OpenShift Container Platform 3.11

クラスターのインストール

OpenShift Container Platform 3.11 クラスターのインストール

Last Updated: 2023-03-25

OpenShift Container Platform 3.11 クラスターのインストール

OpenShift Container Platform 3.11 クラスターのインストール

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

概要

本書を活用した OpenShift Container Platform 3.11 クラスターのインストール

目次

第1章 インストール計画

OpenShift Container Platform は一連の Ansible Playbook を実行してインストールします。クラスターのインストールを準備する際に、環境および OpenShift Container Platform クラスター設定を表すインベントリーファイルを作成します。Ansible についての知識があるとこのプロセスがより容易になりますが、必須ではありません。

Ansible およびその基本的な使用方法については、公式ドキュメント を参照してください。

1.1. 初期計画

実稼働環境用の OpenShift Container Platform クラスターをインストールする前に、以下の質問について検討してください。

- **ご使用のオンプレミスサーバーで IBM POWER または x86_64 プロセッサーを使用しているか？いずれかのタイプのプロセッサーを使用するサーバーに OpenShift Container Platform をインストールできます。**いずれかのプロセッサーを使用するサーバーに OpenShift Container Platform をインストールできます。POWER サーバーを使用する場合は、IBM POWER でのインストールの制限と考慮事項 を確認してください。

- **クラスターで必要な pod の数**サイジングに関する考慮事項 セクションでは、ノードと pod の制限について説明します。これでは、必要な環境の規模を計算することができます。

- **クラスターに必要なホストの数はいくつあるか？**環境シナリオ セクションでは、単一マスターおよび複数マスター設定の複数の設定例について説明します。

- **クラスターに 高可用性 は必要か？**高可用性の設定はフォールトトレランスを改善します。高可用性の設定により、耐障害性が向上します。この場合、環境を設定するために ネイティブの高可用性 (HA) を使用する複数マスター のサンプルの使用を検討されるかもしれません。

- **クラスターモニターリング は必要か？**モニターリングスタックには、追加の システムリソース が必要です。モニターリングスタックは、デフォルトでインストールされていることに注意してください。詳細は、クラスターモニターリングのドキュメント を参照してください。

- **クラスターノードのオペレーティングシステムに Red Hat Enterprise Linux (RHEL) または RHEL Atomic Host を使用する必要があるか？OpenShift Container Platform を RHEL にインストールする場合、RPM ベースのインストールを使用します。**RHEL Atomic Host では、システムコンテナーを使用します。RHEL Atomic Host では、システムコンテナーを使用します。どちらのインストールタイプ も機能する OpenShift Container Platform 環境を提供します。

- **認証に使用する アイデンティティープロバイダー**サポートされているアイデンティティープロバイダーをすでに使用している場合は、インストール時にそのアイデンティティープロバイダーを使用するよう OpenShift Container Platform を設定します。

- **他のテクノロジーと統合する場合は、インストールがサポートされますか？**テスト済みの統合の一覧は、OpenShift Container Platform のテスト済みインテグレーション を参照してください。

1.1.1. IBM POWER でのインストールについての制限および考慮事項

バージョン 3.10.45 の時点では、OpenShift Container Platform を IBM POWER サーバーにインストールできます。

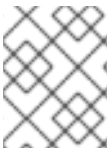
- クラスターは Power ノードおよびマスターのみを使用する必要があります。イメージへのタグの付け方により、OpenShift Container Platform では x86 イメージと Power イメージを区別することができません。
- イメージストリームおよびテンプレートは、アップグレード時にデフォルトでインストールされず、更新されません。イメージストリームは手動でインストールし、更新することができます。
- オンプレミス Power サーバーにのみインストールできます。OpenShift Container Platform をクラウドプロバイダーのノードにインストールすることはできません。
- すべてのストレージプロバイダーがサポートされている訳ではありません。以下のストレージプロバイダーのみを使用できます。
  - GlusterFS
  - NFS
  - ローカルストレージ

1.2. サイジングに関する考慮事項

OpenShift Container Platform クラスターに必要なノードと Pod の数を判別します。クラスターの拡張性はクラスター環境内の Pod の数に相関します。この数は、セットアップの他の数に影響を及ぼします。OpenShift Container Platform のオブジェクトの制限についての最新情報は、クラスターの制限 を参照してください。

1.3. 環境シナリオ

これらの環境シナリオは、実際のサイジングの必要に応じて独自の OpenShift Container Platform クラスターを計画する際に使用してください。



注記

インストール後の単一マスタークラスターから複数マスターへの移行はサポートされていません。

すべての環境において、etcd ホストがマスターホストと同じ場所にある場合、etcd はホストで静的 Pod として実行されます。etcd ホストがマスターホストと同じ場所にない場合、etcd はスタンドアロンプロセスとして実行されます。

注記

RHEL Atomic Host を使用する場合、etcd はマスターホストのみに設定できます。

1.3.1. 1 つのシステムの単一マスターおよびノード

OpenShift Container Platform は開発環境の単一システムでのみインストールできます。**オールインワン環境** は実稼働環境として使用できません。

1.3.2. 単一マスターおよび複数ノード

以下の表では、単一 マスター (etcd が同じホストにインストールされている) および 2 つの ノード のサンプル環境について説明しています。

ホスト名	インストールするインフラストラクチャーコンポーネント
master.example.com	マスター、etcd、ノード
node1.example.com	ノード
node2.example.com	

1.3.3. ネイティブの高可用性 (HA) を使用する複数マスター

以下では、ネイティブ HA メソッドを使用する、3 つのマスター、1 つの HAProxy ロードバランサー、2 つの **ノード** のサンプル環境を説明しています。 etcd はマスターノードで静的 Pod として実行されます。

重要

ルーターとマスターノードの負荷が高くなり、耐障害性のある環境を維持するためには、ルーターとマスターノードの負荷が分散されます。Red Hat は、実稼働環境にエンタープライズレベルの外部ロードバランサーの使用を推奨します。この負荷分散は、OpenShift Container Platform ルーターを実行するホストであるマスターとノードに適用されます。負荷が IP アドレスに分散される Transmission Control Protocol (TCP) レイヤー 4 の負荷分散が推奨されます。参照設計については、External Load Balancer Integrations with OpenShift Enterprise 3 を参照してください。これは、実稼働環境での使用には推奨されません。

ホスト名	インストールするインフラストラクチャーコンポーネント
master1.example.com	マスター (クラスター化、ネイティブ HA を使用) およびノードおよびクラスター化された etcd
master2.example.com	
master3.example.com	
lb.example.com	API マスターエンドポイントの負荷分散を行う HAProxy
node1.example.com	ノード
node2.example.com	

1.3.4. 外部のクラスター化された etcd を含む、ネイティブ HA を使用した複数マスター

以下では、ネイティブ HA メソッドを使用する、3 つの マスター、1 つの HAProxy ロードバランサー、3 つの外部のクラスター化された etcd ホスト、 2 つの **ノード** のサンプル環境を説明しています。

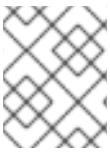
ホスト名	インストールするインフラストラクチャーコンポーネント
master1.example.com	マスター (クラスター化、ネイティブ HA を使用) およびノード
master2.example.com	
master3.example.com	
lb.example.com	API マスターエンドポイントの負荷分散を行う HAProxy
etcd1.example.com	クラスター化された etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	ノード
node2.example.com	

1.3.5. スタンドアロンレジストリー

OpenShift Container Platform は、OpenShift Container Platform の統合レジストリーを使用してスタンドアロンレジストリーとして機能するようにインストールすることもできます。このシナリオの詳細は、スタンドアロンレジストリーのインストール を参照してください。

1.4. サポート対象のオペレーティングシステムのインストールタイプ

OpenShift Container Platform 3.10 以降、RHEL をホストの基礎となる OS として使用する場合、RPM 方式はホストに OpenShift Container Platform コンポーネントをインストールするために使用されます。RHEL Atomic Host を使用する場合、システムコンテナー方式がそのホストで使用されます。いずれのインストールタイプもクラスターに同じ機能を提供しますが、使用するオペレーティングシステムによってサービスおよびホストの更新の管理方法が異なります。



注記

OpenShift Container Platform 3.10 の時点で、includezerized インストール方法は Red Hat Enterprise Linux システムでサポートされなくなりました。

RPM インストールは、パッケージ管理ですべてのサービスをインストールし、サービスを同じユーザー空間で実行されるように設定します。 システムコンテナーのインストールは、システムコンテナーイメージを使用してサービスをインストールし、個別のコンテナーで個々のサービスを実行します。

RHEL で RPM を使用する場合、すべてのサービスが外部ソースのパッケージ管理によってインストールされ、更新されます。これらのパッケージは、同じユーザー空間内のホストの既存設定を変更します。RHEL Atomic Host でのシステムコンテナーインストールの場合は、OpenShift Container Platform の各コンポーネントはコンテナーとして同梱され (自己完結型パッケージ)、ホストのカーネルを使用して実行します。更新された新しいコンテナーはホストの既存のものを置き換えます。

以下の表およびセクションは、インストールタイプごとの詳細な相違点について説明しています。

表1.1 インストールタイプ間の相違点

	Red Hat Enterprise Linux (RHEL)	RHEL Atomic Host
インストールタイプ	RPM ベース	システムコンテナー
配信メカニズム	**yum** を使用する RPM パッケージ	**docker** を使用するシステムコンテナー
サービス管理	**systemd**	**docker** および **systemd** ユニット

1.4.1. システムコンテナーの必須イメージ

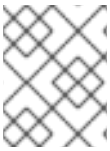
システムコンテナーのインストールタイプは以下のイメージを使用します。

- **openshift3/ose-node**

デフォルトで、上記のイメージはすべて registry.redhat.io の Red Hat Registry からプルされます。

プライベートレジストリーを使用してインストール中にこれらのイメージをプルする必要がある場合は、あらかじめレジストリー情報を指定できます。必要に応じてインベントリーファイルで以下の Ansible 変数を設定できます。

```
oreg_url='<registry_hostname>/openshift3/ose-${component}:${version}'
openshift_docker_insecure_registries=<registry_hostname>
openshift_docker_blocked_registries=<registry_hostname>
```



注記

ホストの IP アドレスに **openshift_docker_insecure_registries** 変数も設定できます。**0.0.0.0/0** は有効な設定ではありません。

デフォルトコンポーネントは、**oreg_url** 値からイメージの接頭辞およびバージョンを継承します。

安全でないブロックされた追加のコンテナーレジストリーの設定はインストールプロセスの開始時に行われ、必要なイメージをプルする前にそれらの設定が適用されるようにします。

1.4.2. systemd サービス名

インストールプロセスでは、通常の **systemctl** コマンドを使用してサービスの起動、停止、ポーリングを実行するために使われる関連の **systemd** ユニットを作成します。システムコンテナーインストールの場合、それらのユニット名は RPM インストールのものと一致します。

1.4.3. ファイルパスの場所

すべての OpenShift Container Platform 設定ファイルは、コンテナー化インストール時に RPM ベースのインストールの場合と同じ場所に置かれ、 **os-tree**アップグレード後も存続します。

ただし、デフォルトのイメージストリームおよびテンプレートファイル は、標準の

/usr/share/openshift/examples/ ディレクトリーが RHEL Atomic Host では読み取り専用であるため、そのディレクトリーにではなく Atomic Host インストールの **/etc/origin/examples/** にインストールされます。

1.4.4. ストレージ要件

RHEL Atomic Host インストールが持つ root ファイルシステムは通常非常に小さいサイズです。ただし、etcd、マスター、ノードコンテナーは **/var/lib/** ディレクトリーにデータを維持します。そのため、OpenShift Container Platform をインストールする前に root ファイルシステムに十分な空き領域があることを確認してください。詳細は システム要件 のセクションを参照してください。

第2章 システムおよび環境要件

2.1. システム要件

OpenShift Container Platform 環境のホストは以下のハードウェア仕様およびシステムレベルの要件を満たしている必要があります。

2.1.1. Red Hat サブスクリプション

まず、お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションがなければなりません。これがない場合は、営業担当者にお問い合わせください。

2.1.2. ハードウェアの最小要件

システムの要件はホストのタイプによって異なります。

マスター	● 物理または仮想システム、またはパブリックまたはプライベート IaaS で実行されるインスタンス。 ● ベース OS: 最低限のインストールオプションと Extras チャンネルの最新パッケージを持つ Red Hat Enterprise Linux (RHEL) 7.5 以降 または RHEL Atomic Host 7.4.5 以降 ○ IBM POWER9: 最低限のインストールオプションと Extras チャンネルの最新パッケージを持つ RHEL-ALT 7.5 ○ IBM POWER8: 最低限のインストールオプションと Extras チャンネルの最新パッケージを持つ RHEL 7.5RHEL を使用する場合は、以下の最小限のカーネルバージョンを使用する必要があります。 ○ RHEL 7.5: 3.10.0-862.31.1 ○ RHEL 7.6: 3.10.0-957.27.2 ○ RHEL 7.7: 3.10.0-1062 ● 4 つ以上の vCPU (追加することを強く推奨します) ● 最小 16 GB RAM (特に etcd がマスター上の同一の場所に配置されている場合、メモリーの追加を強く推奨します) ● **/var/** を含むファイルシステムの最小 40 GB のハードディスク領域。 1 ● **/usr/local/bin/** を含むファイルシステムの最小 1 GB のハードディスク領域。 ● システムの一時ディレクトリーを含むファイルシステムの最小 1 GB のハードディスク領域。 2 ● 同一の場所に配置された etcd を含むマスターは最小 4 コアを必要とします。2 コアのシステムは機能しません。

ノード	• 物理または仮想システム、またはパブリックまたはプライベート IaaS で実行されるインスタンス。
	• ベース OS: 最低限のインストールオプションを持つ RHEL 7.5 以降 または RHEL Atomic Host 7.4.5 以降
	  ◦ IBM POWER9: 最低限のインストールオプションと Extras チャンネルの最新パッケージを持つ RHEL-ALT 7.5
	  ◦ IBM POWER8: 最低限のインストールオプションと Extras チャンネルの最新パッケージを持つ RHEL 7.5RHEL を使用する場合は、以下の最小限のカーネルバージョンを使用する必要があります。
	  ◦ RHEL 7.5: 3.10.0-862.31.1
	  ◦ RHEL 7.6: 3.10.0-957.27.2
	  ◦ RHEL 7.7: 3.10.0-1062
	• NetworkManager 1.0 以降。
	• 1 vCPU。
	• 最小 8 GB の RAM。
	• **/var/** を含むファイルシステムの 15 GB 以上のハードディスク領域。 1
	• **/usr/local/bin/** を含むファイルシステムの最小 1 GB のハードディスク領域。
	• システムの一時ディレクトリーを含むファイルシステムの最小 1 GB のハードディスク領域。 2
	• Docker のストレージバックエンドのコンテナーを実行するシステムごとに使用する 15 GB 以上の追加の未割り当て領域。詳細は、Docker ストレージの設定 を参照してください。ノード上で実行されるコンテナーのサイズおよび数によって、追加の領域が必要になる可能性があります。

外部 etcd ノード	• etcd データ用の最小 20 GB のハードディスク領域。 • etcd ノードを適切にサイズに設定する方法については、CoreOS etcd ドキュメントの Hardware Recommendations セクションを参照してください。 • 現時点で、OpenShift Container Platform はイメージ、ビルド、およびデプロイメントメタデータを etcd に保存します。定期的に 古いリソースをプルーニング する必要があります。これらのリソースを多数使用する予定の場合は、大量のメモリーと高速 SSD ドライブを搭載したマシンに etcd を配置します。
Ansible コントローラー	Ansible Playbook を実行するホストには、ホストあたり 75MiB 以上の空きメモリーがインベントリーで必要になります。

1 RHEL Atomic Host で **/var/**ファイルシステムのサイジング要件を満たすには、デフォルト設定に変更を加える必要があります。インストール時またはインストール後にこの設定を行う方法については、Managing Storage in Red Hat Enterprise Linux Atomic Host を参照してください。

2 システムの一時ディレクトリーは、Python の標準ライブラリーの **tempfile** モジュールで定義されるルールに基づいて決定されます。

コンテナーデーモンを実行する各システムのストレージを設定する必要があります。コンテナー化インストールの場合、マスターにストレージが必要になります。また、Web コンソールはマスターのコンテナーで実行され、マスターには Web コンソールを実行するためにストレージが必要です。コンテナーはノードで実行されるため、ノードにはストレージが常に必要になります。ストレージのサイズはワークロード、コンテナー数、実行中のコンテナーのサイズおよびコンテナーのストレージ要件によって異なります。また、ストレージをコンテナー化された etcd を実行するように設定する必要もあります。

NVMe や SSD などのシリアル書き込み (fsync) を迅速に処理するストレージで etcd を使用することが強く推奨されます。Ceph、NFS、およびスピニングディスクは推奨されません。

2.1.3. 実稼働環境レベルのハードウェア要件

テストまたはサンプル環境は最小要件で機能します。実稼働環境の場合、以下の推奨事項が当てはまります。

マスターホスト

外部 etcd を含む可用性の高い OpenShift Container Platform クラスターにおいて、マスターホストには、上記の表にある最小要件のほかに、1000 Pod に対して 1 CPU コアと 1.5 GB のメモリーが必要になります。したがって、2000 Pod で設定される OpenShift Container Platform クラスターのマスターホストの推奨されるサイズとして、2 CPU コアと 16 GB の RAM に 2 CPU コアと 3 GB の RAM を追加した合計 4 CPU コアと 19 GB の RAM が最小要件として必要になります。

パフォーマンスに関するガイダンスについては、 Recommended Practices for OpenShift Container Platform Master Hosts を参照してください。

ノードホスト

ノードホストのサイズは、そのワークロードの予想されるサイズによって異なります。OpenShift Container Platform クラスターの管理者は、予想されるワークロードを計算し、オーバーヘッドの約 10 パーセントを追加する必要があります。実稼働環境の場合、ノードホストの障害が最大容量に影響を与えることがないよう、十分なリソースを割り当てるようにします。

詳細は、 サイジングに関する考慮事項および Cluster Limits を参照してください。

重要

ノードでの物理リソースの過剰なサブスクライブは、Kubernetes スケジューラーが Pod の配置時に行うリソース保証に影響を与えます。メモリースワップを防ぐ ために実行できる処置について確認してください。

