OpenShift Container Platform 4.5

インストール

OpenShift Container Platform クラスターのインストールおよび設定

Last Updated: 2023-05-22

OpenShift Container Platform 4.5 インストール

OpenShift Container Platform クラスターのインストールおよび設定

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Installing.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、OpenShift Container Platform のインストール方法と、一部の設定プロセスの詳細について説明します。

目次

第1章 非接続インストールのイメージのミラーリング **32**
1.1. 前提条件 32
1.2. ミラーレジストリーについて 32
1.3. ミラーホストの準備 33
1.3.1. バイナリーのダウンロードによる CLI のインストール 33
1.3.1.1. Linux への CLI のインストール 33
1.3.1.2. Windows での CLI のインストール 33
1.3.1.3. macOS への CLI のインストール 34
1.4. イメージのミラーリングを可能にする認証情報の設定 34
1.5. OPENSHIFT CONTAINER PLATFORM イメージリポジトリーのミラーリング 37
1.6. 非接続環境の CLUSTER SAMPLES OPERATOR 40
1.7. 関連情報 40

第2章 AWS へのインストール **41**
2.1. AWS アカウントの設定 41
2.1.1. Route 53 の設定 41
2.1.2. AWS アカウントの制限 41
2.1.3. 必要な AWS パーミッション 44
2.1.4. IAM ユーザーの作成 51
2.1.5. サポートされている AWS リージョン 52
2.1.6. 次のステップ 53
2.2. AWS の IAM の手動作成 53
2.2.1. IAM の手動作成 53
2.2.2. 管理者の認証情報のルートシークレット形式 55
2.2.2.1. アップグレード 55
2.2.3. mint モード 56
2.2.4. 管理者認証情報の削除またはローテーション機能を持つ mint モード 56
2.3. クラスターの AWS へのクイックインストール 56
2.3.1. 前提条件 56
2.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 57
2.3.3. SSH プライベートキーの生成およびエージェントへの追加 57
2.3.4. インストールプログラムの取得 59
2.3.5. クラスターのデプロイ 60
2.3.6. バイナリーのダウンロードによる CLI のインストール 61
2.3.6.1. Linux への CLI のインストール 62
2.3.6.2. Windows での CLI のインストール 62
2.3.6.3. macOS への CLI のインストール 63
2.3.7. クラスターへのログイン 63
2.3.8. 次のステップ 64
2.4. カスタマイズによる AWS へのクラスターのインストール 64
2.4.1. 前提条件 64
2.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 64
2.4.3. SSH プライベートキーの生成およびエージェントへの追加 65
2.4.4. インストールプログラムの取得 66
2.4.5. インストール設定ファイルの作成 67
2.4.5.1. インストール設定パラメーター 69
2.4.5.1.1. 必須設定パラメーター 69
2.4.5.1.2. ネットワーク設定パラメーター 70
2.4.5.1.3. オプションの設定パラメーター 72
2.4.5.1.4. オプションの AWS 設定パラメーター 76
2.4.5.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル 77

2.4.6. クラスターのデプロイ 79
2.4.7. バイナリーのダウンロードによる CLI のインストール 80
2.4.7.1. Linux への CLI のインストール 80
2.4.7.2. Windows での CLI のインストール 81
2.4.7.3. macOS への CLI のインストール 81
2.4.8. クラスターへのログイン 82
2.4.9. 次のステップ 82
2.5. ネットワークのカスタマイズによる AWS へのクラスターのインストール 83
2.5.1. 前提条件 83
2.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 83
2.5.3. SSH プライベートキーの生成およびエージェントへの追加 84
2.5.4. インストールプログラムの取得 85
2.5.5. インストール設定ファイルの作成 86
2.5.5.1. インストール設定パラメーター 87
2.5.5.1.1. 必須設定パラメーター 88
2.5.5.1.2. ネットワーク設定パラメーター 89
2.5.5.1.3. オプションの設定パラメーター 91
2.5.5.1.4. オプションの AWS 設定パラメーター 94
2.5.5.2. ネットワーク設定パラメーター 96
2.5.5.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル 97
2.5.6. 高度なネットワーク設定パラメーターの変更 99
2.5.7. Cluster Network Operator (CNO) の設定 101
2.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 102
2.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター 102
2.5.7.3. Cluster Network Operator の設定例 103
2.5.8. クラスターのデプロイ 103
2.5.9. バイナリーのダウンロードによる CLI のインストール 105
2.5.9.1. Linux への CLI のインストール 105
2.5.9.2. Windows での CLI のインストール 105
2.5.9.3. macOS への CLI のインストール 106
2.5.10. クラスターへのログイン 106
2.5.11. 次のステップ 107
2.6. ネットワークが制限された環境での AWS へのクラスターのインストール 107
2.6.1. 前提条件 107
2.6.2. ネットワークが制限された環境でのインストールについて 108
2.6.2.1. その他の制限 108
2.6.3. カスタム VPC の使用について 108
2.6.3.1. VPC を使用するための要件 109
2.6.3.2. VPC 検証 111
2.6.3.3. パーミッションの区分 112
2.6.3.4. クラスター間の分離 112
2.6.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 112
2.6.5. SSH プライベートキーの生成およびエージェントへの追加 113
2.6.6. インストール設定ファイルの作成 114
2.6.6.1. インストール設定パラメーター 116
2.6.6.1.1. 必須設定パラメーター 117
2.6.6.1.2. ネットワーク設定パラメーター 118
2.6.6.1.3. オプションの設定パラメーター 120
2.6.6.1.4. オプションの AWS 設定パラメーター 123
2.6.6.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル 125
2.6.6.3. インストール時のクラスター全体のプロキシーの設定 127
2.6.7. クラスターのデプロイ 129
2.6.8. バイナリーのダウンロードによる CLI のインストール 130

2.6.8.1. Linux への CLI のインストール 130
2.6.8.2. Windows での CLI のインストール 131
2.6.8.3. macOS への CLI のインストール 131
2.6.9. クラスターへのログイン 132
2.6.10. 次のステップ 132
2.7. AWS のクラスターの既存 VPC へのインストール 132
2.7.1. 前提条件 133
2.7.2. カスタム VPC の使用について 133
2.7.2.1. VPC を使用するための要件 133
2.7.2.2. VPC 検証 136
2.7.2.3. パーミッションの区分 136
2.7.2.4. クラスター間の分離 136
2.7.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 137
2.7.4. SSH プライベートキーの生成およびエージェントへの追加 137
2.7.5. インストールプログラムの取得 139
2.7.6. インストール設定ファイルの作成 140
2.7.6.1. インストール設定パラメーター 141
2.7.6.1.1. 必須設定パラメーター 141
2.7.6.1.2. ネットワーク設定パラメーター 143
2.7.6.1.3. オプションの設定パラメーター 144
2.7.6.1.4. オプションの AWS 設定パラメーター 148
2.7.6.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル 149
2.7.6.3. インストール時のクラスター全体のプロキシーの設定 151
2.7.7. クラスターのデプロイ 153
2.7.8. バイナリーのダウンロードによる CLI のインストール 154
2.7.8.1. Linux への CLI のインストール 154
2.7.8.2. Windows での CLI のインストール 155
2.7.8.3. macOS への CLI のインストール 155
2.7.9. クラスターへのログイン 156
2.7.10. 次のステップ 156
2.8. プライベートクラスターの AWS へのインストール 156
2.8.1. 前提条件 157
2.8.2. プライベートクラスター 157
2.8.2.1. AWS のプライベートクラスター 157
2.8.2.1.1. 制限事項 158
2.8.3. カスタム VPC の使用について 158
2.8.3.1. VPC を使用するための要件 158
2.8.3.2. VPC 検証 161
2.8.3.3. パーミッションの区分 161
2.8.3.4. クラスター間の分離 161
2.8.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 162
2.8.5. SSH プライベートキーの生成およびエージェントへの追加 162
2.8.6. インストールプログラムの取得 164
2.8.7. インストール設定ファイルの手動作成 165
2.8.7.1. インストール設定パラメーター 165
2.8.7.1.1. 必須設定パラメーター 166
2.8.7.1.2. ネットワーク設定パラメーター 167
2.8.7.1.3. オプションの設定パラメーター 169
2.8.7.1.4. オプションの AWS 設定パラメーター 173
2.8.7.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル 174
2.8.7.3. インストール時のクラスター全体のプロキシーの設定 176
2.8.8. クラスターのデプロイ 178
2.8.9. バイナリーのダウンロードによる CLI のインストール 179

2.8.9.1. Linux への CLI のインストール 179
2.8.9.2. Windows での CLI のインストール 180
2.8.9.3. macOS への CLI のインストール 180
2.8.10. クラスターへのログイン 181
2.8.11. 次のステップ 181
2.9. CLOUDFORMATION テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール 181
2.9.1. 前提条件 182
2.9.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 182
2.9.3. 必要な AWS インフラストラクチャーコンポーネント 183
2.9.3.1. クラスターマシン 183
2.9.3.2. 証明書署名要求の管理 185
2.9.3.3. 他のインフラストラクチャーコンポーネント 185
2.9.3.4. 必要な AWS パーミッション 191
2.9.4. インストールプログラムの取得 198
2.9.5. SSH プライベートキーの生成およびエージェントへの追加 199
2.9.6. AWS のインストール設定ファイルの作成 201
2.9.6.1. インストール設定ファイルの作成 201
2.9.6.2. インストール時のクラスター全体のプロキシーの設定 202
2.9.6.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 204
2.9.7. インフラストラクチャー名の抽出 206
2.9.8. AWS での VPC の作成 206
2.9.8.1. VPC の CloudFormation テンプレート 208
2.9.9. AWS でのネットワークおよび負荷分散コンポーネントの作成 213
2.9.9.1. ネットワークおよびロードバランサーの CloudFormation テンプレート 217
2.9.10. AWS でのセキュリティーグループおよびロールの作成 224
2.9.10.1. セキュリティーオブジェクトの CloudFormation テンプレート 227
2.9.11. AWS インフラストラクチャーの RHCOS AMI 236
2.9.12. AWS でのブートストラップノードの作成 236
2.9.12.1. ブートストラップマシンの CloudFormation テンプレート 241
2.9.13. AWS でのコントロールプレーンの作成 245
2.9.13.1. コントロールプレーンマシンの CloudFormation テンプレート 249
2.9.14. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS でのブートストラップノードの初期化 257
2.9.14.1. AWS でのワーカーノードの作成 257
2.9.14.1.1. ワーカーマシンの CloudFormation テンプレート 261
2.9.15. バイナリーのダウンロードによる CLI のインストール 263
2.9.15.1. Linux への CLI のインストール 264
2.9.15.2. Windows での CLI のインストール 264
2.9.15.3. macOS への CLI のインストール 265
2.9.16. クラスターへのログイン 265
2.9.17. マシンの証明書署名要求の承認 266
2.9.18. Operator の初期設定 268
2.9.18.1. イメージレジストリーストレージの設定 269
2.9.18.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定 269
2.9.18.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 270
2.9.19. ブートストラップリソースの削除 271
2.9.20. Ingress DNS レコードの作成 271
2.9.21. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行 274
2.9.22. 次のステップ 275
2.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での AWS へのクラスターのインストール 275

2.10.1. 前提条件 276
2.10.2. ネットワークが制限された環境でのインストールについて 276
2.10.2.1. その他の制限 277
2.10.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 277
2.10.4. 必要な AWS インフラストラクチャーコンポーネント 278
2.10.4.1. クラスターマシン 278
2.10.4.2. 証明書署名要求の管理 279
2.10.4.3. 他のインフラストラクチャーコンポーネント 280
2.10.4.4. 必要な AWS パーミッション 286
2.10.5. SSH プライベートキーの生成およびエージェントへの追加 293
2.10.6. AWS のインストール設定ファイルの作成 294
2.10.6.1. インストール設定ファイルの作成 294
2.10.6.2. インストール時のクラスター全体のプロキシーの設定 297
2.10.6.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 298
2.10.7. インフラストラクチャー名の抽出 300
2.10.8. AWS での VPC の作成 301
2.10.8.1. VPC の CloudFormation テンプレート 302
2.10.9. AWS でのネットワークおよび負荷分散コンポーネントの作成 308
2.10.9.1. ネットワークおよびロードバランサーの CloudFormation テンプレート 311
2.10.10. AWS でのセキュリティーグループおよびロールの作成 319
2.10.10.1. セキュリティーオブジェクトの CloudFormation テンプレート 321
2.10.11. AWS インフラストラクチャーの RHCOS AMI 330
2.10.12. AWS でのブートストラップノードの作成 331
2.10.12.1. ブートストラップマシンの CloudFormation テンプレート 335
2.10.13. AWS でのコントロールプレーンの作成 339
2.10.13.1. コントロールプレーンマシンの CloudFormation テンプレート 344
2.10.14. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS でのブートストラップノードの初期化 351
2.10.14.1. AWS でのワーカーノードの作成 352
2.10.14.1.1. ワーカーマシンの CloudFormation テンプレート 355
2.10.15. クラスターへのログイン 358
2.10.16. マシンの証明書署名要求の承認 358
2.10.17. Operator の初期設定 361
2.10.17.1. イメージレジストリーストレージの設定 362
2.10.17.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定 362
2.10.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 363
2.10.18. ブートストラップリソースの削除 364
2.10.19. Ingress DNS レコードの作成 364
2.10.20. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行 367
2.10.21. 次のステップ 367
2.11. AWS でのクラスターのアンインストール 368
2.11.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 368

第3章 AZURE へのインストール 370
3.1. AZURE アカウントの設定 370
3.1.1. Azure アカウントの制限 370
3.1.2. Azure でのパブリック DNS ゾーンの設定 373
3.1.3. Azure アカウント制限の拡張 373
3.1.4. 必要な Azure ロール 374
3.1.5. サービスプリンシパルの作成 374
3.1.6. サポート対象の Azure リージョン 377

3.1.7. 次のステップ 378
3.2. AZURE の IAM の手動作成 379
3.2.1. IAM の手動作成 379
3.2.2. 管理者の認証情報のルートシークレット形式 380
3.2.2.1. アップグレード 381
3.2.3. mint モード 381
3.3. クラスターの AZURE へのクイックインストール 382
3.3.1. 前提条件 382
3.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 382
3.3.3. SSH プライベートキーの生成およびエージェントへの追加 383
3.3.4. インストールプログラムの取得 384
3.3.5. クラスターのデプロイ 385
3.3.6. バイナリーのダウンロードによる CLI のインストール 387
3.3.6.1. Linux への CLI のインストール 387
3.3.6.2. Windows での CLI のインストール 388
3.3.6.3. macOS への CLI のインストール 388
3.3.7. クラスターへのログイン 389
3.3.8. 次のステップ 389
3.4. カスタマイズによる AZURE へのクラスターのインストール 389
3.4.1. 前提条件 390
3.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 390
3.4.3. SSH プライベートキーの生成およびエージェントへの追加 390
3.4.4. インストールプログラムの取得 392
3.4.5. インストール設定ファイルの作成 393
3.4.5.1. インストール設定パラメーター 394
3.4.5.1.1. 必須設定パラメーター 395
3.4.5.1.2. ネットワーク設定パラメーター 396
3.4.5.1.3. オプションの設定パラメーター 398
3.4.5.1.4. 追加の Azure 設定パラメーター 401
3.4.5.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル 402
3.4.6. クラスターのデプロイ 404
3.4.7. バイナリーのダウンロードによる CLI のインストール 405
3.4.7.1. Linux への CLI のインストール 405
3.4.7.2. Windows での CLI のインストール 406
3.4.7.3. macOS への CLI のインストール 406
3.4.8. クラスターへのログイン 407
3.4.9. 次のステップ 407
3.5. ネットワークのカスタマイズによる AZURE へのクラスターのインストール 408
3.5.1. 前提条件 408
3.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 408
3.5.3. SSH プライベートキーの生成およびエージェントへの追加 409
3.5.4. インストールプログラムの取得 410
3.5.5. インストール設定ファイルの作成 411
3.5.5.1. インストール設定パラメーター 413
3.5.5.1.1. 必須設定パラメーター 413
3.5.5.1.2. ネットワーク設定パラメーター 415
3.5.5.1.3. オプションの設定パラメーター 416
3.5.5.1.4. 追加の Azure 設定パラメーター 420
3.5.5.2. ネットワーク設定パラメーター 421
3.5.5.3. Azure のカスタマイズされた install-config.yaml ファイルのサンプル 422
3.5.6. 高度なネットワーク設定パラメーターの変更 424
3.5.7. Cluster Network Operator (CNO) の設定 425
3.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 426

3.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター 427
3.5.7.3. Cluster Network Operator の設定例 428
3.5.8. クラスターのデプロイ 428
3.5.9. バイナリーのダウンロードによる CLI のインストール 429
3.5.9.1. Linux への CLI のインストール 430
3.5.9.2. Windows での CLI のインストール 430
3.5.9.3. macOS への CLI のインストール 431
3.5.10. クラスターへのログイン 431
3.5.11. 次のステップ 432
3.6. AZURE のクラスターの既存 VNET へのインストール 432
3.6.1. 前提条件 432
3.6.2. OpenShift Container Platform クラスターでの VNet の再利用について 432
3.6.2.1. VNet を使用するための要件 432
3.6.2.1.1. ネットワークセキュリティーグループの要件 433
3.6.2.2. パーミッションの区分 434
3.6.2.3. クラスター間の分離 434
3.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 435
3.6.4. SSH プライベートキーの生成およびエージェントへの追加 435
3.6.5. インストールプログラムの取得 436
3.6.6. インストール設定ファイルの作成 437
3.6.6.1. インストール設定パラメーター 439
3.6.6.1.1. 必須設定パラメーター 439
3.6.6.1.2. ネットワーク設定パラメーター 441
3.6.6.1.3. オプションの設定パラメーター 442
3.6.6.1.4. 追加の Azure 設定パラメーター 446
3.6.6.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル 447
3.6.6.3. インストール時のクラスター全体のプロキシーの設定 449
3.6.7. クラスターのデプロイ 451
3.6.8. バイナリーのダウンロードによる CLI のインストール 452
3.6.8.1. Linux への CLI のインストール 452
3.6.8.2. Windows での CLI のインストール 452
3.6.8.3. macOS への CLI のインストール 453
3.6.9. クラスターへのログイン 453
3.6.10. 次のステップ 454
3.7. プライベートクラスターの AZURE へのインストール 454
3.7.1. 前提条件 454
3.7.2. プライベートクラスター 454
3.7.2.1. Azure のプライベートクラスター 455
3.7.2.1.1. 制限事項 455
3.7.3. OpenShift Container Platform クラスターでの VNet の再利用について 455
3.7.3.1. VNet を使用するための要件 456
3.7.3.1.1. ネットワークセキュリティーグループの要件 457
3.7.3.2. パーミッションの区分 457
3.7.3.3. クラスター間の分離 458
3.7.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 458
3.7.5. SSH プライベートキーの生成およびエージェントへの追加 458
3.7.6. インストールプログラムの取得 460
3.7.7. インストール設定ファイルの手動作成 461
3.7.7.1. インストール設定パラメーター 461
3.7.7.1.1. 必須設定パラメーター 462
3.7.7.1.2. ネットワーク設定パラメーター 463
3.7.7.1.3. オプションの設定パラメーター 465
3.7.7.1.4. 追加の Azure 設定パラメーター 468

3.7.7.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル 469
3.7.7.3. インストール時のクラスター全体のプロキシーの設定 471
3.7.8. クラスターのデプロイ 473
3.7.9. バイナリーのダウンロードによる CLI のインストール 474
3.7.9.1. Linux への CLI のインストール 474
3.7.9.2. Windows での CLI のインストール 475
3.7.9.3. macOS への CLI のインストール 475
3.7.10. クラスターへのログイン 476
3.7.11. 次のステップ 476
3.8. ARM テンプレートを使用したクラスターの AZURE へのインストール 476
3.8.1. 前提条件 477
3.8.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 477
3.8.3. Azure プロジェクトの設定 478
3.8.3.1. Azure アカウントの制限 478
3.8.3.2. Azure でのパブリック DNS ゾーンの設定 481
3.8.3.3. Azure アカウント制限の拡張 481
3.8.3.4. 証明書署名要求の管理 482
3.8.3.5. 必要な Azure ロール 482
3.8.3.6. サービスプリンシパルの作成 482
3.8.3.7. サポート対象の Azure リージョン 485
3.8.4. インストールプログラムの取得 486
3.8.5. SSH プライベートキーの生成およびエージェントへの追加 487
3.8.6. AWS のインストールファイルの作成 489
3.8.6.1. インストール設定ファイルの作成 489
3.8.6.2. インストール時のクラスター全体のプロキシーの設定 490
3.8.6.3. ARM テンプレートの一般的な変数のエクスポート 492
3.8.6.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 493
3.8.7. Azure リソースグループおよびアイデンティティーの作成 496
3.8.8. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード 496
3.8.9. DNS ゾーンの作成例 498
3.8.10. Azure での VNet の作成 499
3.8.10.1. VNet の ARM テンプレート 499
3.8.11. Azure インフラストラクチャー用の RHCOS クラスターイメージのデプロイ 501
3.8.11.1. イメージストレージの ARM テンプレート 502
3.8.12. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 503
ネットワークトポロジー要件 504
ロードバランサー 504
3.8.13. Azure でのネットワークおよび負荷分散コンポーネントの作成 506
3.8.13.1. ネットワークおよびロードバランサーの ARM テンプレート 507
3.8.14. Azure でのブートストラップマシンの作成 512
3.8.14.1. ブートストラップマシンの ARM テンプレート 513
3.8.15. Azure でのコントロールプレーンの作成 517
3.8.15.1. コントロールプレーンマシンの ARM テンプレート 519
3.8.16. ブートストラップの完了を待機し、Azure のブートストラップリソースを削除する 524
3.8.17. Azure での追加のワーカーマシンの作成 525
3.8.17.1. ワーカーマシンの ARM テンプレート 526
3.8.18. バイナリーのダウンロードによる CLI のインストール 531
3.8.18.1. Linux への CLI のインストール 531
3.8.18.2. Windows での CLI のインストール 531
3.8.18.3. macOS への CLI のインストール 532
3.8.19. クラスターへのログイン 532
3.8.20. マシンの証明書署名要求の承認 533
3.8.21. Ingress DNS レコードの追加 535

3.8.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure インストールの実行 537
3.9. AZURE でのクラスターのアンインストール 537
3.9.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 537

第4章 GCP へのインストール 539
4.1. GCP プロジェクトの設定 539
4.1.1. GCP プロジェクトの作成 539
4.1.2. GCP での API サービスの有効化 539
4.1.3. GCP の DNS の設定 540
4.1.4. GCP アカウントの制限 541
4.1.5. GCP でのサービスアカウントの作成 542
4.1.5.1. 必要な GCP パーミッション 543
4.1.6. サポートされている GCP リージョン 544
4.1.7. 次のステップ 545
4.2. GCP の IAM の手動作成 545
4.2.1. IAM の手動作成 545
4.2.2. 管理者の認証情報のルートシークレット形式 547
4.2.2.1. アップグレード 547
4.2.3. mint モード 547
4.3. GCP へのクラスターのクイックインストール 548
4.3.1. 前提条件 548
4.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 548
4.3.3. SSH プライベートキーの生成およびエージェントへの追加 548
4.3.4. インストールプログラムの取得 550
4.3.5. クラスターのデプロイ 551
4.3.6. バイナリーのダウンロードによる CLI のインストール 553
4.3.6.1. Linux への CLI のインストール 553
4.3.6.2. Windows での CLI のインストール 554
4.3.6.3. macOS への CLI のインストール 554
4.3.7. クラスターへのログイン 555
4.3.8. 次のステップ 555
4.4. カスタマイズによる GCP へのクラスターのインストール 555
4.4.1. 前提条件 555
4.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 556
4.4.3. SSH プライベートキーの生成およびエージェントへの追加 556
4.4.4. インストールプログラムの取得 558
4.4.5. インストール設定ファイルの作成 559
4.4.5.1. インストール設定パラメーター 560
4.4.5.1.1. 必須設定パラメーター 561
4.4.5.1.2. ネットワーク設定パラメーター 562
4.4.5.1.3. オプションの設定パラメーター 564
4.4.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 567
4.4.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 568
4.4.6. クラスターのデプロイ 569
4.4.7. バイナリーのダウンロードによる CLI のインストール 571
4.4.7.1. Linux への CLI のインストール 571
4.4.7.2. Windows での CLI のインストール 572
4.4.7.3. macOS への CLI のインストール 572
4.4.8. クラスターへのログイン 572
4.4.9. 次のステップ 573
4.5. ネットワークのカスタマイズによる GCP へのクラスターのインストール 573
4.5.1. 前提条件 573

4.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 574
4.5.3. SSH プライベートキーの生成およびエージェントへの追加 574
4.5.4. インストールプログラムの取得 576
4.5.5. インストール設定ファイルの作成 577
4.5.5.1. インストール設定パラメーター 578
4.5.5.1.1. 必須設定パラメーター 578
4.5.5.1.2. ネットワーク設定パラメーター 580
4.5.5.1.3. オプションの設定パラメーター 581
4.5.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 585
4.5.5.2. ネットワーク設定パラメーター 586
4.5.5.3. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 587
4.5.6. 高度なネットワーク設定パラメーターの変更 588
4.5.7. Cluster Network Operator (CNO) の設定 590
4.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 591
4.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター 592
4.5.7.3. Cluster Network Operator の設定例 592
4.5.8. クラスターのデプロイ 593
4.5.9. バイナリーのダウンロードによる CLI のインストール 594
4.5.9.1. Linux への CLI のインストール 594
4.5.9.2. Windows での CLI のインストール 595
4.5.9.3. macOS への CLI のインストール 595
4.5.10. クラスターへのログイン 596
4.5.11. 次のステップ 596
4.6. ネットワークが制限された環境での GCP へのクラスターのインストール 596
4.6.1. 前提条件 596
4.6.2. ネットワークが制限された環境でのインストールについて 597
4.6.2.1. その他の制限 597
4.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 598
4.6.4. SSH プライベートキーの生成およびエージェントへの追加 598
4.6.5. インストール設定ファイルの作成 600
4.6.5.1. インストール設定パラメーター 602
4.6.5.1.1. 必須設定パラメーター 602
4.6.5.1.2. ネットワーク設定パラメーター 604
4.6.5.1.3. オプションの設定パラメーター 605
4.6.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 609
4.6.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 609
4.6.5.3. インストール時のクラスター全体のプロキシーの設定 612
4.6.6. クラスターのデプロイ 613
4.6.7. バイナリーのダウンロードによる CLI のインストール 614
4.6.7.1. Linux への CLI のインストール 615
4.6.7.2. Windows での CLI のインストール 615
4.6.7.3. macOS への CLI のインストール 616
4.6.8. クラスターへのログイン 616
4.6.9. 次のステップ 617
4.7. GCP のクラスターの既存 VPC へのインストール 617
4.7.1. 前提条件 617
4.7.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 617
4.7.3. SSH プライベートキーの生成およびエージェントへの追加 618
4.7.4. インストールプログラムの取得 620
4.7.5. インストール設定ファイルの作成 620
4.7.5.1. インストール設定パラメーター 622
4.7.5.1.1. 必須設定パラメーター 622
4.7.5.1.2. ネットワーク設定パラメーター 624

4.7.5.1.3. オプションの設定パラメーター 625
4.7.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 629
4.7.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 629
4.7.5.3. インストール時のクラスター全体のプロキシーの設定 631
4.7.6. クラスターのデプロイ 633
4.7.7. バイナリーのダウンロードによる CLI のインストール 634
4.7.7.1. Linux への CLI のインストール 634
4.7.7.2. Windows での CLI のインストール 635
4.7.7.3. macOS への CLI のインストール 635
4.7.8. クラスターへのログイン 636
4.7.9. 次のステップ 636
4.8. GCP へのプライベートクラスターのインストール 637
4.8.1. 前提条件 637
4.8.2. プライベートクラスター 637
4.8.2.1. GCP のプライベートクラスター 637
4.8.2.1.1. 制限事項 638
4.8.3. カスタム VPC の使用について 638
4.8.3.1. VPC を使用するための要件 638
4.8.3.2. パーミッションの区分 639
4.8.3.3. クラスター間の分離 639
4.8.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 640
4.8.5. SSH プライベートキーの生成およびエージェントへの追加 640
4.8.6. インストールプログラムの取得 642
4.8.7. インストール設定ファイルの手動作成 643
4.8.7.1. インストール設定パラメーター 643
4.8.7.1.1. 必須設定パラメーター 644
4.8.7.1.2. ネットワーク設定パラメーター 645
4.8.7.1.3. オプションの設定パラメーター 647
4.8.7.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 650
4.8.7.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 651
4.8.7.3. インストール時のクラスター全体のプロキシーの設定 653
4.8.8. クラスターのデプロイ 654
4.8.9. バイナリーのダウンロードによる CLI のインストール 655
4.8.9.1. Linux への CLI のインストール 656
4.8.9.2. Windows での CLI のインストール 656
4.8.9.3. macOS への CLI のインストール 657
4.8.10. クラスターへのログイン 657
4.8.11. 次のステップ 658
4.9. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール 658
4.9.1. 前提条件 658
4.9.2. 証明書署名要求の管理 659
4.9.3. GCP プロジェクトの設定 659
4.9.3.1. GCP プロジェクトの作成 659
4.9.3.2. GCP での API サービスの有効化 659
4.9.3.3. GCP の DNS の設定 660
4.9.3.4. GCP アカウントの制限 661
4.9.3.5. GCP でのサービスアカウントの作成 662
4.9.3.5.1. 必要な GCP パーミッション 663
4.9.3.6. サポートされている GCP リージョン 664
4.9.3.7. GCP の CLI ツールのインストールおよび設定 665
4.9.4. GCP のインストール設定ファイルの作成 665
4.9.4.1. インストール設定ファイルの作成 665

4.9.4.2. インストール時のクラスター全体のプロキシーの設定 667
4.9.4.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 668
4.9.5. 一般的な変数のエクスポート 671
4.9.5.1. インフラストラクチャー名の抽出 671
4.9.5.2. Deployment Manager テンプレートの一般的な変数のエクスポート 671
4.9.6. GCP での VPC の作成 672
4.9.6.1. VPC の Deployment Manager テンプレート 673
4.9.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 674
ネットワークトポロジー要件 675
ロードバランサー 675
4.9.8. GCP でのロードバランサーの作成 677
4.9.8.1. 外部ロードバランサーの Deployment Manager テンプレート 679
4.9.8.2. 内部ロードバランサーの Deployment Manager テンプレート 680
4.9.9. GCP でのプライベート DNS ゾーンの作成 681
4.9.9.1. プライベート DNS の Deployment Manager テンプレート 683
4.9.10. GCP でのファイアウォールルールの作成 683
4.9.10.1. ファイアウォールルール用の Deployment Manager テンプレート 684
4.9.11. GCP での IAM ロールの作成 687
4.9.11.1. IAM ロールの Deployment Manager テンプレート 688
4.9.12. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成 689
4.9.13. GCP でのブートストラップマシンの作成 690
4.9.13.1. ブートストラップマシンの Deployment Manager テンプレート 692
4.9.14. GCP でのコントロールプレーンマシンの作成 693
4.9.14.1. コントロールプレーンマシンの Deployment Manager テンプレート 695
4.9.15. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する 697
4.9.16. GCP での追加のワーカーマシンの作成 698
4.9.16.1. ワーカーマシンの Deployment Manager テンプレート 700
4.9.17. バイナリーのダウンロードによる CLI のインストール 701
4.9.17.1. Linux への CLI のインストール 701
4.9.17.2. Windows での CLI のインストール 702
4.9.17.3. macOS への CLI のインストール 702
4.9.18. クラスターへのログイン 703
4.9.19. マシンの証明書署名要求の承認 703
4.9.20. オプション: Ingress DNS レコードの追加 706
4.9.21. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了 708
4.9.22. 次のステップ 710
4.10. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへの共有 VPC のあるクラスターのインストール 710
4.10.1. 前提条件 711
4.10.2. 証明書署名要求の管理 711
4.10.3. クラスターをホストする GCP プロジェクトの設定 711
4.10.3.1. GCP プロジェクトの作成 711
4.10.3.2. GCP での API サービスの有効化 711
4.10.3.3. GCP アカウントの制限 712
4.10.3.4. GCP でのサービスアカウントの作成 714
4.10.3.4.1. 必要な GCP パーミッション 714
4.10.3.5. サポートされている GCP リージョン 715
4.10.3.6. GCP の CLI ツールのインストールおよび設定 716
4.10.4. 共有 VPC ネットワークをホストする GCP プロジェクトの設定 717
4.10.4.1. GCP の DNS の設定 717
4.10.4.2. GCP での VPC の作成 718
4.10.4.2.1. VPC の Deployment Manager テンプレート 720
4.10.5. GCP のインストール設定ファイルの作成 721

4.10.5.1. インストール設定ファイルの手動作成 721
4.10.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル 722
4.10.5.3. インストール時のクラスター全体のプロキシーの設定 724
4.10.5.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 725
4.10.6. 一般的な変数のエクスポート 728
4.10.6.1. インフラストラクチャー名の抽出 728
4.10.6.2. Deployment Manager テンプレートの一般的な変数のエクスポート 729
4.10.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 730
ネットワークトポロジー要件 731
ロードバランサー 731
4.10.8. GCP でのロードバランサーの作成 733
4.10.8.1. 外部ロードバランサーの Deployment Manager テンプレート 735
4.10.8.2. 内部ロードバランサーの Deployment Manager テンプレート 736
4.10.9. GCP でのプライベート DNS ゾーンの作成 737
4.10.9.1. プライベート DNS の Deployment Manager テンプレート 739
4.10.10. GCP でのファイアウォールルールの作成 739
4.10.10.1. ファイアウォールルール用の Deployment Manager テンプレート 740
4.10.11. GCP での IAM ロールの作成 743
4.10.11.1. IAM ロールの Deployment Manager テンプレート 745
4.10.12. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成 746
4.10.13. GCP でのブートストラップマシンの作成 746
4.10.13.1. ブートストラップマシンの Deployment Manager テンプレート 748
4.10.14. GCP でのコントロールプレーンマシンの作成 750
4.10.14.1. コントロールプレーンマシンの Deployment Manager テンプレート 752
4.10.15. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する 754
4.10.16. GCP での追加のワーカーマシンの作成 755
4.10.16.1. ワーカーマシンの Deployment Manager テンプレート 757
4.10.17. バイナリーのダウンロードによる CLI のインストール 758
4.10.17.1. Linux への CLI のインストール 758
4.10.17.2. Windows での CLI のインストール 758
4.10.17.3. macOS への CLI のインストール 759
4.10.18. クラスターへのログイン 759
4.10.19. マシンの証明書署名要求の承認 760
4.10.20. Ingress DNS レコードの追加 762
4.10.21. Ingress ファイアウォールルールの追加 764
4.10.21.1. GCP での共有 VPC のクラスター全体のファイアウォールルールの作成 764
4.10.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了 765
4.10.23. 次のステップ 768
4.11. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での GCP へのクラスターのインストール 768
4.11.1. 前提条件 768
4.11.2. GCP プロジェクトの設定 769
4.11.2.1. GCP プロジェクトの作成 769
4.11.2.2. GCP での API サービスの有効化 769
4.11.2.3. GCP の DNS の設定 770
4.11.2.4. GCP アカウントの制限 771
4.11.2.5. GCP でのサービスアカウントの作成 772
4.11.2.5.1. 必要な GCP パーミッション 773
4.11.2.6. サポートされている GCP リージョン 774
4.11.2.7. GCP の CLI ツールのインストールおよび設定 775
4.11.3. GCP のインストール設定ファイルの作成 775
4.11.3.1. インストール設定ファイルの作成 775

4.11.3.2. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 777
4.11.4. 一般的な変数のエクスポート 779
4.11.4.1. インフラストラクチャー名の抽出 779
4.11.4.2. Deployment Manager テンプレートの一般的な変数のエクスポート 779
4.11.5. GCP での VPC の作成 780
4.11.5.1. VPC の Deployment Manager テンプレート 781
4.11.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 782
ネットワークトポロジー要件 783
ロードバランサー 783
4.11.7. GCP でのロードバランサーの作成 785
4.11.7.1. 外部ロードバランサーの Deployment Manager テンプレート 787
4.11.7.2. 内部ロードバランサーの Deployment Manager テンプレート 788
4.11.8. GCP でのプライベート DNS ゾーンの作成 789
4.11.8.1. プライベート DNS の Deployment Manager テンプレート 791
4.11.9. GCP でのファイアウォールルールの作成 791
4.11.9.1. ファイアウォールルール用の Deployment Manager テンプレート 792
4.11.10. GCP での IAM ロールの作成 795
4.11.10.1. IAM ロールの Deployment Manager テンプレート 796
4.11.11. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成 797
4.11.12. GCP でのブートストラップマシンの作成 798
4.11.12.1. ブートストラップマシンの Deployment Manager テンプレート 800
4.11.13. GCP でのコントロールプレーンマシンの作成 801
4.11.13.1. コントロールプレーンマシンの Deployment Manager テンプレート 803
4.11.14. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する 805
4.11.15. GCP での追加のワーカーマシンの作成 806
4.11.15.1. ワーカーマシンの Deployment Manager テンプレート 808
4.11.16. クラスターへのログイン 809
4.11.17. マシンの証明書署名要求の承認 810
4.11.18. オプション: Ingress DNS レコードの追加 812
4.11.19. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了 814
4.11.20. 次のステップ 816
4.12. GCP でのクラスターのアンインストール 816
4.12.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 816

第5章 ベアメタルへのインストール **818**
5.1. クラスターのベアメタルへのインストール 818
5.1.1. 前提条件 818
5.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 818
5.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件 819
5.1.3.1. 必要なマシン 819
5.1.3.2. ネットワーク接続の要件 819
5.1.3.3. 最小リソース要件 820
5.1.3.4. 証明書署名要求の管理 820
5.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 820
5.1.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 821
ネットワークトポロジー要件 821
ロードバランサー 822
5.1.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 823
5.1.5. SSH プライベートキーの生成およびエージェントへの追加 826
5.1.6. インストールプログラムの取得 828
5.1.7. バイナリーのダウンロードによる CLI のインストール 829
5.1.7.1. Linux への CLI のインストール 829
5.1.7.2. Windows での CLI のインストール 829

5.1.7.3. macOS への CLI のインストール 830
5.1.8. インストール設定ファイルの手動作成 830
5.1.8.1. インストール設定パラメーター 831
5.1.8.1.1. 必須設定パラメーター 831
5.1.8.1.2. ネットワーク設定パラメーター 833
5.1.8.1.3. オプションの設定パラメーター 834
5.1.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 838
5.1.8.2. ベアメタルのサンプル install-config.yaml ファイル 838
5.1.8.3. インストール時のクラスター全体のプロキシーの設定 840
5.1.9. 3 ノードクラスターの設定 842
5.1.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 842
5.1.11. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 845
5.1.11.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 845
5.1.11.1.1. 高度なネットワークの設定 847
5.1.11.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 848
5.1.12. クラスターの作成 852
5.1.13. クラスターへのログイン 853
5.1.14. マシンの証明書署名要求の承認 853
5.1.15. Operator の初期設定 856
5.1.15.1. インストール時に削除されたイメージレジストリー 856
5.1.15.2. イメージレジストリーストレージの設定 857
5.1.15.2.1. ベアメタルの場合のレジストリーストレージの設定 857
5.1.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 858
5.1.15.2.3. ベアメタルの場合のブロックレジストリーストレージの設定 859
5.1.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 860
5.1.17. 次のステップ 862
5.2. ネットワークのカスタマイズによるベアメタルへのクラスターのインストール 862
5.2.1. 前提条件 862
5.2.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 862
5.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件 863
5.2.3.1. 必要なマシン 863
5.2.3.2. ネットワーク接続の要件 864
5.2.3.3. 最小リソース要件 864
5.2.3.4. 証明書署名要求の管理 864
5.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 864
5.2.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 865
ネットワークトポロジー要件 866
ロードバランサー 866
5.2.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 868
5.2.5. SSH プライベートキーの生成およびエージェントへの追加 871
5.2.6. インストールプログラムの取得 872
5.2.7. バイナリーのダウンロードによる CLI のインストール 873
5.2.7.1. Linux への CLI のインストール 873
5.2.7.2. Windows での CLI のインストール 874
5.2.7.3. macOS への CLI のインストール 874
5.2.8. インストール設定ファイルの手動作成 875
5.2.8.1. インストール設定パラメーター 875
5.2.8.1.1. 必須設定パラメーター 876
5.2.8.1.2. ネットワーク設定パラメーター 877
5.2.8.1.3. オプションの設定パラメーター 879
5.2.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 883
5.2.8.2. ベアメタルのサンプル install-config.yaml ファイル 883
5.2.8.3. ネットワーク設定パラメーター 885

5.2.9. 高度なネットワーク設定パラメーターの変更 886
5.2.10. Cluster Network Operator (CNO) の設定 888
5.2.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 889
5.2.10.2. Cluster Network Operator の設定例 889
5.2.11. Ignition 設定ファイルの作成 890
5.2.12. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 891
5.2.12.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 891
5.2.12.1.1. 高度なネットワークの設定 893
5.2.12.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 894
5.2.13. クラスターの作成 898
5.2.14. クラスターへのログイン 899
5.2.15. マシンの証明書署名要求の承認 899
5.2.16. Operator の初期設定 902
5.2.16.1. インストール時に削除されたイメージレジストリー 903
5.2.16.2. イメージレジストリーストレージの設定 903
5.2.16.3. ベアメタルの場合のブロックレジストリーストレージの設定 903
5.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 904
5.2.18. 次のステップ 906
5.3. ネットワークが制限された環境でのクラスターのベアメタルへのインストール 906
5.3.1. 前提条件 906
5.3.2. ネットワークが制限された環境でのインストールについて 907
5.3.2.1. その他の制限 907
5.3.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 907
5.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件 908
5.3.4.1. 必要なマシン 908
5.3.4.2. ネットワーク接続の要件 909
5.3.4.3. 最小リソース要件 909
5.3.4.4. 証明書署名要求の管理 909
5.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 909
5.3.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 910
ネットワークトポロジー要件 911
ロードバランサー 911
5.3.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 913
5.3.6. SSH プライベートキーの生成およびエージェントへの追加 916
5.3.7. インストール設定ファイルの手動作成 917
5.3.7.1. インストール設定パラメーター 918
5.3.7.1.1. 必須設定パラメーター 918
5.3.7.1.2. ネットワーク設定パラメーター 920
5.3.7.1.3. オプションの設定パラメーター 921
5.3.7.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 925
5.3.7.2. ベアメタルのサンプル install-config.yaml ファイル 925
5.3.7.3. インストール時のクラスター全体のプロキシーの設定 928
5.3.8. 3 ノードクラスターの設定 929
5.3.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 929
5.3.10. chrony タイムサービスの設定 932
5.3.11. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 933
5.3.11.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 933
5.3.11.1.1. 高度なネットワークの設定 935
5.3.11.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 936
5.3.12. クラスターの作成 940
5.3.13. クラスターへのログイン 941
5.3.14. マシンの証明書署名要求の承認 941
5.3.15. Operator の初期設定 944

5.3.15.1. イメージレジストリーストレージの設定 945
5.3.15.1.1. イメージレジストリーの管理状態の変更 945
5.3.15.1.2. ベアメタルの場合のレジストリーストレージの設定 945
5.3.15.1.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 946
5.3.15.1.4. ベアメタルの場合のブロックレジストリーストレージの設定 947
5.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 948
5.3.17. 次のステップ 950

第6章 IBM Z および LINUXONE へのインストール 951
6.1. クラスターの IBM Z および LINUXONE へのインストール 951
6.1.1. 前提条件 951
6.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 951
6.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件 952
6.1.3.1. 必要なマシン 952
6.1.3.2. ネットワーク接続の要件 953
6.1.3.3. IBM Z ネットワーク接続の要件 953
6.1.3.4. 最小リソース要件 953
6.1.3.5. 最小の IBM Z システム要件 953
ハードウェア要件 953
オペレーティングシステム要件 953
z/VM ゲスト仮想マシンのディスクストレージ 954
ストレージ/メインメモリー 954
6.1.3.6. 推奨される IBM Z システム要件 954
ハードウェア要件 954
オペレーティングシステム要件 954
z/VM ゲスト仮想マシンのディスクストレージ 954
ストレージ/メインメモリー 955
6.1.3.7. 証明書署名要求の管理 955
6.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 955
6.1.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 956
ネットワークトポロジー要件 956
ロードバランサー 957
6.1.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 958
6.1.5. SSH プライベートキーの生成およびエージェントへの追加 961
6.1.6. インストールプログラムの取得 963
6.1.7. バイナリーのダウンロードによる CLI のインストール 964
6.1.7.1. Linux への CLI のインストール 964
6.1.7.2. Windows での CLI のインストール 964
6.1.7.3. macOS への CLI のインストール 965
6.1.8. インストール設定ファイルの手動作成 965
6.1.8.1. IBM Z のサンプル install-config.yaml ファイル 966
6.1.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 968
6.1.10. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 970
6.1.11. クラスターの作成 972
6.1.12. クラスターへのログイン 973
6.1.13. マシンの証明書署名要求の承認 973
6.1.14. Operator の初期設定 976
6.1.14.1. イメージレジストリーストレージの設定 976
6.1.14.1.1. ベアメタルの場合のレジストリーストレージの設定 977
6.1.14.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 978
6.1.15. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 979
6.1.16. デバッグ情報の収集 981
6.1.17. 関連情報 981

6.1.18. 次のステップ 981
6.2. ネットワークが制限された環境でのクラスターの IBM Z および LINUXONE へのインストール 982
6.2.1. ネットワークが制限された環境でのインストールについて 982
6.2.1.1. その他の制限 983
6.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件 983
6.2.2.1. 必要なマシン 983
6.2.2.2. ネットワーク接続の要件 984
6.2.2.3. IBM Z ネットワーク接続の要件 984
6.2.2.4. 最小リソース要件 984
6.2.2.5. 最小の IBM Z システム要件 984
ハードウェア要件 985
オペレーティングシステム要件 985
z/VM ゲスト仮想マシンのディスクストレージ 985
ストレージ/メインメモリー 985
6.2.2.6. 推奨される IBM Z システム要件 985
ハードウェア要件 985
オペレーティングシステム要件 985
z/VM ゲスト仮想マシンのディスクストレージ 986
ストレージ/メインメモリー 986
6.2.2.7. 証明書署名要求の管理 986
6.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 986
6.2.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 987
ネットワークトポロジー要件 988
ロードバランサー 988
6.2.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 989
6.2.4. SSH プライベートキーの生成およびエージェントへの追加 992
6.2.5. インストール設定ファイルの手動作成 994
6.2.5.1. IBM Z のサンプル install-config.yaml ファイル 995
6.2.5.2. インストール時のクラスター全体のプロキシーの設定 997
6.2.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 998
6.2.7. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1000
6.2.8. クラスターの作成 1002
6.2.9. クラスターへのログイン 1003
6.2.10. マシンの証明書署名要求の承認 1004
6.2.11. Operator の初期設定 1006
6.2.11.1. イメージレジストリーストレージの設定 1007
6.2.11.1.1. ベアメタルの場合のレジストリーストレージの設定 1007
6.2.11.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 1008
6.2.12. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1009
6.2.13. デバッグ情報の収集 1011

第7章 IBM POWER へのインストール 1013
7.1. クラスターの IBM POWER へのインストール 1013
7.1.1. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1013
7.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件 1014
7.1.2.1. 必要なマシン 1014
7.1.2.2. ネットワーク接続の要件 1014
7.1.2.3. 最小リソース要件 1015
7.1.2.4. 証明書署名要求の管理 1015
7.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 1015
7.1.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 1016
ネットワークトポロジー要件 1016

ロードバランサー 1017
7.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 1018
7.1.4. SSH プライベートキーの生成およびエージェントへの追加 1021
7.1.5. インストールプログラムの取得 1023
7.1.6. バイナリーのダウンロードによる CLI のインストール 1024
7.1.6.1. Linux への CLI のインストール 1024
7.1.6.2. Windows での CLI のインストール 1024
7.1.6.3. macOS への CLI のインストール 1025
7.1.7. インストール設定ファイルの手動作成 1025
7.1.7.1. IBM Power のサンプル install-config.yaml ファイル 1026
7.1.8. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1028
7.1.9. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1030
7.1.9.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1030
7.1.9.2. PXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1032
7.1.10. クラスターの作成 1035
7.1.11. クラスターへのログイン 1036
7.1.12. マシンの証明書署名要求の承認 1036
7.1.13. Operator の初期設定 1039
7.1.13.1. イメージレジストリーストレージの設定 1039
7.1.13.1.1. ベアメタルの場合のレジストリーストレージの設定 1040
7.1.13.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 1041
7.1.14. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1042
7.2. ネットワークが制限された環境での IBM POWER へのクラスターのインストール 1044
7.2.1. ネットワークが制限された環境でのインストールについて 1045
7.2.1.1. その他の制限 1045
7.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件 1045
7.2.2.1. 必要なマシン 1045
7.2.2.2. ネットワーク接続の要件 1046
7.2.2.3. 最小リソース要件 1046
7.2.2.4. 証明書署名要求の管理 1046
7.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 1047
7.2.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 1047
ネットワークトポロジー要件 1048
ロードバランサー 1048
7.2.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 1050
7.2.4. SSH プライベートキーの生成およびエージェントへの追加 1053
7.2.5. インストール設定ファイルの手動作成 1054
7.2.5.1. IBM Power のサンプル install-config.yaml ファイル 1055
7.2.5.2. インストール時のクラスター全体のプロキシーの設定 1057
7.2.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1059
7.2.7. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1061
7.2.7.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1061
7.2.7.2. PXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1063
7.2.8. クラスターの作成 1066
7.2.9. クラスターへのログイン 1067
7.2.10. マシンの証明書署名要求の承認 1067
7.2.11. Operator の初期設定 1070
7.2.11.1. イメージレジストリーストレージの設定 1070
7.2.11.1.1. ベアメタルの場合のレジストリーストレージの設定 1071
7.2.11.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 1072
7.2.12. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1073

第8章 OPENSTACK へのインストール 1076
8.1. カスタマイズによる OPENSTACK へのクラスターのインストール 1076
8.1.1. 前提条件 1076
8.1.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン 1076
8.1.2.1. コントロールプレーンおよびコンピュートマシン 1077
8.1.2.2. ブートストラップマシン 1077
8.1.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1078
8.1.4. RHOSP での Swift の有効化 1078
8.1.5. 外部ネットワークアクセスの確認 1079
8.1.6. インストールプログラムのパラメーターの定義 1080
8.1.7. インストールプログラムの取得 1082
8.1.8. インストール設定ファイルの作成 1083
8.1.8.1. インストール時のクラスター全体のプロキシーの設定 1084
8.1.9. インストール設定パラメーター 1086
8.1.9.1. 必須設定パラメーター 1086
8.1.9.2. ネットワーク設定パラメーター 1088
8.1.9.3. オプションの設定パラメーター 1089
8.1.9.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター 1093
8.1.9.5. オプションの RHOSP 設定パラメーター 1094
8.1.9.6. 追加の Google Cloud Platform (GCP) 設定パラメーター 1095
8.1.9.7. RHOSP デプロイメントでのカスタムサブネット 1096
8.1.9.8. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル 1096
8.1.10. SSH プライベートキーの生成およびエージェントへの追加 1097
8.1.11. 環境へのアクセスの有効化 1099
8.1.11.1. floating IP アドレスを使ったアクセスの有効化 1099
8.1.11.2. Floating IP アドレスを使用しないアクセスの有効化 1100
8.1.12. クラスターのデプロイ 1100
8.1.13. クラスターステータスの確認 1101
8.1.14. クラスターへのログイン 1102
8.1.15. Floating IP アドレスを使用したアプリケーションアクセスの設定 1102
8.1.16. 次のステップ 1103
8.2. KURYR を使用する OPENSTACK へのクラスターのインストール 1103
8.2.1. 前提条件 1104
8.2.2. Kuryr SDN について 1104
8.2.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン 1105
8.2.3.1. クォータの拡大 1107
8.2.3.2. Neutron の設定 1107
8.2.3.3. Octavia の設定 1107
8.2.3.3.1. Octavia OVN ドライバー 1110
8.2.3.4. Kuryr を使用したインストールについての既知の制限 1111
RHOSP の一般的な制限 1111
RHOSP バージョンの制限 1111
RHOSP 環境の制限 1112
RHOSP のアップグレードの制限 1112
8.2.3.5. コントロールプレーンおよびコンピュートマシン 1113
8.2.3.6. ブートストラップマシン 1113
8.2.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1113
8.2.5. RHOSP での Swift の有効化 1114
8.2.6. 外部ネットワークアクセスの確認 1114
8.2.7. インストールプログラムのパラメーターの定義 1116
8.2.8. インストールプログラムの取得 1117
8.2.9. インストール設定ファイルの作成 1118

8.2.9.1. インストール時のクラスター全体のプロキシーの設定 1120
8.2.10. インストール設定パラメーター 1121
8.2.10.1. 必須設定パラメーター 1122
8.2.10.2. ネットワーク設定パラメーター 1123
8.2.10.3. オプションの設定パラメーター 1125
8.2.10.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター 1128
8.2.10.5. オプションの RHOSP 設定パラメーター 1129
8.2.10.6. 追加の Google Cloud Platform (GCP) 設定パラメーター 1131
8.2.10.7. RHOSP デプロイメントでのカスタムサブネット 1131
8.2.10.8. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル 1132
8.2.11. SSH プライベートキーの生成およびエージェントへの追加 1133
8.2.12. 環境へのアクセスの有効化 1135
8.2.12.1. floating IP アドレスを使ったアクセスの有効化 1135
8.2.12.2. Floating IP アドレスを使用しないアクセスの有効化 1135
8.2.13. クラスターのデプロイ 1136
8.2.14. クラスターステータスの確認 1137
8.2.15. クラスターへのログイン 1138
8.2.16. Floating IP アドレスを使用したアプリケーションアクセスの設定 1138
8.2.17. 次のステップ 1139
8.3. 独自のインフラストラクチャーを使用した OPENSTACK へのクラスターのインストール 1139
8.3.1. 前提条件 1139
8.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1140
8.3.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン 1140
8.3.3.1. コントロールプレーンおよびコンピュートマシン 1141
8.3.3.2. ブートストラップマシン 1142
8.3.4. Playbook 依存関係のダウンロード 1142
8.3.5. インストールプログラムの取得 1143
8.3.6. SSH プライベートキーの生成およびエージェントへの追加 1144
8.3.7. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成 1145
8.3.8. 外部ネットワークアクセスの確認 1147
8.3.9. 環境へのアクセスの有効化 1147
8.3.9.1. floating IP アドレスを使ったアクセスの有効化 1147
8.3.10. インストールプログラムのパラメーターの定義 1148
8.3.11. インストール設定ファイルの作成 1150
8.3.12. インストール設定パラメーター 1151
8.3.12.1. 必須設定パラメーター 1152
8.3.12.2. ネットワーク設定パラメーター 1153
8.3.12.3. オプションの設定パラメーター 1155
8.3.12.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター 1158
8.3.12.5. オプションの RHOSP 設定パラメーター 1159
8.3.12.6. 追加の Google Cloud Platform (GCP) 設定パラメーター 1161
8.3.12.7. RHOSP デプロイメントでのカスタムサブネット 1161
8.3.12.8. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル 1162
8.3.12.9. マシンのカスタムサブネットの設定 1163
8.3.12.10. コンピュートマシンプールを空にする 1163
8.3.13. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1164
8.3.14. ブートストラップ Ignition ファイルの準備 1166
8.3.15. コントロールプレーンの Ignition 設定ファイルの作成 1169
8.3.16. ネットワークリソースの作成 1170
8.3.17. ブートストラップマシンの作成 1180
8.3.18. コントロールプレーンマシンの作成 1181
8.3.19. クラスターへのログイン 1184
8.3.20. ブートストラップリソースの削除 1184

8.3.21. コンピュートマシンの作成 1186
8.3.22. マシンの証明書署名要求の承認 1187
8.3.23. インストールの正常な実行の確認 1190
8.3.24. Floating IP アドレスを使用したアプリケーションアクセスの設定 1190
8.3.25. 次のステップ 1191
8.4. 独自のインフラストラクチャーでの KURYR を使用する OPENSTACK へのクラスターのインストール 1191
8.4.1. 前提条件 1191
8.4.2. Kuryr SDN について 1192
8.4.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン 1192
8.4.3.1. クォータの拡大 1194
8.4.3.2. Neutron の設定 1194
8.4.3.3. Octavia の設定 1195
8.4.3.3.1. Octavia OVN ドライバー 1198
8.4.3.4. Kuryr を使用したインストールについての既知の制限 1199
RHOSP の一般的な制限 1199
RHOSP バージョンの制限 1199
RHOSP 環境の制限 1199
RHOSP のアップグレードの制限 1200
8.4.3.5. コントロールプレーンおよびコンピュートマシン 1200
8.4.3.6. ブートストラップマシン 1201
8.4.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1201
8.4.5. Playbook 依存関係のダウンロード 1201
8.4.6. インストールプログラムの取得 1202
8.4.7. SSH プライベートキーの生成およびエージェントへの追加 1203
8.4.8. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成 1205
8.4.9. 外部ネットワークアクセスの確認 1206
8.4.10. 環境へのアクセスの有効化 1207
8.4.10.1. floating IP アドレスを使ったアクセスの有効化 1207
8.4.11. インストールプログラムのパラメーターの定義 1207
8.4.12. インストール設定ファイルの作成 1209
8.4.13. インストール設定パラメーター 1211
8.4.13.1. 必須設定パラメーター 1211
8.4.13.2. ネットワーク設定パラメーター 1213
8.4.13.3. オプションの設定パラメーター 1214
8.4.13.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター 1218
8.4.13.5. オプションの RHOSP 設定パラメーター 1219
8.4.13.6. 追加の Google Cloud Platform (GCP) 設定パラメーター 1220
8.4.13.7. RHOSP デプロイメントでのカスタムサブネット 1221
8.4.13.8. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル 1222
8.4.13.9. マシンのカスタムサブネットの設定 1223
8.4.13.10. コンピュートマシンプールを空にする 1223
8.4.13.11. ネットワークタイプの変更 1224
8.4.14. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1224
8.4.15. ブートストラップ Ignition ファイルの準備 1227
8.4.16. コントロールプレーンの Ignition 設定ファイルの作成 1230
8.4.17. ネットワークリソースの作成 1230
8.4.18. ブートストラップマシンの作成 1240
8.4.19. コントロールプレーンマシンの作成 1242
8.4.20. クラスターへのログイン 1244
8.4.21. ブートストラップリソースの削除 1245
8.4.22. コンピュートマシンの作成 1246
8.4.23. マシンの証明書署名要求の承認 1248

8.4.24. インストールの正常な実行の確認 1250
8.4.25. Floating IP アドレスを使用したアプリケーションアクセスの設定 1250
8.4.26. 次のステップ 1251
8.5. ネットワークが制限された環境での OPENSTACK へのクラスターのインストール 1251
8.5.1. ネットワークが制限された環境でのインストールについて 1252
8.5.1.1. その他の制限 1252
8.5.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン 1252
8.5.2.1. コントロールプレーンおよびコンピュートマシン 1253
8.5.2.2. ブートストラップマシン 1254
8.5.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1254
8.5.4. RHOSP での Swift の有効化 1255
8.5.5. インストールプログラムのパラメーターの定義 1255
8.5.6. ネットワークが制限されたインストール用の RHCOS イメージの作成 1257
8.5.7. インストール設定ファイルの作成 1258
8.5.7.1. インストール設定パラメーター 1261
8.5.7.1.1. 必須設定パラメーター 1261
8.5.7.1.2. ネットワーク設定パラメーター 1263
8.5.7.1.3. オプションの設定パラメーター 1264
8.5.7.1.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター 1268
8.5.7.1.5. オプションの RHOSP 設定パラメーター 1269
8.5.7.1.6. 追加の Google Cloud Platform (GCP) 設定パラメーター 1270
8.5.7.2. 制限された OpenStack インストールのカスタマイズされた install-config.yaml ファイルのサンプル 1271
8.5.8. SSH プライベートキーの生成およびエージェントへの追加 1272
8.5.9. 環境へのアクセスの有効化 1274
8.5.9.1. floating IP アドレスを使ったアクセスの有効化 1274
8.5.9.2. Floating IP アドレスを使用しないアクセスの有効化 1274
8.5.10. クラスターのデプロイ 1275
8.5.11. クラスターステータスの確認 1276
8.5.12. クラスターへのログイン 1277
8.5.13. Floating IP アドレスを使用したアプリケーションアクセスの設定 1277
8.6. OPENSTACK でのクラスターのアンインストール 1278
8.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 1278
8.7. 独自のインフラストラクチャーからの OPENSTACK のクラスターのアンインストール 1279
8.7.1. 前提条件 1279
8.7.2. Playbook 依存関係のダウンロード 1280
8.7.3. 独自のインフラストラクチャーを使用する RHOSP のクラスターの削除 1280

第9章 RHV へのインストール 1289
9.1. RHV へのクラスターのクイックインストール 1289
9.1.1. 前提条件 1290
9.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1290
9.1.3. RHV 環境の要件 1291
9.1.4. RHV 環境の要件の確認 1292
9.1.5. RHV でのネットワーク環境の準備 1294
9.1.6. RHV 用の CA 証明書の設定 1295
9.1.7. SSH プライベートキーの生成およびエージェントへの追加 1296
9.1.8. インストールプログラムの取得 1297
9.1.9. クラスターのデプロイ 1298

第10章 バイナリーのダウンロードによる CLI のインストール 1302
10.1. LINUX への CLI のインストール 1302
10.2. WINDOWS での CLI のインストール 1302

10.3. MACOS への CLI のインストール 1303

第11章 クラスターへのログイン 1304
11.1. クラスターステータスの確認 1304
11.2. RHV での OPENSHIFT CONTAINER PLATFORM WEB コンソールへのアクセス 1305
11.3. RED HAT VIRTUALIZATION (RHV) へのインストールに関するよくある問題のトラブルシューティング 1305
11.3.1. CPU 負荷が増大し、ノードが Not Ready 状態になる 1305
11.3.2. OpenShift Container Platform クラスター API に接続できない 1306
11.4. インストール後のタスク 1306
11.5. カスタマイズによる RHV へのクラスターのインストール 1307
11.5.1. 前提条件 1308
11.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1309
11.5.3. RHV 環境の要件 1309
11.5.4. RHV 環境の要件の確認 1310
11.5.5. RHV でのネットワーク環境の準備 1312
11.5.6. RHV 用の CA 証明書の設定 1313
11.5.7. SSH プライベートキーの生成およびエージェントへの追加 1314
11.5.8. インストールプログラムの取得 1315
11.5.9. インストール設定ファイルの作成 1316
11.5.9.1. Red Hat Virtualization (RHV) のサンプル install-config.yaml ファイル 1319
11.5.9.2. インストール設定パラメーター 1322
11.5.9.2.1. 必須設定パラメーター 1322
11.5.9.2.2. ネットワーク設定パラメーター 1323
11.5.9.2.3. オプションの設定パラメーター 1325
11.5.9.2.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 1329
11.5.9.2.5. 追加の Red Hat Virtualization (RHV) 設定パラメーター 1329
11.5.9.2.6. マシンプールの追加 RHV パラメーター 1330
11.5.10. クラスターのデプロイ 1331
11.5.11. バイナリーのダウンロードによる CLI のインストール 1333
11.5.11.1. Linux への CLI のインストール 1333
11.5.11.2. Windows での CLI のインストール 1333
11.5.11.3. macOS への CLI のインストール 1334
11.5.12. クラスターへのログイン 1334
11.5.13. クラスターステータスの確認 1335
11.5.14. RHV での OpenShift Container Platform Web コンソールへのアクセス 1336
11.5.15. Red Hat Virtualization (RHV) へのインストールに関するよくある問題のトラブルシューティング 1336
11.5.15.1. CPU 負荷が増大し、ノードが Not Ready 状態になる 1336
11.5.15.2. OpenShift Container Platform クラスター API に接続できない 1337
11.5.16. インストール後のタスク 1337
11.5.17. 次のステップ 1337
11.6. RHV でのクラスターのアンインストール 1337
11.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 1338

第12章 VSPHERE へのインストール 1339
12.1. クラスターの VSPHERE へのインストール 1339
12.1.1. 前提条件 1339
12.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1339
12.1.3. VMware vSphere インフラストラクチャーの要件 1340
12.1.4. vCenter の要件 1341
必要な vCenter アカウントの権限 1341
OpenShift Container Platform と vMotion の使用 1345
クラスターリソース 1346

クラスターの制限 1346
ネットワーク要件 1346
必要な IP アドレス 1346
DNS レコード 1346
12.1.5. SSH プライベートキーの生成およびエージェントへの追加 1347
12.1.6. インストールプログラムの取得 1349
12.1.7. vCenter ルート CA 証明書のシステム信頼への追加 1349
12.1.8. クラスターのデプロイ 1350
12.1.9. バイナリーのダウンロードによる CLI のインストール 1353
12.1.9.1. Linux への CLI のインストール 1353
12.1.9.2. Windows での CLI のインストール 1354
12.1.9.3. macOS への CLI のインストール 1354
12.1.10. クラスターへのログイン 1355
12.1.11. レジストリーストレージの作成 1355
12.1.11.1. インストール時に削除されたイメージレジストリー 1355
12.1.11.2. イメージレジストリーストレージの設定 1356
12.1.11.2.1. VMware vSphere のレジストリーストレージの設定 1356
12.1.11.2.2. VMware vSphere のブロックレジストリーストレージの設定 1357
12.1.12. VMware vSphere ボリュームのバックアップ 1359
12.1.13. 次のステップ 1359
12.2. カスタマイズによる VSPHERE へのクラスターのインストール 1359
12.2.1. 前提条件 1359
12.2.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1360
12.2.3. VMware vSphere インフラストラクチャーの要件 1360
12.2.4. vCenter の要件 1361
必要な vCenter アカウントの権限 1361
OpenShift Container Platform と vMotion の使用 1365
クラスターリソース 1366
クラスターの制限 1366
ネットワーク要件 1366
必要な IP アドレス 1366
DNS レコード 1366
12.2.5. SSH プライベートキーの生成およびエージェントへの追加 1367
12.2.6. インストールプログラムの取得 1369
12.2.7. vCenter ルート CA 証明書のシステム信頼への追加 1369
12.2.8. インストール設定ファイルの作成 1370
12.2.8.1. インストール設定パラメーター 1372
12.2.8.1.1. 必須設定パラメーター 1372
12.2.8.1.2. ネットワーク設定パラメーター 1374
12.2.8.1.3. オプションの設定パラメーター 1375
12.2.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 1379
12.2.8.1.5. オプションの VMware vSphere マシンプール設定パラメーター 1380
12.2.8.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル 1381
12.2.9. クラスターのデプロイ 1382
12.2.10. バイナリーのダウンロードによる CLI のインストール 1384
12.2.10.1. Linux への CLI のインストール 1384
12.2.10.2. Windows での CLI のインストール 1385
12.2.10.3. macOS への CLI のインストール 1385
12.2.11. クラスターへのログイン 1385
12.2.12. レジストリーストレージの作成 1386
12.2.12.1. インストール時に削除されたイメージレジストリー 1386
12.2.12.2. イメージレジストリーストレージの設定 1386

12.2.12.2.1. VMware vSphere のレジストリーストレージの設定 1387
12.2.12.2.2. VMware vSphere のブロックレジストリーストレージの設定 1388
12.2.13. VMware vSphere ボリュームのバックアップ 1389
12.2.14. 次のステップ 1390
12.3. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール 1390
12.3.1. 前提条件 1390
12.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1391
12.3.3. VMware vSphere インフラストラクチャーの要件 1391
12.3.4. vCenter の要件 1392
必要な vCenter アカウントの権限 1392
OpenShift Container Platform と vMotion の使用 1396
クラスターリソース 1397
クラスターの制限 1397
ネットワーク要件 1397
必要な IP アドレス 1397
DNS レコード 1397
12.3.5. SSH プライベートキーの生成およびエージェントへの追加 1398
12.3.6. インストールプログラムの取得 1400
12.3.7. vCenter ルート CA 証明書のシステム信頼への追加 1400
12.3.8. インストール設定ファイルの作成 1401
12.3.8.1. インストール設定パラメーター 1403
12.3.8.1.1. 必須設定パラメーター 1403
12.3.8.1.2. ネットワーク設定パラメーター 1405
12.3.8.1.3. オプションの設定パラメーター 1406
12.3.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 1410
12.3.8.2. ネットワーク設定パラメーター 1410
12.3.8.3. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル 1411
12.3.9. 高度なネットワーク設定パラメーターの変更 1413
12.3.10. Cluster Network Operator (CNO) の設定 1414
12.3.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 1415
12.3.10.2. Cluster Network Operator の設定例 1416
12.3.11. クラスターのデプロイ 1416
12.3.12. バイナリーのダウンロードによる CLI のインストール 1418
12.3.12.1. Linux への CLI のインストール 1418
12.3.12.2. Windows での CLI のインストール 1419
12.3.12.3. macOS への CLI のインストール 1419
12.3.13. クラスターへのログイン 1420
12.3.14. レジストリーストレージの作成 1420
12.3.14.1. インストール時に削除されたイメージレジストリー 1420
12.3.14.2. イメージレジストリーストレージの設定 1421
12.3.14.2.1. VMware vSphere のレジストリーストレージの設定 1421
12.3.14.2.2. VMware vSphere のブロックレジストリーストレージの設定 1422
12.3.15. VMware vSphere ボリュームのバックアップ 1424
12.3.16. 次のステップ 1424
12.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール 1424
12.4.1. 前提条件 1425
12.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1425
12.4.3. VMware vSphere インフラストラクチャーの要件 1426
12.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件 1427
12.4.4.1. 必要なマシン 1427

12.4.4.2. ネットワーク接続の要件 1427
12.4.4.3. 最小リソース要件 1428
12.4.4.4. 証明書署名要求の管理 1428
12.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 1428
12.4.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 1429
ネットワークトポロジー要件 1429
ロードバランサー 1430
12.4.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 1432
12.4.6. SSH プライベートキーの生成およびエージェントへの追加 1435
12.4.7. インストールプログラムの取得 1436
12.4.8. インストール設定ファイルの手動作成 1437
12.4.8.1. VMware vSphere のサンプル install-config.yaml ファイル 1438
12.4.8.2. インストール時のクラスター全体のプロキシーの設定 1440
12.4.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1441
12.4.10. インフラストラクチャー名の抽出 1443
12.4.11. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1444
12.4.12. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1447
12.4.13. バイナリーのダウンロードによる CLI のインストール 1448
12.4.13.1. Linux への CLI のインストール 1449
12.4.13.2. Windows での CLI のインストール 1449
12.4.13.3. macOS への CLI のインストール 1450
12.4.14. クラスターの作成 1450
12.4.15. クラスターへのログイン 1451
12.4.16. マシンの証明書署名要求の承認 1452
12.4.17. Operator の初期設定 1454
12.4.17.1. インストール時に削除されたイメージレジストリー 1455
12.4.17.2. イメージレジストリーストレージの設定 1455
12.4.17.2.1. VMware vSphere のレジストリーストレージの設定 1455
12.4.17.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 1457
12.4.17.2.3. VMware vSphere のブロックレジストリーストレージの設定 1458
12.4.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1459
12.4.19. VMware vSphere ボリュームのバックアップ 1461
12.4.20. 次のステップ 1461
12.5. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール 1462
12.5.1. 前提条件 1462
12.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1462
12.5.3. VMware vSphere インフラストラクチャーの要件 1463
12.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件 1464
12.5.4.1. 必要なマシン 1464
12.5.4.2. ネットワーク接続の要件 1464
12.5.4.3. 最小リソース要件 1465
12.5.4.4. 証明書署名要求の管理 1465
12.5.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 1465
12.5.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 1466
ネットワークトポロジー要件 1466
ロードバランサー 1467
12.5.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 1469
12.5.6. SSH プライベートキーの生成およびエージェントへの追加 1472
12.5.7. インストールプログラムの取得 1473
12.5.8. インストール設定ファイルの手動作成 1474
12.5.8.1. VMware vSphere のサンプル install-config.yaml ファイル 1475
12.5.8.2. ネットワーク設定パラメーター 1477

12.5.9. 高度なネットワーク設定パラメーターの変更 1478
12.5.10. Cluster Network Operator (CNO) の設定 1479
12.5.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター 1480
12.5.10.2. Cluster Network Operator の設定例 1481
12.5.11. Ignition 設定ファイルの作成 1481
12.5.12. インフラストラクチャー名の抽出 1482
12.5.13. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1483
12.5.14. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1487
12.5.15. バイナリーのダウンロードによる CLI のインストール 1488
12.5.15.1. Linux への CLI のインストール 1488
12.5.15.2. Windows での CLI のインストール 1488
12.5.15.3. macOS への CLI のインストール 1489
12.5.16. クラスターの作成 1489
12.5.17. クラスターへのログイン 1490
12.5.18. マシンの証明書署名要求の承認 1491
12.5.19. Operator の初期設定 1493
12.5.19.1. インストール時に削除されたイメージレジストリー 1494
12.5.19.2. イメージレジストリーストレージの設定 1494
12.5.19.2.1. VMware vSphere のブロックレジストリーストレージの設定 1495
12.5.20. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1496
12.5.21. VMware vSphere ボリュームのバックアップ 1498
12.5.22. 次のステップ 1498
12.6. ネットワークが制限された環境での VSPHERE へのクラスターのインストール 1499
12.6.1. 前提条件 1499
12.6.2. ネットワークが制限された環境でのインストールについて 1499
12.6.2.1. その他の制限 1499
12.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1500
12.6.4. VMware vSphere インフラストラクチャーの要件 1500
12.6.5. vCenter の要件 1501
必要な vCenter アカウントの権限 1501
OpenShift Container Platform と vMotion の使用 1505
クラスターリソース 1506
クラスターの制限 1506
ネットワーク要件 1506
必要な IP アドレス 1506
DNS レコード 1507
12.6.6. SSH プライベートキーの生成およびエージェントへの追加 1507
12.6.7. vCenter ルート CA 証明書のシステム信頼への追加 1509
12.6.8. ネットワークが制限されたインストール用の RHCOS イメージの作成 1509
12.6.9. インストール設定ファイルの作成 1510
12.6.9.1. インストール設定パラメーター 1513
12.6.9.1.1. 必須設定パラメーター 1513
12.6.9.1.2. ネットワーク設定パラメーター 1515
12.6.9.1.3. オプションの設定パラメーター 1516
12.6.9.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター 1520
12.6.9.1.5. オプションの VMware vSphere マシンプール設定パラメーター 1521
12.6.9.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル 1522
12.6.10. クラスターのデプロイ 1524
12.6.11. バイナリーのダウンロードによる CLI のインストール 1525
12.6.11.1. Linux への CLI のインストール 1526
12.6.11.2. Windows での CLI のインストール 1526
12.6.11.3. macOS への CLI のインストール 1527

12.6.12. クラスターへのログイン 1527
12.6.13. レジストリーストレージの作成 1528
12.6.13.1. インストール時に削除されたイメージレジストリー 1528
12.6.13.2. イメージレジストリーストレージの設定 1528
12.6.13.2.1. VMware vSphere のレジストリーストレージの設定 1528
12.6.14. 次のステップ 1530
12.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での VSPHERE へのクラスターのインストール 1530
12.7.1. 前提条件 1530
12.7.2. ネットワークが制限された環境でのインストールについて 1531
12.7.2.1. その他の制限 1531
12.7.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス 1531
12.7.4. VMware vSphere インフラストラクチャーの要件 1532
12.7.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件 1533
12.7.5.1. 必要なマシン 1533
12.7.5.2. ネットワーク接続の要件 1534
12.7.5.3. 最小リソース要件 1534
12.7.5.4. 証明書署名要求の管理 1534
12.7.6. ユーザーによってプロビジョニングされるインフラストラクチャーの作成 1534
12.7.6.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件 1535
ネットワークトポロジー要件 1536
ロードバランサー 1536
12.7.6.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件 1538
12.7.7. SSH プライベートキーの生成およびエージェントへの追加 1541
12.7.8. インストール設定ファイルの手動作成 1542
12.7.8.1. VMware vSphere のサンプル install-config.yaml ファイル 1543
12.7.8.2. インストール時のクラスター全体のプロキシーの設定 1546
12.7.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成 1547
12.7.10. chrony タイムサービスの設定 1549
12.7.11. インフラストラクチャー名の抽出 1551
12.7.12. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1551
12.7.13. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成 1555
12.7.14. クラスターの作成 1556
12.7.15. クラスターへのログイン 1556
12.7.16. マシンの証明書署名要求の承認 1557
12.7.17. Operator の初期設定 1559
12.7.17.1. イメージレジストリーストレージの設定 1560
12.7.17.1.1. VMware vSphere のレジストリーストレージの設定 1560
12.7.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定 1562
12.7.17.1.3. VMware vSphere のブロックレジストリーストレージの設定 1563
12.7.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了 1564
12.7.19. VMware vSphere ボリュームのバックアップ 1566
12.7.20. 次のステップ 1566
12.8. インストーラーでプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール 1567
12.8.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除 1567

第13章 インストール設定 1569
13.1. 各種プラットフォームのサポートされているインストール方法 1569
13.2. ノードのカスタマイズ 1570
13.2.1. day-1 カーネル引数の追加 1570
13.2.2. カーネルモジュールのノードへの追加 1571

13.2.2.1. カーネルモジュールコンテナーのビルドおよびテスト 1572
13.2.2.2. カーネルモジュールの OpenShift Container Platform へのプロビジョニング 1575
13.2.2.2.1. MachineConfig オブジェクトでのカーネルモジュールのプロビジョニング 1575
13.2.3. インストール時のディスクの暗号化 1578
13.2.3.1. TPM v2 ディスク暗号化の有効化 1579
13.2.3.2. Tang ディスク暗号化の有効化 1580
13.2.4. chrony タイムサービスの設定 1582
13.2.5. 関連情報 1584
13.3. 利用可能なクラスターのカスタマイズ 1584
13.3.1. クラスター設定リソース 1584
13.3.2. Operator 設定リソース 1585
13.3.3. 追加の設定リソース 1585
13.3.4. 情報リソース 1586
13.3.5. グローバルクラスターのプルシークレットの更新 1586
13.4. ファイアウォールの設定 1587
13.4.1. OpenShift Container Platform のファイアウォールの設定 1587
13.5. プライベートクラスターの設定 1590
13.5.1. プライベートクラスター 1591
DNS 1591
Ingress コントローラー 1591
API サーバー 1591
13.5.2. DNS をプライベートに設定する 1591
13.5.3. Ingress コントローラーをプライベートに設定する 1593
13.5.4. API サーバーをプライベートに制限する 1593

第14章 インストールの問題のトラブルシューティング 1595
14.1. 前提条件 1595
14.2. 失敗したインストールのログの収集 1595
14.3. ホストへの SSH アクセスによるログの手動収集 1596
14.4. ホストへの SSH アクセスを使用しないログの手動収集 1597
14.5. インストールプログラムからのデバッグ情報の取得 1597

第15章 FIPS 暗号のサポート 1599
15.1. OPENSHIFT CONTAINER PLATFORM での FIPS 検証 1599
15.2. クラスターが使用するコンポーネントでの FIPS サポート 1599
15.2.1. etcd 1600
15.2.2. ストレージ 1600
15.2.3. ランタイム 1600
15.3. FIPS モードでのクラスターのインストール 1600

第1章 非接続インストールのイメージのミラーリング

ネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールする前に、必要なコンテナーイメージをその環境にミラーリングする必要があります。この手順を無制限のネットワークで使用して、クラスターが外部コンテンツにちて組織の制御の条件を満たすコンテナーイメージのみを使用するようにすることもできます。

重要

必要なコンテナーイメージを取得するには、インターネットへのアクセスが必要です。この手順では、ご使用のネットワークとインターネットのどちらにもアクセスできるミラーホストにミラーレジストリーを配置します。ミラーホストへのアクセスがない場合は、非接続の手順に従って、イメージをネットワークの境界をまたがって移動できるデバイスにコピーします。

1.1. 前提条件

- 以下のレジストリーのいずれかなど、OpenShift Container Platform クラスターをホストする場所に Docker v2-2 をサポートするコンテナーイメージレジストリーが必要です。
  - Red Hat Quay
  - JFrog Artifactory
  - Sonatype Nexus リポジトリー
  - Harbor

  Red Hat Quay のエンタイトルメントをお持ちの場合は、Red Hat Quay のデプロイに関するドキュメント 概念実証 (実稼働以外) 向けの Red Hat Quay のデプロイ または Quay Operator の使用による OpenShift への Red Hat Quay のデプロイ を参照してください。レジストリーの選択およびインストールがにおいてさらにサポートが必要な場合は、営業担当者または Red Hat サポートにお問い合わせください。

1.2. ミラーレジストリーについて

OpenShift Container Platform インストールおよびミラーレジストリーに対する後続の製品の更新に必要なイメージをミラーリングできます。これらのアクションは同じプロセスを使用します。コンテンツの説明が含まれるリリースイメージ、およびこれが参照するイメージがすべてミラーリングされます。さらに、Operator カタログソースイメージおよびこれが参照するイメージは、使用する Operator ごとにミラーリングする必要があります。コンテンツをミラーリングした後に、各クラスターをミラーレジストリーからこのコンテンツを取得するように設定します。

ミラーレジストリーには、Docker v2-2 をサポートする任意のコンテナーレジストリーを使用できます。すべての主要なクラウドプロバイダーレジストリー、および Red Hat Quay、Artifactory その他には、必要なサポートがあります。これらのレジストリーの 1 つを使用すると、OpenShift Container Platform で非接続環境で各イメージの整合性を検証できるようになります。

ミラーレジストリーは、プロビジョニングするクラスター内のすべてのマシンから到達できる必要があります。レジストリーに到達できない場合、インストール、更新、またはワークロードの再配置などの通常の操作が失敗する可能性があります。そのため、ミラーレジストリーは可用性の高い方法で実行し、ミラーレジストリーは少なくとも OpenShift Container Platform クラスターの実稼働環境の可用性の条件に一致している必要があります。

ミラーレジストリーを OpenShift Container Platform イメージで設定する場合、2 つのシナリオを実行

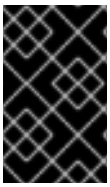
することができます。インターネットとミラーレジストリーの両方にアクセスできるホストがあり、クラスターノードにアクセスできない場合は、そのマシンからコンテンツを直接ミラーリングできます。このプロセスは、**connected mirroring** (接続ミラーリング) と呼ばれます。このようなホストがない場合は、イメージをファイルシステムにミラーリングしてから、そのホストまたはリムーバブルメディアを制限された環境に配置する必要があります。このプロセスは、**disconnected mirroring** (非接続ミラーリング) と呼ばれます。

1.3. ミラーホストの準備

ミラー手順を実行する前に、ホストを準備して、コンテンツを取得し、リモートの場所にプッシュできるようにする必要があります。

1.3.1. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

1.3.1.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

1.3.1.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

1.3.1.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

1.4. イメージのミラーリングを可能にする認証情報の設定

Red Hat からミラーへのイメージのミラーリングを可能にするコンテナーイメージレジストリーの認証情報ファイルを作成します。

警告

クラスターのインストール時に、このイメージレジストリー認証情報ファイルをプルシークレットとして使用しないでください。クラスターのインストール時にこのファイルを指定すると、クラスター内のすべてのマシンにミラーレジストリーへの書き込みアクセスが付与されます。

警告

このプロセスでは、ミラーレジストリーのコンテナーイメージレジストリーへの書き込みアクセスがあり、認証情報をレジストリープルシークレットに追加する必要があります。

前提条件

- ネットワークが制限された環境で使用するミラーレジストリーを設定していること。
- イメージをミラーリングするミラーレジストリー上のイメージリポジトリーの場所を特定している。
- イメージのイメージリポジトリーへのアップロードを許可するミラーレジストリーアカウントをプロビジョニングしている。

手順

インストールホストで以下の手順を実行します。

1. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから **registry.redhat.io** プルシークレットをダウンロードし、これを **.json** ファイルに保存します。
2. ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードまたはトークンを生成します。

```
$ echo -n '<user_name>:<password>' | base64 -w0 (1)
BGVtbYk3ZHAtqXs=
```

(1) **<user_name>** および **<password>** については、レジストリーに設定したユーザー名およびパスワードを指定します。

3. JSON 形式でプルシークレットのコピーを作成します。

```
$ cat ./pull-secret.text | jq .  > <path>/<pull-secret-file> (1)
```

(1) プルシークレットを保存するフォルダーへのパスおよび作成する JSON ファイルの名前を指定します。

ファイルの内容は以下の例のようになります。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

4. 新規ファイルを編集し、レジストリーについて記述するセクションをこれに追加します。

```
  "auths": {
    "<mirror_registry>": { 1
      "auth": "<credentials>", 2
      "email": "you@example.com"
  },
```

1 **<mirror_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000**

2 **<credentials>** については、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

ファイルは以下の例のようになります。

```
{
  "auths": {
    "<mirror_registry>": {
      "auth": "<credentials>",
      "email": "you@example.com"
    },
    "cloud.openshift.com": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
    "quay.io": {
      "auth": "b3BlbnNo...",
      "email": "you@example.com"
    },
```

```
    "registry.connect.redhat.com": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    },
    "registry.redhat.io": {
      "auth": "NTE3Njg5Nj...",
      "email": "you@example.com"
    }
  }
}
```

1.5. OPENSHIFT CONTAINER PLATFORM イメージリポジトリーのミラーリング

クラスターのインストールまたはアップグレード時に使用するために、OpenShift Container Platform イメージリポジトリーをお使いのレジストリーにミラーリングします。

前提条件

- ミラーホストがインターネットにアクセスできる。
- ネットワークが制限された環境で使用するミラーレジストリーを設定し、設定した証明書および認証情報にアクセスできる。
- Red Hat OpenShift Cluster Manager のサイトの Pull Secret ページからプルシークレットをダウンロードしており、ミラーリポジトリーに認証を組み込むようにこれを変更している。
- Subject Alternative Name が設定されていない自己署名証明書を使用する場合は、この手順の **oc** コマンドの前に **GODEBUG=x509ignoreCN=0** を追加する必要があります。この変数を設定しない場合、**oc** コマンドは以下のエラーを出して失敗します。

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common Name matching with GODEBUG=x509ignoreCN=0
```

手順

ミラーホストで以下の手順を実行します。

1. OpenShift Container Platform ダウンロード ページを確認し、インストールする必要のある OpenShift Container Platform のバージョンを判別し、Repository Tags ページで対応するタグを判別します。
2. 必要な環境変数を設定します。
   a. リリースバージョンをエクスポートします。

```
$ OCP_RELEASE=<release_version>
```

   <release_version> について、インストールする OpenShift Container Platform のバージョンに対応するタグを指定します (例: **4.5.4**)。

   b. ローカルレジストリー名とホストポートをエクスポートします。

```
$ LOCAL_REGISTRY='<local_registry_host_name>:<local_registry_host_port>'
```

<local_registry_host_name> については、ミラーレジストリーのレジストリードメイン名を指定し、**<local_registry_host_port>** については、コンテンツの送信に使用するポートを指定します。

c. ローカルリポジトリー名をエクスポートします。

```
$ LOCAL_REPOSITORY='<local_repository_name>'
```

<local_repository_name> については、**ocp4/openshift4** などのレジストリーに作成するリポジトリーの名前を指定します。

d. ミラーリングするリポジトリーの名前をエクスポートします。

```
$ PRODUCT_REPO='openshift-release-dev'
```

実稼働環境のリリースの場合には、**openshift-release-dev** を指定する必要があります。

e. パスをレジストリープルシークレットにエクスポートします。

```
$ LOCAL_SECRET_JSON='<path_to_pull_secret>'
```

<path_to_pull_secret> については、作成したミラーレジストリーのプルシークレットの絶対パスおよびファイル名を指定します。

f. リリースミラーをエクスポートします。

```
$ RELEASE_NAME="ocp-release"
```

実稼働環境のリリースについては、**ocp-release** を指定する必要があります。

g. サーバーのアーキテクチャーのタイプをエクスポートします (例: **x86_64**)。

```
$ ARCHITECTURE=<server_architecture>
```

h. ミラーリングされたイメージをホストするためにディレクトリーへのパスをエクスポートします。

```
$ REMOVABLE_MEDIA_PATH=<path> 1
```

1 最初のスラッシュ (/) 文字を含む完全パスを指定します。

3. バージョンイメージを内部コンテナーレジストリーにミラーリングします。

- ミラーホストがインターネットにアクセスできない場合は、以下の操作を実行します。

  i. リムーバブルメディアをインターネットに接続しているシステムに接続します。

  ii. ミラーリングするイメージおよび設定マニフェストを確認します。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
```

```
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE} --dry-run
```

iii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。

iv. イメージをリムーバブルメディア上のディレクトリーにミラーリングします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-
dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE}
```

v. メディアをネットワークが制限された環境に移し、イメージをローカルコンテナーレジストリーにアップロードします。

```
$ oc image mirror -a ${LOCAL_SECRET_JSON} --from-
dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*"
${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} 1
```

1 **REMOVABLE_MEDIA_PATH** の場合、イメージのミラーリング時に指定した同じパスを使用する必要があります。

- ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下の操作を実行します。

i. 以下のコマンドを使用して、リリースイメージをローカルレジストリーに直接プッシュします。

```
$ oc adm release mirror -a ${LOCAL_SECRET_JSON}  \
    --from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-
${ARCHITECTURE} \
    --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} \
    --to-release-
image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
${ARCHITECTURE}
```

このコマンドは、リリース情報をダイジェストとしてプルします。その出力には、クラスターのインストール時に必要な **imageContentSources** データが含まれます。

ii. 直前のコマンドの出力の **imageContentSources** セクション全体を記録します。ミラーの情報はミラーリングされたリポジトリーに一意であり、インストール時に **imageContentSources** セクションを **install-config.yaml** ファイルに追加する必要があります。

注記

ミラーリングプロセス中にイメージ名に Quay.io のパッチが適用され、podman イメージにはブートストラップ仮想マシンのレジストリーに Quay.io が表示されます。

4. ミラーリングしたコンテンツをベースとしているインストールプログラムを作成するには、これを展開し、リリースに固定します。

   - ミラーホストがインターネットにアクセスできない場合は、以下のコマンドを実行します。

     ```
     $ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
     "${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}"
     ```

   - ローカルコンテナーレジストリーがミラーホストに接続されている場合は、以下のコマンドを実行します。

     ```
     $ oc adm release extract -a ${LOCAL_SECRET_JSON} --command=openshift-install
     "${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-
     ${ARCHITECTURE}"
     ```

重要

選択した OpenShift Container Platform バージョンに適したイメージを使用するには、ミラーリングされたコンテンツからインストールプログラムを展開する必要があります。

インターネット接続のあるマシンで、このステップを実行する必要があります。

非接続環境を使用している場合には、must-gather の一部として **--image** フラグを使用し、ペイロードイメージを参照します。

1.6. 非接続環境の CLUSTER SAMPLES OPERATOR

非接続環境で Cluster Samples Operator を設定するには、クラスターのインストール後に追加の手順を実行する必要があります。

- クラスターにインストールする Operator の OperatorHub イメージを ミラーリング します。
- VMware vSphere、ベアメタル、または Amazon Web Services など、ネットワークが制限された環境でプロビジョニングするインフラストラクチャーにクラスターをインストールします。

1.7. 関連情報

- must-gather の使用についての詳細は、特定の機能についてのデータの収集 について参照してください。

第2章 AWS へのインストール

2.1. AWS アカウントの設定

OpenShift Container Platform をインストールする前に、Amazon Web Services (AWS) アカウントを設定する必要があります。

2.1.1. Route 53 の設定

OpenShift Container Platform をインストールするには、使用する Amazon Web Services (AWS) アカウントに、Route 53 サービスの専用のパブリックホストゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。Route 53 サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、または AWS または他のソースから新規のものを取得できます。

注記

AWS で新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかります。AWS 経由でドメインを購入する方法についての詳細は、AWS ドキュメントの Registering Domain Names Using Amazon Route 53 を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を AWS に移行します。AWS ドキュメントの Making Amazon Route 53 the DNS Service for an Existing Domain を参照してください。

3. ドメインまたはサブドメインのパブリックホストゾーンを作成します。AWS ドキュメントの Creating a Public Hosted Zone を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。AWS ドキュメントの Getting the Name Servers for a Public Hosted Zone を参照してください。

5. ドメインが使用する AWS Route 53 ネームサーバーのレジストラーレコードを更新します。たとえば、別のアカウントを使ってドメインを Route 53 サービスに登録している場合は、AWS ドキュメントの Adding or Changing Name Servers or Glue Records のトピックを参照してください。

6. サブドメインを使用している場合は、その委任レコードを親ドメインに追加します。これにより、サブドメインの Amazon Route 53 の責任が付与されます。親ドメインの DNS プロバイダーによって要約された委任手順に従います。高次元の手順の例については、AWS ドキュメントの Creating a subdomain that uses Amazon Route 53 as the DNS service without migrating the parent domain を参照してください。

2.1.2. AWS アカウントの制限

OpenShift Container Platform クラスターは数多くの Amazon Web Services (AWS) コンポーネントを使用し、デフォルトの サービス制限 は、OpenShift Container Platform クラスターをインストールする機

能に影響を与えます。特定のクラスター設定を使用し、クラスターを特定の AWS リージョンにデプロイするか、またはアカウントを使って複数のクラスターを実行する場合、AWS アカウントの追加リソースを要求することが必要になる場合があります。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある AWS コンポーネントの制限を要約しています。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
インスタンスの制限	変動あり。	変動あり。	デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのマスターノード ● 3 つのワーカーノード これらのインスタンスタイプの数は、新規アカウントのデフォルト制限内の値です。追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの制限を見直し、クラスターが必要なマシンをデプロイできることを確認します。 ほとんどのリージョンでは、ブートストラップおよびワーカーマシンは **m4.large** マシンを使用し、マスターマシンは **m4.xlarge** インスタンスを使用します。これらのインスタンスタイプをサポートしないすべてのリージョンを含む一部のリージョンでは、**m5.large** および **m5.xlarge** インスタンスが代わりに使用されます。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
Elastic IP (EIP)	0 - 1	アカウントごとに 5 つの EIP	クラスターを高可用性設定でプロビジョニングするために、インストールプログラムはそれぞれの リージョン内のアベイラビリティーゾーン にパブリックおよびプライベートのサブネットを作成します。各プライベートサブネットには NAT ゲートウェイ が必要であり、各 NAT ゲートウェイには別個の Elastic IP が必要です。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。デフォルトの高可用性を利用するには、少なくとも 3 つのアベイラビリティーゾーンがあるリージョンにクラスターをインストールします。アベイラビリティーゾーンが 6 つ以上あるリージョンにクラスターをインストールするには、EIP 制限を引き上げる必要があります。  **重要** **us-east-1** リージョンを使用するには、アカウントの EIP 制限を引き上げる必要があります。
Virtual Private Cloud (VPC)	5	リージョンごとに 5 つの VPC	各クラスターは独自の VPC を作成します。
Elastic Load Balancing (ELB/NLB)	3	リージョンごとに 20	デフォルトで、各クラスターは、マスター API サーバーの内部および外部のネットワークロードバランサーおよびルーターの単一の Classic Elastic Load Balancer を作成します。追加の Kubernetes **Service** オブジェクトをタイプ **LoadBalancer** を指定してデプロイすると、追加の ロードバランサー が作成されます。
NAT ゲートウェイ	5	アベイラビリティゾーンごとに 5 つ	クラスターは各アベイラビリティーゾーンに 1 つの NAT ゲートウェイをデプロイします。

コンポーネント	デフォルトで利用できるクラスターの数	デフォルトの AWS の制限	説明
Elastic Network Interface (ENI)	12 以上	リージョンごとに 350	デフォルトのインストールは 21 の ENI を作成し、リージョンの各アベイラビリティーゾーンに 1 つの ENI を作成します。たとえば、**us-east-1** リージョンには 6 つのアベイラビリティーゾーンが含まれるため、そのゾーンにデプロイされるクラスターは 27 の ENI を使用します。AWS リージョンマップ を確認して、各リージョンにあるアベイラビリティーゾーンの数を判別します。 追加の ENI が、クラスターの使用およびデプロイされたワークロード別に作成される追加のマシンおよび Elastic Load Balancer について作成されます。
VPC ゲートウェイ	20	アカウントごとに 20	各クラスターは、S3 アクセス用の単一の VPC ゲートウェイを作成します。
S3 バケット	99	アカウントごとに 100 バケット	インストールプロセスでは 1 つの一時的なバケットを作成し、各クラスターのレジストリーコンポーネントがバケットを作成するため、AWS アカウントごとに 99 の OpenShift Container Platform クラスターのみを作成できます。
セキュリティーグループ	250	アカウントごとに 2,500	各クラスターは、10 の個別のセキュリティーグループを作成します。

2.1.3. 必要な AWS パーミッション

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例2.1 インストールに必要な EC2 パーミッション

- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**

- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**

- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例2.2 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**

注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

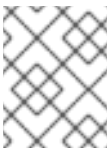
例2.3 インストールに必要な Elastic Load Balancing のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

例2.4 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**

- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例2.5 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**

- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例2.6 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例2.7 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**

- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

例2.8 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例2.9 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**

- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**

注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。

例2.10 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

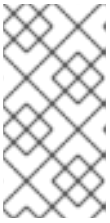
2.1.4. IAM ユーザーの作成

各 Amazon Web Services (AWS) アカウントには、アカウントの作成に使用するメールアドレスに基づく root ユーザーアカウントが含まれます。これは高度な権限が付与されたアカウントであり、初期アカウントにのみ使用し、請求設定また初期のユーザーセットの作成およびアカウントのセキュリティー保護のために使用することが推奨されています。

OpenShift Container Platform をインストールする前に、セカンダリー IAM 管理ユーザーを作成します。AWS ドキュメントの Creating an IAM User in Your AWS Account 手順を実行する際に、以下のオプションを設定します。

手順

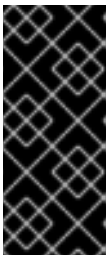
1. IAM ユーザー名を指定し、**Programmatic access** を選択します。
2. **AdministratorAccess** ポリシーを割り当て、アカウントにクラスターを作成するために十分なパーミッションがあることを確認します。このポリシーはクラスターに対し、各 OpenShift Container Platform コンポーネントに認証情報を付与する機能を提供します。クラスターはコンポーネントに対し、それらが必要とする認証情報のみを付与します。



注記

必要なすべての AWS パーミッションを付与し、これをユーザーに割り当てるポリシーを作成することは可能ですが、これは優先されるオプションではありません。クラスターには追加の認証情報を個別コンポーネントに付与する機能がないため、同じ認証情報がすべてのコンポーネントによって使用されます。

3. オプション: タグを割り当て、メタデータをユーザーに追加します。
4. 指定したユーザー名に **AdministratorAccess** ポリシーが付与されていることを確認します。
5. アクセスキー ID およびシークレットアクセスキーの値を記録します。ローカルマシンをインストールプログラムを実行するように設定する際にこれらの値を使用する必要があります。



重要

クラスターのデプロイ時に、マルチファクター認証デバイスの使用中に生成した一時的なセッショントークンを使用して AWS に対する認証を行うことはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。

2.1.5. サポートされている AWS リージョン

OpenShift Container Platform クラスターを以下のリージョンにデプロイできます。

- **ap-northeast-1** (Tokyo)
- **ap-northeast-2** (Seoul)
- **ap-south-1** (Mumbai)
- **ap-southeast-1** (Singapore)
- **ap-southeast-2** (Sydney)
- **ca-central-1** (Central)
- **eu-central-1** (Frankfurt)
- **eu-north-1** (Stockholm)

- **eu-west-1** (Ireland)
- **eu-west-2** (London)
- **eu-west-3** (Paris)
- **me-south-1** (Bahrain)
- **sa-east-1** (São Paulo)
- **us-east-1** (N. Virginia)
- **us-east-2** (Ohio)
- **us-west-1** (N. California)
- **us-west-2** (Oregon)

2.1.6. 次のステップ

- OpenShift Container Platform クラスターをインストールします。
  - インストーラーでプロビジョニングされるインフラストラクチャーのデフォルトオプションを使用した クラスターのクイックインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのクラウドのカスタマイズを使用したクラスターのインストール
  - インストーラーでプロビジョニングされるインフラストラクチャーへのネットワークのカスタマイズを使用したクラスターのインストール
  - CloudFormation テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

2.2. AWS の IAM の手動作成

2.2.1. IAM の手動作成

Cloud Credential Operator は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. OpenShift Container Platform インストーラーを実行し、マニフェストを生成します。

```
$ openshift-install create manifests --dir=mycluster
```

2. Cloud Credential Operator が手動モードになるように、設定マップを manifests ディレクトリーに挿入します。

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. ローカルクラウドの認証情報を使用して作成された **admin** 認証情報シークレットを削除します。この削除により、**admin** 認証情報がクラスターに保存されなくなります。

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. OpenShift Container Platform リリースイメージを取得します。**openshift-install** バイナリーはこれを使用するためにビルドされます。

```
$ bin/openshift-install version
```

出力例

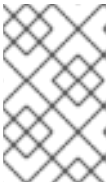
```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```

6. 展開したファイルで **CredentialsRequests** を見つけます。

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```



注記

今後の OpenShift Container Platform リリースでは、**CredentialsRequests** をスキャンし、それらを表示する新規の **oc adm release** コマンドが表示されます。

これにより、それぞれの要求の詳細が表示されます。**spec.providerSpec.kind** がインストールするクラウドプロバイダーと一致しない **CredentialsRequests** については、これを無視するようにしてください。

サンプル CredentialsRequest オブジェクト

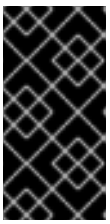
```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  name: cloud-credential-operator-iam-ro
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: cloud-credential-operator-iam-ro-creds
```

```
    namespace: openshift-cloud-credential-operator
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AWSProviderSpec
    statementEntries:
    - effect: Allow
      action:
      - iam:GetUser
      - iam:GetUserPolicy
      - iam:ListAccessKeys
      resource: "*"
```

7. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、各 **request.spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

8. クラスターの作成に進みます。

```
$ openshift-install create cluster --dir=mycluster
```



重要

アップグレードを実行する前に、パーミッションが次のリリースで変更された場合には、認証情報の調整が必要になる場合があります。今後は、Cloud Credential Operator は更新されたパーミッションが処理されることを示唆するまでアップグレードできなくなる可能性があります。

2.2.2. 管理者の認証情報のルートシークレット形式

各クラウドプロバイダーは、**kube-system** namespace の認証情報ルートシークレットを使用します。これは、すべての認証情報要求を満たし、それぞれのシークレットを作成するために使用されます。これは、**Mint モード** で新規の認証情報を作成するか、または **Passthrough モード** で認証情報ルートシークレットをコピーして実行します。

シークレットの形式はクラウドごとに異なり、それぞれの **CredentialsRequest** シークレットにも使用されます。

Amazon Web Services (AWS) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <AccessKeyID>
  aws_secret_access_key: <SecretAccessKey>
```

2.2.2.1. アップグレード

今後のリリースでは、Cloud Credential Operator の改善により、手動でメンテナーンスされる認証情報が今後のリリースのイメージの **CredentialsRequest** オブジェクトに一致するように更新されていないことが原因でアップグレードが失敗する可能性が生じる状況を避けることができます。

2.2.3. mint モード

mint モードは AWS、GCP および Azure でサポートされます。

OpenShift Container Platform を実行するためのデフォルトおよび推奨されるベストプラクティスとして、管理者レベルのクラウド認証情報を使用してインストーラーを実行できます。**admin** 認証情報は **kube-system** namespace に保存され、次に Cloud Credential Operator によってクラスターの **CredentialsRequest** オブジェクトを処理し、特定のパーミッションでそれぞれの新規ユーザーを作成するために使用されます。

mint モードには以下の利点があります。

- 各クラスターコンポーネントにはそれぞれが必要なパーミッションのみがあります。
- アップグレードを含むクラウド認証情報の自動の継続的な調整が行われます。これには、追加の認証情報またはパーミッションが必要になる可能性があります。

1 つの不利な点として、mint モードでは、**admin** 認証情報がクラスターの **kube-system** シークレットに保存される必要があります。

2.2.4. 管理者認証情報の削除またはローテーション機能を持つ mint モード

現時点で、このモードは AWS でのみサポートされます。

このモードでは、ユーザーは通常の mint モードと同様に **admin** 認証情報を使用して OpenShift Container Platform をインストールします。ただし、このモードはクラスターのインストール後に **admin** 認証情報シークレットを削除します。

管理者は、Cloud Credential Operator に読み取り専用の認証情報について独自の要求を行わせることができます。これにより、すべての **CredentialsRequest** オブジェクトに必要なパーミッションがあることの確認が可能になります。そのため、いずれかの変更が必要にならない限り **admin** 認証情報は必要になりません。関連付けられた認証情報が削除された後に、必要な場合は、これは基礎となるクラウドで破棄できます。

アップグレードの前に、**admin** の認証情報を復元する必要があります。今後は、認証情報が存在しない場合に、アップグレードがブロックされる可能性があります。

admin 認証情報はクラスターに永続的に保存されません。

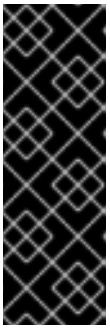
このモードでは、短い期間にクラスターでの **admin** 認証情報が必要になります。また、アップグレードごとに **admin** 認証情報を使用してシークレットを手動で再インストールする必要があります。

2.3. クラスターの AWS へのクイックインストール

OpenShift Container Platform バージョン 4.5 では、デフォルトの設定オプションを使用してクラスターを Amazon Web Services (AWS) にインストールできます。

2.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

2.3.3. SSH プライベートキーの生成およびエージェントへの追加

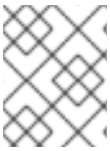
クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインス

トールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.3.4. インストールプログラムの取得

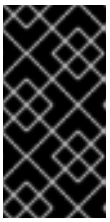
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

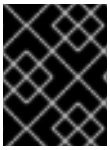
3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

2.3.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

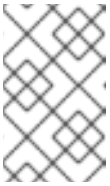
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **aws** を選択します。

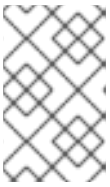
c. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

d. クラスターのデプロイ先とする AWS リージョンを選択します。

e. クラスターに設定した Route 53 サービスのベースドメインを選択します。

f. クラスターの記述名を入力します。

g. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。



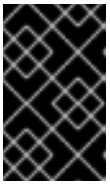
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

2.3.6. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.3.6.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.3.6.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。

PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.3.6.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.3.7. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.3.8. 次のステップ

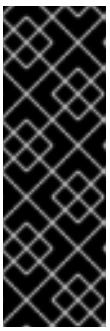
- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

2.4. カスタマイズによる AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストールプログラムが Amazon Web Services (AWS) にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールすることができます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

2.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

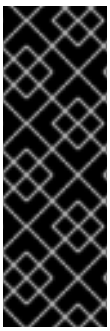
2.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

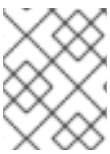
2.4.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
    -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.4.4. インストールプログラムの取得

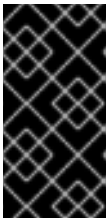
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

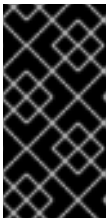
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

2.4.5. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

      iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

      iv. クラスターのデプロイ先とする AWS リージョンを選択します。

      v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

      vi. クラスターの記述名を入力します。

      vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

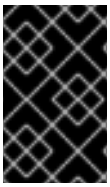
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.4.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

2.4.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

2.4.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表2.2 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

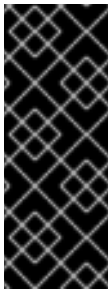
パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

2.4.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表2.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.4.5.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

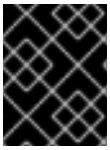
表2.4 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	ルートボリュームのインスタンスタイプ。	有効な AWS EBS インスタンスタイプ (例: **io1**)。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。

パラメーター	説明	値
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

2.4.5.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
      type: m5.xlarge (5)
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (8)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (9)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (10)
    userTags:
      adminContact: jdoe
      costCenter: 7536
fips: false (11)
sshKey: ssh-ed25519 AAAA... (12)
pullSecret: '{"auths": ...}' (13)
```

(1) (9) (10) (13) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (6) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の

4 5 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

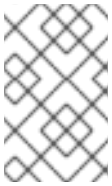
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

8 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

11 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

12 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

2.4.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

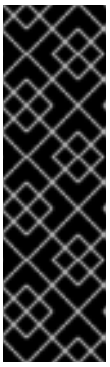
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

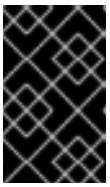
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



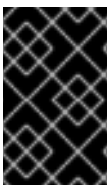
重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

2.4.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.4.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.4.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.4.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.4.8. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.4.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

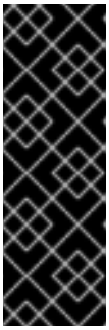
2.5. ネットワークのカスタマイズによる AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、カスタマイズされたネットワーク設定オプションでクラスターを Amazon Web Services (AWS) にインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

2.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

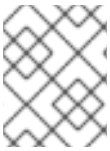
2.5.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.5.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

2.5.5. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

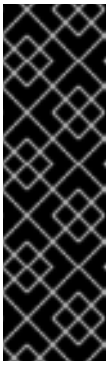
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

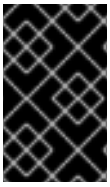
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.5.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

2.5.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.5 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

2.5.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表2.6 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

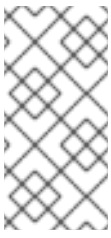
2.5.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表2.7 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.5.5.1.4. オプションの AWS 設定パラメーター

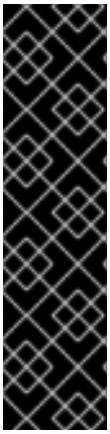
オプションの AWS 設定パラメーターは、以下の表で説明されています。

表2.8 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	ルートボリュームのインスタンスタイプ。	有効な AWS EBS インスタンスタイプ (例: **io1**)。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。

パラメーター	説明	値
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID です。



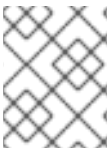
重要

Open Virtual Networking (OVN) Kubernetes ネットワークプラグインは、テクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

OVN テクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/articles/4380121 を参照してください。

2.5.5.2. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。



注記

インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

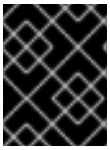
表2.9 必要なネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。**OVNKubernetes** プラグインは、OpenShift Container Platform 4.5 でテクノロジープレビューとしてご利用いただけます。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの **serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