2.1.4. ストレージ管理

表2.1 OpenShift Container Platform コンポーネントがデータを書き込む主なディレクトリー

ディレクトリー	注記	サイジング	予想される拡張

ディレクトリー	注記	サイジング	予想される拡張
/var/lib/openshift	単一マスターモードの場合に etcd ストレージのみに使用され、etcd は **atomic-openshift-master** プロセスで組み込まれます。	10GB 未満。	環境と共に徐々に拡張します。メタデータのみを格納します。
/var/lib/etcd	複数マスターモードの場合や etcd が管理者によってスタンドアロンにされる場合に etcd ストレージに使用されます。	20 GB 未満。	環境と共に徐々に拡張します。メタデータのみを格納します。
/var/lib/docker	ランタイムが docker の場合、これはマウントポイントになります。アクティブなコンテナーランタイム (Pod を含む) およびローカルイメージのストレージに使用されるストレージです (レジストリーストレージには使用されません)。マウントポイントは手動ではなく、docker-storage で管理される必要があります。	16 GB メモリーの場合、1 ノードにつき 50 GB。 メモリーに 8 GB が追加されるたびに 20-25 GB を追加します。	拡張は実行中のコンテナーの容量によって制限されます。
/var/lib/containers	ランタイムが CRI-O の場合、これはマウントポイントになります。アクティブなコンテナーランタイム (Pod を含む) およびローカルイメージのストレージに使用されるストレージです (レジストリーストレージには使用されません)。	16 GB メモリーの場合、1 ノードにつき 50 GB。 メモリーに 8 GB が追加されるたびに 20-25 GB を追加します。	拡張は実行中のコンテナーの容量によって制限されます。
/var/lib/origin/openshift.local.volumes	Pod の一時ボリュームストレージです。これには、ランタイムにコンテナーにマウントされる外部のすべての内容が含まれます。環境変数、kube シークレット、および永続ストレージ PV でサポートされていないデータボリュームが含まれます。	変動あり。	ストレージを必要とする Pod が永続ボリュームを使用している場合は最小になります。一時ストレージを使用する場合はすぐに拡張する可能性があります。

ディレクトリー	注記	サイジング	予想される拡張
/var/log	すべてのコンポーネントのログファイルです。	10 から 30 GB。	ログファイルはすぐに拡張する可能性があります。 サイズは拡張するディスク別に管理するか、ログローテーションを使用して管理できます。

2.1.5. Red Hat Gluster Storage ハードウェア要件

コンバージドモードまたはインデペンデントモードのクラスターで使用されるノードはストレージノードとみなされます。単一ノードは複数のグループに分割できませんが、ストレージノードはそれぞれ別個のクラスターグループに分類できます。ストレージノードの各グループについては、以下が当てはまります。

- Gluster ストレージのボリュームタイプオプションに基づき、1 つのグループあたり最低でも 1 つまたは複数のストレージが必要です。
- 各ストレージノードには 8 GB 以上の RAM が必要です。これにより、Red Hat Gluster Storage Pod、その他のアプリケーションおよび基礎となる OS を実行できます。
  - 各 GlusterFS ボリュームはストレージクラスターにあるすべてのストレージノードのメモリー (約 30 MB) も消費します。RAM の合計量は、コンカレントボリュームがいくつ求められているか、またはいくつ予想されるかによって決める必要があります。
- 各ストレージノードには、現在のデータまたはメタデータを含まない 1 つ以上の raw ブロックデバイスが必要です。それらのブロックデバイス全体は GlusterFS ストレージで使用されます。以下が存在しないことを確認してください。
  - パーティションテーブル (GPT または MSDOS)
  - ファイルシステムまたは未処理のファイルシステムの署名
  - 以前のボリュームグループの LVM2 署名および論理ボリューム
  - LVM2 物理ボリュームの LVM2 メタデータ

  不確かな場合には、**wipefs -a <device>** で上記のすべてを消去する必要があります。

重要

2 つのクラスター、つまりインフラストラクチャーアプリケーション (OpenShift Container レジストリーなど) のストレージ専用のクラスターと一般的なアプリケーションのストレージ専用のクラスターについて計画することをお勧めします。これには、合計で 6 つのストレージノードが必要になります。この設定は I/O およびボリューム作成のパフォーマンスへの潜在的な影響を回避するために推奨されます。

2.1.6. ハードウェア要件のモニターリング

モニターリングスタックは追加のリソース要件を課すもので、デフォルトでインストールされます。コンピューティングリソースの推奨事項 および クラスターモニターリングのドキュメント を参照してください。

2.1.7. SELinux 要件

Security-Enhanced Linux (SELinux) をすべてのサーバーで有効にしてから OpenShift Container Platform をインストールする必要があります。そうでないと、インストーラーは失敗します。 さらに、**/etc/selinux/config** ファイルで **SELINUX=enforcing** および **SELINUXTYPE=targeted** を設定します。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2.1.8. オプション: コアの使用についての設定

デフォルトで、OpenShift Container Platform マスターおよびノードは、それらが実行されるシステムで利用可能なすべてのコアを使用します。**GOMAXPROCS** 環境変数を設定することにより、OpenShift Container Platform で使用するコア数を選択することができます。**GOMAXPROCS** 環境変数の機能などの詳細については、Go Language ドキュメント を参照してください。

たとえば、以下を実行してからサーバーを起動し、OpenShift Container Platform が 1 つのコアでのみ実行されるようにします。

```
# export GOMAXPROCS=1
```

2.1.9. オプション: OverlayFS の使用

OverlayFS は、ファイルシステム上に別のファイルシステムを重ねる (オーバーレイする) ことができるユニオンファイルシステムです。

Red Hat Enterprise Linux 7.4 の時点で、OpenShift Container Platform 環境を OverlayFS を使用できるように設定するオプションがあります。古いバージョンの **overlay** ドライバーのほかにも、**overlay2** グラフドライバーが完全にサポートされています。ただし、Red Hat では、速度と実装の単純さを考慮し、**overlay** ではなく **overlay2** を使用することを推奨しています。

Comparing the Overlay vs. Overlay2 Graph Drivers には、**overlay** および **overlay2** ドライバーの詳細情報が記載されています。

Docker サービスの overlay2 グラフドライバーを有効化する方法については、Atomic Host ドキュメントの **Overlay Graph Driver** セクションを参照してください。

2.1.10. セキュリティー警告

OpenShift Container Platform は、クラスター内のホストでコンテナーを実行し、ビルド操作やレジストリーサービスなど一部のケースでは特権付きコンテナーを使用して実行します。さらに、これらのコンテナーはホストの Docker daemon にアクセスし、**docker build** および **docker push** の操作を実行し

ます。実質的に root アクセスが可能であるため、任意のイメージでの **docker run** 操作の実行については関連するセキュリティーリスクについてクラスター管理者が認識している必要があります。**docker build** の操作についてはとくに注意が必要です。

特定のビルドをノードに割り当て、それらのノードのみにリスクを制限することで有害なコンテナーに関連する危険にさらされるリスクを制限できます。これを実行するには、開発ガイドの 特定のノードへのビルドの割り当て のセクションを参照してください。クラスター管理者の場合は、グローバルビルドのデフォルト設定およびオーバーライドの設定 のセクションを参照してください。

SCC (Security Context Constraints) を使用して、Pod が実行可能なアクションおよび、アクセス可能な機能を制御できます。Dockerfile の **USER** で実行するイメージを有効にする方法は、Managing Security Context Constraints(ユーザーには **cluster-admin** 権限が必要) を参照してください。

詳細は、以下の記事を参照してください。

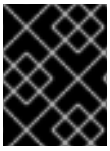
- http://opensource.com/business/14/7/docker-security-selinux
- https://docs.docker.com/engine/security/security/

2.2. 環境要件

以下のセクションでは、OpenShift Container Platform 設定を含む環境の要件を定義します。これには、ネットワークの考慮事項や Git リポジトリーのアクセス、ストレージおよびクラウドインフラストラクチャープロバイダーなどの外部サービスへのアクセスなどの要件が含まれます。

2.2.1. DNS 要件

OpenShift Container Platform では、完全に機能する DNS サーバーが環境になければなりません。この場合、DNS ソフトウェアを実行する別個のホストを使用することが適しており、これによりプラットフォームで実行されるホストおよびコンテナーに対して名前解決を実行することができます。



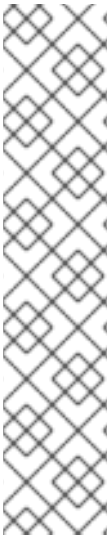
重要

各ホストの **/etc/hosts** ファイルにエントリーを追加するだけでは不十分です。このファイルはプラットフォームで実行されるコンテナーにはコピーされません。

OpenShift Container Platform の主要コンポーネントはコンテナーの内部で実行され、名前解決に以下のプロセスを使用します。

1. デフォルトで、コンテナーはホストから DNS 設定ファイル (**/etc/resolv.conf**) を受信します。
2. OpenShift Container Platform は Pod の最初のネームサーバーをノードの IP アドレスに設定します。

OpenShift Container Platform 3.2 の時点で、**dnsmasq** はすべてのマスターおよびノードで自動的に設定されます。Pod は DNS としてノードを使用し、ノードは要求を転送します。デフォルトで、**dnsmasq** はポート 53 でリッスンするようにノード上に設定されます。 そのため、ノードはその他の種類の DNS アプリケーションを実行することができません。



注記

NetworkManager はネットワークに自動的に接続するシステムの検出と設定を行うプログラムであり、**dnsmasq** を DNS IP アドレスで設定するためにノードで必要となります。

NM_CONTROLLED はデフォルトで **yes** に設定されます。**NM_CONTROLLED** が **no** に設定されている場合、NetworkManager のディスパッチスクリプトは関連する **origin-upstream-dns.conf** dnsmasq ファイルを作成せず、dnsmasq を手動で設定する必要があります。

同様に、ネットワークスクリプト (例: **/etc/sysconfig/network-scripts/ifcfg-em1**) で **PEERDNS** パラメーターが **no** に設定されている場合、dnsmasq ファイルは生成されず、Ansible のインストールは失敗します。**PEERDNS** 設定が **yes** に設定されていることを確認してください。

以下はレコードのサンプルセットです。

```
master1    A   10.64.33.100
master2    A   10.64.33.103
node1      A   10.64.33.101
node2      A   10.64.33.102
```

適切に機能する DNS 環境がない場合には、以下に関連する障害が発生する可能性があります。

- Ansible ベースの参照スクリプトによる製品のインストール
- インフラストラクチャーコンテナー (レジストリー、ルーター) のデプロイ
- OpenShift Container Platform web コンソールへのアクセス (IP アドレスのみではアクセスできないため)

2.2.1.1. ホストを DNS を使用するように設定する

環境内の各ホストが DNS サーバーのホスト名を解決するように設定されていることを確認します。ホストの DNS 解決の設定は、DHCP が有効にされているかどうかによって異なります。DHCP の場合:

- DHCP が無効にされている場合、ネットワークインターフェイスを static (静的) に設定し、DNS ネームサーバーを NetworkManager に追加します。
- DHCP が有効にされている場合、NetworkManager ディスパッチスクリプトは DHCP 設定に基づいて DNS を自動的に設定します。

ホストが DNS サーバーで解決できることを確認するには、以下を実行します。

1. **/etc/resolv.conf** の内容を確認します。

```
$ cat /etc/resolv.conf
# Generated by NetworkManager
search example.com
nameserver 10.64.33.1
# nameserver updated by /etc/NetworkManager/dispatcher.d/99-origin-dns.sh
```

この例では、10.64.33.1 が DNS サーバーのアドレスです。

2. **/etc/resolv.conf** に一覧表示されている DNS サーバーが OpenShift Container Platform 環境のすべてのマスターおよびノードの IP アドレスに対してホスト名を解決できることをテストします。

```
$ dig <node_hostname> @<IP_address> +short
```

以下は例になります。

```
$ dig master.example.com @10.64.33.1 +short
10.64.33.100
$ dig node1.example.com @10.64.33.1 +short
10.64.33.101
```

2.2.1.2. DNS ワイルドカードの設定

オプションとして、ルーターが使用するワイルドカードを設定し、新規ルートが追加される際に DNS 設定を更新しなくてもよいようにします。ルーターのワイルドカードを設定する場合は、Ansible インベントリーファイル の設定時に **openshift_master_default_subdomain** パラメーターをこの値に設定します。

DNS ゾーンのワイルドカードは、最終的には OpenShift Container Platform ルーター の IP アドレスに解決される必要があります。

たとえば、有効期間 (TTL) の低い値が設定されていて、ルーターがデプロイされるホストのパブリック IP アドレスをポイントする **cloudapps** のワイルドカード DNS エントリーを作成します。

```
*.cloudapps.example.com. 300 IN  A 192.168.133.2
```

警告

各ノードホストの **/etc/resolv.conf** ファイルで、ワイルドカードエントリーを持つ DNS サーバーがネームサーバーとして一覧表示されていないこと、またはワイルドカードドメインが検索一覧に表示されていないことを確認してください。そうでない場合、OpenShift Container Platform が管理するコンテナーはホスト名を適切に解決できないことがあります。

2.2.1.3. ノードホスト名の設定

クラウドプロバイダーに統合されていないクラスターを設定する場合、ノードのホスト名を正しく設定する必要があります。各ノードのホスト名は解決可能である必要があり、各ノードは相互に到達できる必要があります。

ノードが他のノードに到達できることを確認するには、以下を実行します。

1. 1 つのノードでホスト名を取得します。

```
$ hostname

master-1.example.com
```

2. 同じノードで、ホストの完全修飾ドメイン名を取得します。

```
$ hostname -f

master-1.example.com
```

3. 別のノードから、この最初のノードに到達できることを確認します。

```
$ ping master-1.example.com -c 1

PING master-1.example.com (172.16.122.9) 56(84) bytes of data.
64 bytes from master-1.example.com (172.16.122.9): icmp_seq=1 ttl=64 time=0.319 ms

--- master-1.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.319/0.319/0.319/0.000 ms
```

2.2.2. ネットワークアクセス要件

共有ネットワークは、マスターとノードホスト間に存在する必要があります。標準のインストール方式を使用して 高可用性のために複数のマスター を設定する計画をしている場合、インストールのプロセスで 仮想 IP (VIP) として設定される IP を選択する必要もあります。選択した IP はすべてのノード間でルーティングできる必要があり、FQDN を使用して設定する場合は、すべてのノード上で解決する必要があります。

2.2.2.1. NetworkManager

NetworkManager はネットワークに自動的に接続するシステムの検出と設定を行うプログラムであり、**dnsmasq** を DNS IP アドレスで設定するためにノードで必要となります。

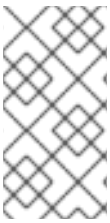
NM_CONTROLLED はデフォルトで **yes** に設定されます。**NM_CONTROLLED** が **no** に設定されている場合、NetworkManager のディスパッチスクリプトは関連する **origin-upstream-dns.conf** dnsmasq ファイルを作成せず、dnsmasq を手動で設定する必要があります。

2.2.2.2. firewalld のファイアウォールとしての設定

iptables はデフォルトのファイアウォールですが、firewalld は新規インストールで推奨されるファイアウォールです。**Ansible インベントリーファイル** で os_firewall_use_firewalld=true を設定することで、firewalld を有効にすることができます。

```
[OSEv3:vars]
os_firewall_use_firewalld=True
```

この変数を **true** に設定することで、必要なポートが開き、ルールがデフォルトゾーンに追加されます。 これにより、firewalld が適切に設定されていることを確認できます。



注記

firewalld のデフォルトの設定オプションを使用する際には設定オプションが制限され、これらをオーバーライドすることはできません。たとえば、ストレージネットワークを複数ゾーンのインターフェイスでセットアップすることができますが、ノードが通信に使用するインターフェイスはデフォルトゾーンになければなりません。

2.2.2.3. 複数のネットワークインターフェイスを持つホスト

ホストに複数のネットワークインターフェイスがある場合、OpenShift Container Platform はインストール、クラスターネットワーク、およびサービスネットワーク用に 1 つのネットワークインターフェイスのみを使用します。OpenShift Container Platform とは関係のない通信に追加のネットワークインターフェイスを使用できますが、あるネットワークインターフェイスや、別のネットワークインターフェイスでクラスター関連のトラフィックが異なるクラスター関連のトラフィックをルーティングするサポートはありません。

2.2.2.4. 必須ポート

OpenShift Container Platform のインストールは、iptables を使用して各ホストに内部のファイアウォールルール一式を自動的に作成します。ただし、ネットワーク設定でハードウェアベースのファイアウォールなどの外部ファイアウォールを使用する場合、インフラストラクチャーコンポーネントが、特定のプロセスまたはサービスの通信エンドポイントとして機能する特定ポートで相互に通信できることを確認する必要があります。

OpenShift Container Platform で必要な以下のポートがネットワーク上で開いており、ホスト間のアクセスを許可するよう設定されていることを確認してください。設定や使用状況によって、一部はポートはオプションになります。

表2.2 ノード間通信

4789	UDP	別個のホストの Pod 間の SDN 通信に必要です。

表2.3 ノードからマスターへの通信

4789	UDP	別個のホストの Pod 間の SDN 通信に必要です。
443 または 8443	TCP	ノードホストがマスター API と通信するために必要です。 ノードホストがステータスをポストバックしたり、タスクを受信したりする際に使用します。

表2.4 マスターからノードへの通信

4789	UDP	別個のホストの Pod 間の SDN 通信に必要です。
10250	TCP	マスターは **oc** コマンドの Kubelet 経由でノードホストのプロキシーします。このポートはマスターおよびインフラストラクチャーノードから任意のマスターおよびノードに対して許可される必要があります。メトリクスについては、ソースがインフラストラクチャーノードにある必要があります。
10010	TCP	CRI-O を使用している場合は、このポートを開き、**oc exec** および **oc rsh** 操作を実行できるようにします。

表2.5 マスター間の通信

2049	TCP/ UDP	NFS ホストをインストーラーの一部としてプロビジョニングする場合に必要です。

2379	TCP	スタンドアロン etcd (クラスター化) が状態の変更を受け取るために使用されます。
2380	TCP	etcd はスタンドアロン etcd (クラスター化) を使用する場合、リーダー選定とピアリング接続のためにこのポートがマスター間で開かれていることを要求します。
4789	UDP	別個のホストの Pod 間の SDN 通信に必要です。

表2.6 外部からロードバランサーへの通信

9000	TCP	**ネイティブ** HA メソッドを選択している場合、HAProxy 統計ページへのアクセスを許可するためのオプションです。

表2.7 外部からマスターへの通信

443 または 8443	TCP	ノードホストがマスター API と通信するために必要です。 ノードホストがステータスをポストバックしたり、タスクを受信したりする際に使用します。
8444	TCP	コントローラーマネージャーおよびスケジューラーサービスがリッスンするポート。**/metrics** および **/healthz** エンドポイント用に開いている必要があります。