2.5.5.3. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
```

```
  hyperthreading: Enabled 3 4
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
      type: m5.xlarge 5
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 8
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster 9
networking: 10
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 11
    userTags:
      adminContact: jdoe
      costCenter: 7536
fips: false 12
sshKey: ssh-ed25519 AAAA... 13
pullSecret: '{"auths": ...}' 14
```

1 9 11 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 10 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができませ

ん。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 5 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

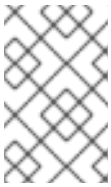
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

8 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

13 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

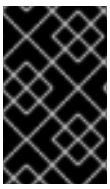


注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

2.5.6. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

2.5.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

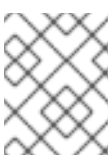
```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
    ...
  kubeProxyConfig: 4
    iptablesSyncPeriod: 30s 5
    proxyArguments:
      iptables-min-sync-period: 6
      - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。



注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go

2.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

1 **install-config.yaml** ファイルに指定されます。

2 OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

3 OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

4 VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

5 すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

2.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OVNKubernetes 1
  ovnKubernetesConfig: 2
```

```
    mtu: 1400 3
    genevePort: 6081 4
```

1 **install-config.yaml** ファイルに指定されます。

2 OVN-Kubernetes 設定の一部を上書きする必要がある場合にのみ指定します。

3 Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。

4 Geneve オーバーレイネットワークの UDP ポート。

2.5.7.3. Cluster Network Operator の設定例

以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
      - 0s
```

2.5.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

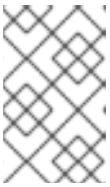
手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

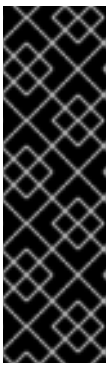
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

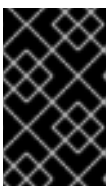
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

2.5.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.5.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.5.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。

5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.5.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.5.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.5.11. 次のステップ

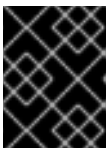
- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

2.6. ネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、既存の Amazon Virtual Private Cloud (VPC) にインストールリリースコンテンツの内部ミラーを作成することにより、制限付きネットワークの Amazon Web Services (AWS) にクラスターをインストールできます。

2.6.1. 前提条件

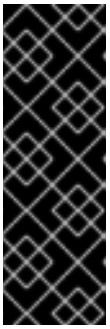
- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- AWS に既存の VPC が必要です。インストーラーでプロビジョニングされるインフラストラクチャーを使用してネットワークが制限された環境にインストールする場合は、インストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- クラスターをホストするために AWS アカウントを設定 している。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールしている。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。
- クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 している (ファイアウォールを使用し、Telemetry サービスを使用する予定の場合)。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.6.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

2.6.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

2.6.3. カスタム VPC の使用について

OpenShift Container Platform 4.5 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

2.6.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC
- VPC DHCP オプション
- VPC エンドポイント

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC は以下の特性を満たす必要があります。

- VPC の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれている必要があります。
- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。

パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。

インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで現在の タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要がありま

す。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。
パブリックサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::SubnetNetworkAclAssociation**	VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。
インターネットゲートウェイ	• **AWS::EC2::InternetGateway** • **AWS::EC2::VPCGatewayAttachment** • **AWS::EC2::RouteTable** • **AWS::EC2::Route** • **AWS::EC2::SubnetRouteTableAssociation** • **AWS::EC2::NatGateway** • **AWS::EC2::EIP**	VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。
ネットワークアクセス制御	• **AWS::EC2::NetworkAcl** • **AWS::EC2::NetworkAclEntry**	VPC が以下のポートにアクセスできるようにする必要があります。 **ポート** / **理由** **80** / インバウンド HTTP トラフィック

コンポーネント	AWS タイプ	説明	
		443	インバウンド HTTPS トラフィック
		22	インバウンド SSH トラフィック
		1024 - 65535	インバウンド一時 (ephemeral) トラフィック
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

2.6.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

2.6.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

2.6.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

2.6.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

追加リソース

- Telemetry サービスの詳細は、リモートヘルスモニターリング を参照してください。

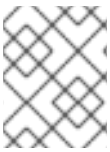
2.6.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.6.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。ネットワークが制限されたインストールでは、これらのファイルが bastion ホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得します。

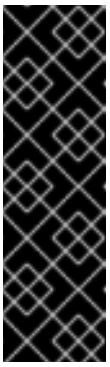
手順

1. **install-config.yaml** ファイルを作成します。

a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。

iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<bastion_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. クラスターをインストールする VPC のサブネットを定義します。

```
subnets:
- subnet-1
- subnet-2
- subnet-3
```

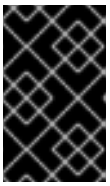
d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.6.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

2.6.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.10 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

2.6.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表2.11 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト  **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

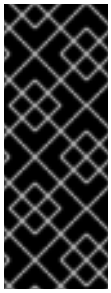
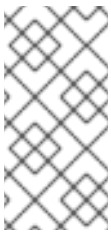
2.6.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表2.12 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.6.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

表2.13 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。

パラメーター	説明	値
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	ルートボリュームのインスタンスタイプ。	有効な AWS EBS インスタンスタイプ (例: **io1**)。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。

パラメーター	説明	値
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

2.6.6.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
      type: m5.xlarge (5)
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    aws:
      rootVolume:
```

```
        iops: 2000
        size: 500
        type: io1 8
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster 9
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 10
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: 11
    - subnet-1
    - subnet-2
    - subnet-3
fips: false 12
sshKey: ssh-ed25519 AAAA... 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
additionalTrustBundle: | 15
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: 16
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

1 9 10 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 5 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

重要

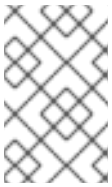
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

8 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

11 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

13 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

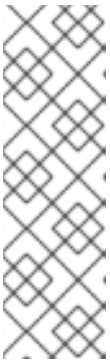
16 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

2.6.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

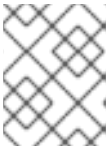
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

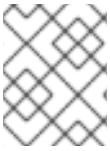


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

2.6.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

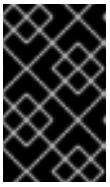
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

2.6.8. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.6.8.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。

PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.6.8.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.6.8.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.6.9. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.6.10. 次のステップ

- クラスターのカスタマイズ
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

2.7. AWS のクラスターの既存 VPC へのインストール

OpenShift Container Platform バージョン 4.5 では、クラスターを Amazon Web Services (AWS) の既存の Amazon Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、カスタ

マイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

2.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.7.2. カスタム VPC の使用について

OpenShift Container Platform 4.5 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

2.7.2.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC

- VPC DHCP オプション
- VPC エンドポイント

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC は以下の特性を満たす必要があります。

- VPC の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれている必要があります。
- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。

パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。

インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで現在の タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

2.7.2.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

2.7.2.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

2.7.2.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

2.7.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

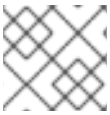
- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

2.7.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
-
```

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.7.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

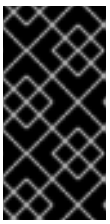
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシー

クレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

2.7.6. インストール設定ファイルの作成

Amazon Web Services (AWS) での OpenShift Container Platform のインストールをカスタマイズできます。

前提条件

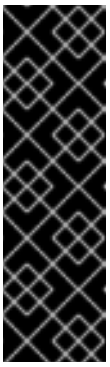
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **AWS** を選択します。
      iii. Amazon Web Services (AWS) プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。
      iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

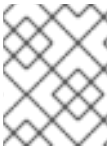
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.7.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

2.7.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.14 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

2.7.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表2.15 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

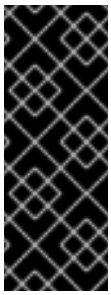
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

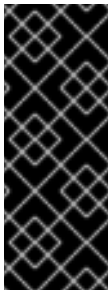
2.7.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

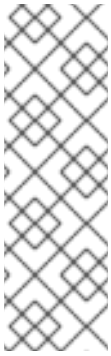
表2.16 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.7.6.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

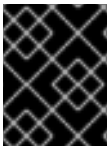
表2.17 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	ルートボリュームのインスタンスタイプ。	有効な AWS EBS インスタンスタイプ (例: **io1**)。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。

パラメーター	説明	値
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

2.7.6.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
      type: m5.xlarge (5)
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (8)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (9)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (10)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (11)
    - subnet-1
    - subnet-2
    - subnet-3
fips: false (12)
sshKey: ssh-ed25519 AAAA... (13)
pullSecret: '{"auths": ...}' (14)
```

1 9 10 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 5 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

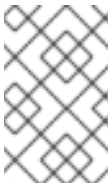
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

8 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

11 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

13 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

2.7.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要がある

かどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名され

ない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

2.7.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2. オプション: クラスターのインストールに使用した IAM アカウントから **AdministratorAccess** ポリシーを削除するか、または無効にします。

2.7.8. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.7.8.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。

4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.7.8.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.7.8.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.7.9. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.7.10. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

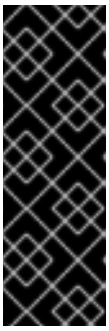
2.8. プライベートクラスターの AWS へのインストール

OpenShift Container Platform バージョン 4.5 では、プライベートクラスターを Amazon Web Services (AWS) の既存の VPC にインストールできます。インストールプログラムは、カスタマイズ可能な残り

の必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

2.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.8.2. プライベートクラスター

お使いの環境で外部のインターネット接続を必要としない場合には、外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセス可能で、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

2.8.2.1. AWS のプライベートクラスター

Amazon Web Services (AWS) でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、プライベートネットワークからのみアクセスできるように Ingress Operator および API サーバーを設定します。

クラスターには、引き続き AWS API にアクセスするためにインターネットへのアクセスが必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット
- パブリック Ingress をサポートするパブリックロードバランサー
- クラスターの **baseDomain** に一致するパブリック Route 53 ゾーン

インストールプログラムは、プライベート Route 53 ゾーンを作成するために指定する **baseDomain** とクラスターに必要なレコードを使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

2.8.2.1.1. 制限事項

プライベートクラスターにパブリック機能を追加する機能には制限があります。

- Kubernetes API エンドポイントは、追加のアクションを実行せずにインストールする場合はパブリックにすることができません。これらのアクションには、使用中のアベイラビリティーゾーンごとに VPC でパブリックサブネットやパブリックのロードバランサーを作成することや、6443 のインターネットからのトラフィックを許可するようにコントロールプレーンのセキュリティーグループを設定することなどが含まれます。
- パブリックのサービスタイプのロードバランサーを使用する場合には、各アベイラビリティーゾーンのパブリックサブネットに **kubernetes.io/cluster/<cluster-infra-id>: shared** のタグを付け、AWS がそれらを使用してパブリックロードバランサーを作成できるようにします。

2.8.3. カスタム VPC の使用について

OpenShift Container Platform 4.5 では、Amazon Web Services (AWS) の既存 Amazon Virtual Private Cloud (VPC) の既存のサブネットにクラスターをデプロイできます。OpenShift Container Platform を既存の AWS VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC を作成するために必要なインフラストラクチャーの作成パーミッションを取得できない場合は、このインストールオプションを使用します。

インストールプログラムは既存のサブネットにある他のコンポーネントを把握できないため、ユーザーの代わりにサブネットの CIDR を選択することはできません。クラスターをインストールするサブネットのネットワークを独自に設定する必要があります。

2.8.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなりました。

- インターネットゲートウェイ
- NAT ゲートウェイ
- サブネット
- ルートテーブル
- VPC

- VPC DHCP オプション
- VPC エンドポイント

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC は以下の特性を満たす必要があります。

- VPC の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれている必要があります。
- VPC は **kubernetes.io/cluster/.*: owned** タグを使用できません。
- VPC で **enableDnsSupport** および **enableDnsHostnames** 属性を有効にし、クラスターが VPC に割り当てられている Route 53 ゾーンを使用してクラスターの内部 DNS レコードを解決できるようにする必要があります。AWS ドキュメントの DNS Support in Your VPC を参照してください。

パブリックアクセスでクラスターを使用する場合、クラスターが使用する各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを作成する必要があります。

インストールプログラムは **kubernetes.io/cluster/.*: shared** タグを追加するようにサブネットを変更するため、サブネットでは 1 つ以上の空のタグスロットが利用可能である必要があります。AWS ドキュメントで現在の タグ制限 を確認し、インストールプログラムでタグを指定する各サブネットに追加できるようにします。

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	• **AWS::EC2::VPC** • **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。

2.8.3.2. VPC 検証

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- プライベートサブネットを指定します。
- サブネットの CIDR は指定されたマシン CIDR に属します。
- 各アベイラビリティーゾーンのサブネットを指定します。それぞれのアベイラビリティーゾーンには、複数のパブリックおよびプライベートサブネットがありません。プライベートクラスターを使用する場合、各アベイラビリティーゾーンのプライベートサブネットのみを指定します。それ以外の場合は、各アベイラビリティーゾーンのパブリックサブネットおよびプライベートサブネットを指定します。
- 各プライベートサブネットアベイラビリティーゾーンのパブリックサブネットを指定します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。VPC から OpenShift Container Platform クラスターを削除する場合、 **kubernetes.io/cluster/.*: shared** タグは、それが使用したサブネットから削除されます。

2.8.3.3. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する AWS の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ELB、セキュリティーグループ、S3 バケットおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

2.8.3.4. クラスター間の分離

OpenShift Container Platform を既存のネットワークにデプロイする場合、クラスターサービスの分離の規模は以下の方法で縮小されます。

- 複数の OpenShift Container Platform クラスターを同じ VPC にインストールできます。
- ICMP Ingress はネットワーク全体から許可されます。
- TCP 22 Ingress (SSH) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 6443 Ingress (Kubernetes API) はネットワーク全体に対して許可されます。
- コントロールプレーンの TCP 22623 Ingress (MCS) はネットワーク全体に対して許可されます。

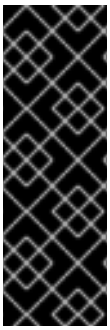
2.8.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

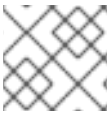


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

2.8.5. SSH プライベートキーの生成およびエージェントへの追加

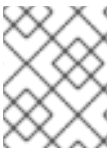
クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
-
```

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

2.8.6. インストールプログラムの取得

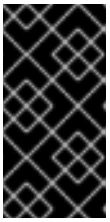
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

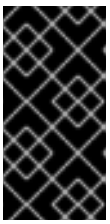
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシー

クレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

2.8.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

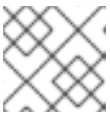
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

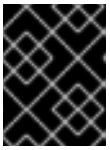
2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

2.8.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際

に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

2.8.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表2.18 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

2.8.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表2.19 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

2.8.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

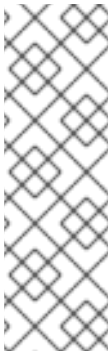
表2.20 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

2.8.7.1.4. オプションの AWS 設定パラメーター

オプションの AWS 設定パラメーターは、以下の表で説明されています。

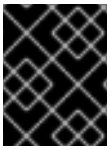
表2.21 オプションの AWS パラメーター

パラメーター	説明	値
compute.platform.aws.rootVolume.iops	ルートボリュームに予約される 1 秒あたりの入出力操作 (IOPS)。	整数 (例: **4000**)。
compute.platform.aws.rootVolume.size	ルートボリュームのサイズ (GiB)。	整数 (例: **500**)。
compute.platform.aws.rootVolume.type	ルートボリュームのインスタンスタイプ。	有効な AWS EBS インスタンスタイプ (例: **io1**)。
compute.platform.aws.type	コンピュートマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
compute.platform.aws.zones	インストールプログラムがコンピュートマシンプールのマシンを作成するアベイラビリティーゾーン。独自の VPC を指定する場合は、そのアベイラビリティーゾーンにサブネットを指定する必要があります。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。

パラメーター	説明	値
compute.aws.region	インストールプログラムがコンピュートリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
controlPlane.platform.aws.type	コントロールプレーンマシンの EC2 インスタンスタイプ。	有効な AWS インスタンスタイプ (例: **c5.9xlarge**)。
controlPlane.platform.aws.zones	インストールプログラムがコントロールプレーンマシンプールのマシンを作成するアベイラビリティーゾーン。	**YAML シーケンス** の us-east-1c などの有効な AWS アベイラビリティーゾーンの一覧。
controlPlane.aws.region	インストールプログラムがコントロールプレーンのリソースを作成する AWS リージョン。	有効な AWS リージョン (例: **us-east-1**)。
platform.aws.userTags	インストールプログラムが、作成するすべてのリソースに対するタグとして追加するキーと値のマップ。	**<key>: <value>** 形式のキー値ペアなどの有効な YAML マップ。AWS タグについての詳細は、AWS ドキュメントの Tagging Your Amazon EC2 Resources を参照してください。
platform.aws.subnets	インストールプログラムによる VPC の作成を許可する代わりに VPC を指定する場合は、使用するクラスターのサブネットを指定します。サブネットは、指定する同じ **machineNetwork[].cidr** 範囲の一部である必要があります。標準クラスターの場合は、各アベイラビリティーゾーンのパブリックおよびプライベートサブネットを指定します。プライベートクラスターについては、各アベイラビリティーゾーンのプライベートサブネットを指定します。	有効なサブネット ID。

2.8.7.2. AWS のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    aws:
      zones:
      - us-west-2a
      - us-west-2b
      rootVolume:
        iops: 4000
        size: 500
        type: io1
      type: m5.xlarge (5)
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    aws:
      rootVolume:
        iops: 2000
        size: 500
        type: io1 (8)
      type: c5.4xlarge
      zones:
      - us-west-2c
  replicas: 3
metadata:
  name: test-cluster (9)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-2 (10)
    userTags:
      adminContact: jdoe
      costCenter: 7536
    subnets: (11)
    - subnet-1
    - subnet-2
    - subnet-3
fips: false (12)
sshKey: ssh-ed25519 AAAA... (13)
publish: Internal (14)
pullSecret: '{"auths": ...}' (15)
```

1 9 10 15 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 5 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

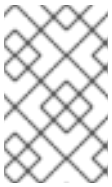
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **m4.2xlarge** または **m5.2xlarge** などの大規模なインスタンスタイプを使用します。

8 大規模なクラスターの場合などに etcd の高速のストレージを設定するには、ストレージタイプを **io1** として設定し、**iops** を **2000** に設定します。

11 独自の VPC を指定する場合は、クラスターが使用する各アベイラビリティーゾーンのサブネットを指定します。

12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

13 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

14 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

2.8.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

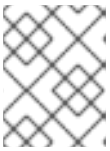
1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これら

のコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

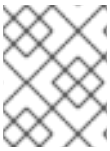


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

2.8.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

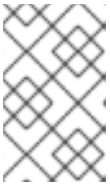
- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



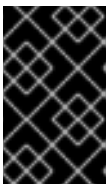
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

2.8.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.8.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.8.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.8.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。

PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.8.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.8.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

2.9. CLOUDFORMATION テンプレートの使用による、AWS でのユーザーによってプロビジョニングされたインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、独自に提供するインフラストラクチャーを使用するクラスターを Amazon Web Services (AWS) にインストールできます。

このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

2.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。

重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールします。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.9.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

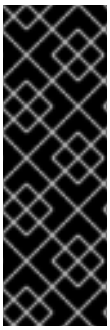
OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供す

るためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

2.9.3. 必要な AWS インフラストラクチャーコンポーネント

OpenShift Container Platform を Amazon Web Services (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、マシンとサポートするインフラストラクチャーの両方を手動で作成する必要があります。

各種プラットフォームの統合テストの詳細については、OpenShift Container Platform 4.x のテスト済みインテグレーション のページを参照してください。

提供される Cloud Formation テンプレートを使用してこのインフラストラクチャーを作成でき、コンポーネントを手動で作成するか、またはクラスターの要件を満たす既存のインフラストラクチャーを再利用できます。コンポーネントの相互関係についての詳細は、Cloud Formation テンプレートを参照してください。

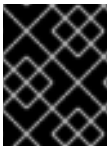
2.9.3.1. クラスターマシン

以下のマシンには **AWS::EC2::Instance** オブジェクトが必要になります。

- ブートストラップマシン。このマシンはインストール時に必要ですが、クラスターのデプロイ後に除去することができます。
- 3 つのコントロールプレーンマシンコントロールプレーンマシンはマシンセットによって制御されません。

- コンピュートマシン。インストール時に 2 つ以上のコンピュートマシン (ワーカーマシンとしても知られる) を作成する必要があります。これらのマシンはマシンセットによって制御されません。

提供される Cloud Formation テンプレートを使用して、クラスターマシンの以下のインスタンスタイプを使用できます。



重要

m4 インスタンスが **eu-west-3** などのリージョンで利用可能ではない場合、**m5** タイプを代わりに使用します。

表2.22 マシンのインスタンスタイプ

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large または **m5.large**			x
m4.xlarge または **m5.xlarge**		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.8xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
c4.large			x
c4.xlarge			x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x

これらのインスタンスタイプの仕様に対応する他のインスタンスタイプを使用できる場合もあります。

2.9.3.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

2.9.3.3. 他のインフラストラクチャーコンポーネント

- 1 つの VPC
- DNS エントリー
- ロードバランサー (classic または network) およびリスナー
- パブリックおよびプライベート Route 53 ゾーン
- セキュリティーグループ
- IAM ロール
- S3 バケット

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>VPC</td><td>AWS::EC2::VPC
AWS::EC2::VPCEndpoint</td><td>使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。</td></tr>
<tr><td>パブリックサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::SubnetNetworkAclAssociation</td><td>VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。</td></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
</table>

コンポーネント	AWS タイプ	説明	
		0 - 65535	アウトバウンド一時 (ephemeral) トラフィック
プライベートサブネット	• **AWS::EC2::Subnet** • **AWS::EC2::RouteTable** • **AWS::EC2::SubnetRouteTableAssociation**	VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。	

必要な DNS および負荷分散コンポーネント

DNS およびロードバランサー設定では、パブリックホストゾーンを使用する必要があり、クラスターのインフラストラクチャーをプロビジョニングする場合にインストールプログラムが使用するものと同様のプライベートホストゾーンを使用できます。ロードバランサーに解決する DNS エントリーを作成する必要があります。**api.<cluster_name>.<domain>** のエントリーは外部ロードバランサーを参照し、**api-int.<cluster_name>.<domain>** のエントリーは内部ロードバランサーを参照する必要があります。

またクラスターには、Kubernetes API とその拡張に必要なポート 6443、および新規マシンの Ignition 設定ファイルに必要なポート 22623 のロードバランサーおよびリスナーが必要です。ターゲットはマスターノードになります。ポート 6443 はクラスター外のクライアントとクラスター内のノードからもアクセスできる必要があります。ポート 22623 はクラスター内のノードからアクセスできる必要があります。

コンポーネント	AWS タイプ	説明
DNS	**AWS::Route53::HostedZone**	内部 DNS のホストゾーン。
etcd レコードセット	**AWS::Route53::RecordSet**	コントロールプレーンマシンの etcd の登録レコード。
パブリックロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	パブリックサブネットのロードバランサー。
外部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	外部 API サーバーのエイリアスレコード。

コンポーネント	AWS タイプ	説明
外部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	外部ロードバランサー用のポート 6443 のリスナー。
外部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	外部ロードバランサーのターゲットグループ。
プライベートロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	プライベートサブネットのロードバランサー。
内部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	内部 API サーバーのエイリアスレコード。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサー用のポート 22623 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサーのポート 6443 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。

セキュリティーグループ

コントロールプレーンおよびワーカーマシンには、以下のポートへのアクセスが必要です。

グループ	タイプ	IP プロトコル	ポート範囲
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**

グループ	タイプ	IP プロトコル	ポート範囲
		tcp	22
		tcp	6443
		tcp	22623
WorkerSecurityGroup	AWS::EC2::SecurityGroup	icmp	0
		tcp	22
BootstrapSecurityGroup	AWS::EC2::SecurityGroup	tcp	22
		tcp	19531

コントロールプレーンの Ingress

コントロールプレーンマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressEtcd	etcd	tcp	2379- 2380
MasterIngressVxlan	Vxlan パケット	udp	4789
MasterIngressWorkerVxlan	Vxlan パケット	udp	4789
MasterIngressInternal	内部クラスター通信および Kubernetes プロキシーメトリクス	tcp	9000 - 9999
MasterIngressWorkerInternal	内部クラスター通信	tcp	9000 - 9999
MasterIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	tcp	10250 - 10259
MasterIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	tcp	10250 - 10259
MasterIngressIngressServices	Kubernetes Ingress サービス	tcp	30000 - 32767

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngress WorkerIngress Services	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

ワーカーの Ingress

ワーカーマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngress Vxlan	Vxlan パケット	**udp**	**4789**
WorkerIngress WorkerVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngress Internal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngress WorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngress Kube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngress WorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngress IngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngress WorkerIngress Services	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

ロールおよびインスタンスプロファイル

マシンには、AWS でのパーミッションを付与する必要があります。提供される CloudFormation テンプレートはマシンに対し、以下の **AWS::IAM::Role** オブジェクトについてのパーミッションを付与し、それぞれのロールセットに **AWS::IAM::InstanceProfile** を指定します。テンプレートを使用しない場合、マシンには以下の広範囲のパーミッションまたは個別のパーミッションを付与することができます。

ロール	結果	アクション	リソース
マスター	**Allow**	**ec2:***	*
	Allow	**elasticloadbalancing:***	*
	Allow	**iam:PassRole**	*
	Allow	**s3:GetObject**	*
ワーカー	**Allow**	**ec2:Describe***	*
ブートストラップ	**Allow**	**ec2:Describe***	*
	Allow	**ec2:AttachVolume**	*
	Allow	**ec2:DetachVolume**	*

2.9.3.4. 必要な AWS パーミッション

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例2.11 インストールに必要な EC2 パーミッション

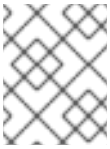
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**

- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**
- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**

- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例2.12 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**
- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例2.13 インストールに必要な Elastic Load Balancing のパーミッション

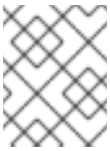
- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**

- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**
- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

例2.14 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**

- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**



注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例2.15 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例2.16 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**
- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例2.17 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

例2.18 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**
- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例2.19 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**

注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。

例2.20 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

2.9.4. インストールプログラムの取得

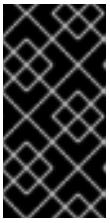
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

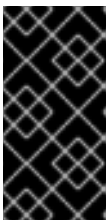
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

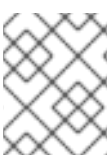
2.9.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

2.9.6. AWS のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

2.9.6.1. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

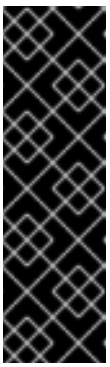
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを取得します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

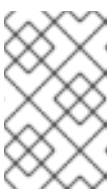
   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **aws** を選択します。

iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。

iv. クラスターのデプロイ先とする AWS リージョンを選択します。

v. クラスターに設定した Route 53 サービスのベースドメインを選択します。

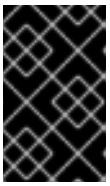
vi. クラスターの記述名を入力します。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを編集し、以下の **compute** スタンザに示されるようにコンピュートレプリカ (ワーカーレプリカとしても知られる) の数を **0** に設定します。

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
```

3. オプション: **install-config.yaml** ファイルをバックアップします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.9.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

2.9.6.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

2.9.7. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

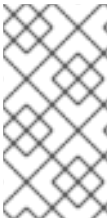
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

2.9.8. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。VPC を作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

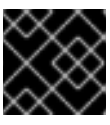
手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1. VPC の CIDR ブロック。
2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
3. テンプレートを起動します。



重要

単一行にコマンドを入力してください。

```
-
```

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1. **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

2.9.8.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例2.21 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
```

```
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 =
/19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]

Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
```

```
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
```

```
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
NAT:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Properties:
    AllocationId:
      "Fn::GetAtt":
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
EIP:
  Type: "AWS::EC2::EIP"
  Properties:
    Domain: vpc
Route:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId:
      Ref: NAT
PrivateSubnet2:
  Type: "AWS::EC2::Subnet"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
    AvailabilityZone: !Select
    - 1
    - Fn::GetAZs: !Ref "AWS::Region"
PrivateRouteTable2:
  Type: "AWS::EC2::RouteTable"
  Condition: DoAz2
  Properties:
    VpcId: !Ref VPC
PrivateSubnetRouteTableAssociation2:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Condition: DoAz2
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable2
NAT2:
  DependsOn:
  - GatewayToInternet
  Type: "AWS::EC2::NatGateway"
  Condition: DoAz2
  Properties:
```

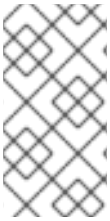
```
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT2
  PrivateSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable3:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation3:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz3
    Properties:
      SubnetId: !Ref PrivateSubnet3
      RouteTableId: !Ref PrivateRouteTable3
  NAT3:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
```

```
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
```

2.9.9. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。これらのコンポーネントを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。 これにより、ホストゾーンおよびサブネットのタグも作成されます。

単一 VPC 内でテンプレートを複数回実行することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ゾーンのホストゾーン ID を取得します。この ID は、AWS コンソールから、または以下のコマンドを実行して取得できます。

重要

単一行にコマンドを入力してください。

```
$ aws route53 list-hosted-zones-by-name |
    jq --arg name "<route53_domain>." \ 1
    -r '.HostedZones | .[] | select(.Name=="\($name)") | .Id'
```

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "ClusterName", 1
    "ParameterValue": "mycluster" 2
  },
  {
    "ParameterKey": "InfrastructureName", 3
    "ParameterValue": "mycluster-<random_string>" 4
  },
  {
    "ParameterKey": "HostedZoneId", 5
```

```
    "ParameterValue": "<random_string>" 6
  },
  {
    "ParameterKey": "HostedZoneName", 7
    "ParameterValue": "example.com" 8
  },
  {
    "ParameterKey": "PublicSubnets", 9
    "ParameterValue": "subnet-<random_string>" 10
  },
  {
    "ParameterKey": "PrivateSubnets", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "VpcId", 13
    "ParameterValue": "vpc-<random_string>" 14
  }
]
```

1. ホスト名などに使用するクラスターを表す短いクラスターの名前。
2. クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。
3. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
4. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
5. ターゲットの登録に使用する Route 53 パブリックトゾーン ID。
6. **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。
7. ターゲットの登録に使用する Route 53 ゾーン。
8. クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
9. VPC 用に作成したパブリックサブネット。
10. VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。
11. VPC 用に作成したプライベートサブネット。
12. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
13. クラスター用に作成した VPC。
14. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セク

ションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM
```

1. **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。
ApiServerDnsName	API サーバーの完全ホスト名。

RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。
InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。
InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

2.9.9.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例2.22 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
```

```
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
        default: "Public Subnets"
      PrivateSubnets:
        default: "Private Subnets"
      HostedZoneName:
        default: "Public Hosted Zone Name"
      HostedZoneId:
        default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
```

```
    Subnets: !Ref PublicSubnets
    Type: network

IntApiElb:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: !Join ["-", [!Ref InfrastructureName, "int"]]
    Scheme: internal
    IpAddressType: ipv4
    Subnets: !Ref PrivateSubnets
    Type: network

IntDns:
  Type: "AWS::Route53::HostedZone"
  Properties:
    HostedZoneConfig:
      Comment: "Managed by CloudFormation"
    Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
    HostedZoneTags:
    - Key: Name
      Value: !Join ["-", [!Ref InfrastructureName, "int"]]
    - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
      Value: "owned"
    VPCs:
    - VPCId: !Ref VpcId
      VPCRegion: !Ref "AWS::Region"

ExternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref HostedZoneId
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt ExtApiElb.DNSName

InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
```

```
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
```

```
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalServiceInternalListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalServiceTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 22623
    Protocol: TCP

InternalServiceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/healthz"
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

RegisterTargetLambdaIamRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
```

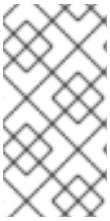
```
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
```

```
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
```

```
            for subnet_id in event['ResourceProperties']['Subnets']:
              ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
    Value: !Ref ExternalApiTargetGroup
  InternalApiTargetGroupArn:
    Description: ARN of the internal API target group.
    Value: !Ref InternalApiTargetGroup
  InternalServiceTargetGroupArn:
    Description: ARN of the internal service target group.
    Value: !Ref InternalServiceTargetGroup
```

2.9.10. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。これらのコンポーネントを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. VPC の CIDR ブロック。
4. **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。
5. VPC 用に作成したプライベートサブネット。

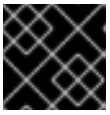
6 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

7 クラスター用に作成した VPC。

8 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。

3. テンプレートを起動します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM
```

1 **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル

WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

2.9.10.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例2.23 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
```

```
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  MasterIngressEtcd:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: etcd
      FromPort: 2379
      ToPort: 2380
      IpProtocol: tcp
```

```
MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressWorkerInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
```

```
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  MasterIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressWorkerInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  MasterIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressWorkerKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Kubernetes kubelet, scheduler and controller manager
      FromPort: 10250
      ToPort: 10259
      IpProtocol: tcp

  MasterIngressIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt MasterSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: tcp

  MasterIngressWorkerIngressServices:
    Type: AWS::EC2::SecurityGroupIngress
```

```
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressMasterVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
```

```
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressMasterInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp

  WorkerIngressKube:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes secure kubelet port
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal Kubernetes communication
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressMasterIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressMasterIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp
```

```
MasterIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action:
          - "ec2:AttachVolume"
          - "ec2:AuthorizeSecurityGroupIngress"
          - "ec2:CreateSecurityGroup"
          - "ec2:CreateTags"
          - "ec2:CreateVolume"
          - "ec2:DeleteSecurityGroup"
          - "ec2:DeleteVolume"
          - "ec2:Describe*"
          - "ec2:DetachVolume"
          - "ec2:ModifyInstanceAttribute"
          - "ec2:ModifyVolume"
          - "ec2:RevokeSecurityGroupIngress"
          - "elasticloadbalancing:AddTags"
          - "elasticloadbalancing:AttachLoadBalancerToSubnets"
          - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
          - "elasticloadbalancing:CreateListener"
          - "elasticloadbalancing:CreateLoadBalancer"
          - "elasticloadbalancing:CreateLoadBalancerPolicy"
          - "elasticloadbalancing:CreateLoadBalancerListeners"
          - "elasticloadbalancing:CreateTargetGroup"
          - "elasticloadbalancing:ConfigureHealthCheck"
          - "elasticloadbalancing:DeleteListener"
          - "elasticloadbalancing:DeleteLoadBalancer"
          - "elasticloadbalancing:DeleteLoadBalancerListeners"
          - "elasticloadbalancing:DeleteTargetGroup"
          - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
          - "elasticloadbalancing:DeregisterTargets"
          - "elasticloadbalancing:Describe*"
          - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
          - "elasticloadbalancing:ModifyListener"
          - "elasticloadbalancing:ModifyLoadBalancerAttributes"
          - "elasticloadbalancing:ModifyTargetGroup"
          - "elasticloadbalancing:ModifyTargetGroupAttributes"
          - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
          - "elasticloadbalancing:RegisterTargets"
          - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
          - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
```

```
            - "kms:DescribeKey"
            Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId

  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

2.9.11. AWS インフラストラクチャーの RHCOS AMI

OpenShift Container Platform ノードについて、Amazon Web Services (AWS) ゾーンの有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を使用する必要があります。

表2.23 RHCOS AMI

AWS ゾーン	AWS AMI
ap-northeast-1	**ami-0530d04240177f118**
ap-northeast-2	**ami-09e4cd700276785d2**
ap-south-1	**ami-0754b15d212830477**
ap-southeast-1	**ami-03b46cc4b1518c5a8**
ap-southeast-2	**ami-0a5b99ab2234a4e6a**
ca-central-1	**ami-012bc4ee3b6c673bc**
eu-central-1	**ami-02e08df1201f1c2f8**
eu-north-1	**ami-0309c9d2fadcb2d5a**
eu-west-1	**ami-0bdd69d8e7cd18188**
eu-west-2	**ami-0e610e967a62dbdfa**
eu-west-3	**ami-0e817e26f638a71ac**
me-south-1	**ami-024117d7c87b7ff08**
sa-east-1	**ami-08e62f746b94950c1**
us-east-1	**ami-077ede5bed2e431ea**
us-east-2	**ami-0f4ecf819275850dd**
us-west-1	**ami-0c4990e435bc6c5fe**
us-west-2	**ami-000d6e92357ac605c**

2.9.12. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。このノードを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。

注記

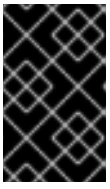
提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

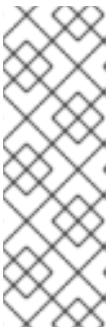
手順

1. **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定します。このファイルはインストールディレクトリーに置かれます。これを実行するための 1 つの方法として、クラスターのリージョンに S3 バケットを作成し、Ignition 設定ファイルをこれにアップロードします。



重要

提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。



注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

   a. バケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

   1 **<cluster-name>-infra** はバケット名です。

   b. **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

   c. ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
  {
    "ParameterKey": "VpcId", 11
    "ParameterValue": "vpc-<random_string>" 12
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", 13
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" 14
  },
  {
    "ParameterKey": "AutoRegisterELB", 15
    "ParameterValue": "yes" 16
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 17
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 18
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 19
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 20
  },
  {
```

```
    "ParameterKey": "InternalApiTargetGroupArn", 21
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 22
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 23
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 24
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. ブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. 有効な **AWS::EC2::Image::Id** 値を指定します。
5. ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。
6. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
7. ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。
8. VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。
9. マスターセキュリティーグループ ID (一時ルールの登録用)。
10. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
11. 作成されたリソースが属する VPC。
12. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。
13. ブートストラップの Ignition 設定ファイルをフェッチする場所。
14. **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。
15. ネットワークロードバランサー (NLB) を登録するかどうか。
16. **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。
17. NLB IP ターゲット登録 lambda グループの ARN。
18. DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。
19. 外部 API ロードバランサーのターゲットグループの ARN。
20. DNS および負荷分散の CloudFormation テンプレートの出力から

21 内部 API ロードバランサーのターゲットグループの ARN。

22 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。

23 内部サービスバランサーのターゲットグループの ARN。

24 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。

3. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM
```

1 **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

BootstrapInstanceId	ブートストラップインスタンス ID。
BootstrapPublicIp	ブートストラップノードのパブリック IP アドレス。
BootstrapPrivateIp	ブートストラップノードのプライベート IP アドレス。

2.9.12.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例2.24 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
```

```
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
```

```
BootstrapIamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
      - Effect: "Allow"
        Principal:
          Service:
          - "ec2.amazonaws.com"
        Action:
        - "sts:AssumeRole"
    Path: "/"
    Policies:
    - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Action: "ec2:Describe*"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:AttachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "ec2:DetachVolume"
          Resource: "*"
        - Effect: "Allow"
          Action: "s3:GetObject"
          Resource: "*"

BootstrapInstanceProfile:
  Type: "AWS::IAM::InstanceProfile"
  Properties:
    Path: "/"
    Roles:
    - Ref: "BootstrapIamRole"

BootstrapSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: Cluster Bootstrap Security Group
    SecurityGroupIngress:
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: !Ref AllowedBootstrapSshCidr
    - IpProtocol: tcp
      ToPort: 19531
      FromPort: 19531
      CidrIp: 0.0.0.0/0
    VpcId: !Ref VpcId

BootstrapInstance:
  Type: AWS::EC2::Instance
  Properties:
```

```
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: "i3.large"
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

2.9.13. AWS でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。これらのノードを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。

注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
```

```
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m4.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
```

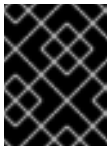
```
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. コントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。
11. 13. 15. コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。
12. 14. 16. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
17. マスターノードに関連付けるマスターセキュリティーグループ ID。
18. セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。
19. コントロールプレーンの Ignition 設定ファイルをフェッチする場所。
20. 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。
21. 使用する base64 でエンコードされた認証局の文字列。
22. インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。
23. マスターロールに関連付ける IAM プロファイル。
24. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。

25 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

26 許可される値:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**



重要

m4 インスタンスタイプが **eu-west-3** などのリージョンで利用可能ではない場合、**m5.xlarge** などのように **m5** タイプを代わりに使用します。

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。

35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2.9.13.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例2.25 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
```

```
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone
information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m4.xlarge
    Type: String
    AllowedValues:
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.8xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
```

```
      - "c4.2xlarge"
      - "c4.4xlarge"
      - "c4.8xlarge"
      - "r4.xlarge"
      - "r4.2xlarge"
      - "r4.4xlarge"
      - "r4.8xlarge"
      - "r4.16xlarge"
    AutoRegisterELB:
      Default: "yes"
      AllowedValues:
      - "yes"
      - "no"
      Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
      Type: String
    RegisterNlbIpTargetsLambdaArn:
      Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
      Type: String
    ExternalApiTargetGroupArn:
      Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
      Type: String
    InternalApiTargetGroupArn:
      Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
      Type: String
    InternalServiceTargetGroupArn:
      Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
      Type: String

  Metadata:
    AWS::CloudFormation::Interface:
      ParameterGroups:
      - Label:
          default: "Cluster Information"
        Parameters:
        - InfrastructureName
      - Label:
          default: "Host Information"
        Parameters:
        - MasterInstanceType
        - RhcosAmi
        - IgnitionLocation
        - CertificateAuthorities
        - MasterSecurityGroupId
        - MasterInstanceProfileName
      - Label:
          default: "Network Configuration"
        Parameters:
        - VpcId
        - AllowedBootstrapSshCidr
        - Master0Subnet
        - Master1Subnet
        - Master2Subnet
```

```
    - Label:
        default: "DNS"
      Parameters:
      - AutoRegisterDNS
      - PrivateHostedZoneName
      - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Master Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterDNS:
        default: "Use Provided DNS Automation"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"
      PrivateHostedZoneName:
        default: "Private Hosted Zone Name"
      PrivateHostedZoneId:
        default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
```

```
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
```

```
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
```

```
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  EtcdSrvRecords:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]],
      ]
```

```
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]],
      ]
      - !Join [
        " ",
        ["0 10 2380", !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]],
      ]
      TTL: 60
      Type: SRV

  Etcd0Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-0", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master0.PrivateIp
      TTL: 60
      Type: A

  Etcd1Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-1", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master1.PrivateIp
      TTL: 60
      Type: A

  Etcd2Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

2.9.14. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS でのブートストラップノードの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、クラスターをインストールできます。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを手動で管理する予定の場合には、ワーカーマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

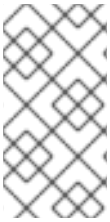
コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2.9.14.1. AWS でのワーカーノードの作成

クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。これらのノードを手動で作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。

重要

CloudFormation テンプレートは、1 つのワーカーマシンを表すスタックを作成します。それぞれのワーカーマシンにスタックを作成する必要があります。



注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "Subnet", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "WorkerSecurityGroupId", 7
    "ParameterValue": "sg-<random_string>" 8
  },
  {
    "ParameterKey": "IgnitionLocation", 9
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
  },
  {
    "ParameterKey": "CertificateAuthorities", 11
    "ParameterValue": "" 12
  },
```

```
  {
    "ParameterKey": "WorkerInstanceProfileName", 13
    "ParameterValue": "" 14
  },
  {
    "ParameterKey": "WorkerInstanceType", 15
    "ParameterValue": "m4.large" 16
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. ワーカーノードに使用する最新の Red Hat Enterprise Linux CoreOS(RHCOS)AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. ワーカーノードを起動するサブネット (プライベートが望ましい)。
6. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
7. ワーカーノードに関連付けるワーカーセキュリティーグループ ID。
8. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。
9. ブートストラップの Ignition 設定ファイルをフェッチする場所。
10. 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**
11. 使用する base64 でエンコードされた認証局の文字列。
12. インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。
13. ワーカーロールに関連付ける IAM プロファイル。
14. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。
15. コントロールプレーンマシンに使用する AWS インスタンスのタイプ。
16. 許可される値:
    - **m4.large**
    - **m4.xlarge**
    - **m4.2xlarge**
    - **m4.4xlarge**

- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.large**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

重要

m4 インスタンスが **eu-west-3** などのリージョンで利用可能ではない場合、**m5** タイプを代わりに使用します。

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. **m5** インスタンスタイプを **WorkerInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. ワーカースタックを作成します。

   a. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

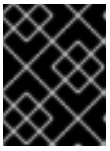
```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml \ (2)
    --parameters file://<parameters>.json (3)
```

1. **<name>** は **cluster-workers** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

b. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。



重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

2.9.14.1.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例2.26 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
```

```
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m4.large
    Type: String
    AllowedValues:
    - "m4.large"
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.8xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "c4.large"
    - "c4.xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "r4.large"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
```

```
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

2.9.15. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

2.9.15.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.9.15.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

2.9.15.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

2.9.16. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.9.17. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

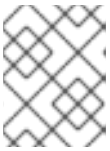
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

2.9.18. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                       4.5.4     True        False         False      69s
cloud-credential                     4.5.4     True        False         False      12m
cluster-autoscaler                   4.5.4     True        False         False      11m
console                              4.5.4     True        False         False      46s
dns                                  4.5.4     True        False         False      11m
image-registry                       4.5.4     True        False         False      5m26s
ingress                              4.5.4     True        False         False      5m36s
```

```
kube-apiserver                 4.5.4     True        False         False      8m53s
kube-controller-manager        4.5.4     True        False         False      7m24s
kube-scheduler                 4.5.4     True        False         False      12m
machine-api                    4.5.4     True        False         False      12m
machine-config                 4.5.4     True        False         False      7m36s
marketplace                    4.5.4     True        False         False      7m54m
monitoring                     4.5.4     True        False         False      7h54s
network                        4.5.4     True        False         False      5m9s
node-tuning                    4.5.4     True        False         False      11m
openshift-apiserver            4.5.4     True        False         False      11m
openshift-controller-manager   4.5.4     True        False         False      5m943s
openshift-samples              4.5.4     True        False         False      3m55s
operator-lifecycle-manager     4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog   4.5.4     True        False         False      11m
service-ca                     4.5.4     True        False         False      11m
service-catalog-apiserver      4.5.4     True        False         False      5m26s
service-catalog-controller-manager   4.5.4     True        False         False      5m25s
storage                        4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

2.9.18.1. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

AWS のユーザーによってプロビジョニングされるインフラストラクチャーのレジストリーストレージを設定し、OpenShift Container Platform を非表示のリージョンにデプロイできます。詳細は、Configuring the registry for AWS user-provisioned infrastructure を参照してください。

2.9.18.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS 上のクラスター。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**

- **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

2.9.18.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

   - 以下を実行します。

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     次に、行を変更します。

     ```
     managementState: Removed
     ```

     上記を以下のように変更します。

     ```
     managementState: Managed
     ```

2.9.19. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。

   ```
   $ aws cloudformation delete-stack --stack-name <name> 1
   ```

   1 **<name>** は、ブートストラップスタックの名前です。

2.9.20. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) をインストールします。

- **jq** パッケージをインストールします。
- AWS CLI をダウンロードし、これをコンピューターにインストールします。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

     ```
     $ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}{end}' routes
     ```

     出力例

     ```
     oauth-openshift.apps.<cluster_name>.<domain_name>
     console-openshift-console.apps.<cluster_name>.<domain_name>
     downloads-openshift-console.apps.<cluster_name>.<domain_name>
     alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
     grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
     prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
     ```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

   ```
   $ oc -n openshift-ingress get service router-default
   ```

   出力例

   ```
   NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)                      AGE
   router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com   80:31499/TCP,443:30693/TCP   5m
   ```

3. ロードバランサーのホストゾーン ID を見つけます。

   ```
   $ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName == "<external_ip>").CanonicalHostedZoneNameID' 1
   ```

   1 **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

   出力例

   ```
   Z3AADJGX6KTTL2
   ```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ 1
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", 2
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", 3
>           "DNSName": "<external_ip>.", 4
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

1 **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。

2 **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

3 **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

4 **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。

(2) **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

(3) **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

(4) **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

2.9.21. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除しています。
- **oc** CLI をインストールし、ログインします。

手順

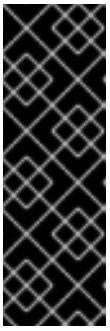
- クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete (1)
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書からの回復** についてのドキュメントを参照してください。

2.9.22. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

2.10. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での AWS へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、各自でプロビジョニングするインフラストラクチャーおよびインストールリリースコンテンツの内部ミラーを使用して、クラスターを Amazon Web Services (AWS) にインストールできます。

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが AWS API を使用するにはインターネットへのアクセスが必要になります。

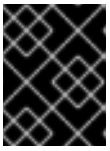
このインフラストラクチャーを作成する 1 つの方法として、提供される CloudFormation テンプレートを使用できます。テンプレートを変更してインフラストラクチャーをカスタマイズしたり、それらに含まれる情報を使用し、所属する会社のポリシーに基づいて AWS オブジェクトを作成したりできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の CloudFormation テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

2.10.1. 前提条件

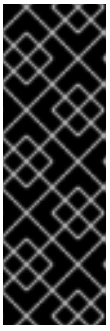
- ミラーホストでミラーレジストリーを作成 しており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- AWS アカウントを設定 してクラスターをホストします。



重要

AWS プロファイルがご使用のコンピューターに保存されている場合、マルチファクター認証デバイスを使用中に生成した一時的なセッショントークンを使用することはできません。クラスターは継続的に現行の AWS 認証情報を使用して、クラスターの有効期間全体にわたって AWS リソースを作成するため、キーをベースとした有効期間の長い認証情報を使用する必要があります。適切なキーを生成するには、AWS ドキュメントの Managing Access Keys for IAM Users を参照してください。キーは、インストールプログラムの実行時に指定できます。

- AWS CLI をダウンロードし、これをコンピューターにインストールします。AWS ドキュメントの Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

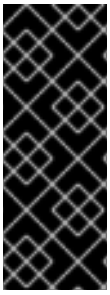
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

2.10.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

2.10.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

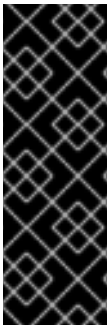
2.10.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

2.10.4. 必要な AWS インフラストラクチャーコンポーネント

OpenShift Container Platform を Amazon Web Services (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーにインストールするには、マシンとサポートするインフラストラクチャーの両方を手動で作成する必要があります。

各種プラットフォームの統合テストの詳細については、OpenShift Container Platform 4.x のテスト済みインテグレーション のページを参照してください。

提供される Cloud Formation テンプレートを使用してこのインフラストラクチャーを作成でき、コンポーネントを手動で作成するか、またはクラスターの要件を満たす既存のインフラストラクチャーを再利用できます。コンポーネントの相互関係についての詳細は、Cloud Formation テンプレートを参照してください。

2.10.4.1. クラスターマシン

以下のマシンには **AWS::EC2::Instance** オブジェクトが必要になります。

- ブートストラップマシン。このマシンはインストール時に必要ですが、クラスターのデプロイ後に除去することができます。
- 3 つのコントロールプレーンマシンコントロールプレーンマシンはマシンセットによって制御されません。
- コンピュートマシン。インストール時に 2 つ以上のコンピュートマシン (ワーカーマシンとしても知られる) を作成する必要があります。これらのマシンはマシンセットによって制御されません。

提供される Cloud Formation テンプレートを使用して、クラスターマシンの以下のインスタンスタイプを使用できます。

重要

m4 インスタンスが **eu-west-3** などのリージョンで利用可能ではない場合、**m5** タイプを代わりに使用します。

表2.24 マシンのインスタンスタイプ

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
i3.large	x		
m4.large または **m5.large**			x

インスタンスタイプ	ブートストラップ	コントロールプレーン	コンピュート
m4.xlarge または **m5.xlarge**		x	x
m4.2xlarge		x	x
m4.4xlarge		x	x
m4.8xlarge		x	x
m4.10xlarge		x	x
m4.16xlarge		x	x
c4.large			x
c4.xlarge			x
c4.2xlarge		x	x
c4.4xlarge		x	x
c4.8xlarge		x	x
r4.large			x
r4.xlarge		x	x
r4.2xlarge		x	x
r4.4xlarge		x	x
r4.8xlarge		x	x
r4.16xlarge		x	x

これらのインスタンスタイプの仕様に対応する他のインスタンスタイプを使用できる場合もあります。

2.10.4.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

2.10.4.3. 他のインフラストラクチャーコンポーネント

- 1 つの VPC
- DNS エントリー
- ロードバランサー (classic または network) およびリスナー
- パブリックおよびプライベート Route 53 ゾーン
- セキュリティーグループ
- IAM ロール
- S3 バケット

非接続環境で作業している場合、EC2 および ELB エンドポイントのパブリック IP アドレスに到達することはできません。これを解決するには、VPC エンドポイントを作成し、これをクラスターが使用するサブネットに割り当てる必要があります。エンドポイントの名前は以下のように指定する必要があります。

- **ec2.<region>.amazonaws.com**
- **elasticloadbalancing.<region>.amazonaws.com**
- **s3.<region>.amazonaws.com**

必要な VPC コンポーネント

お使いのマシンとの通信を可能にする適切な VPC およびサブネットを指定する必要があります。

コンポーネント	AWS タイプ	説明
VPC	- **AWS::EC2::VPC** - **AWS::EC2::VPCEndpoint**	使用するクラスターのパブリック VPC を指定する必要があります。VPC は、各サブネットのルートテーブルを参照するエンドポイントを使用して、S3 でホストされているレジストリーとの通信を強化します。
パブリックサブネット	- **AWS::EC2::Subnet** - **AWS::EC2::SubnetNetworkAclAssociation**	VPC には 1 から 3 のアベイラビリティーゾーンのパブリックサブネットが必要であり、それらを適切な Ingress ルールに関連付ける必要があります。

<table>
<tr><th>コンポーネント</th><th>AWS タイプ</th><th>説明</th></tr>
<tr><td>インターネットゲートウェイ</td><td>AWS::EC2::InternetGateway
AWS::EC2::VPCGatewayAttachment
AWS::EC2::RouteTable
AWS::EC2::Route
AWS::EC2::SubnetRouteTableAssociation
AWS::EC2::NatGateway
AWS::EC2::EIP</td><td>VPC に割り当てられたパブリックルートを持つパブリックインターネットゲートウェイが必要です。提供されるテンプレートでは、各パブリックサブネットに EIP アドレスと NAT ゲートウェイがあります。これらの NAT ゲートウェイは、プライベートサブネットインスタンスなどのクラスターリソースがインターネットに到達できるようにするもので、一部のネットワークが制限された環境またはプロキシーのシナリオでは必要ありません。</td></tr>
<tr><td>ネットワークアクセス制御</td><td>AWS::EC2::NetworkAcl
AWS::EC2::NetworkAclEntry</td><td>VPC が以下のポートにアクセスできるようにする必要があります。<table><tr><th>ポート</th><th>理由</th></tr><tr><td>80</td><td>インバウンド HTTP トラフィック</td></tr><tr><td>443</td><td>インバウンド HTTPS トラフィック</td></tr><tr><td>22</td><td>インバウンド SSH トラフィック</td></tr><tr><td>1024 - 65535</td><td>インバウンド一時 (ephemeral) トラフィック</td></tr><tr><td>0 - 65535</td><td>アウトバウンド一時 (ephemeral) トラフィック</td></tr></table></td></tr>
<tr><td>プライベートサブネット</td><td>AWS::EC2::Subnet
AWS::EC2::RouteTable
AWS::EC2::SubnetRouteTableAssociation</td><td>VPC にはプライベートサブネットを使用できます。提供される CloudFormation テンプレートは 1 から 3 アベイラビリティーゾーンのプライベートサブネットを作成できます。プライベートサブネットを使用できる場合は、それらの適切なルートおよびテーブルを指定する必要があります。</td></tr>
</table>

必要な DNS および負荷分散コンポーネント

DNS およびロードバランサー設定では、パブリックホストゾーンを使用する必要があり、クラスターのインフラストラクチャーをプロビジョニングする場合にインストールプログラムが使用するものと同様のプライベートホストゾーンを使用できます。ロードバランサーに解決する DNS エントリーを作成する必要があります。**api.<cluster_name>.<domain>** のエントリーは外部ロードバランサーを参照し、**api-int.<cluster_name>.<domain>** のエントリーは内部ロードバランサーを参照する必要があります。

またクラスターには、Kubernetes API とその拡張に必要なポート 6443、および新規マシンの Ignition 設定ファイルに必要なポート 22623 のロードバランサーおよびリスナーが必要です。ターゲットはマスターノードになります。ポート 6443 はクラスター外のクライアントとクラスター内のノードからもアクセスできる必要があります。ポート 22623 はクラスター内のノードからアクセスできる必要があります。

コンポーネント	AWS タイプ	説明
DNS	**AWS::Route53::HostedZone**	内部 DNS のホストゾーン。
etcd レコードセット	**AWS::Route53::RecordSet**	コントロールプレーンマシンの etcd の登録レコード。
パブリックロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	パブリックサブネットのロードバランサー。
外部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	外部 API サーバーのエイリアスレコード。
外部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	外部ロードバランサー用のポート 6443 のリスナー。
外部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	外部ロードバランサーのターゲットグループ。
プライベートロードバランサー	**AWS::ElasticLoadBalancingV2::LoadBalancer**	プライベートサブネットのロードバランサー。

コンポーネント	AWS タイプ	説明
内部 API サーバーレコード	**AWS::Route53::RecordSetGroup**	内部 API サーバーのエイリアスレコード。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサー用のポート 22623 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。
内部リスナー	**AWS::ElasticLoadBalancingV2::Listener**	内部ロードバランサーのポート 6443 のリスナー。
内部ターゲットグループ	**AWS::ElasticLoadBalancingV2::TargetGroup**	内部ロードバランサーのターゲットグループ。

セキュリティーグループ

コントロールプレーンおよびワーカーマシンには、以下のポートへのアクセスが必要です。

グループ	タイプ	IP プロトコル	ポート範囲
MasterSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
		tcp	**6443**
		tcp	**22623**
WorkerSecurityGroup	**AWS::EC2::SecurityGroup**	**icmp**	**0**
		tcp	**22**
BootstrapSecurityGroup	**AWS::EC2::SecurityGroup**	**tcp**	**22**
		tcp	**19531**

コントロールプレーンの Ingress

コントロールプレーンマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
MasterIngressEtcd	etcd	**tcp**	**2379- 2380**
MasterIngressVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**
MasterIngressInternal	内部クラスター通信および Kubernetes プロキシーメトリクス	**tcp**	**9000 - 9999**
MasterIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
MasterIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250 - 10259**
MasterIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
MasterIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

ワーカーの Ingress

ワーカーマシンには、以下の Ingress グループが必要です。それぞれの Ingress グループは **AWS::EC2::SecurityGroupIngress** リソースになります。

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressVxlan	Vxlan パケット	**udp**	**4789**
WorkerIngressWorkerVxlan	Vxlan パケット	**udp**	**4789**

Ingress グループ	説明	IP プロトコル	ポート範囲
WorkerIngressInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressWorkerInternal	内部クラスター通信	**tcp**	**9000 - 9999**
WorkerIngressKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressWorkerKube	Kubernetes kubelet、スケジューラーおよびコントローラーマネージャー	**tcp**	**10250**
WorkerIngressIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**
WorkerIngressWorkerIngressServices	Kubernetes Ingress サービス	**tcp**	**30000 - 32767**

ロールおよびインスタンスプロファイル

マシンには、AWS でのパーミッションを付与する必要があります。提供される CloudFormation テンプレートはマシンに対し、以下の **AWS::IAM::Role** オブジェクトについてのパーミッションを付与し、それぞれのロールセットに **AWS::IAM::InstanceProfile** を指定します。テンプレートを使用しない場合、マシンには以下の広範囲のパーミッションまたは個別のパーミッションを付与することができます。

ロール	結果	アクション	リソース
マスター	**Allow**	**ec2:***	*
	Allow	**elasticloadbalancing:***	*
	Allow	**iam:PassRole**	*
	Allow	**s3:GetObject**	*
ワーカー	**Allow**	**ec2:Describe***	*
ブートストラップ	**Allow**	**ec2:Describe***	*
	Allow	**ec2:AttachVolume**	*
	Allow	**ec2:DetachVolume**	*

2.10.4.4. 必要な AWS パーミッション

AdministratorAccess ポリシーを、Amazon Web Services (AWS) で作成する IAM ユーザーに割り当てる場合、そのユーザーには必要なパーミッションすべてを付与します。OpenShift Container Platform クラスターのすべてのコンポーネントをデプロイするために、IAM ユーザーに以下のパーミッションが必要になります。

例2.27 インストールに必要な EC2 パーミッション

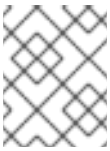
- **tag:TagResources**
- **tag:UntagResources**
- **ec2:AllocateAddress**
- **ec2:AssociateAddress**
- **ec2:AuthorizeSecurityGroupEgress**
- **ec2:AuthorizeSecurityGroupIngress**
- **ec2:CopyImage**
- **ec2:CreateNetworkInterface**
- **ec2:AttachNetworkInterface**
- **ec2:CreateSecurityGroup**
- **ec2:CreateTags**
- **ec2:CreateVolume**
- **ec2:DeleteSecurityGroup**
- **ec2:DeleteSnapshot**
- **ec2:DeleteTags**
- **ec2:DeregisterImage**
- **ec2:DescribeAccountAttributes**
- **ec2:DescribeAddresses**
- **ec2:DescribeAvailabilityZones**
- **ec2:DescribeDhcpOptions**
- **ec2:DescribeImages**
- **ec2:DescribeInstanceAttribute**
- **ec2:DescribeInstanceCreditSpecifications**
- **ec2:DescribeInstances**
- **ec2:DescribeInternetGateways**

- **ec2:DescribeKeyPairs**
- **ec2:DescribeNatGateways**
- **ec2:DescribeNetworkAcls**
- **ec2:DescribeNetworkInterfaces**
- **ec2:DescribePrefixLists**
- **ec2:DescribeRegions**
- **ec2:DescribeRouteTables**
- **ec2:DescribeSecurityGroups**
- **ec2:DescribeSubnets**
- **ec2:DescribeTags**
- **ec2:DescribeVolumes**
- **ec2:DescribeVpcAttribute**
- **ec2:DescribeVpcClassicLink**
- **ec2:DescribeVpcClassicLinkDnsSupport**
- **ec2:DescribeVpcEndpoints**
- **ec2:DescribeVpcs**
- **ec2:GetEbsDefaultKmsKeyId**
- **ec2:ModifyInstanceAttribute**
- **ec2:ModifyNetworkInterfaceAttribute**
- **ec2:ReleaseAddress**
- **ec2:RevokeSecurityGroupEgress**
- **ec2:RevokeSecurityGroupIngress**
- **ec2:RunInstances**
- **ec2:TerminateInstances**

例2.28 インストール時のネットワークリソースの作成に必要なパーミッション

- **ec2:AssociateDhcpOptions**
- **ec2:AssociateRouteTable**
- **ec2:AttachInternetGateway**
- **ec2:CreateDhcpOptions**

- **ec2:CreateInternetGateway**
- **ec2:CreateNatGateway**
- **ec2:CreateRoute**
- **ec2:CreateRouteTable**
- **ec2:CreateSubnet**
- **ec2:CreateVpc**
- **ec2:CreateVpcEndpoint**
- **ec2:ModifySubnetAttribute**
- **ec2:ModifyVpcAttribute**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの作成にこれらのパーミッションを必要としません。

例2.29 インストールに必要な Elastic Load Balancing のパーミッション

- **elasticloadbalancing:AddTags**
- **elasticloadbalancing:ApplySecurityGroupsToLoadBalancer**
- **elasticloadbalancing:AttachLoadBalancerToSubnets**
- **elasticloadbalancing:ConfigureHealthCheck**
- **elasticloadbalancing:CreateListener**
- **elasticloadbalancing:CreateLoadBalancer**
- **elasticloadbalancing:CreateLoadBalancerListeners**
- **elasticloadbalancing:CreateTargetGroup**
- **elasticloadbalancing:DeleteLoadBalancer**
- **elasticloadbalancing:DeregisterInstancesFromLoadBalancer**
- **elasticloadbalancing:DeregisterTargets**
- **elasticloadbalancing:DescribeInstanceHealth**
- **elasticloadbalancing:DescribeListeners**
- **elasticloadbalancing:DescribeLoadBalancerAttributes**
- **elasticloadbalancing:DescribeLoadBalancers**
- **elasticloadbalancing:DescribeTags**

- **elasticloadbalancing:DescribeTargetGroupAttributes**
- **elasticloadbalancing:DescribeTargetHealth**
- **elasticloadbalancing:ModifyLoadBalancerAttributes**
- **elasticloadbalancing:ModifyTargetGroup**
- **elasticloadbalancing:ModifyTargetGroupAttributes**
- **elasticloadbalancing:RegisterInstancesWithLoadBalancer**
- **elasticloadbalancing:RegisterTargets**
- **elasticloadbalancing:SetLoadBalancerPoliciesOfListener**

例2.30 インストールに必要な IAM パーミッション

- **iam:AddRoleToInstanceProfile**
- **iam:CreateInstanceProfile**
- **iam:CreateRole**
- **iam:DeleteInstanceProfile**
- **iam:DeleteRole**
- **iam:DeleteRolePolicy**
- **iam:GetInstanceProfile**
- **iam:GetRole**
- **iam:GetRolePolicy**
- **iam:GetUser**
- **iam:ListInstanceProfilesForRole**
- **iam:ListRoles**
- **iam:ListUsers**
- **iam:PassRole**
- **iam:PutRolePolicy**
- **iam:RemoveRoleFromInstanceProfile**
- **iam:SimulatePrincipalPolicy**
- **iam:TagRole**

注記

AWS アカウントに Elastic Load Balancer (ELB) を作成していない場合、IAM ユーザーには **iam:CreateServiceLinkedRole** パーミッションも必要です。

例2.31 インストールに必要な Route 53 パーミッション

- **route53:ChangeResourceRecordSets**
- **route53:ChangeTagsForResource**
- **route53:CreateHostedZone**
- **route53:DeleteHostedZone**
- **route53:GetChange**
- **route53:GetHostedZone**
- **route53:ListHostedZones**
- **route53:ListHostedZonesByName**
- **route53:ListResourceRecordSets**
- **route53:ListTagsForResource**
- **route53:UpdateHostedZoneComment**

例2.32 インストールに必要な S3 パーミッション

- **s3:CreateBucket**
- **s3:DeleteBucket**
- **s3:GetAccelerateConfiguration**
- **s3:GetBucketAcl**
- **s3:GetBucketCors**
- **s3:GetBucketLocation**
- **s3:GetBucketLogging**
- **s3:GetBucketObjectLockConfiguration**
- **s3:GetBucketReplication**
- **s3:GetBucketRequestPayment**
- **s3:GetBucketTagging**
- **s3:GetBucketVersioning**

- **s3:GetBucketWebsite**
- **s3:GetEncryptionConfiguration**
- **s3:GetLifecycleConfiguration**
- **s3:GetReplicationConfiguration**
- **s3:ListBucket**
- **s3:PutBucketAcl**
- **s3:PutBucketTagging**
- **s3:PutEncryptionConfiguration**

例2.33 クラスター Operator が必要とする S3 パーミッション

- **s3:DeleteObject**
- **s3:GetObject**
- **s3:GetObjectAcl**
- **s3:GetObjectTagging**
- **s3:GetObjectVersion**
- **s3:PutObject**
- **s3:PutObjectAcl**
- **s3:PutObjectTagging**

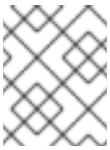
例2.34 ベースクラスターリソースの削除に必要なパーミッション

- **autoscaling:DescribeAutoScalingGroups**
- **ec2:DeleteNetworkInterface**
- **ec2:DeleteVolume**
- **elasticloadbalancing:DeleteTargetGroup**
- **elasticloadbalancing:DescribeTargetGroups**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:ListInstanceProfiles**
- **iam:ListRolePolicies**
- **iam:ListUserPolicies**

- **s3:DeleteObject**
- **s3:ListBucketVersions**
- **tag:GetResources**

例2.35 ネットワークリソースの削除に必要なパーミッション

- **ec2:DeleteDhcpOptions**
- **ec2:DeleteInternetGateway**
- **ec2:DeleteNatGateway**
- **ec2:DeleteRoute**
- **ec2:DeleteRouteTable**
- **ec2:DeleteSubnet**
- **ec2:DeleteVpc**
- **ec2:DeleteVpcEndpoints**
- **ec2:DetachInternetGateway**
- **ec2:DisassociateRouteTable**
- **ec2:ReplaceRouteTableAssociation**



注記

既存の VPC を使用する場合、アカウントではネットワークリソースの削除にこれらのパーミッションを必要としません。

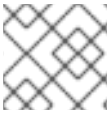
例2.36 マニフェストの作成に必要な追加の IAM および S3 パーミッション

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**

- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **s3:PutBucketPublicAccessBlock**
- **s3:GetBucketPublicAccessBlock**
- **s3:PutLifecycleConfiguration**
- **s3:HeadBucket**
- **s3:ListBucketMultipartUploads**
- **s3:AbortMultipartUpload**

2.10.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

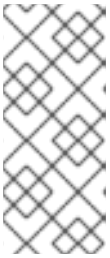
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> (1)
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

(1) **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

2.10.6. AWS のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Amazon Web Services (AWS) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

2.10.6.1. インストール設定ファイルの作成

インストールプログラムがクラスターをデプロイするために必要なインストール設定ファイルを生成し、カスタマイズします。

前提条件

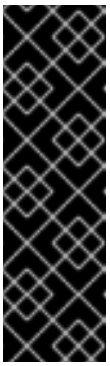
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。

手順

1. **install-config.yaml** ファイルを取得します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **aws** を選択します。
      iii. AWS プロファイルをコンピューターに保存していない場合、インストールプログラムを実行するように設定したユーザーの AWS アクセスキー ID およびシークレットアクセスキーを入力します。
      iv. クラスターのデプロイ先とする AWS リージョンを選択します。
      v. クラスターに設定した Route 53 サービスのベースドメインを選択します。
      vi. クラスターの記述名を入力します。
      vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを編集し、以下の **compute** スタンザに示されるようにコンピュートレプリカ (ワーカーレプリカとしても知られる) の数を **0** に設定します。

```
compute:
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

   a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

   ```
   pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email":
   "you@example.com"}}}'
   ```

   <local_registry> については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

   b. **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

   ```
   additionalTrustBundle: |
     -----BEGIN CERTIFICATE-----

   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
     -----END CERTIFICATE-----
   ```

   c. イメージコンテンツリソースを追加します。

   ```
   imageContentSources:
   - mirrors:
     - <local_registry>/<local_repository_name>/release
     source: quay.io/openshift-release-dev/ocp-release
   - mirrors:
     - <local_registry>/<local_repository_name>/release
     source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
   ```

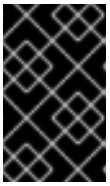
   コマンドの出力の **imageContentSources** セクションを使用して、リポジトリー、またはネットワークが制限されたネットワークに取り込んだメディアからのコンテンツをミラーリングする際に使用した値をミラーリングします。

   d. オプション: パブリッシュストラテジーを **Internal** に設定します。

   ```
   publish: Internal
   ```

   このオプションを設定すると、内部 Ingress コントローラーおよびプライベートロードバランサーを作成します。

4. オプション: **install-config.yaml** ファイルをバックアップします。



重要

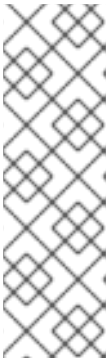
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

2.10.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されま

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

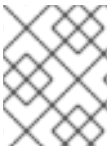
4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

2.10.6.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。
   c. ファイルを保存し、終了します。
5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
```

```
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

2.10.7. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Amazon Web Services (AWS) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される CloudFormation テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

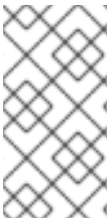
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

2.10.8. AWS での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Amazon Web Services (AWS) で作成する必要があります。VPN およびルートテーブルを含む、各種要件を満たすように VPC をカスタマイズできます。VPC を作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "VpcCidr", 1
    "ParameterValue": "10.0.0.0/16" 2
  },
  {
    "ParameterKey": "AvailabilityZoneCount", 3
    "ParameterValue": "1" 4
  },
  {
    "ParameterKey": "SubnetBits", 5
    "ParameterValue": "12" 6
  }
]
```

1 VPC の CIDR ブロック。

2. **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。
3. VPC をデプロイするアベイラビリティーゾーンの数。
4. **1** から **3** の間の整数を指定します。
5. 各アベイラビリティーゾーン内の各サブネットのサイズ。
6. **5** から **13** の間の整数を指定します。 ここで、**5** は **/27** であり、 **13** は **/19** です。

2. このトピックの**VPC の CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

3. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1. **<name>** は **cluster-vpc** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

VpcId	VPC の ID。
PublicSubnetIds	新規パブリックサブネットの ID。
PrivateSubnetIds	新規プライベートサブネットの ID。

2.10.8.1. VPC の CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例2.37 VPC の CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for Best Practice VPC with 1-3 AZs

Parameters:
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  AvailabilityZoneCount:
    ConstraintDescription: "The number of availability zones. (Min: 1, Max: 3)"
    MinValue: 1
    MaxValue: 3
    Default: 1
    Description: "How many AZs to create VPC subnets for. (Min: 1, Max: 3)"
    Type: Number
  SubnetBits:
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/19-27.
    MinValue: 5
    MaxValue: 13
    Default: 12
    Description: "Size of each subnet to create within the availability zones. (Min: 5 = /27, Max: 13 = /19)"
    Type: Number

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcCidr
      - SubnetBits
    - Label:
        default: "Availability Zones"
      Parameters:
      - AvailabilityZoneCount
    ParameterLabels:
      AvailabilityZoneCount:
        default: "Availability Zone Count"
      VpcCidr:
        default: "VPC CIDR"
      SubnetBits:
        default: "Bits Per Subnet"

Conditions:
  DoAz3: !Equals [3, !Ref AvailabilityZoneCount]
  DoAz2: !Or [!Equals [2, !Ref AvailabilityZoneCount], Condition: DoAz3]
```

```
Resources:
  VPC:
    Type: "AWS::EC2::VPC"
    Properties:
      EnableDnsSupport: "true"
      EnableDnsHostnames: "true"
      CidrBlock: !Ref VpcCidr
  PublicSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [0, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [1, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PublicSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [2, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  InternetGateway:
    Type: "AWS::EC2::InternetGateway"
  GatewayToInternet:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      VpcId: !Ref VPC
      InternetGatewayId: !Ref InternetGateway
  PublicRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PublicRoute:
    Type: "AWS::EC2::Route"
    DependsOn: GatewayToInternet
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway
  PublicSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet
      RouteTableId: !Ref PublicRouteTable
```

```
  PublicSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable
  PublicSubnetRouteTableAssociation3:
    Condition: DoAz3
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PublicSubnet3
      RouteTableId: !Ref PublicRouteTable
  PrivateSubnet:
    Type: "AWS::EC2::Subnet"
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [3, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 0
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable:
    Type: "AWS::EC2::RouteTable"
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Properties:
      SubnetId: !Ref PrivateSubnet
      RouteTableId: !Ref PrivateRouteTable
  NAT:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP
        - AllocationId
      SubnetId: !Ref PublicSubnet
  EIP:
    Type: "AWS::EC2::EIP"
    Properties:
      Domain: vpc
  Route:
    Type: "AWS::EC2::Route"
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT
  PrivateSubnet2:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
```

```
      CidrBlock: !Select [4, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 1
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable2:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz2
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation2:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
    Condition: DoAz2
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable2
  NAT2:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz2
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP2
        - AllocationId
      SubnetId: !Ref PublicSubnet2
  EIP2:
    Type: "AWS::EC2::EIP"
    Condition: DoAz2
    Properties:
      Domain: vpc
  Route2:
    Type: "AWS::EC2::Route"
    Condition: DoAz2
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT2
  PrivateSubnet3:
    Type: "AWS::EC2::Subnet"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
      CidrBlock: !Select [5, !Cidr [!Ref VpcCidr, 6, !Ref SubnetBits]]
      AvailabilityZone: !Select
      - 2
      - Fn::GetAZs: !Ref "AWS::Region"
  PrivateRouteTable3:
    Type: "AWS::EC2::RouteTable"
    Condition: DoAz3
    Properties:
      VpcId: !Ref VPC
  PrivateSubnetRouteTableAssociation3:
    Type: "AWS::EC2::SubnetRouteTableAssociation"
```

```
    Condition: DoAz3
    Properties:
      SubnetId: !Ref PrivateSubnet3
      RouteTableId: !Ref PrivateRouteTable3
  NAT3:
    DependsOn:
    - GatewayToInternet
    Type: "AWS::EC2::NatGateway"
    Condition: DoAz3
    Properties:
      AllocationId:
        "Fn::GetAtt":
        - EIP3
        - AllocationId
      SubnetId: !Ref PublicSubnet3
  EIP3:
    Type: "AWS::EC2::EIP"
    Condition: DoAz3
    Properties:
      Domain: vpc
  Route3:
    Type: "AWS::EC2::Route"
    Condition: DoAz3
    Properties:
      RouteTableId:
        Ref: PrivateRouteTable3
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId:
        Ref: NAT3
  S3Endpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal: '*'
          Action:
          - '*'
          Resource:
          - '*'
      RouteTableIds:
      - !Ref PublicRouteTable
      - !Ref PrivateRouteTable
      - !If [DoAz2, !Ref PrivateRouteTable2, !Ref "AWS::NoValue"]
      - !If [DoAz3, !Ref PrivateRouteTable3, !Ref "AWS::NoValue"]
      ServiceName: !Join
      - ''
      - - com.amazonaws.
        - !Ref 'AWS::Region'
        - .s3
      VpcId: !Ref VPC

Outputs:
  VpcId:
    Description: ID of the new VPC.
```

```
    Value: !Ref VPC
  PublicSubnetIds:
    Description: Subnet IDs of the public subnets.
    Value:
      !Join [
        ",",
        [!Ref PublicSubnet, !If [DoAz2, !Ref PublicSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PublicSubnet3, !Ref "AWS::NoValue"]]
      ]
  PrivateSubnetIds:
    Description: Subnet IDs of the private subnets.
    Value:
      !Join [
        ",",
        [!Ref PrivateSubnet, !If [DoAz2, !Ref PrivateSubnet2, !Ref "AWS::NoValue"], !If [DoAz3, !Ref
PrivateSubnet3, !Ref "AWS::NoValue"]]
      ]
```

2.10.9. AWS でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散 (classic または network) を Amazon Web Services (AWS) で設定する必要があります。これらのコンポーネントを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。 これにより、ホストゾーンおよびサブネットのタグも作成されます。

単一 VPC 内でテンプレートを複数回実行することができます。

注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。

手順

1. クラスターの **install-config.yaml** ファイルに指定した Route 53 ゾーンのホストゾーン ID を取得します。この ID は、AWS コンソールから、または以下のコマンドを実行して取得できます。

重要

単一行にコマンドを入力してください。

```
$ aws route53 list-hosted-zones-by-name |
    jq --arg name "<route53_domain>." \ 1
    -r '.HostedZones | .[] | select(.Name=="\($name)") | .Id'
```

1 **<route53_domain>** について、クラスターの **install-config.yaml** ファイルを生成した時に作成した Route 53 ベースドメインを指定します。

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
 {
  "ParameterKey": "ClusterName", 1
  "ParameterValue": "mycluster" 2
 },
 {
  "ParameterKey": "InfrastructureName", 3
  "ParameterValue": "mycluster-<random_string>" 4
 },
 {
  "ParameterKey": "HostedZoneId", 5
  "ParameterValue": "<random_string>" 6
 },
 {
  "ParameterKey": "HostedZoneName", 7
  "ParameterValue": "example.com" 8
 },
 {
  "ParameterKey": "PublicSubnets", 9
  "ParameterValue": "subnet-<random_string>" 10
 },
 {
  "ParameterKey": "PrivateSubnets", 11
  "ParameterValue": "subnet-<random_string>" 12
 },
 {
  "ParameterKey": "VpcId", 13
  "ParameterValue": "vpc-<random_string>" 14
 }
]
```

1 ホスト名などに使用するクラスターを表す短いクラスターの名前。

2 クラスターの **install-config.yaml** ファイルを生成した時に使用したクラスター名を指定します。

3 クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

4 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

5 ターゲットの登録に使用する Route 53 パブリックトゾーン ID。

6 **Z21IXYZABCZ2A4** に類する形式の Route 53 パブリックゾーン ID を指定します。この値は AWS コンソールから取得できます。

7 ターゲットの登録に使用する Route 53 ゾーン。

8 クラスターの **install-config.yaml** ファイルを生成した時に使用した Route 53 ベースドメインを指定します。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

9 VPC 用に作成したパブリックサブネット。

10 VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

11 VPC 用に作成したプライベートサブネット。

12 VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。

13 クラスター用に作成した VPC。

14 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

3. このトピックの**ネットワークおよびロードバランサーの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM
```

1 **<name>** は **cluster-dns** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

PrivateHostedZoneId	プライベート DNS のホストゾーン ID。
ExternalApiLoadBalancerName	外部 API ロードバランサーのフルネーム。
InternalApiLoadBalancerName	内部 API ロードバランサーのフルネーム。
ApiServerDnsName	API サーバーの完全ホスト名。
RegisterNlbIpTargetsLambda	これらのロードバランサーの登録/登録解除に役立つ Lambda ARN。
ExternalApiTargetGroupArn	外部 API ターゲットグループの ARN。
InternalApiTargetGroupArn	内部 API ターゲットグループの ARN。
InternalServiceTargetGroupArn	内部サービスターゲットグループの ARN。

2.10.9.1. ネットワークおよびロードバランサーの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例2.38 ネットワークおよびロードバランサーの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Network Elements (Route53 & LBs)

Parameters:
  ClusterName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Cluster name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, representative cluster name to use for host names and other identifying
names.
```

```
    Type: String
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  HostedZoneId:
    Description: The Route53 public zone ID to register the targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  HostedZoneName:
    Description: The Route53 zone to register the targets with, such as example.com. Omit the
trailing period.
    Type: String
    Default: "example.com"
  PublicSubnets:
    Description: The internet-facing subnets.
    Type: List<AWS::EC2::Subnet::Id>
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - ClusterName
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - PublicSubnets
      - PrivateSubnets
    - Label:
        default: "DNS"
      Parameters:
      - HostedZoneName
      - HostedZoneId
    ParameterLabels:
      ClusterName:
        default: "Cluster Name"
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      PublicSubnets:
```

```
      default: "Public Subnets"
    PrivateSubnets:
      default: "Private Subnets"
    HostedZoneName:
      default: "Public Hosted Zone Name"
    HostedZoneId:
      default: "Public Hosted Zone ID"

Resources:
  ExtApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "ext"]]
      IpAddressType: ipv4
      Subnets: !Ref PublicSubnets
      Type: network

  IntApiElb:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: !Join ["-", [!Ref InfrastructureName, "int"]]
      Scheme: internal
      IpAddressType: ipv4
      Subnets: !Ref PrivateSubnets
      Type: network

  IntDns:
    Type: "AWS::Route53::HostedZone"
    Properties:
      HostedZoneConfig:
        Comment: "Managed by CloudFormation"
      Name: !Join [".", [!Ref ClusterName, !Ref HostedZoneName]]
      HostedZoneTags:
      - Key: Name
        Value: !Join ["-", [!Ref InfrastructureName, "int"]]
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "owned"
      VPCs:
      - VPCId: !Ref VpcId
        VPCRegion: !Ref "AWS::Region"

  ExternalApiServerRecord:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Alias record for the API server
      HostedZoneId: !Ref HostedZoneId
      RecordSets:
      - Name:
          !Join [
            ".",
            ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
          ]
        Type: A
        AliasTarget:
          HostedZoneId: !GetAtt ExtApiElb.CanonicalHostedZoneID
          DNSName: !GetAtt ExtApiElb.DNSName
```

```
InternalApiServerRecord:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    Comment: Alias record for the API server
    HostedZoneId: !Ref IntDns
    RecordSets:
    - Name:
        !Join [
          ".",
          ["api", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName
    - Name:
        !Join [
          ".",
          ["api-int", !Ref ClusterName, !Join ["", [!Ref HostedZoneName, "."]]],
        ]
      Type: A
      AliasTarget:
        HostedZoneId: !GetAtt IntApiElb.CanonicalHostedZoneID
        DNSName: !GetAtt IntApiElb.DNSName

ExternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: ExternalApiTargetGroup
    LoadBalancerArn:
      Ref: ExtApiElb
    Port: 6443
    Protocol: TCP

ExternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60
```

```
InternalApiListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalApiTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 6443
    Protocol: TCP

InternalApiTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/readyz"
    HealthCheckPort: 6443
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 6443
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

InternalServiceInternalListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
    - Type: forward
      TargetGroupArn:
        Ref: InternalServiceTargetGroup
    LoadBalancerArn:
      Ref: IntApiElb
    Port: 22623
    Protocol: TCP

InternalServiceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    HealthCheckIntervalSeconds: 10
    HealthCheckPath: "/healthz"
    HealthCheckPort: 22623
    HealthCheckProtocol: HTTPS
    HealthyThresholdCount: 2
    UnhealthyThresholdCount: 2
    Port: 22623
    Protocol: TCP
    TargetType: ip
    VpcId:
      Ref: VpcId
```

```
    TargetGroupAttributes:
    - Key: deregistration_delay.timeout_seconds
      Value: 60

  RegisterTargetLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "nlb", "lambda", "role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalApiTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref InternalServiceTargetGroup
          - Effect: "Allow"
            Action:
              [
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:DeregisterTargets",
              ]
            Resource: !Ref ExternalApiTargetGroup

  RegisterNlbIpTargets:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
        - "RegisterTargetLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
```

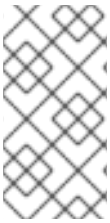
```
          import boto3
          import cfnresponse
          def handler(event, context):
            elb = boto3.client('elbv2')
            if event['RequestType'] == 'Delete':
              elb.deregister_targets(TargetGroupArn=event['ResourceProperties']
['TargetArn'],Targets=[{'Id': event['ResourceProperties']['TargetIp']}])
            elif event['RequestType'] == 'Create':
              elb.register_targets(TargetGroupArn=event['ResourceProperties']['TargetArn'],Targets=
[{'Id': event['ResourceProperties']['TargetIp']}])
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['TargetArn']+event['ResourceProperties']['TargetIp'])
      Runtime: "python3.7"
      Timeout: 120

  RegisterSubnetTagsLambdaIamRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ["-", [!Ref InfrastructureName, "subnet-tags-lambda-role"]]
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "lambda.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "subnet-tagging-policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              [
                "ec2:DeleteTags",
                "ec2:CreateTags"
              ]
            Resource: "arn:aws:ec2:*:*:subnet/*"
          - Effect: "Allow"
            Action:
              [
                "ec2:DescribeSubnets",
                "ec2:DescribeTags"
              ]
            Resource: "*"

  RegisterSubnetTags:
    Type: "AWS::Lambda::Function"
    Properties:
      Handler: "index.handler"
      Role:
        Fn::GetAtt:
```

```
        - "RegisterSubnetTagsLambdaIamRole"
        - "Arn"
      Code:
        ZipFile: |
          import json
          import boto3
          import cfnresponse
          def handler(event, context):
            ec2_client = boto3.client('ec2')
            if event['RequestType'] == 'Delete':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.delete_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName']}]);
            elif event['RequestType'] == 'Create':
              for subnet_id in event['ResourceProperties']['Subnets']:
                ec2_client.create_tags(Resources=[subnet_id], Tags=[{'Key': 'kubernetes.io/cluster/' +
event['ResourceProperties']['InfrastructureName'], 'Value': 'shared'}]);
            responseData = {}
            cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
event['ResourceProperties']['InfrastructureName']+event['ResourceProperties']['Subnets'][0])
      Runtime: "python3.7"
      Timeout: 120

  RegisterPublicSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PublicSubnets

  RegisterPrivateSubnetTags:
    Type: Custom::SubnetRegister
    Properties:
      ServiceToken: !GetAtt RegisterSubnetTags.Arn
      InfrastructureName: !Ref InfrastructureName
      Subnets: !Ref PrivateSubnets

Outputs:
  PrivateHostedZoneId:
    Description: Hosted zone ID for the private DNS, which is required for private records.
    Value: !Ref IntDns
  ExternalApiLoadBalancerName:
    Description: Full name of the external API load balancer.
    Value: !GetAtt ExtApiElb.LoadBalancerFullName
  InternalApiLoadBalancerName:
    Description: Full name of the internal API load balancer.
    Value: !GetAtt IntApiElb.LoadBalancerFullName
  ApiServerDnsName:
    Description: Full hostname of the API server, which is required for the Ignition config files.
    Value: !Join [".", ["api-int", !Ref ClusterName, !Ref HostedZoneName]]
  RegisterNlbIpTargetsLambda:
    Description: Lambda ARN useful to help register or deregister IP targets for these load
balancers.
    Value: !GetAtt RegisterNlbIpTargets.Arn
  ExternalApiTargetGroupArn:
    Description: ARN of the external API target group.
```

```
  Value: !Ref ExternalApiTargetGroup
InternalApiTargetGroupArn:
  Description: ARN of the internal API target group.
  Value: !Ref InternalApiTargetGroup
InternalServiceTargetGroupArn:
  Description: ARN of the internal service target group.
  Value: !Ref InternalServiceTargetGroup
```

2.10.10. AWS でのセキュリティーグループおよびロールの作成

OpenShift Container Platform クラスターで使用するセキュリティーグループおよびロールを Amazon Web Services (AWS) で作成する必要があります。これらのコンポーネントを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。



注記

提供される CloudFormation テンプレートを使用して AWS インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。

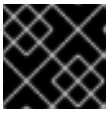
手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "VpcCidr", 3
    "ParameterValue": "10.0.0.0/16" 4
  },
  {
    "ParameterKey": "PrivateSubnets", 5
    "ParameterValue": "subnet-<random_string>" 6
  },
  {
    "ParameterKey": "VpcId", 7
    "ParameterValue": "vpc-<random_string>" 8
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. VPC の CIDR ブロック。
4. **x.x.x.x/16-24** の形式で定義した VPC に使用した CIDR ブロックパラメーターを指定します。
5. VPC 用に作成したプライベートサブネット。
6. VPC の CloudFormation テンプレートの出力から **PrivateSubnetIds** 値を指定します。
7. クラスター用に作成した VPC。
8. VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

2. このトピックの**セキュリティーオブジェクトの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なセキュリティーグループおよびロールについて記述しています。
3. テンプレートを起動します。



重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> (1)
    --template-body file://<template>.yaml (2)
    --parameters file://<parameters>.json (3)
    --capabilities CAPABILITY_NAMED_IAM
```

1. **<name>** は **cluster-secs** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。
2. **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。
3. **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

4. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

MasterSecurityGroupId	マスターセキュリティーグループ ID
WorkerSecurityGroupId	ワーカーセキュリティーグループ ID
MasterInstanceProfile	マスター IAM インスタンスプロファイル
WorkerInstanceProfile	ワーカー IAM インスタンスプロファイル

2.10.10.1. セキュリティーオブジェクトの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なセキュリティーオブジェクトをデプロイすることができます。

例2.39 セキュリティーオブジェクトの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Security Elements (Security Groups & IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or
used by the cluster.
    Type: String
  VpcCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-
4][0-9]|25[0-5])(\/(1[6-9]|2[0-4]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/16-24.
    Default: 10.0.0.0/16
    Description: CIDR block for VPC.
    Type: String
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  PrivateSubnets:
    Description: The internal subnets.
    Type: List<AWS::EC2::Subnet::Id>

Metadata:
  AWS::CloudFormation::Interface:
```

```
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - VpcCidr
      - PrivateSubnets
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      VpcCidr:
        default: "VPC CIDR"
      PrivateSubnets:
        default: "Private Subnets"

Resources:
  MasterSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Master Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        ToPort: 6443
        FromPort: 6443
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22623
        ToPort: 22623
        CidrIp: !Ref VpcCidr
      VpcId: !Ref VpcId

  WorkerSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Worker Security Group
      SecurityGroupIngress:
      - IpProtocol: icmp
        FromPort: 0
        ToPort: 0
        CidrIp: !Ref VpcCidr
      - IpProtocol: tcp
        FromPort: 22
```

```
      ToPort: 22
      CidrIp: !Ref VpcCidr
    VpcId: !Ref VpcId

MasterIngressEtcd:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: etcd
    FromPort: 2379
    ToPort: 2380
    IpProtocol: tcp

MasterIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressWorkerVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

MasterIngressGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressWorkerGeneve:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Geneve packets
    FromPort: 6081
    ToPort: 6081
    IpProtocol: udp

MasterIngressInternal:
  Type: AWS::EC2::SecurityGroupIngress
```

```
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressWorkerInternal:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: tcp

MasterIngressInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressWorkerInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

MasterIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
    IpProtocol: tcp

MasterIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes kubelet, scheduler and controller manager
    FromPort: 10250
    ToPort: 10259
```

```
    IpProtocol: tcp

MasterIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressWorkerIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

MasterIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

MasterIngressWorkerIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt MasterSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: udp

WorkerIngressVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Vxlan packets
    FromPort: 4789
    ToPort: 4789
    IpProtocol: udp

WorkerIngressMasterVxlan:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
```

```
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Vxlan packets
      FromPort: 4789
      ToPort: 4789
      IpProtocol: udp

  WorkerIngressGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressMasterGeneve:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Geneve packets
      FromPort: 6081
      ToPort: 6081
      IpProtocol: udp

  WorkerIngressInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressMasterInternal:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: tcp

  WorkerIngressInternalUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
      Description: Internal cluster communication
      FromPort: 9000
      ToPort: 9999
      IpProtocol: udp
```

```
WorkerIngressMasterInternalUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal cluster communication
    FromPort: 9000
    ToPort: 9999
    IpProtocol: udp

WorkerIngressKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes secure kubelet port
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressWorkerKube:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Internal Kubernetes communication
    FromPort: 10250
    ToPort: 10250
    IpProtocol: tcp

WorkerIngressIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressMasterIngressServices:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
    Description: Kubernetes ingress services
    FromPort: 30000
    ToPort: 32767
    IpProtocol: tcp

WorkerIngressIngressServicesUDP:
  Type: AWS::EC2::SecurityGroupIngress
  Properties:
    GroupId: !GetAtt WorkerSecurityGroup.GroupId
    SourceSecurityGroupId: !GetAtt WorkerSecurityGroup.GroupId
    Description: Kubernetes ingress services
```

```
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  WorkerIngressMasterIngressServicesUDP:
    Type: AWS::EC2::SecurityGroupIngress
    Properties:
      GroupId: !GetAtt WorkerSecurityGroup.GroupId
      SourceSecurityGroupId: !GetAtt MasterSecurityGroup.GroupId
      Description: Kubernetes ingress services
      FromPort: 30000
      ToPort: 32767
      IpProtocol: udp

  MasterIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "master", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:AttachVolume"
            - "ec2:AuthorizeSecurityGroupIngress"
            - "ec2:CreateSecurityGroup"
            - "ec2:CreateTags"
            - "ec2:CreateVolume"
            - "ec2:DeleteSecurityGroup"
            - "ec2:DeleteVolume"
            - "ec2:Describe*"
            - "ec2:DetachVolume"
            - "ec2:ModifyInstanceAttribute"
            - "ec2:ModifyVolume"
            - "ec2:RevokeSecurityGroupIngress"
            - "elasticloadbalancing:AddTags"
            - "elasticloadbalancing:AttachLoadBalancerToSubnets"
            - "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer"
            - "elasticloadbalancing:CreateListener"
            - "elasticloadbalancing:CreateLoadBalancer"
            - "elasticloadbalancing:CreateLoadBalancerPolicy"
            - "elasticloadbalancing:CreateLoadBalancerListeners"
            - "elasticloadbalancing:CreateTargetGroup"
            - "elasticloadbalancing:ConfigureHealthCheck"
            - "elasticloadbalancing:DeleteListener"
            - "elasticloadbalancing:DeleteLoadBalancer"
```

```
            - "elasticloadbalancing:DeleteLoadBalancerListeners"
            - "elasticloadbalancing:DeleteTargetGroup"
            - "elasticloadbalancing:DeregisterInstancesFromLoadBalancer"
            - "elasticloadbalancing:DeregisterTargets"
            - "elasticloadbalancing:Describe*"
            - "elasticloadbalancing:DetachLoadBalancerFromSubnets"
            - "elasticloadbalancing:ModifyListener"
            - "elasticloadbalancing:ModifyLoadBalancerAttributes"
            - "elasticloadbalancing:ModifyTargetGroup"
            - "elasticloadbalancing:ModifyTargetGroupAttributes"
            - "elasticloadbalancing:RegisterInstancesWithLoadBalancer"
            - "elasticloadbalancing:RegisterTargets"
            - "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer"
            - "elasticloadbalancing:SetLoadBalancerPoliciesOfListener"
            - "kms:DescribeKey"
            Resource: "*"

  MasterInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "MasterIamRole"

  WorkerIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "worker", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
            - "ec2:DescribeInstances"
            - "ec2:DescribeRegions"
            Resource: "*"

  WorkerInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Roles:
      - Ref: "WorkerIamRole"

Outputs:
  MasterSecurityGroupId:
    Description: Master Security Group ID
    Value: !GetAtt MasterSecurityGroup.GroupId
```

```
  WorkerSecurityGroupId:
    Description: Worker Security Group ID
    Value: !GetAtt WorkerSecurityGroup.GroupId

  MasterInstanceProfile:
    Description: Master IAM Instance Profile
    Value: !Ref MasterInstanceProfile

  WorkerInstanceProfile:
    Description: Worker IAM Instance Profile
    Value: !Ref WorkerInstanceProfile
```

2.10.11. AWS インフラストラクチャーの RHCOS AMI

OpenShift Container Platform ノードについて、Amazon Web Services (AWS) ゾーンの有効な Red Hat Enterprise Linux CoreOS (RHCOS) AMI を使用する必要があります。

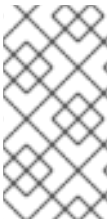
表2.25 RHCOS AMI

AWS ゾーン	AWS AMI
ap-northeast-1	**ami-0530d04240177f118**
ap-northeast-2	**ami-09e4cd700276785d2**
ap-south-1	**ami-0754b15d212830477**
ap-southeast-1	**ami-03b46cc4b1518c5a8**
ap-southeast-2	**ami-0a5b99ab2234a4e6a**
ca-central-1	**ami-012bc4ee3b6c673bc**
eu-central-1	**ami-02e08df1201f1c2f8**
eu-north-1	**ami-0309c9d2fadcb2d5a**
eu-west-1	**ami-0bdd69d8e7cd18188**
eu-west-2	**ami-0e610e967a62dbdfa**
eu-west-3	**ami-0e817e26f638a71ac**
me-south-1	**ami-024117d7c87b7ff08**
sa-east-1	**ami-08e62f746b94950c1**

AWS ゾーン	AWS AMI
us-east-1	**ami-077ede5bed2e431ea**
us-east-2	**ami-0f4ecf819275850dd**
us-west-1	**ami-0c4990e435bc6c5fe**
us-west-2	**ami-000d6e92357ac605c**

2.10.12. AWS でのブートストラップノードの作成

OpenShift Container Platform クラスターの初期化で使用するブートストラップノードを Amazon Web Services (AWS) で作成する必要があります。このノードを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。



注記

提供される CloudFormation テンプレートを使用してブートストラップノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. **bootstrap.ign** Ignition 設定ファイルをクラスターに送るための場所を指定します。このファイルはインストールディレクトリーに置かれます。これを実行するための 1 つの方法として、クラスターのリージョンに S3 バケットを作成し、Ignition 設定ファイルをこれにアップロードします。

重要

提供される CloudFormation テンプレートでは、クラスターの Ignition 設定ファイルは S3 バケットから送られることを前提としています。このファイルを別の場所から送ることを選択する場合は、テンプレートを変更する必要があります。

注記

ブートストラップ Ignition 設定ファイルには、X.509 キーのようなシークレットが含まれません。以下の手順では、S3 バケットの基本的なセキュリティーを提供します。追加のセキュリティーを提供するには、OpenShift IAM ユーザーなどの特定のユーザーのみがバケットに含まれるオブジェクトにアクセスできるように S3 バケットポリシーを有効にできます。S3 を完全に回避し、ブートストラップマシンが到達できるアドレスからブートストラップ Ignition 設定ファイルを送ることができます。

a. バケットを作成します。

```
$ aws s3 mb s3://<cluster-name>-infra 1
```

1 **<cluster-name>-infra** はバケット名です。

b. **bootstrap.ign** Ignition 設定ファイルをバケットにアップロードします。

```
$ aws s3 cp bootstrap.ign s3://<cluster-name>-infra/bootstrap.ign
```

c. ファイルがアップロードされていることを確認します。

```
$ aws s3 ls s3://<cluster-name>-infra/
```

出力例

```
2019-04-03 16:15:16     314878 bootstrap.ign
```

2. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AllowedBootstrapSshCidr", 5
    "ParameterValue": "0.0.0.0/0" 6
  },
  {
    "ParameterKey": "PublicSubnet", 7
    "ParameterValue": "subnet-<random_string>" 8
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 9
    "ParameterValue": "sg-<random_string>" 10
  },
```

```
  {
    "ParameterKey": "VpcId", (11)
    "ParameterValue": "vpc-<random_string>" (12)
  },
  {
    "ParameterKey": "BootstrapIgnitionLocation", (13)
    "ParameterValue": "s3://<bucket_name>/bootstrap.ign" (14)
  },
  {
    "ParameterKey": "AutoRegisterELB", (15)
    "ParameterValue": "yes" (16)
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", (17)
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" (18)
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", (19)
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" (20)
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", (21)
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (22)
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", (23)
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" (24)
  }
]
```

(1) クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。

(2) 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。

(3) ブートストラップノードに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。

(4) 有効な **AWS::EC2::Image::Id** 値を指定します。

(5) ブートストラップノードへの SSH アクセスを許可する CIDR ブロック。

(6) **x.x.x.x/16-24** 形式で CIDR ブロックを指定します。

(7) ブートストラップを起動するために VPC に関連付けられるパブリックサブネット。

(8) VPC の CloudFormation テンプレートの出力から **PublicSubnetIds** 値を指定します。

(9) マスターセキュリティーグループ ID (一時ルールの登録用)。

10 セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

11 作成されたリソースが属する VPC。

12 VPC の CloudFormation テンプレートの出力から **VpcId** 値を指定します。

13 ブートストラップの Ignition 設定ファイルをフェッチする場所。

14 **s3://<bucket_name>/bootstrap.ign** の形式で S3 バケットおよびファイル名を指定します。

15 ネットワークロードバランサー (NLB) を登録するかどうか。

16 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

17 NLB IP ターゲット登録 lambda グループの ARN。

18 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。

19 外部 API ロードバランサーのターゲットグループの ARN。

20 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。

21 内部 API ロードバランサーのターゲットグループの ARN。

22 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。

23 内部サービスバランサーのターゲットグループの ARN。

24 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。

3. このトピックの**ブートストラップマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
    --capabilities CAPABILITY_NAMED_IAM
```

1 **<name>** は **cluster-bootstrap** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

StackStatus が **CREATE_COMPLETE** を表示した後に、出力には以下のパラメーターの値が表示されます。これらのパラメーターの値をクラスターを作成するために実行する他の CloudFormation テンプレートに指定する必要があります。

Bootstrap InstanceId	ブートストラップインスタンス ID。
Bootstrap PublicIp	ブートストラップノードのパブリック IP アドレス。
Bootstrap PrivateIp	ブートストラップノードのプライベート IP アドレス。

2.10.12.1. ブートストラップマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイできます。

例2.40 ブートストラップマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Bootstrap (EC2 Instance, Security Groups and IAM)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a maximum of 27 characters.
    Description: A short, unique cluster ID used to tag cloud resources and identify items owned or used by the cluster.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AllowedBootstrapSshCidr:
    AllowedPattern: ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/([0-9]|1[0-9]|2[0-9]|3[0-2]))$
    ConstraintDescription: CIDR block parameter must be in the form x.x.x.x/0-32.
    Default: 0.0.0.0/0
    Description: CIDR block to allow SSH access to the bootstrap node.
```

```
    Type: String
  PublicSubnet:
    Description: The public subnet to launch the bootstrap node into.
    Type: AWS::EC2::Subnet::Id
  MasterSecurityGroupId:
    Description: The master security group ID for registering temporary rules.
    Type: AWS::EC2::SecurityGroup::Id
  VpcId:
    Description: The VPC-scoped resources will belong to this VPC.
    Type: AWS::EC2::VPC::Id
  BootstrapIgnitionLocation:
    Default: s3://my-s3-bucket/bootstrap.ign
    Description: Ignition config file location.
    Type: String
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group.
    Type: String

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - RhcosAmi
      - BootstrapIgnitionLocation
      - MasterSecurityGroupId
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - PublicSubnet
    - Label:
        default: "Load Balancer Automation"
      Parameters:
```

```
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      AllowedBootstrapSshCidr:
        default: "Allowed SSH Source"
      PublicSubnet:
        default: "Public Subnet"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
        default: "Bootstrap Ignition Source"
      MasterSecurityGroupId:
        default: "Master Security Group ID"
      AutoRegisterELB:
        default: "Use Provided ELB Automation"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]

Resources:
  BootstrapIamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: "Allow"
          Principal:
            Service:
            - "ec2.amazonaws.com"
          Action:
          - "sts:AssumeRole"
      Path: "/"
      Policies:
      - PolicyName: !Join ["-", [!Ref InfrastructureName, "bootstrap", "policy"]]
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action: "ec2:Describe*"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:AttachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "ec2:DetachVolume"
            Resource: "*"
          - Effect: "Allow"
            Action: "s3:GetObject"
```

```
            Resource: "*"

  BootstrapInstanceProfile:
    Type: "AWS::IAM::InstanceProfile"
    Properties:
      Path: "/"
      Roles:
      - Ref: "BootstrapIamRole"

  BootstrapSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster Bootstrap Security Group
      SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: !Ref AllowedBootstrapSshCidr
      - IpProtocol: tcp
        ToPort: 19531
        FromPort: 19531
        CidrIp: 0.0.0.0/0
      VpcId: !Ref VpcId

  BootstrapInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      IamInstanceProfile: !Ref BootstrapInstanceProfile
      InstanceType: "i3.large"
      NetworkInterfaces:
      - AssociatePublicIpAddress: "true"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "BootstrapSecurityGroup"
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "PublicSubnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"replace":{"source":"${S3Loc}","verification":{}}},"timeouts":
{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          S3Loc: !Ref BootstrapIgnitionLocation
        }

  RegisterBootstrapApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
```

```
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

  RegisterBootstrapInternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt BootstrapInstance.PrivateIp

Outputs:
  BootstrapInstanceId:
    Description: Bootstrap Instance ID.
    Value: !Ref BootstrapInstance

  BootstrapPublicIp:
    Description: The bootstrap node public IP address.
    Value: !GetAtt BootstrapInstance.PublicIp

  BootstrapPrivateIp:
    Description: The bootstrap node private IP address.
    Value: !GetAtt BootstrapInstance.PrivateIp
```

2.10.13. AWS でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Amazon Web Services (AWS) で作成する必要があります。これらのノードを作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。

注記

提供される CloudFormation テンプレートを使用してコントロールプレーンノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
    "ParameterKey": "RhcosAmi", 3
    "ParameterValue": "ami-<random_string>" 4
  },
  {
    "ParameterKey": "AutoRegisterDNS", 5
    "ParameterValue": "yes" 6
  },
  {
    "ParameterKey": "PrivateHostedZoneId", 7
    "ParameterValue": "<random_string>" 8
  },
  {
    "ParameterKey": "PrivateHostedZoneName", 9
    "ParameterValue": "mycluster.example.com" 10
  },
  {
    "ParameterKey": "Master0Subnet", 11
    "ParameterValue": "subnet-<random_string>" 12
  },
  {
    "ParameterKey": "Master1Subnet", 13
    "ParameterValue": "subnet-<random_string>" 14
  },
  {
    "ParameterKey": "Master2Subnet", 15
    "ParameterValue": "subnet-<random_string>" 16
  },
  {
    "ParameterKey": "MasterSecurityGroupId", 17
    "ParameterValue": "sg-<random_string>" 18
  },
  {
    "ParameterKey": "IgnitionLocation", 19
    "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/master"
20
  },
  {
    "ParameterKey": "CertificateAuthorities", 21
    "ParameterValue": "data:text/plain;charset=utf-8;base64,ABC...xYz==" 22
  },
  {
    "ParameterKey": "MasterInstanceProfileName", 23
    "ParameterValue": "<roles_stack>-MasterInstanceProfile-<random_string>" 24
  },
  {
```

```
    "ParameterKey": "MasterInstanceType", 25
    "ParameterValue": "m4.xlarge" 26
  },
  {
    "ParameterKey": "AutoRegisterELB", 27
    "ParameterValue": "yes" 28
  },
  {
    "ParameterKey": "RegisterNlbIpTargetsLambdaArn", 29
    "ParameterValue": "arn:aws:lambda:<region>:<account_number>:function:
<dns_stack_name>-RegisterNlbIpTargets-<random_string>" 30
  },
  {
    "ParameterKey": "ExternalApiTargetGroupArn", 31
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Exter-<random_string>" 32
  },
  {
    "ParameterKey": "InternalApiTargetGroupArn", 33
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 34
  },
  {
    "ParameterKey": "InternalServiceTargetGroupArn", 35
    "ParameterValue": "arn:aws:elasticloadbalancing:<region>:
<account_number>:targetgroup/<dns_stack_name>-Inter-<random_string>" 36
  }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. コントロールプレーンマシンに使用する最新の Red Hat Enterprise Linux CoreOS (RHCOS) AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. DNS etcd 登録を実行するかどうか。
6. **yes** または **no** を指定します。**yes** を指定する場合、ホストゾーンの情報を指定する必要があります。
7. etcd ターゲットの登録に使用する Route 53 プライベートゾーン ID。
8. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateHostedZoneId** 値を指定します。
9. ターゲットの登録に使用する Route 53 ゾーン。
10. **<cluster_name>.<domain_name>** を指定します。ここで、**<domain_name>** はクラスターの **install-config.yaml** ファイルの生成時に使用した Route 53 ベースドメインです。AWS コンソールに表示される末尾のピリド (.) は含めないでください。

11 13 15 コントロールプレーンマシンの起動に使用するサブネット (プライベートが望ましい)。

12 14 16 DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。

17 マスターノードに関連付けるマスターセキュリティーグループ ID。

18 セキュリティーグループおよびロールの CloudFormation テンプレートから **MasterSecurityGroupId** 値を指定します。

19 コントロールプレーンの Ignition 設定ファイルをフェッチする場所。

20 生成される Ignition 設定ファイルの場所を指定します (**https://api-int.<cluster_name>.<domain_name>:22623/config/master**)。

21 使用する base64 でエンコードされた認証局の文字列。

22 インストールディレクトリーにある **master.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

23 マスターロールに関連付ける IAM プロファイル。

24 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **MasterInstanceProfile** パラメーターの値を指定します。

25 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

26 許可される値:

- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

重要

m4 インスタンスタイプが **eu-west-3** などのリージョンで利用可能ではない場合、**m5.xlarge** などのように **m5** タイプを代わりに使用します。

27 ネットワークロードバランサー (NLB) を登録するかどうか。

28 **yes** または **no** を指定します。**yes** を指定する場合、Lambda Amazon Resource Name (ARN) の値を指定する必要があります。