表2.8 IaaS デプロイメント

22	TCP	インストーラーまたはシステム管理者が SSH で必要とします。
53 または 8053	TCP/UDP	クラスターサービス (SkyDNS) の DNS 解決に必要です。3.2 よりも前のインストールまたは 3.2 にアップグレードした環境はポート 53 を使用します。新規インストールはデフォルトで 8053 を使用するため、**dnsmasq** が設定される可能性があります。マスターホストの内部で開かれている必要があります。
80 または 443	TCP	ルーターの HTTP/HTTPS 用です。ノードホスト、とくにルーターを実行しているノードで外部に開かれている必要があります。
1936	TCP	(**オプション**) テンプレートルーターを実行して統計にアクセスする際に開かれている必要があります。統計をどのように公開する必要があるかによって、接続に対して外部または内部に開くことができます。この場合、追加の設定が必要になることがあります。詳しくは、以下の注記セクションを参照してください。
2379 および 2380	TCP	スタンドアロン etcd 用です。マスターホストの内部で開かれている必要があります。2379 はサーバークライアント接続用です。2380 はサーバー間の接続用で、クラスター化された etcd がある場合にのみ必要となります。
4789	UDP	VxLAN 用 (OpenShift SDN) です。ノードホストの内部で開かれている必要があります。

8443	TCP	OpenShift Container Platform Web コンソール用で、API サーバーと共有します。
10250	TCP	Kubelet 用です。ノード上で外部に開かれている必要があります。

備考

- 上記の例では、ポート **4789** は UDP (User Datagram Protocol) に使用されます。
- デプロイメントで SDN を使用している場合、レジストリーがデプロイされているのと同じノードからレジストリーにアクセスしているのでない限り、 Pod のネットワークはサービスプロキシー経由でアクセスされます。
- OpenShift Container Platform の内部 DNS は SDN 経由で受け取ることができません。クラウド以外のデプロイメントの場合、これはデフォルトで、マスターホストのデフォルトルートに関連付けられた IP アドレスに設定されます。クラウドデプロイメントの場合、これはデフォルトでクラウドメタデータで定義される最初の内部インターフェイスに関連付けられた IP アドレスに設定されます。
- マスターホストはポート **10250** を使用してノードに到達し、SDN を経由しません。デプロイメントのターゲットホストによって異なりますが、**openshift_public_hostname** の計算された値を使用します。
- iptables ルールにより、ポート **1936** はアクセス不可能な状態になります。ポート **1936** を開くよう iptables を設定するには以下を使用してください。

```
# iptables -A OS_FIREWALL_ALLOW -p tcp -m state --state NEW -m tcp \
    --dport 1936 -j ACCEPT
```

表2.9 集計ロギングおよびメトリクス

9200	TCP	Elasticsearch API 用です。Kibana が表示用にログを取得できるようにインフラストラクチャーノードの内部で開かれている必要があります。ルートを使用して Elasticsearch に直接アクセスできるよう外部に開くこともできます。ルートは **oc expose** を使用して作成できます。
9300	TCP	Elasticsearch のクラスター内での使用向けです。Elasticsearch クラスターのメンバーが相互に通信できるようにインフラストラクチャーノードで内部に開かれている必要があります。
9090	TCP	Prometheus API および Web コンソール用です。
9100	TCP	ハードウェアおよびオペレーティングシステムのメトリクスをエクスポートする Prometheus Node-Exporter 用です。ポート 9100 は、Prometheus サーバーがメトリクスを収集するために各 OpenShift Container Platform ホストで開かれている必要があります。
8443	TCP	ノードホストがマスター API と通信するために必要です。ノードホストがステータスをポストバックしたり、タスクを受信したりする際に使用します。 このポートはマスターおよびインフラストラクチャーノードから任意のマスターノードに対して許可される必要があります。

10250	TCP	Kubernetes cAdvisor、コンテナーリソースの使用状況およびパフォーマンス分析エージェント用です。このポートはマスターおよびインフラストラクチャーノードから任意のマスターノードに対して許可される必要があります。メトリクスについては、ソースがインフラストラクチャーノードにある必要があります。
8444	TCP	コントローラーマネージャーおよびスケジューラーサービスがリッスンするポート。ポート 8444 は各 OpenShift Container Platform ホストで開かれている必要があります。
1936	TCP	(**オプション**) テンプレートルーターを実行して統計にアクセスする際に開かれている必要があります。このポートは、Prometheus インフラストラクチャーメトリクスがルーターで有効にされている場合にインフラストラクチャーノードからルーターをホストするインフラストラクチャーノードに対して許可される必要があります。統計を公開する必要があるかどうかに応じて、接続に対して外部または内部に開くことができます。この場合、追加の設定が必要になることがあります。詳しくは、以下の注記セクションを参照してください。

備考

2.2.3. 永続ストレージ

Kubernetes の 永続ボリューム フレームワークにより、お使いの環境で利用可能なネットワークストレージを使用して、OpenShift Container Platform クラスターに永続ストレージをプロビジョニングできます。これは、アプリケーションのニーズに応じて初回 OpenShift Container Platform インストールの完了後に行うことができ、ユーザーは基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようになります。

クラスターの設定ガイドは、NFS、GlusterFS, Ceph RBD、OpenStack Cinder、AWS Elastic Block Store (EBS)、GCE Persistent Disks、および iSCSI を使用して永続ストレージを OpenShift Container Platform クラスターにプロビジョニングする方法についてのクラスター管理者向けの情報を提供しています。

2.2.4. クラウドプロバイダーの留意事項

OpenShift Container Platform をクラウドプロバイダーにインストールする場合に考慮すべき事柄がいくつかあります。

- Amazon Web Services の場合は、Permissions および Configuring a Security Group のセクションを参照してください。
- OpenStack の場合は、Permissions and the Configuring a Security Group セクションを参照してください。

2.2.4.1. 検出された IP アドレスとホスト名の上書き

一部のデプロイメントでは、ユーザーがホストの検出されたホスト名と IP アドレスを上書きすることが必要です。デフォルト値を確認するには、Playbook ディレクトリーに切り替え、**openshift_facts** Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \
  playbooks/byo/openshift_facts.yml
```

重要

Amazon Web Services の場合は、Overriding Detected IP Addresses and Host Names のセクションを参照してください。

検出された共通の設定を確認してみましょう。それらが想定される内容と異なる場合にはそれらを上書きすることができます。

インベントリーファイルの設定 トピックでは、利用可能な Ansible 変数を詳しく説明します。

変数	使用法
hostname	• インスタンス自体から内部 IP アドレスに解決する。
ip	• インスタンスの内部 IP アドレス。
public_hostname	• クラウド外のホストから外部 IP に解決する。 • プロバイダーの **openshift_public_hostname** が上書きする。
public_ip	• インスタンスに関連付けられた外部からアクセス可能な IP アドレス。 • **openshift_public_ip** が上書きする。
use_openshift_sdn	• GCE 以外のすべてのクラウドについては、**true** に設定する。 • **openshift_use_openshift_sdn** が上書きする。

2.2.4.2. クラウドプロバイダーのインストール後の設定

インストールプロセスの後に、AWS、OpenStack、または GCE 用に OpenShift Container Platform を設定することができます。

第3章 ホストの準備

OpenShift Container Platform をインストールする前に、ノードホストを準備する必要があります。それらは以下の要件を満たす必要があります。

3.1. オペレーティングシステム要件

マスターおよびノードホストのオペレーティングシステムの要件は使用するサーバーのアーキテクチャーによって異なります。

- x86_64 アーキテクチャーを使用するサーバーの場合は、Extras チャンネルからの最新パッケージを含む Red Hat Enterprise Linux (RHEL) 7.5 以降のベースインストールまたは RHEL Atomic Host 7.4.2 以降を使用します。
- クラスドベースのインストールの場合は、Extras チャンネルからの最新パッケージを含む RHEL 7.5 以降のベースインストールを使用します。
- IBM POWER8 アーキテクチャーを使用するサーバーの場合は、Extras チャンネルからの最新パッケージを含む RHEL 7.5 以降のベースインストールを使用します。
- IBM POWER9 アーキテクチャーを使用するサーバーの場合は、Extras チャンネルからの最新パッケージを含む RHEL-ALT 7.5 以降のベースインストールを使用します。

それぞれのインストール方法については、必要に応じて以下のドキュメントを参照してください。

- Red Hat Enterprise Linux 7 インストールガイド
- Red Hat Enterprise Linux Atomic Host 7 インストールと設定ガイド

3.2. サーバータイプの要件

ノードに IBM POWER サーバーを使用する場合は、IBM POWER サーバーのみを使用できます。IBM POWER サーバーで実行されるノードを x86_64 サーバーを使用する既存クラスターに追加したり、クラスターノードを IBM POWER および x86_64 サーバーの混在環境にデプロイできません。

3.3. パスの設定

各ホストの root ユーザーの **PATH** には以下のディレクトリーが含まれている必要があります。

- **/bin**
- **/sbin**
- **/usr/bin**
- **/usr/sbin**

これらのディレクトリーは新規の RHEL 7.x インストールでデフォルトで設定されます。

3.4. ホストアクセスの確保

OpenShift Container インストーラーでは、すべてのホストにアクセスできるユーザーが必要になります。インストーラーを非 root ユーザーとして実行する場合は、まず各ホストでパスワードレス **sudo** 権限を設定します。

1. インストール Playbook を実行するホストで SSH キーを生成します。

```
# ssh-keygen
```

パスワードは使用**しない**でください。

2. キーを他のクラスターホストに配信します。**bash** ループを使用できます。

```
# for host in master.example.com \ 1
    node1.example.com \ 2
    node2.example.com; \ 3
    do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
    done
```

1 2 3 各クラスターホストのホスト名を指定します。

3. SSH 経由でループに一覧表示される各ホストにアクセスできることを確認します。

3.5. プロキシーの上書きの設定

ノードの **/etc/environment** ファイルに **http_proxy** または **https_proxy** 値のいずれかが含まれる場合、OpenShift Container Platform コンポーネント間でのオープンな通信を可能にするため、そのファイルに **no_proxy** 値を設定する必要もあります。

注記

/etc/environment ファイルの **no_proxy** パラメーターは、インベントリーファイルに設定するグローバルプロキシー値と同じ値ではありません。グローバルプロキシー値では、プロキシーの設定を使って特定の OpenShift Container Platform サービスを設定します。詳細は、グローバルプロキシーオプションの設定 を参照してください。

/etc/environment ファイルにプロキシー値が含まれる場合、以下の値を、各ノードでこのファイルの **no_proxy** パラメーターに以下の値を定義します。

- マスターおよびノードのホスト名またはそれらのドメイン接尾辞。
- 他の内部ホスト名またはそれらのドメイン接尾辞。
- etcd IP アドレス。**etcd** アクセスはアドレスで制御されるので、ホスト名ではなく IP アドレスを指定する必要があります。
- Kubernetes IP アドレス (デフォルトは **172.30.0.1**)。インベントリーファイルの **openshift_portal_net** パラメーターに設定される値である必要があります。
- Kubernetes の内部ドメイン接尾辞: **cluster.local**。
- Kubernetes の内部ドメイン接尾辞: **.svc**

注記

no_proxy は CIDR をサポートしないので、ドメイン接尾辞を使用できます。

http_proxy または **https_proxy** 値のいずれかを使用する場合、 **no_proxy** パラメーターの値は以下の例のようになります。

```
no_proxy=.internal.example.com,10.0.0.1,10.0.0.2,10.0.0.3,.cluster.local,.svc,localhost,127.0.0.1,172.30
0.1
```

3.6. ホストの登録

インストールパッケージにアクセスするには、各ホストを Red Hat Subscription Manager (RHSM) に登録し、アクティブな OpenShift Container Platform サブスクリプションを割り当てる必要があります。

1. 各ホストで RHSM に登録します。

```
# subscription-manager register --username=<user_name> --password=<password>
```

2. RHSM から最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

3. 利用可能なサブスクリプションを一覧表示します。

```
# subscription-manager list --available --matches '*OpenShift*'
```

4. 直前のコマンドの出力で、OpenShift Container Platform サブスクリプションのプール ID を見つけ、これをアタッチします。

```
# subscription-manager attach --pool=<pool_id>
```

5. yum リポジトリーをすべて無効にします。
   a. 有効にされている RHSM リポジトリーをすべて無効にします。

```
# subscription-manager repos --disable="*"
```

   b. 残りの yum リポジトリーを一覧表示し、**repo id** にあるそれらの名前をメモします (ある場合)。

```
# yum repolist
```

   c. **yum-config-manager** を使用して、残りの yum リポジトリーを無効にします。

```
# yum-config-manager --disable <repo_id>
```

   または、すべてのリポジトリーを無効にします。

```
 yum-config-manager --disable \*
```

   利用可能なリポジトリーが多い場合には、数分の時間がかかることがあります。

6. OpenShift Container Platform 3.11 で必要なリポジトリーのみを有効にします。

- x86_64 サーバーでのクラウドインストールおよびオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.11-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms"
```

- IBM POWER8 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-for-power-le-rpms" \
    --enable="rhel-7-for-power-le-extras-rpms" \
    --enable="rhel-7-for-power-le-optional-rpms" \
    --enable="rhel-7-server-ansible-2.9-for-power-le-rpms" \
    --enable="rhel-7-server-for-power-le-rhscl-rpms" \
    --enable="rhel-7-for-power-le-ose-3.11-rpms"
```

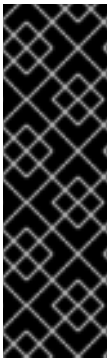
- IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-for-power-9-rpms" \
    --enable="rhel-7-for-power-9-extras-rpms" \
    --enable="rhel-7-for-power-9-optional-rpms" \
    --enable="rhel-7-server-ansible-2.9-for-power-9-rpms" \
    --enable="rhel-7-server-for-power-9-rhscl-rpms" \
    --enable="rhel-7-for-power-9-ose-3.11-rpms"
```

注記

以前のバージョンの OpenShift Container Platform 3.11 は Ansible 2.6 のみをサポートしていました。最新バージョンの Playbook が Ansible 2.9 に対応するようになりました。Ansible 2.9 は、使用する推奨バージョンです。

3.7. 基本パッケージのインストール



重要

ホストが RHEL 7.5 を使用しており、OpenShift Container Platform のデフォルト **docker** 設定 (OverlayFS ストレージおよびすべてのデフォルトロギングオプションを使用) を受け入れる必要がある場合、これらのパッケージを手動でインストールしないでください。これらのパッケージは、インストール 時に **prerequisites.yml** Playbook を実行する場合にインストールされます。

ホストが RHEL 7.4 を使用するか、または RHEL 7.5 を使用し、**docker** 設定をカスタマイズする必要がある場合、以下のパッケージをインストールします。

RHEL 7 システムの場合:

1. 以下の基本パッケージをインストールします。

```
# yum install wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-completion kexec-tools sos psacct
```

2. システムを最新パッケージに更新します。

```
# yum update
# reboot
```

3. 使用するインストール方式に必要なパッケージをインストールします。

- コンテナー化されたインストーラー を使用する予定がある場合、以下のパッケージをインストールします。

```
# yum install atomic
```

- RPM ベースのインストーラー を使用する予定がある場合、以下のパッケージをインストールします。

```
# yum install openshift-ansible
```

注記

yum install openshift-ansible コマンドの実行中にエラーが発生した場合は、解決策を参照してそのエラーを修正してください。

このパッケージはインストーラーユーティリティーを提供し、Ansible、Playbook、および関連する設定ファイルなどの、クラスターインストールプロセスが必要とする他のパッケージをプルします。

RHEL Atomic Host 7 システムの場合:

1. 最新の Atomic ツリーにアップグレードしてホストが最新の状態にあることを確認します (利用可能な場合)。

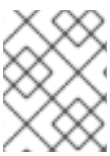
```
# atomic host upgrade
```

2. アップグレードが完了し、以下の起動の準備ができたら、ホストを再起動します。

```
# reboot
```

3.8. DOCKER のインストール

ここで、すべてのマスターおよびノードホストで Docker をインストールする必要があります。これにより、OpenShift Container Platform をインストールする前に Docker ストレージオプション を設定することができます。



注記

クラスターインストールプロセスは、**/etc/sysconfig/docker** ファイルを自動的に変更します。

RHEL 7 システムの場合:

1. Docker 1.13 をインストールします。

```
# yum install docker-1.13.1
```

2. バージョン 1.13 がインストールされていることを確認します。

```
# rpm -V docker-1.13.1
# docker version
```

RHEL Atomic Host 7 システムの場合:

アクションは不要です。Docker はデフォルトでインストールされ、設定され、実行中になります。

3.9. DOCKER ストレージの設定

作成元のコンテナーとイメージは Docker のストレージバックエンドに保存されます。このストレージは一時的なストレージであり、アプリケーションの必要を満たすために割り当てられる 永続ストレージ とは区別されます。**一時ストレージ** の場合、コンテナーに保存されるデータはコンテナーが削除されると失われます。**永続ストレージ** の場合、コンテナーに保存されるデータはコンテナーが削除されてもそのまま残ります。

デフォルトで各システムはコンテナーデーモンを実行するので、すべてのマスターおよびノードホストにストレージを設定する必要があります。コンテナー化インストールの場合、マスターにストレージが必要です。またデフォルトで、ストレージが必要な Web コンソールおよび etcd はマスター上のコンテナーで実行されます。コンテナーはノードで実行されるため、ストレージは常にそれらに必要になります。

ストレージのサイズは、ワークロード、コンテナー数、実行されているコンテナーのサイズ、およびコンテナーのストレージ要件によって変わります。

重要

ホストが RHEL 7.5 を使用しており、OpenShift Container Platform のデフォルト **docker** 設定 (OverlayFS ストレージおよびすべてのデフォルトロギングオプションを使用) を受け入れる必要がある場合、これらのパッケージを手動でインストールしないでください。これらのパッケージは、インストール 時に **prerequisites.yml** Playbook を実行する場合にインストールされます。

ホストが RHEL 7.4 を使用するか、または RHEL 7.5 を使用し、**docker** 設定をカスタマイズする必要がある場合、以下のパッケージをインストールします。

RHEL 7 システムの場合:

RHEL 7 の Docker のデフォルトストレージバックエンドは、ループバックデバイスにあるシンプールです。これは実稼働環境でサポートされておらず、概念実証向けの環境のみに適しています。 実稼働環境の場合、シンプール論理ボリュームを作成し、Docker をそのボリュームを使用するよう再設定する必要があります。

Docker はグラフドライバーにイメージとコンテナーを保存します。これは、**DeviceMapper**、**OverlayFS**、**Btrfs** など、プラグ可能なテクノロジーです。これらにはそれぞれメリットとデメリットがあります。たとえば、OverlayFS は、コンテナーの開始と停止で高速ですが、

Portable Operating System Interface for Unix (POSIX) ではありません。これは、ユニオンファイルシステムの構造的な制限のためです。お使いの RHEL バージョンで OverlayFS を使用する方法についての情報は Red Hat Enterprise Linux リリースノートを参照してください。

DeviceMapper と OverlayFS のメリットと制限に関する情報は、Choosing a Graph Driver を参照してください。

RHEL Atomic Host 7 システムの場合:

RHEL Atomic Host の Docker のデフォルトストレージバックエンドはシンプール論理ボリュームで、実稼働環境でサポートされています。システム要件 にある Docker ストレージ要件に対して、このボリュームに十分なスペースが割り当てられていることを確認する必要があります。

十分なスペースが割り当てられていない場合、docker-storage-setup の使用と RHEL Atomic Host におけるストレージ管理の基本手順については、**Managing Storage with Docker Formatted Containers**を参照してください。

3.9.1. OverlayFS の設定

OverlayFS は、ユニオンファイルシステムのタイプです。OverlayFS により、あるファイルシステムを別のファイルシステムに重ねることができます。上位のファイルシステムで変更が記録されても、下位のファイルシステムは変更されません。

Comparing the Overlay vs. Overlay2 Graph Drivers には、**overlay** および **overlay2** ドライバーの詳細情報が記載されています。

overlay2 ドライバーを使用するには、下層が XFS ファイルシステムを使用する必要があります。下層のファイルシステムは、変更されないファイルシステムです。

Docker サービスの OverlayFS ストレージドライバーの有効化については、Red Hat Enterprise Linux Atomic Host ドキュメント を参照してください。

3.9.2. シンプールストレージの設定

Docker に含まれる **docker-storage-setup** スクリプトを使用してシンプールデバイスを作成し、Docker ストレージドライバーを設定できます。これは Docker のインストール後に実行でき、イメージまたはコンテナーの作成前に実行する必要があります。このスクリプトは **/etc/sysconfig/docker-storage-setup** ファイルから設定オプションを読み取り、論理ボリュームを作成するための 3 つのオプションをサポートします。

- 追加のブロックデバイスを使用する。
- 既存の、指定されたボリュームグループを使用する。
- root ファイルシステムが置かれている残りのボリュームグループの空きスペースを使用する。

追加のブロックデバイスを使用することは最も信頼性の高いオプションですが、Docker ストレージを設定する前に他のブロックデバイスをホストに追加する必要があります。他のオプションはいずれも、ホストのプロビジョニング時に利用可能な空きスペースを残しておく必要があります。ルートファイルシステムボリュームグループの残りの空きスペースを使用すると、Red Hat Mobile Application Platform (RHMAP) などの一部のアプリケーションで問題が生じることが確認されています。

1. 以下の 3 つのオプションのいずれかを使用して **docker-pool** ボリュームを作成します。
    - 追加のブロックデバイスを使用するには、以下を実行します。

a. **/etc/sysconfig/docker-storage-setup** で、使用するブロックデバイスのパスに **DEVS** を設定します。作成するボリュームグループ名に **VG** を設定します (**docker-vg** など)。以下は例になります。

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
DEVS=/dev/vdc
VG=docker-vg
EOF
```

b. **docker-storage-setup** を実行し、出力で **docker-pool** ボリュームが作成されたことを確認します。

```
# docker-storage-setup
[5/1868]
0
Checking that no-one is using this disk right now ...
OK

Disk /dev/vdc: 31207 cylinders, 16 heads, 63 sectors/track
sfdisk:  /dev/vdc: unrecognized partition table type

Old situation:
sfdisk: No partitions found

New situation:
Units: sectors of 512 bytes, counting from 0

   Device Boot    Start       End   #sectors  Id  System
/dev/vdc1          2048  31457279   31455232  8e  Linux LVM
/dev/vdc2             0         -          0   0  Empty
/dev/vdc3             0         -          0   0  Empty
/dev/vdc4             0         -          0   0  Empty
Warning: partition 1 does not start at a cylinder boundary
Warning: partition 1 does not end at a cylinder boundary
Warning: no primary partition is marked bootable (active)
This does not matter for LILO, but the DOS MBR will not boot this disk.
Successfully wrote the new partition table

Re-reading the partition table ...

If you created or changed a DOS partition, /dev/foo7, say, then use dd(1)
to zero the first 512 bytes:  dd if=/dev/zero of=/dev/foo7 bs=512 count=1
(See fdisk(8).)
  Physical volume "/dev/vdc1" successfully created
  Volume group "docker-vg" successfully created
  Rounding up size to full physical extent 16.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume docker-vg/docker-pool and docker-
vg/docker-poolmeta to pool's data and metadata volumes.
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted docker-vg/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

- 既存の、指定されたボリュームグループを使用するには、以下を実行します。

a. **/etc/sysconfig/docker-storage-setup** で、**VG** をボリュームグループに設定します。以下は例になります。

```
# cat <<EOF > /etc/sysconfig/docker-storage-setup
VG=docker-vg
EOF
```

b. 次に **docker-storage-setup** を実行し、出力で **docker-pool** ボリュームが作成されたことを確認します。

```
# docker-storage-setup
  Rounding up size to full physical extent 16.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume docker-vg/docker-pool and docker-
vg/docker-poolmeta to pool's data and metadata volumes.
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted docker-vg/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

- root ファイルシステムが置かれているボリュームグループの残りの空きスペースを使用するには、以下を実行します。

a. root ファイルシステムが置かれているボリュームグループに必要な空きスペースがあることを確認してから、**docker-storage-setup** を実行して、出力で **docker-pool** ボリュームが作成されていることを確認します。

```
# docker-storage-setup
  Rounding up size to full physical extent 32.00 MiB
  Logical volume "docker-poolmeta" created.
  Logical volume "docker-pool" created.
  WARNING: Converting logical volume rhel/docker-pool and rhel/docker-poolmeta to
pool's data and metadata volumes.
  THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
  Converted rhel/docker-pool to thin pool.
  Logical volume "docker-pool" changed.
```

2. 設定を確認します。**/etc/sysconfig/docker-storage** ファイルに **dm.thinpooldev** および **docker-pool** 論理ボリュームの値があることを確認します。

```
# cat /etc/sysconfig/docker-storage
DOCKER_STORAGE_OPTIONS="--storage-driver devicemapper --storage-opt dm.fs=xfs --
storage-opt dm.thinpooldev=/dev/mapper/rhel-docker--pool --storage-opt
dm.use_deferred_removal=true --storage-opt dm.use_deferred_deletion=true "

# lvs
  LV          VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  docker-pool rhel twi-a-t---  9.29g             0.00   0.12
```

重要

Docker または OpenShift Container Platform を使用する前に、**docker-pool** 論理ボリュームが要求を満たすサイズであることを確認します。**docker-pool** ボリュームは利用可能なボリュームグループの 60% である必要があり、これは LVM モニターリングによって拡張し、ボリュームグループを埋めていきます。

3. Docker を起動するか、または再起動します。
   - Docker がホストでまだ起動されていない場合は、サービスを有効にしてから起動し、それが実行されていることを確認します。

```
# systemctl enable docker
# systemctl start docker
# systemctl is-active docker
```

   - Docker がすでに実行されている場合は、以下を実行します。
     a. Docker を再初期化します。

警告

これは現在ホストにあるコンテナーまたはイメージを破棄します。

```
# systemctl stop docker
# rm -rf /var/lib/docker/*
# systemctl restart docker
```

     b. **/var/lib/docker/** フォルダーのコンテンツを削除します。

3.9.3. Docker ストレージの再設定

docker-pool を作成した後に Docker ストレージを再設定する必要がある場合は、以下を実行します。

1. **docker-pool** 論理ボリュームを削除します。
2. 専用のボリュームグループを使用する場合、ボリュームグループおよび関連付けられた物理ボリュームを削除します。
3. **docker-storage-setup** を再び実行します。

LVM 管理の詳細は、論理ボリュームマネージャー管理 を参照してください。

3.9.4. イメージ署名サポートの有効化

OpenShift Container Platform は、イメージが信頼済みのソースのものかを暗号で確認することができます。Container Security Guide には、イメージ署名の仕組みの概要が記載されています。

atomic コマンドラインインターフェイス (CLI) (バージョン 1.12.5 以降) を使用してイメージ署名の検証を設定できます。**atomic** CLI は RHEL Atomic Host システムにプリインストールされています。

注記

atomic CLI の詳細は、Atomic CLI についてのドキュメント を参照してください。

以下のファイルとディレクトリーは、ホストの信頼設定を設定しています。

- **/etc/containers/registries.d/***
- **/etc/containers/policy.json**

信頼設定は、各ノードで直接管理するか、または個別のホストでファイルを管理でき、Ansible などを使用してそれらのファイルを適切なノードに配布できます。Ansible を使用したファイル配布の自動化の例については、Container Image Signing Integration Guide を参照してください。

1. ホストシステムにインストールされていない場合は、**atomic** パッケージをインストールします。

```
$ yum install atomic
```

2. 現在の信頼設定を表示します。

```
$ atomic trust show
* (default)                    accept
```

デフォルト設定はすべてのレジストリーをホワイトリストに入れます。 つまり、署名の検証は設定されません。

3. 信頼設定をカスタマイズします。以下の例では、1 つのレジストリーまたは namespace をホワイトリストに入れ、信頼されていないレジストリーをブラックリストに入れ (拒否) ます。 これには、ベンダーレジストリーでの署名の検証が必要になります。

```
$ atomic trust add --type insecureAcceptAnything 172.30.1.1:5000

$ atomic trust add --sigstoretype atomic \
  --pubkeys pub@example.com \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype atomic \
  --pubkeys /etc/pki/example.com.pub \
  172.30.1.1:5000/production

$ atomic trust add --sigstoretype web \
  --sigstore https://access.redhat.com/webassets/docker/content/sigstore \
  --pubkeys /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release \
  registry.redhat.io

# atomic trust show
* (default)                    accept
172.30.1.1:5000                 accept
172.30.1.1:5000/production      signed security@example.com
registry.redhat.io              signed security@redhat.com,security@redhat.com
```

4. グローバル **reject** デフォルト信頼を追加してノードをさらに強化できます。

```
$ atomic trust default reject
```

```
$ atomic trust show
* (default)                    reject
172.30.1.1:5000                accept
172.30.1.1:5000/production     signed security@example.com
registry.redhat.io             signed security@redhat.com,security@redhat.com
```

5. オプションで、詳細のオプションについて **atomic** man ページ **man atomic-trust** を確認します。

3.9.5. コンテナーログの管理

コンテナーのログファイル (コンテナーが実行されているノード上の **/var/lib/docker/containers/<hash>/<hash>-json.log** ファイル) が問題を生じさせかねないサイズに拡張してしまうことを防ぐために、Docker の **json-file** ロギングドライバーを設定し、ログファイルのサイズと数を制限できます。

オプション	目的
--log-opt max-size	作成される新規ログファイルのサイズを設定します。
--log-opt max-file	ホストごとに保持するログファイルの最大数を設定します。

1. ログファイルを設定するには、**/etc/sysconfig/docker** ファイルを編集します。たとえば、最大ファイルサイズを 1MB に設定し、最大の 3 つのログファイルを保持するには、**max-size=1M** および **max-file=3** を **OPTIONS=** 行に追加し、値が単一引用符のフォーマットをベースとしていることを確認します。

```
OPTIONS='--insecure-registry=172.30.0.0/16 --selinux-enabled --log-opt max-size=1M --log-opt max-file=3'
```

ロギングドライバーの設定 方法に関する詳細は、Docker ドキュメントを参照してください。

2. Docker サービスを再起動します。

```
# systemctl restart docker
```

3.9.6. 利用可能なコンテナーログの表示

コンテナーログは、コンテナーが実行されているノードの **/var/lib/docker/containers/<hash>/** ディレクトリーで確認できます。以下は例になります。

```
# ls -lh
/var/lib/docker/containers/f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8/

total 2.6M
-rw-r--r--. 1 root root 5.6K Nov 24 00:12 config.json
-rw-r--r--. 1 root root 649K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log
-rw-r--r--. 1 root root 977K Nov 24 00:15
```

```
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log.1
-rw-r--r--. 1 root root 977K Nov 24 00:15
f088349cceac173305d3e2c2e4790051799efe363842fdab5732f51f5b001fd8-json.log.2
-rw-r--r--. 1 root root 1.3K Nov 24 00:12 hostconfig.json
drwx------. 2 root root    6 Nov 24 00:12 secrets
```

3.9.7. ローカルボリューム使用のブロック

ボリュームのプロビジョニングが **Dockerfile** の **VOLUME** 指示または **docker run -v <volumename>** コマンドを使用して実行されると、ホストのストレージ領域が使用されます。このストレージを使用すると、予期しない領域不足の問題が生じ、ホストが停止する可能性があります。

OpenShift Container Platform では、独自のイメージを実行しようとするユーザーには、ノードホストのストレージ領域全体が一杯になるリスクがあります。この問題に対する 1 つの解決策として、ユーザーがボリュームを持つイメージを実行できないようにする方法があります。これにより、ユーザーがアクセスできるストレージのみを制限し、クラスター管理者はストレージのクォータを割り当てることができます。

docker-novolume-plugin を使用して、ローカルボリュームが定義されたコンテナーの起動を禁止することにより、この問題を解決することができます。とくに、このプラグインは以下を含む **docker run** コマンドをブロックします。

- **--volumes-from** オプション
- **VOLUME** が定義されたイメージ
- **docker volume** コマンドを使ってプロビジョニングされた既存ボリュームの参照

プラグインはバインドマウントへの参照をブロックしません。

docker-novolume-plugin を有効にするには、各ノードホストで以下の手順を実行します。

1. **docker-novolume-plugin** パッケージをインストールします。

```
$ yum install docker-novolume-plugin
```

2. **docker-novolume-plugin** サービスを有効にし、起動します。

```
$ systemctl enable docker-novolume-plugin
$ systemctl start docker-novolume-plugin
```

3. **/etc/sysconfig/docker** ファイルを編集し、以下を **OPTIONS** 一覧に追加します。

```
--authorization-plugin=docker-novolume-plugin
```

4. **docker** サービスを再起動します。

```
$ systemctl restart docker
```

このプラグインを有効にした後に、ローカルボリュームが定義されたコンテナーは起動に失敗し、以下のエラーメッセージを表示します。

```
runContainer: API error (500): authorization denied by plugin
docker-novolume-plugin: volumes are not allowed
```

3.10. RED HAT GLUSTER STORAGE ソフトウェア要件

GlusterFS ボリュームにアクセスするには、すべてのスケジュール可能なノードで **mount.glusterfs** コマンドを利用できる必要があります。RPM ベースのシステムの場合は、**glusterfs-fuse** パッケージがインストールされている必要があります。

```
# yum install glusterfs-fuse
```

このパッケージはすべての RHEL システムにインストールされています。ただし、サーバーが x86_64 アーキテクチャーを使用する場合は Red Hat Gluster Storage の最新バージョンに更新することを推奨します。そのためには、以下の RPM リポジトリーを有効にする必要があります。

```
# subscription-manager repos --enable=rh-gluster-3-client-for-rhel-7-server-rpms
```

glusterfs-fuse がノードにすでにインストールされている場合、最新バージョンがインストールされていることを確認します。

```
# yum update glusterfs-fuse
```

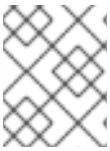
3.11. 次のステップ

ホストの準備が完了した後に、OpenShift Container Platform をインストールしている場合、インベントリーファイルを設定 します。スタンドアロンレジストリーをインストールしている場合は、代わりにスタンドアロンレジストリーのインストール を続行してください。

第4章 インベントリーファイルの設定

4.1. クラスターのインベントリーファイルのカスタマイズ

Ansible インベントリーファイルはクラスター内のホストについての詳細や OpenShift Container Platform インストールのクラスター設定の詳細を記述します。OpenShift Container Platform インストール Playbook は、ホストのセットに OpenShigt Container Platform をインストールする方法を判別するためにインベントリーファイルを読み取ります。



注記

インベントリーファイルの形式についての詳細 (YAML 構文 の基本事項を含む) については、Ansible ドキュメント を参照してください。

ホストの準備で説明されているように openshift-ansible RPM パッケージをインストールする場合、Ansible の依存関係は **/etc/ansible/hosts** のデフォルトの場所でファイルを作成します。ただし、ファイルは単純にデフォルトの Ansible のサンプルであり、OpenShift Container Platform 設定にとくに関連している変数はありません。OpenShift Container Platform を適切にインストールするには、ファイルのデフォルトの内容を、クラスターのトポロジーおよび要件に基づいて独自の設定に置き換える **必要があります**。

以下のセクションでは、クラスターインストール時にインベントリーファイルに設定する一般的に使用される変数について説明します。説明されている Ansible 変数の多くはオプションです。開発環境の場合、必須のパラメーターのデフォルト値を受け入れますが、実稼働環境ではそれらについての適切な値を選択する必要があります。

まず、いくつかの例の インベントリーファイルのサンプル を確認し、クラスターインストールの開始時に使用します。

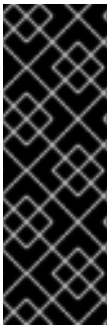
注記

イメージには更新を維持するためにバージョン番号ポリシーが必要です。詳細は Architecture Guide の Image Version Tag Policy のセクションを参照してください。

4.2. クラスター変数の設定

Ansible インストール時にグローバルクラスター環境変数を割り当てるには、それらを **/etc/ansible/hosts** ファイルの **[OSEv3:vars]** セクションに追加します。それぞれのパラメーター値を別個の行に配置する必要があります。以下は例になります。