29 NLB IP ターゲット登録 lambda グループの ARN。

30 DNS および負荷分散の CloudFormation テンプレートの出力から **RegisterNlbIpTargetsLambda** 値を指定します。

31 外部 API ロードバランサーのターゲットグループの ARN。

32 DNS および負荷分散の CloudFormation テンプレートの出力から **ExternalApiTargetGroupArn** 値を指定します。

33 内部 API ロードバランサーのターゲットグループの ARN。

34 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalApiTargetGroupArn** 値を指定します。

35 内部サービスバランサーのターゲットグループの ARN。

36 DNS および負荷分散の CloudFormation テンプレートの出力から **InternalServiceTargetGroupArn** 値を指定します。

2. このトピックの**コントロールプレーンマシンの CloudFormation テンプレート**セクションからテンプレートをコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。

3. **m5** インスタンスタイプを **MasterInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **MasterInstanceType.AllowedValues** パラメーターに追加します。

4. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-control-plane** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

5. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

2.10.13.1. コントロールプレーンマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例2.41 コントロールプレーンマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 master instances)

Parameters:
  InfrastructureName:
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  AutoRegisterDNS:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke DNS etcd registration, which requires Hosted Zone
information?
    Type: String
  PrivateHostedZoneId:
    Description: The Route53 private zone ID to register the etcd targets with, such as
Z21IXYZABCZ2A4.
    Type: String
  PrivateHostedZoneName:
    Description: The Route53 zone to register the targets with, such as cluster.example.com. Omit
the trailing period.
    Type: String
  Master0Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master1Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  Master2Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
```

```
  MasterSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/master
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  MasterInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  MasterInstanceType:
    Default: m4.xlarge
    Type: String
    AllowedValues:
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.8xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"
  AutoRegisterELB:
    Default: "yes"
    AllowedValues:
    - "yes"
    - "no"
    Description: Do you want to invoke NLB registration, which requires a Lambda ARN parameter?
    Type: String
  RegisterNlbIpTargetsLambdaArn:
    Description: ARN for NLB IP target registration lambda. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  ExternalApiTargetGroupArn:
    Description: ARN for external API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalApiTargetGroupArn:
    Description: ARN for internal API load balancer target group. Supply the value from the cluster
infrastructure or select "no" for AutoRegisterELB.
    Type: String
  InternalServiceTargetGroupArn:
    Description: ARN for internal service load balancer target group. Supply the value from the
cluster infrastructure or select "no" for AutoRegisterELB.
    Type: String
```

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - MasterInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - MasterSecurityGroupId
      - MasterInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - VpcId
      - AllowedBootstrapSshCidr
      - Master0Subnet
      - Master1Subnet
      - Master2Subnet
    - Label:
        default: "DNS"
      Parameters:
      - AutoRegisterDNS
      - PrivateHostedZoneName
      - PrivateHostedZoneId
    - Label:
        default: "Load Balancer Automation"
      Parameters:
      - AutoRegisterELB
      - RegisterNlbIpTargetsLambdaArn
      - ExternalApiTargetGroupArn
      - InternalApiTargetGroupArn
      - InternalServiceTargetGroupArn
    ParameterLabels:
      InfrastructureName:
        default: "Infrastructure Name"
      VpcId:
        default: "VPC ID"
      Master0Subnet:
        default: "Master-0 Subnet"
      Master1Subnet:
        default: "Master-1 Subnet"
      Master2Subnet:
        default: "Master-2 Subnet"
      MasterInstanceType:
        default: "Master Instance Type"
      MasterInstanceProfileName:
        default: "Master Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      BootstrapIgnitionLocation:
```

```
      default: "Master Ignition Source"
    CertificateAuthorities:
      default: "Ignition CA String"
    MasterSecurityGroupId:
      default: "Master Security Group ID"
    AutoRegisterDNS:
      default: "Use Provided DNS Automation"
    AutoRegisterELB:
      default: "Use Provided ELB Automation"
    PrivateHostedZoneName:
      default: "Private Hosted Zone Name"
    PrivateHostedZoneId:
      default: "Private Hosted Zone ID"

Conditions:
  DoRegistration: !Equals ["yes", !Ref AutoRegisterELB]
  DoDns: !Equals ["yes", !Ref AutoRegisterDNS]

Resources:
  Master0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master0Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster0:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp
```

```
  RegisterMaster0InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  RegisterMaster0InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master0.PrivateIp

  Master1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master1Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster1:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalApiTarget:
```

```
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalApiTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  RegisterMaster1InternalServiceTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref InternalServiceTargetGroupArn
      TargetIp: !GetAtt Master1.PrivateIp

  Master2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref MasterInstanceProfileName
      InstanceType: !Ref MasterInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "MasterSecurityGroupId"
        SubnetId: !Ref "Master2Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

  RegisterMaster2:
    Condition: DoRegistration
    Type: Custom::NLBRegister
    Properties:
      ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
      TargetArn: !Ref ExternalApiTargetGroupArn
      TargetIp: !GetAtt Master2.PrivateIp

  RegisterMaster2InternalApiTarget:
    Condition: DoRegistration
    Type: Custom::NLBRegister
```

```
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalApiTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

RegisterMaster2InternalServiceTarget:
  Condition: DoRegistration
  Type: Custom::NLBRegister
  Properties:
    ServiceToken: !Ref RegisterNlbIpTargetsLambdaArn
    TargetArn: !Ref InternalServiceTargetGroupArn
    TargetIp: !GetAtt Master2.PrivateIp

EtcdSrvRecords:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["_etcd-server-ssl._tcp", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-0", !Ref PrivateHostedZoneName]]],
    ]
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-1", !Ref PrivateHostedZoneName]]],
    ]
    - !Join [
      " ",
      ["0 10 2380", !Join ["", ["etcd-2", !Ref PrivateHostedZoneName]]],
    ]
    TTL: 60
    Type: SRV

Etcd0Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["etcd-0", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !GetAtt Master0.PrivateIp
    TTL: 60
    Type: A

Etcd1Record:
  Condition: DoDns
  Type: AWS::Route53::RecordSet
  Properties:
    HostedZoneId: !Ref PrivateHostedZoneId
    Name: !Join ["", ["etcd-1", !Ref PrivateHostedZoneName]]
    ResourceRecords:
    - !GetAtt Master1.PrivateIp
    TTL: 60
    Type: A
```

```
  Etcd2Record:
    Condition: DoDns
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref PrivateHostedZoneId
      Name: !Join [".", ["etcd-2", !Ref PrivateHostedZoneName]]
      ResourceRecords:
      - !GetAtt Master2.PrivateIp
      TTL: 60
      Type: A

Outputs:
  PrivateIPs:
    Description: The control-plane node private IP addresses.
    Value:
      !Join [
        ",",
        [!GetAtt Master0.PrivateIp, !GetAtt Master1.PrivateIp, !GetAtt Master2.PrivateIp]
      ]
```

2.10.14. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS でのブートストラップノードの初期化

Amazon Web Services (AWS) ですべての必要なインフラストラクチャーを作成した後に、クラスターをインストールできます。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを手動で管理する予定の場合には、ワーカーマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

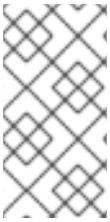
コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2.10.14.1. AWS でのワーカーノードの作成

クラスターで使用するワーカーノードを Amazon Web Services (AWS) で作成できます。これらのノードを手動で作成するための最も簡単な方法として、提供される CloudFormation テンプレートを変更することができます。

重要

CloudFormation テンプレートは、1 つのワーカーマシンを表すスタックを作成します。それぞれのワーカーマシンにスタックを作成する必要があります。



注記

提供される CloudFormation テンプレートを使用してワーカーノードを作成しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- AWS アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- AWS で VPC および関連するサブネットを作成し、設定します。
- AWS で DNS、ロードバランサー、およびリスナーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. CloudFormation テンプレートが必要とするパラメーター値が含まれる JSON ファイルを作成します。

```
[
  {
    "ParameterKey": "InfrastructureName", 1
    "ParameterValue": "mycluster-<random_string>" 2
  },
  {
```

```
  "ParameterKey": "RhcosAmi", 3
  "ParameterValue": "ami-<random_string>" 4
 },
 {
  "ParameterKey": "Subnet", 5
  "ParameterValue": "subnet-<random_string>" 6
 },
 {
  "ParameterKey": "WorkerSecurityGroupId", 7
  "ParameterValue": "sg-<random_string>" 8
 },
 {
  "ParameterKey": "IgnitionLocation", 9
  "ParameterValue": "https://api-int.<cluster_name>.<domain_name>:22623/config/worker"
10
 },
 {
  "ParameterKey": "CertificateAuthorities", 11
  "ParameterValue": "" 12
 },
 {
  "ParameterKey": "WorkerInstanceProfileName", 13
  "ParameterValue": "" 14
 },
 {
  "ParameterKey": "WorkerInstanceType", 15
  "ParameterValue": "m4.large" 16
 }
]
```

1. クラスターの Ingition 設定ファイルでエンコードされるクラスターインフラストラクチャーの名前。
2. 形式が **<cluster-name>-<random-string>** の Ignition 設定ファイルから抽出したインフラストラクチャー名を指定します。
3. ワーカーノードに使用する最新の Red Hat Enterprise Linux CoreOS(RHCOS)AMI。
4. **AWS::EC2::Image::Id** 値を指定します。
5. ワーカーノードを起動するサブネット (プライベートが望ましい)。
6. DNS および負荷分散の CloudFormation テンプレートの出力から **PrivateSubnets** 値のサブネットを指定します。
7. ワーカーノードに関連付けるワーカーセキュリティーグループ ID。
8. セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WorkerSecurityGroupId** 値を指定します。
9. ブートストラップの Ignition 設定ファイルをフェッチする場所。
10. 生成される Ignition 設定の場所を指定します。 **https://api-int.<cluster_name>.<domain_name>:22623/config/worker**

11 使用する base64 でエンコードされた認証局の文字列。

12 インストールディレクトリーにある **worker.ign** ファイルから値を指定します。この値は、**data:text/plain;charset=utf-8;base64,ABC...xYz==** 形式の長い文字列です。

13 ワーカーロールに関連付ける IAM プロファイル。

14 セキュリティーグループおよびロールの CloudFormation テンプレートの出力から **WokerInstanceProfile** パラメーターの値を指定します。

15 コントロールプレーンマシンに使用する AWS インスタンスのタイプ。

16 許可される値:

- **m4.large**
- **m4.xlarge**
- **m4.2xlarge**
- **m4.4xlarge**
- **m4.8xlarge**
- **m4.10xlarge**
- **m4.16xlarge**
- **c4.large**
- **c4.xlarge**
- **c4.2xlarge**
- **c4.4xlarge**
- **c4.8xlarge**
- **r4.large**
- **r4.xlarge**
- **r4.2xlarge**
- **r4.4xlarge**
- **r4.8xlarge**
- **r4.16xlarge**

重要

m4 インスタンスが **eu-west-3** などのリージョンで利用可能ではない場合、**m5** タイプを代わりに使用します。

2. このトピックの**ワーカーマシンの CloudFormation テンプレート**セクションからテンプレート

をコピーし、これをコンピューター上に YAML ファイルとして保存します。このテンプレートは、クラスターに必要なネットワークオブジェクトおよびロードバランサーについて記述しています。

3. **m5** インスタンスタイプを **WorkerInstanceType** の値として指定している場合、そのインスタンスタイプを CloudFormation テンプレートの **WorkerInstanceType.AllowedValues** パラメーターに追加します。

4. ワーカースタックを作成します。

   a. テンプレートを起動します。

重要

単一行にコマンドを入力してください。

```
$ aws cloudformation create-stack --stack-name <name> 1
    --template-body file://<template>.yaml \ 2
    --parameters file://<parameters>.json 3
```

1 **<name>** は **cluster-workers** などの CloudFormation スタックの名前です。クラスターを削除する場合に、このスタックの名前が必要になります。

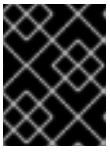
2 **<template>** は、保存した CloudFormation テンプレート YAML ファイルへの相対パスまたはその名前です。

3 **<parameters>** は、CloudFormation パラメーター JSON ファイルへの相対パスまたは名前です。

   b. テンプレートのコンポーネントが存在することを確認します。

```
$ aws cloudformation describe-stacks --stack-name <name>
```

5. クラスターに作成するワーカーマシンが十分な数に達するまでワーカースタックの作成を継続します。



重要

2 つ以上のワーカーマシンを作成する必要があるため、この CloudFormation テンプレートを使用する 2 つ以上のスタックを作成する必要があります。

2.10.14.1.1. ワーカーマシンの CloudFormation テンプレート

以下の CloudFormation テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例2.42 ワーカーマシンの CloudFormation テンプレート

```
AWSTemplateFormatVersion: 2010-09-09
Description: Template for OpenShift Cluster Node Launch (EC2 worker instance)

Parameters:
  InfrastructureName:
```

```
    AllowedPattern: ^([a-zA-Z][a-zA-Z0-9\-]{0,26})$
    MaxLength: 27
    MinLength: 1
    ConstraintDescription: Infrastructure name must be alphanumeric, start with a letter, and have a
maximum of 27 characters.
    Description: A short, unique cluster ID used to tag nodes for the kubelet cloud provider.
    Type: String
  RhcosAmi:
    Description: Current Red Hat Enterprise Linux CoreOS AMI to use for bootstrap.
    Type: AWS::EC2::Image::Id
  Subnet:
    Description: The subnets, recommend private, to launch the master nodes into.
    Type: AWS::EC2::Subnet::Id
  WorkerSecurityGroupId:
    Description: The master security group ID to associate with master nodes.
    Type: AWS::EC2::SecurityGroup::Id
  IgnitionLocation:
    Default: https://api-int.$CLUSTER_NAME.$DOMAIN:22623/config/worker
    Description: Ignition config file location.
    Type: String
  CertificateAuthorities:
    Default: data:text/plain;charset=utf-8;base64,ABC...xYz==
    Description: Base64 encoded certificate authority string to use.
    Type: String
  WorkerInstanceProfileName:
    Description: IAM profile to associate with master nodes.
    Type: String
  WorkerInstanceType:
    Default: m4.large
    Type: String
    AllowedValues:
    - "m4.large"
    - "m4.xlarge"
    - "m4.2xlarge"
    - "m4.4xlarge"
    - "m4.8xlarge"
    - "m4.10xlarge"
    - "m4.16xlarge"
    - "c4.large"
    - "c4.xlarge"
    - "c4.2xlarge"
    - "c4.4xlarge"
    - "c4.8xlarge"
    - "r4.large"
    - "r4.xlarge"
    - "r4.2xlarge"
    - "r4.4xlarge"
    - "r4.8xlarge"
    - "r4.16xlarge"

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
    - Label:
        default: "Cluster Information"
      Parameters:
```

```
      - InfrastructureName
    - Label:
        default: "Host Information"
      Parameters:
      - WorkerInstanceType
      - RhcosAmi
      - IgnitionLocation
      - CertificateAuthorities
      - WorkerSecurityGroupId
      - WorkerInstanceProfileName
    - Label:
        default: "Network Configuration"
      Parameters:
      - Subnet
    ParameterLabels:
      Subnet:
        default: "Subnet"
      InfrastructureName:
        default: "Infrastructure Name"
      WorkerInstanceType:
        default: "Worker Instance Type"
      WorkerInstanceProfileName:
        default: "Worker Instance Profile Name"
      RhcosAmi:
        default: "Red Hat Enterprise Linux CoreOS AMI ID"
      IgnitionLocation:
        default: "Worker Ignition Source"
      CertificateAuthorities:
        default: "Ignition CA String"
      WorkerSecurityGroupId:
        default: "Worker Security Group ID"

Resources:
  Worker0:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref RhcosAmi
      BlockDeviceMappings:
      - DeviceName: /dev/xvda
        Ebs:
          VolumeSize: "120"
          VolumeType: "gp2"
      IamInstanceProfile: !Ref WorkerInstanceProfileName
      InstanceType: !Ref WorkerInstanceType
      NetworkInterfaces:
      - AssociatePublicIpAddress: "false"
        DeviceIndex: "0"
        GroupSet:
        - !Ref "WorkerSecurityGroupId"
        SubnetId: !Ref "Subnet"
      UserData:
        Fn::Base64: !Sub
        - '{"ignition":{"config":{"append":[{"source":"${SOURCE}","verification":{}}]},"security":{"tls":
{"certificateAuthorities":[{"source":"${CA_BUNDLE}","verification":{}}]}},"timeouts":
{},"version":"2.2.0"},"networkd":{},"passwd":{},"storage":{},"systemd":{}}'
        - {
```

```
          SOURCE: !Ref IgnitionLocation,
          CA_BUNDLE: !Ref CertificateAuthorities,
        }
      Tags:
      - Key: !Join ["", ["kubernetes.io/cluster/", !Ref InfrastructureName]]
        Value: "shared"

Outputs:
  PrivateIP:
    Description: The compute node private IP address.
    Value: !GetAtt Worker0.PrivateIp
```

2.10.15. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

2.10.16. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

2.10.17. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

2.10.17.1. イメージレジストリーストレージの設定

Amazon Web Services はデフォルトのストレージを提供します。つまり、Image Registry Operator はインストール後に利用可能になります。ただし、レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、レジストリーストレージを手動で設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

2.10.17.1.1. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS 上のクラスター。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

2.10.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

2.10.18. ブートストラップリソースの削除

クラスターの初期 Operator 設定の完了後に、Amazon Web Services (AWS) からブートストラップリソースを削除します。

前提条件

- クラスターの初期 Operator 設定が完了済みです。

手順

1. ブートストラップリソースを削除します。CloudFormation テンプレートを使用した場合は、そのスタックを削除 します。

```
$ aws cloudformation delete-stack --stack-name <name> 1
```

1 **<name>** は、ブートストラップスタックの名前です。

2.10.19. Ingress DNS レコードの作成

DNS ゾーン設定を削除した場合には、Ingress ロードバランサーを参照する DNS レコードを手動で作成します。ワイルドカードレコードまたは特定のレコードのいずれかを作成できます。以下の手順では A レコードを使用しますが、CNAME やエイリアスなどの必要な他のレコードタイプを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用する OpenShift Container Platform クラスターを Amazon Web Services (AWS) にデプロイしています。
- OpenShift CLI (**oc**) をインストールします。
- **jq** パッケージをインストールします。
- AWS CLI をダウンロードし、これをコンピューターにインストールします。Install the AWS CLI Using the Bundled Installer (Linux, macOS, or Unix) を参照してください。

手順

1. 作成するルートを決定します。
   - ワイルドカードレコードを作成するには、***.apps.<cluster_name>.<domain_name>** を使用します。ここで、**<cluster_name>** はクラスター名で、**<domain_name>** は OpenShift Container Platform クラスターの Route 53 ベースドメインです。
   - 特定のレコードを作成するには、以下のコマンドの出力にあるように、クラスターが使用する各ルートにレコードを作成する必要があります。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.<cluster_name>.<domain_name>
```

```
console-openshift-console.apps.<cluster_name>.<domain_name>
downloads-openshift-console.apps.<cluster_name>.<domain_name>
alertmanager-main-openshift-monitoring.apps.<cluster_name>.<domain_name>
grafana-openshift-monitoring.apps.<cluster_name>.<domain_name>
prometheus-k8s-openshift-monitoring.apps.<cluster_name>.<domain_name>
```

2. Ingress Operator ロードバランサーのステータスを取得し、使用する外部 IP アドレスの値をメモします。これは **EXTERNAL-IP** 列に表示されます。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP                            PORT(S)
AGE
router-default   LoadBalancer   172.30.62.215   ab3...28.us-east-2.elb.amazonaws.com
80:31499/TCP,443:30693/TCP   5m
```

3. ロードバランサーのホストゾーン ID を見つけます。

```
$ aws elb describe-load-balancers | jq -r '.LoadBalancerDescriptions[] | select(.DNSName ==
"<external_ip>").CanonicalHostedZoneNameID' 1
```

1 **<external_ip>** については、取得した Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。

出力例

```
Z3AADJGX6KTTL2
```

このコマンドの出力は、ロードバランサーのホストゾーン ID です。

4. クラスターのドメインのパブリックホストゾーン ID を取得します。

```
$ aws route53 list-hosted-zones-by-name \
        --dns-name "<domain_name>" \ 1
        --query 'HostedZones[? Config.PrivateZone != `true` && Name ==
`<domain_name>.`].Id' 2
        --output text
```

1 2 **<domain_name>** については、OpenShift Container Platform クラスターの Route 53 ベースドメインを指定します。

出力例

```
/hostedzone/Z3URY6TWQ91KVV
```

ドメインのパブリックホストゾーン ID がコマンド出力に表示されます。この例では、これは **Z3URY6TWQ91KVV** になります。

5. プライベートゾーンにエイリアスレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<private_hosted_zone_id>" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) **<private_hosted_zone_id>** については、DNS および負荷分散の CloudFormation テンプレートの出力から値を指定します。

(2) **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

(3) **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

(4) **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

6. パブリックゾーンにレコードを追加します。

```
$ aws route53 change-resource-record-sets --hosted-zone-id "<public_hosted_zone_id>"" --
change-batch '{ (1)
>   "Changes": [
>     {
>       "Action": "CREATE",
>       "ResourceRecordSet": {
>         "Name": "\\052.apps.<cluster_domain>", (2)
>         "Type": "A",
>         "AliasTarget":{
>           "HostedZoneId": "<hosted_zone_id>", (3)
>           "DNSName": "<external_ip>.", (4)
>           "EvaluateTargetHealth": false
>         }
>       }
>     }
>   ]
> }'
```

(1) **<public_hosted_zone_id>** については、ドメインのパブリックホストゾーンを指定します。

2 **<cluster_domain>** については、OpenShift Container Platform クラスターで使用するドメインまたはサブドメインを指定します。

3 **<hosted_zone_id>** については、取得したロードバランサーのパブリックホストゾーン ID を指定します。

4 **<external_ip>** については、Ingress Operator ロードバランサーの外部 IP アドレスの値を指定します。このパラメーターの値に末尾のピリオド (.) が含まれていることを確認します。

2.10.20. ユーザーによってプロビジョニングされるインフラストラクチャーでの AWS インストールの実行

Amazon Web Service (AWS) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後に、デプロイメントを完了するまでモニターします。

前提条件

- OpenShift Container Platform クラスターのブートストラップノードを、ユーザーによってプロビジョニングされた AWS インフラストラクチャーで削除しています。
- **oc** CLI をインストールし、ログインします。

手順

1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Cluster registration ページでクラスターを登録します。

2.10.21. 次のステップ

- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- 必要に応じて、クラウドプロバイダーの認証情報を削除 できます。

2.11. AWS でのクラスターのアンインストール

Amazon Web Services (AWS) にデプロイしたクラスターは削除することができます。

2.11.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第3章 AZURE へのインストール

3.1. AZURE アカウントの設定

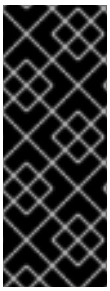
OpenShift Container Platform をインストールする前に、Microsoft Azure アカウントを設定する必要があります。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3.1.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。



重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	40	リージョンごとに 20	デフォルトのクラスターには 40 の vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップマシンは 4 vCPUS を使用する **Standard_D4s_v3** マシンを使用し、コントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、さらにワーカーマシンは、4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、デフォルトクラスターには 40 の vCPU が必要になります。4 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。 デフォルトで、インストールプログラムはコントロールプレーンおよびコンピュートマシンを、リージョン内の すべてのアベイラビリティーゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	6	リージョンごとに 65,536	各デフォルトクラスターには、6 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークセキュリティーグループ	2	5000	各デフォルトクラスター。各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。

3.1.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

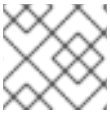
2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

3.1.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。



注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

a. **Issue type** 一覧から、**Service and subscription limits (quotas)** を選択します。

b. **Subscription** 一覧から、変更するサブスクリプションを選択します。

c. **Quota type** 一覧から、引き上げるクォータを選択します。たとえば、**Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。

a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。

b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。

4. **Next: Review + create** をクリックしてから **Create** をクリックします。

3.1.4. 必要な Azure ロール

Microsoft Azure アカウントには、使用するサブスクリプションについて以下のロールが必要です。

- **User Access Administrator**

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

3.1.5. サービスプリンシパルの作成

OpenShift Container Platform およびそのインストールプログラムは Azure Resource Manager 経由で Microsoft Azure リソースを作成する必要があるため、これを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- **jq** パッケージをインストールします。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. Azure CLI にログインします。

```
$ az login
```

認証情報を使用して Web コンソールで Azure にログインします。

2. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が適切なサブスクリプションの UUID であることを確認します。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <id> (1)
```

(1) 使用する必要のあるサブスクリプションの **id** の値を **<id>** の代わりに使用します。

d. アクティブなサブスクリプションを変更したら、アカウント情報を再度表示します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 直前の出力の **tenantId** および **id** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

4. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

1 **<service_principal>** を、サービスプリンシパルに割り当てる名前に置き換えます。

出力例

```
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}
```

5. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

6. サービスプリンシパルに追加パーミッションを付与します。サービスプリンシパルには、クラスターでそのコンポーネントの認証情報を割り当てられるようにレガシーの **Azure Active Directory Graph** → **Application.ReadWrite.OwnedBy** パーミッションおよび **User Access Administrator** ロールが必要です。

   a. **User Access Administrator** ロールを割り当てるには、以下のコマンドを実行します。

```
$ az role assignment create --role "User Access Administrator" \
    --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \ 1
      | jq '.[0].objectId' -r)
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

b. **Azure Active Directory Graph** パーミッションを割り当てるには、以下のコマンドを実行します。

```
$ az ad app permission add --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000 \
    --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

出力例

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

このコマンドで付与する特定のパーミッションについての詳細は、GUID Table for Windows Azure Active Directory Permissions を参照してください。

c. パーミッション要求を承認します。アカウントに Azure Active Directory テナント管理者ロールがない場合は、所属する組織のガイドラインに従い、テナント管理者にパーミッション要求を承認するようにリクエストしてください。

```
$ az ad app permission grant --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

3.1.6. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンの一覧を動的に生成します。以下の Azure リージョンは OpenShift Container Platform バージョン 4.5.4 でテストされ、検証されています。

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)

- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

3.1.7. 次のステップ

- OpenShift Container Platform クラスターを Azure にインストールします。カスタマイズされたクラスターのインストール、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

3.2. AZURE の IAM の手動作成

3.2.1. IAM の手動作成

Cloud Credential Operator は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. OpenShift Container Platform インストーラーを実行し、マニフェストを生成します。

```
$ openshift-install create manifests --dir=mycluster
```

2. Cloud Credential Operator が手動モードになるように、設定マップを manifests ディレクトリーに挿入します。

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. ローカルクラウドの認証情報を使用して作成された **admin** 認証情報シークレットを削除します。この削除により、**admin** 認証情報がクラスターに保存されなくなります。

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. OpenShift Container Platform リリースイメージを取得します。**openshift-install** バイナリーはこれを使用するためにビルドされます。

```
$ bin/openshift-install version
```

出力例

```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```

6. 展開したファイルで **CredentialsRequests** を見つけます。

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```

注記

今後の OpenShift Container Platform リリースでは、**CredentialsRequests** をスキャンし、それらを表示する新規の **oc adm release** コマンドが表示されます。

これにより、それぞれの要求の詳細が表示されます。**spec.providerSpec.kind** がインストールするクラウドプロバイダーと一致しない **CredentialsRequests** については、これを無視するようにしてください。

サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-azure
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: AzureProviderSpec
    roleBindings:
    - role: Contributor
```

7. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、各 **request.spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

8. クラスターの作成に進みます。

```
$ openshift-install create cluster --dir=mycluster
```

重要

アップグレードを実行する前に、パーミッションが次のリリースで変更された場合には、認証情報の調整が必要になる場合があります。今後は、Cloud Credential Operator は更新されたパーミッションが処理されることを示唆するまでアップグレードできなくなる可能性があります。

3.2.2. 管理者の認証情報のルートシークレット形式

各クラウドプロバイダーは、**kube-system** namespace の認証情報ルートシークレットを使用します。これは、すべての認証情報要求を満たし、それぞれのシークレットを作成するために使用されます。これは、**Mint モード** で新規の認証情報を作成するか、または **Passthrough モード** で認証情報ルートシークレットをコピーして実行します。

シークレットの形式はクラウドごとに異なり、それぞれの **CredentialsRequest** シークレットにも使用されます。

Microsoft Azure シークレットの形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: azure-credentials
stringData:
  azure_subscription_id: <SubscriptionID>
  azure_client_id: <ClientID>
  azure_client_secret: <ClientSecret>
  azure_tenant_id: <TenantID>
  azure_resource_prefix: <ResourcePrefix>
  azure_resourcegroup: <ResourceGroup>
  azure_region: <Region>
```

Microsoft Azure では、認証情報シークレット形式には、それぞれのクラスターのインストールにランダムに生成されるクラスターのインフラストラクチャー ID が含まれる必要のある 2 つのプロパティーがあります。この値は、マニフェストの作成後に確認できます。

```
$ cat .openshift_install_state.json | jq '."*installconfig.ClusterID".InfraID' -r
```

出力例

```
mycluster-2mpcn
```

この値は、以下のようにシークレットデータで使用されます。

```
azure_resource_prefix: mycluster-2mpcn
azure_resourcegroup: mycluster-2mpcn-rg
```

3.2.2.1. アップグレード

今後のリリースでは、Cloud Credential Operator の改善により、手動でメンテナーンスされる認証情報が今後のリリースのイメージの **CredentialsRequest** オブジェクトに一致するように更新されていないことが原因でアップグレードが失敗する可能性が生じる状況を避けることができます。

3.2.3. mint モード

mint モードは AWS、GCP および Azure でサポートされます。

OpenShift Container Platform を実行するためのデフォルトおよび推奨されるベストプラクティスとして、管理者レベルのクラウド認証情報を使用してインストーラーを実行できます。**admin** 認証情報は **kube-system** namespace に保存され、次に Cloud Credential Operator によってクラスターの **CredentialsRequest** オブジェクトを処理し、特定のパーミッションでそれぞれの新規ユーザーを作成するために使用されます。

mint モードには以下の利点があります。

- 各クラスターコンポーネントにはそれぞれが必要なパーミッションのみがあります。
- アップグレードを含むクラウド認証情報の自動の継続的な調整が行われます。これには、追加の認証情報またはパーミッションが必要になる可能性があります。

1 つの不利な点として、mint モードでは、**admin** 認証情報がクラスターの **kube-system** シークレットに保存される必要があります。

3.3. クラスターの AZURE へのクイックインストール

OpenShift Container Platform バージョン 4.5 では、デフォルトの設定オプションを使用してクラスターを Microsoft Azure にインストールできます。

3.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- クラスターをホストできるように Azure アカウントを設定 し、クラスターをデプロイするテスト済みの検証されたリージョンを判別します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

3.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

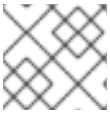
- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.3.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

3.3.4. インストールプログラムの取得

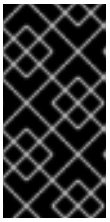
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

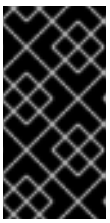
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.3.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

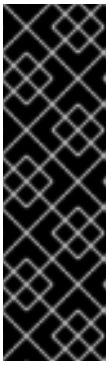
手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

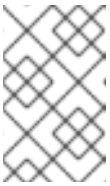


重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **azure** を選択します。

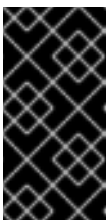
c. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

d. クラスターをデプロイするリージョンを選択します。

e. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

f. クラスターの記述名を入力します。



重要

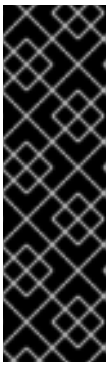
パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

g. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

注記

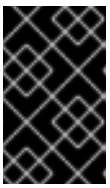
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3.3.6. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.3.6.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.3.6.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.3.6.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。

PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.3.7. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

3.3.8. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

3.4. カスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストールプログラムが Microsoft Azure にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

3.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- クラスターをホストできるように Azure アカウントを設定 し、クラスターをデプロイするテスト済みの検証されたリージョンを判別します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

3.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

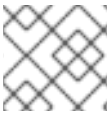
- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.4.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
-
```

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

3.4.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシー

クレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.4.5. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

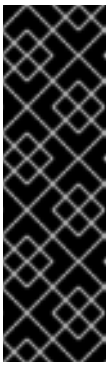
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

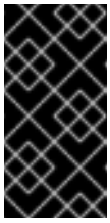
      ii. ターゲットに設定するプラットフォームとして **azure** を選択します。
      iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。



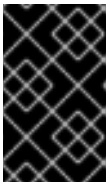
重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

3.4.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

3.4.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表3.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

3.4.5.1.2. ネットワーク設定パラメーター

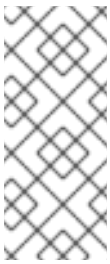
既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表3.2 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

3.4.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表3.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

3.4.5.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表3.4 追加の Azure パラメーター

パラメーター	説明	値
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。サポートされる最小のディスクサイズは **1024** です。

パラメーター	説明	値
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

3.4.5.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
```

```
controlPlane: 2
  hyperthreading: Enabled 3 4
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 5
      type: Standard_D8s_v3
  replicas: 3
compute: 6
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus 11
    baseDomainResourceGroupName: resource_group 12
pullSecret: '{"auths": ...}' 13
ifndef::openshift-origin
fips: false 14
sshKey: ssh-ed25519 AAAA... 15
endif::openshift-origin
ifdef::openshift-origin
sshKey: ssh-ed25519 AAAA... 16
endif::openshift-origin
```

1 10 11 13 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができませ

ん。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。マスターノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

12 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

14 16 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

15 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

3.4.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

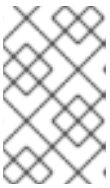
手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



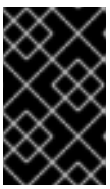
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

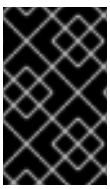


重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3.4.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.4.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.4.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.4.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.4.8. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

3.4.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

3.5. ネットワークのカスタマイズによる AZURE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストールプログラムが Microsoft Azure にプロビジョニングするインフラストラクチャーにカスタマイズされたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

3.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- クラスターをホストできるように Azure アカウントを設定 し、クラスターをデプロイするテスト済みの検証されたリージョンを判別します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

3.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.5.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

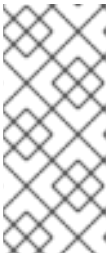
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

3.5.4. インストールプログラムの取得

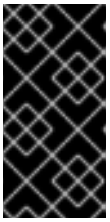
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.5.5. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

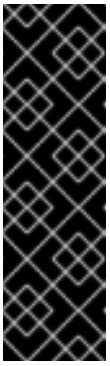
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

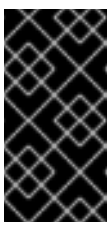
iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。



重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

3.5.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

3.5.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表3.5 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

3.5.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表3.6 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

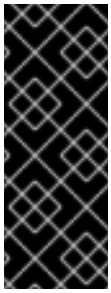
3.5.5.1.3. オプションの設定パラメーター

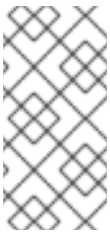
オプションのインストール設定パラメーターは、以下の表で説明されています。

表3.7 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

3.5.5.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表3.8 追加の Azure パラメーター

パラメーター	説明	値
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。サポートされる最小のディスクサイズは **1024** です。
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。

パラメーター	説明	値
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。

注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

重要

Open Virtual Networking (OVN) Kubernetes ネットワークプラグインは、テクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

OVN テクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/articles/4380121 を参照してください。

3.5.5.2. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。

注記

インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

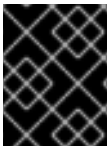
表3.9 必要なネットワークパラメーター

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。**OVNKubernetes** プラグインは、OpenShift Container Platform 4.5 でテクノロジープレビューとしてご利用いただけます。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

パラメーター	説明	値
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの **serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

3.5.5.3. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
```

```
- hyperthreading: Enabled 7
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 8
      zones: 9
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster 10
networking: 11
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus 12
    baseDomainResourceGroupName: resource_group 13
pullSecret: '{"auths": ...}' 14
ifndef::openshift-origin
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
endif::openshift-origin
ifdef::openshift-origin
sshKey: ssh-ed25519 AAAA... 17
endif::openshift-origin
```

1 10 12 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 11 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

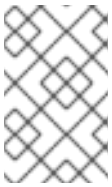
5 8 使用するディスクのサイズは、GB 単位で指定できます。マスターノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

13 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

15 17 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

3.5.6. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

3.5.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
    ...
  kubeProxyConfig: 4
    iptablesSyncPeriod: 30s 5
    proxyArguments:
      iptables-min-sync-period: 6
      - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。

注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time で説明されています。

3.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN (1)
  openshiftSDNConfig: (2)
    mode: NetworkPolicy (3)
    mtu: 1450 (4)
    vxlanPort: 4789 (5)
```

(1) **install-config.yaml** ファイルに指定されます。

(2) OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

(3) OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

(4) VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

(5) すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

3.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OVNKubernetes (1)
  ovnKubernetesConfig: (2)
    mtu: 1400 (3)
    genevePort: 6081 (4)
```

(1) **install-config.yaml** ファイルに指定されます。

(2) OVN-Kubernetes 設定の一部を上書きする必要がある場合にのみ指定します。

3 Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。

4 Geneve オーバーレイネットワークの UDP ポート。

3.5.7.3. Cluster Network Operator の設定例

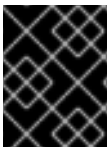
以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
      - 0s
```

3.5.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

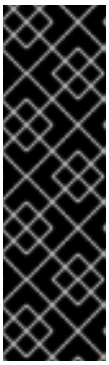
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3.5.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.5.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.5.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.5.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.5.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

3.5.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

3.6. AZURE のクラスターの既存 VNET へのインストール

OpenShift Container Platform バージョン 4.5 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

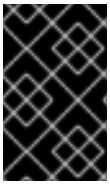
3.6.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- クラスターをホストできるように Azure アカウントを設定 し、クラスターをデプロイするテスト済みの検証されたリージョンを判別します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

3.6.2. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.5 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。



重要

既存の VNet を使用するには、更新された Azure Private DNS (プレビュー) 機能を使用する必要があります。この機能の制限についての詳細は、Announcing Preview Refresh for Azure DNS Private Zones を参照してください。

3.6.2.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。
- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットを指定する必要があります。
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

3.6.2.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。

重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表3.10 必須ポート

ポート	説明	コントロールプレーン	コンピュート
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシン設定サーバーとの通信を許可します。	x	

注記

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

3.6.2.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

3.6.2.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

3.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.6.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 **~/.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

3.6.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.6.6. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

      iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

      - **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
      - **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
      - **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
      - **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

      iv. クラスターをデプロイするリージョンを選択します。

      v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

      vi. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

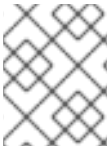
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

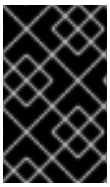
3.6.6.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

3.6.6.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表3.11 必須パラメーター

パラメーター	説明	値

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

3.6.6.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表3.12 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

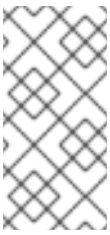
3.6.6.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表3.13 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

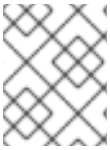
3.6.6.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表3.14 追加の Azure パラメーター

パラメーター	説明	値
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。サポートされる最小のディスクサイズは **1024** です。
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。

パラメーター	説明	値
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。



注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

3.6.6.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus 11
    baseDomainResourceGroupName: resource_group 12
    networkResourceGroupName: vnet_resource_group 13
    virtualNetwork: vnet 14
    controlPlaneSubnet: control_plane_subnet 15
    computeSubnet: compute_subnet 16
pullSecret: '{"auths": ...}' 17
fips: false 18
sshKey: ssh-ed25519 AAAA... 19
```

1 10 11 17 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 6 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 7 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。マスターノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

12 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

13 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

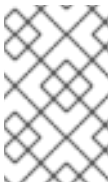
14 既存の VNet を使用する場合は、その名前を指定します。

15 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

16 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

18 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

19 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

3.6.6.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
```

```
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。

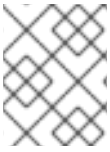
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

3.6.7. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

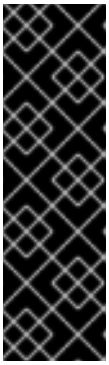
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

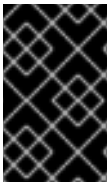
インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3.6.8. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.6.8.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.6.8.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.6.8.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.6.9. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。

- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

3.6.10. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

3.7. プライベートクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.5 では、プライベートクラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

3.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- クラスターをホストできるように Azure アカウントを設定 し、クラスターをデプロイするテスト済みの検証されたリージョンを判別します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

3.7.2. プライベートクラスター

お使いの環境で外部のインターネット接続を必要としない場合には、外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセス可能で、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

3.7.2.1. Azure のプライベートクラスター

Microsoft Azure でプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VNet とサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

ネットワークがプライベート VNET に接続される方法によって、クラスターのプライベート DNS レコードを解決するために DNS フォワーダーを使用する必要がある場合があります。クラスターのマシンは、DNS 解決に **168.63.129.16** を内部で使用します。詳細は、Azure ドキュメントの What is Azure Private DNS? および What is IP address 168.63.129.16? を参照してください。

クラスターには、Azure API にアクセスするためにインターネットへのアクセスが依然として必要です。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- **BaseDomainResourceGroup** (クラスターがパブリックレコードを作成しないため)
- パブリック IP アドレス
- パブリック DNS レコード
- パブリックエンドポイント

> The cluster is configured so that the Operators do not create public records for the cluster and all cluster machines are placed in the private subnets that you specify.

3.7.2.1.1. 制限事項

Azure 上のプライベートクラスターは、既存の VNet の使用に関連する制限のみの制限を受けます。

3.7.3. OpenShift Container Platform クラスターでの VNet の再利用について

OpenShift Container Platform 4.5 では、クラスターを Microsoft Azure の既存の Azure Virtual Network (VNet) にデプロイできます。これを実行する場合、VNet 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の Azure VNet にデプロイすることで、新規アカウントでのサービス制限の制約を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VNet の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

重要

既存の VNet を使用するには、更新された Azure Private DNS (プレビュー) 機能を使用する必要があります。この機能の制限についての詳細は、Announcing Preview Refresh for Azure DNS Private Zones を参照してください。

3.7.3.1. VNet を使用するための要件

既存の VNet を使用してクラスターをデプロイする場合、クラスターをインストールする前に追加のネットワーク設定を実行する必要があります。インストーラーでプロビジョニングされるインフラストラクチャークラスターでは、インストーラーは通常以下のコンポーネントを作成しますが、既存の VNet にインストールする場合にはこれらを作成しません。

- サブネット
- ルートテーブル
- VNets
- ネットワークセキュリティーグループ

カスタム VNet を使用する場合、インストールプログラムおよびクラスターで使用できるようにカスタム VNet およびそのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VNet オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

クラスターは、既存の VNet およびサブネットを含むリソースグループにアクセスできる必要があります。クラスターが作成するすべてのリソースは、作成される別個のリソースグループに配置され、一部のネットワークリソースが別個のグループから使用されます。一部のクラスター Operator は両方のリソースグループのリソースにアクセスできる必要があります。たとえば マシン API コントローラーは、ネットワークリソースグループから、作成される仮想マシンの NIC をサブネットに割り当てます。

VNet には以下の特徴が確認される必要があります。

- VNet の CIDR ブロックには、クラスターマシンの IP アドレスプールである **Networking.MachineCIDR** 範囲が含まれる必要があります。
- VNet およびそのサブネットは同じリソースグループに属する必要があり、サブネットは静的 IP アドレスではなく、Azure で割り当てられた DHCP IP アドレスを使用するように設定される必要があります。

コントロールプレーンマシンのサブネットおよびコンピュートマシン用のサブネットの 2 つのサブネットを VNet 内に指定する必要があります。Azure はマシンを指定するリージョン内の複数の異なるアベイラビリティーゾーンに分散するため、デフォルトのクラスターには高可用性があります。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定したサブネットすべてが存在します。

- コントロールプレーンマシンのサブネットおよびコンピュートマシンのサブネットの 2 つのサブネットを指定する必要があります。
- サブネットの CIDR は指定されたマシン CIDR に属します。マシンは、プライベートサブネットを指定しないアベイラビリティーゾーンにはプロビジョニングされません。必要な場合に、インストールプログラムはコントロールプレーンおよびワーカーノードを管理するパブリックロードバランサーを作成し、Azure はパブリック IP アドレスをそれらに割り当てます。

既存の VNet を使用するクラスターを破棄しても、VNet は削除されません。

3.7.3.1.1. ネットワークセキュリティーグループの要件

コンピュートマシンおよびコントロールプレーンマシンをホストするサブネットのネットワークセキュリティーグループには、クラスターの通信が正しいことを確認するための特定のアクセスが必要です。必要なクラスター通信ポートへのアクセスを許可するルールを作成する必要があります。

重要

ネットワークセキュリティーグループルールは、クラスターのインストール前に有効にされている必要があります。必要なアクセスなしにクラスターのインストールを試行しても、インストールプログラムは Azure API に到達できず、インストールに失敗します。

表3.15 必須ポート

ポート	説明	コントロールプレーン	コンピュート
80	HTTP トラフィックを許可します。		x
443	HTTPS トラフィックを許可します		x
6443	コントロールプレーンマシンとの通信を許可します。	x	
22623	マシン設定サーバーとの通信を許可します。	x	

注記

クラスターコンポーネントは、Kubernetes コントローラーが更新する、ユーザーによって提供されるネットワークセキュリティーグループを変更しないため、擬似セキュリティーグループが環境の残りの部分に影響を及ぼさずに Kubernetes コントローラー用に作成されます。

3.7.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、ストレージ、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VNet、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する Azure の認証情報には、VNet、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VNet 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージアカウントおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

3.7.3.3. クラスター間の分離

クラスターは既存のサブネットのネットワークセキュリティーグループを変更できないため、VNet でクラスターを相互に分離する方法はありません。

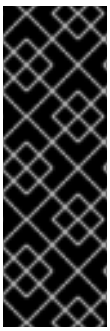
3.7.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

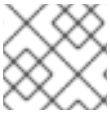


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.7.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

3.7.6. インストールプログラムの取得

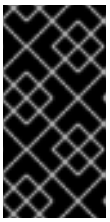
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシー

クレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.7.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

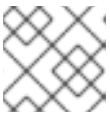
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

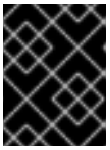
2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



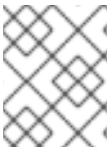
重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

3.7.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際

に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

3.7.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表3.16 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

3.7.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表3.17 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

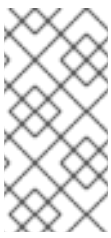
3.7.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表3.18 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

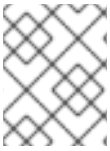
3.7.7.1.4. 追加の Azure 設定パラメーター

追加の Azure 設定パラメーターは以下の表で説明されています。

表3.19 追加の Azure パラメーター

パラメーター	説明	値
controlPlane.platform.azure.osDisk.diskSizeGB	VM の Azure ディスクのサイズ。	GB 単位でディスクのサイズを表す整数。サポートされる最小のディスクサイズは **1024** です。

パラメーター	説明	値
platform.azure.baseDomainResourceGroupName	ベースドメインの DNS ゾーンが含まれるリソースグループの名前。	文字列 (例: **production_cluster**)。
platform.azure.region	クラスターをホストする Azure リージョンの名前。	**centralus** などの有効なリージョン名。
platform.azure.zone	マシンを配置するアベイラビリティーゾーンの一覧。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。	ゾーンの一覧 (例: **["1", "2", "3"]**)。
platform.azure.networkResourceGroupName	クラスターをデプロイする既存の VNet を含むリソースグループの名前。この名前は **platform.azure.baseDomainResourceGroupName** と同じにすることはできません。	文字列。
platform.azure.virtualNetwork	クラスターをデプロイする既存 VNet の名前。	文字列。
platform.azure.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。
platform.azure.computeSubnet	コンピュートマシンをデプロイする VNet 内の既存サブネットの名前。	有効な CIDR (例: **10.0.0.0/16**)。

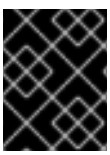


注記

Azure クラスターで、Azure アベイラビリティーゾーン のカスタマイズや タグを使用した Azure リソースの編成 を実行することはできません。

3.7.7.2. Azure のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
```

```
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    azure:
      osDisk:
        diskSizeGB: 1024 (5)
      type: Standard_D8s_v3
  replicas: 3
compute: (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    azure:
      type: Standard_D2s_v3
      osDisk:
        diskSizeGB: 512 (8)
      zones: (9)
      - "1"
      - "2"
      - "3"
  replicas: 5
metadata:
  name: test-cluster (10)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  azure:
    region: centralus (11)
    baseDomainResourceGroupName: resource_group (12)
    networkResourceGroupName: vnet_resource_group (13)
    virtualNetwork: vnet (14)
    controlPlaneSubnet: control_plane_subnet (15)
    computeSubnet: compute_subnet (16)
pullSecret: '{"auths": ...}' (17)
fips: false (18)
sshKey: ssh-ed25519 AAAA... (19)
publish: Internal (20)
```

(1) (10) (11) (17) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (6) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (7) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができませ

ん。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **Standard_D8s_v3** などの大規模な仮想マシンタイプを使用します。

5 8 使用するディスクのサイズは、GB 単位で指定できます。マスターノードの最小推奨値は 1024 GB です。

9 マシンをデプロイするゾーンの一覧を指定します。高可用性を確保するには、少なくとも 2 つのゾーンを指定します。

12 ベースドメインの DNS ゾーンが含まれるリソースグループの名前を指定します。

13 既存の VNet を使用する場合は、それが含まれるリソースグループの名前を指定します。

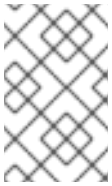
14 既存の VNet を使用する場合は、その名前を指定します。

15 既存の VNet を使用する場合は、コントロールプレーンマシンをホストするサブネットの名前を指定します。

16 既存の VNet を使用する場合は、コンピュートマシンをホストするサブネットの名前を指定します。

18 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

19 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

20 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

3.7.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

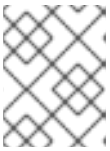
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しませ

ん。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

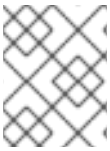


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

3.7.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

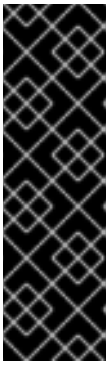
```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

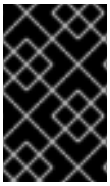
インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3.7.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.7.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。

3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.7.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.7.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。

3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.7.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

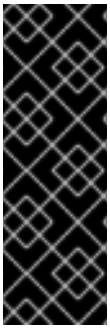
3.7.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

3.8. ARM テンプレートを使用したクラスターの AZURE へのインストール

OpenShift Container Platform バージョン 4.5 では、独自にプロビジョニングするインフラストラクチャーを使用して、クラスターを Microsoft Azure にインストールできます。

これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Azure Resource Manager (ARM) テンプレートが提供されます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の ARM テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

3.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- Azure アカウントを設定 してクラスターをホストします。
- AzureS CLI をダウンロードし、これをコンピューターにインストールします。Azure ドキュメントの Install the Azure CLI を参照してください。以下のドキュメントについては、直近で Azure CLI のバージョン **2.2.0** を使用してテストされていますAzure CLI コマンドは、使用するバージョンによって動作が異なる場合があります。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

3.8.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

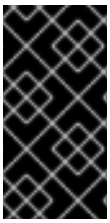
- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

3.8.3. Azure プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするために Azure プロジェクトを設定する必要があります。

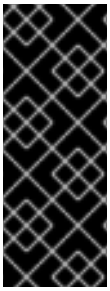


重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

3.8.3.1. Azure アカウントの制限

OpenShift Container Platform クラスターは数多くの Microsoft Azure コンポーネントを使用し、デフォルトの Azure サブスクリプションおよびサービス制限、クォータ、および制約 は、OpenShift Container Platform クラスターをインストールする機能に影響を与えます。



重要

デフォルトの制限は、Free Trial や Pay-As-You-Go、および DV2、F、および G などのシリーズといったカテゴリータイプによって異なります。たとえば、Enterprise Agreement サブスクリプションのデフォルトは 350 コアです。

サブスクリプションタイプの制限を確認し、必要に応じて、デフォルトのクラスターを Azure にインストールする前にアカウントのクォータ制限を引き上げます。

以下の表は、OpenShift Container Platform クラスターのインストールおよび実行機能に影響を与える可能性のある Azure コンポーネントの制限を要約しています。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
vCPU	40	リージョンごとに 20	デフォルトのクラスターには 40 の vCPU が必要であるため、アカウントの上限を引き上げる必要があります。 デフォルトで、各クラスターは以下のインスタンスを作成します。 ● 1 つのブートストラップマシン。これはインストール後に削除されます。 ● 3 つのコントロールプレーンマシン ● 3 つのコンピュートマシン ブートストラップマシンは 4 vCPUS を使用する **Standard_D4s_v3** マシンを使用し、コントロールプレーンマシンは 8 vCPU を使用する **Standard_D8s_v3** 仮想マシンを使用し、さらにワーカーマシンは、4 vCPU を使用する **Standard_D4s_v3** 仮想マシンを使用するため、デフォルトクラスターには 40 の vCPU が必要になります。4 vCPU を使用するブートストラップノードの仮想マシンは、インストール時にのみ使用されます。 追加のワーカーノードをデプロイし、自動スケーリングを有効にし、大規模なワークロードをデプロイするか、または異なるインスタンスタイプを使用するには、アカウントの vCPU 制限をさらに引き上げ、クラスターが必要なマシンをデプロイできるようにする必要があります。 デフォルトで、インストールプログラムはコントロールプレーンおよびコンピュートマシンを、リージョン内の すべてのアベイラビリティーゾーン に分散します。クラスターの高可用性を確保するには、少なくとも 3 つ以上のアベイラビリティーゾーンのあるリージョンを選択します。リージョンに含まれるアベイラビリティーゾーンが 3 つ未満の場合、インストールプログラムは複数のコントロールプレーンマシンを利用可能なゾーンに配置します。
VNet	1	リージョンごとに 1000	各デフォルトクラスターには、2 つのサブネットを含む 1 つの Virtual Network (VNet) が必要です。
ネットワークインターフェイス	6	リージョンごとに 65,536	各デフォルトクラスターには、6 つのネットワークインターフェイスが必要です。さらに多くのマシンを作成したり、デプロイしたワークロードでロードバランサーを作成する場合、クラスターは追加のネットワークインターフェイスを使用します。

コンポーネント	デフォルトで必要なコンポーネントの数	デフォルトの Azure 制限	説明
ネットワークセキュリティーグループ	2	5000	各デフォルトクラスター。各クラスターは VNet の各サブネットにネットワークセキュリティーグループを作成します。デフォルトのクラスターは、コントロールプレーンおよびコンピュートノードのサブネットにネットワークセキュリティーグループを作成します。 **controlplane**: 任意の場所からコントロールプレーンマシンにポート 6443 でアクセスできるようにします。 **node**: インターネットからワーカーノードにポート 80 および 443 でアクセスできるようにします。
ネットワークロードバランサー	3	リージョンごとに 1000	各クラスターは以下の ロードバランサー を作成します。 **default**: ワーカーマシン間でポート 80 および 443 での要求の負荷分散を行うパブリック IP アドレス **internal**: コントロールプレーンマシン間でポート 6443 および 22623 での要求の負荷分散を行うプライベート IP アドレス **external**: コントロールプレーンマシン間でポート 6443 での要求の負荷分散を行うパブリック IP アドレス アプリケーションが追加の Kubernetes **LoadBalancer** サービスオブジェクトを作成すると、クラスターは追加のロードバランサーを使用します。
パブリック IP アドレス	3		2 つのパブリックロードバランサーのそれぞれはパブリック IP アドレスを使用します。ブートストラップマシンは、インストール時のトラブルシューティングのためにマシンに SSH を実行できるようにパブリック IP アドレスも使用します。ブートストラップノードの IP アドレスは、インストール時にのみ使用されます。
プライベート IP アドレス	7		内部ロードバランサー、3 つのコントロールプレーンマシンのそれぞれ、および 3 つのワーカーマシンのそれぞれはプライベート IP アドレスを使用します。

3.8.3.2. Azure でのパブリック DNS ゾーンの設定

OpenShift Container Platform をインストールするには、使用する Microsoft Azure アカウントに、専用のパブリックホスト DNS ゾーンが必要になります。このゾーンはドメインに対する権威を持っている必要があります。このサービスは、クラスターへの外部接続のためのクラスター DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、Azure または別のソースから新規のものを取得できます。

注記

Azure 経由でドメインを購入する方法についての詳細は、Azure ドキュメントの Buy a custom domain name for Azure App Service を参照してください。

2. 既存のドメインおよびレジストラーを使用している場合、その DNS を Azure に移行します。Azure ドキュメントの Migrate an active DNS name to Azure App Service を参照してください。

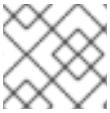
3. ドメインの DNS を設定します。Azure ドキュメントの Tutorial: Host your domain in Azure DNS の手順に従い、ドメインまたはサブドメインのパブリックホストゾーンを作成し、 新規の権威ネームサーバーを抽出し、ドメインが使用するネームサーバーのレジストラーレコードを更新します。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

4. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。

この DNS ゾーンの作成例 を参照し、Azure の DNS ソリューションを確認することができます。

3.8.3.3. Azure アカウント制限の拡張

アカウントの制限を引き上げるには、Azure ポータルでサポートをリクエストします。



注記

サポートリクエストごとに 1 つの種類のクォータのみを増やすことができます。

手順

1. Azure ポータルの左端で **Help + support** をクリックします。

2. **New support request** をクリックしてから必要な値を選択します。

   a. **Issue type** 一覧から、 **Service and subscription limits (quotas)** を選択します。

   b. **Subscription** 一覧から、変更するサブスクリプションを選択します。

   c. **Quota type** 一覧から、引き上げるクォータを選択します。たとえば、 **Compute-VM (cores-vCPUs) subscription limit increases** を選択し、クラスターのインストールに必要な vCPU の数を増やします。

d. **Next: Solutions** をクリックします。

3. **Problem Details**ページで、クォータの引き上げについての必要な情報を指定します。

a. **Provide details**をクリックし、**Quota details**ウィンドウに必要な詳細情報を指定します。

b. SUPPORT METHOD and CONTACT INFO セクションに、問題の重大度および問い合わせ先の詳細を指定します。

4. **Next: Review + create** をクリックしてから **Create** をクリックします。

3.8.3.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

3.8.3.5. 必要な Azure ロール

Microsoft Azure アカウントには、使用するサブスクリプションについて以下のロールが必要です。

- **User Access Administrator**

Azure ポータルでロールを設定するには、Azure ドキュメントの Manage access to Azure resources using RBAC and the Azure portal を参照します。

3.8.3.6. サービスプリンシパルの作成

OpenShift Container Platform およびそのインストールプログラムは Azure Resource Manager 経由で Microsoft Azure リソースを作成する必要があるため、これを表すサービスプリンシパルを作成する必要があります。

前提条件

- Azure CLI のインストールまたは更新を実行します。
- **jq** パッケージをインストールします。
- Azure アカウントには、使用するサブスクリプションに必要なロールがなければなりません。

手順

1. Azure CLI にログインします。

```
$ az login
```

認証情報を使用して Web コンソールで Azure にログインします。

2. Azure アカウントでサブスクリプションを使用している場合は、適切なサブスクリプションを使用していることを確認してください。

a. 利用可能なアカウントの一覧を表示し、クラスターに使用するサブスクリプションの **tenantId** の値を記録します。

```
$ az account list --refresh
```

出力例

```
[
  {
    "cloudName": "AzureCloud",
    "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
    "isDefault": true,
    "name": "Subscription Name",
    "state": "Enabled",
    "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee",
    "user": {
      "name": "you@example.com",
      "type": "user"
    }
  }
]
```

b. アクティブなアカウントの詳細を表示し、**tenantId** 値が使用するサブスクリプションと一致することを確認します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "9bab1460-96d5-40b3-a78e-17b15e978a80",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "6057c7e9-b3ae-489d-a54e-de3f6bf6a8ee", (1)
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

(1) **tenantId** パラメーターの値が適切なサブスクリプションの UUID であることを確認します。

c. 適切なサブスクリプションを使用していない場合には、アクティブなサブスクリプションを変更します。

```
$ az account set -s <id> (1)
```

(1) 使用する必要のあるサブスクリプションの **id** の値を **<id>** の代わりに使用します。

d. アクティブなサブスクリプションを変更したら、アカウント情報を再度表示します。

```
$ az account show
```

出力例

```
{
  "environmentName": "AzureCloud",
  "id": "33212d16-bdf6-45cb-b038-f6565b61edda",
  "isDefault": true,
  "name": "Subscription Name",
  "state": "Enabled",
  "tenantId": "8049c7e9-c3de-762d-a54e-dc3f6be6a7ee",
  "user": {
    "name": "you@example.com",
    "type": "user"
  }
}
```

3. 直前の出力の **tenantId** および **id** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

4. アカウントのサービスプリンシパルを作成します。

```
$ az ad sp create-for-rbac --role Contributor --name <service_principal> 1
```

1 **<service_principal>** を、サービスプリンシパルに割り当てる名前に置き換えます。

出力例

```
Changing "<service_principal>" to a valid URI of "http://<service_principal>", which is the
required format used for service principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
Retrying role assignment creation: 4/36
{
  "appId": "8bd0d04d-0ac2-43a8-928d-705c598c6956",
  "displayName": "<service_principal>",
  "name": "http://<service_principal>",
  "password": "ac461d78-bf4b-4387-ad16-7e32e328aec6",
  "tenant": "6048c7e9-b2ad-488d-a54e-dc3f6be6a7ee"
}
```

5. 直前の出力の **appId** および **password** パラメーターの値を記録します。OpenShift Container Platform のインストール時にこれらの値が必要になります。

6. サービスプリンシパルに追加パーミッションを付与します。サービスプリンシパルには、クラスターでそのコンポーネントの認証情報を割り当てられるようにレガシーの **Azure Active Directory Graph** → **Application.ReadWrite.OwnedBy** パーミッションおよび **User Access Administrator** ロールが必要です。

   a. **User Access Administrator** ロールを割り当てるには、以下のコマンドを実行します。

```
$ az role assignment create --role "User Access Administrator" \
    --assignee-object-id $(az ad sp list --filter "appId eq '<appId>'" \ 1
      | jq '.[0].objectId' -r)
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

b. **Azure Active Directory Graph** パーミッションを割り当てるには、以下のコマンドを実行します。

```
$ az ad app permission add --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000 \
    --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

出力例

```
Invoking "az ad app permission grant --id 46d33abc-b8a3-46d8-8c84-f0fd58177435 --api
00000002-0000-0000-c000-000000000000" is needed to make the change effective
```

このコマンドで付与する特定のパーミッションについての詳細は、GUID Table for Windows Azure Active Directory Permissions を参照してください。

c. パーミッション要求を承認します。アカウントに Azure Active Directory テナント管理者ロールがない場合は、所属する組織のガイドラインに従い、テナント管理者にパーミッション要求を承認するようにリクエストしてください。

```
$ az ad app permission grant --id <appId> \ 1
    --api 00000002-0000-0000-c000-000000000000
```

1 **<appId>** を、サービスプリンシパルの **appId** パラメーター値に置き換えます。

3.8.3.7. サポート対象の Azure リージョン

インストールプログラムは、サブスクリプションに基づいて利用可能な Microsoft Azure リージョンの一覧を動的に生成します。以下の Azure リージョンは OpenShift Container Platform バージョン 4.5.4 でテストされ、検証されています。

- **australiacentral** (Australia Central)
- **australiaeast** (Australia East)
- **australiasoutheast** (Australia South East)
- **brazilsouth** (Brazil South)
- **canadacentral** (Canada Central)
- **canadaeast** (Canada East)
- **centralindia** (Central India)

- **centralus** (Central US)
- **eastasia** (East Asia)
- **eastus** (East US)
- **eastus2** (East US 2)
- **francecentral** (France Central)
- **germanywestcentral** (Germany West Central)
- **japaneast** (Japan East)
- **japanwest** (Japan West)
- **koreacentral** (Korea Central)
- **koreasouth** (Korea South)
- **northcentralus** (North Central US)
- **northeurope** (North Europe)
- **norwayeast** (Norway East)
- **southafricanorth** (South Africa North)
- **southcentralus** (South Central US)
- **southeastasia** (Southeast Asia)
- **southindia** (South India)
- **switzerlandnorth** (Switzerland North)
- **uaenorth** (UAE North)
- **uksouth** (UK South)
- **ukwest** (UK West)
- **westcentralus** (West Central US)
- **westeurope** (West Europe)
- **westindia** (West India)
- **westus** (West US)
- **westus2** (West US 2)

3.8.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

3.8.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

3.8.6. AWS のインストールファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャー を使用して OpenShift Container Platform を Microsoft Azure にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

3.8.6.1. インストール設定ファイルの作成

Microsoft Azure にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

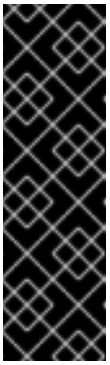
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **azure** を選択します。

iii. お使いのコンピューターに Microsoft Azure プロファイルが保存されていない場合は、サブスクリプションとサービスプリンシパルに以下の Azure パラメーター値を指定します。

- **azure subscription id** クラスターに使用するサブスクリプション ID。アカウント出力に **id** 値を指定します。
- **azure tenant id** テナント ID。アカウント出力に **tenantId** 値を指定します。
- **azure service principal client id** サービスプリンシパルの **appId** パラメーターの値。
- **azure service principal client secret** サービスプリンシパルの **password** パラメーターの値。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成した Azure DNS ゾーンに対応します。

vi. クラスターの記述名を入力します。

重要

パブリックエンドポイントで利用可能なすべての Azure リソースはリソース名の制限を受けるため、特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの Resolve reserved resource name errors を参照してください。

vii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

3.8.6.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

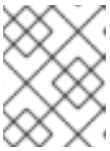
1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。
3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しませ

ん。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

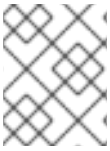
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

3.8.6.3. ARM テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Microsoft Azure で実行するのに役立つ指定の Azure Resource Manager (ARM) テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の ARM テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. 提供される ARM テンプレートで使用される **install-config.yaml** にある一般的な変数をエクスポートします。

```
$ export CLUSTER_NAME=<cluster_name>1
$ export AZURE_REGION=<azure_region>2
$ export SSH_KEY=<ssh_key>3
```

```
$ export BASE_DOMAIN=<base_domain> 4
$ export BASE_DOMAIN_RESOURCE_GROUP=<base_domain_resource_group> 5
```

1. **install-config.yaml** ファイルからの **.metadata.name** 属性の値。
2. クラスターをデプロイするリージョン (例: **centralus**)。これは、**install-config.yaml** ファイルからの **.platform.azure.region** 属性の値です。
3. 文字列としての SSH RSA 公開鍵ファイル。SSH キーは、スペースが含まれているために引用符で囲む必要があります。これは、**install-config.yaml** ファイルからの **.sshKey** 属性の値です。
4. クラスターをデプロイするベースドメイン。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。これは、**install-config.yaml** からの **.baseDomain** 属性の値です。
5. パブリック DNS ゾーンが存在するリソースグループ。これは、**install-config.yaml** ファイルからの **.platform.azure.baseDomainResourceGroupName** 属性の値です。

以下に例を示します。

```
$ export CLUSTER_NAME=test-cluster
$ export AZURE_REGION=centralus
$ export SSH_KEY="ssh-rsa xxx/xxx/xxx= user@email.com"
$ export BASE_DOMAIN=example.com
$ export BASE_DOMAIN_RESOURCE_GROUP=ocp-cluster
```

2. kubeadmin 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

3.8.6.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。

- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
```

```
spec:
  baseDomain: example.openshift.com
  privateZone: (1)
    id: mycluster-100419-private-zone
  publicZone: (2)
    id: example.openshift.com
status: {}
```

(1) (2) このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. ユーザーによってプロビジョニングされるインフラストラクチャーで Azure を設定する場合、Azure Resource Manager (ARM) テンプレートで後に使用するためにマニフェストファイルに定義された一般的な変数の一部をエクスポートする必要があります。

   a. 以下のコマンドを使用してインフラストラクチャー ID をエクスポートします。

   ```
   $ export INFRA_ID=<infra_id> (1)
   ```

   (1) OpenShift Container Platform クラスターには、**<cluster_name>-<random_string>** の形式の識別子 (**INFRA_ID**) が割り当てられます。これは、提供される ARM テンプレートを使用して作成されるほとんどのリソースのベース名として使用されます。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.infrastructureName** 属性の値です。

   b. 以下のコマンドを使用してリソースグループをエクスポートします。

   ```
   $ export RESOURCE_GROUP=<resource_group> (1)
   ```

   (1) この Azure デプロイメントで作成されたすべてのリソースは、リソースグループ の一部として存在します。リソースグループ名は、**<cluster_name>-<random_string>-rg** 形式の **INFRA_ID** をベースとしています。これは、**manifests/cluster-infrastructure-02-config.yml** ファイルからの **.status.platformStatus.azure.resourceGroupName** 属性の値です。

7. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> (1)
```

(1) **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

3.8.7. Azure リソースグループおよびアイデンティティーの作成

Microsoft Azure リソースグループ およびリソースグループのアイデンティティーを作成する必要があります。これらはいずれも Azure での OpenShift Container Platform クラスターのインストール時に使用されます。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. サポートされる Azure リージョンにリソースグループを作成します。

```
$ az group create --name ${RESOURCE_GROUP} --location ${AZURE_REGION}
```

2. リソースグループの Azure アイデンティティーを作成します。

```
$ az identity create -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity
```

これは、クラスター内の Operator に必要なアクセスを付与するために使用されます。たとえば、これにより Ingress Operator はパブリック IP およびそのロードバランサーを作成できます。Azure アイデンティティーをロールに割り当てる必要があります。

3. Contributor ロールを Azure アイデンティティーに付与します。

a. Azure ロールの割り当てで必要な以下の変数をエクスポートします。

```
$ export PRINCIPAL_ID=`az identity show -g ${RESOURCE_GROUP} -n ${INFRA_ID}-identity --query principalId --out tsv`
```

```
$ export RESOURCE_GROUP_ID=`az group show -g ${RESOURCE_GROUP} --query id --out tsv`
```

b. Contributor ロールをアイデンティティーに割り当てます。

```
$ az role assignment create --assignee "${PRINCIPAL_ID}" --role 'Contributor' --scope "${RESOURCE_GROUP_ID}"
```

3.8.8. RHCOS クラスターイメージおよびブートストラップ Ignition 設定ファイルのアップロード

Azure クライアントは、ローカルにあるファイルに基づくデプロイメントをサポートしないため、RHCOS 仮想ハードディスク (VHD) クラスターイメージおよびブートストラップ Ignition 設定ファイルをコピーし、それらをストレージコンテナーに保存し、それらをデプロイメント時にアクセスできるようにする必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. VHD クラスターイメージを保存するために Azure ストレージアカウントを作成します。

```
$ az storage account create -g ${RESOURCE_GROUP} --location ${AZURE_REGION} --
name ${CLUSTER_NAME}sa --kind Storage --sku Standard_LRS
```

警告

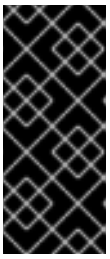
Azure ストレージアカウント名は 3 文字から 24 文字の長さで、数字および小文字のみを使用する必要があります。**CLUSTER_NAME** 変数がこれらの制限に準拠しない場合、Azure ストレージアカウント名を手動で定義する必要があります。Azure ストレージアカウント名の制限についての詳細は、Azure ドキュメントの Resolve errors for storage account names を参照してください。

2. ストレージアカウントキーを環境変数としてエクスポートします。

```
$ export ACCOUNT_KEY=`az storage account keys list -g ${RESOURCE_GROUP} --
account-name ${CLUSTER_NAME}sa --query "[0].value" -o tsv`
```

3. 使用する RHCOS バージョンを選択し、その VHD の URL を環境変数にエクスポートします。

```
$ export VHD_URL=`curl -s https://raw.githubusercontent.com/openshift/installer/release-
4.5/data/data/rhcos.json | jq -r .azure.url`
```



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージを指定する必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

4. 選択した VHD を blob にコピーします。

```
$ az storage container create --name vhd --account-name ${CLUSTER_NAME}sa --account-
key ${ACCOUNT_KEY}
```

```
$ az storage blob copy start --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} --destination-blob "rhcos.vhd" --destination-container vhd --source-uri
"${VHD_URL}"
```

VHD コピータスクの進捗を追跡するには、以下のスクリプトを実行します。

```
status="unknown"
while [ "$status" != "success" ]
do
  status=`az storage blob show --container-name vhd --name "rhcos.vhd" --account-name
${CLUSTER_NAME}sa --account-key ${ACCOUNT_KEY} -o tsv --query
properties.copy.status`
  echo $status
done
```

5. blob ストレージコンテナーを作成し、生成された **bootstrap.ign** ファイルをアップロードします。

```
$ az storage container create --name files --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} --public-access blob
```

```
$ az storage blob upload --account-name ${CLUSTER_NAME}sa --account-key
${ACCOUNT_KEY} -c "files" -f "<installation_directory>/bootstrap.ign" -n "bootstrap.ign"
```

3.8.9. DNS ゾーンの作成例

DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターに必要です。シナリオに適した DNS ストラテジーを選択する必要があります。

この例の場合、Azure の DNS ソリューション が使用されるため、外部 (インターネット) の可視性のために新規パブリック DNS ゾーンと、内部クラスターの解決用にプライベート DNS ゾーンが作成されます。

注記

パブリック DNS ゾーンは、クラスターデプロイメントと同じリソースグループに存在している必要はなく、必要なベースドメイン用にすでに組織内に存在している可能性があります。その場合、パブリック DNS ゾーンの作成を省略できます。先に生成したインストール設定がこのシナリオに基づいていることを確認してください。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. **BASE_DOMAIN_RESOURCE_GROUP** 環境変数でエクスポートされたリソースグループに、新規のパブリック DNS ゾーンを作成します。

```
$ az network dns zone create -g ${BASE_DOMAIN_RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

すでに存在するパブリック DNS ゾーンを使用している場合は、この手順を省略できます。

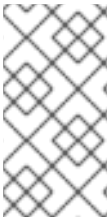
2. このデプロイメントの残りの部分と同じリソースグループにプライベート DNS ゾーンを作成します。

```
$ az network private-dns zone create -g ${RESOURCE_GROUP} -n
${CLUSTER_NAME}.${BASE_DOMAIN}
```

Azure でのパブリック DNS ゾーンの設定 についてのセクションを参照してください。

3.8.10. Azure での VNet の作成

OpenShift Container Platform クラスター用に Microsoft Azure で使用する仮想ネットワーク (VNet) を作成する必要があります。各種の要件を満たすように VPC をカスタマイズできます。VNet を作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。



注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VNet の ARM テンプレート**セクションからテンプレートをコピーし、これを **01_vnet.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要な VNet について記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/01_vnet.json" \
  --parameters baseName="${INFRA_ID}" 1
```

1 リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. VNet テンプレートをプライベート DNS ゾーンにリンクします。

```
$ az network private-dns link vnet create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n ${INFRA_ID}-network-link -v "${INFRA_ID}-vnet"
-e false
```

3.8.10.1. VNet の ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な VNet をデプロイすることができます。

例3.101_vnet.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "addressPrefix" : "10.0.0.0/16",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetPrefix" : "10.0.0.0/24",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetPrefix" : "10.0.1.0/24",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/virtualNetworks",
      "name" : "[variables('virtualNetworkName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[concat('Microsoft.Network/networkSecurityGroups/', variables('clusterNsgName'))]"
      ],
      "properties" : {
        "addressSpace" : {
          "addressPrefixes" : [
            "[variables('addressPrefix')]"
          ]
        },
        "subnets" : [
          {
            "name" : "[variables('masterSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('masterSubnetPrefix')]",
              "serviceEndpoints": [],
              "networkSecurityGroup" : {
                "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
              }
            }
          },
          {
            "name" : "[variables('nodeSubnetName')]",
            "properties" : {
              "addressPrefix" : "[variables('nodeSubnetPrefix')]",
```

```
            "serviceEndpoints": [],
            "networkSecurityGroup" : {
              "id" : "[resourceId('Microsoft.Network/networkSecurityGroups',
variables('clusterNsgName'))]"
            }
          }
        }
      ]
    }
  },
  {
    "type" : "Microsoft.Network/networkSecurityGroups",
    "name" : "[variables('clusterNsgName')]",
    "apiVersion" : "2018-10-01",
    "location" : "[variables('location')]",
    "properties" : {
      "securityRules" : [
        {
          "name" : "apiserver_in",
          "properties" : {
            "protocol" : "Tcp",
            "sourcePortRange" : "*",
            "destinationPortRange" : "6443",
            "sourceAddressPrefix" : "*",
            "destinationAddressPrefix" : "*",
            "access" : "Allow",
            "priority" : 101,
            "direction" : "Inbound"
          }
        }
      ]
    }
  }
 ]
}
```

3.8.11. Azure インフラストラクチャー用の RHCOS クラスターイメージのデプロイ

OpenShift Container Platform ノードに Microsoft Azure 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- RHCOS 仮想ハードディスク (VHD) クラスターイメージを Azure ストレージコンテナーに保存します。
- ブートストラップ Ignition 設定ファイルを Azure ストレージコンテナーに保存します。

手順

1. 本トピックの **イメージストレージの ARM テンプレート**セクションからテンプレートをコピーし、これを **02_storage.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なイメージストレージ について記述しています。

2. RHCOS VHD blob URL を変数としてエクスポートします。

```
$ export VHD_BLOB_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c vhd -n "rhcos.vhd" -o tsv`
```

3. クラスターイメージのデプロイ

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/02_storage.json" \
  --parameters vhdBlobURL="${VHD_BLOB_URL}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. マスターマシンおよびワーカーマシンを作成するために使用される RHCOS VHD の blob URL。

2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3.8.11.1. イメージストレージの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要な保存された Red Hat Enterprise Linux CoreOS (RHCOS) をデプロイすることができます。

例3.2 02_storage.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "vhdBlobURL" : {
      "type" : "string",
      "metadata" : {
        "description" : "URL pointing to the blob where the VHD to be used to create master and
worker machines is located"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "imageName" : "[concat(parameters('baseName'), '-image')]"
  },
  "resources" : [
```

```
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Compute/images",
      "name": "[variables('imageName')]",
      "location" : "[variables('location')]",
      "properties": {
        "storageProfile": {
          "osDisk": {
            "osType": "Linux",
            "osState": "Generalized",
            "blobUri": "[parameters('vhdBlobURL')]",
            "storageAccountType": "Standard_LRS"
          }
        }
      }
    }
  ]
}
```

3.8.12. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表3.20 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。

プロトコル	ポート	説明
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

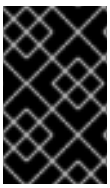
表3.21 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表3.22 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

   表3.23 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

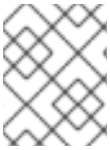
表3.24 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

3.8.13. Azure でのネットワークおよび負荷分散コンポーネントの作成

OpenShift Container Platform クラスターで使用するネットワークおよび負荷分散を Microsoft Azure で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供される ARM テンプレートを使用して Azure インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。

手順

1. 本トピックの **ネットワークおよびロードばランサーの ARM テンプレート**セクションからテンプレートをコピーし、これを **03_infra.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なネットワークおよび負荷分散オブジェクトについて記述しています。
2. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/03_infra.json" \
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" \ 1
  --parameters baseName="${INFRA_ID}" 2
```

1. プライベート DNS ゾーンの名前。
2. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3. API パブリックロードバランサーのパブリックゾーンに **api** DNS レコードを作成します。**${BASE_DOMAIN_RESOURCE_GROUP}** 変数は、パブリック DNS ゾーンがあるリソースグループをポイントする必要があります。

   a. 以下の変数をエクスポートします。

   ```
   $ export PUBLIC_IP=`az network public-ip list -g ${RESOURCE_GROUP} --query "[?
   name=='${INFRA_ID}-master-pip'] | [0].ipAddress" -o tsv`
   ```

   b. 新しいパブリックゾーンに DNS レコードを作成します。

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
   z ${CLUSTER_NAME}.${BASE_DOMAIN} -n api -a ${PUBLIC_IP} --ttl 60
   ```

   c. クラスターを既存のパブリックゾーンに追加する場合は、DNS レコードを代わりに作成できます。

   ```
   $ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
   z ${BASE_DOMAIN} -n api.${CLUSTER_NAME} -a ${PUBLIC_IP} --ttl 60
   ```

3.8.13.1. ネットワークおよびロードバランサーの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用して、OpenShift Container Platform クラスターに必要なネットワークオブジェクトおよびロードバランサーをデプロイすることができます。

例3.3 03_infra.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
```

```
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterPublicIpAddressName" : "[concat(parameters('baseName'), '-master-pip')]",
    "masterPublicIpAddressID" : "[resourceId('Microsoft.Network/publicIPAddresses',
variables('masterPublicIpAddressName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "masterLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('masterLoadBalancerName'))]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "internalLoadBalancerID" : "[resourceId('Microsoft.Network/loadBalancers',
variables('internalLoadBalancerName'))]",
    "skuName": "Standard"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('masterPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('masterPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/loadBalancers",
      "name" : "[variables('masterLoadBalancerName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "[variables('skuName')]"
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/publicIPAddresses/', variables('masterPublicIpAddressName'))]"
      ],
      "properties" : {
        "frontendIPConfigurations" : [
          {
            "name" : "public-lb-ip",
            "properties" : {
              "publicIPAddress" : {
                "id" : "[variables('masterPublicIpAddressID')]"
              }
            }
          }
        ],
        "backendAddressPools" : [
```

```
        {
          "name" : "public-lb-backend"
        }
      ],
      "loadBalancingRules" : [
        {
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" :"[concat(variables('masterLoadBalancerID'), '/frontendIPConfigurations/public-lb-
ip')]"
            },
            "backendAddressPool" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/backendAddressPools/public-lb-
backend')]"
            },
            "protocol" : "Tcp",
            "loadDistribution" : "Default",
            "idleTimeoutInMinutes" : 30,
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "probe" : {
              "id" : "[concat(variables('masterLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion" : "2018-12-01",
    "type" : "Microsoft.Network/loadBalancers",
    "name" : "[variables('internalLoadBalancerName')]",
    "location" : "[variables('location')]",
    "sku": {
      "name": "[variables('skuName')]"
    },
    "properties" : {
      "frontendIPConfigurations" : [
        {
          "name" : "internal-lb-ip",
          "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "subnet" : {
```

```
              "id" : "[variables('masterSubnetRef')]"
            },
            "privateIPAddressVersion" : "IPv4"
          }
        }
      ],
      "backendAddressPools" : [
        {
          "name" : "internal-lb-backend"
        }
      ],
      "loadBalancingRules" : [
        {
          "name" : "api-internal",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 6443,
            "backendPort" : 6443,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/api-internal-probe')]"
            }
          }
        },
        {
          "name" : "sint",
          "properties" : {
            "frontendIPConfiguration" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/frontendIPConfigurations/internal-lb-
ip')]"
            },
            "frontendPort" : 22623,
            "backendPort" : 22623,
            "enableFloatingIP" : false,
            "idleTimeoutInMinutes" : 30,
            "protocol" : "Tcp",
            "enableTcpReset" : false,
            "loadDistribution" : "Default",
            "backendAddressPool" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/backendAddressPools/internal-lb-
backend')]"
            },
            "probe" : {
              "id" : "[concat(variables('internalLoadBalancerID'), '/probes/sint-probe')]"
            }
```

```
          }
        }
      ],
      "probes" : [
        {
          "name" : "api-internal-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 6443,
            "requestPath": "/readyz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        },
        {
          "name" : "sint-probe",
          "properties" : {
            "protocol" : "Https",
            "port" : 22623,
            "requestPath": "/healthz",
            "intervalInSeconds" : 10,
            "numberOfProbes" : 3
          }
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "name": "[concat(parameters('privateDNSZoneName'), '/api')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
    ],
    "properties": {
      "ttl": 60,
      "aRecords": [
        {
          "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "name": "[concat(parameters('privateDNSZoneName'), '/api-int')]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/loadBalancers/', variables('internalLoadBalancerName'))]"
    ],
    "properties": {
      "ttl": 60,
```

```
        "aRecords": [
          {
            "ipv4Address": "
[reference(variables('internalLoadBalancerName')).frontendIPConfigurations[0].properties.privateIP
Address]"
          }
        ]
      }
    }
  ]
}
```

3.8.14. Azure でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Microsoft Azure で作成する必要があります。このマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供されている ARM テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。

手順

1. 本トピックの **ブートストラップマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. ブートストラップマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export BOOTSTRAP_URL=`az storage blob url --account-name ${CLUSTER_NAME}sa --
account-key ${ACCOUNT_KEY} -c "files" -n "bootstrap.ign" -o tsv`
$ export BOOTSTRAP_IGNITION=`jq -rcnM --arg v "2.2.0" --arg url ${BOOTSTRAP_URL}
'{ignition:{version:$v,config:{replace:{source:$url}}}}' | base64 -w0`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/04_bootstrap.json" \
  --parameters bootstrapIgnition="${BOOTSTRAP_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. ブートストラップクラスターのブートストラップ Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3.8.14.1. ブートストラップマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例3.4 04_bootstrap.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "bootstrapIgnition" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Bootstrap ignition content for the bootstrap cluster"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string."
      }
    },
    "bootstrapVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D4s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
```

```
        "Standard_A7",
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
```

```
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Bootstrap Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "vmName" : "[concat(parameters('baseName'), '-bootstrap')]",
    "nicName" : "[concat(variables('vmName'), '-nic')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "clusterNsgName" : "[concat(parameters('baseName'), '-nsg')]",
    "sshPublicIpAddressName" : "[concat(variables('vmName'), '-ssh-pip')]"
  },
  "resources" : [
    {
      "apiVersion" : "2018-12-01",
      "type" : "Microsoft.Network/publicIPAddresses",
      "name" : "[variables('sshPublicIpAddressName')]",
      "location" : "[variables('location')]",
      "sku": {
        "name": "Standard"
      },
      "properties" : {
        "publicIPAllocationMethod" : "Static",
        "dnsSettings" : {
          "domainNameLabel" : "[variables('sshPublicIpAddressName')]"
        }
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "name" : "[variables('nicName')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('sshPublicIpAddressName'))]"
      ],
      "properties" : {
        "ipConfigurations" : [
          {
```

```
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',
variables('sshPublicIpAddressName'))]"
              },
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
              },
              "loadBalancerBackendAddressPools" : [
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
                },
                {
                  "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
                }
              ]
            }
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "name" : "[variables('vmName')]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('bootstrapVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmName')]",
          "adminUsername" : "core",
          "customData" : "[parameters('bootstrapIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
              "publicKeys" : [
                {
                  "path" : "[variables('sshKeyPath')]",
```

```
                "keyData" : "[parameters('sshKeyData')]"
              }
            ]
          }
        }
      },
      "storageProfile" : {
        "imageReference": {
          "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
        },
        "osDisk" : {
          "name": "[concat(variables('vmName'),'_OSDisk')]",
          "osType" : "Linux",
          "createOption" : "FromImage",
          "managedDisk": {
            "storageAccountType": "Premium_LRS"
          },
          "diskSizeGB" : 100
        }
      },
      "networkProfile" : {
        "networkInterfaces" : [
          {
            "id" : "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type": "Microsoft.Network/networkSecurityGroups/securityRules",
      "name" : "[concat(variables('clusterNsgName'), '/bootstrap_ssh_in')]",
      "location" : "[variables('location')]",
      "dependsOn" : [
        "[resourceId('Microsoft.Compute/virtualMachines', variables('vmName'))]"
      ],
      "properties": {
        "protocol" : "Tcp",
        "sourcePortRange" : "*",
        "destinationPortRange" : "22",
        "sourceAddressPrefix" : "*",
        "destinationAddressPrefix" : "*",
        "access" : "Allow",
        "priority" : 100,
        "direction" : "Inbound"
      }
    }
  ]
}
```

3.8.15. Azure でのコントロールプレーンの作成

クラスターで使用するコントロールプレーンマシンを Microsoft Azure で作成する必要があります。これらのマシンを作成する方法として、提供される Azure Resource Manager (ARM) テンプレートを変更することができます。

注記

提供される ARM テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの **コントロールプレーンマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **05_masters.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. コントロールプレーンマシンのデプロイメントに必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign | base64`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/05_masters.json" \
  --parameters masterIgnition="${MASTER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters privateDNSZoneName="${CLUSTER_NAME}.${BASE_DOMAIN}" 3
  --parameters baseName="${INFRA_ID}" 4
```

1. マスターノードの Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. マスターノードが割り当てられているプライベート DNS ゾーンの名前。
4. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3.8.15.1. コントロールプレーンマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例3.5 05_masters.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "masterIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the master nodes"
      }
    },
    "numberOfMasters" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 2,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift masters to deploy"
      }
    },
    "sshKeyData" : {
      "type" : "securestring",
      "metadata" : {
        "description" : "SSH RSA public key file as a string"
      }
    },
    "privateDNSZoneName" : {
      "type" : "string",
      "metadata" : {
        "description" : "Name of the private DNS zone the master nodes are going to be attached to"
      }
    },
    "masterVMSize" : {
      "type" : "string",
      "defaultValue" : "Standard_D8s_v3",
      "allowedValues" : [
        "Standard_A2",
        "Standard_A3",
        "Standard_A4",
        "Standard_A5",
        "Standard_A6",
        "Standard_A7",
```

```
        "Standard_A8",
        "Standard_A9",
        "Standard_A10",
        "Standard_A11",
        "Standard_D2",
        "Standard_D3",
        "Standard_D4",
        "Standard_D11",
        "Standard_D12",
        "Standard_D13",
        "Standard_D14",
        "Standard_D2_v2",
        "Standard_D3_v2",
        "Standard_D4_v2",
        "Standard_D5_v2",
        "Standard_D8_v3",
        "Standard_D11_v2",
        "Standard_D12_v2",
        "Standard_D13_v2",
        "Standard_D14_v2",
        "Standard_E2_v3",
        "Standard_E4_v3",
        "Standard_E8_v3",
        "Standard_E16_v3",
        "Standard_E32_v3",
        "Standard_E64_v3",
        "Standard_E2s_v3",
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
```

```
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the Master Virtual Machines"
      }
    },
    "diskSizeGB" : {
      "type" : "int",
      "defaultValue" : 1024,
      "metadata" : {
        "description" : "Size of the Master VM OS disk, in GB"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "masterSubnetName" : "[concat(parameters('baseName'), '-master-subnet')]",
    "masterSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('masterSubnetName'))]",
    "masterLoadBalancerName" : "[concat(parameters('baseName'), '-public-lb')]",
    "internalLoadBalancerName" : "[concat(parameters('baseName'), '-internal-lb')]",
    "sshKeyPath" : "/home/core/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfMasters')]",
        "input" : "[concat(parameters('baseName'), '-master-', copyIndex('vmNames'))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Network/networkInterfaces",
      "copy" : {
        "name" : "nicCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
      "location" : "[variables('location')]",
      "properties" : {
        "ipConfigurations" : [
          {
            "name" : "pipConfig",
            "properties" : {
              "privateIPAllocationMethod" : "Dynamic",
              "subnet" : {
                "id" : "[variables('masterSubnetRef')]"
```

```
            },
            "loadBalancerBackendAddressPools" : [
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('masterLoadBalancerName'), '/backendAddressPools/public-lb-backend')]"
              },
              {
                "id" : "[concat('/subscriptions/', subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('internalLoadBalancerName'), '/backendAddressPools/internal-lb-backend')]"
              }
            ]
          }
        }
      ]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/SRV",
    "name": "[concat(parameters('privateDNSZoneName'), '/_etcd-server-ssl._tcp')]",
    "location" : "[variables('location')]",
    "properties": {
      "ttl": 60,
      "copy": [{
        "name": "srvRecords",
        "count": "[length(variables('vmNames'))]",
        "input": {
          "priority": 0,
          "weight" : 10,
          "port" : 2380,
          "target" : "[concat('etcd-', copyIndex('srvRecords'), '.',
parameters('privateDNSZoneName'))]"
        }
      }]
    }
  },
  {
    "apiVersion": "2018-09-01",
    "type": "Microsoft.Network/privateDnsZones/A",
    "copy" : {
      "name" : "dnsCopy",
      "count" : "[length(variables('vmNames'))]"
    },
    "name": "[concat(parameters('privateDNSZoneName'), '/etcd-', copyIndex())]",
    "location" : "[variables('location')]",
    "dependsOn" : [
      "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]"
    ],
    "properties": {
      "ttl": 60,
      "aRecords": [
        {
          "ipv4Address": "[reference(concat(variables('vmNames')[copyIndex()], '-
```

```
nic')).ipConfigurations[0].properties.privateIPAddress]"
          }
        ]
      }
    },
    {
      "apiVersion" : "2018-06-01",
      "type" : "Microsoft.Compute/virtualMachines",
      "copy" : {
        "name" : "vmCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "name" : "[variables('vmNames')[copyIndex()]]",
      "location" : "[variables('location')]",
      "identity" : {
        "type" : "userAssigned",
        "userAssignedIdentities" : {
          "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
        }
      },
      "dependsOn" : [
        "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')[copyIndex()], '-
nic'))]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/A/etcd-', copyIndex())]",
        "[concat('Microsoft.Network/privateDnsZones/', parameters('privateDNSZoneName'),
'/SRV/_etcd-server-ssl._tcp')]"
      ],
      "properties" : {
        "hardwareProfile" : {
          "vmSize" : "[parameters('masterVMSize')]"
        },
        "osProfile" : {
          "computerName" : "[variables('vmNames')[copyIndex()]]",
          "adminUsername" : "core",
          "customData" : "[parameters('masterIgnition')]",
          "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
              "publicKeys" : [
                {
                  "path" : "[variables('sshKeyPath')]",
                  "keyData" : "[parameters('sshKeyData')]"
                }
              ]
            }
          }
        },
        "storageProfile" : {
          "imageReference": {
            "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
          },
          "osDisk" : {
            "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
            "osType" : "Linux",
```

```
            "createOption" : "FromImage",
            "caching": "ReadOnly",
            "writeAcceleratorEnabled": false,
            "managedDisk": {
              "storageAccountType": "Premium_LRS"
            },
            "diskSizeGB" : "[parameters('diskSizeGB')]"
          }
        },
        "networkProfile" : {
          "networkInterfaces" : [
            {
              "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')
[copyIndex()], '-nic'))]",
              "properties": {
                "primary": false
              }
            }
          ]
        }
      }
    }
  ]
}
```

3.8.16. ブートストラップの完了を待機し、Azure のブートストラップリソースを削除する

Microsoft Azure ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。
- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ az network nsg rule delete -g ${RESOURCE_GROUP} --nsg-name ${INFRA_ID}-nsg --
name bootstrap_ssh_in
$ az vm stop -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm deallocate -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap
$ az vm delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap --yes
$ az disk delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap_OSDisk --no-
wait --yes
$ az network nic delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-nic --no-
wait
$ az storage blob delete --account-key ${ACCOUNT_KEY} --account-name
${CLUSTER_NAME}sa --container-name files --name bootstrap.ign
$ az network public-ip delete -g ${RESOURCE_GROUP} --name ${INFRA_ID}-bootstrap-
ssh-pip
```

3.8.17. Azure での追加のワーカーマシンの作成

Microsoft Azure でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Azure Resource Manager (ARM) テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_workers.json** というタイプのリソースを追加して起動することができます。

注記

提供される ARM テンプレートを使用してワーカーマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- Azure アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- Azure で VNet および関連するサブネットを作成し、設定します。

- Azure でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの **ワーカーマシンの ARM テンプレート**セクションからテンプレートをコピーし、これを **06_workers.json** としてクラスターのインストールディレクトリーに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. ワーカーマシンのデプロイメントで必要な以下の変数をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign | base64`
```

3. **az** CLI を使用してデプロイメントを作成します。

```
$ az deployment group create -g ${RESOURCE_GROUP} \
  --template-file "<installation_directory>/06_workers.json" \
  --parameters workerIgnition="${WORKER_IGNITION}" \ 1
  --parameters sshKeyData="${SSH_KEY}" \ 2
  --parameters baseName="${INFRA_ID}" 3
```

1. ワーカーノードの Ignition コンテンツ。
2. 文字列としての SSH RSA 公開鍵ファイル。
3. リソース名で使用されるベース名。これは通常クラスターのインフラストラクチャー ID です。

3.8.17.1. ワーカーマシンの ARM テンプレート

以下の Azure Resource Manager (ARM) テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例3.6 06_workers.json ARM テンプレート

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
```

```
    "metadata" : {
      "description" : "Ignition content for the worker nodes"
    }
  },
  "numberOfNodes" : {
    "type" : "int",
    "defaultValue" : 3,
    "minValue" : 2,
    "maxValue" : 30,
    "metadata" : {
      "description" : "Number of OpenShift compute nodes to deploy"
    }
  },
  "sshKeyData" : {
    "type" : "securestring",
    "metadata" : {
      "description" : "SSH RSA public key file as a string"
    }
  },
  "nodeVMSize" : {
    "type" : "string",
    "defaultValue" : "Standard_D4s_v3",
    "allowedValues" : [
      "Standard_A2",
      "Standard_A3",
      "Standard_A4",
      "Standard_A5",
      "Standard_A6",
      "Standard_A7",
      "Standard_A8",
      "Standard_A9",
      "Standard_A10",
      "Standard_A11",
      "Standard_D2",
      "Standard_D3",
      "Standard_D4",
      "Standard_D11",
      "Standard_D12",
      "Standard_D13",
      "Standard_D14",
      "Standard_D2_v2",
      "Standard_D3_v2",
      "Standard_D4_v2",
      "Standard_D5_v2",
      "Standard_D8_v3",
      "Standard_D11_v2",
      "Standard_D12_v2",
      "Standard_D13_v2",
      "Standard_D14_v2",
      "Standard_E2_v3",
      "Standard_E4_v3",
      "Standard_E8_v3",
      "Standard_E16_v3",
      "Standard_E32_v3",
      "Standard_E64_v3",
      "Standard_E2s_v3",
```

```
        "Standard_E4s_v3",
        "Standard_E8s_v3",
        "Standard_E16s_v3",
        "Standard_E32s_v3",
        "Standard_E64s_v3",
        "Standard_G1",
        "Standard_G2",
        "Standard_G3",
        "Standard_G4",
        "Standard_G5",
        "Standard_DS2",
        "Standard_DS3",
        "Standard_DS4",
        "Standard_DS11",
        "Standard_DS12",
        "Standard_DS13",
        "Standard_DS14",
        "Standard_DS2_v2",
        "Standard_DS3_v2",
        "Standard_DS4_v2",
        "Standard_DS5_v2",
        "Standard_DS11_v2",
        "Standard_DS12_v2",
        "Standard_DS13_v2",
        "Standard_DS14_v2",
        "Standard_GS1",
        "Standard_GS2",
        "Standard_GS3",
        "Standard_GS4",
        "Standard_GS5",
        "Standard_D2s_v3",
        "Standard_D4s_v3",
        "Standard_D8s_v3"
      ],
      "metadata" : {
        "description" : "The size of the each Node Virtual Machine"
      }
    }
  },
  "variables" : {
    "location" : "[resourceGroup().location]",
    "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
    "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks',
variables('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/',
variables('nodeSubnetName'))]",
    "infraLoadBalancerName" : "[parameters('baseName')]",
    "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
    "identityName" : "[concat(parameters('baseName'), '-identity')]",
    "imageName" : "[concat(parameters('baseName'), '-image')]",
    "copy" : [
      {
        "name" : "vmNames",
        "count" :  "[parameters('numberOfNodes')]",
        "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-',
```

```
copyIndex('vmNames', 1))]"
      }
    ]
  },
  "resources" : [
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
              "location" : "[variables('location')]",
              "properties" : {
                "ipConfigurations" : [
                  {
                    "name" : "pipConfig",
                    "properties" : {
                      "privateIPAllocationMethod" : "Dynamic",
                      "subnet" : {
                        "id" : "[variables('nodeSubnetRef')]"
                      }
                    }
                  }
                ]
              }
            },
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Compute/virtualMachines",
              "name" : "[variables('vmNames')[copyIndex()]]",
              "location" : "[variables('location')]",
              "tags" : {
                "kubernetes.io-cluster-ffranzupi": "owned"
              },
              "identity" : {
                "type" : "userAssigned",
                "userAssignedIdentities" : {
                  "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentities/',
variables('identityName'))]" : {}
                }
              },
              "dependsOn" : [
                "[concat('Microsoft.Network/networkInterfaces/', concat(variables('vmNames')
```

```
[copyIndex()], '-nic'))]"
            ],
            "properties" : {
              "hardwareProfile" : {
                "vmSize" : "[parameters('nodeVMSize')]"
              },
              "osProfile" : {
                "computerName" : "[variables('vmNames')[copyIndex()]]",
                "adminUsername" : "capi",
                "customData" : "[parameters('workerIgnition')]",
                "linuxConfiguration" : {
                  "disablePasswordAuthentication" : true,
                  "ssh" : {
                    "publicKeys" : [
                      {
                        "path" : "[variables('sshKeyPath')]",
                        "keyData" : "[parameters('sshKeyData')]"
                      }
                    ]
                  }
                }
              },
              "storageProfile" : {
                "imageReference": {
                  "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
                },
                "osDisk" : {
                  "name": "[concat(variables('vmNames')[copyIndex()],'_OSDisk')]",
                  "osType" : "Linux",
                  "createOption" : "FromImage",
                  "managedDisk": {
                    "storageAccountType": "Premium_LRS"
                  },
                  "diskSizeGB": 128
                }
              },
              "networkProfile" : {
                "networkInterfaces" : [
                  {
                    "id" : "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('vmNames')[copyIndex()], '-nic'))]",
                    "properties": {
                      "primary": true
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    }
  ]
}
```

3.8.18. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

3.8.18.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.8.18.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。

5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

3.8.18.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

3.8.19. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

3.8.20. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

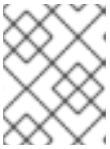
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

3.8.21. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- 独自にプロビジョニングしたインフラストラクチャーを使用して、OpenShift Container Platform クラスターを Microsoft Azure にデプロイしています。
- OpenShift CLI (**oc**) をインストールします。
- **jq** パッケージをインストールします。
- Azure CLI のインストールまたは更新を実行します。

手順

1. Ingress ルーターがロードバランサーを作成し、**EXTERNAL-IP** フィールドにデータを設定していることを確認します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.20.10  35.130.120.110
80:32288/TCP,443:31215/TCP  20
```

2. Ingress ルーター IP を変数としてエクスポートします。

```
$ export PUBLIC_IP_ROUTER=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

3. パブリック DNS ゾーンに ***.apps** レコードを追加します。

   a. このクラスターを新しいパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER} --ttl 300
```

   b. このクラスターを既存のパブリックゾーンに追加する場合は、以下を実行します。

```
$ az network dns record-set a add-record -g ${BASE_DOMAIN_RESOURCE_GROUP} -
z ${BASE_DOMAIN} -n *.apps.${CLUSTER_NAME} -a ${PUBLIC_IP_ROUTER} --ttl 300
```

4. ***.apps** レコードをプライベート DNS ゾーンに追加します。

   a. 以下のコマンドを使用して ***.apps** レコードを作成します。

```
$ az network private-dns record-set a create -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps --ttl 300
```

   b. 以下のコマンドを使用して ***.apps** レコードをプライベート DNS ゾーンに追加します。

```
$ az network private-dns record-set a add-record -g ${RESOURCE_GROUP} -z
${CLUSTER_NAME}.${BASE_DOMAIN} -n *.apps -a ${PUBLIC_IP_ROUTER}
```

ワイルドカードを使用する代わりに明示的なドメインを追加する場合は、クラスターのそれぞれの現行ルートのエントリーを作成できます。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}{"\n"}{end}
{end}' routes
```

出力例

```
oauth-openshift.apps.cluster.basedomain.com
console-openshift-console.apps.cluster.basedomain.com
downloads-openshift-console.apps.cluster.basedomain.com
```

```
alertmanager-main-openshift-monitoring.apps.cluster.basedomain.com
grafana-openshift-monitoring.apps.cluster.basedomain.com
prometheus-k8s-openshift-monitoring.apps.cluster.basedomain.com
```

3.8.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの Azure インストールの実行

Microsoft Azure のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる Azure インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

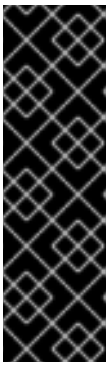
- クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

3.9. AZURE でのクラスターのアンインストール

Microsoft Azure にデプロイしたクラスターは削除することができます。

3.9.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第4章 GCP へのインストール

4.1. GCP プロジェクトの設定

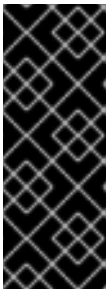
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

4.1.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

4.1.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表4.1 必要な API サービス

API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**

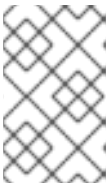
API サービス	コンソールサービス名
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

4.1.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。



注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

4.1.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表4.2 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	コンピュート	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
使用中のグローバル IP アドレス	コンピュート	グローバル	4	1
ヘルスチェック	コンピュート	グローバル	3	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	コンピュート	グローバル	2	0
静的 IP アドレス	コンピュート	リージョン	4	1
ルーター	コンピュート	グローバル	1	0
ルート	コンピュート	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	コンピュート	グローバル	3	0
CPU	コンピュート	リージョン	28	4

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
永続ディスク SSD (GB)	コンピュート	リージョン	896	128

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

4.1.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。

2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

4.1.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表4.3 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

4.1.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.1.7. 次のステップ

- GCP に OpenShift Container Platform クラスターをインストールします。カスタマイズされたクラスターのインストール、またはデフォルトのオプションで クラスターのクイックインストール を実行できます。

4.2. GCP の IAM の手動作成

4.2.1. IAM の手動作成

Cloud Credential Operator は、クラウドアイデンティティーおよびアクセス管理 (IAM) API に到達できない環境にインストールする前に手動モードに配置できます。管理者はクラスター **kube-system** namespace に管理者レベルの認証情報シークレットを保存しないようにします。

手順

1. OpenShift Container Platform インストーラーを実行し、マニフェストを生成します。

```
$ openshift-install create manifests --dir=mycluster
```

2. Cloud Credential Operator が手動モードになるように、設定マップを manifests ディレクトリーに挿入します。

```
$ cat <<EOF > mycluster/manifests/cco-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cloud-credential-operator-config
  namespace: openshift-cloud-credential-operator
  annotations:
    release.openshift.io/create-only: "true"
data:
  disabled: "true"
EOF
```

3. ローカルクラウドの認証情報を使用して作成された **admin** 認証情報シークレットを削除します。この削除により、**admin** 認証情報がクラスターに保存されなくなります。

```
$ rm mycluster/openshift/99_cloud-creds-secret.yaml
```

4. OpenShift Container Platform リリースイメージを取得します。**openshift-install** バイナリーはこれを使用するためにビルドされます。

```
$ bin/openshift-install version
```

出力例

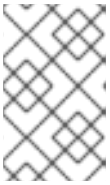
```
release image quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64
```

5. このリリースイメージ内で、デプロイするクラウドをターゲットとする **CredentialsRequest** オブジェクトをすべて特定します。

```
$ oc adm release extract quay.io/openshift-release-dev/ocp-release:4.z.z-x86_64 --to
./release-image
```

6. 展開したファイルで **CredentialsRequests** を見つけます。

```
$ grep -l "apiVersion: cloudcredential.openshift.io" * | xargs cat
```



注記

今後の OpenShift Container Platform リリースでは、**CredentialsRequests** をスキャンし、それらを表示する新規の **oc adm release** コマンドが表示されます。

これにより、それぞれの要求の詳細が表示されます。**spec.providerSpec.kind** がインストールするクラウドプロバイダーと一致しない **CredentialsRequests** については、これを無視するようにしてください。

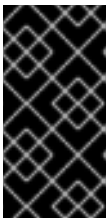
サンプル CredentialsRequest オブジェクト

```
apiVersion: cloudcredential.openshift.io/v1
kind: CredentialsRequest
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: openshift-image-registry-gcs
  namespace: openshift-cloud-credential-operator
spec:
  secretRef:
    name: installer-cloud-credentials
    namespace: openshift-image-registry
  providerSpec:
    apiVersion: cloudcredential.openshift.io/v1
    kind: GCPProviderSpec
    predefinedRoles:
    - roles/storage.admin
    - roles/iam.serviceAccountUser
    skipServiceCheck: true
```

7. 以前に生成した **openshift-install** マニフェストディレクトリーにシークレットの YAML ファイルを作成します。シークレットは、各 **request.spec.secretRef** に定義される namespace およびシークレット名を使用して保存する必要があります。シークレットデータの形式は、クラウドプロバイダーごとに異なります。

8. クラスターの作成に進みます。

```
$ openshift-install create cluster --dir=mycluster
```



重要

アップグレードを実行する前に、パーミッションが次のリリースで変更された場合には、認証情報の調整が必要になる場合があります。今後は、Cloud Credential Operator は更新されたパーミッションが処理されることを示唆するまでアップグレードできなくなる可能性があります。

4.2.2. 管理者の認証情報のルートシークレット形式

各クラウドプロバイダーは、**kube-system** namespace の認証情報ルートシークレットを使用します。これは、すべての認証情報要求を満たし、それぞれのシークレットを作成するために使用されます。これは、**Mint モード** で新規の認証情報を作成するか、または **Passthrough モード** で認証情報ルートシークレットをコピーして実行します。

シークレットの形式はクラウドごとに異なり、それぞれの **CredentialsRequest** シークレットにも使用されます。

Google Cloud Platform (GCP) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <ServiceAccount>
```

4.2.2.1. アップグレード

今後のリリースでは、Cloud Credential Operator の改善により、手動でメンテナーンスされる認証情報が今後のリリースのイメージの **CredentialsRequest** オブジェクトに一致するように更新されていないことが原因でアップグレードが失敗する可能性が生じる状況を避けることができます。

4.2.3. mint モード

mint モードは AWS、GCP および Azure でサポートされます。

OpenShift Container Platform を実行するためのデフォルトおよび推奨されるベストプラクティスとして、管理者レベルのクラウド認証情報を使用してインストーラーを実行できます。**admin** 認証情報は **kube-system** namespace に保存され、次に Cloud Credential Operator によってクラスターの **CredentialsRequest** オブジェクトを処理し、特定のパーミッションでそれぞれの新規ユーザーを作成するために使用されます。

mint モードには以下の利点があります。

- 各クラスターコンポーネントにはそれぞれが必要なパーミッションのみがあります。
- アップグレードを含むクラウド認証情報の自動の継続的な調整が行われます。これには、追加の認証情報またはパーミッションが必要になる可能性があります。

1 つの不利な点として、mint モードでは、**admin** 認証情報がクラスターの **kube-system** シークレットに保存される必要があります。