```
[OSEv3:vars]

openshift_master_identity_providers=[{'name': 'htpasswd_auth',
'login': 'true', 'challenge': 'true',
'kind': 'HTPasswdPasswordIdentityProvider',}]

openshift_master_default_subdomain=apps.test.example.com
```



重要

Ansible インベントリーファイルのパラメーター値に、**#**, **{** or **}** などの特殊文字が含まれている場合、値をダブルエスケープ (double-escape) する必要があります (値を単一と二重引用符で囲みます)。たとえば、**mypasswordwith###hashsigns** を変数 **openshift_cloudprovider_openstack_password** の値として使用し、これを Ansible ホストインベントリーファイルで **openshift_cloudprovider_openstack_password="'mypasswordwith###hashsigns"'** として宣言します。

以下の表は、Ansible インストーラーで使用するグローバルクラスター変数について説明しています。

表4.1 一般的なクラスター変数

変数	目的
ansible_ssh_user	この変数はインストーラーで使用する SSH ユーザーを設定します。これは、デフォルトでは **root** になります。このユーザーは、パスワードなし で SSH ベースの認証を許可する必要があります。SSH キーベースの認証を使用する場合、そのキーは SSH エージェントで管理される必要があります。
ansible_become	**ansible_ssh_user** が **root** でない場合、この変数は **true** に設定する必要があり、ユーザーをパスワードレスの **sudo** 用に設定する必要があります。
debug_level	この変数は、 ログを **systemd-journald.service** に記録する INFO メッセージを設定します。以下のいずれかを設定します。 ● **0**: エラーおよび警告のみをログに記録 ● **2**: 通常の情報をログに記録 (これはデフォルトのレベルです) ● **4**: デバッグレベルの情報をログに記録 ● **6**: API レベルのデバッグ情報 (要求/応答) をログに記録 ● **8**: 本体レベルの API デバッグ情報をログに記録 デバッグログのレベルについての詳細は、ロギングレベルの設定 を参照してください。

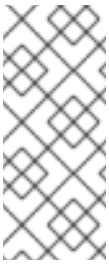
変数	目的
openshift_clock_enabled	クラスターノードでネットワークタイムプロトコル (NTP) を有効にするかどうか。デフォルト値は **true** です。 **chrony** パッケージがインストールされている場合、これは NTP サービスを提供するように設定されます。**chrony** パッケージがインストールされない場合、インストール Playbook が **ntp** パッケージをインストールし、設定して NTP サービスを提供します。  **重要** クラスター内のマスターおよびノードが同期されなくなる状態を防ぐには、このパラメーターのデフォルト値を変更しないでください。

変数	目的
openshift_master_admission_plugin_config	この変数は、インベントリーホストファイルの要件に基づいてパラメーターと任意の JSON 値を設定します。以下は例になります。 openshift_master_admission_plugin_config={"ClusterResourceOverride":{"configuration":{"apiVersion":"v1","kind":"ClusterResourceOverrideConfig","memoryRequestToLimitPercent":"25","cpuRequestToLimitPercent":"25","limitCPUToMemoryPercent":"200"}}} この値では、**openshift_master_admission_plugin_config={"openshift.io/ImagePolicy":{"configuration":{"apiVersion":"v1","executionRules":[{"matchImageAnnotations":[{"key":"images.openshift.io/deny-execution","value":"true"}],"name":"execution-denied","onResources":[{"resource":"pods"},{"resource":"builds"}],"reject":true,"skipOnResolutionFailure":true}],"kind":"ImagePolicyConfig"}}}** がデフォルトのパラメーター値になります。 **重要** カスタム設定を追加する必要がある場合でも、デフォルトの **openshift_master_admission_plugin_config** 値を含める必要があります。
openshift_master_audit_config	この変数は API サービスの監査を有効にします。詳細は、監査の設定 を参照してください。
openshift_master_audit_policyfile	監査ポリシーファイルの場所を指定します。詳細は、監査ポリシー設定 について参照してください。
openshift_master_cluster_hostname	この変数はクラスターのホスト名を上書きします。デフォルトはマスターのホスト名です。

変数	目的
openshift_master_cluster_public_hostname	この変数はクラスターのパブリックホスト名を上書きします。デフォルトはマスターのホスト名です。外部ロードバランサーを使用する場合は、外部ロードバランサーのアドレスを指定します。 以下は例になります。 `openshift_master_cluster_public_hostname=openshift-ansible.public.example.com`
openshift_master_cluster_method	オプション。この変数は複数マスターのデプロイ時の HA メソッドを定義します。**native** メソッドをサポートします。詳細は、複数マスター を参照してください。
openshift_rolling_restart_mode	この変数は、アップグレード Playbook を直接実行する時に HA マスターのローリング再起動 (例: マスターは一度に 1 つずつ停止します) を有効にします。これはデフォルトで **services** となっており、マスターでのサービスのローリング再起動を許可します。また **system** に設定することもでき、これによりマスターノードのローリングおよび完全な再起動が可能になります。 マスターのローリング再起動は、アップグレード時に提供される Ansible Hook を使用して追加の変更を適用するのに必要になる場合があります。実行するように選択するタスクに応じて、サービスを再起動するためにホストの再起動が必要になる可能性があります。
openshift_master_identity_providers	この変数は、アイデンティティープロバイダー を設定します。デフォルト値は Deny All です。サポートされているアイデンティティープロバイダーを使用する場合は、OpenShift Container Platform をそれを使用するように設定します。複数のアイデンティティープロバイダーを設定できます。
openshift_master_named_certificates	これらの変数は、インストールの一部としてデプロイされる カスタム証明書 を設定するために使用されます。詳細は、カスタム証明書の設定 を参照してください。
openshift_master_overwrite_named_certificates	
openshift_hosted_router_certificate	ホストされているルーターの カスタム証明書 の場所を指定します。

変数	目的
openshift_master_ca_certificate	単一証明書、および OpenShift Container Platform 証明書に署名するキーを指定します。新規またはカスタムの OpenShift Container Platform CA の再デプロイ を参照してください。
openshift_additional_ca	**openshift_master_ca_certificate** パラメーターの証明書が中間証明書で署名される場合、CA の中間証明書とルート証明書の完全なチェーンが含まれているバンドルされた証明書を指定します。新規またはカスタムの OpenShift Container Platform CA の再デプロイ を参照してください。
openshift_redeploy_service_signer	パラメーターが **false** に設定されている場合、サービス署名側は **openshift-master/redeploy-certificates.yml** Playbook の実行時に再デプロイされません。デフォルト値は **true** です。
openshift_hosted_registry_cert_expire_days	自動生成されるレジストリー証明書の有効日数。デフォルトで **730** (2 年) に設定されます。
openshift_ca_cert_expire_days	自動生成される CA 証明書の有効日数。デフォルトで **1825** (5 年) に設定されます。
openshift_master_cert_expire_days	自動生成されるマスター証明書の有効日数。デフォルトで **730** (2 年) に設定されます。
etcd_ca_default_days	自動生成される外部 etcd 証明書の有効日数。etcd CA、ピア、サーバー、クライアント証明書の有効性を管理します。デフォルトで **1825** (5 年) に設定されます。
openshift_certificate_expiry_warning_days	この日数内に有効期限が切れる証明書のあるクラスターへのアップグレードを停止します。デフォルトは **365** (1 年) に設定されます。
openshift_certificate_expiry_fail_on_warn	自動生成された証明書が **openshift_certificate_expiry_warning_days** パラメーターで指定された期間に無効になる場合にアップグレードが失敗するかどうか。デフォルトは **True** に設定されます。
os_firewall_use_firewalld	**true** に設定され、デフォルトの iptables ではなく firewalld が使用されます。RHEL Atomic Host では利用できません。詳細は ファイアウォールの設定 のセクションを参照してください。

変数	目的
openshift_master_session_name	これらの変数は OAuth 設定の セッションオプション のデフォルトを上書きします。詳細は セッションオプションの設定 を参照してください。
openshift_master_session_max_seconds	
openshift_master_session_auth_secrets	
openshift_master_session_encryption_secrets	
openshift_master_image_policy_config	マスター設定で **imagePolicyConfig** を設定します。詳細は イメージ設定 を参照してください。
openshift_router_selector	ルーター Pod を自動的にデプロイするためのデフォルトのノードセレクター。詳細は、ノードホストラベルの設定 を参照してください。
openshift_registry_selector	レジストリー Pod を自動的にデプロイするためのデフォルトのノードセレクター。詳細は、ノードホストラベルの設定 を参照してください。
openshift_template_service_broker_namespaces	この変数は、ブローカーが提供するテンプレートの 1 つ以上の namespace を指定することでテンプレートサービスブローカーを有効にします。
openshift_master_bootstrap_auto_approve	この変数は TLS ブートストラップの自動承認を有効にします。Amazon Web Services (AWS) クラスターで クラスターの自動スケーラー を有効にする場合は **true** に設定します。デフォルト値は **false** です。
ansible_service_broker_node_selector	Ansible サービスブローカー Pod を自動的にデプロイするためのデフォルトのノードセレクター。デフォルトは **{"node-role.kubernetes.io/infra":"true"}**。詳細は、ノードホストラベルの設定 を参照してください。
osm_default_node_selector	この変数は、Pod を配置する際にプロジェクトがデフォルトで使用するノードセレクターを上書きします。これは、マスター設定ファイルの **projectConfig.defaultNodeSelector** フィールドで定義されています。これが定義されていない場合はデフォルトで **node-role.kubernetes.io/compute=true** に設定されます。

変数	目的
openshift_docker_additional_registries	OpenShift Container Platform は指定された追加レジストリーを **docker** 設定に追加します。これらは検索対象のレジストリーです。このレジストリーへのアクセスに必要なレジストリーが **80** 以外の場合は、**<address>:<port>** の形式でポート番号を含める必要があります。 以下は例になります。 `openshift_docker_additional_registries=example.com:443`  **注記** クラスターを別のレジストリーを使用するように設定する必要がある場合は、**openshift_docker_additional_registries** を使用するのではなく、**oreg_url** を設定します。
openshift_docker_insecure_registries	OpenShift Container Platform は指定された追加の非セキュアなレジストリーを **docker** 設定に追加します。それらのレジストリーの SSL (Secure Sockets Layer) は検証されません。ホスト名またはホストの IP アドレスに設定できます。**0.0.0.0/0** は IP アドレスの有効な設定ではありません。
openshift_docker_blocked_registries	OpenShift Container Platform は指定されたブロック済みレジストリーを **docker** 設定に追加します。これは一覧表示されるレジストリーをブロックします。これを **all** に設定すると、他の変数で使用されていないすべてのレジストリーがブロックされます。
openshift_docker_ent_reg	**openshift_deployment_type** が **openshift-enterprise** に設定されている場合にコンテナーランタイムによって信頼される追加のレジストリー。デフォルトは **registry.redhat.io** です。**openshift_docker_ent_reg=''** を設定する場合、**registry.redhat.io** は **docker** 設定に追加されません。
openshift_metrics_hawkular_hostname	この変数は、マスター設定でクラスターメトリクスの **metricsPublicURL** を上書きすることで、メトリクスコンソールと統合するホスト名を設定します。この変数を変更する場合は、ホスト名がルーター経由でアクセスできることを確認してください。

変数	目的
openshift_clusterid	この変数は AWS アベイラビリティーゾーン固有のクラスター識別子です。これを使用することで、複数のゾーンまたは複数のクラスターを持つ Amazon Web Service (AWS) での潜在的な問題を回避することができます。詳細は AWS のクラスターへのラベル付け を参照してください。
openshift_encryption_config	この変数を使用して、データストア層の暗号化を設定します。
openshift_image_tag	この変数を使用して、インストールまたは設定するコンテナーイメージタグを指定します。
openshift_pkg_version	この変数を使用して、インストールまたは設定する RPM バージョンを指定します。

警告

クラスターのセットアップ後に **openshift_image_tag** または **openshift_pkg_version** 変数を変更する場合はアップグレードがトリガーされ、ダウンタイムが発生します。

- **openshift_image_tag** が設定されている場合、この値は別のバージョンがインストールされている場合でもシステムコンテナー環境のすべてのホストに使用されます。if
- **openshift_pkg_version** が設定されている場合、この値は別のバージョンがインストールされている場合でも RPM ベースの環境のすべてのホストに使用されます。

表4.2 ネットワーク変数

変数	目的
openshift_master_default_subdomain	この変数は、公開される ルート に使用するデフォルトのサブドメインを上書きします。この変数の値は、小文字の英数字またはダッシュ (-) で設定される必要があります。アルファベット文字で始まり、英数字の文字で終了する必要があります。

変数	目的
os_sdn_network_plugin_name	この変数は、どの OpenShift SDN プラグイン を Pod ネットワークに使用するかを設定します。これは、標準の SDN プラグインでは、**redhat/openshift-ovs-subnet** がデフォルトになります。マルチテナント SDN プラグインを使用するには、変数を **redhat/openshift-ovs-multitenant** に設定します。
osm_cluster_network_cidr	この変数は SDN クラスターネットワーク CIDR ブロックを上書きします。これは、Pod IP の割り当て元のネットワークです。このブロックは非公開ブロックとし、Pod、ノード、またはマスターがアクセスする必要のある可能性があるインフラストラクチャーの既存のネットワークブロックと競合しないようにする必要があります。デフォルトは **10.128.0.0/14** であり、デプロイ後は SDN マスター設定 でこれに一部の変更を加えられることがありますが、任意に再設定することはできません。
openshift_portal_net	この変数は、サービス を OpenShift Container Platform SDN 内で作成する際のサブネットを設定します。このブロックは非公開とし、Pod、ノード、またはマスターがアクセスする必要の可能性があるインフラストラクチャーの既存のネットワークブロックと競合しないようにする必要があります。デフォルトは **172.30.0.0/16** で、デプロイメント後に再設定することはできません。デフォルトは **172.30.0.0/16** であり、デプロイ後はこれを再設定することはできません。 デフォルトを変更する場合は、**docker0** ネットワークブリッジがデフォルトで使用する 172.17.0.0/16 の使用を避けるようにするか、または **docker0** ネットワークを変更します。
osm_host_subnet_length	この変数は、OpenShift Container Platform SDN により Pod IP のホストごとに割り当てられるサブネットのサイズを指定します。デフォルトは **9** であり、これは各ホストにサイズが /23 のサブネットが割り当てられることを意味します (例: デフォルトの 10.128.0.0/14 クラスターネットワーク)。これは、10.128.0.0/23、10.128.2.0/23、10.128.4.0/23 などを割り当てます。これはデプロイ後に再設定することはできません。
openshift_node_proxy_mode	この変数は、使用する サービスプロキシーモード を指定します。 **iptables** (デフォルトの純粋な **iptables** 実装) または **userspace** (ユーザー空間プロキシー) のいずれかを指定します。

変数	目的
openshift_use_flannel	この変数は、デフォルトの SDN の代わりに **flannel** を代替ネットワーキングレイヤーとして有効にします。**flannel** を有効にする場合は、**openshift_use_openshift_sdn** 変数を使用してデフォルトの SDN を無効にしてください。詳細は、Flannel の使用 を参照してください。
openshift_use_openshift_sdn	OpenShift SDN プラグインを無効にするには、**false** に設定します。
openshift_sdn_vxlan_port	この変数は、**cluster network** の **vxlan port** 番号を設定します。デフォルトは **4789** に設定されます。詳細は、クラスターネットワークの VXLAN ポートの変更 を参照してください。
openshift_node_sdn_mtu	この変数は、OpenShift SDN に使用する MTU サイズを指定します。値は、ノードのプライマリーネットワークインターフェイスの MTU よりも **50** バイト未満である必要があります。たとえば、プライマリーネットワークインターフェイスに MTU が **1500** の場合、この値は **1450** になります。デフォルト値は **1450** です。

4.3. デプロイメントタイプの設定

Playbook 全体で使用される各種デフォルト値とインストーラーによって使用されるロールは、デプロイメントタイプの設定 (通常は Ansible インベントリーファイルで定義されます) に基づいて決定されます。

OpenShift Container Platform バリアントをインストールするには、インベントリーファイルの **[OSEv3:vars]** セクションにある **openshift_deployment_type** パラメーターが **openshift-enterprise** に設定されていることを確認してください。

```
[OSEv3:vars]
openshift_deployment_type=openshift-enterprise
```

4.4. ホスト変数の設定

Ansible のインストール時に環境変数をホストに割り当てるには、**[masters]** セクションまたは **[nodes]** セクションにホストを入力した後に **/etc/ansible/hosts** ファイルで必要な変数を指定します。以下は例になります。

```
[masters]
ec2-52-6-179-239.compute-1.amazonaws.com openshift_public_hostname=ose3-
master.public.example.com
```

以下の表は、Ansible インストーラーで使用され、個々のホストエントリーに割り当てることができる変数を示しています。

表4.3 ホスト変数

変数	目的
openshift_public_hostname	この変数は、システムのパブリックホスト名を上書きします。クラウドインストールやネットワークアドレス変換 (NAT) を使用するネットワーク上のホストに使用します。
openshift_public_ip	この変数は、システムのパブリック IP アドレスを上書きします。クラウドインストールやネットワークアドレス変換 (NAT) を使用するネットワーク上のホストに使用します。
openshift_node_labels	この変数は非推奨になっています。 現在のノードラベルの設定方法については、ノードグループおよびホストマッピングの定義 を参照してください。
openshift_docker_options	この変数は、Managing Container Logs で使用されるオプションなど、**/etc/sysconfig/docker** 内に追加の **docker** オプションを設定します。**json-file** を使用することを推奨します。 以下の例は、**json-file** ログドライバーを使用するための Docker の設定を示しています。ここでは、Docker は、3 つの 1 MB ファイル間でローテーションし、署名検証は必要ありません。追加のオプションを指定する場合は、単一引用符のフォーマットが使用されることを確認します。 `OPTIONS=' --selinux-enabled --log-opt max-size=1M --log-opt max-file=3 --insecure-registry 172.30.0.0/16 --log-driver=json-file --signature-verification=false'`
openshift_schedulable	この変数は、ホストがスケジュール可能ノードとしてマークされているかどうか、つまり、新しい Pod を配置できるかどうかを設定します。マスターでのスケジュール可能性の設定 を参照してください。
openshift_node_problem_detector_install	この変数は、Node Problem Detector をアクティブにするために使用されます。**false, the default** に設定される場合、Node Problem Detector はインストールされず、起動しません。

4.5. ノードグループおよびホストマッピングの定義

ノードの設定はマスターから ブートストラップ されるようになりました。ノードおよびサービスが起動されると、ノードは、**kubeconfig** および他のノード設定ファイルが存在するかどうかをクラスター

に参加する前に確認します。存在しない場合には、ノードはマスターから設定をプルしてから、クラスターに参加します。

このプロセスにより、管理者は各ノードホストで一意のノード設定を手動で維持する必要があります。その代わりに、ノードホストの **/etc/origin/node/node-config.yaml** ファイルの内容がマスターから取得される ConfigMap によって提供されるようになりました。

4.5.1. ノードの ConfigMap

ノード設定の定義用の ConfigMap は **openshift-node** プロジェクトで利用できる状態でなければなりません。ConfigMap はノードラベルの信頼できる定義となり、以前の **openshift_node_labels** の値は事実上、無視されます。

デフォルトで、クラスターのインストール時にインストーラーは以下のデフォルト ConfigMap を作成します。

- **node-config-master**
- **node-config-infra**
- **node-config-compute**

以下の ConfigMap も作成され、複数のロールにノードをラベル付けします。

- **node-config-all-in-one**
- **node-config-master-infra**

以下の ConfigMap は、それぞれの既存のデフォルトノードグループの **CRI-O** バリアントです。

- **node-config-master-crio**
- **node-config-infra-crio**
- **node-config-compute-crio**
- **node-config-all-in-one-crio**
- **node-config-master-infra-crio**

重要

ノードホストの **/etc/origin/node/node-config.yaml** ファイルを変更することはできません。変更については、ノードが使用する ConfigMap で定義さえる設定で上書きされます。

4.5.2. ノードグループの定義

最新の **openshift-ansible** パッケージのインストール後に、ノードグループ定義のデフォルトセットを **/usr/share/ansible/openshift-ansible/roles/openshift_facts/defaults/main.yml** ファイル内で YAML 形式で確認することができます。

```
openshift_node_groups:
  - name: node-config-master (1)
    labels:
      - 'node-role.kubernetes.io/master=true' (2)
```