4.3. GCP へのクラスターのクイックインストール

OpenShift Container Platform バージョン 4.5 では、デフォルトの設定オプションを使用してクラスターを Google Cloud Platform (GCP) にインストールできます。

4.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- GCP アカウントを設定 してクラスターをホストします。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

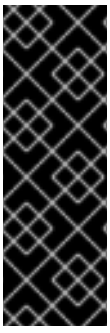
4.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

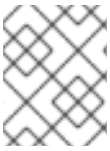
4.3.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.3.4. インストールプログラムの取得

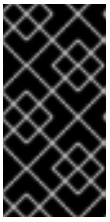
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

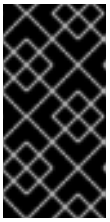
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

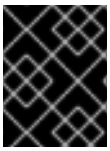
3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.3.5. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2

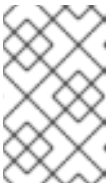
異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **gcp** を選択します。

c. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

d. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

e. クラスターをデプロイするリージョンを選択します。

f. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

g. クラスターの記述名を入力します。7 文字以上の名前を指定すると、クラスター名から生成されるインフラストラクチャー ID で最初の 6 文字のみが使用されます。

h. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

4.3.6. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.3.6.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.3.6.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.3.6.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.3.7. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.3.8. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.4. カスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストールプログラムが Google Cloud Platform (GCP) にプロビジョニングするインフラストラクチャーにカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- GCP アカウントを設定 してクラスターをホストします。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

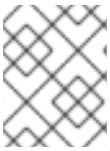
4.4.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.4.4. インストールプログラムの取得

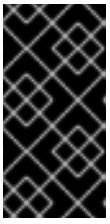
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.4.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.4.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.4.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.4 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.4.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.5 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

パラメーター	説明	値
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.4.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.6 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

4.4.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表4.7 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。

パラメーター	説明	値
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

4.4.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
pullSecret: '{"auths": ...}' 11
fips: false 12
sshKey: ssh-ed25519 AAAA... 13
```

1 8 9 10 11 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

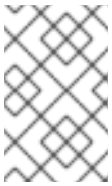
4 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

12 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

13 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.4.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

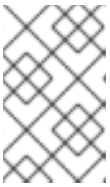
手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

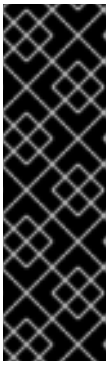
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

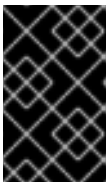
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

4.4.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.4.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
   PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.4.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.4.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.4.8. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターに

ログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.4.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.5. ネットワークのカスタマイズによる GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストールプログラムが Google Cloud Platform (GCP) にプロビジョニングするインフラストラクチャーにカスタマイズされたネットワーク設定でクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

4.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- GCP アカウントを設定 してクラスターをホストします。

- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

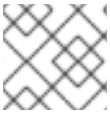
- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

4.5.3. SSH プライベートキーの生成およびエージェントへの追加

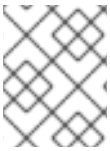
クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

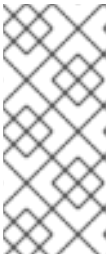
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.5.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

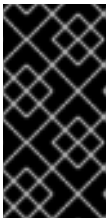
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.5.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

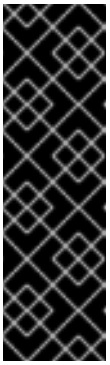
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.5.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.5.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.8 必須パラメーター

パラメーター	説明	値

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.5.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.9 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

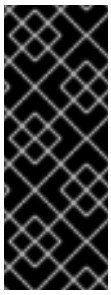
networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

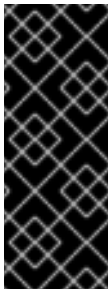
パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.5.5.1.3. オプションの設定パラメーター

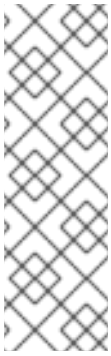
オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.10 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

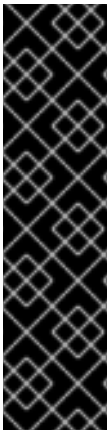
パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

4.5.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表4.11 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。



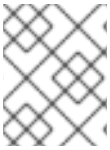
重要

Open Virtual Networking (OVN) Kubernetes ネットワークプラグインは、テクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

OVN テクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/articles/4380121 を参照してください。

4.5.5.2. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。



注記

インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

表4.12 必要なネットワークパラメーター

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。**OVNKubernetes** プラグインは、OpenShift Container Platform 4.5 でテクノロジープレビューとしてご利用いただけます。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの **serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。

パラメーター	説明	値
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

4.5.5.3. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster (8)
networking: (9)
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
```

```
platform:
  gcp:
    projectID: openshift-production 10
    region: us-central1 11
pullSecret: '{"auths": ...}' 12
fips: false 13
sshKey: ssh-ed25519 AAAA... 14
```

1 8 10 11 12 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 5 9 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

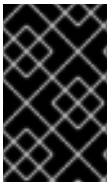
14 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.5.6. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
```

```
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

4.5.7. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
    ...
  kubeProxyConfig: 4
    iptablesSyncPeriod: 30s 5
    proxyArguments:
      iptables-min-sync-period: 6
      - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用し

ない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。

注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time で説明されています。

4.5.7.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

1 **install-config.yaml** ファイルに指定されます。

2 OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

3 OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

4 VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

5 すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

4.5.7.2. OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは OVN-Kubernetes デフォルト CNI ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OVNKubernetes 1
  ovnKubernetesConfig: 2
    mtu: 1400 3
    genevePort: 6081 4
```

1. **install-config.yaml** ファイルに指定されます。
2. OVN-Kubernetes 設定の一部を上書きする必要がある場合にのみ指定します。
3. Geneve (Generic Network Virtualization Encapsulation) オーバーレイネットワークの MTU (maximum transmission unit)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

   自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

   クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **100** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1400** に設定する必要があります。
4. Geneve オーバーレイネットワークの UDP ポート。

4.5.7.3. Cluster Network Operator の設定例

以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
```

```
    mtu: 1450
    vxlanPort: 4789
kubeProxyConfig:
  iptablesSyncPeriod: 30s
  proxyArguments:
    iptables-min-sync-period:
    - 0s
```

4.5.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

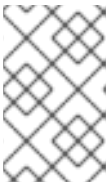
- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

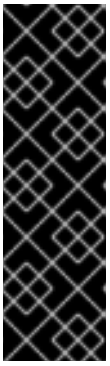
1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

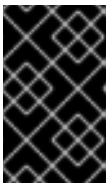
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

4.5.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.5.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.5.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

   ```
   C:\> path
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.5.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.5.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

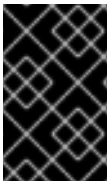
```
system:admin
```

4.5.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.6. ネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform 4.5 では、既存の Google Virtual Private Cloud (VPC) にインストールリリースコンテンツの内部ミラーを作成することで、制限されたネットワークの Google Cloud Platform (GCP) にクラスターをインストールできます。



重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットアクセスが必要になります。

4.6.1. 前提条件

- 非接続インストールのイメージのミラーリング をレジストリーに対して行っており、使用しているバージョンの OpenShift Container Platform の **imageContentSources** データを取得している。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了することができます。

- GCP に既存の VPC がある。インストーラーでプロビジョニングされるインフラストラクチャーを使用するネットワークが制限された環境にクラスターをインストールする場合は、インストーラーでプロビジョニングされる VPC を使用することはできません。以下の要件のいずれかを満たすユーザーによってプロビジョニングされる VPC を使用する必要があります。
  - ミラーレジストリーが含まれる。
  - 別の場所でホストされるミラーレジストリーにアクセスするためのファイアウォールルールまたはピアリング接続がある。
- OpenShift Container Platform のインストールおよび更新 プロセスの詳細を確認している。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.6.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

4.6.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。

- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

4.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

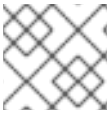
クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

追加リソース

- Telemetry サービスの詳細は、リモートヘルスモニターリング を参照してください。

4.6.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.6.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。ネットワークが制限されたインストールでは、これらのファイルが bastion ホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<bastion_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

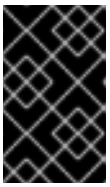
d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

3. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

4. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

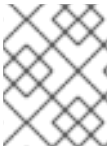


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

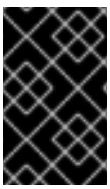
4.6.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.6.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.13 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.6.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.14 ネットワークパラメーター

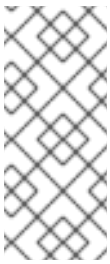
パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

4.6.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.15 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

4.6.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表4.16 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

4.6.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production (9)
    region: us-central1 (10)
    network: existing_vpc (11)
    controlPlaneSubnet: control_plane_subnet (12)
    computeSubnet: compute_subnet (13)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (14)
fips: false (15)
sshKey: ssh-ed25519 AAAA... (16)
additionalTrustBundle: | (17)
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
imageContentSources: (18)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

■

1 8 9 10 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

11 既存 VPC の名前を指定します。

12 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

13 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

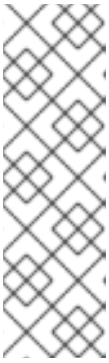
18 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

4.6.5.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前

ワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に . を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

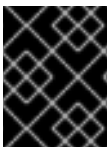
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.6.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数

- ~/.gcp/osServiceAccount.json ファイル
- **gcloud cli** デフォルト認証情報

2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

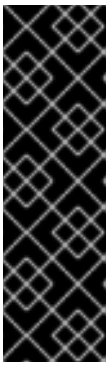
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

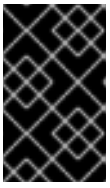
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

4.6.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.6.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.6.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。

PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.6.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.6.8. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.6.9. 次のステップ

- クラスターのカスタマイズ
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.7. GCP のクラスターの既存 VPC へのインストール

OpenShift Container Platform バージョン 4.5 では、クラスターを Google Cloud Platform (GCP) の既存の Virtual Private Cloud (VPC) にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.7.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- GCP アカウントを設定 してクラスターをホストします。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.7.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラス

ターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

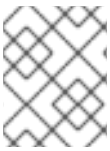
4.7.3. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.7.4. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

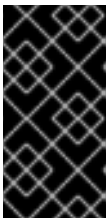
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.7.5. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

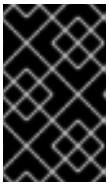
インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. クラスターの記述名を入力します。

viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

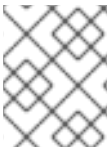


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

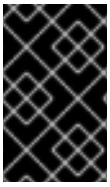
4.7.5.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.7.5.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.17 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

4.7.5.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.18 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

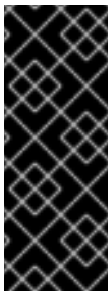
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

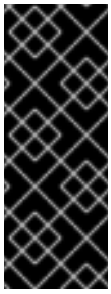
4.7.5.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.19 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

4.7.5.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表4.20 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

4.7.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production (9)
    region: us-central1 (10)
    network: existing_vpc (11)
    controlPlaneSubnet: control_plane_subnet (12)
    computeSubnet: compute_subnet (13)
pullSecret: '{"auths": ...}' (14)
fips: false (15)
sshKey: ssh-ed25519 AAAA... (16)
```

(1) (8) (9) (10) (14) 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

(2) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (6) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義

をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

11 既存 VPC の名前を指定します。

12 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

13 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

4.7.5.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

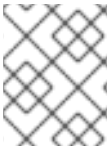
4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.7.6. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

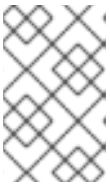
手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

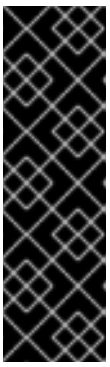
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

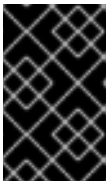
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

4.7.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.7.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.7.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.7.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.7.8. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.7.9. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.8. GCP へのプライベートクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、プライベートクラスターを Google Cloud Platform (GCP) の既存の VPC にインストールできます。インストールプログラムは、カスタマイズ可能な残りの必要なインフラストラクチャーをプロビジョニングします。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

4.8.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- GCP アカウントを設定 してクラスターをホストします。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.8.2. プライベートクラスター

お使いの環境で外部のインターネット接続を必要としない場合には、外部エンドポイントを公開しないプライベート OpenShift Container Platform クラスターをデプロイすることができます。プライベートクラスターは内部ネットワークからのみアクセス可能で、インターネット上では表示されません。

デフォルトで、OpenShift Container Platform はパブリックにアクセス可能な DNS およびエンドポイントを使用できるようにプロビジョニングされます。プライベートクラスターは、クラスターのデプロイ時に DNS、Ingress コントローラー、および API サーバーを private に設定します。つまり、クラスターリソースは内部ネットワークからのみアクセスでき、インターネット上では表示されません。

プライベートクラスターをデプロイするには、要件を満たす既存のネットワークを使用する必要があります。クラスターリソースはネットワーク上の他のクラスター間で共有される可能性があります。

さらに、プロビジョニングするクラウドの API サービスにアクセスできるマシンから、プロビジョニングするネットワーク上のホストおよびインストールメディアを取得するために使用するインターネットにプライベートクラスターをデプロイする必要があります。これらのアクセス要件を満たし、所属する会社のガイドラインに準拠したすべてのマシンを使用することができます。たとえば、このマシンには、クラウドネットワーク上の bastion ホスト、または VPN 経由でネットワークにアクセスできるマシンを使用できます。

4.8.2.1. GCP のプライベートクラスター

Google Cloud Platform (GCP) にプライベートクラスターを作成するには、クラスターをホストするために既存のプライベート VPC およびサブネットを指定する必要があります。インストールプログラムは、クラスターが必要とする DNS レコードを解決できる必要もあります。インストールプログラムは、内部トラフィック用としてのみ Ingress Operator および API サーバーを設定します。

クラスターには、GCP API にアクセスするためにインターネットへのアクセスが依然として必要になります。

以下のアイテムは、プライベートクラスターのインストール時に必要ではなく、作成されません。

- パブリックサブネット

- パブリック Ingress をサポートするパブリックネットワークロードバランサー
- クラスターの **baseDomain** に一致するパブリック DNS ゾーン

インストールプログラムは、プライベート DNS ゾーンおよびクラスターに必要なレコードを作成するために指定する **baseDomain** を使用します。クラスターは、Operator がクラスターのパブリックレコードを作成せず、すべてのクラスターマシンが指定するプライベートサブネットに配置されるように設定されます。

ソースタグに基づいて外部ロードバランサーへのアクセスを制限できないため、プライベートクラスターは内部ロードバランサーのみを使用して内部インスタンスへのアクセスを許可します。

内部ロードバランサーは、ネットワークロードバランサーが使用するターゲットプールではなく、インスタンスグループに依存します。インストールプログラムは、グループにインスタンスがない場合でも、各ゾーンのインスタンスグループを作成します。

- クラスター IP アドレスは内部のみで使用されます。
- 1 つの転送ルールが Kubernetes API およびマシン設定サーバーポートの両方を管理します。
- バックエンドサービスは各ゾーンのインスタンスグループ、および存在する場合はブートストラップインスタンスグループで設定されます。
- ファイアウォールは、内部のソース範囲のみに基づく単一ルールを使用します。

4.8.2.1.1. 制限事項

ロードバランサーの機能の違いにより、マシン設定サーバー **/healthz** のヘルスチェックは実行されません。2 つの内部ロードバランサーが 1 つの IP アドレスを共有できませんが、2 つのネットワークロードバランサーは 1 つの外部 IP アドレスを共有できます。インスタンスが健全であるかどうかについては、ポート 6443 の **/readyz** チェックで完全に判別されます。

4.8.3. カスタム VPC の使用について

OpenShift Container Platform 4.5 では、クラスターを Google Cloud Platform (GCP) の既存の VPC にデプロイできます。これを実行する場合、VPC 内の既存のサブネットおよびルーティングルールも使用する必要があります。

OpenShift Container Platform を既存の GCP VPC にデプロイすると、新規アカウントの制限を回避したり、会社のガイドラインによる運用上の制約をより容易に遵守することが可能になる場合があります。VPC の作成に必要なインフラストラクチャーの作成パーミッションを取得できない場合には、このオプションを使用できます。

4.8.3.1. VPC を使用するための要件

インストールプログラムは、以下のコンポーネントを作成しなくなります。

- VPC
- サブネット
- Cloud Router
- Cloud NAT
- NAT IP アドレス

カスタム VPC を使用する場合は、そのカスタム VPC と使用するインストールプログラムおよびクラスターのサブネットを適切に設定する必要があります。インストールプログラムは、使用するクラスターのネットワーク範囲を細分化できず、サブネットのルートテーブルを設定するか、または DHCP などの VPC オプションを設定します。これは、クラスターのインストール前に設定する必要があります。

VPC およびサブネットは以下の要件を満たす必要があります。

- VPC は、OpenShift Container Platform クラスターをデプロイする同じ GCP プロジェクトに存在する必要があります。
- コントロールプレーンおよびコンピュートマシンからインターネットにアクセスできるようにするには、サブネットで Cloud NAT を設定してこれに対する egress を許可する必要があります。これらのマシンにパブリックアドレスがありません。インターネットへのアクセスが必要ない場合でも、インストールプログラムおよびイメージを取得できるように VPC ネットワークに対して egress を許可する必要があります。複数の Cloud NAT を共有サブネットで設定できないため、インストールプログラムはこれを設定できません。

指定するサブネットが適切であることを確認するには、インストールプログラムが以下のデータを確認します。

- 指定するすべてのサブネットが存在し、指定した VPC に属します。
- サブネットの CIDR はマシン CIDR に属します。
- クラスターのコントロールプレーンおよびコンピュートマシンをデプロイするためにサブネットを指定する必要があります。両方のマシンタイプに同じサブネットを使用できます。

既存の VPC を使用するクラスターを破棄しても、VPC は削除されません。

4.8.3.2. パーミッションの区分

OpenShift Container Platform 4.3 以降、クラスターのデプロイに、インストールプログラムがプロビジョニングするインフラストラクチャークラスターに必要なすべてのパーミッションを必要としなくなりました。この変更は、ある会社で個人がクラウドで他とは異なるリソースを作成できるようにパーミッションが区分された状態に類似するものです。たとえば、インスタンス、バケット、ロードバランサーなどのアプリケーション固有のアイテムを作成することはできますが、VPC、サブネット、または Ingress ルールなどのネットワーク関連のコンポーネントは作成できない可能性があります。

クラスターの作成時に使用する GCP の認証情報には、VPC、およびサブネット、ルーティングテーブル、インターネットゲートウェイ、NAT、VPN などの VPC 内のコアとなるネットワークコンポーネントの作成に必要なネットワークのパーミッションは必要ありません。ロードバランサー、セキュリティーグループ、ストレージおよびノードなどの、クラスター内でマシンに必要なアプリケーションリソースを作成するパーミッションは依然として必要になります。

4.8.3.3. クラスター間の分離

OpenShift Container Platform を既存ネットワークにデプロイする場合、クラスターサービスの分離は、クラスターのインフラストラクチャー ID によるクラスター内のマシンを参照するファイアウォールルールによって保持されます。クラスター内のトラフィックのみが許可されます。

複数のクラスターを同じ VPC にデプロイする場合、以下のコンポーネントはクラスター間のアクセスを共有する可能性があります。

- API: 外部公開ストラテジーでグローバルに利用可能か、または内部公開ストラテジーのネットワーク全体で利用できる。

- デバッグツール: SSH および ICMP アクセス用にマシン CIDR に対して開かれている仮想マシンインスタンス上のポートなど。

4.8.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

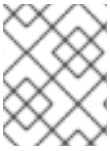
4.8.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

4.8.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

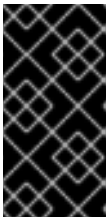
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

4.8.7. インストール設定ファイルの手動作成

内部ネットワークからのみアクセスでき、インターネット上に表示されないプライベート OpenShift Container Platform クラスターのインストールの場合、インストール設定ファイルを手動で生成する必要があります。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

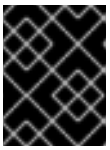
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

4.8.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際

に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

4.8.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表4.21 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

4.8.7.1.2. ネットワーク設定パラメーター

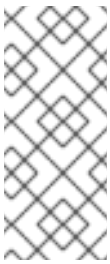
既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表4.22 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

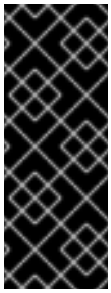
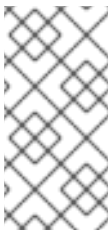
4.8.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表4.23 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

4.8.7.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表4.24 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。

パラメーター	説明	値
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

4.8.7.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2) (3)
  hyperthreading: Enabled (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5) (6)
- hyperthreading: Enabled (7)
  name: worker
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
metadata:
  name: test-cluster (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production 9
    region: us-central1 10
    network: existing_vpc 11
    controlPlaneSubnet: control_plane_subnet 12
    computeSubnet: compute_subnet 13
pullSecret: '{"auths": ...}' 14
fips: false 15
sshKey: ssh-ed25519 AAAA... 16
publish: Internal 17
```

1 8 9 10 14 必須。インストールプログラムはこの値の入力を求めるプロンプトを出します。

2 5 これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

3 6 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

4 7 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

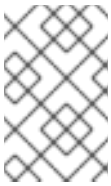
11 既存 VPC の名前を指定します。

12 コントロールプレーンマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

13 コンピュートマシンをデプロイする既存サブネットの名前を指定します。サブネットは、指定した VPC に属している必要があります。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

4.8.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
```

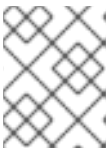
```
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。***** を使用し、すべての宛先のプロキシーをバイパスします。

4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

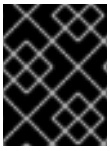
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.8.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

4.8.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.8.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.8.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。

PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.8.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.8.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.8.11. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.9. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、プロビジョニングするインフラストラクチャーを使用するクラスターを Google Cloud Platform (GCP) にインストールできます。

以下に、ユーザーによってプロビジョニングされるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

4.9.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

4.9.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

4.9.3. GCP プロジェクトの設定

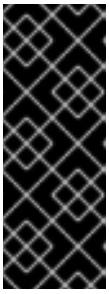
OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

4.9.3.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。



重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

4.9.3.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表4.25 必要な API サービス

API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

4.9.3.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

4.9.3.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表4.26 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

4.9.3.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

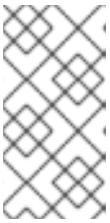
前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。

2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。



注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

4.9.3.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表4.27 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

4.9.3.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.9.3.7. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

4.9.4. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

4.9.4.1. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

      v. クラスターをデプロイするリージョンを選択します。

      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

      vii. クラスターの記述名を入力します。

      viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

   c. オプション: クラスターでコンピュートマシンをプロビジョニングするよう設定する必要がない場合は、 **install-config.yaml** ファイルで **compute** プールの **replicas** を **0** に設定してコンピュートプールを空にします。

   ```
   compute:
   ```

```
- hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 0 1
```

1 **0** に設定します。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

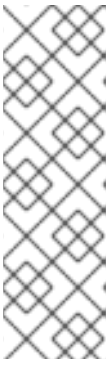
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.9.4.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
```

```
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

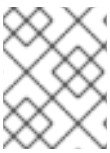
4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.9.4.3. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

   これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション:Ingress DNS レコードの追加

4.9.5. 一般的な変数のエクスポート

4.9.5.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

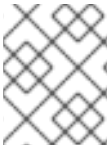
出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

4.9.5.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。



注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

4.9.6. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
```

```
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/19**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.32.0/19**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

4.9.6.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例4.1 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
```

```
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

4.9.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表4.28 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn

プロトコル	ポート	説明
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

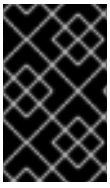
表4.29 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表4.30 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

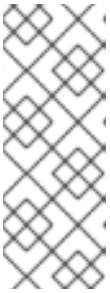
注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.31 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.32 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

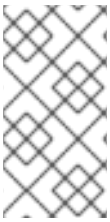
クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

4.9.8. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブ

ジェクトについて記述しています。

3. デプロイメントテンプレートが使用する変数をエクスポートします。

   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

4 **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

5 **control_subnet** は、コントロールサブセットの URL です。

6 **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

4.9.8.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例4.2 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
```

```
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

4.9.8.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例4.3 02_lb_int.py Deployment Manager テンプレート

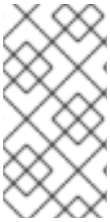
```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
```

```
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

4.9.9. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。
   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

4.9.9.1. プライベート DNS の Deployment Manager テンプレート

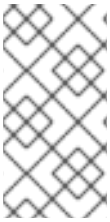
以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例4.4 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

4.9.10. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

4.9.10.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例4.5 03_firewall.py Deployment Manager テンプレート

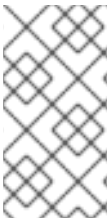
```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
```

```
            'ports': ['30000-32767']
        },{
            'IPProtocol': 'udp',
            'ports': ['30000-32767']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}]

return {'resources': resources}
```

4.9.11. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
```

```
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

4.9.11.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例4.6 03_iam.py Deployment Manager テンプレート

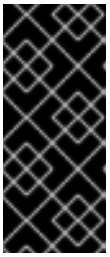
```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

4.9.12. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

4.9.13. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。

(4) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(5) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(6) **image** は RHCOS イメージの **selfLink** URL です。

(7) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(8) **root_volume_size** はブートストラップマシンのブートディスクサイズです。

(9) **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。

   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

   ```
   $ gcloud compute instance-groups unmanaged add-instances \
     ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
   bootstrap
   ```

   b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

   ```
   $ gcloud compute backend-services add-backend \
     ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
   group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
   ```

4.9.13.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例4.7 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '","verification":{}}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":
```

```
{},"systemd":{}}',
                ]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

4.9.14. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)

    ignition: '${MASTER_IGNITION}' (7)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。

(3) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(4) **image** は RHCOS イメージの **selfLink** URL です。

5 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

6 **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。

7 **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.9.14.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例4.8 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-m-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
```

```
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][0]
    }
}, {
    'name': context.properties['infra_id'] + '-m-1',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
```

```
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-m-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

4.9.15. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

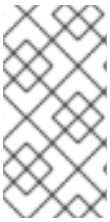
2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.9.16. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。



注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. サービスアカウントのメールアドレスをエクスポートします。

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
```

```
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

1 **name** はワーカーマシンの名前です (例: **worker-0**)。

2 9 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3 10 **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

4 11 **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

5 12 **image** は RHCOS イメージの **selfLink** URL です。

6 13 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

7 14 **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

8 15 **Ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

4.9.16.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例4.9 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
```

```
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

4.9.17. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.9.17.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.9.17.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.9.17.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。

3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。

4. アーカイブを展開し、解凍します。

5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.9.18. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.9.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があ

ります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

4.9.20. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.*** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer   172.30.18.154   35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. A レコードをゾーンに追加します。

- A レコードを使用するには、以下を実行します。

i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.9.21. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. クラスターの稼働状態を確認します。
   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version             False       True          24m     Working towards 4.5.4: 99% complete
```

   b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS    RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running   0          35m
kube-system                                etcd-member-ip-10-0-3-239.us-east-
2.compute.internal             1/1     Running   0          37m
kube-system                                etcd-member-ip-10-0-3-24.us-east-
2.compute.internal             1/1     Running   0          35m
openshift-apiserver-operator               openshift-apiserver-operator-6d6674f4f4-
h7t2t                          1/1     Running   1          37m
openshift-apiserver                        apiserver-fm48r
```

```
1/1     Running   0       30m
openshift-apiserver                                apiserver-fxkvv
1/1     Running   0       29m
openshift-apiserver                                apiserver-q85nm
1/1     Running   0       29m
...
openshift-service-ca-operator                      openshift-service-ca-operator-66ff6dc6cd-
9r257                1/1     Running   0       37m
openshift-service-ca                               apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1     Running   0       35m
openshift-service-ca                               configmap-cabundle-injector-8498544d7-
25qn6                1/1     Running   0       35m
openshift-service-ca                               service-serving-cert-signer-6445fc9c6-wqdqn
1/1     Running   0       35m
openshift-service-catalog-apiserver-operator       openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w     1/1     Running   0       32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm   1/1     Running   0       31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

4.9.22. 次のステップ

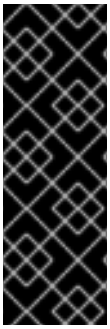
- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.10. DEPLOYMENT MANAGER テンプレートの使用による GCP でのユーザーによってプロビジョニングされるインフラストラクチャーへの共有 VPC のあるクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、プロビジョニングするインフラストラクチャーを使用するクラスターを Google Cloud Platform (GCP) の共有 VPC (Virtual Private Clous) にインストールできます。この場合、共有 VPC にインストールされたクラスターは、クラスターがデプロイされる場所とは異なるプロジェクトから VPC を使用するように設定されるクラスターです。

共有 VPC により、組織は複数のプロジェクトから共通の VPC ネットワークにリソースを接続できるようになります。対象のネットワークの内部 IP を使用して、組織内の通信を安全かつ効率的に実行できます。共有 VPC の詳細は、GCP ドキュメントの Shared VPC overview を参照してください。

以下に、ユーザーによって提供されるインフラストラクチャーの共有 VPC へのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

4.10.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

4.10.2. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

4.10.3. クラスターをホストする GCP プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

4.10.3.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

4.10.3.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表4.33 必要な API サービス

API サービス	コンソールサービス名
Cloud Deployment Manager V2 API	**deploymentmanager.googleapis.com**
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

4.10.3.3. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表4.34 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**

- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

4.10.3.4. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
   クラスターを作成するには、サービスアカウントキーが必要になります。

4.10.3.4.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表4.35 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

4.10.3.5. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)
- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.10.3.6. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**

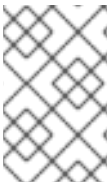
- **gsutil**

GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。

2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

4.10.4. 共有 VPC ネットワークをホストする GCP プロジェクトの設定

共有 VPC (Virtual Private Cloud) を使用して Google Cloud Platform (GCP) で OpenShift Container Platform クラスターをホストする場合、これをホストするプロジェクトを設定する必要があります。



注記

共有 VPC ネットワークをホストするプロジェクトがすでにある場合は、本セクションを参照して、プロジェクトが OpenShift Container Platform クラスターのインストールに必要なすべての要件を満たすことを確認します。

手順

1. OpenShift Container Platform クラスターの共有 VPC をホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。
2. 共有 VPC をホストするプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。
3. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

共有 VPC ネットワークをホストするプロジェクトのサービスアカウントには以下のロールが必要です。

- コンピュートネットワークユーザー
- コンピュートセキュリティー管理者
- Deployment Manager Editor
- DNS 管理者
- セキュリティー管理者
- ネットワーク管理者

4.10.4.1. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカ

ウントに、クラスターをインストールする共有 VPC をホストするプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。

注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

4.10.4.2. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。
   a. コントロールプレーンの CIDR をエクスポートします。

```
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
```

   b. コンピュート CIDR をエクスポートします。

```
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'
```

   c. VPC ネットワークおよびクラスターをデプロイするリージョンを以下にエクスポートします。

```
$ export REGION='<region>'
```

3. 共有 VPC をホストするプロジェクトの ID の変数をエクスポートします。

```
$ export HOST_PROJECT=<host_project>
```

4. ホストプロジェクトに属するサービスアカウントのメールの変数をエクスポートします。

```
$ export HOST_PROJECT_ACCOUNT=<host_service_account_email>
```

5. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '<prefix>' (1)
    region: '${REGION}' (2)
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' (3)
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' (4)
EOF
```

(1) **infra_id** は、ネットワーク名の接頭辞です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/19**)。

(4) **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.32.0/19**)。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create <vpc_deployment_name> --config
01_vpc.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} 1
```

1 **<vpc_deployment_name>** には、デプロイする VPC の名前を指定します。

7. 他のコンポーネントが必要とする VPC 変数をエクスポートします。
   a. ホストプロジェクトネットワークの名前をエクスポートします。

```
$ export HOST_PROJECT_NETWORK=<vpc_network>
```

   b. ホストプロジェクトのコントロールプレーンのサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_CONTROL_SUBNET=<control_plane_subnet>
```

   c. ホストプロジェクトのコンピュートサブネットの名前をエクスポートします。

```
$ export HOST_PROJECT_COMPUTE_SUBNET=<compute_subnet>
```

8. 共有 VPC を設定します。GCP ドキュメントの 共有 VPC の設定 を参照してください。

4.10.4.2.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例4.10 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
```

```
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

4.10.5. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

4.10.5.1. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

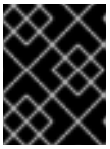
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

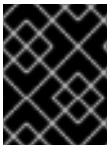


重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

4.10.5.2. GCP のカスタマイズされた install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。



重要

このサンプルの YAML ファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得し、これを変更する必要があります。

```
apiVersion: v1
baseDomain: example.com (1)
controlPlane: (2)
  hyperthreading: Enabled (3) (4)
  name: master
  platform:
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 3
compute: (5)
- hyperthreading: Enabled (6)
  name: worker
  platform:
```

```
    gcp:
      type: n2-standard-4
      zones:
      - us-central1-a
      - us-central1-c
  replicas: 0
metadata:
  name: test-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  gcp:
    projectID: openshift-production (7)
    region: us-central1 (8)
pullSecret: '{"auths": ...}'
fips: false (9)
sshKey: ssh-ed25519 AAAA... (10)
publish: Internal (11)
```

(1) ホストプロジェクトでパブリック DNS を指定します。

(2) (5) これらのパラメーターおよび値を指定しない場合、インストールプログラムはデフォルトの値を指定します。

(3) (6) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(4) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンに対して **n1-standard-8** などの大規模なマシンタイプを使用します。

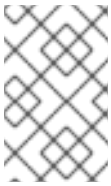
(7) 仮想マシンインスタンスが存在するメインのプロジェクトを指定します。

(8) VPC ネットワークが置かれているリージョンを指定します。

(9) FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされませ

ん。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

10 クラスター内のマシンにアクセスするために使用する **sshKey** 値をオプションで指定できます。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

11 クラスターのユーザーに表示されるエンドポイントをパブリッシュする方法。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。独自にプロビジョニングするインフラストラクチャーを使用するクラスターで共有 VPC を使用するには、**publish** を **Internal** に設定する必要があります。インストールプログラムは、ホストプロジェクトのベースドメインのパブリック DNS ゾーンにアクセスできなくなります。

4.10.5.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
```

```
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

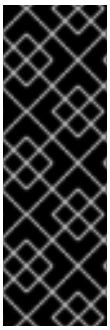
インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

4.10.5.4. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. **<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
status: {}
```

   1 このセクションを完全に削除します。

6. VPC のクラウドプロバイダーを設定します。

   a. **<installation_directory>/manifests/cloud-provider-config.yaml** ファイルを開きます。

   b. **network-project-id** パラメーターを追加し、その値を共有 VPC ネットワークをホストするプロジェクトの ID に設定します。

   c. **network-name** パラメーターを追加し、その値を OpenShift Container Platform クラスターをホストする共有 VPC ネットワークの名前に設定します。

   d. **subnetwork-name** パラメーターの値を、コンピュートマシンをホストする共有 VPC サブネットの値に置き換えます。

   <installation_directory>/manifests/cloud-provider-config.yaml の内容は以下の例のようになります。

```
config: |+
  [global]
  project-id      = example-project
  regional        = true
  multizone       = true
  node-tags       = opensh-ptzzx-master
  node-tags       = opensh-ptzzx-worker
  node-instance-prefix = opensh-ptzzx
  external-instance-groups-prefix = opensh-ptzzx
  network-project-id = example-shared-vpc
  network-name    = example-network
  subnetwork-name = example-worker-subnet
```

7. プライベートネットワーク上にないクラスターをデプロイする場合は、**<installation_directory>/manifests/cluster-ingress-default-ingresscontroller.yaml** ファイルを開き、**scope** パラメーターの値を **External** に置き換えます。ファイルの内容は以下の例のようになります。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  creationTimestamp: null
  name: default
  namespace: openshift-ingress-operator
spec:
  endpointPublishingStrategy:
    loadBalancer:
      scope: External
    type: LoadBalancerService
status:
  availableReplicas: 0
  domain: ''
  selector: ''
```

8. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

4.10.6. 一般的な変数のエクスポート

4.10.6.1. インフラストラクチャー名の抽出

関連情報

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

4.10.6.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>' 1
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>' 2
$ export NETWORK_CIDR='10.0.0.0/16'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 3
```

```
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
```

1 2 ホストプロジェクトの値を指定します。

3 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

4.10.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表4.36 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表4.37 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表4.38 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

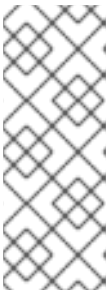
API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.39 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

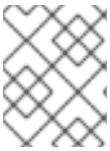
   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.40 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

4.10.8. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブジェクトについて記述しています。
3. デプロイメントテンプレートが使用する変数をエクスポートします。
   a. クラスターネットワークの場所をエクスポートします。

```
$ export CLUSTER_NETWORK=(`gcloud compute networks describe
${HOST_PROJECT_NETWORK} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT} --format json | jq -r .selfLink`)
```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

```
$ export CONTROL_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_CONTROL_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

```
$ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[0] | cut -d "/" -f9`)
```

```
$ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[1] | cut -d "/" -f9`)
```

```
$ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r
.zones[2] | cut -d "/" -f9`)
```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(4) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(5) **control_subnet** は、コントロールサブセットの URL です。

(6) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --
region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-
public-ip --region=${REGION} --format json | jq -r .address`)
```

4.10.8.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例4.11 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

4.10.8.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例4.12 02_lb_int.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
```

```
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

4.10.9. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。

2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' 1
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' 2
    cluster_network: '${CLUSTER_NETWORK}' 3
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。
3. **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。

   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

   b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT} dns
record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

4.10.9.1. プライベート DNS の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例4.13 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

4.10.10. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。

2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml --project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

4.10.10.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例4.14 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
```

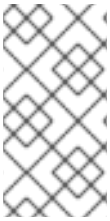
```
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
```

```
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
                'ports': ['30000-32767']
            },{
                'IPProtocol': 'udp',
                'ports': ['30000-32767']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
```

```
    }]

    return {'resources': resources}
```

4.10.11. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
  properties:
    infra_id: '${INFRA_ID}' (1)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コントロールプレーンおよびコンピュートサブネットをホストするサブネットのサービスアカウントに、インストールプログラムが必要とするパーミッションを割り当てます。

   a. 共有 VPC をホストするプロジェクトの **networkViewer** ロールをマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
projects add-iam-policy-binding ${HOST_PROJECT} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role
"roles/compute.networkViewer"
```

   b. **networkUser** ロールをコントロールプレーンサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

   c. **networkUser** ロールをコントロールプレーンサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_CONTROL_SUBNET}" --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

   d. **networkUser** ロールをコンピュートサブネットのマスターサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkUser"
--region ${REGION}
```

   e. **networkUser** ロールをコンピュートサブネットのワーカーサービスアカウントに付与します。

```
$ gcloud --account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
compute networks subnets add-iam-policy-binding
"${HOST_PROJECT_COMPUTE_SUBNET}" --member
```

```
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role
"roles/compute.networkUser" --region ${REGION}
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

4.10.11.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例4.15 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

4.10.12. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

4.10.13. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    zone: '${ZONE_0}' 3

    cluster_network: '${CLUSTER_NETWORK}' 4
    control_subnet: '${CONTROL_SUBNET}' 5
    image: '${CLUSTER_IMAGE}' 6
```

```
    machine_type: 'n1-standard-4' 7
    root_volume_size: '128' 8

    bootstrap_ign: '${BOOTSTRAP_IGN}' 9
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。
4. **cluster_network** はクラスターネットワークの **selfLink** URL です。
5. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
6. **image** は RHCOS イメージの **selfLink** URL です。
7. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
8. **root_volume_size** はブートストラップマシンのブートディスクサイズです。
9. **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-bootstrap-
instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-bootstrap
```

8. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

```
$ gcloud compute backend-services add-backend ${INFRA_ID}-api-internal-backend-service
--region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --instance-
group-zone=${ZONE_0}
```

4.10.13.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例4.16 04_bootstrap.py Deployment Manager テンプレート

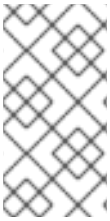
```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-bootstrap-public-ip',
```

```
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '","verification":{}}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":
{},"systemd":{}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
```

```
}]

return {'resources': resources}
```

4.10.14. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' 1
```

```
    zones: 2
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' 3
    image: '${CLUSTER_IMAGE}' 4
    machine_type: 'n1-standard-4' 5
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' 6

    ignition: '${MASTER_IGNITION}' 7
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。
3. **control_subnet** は、コントロールサブセットの **selfLink** URL です。
4. **image** は RHCOS イメージの **selfLink** URL です。
5. **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
6. **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。
7. **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

   - 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

     ```
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
     $ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
     ${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
     ```

   - 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

     ```
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
     $ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
     ```

```
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.10.14.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例4.17 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-m-0',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][0]
        }
    }, {
        'name': context.properties['infra_id'] + '-m-1',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
```

```
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][1]
    }
  }, {
    'name': context.properties['infra_id'] + '-m-2',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
```

```
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

4.10.15. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

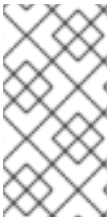
2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.10.16. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。



注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
${HOST_PROJECT_COMPUTE_SUBNET} --region=${REGION} --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT} --format json | jq -r
.selfLink`)
```

b. サービスアカウントのメールアドレスをエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

```
$ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (7)
    ignition: '${WORKER_IGNITION}' (8)
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (9)
    zone: '${ZONE_1}' (10)
    compute_subnet: '${COMPUTE_SUBNET}' (11)
    image: '${CLUSTER_IMAGE}' (12)
    machine_type: 'n1-standard-4' (13)
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' (14)
    ignition: '${WORKER_IGNITION}' (15)
EOF
```

(1) **name** はワーカーマシンの名前です (例: **worker-0**)。

(2) (9) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) (10) **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。

(4) (11) **compute_subnet** はコンピュートサブネットの **selfLink** URL です。

(5) (12) **image** は RHCOS イメージの **selfLink** URL です。

(6) (13) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(7) (14) **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。

8 15 **ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

4.10.16.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例4.18 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-' + context.env['name'],
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
```

```
    }]

    return {'resources': resources}
```

4.10.17. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

4.10.17.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.10.17.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

4.10.17.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

4.10.18. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。

- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.10.19. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

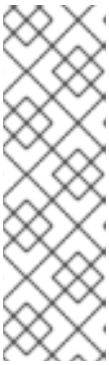
■

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal                        Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal                        Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

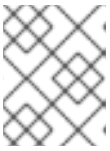
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

4.10.20. Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定が削除されます。Ingress ロードバランサーを参照する DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。

- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default   LoadBalancer   172.30.18.154   35.233.157.184
80:32288/TCP,443:31215/TCP   98
```

2. A レコードをゾーンに追加します。
   - A レコードを使用するには、以下を実行します。
     i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone --project
${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone --
project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
```

     iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME} -
-project ${HOST_PROJECT} --account ${HOST_PROJECT_ACCOUNT}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

```
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME} --project ${HOST_PROJECT} --account
${HOST_PROJECT_ACCOUNT}
```

- ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.10.21. Ingress ファイアウォールルールの追加

クラスターには複数のファイアウォールルールが必要です。共有 VPC を使用しない場合、これらのルールは GCP クラウドプロバイダーを介して Ingress コントローラーによって作成されます。共有 VPC を使用する場合は、現在すべてのサービスのクラスター全体のファイアウォールルールを作成するか、またはクラスターがアクセスを要求する際にイベントに基づいて各ルールを作成できます。クラスターがアクセスを要求する際に各ルールを作成すると、どのファイアウォールルールが必要であるかを正確に把握できます。クラスター全体のファイアウォールルールを作成する場合、同じルールセットを複数のクラスターに適用できます。

イベントに基づいて各ルールを作成する選択をする場合、クラスターをプロビジョニングした後、またはクラスターの有効期間中にコンソールがルールが見つからないことを通知する場合にファイアウォールルールを作成する必要があります。以下のイベントと同様のイベントが表示され、必要なファイアウォールルールを追加する必要があります。

```
$ oc get events -n openshift-ingress --field-selector="reason=LoadBalancerManualChange"
```

出力例

```
Firewall change required by security admin: `gcloud compute firewall-rules create k8s-fw-
a26e631036a3f46cba28f8df67266d55 --network example-network --description "
{\"kubernetes.io/service-name\":\"openshift-ingress/router-default\", \"kubernetes.io/service-
ip\":\"35.237.236.234\"}\" --allow tcp:443,tcp:80 --source-ranges 0.0.0.0/0 --target-tags exampl-fqzq7-
master,exampl-fqzq7-worker --project example-project`
```

これらのルールベースのイベントの作成時に問題が発生した場合には、クラスターの実行中にクラスター全体のファイアウォールルールを設定できます。

4.10.21.1. GCP での共有 VPC のクラスター全体のファイアウォールルールの作成

クラスター全体のファイアウォールルールを作成して、OpenShift Container Platform クラスターに必要なアクセスを許可します。

警告

クラスターイベントに基づいてファイアウォールルールを作成しない場合、クラスター全体のファイアウォールルールを作成する必要があります。

前提条件

- Deployment Manager テンプレートがクラスターをデプロイするために必要な変数をエクスポートしています。
- GCP にクラスターに必要なネットワークおよび負荷分散コンポーネントを作成しています。

手順

1. 単一のファイアウォールルールを追加して、Google Cloud Engine のヘルスチェックがすべてのサービスにアクセスできるようにします。このルールにより、Ingress ロードバランサーはインスタンスのヘルスステータスを判別できます。

```
$ gcloud compute firewall-rules create --allow='tcp:30000-32767,udp:30000-32767' --
network="${CLUSTER_NETWORK}" --source-
ranges='130.211.0.0/22,35.191.0.0/16,209.85.152.0/22,209.85.204.0/22' --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress-hc --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

2. 単一のファイアウォールルールを追加して、すべてのクラスターサービスへのアクセスを許可します。
   - 外部クラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="${CLUSTER_NETWORK}" --source-ranges="0.0.0.0/0" --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

   - プライベートクラスターの場合:

```
$ gcloud compute firewall-rules create --allow='tcp:80,tcp:443' --
network="${CLUSTER_NETWORK}" --source-ranges=${NETWORK_CIDR} --target-
tags="${INFRA_ID}-master,${INFRA_ID}-worker" ${INFRA_ID}-ingress --
account=${HOST_PROJECT_ACCOUNT} --project=${HOST_PROJECT}
```

   このルールでは、TCP ポート **80** および **443** でのトラフィックのみを許可するため、サービスが使用するすべてのポートを追加するようにしてください。

4.10.22. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. クラスターの稼働状態を確認します。
   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version             False       True          24m     Working towards 4.5.4: 99% complete
```

   b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False     7m56s
```

```
cloud-credential                           4.5.4     True        False       False      31m
cluster-autoscaler                         4.5.4     True        False       False      16m
console                                    4.5.4     True        False       False      10m
csi-snapshot-controller                    4.5.4     True        False       False      16m
dns                                        4.5.4     True        False       False      22m
etcd                                       4.5.4     False       False       False      25s
image-registry                             4.5.4     True        False       False      16m
ingress                                    4.5.4     True        False       False      16m
insights                                   4.5.4     True        False       False      17m
kube-apiserver                             4.5.4     True        False       False      19m
kube-controller-manager                    4.5.4     True        False       False      20m
kube-scheduler                             4.5.4     True        False       False      20m
kube-storage-version-migrator              4.5.4     True        False       False      16m
machine-api                                4.5.4     True        False       False      22m
machine-config                             4.5.4     True        False       False      22m
marketplace                                4.5.4     True        False       False      16m
monitoring                                 4.5.4     True        False       False      10m
network                                    4.5.4     True        False       False      23m
node-tuning                                4.5.4     True        False       False      23m
openshift-apiserver                        4.5.4     True        False       False      17m
openshift-controller-manager               4.5.4     True        False       False      15m
openshift-samples                          4.5.4     True        False       False      16m
operator-lifecycle-manager                 4.5.4     True        False       False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False       False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False       False      18m
service-ca                                 4.5.4     True        False       False      23m
service-catalog-apiserver                  4.5.4     True        False       False      23m
service-catalog-controller-manager         4.5.4     True        False       False      23m
storage                                    4.5.4     True        False       False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS      RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
2.compute.internal             1/1     Running     0          35m
kube-system                                etcd-member-ip-10-0-3-239.us-east-
2.compute.internal             1/1     Running     0          37m
kube-system                                etcd-member-ip-10-0-3-24.us-east-
2.compute.internal             1/1     Running     0          35m
openshift-apiserver-operator               openshift-apiserver-operator-6d6674f4f4-
h7t2t                          1/1     Running     1          37m
openshift-apiserver                        apiserver-fm48r
1/1     Running     0          30m
openshift-apiserver                        apiserver-fxkvv
1/1     Running     0          29m
openshift-apiserver                        apiserver-q85nm
1/1     Running     0          29m
...
openshift-service-ca-operator              openshift-service-ca-operator-66ff6dc6cd-
9r257                          1/1     Running     0          37m
```

```
openshift-service-ca                                    apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                                    configmap-cabundle-injector-8498544d7-
25qn6                   1/1       Running     0          35m
openshift-service-ca                                    service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator            openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

4.10.23. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

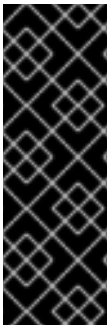
4.11. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での GCP へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、各自でプロビジョニングするインフラストラクチャーおよびインストールリリースコンテンツの内部ミラーを使用して、クラスターを Google Cloud Platform (GCP) にインストールできます

重要

ミラーリングされたインストールリリースのコンテンツを使用して OpenShift Container Platform クラスターをインストールすることは可能ですが、クラスターが GCP API を使用するにはインターネットへのアクセスが必要になります。

以下に、ユーザーによって提供されるインフラストラクチャーのインストールを実行する手順を要約します。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。



重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、クラウドプロバイダーおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。これらの手順を実行するか、独自の手順を作成するのに役立つ複数の Deployment Manager テンプレートが提供されます。他の方法を使用して必要なリソースを作成することもできます。これらのテンプレートはサンプルとしてのみ提供されます。

4.11.1. 前提条件

- ミラーホストでレジストリーを作成 し、OpenShift Container Platform の使用しているバージョン用の **imageContentSources** データを取得します。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了します。

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。他のサイトへのアクセスを付与する必要がある場合もありますが、***.googleapis.com** および **accounts.google.com** へのアクセスを付与する必要があります。
- システムが IAM(アイデンティティーおよびアクセス管理) を管理できない場合、クラスター管理者は IAM 認証情報を手動で作成し、維持 できます。手動モードは、クラウド IAM API に到達できない環境でも使用できます。

4.11.2. GCP プロジェクトの設定

OpenShift Container Platform をインストールする前に、これをホストするように Google Cloud Platform (GCP) プロジェクトを設定する必要があります。

4.11.2.1. GCP プロジェクトの作成

OpenShift Container Platform をインストールするには、クラスターをホストするために Google Cloud Platform (GCP) アカウントでプロジェクトを作成する必要があります。

手順

- OpenShift Container Platform クラスターをホストするプロジェクトを作成します。GCP ドキュメントの プロジェクトの作成と管理 を参照してください。

重要

GCP プロジェクトは、インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合には、Premium Network Service 階層を使用する必要があります。インストールプログラムを使用してインストールしたクラスターでは、Standard Network Service 階層はサポートされません。インストールプログラムは、**api-int.<cluster_name>.<base_domain>** の内部負荷分散を設定します。内部負荷分散には Premium Tier が必要です。

4.11.2.2. GCP での API サービスの有効化

Google Cloud Platform (GCP) プロジェクトでは、OpenShift Container Platform インストールを完了するために複数の API サービスへのアクセスが必要です。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

- クラスターをホストするプロジェクトで以下の必要な API サービスを有効にします。GCP ドキュメントの サービスの有効化 を参照してください。

表4.41 必要な API サービス

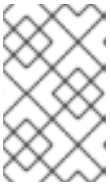
API サービス	コンソールサービス名
Compute Engine API	**compute.googleapis.com**
Google Cloud API	**cloudapis.googleapis.com**
Cloud Resource Manager API	**cloudresourcemanager.googleapis.com**
Google DNS API	**dns.googleapis.com**
IAM Service Account Credentials API	**iamcredentials.googleapis.com**
Identity and Access Management (IAM) API	**iam.googleapis.com**
Service Management API	**servicemanagement.googleapis.com**
Service Usage API	**serviceusage.googleapis.com**
Google Cloud Storage JSON API	**storage-api.googleapis.com**
Cloud Storage	**storage-component.googleapis.com**

4.11.2.3. GCP の DNS の設定

OpenShift Container Platform をインストールするには、使用する Google Cloud Platform (GCP) アカウントに、OpenShift Container Platform クラスターをホストする同じプロジェクトに専用のパブリックホストゾーンがなければなりません。このゾーンはドメインに対する権威を持っている必要があります。DNS サービスは、クラスターへの外部接続のためのクラスターの DNS 解決および名前検索を提供します。

手順

1. ドメイン、またはサブドメイン、およびレジストラーを特定します。既存のドメインおよびレジストラーを移行するか、GCP または他のソースから新規のものを取得できます。



注記

新規ドメインを購入する場合、関連する DNS の変更が伝播するのに時間がかかる場合があります。Google 経由でドメインを購入する方法についての詳細は、Google ドメイン を参照してください。

2. GCP プロジェクトにドメインまたはサブドメインのパブリックホストゾーンを作成します。GCP ドキュメントの ゾーンの管理 を参照してください。
openshiftcorp.com などのルートドメインや、 **clusters.openshiftcorp.com** などのサブドメインを使用します。

3. ホストゾーンレコードから新規の権威ネームサーバーを抽出します。GCP ドキュメントの Cloud DNS ネームサーバーを検索する を参照してください。
通常は、4 つのネームサーバーがあります。

4. ドメインが使用するネームサーバーのレジストラーレコードを更新します。たとえば、ドメインを Google ドメインに登録している場合は、Google Domains Help で How to switch to custom name servers のトピックを参照してください。

5. ルートドメインを Google Cloud DNS に移行している場合は、DNS レコードを移行します。GCP ドキュメントの Cloud DNS への移行 を参照してください。

6. サブドメインを使用する場合は、所属する会社の手順に従ってその委任レコードを親ドメインに追加します。このプロセスには、所属企業の IT 部門や、会社のルートドメインと DNS サービスを制御する部門へのリクエストが含まれる場合があります。

4.11.2.4. GCP アカウントの制限

OpenShift Container Platform クラスターは多くの Google Cloud Platform (GCP) コンポーネントを使用しますが、デフォルトの 割り当て (Quota) はデフォルトの OpenShift Container Platform クラスターをインストールする機能に影響を与えません。

3 つのコンピュートマシンおよび 3 つのコントロールプレーンマシンが含まれるデフォルトクラスターは以下のリソースを使用します。一部のリソースはブートストラッププロセス時にのみ必要となり、クラスターのデプロイ後に削除されることに注意してください。

表4.42 デフォルトのクラスターで使用される GCP リソース

サービス	コンポーネント	場所	必要なリソースの合計	ブートストラップ後に削除されるリソース
サービスアカウント	IAM	グローバル	5	0
ファイアウォールのルール	ネットワーク	グローバル	11	1
転送ルール	コンピュート	グローバル	2	0
ヘルスチェック	コンピュート	グローバル	2	0
イメージ	コンピュート	グローバル	1	0
ネットワーク	ネットワーク	グローバル	1	0
ルーター	ネットワーク	グローバル	1	0
ルート	ネットワーク	グローバル	2	0
サブネットワーク	コンピュート	グローバル	2	0
ターゲットプール	ネットワーク	グローバル	2	0

注記

インストール時にクォータが十分ではない場合、インストールプログラムは超過したクォータとリージョンの両方を示すエラーを表示します。

実際のクラスターサイズ、計画されるクラスターの拡張、およびアカウントに関連付けられた他のクラスターからの使用法を考慮してください。CPU、静的 IP アドレス、および永続ディスク SSD(ストレージ) のクォータは、ほとんどの場合に不十分になる可能性のあるものです。

以下のリージョンのいずれかにクラスターをデプロイする予定の場合、ストレージクォータの最大値を超え、CPU クォータ制限を超える可能性が高くなります。

- **asia-east2**
- **asia-northeast2**
- **asia-south1**
- **australia-southeast1**
- **europe-north1**
- **europe-west2**
- **europe-west3**
- **europe-west6**
- **northamerica-northeast1**
- **southamerica-east1**
- **us-west2**

GCP コンソール からリソースクォータを増やすことは可能ですが、サポートチケットを作成する必要がある場合があります。OpenShift Container Platform クラスターをインストールする前にサポートチケットを解決できるように、クラスターのサイズを早期に計画してください。

4.11.2.5. GCP でのサービスアカウントの作成

OpenShift Container Platform には、Google API でデータにアクセスするための認証および承認を提供する Google Cloud Platform (GCP) サービスアカウントが必要です。プロジェクトに必要なロールが含まれる既存の IAM サービスアカウントがない場合は、これを作成する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。

手順

1. OpenShift Container Platform クラスターをホストするために使用するプロジェクトでサービスアカウントを作成します。GCP ドキュメントで サービスアカウントの作成 を参照してください。

2. サービスアカウントに適切なパーミッションを付与します。付随する個別のパーミッションを付与したり、**オーナー** ロールをこれに割り当てることができます。特定のリソースのサービスアカウントへのロールの付与 を参照してください。

注記

サービスアカウントをプロジェクトの所有者にすることが必要なパーミッションを取得する最も簡単な方法になります。 つまりこれは、サービスアカウントはプロジェクトを完全に制御できることを意味します。この権限を提供することに伴うリスクが受け入れ可能であるかどうかを判別する必要があります。

3. JSON 形式でサービスアカウントキーを作成します。GCP ドキュメントの サービスアカウントキーの作成 を参照してください。
クラスターを作成するには、サービスアカウントキーが必要になります。

4.11.2.5.1. 必要な GCP パーミッション

作成するサービスアカウントに **オーナー** ロールを割り当てると、OpenShift Container Platform のインストールに必要なパーミッションも含め、そのサービスアカウントにすべてのパーミッションが付与されます。OpenShift Container Platform クラスターをデプロイするには、サービスアカウントに以下のパーミッションが必要です。クラスターを既存の VPC にデプロイする場合、サービスアカウントでは一部のネットワークのパーミッションを必要としません。これについては、以下の一覧に記載されています。

インストールプログラムに必要なロール

- Compute 管理者
- セキュリティー管理者
- サービスアカウント管理者
- サービスアカウントユーザー
- ストレージ管理者

インストール時のネットワークリソースの作成に必要なロール

- DNS 管理者

ユーザーによってプロビジョニングされる GCP インフラストラクチャーに必要なロール

- Deployment Manager Editor
- サービスアカウントキー管理者

オプションのロール

クラスターで Operator の制限された認証情報を新たに作成できるようにするには、以下のロールを追加します。

- サービスアカウントキー管理者

ロールは、コントロールプレーンおよびコンピュートマシンが使用するサービスアカウントに適用されます。

表4.43 GCP サービスアカウントのパーミッション

アカウント	ロール
コントロールプレーン	**roles/compute.instanceAdmin**
	roles/compute.networkAdmin
	roles/compute.securityAdmin
	roles/storage.admin
	roles/iam.serviceAccountUser
コンピュート	**roles/compute.viewer**
	roles/storage.admin

4.11.2.6. サポートされている GCP リージョン

OpenShift Container Platform クラスターを以下の Google Cloud Platform (GCP) リージョンにデプロイできます。

- **asia-east1** (Changhua County, Taiwan)
- **asia-east2** (Hong Kong)
- **asia-northeast1** (Tokyo, Japan)
- **asia-northeast2** (Osaka, Japan)
- **asia-south1** (Mumbai, India)
- **asia-southeast1** (Jurong West, Singapore)
- **australia-southeast1** (Sydney, Australia)
- **europe-north1** (Hamina, Finland)
- **europe-west1** (St. Ghislain, Belgium)
- **europe-west2** (London, England, UK)
- **europe-west3** (Frankfurt, Germany)
- **europe-west4** (Eemshaven, Netherlands)
- **europe-west6** (Zürich, Switzerland)
- **northamerica-northeast1** (Montréal, Québec, Canada)
- **southamerica-east1** (São Paulo, Brazil)

- **us-central1** (Council Bluffs, Iowa, USA)
- **us-east1** (Moncks Corner, South Carolina, USA)
- **us-east4** (Ashburn, Northern Virginia, USA)
- **us-west1** (The Dalles, Oregon, USA)
- **us-west2** (Los Angeles, California, USA)

4.11.2.7. GCP の CLI ツールのインストールおよび設定

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して Google Cloud Platform (GCP) に OpenShift Container Platform をインストールするには、GCP の CLI ツールをインストールし、設定する必要があります。

前提条件

- クラスターをホストするプロジェクトを作成しています。
- サービスアカウントを作成し、これに必要なパーミッションを付与しています。

手順

1. **$PATH** で以下のバイナリーをインストールします。
   - **gcloud**
   - **gsutil**

   GCP ドキュメントの Google Cloud SDK のドキュメント を参照してください。
2. 設定したサービスアカウントで、**gcloud** ツールを使用して認証します。
   GCP ドキュメントで、サービスアカウントでの認証 について参照してください。

4.11.3. GCP のインストール設定ファイルの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用して OpenShift Container Platform を Google Cloud Platform (GCP) にインストールするには、インストールプログラムがクラスターをデプロイするために必要なファイルを生成し、クラスターが使用するマシンのみを作成するようにそれらのファイルを変更する必要があります。**install-config.yaml** ファイル、Kubernetes マニフェスト、および Ignition 設定ファイルを生成し、カスタマイズします。

4.11.3.1. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

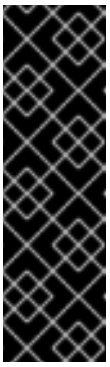
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

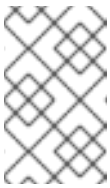
   

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

      v. クラスターをデプロイするリージョンを選択します。

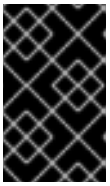
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

      vii. クラスターの記述名を入力します。

      viii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

4.11.3.2. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

関連情報

- オプション:Ingress DNS レコードの追加

4.11.4. 一般的な変数のエクスポート

4.11.4.1. インフラストラクチャー名の抽出

Ignition 設定ファイルには、Google Cloud Platform (GCP) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される Deployment Manager テンプレートにはこのインフラストラクチャー名への参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

4.11.4.2. Deployment Manager テンプレートの一般的な変数のエクスポート

ユーザーによって提供されるインフラストラクチャーのインストールを Google Cloud Platform (GCP) で実行するのに役立つ指定の Deployment Manager テンプレートで使用される一般的な変数のセットをエクスポートする必要があります。

注記

特定の Deployment Manager テンプレートには、追加のエクスポートされる変数が必要になる場合があります。これについては、関連する手順で詳しく説明されています。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

1. 提供される Deployment Manager テンプレートで使用される以下の一般的な変数をエクスポートします。

```
$ export BASE_DOMAIN='<base_domain>'
$ export BASE_DOMAIN_ZONE_NAME='<base_domain_zone_name>'
$ export NETWORK_CIDR='10.0.0.0/16'
$ export MASTER_SUBNET_CIDR='10.0.0.0/19'
$ export WORKER_SUBNET_CIDR='10.0.32.0/19'

$ export KUBECONFIG=<installation_directory>/auth/kubeconfig (1)
$ export CLUSTER_NAME=`jq -r .clusterName <installation_directory>/metadata.json`
$ export INFRA_ID=`jq -r .infraID <installation_directory>/metadata.json`
$ export PROJECT_NAME=`jq -r .gcp.projectID <installation_directory>/metadata.json`
$ export REGION=`jq -r .gcp.region <installation_directory>/metadata.json`
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

4.11.5. GCP での VPC の作成

OpenShift Container Platform クラスターで使用する VPC を Google Cloud Platform (GCP) で作成する必要があります。各種の要件を満たすよう VPC をカスタマイズできます。VPC を作成する 1 つの方法として、提供されている Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 本トピックの **VPC の Deployment Manager テンプレート**セクションを確認し、これを **01_vpc.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な VPC について記述しています。

2. **01_xvdb.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >01_vpc.yaml
imports:
- path: 01_vpc.py

resources:
- name: cluster-vpc
  type: 01_vpc.py
  properties:
    infra_id: '${INFRA_ID}' 1
    region: '${REGION}' 2
    master_subnet_cidr: '${MASTER_SUBNET_CIDR}' 3
    worker_subnet_cidr: '${WORKER_SUBNET_CIDR}' 4
EOF
```

1. **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
2. **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。
3. **master_subnet_cidr** はマスターサブネットの CIDR です (例: **10.0.0.0/19**)。
4. **worker_subnet_cidr** はワーカーサブネットの CIDR です (例: **10.0.32.0/19**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-vpc --config 01_vpc.yaml
```

4.11.5.1. VPC の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な VPC をデプロイすることができます。

例4.19 01_vpc.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-network',
        'type': 'compute.v1.network',
        'properties': {
            'region': context.properties['region'],
            'autoCreateSubnetworks': False
        }
    }, {
        'name': context.properties['infra_id'] + '-master-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['master_subnet_cidr']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-worker-subnet',
        'type': 'compute.v1.subnetwork',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'ipCidrRange': context.properties['worker_subnet_cidr']
        }
    }, {
        'name': context.properties['infra_id'] + '-router',
        'type': 'compute.v1.router',
        'properties': {
            'region': context.properties['region'],
            'network': '$(ref.' + context.properties['infra_id'] + '-network.selfLink)',
            'nats': [{
                'name': context.properties['infra_id'] + '-nat-master',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 7168,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-master-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }, {
                'name': context.properties['infra_id'] + '-nat-worker',
                'natIpAllocateOption': 'AUTO_ONLY',
                'minPortsPerVm': 512,
                'sourceSubnetworkIpRangesToNat': 'LIST_OF_SUBNETWORKS',
                'subnetworks': [{
                    'name': '$(ref.' + context.properties['infra_id'] + '-worker-subnet.selfLink)',
                    'sourceIpRangesToNat': ['ALL_IP_RANGES']
                }]
            }]
        }
    }]

    return {'resources': resources}
```

4.11.6. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表4.44 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス

プロトコル	ポート	説明
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表4.45 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表4.46 コントロールプレーンマシンへのコントロールプレーンマシン

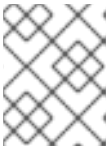
プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.47 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表4.48 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

4.11.7. GCP でのロードバランサーの作成

OpenShift Container Platform クラスターで使用するロードバランシングを Google Cloud Platform (GCP) で設定する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**内部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_int.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な内部負荷分散オブジェクトについて記述しています。
2. また、外部クラスターについては、本トピックの**外部ロードバランサーの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **02_lb_ext.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な外部負荷分散オブ

ジェクトについて記述しています。

3. デプロイメントテンプレートが使用する変数をエクスポートします。

   a. クラスターネットワークの場所をエクスポートします。

   ```
   $ export CLUSTER_NETWORK=(`gcloud compute networks describe ${INFRA_ID}-network --format json | jq -r .selfLink`)
   ```

   b. コントロールプレーンのサブネットの場所をエクスポートします。

   ```
   $ export CONTROL_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-master-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   c. クラスターが使用する 3 つのゾーンをエクスポートします。

   ```
   $ export ZONE_0=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[0] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_1=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[1] | cut -d "/" -f9`)
   ```

   ```
   $ export ZONE_2=(`gcloud compute regions describe ${REGION} --format=json | jq -r .zones[2] | cut -d "/" -f9`)
   ```

4. **02_infra.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_infra.yaml
imports:
- path: 02_lb_ext.py
- path: 02_lb_int.py (1)
resources:
- name: cluster-lb-ext (2)
  type: 02_lb_ext.py
  properties:
    infra_id: '${INFRA_ID}' (3)
    region: '${REGION}' (4)
- name: cluster-lb-int
  type: 02_lb_int.py
  properties:
    cluster_network: '${CLUSTER_NETWORK}'
    control_subnet: '${CONTROL_SUBNET}' (5)
    infra_id: '${INFRA_ID}'
    region: '${REGION}'
    zones: (6)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'
EOF
```

(1) (2) 外部クラスターをデプロイする場合にのみ必要です。

(3) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

4 **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

5 **control_subnet** は、コントロールサブセットの URL です。

6 **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-east1-b**、**us-east1-c**、および **us-east1-d**)。

5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-infra --config 02_infra.yaml
```

6. クラスター IP アドレスをエクスポートします。

```
$ export CLUSTER_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-ip --region=${REGION} --format json | jq -r .address`)
```

7. 外部クラスターの場合、クラスターのパブリック IP アドレスもエクスポートします。

```
$ export CLUSTER_PUBLIC_IP=(`gcloud compute addresses describe ${INFRA_ID}-cluster-public-ip --region=${REGION} --format json | jq -r .address`)
```

4.11.7.1. 外部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な外部ロードバランサーをデプロイすることができます。

例4.20 02_lb_ext.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-http-health-check',
        'type': 'compute.v1.httpHealthCheck',
        'properties': {
            'port': 6080,
            'requestPath': '/readyz'
        }
    }, {
        'name': context.properties['infra_id'] + '-api-target-pool',
        'type': 'compute.v1.targetPool',
        'properties': {
            'region': context.properties['region'],
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-http-health-check.selfLink)'],
            'instances': []
        }
```

```
    }, {
        'name': context.properties['infra_id'] + '-api-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'region': context.properties['region'],
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-public-ip.selfLink)',
            'target': '$(ref.' + context.properties['infra_id'] + '-api-target-pool.selfLink)',
            'portRange': '6443'
        }
    }]

    return {'resources': resources}
```

4.11.7.2. 内部ロードバランサーの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な内部ロードバランサーをデプロイすることができます。

例4.21 02_lb_int.py Deployment Manager テンプレート

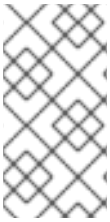
```
def GenerateConfig(context):

    backends = []
    for zone in context.properties['zones']:
        backends.append({
            'group': '$(ref.' + context.properties['infra_id'] + '-master-' + zone + '-instance-group' +
'.selfLink)'
        })

    resources = [{
        'name': context.properties['infra_id'] + '-cluster-ip',
        'type': 'compute.v1.address',
        'properties': {
            'addressType': 'INTERNAL',
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }, {
        # Refer to docs/dev/kube-apiserver-health-check.md on how to correctly setup health check
probe for kube-apiserver
        'name': context.properties['infra_id'] + '-api-internal-health-check',
        'type': 'compute.v1.healthCheck',
        'properties': {
            'httpsHealthCheck': {
                'port': 6443,
                'requestPath': '/readyz'
            },
            'type': "HTTPS"
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-backend-service',
        'type': 'compute.v1.regionBackendService',
        'properties': {
            'backends': backends,
```

```
            'healthChecks': ['$(ref.' + context.properties['infra_id'] + '-api-internal-health-
check.selfLink)'],
            'loadBalancingScheme': 'INTERNAL',
            'region': context.properties['region'],
            'protocol': 'TCP',
            'timeoutSec': 120
        }
    }, {
        'name': context.properties['infra_id'] + '-api-internal-forwarding-rule',
        'type': 'compute.v1.forwardingRule',
        'properties': {
            'backendService': '$(ref.' + context.properties['infra_id'] + '-api-internal-backend-
service.selfLink)',
            'IPAddress': '$(ref.' + context.properties['infra_id'] + '-cluster-ip.selfLink)',
            'loadBalancingScheme': 'INTERNAL',
            'ports': ['6443','22623'],
            'region': context.properties['region'],
            'subnetwork': context.properties['control_subnet']
        }
    }]

    for zone in context.properties['zones']:
        resources.append({
            'name': context.properties['infra_id'] + '-master-' + zone + '-instance-group',
            'type': 'compute.v1.instanceGroup',
            'properties': {
                'namedPorts': [
                    {
                        'name': 'ignition',
                        'port': 22623
                    }, {
                        'name': 'https',
                        'port': 6443
                    }
                ],
                'network': context.properties['cluster_network'],
                'zone': zone
            }
        })

    return {'resources': resources}
```

外部クラスターの作成時に、**02_lb_ext.py** テンプレートに加えてこのテンプレートが必要になります。

4.11.8. GCP でのプライベート DNS ゾーンの作成

OpenShift Container Platform クラスターで使用するプライベート DNS ゾーンを Google Cloud Platform (GCP) で設定する必要があります。このコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**プライベート DNS の Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **02_dns.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なプライベート DNS オブジェクトについて記述しています。
2. **02_dns.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >02_dns.yaml
imports:
- path: 02_dns.py

resources:
- name: cluster-dns
  type: 02_dns.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    cluster_domain: '${CLUSTER_NAME}.${BASE_DOMAIN}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **cluster_domain** はクラスターのドメインです (例: **openshift.example.com**)。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-dns --config 02_dns.yaml
```

4. このテンプレートは Deployment Manager の制限により DNS エントリーを作成しないので、手動で作成する必要があります。
   a. 内部 DNS エントリーを追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name
```

```
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction add ${CLUSTER_IP} --name api-
int.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone ${INFRA_ID}-
private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

b. 外部クラスターの場合、外部 DNS エントリーも追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${CLUSTER_PUBLIC_IP} --name
api.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 60 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone ${BASE_DOMAIN_ZONE_NAME}
```

4.11.8.1. プライベート DNS の Deployment Manager テンプレート

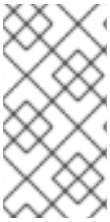
以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なプライベート DNS をデプロイすることができます。

例4.22 02_dns.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-private-zone',
        'type': 'dns.v1.managedZone',
        'properties': {
            'description': '',
            'dnsName': context.properties['cluster_domain'] + '.',
            'visibility': 'private',
            'privateVisibilityConfig': {
                'networks': [{
                    'networkUrl': context.properties['cluster_network']
                }]
            }
        }
    }]

    return {'resources': resources}
```

4.11.9. GCP でのファイアウォールルールの作成

OpenShift Container Platform クラスターで使用するファイアウォールルールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。



注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**ファイアウォールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_firewall.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なセキュリティーグループについて記述しています。
2. **03_firewall.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_firewall.yaml
imports:
- path: 03_firewall.py

resources:
- name: cluster-firewall
  type: 03_firewall.py
  properties:
    allowed_external_cidr: '0.0.0.0/0' (1)
    infra_id: '${INFRA_ID}' (2)
    cluster_network: '${CLUSTER_NETWORK}' (3)
    network_cidr: '${NETWORK_CIDR}' (4)
EOF
```

(1) **allowed_external_cidr** は、クラスター API にアクセスでき、ブートストラップホストに対して SSH を実行できる CIDR 範囲です。内部クラスターの場合、この値を **${NETWORK_CIDR}** に設定します。

(2) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(3) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(4) **network_cidr** は VPC ネットワークの CIDR です (例: **10.0.0.0/16**)。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-firewall --config
03_firewall.yaml
```

4.11.9.1. ファイアウォールルール用の Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なファイアウォールルールをデプロイすることができます。

例4.23 03_firewall.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-in-ssh',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-bootstrap']
        }
    }, {
        'name': context.properties['infra_id'] + '-api',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6443']
            }],
            'sourceRanges': [context.properties['allowed_external_cidr']],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-health-checks',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['6080', '6443', '22624']
            }],
            'sourceRanges': ['35.191.0.0/16', '130.211.0.0/22', '209.85.152.0/22', '209.85.204.0/22'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-etcd',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['2379-2380']
            }],
            'sourceTags': [context.properties['infra_id'] + '-master'],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
```

```
        'name': context.properties['infra_id'] + '-control-plane',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'tcp',
                'ports': ['10257']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10259']
            },{
                'IPProtocol': 'tcp',
                'ports': ['22623']
            }],
            'sourceTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ],
            'targetTags': [context.properties['infra_id'] + '-master']
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-network',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'icmp'
            },{
                'IPProtocol': 'tcp',
                'ports': ['22']
            }],
            'sourceRanges': [context.properties['network_cidr']],
            'targetTags': [
                context.properties['infra_id'] + '-master',
                context.properties['infra_id'] + '-worker'
            ]
        }
    }, {
        'name': context.properties['infra_id'] + '-internal-cluster',
        'type': 'compute.v1.firewall',
        'properties': {
            'network': context.properties['cluster_network'],
            'allowed': [{
                'IPProtocol': 'udp',
                'ports': ['4789', '6081']
            },{
                'IPProtocol': 'tcp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'udp',
                'ports': ['9000-9999']
            },{
                'IPProtocol': 'tcp',
                'ports': ['10250']
            },{
                'IPProtocol': 'tcp',
```

```
            'ports': ['30000-32767']
        },{
            'IPProtocol': 'udp',
            'ports': ['30000-32767']
        }],
        'sourceTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ],
        'targetTags': [
            context.properties['infra_id'] + '-master',
            context.properties['infra_id'] + '-worker'
        ]
    }
}]

return {'resources': resources}
```

4.11.10. GCP での IAM ロールの作成

OpenShift Container Platform クラスターで使用する IAM ロールを Google Cloud Platform (GCP) で作成する必要があります。これらのコンポーネントを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用して GCP インフラストラクチャーを使用しない場合、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。

手順

1. 本トピックの**IAM ロールの Deployment Manager テンプレート**セクションのテンプレートをコピーし、これを **03_iam.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要な IAM ロールについて記述しています。
2. **03_iam.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >03_iam.yaml
imports:
- path: 03_iam.py
resources:
- name: cluster-iam
  type: 03_iam.py
```

```
  properties:
    infra_id: '${INFRA_ID}' 1
EOF
```

1 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

3. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-iam --config 03_iam.yaml
```

4. マスターサービスアカウントの変数をエクスポートします。

```
$ export MASTER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-m@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

5. ワーカーサービスアカウントの変数をエクスポートします。

```
$ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
"email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
```

6. コンピュートマシンをホストするサブネットの変数をエクスポートします。

```
$ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe ${INFRA_ID}-
worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
```

7. このテンプレートは Deployment Manager の制限によりポリシーバインディングを作成しないため、これらを手動で作成する必要があります。

```
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.instanceAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.networkAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/compute.securityAdmin"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/iam.serviceAccountUser"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${MASTER_SERVICE_ACCOUNT}" --role "roles/storage.admin"

$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/compute.viewer"
$ gcloud projects add-iam-policy-binding ${PROJECT_NAME} --member
"serviceAccount:${WORKER_SERVICE_ACCOUNT}" --role "roles/storage.admin"
```

8. サービスアカウントキーを作成し、後で使用できるようにこれをローカルに保存します。

```
$ gcloud iam service-accounts keys create service-account-key.json --iam-
account=${MASTER_SERVICE_ACCOUNT}
```

4.11.10.1. IAM ロールの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要な IAM ロールをデプロイすることができます。

例4.24 03_iam.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-master-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-m',
            'displayName': context.properties['infra_id'] + '-master-node'
        }
    }, {
        'name': context.properties['infra_id'] + '-worker-node-sa',
        'type': 'iam.v1.serviceAccount',
        'properties': {
            'accountId': context.properties['infra_id'] + '-w',
            'displayName': context.properties['infra_id'] + '-worker-node'
        }
    }]

    return {'resources': resources}
```

4.11.11. GCP インフラストラクチャー用の RHCOS クラスターイメージの作成

OpenShift Container Platform ノードに Google Cloud Platform (GCP) 用の有効な Red Hat Enterprise Linux CoreOS (RHCOS) イメージを使用する必要があります。

手順

1. RHCOS イメージミラー ページから RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-<version>-<arch>-gcp.<arch>.tar.gz** 形式の OpenShift Container Platform のバージョン番号が含まれます。

2. Google ストレージバケットを作成します。

```
$ gsutil mb gs://<bucket_name>
```

3. RHCOS イメージを Google ストレージバケットにアップロードします。

```
$ gsutil cp <downloaded_image_file_path>/rhcos-<version>-x86_64-gcp.x86_64.tar.gz
gs://<bucket_name>
```

4. アップロードした RHCOS イメージの場所を変数としてエクスポートします。

```
$ export IMAGE_SOURCE=`gs://<bucket_name>/rhcos-<version>-x86_64-
gcp.x86_64.tar.gz`
```

5. クラスターイメージを作成します。

```
$ gcloud compute images create "${INFRA_ID}-rhcos-image" \
    --source-uri="${IMAGE_SOURCE}"
```

4.11.12. GCP でのブートストラップマシンの作成

OpenShift Container Platform クラスターの初期化を実行する際に使用するブートストラップマシンを Google Cloud Platform (GCP) で作成する必要があります。このマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供されている Deployment Manager テンプレートを使用してブートストラップマシンを作成しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- pyOpenSSL がインストールされていることを確認します。

手順

1. 本トピックの**ブートストラップマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **04_bootstrap.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なブートストラップマシンについて記述しています。
2. インストールプログラムで必要な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所をエクスポートします。

```
$ export CLUSTER_IMAGE=(`gcloud compute images describe ${INFRA_ID}-rhcos-image --
format json | jq -r .selfLink`)
```

3. バケットを作成し、**bootstrap.ign** ファイルをアップロードします。

```
$ gsutil mb gs://${INFRA_ID}-bootstrap-ignition
$ gsutil cp <installation_directory>/bootstrap.ign gs://${INFRA_ID}-bootstrap-ignition/
```

4. Ignition 設定にアクセスするために使用するブートストラップインスタンスの署名付き URL を作成します。出力から URL を変数としてエクスポートします。

```
$ export BOOTSTRAP_IGN=`gsutil signurl -d 1h service-account-key.json gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign | grep "^gs:" | awk '{print $5}'`
```

5. **04_bootstrap.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >04_bootstrap.yaml
imports:
- path: 04_bootstrap.py

resources:
- name: cluster-bootstrap
  type: 04_bootstrap.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    region: '${REGION}' (2)
    zone: '${ZONE_0}' (3)

    cluster_network: '${CLUSTER_NETWORK}' (4)
    control_subnet: '${CONTROL_SUBNET}' (5)
    image: '${CLUSTER_IMAGE}' (6)
    machine_type: 'n1-standard-4' (7)
    root_volume_size: '128' (8)

    bootstrap_ign: '${BOOTSTRAP_IGN}' (9)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **region** はクラスターをデプロイするリージョンです (例: **us-central1**)。

(3) **zone** はブートストラップインスタンスをデプロイするゾーンです (例: **us-central1-b**)。

(4) **cluster_network** はクラスターネットワークの **selfLink** URL です。

(5) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(6) **image** は RHCOS イメージの **selfLink** URL です。

(7) **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

(8) **root_volume_size** はブートストラップマシンのブートディスクサイズです。

(9) **bootstrap_ign** は署名付き URL の作成時の URL 出力です。

6. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-bootstrap --config
04_bootstrap.yaml
```

7. Deployment Manager の制限によりテンプレートではロードバランサーのメンバーシップを管理しないため、ブートストラップマシンは手動で追加する必要があります。

   a. ブートストラップインスタンスを内部ロードバランサーのインスタンスグループに追加します。

   ```
   $ gcloud compute instance-groups unmanaged add-instances \
     ${INFRA_ID}-bootstrap-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-
   bootstrap
   ```

   b. ブートストラップインスタンスグループを内部ロードバランサーのバックエンドサービスに追加します。

   ```
   $ gcloud compute backend-services add-backend \
     ${INFRA_ID}-api-internal-backend-service --region=${REGION} --instance-
   group=${INFRA_ID}-bootstrap-instance-group --instance-group-zone=${ZONE_0}
   ```

4.11.12.1. ブートストラップマシンの Deployment Manager テンプレート

以下の Deployment Mananger テンプレートを使用し、OpenShift Container Platform クラスターに必要なブートストラップマシンをデプロイすることができます。

例4.25 04_bootstrap.py Deployment Manager テンプレート

```
def GenerateConfig(context):

    resources = [{
        'name': context.properties['infra_id'] + '-bootstrap-public-ip',
        'type': 'compute.v1.address',
        'properties': {
            'region': context.properties['region']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': '{"ignition":{"config":{"replace":{"source":"' + context.properties['bootstrap_ign']
+ '","verification":{}}},"timeouts":{},"version":"2.1.0"},"networkd":{},"passwd":{},"storage":
```

```
{},"systemd":{}}',
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet'],
                'accessConfigs': [{
                    'natIP': '$(ref.' + context.properties['infra_id'] + '-bootstrap-public-ip.address)'
                }]
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                    context.properties['infra_id'] + '-bootstrap'
                ]
            },
            'zone': context.properties['zone']
        }
    }, {
        'name': context.properties['infra_id'] + '-bootstrap-instance-group',
        'type': 'compute.v1.instanceGroup',
        'properties': {
            'namedPorts': [
                {
                    'name': 'ignition',
                    'port': 22623
                }, {
                    'name': 'https',
                    'port': 6443
                }
            ],
            'network': context.properties['cluster_network'],
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

4.11.13. GCP でのコントロールプレーンマシンの作成

クラスターで使用するコントロールプレーンマシンを Google Cloud Platform (GCP) で作成する必要があります。これらのマシンを作成する方法として、提供される Deployment Manager テンプレートを変更することができます。

注記

提供される Deployment Manager テンプレートを使用してコントロールプレーンマシンを使用しない場合、指定される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。

- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

手順

1. 本トピックの**コントロールプレーンマシンの Deployment Manager テンプレート**セクションからテンプレートをコピーし、これを **05_control_plane.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なコントロールプレーンのマシンについて記述しています。
2. リソース定義で必要な以下の変数をエクスポートします。

```
$ export MASTER_IGNITION=`cat <installation_directory>/master.ign`
```

3. **05_control_plane.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >05_control_plane.yaml
imports:
- path: 05_control_plane.py

resources:
- name: cluster-control-plane
  type: 05_control_plane.py
  properties:
    infra_id: '${INFRA_ID}' (1)
    zones: (2)
    - '${ZONE_0}'
    - '${ZONE_1}'
    - '${ZONE_2}'

    control_subnet: '${CONTROL_SUBNET}' (3)
    image: '${CLUSTER_IMAGE}' (4)
    machine_type: 'n1-standard-4' (5)
    root_volume_size: '128'
    service_account_email: '${MASTER_SERVICE_ACCOUNT}' (6)

    ignition: '${MASTER_IGNITION}' (7)
EOF
```

(1) **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。

(2) **zones** は、コントロールプレーンインスタンスをデプロイするゾーンです (例: **us-central1-a**、**us-central1-b**、および **us-central1-c**)。

(3) **control_subnet** は、コントロールサブセットの **selfLink** URL です。

(4) **image** は RHCOS イメージの **selfLink** URL です。

5 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。

6 **service_account_email** は作成したマスターサービスアカウントのメールアドレスです。

7 **ignition** は **master.ign** ファイルの内容です。

4. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-control-plane --config
05_control_plane.yaml
```

5. Deployment Manager の制限により、テンプレートではロードバランサーのメンバーシップを管理しないため、コントロールプレーンマシンを手動で追加する必要があります。

- 以下のコマンドを実行してコントロールプレーンマシンを適切なインスタンスグループに追加します。

```
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_0}-instance-group --zone=${ZONE_0} --instances=${INFRA_ID}-m-0
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_1}-instance-group --zone=${ZONE_1} --instances=${INFRA_ID}-m-1
$ gcloud compute instance-groups unmanaged add-instances ${INFRA_ID}-master-
${ZONE_2}-instance-group --zone=${ZONE_2} --instances=${INFRA_ID}-m-2
```

- 外部クラスターの場合、以下のコマンドを実行してコントロールプレーンマシンをターゲットプールに追加する必要もあります。

```
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_0}" --instances=${INFRA_ID}-m-0
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_1}" --instances=${INFRA_ID}-m-1
$ gcloud compute target-pools add-instances ${INFRA_ID}-api-target-pool --instances-
zone="${ZONE_2}" --instances=${INFRA_ID}-m-2
```

4.11.13.1. コントロールプレーンマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用して、OpenShift Container Platform クラスターに必要なコントロールプレーンマシンをデプロイすることができます。

例4.26 05_control_plane.py Deployment Manager テンプレート

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-m-0',
    'type': 'compute.v1.instance',
    'properties': {
      'disks': [{
        'autoDelete': True,
        'boot': True,
        'initializeParams': {
          'diskSizeGb': context.properties['root_volume_size'],
          'diskType': 'zones/' + context.properties['zones'][0] + '/diskTypes/pd-ssd',
          'sourceImage': context.properties['image']
```

```
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][0] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
        },
        'zone': context.properties['zones'][0]
    }
}, {
    'name': context.properties['infra_id'] + '-m-1',
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
            'autoDelete': True,
            'boot': True,
            'initializeParams': {
                'diskSizeGb': context.properties['root_volume_size'],
                'diskType': 'zones/' + context.properties['zones'][1] + '/diskTypes/pd-ssd',
                'sourceImage': context.properties['image']
            }
        }],
        'machineType': 'zones/' + context.properties['zones'][1] + '/machineTypes/' +
context.properties['machine_type'],
        'metadata': {
            'items': [{
                'key': 'user-data',
                'value': context.properties['ignition']
            }]
        },
        'networkInterfaces': [{
            'subnetwork': context.properties['control_subnet']
        }],
        'serviceAccounts': [{
            'email': context.properties['service_account_email'],
            'scopes': ['https://www.googleapis.com/auth/cloud-platform']
        }],
        'tags': {
            'items': [
                context.properties['infra_id'] + '-master',
            ]
```

```
            },
            'zone': context.properties['zones'][1]
        }
    }, {
        'name': context.properties['infra_id'] + '-m-2',
        'type': 'compute.v1.instance',
        'properties': {
            'disks': [{
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'diskType': 'zones/' + context.properties['zones'][2] + '/diskTypes/pd-ssd',
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zones'][2] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['control_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-master',
                ]
            },
            'zone': context.properties['zones'][2]
        }
    }]

    return {'resources': resources}
```

4.11.14. ブートストラップの完了を待機し、GCP のブートストラップリソースを削除する

Google Cloud Platform (GCP) ですべての必要なインフラストラクチャーを作成した後に、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。

- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> \ 1
    --log-level info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

コマンドが **FATAL** 警告を出さずに終了する場合、実稼働用のコントロールプレーンは初期化されています。

2. ブートストラップリソースを削除します。

```
$ gcloud compute backend-services remove-backend ${INFRA_ID}-api-internal-backend-
service --region=${REGION} --instance-group=${INFRA_ID}-bootstrap-instance-group --
instance-group-zone=${ZONE_0}
$ gsutil rm gs://${INFRA_ID}-bootstrap-ignition/bootstrap.ign
$ gsutil rb gs://${INFRA_ID}-bootstrap-ignition
$ gcloud deployment-manager deployments delete ${INFRA_ID}-bootstrap
```

4.11.15. GCP での追加のワーカーマシンの作成

Google Cloud Platform (GCP) でクラスターが使用するワーカーマシンを作成するには、それぞれのインスタンスを個別に起動するか、または自動スケーリンググループなどのクラスター外にある自動プロセスを実行します。OpenShift Container Platform の組み込まれたクラスタースケーリングメカニズムやマシン API を利用できます。

この例では、Deployment Manager テンプレートを使用して 1 つのインスタンスを手動で起動します。追加のインスタンスは、ファイル内に **06_worker.py** というタイプのリソースを追加して起動することができます。

注記

ワーカーマシンを使用するために提供される Deployment Manager テンプレートを使用しない場合は、提供される情報を確認し、インフラストラクチャーを手動で作成する必要があります。クラスターが適切に初期化されない場合、インストールログを用意して Red Hat サポートに問い合わせする必要がある可能性があります。

前提条件

- GCP アカウントを設定します。
- クラスターの Ignition 設定ファイルを生成します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。
- コントロールプレーンマシンを作成します。

手順

1. 本トピックの**ワーカーマシンの Deployment Manager テンプレート**からテンプレートをコピーし、これを **06_worker.py** としてコンピューターに保存します。このテンプレートは、クラスターに必要なワーカーマシンについて記述しています。
2. リソース定義が使用する変数をエクスポートします。
   a. コンピュートマシンをホストするサブネットをエクスポートします。

   ```
   $ export COMPUTE_SUBNET=(`gcloud compute networks subnets describe
   ${INFRA_ID}-worker-subnet --region=${REGION} --format json | jq -r .selfLink`)
   ```

   b. サービスアカウントのメールアドレスをエクスポートします。

   ```
   $ export WORKER_SERVICE_ACCOUNT=(`gcloud iam service-accounts list --filter
   "email~^${INFRA_ID}-w@${PROJECT_NAME}." --format json | jq -r '.[0].email'`)
   ```

   c. コンピュートマシンの Ignition 設定ファイルの場所をエクスポートします。

   ```
   $ export WORKER_IGNITION=`cat <installation_directory>/worker.ign`
   ```

3. **06_worker.yaml** リソース定義ファイルを作成します。

```
$ cat <<EOF >06_worker.yaml
imports:
- path: 06_worker.py

resources:
- name: 'worker-0' (1)
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' (2)
    zone: '${ZONE_0}' (3)
    compute_subnet: '${COMPUTE_SUBNET}' (4)
    image: '${CLUSTER_IMAGE}' (5)
    machine_type: 'n1-standard-4' (6)
    root_volume_size: '128'
```

```
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 7
    ignition: '${WORKER_IGNITION}' 8
- name: 'worker-1'
  type: 06_worker.py
  properties:
    infra_id: '${INFRA_ID}' 9
    zone: '${ZONE_1}' 10
    compute_subnet: '${COMPUTE_SUBNET}' 11
    image: '${CLUSTER_IMAGE}' 12
    machine_type: 'n1-standard-4' 13
    root_volume_size: '128'
    service_account_email: '${WORKER_SERVICE_ACCOUNT}' 14
    ignition: '${WORKER_IGNITION}' 15
EOF
```

1. **name** はワーカーマシンの名前です (例: **worker-0**)。
2. 9 **infra_id** は抽出手順で得られる **INFRA_ID** インフラストラクチャー名です。
3. 10 **zone** はワーカーマシンをデプロイするゾーンです (例: **us-central1-a**)。
4. 11 **compute_subnet** はコンピュートサブネットの **selfLink** URL です。
5. 12 **image** は RHCOS イメージの **selfLink** URL です。
6. 13 **machine_type** はインスタンスのマシンタイプです (例: **n1-standard-4**)。
7. 14 **service_account_email** は作成したワーカーサービスアカウントのメールアドレスです。
8. 15 **Ignition** は **worker.ign** ファイルの内容です。

4. オプション: 追加のインスタンスを起動する必要がある場合には、**06_worker.py** タイプの追加のリソースを **06_worker.yaml** リソース定義ファイルに組み込みます。
5. **gcloud** CLI を使用してデプロイメントを作成します。

```
$ gcloud deployment-manager deployments create ${INFRA_ID}-worker --config
06_worker.yaml
```

4.11.15.1. ワーカーマシンの Deployment Manager テンプレート

以下の Deployment Manager テンプレートを使用し、OpenShift Container Platform クラスターに必要なワーカーマシンをデプロイすることができます。

例4.27 06_worker.py Deployment Manager テンプレート

```
def GenerateConfig(context):

  resources = [{
    'name': context.properties['infra_id'] + '-' + context.env['name'],
    'type': 'compute.v1.instance',
    'properties': {
        'disks': [{
```

```
                'autoDelete': True,
                'boot': True,
                'initializeParams': {
                    'diskSizeGb': context.properties['root_volume_size'],
                    'sourceImage': context.properties['image']
                }
            }],
            'machineType': 'zones/' + context.properties['zone'] + '/machineTypes/' +
context.properties['machine_type'],
            'metadata': {
                'items': [{
                    'key': 'user-data',
                    'value': context.properties['ignition']
                }]
            },
            'networkInterfaces': [{
                'subnetwork': context.properties['compute_subnet']
            }],
            'serviceAccounts': [{
                'email': context.properties['service_account_email'],
                'scopes': ['https://www.googleapis.com/auth/cloud-platform']
            }],
            'tags': {
                'items': [
                    context.properties['infra_id'] + '-worker',
                ]
            },
            'zone': context.properties['zone']
        }
    }]

    return {'resources': resources}
```

4.11.16. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

4.11.17. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

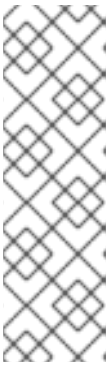
出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

4.11.18. オプション: Ingress DNS レコードの追加

Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除した場合、Ingress ロードバランサーをポイントする DNS レコードを手動で作成する必要があります。ワイルドカード ***.apps.{baseDomain}.** または特定のレコードのいずれかを作成できます。要件に基づいて A、CNAME その他のレコードを使用できます。

前提条件

- GCP アカウントを設定します。
- Kubernetes マニフェストの作成および Ignition 設定の生成時に DNS ゾーン設定を削除します。
- GCP で VPC および関連するサブネットを作成し、設定します。
- GCP でネットワークおよびロードバランサーを作成し、設定します。
- コントロールプレーンおよびコンピュートロールを作成します。
- ブートストラップマシンを作成します。

- コントロールプレーンマシンを作成します。
- ワーカーマシンを作成します。

手順

1. Ingress ルーターがロードバランサーを作成し、 **EXTERNAL-IP** フィールドにデータを設定するのを待機します。

```
$ oc -n openshift-ingress get service router-default
```

出力例

```
NAME            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
router-default  LoadBalancer  172.30.18.154  35.233.157.184
80:32288/TCP,443:31215/TCP  98
```

2. A レコードをゾーンに追加します。
   - A レコードを使用するには、以下を実行します。
     i. ルーター IP アドレスの変数をエクスポートします。

```
$ export ROUTER_IP=`oc -n openshift-ingress get service router-default --no-
headers | awk '{print $4}'`
```

     ii. A レコードをプライベートゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${INFRA_ID}-private-zone
$ gcloud dns record-sets transaction execute --zone ${INFRA_ID}-private-zone
```

     iii. また、外部クラスターの場合は、A レコードをパブリックゾーンに追加します。

```
$ if [ -f transaction.yaml ]; then rm transaction.yaml; fi
$ gcloud dns record-sets transaction start --zone ${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction add ${ROUTER_IP} --name
\*.apps.${CLUSTER_NAME}.${BASE_DOMAIN}. --ttl 300 --type A --zone
${BASE_DOMAIN_ZONE_NAME}
$ gcloud dns record-sets transaction execute --zone
${BASE_DOMAIN_ZONE_NAME}
```

   - ワイルドカードを使用する代わりに明示的なドメインを追加するには、クラスターのそれぞれの現行ルートのエントリーを作成します。

```
$ oc get --all-namespaces -o jsonpath='{range .items[*]}{range .status.ingress[*]}{.host}
{"\n"}{end}{end}' routes
```

出力例

```
oauth-openshift.apps.your.cluster.domain.example.com
console-openshift-console.apps.your.cluster.domain.example.com
downloads-openshift-console.apps.your.cluster.domain.example.com
alertmanager-main-openshift-monitoring.apps.your.cluster.domain.example.com
grafana-openshift-monitoring.apps.your.cluster.domain.example.com
prometheus-k8s-openshift-monitoring.apps.your.cluster.domain.example.com
```

4.11.19. ユーザーによってプロビジョニングされるインフラストラクチャーでの GCP インストールの完了

Google Cloud Platform (GCP) のユーザーによってプロビジョニングされるインフラストラクチャーで OpenShift Container Platform のインストールを開始した後は、クラスターが準備状態になるまでクラスターのイベントをモニターできます。

前提条件

- OpenShift Container Platform クラスターのブートストラップマシンを、ユーザーによってプロビジョニングされる GCP インフラストラクチャーにデプロイします。
- **oc** CLI をインストールし、ログインします。

手順

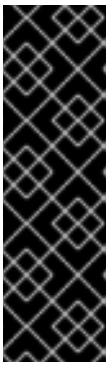
1. クラスターのインストールを完了します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. クラスターの稼働状態を確認します。
   a. 以下のコマンドを実行し、現在のクラスターバージョンとステータスを表示します。

```
$ oc get clusterversion
```

出力例

```
NAME     VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version            False       True          24m     Working towards 4.5.4: 99% complete
```

b. 以下のコマンドを実行し、 Cluster Version Operator (CVO) を使用してコントロールプレーンで管理される Operator を表示します。

```
$ oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

c. 以下のコマンドを実行して、クラスター Pod を表示します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                                  NAME
READY   STATUS    RESTARTS   AGE
kube-system                                etcd-member-ip-10-0-3-111.us-east-
```

```
2.compute.internal              1/1       Running     0          35m
kube-system                                 etcd-member-ip-10-0-3-239.us-east-
2.compute.internal              1/1       Running     0          37m
kube-system                                 etcd-member-ip-10-0-3-24.us-east-
2.compute.internal              1/1       Running     0          35m
openshift-apiserver-operator                openshift-apiserver-operator-6d6674f4f4-
h7t2t                   1/1       Running     1          37m
openshift-apiserver                         apiserver-fm48r
1/1       Running     0          30m
openshift-apiserver                         apiserver-fxkvv
1/1       Running     0          29m
openshift-apiserver                         apiserver-q85nm
1/1       Running     0          29m
...
openshift-service-ca-operator               openshift-service-ca-operator-66ff6dc6cd-
9r257                   1/1       Running     0          37m
openshift-service-ca                        apiservice-cabundle-injector-695b6bcbc-cl5hm
1/1       Running     0          35m
openshift-service-ca                        configmap-cabundle-injector-8498544d7-
25qn6                   1/1       Running     0          35m
openshift-service-ca                        service-serving-cert-signer-6445fc9c6-wqdqn
1/1       Running     0          35m
openshift-service-catalog-apiserver-operator          openshift-service-catalog-apiserver-
operator-549f44668b-b5q2w       1/1       Running     0          32m
openshift-service-catalog-controller-manager-operator   openshift-service-catalog-
controller-manager-operator-b78cr2lnm     1/1       Running     0          31m
```

現在のクラスターバージョンが **AVAILABLE** の場合、インストールが完了します。

4.11.20. 次のステップ

- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

4.12. GCP でのクラスターのアンインストール

Google Cloud Platform (GCP) にデプロイしたクラスターを削除できます。

4.12.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、または 削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

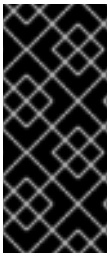
クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第5章 ベアメタルへのインストール

5.1. クラスターのベアメタルへのインストール

OpenShift Container Platform version 4.5 では、クラスターをプロビジョニングするベアメタルのインフラストラクチャーにインストールできます。



重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

5.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

5.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

5.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

5.1.3.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。3 ノードクラスターを実行している場合は、サポートされるのは、実行されるコンピュートマシンがゼロの場合です。1 つのコンピュートマシンの実行はサポートされていません。

注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

5.1.3.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる

必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

5.1.3.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

5.1.3.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

5.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。

5. ネットワーク接続を確認します。

5.1.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表5.1 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表5.2 コントロールプレーンへのすべてのマシン

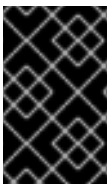
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表5.3 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件

クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



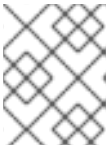
重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.4 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.5 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

5.1.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat

Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表5.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
ワーカーホスト	**<worker><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例5.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
```

```
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例5.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

5.1.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

5.1.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

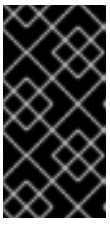
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.1.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

5.1.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

5.1.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

5.1.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

5.1.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

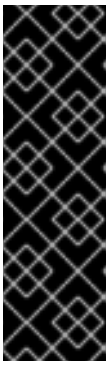
前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

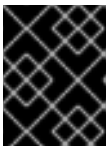
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

5.1.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.1.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.7 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.1.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.8 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.1.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.9 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

5.1.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表5.10 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

5.1.8.2. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
```

```
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

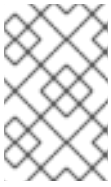
11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。ベアメタルインフラストラクチャー用に追加のプラットフォーム設定変数を指定することはできません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

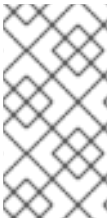


注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

5.1.8.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。



注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

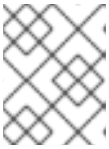
手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。
2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。
3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする

trusted-ca-bundle 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.1.9. 3 ノードクラスターの設定

オプションで、ワーカーなしで OpenShift Container Platform に 3 ノードクラスターをインストールし、実行できます。これにより、クラスター管理者および開発者が開発、実稼働およびテストに使用するための小規模なリソース効率の高いクラスターが提供されます。

手順

- **install-config.yaml** ファイルを編集し、以下の **compute** スタンザに示されるようにコンピュートレプリカ (ワーカーレプリカとしても知られる) の数を **0** に設定します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

5.1.10. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

警告

3 ノードクラスターを実行している場合は、以下の手順を省略してマスターをスケジュール対象にします。

1. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。
   c. ファイルを保存し、終了します。
2. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

   これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

3. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
```

```
└── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.1.11. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用する手順を実行してマシンを作成することができます。

5.1.11.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

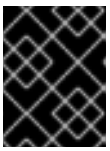
ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあること。

手順

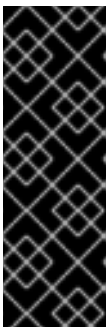
1. インストールプログラムが作成したコントロールプレーン、コンピュート、およびブートストラップ Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. RHCOS イメージミラー ページからオペレーティングシステムのインスタンスをインストールするために優先される方法で必要な RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ISO ファイルおよび RAW ディスクファイルをダウンロードする必要があります。これらのファイルの名前は以下の例のようになります。

- ISO: **rhcos-<version>-installer.<architecture>.iso**

- 圧縮された metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**

3. RAW RHCOS イメージファイルのいずれかを HTTP サーバーにアップロードし、その URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。

- ディスクに ISO イメージを書き込み、これを直接起動します。
- LOM インターフェイスで ISO リダイレクトを使用します。

5. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。

6. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

1. インストール先のシステムのブロックデバイスを指定します。
2. サーバーにアップロードした RAW イメージの URL を指定します。
3. このマシンタイプの Ignition 設定ファイルの URL を指定します。
4. **ip=dhcp** を設定するか、各ノードに個別の静的 IP アドレス (**ip=**) および DNS サーバー (**nameserver=**) を設定します。詳細は、**高度なネットワークの設定**を参照してください。
5. 複数のネットワークインターフェイスまたは DNS サーバーを使用する場合は、**高度なネットワークの設定**を参照してください。
6. オプションで、**高度なネットワークの設定**で説明されているように、**bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。

7. Enter を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

8. 継続してクラスターのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.1.11.1.1. 高度なネットワークの設定

ISO イメージから Red Hat Enterprise Linux CoreOS (RHCOS) をインストールする場合、そのイメージを起動してノードのネットワークを設定する際にカーネル引数を追加できます。以下の表は、これらのカーネル引数の使用方法について説明しています。

表5.11 高度なネットワークの設定

説明	例
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。次に、各ノードの DNS サーバーの IP アドレス (**nameserver=<dns_ip>**) を特定します。この例では、以下を設定します。 ● ノードの IP アドレス: **10.10.10.2** ● ゲートウェイアドレス: **10.10.10.254** ● ネットワーク: **255.255.255.0** ● ホスト名: **core0.example.com** ● DNS サーバーアドレス: **4.4.4.41**	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41
複数の **ip=** エントリーを指定して、複数のネットワークインターフェイスを指定します。	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。	nameserver=1.1.1.1 nameserver=8.8.8.8

説明	例

複数のネットワークインターフェイスを単一のインターフェイスにボンディングすることは、オプションとして **bond=** オプションを使用によってサポートされます。次の 2 つの例では、以下のようになります。
• ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。
• name は、ボンディングデバイス名 (**bond0**) で、network_interfaces は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。options はボンディングオプションのコンマ区切りの一覧です。(**modinfo bonding** を入力して、利用可能なオプションを表示します。)
• **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。
	DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。
	`bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp`
	静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。
	`bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none`
	重要
	高度なネットワークオプションを使用する場合、静的に設定されたアドレスが存在しないか、または適切にアクティブ化されていない RHCOS の初回起動時に問題が発生する可能性があります。この場合、問題を回避するには、RHCOS マシンを手動で再起動する必要がある場合があります。この問題は RHCOS の以降のバージョンの systemd で解決されています。詳細は、BZ#1902584 を参照してください。

5.1.11.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

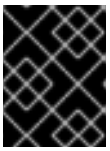
ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。PXE または iPXE ブートを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- PXE または iPXE インフラストラクチャーを提供するのに必要な DHCP、TFTP、および HTTP サービスの設定についての理解。
- 使用しているコンピューターからアクセス可能な HTTP サーバーおよび TFTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したマスター、ワーカーおよびブートストラップの Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから圧縮された metal RAW イメージ、**kernel** および **initramfs** ファイルを取得します。

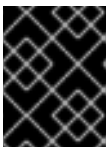
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には RAW イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン名が含まれます。以下の例のようになります。

- 圧縮されたメタル RAW イメージ: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
- **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**