```
    edits: [] 3
  - name: node-config-infra
    labels:
      - 'node-role.kubernetes.io/infra=true'
    edits: []
  - name: node-config-compute
    labels:
      - 'node-role.kubernetes.io/compute=true'
    edits: []
  - name: node-config-master-infra
    labels:
      - 'node-role.kubernetes.io/infra=true,node-role.kubernetes.io/master=true'
    edits: []
  - name: node-config-all-in-one
    labels:
      - 'node-role.kubernetes.io/infra=true,node-role.kubernetes.io/master=true,node-
role.kubernetes.io/compute=true'
    edits: []
```

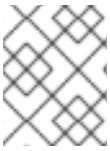
1 ノードのグループ名です。

2 ノードグループに関連付けられるノードラベルの一覧です。詳細は、ノードホストラベル を参照してください。

3 ノードグループの設定の編集です。

openshift_node_groups 変数をインベントリーファイルの **[OSEv3:vars]** グループに設定しない場合、これらのデフォルト値が使用されます。ただし、カスタムノードグループを設定する必要がある場合、計画されているすべてのノードグループを含む、**openshift_node_groups** 構造をインベントリーファイルに定義する必要があります。

openshift_node_groups 値はデフォルト値にマージされないため、YAML 定義を Python ディクショナリーに変換する必要があります。次に、**edits** フィールドを設定し、キーと値のペアを指定してノード設定変数を変更できます。



注記

設定可能なノード変数に関する情報は、Master and Node Configuration Files を参照してください。

たとえば、インベントリーファイルの以下のエントリーは、**node-config-master**、**node-config-infra** および **node-config-compute** という名前のグループを定義します。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true']}]
```

新規ノードグループ名を他のラベルで定義することもできます。 インベントリーファイルの以下のエントリーは、**node-config-master**、**node-config-infra**、**node-config-compute** および **node-config-compute-storage** という名前のグループを定義します。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
```

```
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true']}, {'name': 'node-config-compute-storage', 'labels': ['node-
role.kubernetes.io/compute-storage=true']}]
```

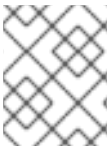
- **node-config-compute** グループを **kubelet** に 2 つのパラメーターを追加するために変更するなど、複数のキーと値のペアを変更するために一覧を使用できます。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'kubeletArguments.experimental-allocatable-ignore-
eviction','value': ['true']}, {'key': 'kubeletArguments.eviction-hard', 'value': ['memory.available<1Ki']}]}]
```

- **node-config-compute** グループを **perFSGroup** を **512Mi** に設定するために変更するなど、ディクショナリーを値として使用することができます。

```
openshift_node_groups=[{'name': 'node-config-master', 'labels': ['node-
role.kubernetes.io/master=true']}, {'name': 'node-config-infra', 'labels': ['node-
role.kubernetes.io/infra=true']}, {'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{ 'key': 'volumeConfig.localQuota','value':
{'perFSGroup':'512Mi'}}]}]
```

openshift_node_group.yml Playbook が実行されるたびに、**edits** フォールドで定義した変更により、関連の ConfigMap (この例では **node-config-compute**) が更新され、最終的にホスト上のノードの設定ファイルに影響を与えます。



注記

openshift_node_group.yaml Playbook を実行すると、新規ノードのみが更新されます。クラスター内の既存ノードを更新するために実行することはできません。

4.5.3. ホストとノードグループのマッピング

どのノードホストにどの ConfigMap を使用するかについてのマッピングでは、インベントリーの **[nodes]** グループで定義されるすべてのホストが **openshift_node_group_name** 変数を使用して **node group** に割り当てられる必要があります。

重要

ホストごとに **openshift_node_group_name** をノードグループに設定することは、デフォルトのノードグループ定義および ConfigMap を使用しているか、または独自の設定をカスタマイズしているかにかかわらず、すべてのクラスターインストールで必要です。

openshift_node_group_name の値は、各ノードを設定する ConfigMap を選択するために使用されます。以下は例になります。

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
```

他のカスタム ConfigMaps が **openshift_node_groups** に定義されている場合、それらを使用することもできます。以下は例になります。

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
gluster[1:6].example.com openshift_node_group_name='node-config-compute-storage'
```

4.5.4. ノードホストラベル

クラスターインストール時に、ラベル をノードホストに割り当てることができます。これらのラベルを使用し、スケジューラー を使用して Pod のノードへの配置を判別できます。

ノードホストに割り当てられるデフォルトラベルを変更する必要がある場合、独自のカスタムノードグループを作成する必要があります。**openshift_node_labels** 変数を設定してラベルを変更することはできなくなりました。デフォルトノードグループを変更するには、ノードグループ定義 を参照してください。

node-role.kubernetes.io/infra=true (このグループを使用するホストは **専用インフラストラクチャーノード** とも呼ばれ、さらに 専用インフラストラクチャーノードの設定 で説明されています) 以外には、実際のラベル名および値は任意であり、クラスターの要件に基づいて適切とみなされる方法で割り当てることができます。

4.5.4.1. マスターでの Pod スケジュール可能性の設定

インストールプロセス時にマスターとして指定するすべてのホストをノードとして設定します。これにより、マスターは OpenShift SDN の一部として設定されます。マスターホストのエントリーを **[nodes]** セクションに追加する必要があります。

```
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
```

インストール後にホストのスケジュール可能性を変更したい場合は、ノードをスケジュール対象外 (Unschedulable) またはスケジュール対象 (Schedulable) としてマークする を参照してください。

4.5.4.2. ノードでの Pod スケジュール可能性の設定

マスターはデフォルトでスケジュール対象ノードとしてマークされるため、デフォルトノードセレクターは、クラスターのインストール時にデフォルトで設定されます。デフォルトノードセレクターは、Pod を配置する際にデフォルトでプロジェクトが使用するノードを判別するためにマスター設定ファイルの **projectConfig.defaultNodeSelector** フィールドに定義されます。これは、**osm_default_node_selector** 変数を使用して上書きされない限り、**node-role.kubernetes.io/compute=true** に設定されます。

重要

インストール時にデフォルトノードセレクター **node-role.kubernetes.io/compute=true** を受け入れる場合、クラスターで非マスターノードとして定義されているのが専用インフラストラクチャーノードだけでないことを確認してください。この場合、アプリケーション Pod はデプロイに失敗します。 プロジェクトの Pod のスケジュール時に、デフォルトノードセレクターに一致する **node-role.kubernetes.io/compute=true** ラベル付きのノードが存在しないためです。

インストール後に必要に応じてこの設定を調整する手順については、クラスター全体でのデフォルトノードセレクターの設定 を参照してください。

4.5.4.3. 専用インフラストラクチャーノードの設定

実稼働環境では、レジストリー Pod とルーター Pod をユーザーアプリケーション用の Pod とは別に実行できる専用インフラストラクチャーノードを保持することを推奨します。

openshift_router_selector および **openshift_registry_selector** Ansible 設定は、レジストリー Pod とルーター Pod を配置する際に使用されるラベルセレクターを決定します。これらはデフォルトで **node-role.kubernetes.io/infra=true** に設定されます。

```
# default selectors for router and registry services
# openshift_router_selector='node-role.kubernetes.io/infra=true'
# openshift_registry_selector='node-role.kubernetes.io/infra=true'
```

レジストリーとルーターは、**node-role.kubernetes.io/infra=true** ラベルが付いた、専用インフラストラクチャーノードと見なされるノードホスト上でのみ実行できます。お使いの OpenShift Container Platform 環境に、**node-role.kubernetes.io/infra=true** ラベルが付いたノードホストが 1 つ以上存在することを確認してください。 デフォルトの **node-config-infra** を使用してこのラベルを設定できます。

```
[nodes]
infra-node1.example.com openshift_node_group_name='node-config-infra'
```

重要

セレクター設定に一致するノードが **[nodes]** セクションにない場合、デフォルトのルーターとレジストリーはデプロイに失敗し、**Pending** ステータスになります。

レジストリーとルーターの管理に OpenShift Container Platform を使用しない場合は、以下のように Ansible 設定を行います。

```
openshift_hosted_manage_registry=false
openshift_hosted_manage_router=false
```

デフォルトの **registry.redhat.io** 以外のイメージレジストリーを使用する場合は、**/etc/ansible/hosts** ファイルで レジストリーを指定 する必要があります。

マスターでのスケジュール可能性の設定 で説明されているように、マスターホストはデフォルトでスケジュール可能としてマークされます。マスターホストに **node-role.kubernetes.io/infra=true** ラベルを付けており、他に専用インフラストラクチャーノードがない場合、マスターホストはスケジュール対象としてマークされる必要もあります。そうしないと、レジストリー Pod とルーター Pod をどこにも配置できなくなります。

これを実行するには、デフォルトの **node-config-master-infra** ノードグループを使用できます。

```
[nodes]
master.example.com openshift_node_group_name='node-config-master-infra'
```

4.6. プロジェクトパラメーターの設定

デフォルトのプロジェクト設定を設定するには、以下の変数を **/etc/ansible/hosts** ファイルに設定します。

表4.4 プロジェクトパラメーター

パラメーター	説明	タイプ	デフォルト値
osm_project_request_message	この文字列は、**projectrequest** API エンドポイントからプロジェクトを要求できない場合にユーザーに提示されます。	文字列	null
osm_project_request_template	projectrequest への応答としてプロジェクトを作成する際に使用されるテンプレートです。値を指定しない場合は、デフォルトのテンプレートが使用されます。	**<namespace>/<template>** 形式の文字列	null
osm_mcs_allocator_range	namespace に割り当てる MCS カテゴリーの範囲を定義します。この値が起動後に変更される場合、新規プロジェクトは他のプロジェクトにすでに割り当てられているラベルを受信する可能性があります。接頭辞には、ユーザー、ロール、およびタイプを含め、一連の有効な SELinux 用語を使用できます。ただし、接頭辞をデフォルトとして残しておくと、サーバーはそれらを自動的に設定できます。たとえば、**s0:/2** は s0:c0,c0 から s0:c511,c511 にラベルを割り当て、**s0:/2,512** は s0:c0,c0,c0 から s0:c511,c511,511 にラベルを割り当てます。	**<prefix>/<numberOfLabels>[,<maxCategory>]** 形式の文字列	**s0:/2**

パラメーター	説明	タイプ	デフォルト値
osm_mcs_labels_per_project	プロジェクトごとに保持するラベルの数を定義します。	整数	**5**
osm_uid_allocator_range	プロジェクトに自動的に割り当てられる Unix ユーザー ID (UID) の合計セット数と、各 namespace が取得するブロックのサイズを定義します。たとえば、**1000-1999/10** は namespace ごとに 10 の UID を割り当て、スペースが不足する前に最大 100 ブロックを割り当てることができます。デフォルト値は、ユーザー namespace の起動時にコンテナーイメージが使用する範囲の予想されるサイズです。	**<block_range>/<number_of_UIDs>** 形式の文字列	**1000000000-1999999999/10000**

4.7. マスター API ポートの設定

マスター API で使用するデフォルトのポートを設定するには、**/etc/ansible/hosts** ファイルに以下の変数を設定します。

表4.5 マスター API ポート

変数	目的
openshift_master_api_port	この変数は、OpenShift Container Platform API へのアクセスに使用するポート番号を設定します。

以下は例になります。

```
openshift_master_api_port=3443
```

Web コンソールポート設定 (**openshift_master_console_port**) は、 API サーバーのポート (**openshift_master_api_port**) に一致している必要があります。

4.8. クラスターのプレインストールチェックの設定

プレインストールチェックは、**openshift_health_checker** Ansible ロールの一部として実行される診断タスクのセットです。OpenShift Container Platform の Ansible インストールの前に実行され、必要なインベントリー値が設定されていることを確認し、正常なインストールを妨げたり干渉したりする可能性があるホストの潜在的な問題を特定します。

以下の表は、OpenShift Container Platform のすべての Ansible インストールの前に実行される、使用可能なプレインストールチェックを示しています。

表4.6 プレインストールチェック

チェック名	目的
memory_availability	このチェックでは、OpenShift Container Platform の特定のデプロイメントで推奨されるメモリー容量がホストにあることを確認します。デフォルト値は、最新のインストールドキュメント から取得されたものです。インベントリーファイルで **openshift_check_min_host_memory_gb** クラスター変数を設定することで、最小メモリー要件のユーザー定義値を設定できます。
disk_availability	このチェックは、etcd、マスター、およびノードホストに対してのみ実行されます。OpenShift Container Platform インストールのマウントパスに十分なディスク領域が残っていることを確認します。推奨されるディスク値は、最新のインストールドキュメント から取得されます。ディスク領域最小要件のユーザー定義の値は、インベントリーファイルで **openshift_check_min_host_disk_gb** クラスター変数を設定することで設定できます。
docker_storage	**docker** デーモン (ノードとコンテナー化インストール) に依存するホストでのみ実行されます。ユーザー定義の上限が設定されていない場合、**docker** の最大使用率のしきい値はデフォルトで使用可能な合計サイズの 90% になります。ユーザー定義の制限が設定されていない場合、**docker** 使用量の最大しきい値のデフォルトは利用可能な合計サイズの 90% になります。(**max_thinpool_data_usage_percent=90**)。最大シンプール使用率のユーザー定義の上限は、インベントリーファイルで **max_thinpool_data_usage_percent** クラスター変数を設定することで設定できます。
docker_storage_driver	**docker** デーモンが OpenShift Containe Platform でサポートされているストレージドライバーを使用していることを確認します。**devicemapper** ストレージドライバーが使用されている場合は、ループバックデバイスが使用されていないことも確認します。詳細については、Docker's Use the Device Mapper Storage Driver ガイドを参照してください。
docker_image_availability	OpenShift Container Platform インストールで必要なイメージがローカルで、またはホストマシン上の1つ以上の設定済みコンテナーイメージレジストリー で使用可能であることの確認を試行します。

チェック名	目的
package_version	**yum** ベースのシステムで実行され、必要な OpenShift Container Platform パッケージの複数のリリースが利用可能かどうかを確認します。OpenShift の **enterprise** インストールの実行時にパッケージの複数のリリースが利用可能である場合は、各リリース用に複数の **yum** リポジトリーが有効になっていることが示され、インストールの際に問題になることがあります。
package_availability	OpenShift Container Platform の RPM インストールの前に実行されます。現在のインストールに必要な RPM パッケージが利用可能であることを確認します。
package_update	**yum** 更新またはパッケージのインストールが成功するかどうかを、実際にインストールを行ったり、ホストで **yum** を実行したりせずに確認します。

特定のプレインストールチェックを無効にするには、コンマ区切りのチェック名の一覧を指定した変数 **openshift_disable_check** をインベントリーファイルに組み込みます。以下は例になります。

```
openshift_disable_check=memory_availability,disk_availability
```

注記

既存のクラスターの診断用に実行するための類似のヘルスチェックセットが Ansible ベースのヘルスチェック に用意されています。証明書の有効期限をチェックするための別のチェックのセットについては、Redeploying Certificates を参照してください。

4.9. レジストリーの場所の設定

registry.redhat.io でデフォルトのレジストリーを使用する場合、以下の変数を設定する必要があります。

```
oreg_url=registry.redhat.io/openshift3/ose-${component}:${version}
oreg_auth_user="<user>"
oreg_auth_password="<password>"
```

レジストリーのアクセストークンの設定に関する詳細は、Red Hat コンテナーレジストリーの認証 を参照してください。

registry.redhat.io のデフォルト以外のイメージレジストリーを使用する場合、レジストリーを **/etc/ansible/hosts** ファイルに指定します。

```
oreg_url=example.com/openshift3/ose-${component}:${version}
openshift_examples_modify_imagestreams=true
```

表4.7 レジストリー変数

変数	目的
oreg_url	別のイメージの場所に設定されます。**registry.redhat.io** でデフォルトのレジストリーを使用していない場合に必要です。デフォルトコンポーネントは、**oreg_url** 値からイメージの接頭辞およびバージョンを継承します。デフォルトレジストリーおよび認証が必要なレジストリーの場合は、**oreg_auth_user** および **oreg_auth_password** を指定する必要があります。
openshift_examples_modify_imagestreams	デフォルト以外のレジストリーを参照している場合は **true** に設定します。イメージストリームの場所を **oreg_url** の値に変更します。
oreg_auth_user	**oreg_url** が認証を必要とするレジストリーを参照する場合、**oreg_auth_user** 変数を使用してユーザー名を指定します。また、パスワードを **oreg_auth_password** パラメーターの値として指定する必要もあります。デフォルトレジストリーを使用する場合、**registry.redhat.io** にアクセスできるユーザーを指定します。
oreg_auth_password	**oreg_url** が認証を必要とするレジストリーを参照する場合、**oreg_auth_password** 変数を使用してパスワードを指定します。また、ユーザー名を **oreg_auth_user** パラメーターの値として指定する必要もあります。デフォルトレジストリーを使用する場合、そのユーザーのパスワードまたはトークンを指定します。

注記

デフォルトレジストリーには、認証トークンが必要です。詳細は、Red Hat レジストリーへのアクセスおよびその設定 を参照してください。

以下は例になります。

```
oreg_url=example.com/openshift3/ose-${component}:${version}
oreg_auth_user=${user_name}
oreg_auth_password=${password}
openshift_examples_modify_imagestreams=true
```

4.10. レジストリールートの設定

ユーザーが OpenShift Container Platform クラスターの外部からイメージをプルして内部のコンテナーイメージレジストリーにプッシュできるように、**/etc/ansible/hosts** ファイルにレジストリールートを設定します。デフォルトでは、レジストリールートは **docker-registry-default.router.default.svc.cluster.local** です。

表4.8 レジストリールート変数

変数	目的
openshift_hosted_registry_routehost	必要なレジストリールートの値に設定します。ルートには、ルーターによって通信が管理されるインフラストラクチャーノードに解決される名前、またはデフォルトのアプリケーションサブドメインのワイルドカード値として設定したサブドメインのいずれかが含まれます。たとえば、**openshift_master_default_subdomain** パラメーターを **apps.example.com** に設定し、***.apps.example.com** がインフラストラクチャーノードまたはロードバランサーに解決される場合は、**registry.apps.example.com** をレジストリールートとして使用できます。
openshift_hosted_registry_routecertificates	レジストリー証明書へのパスを設定します。証明書の場所の値を指定しない場合、証明書が生成されます。以下の証明書の場所を定義できます。 • **certfile** • **keyfile** • **cafile**
openshift_hosted_registry_routetermination	以下のいずれかの値に設定します。 • edge ルーターで暗号化を終了し、送信先から提供される新しい証明書で再暗号化するには、**reencrypt** に設定します。 • 送信先で暗号化を終了するには、**passthrough** に設定します。トラフィックは送信先で復号化されます。

以下は例になります。

```
openshift_hosted_registry_routehost=<path>
openshift_hosted_registry_routetermination=reencrypt
openshift_hosted_registry_routecertificates= "{'certfile': '<path>/org-cert.pem', 'keyfile': '<path>/org-privkey.pem', 'cafile': '<path>/org-chain.pem'}"
```