3. RAW イメージを HTTP サーバーにアップロードします。
4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーにアップロードします。
   - iPXE の場合、**kernel** および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージに PXE または iPXE インストールを設定します。
ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2)
(3)
```

(1) TFTP サーバーで利用可能な **kernel** ファイルの場所を指定します。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP または TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は、TFTP サーバーの **initramfs** ファイルの場所です。**coreos.inst.image_url** パラメーター値は、HTTP サーバーの圧縮されたメタル RAW イメージの場所であり、 **coreos.inst.ignition_url** パラメーター値は HTTP サーバーのブートストラップ Ignition 設定ファイルの場所になります。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

- iPXE の場合:

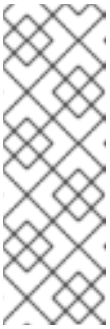
```
kernel  http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1)
(2)
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.
<architecture>.img (3)
boot
```

(1) HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd** パラメーター値は、以下の **initrd** 行で提供される **initramfs** ファイルの名前を参照し、**coreos.inst.image_url** パラメーター値は圧縮されたメタル RAW イメージの場所、 **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します

ます。

3 HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. UEFI を使用する場合は、以下の操作を実行します。

a. システムの起動に必要な EFI バイナリーおよび **grub.cfg** ファイルを提供します。**shim.efi** バイナリーと **grubx64.efi** バイナリーが必要です。

- RHCOS ISO をホストにマウントし、**images/efiboot.img** ファイルをホストにマウントして、必要な EFI バイナリーを展開します。**efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

b. RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

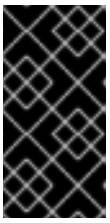
c. **grub.cfg** ファイルを編集し、以下の引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
 initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

1 **linux** 行の項目の最初の引数は、TFTP サーバーにアップロードした **kernel** ファイルの場所です。**coreos.inst.image_url** パラメーター値には、HTTP サーバーにアップロードした圧縮されたメタル RAW イメージの場所を指定します。**coreos.inst.ignition_url** パラメーターには、HTTP サーバーにアップロードしたブートストラップ Ingition 設定ファイルの場所を指定します。

2 TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.1.12. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

5.1.13. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

5.1.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

5.1.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

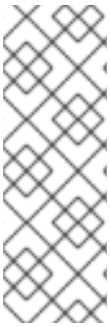
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

5.1.15.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないフラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

5.1.15.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

5.1.15.2.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

5.1.15.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

   - 以下を実行します。

     ```
     $ oc edit configs.imageregistry/cluster
     ```

     次に、行を変更します。

     ```
     managementState: Removed
     ```

     上記を以下のように変更します。

     ```
     managementState: Managed
     ```

5.1.15.2.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

   ```
   $ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
   {"rolloutStrategy":"Recreate","replicas":1}}'
   ```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

5.1.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
```

```
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

5.1.17. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

5.2. ネットワークのカスタマイズによるベアメタルへのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、カスタマイズされたネットワーク設定オプションでプロビジョニングするベアメタルインフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

5.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、Red Hat Insights にアクセスできるように設定 する必要があります。

5.2.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

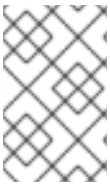
5.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

5.2.3.1. 必要なマシン

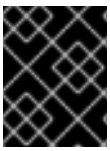
最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。3 ノードクラスターを実行している場合は、サポートされるのは、実行されるコンピュートマシンがゼロの場合です。1 つのコンピュートマシンの実行はサポートされていません。



注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

5.2.3.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

5.2.3.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

5.2.3.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

5.2.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。
5. ネットワーク接続を確認します。

5.2.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表5.12 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

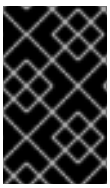
表5.13 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表5.14 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

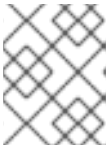


重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.15 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.16 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

5.2.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表5.17 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>. <base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例5.3 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H  ; refresh (3 hours)
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
  IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
```

```
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例5.4 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

5.2.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 **~/.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

5.2.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

5.2.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

5.2.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

5.2.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

5.2.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

5.2.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

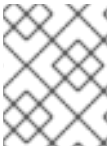
重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

5.2.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプ

ラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.2.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.18 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト

パラメーター	説明	値
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.2.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.19 ネットワークパラメーター

パラメーター	説明	値

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

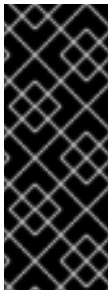
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

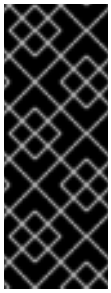
5.2.8.1.3. オプションの設定パラメーター

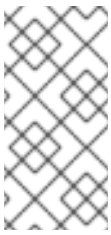
オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.20 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

5.2.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表5.21 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

5.2.8.2. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

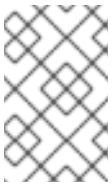
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
```

```
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

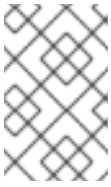
11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。ベアメタルインフラストラクチャー用に追加のプラットフォーム設定変数を指定することはできません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

5.2.8.3. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。

注記

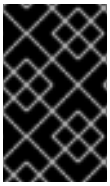
インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

表5.22 必要なネットワークパラメーター

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。	デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの**serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

5.2.9. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。



重要

インストールプロブラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

5.2.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
    ...
  kubeProxyConfig: 4
    iptablesSyncPeriod: 30s 5
    proxyArguments:
      iptables-min-sync-period: 6
      - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。

注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time で説明されています。

5.2.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

1 **install-config.yaml** ファイルに指定されます。

2 OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

3 OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

4 VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

5 すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

5.2.10.2. Cluster Network Operator の設定例

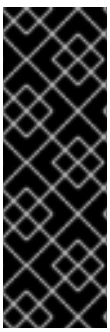
以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
      - 0s
```

5.2.11. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.2.12. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用する手順を実行してマシンを作成することができます。

5.2.12.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあること。

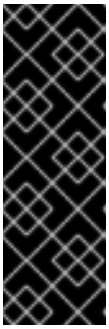
手順

1. インストールプログラムが作成したコントロールプレーン、コンピュート、およびブートストラップ Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. RHCOS イメージミラー ページからオペレーティングシステムのインスタンスをインストールするために優先される方法で必要な RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ISO ファイルおよび RAW ディスクファイルをダウンロードする必要があります。これらのファイルの名前は以下の例のようになります。

- ISO: **rhcos-<version>-installer.<architecture>.iso**
- 圧縮された metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**

3. RAW RHCOS イメージファイルのいずれかを HTTP サーバーにアップロードし、その URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - LOM インターフェイスで ISO リダイレクトを使用します。
5. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。
6. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

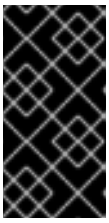
1. インストール先のシステムのブロックデバイスを指定します。
2. サーバーにアップロードした RAW イメージの URL を指定します。
3. このマシンタイプの Ignition 設定ファイルの URL を指定します。
4. **ip=dhcp** を設定するか、各ノードに個別の静的 IP アドレス (**ip=**) および DNS サーバー (**nameserver=**) を設定します。詳細は、**高度なネットワークの設定**を参照してください。

5 複数のネットワークインターフェイスまたは DNS サーバーを使用する場合は、**高度なネットワークの設定**を参照してください。

6 オプションで、**高度なネットワークの設定**で説明されているように、**bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。

7. Enter を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.2.12.1.1. 高度なネットワークの設定

ISO イメージから Red Hat Enterprise Linux CoreOS (RHCOS) をインストールする場合、そのイメージを起動してノードのネットワークを設定する際にカーネル引数を追加できます。以下の表は、これらのカーネル引数の使用方法について説明しています。

表5.23 高度なネットワークの設定

説明	例
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。次に、各ノードの DNS サーバーの IP アドレス (**nameserver=<dns_ip>**) を特定します。この例では、以下を設定します。 • ノードの IP アドレス: **10.10.10.2** • ゲートウェイアドレス: **10.10.10.254** • ネットワーク: **255.255.255.0** • ホスト名: **core0.example.com** • DNS サーバーアドレス: **4.4.4.41**	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41
複数の **ip=** エントリーを指定して、複数のネットワークインターフェイスを指定します。	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none

説明	例
複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。	nameserver=1.1.1.1 nameserver=8.8.8.8
複数のネットワークインターフェイスを単一のインターフェイスにボンディングすることは、オプションとして **bond=** オプションを使用によってサポートされます。次の 2 つの例では、以下のようになります。 ● ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。 ● name は、ボンディングデバイス名 (**bond0**) で、network_interfaces は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。options はボンディングオプションのコンマ区切りの一覧です。(**modinfo bonding** を入力して、利用可能なオプションを表示します。) ● **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。	DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。 bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。 bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none **重要** 高度なネットワークオプションを使用する場合、静的に設定されたアドレスが存在しないか、または適切にアクティブ化されていない RHCOS の初回起動時に問題が発生する可能性があります。この場合、問題を回避するには、RHCOS マシンを手動で再起動する必要がある場合があります。この問題は RHCOS の以降のバージョンの systemd で解決されています。詳細は、BZ#1902584 を参照してください。

5.2.12.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。PXE または iPXE ブートを使用してマシンを作成することができます。

前提条件

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- PXE または iPXE インフラストラクチャーを提供するのに必要な DHCP、TFTP、および HTTP サービスの設定についての理解。
- 使用しているコンピューターからアクセス可能な HTTP サーバーおよび TFTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したマスター、ワーカーおよびブートストラップの Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから圧縮された metal RAW イメージ、**kernel** および **initramfs** ファイルを取得します。

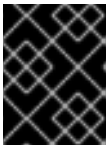
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には RAW イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン名が含まれます。以下の例のようになります。

- 圧縮されたメタル RAW イメージ: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
- **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**

3. RAW イメージを HTTP サーバーにアップロードします。
4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーにアップロードします。
   - iPXE の場合、**kernel** および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージに PXE または iPXE インストールを設定します。
ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

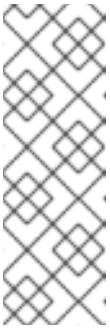
- PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2)
(3)
```

(1) TFTP サーバーで利用可能な **kernel** ファイルの場所を指定します。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

(3) HTTP または TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は、TFTP サーバーの **initramfs** ファイルの場所です。**coreos.inst.image_url** パラメーター値は、HTTP サーバーの圧縮されたメタル RAW イメージの場所であり、 **coreos.inst.ignition_url** パラメーター値は HTTP サーバーのブートストラップ Ignition 設定ファイルの場所になります。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

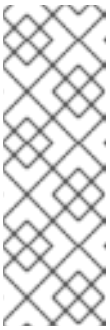
- iPXE の場合:

```
kernel  http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1)
(2)
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.
<architecture>.img (3)
boot
```

1. HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd** パラメーター値は、以下の **initrd** 行で提供される **initramfs** ファイルの名前を参照し、**coreos.inst.image_url** パラメーター値は圧縮されたメタル RAW イメージの場所、 **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

3. HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な EFI バイナリーおよび **grub.cfg** ファイルを提供します。**shim.efi** バイナリーと **grubx64.efi** バイナリーが必要です。

      - RHCOS ISO をホストにマウントし、**images/efiboot.img** ファイルをホストにマウントして、必要な EFI バイナリーを展開します。**efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

   b. RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

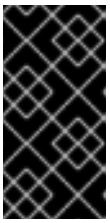
   c. **grub.cfg** ファイルを編集し、以下の引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
 initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

1. **linux** 行の項目の最初の引数は、TFTP サーバーにアップロードした **kernel** ファイルの場所です。**coreos.inst.image_url** パラメーター値には、HTTP サーバーにアップ

2. TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.2.13. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

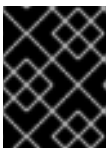
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

5.2.14. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

5.2.15. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

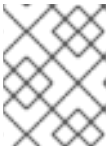
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

5.2.16. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

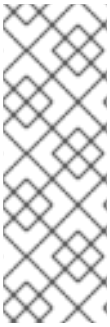
```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

5.2.16.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

5.2.16.2. イメージレジストリーストレージの設定

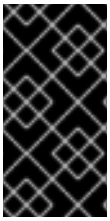
イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

5.2.16.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

5.2.17. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
```

```
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running   0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running   0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running   0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8     1/1
Running   0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

5.2.18. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

5.3. ネットワークが制限された環境でのクラスターのベアメタルへのインストール

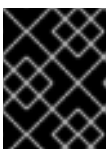
OpenShift Container Platform バージョン 4.5 では、クラスターをネットワークが制限された環境でプロビジョニングするベアメタルインフラストラクチャーにインストールできます。

重要

以下の手順に従って仮想化環境またはクラウド環境にクラスターをデプロイすることができますが、ベアメタルプラットフォーム以外の場合は追加の考慮事項に注意してください。このような環境で OpenShift Container Platform クラスターのインストールを試行する前に、Deploying OpenShift 4.x on non-tested platforms using the bare metal install method にある情報を確認してください。

5.3.1. 前提条件

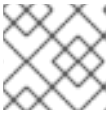
- ミラーホストでレジストリーを作成 し、OpenShift Container Platform の使用しているバージョン用の **imageContentSources** データを取得します。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了します。

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。



注記

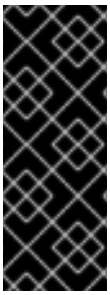
プロキシーを設定する場合は、このサイト一覧も確認してください。

5.3.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

5.3.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

5.3.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

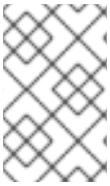
5.3.4. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

5.3.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。3 ノードクラスターを実行している場合は、サポートされるのは、実行されるコンピュートマシンがゼロの場合です。1 つのコンピュートマシンの実行はサポートされていません。



注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

5.3.4.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

5.3.4.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

5.3.4.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

5.3.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。
5. ネットワーク接続を確認します。

5.3.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表5.24 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表5.25 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表5.26 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.27 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

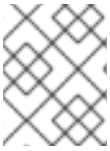
   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表5.28 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

5.3.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表5.29 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>. <base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例5.5 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
```

```
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例5.6 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

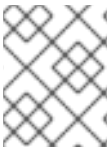
5.3.6. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

5.3.7. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

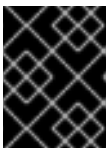
2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

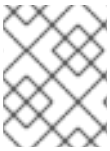


重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

5.3.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

5.3.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表5.30 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

5.3.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表5.31 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

5.3.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表5.32 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

5.3.7.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表5.33 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

5.3.7.2. ベアメタルのサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

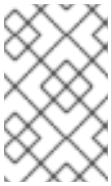
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
```

```
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。ベアメタルインフラストラクチャー用に追加のプラットフォーム設定変数を指定することはできません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを

5.3.7.3. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

注記

ベアメタルインストールでは、**install-config.yaml** ファイルの **networking.machineNetwork[].cidr** フィールドで指定される範囲にあるノード IP アドレスを割り当てない場合、それらを **proxy.noProxy** フィールドに含める必要があります。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

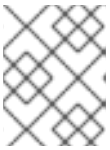
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されま

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

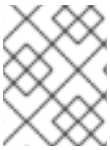


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

5.3.8. 3 ノードクラスターの設定

オプションで、ワーカーなしで OpenShift Container Platform に 3 ノードクラスターをインストールし、実行できます。これにより、クラスター管理者および開発者が開発、実稼働およびテストに使用するための小規模なリソース効率の高いクラスターが提供されます。

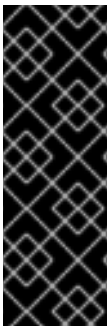
手順

- **install-config.yaml** ファイルを編集し、以下の **compute** スタンザに示されるようにコンピュートレプリカ (ワーカーレプリカとしても知られる) の数を **0** に設定します。

```
compute:
- name: worker
  platform: {}
  replicas: 0
```

5.3.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

警告

3 ノードクラスターを実行している場合は、以下の手順を省略してマスターをスケジュール対象にします。

1. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。
   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。
   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。
   c. ファイルを保存し、終了します。
2. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

   これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

3. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
```

```
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

5.3.10. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを作成し、これを base64 でエンコードします。以下に例を示します。

```
$ cat << EOF | base64
   pool 0.rhel.pool.ntp.org iburst (1)
   driftfile /var/lib/chrony/drift
   makestep 1.0 3
   rtcsync
   logdir /var/log/chrony
EOF
```

(1) DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

出力例

```
ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2Rp
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. **MachineConfig** ファイルを作成します。base64 文字列を独自に作成した文字列に置き換えます。この例では、ファイルを **master** ノードに追加します。これを **worker** に切り替えたり、**worker** ロールの追加の MachineConfig を作成したりできます。クラスターが使用するそれぞれのタイプのマシンについて MachineConfig ファイルを作成します。

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
```

```
    version: 2.2.0
  networkd: {}
  passwd: {}
  storage:
    files:
    - contents:
        source: data:text/plain;charset=utf-
8;base64,ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92Y
XIvbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
        verification: {}
      filesystem: root
      mode: 420
      path: /etc/chrony.conf
 osImageURL: ""
EOF
```

3. 設定ファイルのバックアップコピーを作成します。

4. 以下の 2 つの方法のいずれかで設定を適用します。

   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、そのファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を継続します。

   - クラスターがすでに実行中の場合は、ファイルを適用します。

     ```
     $ oc apply -f ./99-masters-chrony-configuration.yaml
     ```

5.3.11. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用する手順を実行してマシンを作成することができます。

5.3.11.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

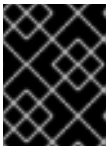
ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したコントロールプレーン、コンピュート、およびブートストラップ Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. RHCOS イメージミラー ページからオペレーティングシステムのインスタンスをインストールするために優先される方法で必要な RHCOS イメージを取得します。

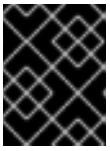
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ISO ファイルおよび RAW ディスクファイルをダウンロードする必要があります。これらのファイルの名前は以下の例のようになります。

- ISO: **rhcos-<version>-installer.<architecture>.iso**
- 圧縮された metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**

3. RAW RHCOS イメージファイルのいずれかを HTTP サーバーにアップロードし、その URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - LOM インターフェイスで ISO リダイレクトを使用します。
5. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。
6. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

1. インストール先のシステムのブロックデバイスを指定します。
2. サーバーにアップロードした RAW イメージの URL を指定します。

3 このマシンタイプの Ignition 設定ファイルの URL を指定します。

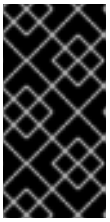
4 **ip=dhcp** を設定するか、各ノードに個別の静的 IP アドレス (**ip=**) および DNS サーバー (**nameserver=**) を設定します。詳細は、**高度なネットワークの設定**を参照してください。

5 複数のネットワークインターフェイスまたは DNS サーバーを使用する場合は、**高度なネットワークの設定**を参照してください。

6 オプションで、**高度なネットワークの設定**で説明されているように、**bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。

7. Enter を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.3.11.1.1. 高度なネットワークの設定

ISO イメージから Red Hat Enterprise Linux CoreOS (RHCOS) をインストールする場合、そのイメージを起動してノードのネットワークを設定する際にカーネル引数を追加できます。以下の表は、これらのカーネル引数の使用方法について説明しています。

表5.34 高度なネットワークの設定

説明	例
IP アドレスを設定するには、DHCP (**ip=dhcp**) を使用するか、または個別の静的 IP アドレス (**ip=<host_ip>**) を設定します。次に、各ノードの DNS サーバーの IP アドレス (**nameserver=<dns_ip>**) を特定します。この例では、以下を設定します。 • ノードの IP アドレス: **10.10.10.2** • ゲートウェイアドレス: **10.10.10.254** • ネットワーク: **255.255.255.0** • ホスト名: **core0.example.com** • DNS サーバーアドレス: **4.4.4.41**	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none nameserver=4.4.4.41

説明	例
複数の **ip=** エントリーを指定して、複数のネットワークインターフェイスを指定します。	ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp1s0:none ip=10.10.10.3::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
複数のネットワークインターフェイスを持つシステムで、DHCP および静的 IP 設定を組み合わせることができます。	ip=enp1s0:dhcp ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:enp2s0:none
各サーバーに **nameserver=** エントリーを追加して、複数の DNS サーバーを指定できます。	nameserver=1.1.1.1 nameserver=8.8.8.8
複数のネットワークインターフェイスを単一のインターフェイスにボンディングすることは、オプションとして **bond=** オプションを使用によってサポートされます。次の 2 つの例では、以下のようになります。 ● ボンディングされたインターフェイスを設定する構文は **bond=name[:network_interfaces][:options]** です。 ● name は、ボンディングデバイス名 (**bond0**) で、network_interfaces は物理 (イーサネット) インターフェイス (**em1,em2**) のコンマ区切り一覧を表します。options はボンディングオプションのコンマ区切りの一覧です。(**modinfo bonding** を入力して、利用可能なオプションを表示します。) ● **Bond=** を使用してボンディングされたインターフェイスを作成する場合は、IP アドレスの割り当て方法とボンディングされたインターフェイスのその他の情報を指定する必要があります。	DHCP を使用するようにボンディングされたインターフェイスを設定するには、ボンドの IP アドレスを **dhcp** に設定します。以下に例を示します。 bond=bond0:em1,em2:mode=active-backup ip=bond0:dhcp 静的 IP アドレスを使用するようにボンディングされたインターフェイスを設定するには、必要な特定の IP アドレスと関連情報を入力します。以下に例を示します。 bond=bond0:em1,em2:mode=active-backup ip=10.10.10.2::10.10.10.254:255.255.255.0:core0.example.com:bond0:none **重要** 高度なネットワークオプションを使用する場合、静的に設定されたアドレスが存在しないか、または適切にアクティブ化されていない RHCOS の初回起動時に問題が発生する可能性があります。この場合、問題を回避するには、RHCOS マシンを手動で再起動する必要がある場合があります。この問題は RHCOS の以降のバージョンの systemd で解決されています。詳細は、BZ#1902584 を参照してください。

5.3.11.2. PXE または iPXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるベアメタルインフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。PXE または iPXE ブートを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- PXE または iPXE インフラストラクチャーを提供するのに必要な DHCP、TFTP、および HTTP サービスの設定についての理解。
- 使用しているコンピューターからアクセス可能な HTTP サーバーおよび TFTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したマスター、ワーカーおよびブートストラップの Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから圧縮された metal RAW イメージ、**kernel** および **initramfs** ファイルを取得します。

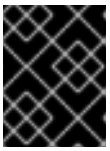
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には RAW イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン名が含まれます。以下の例のようになります。

- 圧縮されたメタル RAW イメージ: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
- **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**

3. RAW イメージを HTTP サーバーにアップロードします。
4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーにアップロードします。
   - iPXE の場合、**kernel** および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。

6. RHCOS イメージに PXE または iPXE インストールを設定します。
ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。

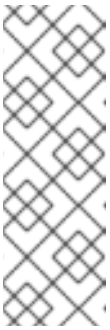
- PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2)
(3)
```

(1) TFTP サーバーで利用可能な **kernel** ファイルの場所を指定します。

(2) 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、**ip=eno1:dhcp** を設定します。

(3) HTTP または TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は、TFTP サーバーの **initramfs** ファイルの場所です。**coreos.inst.image_url** パラメーター値は、HTTP サーバーの圧縮されたメタル RAW イメージの場所であり、**coreos.inst.ignition_url** パラメーター値は HTTP サーバーのブートストラップ Ignition 設定ファイルの場所になります。



注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

- iPXE の場合:

```
kernel  http://<HTTP_server>/rhcos-<version>-<architecture>-installer-kernel-
<architecture> ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
```

```
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
2
initrd http://<HTTP_server>/rhcos-<version>-<architecture>-installer-initramfs.
<architecture>.img 3
boot
```

1. HTTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**kernel** パラメーター値は **kernel** ファイルの場所であり、**initrd** パラメーター値は、以下の **initrd** 行で提供される **initramfs** ファイルの名前を参照し、**coreos.inst.image_url** パラメーター値は圧縮されたメタル RAW イメージの場所、 **coreos.inst.ignition_url** パラメーター値はブートストラップ Ignition 設定ファイルの場所になります。

2. 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定します。

3. HTTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**kernel** 行に **console=** 引数を 1 つ以上追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. UEFI を使用する場合は、以下の操作を実行します。

   a. システムの起動に必要な EFI バイナリーおよび **grub.cfg** ファイルを提供します。**shim.efi** バイナリーと **grubx64.efi** バイナリーが必要です。

      - RHCOS ISO をホストにマウントし、**images/efiboot.img** ファイルをホストにマウントして、必要な EFI バイナリーを展開します。**efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

   b. RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

   c. **grub.cfg** ファイルを編集し、以下の引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
```

```
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (1)
 initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img (2)
}
```

(1) **linux** 行の項目の最初の引数は、TFTP サーバーにアップロードした **kernel** ファイルの場所です。**coreos.inst.image_url** パラメーター値には、HTTP サーバーにアップロードした圧縮されたメタル RAW イメージの場所を指定します。**coreos.inst.ignition_url** パラメーターには、HTTP サーバーにアップロードしたブートストラップ Ingition 設定ファイルの場所を指定します。

(2) TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

8. 継続してクラスターのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

5.3.12. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ (1)
    --log-level=info (2)
```

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

(2) 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

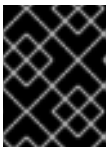
出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
```

```
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

5.3.13. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

5.3.14. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

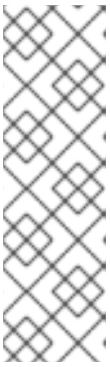
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

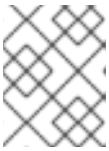
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

5.3.15. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

5.3.15.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

5.3.15.1.1. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、イメージレジストリー Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **ManagementState** イメージレジストリー Operator 設定を **Removed** から **Managed** に変更します。以下は例になります。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

5.3.15.1.2. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

5.3.15.1.3. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

5.3.15.1.4. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

3. 正しい PVC を参照するようにレジストリー設定を編集します。

5.3.16. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
```

```
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running   0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running   0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running   0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8     1/1
Running   0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. Cluster registration ページでクラスターを登録します。

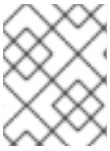
5.3.17. 次のステップ

- クラスターをカスタマイズ します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

第6章 IBM Z および LINUXONE へのインストール

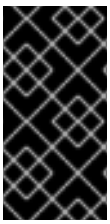
6.1. クラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform version 4.5 では、プロビジョニングする IBM Z または LinuxONE インフラストラクチャーにクラスターをインストールできます。



注記

本書は IBM Z のみを参照しますが、これに含まれるすべての情報は LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

6.1.1. 前提条件

- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- クラスターの NFS を使用した永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

6.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

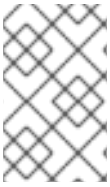
6.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

6.1.3.1. 必要なマシン

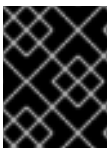
最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。



注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。



重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

6.1.3.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。

6.1.3.3. IBM Z ネットワーク接続の要件

IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM VSWITCH のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

6.1.3.4. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

6.1.3.5. 最小の IBM Z システム要件

OpenShift Container Platform バージョン 4.5 は、以下の IBM ハードウェアにインストールできます。

- IBM Z: z13、z13s、すべての z14 モデル、すべての z15 モデル
- LinuxONE: すべてのモデル

ハードウェア要件

- SMT2 をサポートする 3 IFL 搭載の 1 LPAR
- 1 OSA または RoCE ネットワークアダプター

オペレーティングシステム要件

- z/VM 7.1 の 1 インスタンス

z/VM インスタンスで以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

6.1.3.6. 推奨される IBM Z システム要件

ハードウェア要件

- SMT2 をサポートする 6 IFL 搭載の 3 LPAR
- 1 または 2 OSA または RoCE ネットワークアダプター、またはその両方
- HiperSockets。ノードに直接割り当てられるか、または z/VM ゲストに対して透過性を持たせるために z/VM VSWITCH でブリッジしてノードに割り当てられます。Hipersockets をノードに直接接続するには、RHEL 8 ゲスト経由で外部ネットワークにゲートウェイを設定し、Hipersockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.1 の 2 または 3 インスタンス

z/VM インスタンスで以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)
- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストール

に必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV および High Performance FICON (zHPF) を使用して最適なパフォーマンスを確保します。

- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

6.1.3.7. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

追加リソース

- IBM Knowledge Center の Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。
- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。

6.1.4. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 静的 IP アドレスをセットアップします。
2. FTP サーバーをセットアップします。
3. 必要なロードバランサーをプロビジョニングします。
4. マシンのポートを設定します。
5. DNS を設定します。
6. ネットワーク接続を確認します。

6.1.4.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表6.1 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表6.2 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表6.3 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

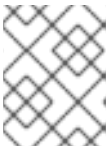
重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

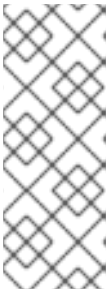
   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表6.4 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表6.5 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

6.1.4.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat

Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表6.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例6.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
```

```
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例6.2 逆引きレコードの DNS ゾーンデータベースの例

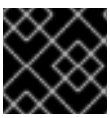
```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

6.1.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。



重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.1.6. インストールプログラムの取得

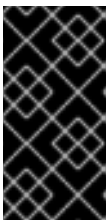
OpenShift Container Platform をインストールする前に、インストールファイルをプロビジョニングマシンにダウンロードします。

前提条件

- Linux を実行するマシンからクラスターをインストールする必要があります (例: Red Hat Enterprise Linux 8)。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

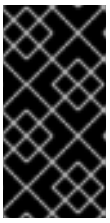
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

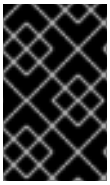
```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、

OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

6.1.7. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

6.1.7.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

6.1.7.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。

3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

6.1.7.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

6.1.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

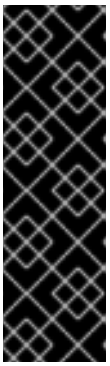
前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```



重要

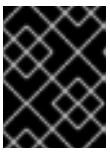
ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

6.1.8.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

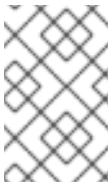
```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
```

```
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場

合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

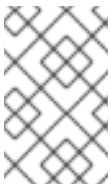
11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

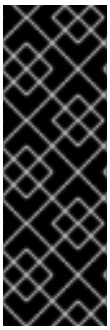


注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

6.1.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
```

```
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.1.10. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

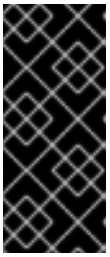
プロビジョニングする IBM Z インフラストラクチャーにクラスターをインストールする前に、クラスターが使用する RHCOS を z/VM ゲスト仮想マシンにインストールする必要があります。マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している FTP サーバー。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) インストールファイルをダウンロードします。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

以下のファイルをダウンロードします。

- initramfs: **rhcos-<version>-installer-initramfs.img**
- kernel: **rhcos-<version>-installer-kernel**
- RHCOS をインストールするディスクのオペレーティングシステムイメージ。このタイプは仮想マシンによって異なる場合があります。
  rhcos-<version>-s390x-dasd.s390x.raw.gz (DASD 用)

  rhcos-<version>-s390x-metal.s390x.raw.gz (FCP 用)

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **coreos.inst.install_dev=** の場合、DASD インストールに **dasda** を指定するか、または FCP に **sda** を指定します。FCP には **zfcp.allow_lun_scan=0** が必要なことに注意してください。
   - **rd.dasd=** の場合、RHCOS がインストールされる DASD を指定します。
   - **rd.zfcp=<adapter>,<wwpn>,<lun>** は、RHCOS をインストールする FCP ディスクを指定します。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。
     v. **hostname.domainname** 形式のマシンホストおよびドメイン名。RHCOS に決定し、設定させる場合は、この値を省略します。
     vi. ネットワークインターフェイス名。RHCOS に決定し、設定させる場合は、この値を省略します。
     vii. 静的 IP アドレスを使用する場合、空の文字列になります。
   - **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。
   - その他のパラメーターはそのまま利用できます。
     ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

     ```
     rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda coreos.inst.image_url=ftp://
     cl1.provide.example.com:8080/assets/rhcos-43.80.20200430.0-s390x-dasd.390x.raw.gz
     coreos.inst.ignition_url=ftp://cl1.provide.example.com:8080/ignition-bootstrap-0
     ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1
     rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 zfcp.allow_lun_scan=0
     cio_ignore=all,
     !condev rd.dasd=0.0.3490
     ```

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM Knowledge Center で PUNCH を参照してください。

ヒント

CP PUNCH コマンドを使用するか、Linux を使用している場合は、 **vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。

7. リーダーからブートストラップマシンに対して IPL を実行します。

```
$ ipl c
```

IBM Knowledge Center で IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

6.1.11. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

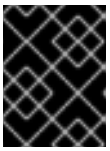
出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
```

```
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

6.1.12. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.1.13. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

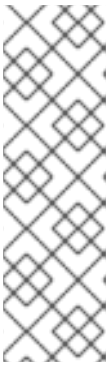
2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE   REQUESTOR                                 CONDITION
csr-mddf5   20m   system:node:master-01.example.com   Approved,Issued
csr-z5rln   16m   system:node:worker-21.example.com   Approved,Issued
```

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.1.14. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

6.1.14.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

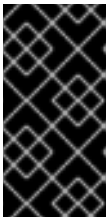
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

6.1.14.1.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



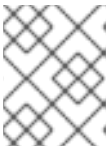
重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

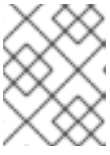


注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```



注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

6.1.14.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

6.1.15. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
```

```
operator-lifecycle-manager-catalog        4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver  4.5.4     True        False         False      18m
service-ca                                4.5.4     True        False         False      23m
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                      NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator   openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running   1          9m
openshift-apiserver            apiserver-67b9g                              1/1     Running   0
3m
openshift-apiserver            apiserver-ljcmx                              1/1     Running   0
1m
openshift-apiserver            apiserver-z25h4                              1/1     Running   0
2m
```

```
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

6.1.16. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

6.1.17. 関連情報

- How to generate SOSREPORT within OpenShift4 nodes without SSH も参照してください。

6.1.18. 次のステップ

- クラスターをカスタマイズ します。

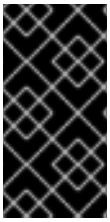
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

6.2. ネットワークが制限された環境でのクラスターの IBM Z および LINUXONE へのインストール

OpenShift Container Platform バージョン 4.5 では、クラスターを制限されたネットワークでプロビジョニングする IBM Z および LinuxONE インフラストラクチャーにクラスターをインストールできます。

注記

本書は IBM Z のみを参照しますが、これに含まれるすべての情報は LinuxONE にも適用されます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

前提条件

- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得します。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- クラスターの NFS を使用して 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

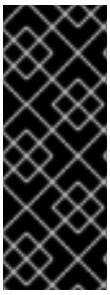
6.2.1. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境

のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

6.2.1.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

6.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

6.2.2.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。

注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

重要

クラスターの高可用性を改善するには、2 つ以上の物理マシンの複数の異なる z/VM インスタンスにコントロールプレーンマシンを分散します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

6.2.2.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。マシンは静的 IP アドレスで設定されます。DHCP サーバーは必要ありません。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。

6.2.2.3. IBM Z ネットワーク接続の要件

IBM Z の z/VM でインストールするには、 レイヤー 2 モードの単一 z/VM 仮想 NIC が必要になります。以下も必要になります。

- 直接接続された OSA または RoCE ネットワークアダプター
- z/VM VSWITCH のセットアップ。推奨されるセットアップでは、OSA リンクアグリゲーションを使用します。

6.2.2.4. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

6.2.2.5. 最小の IBM Z システム要件

OpenShift Container Platform バージョン 4.5 は、以下の IBM ハードウェアにインストールできます。

- IBM Z: z13、z13s、すべての z14 モデル、すべての z15 モデル
- LinuxONE: すべてのモデル

ハードウェア要件

- SMT2 をサポートする 3 IFL 搭載の 1 LPAR
- 1 OSA または RoCE ネットワークアダプター

オペレーティングシステム要件

- z/VM 7.1 の 1 インスタンス

z/VM インスタンスで以下をセットアップします。

- OpenShift Container Platform コントロールプレーンマシンの 3 ゲスト仮想マシン
- OpenShift Container Platform コンピュートマシンの 2 ゲスト仮想マシン
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

6.2.2.6. 推奨される IBM Z システム要件

ハードウェア要件

- SMT2 をサポートする 6 IFL 搭載の 3 LPAR
- 1 または 2 OSA または RoCE ネットワークアダプター、またはその両方
- HiperSockets。ノードに直接割り当てられるか、または z/VM ゲストに対して透過性を持たせるために z/VM VSWITCH でブリッジしてノードに割り当てられます。Hipersockets をノードに直接接続するには、RHEL 8 ゲスト経由で外部ネットワークにゲートウェイを設定し、Hipersockets ネットワークにブリッジする必要があります。

オペレーティングシステム要件

- 高可用性を確保する場合は z/VM 7.1 の 2 または 3 インスタンス

z/VM インスタンスで以下を設定します。

- OpenShift Container Platform コントロールプレーンマシン用に 3 ゲスト仮想マシン (z/VM インスタンスごとに 1 つ)
- OpenShift Container Platform コンピュートマシン用に 6 以上のゲスト仮想マシン (z/VM インスタンス全体に分散)
- 一時 OpenShift Container Platform ブートストラップマシンの 1 ゲスト仮想マシン

z/VM ゲスト仮想マシンのディスクストレージ

- FICON 接続のディスクストレージ (DASD)これらには z/VM ミニディスク、フルパックミニディスク、または専用の DASD を使用でき、これらすべてはデフォルトである CDL としてフォーマットする必要があります。Red Hat Enterprise Linux CoreOS (RHCOS) インストールに必要な最低限の DASD サイズに達するには、拡張アドレスボリューム (EAV) が必要です。利用可能な場合は、HyperPAV および High Performance FICON (zHPF) を使用して最適なパフォーマンスを確保します。
- FCP 接続のディスクストレージ

ストレージ/メインメモリー

- OpenShift Container Platform コントロールプレーンマシン用に 16 GB
- OpenShift Container Platform コンピュートマシン用に 8 GB
- 一時 OpenShift Container Platform ブートストラップマシン用に 16 GB

6.2.2.7. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

追加リソース

- IBM Knowledge Center の Bridging a HiperSockets LAN with a z/VM Virtual Switch を参照してください。
- パフォーマンスの最適化については、Scaling HyperPAV alias devices on Linux guests on z/VM を参照してください。

6.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。
5. ネットワーク接続を確認します。

6.2.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表6.7 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表6.8 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表6.9 コントロールプレーンマシンへのコントロールプレーンマシン

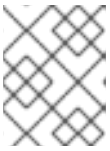
プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

    - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
    - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表6.10 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

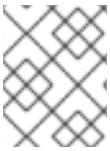
   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表6.11 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

6.2.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表6.12 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>. <base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
マスターホスト	**<master><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例6.3 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
```

```
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例6.4 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

6.2.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

重要

障害復旧およびデバッグが必要な実稼働環境では、この手順を省略しないでください。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

6.2.5. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

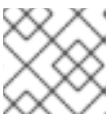
1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

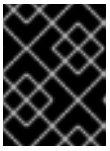
2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

6.2.5.1. IBM Z のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 0 (4)
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 (9)
    hostPrefix: 23 (10)
  networkType: OpenShiftSDN
  serviceNetwork: (11)
  - 172.30.0.0/16
platform:
  none: {} (12)
fips: false (13)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (14)
sshKey: 'ssh-ed25519 AAAA...' (15)
additionalTrustBundle: | (16)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (17)
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_repository>/ocp4/openshift4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container

Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

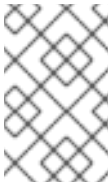
11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Z インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 **additionalTrustBundle** パラメーターおよび値を追加します。この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

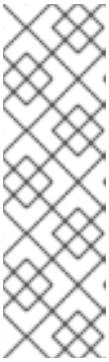
17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

6.2.5.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
```

```
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

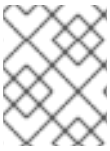
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

6.2.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

   これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

   以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

6.2.7. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

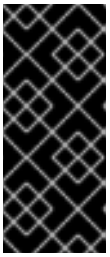
プロビジョニングする IBM Z インフラストラクチャーにクラスターをインストールする前に、クラスターが使用する RHCOS を z/VM ゲスト仮想マシンにインストールする必要があります。マシンを作成するには、以下の手順を実行します。

前提条件

- 作成するマシンがアクセスできるプロビジョニングマシンで稼働している FTP サーバー。

手順

1. プロビジョニングマシンで Linux にログインします。
2. RHCOS イメージミラー から Red Hat Enterprise Linux CoreOS (RHCOS) インストールファイルをダウンロードします。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

以下のファイルをダウンロードします。

- initramfs: **rhcos-<version>-installer-initramfs.img**
- kernel: **rhcos-<version>-installer-kernel**
- RHCOS をインストールするディスクのオペレーティングシステムイメージ。このタイプは仮想マシンによって異なる場合があります。
  rhcos-<version>-s390x-dasd.s390x.raw.gz (DASD 用)

  rhcos-<version>-s390x-metal.s390x.raw.gz (FCP 用)

3. パラメーターファイルを作成します。以下のパラメーターは特定の仮想マシンに固有のものです。
   - **coreos.inst.install_dev=** の場合、DASD インストールに **dasda** を指定するか、または FCP に **sda** を指定します。FCP には **zfcp.allow_lun_scan=0** が必要なことに注意してください。
   - **rd.dasd=** の場合、RHCOS がインストールされる DASD を指定します。
   - **rd.zfcp=<adapter>,<wwpn>,<lun>** は、RHCOS をインストールする FCP ディスクを指定します。
   - **ip=** には、以下の 7 つのエントリーを指定します。
     i. マシンの IP アドレス。
     ii. 空の文字列。
     iii. ゲートウェイ。
     iv. ネットマスク。

v. **hostname.domainname** 形式のマシンホストおよびドメイン名。RHCOS に決定し、設定させる場合は、この値を省略します。

vi. ネットワークインターフェイス名。RHCOS に決定し、設定させる場合は、この値を省略します。

vii. 静的 IP アドレスを使用する場合、空の文字列になります。

- **coreos.inst.ignition_url=** の場合、マシンロールの Ignition ファイルを指定します。**bootstrap.ign**、**master.ign**、または **worker.ign** を使用します。

- その他のパラメーターはそのまま利用できます。
ブートストラップマシンのパラメーターファイルのサンプル **bootstrap-0.parm**:

```
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=dasda coreos.inst.image_url=ftp://
cl1.provide.example.com:8080/assets/rhcos-43.80.20200430.0-s390x-dasd.390x.raw.gz
coreos.inst.ignition_url=ftp://cl1.provide.example.com:8080/ignition-bootstrap-0
ip=172.18.78.2::172.18.78.1:255.255.255.0:::none nameserver=172.18.78.1
rd.znet=qeth,0.0.bdf0,0.0.bdf1,0.0.bdf2,layer2=1,portno=0 zfcp.allow_lun_scan=0
cio_ignore=all,
!condev rd.dasd=0.0.3490
```

4. FTP などを使用し、initramfs、kernel、パラメーターファイル、および RHCOS イメージを z/VM に転送します。FTP でファイルを転送し、仮想リーダーから起動する方法については、Z/VM 環境へのインストール を参照してください。

5. ブートストラップノードになる z/VM ゲスト仮想マシンの仮想リーダーに対してファイルの punch を実行します。
IBM Knowledge Center で PUNCH を参照してください。

ヒント

CP PUNCH コマンドを使用するか、Linux を使用している場合は、 **vmur** コマンドを使用して 2 つの z/VM ゲスト仮想マシン間でファイルを転送できます。

6. ブートストラップマシンで CMS にログインします。

7. リーダーからブートストラップマシンに対して IPL を実行します。

```
$ ipl c
```

IBM Knowledge Center で IPL を参照してください。

8. クラスター内の他のマシンについてこの手順を繰り返します。

6.2.8. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。

- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

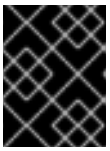
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

6.2.9. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

```
-
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

6.2.10. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

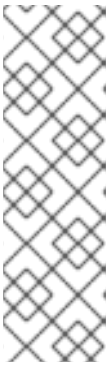
出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

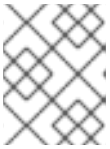
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

6.2.11. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
```

```
authentication                 4.5.4     True        False         False      69s
cloud-credential               4.5.4     True        False         False      12m
cluster-autoscaler             4.5.4     True        False         False      11m
console                        4.5.4     True        False         False      46s
dns                            4.5.4     True        False         False      11m
image-registry                 4.5.4     True        False         False      5m26s
ingress                        4.5.4     True        False         False      5m36s
kube-apiserver                 4.5.4     True        False         False      8m53s
kube-controller-manager        4.5.4     True        False         False      7m24s
kube-scheduler                 4.5.4     True        False         False      12m
machine-api                    4.5.4     True        False         False      12m
machine-config                 4.5.4     True        False         False      7m36s
marketplace                    4.5.4     True        False         False      7m54m
monitoring                     4.5.4     True        False         False      7h54s
network                        4.5.4     True        False         False      5m9s
node-tuning                    4.5.4     True        False         False      11m
openshift-apiserver            4.5.4     True        False         False      11m
openshift-controller-manager   4.5.4     True        False         False      5m943s
openshift-samples              4.5.4     True        False         False      3m55s
operator-lifecycle-manager     4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog   4.5.4     True        False         False      11m
service-ca                     4.5.4     True        False         False      11m
service-catalog-apiserver      4.5.4     True        False         False      5m26s
service-catalog-controller-manager   4.5.4     True        False         False      5m25s
storage                        4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

6.2.11.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

6.2.11.1.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

6.2.11.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

6.2.12. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

   ```
   NAMESPACE                       NAME                                        READY   STATUS
   RESTARTS   AGE
   openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
   Running    1          9m
   openshift-apiserver             apiserver-67b9g                             1/1     Running    0
   3m
   openshift-apiserver             apiserver-ljcmx                             1/1     Running    0
   1m
   openshift-apiserver             apiserver-z25h4                             1/1     Running    0
   2m
   openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8      1/1
   Running    0          5m
   ...
   ```

   b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

   ```
   $ oc logs <pod_name> -n <namespace> 1
   ```

   1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

   Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. Cluster registration ページでクラスターを登録します。

6.2.13. デバッグ情報の収集

IBM Z での OpenShift Container Platform インストールに関する特定の問題のトラブルシューティングおよびデバッグに役立つ可能性のあるデバッグ情報を収集できます。

前提条件

- **oc** CLI ツールをインストールしていること。

手順

1. クラスターにログインします。

```
$ oc login
```

2. ハードウェア情報を収集するノードで、デバッグコンテナーを起動します。

```
$ oc debug node/<nodename>
```

3. **/host** ファイルシステムに切り替え、**toolbox** を起動します。

```
$ chroot /host
$ toolbox
```

4. **dbginfo** データを収集します。

```
$ dbginfo.sh
```

5. その後に、**scp** を使用するなどしてデータを取得できます。

追加リソース

- How to generate SOSREPORT within OpenShift Container Platform version 4 nodes without SSH も参照してください。

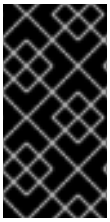
次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。

第7章 IBM POWER へのインストール

7.1. クラスターの IBM POWER へのインストール

OpenShift Container Platform バージョン 4.5 では、プロビジョニングする IBM Power インフラストラクチャーにクラスターをインストールできます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

前提条件

- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。
- クラスターの NFS を使用した永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

7.1.1. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。

- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

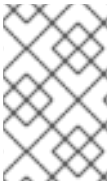
7.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーでのクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

7.1.2.1. 必要なマシン

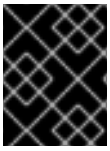
最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。



注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.1.2.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内

の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

7.1.2.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	2	16 GB	120 GB
コントロールプレーン	RHCOS	2	16 GB	120 GB
コンピュート	RHCOS	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

7.1.2.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

7.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。

5. ネットワーク接続を確認します。

7.1.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表7.1 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表7.2 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.3 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件

クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

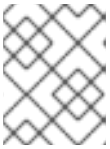
重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。



注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.4 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表7.5 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

7.1.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat

Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表7.6 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

コンポーネント	レコード	説明
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例7.1 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
```

```
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例7.2 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

7.1.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.1.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

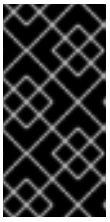
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

7.1.6. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

7.1.6.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.1.6.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。

2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

7.1.6.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

7.1.7. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

7.1.7.1. IBM Power のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

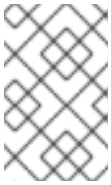
```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
```

```
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths": ...}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。



注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP

アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Power インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

7.1.8. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
```

```
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

7.1.9. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用する手順を実行してマシンを作成することができます。

7.1.9.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

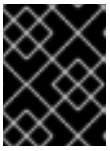
ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したコントロールプレーン、コンピュート、およびブートストラップ Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. RHCOS イメージミラー ページからオペレーティングシステムのインスタンスをインストールするために優先される方法で必要な RHCOS イメージを取得します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ISO ファイルおよび RAW ディスクファイルをダウンロードする必要があります。これらのファイルの名前は以下の例のようになります。

- ISO: **rhcos-<version>-installer.<architecture>.iso**
- 圧縮された metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**

3. RAW RHCOS イメージファイルのいずれかを HTTP サーバーにアップロードし、その URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - LOM インターフェイスで ISO リダイレクトを使用します。
5. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。
6. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

1 インストール先のシステムのブロックデバイスを指定します。

2 サーバーにアップロードした RAW イメージの URL を指定します。

3 このマシンタイプの Ignition 設定ファイルの URL を指定します。

4 **ip=dhcp** を設定するか、各ノードに個別の静的 IP アドレス (**ip=**) および DNS サーバー (**nameserver=**) を設定します。詳細は、**高度なネットワークの設定**を参照してください。

5 複数のネットワークインターフェイスまたは DNS サーバーを使用する場合は、**高度なネットワークの設定**を参照してください。

6 オプションで、**高度なネットワークの設定**で説明されているように、**bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。

7. Enter を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。
8. 継続してクラスターのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

7.1.9.2. PXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。PXE ブートを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- 使用しているコンピューターからアクセス可能な HTTP サーバーおよび TFTP サーバーへのアクセスがあること。

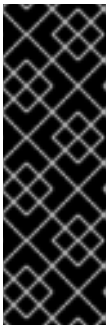
手順

1. インストールプログラムが作成したマスター、ワーカーおよびブートストラップの Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから圧縮された metal RAW イメージ、**kernel** および **initramfs** ファイルを取得します。



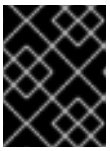
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には RAW イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン名が含まれます。以下の例のようになります。

- 圧縮されたメタル RAW イメージ: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
- **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**

3. RAW イメージを HTTP サーバーにアップロードします。
4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーにアップロードします。
   - iPXE の場合、**kernel** および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージに PXE インストールを設定します。
   ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2)
(3)
```

(1) TFTP サーバーで利用可能な **kernel** ファイルの場所を指定します。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定し

3 HTTP または TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は、TFTP サーバーの **initramfs** ファイルの場所です。**coreos.inst.image_url** パラメーター値は、HTTP サーバーの圧縮されたメタル RAW イメージの場所であり、 **coreos.inst.ignition_url** パラメーター値は HTTP サーバーのブートストラップ Ignition 設定ファイルの場所になります。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. UEFI を使用する場合は、以下の操作を実行します。

a. システムの起動に必要な EFI バイナリーおよび **grub.cfg** ファイルを提供します。**shim.efi** バイナリーと **grubx64.efi** バイナリーが必要です。

- RHCOS ISO をホストにマウントし、**images/efiboot.img** ファイルをホストにマウントして、必要な EFI バイナリーを展開します。**efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

b. RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

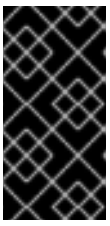
c. **grub.cfg** ファイルを編集し、以下の引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
 initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

1 **linux** 行の項目の最初の引数は、TFTP サーバーにアップロードした **kernel** ファイルの場所です。**coreos.inst.image_url** パラメーター値には、HTTP サーバーにアップロードした圧縮されたメタル RAW イメージの場所を指定します。**coreos.inst.ignition_url** パラメーターには、HTTP サーバーにアップロードしたブートストラップ Ingition 設定ファイルの場所を指定します。

2 TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

7.1.10. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

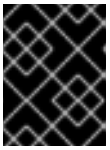
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

7.1.11. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.1.12. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.1.13. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

7.1.13.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

7.1.13.1.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

7.1.13.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

7.1.14. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
```

```
operator-lifecycle-manager-catalog        4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver  4.5.4     True        False         False      18m
service-ca                                4.5.4     True        False         False      23m
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

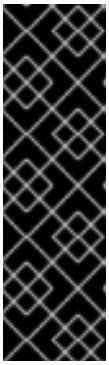
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running     0
2m
```

```
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

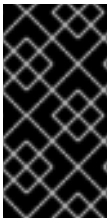
Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

7.2. ネットワークが制限された環境での IBM POWER へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、クラスターをネットワークが制限された環境でプロビジョニングする IBM Power インフラストラクチャーにインストールできます。



重要

ベアメタルプラットフォーム以外の場合には、追加の考慮点を検討する必要があります。OpenShift Container Platform クラスターをインストールする前に、guidelines for deploying OpenShift Container Platform on non-tested platforms にある情報を確認してください。

前提条件

- ネットワークが制限された環境でインストールのミラーレジストリーを作成 し、お使いの OpenShift Container Platform のバージョンの **imageContentSources** データを取得します。**docker.io/ibmcom/registry-ppc64le:2.6.2.5** イメージを使用します。
- インストールプロセスを開始する前に、既存のインストールファイルを移動するか、または削除する必要があります。これにより、インストールプロセス時に必要なインストールファイルが作成され、更新されます。

重要

インストールメディアにアクセスできるマシンからインストール手順が実行されるようにします。

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

7.2.1. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

7.2.1.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

7.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

7.2.2.1. 必要なマシン

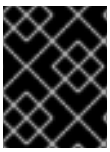
最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン

- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。

注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

7.2.2.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

7.2.2.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	2	16 GB	120 GB
コントロールプレーン	RHCOS	2	16 GB	120 GB
コンピュート	RHCOS	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

7.2.2.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

7.2.3. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。
5. ネットワーク接続を確認します。

7.2.3.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表7.7 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn

プロトコル	ポート	説明
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表7.8 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表7.9 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

   注記

   API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

   表7.10 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

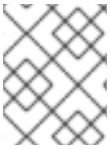
表7.11 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

追加リソース

- chrony タイムサービスの設定

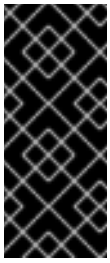
7.2.3.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表7.12 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例7.3 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例7.4 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
```

```
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

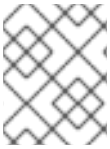
7.2.4. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

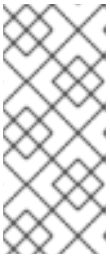
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

7.2.5. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

7.2.5.1. IBM Power のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com 1
compute: 2
- hyperthreading: Enabled 3
  name: worker
  replicas: 0 4
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3 7
metadata:
  name: test 8
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14 9
    hostPrefix: 23 10
  networkType: OpenShiftSDN
  serviceNetwork: 11
  - 172.30.0.0/16
platform:
  none: {} 12
fips: false 13
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 14
sshKey: 'ssh-ed25519 AAAA...' 15
additionalTrustBundle: | 16
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 17
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、**compute** セクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッド (SMT) または **hyperthreading** を有効/無効にするかどうか。デフォルトでは、SMT はマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。SMT を無効にする場合、これをすべてのクラスターマシンで無効にする必要があります。これにはコントロールプレーンとコンピュートマシンの両方が含まれます。

注記

同時マルチスレッド (SMT) はデフォルトで有効になっています。SMT が BIOS 設定で有効になっていない場合は、**hyperthreading** パラメーターは効果がありません。

重要

BIOS または **install-config.yaml** であるかに関係なく **hyperthreading** を無効にする場合、容量計画においてマシンのパフォーマンスの大幅な低下が考慮に入れられていることを確認します。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるイ

ンフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 Pod IP アドレスの割り当てに使用する IP アドレスのブロック。このブロックは既存の物理ネットワークと重複できません。これらの IP アドレスは Pod ネットワークに使用されます。外部ネットワークから Pod にアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定する必要があります。

10 それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定され、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。外部ネットワークからのノードへのアクセスを提供する必要がある場合には、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

11 サービス IP アドレスに使用する IP アドレスプール。1 つの IP アドレスプールのみを入力できます。外部ネットワークからサービスにアクセスする必要がある場合、ロードバランサーおよびルーターを、トラフィックを管理するように設定します。

12 プラットフォームを **none** に設定する必要があります。IBM Power インフラストラクチャー用に追加のプラットフォーム設定変数を指定できません。

13 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

14 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

15 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

16 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

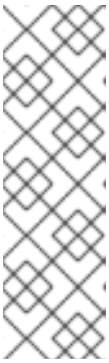
17 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

7.2.5.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

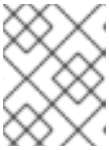
1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これら

のコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

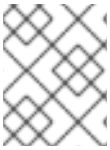


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

7.2.6. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

5. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

6. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

7.2.7. Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージまたはネットワーク PXE ブートを使用する手順を実行してマシンを作成することができます。

7.2.7.1. ISO イメージを使用した Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。ISO イメージを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあること。

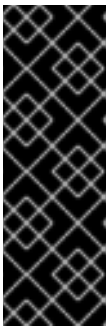
手順

1. インストールプログラムが作成したコントロールプレーン、コンピュート、およびブートストラップ Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. RHCOS イメージミラー ページからオペレーティングシステムのインスタンスをインストールするために優先される方法で必要な RHCOS イメージを取得します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には ISO イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ISO ファイルおよび RAW ディスクファイルをダウンロードする必要があります。これらのファイルの名前は以下の例のようになります。

- ISO: **rhcos-<version>-installer.<architecture>.iso**
- 圧縮された metal RAW: **rhcos-<version>-metal.<architecture>.raw.gz**

3. RAW RHCOS イメージファイルのいずれかを HTTP サーバーにアップロードし、その URL をメモします。

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. ISO を使用し、RHCOS インストールを開始します。以下のインストールオプションのいずれかを使用します。
   - ディスクに ISO イメージを書き込み、これを直接起動します。
   - LOM インターフェイスで ISO リダイレクトを使用します。
5. インスタンスの起動後に、**TAB** または **E** キーを押してカーネルコマンドラインを編集します。
6. パラメーターをカーネルコマンドラインに追加します。

```
coreos.inst=yes
coreos.inst.install_dev=sda 1
coreos.inst.image_url=<image_URL> 2
coreos.inst.ignition_url=http://example.com/config.ign 3
ip=<dhcp or static IP address> 4 5
bond=<bonded_interface> 6
```

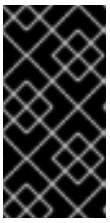
1. インストール先のシステムのブロックデバイスを指定します。
2. サーバーにアップロードした RAW イメージの URL を指定します。
3. このマシンタイプの Ignition 設定ファイルの URL を指定します。
4. **ip=dhcp** を設定するか、各ノードに個別の静的 IP アドレス (**ip=**) および DNS サーバー (**nameserver=**) を設定します。詳細は、**高度なネットワークの設定**を参照してください。

5 複数のネットワークインターフェイスまたは DNS サーバーを使用する場合は、**高度なネットワークの設定**を参照してください。

6 オプションで、**高度なネットワークの設定**で説明されているように、**bond=** オプションを使用して、複数のネットワークインターフェイスを単一のインターフェイスにボンディングできます。

7. Enter を押してインストールを完了します。RHCOS のインストール後に、システムは再起動します。システムの再起動後、指定した Ignition 設定ファイルを適用します。

8. 継続してクラスターのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

7.2.7.2. PXE ブートによる Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

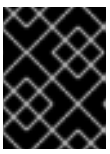
ユーザーによってプロビジョニングされる IBM Power インフラストラクチャーにクラスターをインストールする前に、それが使用する RHCOS マシンを作成する必要があります。PXE ブートを使用してマシンを作成することができます。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- 使用しているコンピューターからアクセス可能な HTTP サーバーおよび TFTP サーバーへのアクセスがあること。

手順

1. インストールプログラムが作成したマスター、ワーカーおよびブートストラップの Ignition 設定を HTTP サーバーにアップロードします。これらのファイルの URL をメモします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

2. Red Hat カスタマーポータルの 製品のダウンロード ページまたは RHCOS イメージミラー ページから圧縮された metal RAW イメージ、**kernel** および **initramfs** ファイルを取得します。

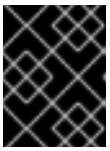
重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。この手順には RAW イメージのみを使用します。RHCOS qcow2 イメージは、ベアメタルのインストールではサポートされません。

ファイル名には、OpenShift Container Platform のバージョン名が含まれます。以下の例のようになります。

- 圧縮されたメタル RAW イメージ: **rhcos-<version>-<architecture>-metal.<architecture>.raw.gz**
- **kernel**: **rhcos-<version>-<architecture>-installer-kernel-<architecture>**
- **initramfs**: **rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img**

3. RAW イメージを HTTP サーバーにアップロードします。
4. 使用する起動方法に必要な追加ファイルをアップロードします。
   - 従来の PXE の場合、**kernel** および **initramfs** ファイルを TFTP サーバーにアップロードします。
   - iPXE の場合、**kernel** および **initramfs** ファイルを HTTP サーバーにアップロードします。



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

5. RHCOS のインストール後にマシンがローカルディスクから起動されるようにネットワークブートインフラストラクチャーを設定します。
6. RHCOS イメージに PXE インストールを設定します。
   ご使用の環境についての以下の例で示されるメニューエントリーのいずれかを変更し、イメージおよび Ignition ファイルが適切にアクセスできることを確認します。
   - PXE の場合:

```
DEFAULT pxeboot
TIMEOUT 20
PROMPT 0
LABEL pxeboot
    KERNEL rhcos-<version>-<architecture>-installer-kernel-<architecture> (1)
    APPEND ip=dhcp rd.neednet=1 initrd=rhcos-<version>-<architecture>-installer-
initramfs.<architecture>.img coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign (2)
(3)
```

(1) TFTP サーバーで利用可能な **kernel** ファイルの場所を指定します。

2 複数の NIC を使用する場合、**ip** オプションに単一インターフェイスを指定します。たとえば、**eno1** という名前の NIC で DHCP を使用するには、 **ip=eno1:dhcp** を設定し

3 HTTP または TFTP サーバーにアップロードした RHCOS ファイルの場所を指定します。**initrd** パラメーター値は、TFTP サーバーの **initramfs** ファイルの場所です。**coreos.inst.image_url** パラメーター値は、HTTP サーバーの圧縮されたメタル RAW イメージの場所であり、 **coreos.inst.ignition_url** パラメーター値は HTTP サーバーのブートストラップ Ignition 設定ファイルの場所になります。

注記

この設定では、グラフィカルコンソールを使用するマシンでシリアルコンソールアクセスを有効にしません。別のコンソールを設定するには、**APPEND** 行に 1 つ以上の **console=** 引数を追加します。たとえば、**console=tty0 console=ttyS0** を追加して、最初の PC シリアルポートをプライマリーコンソールとして、グラフィカルコンソールをセカンダリーコンソールとして設定します。詳細は、How does one set up a serial terminal and/or console in Red Hat Enterprise Linux? を参照してください。

7. UEFI を使用する場合は、以下の操作を実行します。

a. システムの起動に必要な EFI バイナリーおよび **grub.cfg** ファイルを提供します。**shim.efi** バイナリーと **grubx64.efi** バイナリーが必要です。

- RHCOS ISO をホストにマウントし、**images/efiboot.img** ファイルをホストにマウントして、必要な EFI バイナリーを展開します。**efiboot.img** マウントポイントから、**EFI/redhat/shimx64.efi** および **EFI/redhat/grubx64.efi** ファイルを TFTP サーバーにコピーします。

```
# mkdir -p /mnt/{iso,efiboot}
# mount -o loop rhcos-installer.x86_64.iso /mnt/iso
# mount -o loop,ro /mnt/iso/images/efiboot.img /mnt/efiboot
# cp /mnt/efiboot/EFI/redhat/{shimx64.efi,grubx64.efi} .
# umount /mnt/{efiboot,iso}
```

b. RHCOS ISO に含まれている **EFI/redhat/grub.cfg** ファイルを TFTP サーバーにコピーします。

c. **grub.cfg** ファイルを編集し、以下の引数を追加します。

```
menuentry 'Install Red Hat Enterprise Linux CoreOS' --class fedora --class gnu-linux --
class gnu --class os {
 linux rhcos-<version>-<architecture>-installer-kernel-<architecture> nomodeset
rd.neednet=1 coreos.inst=yes coreos.inst.install_dev=sda
coreos.inst.image_url=http://<HTTP_server>/rhcos-<version>-<architecture>-metal.
<architecture>.raw.gz coreos.inst.ignition_url=http://<HTTP_server>/bootstrap.ign 1
 initrd rhcos-<version>-<architecture>-installer-initramfs.<architecture>.img 2
}
```

1 **linux** 行の項目の最初の引数は、TFTP サーバーにアップロードした **kernel** ファイルの場所です。**coreos.inst.image_url** パラメーター値には、HTTP サーバーにアップロードした圧縮されたメタル RAW イメージの場所を指定します。**coreos.inst.ignition_url** パラメーターには、HTTP サーバーにアップロードしたブートストラップ Ingition 設定ファイルの場所を指定します。

2 TFTP サーバーにアップロードした **initramfs** ファイルの場所を指定します。

8. 継続してクラスターのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。コントロールプレーンマシンがデフォルトのスケジュール対象にされていない場合、クラスターのインストール前に少なくとも 2 つのコンピュートマシンを作成します。

7.2.8. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

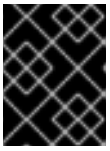
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

7.2.9. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

7.2.10. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

7.2.11. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

7.2.11.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

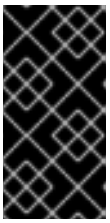
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

7.2.11.1.1. ベアメタルの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- ベアメタル上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

7.2.11.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

7.2.12. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
```

```
operator-lifecycle-manager-catalog        4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver  4.5.4     True        False         False      18m
service-ca                                4.5.4     True        False         False      23m
service-catalog-apiserver                 4.5.4     True        False         False      23m
service-catalog-controller-manager        4.5.4     True        False         False      23m
storage                                   4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

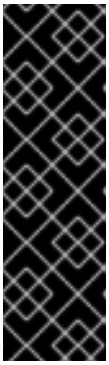
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                       NAME                                         READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver             apiserver-67b9g                              1/1     Running     0
3m
openshift-apiserver             apiserver-ljcmx                              1/1     Running     0
1m
openshift-apiserver             apiserver-z25h4                              1/1     Running     0
2m
```

```
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。

第8章 OPENSTACK へのインストール

8.1. カスタマイズによる OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、Red Hat OpenStack Platform (RHOSP) にカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に **install-config.yaml** でパラメーターを変更します。

8.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
    - OpenShift Container Platform 4.5 が **Available platforms** セクションの RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。
- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている必要があります。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、Optimizing storage を参照してください。
- RHOSP でメタデータサービスが有効化されています。

8.1.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

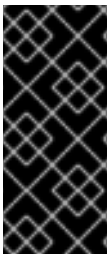
OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表8.1 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB

リソース	値
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

8.1.2.1. コントロールプレーンおよびコンピュートマシン

デフォルトで、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンおよび 3 つのコンピュートマシンを使用します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

8.1.2.2. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

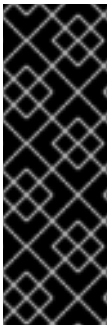
8.1.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

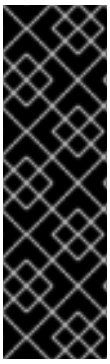


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.1.4. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。



重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

8.1.5. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

-

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

重要

外部ネットワークの CIDR 範囲がデフォルトのネットワーク範囲のいずれかと重複している場合、インストールプロセスを開始する前に、**install-config.yaml** ファイルで一致するネットワーク範囲を変更する必要があります。

デフォルトのネットワーク範囲は以下のとおりです。

ネットワーク	範囲
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

警告

インストールプログラムにより同じ名前を持つ複数のネットワークが見つかる場合、それらのネットワークのいずれかがランダムに設定されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

注記

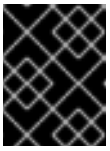
Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

8.1.6. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。



重要

パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

- RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. マシンを認証局の信頼バンドルに追加します。

   ```
   $ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
   ```

   c. 信頼バンドルを更新します。

   ```
   $ sudo update-ca-trust extract
   ```

   d. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

   ```
   clouds:
     shiftstack:
       ...
       cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
   ```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

8.1.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

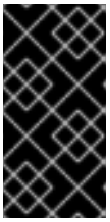
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.1.8. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat OpenStack Platform (RHOSP)

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

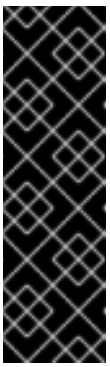
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

viii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

ix. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

x. コントロールプレーンおよびコンピュートノードに使用する 16 GB 以上の RAM で RHOSP フレーバーを指定します。

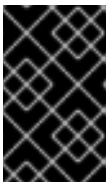
xi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

xii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

xiii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

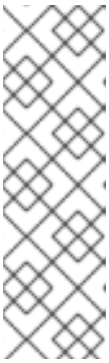
install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

8.1.8.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。



注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

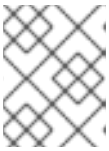
1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これら

openshift-config namespace を生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。



注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

8.1.9. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

8.1.9.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.2 必須パラメーター

パラメーター	説明	値

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.1.9.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表8.3 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

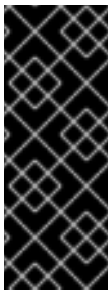
パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

8.1.9.3. オプションの設定パラメーター

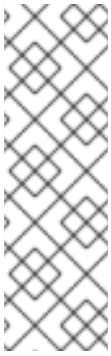
オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.4 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**

パラメーター	説明	値
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

8.1.9.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表8.5 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。

パラメーター	説明	値
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。	文字列 (例: **m1.xlarge**)。
platform.openstack.lbFloatingIP	ロードバランサー API に関連付ける既存の Floating IP アドレス。	IP アドレス (例: **128.0.0.1**)。

8.1.9.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表8.6 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。 この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

8.1.9.6. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表8.7 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

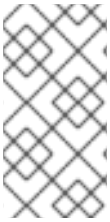
8.1.9.7. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。ノードとポートはこの上に作成されます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、以下を確認します。

- ターゲットネットワークおよびサブネットが利用可能である。
- DHCP がターゲットサブネットで有効にされている。
- ターゲットネットワーク上でポートを作成するためのパーミッションがあるインストーラー認証情報を指定できます。
- ネットワーク設定にルーターが必要な場合、これは RHOSP で作成されます。一部の設定は、Floating IP アドレスの変換用のルーターに依存します。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。



注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

8.1.9.8. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

8.1.10. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.1.11. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

OpenShift Container Platform API およびクラスターで実行されるアプリケーションを、floating IP アドレスを使用/不使用でアクセス可能になるように設定できます。

8.1.11.1. floating IP アドレスを使ったアクセスの有効化

2 つの floating IP (FIP) アドレスを作成します。1 つ目は OpenShift Container Platform API への外部アクセス用の **API FIP** であり、2 つ目は OpenShift Container Platform アプリケーション用の **apps FIP** です。

重要

API FIP も **install-config.yaml** ファイルで使用されます。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. 新しい FIP を反映させるには、以下のパターンに続くレコードを DNS サーバーに追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御しない場合は、代わりに **/etc/hosts** ファイルにレコードを追加します。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

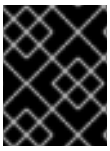
8.1.11.2. Floating IP アドレスを使用しないアクセスの有効化

Floating IP アドレスを使用できない場合でも、OpenShift Container Platform のインストールは終了できる可能性があります。ただし、インストールプログラムは API アクセスを待機してタイムアウトする場合は失敗します。

インストールプログラムがタイムアウトすると、クラスターは初期化される可能性があります。ブートストラップ処理が開始されたら、これを完了する必要があります。デプロイ後にクラスターのネットワーク設定を編集する必要があります。

8.1.12. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

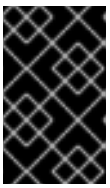
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

8.1.13. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

8.1.14. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.1.15. Floating IP アドレスを使用したアプリケーションアクセスの設定

OpenShift Container Platform をインストールした後に、アプリケーションネットワークトラフィックを許可するように Red Hat OpenStack Platform (RHOSP) を設定します。

前提条件

- OpenShift Container Platform クラスターがインストールされている必要があります。
- **環境へのアクセスの有効化**で説明されているように、Floating IP アドレスが有効化されています。

手順

OpenShift Container Platform クラスターをインストールした後に、Floating IP アドレスを Ingress ポートに割り当てます。

1. ポートを表示します。

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. ポートを IP アドレスに接続します。

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. ***apps.** のワイルドカード **A** レコードを DNS ファイルに追加します。

```
*.apps.<cluster name>.<base domain>  IN  A  <apps FIP>
```

注記

DNS サーバーを制御せず、非実稼働環境でアプリケーションアクセスを有効にする必要がある場合は、これらのホスト名を **/etc/hosts** に追加できます。

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.1.16. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラストラフィックを設定 します。

8.2. KURYR を使用する OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、Kuryr SDN を使用する Red Hat OpenStack Platform (RHOSP) にカスタマイズされたクラスターをインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に **install-config.yaml** でパラメーターを変更します。

8.2.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
  - OpenShift Container Platform 4.5 が **Available platforms** セクションの RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。
- ブロックストレージ (Cinder) またはオブジェクトストレージ (Swift) などのストレージサービスが RHOSP にインストールされている必要があります。オブジェクトストレージは、OpenShift Container Platform レジストリークラスターデプロイメントに推奨されるストレージ技術です。詳細は、Optimizing storage を参照してください。

8.2.2. Kuryr SDN について

Kuryr は、Neutron および Octavia Red Hat OpenStack Platform (RHOSP) サービスを使用して Pod およびサービスのネットワークを提供する Container Network Interface (CNI) プラグインです。

Kuryr と OpenShift Container Platform の統合は主に、RHOSP の仮想マシンで実行する OpenShift Container Platform クラスター用に設計されました。Kuryr は、OpenShift Container Platform Pod を RHOSP SDN にプラグインしてネットワークのパフォーマンスを強化します。さらに、これは Pod と RHOSP 仮想インスタンス間の接続を可能にします。

Kuryr コンポーネントは **openshift-kuryr** namespace を使用して OpenShift Container Platform の Pod としてインストールされます。

- **kuryr-controller**: **master** ノードにインストールされる単一のサービスインスタンスです。これは、OpenShift Container Platform で **Deployment** としてモデリングされます。
- **kuryr-cni**: 各 OpenShift Container Platform ノードで Kuryr を CNI ドライバーとしてインストールし、設定するコンテナーです。これは、OpenShift Container Platform で **DaemonSet** オブジェクトとしてモデリングされます。

Kuryr コントローラーは OpenShift Container Platform API サーバーで Pod、サービスおよび namespace の作成、更新、および削除イベントについて監視します。これは、OpenShift Container Platform API 呼び出しを Neutron および Octavia の対応するオブジェクトにマップします。そのため、Neutron トランクポート機能を実装するすべてのネットワークソリューションを使用して、Kuryr 経由で OpenShift Container Platform をサポートすることができます。これには、Open vSwitch (OVS) および Open Virtual Network (OVN) などのオープンソースソリューションや Neutron と互換性のある市販の SDN が含まれます。

Kuryr は、カプセル化された RHOSP テナントネットワーク上の OpenShift Container Platform デプロイメントに使用することが推奨されています。これは、 RHOSP ネットワークでカプセル化された OpenShift Container Platform SDN を実行するなど、二重のカプセル化を防ぐために必要です。

プロバイダーネットワークまたはテナント VLAN を使用する場合は、二重のカプセル化を防ぐために Kuryr を使用する必要はありません。パフォーマンス上の利点はそれほど多くありません。ただし、設定によっては、Kuryr を使用して 2 つのオーバーレイが使用されないようにすることには利点がある場合があります。

Kuryr は、以下のすべての基準が true であるデプロイメントでは推奨されません。

- RHOSP のバージョンが 16 よりも前のバージョンである。
- デプロイメントで UDP サービスが使用されているか、または少数のハイパーバイザーで多数の TCP サービスが使用されている。

または、以下を実行します。

- **ovn-octavia** Octavia ドライバーが無効にされている。
- デプロイメントで、少数のハイパーバイザーで多数の TCP サービスが使用されている。

8.2.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン

Kuryr SDN を使用する場合、Pod、サービス、namespace およびネットワークポリシーは RHOSP クォータのリソースを使用します。これにより、最小要件が増加します。また、Kuryr にはデフォルトインストールに必要な要件以外の追加要件があります。

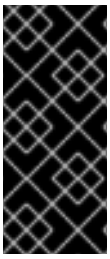
以下のクォータを使用してデフォルトのクラスターの最小要件を満たすようにします。

表8.8 Kuryr を使用する RHOSP のデフォルト OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3: LoadBalancer タイプに予想されるサービス数
ポート	1500: Pod ごとに 1 つ必要
ルーター	1
サブネット	250: namespace/プロジェクトごとに 1 つ必要
ネットワーク	250: namespace/プロジェクトごとに 1 つ必要
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	250: サービスおよび NetworkPolicy ごとに 1 つ必要
セキュリティーグループルール	1000
ロードバランサー	100: サービスごとに 1 つ必要
ロードバランサーリスナー	500: サービスで公開されるポートごとに 1 つ必要

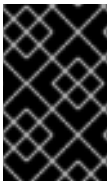
リソース	値
ロードバランサーノード	500: サービスで公開されるポートごとに 1 つ必要

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



重要

OVN Octavia ドライバーではなく Amphora ドライバーで Red Hat OpenStack Platform(RHOSP) バージョン 16 を使用している場合、セキュリティーグループはユーザープロジェクトではなくサービスアカウントに関連付けられます。

リソースを設定する際には、以下の点に注意してください。

- 必要なポート数は Pod 数よりも大きくなる。Kuryr はポートプールを使用して、事前に作成済みのポートを Pod で使用できるようにし、Pod の起動時間を短縮します。
- 各ネットワークポリシーは RHOSP セキュリティーグループにマップされ、**NetworkPolicy** 仕様によっては 1 つ以上のルールがセキュリティーグループに追加される。
- 各サービスは RHOSP ロードバランサーにマップされる。クォータに必要なセキュリティーグループの数を見積もる場合には、この要件を考慮してください。
  RHOSP バージョン 15 以前のバージョン、または **ovn-octavia driver** を使用している場合、各ロードバランサーにはユーザープロジェクトと共にセキュリティーグループがあります。
- クォータはロードバランサーのリソース (VM リソースなど) を考慮しませんが、RHOSP デプロイメントのサイズを決定する際にはこれらのリソースを考慮する必要があります。デフォルトのインストールには 50 を超えるロードバランサーが あり、クラスターはそれらのロードバランサーに対応できる必要があります。
  OVN Octavia ドライバーを有効にして RHOSP バージョン 16 を使用している場合は、1 つのロードバランサー仮想マシンのみが生成され、サービスは OVN フロー経由で負荷分散されます。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

Kuryr SDN を有効にするには、使用する環境が以下の要件を満たしている必要があります。

- RHOSP 13+ を実行します。
- オーバークラウドと Octavia を使用します。
- Neutron トランクポートの拡張を使用します。

- ML2/OVS Neutron ドライバーが **ovs-hybrid** の代わりに使用れる場合、**openvswitch** ファイアウォールドライバーを使用します。

8.2.3.1. クォータの拡大

Kuryr SDN を使用する場合、Pod、サービス、namespace、およびネットワークポリシーが使用する Red Hat OpenStack Platform (RHOSP) リソースに対応するためにクォータを引き上げる必要があります。

手順

- 以下のコマンドを実行して、プロジェクトのクォータを増やします。

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets 250 --networks 250 <project>
```

8.2.3.2. Neutron の設定

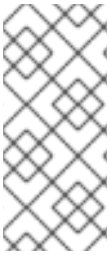
Kuryr CNI は Neutron トランクの拡張を使用してコンテナーを Red Hat OpenStack Platform (RHOSP) SDN にプラグインします。したがって、Kuryr が適切に機能するには **trunks** 拡張を使用する必要があります。

さらにデフォルトの ML2/OVS Neutron ドライバーを使用する場合には、セキュリティーグループがトランクサブポートで実行され、Kuryr がネットワークポリシーを適切に処理できるように、**ovs_hybrid** ではなく **openvswitch** に設定される必要があります。

8.2.3.3. Octavia の設定

Kuryr SDN は Red Hat OpenStack Platform (RHOSP) の Octavia LBaaS を使用して OpenShift Container Platform サービスを実装します。したがって、Kuryr SDN を使用するように RHOSP に Octavia コンポーネントをインストールし、設定する必要があります。

Octavia を有効にするには、Octavia サービスを RHOSP オーバークラウドのインストール時に組み込むか、またはオーバークラウドがすでに存在する場合は Octavia サービスをアップグレードする必要があります。Octavia を有効にする以下の手順は、オーバークラウドのクリーンインストールまたはオーバークラウドの更新の両方に適用されます。



注記

以下の手順では、Octavia を使用する場合に RHOSP のデプロイメント 時に必要となる主な手順のみを説明します。また、レジストリーメソッド が変更されることにも留意してください。

以下の例では、ローカルレジストリーの方法を使用しています。

手順

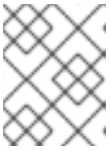
1. ローカルレジストリーを使用している場合、イメージをレジストリーにアップロードするためのテンプレートを作成します。以下に例を示します。

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
```

```
--prefix=openstack- \
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. **local_registry_images.yaml** ファイルに Octavia イメージが含まれることを確認します。以下に例を示します。

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```



注記

Octavia コンテナーのバージョンは、インストールされている特定の RHOSP リリースによって異なります。

3. コンテナーイメージを **registry.redhat.io** からアンダークラウドノードにプルします。

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

これには、ネットワークおよびアンダークラウドディスクの速度に応じて多少の時間がかかる可能性があります。

4. Octavia ロードバランサーは OpenShift Container Platform API にアクセスするために使用されるため、それらのリスナーの接続のデフォルトタイムアウトを増やす必要があります。デフォルトのタイムアウトは 50 秒です。以下のファイルをオーバークラウドのデプロイコマンドに渡し、タイムアウトを 20 分に増やします。

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```

注記

これは RHOSP 13.0.13+ では不要です。

5. Octavia を使用してオーバークラウドをインストールまたは更新します。

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

注記

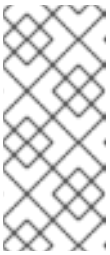
このコマンドには、Octavia に関連付けられたファイルのみが含まれます。これは、RHOSP の特定のインストールによって異なります。詳細は RHOSP のドキュメントを参照してください。Octavia インストールのカスタマイズについての詳細は、Octavia デプロイメントのプランニング を参照してください。

注記

Kuryr SDN を利用する際には、オーバークラウドのインストールに Neutron の **trunk** 拡張機能が必要です。これは、Director デプロイメントでデフォルトで有効にされます。Neutron バックエンドが ML2/OVS の場合、デフォルトの **ovs-hybrid** の代わりに **openvswitch** ファイアウォールを使用します。バックエンドが ML2/OVN の場合には変更の必要がありません。

6. RHOSP の 13.0.13 よりも前のバージョンでは、プロジェクトの作成後にプロジェクト ID を **octavia.conf** 設定ファイルに追加します。

- トラフィックが Octavia ロードバランサーを通過する場合など、複数のサービス全体でネットワークポリシーを実行するには、Octavia がユーザープロジェクトで Amphora 仮想マシンセキュリティーグループを作成するようにする必要があります。
この変更により、必要なロードバランサーのセキュリティーグループがそのプロジェクトに属し、それらをサービスの分離を実行するように更新できます。



注記

RHOSP の 13.0.13 よりも後のバージョンでは、このタスクは必要ありません。

Octavia は、ロードバランサー VIP へのアクセスを制限する新しい ACL API を実装します。

a. プロジェクト ID を取得します。

```
$ openstack project show <project>
```

出力例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. プロジェクト ID をコントローラーの **octavia.conf** に追加します。

i. **stackrc** ファイルを取得します。

```
$ source stackrc  # Undercloud credentials
```

ii. オーバークラウドコントローラーを一覧表示します。

```
$ openstack server list
```

出力例

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii. コントローラーに対して SSH を実行します。

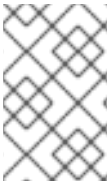
```
$ ssh heat-admin@192.168.24.8
```

iv. **octavia.conf** ファイルを編集して、プロジェクトを Amphora セキュリティーグループがユーザーのアカウントに設定されているプロジェクトの一覧に追加します。

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c. 新しい設定が読み込まれるように Octavia ワーカーを再起動します。

```
controller-0$ sudo docker restart octavia_worker
```



注記

RHOSP 環境によっては、Octavia は UDP リスナーをサポートしない可能性があります。RHOSP の 13.0.13 よりも前のバージョンで Kuryr SDN を使用する場合、UDP サービスはサポートされません。RHOSP バージョン 16 以降は UDP をサポートします。

8.2.3.3.1. Octavia OVN ドライバー

Octavia は Octavia API を使用して複数のプロバイダードライバーをサポートします。

利用可能なすべての Octavia プロバイダードライバーをコマンドラインで表示するには、以下を入力します。

```
$ openstack loadbalancer provider list
```

出力例

```
+---------+-------------------------------------------------+
| name    | description                                     |
+---------+-------------------------------------------------+
| amphora | The Octavia Amphora driver.                     |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                             |
+---------+-------------------------------------------------+
```

RHOSP バージョン 16 以降、Octavia OVN プロバイダードライバー (**ovn**) は RHOSP デプロイメントの OpenShift Container Platform でサポートされます。

ovn は、Octavia および OVN が提供する負荷分散用の統合ドライバーです。これは基本的な負荷分散機能をサポートし、OpenFlow ルールに基づいています。このドライバーは、OVN Neutron ML2 を使用するデプロイメント上の director により Octavia で自動的に有効にされます。

Amphora プロバイダードライバーがデフォルトのドライバーです。ただし、**ovn** が有効にされる場合には、Kuryr がこれを使用します。

Kuryr が Amphora の代わりに **ovn** を使用する場合は、以下の利点があります。

- リソース要件が減少します。Kuryr は、各サービスにロードバランサーの仮想マシンを必要としません。
- ネットワークレイテンシーが短縮されます。
- サービスごとに仮想マシンを使用する代わりに、OpenFlow ルールを使用することで、サービスの作成速度が上がります。
- Amphora 仮想マシンで集中管理されるのではなく、すべてのノードに分散負荷分散アクションが分散されます。

RHOSP クラウドがバージョン 13 から 16 にアップグレードした後に、クラスターを Octavia OVN ドライバーを使用するように設定 できます。

8.2.3.4. Kuryr を使用したインストールについての既知の制限

OpenShift Container Platform を Kuryr SDN で使用する場合、いくつかの既知の制限があります。

RHOSP の一般的な制限
Kuryr SDN を使用する OpenShift Container Platform は、タイプが **NodePort** の **Service** オブジェクトをサポートしません。

RHOSP バージョンの制限
OpenShift Container Platform を Kuryr SDN で使用する場合は、RHOSP バージョンに依存するいくつかの制限があります。

- RHOSP の 16 よりも前のバージョンでは、デフォルトの Octavia ロードバランサードライバー (Amphora) を使用します。このドライバーでは、OpenShift Container Platform サービスごとに 1 つの Amphora ロードバランサー仮想マシンをデプロイする必要があります。サービス数が

多すぎると、リソースが不足する可能性があります。
OVN Octavia ドライバーが無効にされている以降のバージョンの RHOSP のデプロイメントでも Amphora ドライバーを使用します。この場合も、RHOSP の以前のバージョンと同じリソースに関する懸念事項を考慮する必要があります。

- バージョン 13.0.13 よりも前の Octavia RHOSP バージョンは UDP リスナーをサポートしません。そのため、OpenShift Container Platform UDP サービスはサポートされません。
- 13.0.13 よりも前の Octavia RHOSP バージョンは、同じポートで複数のプロトコルをリッスンできません。TCP や UDP など、同じポートを異なるプロトコルに公開するサービスはサポートされません。

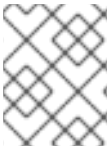
RHOSP 環境の制限
Kuryr SDN を使用する場合に、デプロイメント環境に依存する制限事項があります。

Octavia には UDP プロトコルおよび複数のリスナーのサポートがないため、RHOCP バージョンが 13.0.13 よりも前のバージョンの場合、Kuryr は Pod が DNS 解決に TCP を使用するように強制します。

Go バージョン 1.12 以前では、CGO サポートが無効にされた状態でコンパイルされたアプリケーションは UDP のみを使用します。この場合、ネイティブの Go リゾルバーは、TCP が DNS 解決に強制的に実行されるかどうかを制御する、**resolv.conf** の **use-vc** オプションを認識しません。その結果、UDP は引き続き DNS 解決に使用されますが、これは失敗します。

TCP の強制を許可するには、環境変数 **CGO_ENABLED** を **1** に設定 (例: **CGO_ENABLED=1**) されている状態でアプリケーションをコンパイルするか、または変数がないことを確認します。

Go バージョン 1.13 以降では、UDP を使用した DNS 解決が失敗する場合に TCP が自動的に使用されます。



注記

Alpine ベースのコンテナーを含む musl ベースのコンテナーは **use-vc** オプションをサポートしません。

RHOSP のアップグレードの制限
RHOSP のアップグレードプロセスにより、Octavia API が変更され、ロードバランサーに使用される Amphora イメージへのアップグレードが必要になる可能性があります。

API の変更に個別に対応できます。

Amphora イメージがアップグレードされると、RHOSP Operator は既存のロードバランサー仮想マシンを 2 つの方法で処理できます。

- ロードバランサーのフェイルオーバー をトリガーしてそれぞれの仮想マシンをアップグレードします。
- ユーザーが仮想マシンのアップグレードを行う必要があります。

Operator が最初のオプションを選択する場合、フェイルオーバー時に短い時間のダウンタイムが生じる可能性があります。

Operator が 2 つ目のオプションを選択する場合、既存のロードバランサーは UDP リスナーなどのアップグレードされた Octavia API 機能をサポートしません。この場合、ユーザーはこれらの機能を使用するためにサービスを再作成する必要があります。

重要

OpenShift Container Platform が UDP の負荷分散をサポートする新規の Octavia バージョンを検出する場合、これは DNS サービスを自動的に再作成します。サービスの再作成により、サービスのデフォルトが UDP の負荷分散をサポートするようになります。

再作成により、DNS サービスに約 1 分間のダウンタイムが発生します。

8.2.3.5. コントロールプレーンおよびコンピュートマシン

デフォルトで、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンおよび 3 つのコンピュートマシンを使用します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

8.2.3.6. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

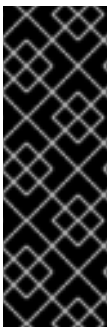
8.2.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

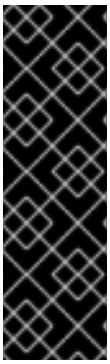


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.2.5. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。



重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

8.2.6. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

重要

外部ネットワークの CIDR 範囲がデフォルトのネットワーク範囲のいずれかと重複している場合、インストールプロセスを開始する前に、**install-config.yaml** ファイルで一致するネットワーク範囲を変更する必要があります。

デフォルトのネットワーク範囲は以下のとおりです。

ネットワーク	範囲
machineNetwork	10.0.0.0/16
serviceNetwork	172.30.0.0/16
clusterNetwork	10.128.0.0/14

警告

インストールプログラムにより同じ名前を持つ複数のネットワークが見つかる場合、それらのネットワークのいずれかがランダムに設定されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

8.2.7. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. マシンを認証局の信頼バンドルに追加します。

   ```
   $ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
   ```

   c. 信頼バンドルを更新します。

   ```
   $ sudo update-ca-trust extract
   ```

   d. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

   ```
   clouds:
     shiftstack:
       ...
       cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
   ```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

   ```
   $ oc edit configmap -n openshift-config cloud-provider-config
   ```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

   b. 現行ディレクトリー

   c. Unix 固有のユーザー設定ディレクトリー (例: ~/**.config**/**openstack**/**clouds.yaml**)

   d. Unix 固有のサイト設定ディレクトリー (例: /**etc**/**openstack**/**clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

8.2.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

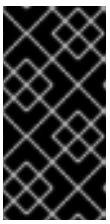
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。

2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.2.9. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat OpenStack Platform (RHOSP)

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

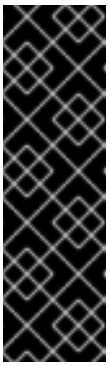
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

viii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

ix. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

x. コントロールプレーンおよびコンピュートノードに使用する 16 GB 以上の RAM で RHOSP フレーバーを指定します。

xi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

xii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

xiii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

8.2.9.1. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。

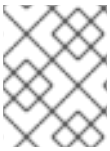
4. 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。



注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

8.2.10. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

8.2.10.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.9 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.2.10.2. ネットワーク設定パラメーター

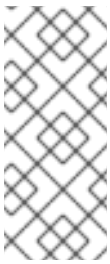
既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表8.10 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

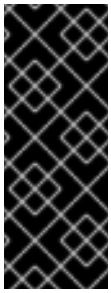
パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

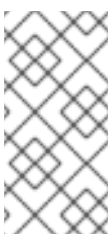
8.2.10.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.11 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

8.2.10.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表8.12 追加の RHOSP パラメーター

パラメーター	説明	値

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。	文字列 (例: **m1.xlarge**)。
platform.openstack.lbFloatingIP	ロードバランサー API に関連付ける既存の Floating IP アドレス。	IP アドレス (例: **128.0.0.1**)。

8.2.10.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表8.13 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。 この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**

パラメーター	説明	値
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

8.2.10.6. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表8.14 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

8.2.10.7. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。ノードとポートはこの上に作成されます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、以下を確認します。

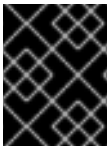
- ターゲットネットワークおよびサブネットが利用可能である。
- DHCP がターゲットサブネットで有効にされている。
- ターゲットネットワーク上でポートを作成するためのパーミッションがあるインストーラー認証情報を指定できます。
- ネットワーク設定にルーターが必要な場合、これは RHOSP で作成されます。一部の設定は、Floating IP アドレスの変換用のルーターに依存します。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。

注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

8.2.10.8. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル

デフォルトの OpenShift SDN ではなく Kuryr SDN を使用してデプロイするには、**install-config.yaml** ファイルを変更して **Kuryr** を必要な **networking.networkType** として追加してから、デフォルトの OpenShift Container Platform SDN インストール手順に進む必要があります。このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
```

```
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 (1)
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: true (2)
    octaviaSupport: true (3)
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

(1) Amphora Octavia ドライバーは、ロードバランサーごとに 2 つのポートを作成します。そのため、インストーラーが作成するサービスサブネットは、**serviceNetwork** プロパティーの値として指定される CIDR のサイズは 2 倍になります。IP アドレスの競合を防ぐには、範囲をより広くする必要があります。

(2) (3) **trunkSupport** と **octaviaSupport** の両方はインストーラーによって自動的に検出されるため、それらを設定する必要はありません。ただし、ご使用の環境がこれらの両方の要件を満たさないと、Kuryr SDN は適切に機能しません。トランクは Pod を RHOSP ネットワークに接続するために必要であり、Octavia は OpenShift Container Platform サービスを作成するために必要です。

8.2.11. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

   -

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

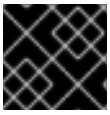
8.2.12. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

OpenShift Container Platform API およびクラスターで実行されるアプリケーションを、floating IP アドレスを使用/不使用でアクセス可能になるように設定できます。

8.2.12.1. floating IP アドレスを使ったアクセスの有効化

2 つの floating IP (FIP) アドレスを作成します。1 つ目は OpenShift Container Platform API への外部アクセス用の **API FIP** であり、2 つ目は OpenShift Container Platform アプリケーション用の **apps FIP** です。



重要

API FIP も **install-config.yaml** ファイルで使用されます。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

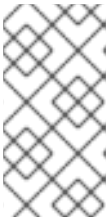
```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. 新しい FIP を反映させるには、以下のパターンに続くレコードを DNS サーバーに追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



注記

DNS サーバーを制御しない場合は、代わりに **/etc/hosts** ファイルにレコードを追加します。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

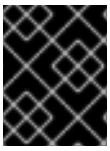
8.2.12.2. Floating IP アドレスを使用しないアクセスの有効化

Floating IP アドレスを使用できない場合でも、OpenShift Container Platform のインストールは終了できる可能性があります。ただし、インストールプラグラムは API アクセスを待機してタイムアウトする場合は失敗します。

インストールプログラムがタイムアウトすると、クラスターは初期化される可能性があります。ブートストラップ処理が開始されたら、これを完了する必要があります。デプロイ後にクラスターのネットワーク設定を編集する必要があります。

8.2.13. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

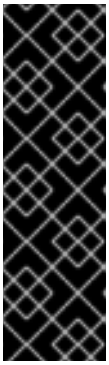
```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

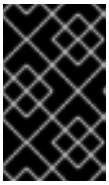
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

8.2.14. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

8.2.15. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.2.16. Floating IP アドレスを使用したアプリケーションアクセスの設定

OpenShift Container Platform をインストールした後に、アプリケーションネットワークトラフィックを許可するように Red Hat OpenStack Platform (RHOSP) を設定します。

前提条件

- OpenShift Container Platform クラスターがインストールされている必要があります。
- **環境へのアクセスの有効化**で説明されているように、Floating IP アドレスが有効化されています。

手順

OpenShift Container Platform クラスターをインストールした後に、Floating IP アドレスを Ingress ポートに割り当てます。

1. ポートを表示します。

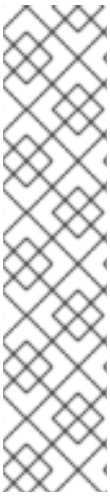
```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. ポートを IP アドレスに接続します。

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. ***apps.** のワイルドカード **A** レコードを DNS ファイルに追加します。

```
*.apps.<cluster name>.<base domain>  IN  A  <apps FIP>
```



注記

DNS サーバーを制御せず、非実稼働環境でアプリケーションアクセスを有効にする必要がある場合は、これらのホスト名を **/etc/hosts** に追加できます。

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.2.17. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラストラフィックを設定 します。

8.3. 独自のインフラストラクチャーを使用した OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、ユーザーによってプロビジョニングされたインフラストラクチャーを実行するクラスターを Red Hat OpenStack Platform (RHOSP) にインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

8.3.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。

    - OpenShift Container Platform 4.5 が **Available platforms** セクションの RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。

- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。
- OpenShift Container Platform をインストールする RHOSP アカウントがあります。
- インストールプログラムを実行するマシンには、以下が含まれます。
    - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー
    - Python 3

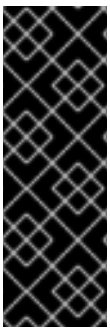
8.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

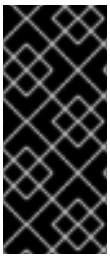
8.3.3. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表8.15 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

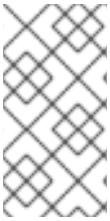
リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。



注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

8.3.3.1. コントロールプレーンおよびコンピュートマシン

デフォルトで、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンおよび 3 つのコンピュートマシンを使用します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

8.3.3.2. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

8.3.4. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされています。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

8.3.5. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

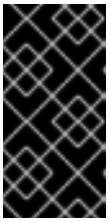
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.3.6. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/**.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.3.7. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

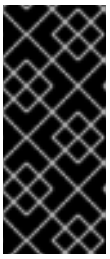
OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.5 の最新リリースを選択します。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。
4. イメージを展開します。

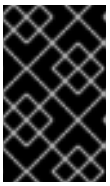
注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```



重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

8.3.8. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

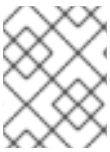
1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+----------------+-------------+
| ID                                   | Name           | Router Type |
+--------------------------------------+----------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+----------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。



注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

8.3.9. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

OpenShift Container Platform API およびクラスターで実行されるアプリケーションを、floating IP アドレスを使用してアクセス可能になるように設定できます。

8.3.9.1. floating IP アドレスを使ったアクセスの有効化

2 つの floating IP (FIP) アドレスを作成します。1 つ目は OpenShift Container Platform API への外部アクセス用の **API FIP** であり、2 つ目は OpenShift Container Platform アプリケーション用の **apps FIP** です。

重要

API FIP も **install-config.yaml** ファイルで使用されます。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. 新しい FIP を反映させるには、以下のパターンに続くレコードを DNS サーバーに追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御しない場合は、代わりに **/etc/hosts** ファイルにレコードを追加します。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

ヒント

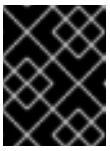
Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

8.3.10. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。



重要

パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

- RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. マシンを認証局の信頼バンドルに追加します。

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

   c. 信頼バンドルを更新します。

```
$ sudo update-ca-trust extract
```

   d. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

```
clouds:
  shiftstack:
    ...
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

   ヒント

   カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。

   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値

b. 現行ディレクトリー

c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)

d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
インストールプログラムはこの順序で **clouds.yaml** を検索します。

8.3.11. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat OpenStack Platform (RHOSP)

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

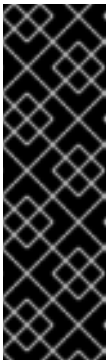
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

viii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

ix. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

x. コントロールプレーンおよびコンピュートノードに使用する 16 GB 以上の RAM で RHOSP フレーバーを指定します。

xi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

xii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

xiii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

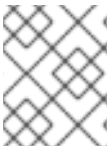
重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

8.3.12. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

8.3.12.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.16 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。

パラメーター	説明	値
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.3.12.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表8.17 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。

パラメーター	説明	値
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork	Pod の IP アドレスブロック。 デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: clusterNetwork: - cidr: 10.128.0.0/14 hostPrefix: 23
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16

パラメーター	説明	値
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16**  **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

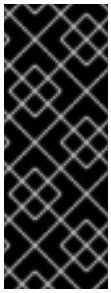
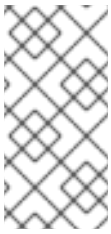
8.3.12.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.18 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**

パラメーター	説明	値
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。 **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

8.3.12.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表8.19 追加の RHOSP パラメーター

パラメーター	説明	値

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。	文字列 (例: **m1.xlarge**)。
platform.openstack.lbFloatingIP	ロードバランサー API に関連付ける既存の Floating IP アドレス。	IP アドレス (例: **128.0.0.1**)。

8.3.12.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表8.20 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。 この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**

パラメーター	説明	値
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

8.3.12.6. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表8.21 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

8.3.12.7. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。ノードとポートはこの上に作成されます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、以下を確認します。

- ターゲットネットワークおよびサブネットが利用可能である。
- DHCP がターゲットサブネットで有効にされている。
- ターゲットネットワーク上でポートを作成するためのパーミッションがあるインストーラー認証情報を指定できます。
- ネットワーク設定にルーターが必要な場合、これは RHOSP で作成されます。一部の設定は、Floating IP アドレスの変換用のルーターに依存します。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。

注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

8.3.12.8. RHOSP のカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
```

```
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

8.3.12.9. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; (1)
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

(1) 必要な Neutron サブネットに一致する値 (例: **192.0.2.0/24**) を挿入します。

   - 値を手動で設定するには、ファイルを開き、**networking.machineCIDR** の値を必要な Neutron サブネットに一致する値に設定します。

8.3.12.10. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

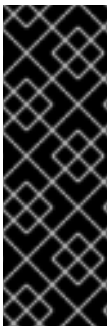
1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

8.3.13. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-
cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
```

```
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

8. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

8.3.14. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
        'verification': {}
    },
    'filesystem': 'root',
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
            'verification': {}
```

```
        },
        'filesystem': 'root',
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

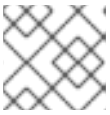
2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。



注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", 1
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
```

```
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", 4
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

8.3.15. コントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。

注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成**を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
files = ignition['storage'].get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
ignition['storage']['files'] = files;
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

8.3.16. ネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セキュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

手順

1. **common.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.1 common.yaml Ansible Playbook

```
- hosts: localhost
  gather_facts: no

  vars_files:
  - metadata.json

  tasks:
  - name: 'Compute resource names'
    set_fact:
      cluster_id_tag: "openshiftClusterID={{ infraID }}"
      os_network: "{{ infraID }}-network"
      os_subnet: "{{ infraID }}-nodes"
      os_router: "{{ infraID }}-external-router"
      # Port names
      os_port_api: "{{ infraID }}-api-port"
      os_port_ingress: "{{ infraID }}-ingress-port"
```

```
  os_port_bootstrap: "{{ infraID }}-bootstrap-port"
  os_port_master: "{{ infraID }}-master-port"
  os_port_worker: "{{ infraID }}-worker-port"
  # Security groups names
  os_sg_master: "{{ infraID }}-master"
  os_sg_worker: "{{ infraID }}-worker"
  # Server names
  os_bootstrap_server_name: "{{ infraID }}-bootstrap"
  os_cp_server_name: "{{ infraID }}-master"
  os_cp_server_group_name: "{{ infraID }}-master"
  os_compute_server_name: "{{ infraID }}-worker"
  # Trunk names
  os_cp_trunk_name: "{{ infraID }}-master-trunk"
  os_compute_trunk_name: "{{ infraID }}-worker-trunk"
  # Subnet pool name
  subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
  # Service network name
  os_svc_network: "{{ infraID }}-kuryr-service-network"
  # Service subnet name
  os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
  # Ignition files
  os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. **inventory.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.2 inventory.yaml Ansible Playbook

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'm1.xlarge'
      os_flavor_worker: 'm1.large'
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
      # OpenShift API floating IP address
      os_api_fip: '203.0.113.23'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '203.0.113.19'
      # Service subnet cidr
      svc_subnet_range: '172.30.0.0/16'
      os_svc_network_range: '172.30.0.0/15'
      # Subnet pool prefixes
      cluster_network_cidrs: '10.128.0.0/14'
      # Subnet pool prefix length
      host_prefix: '23'
      # Name of the SDN.
      # Possible values are OpenshiftSDN or Kuryr.
      os_networking_type: 'OpenshiftSDN'

      # Number of provisioned Control Plane nodes
```

```
    # 3 is the minimum number for a fully-functional cluster.
    os_cp_nodes_number: 3

    # Number of provisioned Compute nodes.
    # 3 is the minimum number for a fully-functional cluster.
    os_compute_nodes_number: 3
```

3. **security-groups.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.3 security-groups.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the master security group'
    os_security_group:
      name: "{{ os_sg_master }}"

  - name: 'Set master security group tag'
    command:
      cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }} "

  - name: 'Create the worker security group'
    os_security_group:
      name: "{{ os_sg_worker }}"

  - name: 'Set worker security group tag'
    command:
      cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }} "

  - name: 'Create master-sg rule "ICMP"'
    os_security_group_rule:
      security_group: "{{ os_sg_master }}"
      protocol: icmp

  - name: 'Create master-sg rule "machine config server"'
    os_security_group_rule:
      security_group: "{{ os_sg_master }}"
      protocol: tcp
      remote_ip_prefix: "{{ os_subnet_range }}"
      port_range_min: 22623
      port_range_max: 22623

  - name: 'Create master-sg rule "SSH"'
    os_security_group_rule:
      security_group: "{{ os_sg_master }}"
```

```
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create master-sg rule "DNS (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: tcp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "DNS (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 6443
    port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
```

```
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 2379
    port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
```

```
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"


- name: 'Create worker-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: icmp

- name: 'Create worker-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create worker-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 80
    port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 443
    port_range_max: 443

- name: 'Create worker-sg rule "router"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
```

```
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 1936
    port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet insecure"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
```

```
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"
```

4. **network.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.4 network.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the cluster network'
    os_network:
      name: "{{ os_network }}"

  - name: 'Set the cluster network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

  - name: 'Create a subnet'
    os_subnet:
      name: "{{ os_subnet }}"
      network_name: "{{ os_network }}"
      cidr: "{{ os_subnet_range }}"
      allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
      allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

  - name: 'Set the cluster subnet tag'
    command:
      cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

  - name: 'Create the service network'
    os_network:
      name: "{{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

  - name: 'Set the service network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
```

```
    when: os_networking_type == "Kuryr"

  - name: 'Computing facts for service subnet'
    set_fact:
      first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
      last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') |ipmath(1) }}"
      first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
      last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable')
|ipmath(1) }}"
      allocation_pool: ""
    when: os_networking_type == "Kuryr"

  - name: 'Get first part of OpenStack network'
    set_fact:
      allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range |ipmath(-
1) }}' }}"
    when:
    - os_networking_type == "Kuryr"
    - first_ip_svc_subnet_range != first_ip_os_svc_network_range

  - name: 'Get last part of OpenStack network'
    set_fact:
      allocation_pool: "{{ allocation_pool + ' --allocation-pool start={{
last_ip_svc_subnet_range | ipmath(1) }},end={{ last_ip_os_svc_network_range |ipmath(-
1) }}' }}"
    when:
    - os_networking_type == "Kuryr"
    - last_ip_svc_subnet_range != last_ip_os_svc_network_range

  - name: 'Get end of allocation'
    set_fact:
      gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
    when: os_networking_type == "Kuryr"

  - name: 'replace last IP'
    set_fact:
      allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1))}}"
    when: os_networking_type == "Kuryr"

  - name: 'list service subnet'
    command:
      cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
    when: os_networking_type == "Kuryr"
    register: svc_subnet

  - name: 'Create the service subnet'
    command:
      cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-
range {{ os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{
os_svc_network }} --tag {{ cluster_id_tag }} {{ os_svc_subnet }}"
    when:
    - os_networking_type == "Kuryr"
    - svc_subnet.stdout == ""

  - name: 'list subnet pool'
```

```
    command:
      cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
    when: os_networking_type == "Kuryr"
    register: pods_subnet_pool

  - name: 'Create pods subnet pool'
    command:
      cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-
prefix {{ cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
    when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout == ""

  - name: 'Create external router'
    os_router:
      name: "{{ os_router }}"
      network: "{{ os_external_network }}"
      interfaces:
      - "{{ os_subnet }}"

  - name: 'Set external router tag'
    command:
      cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
    when: os_networking_type == "Kuryr"

  - name: 'Create the API port'
    os_port:
      name: "{{ os_port_api }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

  - name: 'Set API port tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

  - name: 'Create the Ingress port'
    os_port:
      name: "{{ os_port_ingress }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_worker }}"
      fixed_ips:
      - subnet: "{{ os_subnet }}"
        ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

  - name: 'Set the Ingress port tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

  # NOTE: openstack ansible module doesn't allow attaching Floating IPs to
  # ports, let's use the CLI instead
  - name: 'Attach the API floating IP to API port'
```

```
    command:
      cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

  # NOTE: openstack ansible module doesn't allow attaching Floating IPs to
  # ports, let's use the CLI instead
  - name: 'Attach the Ingress floating IP to Ingress port'
    command:
      cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"
```

5. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

6. コマンドラインで、**network.yaml** Playbook を実行して、ネットワーク、サブネット、およびルーターを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```

7. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

8.3.17. ブートストラップマシンの作成

ブートストラップマシンを作成し、Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. **bootstrap.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.5 bootstrap.yaml

```
# Required Python packages:
#
# ansible
```

```
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"

  - name: 'Set bootstrap port tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

  - name: 'Create the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_master }}"
      userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
      auto_ip: no
      nics:
      - port-name: "{{ os_port_bootstrap }}"

  - name: 'Create the bootstrap floating IP'
    os_floating_ip:
      state: present
      network: "{{ os_external_network }}"
      server: "{{ os_bootstrap_server_name }}"
```

3. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

4. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

8.3.18. コントロールプレーンマシンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。

前提条件

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- **コントロールプレーン Ignition 設定ファイルの作成**で作成された 3 つの Ignition ファイル。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. コントロールプレーン Ignition 設定ファイルが作業ディレクトリーにない場合、それらをここにコピーします。
3. **control-plane.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.6 control-plane.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the Control Plane ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
    register: ports

  - name: 'Set Control Plane ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

  - name: 'List the Control Plane Trunks'
    command:
      cmd: "openstack network trunk list"
```

```
    when: os_networking_type == "Kuryr"
    register: control_plane_trunks

  - name: 'Create the Control Plane trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_cp_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

  - name: 'List the Server groups'
    command:
      cmd: "openstack server group list -f json -c ID -c Name"
    register: server_group_list

  - name: 'Parse the Server group ID from existing'
    set_fact:
      server_group_id: "{{ (server_group_list.stdout | from_json | json_query(list_query) |
first).ID }}"
    vars:
      list_query: "[?Name=='{{ os_cp_server_group_name }}']"
    when:
    - "os_cp_server_group_name|string in server_group_list.stdout"

  - name: 'Create the Control Plane server group'
    command:
      cmd: "openstack --os-compute-api-version=2.15 server group create -f json -c id --
policy=soft-anti-affinity {{ os_cp_server_group_name }}"
    register: server_group_created
    when:
    - server_group_id is not defined

  - name: 'Parse the Server group ID from creation'
    set_fact:
      server_group_id: "{{ (server_group_created.stdout | from_json).id }}"
    when:
    - server_group_id is not defined

  - name: 'Create the Control Plane servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_master }}"
      auto_ip: no
      # The ignition filename will be concatenated with the Control Plane node
      # name and its 0-indexed serial number.
      # In this case, the first node will look for this filename:
      #    "{{ infraID }}-master-0-ignition.json"
      userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"
      nics:
      - port-name: "{{ os_port_master }}-{{ item.0 }}"
      scheduler_hints:
        group: "{{ server_group_id }}"
    with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"
```

```
```

4. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

5. 以下のコマンドを実行してブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete
```

コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