4.11. ルーターのシャード化の設定

ルーターのシャード化 サポートは、インベントリーに適切なデータを指定することで有効になります。変数 **openshift_hosted_routers** は、一覧の形式のデータを保持します。データが渡されない場合、デフォルトのルーターが作成されます。ルーターのシャード化については、複数の組み合わせがあります。以下の例は、別個のノードでのルーターをサポートしています。

```
openshift_hosted_routers=[{'name': 'router1', 'certificate': {'certfile': '/path/to/certificate/abc.crt', 'keyfile': '/path/to/certificate/abc.key', 'cafile':
```

```
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router1', 'ports':
['80:80', '443:443']},

{'name': 'router2', 'certificate': {'certfile': '/path/to/certificate/xyz.crt',
'keyfile': '/path/to/certificate/xyz.key', 'cafile':
'/path/to/certificate/ca.crt'}, 'replicas': 1, 'serviceaccount': 'router',
'namespace': 'default', 'stats_port': 1936, 'edits': [{'action': 'append',
'key': 'spec.template.spec.containers[0].env', 'value': {'name': 'ROUTE_LABELS',
'value': 'route=external'}}], 'images':
'openshift3/ose-${component}:${version}', 'selector': 'type=router2', 'ports':
['80:80', '443:443']}]
```

4.12. RED HAT GLUSTER STORAGE の永続ストレージの設定

Red Hat Gluster Storage は、OpenShift Container Platform の 永続ストレージ および動的プロビジョニングを提供するように設定できます。OpenShift Container Platform 内のコンテナー化ストレージ (**コンバージドモード**) と、独自のノードでコンテナー化されていないノード (**インデペンデントモード**) の両方を使用することができます。

OpenShift Container Platform クラスターと対話する変数を使用して Red Hat Gluster Storage クラスターを設定します。**[OSEv3:vars]** グループで定義する変数は、ホスト変数、ロール変数、およびイメージ名およびバージョンタグ変数が含まれます。

glusterfs_devices ホスト変数を使用して、Red Hat Gluster Storage クラスターを管理するブロックデバイスの一覧を定義します。設定の各ホストに最低でも **glusterfs_devices** 変数が必要で、すべての設定には、パーティションや LVM PV のない 1 つのベアメタルデバイスが必要です。

ロール変数は、Red Hat Gluster Storage クラスターの新規または既存の OpenShift Container Platform クラスターに統合を制御します。ロール変数を複数定義することもできます。それぞれの変数には、統合 Docker レジストリーのストレージとして使用するための対応する変数もあり、オプションで個別の Red Hat Gluster Storage クラスターを設定します。

イメージ名とバージョンタグ変数を定義して、OpenShift Container Platform Pod が停止後にアップグレードされないようにすることができます。これにより、別の OpenShift Container Platform バージョンでクラスターが破損する可能性があります。これらの変数を定義して、すべてのコンテナー化コンポーネントのイメージ名とバージョンタグを指定することもできます。

追加情報と以下を含む例については、Red Hat Gluster Storage を使用する永続ストレージ を参照してください。

4.12.1. コンバージドモードの設定

重要

具体的なホストの準備と前提条件については、コンバージドモードに関する考慮事項 を参照してください。

1. インベントリーファイルの **[OSEv3:vars]** セクションに次の変数を追加し、設定に合わせてそれらを調整します。

```
[OSEv3:vars]
...
```

```
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
```

2. **[OSEv3:children]** セクションに **glusterfs** を追加して、 **[glusterfs]** グループを有効にします。

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. GlusterFS ストレージをホストする各ストレージノードのエントリーを含む **[glusterfs]** セクションを追加します。ノードごとに、 **glusterfs_devices** を GlusterFS クラスターの一部として完全に管理される raw ブロックデバイスの一覧に設定します。少なくとも 1 つのデバイスを一覧に含める必要があります。各デバイスはパーティションや LVM PV がないベアでなければなりません。変数は次の形式で指定します。

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "</path/to/device2>", ... ]'
```

以下に例を示します。

```
[glusterfs]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. **[glusterfs]** の下に一覧表示されているホストを **[nodes]** グループに追加します。

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-compute"
node12.example.com openshift_node_group_name="node-config-compute"
node13.example.com openshift_node_group_name="node-config-compute"
```

デプロイを正常に実行するには、有効なイメージタグが必要です。インベントリーファイルの以下の変数について interoperability matrix で説明されているように、**<tag>** を OpenShift Container Platform 3.11 と互換性のある Red Hat Gluster Storage のバージョンに置き換えます。

- **openshift_storage_glusterfs_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>**
- **openshift_storage_glusterfs_block_image=registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>**
- **openshift_storage_glusterfs_s3_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>**
- **openshift_storage_glusterfs_heketi_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>**
- **openshift_storage_glusterfs_registry_image=registry.redhat.io/rhgs3/rhgs-server-rhel7:<tag>**

- **openshift_storage_glusterfs_block_registry_image=registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7:<tag>**
- **openshift_storage_glusterfs_s3_registry_image=registry.redhat.io/rhgs3/rhgs-s3-server-rhel7:<tag>**
- **openshift_storage_glusterfs_heketi_registry_image=registry.redhat.io/rhgs3/rhgs-volmanager-rhel7:<tag>**

4.12.2. インデペンデントモードの設定

1. インベントリーファイルの **[OSEv3:vars]** セクションに次の変数を追加し、設定に合わせてそれらを調整します。

```
[OSEv3:vars]
...
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=true
openshift_storage_glusterfs_block_host_vol_size=100
openshift_storage_glusterfs_block_storageclass=true
openshift_storage_glusterfs_block_storageclass_default=false
openshift_storage_glusterfs_is_native=false
openshift_storage_glusterfs_heketi_is_native=true
openshift_storage_glusterfs_heketi_executor=ssh
openshift_storage_glusterfs_heketi_ssh_port=22
openshift_storage_glusterfs_heketi_ssh_user=root
openshift_storage_glusterfs_heketi_ssh_sudo=false
openshift_storage_glusterfs_heketi_ssh_keyfile="/root/.ssh/id_rsa"
```

2. **[OSEv3:children]** セクションに **glusterfs** を追加して、 **[glusterfs]** グループを有効にします。

```
[OSEv3:children]
masters
nodes
glusterfs
```

3. GlusterFS ストレージをホストする各ストレージノードのエントリーを含む **[glusterfs]** セクションを追加します。ノードごとに、 **glusterfs_devices** を GlusterFS クラスターの一部として完全に管理される raw ブロックデバイスの一覧に設定します。少なくとも 1 つのデバイスを一覧に含める必要があります。各デバイスはパーティションや LVM PV がないベアでなければなりません。また、**glusterfs_ip** をノードの IP アドレスに設定します。変数は次の形式で指定します。

```
<hostname_or_ip> glusterfs_ip=<ip_address> glusterfs_devices='[ "</path/to/device1/>", "
</path/to/device2>", ... ]'
```

以下は例になります。

```
[glusterfs]
gluster1.example.com glusterfs_ip=192.168.10.11 glusterfs_devices='[ "/dev/xvdc",
"/dev/xvdd" ]'
gluster2.example.com glusterfs_ip=192.168.10.12 glusterfs_devices='[ "/dev/xvdc",
```

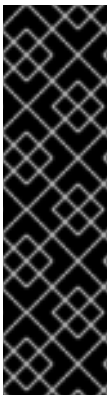
```
"/dev/xvdd" ]'
gluster3.example.com glusterfs_ip=192.168.10.13 glusterfs_devices='[ "/dev/xvdc",
"/dev/xvdd" ]'
```

4.13. OPENSHIFT CONTAINER レジストリーの設定

統合された OpenShift Container レジストリー は、インストーラーを使用してデプロイできます。

4.13.1. レジストリーストレージの設定

レジストリーストレージのオプションが使用されていない場合、デフォルトの OpenShift Container レジストリーは一時的で、Pod が存在しなくなるとすべてのデータが失われます。



重要

テストにより、RHEL NFS サーバーをコンテナーイメージレジストリー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 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 サーバー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 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。

通常インストーラー (advanced installer) を使用している場合にレジストリーストレージを有効にするには、以下のいくつかのオプションを選択できます。

オプション A: NFS ホストグループ
次の変数が設定されている場合、クラスターインストール時に **[nfs]** ホストグループ内のホストのパス **<nfs_directory>/<volume_name>** に NFS ボリュームが作成されます。たとえば、次のオプションを使用した場合、ボリュームパスは **/exports/registry** になります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_nfs_directory=/exports
openshift_hosted_registry_storage_nfs_options='*(rw,root_squash)'
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
```

オプション B: 外部 NFS ホスト
外部 NFS ボリュームを使用するには、該当する NFS ボリュームがストレージホストの **<nfs_directory>/<volume_name>** パスにすでに存在している必要があります。次のオプションを使用した場合、リモートボリュームパスは **nfs.example.com:/exports/registry** になります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character
```

```
openshift_hosted_registry_storage_kind=nfs
openshift_hosted_registry_storage_access_modes=['ReadWriteMany']
openshift_hosted_registry_storage_host=nfs.example.com
openshift_hosted_registry_storage_nfs_directory=/exports
openshift_hosted_registry_storage_volume_name=registry
openshift_hosted_registry_storage_volume_size=10Gi
```

NFS を使用した OpenShift Container Platform のアップグレードまたはインストール
オプション C: OpenStack プラットフォーム
OpenStack ストレージ設定がすでに存在している必要があります。

```
[OSEv3:vars]

openshift_hosted_registry_storage_kind=openstack
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_openstack_filesystem=ext4
openshift_hosted_registry_storage_openstack_volumeID=3a650b4f-c8c5-4e0a-8ca5-eaee11f16c57
openshift_hosted_registry_storage_volume_size=10Gi
```

オプション D: AWS または別の S3 ストレージソリューション
シンプルストレージソリューション (S3) バケットがすでに存在している必要があります。

```
[OSEv3:vars]

#openshift_hosted_registry_storage_kind=object
#openshift_hosted_registry_storage_provider=s3
#openshift_hosted_registry_storage_s3_accesskey=access_key_id
#openshift_hosted_registry_storage_s3_secretkey=secret_access_key
#openshift_hosted_registry_storage_s3_bucket=bucket_name
#openshift_hosted_registry_storage_s3_region=bucket_region
#openshift_hosted_registry_storage_s3_chunksize=26214400
#openshift_hosted_registry_storage_s3_rootdirectory=/registry
#openshift_hosted_registry_pullthrough=true
#openshift_hosted_registry_acceptschema2=true
#openshift_hosted_registry_enforcequota=true
```

Minio や ExoScale などの別の S3 サービスを使用している場合は、リージョンエンドポイントパラメーターも追加します。

```
openshift_hosted_registry_storage_s3_regionendpoint=https://myendpoint.example.com/
```

オプション E: コンバージドモード
コンバージドモードの設定 と同様に、Red Hat Gluster Storage はクラスターの初期インストール時に OpenShift Container レジストリーのストレージを提供するように設定できます。 これにより、冗長で信頼性の高いレジストリーのストレージを確保できます。

重要

具体的なホストの準備と前提条件については、コンバージドモードに関する考慮事項 を参照してください。

1. インベントリーファイルの **[OSEv3:vars]** セクションに次の変数を追加し、設定に合わせてそれらを調整します。

```
[OSEv3:vars]
...
openshift_hosted_registry_storage_kind=glusterfs (1)
openshift_hosted_registry_storage_volume_size=5Gi
openshift_hosted_registry_selector='node-role.kubernetes.io/infra=true'
```

(1) 統合 OpenShift Container Registry をインフラストラクチャーノードで実行することが推奨されます。インフラストラクチャーノードは、OpenShift Container Platform クラスターのサービスを提供するために管理者がデプロイするアプリケーションを実行する専用ノードです。

2. **[OSEv3:children]** セクションに **glusterfs_registry** を追加して、**[glusterfs_registry]** グループを有効にします。

```
[OSEv3:children]
masters
nodes
glusterfs_registry
```

3. GlusterFS ストレージをホストする各ストレージノードのエントリーを含む **[glusterfs_registry]** セクションを追加します。ノードごとに、 **glusterfs_devices** を GlusterFS クラスターの一部として完全に管理される raw ブロックデバイスの一覧に設定します。少なくとも 1 つのデバイスを一覧に含める必要があります。各デバイスはパーティションや LVM PV がないベアでなければなりません。変数は次の形式で指定します。

```
<hostname_or_ip> glusterfs_devices='[ "</path/to/device1/>", "</path/to/device2>", ... ]'
```

以下に例を示します。

```
[glusterfs_registry]
node11.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node12.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
node13.example.com glusterfs_devices='[ "/dev/xvdc", "/dev/xvdd" ]'
```

4. **[glusterfs_registry]** の下に一覧表示されているホストを **[nodes]** グループに追加します。

```
[nodes]
...
node11.example.com openshift_node_group_name="node-config-infra"
node12.example.com openshift_node_group_name="node-config-infra"
node13.example.com openshift_node_group_name="node-config-infra"
```

オプション F: Google Compute Engine (GCE) 上の Google Cloud Storage (GCS) バケット
GCS バケットがすでに存在している必要があります。

```
[OSEv3:vars]

openshift_hosted_registry_storage_provider=gcs
openshift_hosted_registry_storage_gcs_bucket=bucket01
openshift_hosted_registry_storage_gcs_keyfile=test.key
openshift_hosted_registry_storage_gcs_rootdirectory=/registry
```

オプション G: vSphere ボリュームおよび vSphere Cloud Provider (VCP)

vSphere Cloud Provider は、OpenShift Container Platform ノードでアクセスできるデータストアで設定される必要があります。

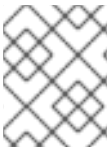
レジストリーに vSphere ボリュームを使用する場合、ストレージアクセスモードを **ReadWriteOnce** に設定し、レプリカ数を **1** に設定する必要があります。

```
[OSEv3:vars]

openshift_hosted_registry_storage_kind=vsphere
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_annotations=['volume.beta.kubernetes.io/storage-provisioner:
kubernetes.io/vsphere-volume']
openshift_hosted_registry_replicas=1
```

4.14. グローバルプロキシーオプションの設定

ホストが外部ホストに接続するために HTTP または HTTPS プロキシーを使用する必要がある場合は、プロキシーを使用するためにマスター、Docker、ビルドなどの多数のコンポーネントを設定する必要があります。ノードサービスは外部アクセスを必要としないマスター API にのみ接続するため、プロキシーを使用するように設定する必要はありません。

この設定を単純化するため、クラスターまたはホストレベルで次の Ansible 変数を指定し、これらの設定を環境全体に均一に適用することができます。



注記

ビルド用のプロキシー環境の定義方法の詳細については、グローバルビルドのデフォルトとオーバーライドの設定 を参照してください。

表4.9 クラスタープロキシー変数

変数	目的
openshift_http_proxy	この変数はマスターおよび Docker デーモンの **HTTP_PROXY** 環境変数を指定します。
openshift_https_proxy	この変数は、マスターおよび Docker デーモンの **HTTPS_PROXY** 環境変数を指定します。

変数	目的
openshift_no_proxy	この変数は、マスターおよび Docker デーモンに **NO_PROXY** 環境変数を設定するために使用されます。定義されたプロキシーを使用しないホスト名、ドメイン名、またはワイルドカードホスト名のコンマ区切りの一覧を提供します。デフォルトでは、この一覧は、定義済みのすべての OpenShift Container Platform ホスト名の一覧で拡張されます。 定義されたプロキシーを使用しないホスト名には、以下が含まれます。 ● マスターおよびノードのホスト名。ドメイン接尾辞を含める必要があります。 ● 他の内部ホスト名。ドメイン接尾辞を含める必要があります。 ● etcd IP アドレス。etcd アクセスは IP アドレスで管理されるため、IP アドレスを指定する必要があります。 ● コンテナーイメージレジストリー IP アドレス。 ● Kubernetes IP アドレス。この値はデフォルトで **172.30.0.1** であり、指定している場合は **openshift_portal_net** パラメーター値になります。 ● **cluster.local** Kubernetes 内部ドメイン接尾辞。 ● **svc** Kubernetes 内部ドメイン接尾辞。
openshift_generate_no_proxy_hosts	このブール変数は、すべての定義済み OpenShift ホストの名前と ***.cluster.local** が自動的に **NO_PROXY** 一覧に追加されるかどうかを指定します。デフォルトは **true** です。 このオプションを上書きするには **false** に設定します。
openshift_builddefaults_http_proxy	この変数は、**BuildDefaults** 受付コントローラーを使用して、ビルドに挿入される **HTTP_PROXY** 環境変数を定義します。このパラメーターを定義せず、**openshift_http_proxy** パラメーターを定義する場合、**openshift_http_proxy** 値が使用されます。**openshift_http_proxy** の値を問わず、デフォルトの http プロキシーを無効にするには、**openshift_builddefaults_http_proxy** 値を **False** に設定します。

変数	目的
openshift_builddefaults_https_proxy	この変数は、**BuildDefaults** 受付コントローラーを使用して、ビルドに挿入される **HTTPS_PROXY** 環境変数を定義します。このパラメーターを定義せず、**openshift_http_proxy** パラメーターを定義する場合、**openshift_https_proxy** 値が使用されます。**openshift_https_proxy** の値を問わず、デフォルトの http プロキシーを無効にするには、**openshift_builddefaults_https_proxy** 値を **False** に設定します。
openshift_builddefaults_no_proxy	この変数は、**BuildDefaults** 受付コントローラーを使用して、ビルドに挿入される **NO_PROXY** 環境変数を定義します。**openshift_no_proxy** 値の種類を問わず、ビルドのデフォルトの no proxy 設定を無効にするには、**openshift_builddefaults_no_proxy** 値を **False** に設定します。
openshift_builddefaults_git_http_proxy	この変数は、ビルド時に **git clone** 操作で使用される HTTP プロキシーを定義します。これは、**BuildDefaults** 管理者コントローラーを使用して定義されます。**openshift_http_proxy** 値の種類を問わず、ビルド時に **git clone** 操作のデフォルトの http プロキシーを無効にするには、**openshift_builddefaults_git _http_proxy** 値を **False** に設定します。
openshift_builddefaults_git_https_proxy	この変数は、ビルド時に **git clone** 操作で使用される HTTPS プロキシーを定義します。これは **BuildDefaults** 受付コントローラーを使用して定義されます。**openshift_https_proxy** 値の種類を問わず、ビルド時に **git clone** 操作のデフォルトの https プロキシーを無効にするには、**openshift_builddefaults_git_https_proxy** 値を **False** に設定します。

4.15. ファイアウォールの設定

重要

- デフォルトのファイアウォールを変更する場合は、不一致を防ぐために、クラスター内の各ホストが同じファイアウォールタイプを使用していることを確認してください。
- Atomic Host にインストールされた OpenShift Container Platform でファイアウォールを使用しないでください。 ファイアウォールは Atomic Host ではサポートされていません。

注記

iptables はデフォルトのファイアウォールですが、firewalld は新規インストールで推奨されるファイアウォールです。

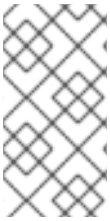
OpenShift Container Platform は iptables をデフォルトのファイアウォールとして使用しますが、クラスターをインストールプロセス時に firewalld を使用するように設定できます。

iptables はデフォルトのファイアウォールであるため、OpenShift Container Platform は iptables を自動的に設定するように設計されています。ただし、iptables ルールが適切に設定されていない場合、iptables ルールによって OpenShift Container Platform が中断する可能性があります。firewalld の利点の 1 つは、複数のオブジェクトでファイアウォールルールを安全に共有できることです。

firewalld を OpenShift Container Platform インストールのファイアウォールとして使用するには、インストール時に **os_firewall_use_firewalld** 変数を Ansible ホストファイルの設定変数の一覧に追加します。