```
INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

8.3.19. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.3.20. ブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- コントロールプレーンマシンを実行中です。
  - マシンのステータスが分からない場合は、**クラスターステータスの確認**を参照してください。

手順

1. **down-bootstrap.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.7 down-bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      state: absent
      delete_fip: yes

  - name: 'Remove the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      state: absent
```

2. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL を無効にしていない場合は、無効にしてください。

8.3.21. コンピュートマシンの作成

コントロールプレーンの起動後、コンピュートマシンを作成します。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。
- コントロールプレーンがアクティブです。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. **compute-nodes.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.8 compute-nodes.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the Compute ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_worker }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
    register: ports

  - name: 'Set Compute ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

  - name: 'List the Compute Trunks'
    command:
      cmd: "openstack network trunk list"
```

```
    when: os_networking_type == "Kuryr"
    register: compute_trunks

  - name: 'Create the Compute trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

  - name: 'Create the Compute servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_worker }}"
      auto_ip: no
      userdata: "{{ lookup('file', 'worker.ign') | string }}"
      nics:
      - port-name: "{{ os_port_worker }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"
```

3. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

次のステップ

- マシンの証明書署名要求を承認します。

8.3.22. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
```

```
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

8.3.23. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

8.3.24. Floating IP アドレスを使用したアプリケーションアクセスの設定

OpenShift Container Platform をインストールした後に、アプリケーションネットワークトラフィックを許可するように Red Hat OpenStack Platform (RHOSP) を設定します。

前提条件

- OpenShift Container Platform クラスターがインストールされている必要があります。
- **環境へのアクセスの有効化**で説明されているように、Floating IP アドレスが有効化されています。

手順

OpenShift Container Platform クラスターをインストールした後に、Floating IP アドレスを Ingress ポートに割り当てます。

1. ポートを表示します。

```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. ポートを IP アドレスに接続します。

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. ***apps.** のワイルドカード **A** レコードを DNS ファイルに追加します。

```
*.apps.<cluster name>.<base domain>  IN  A  <apps FIP>
```

注記

DNS サーバーを制御せず、非実稼働環境でアプリケーションアクセスを有効にする必要がある場合は、これらのホスト名を **/etc/hosts** に追加できます。

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.3.25. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラスタートラフィックを設定 します。

8.4. 独自のインフラストラクチャーでの KURYR を使用する OPENSTACK へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、ユーザーによってプロビジョニングされたインフラストラクチャーを実行するクラスターを Red Hat OpenStack Platform (RHOSP) にインストールできます。

独自のインフラストラクチャーを使用することで、クラスターを既存のインフラストラクチャーおよび変更と統合できます。このプロセスでは、インストーラーでプロビジョニングされるインストールの場合よりも多くの手作業が必要になります。Nova サーバー、Neutron ポート、セキュリティーグループなどのすべての RHOSP リソースを作成する必要があるためです。ただし、Red Hat では、デプロイメントプロセスを支援する Ansible Playbook を提供しています。

8.4.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
  - OpenShift Container Platform 4.5 が **Available platforms** セクションの RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。
- OpenShift Container Platform をインストールする RHOSP アカウントがあります。
- インストールプログラムを実行するマシンには、以下が含まれます。
  - インストールプロセス時に作成したファイルを保持できる単一ディレクトリー

- Python 3

8.4.2. Kuryr SDN について

Kuryr は、Neutron および Octavia Red Hat OpenStack Platform (RHOSP) サービスを使用して Pod およびサービスのネットワークを提供する Container Network Interface (CNI) プラグインです。

Kuryr と OpenShift Container Platform の統合は主に、RHOSP の仮想マシンで実行する OpenShift Container Platform クラスター用に設計されました。Kuryr は、OpenShift Container Platform Pod を RHOSP SDN にプラグインしてネットワークのパフォーマンスを強化します。さらに、これは Pod と RHOSP 仮想インスタンス間の接続を可能にします。

Kuryr コンポーネントは **openshift-kuryr** namespace を使用して OpenShift Container Platform の Pod としてインストールされます。

- **kuryr-controller**: **master** ノードにインストールされる単一のサービスインスタンスです。これは、OpenShift Container Platform で **Deployment** としてモデリングされます。
- **kuryr-cni**: 各 OpenShift Container Platform ノードで Kuryr を CNI ドライバーとしてインストールし、設定するコンテナーです。これは、OpenShift Container Platform で **DaemonSet** オブジェクトとしてモデリングされます。

Kuryr コントローラーは OpenShift Container Platform API サーバーで Pod、サービスおよび namespace の作成、更新、および削除イベントについて監視します。これは、OpenShift Container Platform API 呼び出しを Neutron および Octavia の対応するオブジェクトにマップします。そのため、Neutron トランクポート機能を実装するすべてのネットワークソリューションを使用して、Kuryr 経由で OpenShift Container Platform をサポートすることができます。これには、Open vSwitch (OVS) および Open Virtual Network (OVN) などのオープンソースソリューションや Neutron と互換性のある市販の SDN が含まれます。

Kuryr は、カプセル化された RHOSP テナントネットワーク上の OpenShift Container Platform デプロイメントに使用することが推奨されています。これは、 RHOSP ネットワークでカプセル化された OpenShift Container Platform SDN を実行するなど、二重のカプセル化を防ぐために必要です。

プロバイダーネットワークまたはテナント VLAN を使用する場合は、二重のカプセル化を防ぐために Kuryr を使用する必要はありません。パフォーマンス上の利点はそれほど多くありません。ただし、設定によっては、Kuryr を使用して 2 つのオーバーレイが使用されないようにすることには利点がある場合があります。

Kuryr は、以下のすべての基準が true であるデプロイメントでは推奨されません。

- RHOSP のバージョンが 16 よりも前のバージョンである。
- デプロイメントで UDP サービスが使用されているか、または少数のハイパーバイザーで多数の TCP サービスが使用されている。

または、以下を実行します。

- **ovn-octavia** Octavia ドライバーが無効にされている。
- デプロイメントで、少数のハイパーバイザーで多数の TCP サービスが使用されている。

8.4.3. Kuryr を使用して OpenShift Container Platform を RHOSP にインストールするためのリソースのガイドライン

Kuryr SDN を使用する場合、Pod、サービス、namespace およびネットワークポリシーは RHOSP クォータのリソースを使用します。これにより、最小要件が増加します。また、Kuryr にはデフォルトインストールに必要な要件以外の追加要件があります。

以下のクォータを使用してデフォルトのクラスターの最小要件を満たすようにします。

表8.22 Kuryr を使用する RHOSP のデフォルト OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3: LoadBalancer タイプに予想されるサービス数
ポート	1500: Pod ごとに 1 つ必要
ルーター	1
サブネット	250: namespace/プロジェクトごとに 1 つ必要
ネットワーク	250: namespace/プロジェクトごとに 1 つ必要
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	250: サービスおよび NetworkPolicy ごとに 1 つ必要
セキュリティーグループルール	1000
ロードバランサー	100: サービスごとに 1 つ必要
ロードバランサーリスナー	500: サービスで公開されるポートごとに 1 つ必要
ロードバランサーノード	500: サービスで公開されるポートごとに 1 つ必要

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。

重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

重要

OVN Octavia ドライバーではなく Amphora ドライバーで Red Hat OpenStack Platform(RHOSP) バージョン 16 を使用している場合、セキュリティーグループはユーザープロジェクトではなくサービスアカウントに関連付けられます。

リソースを設定する際には、以下の点に注意してください。

- 必要なポート数は Pod 数よりも大きくなる。Kuryr はポートプールを使用して、事前に作成済みのポートを Pod で使用できるようにし、Pod の起動時間を短縮します。
- 各ネットワークポリシーは RHOSP セキュリティーグループにマップされ、**NetworkPolicy** 仕様によっては 1 つ以上のルールがセキュリティーグループに追加される。
- 各サービスは RHOSP ロードバランサーにマップされる。クォータに必要なセキュリティーグループの数を見積もる場合には、この要件を考慮してください。
  RHOSP バージョン 15 以前のバージョン、または **ovn-octavia driver** を使用している場合、各ロードバランサーにはユーザープロジェクトと共にセキュリティーグループがあります。
- クォータはロードバランサーのリソース (VM リソースなど) を考慮しませんが、RHOSP デプロイメントのサイズを決定する際にはこれらのリソースを考慮する必要があります。デフォルトのインストールには 50 を超えるロードバランサーが あり、クラスターはそれらのロードバランサーに対応できる必要があります。
  OVN Octavia ドライバーを有効にして RHOSP バージョン 16 を使用している場合は、1 つのロードバランサー仮想マシンのみが生成され、サービスは OVN フロー経由で負荷分散されます。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

Kuryr SDN を有効にするには、使用する環境が以下の要件を満たしている必要があります。

- RHOSP 13+ を実行します。
- オーバークラウドと Octavia を使用します。
- Neutron トランクポートの拡張を使用します。
- ML2/OVS Neutron ドライバーが **ovs-hybrid** の代わりに使用れる場合、**openvswitch** ファイアウォールドライバーを使用します。

8.4.3.1. クォータの拡大

Kuryr SDN を使用する場合、Pod、サービス、namespace、およびネットワークポリシーが使用する Red Hat OpenStack Platform (RHOSP) リソースに対応するためにクォータを引き上げる必要があります。

手順

- 以下のコマンドを実行して、プロジェクトのクォータを増やします。

```
$ sudo openstack quota set --secgroups 250 --secgroup-rules 1000 --ports 1500 --subnets
250 --networks 250 <project>
```

8.4.3.2. Neutron の設定

Kuryr CNI は Neutron トランクの拡張を使用してコンテナーを Red Hat OpenStack Platform (RHOSP) SDN にプラグインします。したがって、Kuryr が適切に機能するには **trunks** 拡張を使用する必要があります。

さらにデフォルトの ML2/OVS Neutron ドライバーを使用する場合には、セキュリティーグループがトランクサブポートで実行され、Kuryr がネットワークポリシーを適切に処理できるように、**ovs_hybrid** ではなく **openvswitch** に設定される必要があります。

8.4.3.3. Octavia の設定

Kuryr SDN は Red Hat OpenStack Platform (RHOSP) の Octavia LBaaS を使用して OpenShift Container Platform サービスを実装します。したがって、Kuryr SDN を使用するように RHOSP に Octavia コンポーネントをインストールし、設定する必要があります。

Octavia を有効にするには、Octavia サービスを RHOSP オーバークラウドのインストール時に組み込むか、またはオーバークラウドがすでに存在する場合は Octavia サービスをアップグレードする必要があります。Octavia を有効にする以下の手順は、オーバークラウドのクリーンインストールまたはオーバークラウドの更新の両方に適用されます。

注記

以下の手順では、Octavia を使用する場合に RHOSP のデプロイメント 時に必要となる主な手順のみを説明します。また、レジストリーメソッド が変更されることにも留意してください。

以下の例では、ローカルレジストリーの方法を使用しています。

手順

1. ローカルレジストリーを使用している場合、イメージをレジストリーにアップロードするためのテンプレートを作成します。以下に例を示します。

```
(undercloud) $ openstack overcloud container image prepare \
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
--namespace=registry.access.redhat.com/rhosp13 \
--push-destination=<local-ip-from-undercloud.conf>:8787 \
--prefix=openstack- \
--tag-from-label {version}-{release} \
--output-env-file=/home/stack/templates/overcloud_images.yaml \
--output-images-file /home/stack/local_registry_images.yaml
```

2. **local_registry_images.yaml** ファイルに Octavia イメージが含まれることを確認します。以下に例を示します。

```
...
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-api:13.0-43
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-health-manager:13.0-
45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-housekeeping:13.0-45
  push_destination: <local-ip-from-undercloud.conf>:8787
- imagename: registry.access.redhat.com/rhosp13/openstack-octavia-worker:13.0-44
  push_destination: <local-ip-from-undercloud.conf>:8787
```

注記

Octavia コンテナーのバージョンは、インストールされている特定の RHOSP リリースによって異なります。

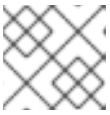
3. コンテナーイメージを **registry.redhat.io** からアンダークラウドノードにプルします。

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file  /home/stack/local_registry_images.yaml \
  --verbose
```

これには、ネットワークおよびアンダークラウドディスクの速度に応じて多少の時間がかかる可能性があります。

4. Octavia ロードバランサーは OpenShift Container Platform API にアクセスするために使用されるため、それらのリスナーの接続のデフォルトタイムアウトを増やす必要があります。デフォルトのタイムアウトは 50 秒です。以下のファイルをオーバークラウドのデプロイコマンドに渡し、タイムアウトを 20 分に増やします。

```
(undercloud) $ cat octavia_timeouts.yaml
parameter_defaults:
  OctaviaTimeoutClientData: 1200000
  OctaviaTimeoutMemberData: 1200000
```



注記

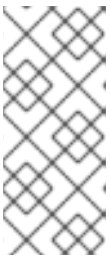
これは RHOSP 13.0.13+ では不要です。

5. Octavia を使用してオーバークラウドをインストールまたは更新します。

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml \
  -e octavia_timeouts.yaml
```

注記

このコマンドには、Octavia に関連付けられたファイルのみが含まれます。これは、RHOSP の特定のインストールによって異なります。詳細は RHOSP のドキュメントを参照してください。Octavia インストールのカスタマイズについての詳細は、Octavia デプロイメントのプランニング を参照してください。



注記

Kuryr SDN を利用する際には、オーバークラウドのインストールに Neutron の **trunk** 拡張機能が必要です。これは、Director デプロイメントでデフォルトで有効にされます。Neutron バックエンドが ML2/OVS の場合、デフォルトの **ovs-hybrid** の代わりに **openvswitch** ファイアウォールを使用します。バックエンドが ML2/OVN の場合には変更の必要がありません。

6. RHOSP の 13.0.13 よりも前のバージョンでは、プロジェクトの作成後にプロジェクト ID を **octavia.conf** 設定ファイルに追加します。

- トラフィックが Octavia ロードバランサーを通過する場合など、複数のサービス全体でネットワークポリシーを実行するには、Octavia がユーザープロジェクトで Amphora 仮想マシンセキュリティーグループを作成するようにする必要があります。
この変更により、必要なロードバランサーのセキュリティーグループがそのプロジェクトに属し、それらをサービスの分離を実行するように更新できます。

注記

RHOSP の 13.0.13 よりも後のバージョンでは、このタスクは必要ありません。

Octavia は、ロードバランサー VIP へのアクセスを制限する新しい ACL API を実装します。

a. プロジェクト ID を取得します。

```
$ openstack project show <project>
```

出力例

```
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description |                                  |
| domain_id   | default                          |
| enabled     | True                             |
| id          | PROJECT_ID                       |
| is_domain   | False                            |
| name        | *<project>*                      |
| parent_id   | default                          |
| tags        | []                               |
+-------------+----------------------------------+
```

b. プロジェクト ID をコントローラーの **octavia.conf** に追加します。

i. **stackrc** ファイルを取得します。

```
$ source stackrc  # Undercloud credentials
```

ii. オーバークラウドコントローラーを一覧表示します。

```
$ openstack server list
```

出力例

```
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
| ID                                   | Name        | Status | Networks
| Image         | Flavor     |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
|
```

```
| 6bef8e73-2ba5-4860-a0b1-3937f8ca7e01 | controller-0 | ACTIVE |
ctlplane=192.168.24.8 | overcloud-full | controller |
|
| dda3173a-ab26-47f8-a2dc-8473b4a67ab9 | compute-0    | ACTIVE |
ctlplane=192.168.24.6 | overcloud-full | compute    |
|
+--------------------------------------+-------------+--------+----------------------+-------
---------+------------+
```

iii. コントローラーに対して SSH を実行します。

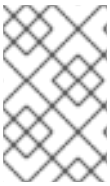
```
$ ssh heat-admin@192.168.24.8
```

iv. **octavia.conf** ファイルを編集して、プロジェクトを Amphora セキュリティーグループがユーザーのアカウントに設定されているプロジェクトの一覧に追加します。

```
# List of project IDs that are allowed to have Load balancer security groups
# belonging to them.
amp_secgroup_allowed_projects = PROJECT_ID
```

c. 新しい設定が読み込まれるように Octavia ワーカーを再起動します。

```
controller-0$ sudo docker restart octavia_worker
```



注記

RHOSP 環境によっては、Octavia は UDP リスナーをサポートしない可能性があります。RHOSP の 13.0.13 よりも前のバージョンで Kuryr SDN を使用する場合、UDP サービスはサポートされません。RHOSP バージョン 16 以降は UDP をサポートします。

8.4.3.3.1. Octavia OVN ドライバー

Octavia は Octavia API を使用して複数のプロバイダードライバーをサポートします。

利用可能なすべての Octavia プロバイダードライバーをコマンドラインで表示するには、以下を入力します。

```
$ openstack loadbalancer provider list
```

出力例

```
+---------+-------------------------------------------------+
| name    | description                                     |
+---------+-------------------------------------------------+
| amphora | The Octavia Amphora driver.                     |
| octavia | Deprecated alias of the Octavia Amphora driver. |
| ovn     | Octavia OVN driver.                             |
+---------+-------------------------------------------------+
```

RHOSP バージョン 16 以降、Octavia OVN プロバイダードライバー (**ovn**) は RHOSP デプロイメントの OpenShift Container Platform でサポートされます。

ovn は、Octavia および OVN が提供する負荷分散用の統合ドライバーです。これは基本的な負荷分散機能をサポートし、OpenFlow ルールに基づいています。このドライバーは、OVN Neutron ML2 を使用するデプロイメント上の director により Octavia で自動的に有効にされます。

Amphora プロバイダードライバーがデフォルトのドライバーです。ただし、**ovn** が有効にされる場合には、Kuryr がこれを使用します。

Kuryr が Amphora の代わりに **ovn** を使用する場合は、以下の利点があります。

- リソース要件が減少します。Kuryr は、各サービスにロードバランサーの仮想マシンを必要としません。
- ネットワークレイテンシーが短縮されます。
- サービスごとに仮想マシンを使用する代わりに、OpenFlow ルールを使用することで、サービスの作成速度が上がります。
- Amphora 仮想マシンで集中管理されるのではなく、すべてのノードに分散負荷分散アクションが分散されます。

8.4.3.4. Kuryr を使用したインストールについての既知の制限

OpenShift Container Platform を Kuryr SDN で使用する場合、いくつかの既知の制限があります。

RHOSP の一般的な制限
Kuryr SDN を使用する OpenShift Container Platform は、タイプが **NodePort** の **Service** オブジェクトをサポートしません。

RHOSP バージョンの制限
OpenShift Container Platform を Kuryr SDN で使用する場合は、RHOSP バージョンに依存するいくつかの制限があります。

- RHOSP の 16 よりも前のバージョンでは、デフォルトの Octavia ロードバランサードライバー (Amphora) を使用します。このドライバーでは、OpenShift Container Platform サービスごとに 1 つの Amphora ロードバランサー仮想マシンをデプロイする必要があります。サービス数が多すぎると、リソースが不足する可能性があります。
  OVN Octavia ドライバーが無効にされている以降のバージョンの RHOSP のデプロイメントでも Amphora ドライバーを使用します。この場合も、RHOSP の以前のバージョンと同じリソースに関する懸念事項を考慮する必要があります。
- バージョン 13.0.13 よりも前の Octavia RHOSP バージョンは UDP リスナーをサポートしません。そのため、OpenShift Container Platform UDP サービスはサポートされません。
- 13.0.13 よりも前の Octavia RHOSP バージョンは、同じポートで複数のプロトコルをリッスンできません。TCP や UDP など、同じポートを異なるプロトコルに公開するサービスはサポートされません。

RHOSP 環境の制限
Kuryr SDN を使用する場合に、デプロイメント環境に依存する制限事項があります。

Octavia には UDP プロトコルおよび複数のリスナーのサポートがないため、RHOCP バージョンが 13.0.13 よりも前のバージョンの場合、Kuryr は Pod が DNS 解決に TCP を使用するように強制します。

Go バージョン 1.12 以前では、CGO サポートが無効にされた状態でコンパイルされたアプリケーションは UDP のみを使用します。この場合、ネイティブの Go リゾルバーは、TCP が DNS 解決に強制的に実行されるかどうかを制御する、**resolv.conf** の **use-vc** オプションを認識しません。その結果、UDP

は引き続き DNS 解決に使用されますが、これは失敗します。

TCP の強制を許可するには、環境変数 **CGO_ENABLED** を **1** に設定 (例: **CGO_ENABLED=1**) されている状態でアプリケーションをコンパイルするか、または変数がないことを確認します。

Go バージョン 1.13 以降では、UDP を使用した DNS 解決が失敗する場合に TCP が自動的に使用されます。

注記

Alpine ベースのコンテナーを含む musl ベースのコンテナーは **use-vc** オプションをサポートしません。

RHOSP のアップグレードの制限
RHOSP のアップグレードプロセスにより、Octavia API が変更され、ロードバランサーに使用される Amphora イメージへのアップグレードが必要になる可能性があります。

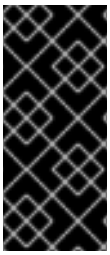
API の変更に個別に対応できます。

Amphora イメージがアップグレードされると、RHOSP Operator は既存のロードバランサー仮想マシンを 2 つの方法で処理できます。

- ロードバランサーのフェイルオーバー をトリガーしてそれぞれの仮想マシンをアップグレードします。
- ユーザーが仮想マシンのアップグレードを行う必要があります。

Operator が最初のオプションを選択する場合、フェイルオーバー時に短い時間のダウンタイムが生じる可能性があります。

Operator が 2 つ目のオプションを選択する場合、既存のロードバランサーは UDP リスナーなどのアップグレードされた Octavia API 機能をサポートしません。この場合、ユーザーはこれらの機能を使用するためにサービスを再作成する必要があります。



重要

OpenShift Container Platform が UDP の負荷分散をサポートする新規の Octavia バージョンを検出する場合、これは DNS サービスを自動的に再作成します。サービスの再作成により、サービスのデフォルトが UDP の負荷分散をサポートするようになります。

再作成により、DNS サービスに約 1 分間のダウンタイムが発生します。

8.4.3.5. コントロールプレーンおよびコンピュートマシン

デフォルトで、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンおよび 3 つのコンピュートマシンを使用します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

8.4.3.6. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

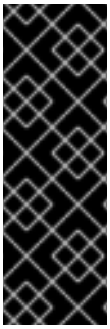
8.4.4. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

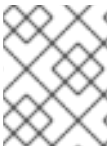


重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.4.5. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでのインストールプロセスを単純化する Ansible Playbook には、複数の Python モジュールが必要です。インストーラーを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。



注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされています。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk python3-netaddr
```

3. **python** コマンドが **python3** を参照していることを確認します。

```
$ sudo alternatives --set python /usr/bin/python3
```

8.4.6. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

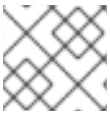
3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

8.4.7. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 **~/.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.4.8. Red Hat Enterprise Linux CoreOS (RHCOS) イメージの作成

OpenShift Container Platform インストールプログラムでは、Red Hat Enterprise Linux CoreOS (RHCOS) イメージが Red Hat OpenStack Platform (RHOSP) クラスターに存在する必要があります。最新の RHCOS イメージを取得した後、RHOSP CLI を使用してこれをアップロードします。

前提条件

- RHOSP CLI がインストールされています。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、Red Hat Enterprise Linux (RHEL) 8 用の OpenShift Container Platform 4.5 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**をダウンロードします。
4. イメージを展開します。

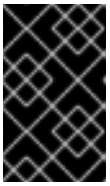
注記

クラスターが使用する前に RHOSP イメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. ダウンロードしたイメージから、RHOSP CLI を使用して **rhcos** という名前のイメージをクラスターに作成します。

```
$ openstack image create --container-format=bare --disk-format=qcow2 --file rhcos-
${RHCOS_VERSION}-openstack.qcow2 rhcos
```



重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

RHOSP にイメージをアップロードした後は、インストールプログラムでイメージを利用できます。

8.4.9. 外部ネットワークアクセスの確認

OpenShift Container Platform インストールプロセスでは、外部ネットワークへのアクセスが必要です。外部ネットワーク値をこれに指定する必要があります。指定しない場合には、デプロイメントは失敗します。このプロセスを実行する前に、外部ルータータイプのネットワークが Red Hat OpenStack Platform (RHOSP) に存在することを確認します。

前提条件

- OpenStack のネットワークサービスを、DHCP エージェントがインスタンスの DNS クエリーを転送できるように設定します。

手順

1. RHOSP CLI を使用して、'External' ネットワークの名前と ID を確認します。

```
$ openstack network list --long -c ID -c Name -c "Router Type"
```

出力例

```
+--------------------------------------+---------------+-------------+
| ID                                   | Name          | Router Type |
+--------------------------------------+---------------+-------------+
| 148a8023-62a7-4672-b018-003462f8d7dc | public_network | External    |
+--------------------------------------+---------------+-------------+
```

外部ルータータイプのあるネットワークがネットワーク一覧に表示されます。1 つ以上のネットワークが表示されない場合は、デフォルトの Floating IP ネットワークの作成 および デフォルトのプロバイダーネットワークの作成 を参照してください。

注記

Neutron トランクサービスプラグインが有効にされると、トランクポートがデフォルトで作成されます。詳細は、Neutron trunk port を参照してください。

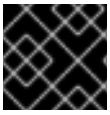
8.4.10. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

OpenShift Container Platform API およびクラスターで実行されるアプリケーションを、floating IP アドレスを使用してアクセス可能になるように設定できます。

8.4.10.1. floating IP アドレスを使ったアクセスの有効化

2 つの floating IP (FIP) アドレスを作成します。1 つ目は OpenShift Container Platform API への外部アクセス用の **API FIP** であり、2 つ目は OpenShift Container Platform アプリケーション用の **apps FIP** です。



重要

API FIP も **install-config.yaml** ファイルで使用されます。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

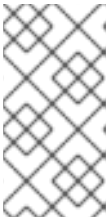
```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. 新しい FIP を反映させるには、以下のパターンに続くレコードを DNS サーバーに追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```



注記

DNS サーバーを制御しない場合は、代わりに **/etc/hosts** ファイルにレコードを追加します。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

8.4.11. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに **clouds.yaml** ファイルを生成します。

     重要

     パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

   - RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

     ```
     clouds:
       shiftstack:
         auth:
           auth_url: http://10.10.14.42:5000/v3
           project_name: shiftstack
           username: shiftstack_user
           password: XXX
           user_domain_name: Default
           project_domain_name: Default
       dev-env:
         region_name: RegionOne
         auth:
           username: 'devuser'
           password: XXX
           project_name: 'devonly'
           auth_url: 'https://10.10.14.22:5001/v2.0'
     ```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。
   a. 認証局ファイルをマシンにコピーします。
   b. マシンを認証局の信頼バンドルに追加します。

      ```
      $ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
      ```

   c. 信頼バンドルを更新します。

      ```
      $ sudo update-ca-trust extract
      ```

   d. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

      ```
      clouds:
      ```

```
shiftstack:
  ...
  cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。
   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値
   b. 現行ディレクトリー
   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)
   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

8.4.12. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat OpenStack Platform (RHOSP)

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

viii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

ix. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

x. コントロールプレーンおよびコンピュートノードに使用する 16 GB 以上の RAM で RHOSP フレーバーを指定します。

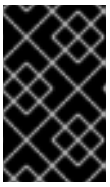
xi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

xii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

xiii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



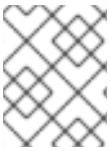
重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

これで、指定したディレクトリーに **install-config.yaml** ファイルが作成されます。

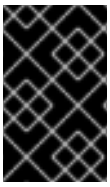
8.4.13. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

8.4.13.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.23 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

8.4.13.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表8.24 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`  hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

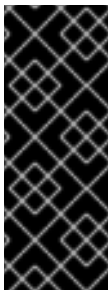
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

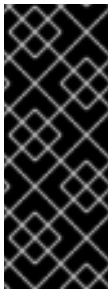
8.4.13.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.25 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

8.4.13.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表8.26 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。

パラメーター	説明	値
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。	文字列 (例: **m1.xlarge**)。
platform.openstack.lbFloatingIP	ロードバランサー API に関連付ける既存の Floating IP アドレス。	IP アドレス (例: **128.0.0.1**)。

8.4.13.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表8.27 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。 この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

8.4.13.6. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表8.28 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

8.4.13.7. RHOSP デプロイメントでのカスタムサブネット

オプションで、選択する Red Hat OpenStack Platform (RHOSP) サブネットにクラスターをデプロイすることができます。サブネットの GUID は、**install-config.yaml** ファイルの **platform.openstack.machinesSubnet** の値として渡されます。

このサブネットはクラスターのプライマリーサブネットとして使用されます。ノードとポートはこの上に作成されます。

カスタムサブネットを使用して OpenShift Container Platform インストーラーを実行する前に、以下を確認します。

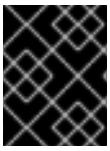
- ターゲットネットワークおよびサブネットが利用可能である。
- DHCP がターゲットサブネットで有効にされている。
- ターゲットネットワーク上でポートを作成するためのパーミッションがあるインストーラー認証情報を指定できます。
- ネットワーク設定にルーターが必要な場合、これは RHOSP で作成されます。一部の設定は、Floating IP アドレスの変換用のルーターに依存します。
- ネットワーク設定は、プロバイダーのネットワークに依存しません。プロバイダーネットワークはサポートされません。

注記

デフォルトでは、API VIP は x.x.x.5 を取得し、Ingress VIP はネットワークの CIDR ブロックから x.x.x.7 を取得します。これらのデフォルト値を上書きするには、DHCP 割り当てプール外の **platform.openstack.apiVIP** および **platform.openstack.ingressVIP** の値を設定します。

8.4.13.8. Kuryr を使用した OpenStack のカスタマイズされた install-config.yaml ファイルのサンプル

デフォルトの OpenShift SDN ではなく Kuryr SDN を使用してデプロイするには、**install-config.yaml** ファイルを変更して **Kuryr** を必要な **networking.networkType** として追加してから、デフォルトの OpenShift Container Platform SDN インストール手順に進む必要があります。このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。



重要

このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  serviceNetwork:
  - 172.30.0.0/16 1
  networkType: Kuryr
platform:
  openstack:
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
    trunkSupport: true 2
    octaviaSupport: true 3
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

1 Amphora Octavia ドライバーは、ロードバランサーごとに 2 つのポートを作成します。そのため、インストーラーが作成するサービスサブネットは、**serviceNetwork** プロパティーの値として指定される CIDR のサイズは 2 倍になります。IP アドレスの競合を防ぐには、範囲をより広くする必要があります。

2 3

trunkSupport と **octaviaSupport** の両方はインストーラーによって自動的に検出されるため、それらを設定する必要はありません。ただし、ご使用の環境がこれらの両方の要件を満たさないと、Kuryr SDN は適切に機能しません。トランクは Pod を RHOSP ネットワークに接続するために必要であり、Octavia は OpenShift Container Platform サービスを作成するために必要です。

8.4.13.9. マシンのカスタムサブネットの設定

インストールプログラムがデフォルトで使用する IP 範囲は、OpenShift Container Platform のインストール時に作成する Neutron サブネットと一致しない可能性があります。必要な場合は、インストール設定ファイルを編集して、新規マシンの CIDR 値を更新します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["machineNetwork"] = [{"cidr": "192.168.0.0/18"}]; 1
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

1 必要な Neutron サブネットに一致する値 (例: **192.0.2.0/24**) を挿入します。

   - 値を手動で設定するには、ファイルを開き、**networking.machineCIDR** の値を必要な Neutron サブネットに一致する値に設定します。

8.4.13.10. コンピュートマシンプールを空にする

独自のインフラストラクチャーを使用するインストールを実行するには、インストール設定ファイルのコンピュートマシンの数をゼロに設定します。その後、これらのマシンを手動で作成します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドラインで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。

- スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["compute"][0]["replicas"] = 0;
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

- 値を手動で設定するには、ファイルを開き、**compute.<first entry>.replicas** の値を **0** に設定します。

8.4.13.11. ネットワークタイプの変更

デフォルトで、インストールプログラムは **OpenShiftSDN** ネットワークタイプを選択します。代わりに Kuryr を使用するには、プログラムが生成したインストール設定ファイルの値を変更します。

前提条件

- OpenShift Container Platform インストールプログラムで生成された **install-config.yaml** ファイルがあります。

手順

1. コマンドプロンプトで、**install-config.yaml** が含まれるディレクトリーを参照します。
2. そのディレクトリーからスクリプトを実行して **install-config.yaml** ファイルを編集するか、または手動でファイルを更新します。
   - スクリプトを使用して値を設定するには、以下を実行します。

```
$ python -c '
import yaml;
path = "install-config.yaml";
data = yaml.safe_load(open(path));
data["networking"]["networkType"] = "Kuryr";
open(path, "w").write(yaml.dump(data, default_flow_style=False))'
```

   - 値を手動で設定するには、ファイルを開き、**networking.networkType** を **"Kuryr"** に設定します。

8.4.14. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
```

```
├── master.ign
├── metadata.json
└── worker.ign
```

8. メタデータファイルの **infraID** キーを環境変数としてエクスポートします。

```
$ export INFRA_ID=$(jq -r .infraID metadata.json)
```

ヒント

metadata.json から **infraID** キーを抽出し、作成するすべての RHOSP リソースの接頭辞として使用します。これを実行することで、同じプロジェクトで複数のデプロイメントを実行する際に名前の競合が発生しないようにします。

8.4.15. ブートストラップ Ignition ファイルの準備

OpenShift Container Platform インストールプロセスは、ブートストラップ Ignition 設定ファイルから作成されるブートストラップマシンに依存します。

ファイルを編集し、アップロードします。次に、Red Hat OpenStack Platform (RHOSP) がプライマリーファイルをダウンロードする際に使用するセカンダリーブートストラップ Ignition 設定ファイルを作成します。

前提条件

- インストーラープログラムが生成するブートストラップ Ignition ファイル **bootstrap.ign** があります。
- インストーラーのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成** を参照してください。
- HTTP(S) でアクセス可能な方法でブートストラップ Ignition ファイルを保存できます。
  - 記載された手順では RHOSP イメージサービス (Glance) を使用しますが、RHOSP ストレージサービス (Swift)、Amazon S3、内部 HTTP サーバー、またはアドホックの Nova サーバーを使用することもできます。

手順

1. 以下の Python スクリプトを実行します。スクリプトはブートストラップ Ignition ファイルを変更して、ホスト名および利用可能な場合は、実行時の CA 証明書ファイルを設定します。

```
import base64
import json
import os

with open('bootstrap.ign', 'r') as f:
    ignition = json.load(f)

files = ignition['storage'].get('files', [])

infra_id = os.environ.get('INFRA_ID', 'openshift').encode()
```

```
hostname_b64 = base64.standard_b64encode(infra_id + b'-bootstrap\n').decode().strip()
files.append(
{
    'path': '/etc/hostname',
    'mode': 420,
    'contents': {
        'source': 'data:text/plain;charset=utf-8;base64,' + hostname_b64,
        'verification': {}
    },
    'filesystem': 'root',
})

ca_cert_path = os.environ.get('OS_CACERT', '')
if ca_cert_path:
    with open(ca_cert_path, 'r') as f:
        ca_cert = f.read().encode()
        ca_cert_b64 = base64.standard_b64encode(ca_cert).decode().strip()

    files.append(
    {
        'path': '/opt/openshift/tls/cloud-ca-cert.pem',
        'mode': 420,
        'contents': {
            'source': 'data:text/plain;charset=utf-8;base64,' + ca_cert_b64,
            'verification': {}
        },
        'filesystem': 'root',
    })

ignition['storage']['files'] = files;

with open('bootstrap.ign', 'w') as f:
    json.dump(ignition, f)
```

2. RHOSP CLI を使用して、ブートストラップ Ignition ファイルを使用するイメージを作成します。

```
$ openstack image create --disk-format=raw --container-format=bare --file bootstrap.ign
<image_name>
```

3. イメージの詳細を取得します。

```
$ openstack image show <image_name>
```

file 値をメモします。これは **v2/images/<image_ID>/file** パターンをベースとしています。

注記

作成したイメージがアクティブであることを確認します。

4. イメージサービスのパブリックアドレスを取得します。

```
$ openstack catalog show image
```

5. パブリックアドレスとイメージ **file** 値を組み合わせ、結果を保存場所として保存します。この場所は、**<image_service_public_URL>/v2/images/<image_ID>/file** パターンをベースとしています。

6. 認証トークンを生成し、トークン ID を保存します。

```
$ openstack token issue -c id -f value
```

7. **$INFRA_ID-bootstrap-ignition.json** というファイルに以下のコンテンツを挿入し、独自の値に一致するようにプレースホルダーを編集します。

```
{
  "ignition": {
    "config": {
      "append": [{
        "source": "<storage_url>", 1
        "verification": {},
        "httpHeaders": [{
          "name": "X-Auth-Token", 2
          "value": "<token_ID>" 3
        }]
      }]
    },
    "security": {
      "tls": {
        "certificateAuthorities": [{
          "source": "data:text/plain;charset=utf-8;base64,<base64_encoded_certificate>", 4
          "verification": {}
        }]
      }
    },
    "timeouts": {},
    "version": "2.4.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1. **ignition.config.append.source** の値をブートストラップ Ignition ファイルのストレージ URL に置き換えます。
2. **httpHeaders** の **name** を **"X-Auth-Token"** に設定します。
3. **httpHeaders** の **value** をトークンの ID に設定します。
4. ブートストラップ Ignition ファイルサーバーが自己署名証明書を使用する場合は、base64 でエンコードされた証明書を含めます。

8. セカンダリー Ignition 設定ファイルを保存します。

ブートストラップ Ignition データはインストール時に RHOSP に渡されます。

警告

ブートストラップ Ignition ファイルには、**clouds.yaml** 認証情報などの機密情報が含まれます。これを安全な場所に保存し、インストールプロセスの完了後に削除します。

8.4.16. コントロールプレーンの Ignition 設定ファイルの作成

独自のインフラストラクチャーを使用して OpenShift Container Platform を Red Hat OpenStack Platform (RHOSP) にインストールするには、コントロールプレーンの Ignition 設定ファイルが必要です。複数の設定ファイルを作成する必要があります。

注記

ブートストラップ Ignition 設定と同様に、各コントロールプレーンマシンのホスト名を明示的に定義する必要があります。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
  - 変数が設定されていない場合は、**Kubernetes マニフェストおよび Ignition 設定ファイルの作成**を参照してください。

手順

- コマンドラインで、以下の Python スクリプトを実行します。

```
$ for index in $(seq 0 2); do
    MASTER_HOSTNAME="$INFRA_ID-master-$index\n"
    python -c "import base64, json, sys;
ignition = json.load(sys.stdin);
files = ignition['storage'].get('files', []);
files.append({'path': '/etc/hostname', 'mode': 420, 'contents': {'source':
'data:text/plain;charset=utf-8;base64,' +
base64.standard_b64encode(b'$MASTER_HOSTNAME').decode().strip(), 'verification': {}},
'filesystem': 'root'});
ignition['storage']['files'] = files;
json.dump(ignition, sys.stdout)" <master.ign >"$INFRA_ID-master-$index-ignition.json"
done
```

  以下の 3 つのコントロールプレーン Ignition ファイルが作成されます。**<INFRA_ID>-master-0-ignition.json**、**<INFRA_ID>-master-1-ignition.json**、および **<INFRA_ID>-master-2-ignition.json**。

8.4.17. ネットワークリソースの作成

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) インストールの OpenShift Container Platform に必要なネットワークリソースを作成します。時間を節約するには、セ

キュリティーグループ、ネットワーク、サブネット、ルーター、およびポートを生成する指定された Ansible Playbook を実行します。

手順

1. **common.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.9 common.yaml Ansible Playbook

```
- hosts: localhost
  gather_facts: no

  vars_files:
  - metadata.json

  tasks:
  - name: 'Compute resource names'
    set_fact:
      cluster_id_tag: "openshiftClusterID={{ infraID }}"
      os_network: "{{ infraID }}-network"
      os_subnet: "{{ infraID }}-nodes"
      os_router: "{{ infraID }}-external-router"
      # Port names
      os_port_api: "{{ infraID }}-api-port"
      os_port_ingress: "{{ infraID }}-ingress-port"
      os_port_bootstrap: "{{ infraID }}-bootstrap-port"
      os_port_master: "{{ infraID }}-master-port"
      os_port_worker: "{{ infraID }}-worker-port"
      # Security groups names
      os_sg_master: "{{ infraID }}-master"
      os_sg_worker: "{{ infraID }}-worker"
      # Server names
      os_bootstrap_server_name: "{{ infraID }}-bootstrap"
      os_cp_server_name: "{{ infraID }}-master"
      os_cp_server_group_name: "{{ infraID }}-master"
      os_compute_server_name: "{{ infraID }}-worker"
      # Trunk names
      os_cp_trunk_name: "{{ infraID }}-master-trunk"
      os_compute_trunk_name: "{{ infraID }}-worker-trunk"
      # Subnet pool name
      subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
      # Service network name
      os_svc_network: "{{ infraID }}-kuryr-service-network"
      # Service subnet name
      os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
      # Ignition files
      os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. **inventory.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.10 inventory.yaml Ansible Playbook

```
all:
  hosts:
    localhost:
```

```
    ansible_connection: local
    ansible_python_interpreter: "{{ansible_playbook_python}}"

    # User-provided values
    os_subnet_range: '10.0.0.0/16'
    os_flavor_master: 'm1.xlarge'
    os_flavor_worker: 'm1.large'
    os_image_rhcos: 'rhcos'
    os_external_network: 'external'
    # OpenShift API floating IP address
    os_api_fip: '203.0.113.23'
    # OpenShift Ingress floating IP address
    os_ingress_fip: '203.0.113.19'
    # Service subnet cidr
    svc_subnet_range: '172.30.0.0/16'
    os_svc_network_range: '172.30.0.0/15'
    # Subnet pool prefixes
    cluster_network_cidrs: '10.128.0.0/14'
    # Subnet pool prefix length
    host_prefix: '23'
    # Name of the SDN.
    # Possible values are OpenshiftSDN or Kuryr.
    os_networking_type: 'OpenshiftSDN'

    # Number of provisioned Control Plane nodes
    # 3 is the minimum number for a fully-functional cluster.
    os_cp_nodes_number: 3

    # Number of provisioned Compute nodes.
    # 3 is the minimum number for a fully-functional cluster.
    os_compute_nodes_number: 3
```

3. **security-groups.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.11 security-groups.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the master security group'
    os_security_group:
      name: "{{ os_sg_master }}"

  - name: 'Set master security group tag'
    command:
      cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_master }} "
```

```
- name: 'Create the worker security group'
  os_security_group:
    name: "{{ os_sg_worker }}"

- name: 'Set worker security group tag'
  command:
    cmd: "openstack security group set --tag {{ cluster_id_tag }} {{ os_sg_worker }} "

- name: 'Create master-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: icmp

- name: 'Create master-sg rule "machine config server"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 22623
    port_range_max: 22623

- name: 'Create master-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create master-sg rule "DNS (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: tcp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "DNS (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 53
    port_range_max: 53

- name: 'Create master-sg rule "mDNS"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    remote_ip_prefix: "{{ os_subnet_range }}"
    protocol: udp
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create master-sg rule "OpenShift API"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
```

```
    protocol: tcp
    port_range_min: 6443
    port_range_max: 6443

- name: 'Create master-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create master-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create master-sg rule "ovndb"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 6641
    port_range_max: 6642

- name: 'Create master-sg rule "master ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "master ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create master-sg rule "kube scheduler"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10259
    port_range_max: 10259

- name: 'Create master-sg rule "kube controller manager"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
```

```
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10257
    port_range_max: 10257

- name: 'Create master-sg rule "master ingress kubelet secure"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create master-sg rule "etcd"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 2379
    port_range_max: 2380

- name: 'Create master-sg rule "master ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "master ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create master-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_master }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"


- name: 'Create worker-sg rule "ICMP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: icmp

- name: 'Create worker-sg rule "SSH"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 22
    port_range_max: 22

- name: 'Create worker-sg rule "mDNS"'
```

```
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 5353
    port_range_max: 5353

- name: 'Create worker-sg rule "Ingress HTTP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 80
    port_range_max: 80

- name: 'Create worker-sg rule "Ingress HTTPS"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    port_range_min: 443
    port_range_max: 443

- name: 'Create worker-sg rule "router"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 1936
    port_range_max: 1936

- name: 'Create worker-sg rule "VXLAN"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 4789
    port_range_max: 4789

- name: 'Create worker-sg rule "Geneve"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 6081
    port_range_max: 6081

- name: 'Create worker-sg rule "worker ingress internal (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress internal (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
```

```
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 9000
    port_range_max: 9999

- name: 'Create worker-sg rule "worker ingress kubelet insecure"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 10250
    port_range_max: 10250

- name: 'Create worker-sg rule "worker ingress services (TCP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: tcp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "worker ingress services (UDP)"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: udp
    remote_ip_prefix: "{{ os_subnet_range }}"
    port_range_min: 30000
    port_range_max: 32767

- name: 'Create worker-sg rule "VRRP"'
  os_security_group_rule:
    security_group: "{{ os_sg_worker }}"
    protocol: '112'
    remote_ip_prefix: "{{ os_subnet_range }}"
```

4. **network.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.12 network.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the cluster network'
    os_network:
      name: "{{ os_network }}"
```

```
  - name: 'Set the cluster network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_network }}"

  - name: 'Create a subnet'
    os_subnet:
      name: "{{ os_subnet }}"
      network_name: "{{ os_network }}"
      cidr: "{{ os_subnet_range }}"
      allocation_pool_start: "{{ os_subnet_range | next_nth_usable(10) }}"
      allocation_pool_end: "{{ os_subnet_range | ipaddr('last_usable') }}"

  - name: 'Set the cluster subnet tag'
    command:
      cmd: "openstack subnet set --tag {{ cluster_id_tag }} {{ os_subnet }}"

  - name: 'Create the service network'
    os_network:
      name: "{{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

  - name: 'Set the service network tag'
    command:
      cmd: "openstack network set --tag {{ cluster_id_tag }} {{ os_svc_network }}"
    when: os_networking_type == "Kuryr"

  - name: 'Computing facts for service subnet'
    set_fact:
      first_ip_svc_subnet_range: "{{ svc_subnet_range | ipv4('network') }}"
      last_ip_svc_subnet_range: "{{ svc_subnet_range | ipaddr('last_usable') |ipmath(1) }}"
      first_ip_os_svc_network_range: "{{ os_svc_network_range | ipv4('network') }}"
      last_ip_os_svc_network_range: "{{ os_svc_network_range | ipaddr('last_usable')
|ipmath(1) }}"
      allocation_pool: ""
    when: os_networking_type == "Kuryr"

  - name: 'Get first part of OpenStack network'
    set_fact:
      allocation_pool: "{{ allocation_pool + '--allocation-pool start={{
first_ip_os_svc_network_range | ipmath(1) }},end={{ first_ip_svc_subnet_range |ipmath(-
1) }}' }}"
    when:
    - os_networking_type == "Kuryr"
    - first_ip_svc_subnet_range != first_ip_os_svc_network_range

  - name: 'Get last part of OpenStack network'
    set_fact:
      allocation_pool: "{{ allocation_pool + ' --allocation-pool start={{
last_ip_svc_subnet_range | ipmath(1) }},end={{ last_ip_os_svc_network_range |ipmath(-
1) }}' }}"
    when:
    - os_networking_type == "Kuryr"
    - last_ip_svc_subnet_range != last_ip_os_svc_network_range

  - name: 'Get end of allocation'
```

```
    set_fact:
      gateway_ip: "{{ allocation_pool.split('=')[-1] }}"
    when: os_networking_type == "Kuryr"

  - name: 'replace last IP'
    set_fact:
      allocation_pool: "{{ allocation_pool | replace(gateway_ip, gateway_ip | ipmath(-1))}}"
    when: os_networking_type == "Kuryr"

  - name: 'list service subnet'
    command:
      cmd: "openstack subnet list --name {{ os_svc_subnet }} --tag {{ cluster_id_tag }}"
    when: os_networking_type == "Kuryr"
    register: svc_subnet

  - name: 'Create the service subnet'
    command:
      cmd: "openstack subnet create --ip-version 4 --gateway {{ gateway_ip }} --subnet-
range {{ os_svc_network_range }} {{ allocation_pool }} --no-dhcp --network {{
os_svc_network }} --tag {{ cluster_id_tag }} {{ os_svc_subnet }}"
    when:
    - os_networking_type == "Kuryr"
    - svc_subnet.stdout == ""

  - name: 'list subnet pool'
    command:
      cmd: "openstack subnet pool list --name {{ subnet_pool }} --tags {{ cluster_id_tag }}"
    when: os_networking_type == "Kuryr"
    register: pods_subnet_pool

  - name: 'Create pods subnet pool'
    command:
      cmd: "openstack subnet pool create --default-prefix-length {{ host_prefix }} --pool-
prefix {{ cluster_network_cidrs }} --tag {{ cluster_id_tag }} {{ subnet_pool }}"
    when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout == ""

  - name: 'Create external router'
    os_router:
      name: "{{ os_router }}"
      network: "{{ os_external_network }}"
      interfaces:
      - "{{ os_subnet }}"

  - name: 'Set external router tag'
    command:
      cmd: "openstack router set --tag {{ cluster_id_tag }} {{ os_router }}"
    when: os_networking_type == "Kuryr"

  - name: 'Create the API port'
    os_port:
      name: "{{ os_port_api }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
```

```
    fixed_ips:
    - subnet: "{{ os_subnet }}"
      ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"

- name: 'Set API port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_api }}"

- name: 'Create the Ingress port'
  os_port:
    name: "{{ os_port_ingress }}"
    network: "{{ os_network }}"
    security_groups:
    - "{{ os_sg_worker }}"
    fixed_ips:
    - subnet: "{{ os_subnet }}"
      ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"

- name: 'Set the Ingress port tag'
  command:
    cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_ingress }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the API floating IP to API port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_api }} {{ os_api_fip }}"

# NOTE: openstack ansible module doesn't allow attaching Floating IPs to
# ports, let's use the CLI instead
- name: 'Attach the Ingress floating IP to Ingress port'
  command:
    cmd: "openstack floating ip set --port {{ os_port_ingress }} {{ os_ingress_fip }}"
```

5. コマンドラインで、**security-groups.yaml** Playbook を実行してセキュリティーグループを作成します。

```
$ ansible-playbook -i inventory.yaml security-groups.yaml
```

6. コマンドラインで、**network.yaml** Playbook を実行して、ネットワーク、サブネット、およびルーターを作成します。

```
$ ansible-playbook -i inventory.yaml network.yaml
```

7. オプション: Nova サーバーが使用するデフォルトのリゾルバーを制御する必要がある場合は、RHOSP CLI コマンドを実行します。

```
$ openstack subnet set --dns-nameserver <server_1> --dns-nameserver <server_2>
"$INFRA_ID-nodes"
```

8.4.18. ブートストラップマシンの作成

ブートストラップマシンを作成し、Red Hat OpenStack Platform (RHOSP) で実行するために必要なネットワークアクセスを付与します。Red Hat は、このプロセスを単純化するために実行する Ansible Playbook を提供しています。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. **bootstrap.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.13 bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"

  - name: 'Set bootstrap port tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ os_port_bootstrap }}"

  - name: 'Create the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_master }}"
      userdata: "{{ lookup('file', os_bootstrap_ignition) | string }}"
      auto_ip: no
```

```
    nics:
    - port-name: "{{ os_port_bootstrap }}"

- name: 'Create the bootstrap floating IP'
  os_floating_ip:
    state: present
    network: "{{ os_external_network }}"
    server: "{{ os_bootstrap_server_name }}"
```

3. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml bootstrap.yaml
```

4. ブートストラップサーバーがアクティブになった後に、ログを表示し、Ignition ファイルが受信されたことを確認します。

```
$ openstack console log show "$INFRA_ID-bootstrap"
```

8.4.19. コントロールプレーンマシンの作成

生成した Ignition 設定ファイルを使用して 3 つのコントロールプレーンマシンを作成します。

前提条件

- インストールプログラムのメタデータファイルのインフラストラクチャー ID は環境変数 (**$INFRA_ID**) として設定されます。
- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- **コントロールプレーン Ignition 設定ファイルの作成**で作成された 3 つの Ignition ファイル。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. コントロールプレーン Ignition 設定ファイルが作業ディレクトリーにない場合、それらをここにコピーします。
3. **control-plane.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.14 control-plane.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk
# netaddr

- import_playbook: common.yaml
```

```
- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the Control Plane ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_master }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(5) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(6) }}"
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
    register: ports

  - name: 'Set Control Plane ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"

  - name: 'List the Control Plane Trunks'
    command:
      cmd: "openstack network trunk list"
    when: os_networking_type == "Kuryr"
    register: control_plane_trunks

  - name: 'Create the Control Plane trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_cp_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_cp_trunk_name|string not in control_plane_trunks.stdout"

  - name: 'List the Server groups'
    command:
      cmd: "openstack server group list -f json -c ID -c Name"
    register: server_group_list

  - name: 'Parse the Server group ID from existing'
    set_fact:
      server_group_id: "{{ (server_group_list.stdout | from_json | json_query(list_query) |
first).ID }}"
    vars:
      list_query: "[?Name=='{{ os_cp_server_group_name }}']"
    when:
    - "os_cp_server_group_name|string in server_group_list.stdout"

  - name: 'Create the Control Plane server group'
    command:
      cmd: "openstack --os-compute-api-version=2.15 server group create -f json -c id --
policy=soft-anti-affinity {{ os_cp_server_group_name }}"
```

```
  register: server_group_created
  when:
  - server_group_id is not defined

- name: 'Parse the Server group ID from creation'
  set_fact:
    server_group_id: "{{ (server_group_created.stdout | from_json).id }}"
  when:
  - server_group_id is not defined

- name: 'Create the Control Plane servers'
  os_server:
    name: "{{ item.1 }}-{{ item.0 }}"
    image: "{{ os_image_rhcos }}"
    flavor: "{{ os_flavor_master }}"
    auto_ip: no
    # The ignition filename will be concatenated with the Control Plane node
    # name and its 0-indexed serial number.
    # In this case, the first node will look for this filename:
    #    "{{ infraID }}-master-0-ignition.json"
    userdata: "{{ lookup('file', [item.1, item.0, 'ignition.json'] | join('-')) | string }}"
    nics:
    - port-name: "{{ os_port_master }}-{{ item.0 }}"
    scheduler_hints:
      group: "{{ server_group_id }}"
  with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"
```

4. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml control-plane.yaml
```

5. 以下のコマンドを実行してブートストラッププロセスをモニターします。

```
$ openshift-install wait-for bootstrap-complete
```

コントロールプレーンマシンが実行され、クラスターに参加していることを確認できるメッセージが表示されます。

```
INFO API v1.14.6+f9b5405 up
INFO Waiting up to 30m0s for bootstrapping to complete...
...
INFO It is now safe to remove the bootstrap resources
```

8.4.20. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。

- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.4.21. ブートストラップリソースの削除

不要になったブートストラップリソースを削除します。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- コントロールプレーンマシンを実行中です。
  - マシンのステータスが分からない場合は、**クラスターステータスの確認**を参照してください。

手順

1. **down-bootstrap.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.15 down-bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the bootstrap server'
    os_server:
```

```
      name: "{{ os_bootstrap_server_name }}"
      state: absent
      delete_fip: yes

  - name: 'Remove the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      state: absent
```

2. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml down-bootstrap.yaml
```

ブートストラップポート、サーバー、および Floating IP アドレスが削除されます。

警告

ブートストラップ Ignition ファイル URL を無効にしていない場合は、無効にしてください。

8.4.22. コンピュートマシンの作成

コントロールプレーンの起動後、コンピュートマシンを作成します。

前提条件

- 共通ディレクトリー内の **inventory.yaml** および **common.yaml** Ansible Playbook。
  - これらのファイルが必要な場合は、**ネットワークリソースの作成**からこれらのファイルをコピーします。
- インストールプログラムが作成した **metadata.json** ファイルが Ansible Playbook と同じディレクトリーにあります。
- コントロールプレーンがアクティブです。

手順

1. コマンドラインで、作業ディレクトリーを **inventory.yaml** および **common.yaml** ファイルの場所に変更します。
2. **compute-nodes.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.16 compute-nodes.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
```

```
# openstacksdk
# netaddr

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Create the Compute ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      network: "{{ os_network }}"
      security_groups:
      - "{{ os_sg_worker }}"
      allowed_address_pairs:
      - ip_address: "{{ os_subnet_range | next_nth_usable(7) }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
    register: ports

  - name: 'Set Compute ports tag'
    command:
      cmd: "openstack port set --tag {{ cluster_id_tag }} {{ item.1 }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"

  - name: 'List the Compute Trunks'
    command:
      cmd: "openstack network trunk list"
    when: os_networking_type == "Kuryr"
    register: compute_trunks

  - name: 'Create the Compute trunks'
    command:
      cmd: "openstack network trunk create --parent-port {{ item.1.id }} {{
os_compute_trunk_name }}-{{ item.0 }}"
    with_indexed_items: "{{ ports.results }}"
    when:
    - os_networking_type == "Kuryr"
    - "os_compute_trunk_name|string not in compute_trunks.stdout"

  - name: 'Create the Compute servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      image: "{{ os_image_rhcos }}"
      flavor: "{{ os_flavor_worker }}"
      auto_ip: no
      userdata: "{{ lookup('file', 'worker.ign') | string }}"
      nics:
      - port-name: "{{ os_port_worker }}-{{ item.0 }}"
    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"
```

3. コマンドラインで Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml compute-nodes.yaml
```

次のステップ

- マシンの証明書署名要求を承認します。

8.4.23. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

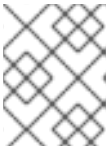
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

8.4.24. インストールの正常な実行の確認

OpenShift Container Platform のインストールが完了していることを確認します。

前提条件

- インストールプログラム (**openshift-install**) があります。

手順

- コマンドラインで、以下を入力します。

```
$ openshift-install --log-level debug wait-for install-complete
```

プログラムはコンソール URL と管理者のログイン情報を出力します。

8.4.25. Floating IP アドレスを使用したアプリケーションアクセスの設定

OpenShift Container Platform をインストールした後に、アプリケーションネットワークトラフィックを許可するように Red Hat OpenStack Platform (RHOSP) を設定します。

前提条件

- OpenShift Container Platform クラスターがインストールされている必要があります。
- **環境へのアクセスの有効化**で説明されているように、Floating IP アドレスが有効化されています。

手順

OpenShift Container Platform クラスターをインストールした後に、Floating IP アドレスを Ingress ポートに割り当てます。

1. ポートを表示します。

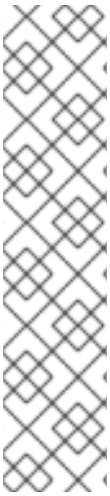
```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. ポートを IP アドレスに接続します。

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. ***apps.** のワイルドカード **A** レコードを DNS ファイルに追加します。

```
*.apps.<cluster name>.<base domain>  IN  A  <apps FIP>
```



注記

DNS サーバーを制御せず、非実稼働環境でアプリケーションアクセスを有効にする必要がある場合は、これらのホスト名を **/etc/hosts** に追加できます。

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

8.4.26. 次のステップ

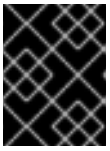
- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ノードポートへの外部アクセスを有効にする必要がある場合は、ノードポートを使用して Ingress クラストラフィックを設定 します。

8.5. ネットワークが制限された環境での OPENSTACK へのクラスターのインストール

OpenShift Container Platform 4.5 では、インストールリリースコンテンツの内部ミラーを作成して、クラスターをネットワークが制限された環境で Red Hat OpenStack Platform (RHOSP) にインストールできます。

前提条件

- ミラーホストでレジストリーを作成 し、OpenShift Container Platform の使用しているバージョン用の **imageContentSources** データを取得します。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了します。

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
    - アーキテクチャードキュメントの 利用可能なプラットフォームの一覧 を参照して、OpenShift Container Platform 4.5 が RHOSP バージョンと互換性があることを確認します。RHOSP サポートマトリックスの OpenShift Container Platform を参照して、プラットフォームのサポートを異なるバージョン間で比較することもできます。
- ネットワーク設定がプロバイダーのネットワークに依存しないことを確認します。プロバイダーネットワークはサポートされません。
- RHOSP でメタデータサービスが有効化されています。

8.5.1. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

8.5.1.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

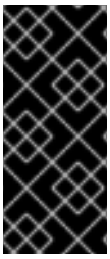
8.5.2. OpenShift Container Platform を RHOSP にインストールするリソースのガイドライン

OpenShift Container Platform のインストールをサポートするために、Red Hat OpenStack Platform (RHOSP) クォータは以下の要件を満たす必要があります。

表8.29 RHOSP のデフォルトの OpenShift Container Platform クラスターについての推奨リソース

リソース	値
Floating IP アドレス	3
ポート	15
ルーター	1
サブネット	1
RAM	112 GB
vCPU	28
ボリュームストレージ	275 GB
インスタンス	7
セキュリティーグループ	3
セキュリティーグループルール	60

クラスターは推奨されるリソースよりもリソースが少ない場合にも機能する場合がありますが、その場合のパフォーマンスは保証されません。



重要

RHOSP オブジェクトストレージ (Swift) が利用可能で、**swiftoperator** ロールを持つユーザーアカウントによって操作されている場合、これは OpenShift Container Platform イメージレジストリーのデフォルトバックエンドとして使用されます。この場合、ボリュームストレージ要件は 175 GB です。Swift 領域要件は、イメージレジストリーのサイズによって異なります。

注記

デフォルトで、セキュリティーグループおよびセキュリティーグループルールのクォータは低く設定される可能性があります。問題が生じた場合には、管理者として **openstack quota set --secgroups 3 --secgroup-rules 60 <project>** を実行して値を増やします。

OpenShift Container Platform デプロイメントは、コントロールプレーンマシン、コンピュートマシン、およびブートストラップマシンで設定されます。

8.5.2.1. コントロールプレーンおよびコンピュートマシン

デフォルトで、OpenShift Container Platform インストールプロセスは 3 つのコントロールプレーンおよび 3 つのコンピュートマシンを使用します。

それぞれのマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

ヒント

コンピュートマシンは、OpenShift Container Platform で実行されるアプリケーションをホストします。できるだけ多くのアプリケーションを実行することが意図されています。

8.5.2.2. ブートストラップマシン

インストール時に、ブートストラップマシンは一時的にプロビジョニングされ、コントロールプレーンを初期化します。実稼働環境用のコントロールプレーンの準備ができた後に、ブートストラップマシンのプロビジョニングは解除されます。

ブートストラップマシンには以下が必要です。

- RHOSP クォータからのインスタンス
- RHOSP クォータからのポート
- 少なくとも 16 GB のメモリー、4 つの vCPU および 25 GB のストレージ領域があるフレーバー

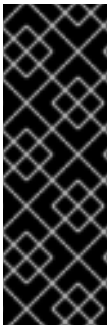
8.5.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

8.5.4. RHOSP での Swift の有効化

Swift は、**swiftoperator** ロールのあるユーザーアカウントによって操作されます。インストールプログラムを実行する前に、ロールをアカウントに追加します。

重要

Swift として知られる Red Hat OpenStack Platform (RHOSP) オブジェクトストレージサービス が利用可能な場合、OpenShift Container Platform はこれをイメージレジストリーストレージとして使用します。利用できない場合、インストールプログラムは Cinder として知られる RHOSP ブロックストレージサービスに依存します。

Swift が存在し、これを使用する必要がある場合は、Swift へのアクセスを有効にする必要があります。これが存在しない場合や使用する必要がない場合は、このセクションを省略してください。

前提条件

- ターゲット環境に RHOSP 管理者アカウントがあります。
- Swift サービスがインストールされています。
- Ceph RGW で、**account in url** オプションが有効化されています。

手順

RHOSP 上で Swift を有効にするには、以下を実行します。

1. RHOSP CLI の管理者として、**swiftoperator** ロールを Swift にアクセスするアカウントに追加します。

```
$ openstack role add --user <user> --project <project> swiftoperator
```

RHOSP デプロイメントでは、イメージレジストリーに Swift を使用することができます。

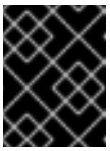
8.5.5. インストールプログラムのパラメーターの定義

OpenShift Container Platform インストールプログラムは、**clouds.yaml** というファイルを使用します。このファイルは、プロジェクト名、ログイン情報、認可サービスの URL を含む Red Hat OpenStack Platform (RHOSP) 設定パラメーターを説明します。

手順

1. **clouds.yaml** ファイルを作成します。
   - RHOSP ディストリビューションに Horizon Web UI が含まれる場合には、そこに

clouds.yaml ファイルを生成します。



重要

パスワードを必ず **auth** フィールドに追加してください。シークレットは、**clouds.yaml** の 別のファイル に保持できます。

- RHOSP ディストリビューションに Horizon Web UI が含まれない場合や Horizon を使用する必要がない場合には、このファイルを独自に作成します。**clouds.yaml** についての詳細は、RHOSP ドキュメントの Config files を参照してください。

```
clouds:
  shiftstack:
    auth:
      auth_url: http://10.10.14.42:5000/v3
      project_name: shiftstack
      username: shiftstack_user
      password: XXX
      user_domain_name: Default
      project_domain_name: Default
  dev-env:
    region_name: RegionOne
    auth:
      username: 'devuser'
      password: XXX
      project_name: 'devonly'
      auth_url: 'https://10.10.14.22:5001/v2.0'
```

2. RHOSP インストールでエンドポイント認証用に自己署名認証局 (CA) を使用する場合、以下を実行します。

   a. 認証局ファイルをマシンにコピーします。

   b. マシンを認証局の信頼バンドルに追加します。

   ```
   $ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
   ```

   c. 信頼バンドルを更新します。

   ```
   $ sudo update-ca-trust extract
   ```

   d. **cacerts** キーを **clouds.yaml** ファイルに追加します。この値は、CA 証明書への絶対的な root 以外によるアクセスが可能なパスである必要があります。

   ```
   clouds:
     shiftstack:
       ...
       cacert: "/etc/pki/ca-trust/source/anchors/ca.crt.pem"
   ```

ヒント

カスタム CA 証明書を使用してインストーラーを実行した後に、**cloud-provider-config** キーマップの **ca-cert.pem** キーの値を編集して証明書を更新できます。コマンドラインで、以下を実行します。

```
$ oc edit configmap -n openshift-config cloud-provider-config
```

3. **clouds.yaml** ファイルを以下の場所のいずれかに置きます。
   a. **OS_CLIENT_CONFIG_FILE** 環境変数の値
   b. 現行ディレクトリー
   c. Unix 固有のユーザー設定ディレクトリー (例: **~/.config/openstack/clouds.yaml**)
   d. Unix 固有のサイト設定ディレクトリー (例: **/etc/openstack/clouds.yaml**)
   インストールプログラムはこの順序で **clouds.yaml** を検索します。

8.5.6. ネットワークが制限されたインストール用の RHCOS イメージの作成

Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された Red Hat OpenStack Platform (RHOSP) 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.5 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - OpenStack Image (QCOW)**イメージをダウンロードします。
4. イメージを展開します。

注記

クラスターが使用する前にイメージを圧縮解除する必要があります。ダウンロードしたファイルの名前に、**.gz** または **.tgz** などの圧縮拡張子が含まれていない場合があります。ファイルを圧縮するか、またはどのように圧縮するかを確認するには、コマンドラインで以下を入力します。

```
$ file <name_of_downloaded_file>
```

5. 圧縮解除したイメージを、Glance などの bastion サーバーからアクセス可能な場所にアップロードします。以下に例を示します。

```
$ openstack image create --file rhcos-44.81.202003110027-0-openstack.x86_64.qcow2 --
disk-format qcow2 rhcos-${RHCOS_VERSION}
```

重要

RHOSP 環境によっては、**.raw** または **.qcow2** 形式 のいずれかでイメージをアップロードできる場合があります。Ceph を使用する場合は、**.raw** 形式を使用する必要があります。

警告

インストールプログラムが同じ名前を持つ複数のイメージを見つける場合、それらのイメージのいずれかがランダムに選択されます。この動作を回避するには、RHOSP でリソースの一意の名前を作成します。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

8.5.7. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat OpenStack Platform (RHOSP)

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。ネットワークが制限されたインストールでは、これらのファイルが bastion ホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得します。

手順

1. **install-config.yaml** ファイルを作成します。

a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **openstack** を選択します。

viii. クラスターのインストールに使用する Red Hat OpenStack Platform (RHOSP) の外部ネットワーク名を指定します。

ix. OpenShift API への外部アクセスに使用する floating IP アドレスを指定します。

x. コントロールプレーンおよびコンピュートノードに使用する 16 GB 以上の RAM で RHOSP フレーバーを指定します。

xi. クラスターをデプロイするベースドメインを選択します。すべての DNS レコードはこのベースのサブドメインとなり、クラスター名も含まれます。

xii. クラスターの名前を入力します。名前は 14 文字以下でなければなりません。

xiii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルで、**platform.openstack.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  openstack:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
openstack.x86_64.qcow2.gz?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<bastion_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、**<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既存のサブネットを指定します。

d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
```

```
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

8.5.7.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

8.5.7.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表8.30 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列

パラメーター	説明	値
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。文字列は 14 文字以上でなければなりません。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

8.5.7.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表8.31 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		``
		networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		    hostPrefix: 23
		``

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

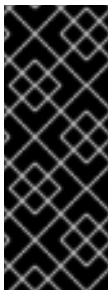
パラメーター	説明	値
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

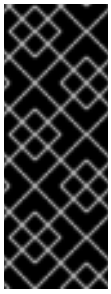
8.5.7.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表8.32 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。

パラメーター	説明	値
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

8.5.7.1.4. 追加の Red Hat OpenStack Platform (RHOSP) 設定パラメーター

追加の RHOSP 設定パラメーターは以下の表で説明されています。

表8.33 追加の RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.rootVolume.size	コンピュートマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
compute.platform.openstack.rootVolume.type	コンピュートマシンの場合、root のボリュームタイプです。	文字列 (例: **performance**)。
controlPlane.platform.openstack.rootVolume.size	コントロールプレーンマシンの場合、root ボリュームのギガバイトのサイズになります。この値を設定しない場合、マシンは一時ストレージを使用します。	整数 (例: **30**)。
controlPlane.platform.openstack.rootVolume.type	コントロールプレーンマシンの場合、root ボリュームのタイプです。	文字列 (例: **performance**)。
platform.openstack.cloud	**clouds.yaml** ファイルのクラウド一覧にある使用する RHOCP クラウドの名前。	文字列 (例: **MyCloud**)。

パラメーター	説明	値
platform.openstack.externalNetwork	インストールに使用される RHOSP の外部ネットワーク名。	文字列 (例: **external**)。
platform.openstack.computeFlavor	コントロールプレーンおよびコンピュートマシンに使用する RHOSP フレーバー。	文字列 (例: **m1.xlarge**)。
platform.openstack.lbFloatingIP	ロードバランサー API に関連付ける既存の Floating IP アドレス。	IP アドレス (例: **128.0.0.1**)。

8.5.7.1.5. オプションの RHOSP 設定パラメーター

オプションの RHOSP 設定パラメーターは、以下の表で説明されています。

表8.34 オプションの RHOSP パラメーター

パラメーター	説明	値
compute.platform.openstack.additionalNetworkIDs	コンピュートマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
compute.platform.openstack.additionalSecurityGroupIDs	コンピュートマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.
controlPlane.platform.openstack.additionalNetworkIDs	コントロールプレーンマシンに関連付けられた追加のネットワーク。追加ネットワーク用に許可されるアドレスのペアは作成されません。	文字列としての 1 つ以上の UUID の一覧。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**。
controlPlane.platform.openstack.additionalSecurityGroupIDs	コントロールプレーンマシンに関連付けられた追加のセキュリティーグループ。	文字列としての 1 つ以上の UUID の一覧。例: **7ee219f3-d2e9-48a1-96c2-e7429f1b0da7**.

パラメーター	説明	値
platform.openstack.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。 ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。 例: **http://mirror.example.com/images/rhcos-43.81.201912131630.0-openstack.x86_64.qcow2.gz?sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d**。 この値は、既存の Glance イメージの名前にもなり得ます (例: **my-rhcos**)。

platform.openstack.defaultMachinePlatform	デフォルトのマシンプールプラットフォームの設定。	<pre>{
		  "type": "ml.large",
		  "rootVolume": {
		  "size": 30,
		  "type": "performance"
		  }
		}</pre>

platform.openstack.externalDNS	クラスターインスタンスが DNS 解決に使用する外部 DNS サーバーの IP アドレス。	文字列としての IP アドレスの一覧。例: **["8.8.8.8", "192.168.1.12"]**
platform.openstack.machinesSubnet	クラスターのノードが使用する RHOSP サブネットの UUID。ノードおよび仮想 IP (VIP) ポートがこのサブネットに作成されます。 **networking.machineNetwork** の最初の項目は **machinesSubnet** の値に一致する必要があります。 カスタムサブネットにデプロイする場合、OpenShift Container Platform インストーラーに外部 DNS サーバーを指定することはできません。代わりに、DNS を RHOSP のサブネットに追加します。	文字列としての UUID。例: **fa806b2f-ac49-4bce-b9db-124bc64209bf**

8.5.7.1.6. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表8.35 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

8.5.7.2. 制限された OpenStack インストールのカスタマイズされた install-config.yaml ファイルのサンプル

このサンプル **install-config.yaml** は、すべての可能な Red Hat OpenStack Platform (RHOSP) カスタマイズオプションを示しています。

重要

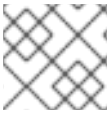
このサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。

```
apiVersion: v1
baseDomain: example.com
clusterID: os-test
controlPlane:
  name: master
  platform: {}
  replicas: 3
compute:
- name: worker
  platform:
    openstack:
      type: ml.large
  replicas: 3
metadata:
  name: example
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineCIDR: 10.0.0.0/16
  serviceNetwork:
```

```
  - 172.30.0.0/16
  networkType: OpenShiftSDN
platform:
  openstack:
    region: region1
    cloud: mycloud
    externalNetwork: external
    computeFlavor: m1.xlarge
    lbFloatingIP: 128.0.0.1
fips: false
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
additionalTrustBundle: |

  -----BEGIN CERTIFICATE-----

  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ

  -----END CERTIFICATE-----

imageContentSources:
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <mirror_registry>/<repo_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

8.5.8. SSH プライベートキーの生成およびエージェントへの追加

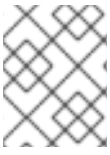
クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

8.5.9. 環境へのアクセスの有効化

デプロイ時に、OpenShift Container Platform マシンはすべて Red Hat OpenStack Platform (RHOSP) テナントネットワークに作成されます。したがって、ほとんどの RHOSP デプロイメントでは直接アクセスできません。

OpenShift Container Platform API およびクラスターで実行されるアプリケーションを、floating IP アドレスを使用/不使用でアクセス可能になるように設定できます。

8.5.9.1. floating IP アドレスを使ったアクセスの有効化

2 つの floating IP (FIP) アドレスを作成します。1 つ目は OpenShift Container Platform API への外部アクセス用の **API FIP** であり、2 つ目は OpenShift Container Platform アプリケーション用の **apps FIP** です。

重要

API FIP も **install-config.yaml** ファイルで使用されます。

手順

1. Red Hat OpenStack Platform (RHOSP) CLI を使用して、API FIP を作成します。

```
$ openstack floating ip create --description "API <cluster_name>.<base_domain>" <external network>
```

2. Red Hat OpenStack Platform (RHOSP) CLI を使用して、apps (アプリ)、または Ingress、FIP を作成します。

```
$ openstack floating ip create --description "Ingress <cluster_name>.<base_domain>" <external network>
```

3. 新しい FIP を反映させるには、以下のパターンに続くレコードを DNS サーバーに追加します。

```
api.<cluster_name>.<base_domain>.  IN  A  <API_FIP>
*.apps.<cluster_name>.<base_domain>. IN  A <apps_FIP>
```

注記

DNS サーバーを制御しない場合は、代わりに **/etc/hosts** ファイルにレコードを追加します。このアクションにより、API は他者のアクセスできない状態になり、この状態は実稼働デプロイメントには適していませんが、開発およびテスト目的のインストールが可能になります。

ヒント

Floating IP アドレスを割り当て、ファイアウォール設定を更新することで、OpenShift Container Platform リソースがクラスター外で利用できる状態にすることができます。

8.5.9.2. Floating IP アドレスを使用しないアクセスの有効化

Floating IP アドレスを使用できない場合でも、OpenShift Container Platform のインストールは終了できる可能性があります。ただし、インストールプログラムは API アクセスを待機してタイムアウトする場合は失敗します。

インストールプログラムがタイムアウトすると、クラスターは初期化される可能性があります。ブートストラップ処理が開始されたら、これを完了する必要があります。デプロイ後にクラスターのネットワーク設定を編集する必要があります。

8.5.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

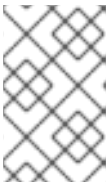
- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

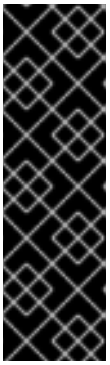
1. **<installation_directory>** については、以下を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

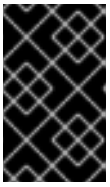
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

8.5.11. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

8.5.12. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

8.5.13. Floating IP アドレスを使用したアプリケーションアクセスの設定

OpenShift Container Platform をインストールした後に、アプリケーションネットワークトラフィックを許可するように Red Hat OpenStack Platform (RHOSP) を設定します。

前提条件

- OpenShift Container Platform クラスターがインストールされている必要があります。
- **環境へのアクセスの有効化**で説明されているように、Floating IP アドレスが有効化されています。

手順

OpenShift Container Platform クラスターをインストールした後に、Floating IP アドレスを Ingress ポートに割り当てます。

1. ポートを表示します。

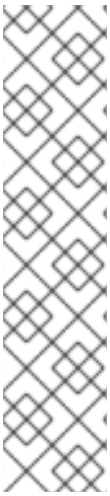
```
$ openstack port show <cluster name>-<clusterID>-ingress-port
```

2. ポートを IP アドレスに接続します。

```
$ openstack floating ip set --port <ingress port ID> <apps FIP>
```

3. ***apps.** のワイルドカード **A** レコードを DNS ファイルに追加します。

```
*.apps.<cluster name>.<base domain>  IN  A  <apps FIP>
```



注記

DNS サーバーを制御せず、非実稼働環境でアプリケーションアクセスを有効にする必要がある場合は、これらのホスト名を **/etc/hosts** に追加できます。

```
<apps FIP> console-openshift-console.apps.<cluster name>.<base domain>
<apps FIP> integrated-oauth-server-openshift-authentication.apps.<cluster name>.
<base domain>
<apps FIP> oauth-openshift.apps.<cluster name>.<base domain>
<apps FIP> prometheus-k8s-openshift-monitoring.apps.<cluster name>.<base
domain>
<apps FIP> grafana-openshift-monitoring.apps.<cluster name>.<base domain>
<apps FIP> <app name>.apps.<cluster name>.<base domain>
```

次のステップ

- クラスターをカスタマイズ します。
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- ネットワークが制限された環境での Operator Lifecycle Manager (OLM) の使用 方法について参照します。
- Cluster Samples Operator および **must-gather** ツールの イメージストリームを設定 します。

8.6. OPENSTACK でのクラスターのアンインストール

Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除できます。

8.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、または 削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

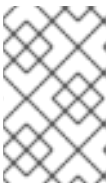
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

8.7. 独自のインフラストラクチャーからの OPENSTACK のクラスターのアンインストール

ユーザーによってプロビジョニングされたインフラストラクチャーの Red Hat OpenStack Platform (RHOSP) にデプロイしたクラスターを削除することができます。

8.7.1. 前提条件

- 以下がマシン上にあります。

- 削除プロセスに使用するファイルを作成できる単一ディレクトリー
- Python 3

8.7.2. Playbook 依存関係のダウンロード

ユーザーによってプロビジョニングされたインフラストラクチャーでの削除プロセスを簡素化する Ansible Playbook には、複数の Python モジュールが必要です。プロセスを実行するマシンで、モジュールのリポジトリーを追加し、それらをダウンロードします。

注記

この手順では、Red Hat Enterprise Linux (RHEL) 8 を使用していることを前提としています。

前提条件

- Python 3 がマシンにインストールされています。

手順

1. コマンドラインで、リポジトリーを追加します。
   a. Red Hat Subscription Manager に登録します。

```
$ sudo subscription-manager register # If not done already
```

   b. 最新のサブスクリプションデータをプルします。

```
$ sudo subscription-manager attach --pool=$YOUR_POOLID # If not done already
```

   c. 現在のリポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=* # If not done already
```

   d. 必要なリポジトリーを追加します。

```
$ sudo subscription-manager repos \
  --enable=rhel-8-for-x86_64-baseos-rpms \
  --enable=openstack-16-tools-for-rhel-8-x86_64-rpms \
  --enable=ansible-2.9-for-rhel-8-x86_64-rpms \
  --enable=rhel-8-for-x86_64-appstream-rpms
```

2. モジュールをインストールします。

```
$ sudo yum install python3-openstackclient ansible python3-openstacksdk
```

3. **python** コマンドが **python3** を参照していることを確認します。

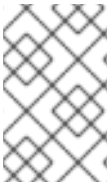
```
$ sudo alternatives --set python /usr/bin/python3
```

8.7.3. 独自のインフラストラクチャーを使用する RHOSP のクラスターの削除

独自のインフラストラクチャーを使用する Red Hat OpenStack Platform (RHOSP) の OpenShift Container Platform クラスターを削除できます。削除プロセスを迅速に完了するには、複数の Ansible Playbook を作成し、実行します。

前提条件

- Python 3 がマシンにインストールされています。
- Downloading playbook dependencies でモジュールをダウンロードしています。



手順

OpenShift Container Platform のインストール時より **common.yaml** および **inventory.yaml** Playbook が残っている場合があります。その場合、手順の最初の 2 つのステップを省略できます。

1. **common.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.17 common.yaml Ansible Playbook

```
- hosts: localhost
  gather_facts: no

  vars_files:
  - metadata.json

  tasks:
  - name: 'Compute resource names'
    set_fact:
      cluster_id_tag: "openshiftClusterID={{ infraID }}"
      os_network: "{{ infraID }}-network"
      os_subnet: "{{ infraID }}-nodes"
      os_router: "{{ infraID }}-external-router"
      # Port names
      os_port_api: "{{ infraID }}-api-port"
      os_port_ingress: "{{ infraID }}-ingress-port"
      os_port_bootstrap: "{{ infraID }}-bootstrap-port"
      os_port_master: "{{ infraID }}-master-port"
      os_port_worker: "{{ infraID }}-worker-port"
      # Security groups names
      os_sg_master: "{{ infraID }}-master"
      os_sg_worker: "{{ infraID }}-worker"
      # Server names
      os_bootstrap_server_name: "{{ infraID }}-bootstrap"
      os_cp_server_name: "{{ infraID }}-master"
      os_cp_server_group_name: "{{ infraID }}-master"
      os_compute_server_name: "{{ infraID }}-worker"
      # Trunk names
      os_cp_trunk_name: "{{ infraID }}-master-trunk"
      os_compute_trunk_name: "{{ infraID }}-worker-trunk"
      # Subnet pool name
      subnet_pool: "{{ infraID }}-kuryr-pod-subnetpool"
      # Service network name
      os_svc_network: "{{ infraID }}-kuryr-service-network"
      # Service subnet name
```

```
    os_svc_subnet: "{{ infraID }}-kuryr-service-subnet"
    # Ignition files
    os_bootstrap_ignition: "{{ infraID }}-bootstrap-ignition.json"
```

2. **inventory.yaml** というローカルファイルに以下のコンテンツを挿入し、独自の値に一致するように値を編集します。

例8.18 inventory.yaml Ansible Playbook

```
all:
  hosts:
    localhost:
      ansible_connection: local
      ansible_python_interpreter: "{{ansible_playbook_python}}"

      # User-provided values
      os_subnet_range: '10.0.0.0/16'
      os_flavor_master: 'm1.xlarge'
      os_flavor_worker: 'm1.large'
      os_image_rhcos: 'rhcos'
      os_external_network: 'external'
      # OpenShift API floating IP address
      os_api_fip: '203.0.113.23'
      # OpenShift Ingress floating IP address
      os_ingress_fip: '203.0.113.19'
      # Service subnet cidr
      svc_subnet_range: '172.30.0.0/16'
      os_svc_network_range: '172.30.0.0/15'
      # Subnet pool prefixes
      cluster_network_cidrs: '10.128.0.0/14'
      # Subnet pool prefix length
      host_prefix: '23'
      # Name of the SDN.
      # Possible values are OpenshiftSDN or Kuryr.
      os_networking_type: 'OpenshiftSDN'

      # Number of provisioned Control Plane nodes
      # 3 is the minimum number for a fully-functional cluster.
      os_cp_nodes_number: 3

      # Number of provisioned Compute nodes.
      # 3 is the minimum number for a fully-functional cluster.
      os_compute_nodes_number: 3
```

3. **オプション**: クラスターで Kuryr を使用する場合は、**down-load-balancers.yaml** というローカルファイルに以下のコンテンツを挿入します。

例8.19 down-load-balancers.yaml

```
# Required Python packages:
#
# ansible
# openstackcli
# openstacksdk
```

```
- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Get an auth token'
    os_auth:
    register: cloud
    when: os_networking_type == "Kuryr"

  - name: 'List octavia versions'
    uri:
      method: GET
      headers:
        X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
        Content-Type: 'application/json'
      url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url') }}/"
    register: octavia_versions
    when: os_networking_type == "Kuryr"

  - set_fact:
      versions: "{{ octavia_versions.json.versions | selectattr('id', 'match', 'v2.5') |
map(attribute='id') | list }}"
    when: os_networking_type == "Kuryr"

  - name: 'List tagged loadbalancers'
    uri:
      method: GET
      headers:
        X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
      url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
}}/v2.0/lbaas/loadbalancers?tags={{cluster_id_tag}}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length > 0
    register: lbs_tagged

  # NOTE: Kuryr creates an Octavia load balancer
  # for each service present on the cluster. Let's make
  # sure to remove the resources generated.
  - name: 'Remove the cluster load balancers'
    command:
      cmd: "openstack loadbalancer delete --cascade {{ item.id }}"
    with_items: "{{ lbs_tagged.json.loadbalancers }}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length > 0
    - '"PENDING" not in item.provisioning_status'

  - name: 'List loadbalancers tagged on description'
    uri:
      method: GET
```

```
      headers:
        X-Auth-Token: "{{ cloud.ansible_facts.auth_token }}"
      url: "{{ cloud.ansible_facts.service_catalog | selectattr('name', 'match', 'octavia') | first |
json_query('endpoints') | selectattr('interface', 'match', 'public') | first | json_query('url')
}}/v2.0/lbaas/loadbalancers?description={{cluster_id_tag}}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length == 0
    register: lbs_description

  # NOTE: Kuryr creates an Octavia load balancer
  # for each service present on the cluster. Let's make
  # sure to remove the resources generated.
  - name: 'Remove the cluster load balancers'
    command:
      cmd: "openstack loadbalancer delete --cascade {{ item.id }}"
    with_items: "{{ lbs_description.json.loadbalancers }}"
    when:
    - os_networking_type == "Kuryr"
    - versions | length == 0
    - '"PENDING" not in item.provisioning_status'
```

4. **down-compute-nodes.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.20 down-compute-nodes.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the Compute servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      state: absent
    with_indexed_items: "{{ [os_compute_server_name] * os_compute_nodes_number }}"

  - name: 'List the Compute trunks'
    command:
      cmd: "openstack network trunk list -c Name -f value"
    when: os_networking_type == "Kuryr"
    register: trunks

  - name: 'Remove the Compute trunks'
    command:
      cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
    when:
    - os_networking_type == "Kuryr"
```

```
  - (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
  with_indexed_items: "{{ [os_compute_trunk_name] * os_compute_nodes_number }}"

- name: 'Remove the Compute ports'
  os_port:
    name: "{{ item.1 }}-{{ item.0 }}"
    state: absent
  with_indexed_items: "{{ [os_port_worker] * os_compute_nodes_number }}"
```

5. **down-control-plane.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.21 down-control-plane.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the Control Plane servers'
    os_server:
      name: "{{ item.1 }}-{{ item.0 }}"
      state: absent
    with_indexed_items: "{{ [os_cp_server_name] * os_cp_nodes_number }}"

  - name: 'Remove the Control Plane server group'
    os_server_group:
      name: "{{ os_cp_server_group_name }}"
      state: absent

  - name: 'List the Compute trunks'
    command:
      cmd: "openstack network trunk list -c Name -f value"
    when: os_networking_type == "Kuryr"
    register: trunks

  - name: 'Remove the Control Plane trunks'
    command:
      cmd: "openstack network trunk delete {{ item.1 }}-{{ item.0 }}"
    when:
    - os_networking_type == "Kuryr"
    - (item.1|string + '-' + item.0|string) in trunks.stdout_lines|list
    with_indexed_items: "{{ [os_cp_trunk_name] * os_cp_nodes_number }}"

  - name: 'Remove the Control Plane ports'
    os_port:
      name: "{{ item.1 }}-{{ item.0 }}"
      state: absent
    with_indexed_items: "{{ [os_port_master] * os_cp_nodes_number }}"
```

6. **down-bootstrap.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.22 down-bootstrap.yaml

```
# Required Python packages:
#
# ansible
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'Remove the bootstrap server'
    os_server:
      name: "{{ os_bootstrap_server_name }}"
      state: absent
      delete_fip: yes

  - name: 'Remove the bootstrap server port'
    os_port:
      name: "{{ os_port_bootstrap }}"
      state: absent
```

7. **down-network.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.23 down-network.yaml

```
# Required Python packages:
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'List ports attatched to router'
    command:
      cmd: "openstack port list --device-owner=network:router_interface --tags {{
cluster_id_tag }} -f value -c id"
    register: router_ports

  - name: 'Remove the ports from router'
    command:
      cmd: "openstack router remove port {{ os_router }} {{ item.1}}"
    with_indexed_items: "{{ router_ports.stdout_lines }}"
```

```
  - name: 'List ha ports attached to router'
    command:
      cmd: "openstack port list --device-owner=network:ha_router_replicated_interface --
tags {{ cluster_id_tag }} -f value -c id"
    register: ha_router_ports

  - name: 'Remove the ha ports from router'
    command:
      cmd: "openstack router remove port {{ os_router }} {{ item.1}}"
    with_indexed_items: "{{ ha_router_ports.stdout_lines }}"

  - name: 'List ports'
    command:
      cmd: "openstack port list --tags {{ cluster_id_tag }} -f value -c id "
    register: ports

  - name: 'Remove the cluster ports'
    command:
      cmd: "openstack port delete {{ item.1}}"
    with_indexed_items: "{{ ports.stdout_lines }}"

  - name: 'Remove the cluster router'
    os_router:
      name: "{{ os_router }}"
      state: absent

  - name: 'List cluster networks'
    command:
      cmd: "openstack network list --tags {{ cluster_id_tag }} -f value -c Name"
    register: networks

  - name: 'Remove the cluster networks'
    os_network:
      name: "{{ item.1}}"
      state: absent
    with_indexed_items: "{{ networks.stdout_lines }}"

  - name: 'List the cluster subnet pool'
    command:
      cmd: "openstack subnet pool list --name {{ subnet_pool }}"
    when: os_networking_type == "Kuryr"
    register: pods_subnet_pool

  - name: 'Remove the cluster subnet pool'
    command:
      cmd: "openstack subnet pool delete {{ subnet_pool }}"
    when:
    - os_networking_type == "Kuryr"
    - pods_subnet_pool.stdout != ""
```

8. **down-security-groups.yaml** というローカルファイルに、以下のコンテンツを挿入します。

例8.24 down-security-groups.yaml

```
# Required Python packages:
```

```
#
# ansible
# openstackclient
# openstacksdk

- import_playbook: common.yaml

- hosts: all
  gather_facts: no

  tasks:
  - name: 'List security groups'
    command:
      cmd: "openstack security group list --tags {{ cluster_id_tag }} -f value -c ID"
    register: security_groups

  - name: 'Remove the cluster security groups'
    command:
      cmd: "openstack security group delete {{ item.1 }}"
    with_indexed_items: "{{ security_groups.stdout_lines }}"
```

9. コマンドラインで、作成した Playbook を実行します。

```
$ ansible-playbook -i inventory.yaml  \
  down-bootstrap.yaml      \
  down-control-plane.yaml  \
  down-compute-nodes.yaml  \
  down-load-balancers.yaml \
  down-network.yaml        \
  down-security-groups.yaml
```

10. OpenShift Container Platform インストールに対して加えた DNS レコードの変更を削除します。

OpenShift Container Platform はお使いのインフラストラクチャーから削除されます。

第9章 RHV へのインストール

9.1. RHV へのクラスターのクイックインストール

警告

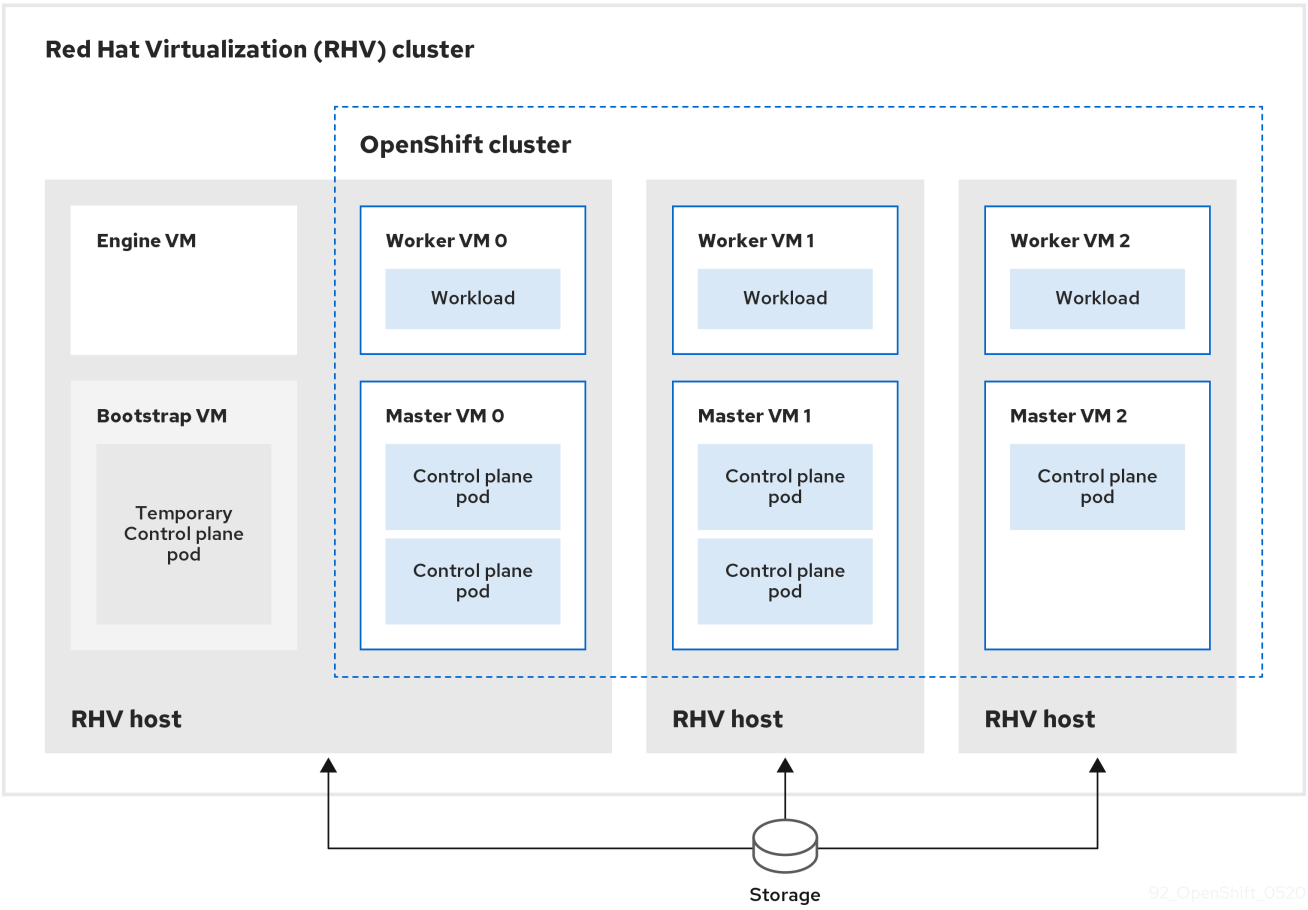
既知の問題により、このデフォルトのインストール手順は Red Hat Virtualization (RHV) 4.4.1 の OpenShift Container Platform バージョン 4.4 および 4.5 では機能しません。この不具合は RHV 4.4.2 で修正されました。

代わりに、カスタマイズを使用した RHV へのクラスターのインストール の手順を実行します。

警告

OpenShift Container Platform バージョン 4.6 を Red Hat Virtualization (RHV) にインストールするには、RHV バージョン 4.4 が必要です。RHV 4.3 で以前のバージョンの OpenShift Container Platform を実行している場合は、これを OpenShift Container Platform バージョン 4.6 に更新しないでください。Red Hat は、RHV バージョン 4.3 での OpenShift Container Platform バージョン 4.6 の実行をテストしていないため、この組み合わせをサポートしません。詳細は、OpenShift Container Platform 4.x Tested Integrations (x86_x64) を参照してください。

以下の図に示されるように、デフォルトの、カスタマイズされていない OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) クラスターにすばやくインストールできます。



インストールプログラムは、インストーラーでプロビジョニングされるインフラストラクチャーを使用してクラスターの作成およびデプロイを自動化します。

デフォルトのクラスターをインストールするには、環境を準備し、インストールプログラムを実行してプロンプトに応答します。次に、インストールプログラムは OpenShift Container Platform クラスターを作成します。

デフォルトクラスターの代替インストール方法については、カスタマイズによるクラスターのインストール について参照してください。

注記

このインストールプログラムは、Linux および macOS でのみ利用できます。

9.1.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 します。

9.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供す

るためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

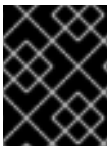
クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

9.1.3. RHV 環境の要件

OpenShift Container Platform クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。これらの要件を満たさないと、エラーが発生する可能性があります。

CPU、メモリー、ストレージについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。

デフォルトでは、インストールプログラムは、1 つのブートストラップマシンを含む 7 つのマシンをインストールプロセスで作成します。インストールプログラムが完了すると、ブートストラップマシンが削除され、そのリソースが解放されます。カスタムインストールを実行する場合は、インストールプログラムが作成する仮想マシンの数を増やすことができます。



重要

install_config.yaml ファイルで仮想マシンまたはリソースの数を増やす場合は、これらの要件も増やす必要があります。

要件

- RHV バージョン 4.3.10 以降。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。

- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
  - 最小 28 vCPU (インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU)。
  - 以下を含む 112 GiB 以上の RAM。
    - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
    - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
    - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。
- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- 実稼働環境では各仮想マシンに 120 GiB 以上を指定する必要があるため、ストレージドメインには OpenShift Container Platform クラスター用に 840 GiB 以上が必要になります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインには OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- RHV クラスターのインターネット接続へのアクセス。これは、インストールおよび更新時に Red Hat Ecosystem Catalog からイメージをダウンロードし、Telemetry サービスでサブスクリプションとエンタイトルメントのプロセスを単純化できるようにするために必要です。
- RHV クラスターに RHV Manager の REST API にアクセスできる仮想ネットワークがあること。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。

注記

- すべてのホストには、それらの操作および OpenShift Container Platform 以外の操作に使用するもの**のほかに**、必要となるメモリーおよび CPU リソースがなければなりません。
- OpenShift Container Platform と RHV のリリースサイクルは異なり、テストされるバージョンは両方の製品のリリース日によって変わる可能性があります。
- ブートストラップマシンは、インストールプログラムが OpenShift Container Platform クラスターを作成する間に一時的なコントロールプレーンを提供します。クラスターの作成後、インストールプログラムはブートストラップマシンを削除し、そのリソースを解放します。

9.1.4. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。

重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンを確認します。
   a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
   b. 開いているウィンドウで、**RHV Software Version** が **4.3.10** 以上であることを確認します。
2. データセンター、クラスター、およびストレージを検査します。
   a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
   b. OpenShift Container Platform をインストールする予定のデータセンターに緑色の上矢印 (Up) が表示されることを確認します。
   c. そのデータセンターの名前をクリックします。
   d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。
   e. 後で使用できるように **ドメイン名** を記録します。
   f. **空き領域** に 230 GiB 以上あることを確認します。
   g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定 できます。
   h. データセンターの詳細で、**Clusters** タブをクリックします。
   i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。
3. RHV ホストリソースを確認します。
   a. RHV 管理ポータルで、**Compute > Clusters** をクリックします。
   b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。
   c. クラスターの詳細で、**Hosts** タブをクリックします。
   d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。
   e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。
   f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API で curl コマンドを使用します。以下の形式を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、管理者パスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u rhvadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

9.1.5. RHV でのネットワーク環境の準備

OpenShift Container Platform クラスターの 3 つの静的 IP アドレスを設定し、これらのアドレスの 2 つを使用して DNS エントリーを作成します。

手順

1. 静的 IP アドレスを予約します。

a. OpenShift Container Platform をインストールするネットワークで、DHCP リースプール外にある 3 つの静的 IP アドレスを特定します。

b. このネットワーク上のホストに接続し、それぞれの IP アドレスが使用されていないことを確認します。たとえば、Address Resolution Protocol (ARP) を使用して、IP アドレスのいずれにもエントリーがないことを確認します。

```
$ arp 10.35.1.19
```

出力例

```
10.35.1.19 (10.35.1.19) -- no entry
```

c. ネットワーク環境の標準的な方法に従って、3 つの静的 IP アドレスを予約します。

d. 今後の参照用にこれらの IP アドレスを記録します。

2. 以下の形式を使用して、OpenShift Container Platform REST API およびアプリケーションドメイン名の DNS エントリーを作成します。

```
api.<cluster-name>.<base-domain>  <ip-address> 1
*.apps.<cluster-name>.<base-domain>  <ip-address> 2
```

1 **<cluster-name>**、**<base-domain>**、および **<ip-address>** には、クラスター名、ベースドメイン、および OpenShift Container Platform API の静的 IP アドレスを指定します。

2 Ingress およびロードバランサー用に OpenShift Container Platform アプリケーションのクラスター名、ベースドメイン、および静的 IP アドレスを指定します。

以下は例になります。

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

注記

3 つ目の静的 IP アドレスには DNS エントリーは必要ありません。OpenShift Container Platform クラスターは、その内部 DNS サービスにこのアドレスを使用します。

9.1.6. RHV 用の CA 証明書の設定

Red Hat Virtualization (RHV) Manager から CA 証明書をダウンロードし、インストールマシンにこれを設定します。

RHV Manager からの Web サイトまたは **curl** コマンドを使用して、証明書をダウンロードできます。

その後、インストールプログラムに証明書を提供します。

手順

1. 以下の 2 つの方法のいずれかを使用して CA 証明書をダウンロードします。

   - Manager の Web ページ (**https://<engine-fqdn>/ovirt-engine/**) に移動します。次に、**Downloads** で **CA Certificate** のリンクをクリックします。

   - 以下のコマンドを実行します。

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem 1
```

1 **<engine-fqdn>** には、RHV Manager の完全修飾ドメイン名 (例: **rhv-env.virtlab.example.com**) を指定します。

2. ルートレスユーザーに Manager へのアクセスを付与するように CA ファイルを設定します。CA ファイルのパーミッションを 8 進数の **0644** に設定します (シンボリック値: **-rw-r—r--**):

```
$ sudo chmod 0644 /tmp/ca.pem
```

3. Linux の場合は、サーバー証明書のディレクトリーに CA 証明書をコピーします。**-p** を使用してパーミッションを保存します。

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

4. オペレーティングシステム用の証明書マネージャーに証明書を追加します。
   - MacOS の場合は、証明書ファイルをダブルクリックして、**Keychain Access** ユーティリティーを使用してファイルを **System** キーチェーンに追加します。
   - Linux の場合は、CA 信頼を更新します。

```
$ sudo update-ca-trust
```

注記

独自の認証局を使用する場合は、システムがこれを信頼することを確認します。

関連情報

詳細は、RHV ドキュメントの Authentication and Security を参照してください。

9.1.7. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

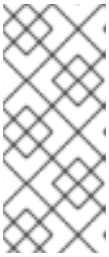
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

9.1.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

9.1.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- インストーラーを実行するマシンから、**ovirt-imageio** ポートを Engine に開きます。デフォルトでは、ポートは **54322** です。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

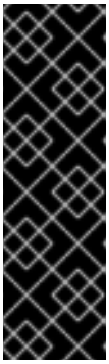
手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

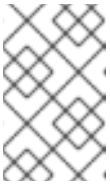


重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

インストールプログラムのプロンプトに対応します。

a. オプション: **SSH Public Key** には、 パスワードなしのパブリックキー (例: **~/.ssh/id_rsa.pub**) を選択します。このキーは、新規 OpenShift Container Platform クラスターとの接続を認証します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターには、**ssh-agent** プロセスが使用する SSH キーを選択します。

b. **Platform** には、**ovirt** を選択します。

c. **Enter oVirt's API endpoint URL** に、この形式を使用して RHV API の URL を入力します。

```
https://<engine-fqdn>/ovirt-engine/api (1)
```

(1) **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ovirtadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

d. **Is the oVirt CA trusted?** には、CA 証明書がすでに設定されているため **Yes** を入力します。そうでない場合は、**No** と入力します。

e. **oVirt's CA bundle** には、前の質問で **Yes** を入力している場合には、 **/etc/pki/ca-trust/source/anchors/ca.pem** の内容をコピーし、ここに貼り付けます。その後、**Enter** を 2 回押します。そうでない場合、つまり、前の質問で **No** と入力している場合は、この質問は表示されません。

f. **oVirt engine username** には、この形式を使用して RHV 管理者のユーザー名およびプロファイルを入力します。

```
<username>@<profile> (1)
```

(1) **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。ユーザー名とプロファイルは以下のようになります。

```
admin@internal
```

g. **oVirt engine password** に、RHV 管理者パスワードを入力します。

h. **oVirt cluster** には、OpenShift Container Platform をインストールするためのクラスターを選択します。

i. **oVirt storage domain** には、OpenShift Container Platform をインストールするためのストレージドメインを選択します。

j. **oVirt network** には、RHV Manager REST API へのアクセスのある仮想ネットワークを選択します。

k. **Internal API Virtual IP** に、クラスターの REST API とは別の静的 IP アドレスを入力します。

l. **Internal DNS Virtual IP** に、クラスターの内部 DNS サービスとは別の静的 IP アドレスを入力します。

m. **Ingress virtual IP** に、ワイルドカードアプリドメイン用に予約した静的 IP アドレスを入力します。

n. **Base Domain** に、OpenShift Container Platform クラスターのベースドメインを入力します。このクラスターが外部に公開される場合、これは DNS インフラストラクチャーが認識

する有効なドメインである必要があります。たとえば、**virtlab.example.com** を入力します。

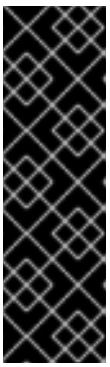
o. **Cluster Name** に、クラスターの名前を入力します。例: **my-cluster**OpenShift Container Platform REST API およびアプリケーションドメイン名向けに作成した外部登録/解決可能な DNS エントリーのクラスター名を使用します。インストールプログラムは、この名前を RHV 環境のクラスターにも指定します。

p. **Pull secret** には、先にダウンロードした **pull-secret.txt** ファイルからプルシークレットをコピーし、ここに貼り付けます。Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから同じプルシークレットのコピーを取得することもできます。

注記

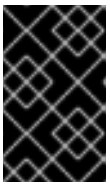
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。

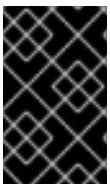
- **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

重要

クラスターのインストールに必要な手順を完了している必要があります。残りの手順では、クラスターを検証し、インストールのトラブルシューティングを行う方法を説明します。

第10章 バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

10.1. LINUX への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

10.2. WINDOWS での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。

4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
   PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

10.3. MACOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

詳細は、Getting started with the CLI を参照してください。

第11章 クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

11.1. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

トラブルシューティング

インストールが失敗すると、インストールプログラムがタイムアウトし、エラーメッセージが表示されます。詳細は、インストールに関する問題のトラブルシューティング を参照してください。

11.2. RHV での OPENSHIFT CONTAINER PLATFORM WEB コンソールへのアクセス

OpenShift Container Platform クラスターの初期化後に、OpenShift Container Platform Web コンソールにログインできます。

手順

1. オプション: Red Hat Virtualization (RHV) Administration Portal で、**Compute → Cluster** を開きます。
2. インストールプログラムが仮想マシンを作成することを確認します。
3. インストールプログラムが実行されているコマンドラインに戻ります。インストールプログラムが完了すると、OpenShift Container Platform Web コンソールにログインするためのユーザー名およびパスワードの一時パスワードが表示されます。
4. ブラウザーから OpenShift Container Platform の Web コンソールの URL を開きます。URL は以下の形式を使用します。

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>.<basedomain>** に、クラスター名およびベースドメインを指定します。

以下に例を示します。

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

11.3. RED HAT VIRTUALIZATION (RHV) へのインストールに関するよくある問題のトラブルシューティング

以下に、一般的な問題およびそれらについて考えられる原因および解決策を記載します。

11.3.1. CPU 負荷が増大し、ノードが Not Ready 状態になる

- **現象**: CPU 負荷が大幅に増大し、ノードが **Not Ready** 状態に切り替わり始める。
- **原因**: ストレージドメインのレイテンシーが高すぎる可能性があります (特にマスターノードの場合)。
- **解決策**:
  Kubelet サービスを再起動して、ノードを再度 Ready 状態にします。以下を入力します。

```
$ systemctl restart kubelet
```

  OpenShift Container Platform メトリクスサービスを検査します。これは、etcd ディスクの同期期間などの有用なデータを収集し、これについて報告します。クラスターが機能している場合は、このデータを使用して、ストレージのレイテンシーまたはスループットが根本的な問題かどうかを判断します。その場合、レイテンシーが短く、スループットの高いストレージリソースの使用を検討してください。

  未加工メトリクスを取得するには、kubeadmin または cluster-admin 権限を持つユーザーで以下のコマンドを実行します。

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics`
```

  詳細は、Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x を参照してください。

11.3.2. OpenShift Container Platform クラスター API に接続できない

- **現象**: インストールプログラムは完了するが、OpenShift Container Platform クラスター API は利用できない。ブートストラップの仮想マシンは、ブートストラッププロセスの完了後も起動した状態になります。以下のコマンドを入力すると、応答がタイムアウトします。

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- **原因**: ブートストラップ仮想マシンがインストールプログラムによって削除されず、クラスターの API IP アドレスをリリースしない。
- **解決策**: **wait-for** サブコマンドを使用して、ブートストラッププロセスの完了時に通知を受信する。

```
$ ./openshift-install wait-for bootstrap-complete
```

  ブートストラッププロセスが完了したら、ブートストラップ仮想マシンを削除します。

```
$ ./openshift-install destroy bootstrap
```

11.4. インストール後のタスク

OpenShift Container Platform クラスターの初期化後に、以下のタスクを実行できます。

- オプション: デプロイメント後に、OpenShift Container Platform で Machine Config Operator (MCO) を使用して SSH キーを追加するか、または置き換えます。
- オプション: **kubeadmin** ユーザーを削除します。代わりに、認証プロバイダーを使用して cluster-admin 権限を持つユーザーを作成します。

11.5. カスタマイズによる RHV へのクラスターのインストール

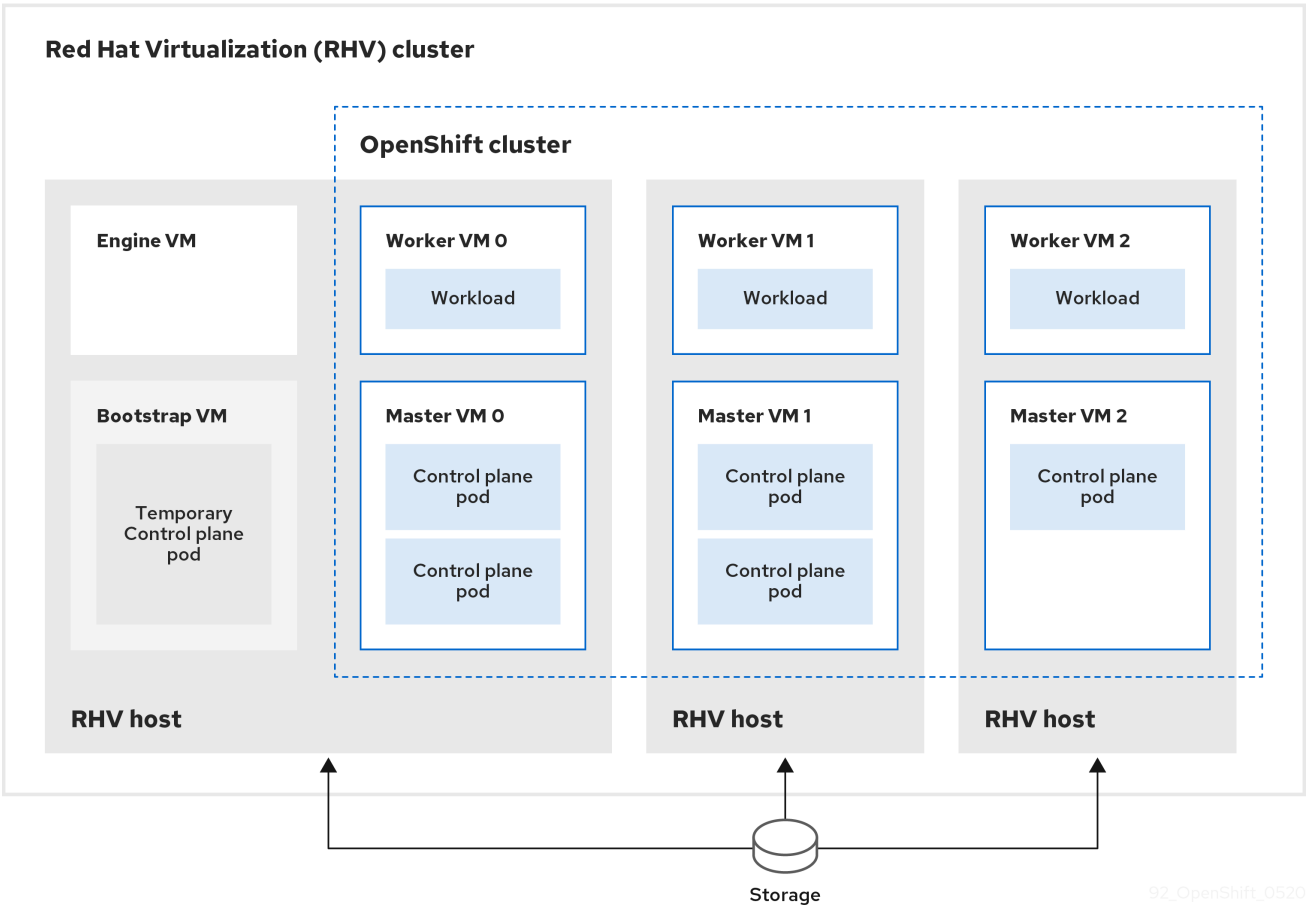
警告

Red Hat Virtualization (RHV) 4.4.1 に OpenShift Container Platform バージョン 4.4 および 4.5 をインストールする際の既知の問題により、OpenShift IPI installation on RHV-4.x failed with "Error: timeout while waiting for state to become 'up' (last state: 'down', timeout: 10m0s)" で説明されているように **install-config.yaml** をカスタマイズする必要があります。この不具合は RHV 4.4.2 で修正されました。

警告

OpenShift Container Platform バージョン 4.6 を Red Hat Virtualization (RHV) にインストールするには、RHV バージョン 4.4 が必要です。RHV 4.3 で以前のバージョンの OpenShift Container Platform を実行している場合は、これを OpenShift Container Platform バージョン 4.6 に更新しないでください。Red Hat は、RHV バージョン 4.3 での OpenShift Container Platform バージョン 4.6 の実行をテストしていないため、この組み合わせをサポートしません。詳細は、OpenShift Container Platform 4.x Tested Integrations (x86_x64) を参照してください。

以下の図に示されるように、OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) でカスタマイズし、インストールすることができます。



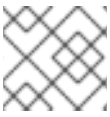
インストールプログラムは、インストーラーでプロビジョニングされるインフラストラクチャーを使用してクラスターの作成およびデプロイを自動化します。

カスタマイズされたクラスターをインストールするには、環境を準備し、以下の手順を実行します。

1. インストールプログラムを実行し、そのプロンプトに応答して、インストール設定ファイル **install-config.yaml** ファイルを作成します。
2. **install-config.yaml** ファイルでパラメーターを検査し、変更します。
3. **install-config.yaml** ファイルの作業用コピーを作成します。
4. **install-config.yaml** ファイルのコピーを使ってインストールプログラムを実行します。

次に、インストールプログラムは OpenShift Container Platform クラスターを作成します。

カスタマイズされたクラスターをインストールする代替方法については、デフォルトのクラスターのインストール を参照してください。



注記

このインストールプログラムは、Linux および macOS でのみ利用できます。

11.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。

- ファイアウォールを使用する場合、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 します。

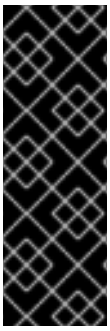
11.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

11.5.3. RHV 環境の要件

OpenShift Container Platform クラスターをインストールし、実行するには、RHV 環境が以下の要件を満たしている必要があります。これらの要件を満たさないと、エラーが発生する可能性があります。

CPU、メモリー、ストレージについての以下の要件は、インストールプログラムが作成する仮想マシンのデフォルト数で乗算した **デフォルト** 値に基づいています。

デフォルトでは、インストールプログラムは、1 つのブートストラップマシンを含む 7 つのマシンをインストールプロセスで作成します。インストールプログラムが完了すると、ブートストラップマシンが削除され、そのリソースが解放されます。カスタムインストールを実行する場合は、インストールプログラムが作成する仮想マシンの数を増やすことができます。

重要

install_config.yaml ファイルで仮想マシンまたはリソースの数を増やす場合は、これらの要件も増やす必要があります。

要件

- RHV バージョン 4.3.10 以降。
- RHV 環境に **Up** 状態のデータセンターが 1 つあること。
- RHV データセンターに RHV クラスターが含まれていること。
- RHV クラスターに OpenShift Container Platform クラスター専用の以下のリソースがあること。
  - 最小 28 vCPU (インストール時に作成される 7 仮想マシンのそれぞれに 4 vCPU)。
  - 以下を含む 112 GiB 以上の RAM。
    - 一時的なコントロールプレーンを提供するブートストラップマシン用に 16 GiB 以上。
    - コントロールプレーンを提供する 3 つのコントロールプレーンマシンのそれぞれに 16 GiB 以上。
    - アプリケーションワークロードを実行する 3 つのコンピュートマシンのそれぞれに 16 GiB 以上。
- RHV ストレージドメインは、これらの etcd バックエンドのパフォーマンス要件 を満たす必要があります。
- 実稼働環境では各仮想マシンに 120 GiB 以上を指定する必要があるため、ストレージドメインには OpenShift Container Platform クラスター用に 840 GiB 以上が必要になります。リソースに制約のある環境または非実稼働環境では、各仮想マシンに 32 GiB 以上を指定する必要があるため、ストレージドメインには OpenShift Container Platform クラスター用に 230 GiB 以上が必要になります。
- RHV クラスターのインターネット接続へのアクセス。これは、インストールおよび更新時に Red Hat Ecosystem Catalog からイメージをダウンロードし、Telemetry サービスでサブスクリプションとエンタイトルメントのプロセスを単純化できるようにするために必要です。
- RHV クラスターに RHV Manager の REST API にアクセスできる仮想ネットワークがあること。インストーラーが作成する仮想マシンが DHCP を使用して IP アドレスを取得するため、DHCP がこのネットワークで有効にされていることを確認します。

注記

- すべてのホストには、それらの操作および OpenShift Container Platform 以外の操作に使用するもの**のほかに**、必要となるメモリーおよび CPU リソースがなければなりません。
- OpenShift Container Platform と RHV のリリースサイクルは異なり、テストされるバージョンは両方の製品のリリース日によって変わる可能性があります。
- ブートストラップマシンは、インストールプログラムが OpenShift Container Platform クラスターを作成する間に一時的なコントロールプレーンを提供します。クラスターの作成後、インストールプログラムはブートストラップマシンを削除し、そのリソースを解放します。

11.5.4. RHV 環境の要件の確認

RHV 環境が OpenShift Container Platform クラスターをインストールし、実行するための要件を満たしていることを確認します。これらの要件を満たさないと、エラーが発生する可能性があります。

重要

これらの要件は、インストールプログラムがコントロールプレーンおよびコンピュートマシンの作成に使用するデフォルトのリソースに基づいています。これらのリソースには、vCPU、メモリー、およびストレージが含まれます。これらのリソースを変更するか、または OpenShift Container Platform マシンの数を増やす場合は、これらの要件を適宜調整します。

手順

1. RHV のバージョンを確認します。
    a. RHV Administration Portal の右上にある ? ヘルプアイコンをクリックし、**About** を選択します。
    b. 開いているウィンドウで、**RHV Software Version** が **4.3.10** 以上であることを確認します。
2. データセンター、クラスター、およびストレージを検査します。
    a. RHV 管理ポータルで、**Compute → Data Centers** をクリックします。
    b. OpenShift Container Platform をインストールする予定のデータセンターに緑色の上矢印 (Up) が表示されることを確認します。
    c. そのデータセンターの名前をクリックします。
    d. データセンターの詳細の **Storage** タブで、OpenShift Container Platform をインストールする予定のストレージドメインが **Active** であることを確認します。
    e. 後で使用できるように **ドメイン名** を記録します。
    f. **空き領域** に 230 GiB 以上あることを確認します。
    g. ストレージドメインが これらの etcd バックエンドのパフォーマンス要件 を満たしていることを確認します。これは、fio パフォーマンスベンチマークツールを使用して測定 できます。
    h. データセンターの詳細で、**Clusters** タブをクリックします。
    i. OpenShift Container Platform をインストールする予定の RHV クラスターを見つけます。後で使用できるようにクラスター名を記録します。
3. RHV ホストリソースを確認します。
    a. RHV 管理ポータルで、**Compute > Clusters** をクリックします。
    b. OpenShift Container Platform をインストールする予定のクラスターをクリックします。
    c. クラスターの詳細で、**Hosts** タブをクリックします。
    d. ホストを検査し、それらに OpenShift Container Platform クラスター **専用** として利用可能な **論理 CPU コア** の合計が 28 つ以上であることを確認します。

e. 後で使用できるように、利用可能な **論理 CPU コア** の数を記録します。

f. これらの CPU コアが分散され、インストール時に作成された 7 つの仮想マシンのそれぞれに 4 つのコアを持たせることができることを確認します。

g. ホストには、以下の OpenShift Container Platform マシンのそれぞれの要件を満たすように **新規仮想マシンをスケジュールするための最大空きメモリー** として 112 GiB があることを確認します。

- ブートストラップマシンに 16 GiB が必要です。
- 3 つのコントロールプレーンマシンのそれぞれに 16 GiB が必要です。
- 3 つのコンピュートマシンのそれぞれに 16 GiB が必要です。

h. 後で使用できるように **新規仮想マシンをスケジュールするための最大空きメモリー** の量を記録します。

4. OpenShift Container Platform をインストールするための仮想ネットワークが RHV Manager の REST API にアクセスできることを確認します。このネットワーク上の仮想マシンから、RHV Manager の REST API で curl コマンドを使用します。以下の形式を使用します。

```
$ curl -k -u <username>@<profile>:<password> \ 1
https://<engine-fqdn>/ovirt-engine/api 2
```

1 **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、 **Profile** ドロップダウンリストで確認できます。**<password>** に、管理者パスワードを指定します。

2 **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u rhvadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

11.5.5. RHV でのネットワーク環境の準備

OpenShift Container Platform クラスターの 3 つの静的 IP アドレスを設定し、これらのアドレスの 2 つを使用して DNS エントリーを作成します。

手順

1. 静的 IP アドレスを予約します。

a. OpenShift Container Platform をインストールするネットワークで、DHCP リースプール外にある 3 つの静的 IP アドレスを特定します。

b. このネットワーク上のホストに接続し、それぞれの IP アドレスが使用されていないことを確認します。たとえば、Address Resolution Protocol (ARP) を使用して、IP アドレスのいずれにもエントリーがないことを確認します。

```
$ arp 10.35.1.19
```

出力例

```
10.35.1.19 (10.35.1.19) -- no entry
```

c. ネットワーク環境の標準的な方法に従って、3 つの静的 IP アドレスを予約します。

d. 今後の参照用にこれらの IP アドレスを記録します。

2. 以下の形式を使用して、OpenShift Container Platform REST API およびアプリケーションドメイン名の DNS エントリーを作成します。

```
api.<cluster-name>.<base-domain>  <ip-address> 1
*.apps.<cluster-name>.<base-domain>  <ip-address> 2
```

1 **<cluster-name>**、**<base-domain>**、および **<ip-address>** には、クラスター名、ベースドメイン、および OpenShift Container Platform API の静的 IP アドレスを指定します。

2 Ingress およびロードバランサー用に OpenShift Container Platform アプリケーションのクラスター名、ベースドメイン、および静的 IP アドレスを指定します。

以下は例になります。

```
api.my-cluster.virtlab.example.com 10.35.1.19
*.apps.my-cluster.virtlab.example.com 10.35.1.20
```

注記

3 つ目の静的 IP アドレスには DNS エントリーは必要ありません。OpenShift Container Platform クラスターは、その内部 DNS サービスにこのアドレスを使用します。

11.5.6. RHV 用の CA 証明書の設定

Red Hat Virtualization (RHV) Manager から CA 証明書をダウンロードし、インストールマシンにこれを設定します。

RHV Manager からの Web サイトまたは **curl** コマンドを使用して、証明書をダウンロードできます。

その後、インストールプログラムに証明書を提供します。

手順

1. 以下の 2 つの方法のいずれかを使用して CA 証明書をダウンロードします。

- Manager の Web ページ (**https://<engine-fqdn>/ovirt-engine/**) に移動します。次に、**Downloads** で **CA Certificate** のリンクをクリックします。
- 以下のコマンドを実行します。

```
$ curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem 1
```

1 **<engine-fqdn>** には、RHV Manager の完全修飾ドメイン名 (例: **rhv-**

env.virtlab.example.com) を指定します。

2. ルートレスユーザーに Manager へのアクセスを付与するように CA ファイルを設定します。CA ファイルのパーミッションを 8 進数の **0644** に設定します (シンボリック値: **-rw-r—r--**):

```
$ sudo chmod 0644 /tmp/ca.pem
```

3. Linux の場合は、サーバー証明書のディレクトリーに CA 証明書をコピーします。**-p** を使用してパーミッションを保存します。

```
$ sudo cp -p /tmp/ca.pem /etc/pki/ca-trust/source/anchors/ca.pem
```

4. オペレーティングシステム用の証明書マネージャーに証明書を追加します。
   - MacOS の場合は、証明書ファイルをダブルクリックして、**Keychain Access** ユーティリティーを使用してファイルを **System** キーチェーンに追加します。
   - Linux の場合は、CA 信頼を更新します。

```
$ sudo update-ca-trust
```

注記

独自の認証局を使用する場合は、システムがこれを信頼することを確認します。

関連情報

詳細は、RHV ドキュメントの Authentication and Security を参照してください。

11.5.7. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

11.5.8. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

11.5.9. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。Red Hat Virtualization (RHV)

警告

Red Hat Virtualization (RHV) 4.4.1 に OpenShift Container Platform バージョン 4.4 および 4.5 をインストールする際の既知の問題により、OpenShift IPI installation on RHV-4.x failed with "Error: timeout while waiting for state to become 'up' (last state: 'down', timeout: 10m0s)" で説明されているように **install-config.yaml** をカスタマイズする必要があります。この不具合は RHV 4.4.2 で修正されました。

警告

OpenShift Container Platform (OCP) バージョン 4.6 を Red Hat Virtualization (RHV) にインストールするには、RHV バージョン 4.4 が必要です。RHV 4.3 で以前のバージョンの OCP を実行している場合は、これを OCP バージョン 4.6 に更新しないでください。Red Hat は、RHV バージョン 4.3 での OCP バージョン 4.6 の実行をテストしていないため、この組み合わせをサポートしません。OpenShift Container Platform 4.x Tested Integrations (x86_x64) も参照してください。

前提条件

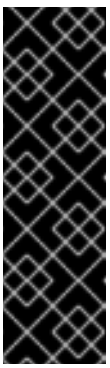
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   i. ターゲットに設定するプラットフォームとして **gcp** を選択します。

ii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iii. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

iv. クラスターをデプロイするリージョンを選択します。

v. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

b. インストールプログラムのプロンプトに対応します。

i. **SSH Public Key** では、パスワードなしのパブリックキー (例: **~/.ssh/id_rsa.pub**) を選択します。このキーは、新規 OpenShift Container Platform クラスターとの接続を認証します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターには、**ssh-agent** プロセスが使用する SSH キーを選択します。

ii. **Platform** には、**ovirt** を選択します。

iii. **Enter oVirt's API endpoint URL** に、この形式を使用して RHV API の URL を入力します。

```
https://<engine-fqdn>/ovirt-engine/api (1)
```

(1) **<engine-fqdn>** に、RHV 環境の完全修飾ドメイン名を指定します。

以下に例を示します。

```
$ curl -k -u ovirtadmin@internal:pw123 \
https://rhv-env.virtlab.example.com/ovirt-engine/api
```

iv. **Is the oVirt CA trusted?** には、CA 証明書がすでに設定されているため **Yes** を入力します。そうでない場合は、**No** と入力します。

v. **oVirt's CA bundle** には、前の質問で **Yes** を入力している場合には、**/etc/pki/ca-trust/source/anchors/ca.pem** の内容をコピーし、ここに貼り付けます。その後、**Enter** を 2 回押します。そうでない場合、つまり、前の質問で **No** と入力している場合は、この質問は表示されません。

vi. **oVirt engine username** には、この形式を使用して RHV 管理者のユーザー名およびプロファイルを入力します。

```
<username>@<profile> (1)
```

(1) **<username>** に、RHV 管理者のユーザー名を指定します。**<profile>** には、ログインプロファイルを指定します。ログインプロファイルは、RHV Administration Portal ログインページに移動し、**Profile** ドロップダウンリストで確認できます。

ユーザー名とプロファイルは以下のようになります。

```
admin@internal
```

vii. **oVirt engine password** に、RHV 管理者パスワードを入力します。

viii. **oVirt cluster** には、OpenShift Container Platform をインストールするためのクラスターを選択します。

ix. **oVirt storage domain** には、OpenShift Container Platform をインストールするためのストレージドメインを選択します。

x. **oVirt network** には、RHV Manager REST API へのアクセスのある仮想ネットワークを選択します。

xi. **Internal API Virtual IP** に、クラスターの REST API とは別の静的 IP アドレスを入力します。

xii. **Internal DNS Virtual IP** に、クラスターの内部 DNS サービスとは別の静的 IP アドレスを入力します。

xiii. **Ingress virtual IP** に、ワイルドカードアプリドメイン用に予約した静的 IP アドレスを入力します。

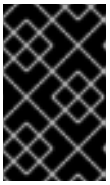
xiv. **Base Domain** に、OpenShift Container Platform クラスターのベースドメインを入力します。このクラスターが外部に公開される場合、これは DNS インフラストラクチャーが認識する有効なドメインである必要があります。たとえば、**virtlab.example.com** を入力します。

xv. **Cluster Name** に、クラスターの名前を入力します。例: **my-cluster**OpenShift Container Platform REST API およびアプリケーションドメイン名向けに作成した外部登録/解決可能な DNS エントリーのクラスター名を使用します。インストールプログラムは、この名前を RHV 環境のクラスターにも指定します。

xvi. **Pull secret** には、先にダウンロードした **pull-secret.txt** ファイルからプルシークレットをコピーし、ここに貼り付けます。Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから同じプルシークレットのコピーを取得することもできます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。



重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

11.5.9.1. Red Hat Virtualization (RHV) のサンプル install-config.yaml ファイル

install-config.yaml ファイルのパラメーターおよびパラメーター値を変更して、インストールプログラムが作成する OpenShift Container Platform クラスターをカスタマイズできます。

以下の例は、RHV への OpenShift Container Platform のインストールに固有の例です。

このファイルは、以下のコマンドを実行する際に指定する **<installation_directory>** にあります。

```
$ ./openshift-install create install-config --dir=<installation_directory>
```

注記

- これらのサンプルファイルは参照用にのみ提供されます。インストールプログラムを使用して **install-config.yaml** ファイルを取得する必要があります。
- **install-config.yaml** ファイルを変更すると、クラスターに必要なリソースを増やすことができます。RHV 環境にそれらの追加リソースがあることを確認します。これらがない場合は、インストールまたはクラスターが失敗します。

例: これはデフォルトの install-config.yaml ファイルです。

```
apiVersion: v1
baseDomain: example.com
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: my-cluster
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 192.168.1.5
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
publish: External
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

例: 最小の install-config.yaml ファイル

```
apiVersion: v1
baseDomain: example.com
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed12345 AAAA...
```

例: install-config.yaml ファイルのカスタムマシンプール

```
apiVersion: v1
baseDomain: example.com
controlPlane:
  name: master
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 2
      memoryMB: 65536
      osDisk:
        sizeGB: 100
      vmType: high_performance
  replicas: 3
compute:
- name: worker
  platform:
    ovirt:
      cpu:
        cores: 4
        sockets: 4
      memoryMB: 65536
      osDisk:
        sizeGB: 200
      vmType: high_performance
  replicas: 5
metadata:
  name: test-cluster
platform:
  ovirt:
    api_vip: 10.46.8.230
    ingress_vip: 10.46.8.232
    ovirt_cluster_id: 68833f9f-e89c-4891-b768-e2ba0815b76b
    ovirt_storage_domain_id: ed7b0f4e-0e96-492a-8fff-279213ee1468
    ovirt_network_name: ovirtmgmt
```

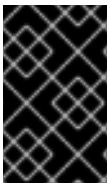
```
    vnicProfileID: 3fa86930-0be5-4052-b667-b79f0a729692
pullSecret: '{"auths": ...}'
sshKey: ssh-ed25519 AAAA...
```

11.5.9.2. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

11.5.9.2.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表11.1 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。

パラメーター	説明	値
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	{
		"auths":{
		"cloud.openshift.com":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		},
		"quay.io":{
		"auth":"b3Blb=",
		"email":"you@example.com"
		}
		}
		}

11.5.9.2.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表11.2 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  clusterNetwork:`
		`  - cidr: 10.128.0.0/14`
		`    hostPrefix: 23`

networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

パラメーター	説明	値
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

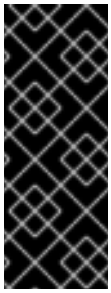
11.5.9.2.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表11.3 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列

パラメーター	説明	値
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

11.5.9.2.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表11.4 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

11.5.9.2.5. 追加の Red Hat Virtualization (RHV) 設定パラメーター

追加の RHV 設定パラメーターは以下の表で説明されています。

表11.5 クラスターの追加 RHV パラメーター

パラメーター	説明	値
platform.ovirt.ovirt_cluster_id	必須。仮想マシンが作成されるクラスター。	文字列。例: **68833f9f-e89c-4891-b768-e2ba0815b76b**
platform.ovirt.ovirt_storage_domain_id	必須。仮想マシンディスクが作成されるストレージドメイン ID。	文字列。例: **ed7b0f4e-0e96-492a-8fff-279213ee1468**
platform.ovirt.ovirt_network_name	必須。仮想マシン NIC が作成されるネットワーク名。	文字列。例: **ocpcluster**
platform.ovirt.vnicProfileID	必須。仮想マシンネットワークインターフェイスの vNIC プロファイル ID。これは、クラスターネットワークに単一のプロファイルがある場合に示唆されます。	文字列。例: **3fa86930-0be5-4052-b667-b79f0a729692**
platform.ovirt.api_vip	必須。API 仮想 IP (VIP) に割り当てられるマシンネットワークの IP アドレス。このエンドポイントで OpenShift API にアクセスできます。	文字列。例: **10.46.8.230**
platform.ovirt.ingress_vip	必須。Ingress 仮想 IP (VIP) に割り当てられるマシンネットワークの IP アドレス。	文字列。例: **10.46.8.232**

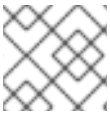
11.5.9.2.6. マシンプールの追加 RHV パラメーター

マシンプールの追加の RHV 設定パラメーターは以下の表で説明されています。

表11.6 マシンプールの追加 RHV パラメーター

パラメーター	説明	値
<machine-pool>.platform.ovirt.cpu	オプション。仮想マシンの CPU を定義します。	オブジェクト
<machine-pool>.platform.ovirt.cpu.cores	**<machine-pool>.platform.ovirt.cpu** を使用する場合に必須です。コア数。仮想 CPU (vCPU) の合計はコア * ソケットです。	整数
<machine-pool>.platform.ovirt.cpu.sockets	**<machine-pool>.platform.ovirt.cpu** を使用する場合に必須です。コアあたりのソケット数。仮想 CPU (vCPU) の合計はコア * ソケットです。	整数

パラメーター	説明	値
<machine-pool>.platform.ovirt.memoryMB	オプション。仮想マシンのメモリー (MiB 単位)。	整数
<machine-pool>.platform.ovirt.instanceTypeID	オプション。**00000009-0009-0009-0009-0000000000f1** などのインスタンスタイプ UUID。これは **https://<engine-fqdn>/ovirt-engine/api/instancetypes** エンドポイントから取得できます。	UUID の文字列
<machine-pool>.platform.ovirt.osDisk	オプション。仮想マシンの起動可能な初回の、および起動可能なディスクを定義します。	文字列
<machine-pool>.platform.ovirt.osDisk.sizeGB	**<machine-pool>.platform.ovirt.osDisk** を使用する場合に必須です。ディスクのサイズ (GiB 単位)。	数字
<machine-pool>.platform.ovirt.vmType	オプション。**high-performance**、**server**、 または **desktop** などの仮想マシンワークロードタイプ。	文字列



注記

<machine-pool> を **controlPlane** または **compute** に置き換えることができます。

11.5.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。

重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- インストーラーを実行するマシンから、**ovirt-imageio** ポートを Engine に開きます。デフォルトでは、ポートは **54322** です。
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1. **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。
2. 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。

- **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

重要

クラスターのインストールに必要な手順を完了している必要があります。残りの手順では、クラスターを検証し、インストールのトラブルシューティングを行う方法を説明します。

11.5.11. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

11.5.11.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

11.5.11.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

11.5.11.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

11.5.12. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig
```
(1)

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

詳細は、Getting started with the CLI を参照してください。

11.5.13. クラスターステータスの確認

インストール時またはインストール後に OpenShift Container Platform クラスターのステータスを確認することができます。

手順

1. クラスター環境で、管理者の kubeconfig ファイルをエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig
```
(1)

(1) **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

kubeconfig ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。

2. デプロイメント後に作成されたコントロールプレーンおよびコンピュートマシンを表示します。

```
$ oc get nodes
```

3. クラスターのバージョンを表示します。

```
$ oc get clusterversion
```

4. Operator のステータスを表示します。

```
$ oc get clusteroperator
```

5. クラスター内のすべての実行中の Pod を表示します。

```
$ oc get pods -A
```

トラブルシューティング

インストールが失敗すると、インストールプログラムがタイムアウトし、エラーメッセージが表示されます。詳細は、インストールに関する問題のトラブルシューティング を参照してください。

11.5.14. RHV での OpenShift Container Platform Web コンソールへのアクセス

OpenShift Container Platform クラスターの初期化後に、OpenShift Container Platform Web コンソールにログインできます。

手順

1. オプション: Red Hat Virtualization (RHV) Administration Portal で、**Compute → Cluster** を開きます。
2. インストールプログラムが仮想マシンを作成することを確認します。
3. インストールプログラムが実行されているコマンドラインに戻ります。インストールプログラムが完了すると、OpenShift Container Platform Web コンソールにログインするためのユーザー名およびパスワードの一時パスワードが表示されます。
4. ブラウザーから OpenShift Container Platform の Web コンソールの URL を開きます。URL は以下の形式を使用します。

```
console-openshift-console.apps.<clustername>.<basedomain> 1
```

1 **<clustername>.<basedomain>** に、クラスター名およびベースドメインを指定します。

以下に例を示します。

```
console-openshift-console.apps.my-cluster.virtlab.example.com
```

11.5.15. Red Hat Virtualization (RHV) へのインストールに関するよくある問題のトラブルシューティング

以下に、一般的な問題およびそれらについて考えられる原因および解決策を記載します。

11.5.15.1. CPU 負荷が増大し、ノードが Not Ready 状態になる

- **現象**: CPU 負荷が大幅に増大し、ノードが **Not Ready** 状態に切り替わり始める。
- **原因**: ストレージドメインのレイテンシーが高すぎる可能性があります (特にマスターノードの場合)。
- **解決策**:
  Kubelet サービスを再起動して、ノードを再度 Ready 状態にします。以下を入力します。

```
$ systemctl restart kubelet
```

OpenShift Container Platform メトリクスサービスを検査します。これは、etcd ディスクの同期期間などの有用なデータを収集し、これについて報告します。クラスターが機能している場合は、このデータを使用して、ストレージのレイテンシーまたはスループットが根本的な問題かどうかを判断します。その場合、レイテンシーが短く、スループットの高いストレージリソースの使用を検討してください。

未加工メトリクスを取得するには、kubeadmin または cluster-admin 権限を持つユーザーで以下のコマンドを実行します。

```
$ oc get --insecure-skip-tls-verify --server=https://localhost:<port> --raw=/metrics`
```

詳細は、Exploring Application Endpoints for the purposes of Debugging with OpenShift 4.x を参照してください。

11.5.15.2. OpenShift Container Platform クラスター API に接続できない

- **現象**: インストールプログラムは完了するが、OpenShift Container Platform クラスター API は利用できない。ブートストラップの仮想マシンは、ブートストラッププロセスの完了後も起動した状態になります。以下のコマンドを入力すると、応答がタイムアウトします。

```
$ oc login -u kubeadmin -p *** <apiurl>
```

- **原因**: ブートストラップ仮想マシンがインストールプログラムによって削除されず、クラスターの API IP アドレスをリリースしない。
- **解決策**: **wait-for** サブコマンドを使用して、ブートストラッププロセスの完了時に通知を受信する。

```
$ ./openshift-install wait-for bootstrap-complete
```

ブートストラッププロセスが完了したら、ブートストラップ仮想マシンを削除します。

```
$ ./openshift-install destroy bootstrap
```

11.5.16. インストール後のタスク

OpenShift Container Platform クラスターの初期化後に、以下のタスクを実行できます。

- オプション: デプロイメント後に、OpenShift Container Platform で Machine Config Operator (MCO) を使用して SSH キーを追加するか、または置き換えます。
- オプション: **kubeadmin** ユーザーを削除します。代わりに、認証プロバイダーを使用して cluster-admin 権限を持つユーザーを作成します。

11.5.17. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

11.6. RHV でのクラスターのアンインストール

OpenShift Container Platform クラスターを Red Hat Virtualization (RHV) から削除することができます。

11.6.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。

注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、または 削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

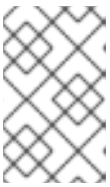
- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。
2. 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

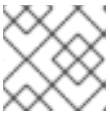
第12章 VSPHERE へのインストール

12.1. クラスターの VSPHERE へのインストール

OpenShift Container Platform バージョン 4.5 では、インストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。

12.1.1. 前提条件

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

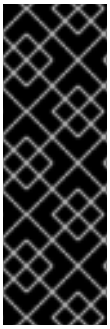
12.1.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

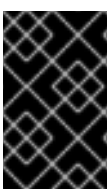
12.1.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表12.1 VMware コンポーネントのサポートされる vSphere の最小バージョン

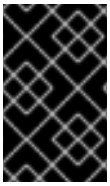
コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。



重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.1.4. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例12.1 インストールに必要なロールおよび特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	Always	**Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.Add NewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	Always	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone Folder.Create Folder.Delete**

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例12.2 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用

重要

OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。

Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスおよびホスト名をクラスターマシンに提供するように設定されていることを確認する必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード

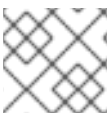
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.2 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

12.1.5. SSH プライベートキーの生成およびエージェントへの追加

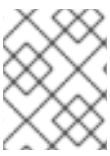
クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

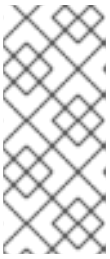
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.1.6. インストールプログラムの取得

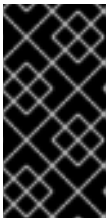
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.1.7. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。
2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

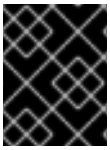
```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

12.1.8. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

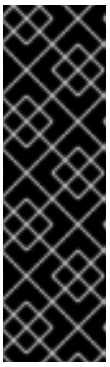
手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

プロンプト時に値を指定します。

a. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

b. ターゲットに設定するプラットフォームとして **gcp** を選択します。

c. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

d. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

e. クラスターをデプロイするリージョンを選択します。

f. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

g. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

h. vCenter インスタンスの名前を指定します。

i. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

j. 接続する vCenter インスタンスにあるデータセンターを選択します。

k. 使用するデフォルトの vCenter データストアを選択します。

l. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。

m. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

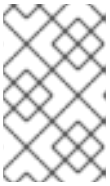
n. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

o. クラスター Ingress に設定した仮想 IP アドレスを入力します。

p. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

q. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。7 文字以上の名前を指定すると、クラスター名から生成されるインフラストラクチャー ID で最初の 6 文字のみが使用されます。

r. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。



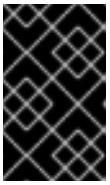
注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

12.1.9. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.1.9.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.1.9.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.1.9.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.1.10. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

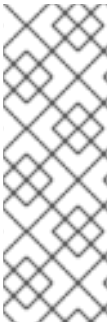
12.1.11. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

12.1.11.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.1.11.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

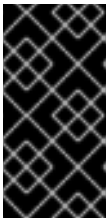
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.1.11.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

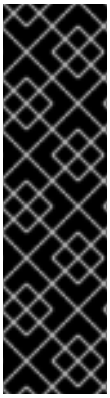
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.1.11.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
spec:
  accessModes:
  - ReadWriteOnce (2)
  resources:
    requests:
      storage: 100Gi (3)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (3) PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.1.12. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.1.13. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.2. カスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、インストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

12.2.1. 前提条件

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

12.2.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

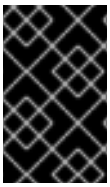
12.2.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表12.3 VMware コンポーネントのサポートされる vSphere の最小バージョン

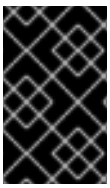
コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。



重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。



重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.2.4. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラス

ターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例12.3 インストールに必要なロールおよび特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	Always	**Cns.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	Always	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone Folder.Create Folder.Delete**

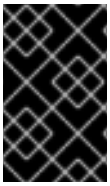
また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例12.4 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用



重要

OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。

Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスおよびホスト名をクラスターマシンに提供するように設定されていることを確認する必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード

OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.4 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

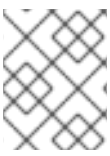
12.2.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.2.6. インストールプログラムの取得

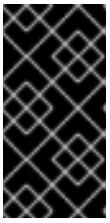
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

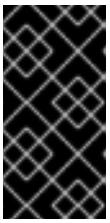
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.2.7. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。
2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

12.2.8. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。VMware vSphere

前提条件

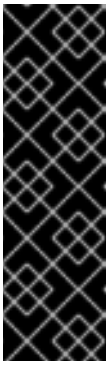
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

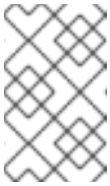
   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。
      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。
      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。
      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。
      v. クラスターをデプロイするリージョンを選択します。
      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。
      vii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。
      viii. vCenter インスタンスの名前を指定します。
      ix. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

x. 接続する vCenter インスタンスにあるデータセンターを選択します。

xi. 使用するデフォルトの vCenter データストアを選択します。

xii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。

xiii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

xiv. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

xv. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xvi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xvii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xviii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

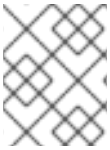
3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

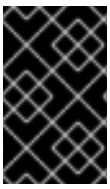
12.2.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.2.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.5 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.2.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.6 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

12.2.8.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.7 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

12.2.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表12.8 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

表12.9 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列

パラメーター	説明	値
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

12.2.8.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表12.10 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの CPU (vCPU) の数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

12.2.8.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
```

```
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster 8
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

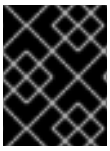
同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

12.2.9. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

12.2.10. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.2.10.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
   PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.2.10.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.2.10.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.2.11. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターに

ログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.2.12. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

12.2.12.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.2.12.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

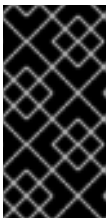
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.2.12.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

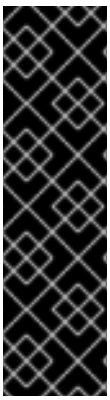
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

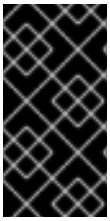
1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.2.12.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
  - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

   1 **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   2 PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   3 PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.2.13. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングしま

す。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.2.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.3. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、カスタマイズされるネットワーク設定オプションと共にインストーラーでプロビジョニングされるインフラストラクチャーを使用して、クラスターを VMware vSphere インスタンスにインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。インストールをカスタマイズするには、クラスターをインストールする前に、**install-config.yaml** ファイルでパラメーターを変更します。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

12.3.1. 前提条件

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

12.3.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.3.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表12.11 VMware コンポーネントのサポートされる vSphere の最小バージョン

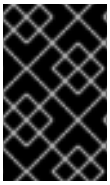
コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。

コンポーネント	サポートされる最小バージョン	説明
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。



重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.3.4. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例12.5 インストールに必要なロールおよび特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	Always	**Cns.Searchable InventoryService.Tagging.AttachTag InventoryService.Tagging.CreateCategory InventoryService.Tagging.CreateTag InventoryService.Tagging.DeleteCategory InventoryService.Tagging.DeleteTag InventoryService.Tagging.EditCategory InventoryService.Tagging.EditTag Sessions.ValidateSession StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage Resource.AssignVMToPool VApp.AssignResourcePool VApp.Import VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace Datastore.Browse Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	Always	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone Folder.Create Folder.Delete**

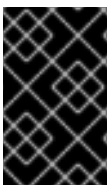
また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例12.6 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用



重要

OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。

Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスおよびホスト名をクラスターマシンに提供するように設定されていることを確認する必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード

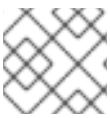
OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.12 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

12.3.5. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.3.6. インストールプログラムの取得

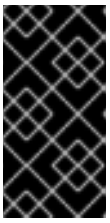
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。

重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.3.7. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates** をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

12.3.8. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。VMware vSphere

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

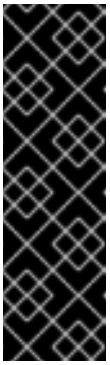
手順

1. **install-config.yaml** ファイルを作成します。

   a. 以下のコマンドを実行します。

   ```
   $ ./openshift-install create install-config --dir=<installation_directory> 1
   ```

   1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

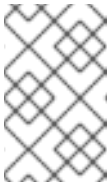
   

   重要

   空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

   b. プロンプト時に、クラウドの設定の詳細情報を指定します。

      i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

      注記

      インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

      ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

      iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

      iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

      v. クラスターをデプロイするリージョンを選択します。

      vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

      vii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

      viii. vCenter インスタンスの名前を指定します。

      ix. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
      インストールプログラムは vCenter インスタンスに接続します。

x. 接続する vCenter インスタンスにあるデータセンターを選択します。

xi. 使用するデフォルトの vCenter データストアを選択します。

xii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。

xiii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

xiv. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

xv. クラスター Ingress に設定した仮想 IP アドレスを入力します。

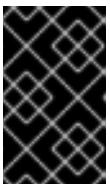
xvi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xvii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xviii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルを変更します。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

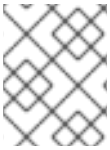


重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

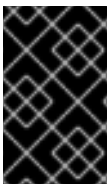
12.3.8.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。



注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。



重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.3.8.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.13 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.3.8.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.14 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。

networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。
	
	OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。
		CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。
		
		`networking:`
		`  serviceNetwork:`
		`  - 172.30.0.0/16`

networking.machineNetwork	マシンの IP アドレスブロック。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		`networking:`
		`  machineNetwork:`
		`  - cidr: 10.0.0.0/16`

networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

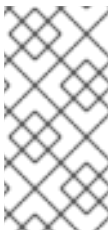
12.3.8.1.3. オプションの設定パラメーター

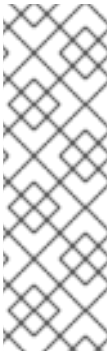
オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.15 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

12.3.8.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

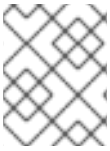
追加の GCP 設定パラメーターは以下の表で説明されています。

表12.16 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

12.3.8.2. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。



注記

インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

表12.17 必要なネットワークパラメーター

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。	デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの **serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

12.3.8.3. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
```

```
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: 5
  hyperthreading: Enabled 6
  name: master
  replicas: 3
  platform:
    vsphere: 7
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster 8
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name
    apiVIP: api_vip
    ingressVIP: ingress_vip
fips: false
pullSecret: '{"auths": ...}'
sshKey: 'ssh-ed25519 AAAA...'
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

12.3.9. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。

重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。

5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。

12.3.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
```

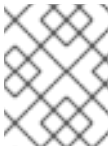
```
spec:
  clusterNetwork: 1
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork: 2
  - 172.30.0.0/16
  defaultNetwork: 3
    ...
  kubeProxyConfig: 4
    iptablesSyncPeriod: 30s 5
    proxyArguments:
      iptables-min-sync-period: 6
      - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。



注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time で説明されています。

12.3.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

1 **install-config.yaml** ファイルに指定されます。

2 OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

3 OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

4 VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

5 すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

12.3.10.2. Cluster Network Operator の設定例

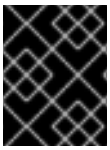
以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
      - 0s
```

12.3.11. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

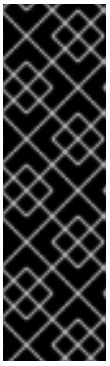
1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

注記

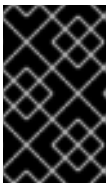
ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。



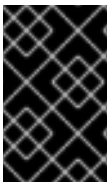
重要

インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。
   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

12.3.12. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.3.12.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.3.12.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.3.12.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.3.13. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

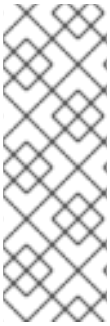
12.3.14. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

12.3.14.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.3.14.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

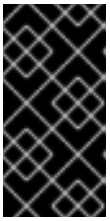
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.3.14.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

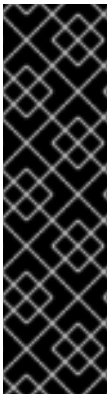
- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。



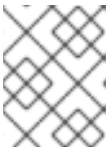
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.3.14.2.2. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
spec:
  accessModes:
  - ReadWriteOnce (2)
  resources:
    requests:
      storage: 100Gi (3)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (3) PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.3.15. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

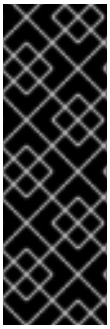
1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.3.16. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、プロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。

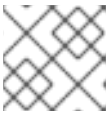


重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

12.4.1. 前提条件

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、クラスターがアクセスを必要とする サイトを許可するようにファイアウォールを設定 する必要があります。



注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

12.4.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.4.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表12.18 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.4.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

12.4.4.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。

注記

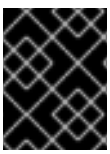
クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。



重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

12.4.4.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる

必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

12.4.4.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

12.4.4.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

12.4.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。

5. ネットワーク接続を確認します。

12.4.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表12.19 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表12.20 コントロールプレーンへのすべてのマシン

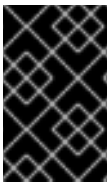
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表12.21 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件

クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.22 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

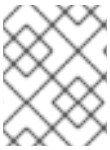
   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.23 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**

- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** - **00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

関連情報

- chrony タイムサービスの設定

12.4.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.24 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>. <base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例12.7 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H  ; refresh (3 hours)
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
  IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
```

```
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例12.8 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

12.4.6. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの **~/.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 **~/.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が **~/.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

12.4.7. インストールプログラムの取得

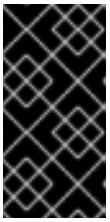
OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

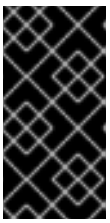
手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。



重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.4.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

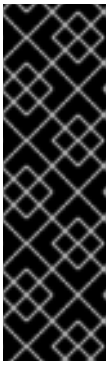
前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

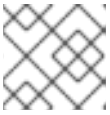
```
$ mkdir <installation_directory>
```



重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。



注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

12.4.8.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths": ...}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

Disabled に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

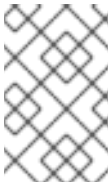
13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

12.4.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1 クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2 クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されま

3 プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。サブドメインのみと一致するように、ドメインの前に **.** を付けます。たとえば、**.y.com** は **x.y.com** に一致しますが、 **y.com** には一致しません。* を使用し、すべての宛先のプロキシーをバイパスします。vCenter の IP アドレスと、そのマシンに使用する IP 範囲を含める必要があります。

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを **openshift-config** namespace に生成します。次に Cluster Network Operator は、これらのコンテンツを Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルにマージする **trusted-ca-bundle** 設定マップを作成し、この設定マップは **Proxy** オブジェクトの **trustedCA** フィールドで参照されます。**additionalTrustBundle** フィールドは、プロキシーのアイデンティティー証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.4.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-
cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.4.10. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere (vSphere) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される {cp-template} テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

12.4.11. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるインフラストラクチャーが含まれるクラスターを VMware vSphere にインストールする前に、それが使用する RHCOS マシンを vSphere ホストに作成する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあります。
- vSphere クラスター を作成します。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
   ブートストラップ Ignition 設定ファイルはサイズが大きすぎて vApp プロパティーに適さないため、これをホストする必要があります。
2. ブートストラップノードの以下の二次的な Ignition 設定ファイルを、**<installation_directory>/append-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
```

```
    "config": {
      "append": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "2.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

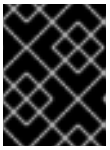
ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. マスター、ワーカー、および二次的なブートストラップ Ignition 設定ファイルを base64 エンコーディングに変換します。
たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

5. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。

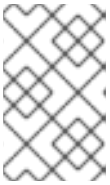
   a. **VMs and Templates** ビューをクリックします。

   b. データセンターの名前を右クリックします。

   c. **New Folder → New VM and Template Folder**をクリックします。

   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。

6. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。



注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。

   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。

   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。

   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。

   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。

      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。

   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。

   g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。

重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

7. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。

   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。

b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。

c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。

d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。

e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。

f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。

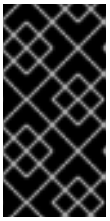
g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。

- オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。
- **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
  - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードした Ignition 設定ファイルの内容を貼り付けます。
  - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
  - **disk.EnableUUID**: **TRUE** を指定します。
- または、仮想マシンの電源を入れる前に vApp プロパティーを使用して追加します。
  - vCenter Server インベントリーから仮想マシンに移動します。
  - **Configure** タブで **Settings** を展開し、**vAPP options** を選択します。
  - スクロールダウンし、**Properties** の下で上記の設定を適用します。

h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

8. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。



重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

12.4.12. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

VMware vSphere でユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのコンピュートマシンを追加で作成できます。

前提条件

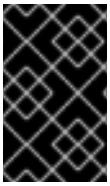
- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。
   i. 設定を完了し、仮想マシンの電源をオンにします。
2. 継続してクラスター用の追加のコンピュートマシンを作成します。

12.4.13. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.4.13.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.4.13.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.4.13.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.4.14. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。
- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

   ```
   $ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
       --log-level=info 2
   ```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

12.4.15. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.4.16. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

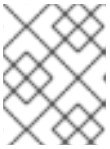
```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  73m  v1.20.0
master-1  Ready     master  73m  v1.20.0
master-2  Ready     master  74m  v1.20.0
worker-0  Ready     worker  11m  v1.20.0
worker-1  Ready     worker  11m  v1.20.0
```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.4.17. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.5.4     True        False         False      69s
cloud-credential           4.5.4     True        False         False      12m
cluster-autoscaler         4.5.4     True        False         False      11m
console                    4.5.4     True        False         False      46s
dns                        4.5.4     True        False         False      11m
image-registry             4.5.4     True        False         False      5m26s
ingress                    4.5.4     True        False         False      5m36s
kube-apiserver             4.5.4     True        False         False      8m53s
kube-controller-manager    4.5.4     True        False         False      7m24s
kube-scheduler             4.5.4     True        False         False      12m
machine-api                4.5.4     True        False         False      12m
```

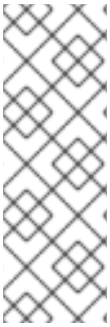
```
machine-config                      4.5.4     True        False         False      7m36s
marketplace                         4.5.4     True        False         False      7m54m
monitoring                          4.5.4     True        False         False      7h54s
network                             4.5.4     True        False         False      5m9s
node-tuning                         4.5.4     True        False         False      11m
openshift-apiserver                 4.5.4     True        False         False      11m
openshift-controller-manager        4.5.4     True        False         False      5m943s
openshift-samples                   4.5.4     True        False         False      3m55s
operator-lifecycle-manager          4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog  4.5.4     True        False         False      11m
service-ca                          4.5.4     True        False         False      11m
service-catalog-apiserver           4.5.4     True        False         False      5m26s
service-catalog-controller-manager  4.5.4     True        False         False      5m25s
storage                             4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

12.4.17.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.4.17.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

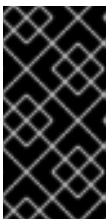
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.4.17.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

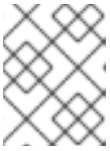
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.4.17.2.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

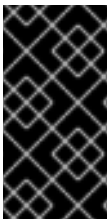
```
managementState: Removed
```

上記を以下のように変更します。

```
  managementState: Managed
```

12.4.17.2.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
spec:
  accessModes:
  - ReadWriteOnce (2)
  resources:
    requests:
      storage: 100Gi (3)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (3) PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.4.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      7m56s
cloud-credential                           4.5.4     True        False         False      31m
cluster-autoscaler                         4.5.4     True        False         False      16m
console                                    4.5.4     True        False         False      10m
csi-snapshot-controller                    4.5.4     True        False         False      16m
dns                                        4.5.4     True        False         False      22m
etcd                                       4.5.4     False       False         False      25s
image-registry                             4.5.4     True        False         False      16m
ingress                                    4.5.4     True        False         False      16m
insights                                   4.5.4     True        False         False      17m
kube-apiserver                             4.5.4     True        False         False      19m
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
```

```
machine-config                             4.5.4     True        False        False      22m
marketplace                                4.5.4     True        False        False      16m
monitoring                                 4.5.4     True        False        False      10m
network                                    4.5.4     True        False        False      23m
node-tuning                                4.5.4     True        False        False      23m
openshift-apiserver                        4.5.4     True        False        False      17m
openshift-controller-manager               4.5.4     True        False        False      15m
openshift-samples                          4.5.4     True        False        False      16m
operator-lifecycle-manager                 4.5.4     True        False        False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False        False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False        False      18m
service-ca                                 4.5.4     True        False        False      23m
service-catalog-apiserver                  4.5.4     True        False        False      23m
service-catalog-controller-manager         4.5.4     True        False        False      23m
storage                                    4.5.4     True        False        False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。

重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。
   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

```
$ oc get pods --all-namespaces
```

出力例

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
```

```
openshift-apiserver-operator      openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running     1          9m
openshift-apiserver               apiserver-67b9g                                 1/1     Running     0
3m
openshift-apiserver               apiserver-ljcmx                                 1/1     Running     0
1m
openshift-apiserver               apiserver-z25h4                                 1/1     Running     0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8        1/1
Running     0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace> 1
```

1 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

12.4.19. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.4.20. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.5. ネットワークのカスタマイズによる VSPHERE へのクラスターのインストール

OpenShift Container Platform バージョン 4.5 では、カスタマイズされたネットワーク設定オプションでプロビジョニングする VMware vSphere インフラストラクチャーにクラスターをインストールできます。ネットワーク設定をカスタマイズすることにより、クラスターは環境内の既存の IP アドレスの割り当てと共存でき、既存の MTU および VXLAN 設定と統合できます。

大半のネットワーク設定パラメーターはインストール時に設定する必要があり、実行中のクラスターで変更できるのは **kubeProxy** 設定パラメーターのみになります。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

12.5.1. 前提条件

- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用する場合、Red Hat Insights にアクセスできるように設定 する必要があります。

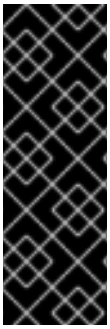
12.5.2. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするためにインターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.5.3. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

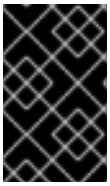
表12.25 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。



重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

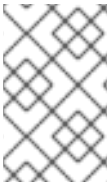
12.5.4. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

12.5.4.1. 必要なマシン

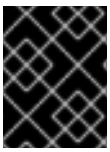
最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。



注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。



重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。

重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

12.5.4.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる

必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

12.5.4.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

12.5.4.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

12.5.5. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。

5. ネットワーク接続を確認します。

12.5.5.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表12.26 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

表12.27 コントロールプレーンへのすべてのマシン

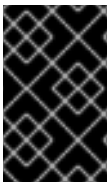
プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表12.28 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379**-**2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件

クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

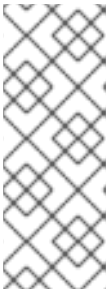
注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.29 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー



注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

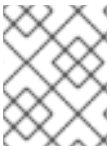
   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.30 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。



注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**

- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** - **00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

関連情報

- chrony タイムサービスの設定

12.5.5.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.31 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。

コンポーネント	レコード	説明
ルート	***.apps.<cluster_name>. <base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>. <cluster_name>. <base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例12.9 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
    2019070700 ; serial
    3H  ; refresh (3 hours)
    30M  ; retry (30 minutes)
    2W  ; expiry (2 weeks)
    1W )  ; minimum (1 week)
  IN NS ns1.example.com.
  IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
```

```
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

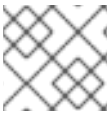
以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例12.10 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

12.5.6. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。



注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。

注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

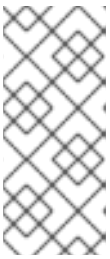
手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。



注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.5.7. インストールプログラムの取得

OpenShift Container Platform をインストールする前に、インストールファイルをローカルコンピューターにダウンロードします。

前提条件

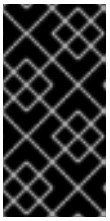
- Linux または macOS を使用するコンピューターからクラスターをインストールする必要があります。
- インストールプログラムをダウンロードするには、500 MB のローカルディスク領域が必要です。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページにアクセスします。Red Hat アカウントがある場合は、認証情報を使ってログインします。アカウントがない場合はこれを作成します。
2. 選択するインストールタイプのページに移動し、オペレーティングシステムのインストールプログラムをダウンロードし、ファイルをインストール設定ファイルを保存するディレクトリーに配置します。

重要

インストールプログラムは、クラスターのインストールに使用するコンピューターにいくつかのファイルを作成します。クラスターインストールの完了後は、インストールプログラムおよびインストールプログラムが作成するファイルの両方を保持する必要があります。



重要

インストールプログラムで作成されたファイルを削除しても、クラスターがインストール時に失敗した場合でもクラスターは削除されません。特定のクラウドプロバイダー用に記載された OpenShift Container Platform のアンインストール手順を完了して、クラスターを完全に削除する必要があります。

3. インストールプログラムを展開します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar xvf <installation_program>.tar.gz
```

4. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから、インストールプルシークレットを **.txt** ファイルとしてダウンロードします。このプルシークレットを使用し、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスで認証できます。

12.5.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

12.5.8.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths": ...}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を

Disabled に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレット。このプルシークレットを使用すると、OpenShift Container Platform コンポーネントのコンテナーイメージを提供する Quay.io など、組み込まれた各種の認証局によって提供されるサービスを使用して認証できます。

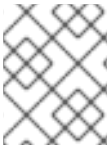
17 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

12.5.8.2. ネットワーク設定パラメーター

クラスターのネットワーク設定パラメーターは **install-config.yaml** 設定ファイルで変更できます。以下の表では、これらのパラメーターについて説明しています。



注記

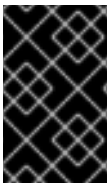
インストール後は、**install-config.yaml** ファイルでこれらのパラメーターを変更することはできません。

表12.32 必要なネットワークパラメーター

パラメーター	説明	値
networking.networkType	デプロイするデフォルトの Container Network Interface (CNI) ネットワークプロバイダープラグイン。**OpenShiftSDN** プラグインのみが OpenShift Container Platform 4.5 でサポートされているプラグインです。	デフォルト値は **OpenShiftSDN** です。
networking.clusterNetwork[].cidr	Pod IP アドレスの割り当てに使用する IP アドレスのブロック。**OpenShiftSDN** ネットワークプラグインは複数のクラスターネットワークをサポートします。複数のクラスターネットワークのアドレスブロックには重複が許可されません。予想されるワークロードに適したサイズのアドレスプールを選択してください。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.128.0.0/14** です。
networking.clusterNetwork[].hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞の長さ。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます (510 (2^(32 - 23) - 2) Pod IP アドレスが許可されます)。	サブネット接頭辞。デフォルト値は **23** です。
networking.serviceNetwork[]	サービスの IP アドレスのブロック。**OpenShiftSDN** は 1 つの **serviceNetwork** ブロックのみを許可します。このアドレスブロックは他のネットワークブロックと重複できません。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **172.30.0.0/16** です。
networking.machineNetwork[].cidr	クラスターのインストール中に OpenShift Container Platform インストールプログラムによって使用されるノードに割り当てられる IP アドレスのブロック。このアドレスブロックは他のネットワークブロックと重複できません。複数の CIDR 範囲を指定できます。	CIDR 形式の IP アドレスの割り当て。デフォルト値は **10.0.0.0/16** です。

12.5.9. 高度なネットワーク設定パラメーターの変更

高度なネットワーク設定パラメーターは、クラスターのインストール前にのみ変更することができます。高度な設定のカスタマイズにより、クラスターを既存のネットワーク環境に統合させることができます。 これを実行するには、MTU または VXLAN ポートを指定し、kube-proxy 設定のカスタマイズを許可し、**openshiftSDNConfig** パラメーターに異なる **mode** を指定します。



重要

インストールプログラムで作成される OpenShift Container Platform マニフェストファイルの変更はサポートされていません。以下の手順のように、作成するマニフェストファイルを適用することがサポートされています。

前提条件

- **install-config.yaml** ファイルを作成し、これに対する変更を完了します。
- クラスターの Ignition 設定ファイルを生成します。

手順

1. 以下のコマンドを使用してマニフェストを作成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

1 **<installation_directory>** については、クラスターの **install-config.yaml** ファイルが含まれるディレクトリーの名前を指定します。

2. **cluster-network-03-config.yml** という名前のファイルを **<installation_directory>/manifests/** ディレクトリーに作成します。

```
$ touch <installation_directory>/manifests/cluster-network-03-config.yml 1
```

1 **<installation_directory>** については、クラスターの **manifests/** ディレクトリーが含まれるディレクトリー名を指定します。

ファイルの作成後は、以下のようにいくつかのネットワーク設定ファイルが **manifests/** ディレクトリーに置かれます。

```
$ ls <installation_directory>/manifests/cluster-network-*
```

出力例

```
cluster-network-01-crd.yml
cluster-network-02-config.yml
cluster-network-03-config.yml
```

3. エディターで **cluster-network-03-config.yml** ファイルを開き、必要な Operator 設定を記述する CR を入力します。

```
apiVersion: operator.openshift.io/v1
kind: Network
```

```
metadata:
  name: cluster
spec: 1
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
```

1 **spec** パラメーターのパラメーターは例です。CR に Cluster Network Operator の設定を指定します。

CNO は CR にパラメーターのデフォルト値を提供するため、変更が必要なパラメーターのみを指定する必要があります。

4. **cluster-network-03-config.yml** ファイルを保存し、テキストエディターを終了します。
5. オプション: **manifests/cluster-network-03-config.yml** ファイルをバックアップします。インストールプログラムは、クラスターの作成時に **manifests/** ディレクトリーを削除します。
6. コントロールプレーンマシンおよび compute machineSets を定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- MachineSet ファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

12.5.10. Cluster Network Operator (CNO) の設定

クラスターネットワークの設定は、Cluster Network Operator (CNO) 設定の一部として指定され、**cluster** という名前の CR オブジェクトに保存されます。CR は **operator.openshift.io** API グループの **Network** API のパラメーターを指定します。

defaultNetwork パラメーターのパラメーター値を CNO CR に設定することにより、OpenShift Container Platform クラスターのクラスターネットワーク設定を指定できます。以下の CR は、CNO のデフォルト設定を表示し、設定可能なパラメーターと有効なパラメーターの値の両方について説明しています。

Cluster Network Operator CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
```

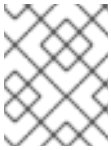
```
clusterNetwork: 1
- cidr: 10.128.0.0/14
  hostPrefix: 23
serviceNetwork: 2
- 172.30.0.0/16
defaultNetwork: 3
  ...
kubeProxyConfig: 4
  iptablesSyncPeriod: 30s 5
  proxyArguments:
    iptables-min-sync-period: 6
    - 0s
```

1 2 **install-config.yaml** ファイルに指定されます。

3 クラスターネットワークのデフォルトの Container Network Interface (CNI) ネットワークプロバイダーを設定します。

4 このオブジェクトのパラメーターは、**kube-proxy** 設定を指定します。パラメーターの値を指定しない場合、クラスターネットワーク Operator は表示されるデフォルトのパラメーター値を適用します。OVN-Kubernetes デフォルト CNI ネットワークプロバイダーを使用している場合、kube-proxy 設定は機能しません。

5 **iptables** ルールの更新期間。デフォルト値は **30s** です。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time ドキュメントで説明されています。



注記

OpenShift Container Platform 4.3 以降で強化されたパフォーマンスの向上により、**iptablesSyncPeriod** パラメーターを調整する必要はなくなりました。

6 **iptables** ルールを更新する前の最小期間。このパラメーターにより、更新の頻度が高くなり過ぎないようにできます。有効な接尾辞には、**s**、**m**、および **h** などが含まれ、これらについては、Go Package time で説明されています。

12.5.10.1. OpenShift SDN デフォルト CNI ネットワークプロバイダーの設定パラメーター

以下の YAML オブジェクトは、OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーの設定パラメーターについて説明しています。

```
defaultNetwork:
  type: OpenShiftSDN 1
  openshiftSDNConfig: 2
    mode: NetworkPolicy 3
    mtu: 1450 4
    vxlanPort: 4789 5
```

1 **install-config.yaml** ファイルに指定されます。

2 OpenShift SDN 設定の一部を上書きする必要がある場合にのみ指定します。

3 OpenShift SDN のネットワーク分離モードを設定します。許可される値は **Multitenant**、**Subnet**、または **NetworkPolicy** です。デフォルト値は **NetworkPolicy** です。

multitenant、**subnet**、または **NetworkPolicy** モード。デフォルト値は **NetworkPolicy** モード。

4 VXLAN オーバーレイネットワークの最大転送単位 (MTU)。これは、プライマリーネットワークインターフェイスの MTU に基づいて自動的に検出されます。通常、検出された MTU を上書きする必要はありません。

自動検出した値が予想される値ではない場合は、ノード上のプライマリーネットワークインターフェイスの MTU が正しいことを確認します。このオプションを使用して、ノード上のプライマリーネットワークインターフェイスの MTU 値を変更することはできません。

クラスターで異なるノードに異なる MTU 値が必要な場合、この値をクラスター内の最小の MTU 値よりも **50** 小さく設定する必要があります。たとえば、クラスター内の一部のノードでは MTU が **9001** であり、MTU が **1500** のクラスターもある場合には、この値を **1450** に設定する必要があります。

5 すべての VXLAN パケットに使用するポート。デフォルト値は **4789** です。別の VXLAN ネットワークの一部である既存ノードと共に仮想化環境で実行している場合は、これを変更する必要がある可能性があります。たとえば、OpenShift SDN オーバーレイを VMware NSX-T 上で実行する場合は、両方の SDN が同じデフォルトの VXLAN ポート番号を使用するため、VXLAN の別のポートを選択する必要があります。

Amazon Web Services (AWS) では、VXLAN にポート **9000** とポート **9999** 間の代替ポートを選択できます。

12.5.10.2. Cluster Network Operator の設定例

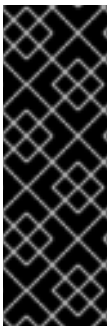
以下の例のように、CNO の完全な CR オブジェクトが表示されます。

Cluster Network Operator のサンプル CR

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      mtu: 1450
      vxlanPort: 4789
  kubeProxyConfig:
    iptablesSyncPeriod: 30s
    proxyArguments:
      iptables-min-sync-period:
      - 0s
```

12.5.11. Ignition 設定ファイルの作成

クラスターマシンは手動で起動する必要があるため、クラスターがマシンを作成するために必要な Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

手順

- Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。

重要

install-config.yaml ファイルを作成している場合、それが含まれるディレクトリーを指定します。または、空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.5.12. インフラストラクチャー名の抽出

Ignition 設定ファイルには、{cp-first} ({cp}) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される {cp-template} テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

12.5.13. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるインフラストラクチャーが含まれるクラスターを VMware vSphere にインストールする前に、それが使用する RHCOS マシンを vSphere ホストに作成する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあります。
- vSphere クラスター を作成します。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
   ブートストラップ Ignition 設定ファイルはサイズが大きすぎて vApp プロパティーに適さないため、これをホストする必要があります。

2. ブートストラップノードの以下の二次的な Ignition 設定ファイル
を、**<installation_directory>/append-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "append": [
        {
          "source": "<bootstrap_ignition_config_url>", (1)
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "2.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

(1) ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

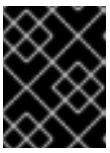
ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. マスター、ワーカー、および二次的なブートストラップ Ignition 設定ファイルを base64 エンコーディングに変換します。
たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

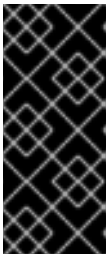
```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```



重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。



重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

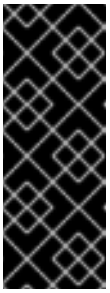
ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

5. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。
6. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。
   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。
   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。
   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。
      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。
   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。
   g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

7. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
    a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
    b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
    c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
    d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
    e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
    f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
    g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
        - オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。
        - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
            - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードした Ignition 設定ファイルの内容を貼り付けます。
            - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
            - **disk.EnableUUID**: **TRUE** を指定します。
        - または、仮想マシンの電源を入れる前に vApp プロパティーを使用して追加します。
            - vCenter Server インベントリーから仮想マシンに移動します。
            - **Configure** タブで **Settings** を展開し、**vAPP options** を選択します。
            - スクロールダウンし、**Properties** の下で上記の設定を適用します。
    h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。
    i. 設定を完了し、仮想マシンの電源をオンにします。
8. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

12.5.14. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

VMware vSphere でユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのコンピュートマシンを追加で作成できます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

2. 継続してクラスター用の追加のコンピュートマシンを作成します。

12.5.15. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。

重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.5.15.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.5.15.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。

3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.5.15.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.5.16. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。

- お使いのマシンでインターネットに直接アクセスできるか、または HTTP または HTTPS プロキシーが利用できる。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

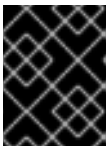
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。



重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

12.5.17. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.5.18. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
```

```
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-
bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。

注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
{{end}}{{end}}' | xargs oc adm certificate approve
```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

```
$ oc get nodes
```

出力例

```
NAME     STATUS   ROLES   AGE  VERSION
master-0 Ready    master  73m  v1.20.0
master-1 Ready    master  73m  v1.20.0
master-2 Ready    master  74m  v1.20.0
worker-0 Ready    worker  11m  v1.20.0
worker-1 Ready    worker  11m  v1.20.0
```

注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.5.19. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
```

```
authentication                 4.5.4     True        False         False      69s
cloud-credential               4.5.4     True        False         False      12m
cluster-autoscaler             4.5.4     True        False         False      11m
console                        4.5.4     True        False         False      46s
dns                            4.5.4     True        False         False      11m
image-registry                 4.5.4     True        False         False      5m26s
ingress                        4.5.4     True        False         False      5m36s
kube-apiserver                 4.5.4     True        False         False      8m53s
kube-controller-manager        4.5.4     True        False         False      7m24s
kube-scheduler                 4.5.4     True        False         False      12m
machine-api                    4.5.4     True        False         False      12m
machine-config                 4.5.4     True        False         False      7m36s
marketplace                    4.5.4     True        False         False      7m54m
monitoring                     4.5.4     True        False         False      7h54s
network                        4.5.4     True        False         False      5m9s
node-tuning                    4.5.4     True        False         False      11m
openshift-apiserver            4.5.4     True        False         False      11m
openshift-controller-manager   4.5.4     True        False         False      5m943s
openshift-samples              4.5.4     True        False         False      3m55s
operator-lifecycle-manager     4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog   4.5.4     True        False         False      11m
service-ca                     4.5.4     True        False         False      11m
service-catalog-apiserver      4.5.4     True        False         False      5m26s
service-catalog-controller-manager   4.5.4     True        False         False      5m25s
storage                        4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

12.5.19.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。

注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.5.19.2. イメージレジストリーストレージの設定

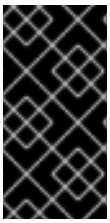
イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.5.19.2.1. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。



重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
spec:
  accessModes:
  - ReadWriteOnce 2
  resources:
    requests:
      storage: 100Gi 3
```

   1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
   2. PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
   3. PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.5.20. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.5.4     True        False         False      7m56s
cloud-credential           4.5.4     True        False         False      31m
cluster-autoscaler         4.5.4     True        False         False      16m
console                    4.5.4     True        False         False      10m
csi-snapshot-controller    4.5.4     True        False         False      16m
dns                        4.5.4     True        False         False      22m
etcd                       4.5.4     False       False         False      25s
image-registry             4.5.4     True        False         False      16m
ingress                    4.5.4     True        False         False      16m
insights                   4.5.4     True        False         False      17m
kube-apiserver             4.5.4     True        False         False      19m
```

```
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

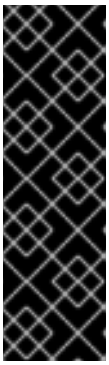
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

```
NAMESPACE                    NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver          apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver          apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver          apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

12.5.21. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.5.22. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.6. ネットワークが制限された環境での VSPHERE へのクラスターのインストール

OpenShift Container Platform 4.5 では、インストールリリースコンテンツの内部ミラーを作成して、クラスターをネットワークが制限された環境で VMware vSphere インフラストラクチャーにインストールできます。

12.6.1. 前提条件

- ミラーホストでレジストリーを作成 し、OpenShift Container Platform の使用しているバージョン用の **imageContentSources** データを取得します。

重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了します。

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで ReadWriteMany アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。

注記

プロキシーを設定する場合は、このサイト一覧も確認してください。

12.6.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。

12.6.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

12.6.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。

重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.6.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

表12.33 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。

重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.6.5. vCenter の要件

インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターを vCenter にインストールする前に、環境を準備する必要があります。

必要な vCenter アカウントの権限
OpenShift Container Platform クラスターを vCenter にインストールするには、インストールプログラムには、必要なリソースの読み取りおよび作成権限を持つアカウントへのアクセスが必要になります。グローバル管理者権限のあるアカウントを使用すること方法が、必要なすべてのパーミッションにアクセスするための最も簡単な方法です。

グローバル管理者権限を持つアカウントを使用できない場合、OpenShift Container Platform クラスターのインストールに必要な権限を付与するためのロールを作成する必要があります。ほとんどの特権

は常に必要になりますが、デフォルト動作であるインストールプログラムでの vCenter インスタンスへの OpenShift Container Platform クラスターが含まれるフォルダーのプロビジョニングを実行する場合にのみ必要となる特権もあります。必要な特権を付与するには、指定されたオブジェクトに vSphere ロールを作成するか、またはこれを修正する必要があります。

インストールプログラムが vSphere 仮想マシンフォルダーを作成するために使用される場合には、追加のロールが必要です。

例12.11 インストールに必要なロールおよび特権

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter	Always	**Cns.Searchable** **InventoryService.Tagging.AttachTag** **InventoryService.Tagging.CreateCategory** **InventoryService.Tagging.CreateTag** **InventoryService.Tagging.DeleteCategory** **InventoryService.Tagging.DeleteTag** **InventoryService.Tagging.EditCategory** **InventoryService.Tagging.EditTag** **Sessions.ValidateSession** **StorageProfile.View**
vSphere vCenter Cluster	Always	**Host.Config.Storage** **Resource.AssignVMToPool** **VApp.AssignResourcePool** **VApp.Import** **VirtualMachine.Config.AddNewDisk**
vSphere Datastore	Always	**Datastore.AllocateSpace** **Datastore.Browse** **Datastore.FileManagement**
vSphere ポートグループ	Always	**Network.Assign**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
仮想マシンフォルダー	Always	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone**

ロールの vSphere オブジェクト	必要になる場合	必要な特権
vSphere vCenter Datacenter	インストールプログラムが仮想マシンフォルダーを作成する場合	**Resource.AssignVMToPool VApp.Import VirtualMachine.Config.Add ExistingDisk VirtualMachine.Config.Add NewDisk VirtualMachine.Config.Add RemoveDevice VirtualMachine.Config.Adva ncedConfig VirtualMachine.Config.Anno tation VirtualMachine.Config.CPU Count VirtualMachine.Config.Disk Extend VirtualMachine.Config.Disk Lease VirtualMachine.Config.Edit Device VirtualMachine.Config.Mem ory VirtualMachine.Config.Rem oveDisk VirtualMachine.Config.Rena me VirtualMachine.Config.Rese tGuestInfo VirtualMachine.Config.Reso urce VirtualMachine.Config.Setti ngs VirtualMachine.Config.Upgr adeVirtualHardware VirtualMachine.Interact.Gue stControl VirtualMachine.Interact.Pow erOff VirtualMachine.Interact.Pow erOn VirtualMachine.Interact.Res et VirtualMachine.Inventory.Cr eate VirtualMachine.Inventory.Cr eateFromExisting VirtualMachine.Inventory.D elete VirtualMachine.Provisionin g.Clone Folder.Create Folder.Delete**

また、ユーザーには一部の **ReadOnly** パーミッションが必要であり、一部のロールでは、パーミッションを子オブジェクトに伝播するパーミッションが必要です。これらの設定は、クラスターを既存のフォルダーにインストールするかどうかによって異なります。

例12.12 必要なパーミッションおよび伝播の設定

vSphere オブジェクト	フォルダータイプ	子への伝播	パーミッションが必要
vSphere vCenter	Always	False	必要な特権が一覧表示
vSphere vCenter Datacenter	既存のフォルダー	False	**ReadOnly** パーミッション
	インストールプログラムがフォルダーを作成する	True	必要な特権が一覧表示
vSphere vCenter Cluster	Always	True	必要な特権が一覧表示
vSphere vCenter Datastore	Always	False	必要な特権が一覧表示
vSphere Switch	Always	False	**ReadOnly** パーミッション
vSphere ポートグループ	Always	False	必要な特権が一覧表示
vSphere vCenter 仮想マシンフォルダー	既存のフォルダー	True	必要な特権が一覧表示

必要な権限のみを持つアカウントの作成に関する詳細は、vSphere ドキュメントの vSphere Permissions and User Management Tasks を参照してください。

OpenShift Container Platform と vMotion の使用

重要

OpenShift Container Platform は通常、コンピュートのみの vMotion をサポートします。Storage vMotion を使用すると問題が発生する可能性があるため、これはサポートされていません。

Pod で vSphere ボリュームを使用している場合、手動でまたは Storage vMotion を使用して仮想マシンをデータストア間で移行すると、OpenShift Container Platform 永続ボリューム (PV) オブジェクト内で無効な参照が発生します。これらの参照により、影響を受ける Pod が起動しなくなり、データが失われる可能性があります。

同様に、OpenShift Container Platform は、仮想マシンのプロビジョニング用にデータストアクラスターを、または PV の動的または静的プロビジョニング用にデータストアクラスターを使用するか、PV の動的または静的プロビジョニング用にデータストアクラスターの一部であるデータストアを使用した VMDK のデータストア間での選択的な移行をサポートしません。

クラスターリソース
インストーラーでプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする場合、インストールプログラムは vCenter インスタンスに複数のリソースを作成できる必要があります。

標準的な OpenShift Container Platform インストールでは、以下の vCenter リソースを作成します。

- 1 フォルダー
- 1 タグカテゴリー
- 1 タグ
- 仮想マシン:
  - 1 テンプレート
  - 1 一時的ブートストラップノード
  - 3 コントロールプレーンノード
  - 3 コンピュートマシン

これらのリソースは 856 GB のストレージを使用しますが、ブートストラップノードはクラスターのインストールプロセス時に破棄されます。標準クラスターを使用するには、最低 800 GB のストレージが必要です。

追加のコンピュートマシンをデプロイする場合、OpenShift Container Platform クラスターは追加のストレージを使用します。

クラスターの制限
利用可能なリソースはクラスターによって異なります。vCenter 内の予想されるクラスター数は、主に利用可能なストレージ容量と必要なリソース数の制限によって制限されます。クラスターが作成する vCenter リソースと、IP アドレスやネットワークなどのクラスターのデプロイに必要なリソースの両方の制限を考慮してください。

ネットワーク要件
ネットワークに DHCP を使用し、DHCP サーバーが永続 IP アドレスおよびホスト名をクラスターマシンに提供するように設定されていることを確認する必要があります。ネットワークが制限された環境の仮想マシンは、ノード、永続ボリューム要求 (PVC) および他のリソースをプロビジョニングし、管理できるように vCenter にアクセスできる必要があります。さらに、OpenShift Container Platform クラスターをインストールする前に以下のネットワークリソースを作成する必要があります。

必要な IP アドレス
インストーラーでプロビジョニングされる vSphere のインストールには、2 つの静的 IP アドレスが必要です。

- **API** アドレスは、クラスター API にアクセスするために使用されます。
- **Ingress** アドレスは、クラスターの Ingress トラフィックに使用されます。

OpenShift Container Platform クラスターのインストール時にこれらの IP アドレスをインストールプログラムに指定する必要があります。

DNS レコード

OpenShift Container Platform クラスターをホストする vCenter インスタンスについて 2 つの静的 IP アドレスの DNS レコードを適切な DNS サーバーに作成する必要があります。各レコードで、**<cluster_name>** はクラスター名で、**<base_domain>** は、クラスターのインストール時に指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.34 必要な DNS レコード

コンポーネント	レコード	説明
API VIP	**api.<cluster_name>.<base_domain>.**	この DNS A/AAAA または CNAME レコードは、コントロールプレーンマシンのロードバランサーを参照する必要があります。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
Ingress VIP	***.apps.<cluster_name>.<base_domain>.**	Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードです。このレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

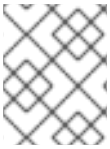
12.6.6. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/.ssh/id_rsa などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/.ssh ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 ~/.ssh/id_rsa などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。

12.6.7. vCenter ルート CA 証明書のシステム信頼への追加

インストールプログラムは vCenter の API へのアクセスが必要であるため、OpenShift Container Platform クラスターをインストールする前に vCenter の信頼されたルート CA 証明書をシステム信頼に追加する必要があります。

手順

1. vCenter ホームページから、vCenter のルート CA 証明書をダウンロードします。vSphere Web Services SDK セクションで、**Download trusted root CA certificates**をクリックします。**<vCenter>/certs/download.zip** ファイルがダウンロードされます。

2. vCenter ルート CA 証明書が含まれる圧縮ファイルを展開します。圧縮ファイルの内容は、以下のファイル構造のようになります。

```
certs
├── lin
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
├── mac
│   ├── 108f4d17.0
│   ├── 108f4d17.r1
│   ├── 7e757f6a.0
│   ├── 8e4f8471.0
│   └── 8e4f8471.r0
└── win
    ├── 108f4d17.0.crt
    ├── 108f4d17.r1.crl
    ├── 7e757f6a.0.crt
    ├── 8e4f8471.0.crt
    └── 8e4f8471.r0.crl

3 directories, 15 files
```

3. オペレーティングシステム用のファイルをシステム信頼に追加します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

4. システム信頼を更新します。たとえば、Fedora オペレーティングシステムで以下のコマンドを実行します。

```
# update-ca-trust extract
```

12.6.8. ネットワークが制限されたインストール用の RHCOS イメージの作成

Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードし、OpenShift Container Platform をネットワークが制限された VMware vSphere 環境にインストールします。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、プログラムはミラーレジストリースト上に置かれます。

手順

1. Red Hat カスタマーポータルの 製品ダウンロードページ にログインします。
2. **Version** で、RHEL 8 用の OpenShift Container Platform 4.5 の最新リリースを選択します。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

3. **Red Hat Enterprise Linux CoreOS (RHCOS) - vSphere**イメージをダウンロードします。
4. ダウンロードしたイメージを、bastion サーバーからアクセス可能な場所にアップロードします。

これで、イメージが制限されたインストールで利用可能になります。OpenShift Container Platform デプロイメントで使用するイメージの名前または場所をメモします。

12.6.9. インストール設定ファイルの作成

Google Cloud Platform (GCP) にインストールする OpenShift Container Platform クラスターをカスタマイズできます。VMware vSphere

前提条件

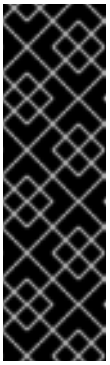
- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。ネットワークが制限されたインストールでは、これらのファイルが bastion ホスト上に置かれます。
- ミラーレジストリーの作成時に生成された **imageContentSources** 値を使用します。
- ミラーレジストリーの証明書の内容を取得します。

手順

1. **install-config.yaml** ファイルを作成します。
   a. 以下のコマンドを実行します。

```
$ ./openshift-install create install-config --dir=<installation_directory> 1
```

1 **<installation_directory>** の場合、インストールプログラムが作成するファイルを保存するためにディレクトリー名を指定します。



重要

空のディレクトリーを指定します。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

b. プロンプト時に、クラウドの設定の詳細情報を指定します。

i. オプション: クラスターマシンにアクセスするために使用する SSH キーを選択します。

注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

ii. ターゲットに設定するプラットフォームとして **gcp** を選択します。

iii. コンピューター上で GCP アカウント用のサービスアカウントキーを設定していない場合、GCP からこれを取得してファイルの内容を貼り付けるか、またはファイルへの絶対パスを入力する必要があります。

iv. クラスターのプロビジョニングに使用するプロジェクト ID を選択します。デフォルト値は、設定したサービスアカウントによって指定されます。

v. クラスターをデプロイするリージョンを選択します。

vi. クラスターをデプロイするベースドメインを選択します。ベースドメインは、クラスターに作成したパブリック DNS ゾーンに対応します。

vii. ターゲットに設定するプラットフォームとして **vsphere** を選択します。

viii. vCenter インスタンスの名前を指定します。

ix. クラスターを作成するのに必要なパーミッションを持つ vCenter アカウントのユーザー名およびパスワードを指定します。
インストールプログラムは vCenter インスタンスに接続します。

x. 接続する vCenter インスタンスにあるデータセンターを選択します。

xi. 使用するデフォルトの vCenter データストアを選択します。

xii. OpenShift Container Platform クラスターをインストールする vCenter クラスターを選択します。

xiii. 設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワークを選択します。

xiv. コントロールプレーン API のアクセス用に設定した仮想 IP アドレスを入力します。

xv. クラスター Ingress に設定した仮想 IP アドレスを入力します。

xvi. ベースドメインを入力します。このベースドメインは、設定した DNS レコードで使用したものと同じである必要があります。

xvii. クラスターの記述名を入力します。クラスター名は、設定した DNS レコードで使用したものと同じである必要があります。

xviii. Red Hat OpenShift Cluster Manager サイトの Pull Secret ページから取得したプルシークレットを貼り付けます。

2. **install-config.yaml** ファイルで **platform.vsphere.clusterOSImage** の値をイメージの場所または名前に設定します。以下に例を示します。

```
platform:
  vsphere:
    clusterOSImage: http://mirror.example.com/images/rhcos-43.81.201912131630.0-
vmware.x86_64.ova?
sha256=ffebbd68e8a1f2a245ca19522c16c86f67f9ac8e4e0c1f0a812b068b16f7265d
```

3. **install-config.yaml** ファイルを編集し、ネットワークが制限された環境でのインストールに必要な追加の情報を提供します。

a. **pullSecret** の値を更新して、レジストリーの認証情報を追加します。

```
pullSecret: '{"auths":{"<bastion_host_name>:5000": {"auth": "<credentials>","email":
"you@example.com"}}}'
```

<bastion_host_name> の場合、ミラーレジストリーの証明書で指定したレジストリードメイン名を指定し、 **<credentials>** の場合は、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

b. **additionalTrustBundle** パラメーターおよび値を追加します。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----

ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
```

この値は、ミラーレジストリーに使用した証明書ファイルの内容である必要があります。これはミラーレジストリー用に生成した既存の、信頼される認証局または自己署名証明書である可能性があります。

c. VPC のネットワークおよびサブネットを定義して、親の **platform.gcp** フィールドの下にクラスターをインストールします。

```
network: <existing_vpc>
controlPlaneSubnet: <control_plane_subnet>
computeSubnet: <compute_subnet>
```

platform.gcp.network には、既存の Google VPC の名前を指定します。**platform.gcp.controlPlaneSubnet** および **platform.gcp.computeSubnet** の場合には、コントロールプレーンマシンとコンピュートマシンをそれぞれデプロイするために既

存のサブネットを指定します。

d. 以下のようなイメージコンテンツリソースを追加します。

```
imageContentSources:
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: quay.example.com/openshift-release-dev/ocp-release
- mirrors:
  - <bastion_host_name>:5000/<repo_name>/release
  source: registry.example.com/ocp/release
```

これらの値を完了するには、ミラーレジストリーの作成時に記録された **imageContentSources** を使用します。

4. 必要な **install-config.yaml** ファイルに他の変更を加えます。利用可能なパラメーターの詳細については、**インストール設定パラメーター**セクションを参照してください。

5. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルはインストールプロセス時に使用されます。このファイルを再利用する必要がある場合は、この段階でこれをバックアップしてください。

12.6.9.1. インストール設定パラメーター

OpenShift Container Platform クラスターをデプロイする前に、クラスターをホストするクラウドプラットフォームでアカウントを記述し、クラスターのプラットフォームをオプションでカスタマイズするためにパラメーターの値を指定します。**install-config.yaml** インストール設定ファイルを作成する際に、コマンドラインで必要なパラメーターの値を指定します。クラスターをカスタマイズする場合、**install-config.yaml** ファイルを変更して、プラットフォームについての詳細情報を指定できます。

注記

インストール後は、これらのパラメーターを **install-config.yaml** ファイルで変更することはできません。

重要

openshift-install コマンドは、パラメーターのフィールド名を検証しません。正しくない名前を指定すると、関連するファイルまたはオブジェクトは作成されず、エラーが報告されません。指定されたパラメーターのフィールド名が正しいことを確認します。

12.6.9.1.1. 必須設定パラメーター

必須のインストール設定パラメーターは、以下の表で説明されています。

表12.35 必須パラメーター

パラメーター	説明	値
apiVersion	**install-config.yaml** コンテンツの API バージョン。現在のバージョンは **v1** です。インストーラーは、古い API バージョンをサポートすることもできます。	文字列
baseDomain	クラウドプロバイダーのベースドメイン。ベースドメインは、OpenShift Container Platform クラスターコンポーネントへのルートを作成するために使用されます。クラスターの完全な DNS 名は、**baseDomain** と **<metadata.name>.<baseDomain>** 形式を使用する **metadata.name** パラメーターの値の組み合わせです。	**example.com** などの完全修飾ドメインまたはサブドメイン名。
metadata	Kubernetes リソース **ObjectMeta**。ここからは **name** パラメーターのみが消費されます。	オブジェクト
metadata.name	クラスターの名前。クラスターの DNS レコードはすべて **{{.metadata.name}}.{{.baseDomain}}** のサブドメインです。	**dev** などの小文字、ハイフン (**-**)、およびピリオド (**.**) が含まれる文字列。
platform	インストールの実行に使用する特定プラットフォームの設定: **aws**、**baremetal**、**azure**、**openstack**、**ovirt**、**vsphere**。**platform.<platform>** パラメーターに関する追加情報は、以下の表で特定のプラットフォームについて参照してください。	オブジェクト

パラメーター	説明	値

pullSecret	https://cloud.redhat.com/openshift/install/pull-secret からプルシークレットを取得し、Quay.io などのサービスから OpenShift Container Platform コンポーネントのコンテナーイメージのダウンロードを認証します。	<pre>{
		  "auths":{
		  "cloud.openshift.com":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  },
		  "quay.io":{
		  "auth":"b3Blb=",
		  "email":"you@example.com"
		  }
		  }
		}</pre>

12.6.9.1.2. ネットワーク設定パラメーター

既存のネットワークインフラストラクチャーの要件に基づいて、インストール設定をカスタマイズできます。たとえば、クラスターネットワークの IP アドレスブロックを拡張するか、デフォルトとは異なる IP アドレスブロックを指定できます。

IPv4 アドレスのみがサポートされます。

表12.36 ネットワークパラメーター

パラメーター	説明	値
networking	クラスターのネットワークの設定。	オブジェクト **注記** インストール後に **networking** オブジェクトで指定したパラメーターを変更することはできません。
networking.networkType	インストールするクラスターネットワークプロバイダー Container Network Interface (CNI) プラグイン。	**OpenShiftSDN** または **OVNKubernetes** のいずれか。デフォルト値は **OpenShiftSDN** です。

networking.clusterNetwork	Pod の IP アドレスブロック。
	
	デフォルト値は **10.128.0.0/14** で、ホストの接頭辞は **/23** です。
	
	複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。
		オブジェクトの配列。以下に例を示します。
		
		<pre>networking:
		  clusterNetwork:
		  - cidr: 10.128.0.0/14
		  hostPrefix: 23</pre>

パラメーター	説明	値
networking.clusterNetwork.cidr	**networking.clusterNetwork** を使用する場合に必須です。IP アドレスブロック。 IPv4 ネットワーク	CIDR (Classless Inter-Domain Routing) 表記の IP アドレスブロック。IPv4 ブロックの接頭辞長は **0** から **32** の間になります。
networking.clusterNetwork.hostPrefix	それぞれの個別ノードに割り当てるサブネット接頭辞長。たとえば、**hostPrefix** が **23** に設定される場合、各ノードに指定の **cidr** から **/23** サブネットが割り当てられます。**hostPrefix** 値の **23** は、510 (2^(32 - 23) - 2) Pod IP アドレスを提供します。	サブネット接頭辞。 デフォルト値は **23** です。
networking.serviceNetwork	サービスの IP アドレスブロック。デフォルト値は **172.30.0.0/16** です。 OpenShift SDN および OVN-Kubernetes ネットワークプロバイダーは、サービスネットワークの単一 IP アドレスブロックのみをサポートします。	CIDR 形式の IP アドレスブロックを持つ配列。以下に例を示します。 networking: serviceNetwork: - 172.30.0.0/16
networking.machineNetwork	マシンの IP アドレスブロック。 複数の IP アドレスブロックを指定する場合は、ブロックが重複しないようにしてください。	オブジェクトの配列。以下に例を示します。 networking: machineNetwork: - cidr: 10.0.0.0/16
networking.machineNetwork.cidr	**networking.machineNetwork** を使用する場合に必須です。IP アドレスブロック。libvirt 以外のすべてのプラットフォームでは、デフォルト値は **10.0.0.0/16** です。libvirt の場合、デフォルト値は **192.168.126.0/24** です。	CIDR 表記の IP ネットワークブロック。 例: **10.0.0.0/16** **注記** 優先される NIC が置かれている CIDR に一致する **networking.machineNetwork** を設定します。

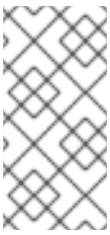
12.6.9.1.3. オプションの設定パラメーター

オプションのインストール設定パラメーターは、以下の表で説明されています。

表12.37 オプションのパラメーター

パラメーター	説明	値
additionalTrustBundle	ノードの信頼済み証明書ストアに追加される PEM でエンコードされた X.509 証明書バンドル。この信頼バンドルは、プロキシーが設定される際にも使用できます。	文字列
compute	コンピュートノードを設定するマシンの設定。	machine-pool オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
compute.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
compute.hyperthreading	コンピュートマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
compute.name	**compute** を使用する場合に必須です。マシンプールの名前。	**worker**
compute.platform	**compute** を使用する場合に必須です。このパラメーターを使用して、ワーカーマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **controlPlane.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
compute.replicas	プロビジョニングするコンピュートマシン (ワーカーマシンとしても知られる) の数。	**2** 以上の正の整数。デフォルト値は **3** です。

パラメーター	説明	値
controlPlane	コントロールプレーンを設定するマシンの設定。	**MachinePool** オブジェクトの配列。詳細は、以下の Machine-pool の表を参照してください。
controlPlane.architecture	プール内のマシンの命令セットアーキテクチャーを決定します。現時点で異種クラスターはサポートされていないため、すべてのプールが同じアーキテクチャーを指定する必要があります。有効な値は **amd64** (デフォルト) です。	文字列
controlPlane.hyperthreading	コントロールプレーンマシンで同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。  **重要** 同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。	**Enabled** または **Disabled**
controlPlane.name	**controlPlane** を使用する場合に必須です。マシンプールの名前。	**master**
controlPlane.platform	**controlPlane** を使用する場合に必須です。このパラメーターを使用して、コントロールプレーンマシンをホストするクラウドプロバイダーを指定します。このパラメーターの値は **compute.platform** パラメーターの値に一致する必要があります。	**aws**、**azure**、**gcp**、**openstack**、**ovirt**、**vsphere**、または **{}**
controlPlane.replicas	プロビジョニングするコントロールプレーンマシンの数。	サポートされる値は **3** のみです (これはデフォルト値です)。

パラメーター	説明	値
fips	FIPS モードを有効または無効にします。デフォルトは **false** (無効) です。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。  **注記** Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。	**false** または **true**
imageContentSources	release-image コンテンツのソースおよびリポジトリー。	オブジェクトの配列。この表の以下の行で説明されているように、**source** およびオプションで **mirrors** が含まれます。
imageContentSources.source	**imageContentSources** を使用する場合に必須です。ユーザーが参照するリポジトリーを指定します (例: イメージプル仕様)。	文字列
imageContentSources.mirrors	同じイメージが含まれる可能性のあるリポジトリーを 1 つ以上指定します。	文字列の配列。
publish	Kubernetes API、OpenShift ルートなどのクラスターのユーザーに表示されるエンドポイントをパブリッシュまたは公開する方法。	**Internal** または **External**。プライベートクラスターをデプロイするには、**publish** を **Internal** に設定します。これはインターネットからアクセスできません。デフォルト値は **External** です。

パラメーター	説明	値
sshKey	クラスターマシンへのアクセスを認証するための SSH キー。  **注記** インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。	たとえば、**sshKey: ssh-ed25519 AAAA..** です。

12.6.9.1.4. 追加の Google Cloud Platform (GCP) 設定パラメーター

追加の GCP 設定パラメーターは以下の表で説明されています。

表12.38 追加の GCP パラメーター

パラメーター	説明	値
platform.gcp.network	クラスターをデプロイする既存 VPC の名前。	文字列。
platform.gcp.type	GCP マシンタイプ。	GCP マシンタイプ。
platform.gcp.zones	インストールプログラムが指定される MachinePool のマシンを作成するアベイラビリティーゾーン。	YAML シーケンス の **us-central1-a** などの有効な GCP アベイラビリティーゾーン の一覧。
platform.gcp.controlPlaneSubnet	コントロールプレーンマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。
platform.gcp.computeSubnet	コンピュートマシンをデプロイする VPC の既存サブネットの名前。	サブネット名。

表12.39 追加の VMware vSphere クラスターパラメーター

パラメーター	説明	値
platform.vsphere.vCenter	vCenter サーバーの完全修飾ホスト名または IP アドレス。	文字列

パラメーター	説明	値
platform.vsphere.username	vCenter インスタンスに接続するために使用するユーザー名。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。	文字列
platform.vsphere.password	vCenter ユーザー名のパスワード。	文字列
platform.vsphere.datacenter	vCenter インスタンスで使用するデータセンターの名前。	文字列
platform.vsphere.defaultDatastore	ボリュームのプロビジョニングに使用するデフォルトデータストアの名前。	文字列
platform.vsphere.folder	**オプション**。インストールプログラムが仮想マシンを作成する既存のフォルダーの絶対パス。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられるフォルダーを作成します。	文字列 (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。
platform.vsphere.network	設定した仮想 IP アドレスおよび DNS レコードが含まれる vCenter インスタンスのネットワーク。	文字列
platform.vsphere.cluster	OpenShift Container Platform クラスターをインストールする vCenter クラスター。	文字列
platform.vsphere.apiVIP	コントロールプレーン API のアクセス用に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。
platform.vsphere.ingressVIP	クラスター Ingress に設定した仮想 IP (VIP) アドレス。	IP アドレス (例: **128.0.0.1**)。

12.6.9.1.5. オプションの VMware vSphere マシンプール設定パラメーター

オプションの VMware vSphere マシンプール設定パラメーターは、以下の表で説明されています。

表12.40 オプションの VMware vSphere マシンプールパラメーター

パラメーター	説明	値
platform.vsphere.clusterOSImage	インストーラーが RHCOS イメージをダウンロードする場所。ネットワークが制限された環境でインストールを実行するには、このパラメーターを設定する必要があります。	HTTP または HTTPS の URL (オプションで SHA-256 形式のチェックサムを使用)。例: **https://mirror.openshift.com/images/rhcos-<version>-vmware.<architecture>.ova**
platform.vsphere.osDisk.diskSizeGB	ディスクのサイズ (ギガバイト単位)。	整数
platform.vsphere.cpus	仮想マシンを割り当てる仮想プロセッサーコアの合計数	整数
platform.vsphere.coresPerSocket	仮想マシンのソケットあたりのコア数。仮想マシンの CPU (vCPU) の数は **platform.vsphere.cpus**/**platform.vsphere.coresPerSocket** になります。デフォルト値は **1** です。	整数
platform.vsphere.memoryMB	仮想マシンのメモリーのサイズ (メガバイト単位)。	整数

12.6.9.2. インストーラーでプロビジョニングされる VMware vSphere クラスターの install-config.yaml ファイルのサンプル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
baseDomain: example.com (1)
compute: (2)
- hyperthreading: Enabled (3)
  name: worker
  replicas: 3
  platform:
    vsphere: (4)
      cpus: 2
      coresPerSocket: 2
      memoryMB: 8196
      osDisk:
        diskSizeGB: 120
controlPlane: (5)
  hyperthreading: Enabled (6)
  name: master
  replicas: 3
  platform:
    vsphere: (7)
```

```
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 120
metadata:
  name: cluster 8
platform:
  vsphere:
    vcenter: your.vcenter.server
    username: username
    password: password
    datacenter: datacenter
    defaultDatastore: datastore
    folder: folder
    network: VM_Network
    cluster: vsphere_cluster_name
    apiVIP: api_vip
    ingressVIP: ingress_vip
    clusterOSImage: http://mirror.example.com/images/rhcos-48.83.202103221318-0-
vmware.x86_64.ova 9
fips: false
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' 10
sshKey: 'ssh-ed25519 AAAA...'
additionalTrustBundle: | 11
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: 12
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

1 クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

2 5 **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

3 6 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 7 オプション: コンピュートおよびコントロールプレーンマシンのマシンプールパラメーターの追加設定を指定します。

8 DNS レコードに指定したクラスター名。

9 bastion サーバーからアクセス可能な Red Hat Enterprise Linux CoreOS (RHCOS) イメージの場所。

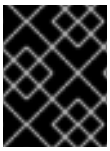
10 **<local_registry>** については、レジストリードメイン名と、ミラーレジストリーがコンテンツを提供するために使用するポートをオプションで指定します。例: **registry.example.com** または **registry.example.com:5000<credentials>** について、ミラーレジストリーの base64 でエンコードされたユーザー名およびパスワードを指定します。

11 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

12 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

12.6.10. クラスターのデプロイ

互換性のあるクラウドプラットフォームに OpenShift Container Platform をインストールできます。



重要

インストールプログラムの **create cluster** コマンドは、初期インストール時に 1 回だけ実行できます。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。

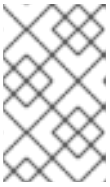
手順

1. クラスターに設定した GCP アカウントのサービスアカウントキーを使用しない既存の GCP 認証情報で、以下の場所に保存されているものを削除します。
   - **GOOGLE_CREDENTIALS**、 **GOOGLE_CLOUD_KEYFILE_JSON**、または **GCLOUD_KEYFILE_JSON** 環境変数
   - **~/.gcp/osServiceAccount.json** ファイル
   - **gcloud cli** デフォルト認証情報
2. インストールプログラムを実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> \ 1
    --log-level=info 2
```

1 **<installation_directory>** については、カスタマイズした **./install-config.yaml** ファイルの場所を指定します。

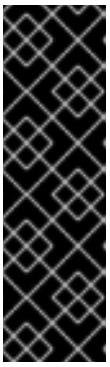
2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

ホストに設定した AWS アカウントにクラスターをデプロイするための十分なパーミッションがない場合、インストールプログラムは停止し、不足しているパーミッションが表示されます。

クラスターのデプロイメントが完了すると、Web コンソールへのリンクや **kubeadmin** ユーザーの認証情報を含む、クラスターにアクセスするための指示がターミナルに表示されます。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

重要

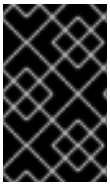
インストールプログラム、またはインストールプログラムが作成するファイルを削除することはできません。これらはいずれもクラスターを削除するために必要になります。

3. オプション: クラスターをインストールするために使用したサービスアカウントのパーミッションの数を減らすことができます。

   - **Owner** ロールをサービスアカウントに割り当てている場合、 そのロールを削除し、これを **Viewer** ロールに置き換えることができます。
   - **Service Account Key Admin** ロールが含まれている場合は、これを削除することができます。

12.6.11. バイナリーのダウンロードによる CLI のインストール

コマンドラインインターフェイスを使用して OpenShift Container Platform と対話するために CLI (**oc**) をインストールすることができます。**oc** は Linux、Windows、または macOS にインストールできます。



重要

以前のバージョンの **oc** をインストールしている場合、これを使用して OpenShift Container Platform 4.5 のすべてのコマンドを実行することはできません。新規バージョンの **oc** をダウンロードし、インストールします。

12.6.11.1. Linux への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Linux にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Linux** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開します。

```
$ tar xvzf <file>
```

5. **oc** バイナリーを、**PATH** にあるディレクトリーに配置します。
PATH を確認するには、以下のコマンドを実行します。

```
$ echo $PATH
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.6.11.2. Windows での CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを Windows にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **Windows** を選択し、**Download command-line tools** をクリックします。
4. ZIP プログラムでアーカイブを解凍します。
5. **oc** バイナリーを、**PATH** にあるディレクトリーに移動します。
PATH を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
C:\> oc <command>
```

12.6.11.3. macOS への CLI のインストール

以下の手順を使用して、OpenShift CLI (**oc**) バイナリーを macOS にインストールできます。

手順

1. Red Hat OpenShift Cluster Manager サイトの Infrastructure Provider ページに移動します。
2. インフラストラクチャープロバイダーを選択し、(該当する場合は) インストールタイプを選択します。
3. **Command-line interface** セクションで、ドロップダウンメニューの **MacOS** を選択し、**Download command-line tools** をクリックします。
4. アーカイブを展開し、解凍します。
5. **oc** バイナリーをパスにあるディレクトリーに移動します。
   PATH を確認するには、ターミナルを開き、以下のコマンドを実行します。

   ```
   $ echo $PATH
   ```

CLI のインストール後は、**oc** コマンドを使用して利用できます。

```
$ oc <command>
```

12.6.12. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバーに接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

   ```
   $ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
   ```

   1. **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

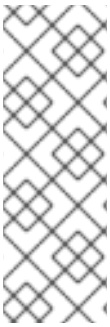
12.6.13. レジストリーストレージの作成

クラスターのインストール後に、レジストリー Operator のストレージを作成する必要があります。

12.6.13.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift イメージレジストリー Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、イメージレジストリー Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。



注記

Prometheus コンソールは、以下のような **ImageRegistryRemoved** アラートを提供します。

"Image Registry has been removed.**ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected.ストレージを設定して、configs.imageregistry.operator.openshift.io を編集して設定を **Managed** 状態に更新してください。

12.6.13.2. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

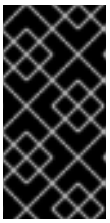
12.6.13.2.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。

- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.6.14. 次のステップ

- クラスターをカスタマイズ します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。
- レジストリーをセットアップし、レジストリーストレージを設定 します。

12.7. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワークが制限された環境での VSPHERE へのクラスターのインストール

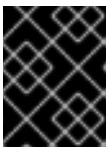
OpenShift Container Platform バージョン 4.5 では、クラスターを制限されたネットワークでプロビジョニングする VMware vSphere インフラストラクチャーにインストールできます。

重要

ユーザーによってプロビジョニングされるインフラストラクチャーのインストールする手順は、例としてのみ提供されます。独自にプロビジョニングするインフラストラクチャーでクラスターをインストールするには、vSphere プラットフォームおよび OpenShift Container Platform のインストールプロセスについて理解している必要があります。ユーザーによってプロビジョニングされるインフラストラクチャーのインストール手順をガイドとして使用します。他の方法で必要なリソースを作成することもできます。

12.7.1. 前提条件

- ミラーホストでレジストリーを作成 し、OpenShift Container Platform の使用しているバージョン用の **imageContentSources** データを取得します。



重要

インストールメディアはミラーホストにあるため、そのコンピューターを使用してすべてのインストール手順を完了します。

- クラスターの 永続ストレージ をプロビジョニングします。プライベートイメージレジストリーをデプロイするには、ストレージで **ReadWriteMany** アクセスモードを指定する必要があります。
- OpenShift Container Platform のインストールおよび更新 プロセスについての詳細を確認します。
- ファイアウォールを使用し、Telemetry を使用する予定がある場合は、クラスターがアクセスする必要のある サイトを許可するようにファイアウォールを設定 する必要があります。

注記

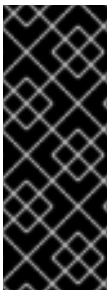
プロキシーを設定する場合は、このサイト一覧も確認してください。

12.7.2. ネットワークが制限された環境でのインストールについて

OpenShift Container Platform 4.5 では、ソフトウェアコンポーネントを取得するためにインターネットへのアクティブな接続を必要としないインストールを実行できます。ネットワークが制限された環境のインストールは、クラスターのインストール先となるクラウドプラットフォームに応じて、インストーラーでプロビジョニングされるインフラストラクチャーまたはユーザーによってプロビジョニングされるインフラストラクチャーを使用して実行できます。

クラウドプラットフォーム上でネットワークが制限されたインストールの実行を選択した場合でも、そのクラウド API へのアクセスが必要になります。Amazon Web Service の IAM サービスなどの一部のクラウド機能はインターネットアクセスを必要とするため、インターネットアクセスが依然として必要になる場合があります。ネットワークによっては、ベアメタルハードウェアまたは VMware vSphere へのインストールには、インターネットアクセスが必要になる場合があります。

ネットワークが制限されたインストールを完了するには、OpenShift Container Platform レジストリーのコンテンツをミラーリングし、インストールメディアを含むレジストリーを作成する必要があります。このミラーは、インターネットと制限されたネットワークの両方にアクセスできるミラーホストで、または制限に対応する他の方法を使用して作成できます。



重要

ユーザーによってプロビジョニングされるインストールの設定は複雑であるため、ユーザーによってプロビジョニングされるインフラストラクチャーを使用してネットワークが制限されたインストールを試行する前に、標準的なユーザーによってプロビジョニングされるインフラストラクチャーを実行することを検討してください。このテストが完了すると、ネットワークが制限されたインストール時に発生する可能性のある問題の切り分けやトラブルシューティングがより容易になります。

12.7.2.1. その他の制限

ネットワークが制限された環境のクラスターには、以下の追加の制限および制約があります。

- **ClusterVersion** ステータスには **Unable to retrieve available updates** エラーが含まれます。
- デフォルトで、開発者カタログのコンテンツは、必要とされるイメージストリームタグにアクセスできないために使用できません。

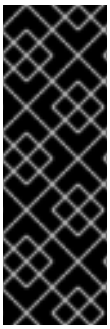
12.7.3. OpenShift Container Platform のインターネットアクセスおよび Telemetry アクセス

OpenShift Container Platform 4.5 では、クラスターをインストールするために必要なイメージを取得するために、インターネットアクセスが必要になります。クラスターの健全性および正常に実行された更新についてのメトリクスを提供するためにデフォルトで実行される Telemetry サービスにもインターネットアクセスが必要です。クラスターがインターネットに接続されている場合、Telemetry は自動的に実行され、クラスターは Red Hat OpenShift Cluster Manager (OCM) に登録されます。

Red Hat OpenShift Cluster Manager インベントリーが Telemetry によって自動的に維持されるか、または OCM を手動で使用しているかのいずれによって正常であることを確認した後に、subscription watch を使用して、アカウントまたはマルチクラスターレベルで OpenShift Container Platform サブスクリプションを追跡します。

インターネットへのアクセスは以下を実行するために必要です。

- Red Hat OpenShift Cluster Manager ページにアクセスし、インストールプログラムをダウンロードし、サブスクリプション管理を実行します。クラスターにインターネットアクセスがあり、Telemetry を無効にしない場合、そのサービスは有効なサブスクリプションでクラスターを自動的に使用します。
- クラスターのインストールに必要なパッケージを取得するために Quay.io にアクセスします。
- クラスターの更新を実行するために必要なパッケージを取得します。



重要

クラスターでインターネットに直接アクセスできない場合、プロビジョニングする一部のタイプのインフラストラクチャーでネットワークが制限されたインストールを実行できます。このプロセスで、必要なコンテンツをダウンロードし、これを使用してミラーレジストリーにクラスターのインストールおよびインストールプログラムの生成に必要なパッケージを設定します。インストールタイプによっては、クラスターのインストール環境でインターネットアクセスが不要となる場合があります。クラスターを更新する前に、ミラーレジストリーのコンテンツを更新します。

12.7.4. VMware vSphere インフラストラクチャーの要件

使用するコンポーネントの要件を満たす VMware vSphere バージョン 6 インスタンスに OpenShift Container Platform クラスターをインストールする必要があります。

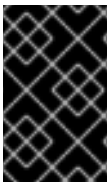
表12.41 VMware コンポーネントのサポートされる vSphere の最小バージョン

コンポーネント	サポートされる最小バージョン	説明
ハイパーバイザー	vSphere 6.5 および HW バージョン 13	このバージョンは、Red Hat Enterprise Linux CoreOS (RHCOS) がサポートする最小バージョンです。Red Hat Enterprise Linux 8 でサポートされるハイパーバイザーの一覧 を参照してください。
ネットワーク (NSX-T)	vSphere 6.5U3 または vSphere 6.7U2 以降	OpenShift Container Platform には vSphere 6.5U3 または vSphere 6.7U2+ が必要です。VMware の NSX Container Plug-in (NCP) 3.0.2 は OpenShift Container Platform 4.5 および NSX-T 3.x+ で認定されています。
ストレージおよび In-tree ドライバー	vSphere 6.5 以降	このプラグインは、OpenShift Container Platform に含まれる vSphere の In-tree ストレージドライバーを使用して vSphere ストレージを作成します。

vSphere バージョン 6.5 インスタンスを使用している場合は、OpenShift Container Platform をインストールする前に 6.7U2 にアップグレードすることを検討してください。

重要

OpenShift Container Platform をインストールする前に、ESXi ホストの時間が同期されていることを確認する必要があります。VMware ドキュメントの Edit Time Configuration for a Host を参照してください。



重要

VPC の使用の制限とは、Storage Distributed Resource Scheduler (SDRS) がサポートされないことを意味します。VMware ドキュメントの vSphere Storage for Kubernetes FAQs を参照してください。

12.7.5. ユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのマシン要件

ユーザーによってプロビジョニングされるインフラストラクチャーを含むクラスターの場合、必要なマシンすべてをデプロイする必要があります。

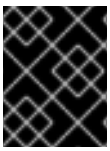
12.7.5.1. 必要なマシン

最小の OpenShift Container Platform クラスターでは以下のホストが必要です。

- 1 つの一時的なブートストラップマシン
- 3 つのコントロールプレーン、またはマスター、マシン
- 少なくとも 2 つのコンピュートマシン (ワーカーマシンとしても知られる)。

注記

クラスターでは、ブートストラップマシンが OpenShift Container Platform クラスターを 3 つのコントロールプレーンマシンにデプロイする必要があります。クラスターのインストール後にブートストラップマシンを削除できます。

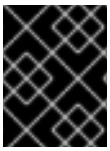


重要

クラスターの高可用性を維持するには、これらのクラスターマシンについて別個の物理ホストを使用します。

ブートストラップおよびコントロールプレーンマシンでは、Red Hat Enterprise Linux CoreOS (RHCOS) をオペレーティングシステムとして使用する必要があります。

RHCOS は Red Hat Enterprise Linux (RHEL) 8 をベースとしており、そのハードウェア認定および要件が継承されることに注意してください。Red Hat Enterprise Linux テクノロジーの機能と制限 を参照してください。



重要

すべての仮想マシンは、インストーラーと同じデータストアおよびフォルダーになければなりません。

12.7.5.2. ネットワーク接続の要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定ファイルをフェッチする必要があります。初回の起動時に、Ignition 設定ファイルをダウンロードできるようネットワーク接続を確立するために、マシンには DHCP サーバーまたはその静的 IP アドレスが設定されている必要になります。さらに、クラスター内の各 OpenShift Container Platform ノードは Network Time Protocol (NTP) サーバーにアクセスできる必要があります。DHCP サーバーが NTP サーバー情報を提供する場合、Red Hat Enterprise Linux CoreOS (RHCOS) マシンの chrony タイムサービスは情報を読み取り、NTP サーバーとクロックを同期できます。

12.7.5.3. 最小リソース要件

それぞれのクラスターマシンは、以下の最小要件を満たしている必要があります。

マシン	オペレーティングシステム	vCPU [1]	仮想 RAM	ストレージ
ブートストラップ	RHCOS	4	16 GB	120 GB
コントロールプレーン	RHCOS	4	16 GB	120 GB
コンピュート	RHCOS または RHEL 7.8 - 7.9	2	8 GB	120 GB

1. 1 vCPU は、同時マルチスレッド (SMT) またはハイパースレッディングが有効にされていない場合に 1 つの物理コアと同等です。これが有効にされている場合、以下の数式を使用して対応する比率を計算します: (コアごとのスレッド × コア数) × ソケット数 = vCPU

12.7.5.4. 証明書署名要求の管理

ユーザーがプロビジョニングするインフラストラクチャーを使用する場合、クラスターの自動マシン管理へのアクセスは制限されるため、インストール後にクラスターの証明書署名要求 (CSR) のメカニズムを提供する必要があります。**kube-controller-manager** は kubelet クライアント CSR のみを承認します。**machine-approver** は、kubelet 認証情報を使用して要求される提供証明書の有効性を保証できません。適切なマシンがこの要求を発行したかどうかを確認できないためです。kubelet 提供証明書の要求の有効性を検証し、それらを承認する方法を判別し、実装する必要があります。

12.7.6. ユーザーによってプロビジョニングされるインフラストラクチャーの作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターをデプロイする前に、基礎となるインフラストラクチャーを作成する必要があります。

前提条件

- クラスターでサポートするインフラストラクチャーを作成する前に、OpenShift Container Platform 4.x のテスト済みインテグレーション ページを参照してください。

手順

1. 各ノードに DHCP を設定するか、または静的 IP アドレスを設定します。
2. 必要なロードバランサーをプロビジョニングします。
3. マシンのポートを設定します。
4. DNS を設定します。
5. ネットワーク接続を確認します。

12.7.6.1. ユーザーによってプロビジョニングされるインフラストラクチャーのネットワーク要件

すべての Red Hat Enterprise Linux CoreOS (RHCOS) マシンでは、起動時に **initramfs** のネットワークがマシン設定サーバーから Ignition 設定をフェッチする必要があります。

マシン間のネットワーク接続を、クラスターのコンポーネントが通信できるように設定する必要があります。すべてのマシンではクラスターの他のすべてのマシンのホスト名を解決できる必要があります。

表12.42 すべてのマシンに対応するすべてのマシン

プロトコル	ポート	説明
ICMP	該当なし	ネットワーク到達性のテスト
TCP	**1936**	メトリクス
	9000-**9999**	ホストレベルのサービス。 ポート **9100**-**9101** のノードエクスポーター、ポート **9099** の Cluster Version Operator が含まれます。
	10250-**10259**	Kubernetes が予約するデフォルトポート
	10256	openshift-sdn
UDP	**4789**	VXLAN および Geneve
	6081	VXLAN および Geneve
	9000-**9999**	ポート **9100**-**9101** のノードエクスポーターを含む、ホストレベルのサービス。
TCP/UDP	**30000**-**32767**	Kubernetes ノードポート

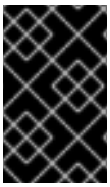
表12.43 コントロールプレーンへのすべてのマシン

プロトコル	ポート	説明
TCP	**6443**	Kubernetes API

表12.44 コントロールプレーンマシンへのコントロールプレーンマシン

プロトコル	ポート	説明
TCP	**2379-2380**	etcd サーバーおよびピアポート

ネットワークトポロジー要件
クラスター用にプロビジョニングするインフラストラクチャーは、ネットワークトポロジーの以下の要件を満たす必要があります。



重要

OpenShift Container Platform では、すべてのノードが、プラットフォームコンテナーのイメージをプルし、Telemetry データを Red Hat に提供するためにインターネットへの直接のアクセスが必要です。

ロードバランサー
OpenShift Container Platform をインストールする前に、以下の要件を満たす 2 つのロードバランサーをプロビジョニングする必要があります。

1. **API ロードバランサー**: プラットフォームと対話およびプラットフォームを設定するためのユーザー向けの共通のエンドポイントを提供します。以下の条件を設定します。
   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、API ルートの Server Name Indication (SNI) を有効にする必要があります。
   - ステートレス負荷分散アルゴリズム。オプションは、ロードバランサーの実装によって異なります。

注記

API ロードバランサーが適切に機能するには、セッション永続性は必要ありません。

ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.45 API ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
6443	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。API サーバーのヘルスチェックプローブの **/readyz** エンドポイントを設定する必要があります。	X	X	Kubernetes API サーバー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
22623	ブートストラップおよびコントロールプレーン。ブートストラップマシンがクラスターのコントロールプレーンを初期化した後に、ブートストラップマシンをロードバランサーから削除します。	X		マシン設定サーバー

注記

ロードバランサーは、API サーバーが **/readyz** エンドポイントをオフにしてからプールから API サーバーインスタンスを削除するまで最大 30 秒かかるように設定する必要があります。**/readyz** の後の時間枠内でエラーが返されたり、正常になったりする場合は、エンドポイントが削除または追加されているはずです。5 秒または 10 秒ごとにプローブし、2 つの正常な要求が正常な状態になり、3 つの要求が正常な状態になりません。これらは十分にテストされた値です。

2. **Application Ingress ロードバランサー**: クラスター外から送られるアプリケーショントラフィックの Ingress ポイントを提供します。以下の条件を設定します。

   - Layer 4 の負荷分散のみ。これは、Raw TCP、SSL パススルー、または SSL ブリッジモードと呼ばれます。SSL ブリッジモードを使用する場合は、Ingress ルートの Server Name Indication (SNI) を有効にする必要があります。
   - 選択可能なオプションやプラットフォーム上でホストされるアプリケーションの種類に基づいて、接続ベースの永続化またはセッションベースの永続化が推奨されます。

   ロードバランサーのフロントとバックの両方で以下のポートを設定します。

表12.46 アプリケーション Ingress ロードバランサー

ポート	バックエンドマシン (プールメンバー)	内部	外部	説明
443	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTPS トラフィック
80	デフォルトで Ingress ルーター Pod、コンピュート、またはワーカーを実行するマシン。	X	X	HTTP トラフィック

ヒント

クライアントの実際の IP アドレスがロードバランサーによって確認できる場合、ソースの IP ベースのセッション永続化を有効にすると、エンドツーエンドの TLS 暗号化を使用するアプリケーションのパフォーマンスを強化できます。

注記

Ingress ルーターの作業用の設定が OpenShift Container Platform クラスターに必要です。コントロールプレーンの初期化後に Ingress ルーターを設定する必要があります。

Ethernet アダプターのハードウェアアドレス要件
クラスターの仮想マシンをプロビジョニングする場合、各仮想マシンに設定されたイーサネットインターフェイスは VMware Organizationally Unique Identifier (OUI) 割り当て範囲から MAC アドレスを使用する必要があります。

- **00:05:69:00:00:00** - **00:05:69:FF:FF:FF**
- **00:0c:29:00:00:00** - **00:0c:29:FF:FF:FF**
- **00:1c:14:00:00:00** - **00:1c:14:FF:FF:FF**
- **00:50:56:00:00:00** - **00:50:56:FF:FF:FF**

VMware OUI 外の MAC アドレスが使用される場合、クラスターのインストールは成功しません。

関連情報

- chrony タイムサービスの設定

12.7.6.2. ユーザーによってプロビジョニングされるインフラストラクチャーの DNS 要件

DNS は、名前解決および逆引き名前解決に使用されます。DNS A/AAAA または CNAME レコードは名前解決に使用され、PTR レコードは逆引き名前解決に使用されます。逆引きレコードは、Red Hat Enterprise Linux CoreOS (RHCOS) は逆引きレコードを使用してすべてのノードのホスト名を設定するために重要です。さらに、逆引きレコードは、OpenShift Container Platform が動作するために必要な証明書署名要求 (CSR) を生成するために使用されます。

以下の DNS レコードは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform クラスターに必要です。各レコードで、 **<cluster_name>** はクラスター名で、**<base_domain>** は、**install-config.yaml** ファイルに指定するクラスターのベースドメインです。完全な DNS レコードは **<component>.<cluster_name>.<base_domain>.** の形式を取ります。

表12.47 必要な DNS レコード

コンポーネント	レコード	説明
Kubernetes API	**api.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。

コンポーネント	レコード	説明
	api-int.<cluster_name>.<base_domain>.	DNS A/AAAA または CNAME レコード、および DNS PTR レコードを、コントロールプレーンマシンのロードバランサーを特定するために追加します。これらのレコードは、クラスター内のすべてのノードで解決できる必要があります。  **重要** API サーバーは、 Kubernetes に記録されるホスト名でワーカーノードを解決できる必要があります。API サーバーがノード名を解決できない場合、プロキシーされる API 呼び出しが失敗し、Pod からログを取得できなくなる可能性があります。
ルート	***.apps.<cluster_name>.<base_domain>.**	デフォルトでワーカーノードの Ingress ルーター Pod を実行するマシンをターゲットにするロードバランサーを参照するワイルドカード DNS A/AAAA または CNAME レコードを追加します。これらのレコードは、クラスター外のクライアントおよびクラスター内のすべてのノードで解決できる必要があります。
ブートストラップ	**bootstrap.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ブートストラップマシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
マスターホスト	**<master><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、マスターノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。
ワーカーホスト	**<worker><n>.<cluster_name>.<base_domain>.**	DNS A/AAAA または CNAME レコードおよび DNS PTR レコードを、ワーカーノードの各マシンを特定するために追加します。これらのレコードは、クラスター内のノードで解決できる必要があります。

ヒント

nslookup <hostname> コマンドを使用して、名前解決を確認することができます。**dig -x <ip_address>** コマンドを使用して、PTR レコードの逆引き名前解決を確認できます。

BIND ゾーンファイルの以下の例は、名前解決の A レコードの例を示しています。この例の目的は、必要なレコードを表示することです。この例では、特定の名前解決サービスを選択するためのアドバイスを提供することを目的としていません。

例12.13 DNS ゾーンデータベースのサンプル

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
 IN MX 10 smtp.example.com.
;
;
ns1 IN A 192.168.1.5
smtp IN A 192.168.1.5
;
helper IN A 192.168.1.5
helper.ocp4 IN A 192.168.1.5
;
; The api identifies the IP of your load balancer.
api.ocp4  IN A 192.168.1.5
api-int.ocp4  IN A 192.168.1.5
;
; The wildcard also identifies the load balancer.
*.apps.ocp4  IN A 192.168.1.5
;
; Create an entry for the bootstrap host.
bootstrap.ocp4 IN A 192.168.1.96
;
; Create entries for the master hosts.
master0.ocp4  IN A 192.168.1.97
master1.ocp4  IN A 192.168.1.98
master2.ocp4  IN A 192.168.1.99
;
; Create entries for the worker hosts.
worker0.ocp4  IN A 192.168.1.11
worker1.ocp4  IN A 192.168.1.7
;
;EOF
```

以下の BIND ゾーンファイルの例では、逆引き名前解決の PTR レコードの例を示しています。

例12.14 逆引きレコードの DNS ゾーンデータベースの例

```
$TTL 1W
@ IN SOA ns1.example.com. root (
   2019070700 ; serial
   3H  ; refresh (3 hours)
   30M  ; retry (30 minutes)
   2W  ; expiry (2 weeks)
   1W )  ; minimum (1 week)
 IN NS ns1.example.com.
;
; The syntax is "last octet" and the host must have an FQDN
; with a trailing dot.
97 IN PTR master0.ocp4.example.com.
```

```
98 IN PTR master1.ocp4.example.com.
99 IN PTR master2.ocp4.example.com.
;
96 IN PTR bootstrap.ocp4.example.com.
;
5 IN PTR api.ocp4.example.com.
5 IN PTR api-int.ocp4.example.com.
;
11 IN PTR worker0.ocp4.example.com.
7 IN PTR worker1.ocp4.example.com.
;
;EOF
```

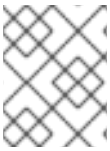
12.7.7. SSH プライベートキーの生成およびエージェントへの追加

クラスターでインストールのデバッグまたは障害復旧を実行する必要がある場合、**ssh-agent** とインストールプログラムの両方に SSH キーを指定する必要があります。このキーを使用してパブリッククラスターのブートストラップマシンにアクセスし、インストールの問題をトラブルシューティングできます。

注記

実稼働環境では、障害復旧およびデバッグが必要です。

このキーを使用して、ユーザー **core** としてマスターノードに対して SSH を実行できます。クラスターをデプロイする際に、キーは **core** ユーザーの ~/**.ssh/authorized_keys** 一覧に追加されます。



注記

AWS キーペア などのプラットフォームに固有の方法で設定したキーではなく、ローカルキーを使用する必要があります。

手順

1. パスワードなしの認証に設定されている SSH キーがコンピューター上にない場合は、これを作成します。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ ssh-keygen -t ed25519 -N '' \
  -f <path>/<file_name> 1
```

1 ~/**.ssh/id_rsa** などの、新規 SSH キーのパスおよびファイル名を指定します。既存のキーペアがある場合は、公開鍵が ~/**.ssh** ディレクトリーにあることを確認します。

このコマンドを実行すると、指定した場所にパスワードを必要としない SSH キーが生成されます。

注記

FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターを **x86_64** アーキテクチャーにインストールする予定の場合は、**ed25519** アルゴリズムを使用するキーは作成しないでください。代わりに、**rsa** アルゴリズムまたは **ecdsa** アルゴリズムを使用するキーを作成します。

2. **ssh-agent** プロセスをバックグラウンドタスクとして開始します。

```
$ eval "$(ssh-agent -s)"
```

出力例

```
Agent pid 31874
```

クラスターが FIPS モードにある場合は、FIPS 準拠のアルゴリズムのみを使用して SSH キーを生成します。鍵は RSA または ECDSA のいずれかである必要があります。

1. SSH プライベートキーを **ssh-agent** に追加します。

```
$ ssh-add <path>/<file_name> 1
```

出力例

```
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

1 **~/.ssh/id_rsa** などの、SSH プライベートキーのパスおよびファイル名を指定します。

2. **GOOGLE_APPLICATION_CREDENTIALS** 環境変数をサービスアカウントのプライベートキーファイルへのフルパスに設定します。

```
$ export GOOGLE_APPLICATION_CREDENTIALS="<your_service_account_file>"
```

3. 認証情報が適用されていることを確認します。

```
$ gcloud auth list
```

次のステップ

- OpenShift Container Platform をインストールする際に、SSH パブリックキーをインストールプログラムに指定します。クラスターを独自にプロビジョニングするインフラストラクチャーにインストールする場合は、このキーをクラスターのマシンに指定する必要があります。

12.7.8. インストール設定ファイルの手動作成

ユーザーによってプロビジョニングされるインフラストラクチャーを使用する OpenShift Container Platform のインストールでは、インストール設定ファイルを手動で生成します。

前提条件

- OpenShift Container Platform インストーラープログラムおよびクラスターのアクセストークンを取得します。
- リポジトリーのミラーリングに使用するコマンドの出力で **imageContentSources** セクションを取得します。
- ミラーレジストリーの証明書の内容を取得します。

手順

1. 必要なインストールアセットを保存するためのインストールディレクトリーを作成します。

```
$ mkdir <installation_directory>
```

重要

ディレクトリーを作成する必要があります。ブートストラップ X.509 証明書などの一部のインストールアセットの有効期限は短く設定されているため、インストールディレクトリーを再利用することができません。別のクラスターインストールの個別のファイルを再利用する必要がある場合は、それらをディレクトリーにコピーすることができます。ただし、インストールアセットのファイル名はリリース間で変更される可能性があります。インストールファイルを以前のバージョンの OpenShift Container Platform からコピーする場合は注意してコピーを行ってください。

2. 以下の **install-config.yaml** ファイルテンプレートをカスタマイズし、これを **<installation_directory>** に保存します。

注記

この設定ファイル **install-config.yaml** に名前を付ける必要があります。

- **docker.io** などの、RHCOS がデフォルトで信頼するレジストリーを使用しない限り、**additionalTrustBundle** セクションにミラーリポジトリーの証明書の内容を指定する必要があります。ほとんどの場合、ミラーの証明書を指定する必要があります。
- リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを組み込む必要があります。

3. **install-config.yaml** ファイルをバックアップし、複数のクラスターをインストールするのに使用できるようにします。

重要

install-config.yaml ファイルは、インストールプロセスの次の手順で使用されます。この時点でこれをバックアップする必要があります。

12.7.8.1. VMware vSphere のサンプル install-config.yaml ファイル

install-config.yaml ファイルをカスタマイズして、OpenShift Container Platform クラスターのプラットフォームについての詳細を指定するか、または必要なパラメーターの値を変更することができます。

```
apiVersion: v1
```

```
baseDomain: example.com (1)
compute:
- hyperthreading: Enabled (2) (3)
  name: worker
  replicas: 0 (4)
controlPlane:
  hyperthreading: Enabled (5) (6)
  name: master
  replicas: 3 (7)
metadata:
  name: test (8)
platform:
  vsphere:
    vcenter: your.vcenter.server (9)
    username: username (10)
    password: password (11)
    datacenter: datacenter (12)
    defaultDatastore: datastore (13)
    folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>" (14)
fips: false (15)
pullSecret: '{"auths":{"<local_registry>": {"auth": "<credentials>","email": "you@example.com"}}}' (16)
sshKey: 'ssh-ed25519 AAAA...' (17)
additionalTrustBundle: | (18)
  -----BEGIN CERTIFICATE-----
  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
  -----END CERTIFICATE-----
imageContentSources: (19)
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - <local_registry>/<local_repository_name>/release
  source: registry.svc.ci.openshift.org/ocp/release
```

(1) クラスターのベースドメイン。すべての DNS レコードはこのベースのサブドメインである必要があり、クラスター名が含まれる必要があります。

(2) (5) **controlPlane** セクションは単一マッピングですが、コンピュートセクションはマッピングのシーケンスになります。複数の異なるデータ構造の要件を満たすには、 **compute** セクションの最初の行はハイフン **-** で始め、**controlPlane** セクションの最初の行はハイフンで始めることができません。どちらのセクションも、現時点では単一のマシンプールを定義しますが、OpenShift Container Platform の今後のバージョンでは、インストール時の複数のコンピュートプールの定義をサポートする可能性があります。1 つのコントロールプレーンプールのみが使用されます。

(3) (6) 同時マルチスレッドまたは **hyperthreading** を有効/無効にするかどうか。デフォルトでは、同時スレッドはマシンのコアのパフォーマンスを上げるために有効にされます。パラメーター値を **Disabled** に設定するとこれを無効にすることができます。一部のクラスターマシンで同時マルチスレッドを無効にする場合は、これをすべてのクラスターマシンで無効にする必要があります。

重要

同時スレッドを無効にする場合は、容量計画においてマシンパフォーマンスの大幅な低下が考慮に入れられていることを確認します。同時マルチスレッドを無効にする場合は、マシンで最低でも 8 CPU および 32 GB の RAM を使用する必要があります。

4 **replicas** パラメーターの値を **0** に設定する必要があります。このパラメーターはクラスターが作成し、管理するワーカーの数を制御します。これは、ユーザーによってプロビジョニングされるインフラストラクチャーを使用する場合にクラスターが実行しない機能です。OpenShift Container Platform のインストールが終了する前に、クラスターが使用するワーカーマシンを手動でデプロイする必要があります。

7 クラスターに追加するコントロールプレーンマシンの数。クラスターをこの値をクラスターの etcd エンドポイント数として使用するため、値はデプロイするコントロールプレーンマシンの数に一致する必要があります。

8 DNS レコードに指定したクラスター名。

9 vCenter サーバーの完全修飾ホスト名または IP アドレス。

10 サーバーにアクセスするユーザーの名前。このユーザーには、少なくとも vSphere の 静的または動的な永続ボリュームのプロビジョニング に必要なロールおよび権限がなければなりません。

11 vSphere ユーザーに関連付けられたパスワード。

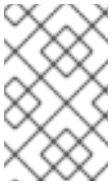
12 vSphere データセンター。

13 使用するデフォルトの vSphere データストア。

14 オプション: インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムが仮想マシンを作成する既存フォルダーの絶対パス (例: **/<datacenter_name>/vm/<folder_name>/<subfolder_name>**)。この値を指定しない場合、インストールプログラムは、データセンターの仮想マシンフォルダーにインフラストラクチャー ID を使用して名前が付けられる上位レベルのフォルダーを作成します。クラスターのインフラストラクチャーを提供する場合は、このパラメーターを省略します。

15 FIPS モードを有効または無効にするかどうか。デフォルトでは、FIPS モードは有効にされません。FIPS モードが有効にされている場合、OpenShift Container Platform が実行される Red Hat Enterprise Linux CoreOS (RHCOS) マシンがデフォルトの Kubernetes 暗号スイートをバイパスし、代わりに RHCOS で提供される暗号モジュールを使用します。

16 Red Hat Enterprise Linux CoreOS (RHCOS) の **core** ユーザーのデフォルト SSH キーの公開部分。



注記

インストールのデバッグまたは障害復旧を実行する必要のある実稼働用の OpenShift Container Platform クラスターでは、**ssh-agent** プロセスが使用する SSH キーを指定します。

17 ミラーレジストリーに使用した証明書ファイルの内容を指定します。

18 リポジトリーのミラーリングに使用するコマンドの出力の **imageContentSources** セクションを指定します。

12.7.8.2. インストール時のクラスター全体のプロキシーの設定

実稼働環境では、インターネットへの直接アクセスを拒否し、代わりに HTTP または HTTPS プロキシーを使用することができます。プロキシー設定を **install-config.yaml** ファイルで行うことにより、新規の OpenShift Container Platform クラスターをプロキシーを使用するように設定できます。

前提条件

- 既存の **install-config.yaml** ファイルが必要です。
- クラスターがアクセスする必要のあるサイトを確認し、プロキシーをバイパスする必要があるかどうかを判別します。デフォルトで、すべてのクラスター egress トラフィック (クラスターをホストするクラウドについてのクラウドプロバイダー API に対する呼び出しを含む) はプロキシーされます。**Proxy** オブジェクトの **spec.noProxy** フィールドにサイトを追加し、必要に応じてプロキシーをバイパスします。

注記

Proxy オブジェクトの **status.noProxy** フィールドには、インストール設定の **networking.machineNetwork[].cidr**、 **networking.clusterNetwork[].cidr**、および **networking.serviceNetwork[]** フィールドの値が設定されます。

Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Red Hat OpenStack Platform (RHOSP) へのインストールの場合、**Proxy** オブジェクトの **status.noProxy** フィールドには、インスタンスメタデータのエンドポイント (**169.254.169.254**) も設定されます。

手順

1. **install-config.yaml** ファイルを編集し、プロキシー設定を追加します。以下に例を示します。

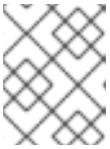
```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: http://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
...
```

1. クラスター外の HTTP 接続を作成するために使用するプロキシー URL。URL スキームは **http** である必要があります。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpProxy** 値を指定することはできません。

2. クラスター外で HTTPS 接続を作成するために使用するプロキシー URL。このフィールドが指定されていない場合、HTTP および HTTPS 接続の両方に **httpProxy** が使用されます。追加のプロキシー設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシーネットワークを使用する場合には、**httpsProxy** 値を指定することはできません。

3. プロキシーを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネット

4 指定されている場合、インストールプログラムは HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書が含まれる **user-ca-bundle** という名前の設定マップを

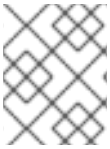


注記

インストールプログラムは、プロキシーの **readinessEndpoints** フィールドをサポートしません。

2. ファイルを保存し、OpenShift Container Platform のインストール時にこれを参照します。

インストールプログラムは、指定の **install-config.yaml** ファイルのプロキシー設定を使用する **cluster** という名前のクラスター全体のプロキシーを作成します。プロキシー設定が指定されていない場合、**cluster Proxy** オブジェクトが依然として作成されますが、これには **spec** がありません。

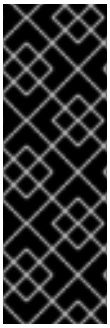


注記

cluster という名前の **Proxy** オブジェクトのみがサポートされ、追加のプロキシーを作成することはできません。

12.7.9. Kubernetes マニフェストおよび Ignition 設定ファイルの作成

一部のクラスター定義ファイルを変更し、クラスターマシンを手動で起動する必要があるため、クラスターがマシンを作成するために必要な Kubernetes マニフェストと Ignition 設定ファイルを生成する必要があります。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

前提条件

- OpenShift Container Platform インストールプログラムを取得します。ネットワークが制限されたインストールでは、これらのファイルがミラーホスト上に置かれます。
- **install-config.yaml** インストール設定ファイルを作成します。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory> 1
```

出力例

```
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for
Scheduler cluster settings
```

1 **<installation_directory>** については、作成した **install-config.yaml** ファイルが含まれるインストールディレクトリーを指定します。

インストールプロセスの後の部分で独自のコンピュートマシンを作成するため、この警告を無視しても問題がありません。

2. コントロールプレーンマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_master-machines-*.yaml
```

これらのファイルを削除することで、クラスターがコントロールプレーンマシンを自動的に生成するのを防ぐことができます。

3. オプション: クラスターでコンピュートマシンをプロビジョニングする必要がない場合は、ワーカーマシンを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f <installation_directory>/openshift/99_openshift-cluster-api_worker-machineset-*.yaml
```

ワーカーマシンは独自に作成し、管理するため、これらのマシンを初期化する必要はありません。

4. コントロールプレーンマシンおよびコンピュートマシンセットを定義する Kubernetes マニフェストファイルを削除します。

```
$ rm -f openshift/99_openshift-cluster-api_master-machines-*.yaml openshift/99_openshift-
cluster-api_worker-machineset-*.yaml
```

これらのリソースを独自に作成および管理するため、それらを初期化する必要はありません。

- マシンセットファイルを保存して、マシン API を使用してコンピュートマシンを作成することができますが、環境に合わせてそれらへの参照を更新する必要があります。

5. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** Kubernetes マニフェストファイルを変更し、Pod がコントロールプレーンマシンにスケジュールされないようにします。

   a. **<installation_directory>/manifests/cluster-scheduler-02-config.yml** ファイルを開きます。

   b. **mastersSchedulable** パラメーターを見つけ、その値を **False** に設定します。

   c. ファイルを保存し、終了します。

6. オプション: Ingress Operator を DNS レコードを作成するよう設定する必要がない場合は、**<installation_directory>/manifests/cluster-dns-02-config.yml** DNS 設定ファイルから **privateZone** および **publicZone** セクションを削除します。

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: null
```

```
  name: cluster
spec:
  baseDomain: example.openshift.com
  privateZone: 1
    id: mycluster-100419-private-zone
  publicZone: 2
    id: example.openshift.com
status: {}
```

1 2 このセクションを完全に削除します。

これを実行する場合、後のステップで Ingress DNS レコードを手動で追加する必要があります。

7. Ignition 設定ファイルを取得します。

```
$ ./openshift-install create ignition-configs --dir=<installation_directory> 1
```

1 **<installation_directory>** については、同じインストールディレクトリーを指定します。

以下のファイルはディレクトリーに生成されます。

```
.
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

12.7.10. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定する必要があります。

手順

1. **chrony.conf** ファイルのコンテンツを作成し、これを base64 でエンコードします。以下に例を示します。

```
$ cat << EOF | base64
   pool 0.rhel.pool.ntp.org iburst 1
   driftfile /var/lib/chrony/drift
   makestep 1.0 3
   rtcsync
   logdir /var/log/chrony
EOF
```

1 DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。

出力例

```
ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. **MachineConfig** ファイルを作成します。base64 文字列を独自に作成した文字列に置き換えます。この例では、ファイルを **master** ノードに追加します。これを **worker** に切り替えたり、**worker** ロールの追加の MachineConfig を作成したりできます。クラスターが使用するそれぞれのタイプのマシンについて MachineConfig ファイルを作成します。

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92Y
XIvbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
          verification: {}
        filesystem: root
        mode: 420
        path: /etc/chrony.conf
  osImageURL: ""
EOF
```

3. 設定ファイルのバックアップコピーを作成します。

4. 以下の 2 つの方法のいずれかで設定を適用します。

   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、そのファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を継続します。

   - クラスターがすでに実行中の場合は、ファイルを適用します。

     ```
     $ oc apply -f ./99-masters-chrony-configuration.yaml
     ```

12.7.11. インフラストラクチャー名の抽出

Ignition 設定ファイルには、VMware vSphere (vSphere) でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。提供される {cp-template} テンプレートにはこのインフラストラクチャー名の参照が含まれるため、これを抽出する必要があります。

Ignition 設定ファイルには、VMware vSphere でクラスターを一意に識別するために使用できる一意のクラスター ID が含まれます。クラスター ID を仮想マシンフォルダーの名前として使用する予定がある場合、これを抽出する必要があります。

前提条件

- OpenShift Container Platform インストールプログラム、およびクラスターのプルシークレットを取得します。
- クラスターの Ignition 設定ファイルを生成します。
- **jq** パッケージをインストールします。

手順

- Ignition 設定ファイルメタデータからインフラストラクチャー名を抽出し、表示するには、以下のコマンドを実行します。

```
$ jq -r .infraID <installation_directory>/metadata.json 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
openshift-vw9j6 1
```

1 このコマンドの出力はクラスター名とランダムな文字列です。

12.7.12. vSphere での Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

ユーザーによってプロビジョニングされるインフラストラクチャーが含まれるクラスターを VMware vSphere にインストールする前に、それが使用する RHCOS マシンを vSphere ホストに作成する必要があります。

前提条件

- クラスターの Ignition 設定ファイルを取得していること。
- お使いのコンピューターからアクセスでき、作成するマシンがアクセスできる HTTP サーバーへのアクセスがあります。
- vSphere クラスター を作成します。

手順

1. **<installation_directory>/bootstrap.ign** という名前のインストールプログラムが作成したブートストラップ Ignition 設定ファイルを HTTP サーバーにアップロードします。このファイルの URL をメモします。
ブートストラップ Ignition 設定ファイルはサイズが大きすぎて vApp プロパティーに適さないため、これをホストする必要があります。

2. ブートストラップノードの以下の二次的な Ignition 設定ファイル
を、**<installation_directory>/append-bootstrap.ign** としてコンピューターに保存します。

```
{
  "ignition": {
    "config": {
      "append": [
        {
          "source": "<bootstrap_ignition_config_url>", 1
          "verification": {}
        }
      ]
    },
    "timeouts": {},
    "version": "2.2.0"
  },
  "networkd": {},
  "passwd": {},
  "storage": {},
  "systemd": {}
}
```

1 ホストしているブートストラップの Ignition 設定ファイルの URL を指定します。

ブートストラップマシンの仮想マシン (VM) を作成する場合に、この Ignition 設定ファイルを使用します。

3. マスター、ワーカー、および二次的なブートストラップ Ignition 設定ファイルを base64 エンコーディングに変換します。
たとえば、Linux オペレーティングシステムを使用する場合、 **base64** コマンドを使用してファイルをエンコードできます。

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64
```

```
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64
```

```
$ base64 -w0 <installation_directory>/append-bootstrap.ign >
<installation_directory>/append-bootstrap.64
```

重要

インストールの完了後にコンピュートマシンをさらにクラスターに追加する予定の場合には、これらのファイルを削除しないでください。

4. RHCOS OVA イメージを取得します。イメージは RHCOS イメージミラー ページで入手できます。

重要

RHCOS イメージは OpenShift Container Platform の各リリースごとに変更されない可能性があります。インストールする OpenShift Container Platform バージョンと等しいか、それ以下のバージョンの内で最も新しいバージョンのイメージをダウンロードする必要があります。利用可能な場合は、OpenShift Container Platform バージョンに一致するイメージのバージョンを使用します。

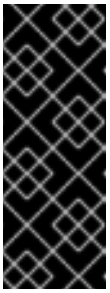
ファイル名には、**rhcos-vmware.<architecture>.ova** 形式の OpenShift Container Platform のバージョン番号が含まれます。

5. vSphere クライアントで、仮想マシンを保管するフォルダーをデータセンターに作成します。
   a. **VMs and Templates** ビューをクリックします。
   b. データセンターの名前を右クリックします。
   c. **New Folder → New VM and Template Folder**をクリックします。
   d. 表示されるウィンドウで、フォルダー名を入力します。**install-config.yaml** ファイルに既存のフォルダーを指定していない場合には、インフラストラクチャー ID と同じ名前を持つフォルダーを作成します。
6. vSphere クライアントで、OVA イメージのテンプレートを作成してから、必要に応じてテンプレートのクローンを作成します。

注記

以下の手順では、テンプレートを作成してから、すべてのクラスターマシンのテンプレートのクローンを作成します。次に、仮想マシンのプロビジョニング時にクローン作成されたマシンタイプの Ignition 設定ファイルの場所を指定します。

   a. **Hosts and Clusters** タブで、クラスターの名前を右クリックし、**Deploy OVF Template** を選択します。
   b. **Select an OVF** タブで、ダウンロードした RHCOS OVA ファイルの名前を指定します。
   c. **Select a name and folder**タブで、**Template-RHCOS** などの **Virtual machine name**をテンプレートに設定します。vSphere クラスターの名前をクリックし、直前の手順で作成したフォルダーを選択します。
   d. **Select a compute resource** タブで、vSphere クラスターの名前をクリックします。
   e. **Select storage** タブで、仮想マシンのストレージオプションを設定します。
      - ストレージ設定に応じて、**Thin Provision** または **Thick Provision** を選択します。
      - **install-config.yaml** ファイルで指定したデータストアを選択します。
   f. **Select network** タブで、クラスターに設定したネットワークを指定します (ある場合)。
   g. OVF テンプレートの作成時には、**Customize template** タブで値を指定したり、テンプレートに追加の設定をしないようにしてください。



重要

元の仮想マシンテンプレートは開始しないでください。仮想マシンテンプレートは停止した状態でなければなりません。また、新規 RHCOS マシン用にクローン作成する必要があります。仮想マシンテンプレートを起動すると、仮想マシンテンプレートがプラットフォームの仮想マシンとして設定されるので、これをマシンセットで設定を適用できるテンプレートとして使用できなくなります。

7. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
    a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
    b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**control-plane-0** または **compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
    c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
    d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
    e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
    f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
    g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
        - オプション: クラスターのパフォーマンスに問題が生じる場合は、**Latency Sensitivity** 一覧から **High** を選択します。
        - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
            - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードした Ignition 設定ファイルの内容を貼り付けます。
            - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
            - **disk.EnableUUID**: **TRUE** を指定します。
        - または、仮想マシンの電源を入れる前に vApp プロパティーを使用して追加します。
            - vCenter Server インベントリーから仮想マシンに移動します。
            - **Configure** タブで **Settings** を展開し、**vAPP options** を選択します。
            - スクロールダウンし、**Properties** の下で上記の設定を適用します。
    h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。
    i. 設定を完了し、仮想マシンの電源をオンにします。
8. 各マシンごとに先の手順に従って、クラスターの残りのマシンを作成します。

重要

この時点でブートストラップおよびコントロールプレーンマシンを作成する必要があります。一部の Pod はデフォルトでコンピュートマシンにデプロイされるため、クラスターのインストール前に、2 つ以上のコンピュートマシンを作成します。

12.7.13. vSphere での追加の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの作成

VMware vSphere でユーザーによってプロビジョニングされるインフラストラクチャーを使用するクラスターのコンピュートマシンを追加で作成できます。

前提条件

- コンピュートマシンの base64 でエンコードされた Ignition ファイルを取得します。
- クラスター用に作成した vSphere テンプレートにアクセスできる必要があります。

手順

1. テンプレートがデプロイされた後に、マシンの仮想マシンをクラスターにデプロイします。
   a. テンプレートの名前を右クリックし、**Clone → Clone to Virtual Machine**をクリックします。
   b. **Select a name and folder**タブで、仮想マシンの名前を指定します。**compute-1** などのように、マシンタイプを名前に含めることができるかもしれません。
   c. **Select a name and folder**タブで、クラスターに作成したフォルダーの名前を選択します。
   d. **Select a compute resource** タブで、データセンター内のホストの名前を選択します。
   e. オプション: **Select storage** タブで、ストレージオプションをカスタマイズします。
   f. **Select clone options** で、**Customize this virtual machine's hardware**を選択します。
   g. **Customize hardware** タブで、**VM Options→ Advanced** をクリックします。
      - **Latency Sensitivity** 一覧から、**High** を選択します。
      - **Edit Configuration** をクリックし、**Configuration Parameters** ウィンドウで **Add Configuration Params** をクリックします。以下のパラメーター名および値を定義します。
        - **guestinfo.ignition.config.data**: このマシンファイルの base64 でエンコードしたコンピュート Ignition 設定ファイルの内容を貼り付けます。
        - **guestinfo.ignition.config.data.encoding**: **base64** を指定します。
        - **disk.EnableUUID**: **TRUE** を指定します。
   h. **Customize hardware** タブの **Virtual Hardware** パネルで、必要に応じて指定した値を変更します。RAM、CPU、およびディスクストレージの量がマシンタイプの最小要件を満たすことを確認してください。また、ネットワークが複数利用可能な場合は、必ず **Add network adapter** に正しいネットワークを選択してください。

i. 設定を完了し、仮想マシンの電源をオンにします。

2. 継続してクラスター用の追加のコンピュートマシンを作成します。

12.7.14. クラスターの作成

OpenShift Container Platform クラスターを作成するには、ブートストラッププロセスが、インストールプログラムで生成した Ignition 設定ファイルを使用してプロビジョニングしたマシンで完了するのを待機します。

前提条件

- クラスターに必要なインフラストラクチャーを作成する。
- インストールプログラムを取得し、クラスターの Ignition 設定ファイルを生成している。
- Ignition 設定ファイルを使用して、クラスターの RHCOS マシンを作成済している。

手順

1. ブートストラッププロセスをモニターします。

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete \ 1
    --log-level=info 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なるインストールの詳細情報を表示するには、**info** ではなく、**warn**、**debug**、または **error** を指定します。

出力例

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.18.3 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

Kubernetes API サーバーでこれがコントロールプレーンマシンにブートストラップされていることを示すシグナルが出されるとコマンドは成功します。

2. ブートストラッププロセスが完了したら、ブートストラップマシンをロードバランサーから削除します。

重要

この時点で、ブートストラップマシンをロードバランサーから削除する必要があります。さらに、マシン自体を削除し、再フォーマットすることができます。

12.7.15. クラスターへのログイン

クラスター **kubeconfig** ファイルをエクスポートし、デフォルトシステムユーザーとしてクラスターにログインできます。**kubeconfig** ファイルには、クライアントを正しいクラスターおよび API サーバー

に接続するために CLI で使用されるクラスターについての情報が含まれます。このファイルはクラスターに固有のファイルであり、OpenShift Container Platform のインストール時に作成されます。

前提条件

- OpenShift Container Platform クラスターをデプロイします。
- **oc** CLI をインストールします。

手順

1. **kubeadmin** 認証情報をエクスポートします。

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2. エクスポートされた設定を使用して、**oc** コマンドを正常に実行できることを確認します。

```
$ oc whoami
```

出力例

```
system:admin
```

12.7.16. マシンの証明書署名要求の承認

マシンをクラスターに追加する際に、追加したそれぞれのマシンについて 2 つの保留状態の証明書署名要求 (CSR) が生成されます。これらの CSR が承認されていることを確認するか、または必要な場合はそれらを承認してください。最初にクライアント要求を承認し、次にサーバー要求を承認する必要があります。

前提条件

- マシンがクラスターに追加されています。

手順

1. クラスターがマシンを認識していることを確認します。

```
$ oc get nodes
```

出力例

```
NAME      STATUS    ROLES   AGE  VERSION
master-0  Ready     master  63m  v1.18.3
master-1  Ready     master  63m  v1.18.3
master-2  Ready     master  64m  v1.18.3
worker-0  NotReady  worker  76s  v1.18.3
worker-1  NotReady  worker  70s  v1.18.3
```

出力には作成したすべてのマシンが一覧表示されます。

2. 保留中の証明書署名要求 (CSR) を確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

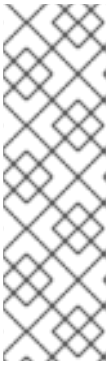
```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
...
```

この例では、2 つのマシンがクラスターに参加しています。この一覧にはさらに多くの承認された CSR が表示される可能性があります。

3. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。クライアントの CSR が承認されたら、Kubelet は提供証明書のセカンダリー CSR を作成します。これには、手動の承認が必要です。次に、後続の提供証明書の更新要求は、Kubelet が同じパラメーターを持つ新規証明書を要求する場合に **machine-approver** によって自動的に承認されます。

- それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs --no-run-if-empty oc adm certificate approve
```

4. クライアント要求が承認されたら、クラスターに追加した各マシンのサーバー要求を確認する必要があります。

```
$ oc get csr
```

出力例

```
NAME        AGE     REQUESTOR                                                   CONDITION
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal
Pending
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal
Pending
...
```

5. 残りの CSR が承認されず、それらが **Pending** ステータスにある場合、クラスターマシンの CSR を承認します。

   - それらを個別に承認するには、それぞれの有効な CSR について以下のコマンドを実行します。

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

     1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

   - すべての保留中の CSR を承認するには、以下のコマンドを実行します。

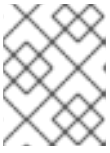
     ```
     $ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}
     {{end}}{{end}}' | xargs oc adm certificate approve
     ```

6. すべてのクライアントおよびサーバーの CSR が承認された後に、マシンのステータスが **Ready** になります。以下のコマンドを実行して、これを確認します。

   ```
   $ oc get nodes
   ```

   出力例

   ```
   NAME      STATUS    ROLES   AGE  VERSION
   master-0  Ready     master  73m  v1.20.0
   master-1  Ready     master  73m  v1.20.0
   master-2  Ready     master  74m  v1.20.0
   worker-0  Ready     worker  11m  v1.20.0
   worker-1  Ready     worker  11m  v1.20.0
   ```



注記

サーバー CSR の承認後にマシンが **Ready** ステータスに移行するまでに数分の時間がかかる場合があります。

関連情報

- CSR の詳細は、Certificate Signing Requests を参照してください。

12.7.17. Operator の初期設定

コントロールプレーンの初期化後に、一部の Operator を利用可能にするためにそれらをすぐに設定する必要があります。

前提条件

- コントロールプレーンが初期化されています。

手順

1. クラスターコンポーネントがオンラインになることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.5.4     True        False         False      69s
cloud-credential                           4.5.4     True        False         False      12m
cluster-autoscaler                         4.5.4     True        False         False      11m
console                                    4.5.4     True        False         False      46s
dns                                        4.5.4     True        False         False      11m
image-registry                             4.5.4     True        False         False      5m26s
ingress                                    4.5.4     True        False         False      5m36s
kube-apiserver                             4.5.4     True        False         False      8m53s
kube-controller-manager                    4.5.4     True        False         False      7m24s
kube-scheduler                             4.5.4     True        False         False      12m
machine-api                                4.5.4     True        False         False      12m
machine-config                             4.5.4     True        False         False      7m36s
marketplace                                4.5.4     True        False         False      7m54m
monitoring                                 4.5.4     True        False         False      7h54s
network                                    4.5.4     True        False         False      5m9s
node-tuning                                4.5.4     True        False         False      11m
openshift-apiserver                        4.5.4     True        False         False      11m
openshift-controller-manager               4.5.4     True        False         False      5m943s
openshift-samples                          4.5.4     True        False         False      3m55s
operator-lifecycle-manager                 4.5.4     True        False         False      11m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      11m
service-ca                                 4.5.4     True        False         False      11m
service-catalog-apiserver                  4.5.4     True        False         False      5m26s
service-catalog-controller-manager         4.5.4     True        False         False      5m25s
storage                                    4.5.4     True        False         False      5m30s
```

2. 利用不可の Operator を設定します。

12.7.17.1. イメージレジストリーストレージの設定

イメージレジストリー Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定についての手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

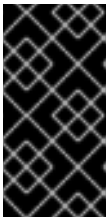
アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

12.7.17.1.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Container Storage などのクラスターのプロビジョニングされた永続ストレージ。



重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100Gi の容量が必要です。

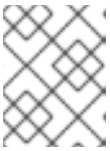
重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリクスストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。



注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry
```

注記

ストレージタイプが **emptyDIR** の場合、レプリカ数が **1** を超えることはありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 **claim** フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

12.7.17.1.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

イメージレジストリー Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

1. イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

イメージレジストリー Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

2. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

上記を以下のように変更します。

```
managementState: Managed
```

12.7.17.1.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. イメージレジストリーストレージをブロックストレージタイプとして設定するには、レジストリーが **Recreate** ロールアウトストラテジーを使用し、**1** レプリカのみで実行されるように、レジストリーにパッチを適用します。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
spec:
  accessModes:
  - ReadWriteOnce (2)
  resources:
    requests:
      storage: 100Gi (3)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) PersistentVolumeClaim のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (3) PersistentVolumeClaim のサイズ。

   b. ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: 1
```

1 カスタム PVC を作成すると、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにすることができます。

正しい PVC を参照するようにレジストリーストレージを設定する方法については、vSphere のレジストリーの設定 を参照してください。

12.7.18. ユーザーによってプロビジョニングされるインフラストラクチャーでのインストールの完了

Operator 設定の完了後に、提供するインフラストラクチャーでのクラスターのインストールを終了できます。

前提条件

- コントロールプレーンが初期化されています。
- Operator の初期設定を完了済みです。

手順

1. 以下のコマンドを使用して、すべてのクラスターコンポーネントがオンラインであることを確認します。

```
$ watch -n5 oc get clusteroperators
```

出力例

```
NAME                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication             4.5.4     True        False         False      7m56s
cloud-credential           4.5.4     True        False         False      31m
cluster-autoscaler         4.5.4     True        False         False      16m
console                    4.5.4     True        False         False      10m
csi-snapshot-controller    4.5.4     True        False         False      16m
dns                        4.5.4     True        False         False      22m
etcd                       4.5.4     False       False         False      25s
image-registry             4.5.4     True        False         False      16m
ingress                    4.5.4     True        False         False      16m
insights                   4.5.4     True        False         False      17m
kube-apiserver             4.5.4     True        False         False      19m
```

```
kube-controller-manager                    4.5.4     True        False         False      20m
kube-scheduler                             4.5.4     True        False         False      20m
kube-storage-version-migrator              4.5.4     True        False         False      16m
machine-api                                4.5.4     True        False         False      22m
machine-config                             4.5.4     True        False         False      22m
marketplace                                4.5.4     True        False         False      16m
monitoring                                 4.5.4     True        False         False      10m
network                                    4.5.4     True        False         False      23m
node-tuning                                4.5.4     True        False         False      23m
openshift-apiserver                        4.5.4     True        False         False      17m
openshift-controller-manager               4.5.4     True        False         False      15m
openshift-samples                          4.5.4     True        False         False      16m
operator-lifecycle-manager                 4.5.4     True        False         False      22m
operator-lifecycle-manager-catalog         4.5.4     True        False         False      22m
operator-lifecycle-manager-packageserver   4.5.4     True        False         False      18m
service-ca                                 4.5.4     True        False         False      23m
service-catalog-apiserver                  4.5.4     True        False         False      23m
service-catalog-controller-manager         4.5.4     True        False         False      23m
storage                                    4.5.4     True        False         False      17m
```

あるいは、以下のコマンドを使用すると、すべてのクラスターが利用可能な場合に通知されます。また、このコマンドは認証情報を取得して表示します。

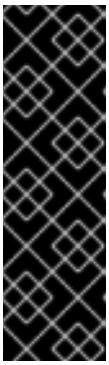
```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete 1
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

出力例

```
INFO Waiting up to 30m0s for the cluster to initialize...
```

Cluster Version Operator が Kubernetes API サーバーから OpenShift Container Platform クラスターのデプロイを終了するとコマンドは成功します。



重要

インストールプログラムが生成する Ignition 設定ファイルには、24 時間が経過すると期限切れになり、その後に更新される証明書が含まれます。証明書を更新する前にクラスターが停止し、24 時間経過した後にクラスターを再起動すると、クラスターは期限切れの証明書を自動的に復元します。例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。詳細は、**コントロールプレーン証明書の期限切れの状態からのリカバリー** についてのドキュメントを参照してください。

2. Kubernetes API サーバーが Pod と通信していることを確認します。

   a. すべての Pod の一覧を表示するには、以下のコマンドを使用します。

   ```
   $ oc get pods --all-namespaces
   ```

   出力例

```
NAMESPACE                  NAME                                        READY   STATUS
RESTARTS   AGE
openshift-apiserver-operator    openshift-apiserver-operator-85cb746d55-zqhs8   1/1
Running    1          9m
openshift-apiserver        apiserver-67b9g                             1/1     Running    0
3m
openshift-apiserver        apiserver-ljcmx                             1/1     Running    0
1m
openshift-apiserver        apiserver-z25h4                             1/1     Running    0
2m
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8       1/1
Running    0          5m
...
```

b. 以下のコマンドを使用して、直前のコマンドの出力に一覧表示される Pod のログを表示します。

```
$ oc logs <pod_name> -n <namespace>
```
(1)

(1) 直前のコマンドの出力にあるように、Pod 名および namespace を指定します。

Pod のログが表示される場合、Kubernetes API サーバーはクラスターマシンと通信できます。

3. Cluster registration ページでクラスターを登録します。

Adding compute machines to vSphere に従い、クラスターのインストールの完了後に追加のコンピュートマシンを追加できます。

12.7.19. VMware vSphere ボリュームのバックアップ

OpenShift Container Platform は、自由にクラスターないのノードにあるボリュームをアタッチしたり、アタッチ解除できるように、個別の永続ディスクとして新規ボリュームをプロビジョニングします。そのため、スナップショットを使用するボリュームはバックアップしたり、スナップショットからボリュームを復元したりすることはできません。詳細は、スナップショットの制限 を参照してください。

手順

永続ボリュームのバックアップを作成すには、以下を実行します。

1. 永続ボリュームを使用しているアプリケーションを停止します。
2. 永続ボリュームのクローンを作成します。
3. アプリケーションを再起動します。
4. クローンを作成したボリュームのバックアップを作成します。
5. クローンを作成したボリュームを削除します。

12.7.20. 次のステップ

- クラスターをカスタマイズ します。

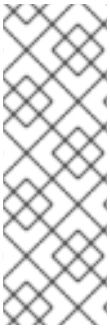
- クラスターのインストールに使用したミラーレジストリーに信頼される CA がある場合、信頼ストアを設定 してこれをクラスターに追加します。
- 必要な場合は、リモートの健全性レポートをオプトアウト することができます。

12.8. インストーラーでプロビジョニングされるインフラストラクチャーを使用した VSPHERE へのクラスターのインストール

インストーラーでプロビジョニングされるインフラストラクチャーを使用して、VMware vSphere インスタンスにデプロイしたクラスターを削除できます。

12.8.1. インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターの削除

インストーラーでプロビジョニングされるインフラストラクチャーを使用するクラスターは、クラウドから削除できます。



注記

アンインストール後に、とくにユーザーによってプロビジョニングされるインフラストラクチャー (UPI) クラスターで適切に削除されていないリソースがあるかどうかについて、クラウドプロバイダーを確認します。インストーラーが作成されなかったり、インストーラーがアクセスできない場合には、リソースがある可能性があります。たとえば、一部の Google Cloud リソースには共有 VPC ホストプロジェクトで IAM パーミッション が必要になるか、または 削除する必要のあるヘルスチェック が使用されていない可能性があります。

前提条件

- クラスターをデプロイするために使用したインストールプログラムのコピーがあります。
- クラスター作成時にインストールプログラムが生成したファイルがあります。

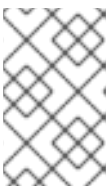
手順

1. クラスターをインストールするために使用したコンピューターから、以下のコマンドを実行します。

```
$ ./openshift-install destroy cluster \
--dir=<installation_directory> --log-level=info 1 2
```

1 **<installation_directory>** には、インストールファイルを保存したディレクトリーへのパスを指定します。

2 異なる詳細情報を表示するには、 **info** ではなく、**warn**、**debug**、または **error** を指定します。



注記

クラスターのクラスター定義ファイルが含まれるディレクトリーを指定する必要があります。クラスターを削除するには、インストールプログラムでこのディレクトリーにある **metadata.json** ファイルが必要になります。

2. オプション: **<installation_directory>** ディレクトリーおよび OpenShift Container Platform インストールプログラムを削除します。

第13章 インストール設定

13.1. 各種プラットフォームのサポートされているインストール方法

各種のプラットフォームで各種のインストールを実行できます。

注記

以下の表にあるように、すべてのプラットフォームですべてのインストールオプションがサポートされている訳ではありません。

表13.1 インストーラーでプロビジョニングされるインフラストラクチャーのオプション

	AWS	Azure	GCP	OpenStack	RHV	ベアメタル	vSphere	IBM Z
デフォルト	X	X	X		X		X	
カスタム	X	X	X	X	X		X	
Network Operator	X	X	X				X	
ネットワークが制限されたインストール	X		X	X			X	
プライベートクラスター	X	X	X					
既存の仮想プライベートネットワーク	X	X	X					

表13.2 ユーザーによってプロビジョニングされるインフラストラクチャーのオプション

	AWS	Azure	GCP	OpenStack	RHV	ベアメタル	vSphere	IBM Z
カスタム	X	X	X	X		X	X	

	AWS	Azure	GCP	OpenStack	RHV	ベアメタル	vSphere	IBM Z
Network Operator						X	X	
ネットワークが制限されたインストール	X		X			X	X	
クラスタープロジェクト外でホストされる共有 VPC			X					

13.2. ノードのカスタマイズ

OpenShift Container Platform ノードへの直接の変更は推奨されませんが、必要とされる低レベルのセキュリティー、ネットワーク、またはパフォーマンス機能を実装することが必要になる場合があります。OpenShift Container Platform ノードへの直接の変更は、以下によって実行できます。

- **openshift-install** の実行時にクラスターを起動するためにマニフェストファイルに組み込まれるマシン設定を作成します。
- Machine Config Operator を使用して実行中の OpenShift Container Platform ノードに渡されるマシン設定を作成します。

以下のセクションでは、この方法でノード上で設定する必要が生じる可能性のある機能について説明します。

13.2.1. day-1 カーネル引数の追加

多くの場合、カーネル引数を day-2 アクティビティーとして変更することが推奨されますが、初期クラスターのインストール時にすべてのマスターまたはワーカーノードにカーネル引数を追加することができます。以下は、クラスターのインストール時にカーネル引数を追加して、システムの初回起動前に有効にする必要が生じる可能性のある理由です。

- SELinux などの機能を無効にし、初回起動時にシステムに影響を与えないようにする必要がある場合。
- システムの起動前に、低レベルのネットワーク設定を実行する必要がある場合。

カーネル引数をマスターまたはワーカーノードに追加するには、**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

起動時に RHEL 8 カーネルに渡すことのできる引数の一覧については、Kernel.org カーネルパラメーター を参照してください。カーネル引数が OpenShift Container Platform の初回インストールを完了するために必要な場合は、この手順でカーネル引数のみを追加することが推奨されます。

手順

1. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

2. カーネル引数をワーカーまたはマスターノードに追加するかどうかを決定します。

3. **openshift** ディレクトリーでファイル (例: **99-openshift-machineconfig-master-kargs.yaml**) を作成し、カーネル設定を追加するために **MachineConfig** オブジェクトを定義します。以下の例では、**loglevel=7** カーネル引数をマスターノードに追加します。

```
$ cat << EOF > 99-openshift-machineconfig-master-kargs.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-openshift-machineconfig-master-kargs
spec:
  kernelArguments:
    - 'loglevel=7'
EOF
```

   カーネル引数をワーカーノードに追加する場合は、**master** を **worker** に切り替えます。マスターおよびワーカーノードの両方に追加するために別々の YAML ファイルを作成します。

クラスターの作成を継続できます。

13.2.2. カーネルモジュールのノードへの追加

大半の一般的なハードウェアの場合、Linux カーネルには、コンピューターの起動時にそのハードウェアを使用するために必要となるデバイスドライバーモジュールが含まれます。ただし、一部のハードウェアの場合、Linux でモジュールを利用できません。したがって、各ホストコンピューターにこれらのモジュールを提供する方法を確保する必要があります。この手順では、OpenShift Container Platform クラスターのノードについてこれを実行する方法を説明します。

この手順に従ってカーネルモジュールを最初にデプロイする際、モジュールは現行のカーネルに対して利用可能になります。新規カーネルがインストールされると、kmods-via-containers ソフトウェアはモジュールを再ビルドし、デプロイしてそのモジュールの新規カーネルと互換性のあるバージョンが利用可能になるようにします。

この機能によって各ノードでモジュールが最新の状態に保てるようにするために、以下が実行されます。

- 新規カーネルがインストールされているかどうかを検出するために、システムの起動時に起動する各ノードに systemd サービスを追加します。
- 新規カーネルが検出されると、サービスはモジュールを再ビルドし、これをカーネルにインストールします。

この手順に必要なソフトウェアの詳細については、kmods-via-containers github サイトを参照してください。

以下の重要な点に留意してください。

- この手順はテクノロジープレビューです。
- ソフトウェアのツールおよびサンプルは公式の RPM 形式で利用できず、現時点ではこの手順に記載されている非公式の **github.com** サイトからしか取得できません。
- この手順で追加する必要がある可能性のあるサードパーティーのカーネルモジュールについては Red Hat はサポートしません。
- この手順では、カーネルモジュールのビルドに必要なソフトウェアは RHEL 8 コンテナーにデプロイされます。モジュールは、ノードが新規カーネルを取得する際に各ノードで自動的に再ビルドされることに注意してください。このため、各ノードには、モジュールの再ビルドに必要なカーネルと関連パッケージを含む **yum** リポジトリーへのアクセスが必要です。このコンテンツは、有効な RHEL サブスクリプションを使用して効果的に利用できます。

13.2.2.1. カーネルモジュールコンテナーのビルドおよびテスト

カーネルモジュールを OpenShift Container Platform クラスターにデプロイする前に、プロセスを別の RHEL システムでテストできます。カーネルモジュールのソースコード、KVC フレームワーク、および kmod-via-containers ソフトウェアを収集します。次にモジュールをビルドし、テストします。RHEL 8 システムでこれを行うには、以下を実行します。

手順

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアとコンテナーのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. **kmod-via-containers** リポジトリーのクローンを作成します。
   a. リポジトリーのフォルダーを作成します。

```
$ mkdir kmods; cd kmods
```

   b. リポジトリーのクローンを作成します。

```
$ git clone https://github.com/kmods-via-containers/kmods-via-containers
```

5. RHEL 8 ビルドホストに KVC フレームワークインスタンスをインストールし、モジュールをテストします。これにより、**kmods-via-container** systemd サービスが追加され、読み込まれます。

a. **kmod-via-containers** ディレクトリーに移動します。

```
$ cd kmods-via-containers/
```

b. KVC フレームワークインスタンスをインストールします。

```
$ sudo make install
```

c. systemd マネージャー設定を再読み込みします。

```
$ sudo systemctl daemon-reload
```

6. カーネルモジュールのソースコードを取得します。ソースコードは、制御下になく、他から提供されるサードパーティーモジュールをビルドするために使用される可能性があります。システムに対してクローン作成できる以下の **kvc-simple-kmod** サンプルのコンテンツと同様のコンテンツが必要になります。

```
$ cd .. ; git clone https://github.com/kmods-via-containers/kvc-simple-kmod
```

7. この例では、設定ファイル **simple-kmod.conf** を編集し、Dockerfile の名前を **Dockerfile.rhel** に変更します。

a. **kvc-simple-kmod** ディレクトリーに移動します。

```
$ cd kvc-simple-kmod
```

b. Dockerfile の名前を変更します。

```
$ cat simple-kmod.conf
```

Dockerfile の例

```
KMOD_CONTAINER_BUILD_CONTEXT="https://github.com/kmods-via-containers/kvc-simple-kmod.git"
KMOD_CONTAINER_BUILD_FILE=Dockerfile.rhel
KMOD_SOFTWARE_VERSION=dd1a7d4
KMOD_NAMES="simple-kmod simple-procfs-kmod"
```

8. この例ではカーネルモジュール **simple-kmod** の **kmods-via-containers@.service** のインスタンスを作成します。

```
$ sudo make install
```

9. **kmods-via-containers@.service** インスタンスを有効にします。

```
$ sudo kmods-via-containers build simple-kmod $(uname -r)
```

10. systemd サービスを有効にし、起動します。

a. サービスを有効にします。

```
$ sudo systemctl enable kmods-via-containers@simple-kmod.service
```

b. サービスを起動します。

```
$ sudo systemctl start kmods-via-containers@simple-kmod.service
```

c. サービスのステータスを確認します。

```
$ sudo systemctl status kmods-via-containers@simple-kmod.service
```

出力例

```
● kmods-via-containers@simple-kmod.service - Kmods Via Containers - simple-kmod
Loaded: loaded (/etc/systemd/system/kmods-via-containers@.service;
       enabled; vendor preset: disabled)
Active: active (exited) since Sun 2020-01-12 23:49:49 EST; 5s ago...
```

11. カーネルモジュールがロードされていることを確認するには、**lsmod** コマンドを使用してモジュールを一覧表示します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

12. オプション。他の方法を使用して **simple-kmod** のサンプルが機能していることを確認します。

- **dmesg** を使ってカーネルリングバッファーで Hello world メッセージを探します。

```
$ dmesg | grep 'Hello world'
```

出力例

```
[ 6420.761332] Hello world from simple_kmod.
```

- /**proc** で **simple-procfs-kmod** の値を確認します。

```
$ sudo cat /proc/simple-procfs-kmod
```

出力例

```
simple-procfs-kmod number = 0
```

- **spkut** コマンドを実行して、モジュールの詳細情報を取得します。

```
$ sudo spkut 44
```

出力例

```
KVC: wrapper simple-kmod for 4.18.0-147.3.1.el8_1.x86_64
Running userspace wrapper using the kernel module container...
```

```
+ podman run -i --rm --privileged
  simple-kmod-dd1a7d4:4.18.0-147.3.1.el8_1.x86_64 spkut 44
simple-procfs-kmod number = 0
simple-procfs-kmod number = 44
```

その後は、システムの起動時に、このサービスは新規カーネルが実行中であるかどうかをチェックします。新規カーネルがある場合は、サービスは新規バージョンのカーネルモジュールをビルドし、これをロードします。モジュールがすでにビルドされている場合は、これをロードします。

13.2.2.2. カーネルモジュールの OpenShift Container Platform へのプロビジョニング

OpenShift Container Platform クラスターの初回起動時にカーネルモジュールを有効にする必要があるかどうかに応じて、以下のいずれかの方法でデプロイするようにカーネルモジュールを設定できます。

- **クラスターインストール時のカーネルモジュールのプロビジョニング (day-1)**: コンテンツを **MachineConfig** として作成し、これをマニフェストファイルのセットと共に組み込み、これを **openshift-install** に提供できます。
- **Machine Config Operator によるカーネルモジュールのプロビジョニング (day-2)**: カーネルモジュールを追加する際にクラスターが稼働するまで待機できる場合、Machine Config Operator (MCO) を使用してカーネルモジュールソフトウェアをデプロイできます。

いずれの場合も、各ノードは、新規カーネルの検出時にカーネルパッケージと関連ソフトウェアパッケージを取得できる必要があります。該当するコンテンツを取得できるように各ノードをセットアップする方法はいくつかあります。

- 各ノードに RHEL エンタイトルメントを提供します。
- **/etc/pki/entitlement** ディレクトリーから、既存 RHEL ホストの RHEL エンタイトルメントを取得し、それらを Ignition 設定の作成時に提供する他のファイルと同じ場所にコピーします。
- Dockerfile 内で、カーネルおよびその他のパッケージを含む **yum** リポジトリーへのポインターを追加します。これには、新たにインストールされたカーネルと一致させる必要があるため、新規のカーネルパッケージが含まれている必要があります。

13.2.2.2.1. MachineConfig オブジェクトでのカーネルモジュールのプロビジョニング

MachineConfig オブジェクト でカーネルモジュールソフトウェアをパッケージ化することで、そのソフトウェアをインストール時に、または Machine Config Operator を使用してワーカーまたはマスターノードに配信できます。

まず、使用するベース Ignition 設定を作成します。インストール時に、Ignition 設定にはクラスターの **core** ユーザーの **authorized_keys** ファイルに追加する ssh パブリックキーが含まれます。後で MCO を使用して **MachineConfig** を追加する場合、ssh パブリックキーは不要になります。どちらのタイプでも、サンプルの simple-kmod サービスは **kmods-via-containers@simple-kmod.service** を必要とする systemd ユニットファイルを作成します。

注記

systemd ユニットは アップストリームのバグ に対する回避策であり、**kmods-via-containers@simple-kmod.service** が起動時に開始するようにします。

1. RHEL 8 システムを登録します。

```
# subscription-manager register
```

2. RHEL 8 システムにサブスクリプションを割り当てます。

```
# subscription-manager attach --auto
```

3. ソフトウェアのビルドに必要なソフトウェアをインストールします。

```
# yum install podman make git -y
```

4. systemd ユニットファイルを作成する Ignition 設定ファイルを作成します。

   a. Ignition 設定ファイルをホストするディレクトリーを作成します。

```
$ mkdir kmods; cd kmods
```

   b. systemd ユニットファイルを作成する Ignition 設定ファイルを作成します。

```
$ cat <<EOF > ./baseconfig.ign
{
  "ignition": { "version": "2.2.0" },
  "passwd": {
    "users": [
      {
        "name": "core",
        "groups": ["sudo"],
        "sshAuthorizedKeys": [
          "ssh-rsa AAAA"
        ]
      }
    ]
  },
  "systemd": {
    "units": [{
      "name": "require-kvc-simple-kmod.service",
      "enabled": true,
      "contents": "[Unit]\nRequires=kmods-via-containers@simple-
kmod.service\n[Service]\nType=oneshot\nExecStart=/usr/bin/true\n\n[Install]\nWantedBy=m
ulti-user.target"
    }]
  }
}
EOF
```

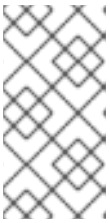
注記

openshift-install の実行時に、パブリック SSH キーを使用する **baseconfig.ign** ファイルに追加する必要があります。MCO を使用して **MachineConfig** オブジェクトを作成する場合、パブリック SSH キーは必要ありません。

5. 以下の設定を使用するベース MCO YAML スニペットを作成します。

```
$ cat <<EOF > mc-base.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
```

```
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 10-kvc-simple-kmod
spec:
  config:
EOF
```



注記

mc-base.yaml は、**worker** ノードにカーネルモジュールをデプロイするように設定されます。マスターノードでデプロイするには、ロールを **worker** から **master** に変更します。どちらの方法でも、デプロイメントの種類ごとに異なるファイル名を使用して手順全体を繰り返すことができます。

6. **kmods-via-containers** ソフトウェアを取得します。
   a. **kmods-via-containers** リポジトリーのクローンを作成します。

      ```
      $ git clone https://github.com/kmods-via-containers/kmods-via-containers
      ```

   b. **kvc-simple-kmod** リポジトリーのクローンを作成します。

      ```
      $ git clone https://github.com/kmods-via-containers/kvc-simple-kmod
      ```

7. モジュールソフトウェアを取得します。この例では、**kvc-simple-kmod** が使用されます。
8. fakeroot ディレクトリーを作成し、先にクローン作成したリポジトリーを使用して Ignition で配信するファイルを使用してこれを設定します。
   a. ディレクトリーを作成します。

      ```
      $ FAKEROOT=$(mktemp -d)
      ```

   b. **kmod-via-containers** ディレクトリーに移動します。

      ```
      $ cd kmods-via-containers
      ```

   c. KVC フレームワークインスタンスをインストールします。

      ```
      $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
      ```

   d. **kvc-simple-kmod** ディレクトリーに移動します。

      ```
      $ cd ../kvc-simple-kmod
      ```

   e. インスタンスを作成します。

      ```
      $ make install DESTDIR=${FAKEROOT}/usr/local CONFDIR=${FAKEROOT}/etc/
      ```

9. **filetranspiler** というツールおよび依存するソフトウェアを取得します。

```
$ cd .. ; sudo yum install -y python3
git clone https://github.com/ashcrow/filetranspiler.git
```

10. 最終的なマシン設定 YAML(**mc.yaml**) を生成し、これに配信するファイルと共にベース Ignition 設定、ベースマシン設定、および fakeroot ディレクトリーを含めます。

```
$ ./filetranspiler/filetranspile -i ./baseconfig.ign \
    -f ${FAKEROOT} --format=yaml --dereference-symlinks \
    | sed 's/^/     /' | (cat mc-base.yaml -) > 99-simple-kmod.yaml
```

11. クラスターがまだ起動していない場合は、マニフェストファイルを生成し、そのファイルを **openshift** ディレクトリーに追加します。クラスターがすでに実行中の場合は、ファイルを以下のように適用します。

```
$ oc create -f 99-simple-kmod.yaml
```

ノードは **kmods-via-containers@simple-kmod.service** サービスを起動し、カーネルモジュールがロードされます。

12. カーネルモジュールがロードされていることを確認するには、ノードにログインすることができます (**oc debug node/<openshift-node>** を使用してから **chroot /host** を使用します)。モジュールを一覧表示するには、**lsmod** コマンドを使用します。

```
$ lsmod | grep simple_
```

出力例

```
simple_procfs_kmod     16384  0
simple_kmod            16384  0
```

13.2.3. インストール時のディスクの暗号化

OpenShift Container Platform のインストール時に、すべてのマスターおよびノードノードでディスクの暗号化を有効にできます。この機能には以下の特徴があります。

- インストーラーでプロビジョニングされるインフラストラクチャーおよびユーザーによってプロビジョニングされるインフラストラクチャーのデプロイメントで利用可能である。
- Red Hat Enterprise Linux CoreOS (RHCOS) システムのみでサポートされる。
- マニフェストのインストールフェーズでディスク暗号化が設定される。これにより、初回起動時からディスクに書き込まれたすべてのデータが暗号化されます。
- ルートファイルシステムのみでデータを暗号化する (**/dev/mapper/coreos-luks-root** は / マウントポイントを表す)。
- パスフレーズを提供するのにユーザーの介入を必要としない。
- AES-256-CBC 暗号化を使用する。

サポートされている暗号化モードとして、以下の 2 つのモードを使用できます。

- TPM v2: これは優先されるモードです。TPM v2 はパスフレーズを安全な暗号プロセッサーに保存します。TPM v2 ディスク暗号化を実装するには、以下で説明されているように Ignition 設定ファイルを作成します。

- Tang: Tang を使用してクラスターを暗号化するには、Tang サーバーを使用する必要があります。Clevis はクライアント側に復号化を実装します。Tang 暗号化モードは、ベアメタルのインストールの場合にのみサポートされます。

クラスター内のノードのディスク暗号化を有効にするには、以下の 2 つの手順の内のいずれかに従います。

13.2.3.1. TPM v2 ディスク暗号化の有効化

以下の手順を使用して、OpenShift Container Platform のデプロイメント時に TPM v2 モードのディスク暗号化を有効にします。

手順

1. TPM v2 暗号化を各ノードの BIOS で有効にする必要があるかどうかを確認します。これは、ほとんどの Dell システムで必要になります。コンピューターのマニュアルを確認してください。

2. クラスターの Kubernetes マニフェストを生成します。

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

3. **openshift** ディレクトリーで、マスターまたはワーカーファイルを作成し、それらのノードのディスクを暗号化します。

   - ワーカーファイルを作成するには、以下のコマンドを実行します。

```
$ cat << EOF > ./99-openshift-worker-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tpm
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

   - マスターファイルを作成するには、以下のコマンドを実行します。

```
$ cat << EOF > ./99-openshift-master-tpmv2-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
```

```
metadata:
  name: master-tpm
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

4. YAML ファイルのバックアップコピーを作成します。ファイルはクラスターの作成時に削除されるため、これを実行する必要があります。

5. 残りの OpenShift Container Platform のデプロイメントを継続します。

13.2.3.2. Tang ディスク暗号化の有効化

以下の手順を使用して、OpenShift Container Platform のデプロイメント時に Tang モードのディスク暗号化を有効にします。

手順

1. 暗号化の設定を設定し、**openshift-install** を実行してクラスターをインストールし、**oc** を使用してクラスターを操作するために Red Hat Enterprise Linux サーバーにアクセスします。

2. 既存の Tang サーバーを設定するか、またはこれにアクセスします。手順については、NBDE (Network-Bound Disk Encryption) を参照してください。タグの表示についての詳細は、Securing Automated Decryption New Cryptography and Techniques を参照してください。

3. クラスターについて Red Hat Enterprise Linux CoreOS (RHCOS) インストールを実行する際にネットワークを設定するためにカーネル引数を追加します。たとえば、DHCP ネットワークを設定するには、**ip=dhcp** を特定するか、またはカーネルコマンドラインにパラメーターを追加する際に静的ネットワークを設定します。DHCP と静的ネットワークの両方の場合、**rd.neednet=1** カーネル引数も指定する必要があります。

重要

このステップを省略すると、2 番目の起動に失敗します。

4. **clevis** パッケージがインストールされていない場合はインストールします。

```
$ sudo yum install clevis -y
```

5. Tang サーバーからサムプリントを生成します。

   a. 以下のコマンドでは、**url** の値を Tang サーバーの URL に置き換えます。

```
$ echo nifty random wordwords \
    | clevis-encrypt-tang \
     '{"url":"https://tang.example.org"}'
```

出力例

```
The advertisement contains the following signing keys:

PLjNyRdGw03zlRoGjQYMahSZGu9
```

b. **Do you want to trust these key? [ynYN]** プロンプトが表示されたら、**Y** を入力します。その後サムプリントが表示されます。

出力例

```
eyJhbmc3SlRyMXpPenc3ajhEQ01tZVJiTi1oM...
```

6. Base64 でエンコードされたファイルを作成します。値を Tang サーバーの URL (**url**) と生成したサムプリント (**thp**) で置き換えます。

```
$ (cat <<EOM
{
 "url": "https://tang.example.com",
 "thp": "PLjNyRdGw03zlRoGjQYMahSZGu9"
}
EOM
) | base64 -w0
```

出力例

```
ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogInRocCI6ICJaUk1leTFjR3cw
N3psVExHYlhuUWFoUzBHdTAiCn0K
```

7. **openshift** ディレクトリーで、マスターまたはワーカーファイルを作成し、それらのノードのディスクを暗号化します。

- ワーカーノードの場合は、以下のコマンドを使用します。

```
$ cat << EOF > ./99-openshift-worker-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: worker-tang
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
```

```
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogIn
RocCI6ICJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

- マスターノードの場合は、以下のコマンドを使用します。

```
$ cat << EOF > ./99-openshift-master-tang-encryption.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: master-tang
  labels:
    machineconfiguration.openshift.io/role: master
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;base64,e30K
          source:
data:text/plain;base64,ewogInVybCI6ICJodHRwczovL3RhbmcuZXhhbXBsZS5jb20iLAogIn
RocCI6ICJaUk1leTFjR3cwN3psVExHYlhuUWFoUzBHdTAiCn0K
        filesystem: root
        mode: 420
        path: /etc/clevis.json
EOF
```

8. 以下の例で示すように **rd.neednet=1** カーネル引数を追加します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: <node_type>-tang (1)
spec:
  config:
    ignition:
      version: 3.1.0
  kernelArguments:
    - rd.neednet=1 (2)
```

(1) 設定しているノードタイプに基づいて、上記の例で定義した名前を使用します (例: **name: worker-tang**)。

(2) 必須。

9. 残りの OpenShift Container Platform のデプロイメントを継続します。

13.2.4. chrony タイムサービスの設定

chrony タイムサービス (**chronyd**) で使用されるタイムサーバーおよび関連する設定は、**chrony.conf** ファイルのコンテンツを変更し、それらのコンテンツをマシン設定としてノードに渡して設定できます。

手順

1. **chrony.conf** ファイルのコンテンツを作成し、これを base64 でエンコードします。以下に例を示します。

```
$ cat << EOF | base64
   pool 0.rhel.pool.ntp.org iburst (1)
   driftfile /var/lib/chrony/drift
   makestep 1.0 3
   rtcsync
   logdir /var/log/chrony
EOF
```

(1) DHCP サーバーが提供するものなど、有効な到達可能なタイムソースを指定します。または、NTP サーバーの **1.rhel.pool.ntp.org**、**2.rhel.pool.ntp.org**、または **3.rhel.pool.ntp.org** のいずれかを指定できます。

出力例

```
ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92YXIvbGli
L2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAgIGxvZ2Rp
RpciAv
dmFyL2xvZy9jaHJvbnkK
```

2. **MachineConfig** ファイルを作成します。base64 文字列を独自に作成した文字列に置き換えます。この例では、ファイルを **master** ノードに追加します。これを **worker** に切り替えたり、**worker** ロールの追加の MachineConfig を作成したりできます。クラスターが使用するそれぞれのタイプのマシンについて MachineConfig ファイルを作成します。

```
$ cat << EOF > ./99-masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-masters-chrony-configuration
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
```

```
8;base64,ICAgIHNlcnZlciBjbG9jay5yZWRoYXQuY29tIGlidXJzdAogICAgZHJpZnRmaWxlIC92Y
XIvbGliL2Nocm9ueS9kcmlmdAogICAgbWFrZXN0ZXAgMS4wIDMKICAgIHJ0Y3N5bmMKICAg
IGxvZ2RpciAvdmFyL2xvZy9jaHJvbnkK
        verification: {}
      filesystem: root
      mode: 420
      path: /etc/chrony.conf
  osImageURL: ""
EOF
```

3. 設定ファイルのバックアップコピーを作成します。
4. 以下の 2 つの方法のいずれかで設定を適用します。
   - クラスターがまだ起動していない場合は、マニフェストファイルを生成した後に、そのファイルを **<installation_directory>/openshift** ディレクトリーに追加してから、クラスターの作成を継続します。
   - クラスターがすでに実行中の場合は、ファイルを適用します。

     ```
     $ oc apply -f ./99-masters-chrony-configuration.yaml
     ```

13.2.5. 関連情報

FIPS サポートについての詳細は、FIPS 暗号のサポート を参照してください。

13.3. 利用可能なクラスターのカスタマイズ

OpenShift Container Platform クラスターのデプロイ後は、大半のクラスター設定およびカスタマイズが終了していることになります。数多くの**設定リソース**が利用可能です。

イメージレジストリー、ネットワーク設定、イメージビルドの動作およびアイデンティティープロバイダーなどのクラスターの主要な機能を設定するために設定リソースを変更します。

これらのリソースを使用して制御する設定の現在の記述については、 **oc explain** コマンドを使用します (例: **oc explain builds --api-version=config.openshift.io/v1**)。

13.3.1. クラスター設定リソース

すべてのクラスター設定リソースはグローバルにスコープが設定され (namespace は設定されない)、**cluster** という名前が付けられます。

リソース名	説明
apiserver.config.openshift.io	証明書および認証局 などの API サーバー設定を提供します。
authentication.config.openshift.io	クラスターの アイデンティティープロバイダー および認証設定を制御します。

リソース名	説明
build.config.openshift.io	クラスターのすべてのビルドについてのデフォルトおよび有効にされている 設定 を制御します。
console.config.openshift.io	ログアウト動作 を含む Web コンソールインターフェイスの動作を設定します。
featuregate.config.openshift.io	FeatureGates を有効にして、テクノロジープレビュー機能を使用できるようにします。
image.config.openshift.io	特定の イメージレジストリー が処理される方法を設定します (allowed、disallowed、insecure、CA details)。
ingress.config.openshift.io	ルートのデフォルトドメインなどの ルーティング に関連する設定の詳細。
oauth.config.openshift.io	内部 OAuth サーバー フローに関連するアイデンティティープロバイダーおよび他の動作を設定します。
project.config.openshift.io	プロジェクトテンプレートを含む、プロジェクトの作成方法 を設定します。
proxy.config.openshift.io	外部ネットワークアクセスを必要とするコンポーネントで使用されるプロキシーを定義します。注: すべてのコンポーネントがこの値を使用する訳ではありません。
scheduler.config.openshift.io	ポリシーやデフォルトノードセレクターなどの スケジューラー の動作を設定します。

13.3.2. Operator 設定リソース

これらの設定リソースは、**cluster** という名前のクラススコープのインスタンスです。これは、特定の Operator によって所有される特定コンポーネントの動作を制御します。

リソース名	説明
console.operator.openshift.io	ブランドのカスタマイズなどのコンソールの外観の制御
config.imageregistry.operator.openshift.io	パブリックルーティング、プロキシー設定、リソース設定、レプリカ数およびストレージタイプなどの 内部イメージレジストリー設定 を設定します。
config.samples.operator.openshift.io	Samples Operator を設定して、クラスターにインストールされるイメージストリームとテンプレートのサンプルを制御します。

13.3.3. 追加の設定リソース

これらの設定リソースは、特定コンポーネントの単一インスタンスを表します。場合によっては、リソースの複数のインスタンスを作成して、複数のインスタンスを要求できます。他の場合には、Operator は特定の namespace の特定のリソースインスタンス名のみを使用できます。追加のリソースインスタンスの作成方法や作成するタイミングについての詳細は、コンポーネント固有のドキュメントを参照してください。

リソース名	インスタンス名	Namespace	説明
alertmanager.monitoring.coreos.com	**main**	**openshift-monitoring**	Alertmanager デプロイメントパラメーターを制御します。
ingresscontroller.operator.openshift.io	**default**	**openshift-ingress-operator**	ドメイン、レプリカ数、証明書、およびコントローラーの配置などの Ingress Operator 動作を設定します。

13.3.4. 情報リソース

これらのリソースを使用して、クラスターについての情報を取得します。これらのリソースは直接編集しないでください。

リソース名	インスタンス名	説明
clusterversion.config.openshift.io	**version**	OpenShift Container Platform 4.5 では、実稼働クラスターの **ClusterVersion** リソースをカスタマイズすることはできません。その代わりとして、 クラスターの更新 プロセスを実行します。
dns.config.openshift.io	**cluster**	クラスターの DNS 設定を変更することはできません。DNS Operator ステータスを表示 できます。
infrastructure.config.openshift.io	**cluster**	クラスターはそのクラウドプロバイダーとの対話を可能にする設定の詳細。
network.config.openshift.io	**cluster**	インストール後にクラスターのネットワークを変更することはできません。ネットワークをカスタマイズするには、インストール時にネットワークをカスタマイズ するプロセスを実行します。

13.3.5. グローバルクラスターのプルシークレットの更新

クラスターのグローバルプルシークレットを更新できます。

警告

クラスターリソースは新規のプルシークレットに合わせて調整する必要がありますが、これにより、クラスターのユーザービリティーが一時的に制限される可能性があります。

警告

グローバルプルシークレットを更新すると、Machine Config Operator (MCO) が変更を同期している間にノードが再起動します。

前提条件

- アップロードする新規または変更されたプルシークレットファイルがある。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- 以下のコマンドを実行して、クラスターのグローバルプルシークレットを更新します。

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=<pull-secret-location> 1
```

1 新規プルシークレットファイルへのパスを指定します。

この更新はすべてのノードにロールアウトされます。これには、クラスターのサイズに応じて多少時間がかかる場合があります。この間に、ノードがドレイン (解放) され、Pod は残りのノードで再スケジュールされます。

13.4. ファイアウォールの設定

ファイアウォールを使用する場合、OpenShift Container Platform が機能するために必要なサイトにアクセスできるように設定する必要があります。一部のサイトにはアクセスを常に付与し、クラスターをホストするために Red Hat Insights、Telemetry サービス、クラウドを使用したり、特定のビルドストラテジーをホストする場合に追加のアクセスを付与する必要があります。

13.4.1. OpenShift Container Platform のファイアウォールの設定

OpenShift Container Platform をインストールする前に、ファイアウォールを、OpenShift Container Platform が必要とするサイトへのアクセスを付与するように設定する必要があります。

コントローラーノードのみで実行されるサービスとワーカーノードで実行されるサービスの設定に関する特別な考慮事項はありません。

手順

1. 以下のレジストリー URL を許可リストに指定します。

URL	ポート	機能
registry.redhat.io	443、80	コアコンテナーイメージを指定します。
quay.io	443、80	コアコンテナーイメージを指定します。
***.quay.io**	443, 80	コアコンテナーイメージを指定します。
sso.redhat.com	443, 80	**https://cloud.redhat.com/openshift** サイトでは、 **sso.redhat.com** からの認証を使用します。
openshift.org	443, 80	Red Hat Enterprise Linux CoreOS (RHCOS) イメージを提供します。

   quay.io などのサイトを許可リストに追加するには、***.quay.io** などのワイルドカードエントリーを拒否リストに加えないでください。ほとんどの場合、イメージレジストリーはコンテンツ配信ネットワーク (CDN) を使用してイメージを提供します。ファイアウォールがアクセスをブロックすると、初回のダウンロード要求が **cdn01.quay.io** などのホスト名にリダイレクトされると、イメージのダウンロードが拒否されます。

   cdn01.quay.io などの CDN ホスト名は、許可リストに ***.quay.io** などのワイルドカードエントリーを追加する場合に説明されます。

2. ビルドに必要な言語またはフレームワークのリソースを提供するサイトを許可リストに指定します。

3. Telemetry を無効にしていない場合は、以下の URL へのアクセスを許可して Red Hat Insights にアクセスできるようにする必要があります。

URL	ポート	機能
cert-api.access.redhat.com	443、80	Telemetry で必須
api.access.redhat.com	443、80	Telemetry で必須
infogw.api.openshift.com	443、80	Telemetry で必須
https://cloud.redhat.com/api/ingress	443, 80	Telemetry および **insights-operator** で必須

4. Amazon Web Services (AWS) 、Microsoft Azure、または Google Cloud Platform (GCP) を使用してクラスターをホストする場合、クラウドプロバイダー API およびそのクラウドの DNS を提供する URL へのアクセスを付与する必要があります。

クラウド	URL	ポート	機能
AWS	***.amazonaws.com**	443、80	AWS サービスおよびリソースへのアクセスに必要です。AWS ドキュメントの AWS Service Endpoints を参照し、使用するリージョンを許可するエンドポイントを判別します。
	oso-rhc4tp-docker-registry.s3-us-west-2.amazonaws.com	443、80	厳密なセキュリティー要件を適用する際に AWS サービスおよびリソースにアクセスするために必要です。AWS ドキュメントの AWS Service Endpoints を参照し、使用するリージョンを許可するエンドポイントを判別します。
GCP	***.googleapis.com**	443、80	GCP サービスおよびリソースへのアクセスに必要です。GCP ドキュメントの Cloud Endpoints を参照し、API を許可するエンドポイントを判別します。
	accounts.google.com	443、80	GCP アカウントへのアクセスに必要です。
Azure	**management.azure.com**	443、80	Azure サービスおよびリソースへのアクセスに必要です。Azure ドキュメントで Azure REST API Reference を参照し、API を許可するエンドポイントを判別します。

5. 以下の URL を許可リストに指定します。

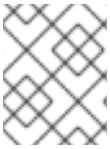
URL	ポート	機能
mirror.openshift.com	443、80	ミラーリングされたインストールのコンテンツおよびイメージへのアクセスに必要。Cluster Version Operator には単一の機能ソースのみが必要ですが、このサイトはリリースイメージ署名のソースでもあります。
storage.googleapis.com/openshift-release	443、80	リリースイメージ署名のソース (ただし、Cluster Version Operator には単一の機能ソースのみが必要)。
***.apps.<cluster_name>.<base_domain>**	443、80	Ingress ワイルドカードをインストール時に設定しない限り、デフォルトのクラスタールートへのアクセスに必要。

URL	ポート	機能
quay-registry.s3.amazonaws.com	443, 80	AWS で Quay イメージコンテンツにアクセスするために必要。
api.openshift.com	443, 80	クラスターに更新が利用可能かどうかを確認するために必要。
art-rhcos-ci.s3.amazonaws.com	443, 80	Red Hat Enterprise Linux CoreOS (RHCOS) イメージをダウンロードするために必要。
api.openshift.com	443, 80	クラスタートークンに必須
cloud.redhat.com/openshift	443, 80	クラスタートークンに必須
registry.access.redhat.com	443、80	**odo** CLI に必須

Operator にはヘルスチェックを実行するためのルートアクセスが必要です。具体的には、認証および Web コンソール Operator は 2 つのルートに接続し、ルートが機能することを確認します。クラスター管理者として操作を実行しており、***.apps.<cluster_name>.<base_domain>** を許可しない場合は、これらのルートを許可します。

- **oauth-openshift.apps.<cluster_name>.<base_domain>**
- **console-openshift-console.apps.<cluster_name>.<base_domain>**、またはフィールドが空でない場合に **consoles.operator/cluster** オブジェクトの **spec.route.hostname** フィールドに指定されるホスト名。

6. デフォルトの Red Hat Network Time Protocol (NTP) サーバーを使用する場合は、以下の URL を許可します。
   - **1.rhel.pool.ntp.org**
   - **2.rhel.pool.ntp.org**
   - **3.rhel.pool.ntp.org**

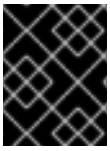


注記

デフォルトの Red Hat NTP サーバーを使用しない場合は、プラットフォームの NTP サーバーを確認し、ファイアウォールでこれを許可します。

13.5. プライベートクラスターの設定

OpenShift Container Platform バージョン 4.5 クラスターのインストール後に、そのコアコンポーネントの一部を private に設定できます。



重要

この変更は、クラウドプロバイダーにプロビジョニングするインフラストラクチャーを使用するクラスターにのみ設定できます。

13.5.1. プライベートクラスター

デフォルトで、OpenShift Container Platform は一般にアクセス可能な DNS およびエンドポイントを使用してプロビジョニングされます。クラスターのデプロイ後に DNS、Ingress コントローラー、および API サーバーを private に設定できます。

DNS
OpenShift Container Platform をインストーラーでプロビジョニングされるインフラストラクチャーにインストールする場合、インストールプログラムは既存のパブリックゾーンにレコードを作成し、可能な場合はクラスター独自の DNS 解決用のプライベートゾーンを作成します。パブリックゾーンおよびプライベートゾーンの両方で、インストールプログラムまたはクラスターが **Ingress** オブジェクトの ***.apps**、および API サーバーの **api** の DNS エントリーを作成します。

***.apps** レコードはパブリックゾーンとプライベートゾーンのどちらでも同じであるため、パブリックゾーンを削除する際に、プライベートゾーンではクラスターのすべての DNS 解決をシームレスに提供します。

Ingress コントローラー
デフォルトの **Ingress** オブジェクトはパブリックとして作成されるため、ロードバランサーはインターネットに接続され、パブリックサブネットで使用されます。デフォルト Ingress コントローラーは内部コントローラーに置き換えることができます。

API サーバー
デフォルトでは、インストールプログラムは内部トラフィックと外部トラフィックの両方で使用するための API サーバーの適切なネットワークロードバランサーを作成します。

Amazon Web Services (AWS) では、個別のパブリックロードバランサーおよびプライベートロードバランサーが作成されます。ロードバランサーは、クラスター内で使用するために追加ポートが内部で利用可能な場合を除き、常に同一です。インストールプログラムは API サーバー要件に基づいてロードバランサーを自動的に作成または破棄しますが、クラスターはそれらを管理または維持しません。クラスターの API サーバーへのアクセスを保持する限り、ロードバランサーを手動で変更または移動できます。パブリックロードバランサーの場合、ポート 6443 は開放され、ヘルスチェックが HTTPS について **/readyz** パスに対して設定されます。

Google Cloud Platform では、内部および外部 API トラフィックの両方を管理するために単一のロードバランサーが作成されるため、ロードバランサーを変更する必要はありません。

Microsoft Azure では、パブリックおよびプライベートロードバランサーの両方が作成されます。ただし、現在の実装には制限があるため、プライベートクラスターで両方のロードバランサーを保持します。

13.5.2. DNS をプライベートに設定する

クラスターのデプロイ後に、プライベートゾーンのみを使用するように DNS を変更できます。

手順

1. クラスターの **DNS** カスタムリソースを確認します。

```
$ oc get dnses.config.openshift.io/cluster -o yaml
```

出力例

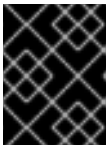
```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>: owned
  publicZone:
    id: Z2XXXXXXXXXXA4
status: {}
```

spec セクションには、プライベートゾーンとパブリックゾーンの両方が含まれることに注意してください。

2. **DNS** カスタムリソースにパッチを適用して、パブリックゾーンを削除します。

```
$ oc patch dnses.config.openshift.io/cluster --type=merge --patch='{"spec": {"publicZone": null}}'
dns.config.openshift.io/cluster patched
```

Ingress コントローラーは **Ingress** オブジェクトの作成時に **DNS** 定義を参照するため、**Ingress** オブジェクトを作成または変更する場合、プライベートレコードのみが作成されます。



重要

既存の Ingress オブジェクトの DNS レコードは、パブリックゾーンの削除時に変更されません。

3. オプション: クラスターの **DNS** カスタムリソースを確認し、パブリックゾーンが削除されていることを確認します。

```
$ oc get dnses.config.openshift.io/cluster -o yaml
```

出力例

```
apiVersion: config.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-10-25T18:27:09Z"
  generation: 2
  name: cluster
  resourceVersion: "37966"
  selfLink: /apis/config.openshift.io/v1/dnses/cluster
```

```
  uid: 0e714746-f755-11f9-9cb1-02ff55d8f976
spec:
  baseDomain: <base_domain>
  privateZone:
    tags:
      Name: <infrastructureID>-int
      kubernetes.io/cluster/<infrastructureID>-wfpg4: owned
status: {}
```

13.5.3. Ingress コントローラーをプライベートに設定する

クラスターのデプロイ後に、その Ingress コントローラーをプライベートゾーンのみを使用するように変更できます。

手順

1. 内部エンドポイントのみを使用するようにデフォルト Ingress コントローラーを変更します。

```
$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal
EOF
```

出力例

```
ingresscontroller.operator.openshift.io "default" deleted
ingresscontroller.operator.openshift.io/default replaced
```

パブリック DNS エントリーが削除され、プライベートゾーンエントリーが更新されます。

13.5.4. API サーバーをプライベートに制限する

クラスターを Amazon Web Services (AWS) または Microsoft Azure にデプロイした後に、プライベートゾーンのみを使用するように API サーバーを再設定することができます。

前提条件

- OpenShift CLI (**oc**) をインストールしている。
- **admin** 権限を持つユーザーとして Web コンソールにアクセスできること。

手順

1. AWS または Azure の Web ポータルまたはコンソールで、以下のアクションを実行します。
   a. 適切なロードバランサーコンポーネントを見つけ、削除します。

- AWS の場合は、外部ロードバランサーを削除します。プライベートゾーンの API DNS エントリーは、同一の設定を使用する内部ロードバランサーをすでに参照するため、内部ロードバランサーを変更する必要はありません。
- Azure の場合、ロードバランサーの **api-internal** ルールを削除します。

b. パブリックゾーンの **api.$clustername.$yourdomain** DNS エントリーを削除します。

2. ターミナルで、クラスターマシンを一覧表示します。

```
$ oc get machine -n openshift-machine-api
```

出力例

```
NAME                      STATE    TYPE       REGION     ZONE       AGE
lk4pj-master-0            running  m4.xlarge  us-east-1  us-east-1a  17m
lk4pj-master-1            running  m4.xlarge  us-east-1  us-east-1b  17m
lk4pj-master-2            running  m4.xlarge  us-east-1  us-east-1a  17m
lk4pj-worker-us-east-1a-5fzfj  running  m4.xlarge  us-east-1  us-east-1a  15m
lk4pj-worker-us-east-1a-vbghs  running  m4.xlarge  us-east-1  us-east-1a  15m
lk4pj-worker-us-east-1b-zgpzg  running  m4.xlarge  us-east-1  us-east-1b  15m
```

以下の手順で、名前に **master** が含まれるコントロールプレーンマシンを変更します。

3. 各コントロールプレーンマシンから外部ロードバランサーを削除します。

a. コントロールプレーンの **Machine** オブジェクトを編集し、外部ロードバランサーへの参照を削除します。

```
$ oc edit machines -n openshift-machine-api <master_name> 1
```

1 変更するコントロールプレーン (またはマスター) **Machine** オブジェクトの名前を指定します。

b. 以下の例でマークが付けられている外部ロードバランサーを記述する行を削除し、オブジェクト仕様を保存し、終了します。

```
...
spec:
  providerSpec:
    value:
      ...
        loadBalancers:
        - name: lk4pj-ext 1
          type: network 2
        - name: lk4pj-int
          type: network
```

1 2 この行を削除します。

c. 名前に **master** が含まれるマシンにこのプロセスを繰り返します。

第14章 インストールの問題のトラブルシューティング

OpenShift Container Platform のインストールした場合のトラブルシューティングのために、ブートストラップおよびコントロールプレーン (またはマスター) マシンからログを収集できます。インストールプログラムからデバッグ情報を取得することもできます。

14.1. 前提条件

- OpenShift Container Platform クラスターのインストールを試みたが、インストールに失敗している。

14.2. 失敗したインストールのログの収集

SSH キーをインストールプログラムに指定している場合、失敗したインストールについてのデータを収集することができます。

注記

実行中のクラスターからログを収集する場合とは異なるコマンドを使用して失敗したインストールについてのログを収集します。実行中のクラスターからログを収集する必要がある場合は、**oc adm must-gather** コマンドを使用します。

前提条件

- OpenShift Container Platform のインストールがブートストラッププロセスの終了前に失敗している。ブートストラップノードは実行中であり、SSH でアクセスできる。
- **ssh-agent** プロセスはコンピューター上でアクティブであり、**ssh-agent** プロセスとインストールプログラムの両方に同じ SSH キーを提供している。
- 独自にプロビジョニングしたインフラストラクチャーにクラスターのインストールを試行した場合には、ブートストラップおよびマスターノードの完全修飾ドメイン名がある。

手順

1. ブートストラップおよびコントロールプレーンマシンからインストールログを収集するために必要なコマンドを生成します。
   - インストーラーでプロビジョニングされるインフラストラクチャーを使用している場合は、以下のコマンドを実行します。

     ```
     $ ./openshift-install gather bootstrap --dir=<installation_directory> 1
     ```

     1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

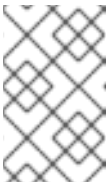
     インストーラーでプロビジョニングされるインフラストラクチャーの場合、インストールプログラムは、ホスト名または IP アドレスを指定しなくてもよいようにクラスターについての情報を保存します。
   - 独自にプロビジョニングしたインフラストラクチャーを使用している場合は、以下のコマンドを実行します。

```
$ ./openshift-install gather bootstrap --dir=<installation_directory> \ 1
  --bootstrap <bootstrap_address> \ 2
  --master <master_1_address> \ 3
  --master <master_2_address> \ 4
  --master <master_3_address>" 5
```

1. **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2. **<bootstrap_address>** は、クラスターのブートストラップマシンの完全修飾ドメイン名または IP アドレスです。

3 4 5. クラスター内のそれぞれのコントロールプレーン (またはマスター) マシンについては、**<master_*_address>** をその完全修飾ドメイン名または IP アドレスに置き換えます。



注記

デフォルトクラスターには 3 つのコントロールプレーンマシンが含まれます。クラスターが使用する数にかかわらず、表示されるようにすべてのコントロールプレーンマシンを一覧表示します。

出力例

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

インストールの失敗についての Red Hat サポートケースを作成する場合は、圧縮したログをケースに含めるようにしてください。

14.3. ホストへの SSH アクセスによるログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

前提条件

- ホストへの SSH アクセスがあること。

手順

1. 以下を実行し、**journalctl** コマンドを使用してブートストラップホストから **bootkube.service** サービスログを収集します。

```
$ journalctl -b -f -u bootkube.service
```

2. podman ログを使用して、ブートストラップホストのコンテナーログを収集します。これは、ホストからすべてのコンテナーログを取得するためにループで表示されます。

```
$ for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done
```

3. または、以下を実行し、**tail** コマンドを使用してホストのコンテナーログを収集します。

```
# tail -f /var/lib/containers/storage/overlay-containers/*/userdata/ctr.log
```

4. 以下を実行し、**journalctl** コマンドを使用して **kubelet.service** および **crio.service** サービスログをマスターホストおよびワーカーホストから収集します。

```
$ journalctl -b -f -u kubelet.service -u crio.service
```

5. 以下を実行し、**tail** コマンドを使用してマスターホストおよびワーカーホストのコンテナーログを収集します。

```
$ sudo tail -f /var/log/containers/*
```

14.4. ホストへの SSH アクセスを使用しないログの手動収集

must-gather または自動化された収集方法が機能しない場合にログを手動で収集します。

ノードへの SSH アクセスがない場合は、システムジャーナルにアクセスし、ホストで生じていることを調査できます。

前提条件

- OpenShift Container Platform のインストールが完了している。
- API サービスが機能している。
- システム管理者権限がある。

手順

1. 以下を実行し、**/var/log** の下にある **journald** ユニットログにアクセスします。

```
$ oc adm node-logs --role=master -u kubelet
```

2. 以下を実行し、**/var/log** の下にあるホストファイルのパスにアクセスします。

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

14.5. インストールプログラムからのデバッグ情報の取得

以下のアクションのいずれかを使用して、インストールプログラムからデバッグ情報を取得できます。

- 非表示の **.openshift_install.log** ファイルで過去のインストールからのデバッグ情報を確認します。たとえば、 以下を入力します。

```
$ cat ~/<installation_directory>/.openshift_install.log 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

- **--log-level=debug** を使用してインストールプログラムを再実行します。

```
$ ./openshift-install create cluster --dir=<installation_directory> --log-level=debug 1
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。

第15章 FIPS 暗号のサポート

バージョン 4.3 以降、FIPS で検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーを使用する OpenShift Container Platform クラスターをインストールすることができます。

クラスター内の Red Hat Enterprise Linux CoreOS (RHCOS) マシンの場合、この変更は、ユーザーがクラスターのデプロイメント時に変更できるクラスターオプションを制御する **install-config.yaml** ファイルのオプションのステータスに基づいてマシンがデプロイされる際に適用されます。Red Hat Enterprise Linux (RHEL) マシンでは、ワーカーマシンとして使用する予定のマシンにオペレーティングシステムをインストールする場合に FIPS モードを有効にする必要があります。これらの設定方法により、クラスターが FIPS コンプライアンス監査の要件を満たすことを確認できます。初期システムの起動前は、FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号パッケージのみが有効になります。

FIPS はクラスターが使用するオペレーティングシステムの初回の起動前に有効にされている必要があり、クラスターをデプロイしてから FIPS を有効にすることはできません。

15.1. OPENSHIFT CONTAINER PLATFORM での FIPS 検証

OpenShift Container Platform は、それが使用するオペレーティングシステムのコンポーネント用に RHEL および RHCOS 内の特定の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用します。RHEL7 core crypto components を参照してください。たとえば、ユーザーが OpenShift Container Platform クラスターおよびコンテナーに対して SSH を実行する場合、それらの接続は適切に暗号化されます。

OpenShift Container Platform コンポーネントは Go で作成され、Red Hat の golang コンパイラーを使用してビルドされます。クラスターの FIPS モードを有効にすると、暗号署名を必要とするすべての OpenShift Container Platform コンポーネントは RHEL および RHCOS 暗号ライブラリーを呼び出します。

表15.1 OpenShift Container Platform 4.5 における FIPS モード属性および制限

属性	制限
RHEL 7 オペレーティングシステムの FIPS サポート。	FIPS 実装は、ハッシュ関数を計算し、そのハッシュに基づくキーを検証する単一の機能を提供しません。この制限については、今後の OpenShift Container Platform リリースで継続的に評価され、改善されます。
CRI-O ランタイムの FIPS サポート。	
OpenShift Container Platform サービスの FIPS サポート。	
RHEL 7 および RHCOS バイナリーおよびイメージから取得される FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号化モジュールおよびアルゴリズム。	
FIPS と互換性のある golang コンパイラーの使用。	TLS FIPS サポートは完全に実装されていませんが、今後の OpenShift Container Platform リリースで予定されています。

15.2. クラスターが使用するコンポーネントでの FIPS サポート

OpenShift Container Platform クラスター自体は FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用しますが、OpenShift Container Platform クラスターをサポートするシステムが暗号化の FIPS 検証済み/進行中のモジュール (Modules in Process) モジュールを使用していることを確認してください。

15.2.1. etcd

etcd に保存されるシークレットが FIPS 検証済み/進行中のモジュール (Modules in Process) の暗号を使用できるようにするには、ノードを FIPS モードで起動します。クラスターを FIPS モードでインストールした後に、 FIPS で承認される **aes cbc** 暗号アルゴリズムを使用して etcd データを暗号化 できます。

15.2.2. ストレージ

ローカルストレージの場合は、RHEL が提供するディスク暗号化または RHEL が提供するディスク暗号化を使用する Container Native Storage を使用します。RHEL が提供するディスク暗号を使用するボリュームにすべてのデータを保存し、クラスター用に FIPS モードを有効にすることで、移動しないデータと移動するデータまたはネットワークデータは FIPS の検証済み/進行中のモジュール (Modules in Process) の暗号化によって保護されます。ノードのカスタマイズ で説明されているように、各ノードのルートファイルシステムを暗号化するようにクラスターを設定できます。

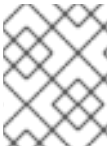
15.2.3. ランタイム

コンテナーに対して FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号モジュールを使用しているホストで実行されていることを認識させるには、CRI-O を使用してランタイムを管理します。CRI-O は FIPS モードをサポートします。このモードでは、コンテナーが FIPS モードで実行されていることを認識できるように設定されます。

15.3. FIPS モードでのクラスターのインストール

FIPS モードでクラスターをインストールするには、必要なインフラストラクチャーにカスタマイズされたクラスターをインストールする方法についての説明に従ってください。クラスターをデプロイする前に、**fips: true** を **install-config.yaml** ファイルに設定していることを確認します。

- Amazon Web Services
- Microsoft Azure
- ベアメタル
- Google Cloud Platform
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere



注記

Azure File ストレージを使用している場合、FIPS モードを有効にすることはできません。

AES CBC 暗号化を etcd データストアに適用するには、クラスターのインストール後に etcd データの暗号化 プロセスに従ってください。

RHEL ノードをクラスターに追加する場合は、初回の起動前に FIPS モードをマシン上で有効にしていることを確認してください。RHEL コンピュートマシンの OpenShift Container Platform クラスターへの追加、およびRHEL 7 ドキュメントの FIPS モードの有効化 を参照してください。