```
[OSEv3:vars]
os_firewall_use_firewalld=True 1
```

1 この変数を **true** に設定することで、必要なポートが開き、ルールがデフォルトゾーンに追加されます。 これにより、firewalld が適切に設定されていることを確認できます。



注記

firewalld のデフォルトの設定オプションを使用する際には設定オプションが制限され、これらをオーバーライドすることはできません。たとえば、ストレージネットワークを複数ゾーンのインターフェイスでセットアップすることができますが、ノードが通信に使用するインターフェイスはデフォルトゾーンになければなりません。

4.16. セッションオプションの設定

OAuth 設定の セッションオプション はインベントリーファイルで設定できます。デフォルトで、Ansible は **sessionSecretsFile** を生成された認証および暗号化シークレットで設定し、1 つのマスターで生成されたセッションが他のマスターによって復号化されるようにできます。デフォルトの場所は **/etc/origin/master/session-secrets.yaml** であり、このファイルはすべてのマスターで削除された場合にのみ再作成されます。

openshift_master_session_name および **openshift_master_session_max_seconds** を使用してセッション名と最大秒数を設定できます。

```
openshift_master_session_name=ssn
openshift_master_session_max_seconds=3600
```

設定されている場合、**openshift_master_session_auth_secrets** および **openshift_master_encryption_secrets** は同じ長さでなければなりません。

HMAC を使用したセッションの認証に使用される **openshift_master_session_auth_secrets** の場合、32 バイトまたは 64 バイトのシークレットを使用することを推奨します。

```
openshift_master_session_auth_secrets=['DONT+USE+THIS+SECRET+b4NV+pmZNSO']
```

セッションの暗号化に使用される **openshift_master_encryption_secrets** の場合、シークレットの長さは AES-128、AES-192、または AES-256 を選択するできるようにそれぞれ 16、24、または 32 文字にする必要があります。

```
openshift_master_session_encryption_secrets=['DONT+USE+THIS+SECRET+b4NV+pmZNSO']
```

4.17. カスタム証明書の設定

OpenShift Container Platform API のパブリックホスト名と Web コンソール の カスタム提供証明書 は、クラスターインストール時にデプロイでき、インベントリーファイルで設定できます。

注記

openshift_master_cluster_public_hostname パラメーター値として設定する、**publicMasterURL** に関連付けられたホスト名のカスタム証明書を設定します。**masterURL** に関連付けられるホスト名 (**openshift_master_cluster_hostname**) のカスタム提供証明書を使用することにより、インフラストラクチャーコンポーネントが内部の **masterURL** ホストを使用してマスター API への接続を試行するために TLS エラーが生じます。

証明書とキーファイルのパスは、**openshift_master_named_certificates** クラスター変数を使用して設定できます。

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt", "keyfile":
"/path/to/custom1.key", "cafile": "/path/to/custom-ca1.crt"}]
```

ファイルパスは、Ansible が実行されるシステムに対してローカルである必要があります。証明書はマスターホストにコピーされ、**/etc/origin/master/named_certificates/** ディレクトリー内にデプロイされます。

Ansible は、証明書の **Common Name** と **Subject Alternative Names** を検出します。検出された名前は、**openshift_master_named_certificates** の設定時に **"names"** キーを指定して上書きできます。

```
openshift_master_named_certificates=[{"certfile": "/path/to/custom1.crt", "keyfile":
"/path/to/custom1.key", "names": ["public-master-host.com"], "cafile": "/path/to/custom-ca1.crt"}]
```

openshift_master_named_certificates を使用して設定される証明書はマスターにキャッシュされます。 つまり、別の証明書セットで Ansible を実行するたびに、以前にデプロイされたすべての証明書がマスターホストとマスターの設定ファイル内に残ることになります。

openshift_master_named_certificates を提供される値 (または値なし) で上書きする必要がある場合、**openshift_master_overwrite_named_certificates** クラスター変数を指定します。

```
openshift_master_overwrite_named_certificates=true
```

さらに詳細の例が必要な場合には、次のクラスター変数をインベントリーファイルに追加することを検討してください。

```
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lb-internal.openshift.com
openshift_master_cluster_public_hostname=custom.openshift.com
```

以降の Ansible 実行で証明書を上書きするには、以下のパラメーター値を設定します。

```
openshift_master_named_certificates=[{"certfile": "/root/STAR.openshift.com.crt", "keyfile":
"/root/STAR.openshift.com.key", "names": ["custom.openshift.com"], "cafile": "/root/ca-file.crt"}]
openshift_master_overwrite_named_certificates=true
```

重要

The **cafile** 証明書は、証明書のインストール時または再デプロイメント時にマスターの **ca-bundle.crt** ファイルにインポートされます。**ca-bundle.crt** ファイルは、OpenShift Container Platform で実行されるすべての Pod にマウントされます。複数の OpenShift Container Platform コンポーネントはデフォルトでは、**masterPublicURL** エンドポイントにアクセスする時に名前付き証明書を自動的に信頼します。certificates パラメーターから **cafile** オプションを省略すると、Web コンソールと他のコンポーネントの複数の機能は削減されます。

4.18. 証明書の有効性の設定

デフォルトで、etcd、マスター、kubelet の管理に使用される証明書は 2 から 5 年で有効期限切れになります。自動生成されるレジストリー、CA、ノードおよびマスター証明書の有効性 (有効期限が切れるまでの日数) は、以下の変数 (デフォルト値が表示されています) を使用してインストール時に設定できます。

```
[OSEv3:vars]

openshift_hosted_registry_cert_expire_days=730
openshift_ca_cert_expire_days=1825
openshift_master_cert_expire_days=730
etcd_ca_default_days=1825
```

これらの値は、 Ansible のインストール後での 証明書の再デプロイ 時にも使用されます。

4.19. クラスターモニターリングの設定

Prometheus クラスターモニターリングは、自動的にデプロイされるように設定されています。その自動的なデプロイメントを防ぐには、以下を設定します。

```
[OSEv3:vars]

openshift_cluster_monitoring_operator_install=false
```

Prometheus クラスターモニターリングおよびその設定についての詳細は、Prometheus クラスターモニターリングのドキュメント を参照してください。

4.20. クラスターメトリクスの設定

クラスターメトリクスは、自動的にデプロイされるように設定されていません。クラスターインストール時にクラスターメトリクスを有効にするには、以下を設定します。

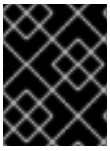
```
[OSEv3:vars]

openshift_metrics_install_metrics=true
```

メトリクスのパブリック URL は、クラスターのインストール時に **openshift_metrics_hawkular_hostname** Ansible 変数を使用して設定できます。 デフォルト値は以下の通りです。

https://hawkular-metrics.{{openshift_master_default_subdomain}}/hawkular/metrics

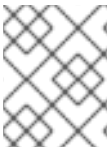
この変数を変更する場合は、ホスト名がルーター経由でアクセスできることを確認してください。

openshift_metrics_hawkular_hostname=hawkular-metrics.
{{openshift_master_default_subdomain}}



重要

アップストリームの Kubernetes ルールに応じて、**eth0** のデフォルトインターフェイスでのみメトリクスを収集できます。



注記

メトリクスをデプロイするために **openshift_master_default_subdomain** 値を設定する必要があります。

4.20.1. メトリクスストレージの設定

メトリクスに永続ストレージを使用するには、**openshift_metrics_cassandra_storage_type** 変数を設定する必要があります。**openshift_metrics_cassandra_storage_type** が設定されていない場合、クラスターのメトリクスデータは **emptyDir** ボリュームに保存されます。 このボリュームは、Cassandra Pod が終了すると削除されます。

重要

テストにより、RHEL NFS サーバーをコンテナーイメージレジストリー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、メトリクスストレージの Cassandra 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 サーバーを使用することは推奨されていません。

Cassandra は複数の独立したインスタンスにより冗長性を提供することを目的として設計されています。そのため、データディレクトリーに NFS または SAN を使用することは適切ではなく、推奨されていません。

ただし、他の NFS/SAN の実装ではこのコンポーネントのサポートやこのコンポーネントへのストレージの提供に関して問題が検出されない可能性があります。OpenShift 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/SAN 実装ベンダーにお問い合わせください。

クラスターインストール時にクラスターメトリクスストレージを有効にするには、次の 3 つのオプションを選択できます。

オプション A: 動的
OpenShift Container Platform 環境がクラウドプロバイダーの 動的ボリュームプロビジョニング をサポートする場合、以下の変数を使用します。

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=dynamic
```

gluster-storage および glusterfs-storage-block などのデフォルトで動的にプロビジョニングされたボリュームタイプが複数ある場合、変数でプロビジョニングされたボリュームタイプを指定できます。以下の変数を使用します。

```
[OSEv3:vars]

openshift_metrics_cassandra_storage_type=pv
openshift_metrics_cassandra_pvc_storage_class_name=glusterfs-storage-block
```

動的プロビジョニングを有効または無効にするために DynamicProvisioningEnabled を使用する方法についての詳細は、**Volume Configuration** を参照してください。

オプション B: NFS ホストグループ
次の変数が設定されている場合、NFS ボリュームはクラスターインストール時に **[nfs]** ホストグループ内のホストのパス **<nfs_directory>/<volume_name>** に作成されます。たとえば、以下のオプションを使用した場合、ボリュームパスは **/exports/metrics** になります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_nfs_directory=/exports
openshift_metrics_storage_nfs_options='*(rw,root_squash)'
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

オプション C: 外部 NFS ホスト
外部 NFS ボリュームを使用するには、該当する NFS ボリュームがストレージホストの **<nfs_directory>/<volume_name>** パスにすでに存在している必要があります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_metrics_storage_kind=nfs
openshift_metrics_storage_access_modes=['ReadWriteOnce']
openshift_metrics_storage_host=nfs.example.com
openshift_metrics_storage_nfs_directory=/exports
openshift_metrics_storage_volume_name=metrics
openshift_metrics_storage_volume_size=10Gi
```

以下のオプションを使用した場合、リモートボリュームのパスは **nfs.example.com:/exports/metrics** になります。

NFS を使用した OpenShift Container Platform のアップグレードまたはインストール
コアの OpenShift Container Platform コンポーネントでの NFS の使用は推奨されていません。 NFS (および NFS プロトコル) を使用すると、OpenShift Container Platform インフラストラクチャーを設定するアプリケーションに必要な適切な整合性が確保されなくなるためです。

そのため、インストーラーおよび更新 Playbook には、コアインフラストラクチャーコンポーネントで NFS の使用を有効にするオプションが必要になります。

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

クラスターのアップグレードまたはインストール時に以下のメッセージが表示される場合、追加の手順が必要になります。

```
TASK [Run variable sanity checks] ***********************************************
fatal: [host.example.com]: FAILED! => {"failed": true, "msg": "last_checked_host: host.example.com,
last_checked_var: openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind. openshift_enable_unsupported_configurations=True
mustbe specified to continue with this configuration."}
```

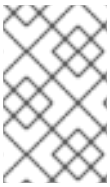
Ansible インベントリーファイルで、以下のパラメーターを指定します。

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.21. クラスターロギングの設定

クラスターロギングは、デフォルトで自動的にデプロイされるように設定されていません。クラスターインストール時にクラスターロギングを有効にするには、以下を設定します。

```
[OSEv3:vars]

openshift_logging_install_logging=true
```



注記

クラスターロギングのインストール時に、ノードセレクターも指定する必要があります (例: **openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra": "true"})** 。

利用可能なクラスターロギング変数についての詳細は、ロギング Ansible 変数の指定 を参照してください。

4.21.1. ロギングストレージの設定

ロギングに永続ストレージを使用するには、**openshift_logging_es_pvc_dynamic** 変数を設定する必要があります。**openshift_logging_es_pvc_dynamic** が設定されていない場合、クラスターのロギングデータは **emptyDir** ボリュームに保存されます。 このボリュームは、Elasticsearch Pod が終了すると削除されます。

重要

テストにより、RHEL NFS サーバーをコンテナーイメージレジストリー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、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 サーバーを使用することは推奨されていません。

ElasticSearch はカスタム deletionPolicy を実装しないため、NFS ストレージをボリュームまたは永続ボリュームとして使用することは Elasticsearch ストレージではサポートされていません。Lucene が NFS が指定しないファイルシステムの動作に依存するためです。 データの破損およびその他の問題が発生する可能性があります。

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 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。

クラスターインストール時にクラスターロギングストレージを有効にするには、次の 3 つのオプションを選択できます。

オプション A: 動的
OpenShift Container Platform 環境に動的ボリュームプロビジョニングがある場合、クラウドプロバイダー経由か、または独立したストレージプロバイダーによって設定されている可能性があります。たとえば、クラウドプロバイダーには GCE にプロビジョナー **kubernetes.io/gce-pd** が指定された StorageClass があり、GlusterFS などの独立したストレージプロバイダーには、プロビジョナー **kubernetes.io/glusterfs** が指定された **StorageClass** がある可能性があります。いずれの場合も、以下の変数を使用します。

```
[OSEv3:vars]

openshift_logging_es_pvc_dynamic=true
```

動的プロビジョニングについての詳細は、動的プロビジョニングとストレージクラスの作成 を参照してください。

gluster-storage および glusterfs-storage-block などのデフォルトで動的にプロビジョニングされたボリュームタイプが複数ある場合、変数でプロビジョニングされたボリュームタイプを指定できます。以下の変数を使用します。

```
[OSEv3:vars]

openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_storage_class_name=glusterfs-storage-block
```

動的プロビジョニングを有効または無効にするために DynamicProvisioningEnabled を使用する方法についての詳細は、**Volume Configuration** を参照してください。

オプション B: NFS ホストグループ
次の変数が設定されている場合、NFS ボリュームはクラスターインストール時に **[nfs]** ホストグループ内のホストのパス **<nfs_directory>/<volume_name>** に作成されます。たとえば、以下のオプションを使用した場合、ボリュームパスは **/exports/logging** になります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character
```

```
openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_nfs_directory=/exports 1
openshift_logging_storage_nfs_options='*(rw,root_squash)' 2
openshift_logging_storage_volume_name=logging 3
openshift_logging_storage_volume_size=10Gi
openshift_enable_unsupported_configurations=true
openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_size=10Gi
openshift_logging_es_pvc_storage_class_name=''
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_prefix=logging
```

1 2 これらのパラメーターは、**/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml** インストール Playbook でのみ動作します。このパラメーターは **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** Playbook では機能しません。

3 NFS ボリューム名は **logging** する必要があります。

オプション C: 外部 NFS ホスト
外部 NFS ボリュームを使用するには、該当する NFS ボリュームがストレージホストの **<nfs_directory>/<volume_name>** パスにすでに存在している必要があります。

```
[OSEv3:vars]

# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_logging_storage_kind=nfs
openshift_logging_storage_access_modes=['ReadWriteOnce']
openshift_logging_storage_host=nfs.example.com 1
openshift_logging_storage_nfs_directory=/exports 2
openshift_logging_storage_volume_name=logging 3
openshift_logging_storage_volume_size=10Gi
openshift_enable_unsupported_configurations=true
openshift_logging_elasticsearch_storage_type=pvc
openshift_logging_es_pvc_size=10Gi
openshift_logging_es_pvc_storage_class_name=''
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_prefix=logging
```

1 2 これらのパラメーターは、**/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml** インストール Playbook でのみ動作します。このパラメーターは **/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml** Playbook では機能しません。

3 NFS ボリューム名は **logging** する必要があります。

以下のオプションを使用した場合、リモートボリュームのパスは **nfs.example.com:/exports/logging** になります。

NFS を使用した OpenShift Container Platform のアップグレードまたはインストール

コアの OpenShift Container Platform コンポーネントでの NFS の使用は推奨されていません。 NFS (および NFS プロトコル) を使用すると、OpenShift Container Platform インフラストラクチャーを設定するアプリケーションに必要な適切な整合性が確保されなくなるためです。

そのため、インストーラーおよび更新 Playbook には、コアインフラストラクチャーコンポーネントで NFS の使用を有効にするオプションが必要になります。

```
# Enable unsupported configurations, things that will yield a partially
# functioning cluster but would not be supported for production use
#openshift_enable_unsupported_configurations=false
```

クラスターのアップグレードまたはインストール時に以下のメッセージが表示される場合、追加の手順が必要になります。

```
TASK [Run variable sanity checks] ***********************************************
fatal: [host.example.com]: FAILED! => {"failed": true, "msg": "last_checked_host: host.example.com,
last_checked_var: openshift_hosted_registry_storage_kind;nfs is an unsupported type for
openshift_hosted_registry_storage_kind. openshift_enable_unsupported_configurations=True
mustbe specified to continue with this configuration."}
```

Ansible インベントリーファイルで、以下のパラメーターを指定します。

```
[OSEv3:vars]
openshift_enable_unsupported_configurations=True
```

4.22. サービスカタログオプションのカスタマイズ

サービスカタログ はインストール時にデフォルトで有効にされます。サービスブローカーを有効にすると、サービスブローカーをカタログに登録できます。サービスカタログが有効にされると、OpenShift Ansible Broker およびテンプレートサービスブローカーの両方もインストールされます。詳細は、OpenShift Ansible Broker の設定 および テンプレートサービスブローカーの設定 を参照してください。サービスカタログを無効にする場合は、OpenShift Ansible Broker およびテンプレートサービスブローカーはインストールされません。

サービスカタログの自動デプロイメントを無効にするには、以下のクラスター変数をインベントリーファイルに設定します。

```
openshift_enable_service_catalog=false
```

独自のレジストリーを使用する場合、以下を追加する必要があります。

- **openshift_service_catalog_image_prefix**: サービスカタログイメージをプルする際に、特定の接頭辞 (例: **registry**) の使用を強制的に実行します。(イメージ名までの) 詳細なレジストリー名を指定する必要があります。
- **openshift_service_catalog_image_version**: サービスカタログイメージをプルする際に、特定のイメージバージョンの使用を強制的に実行します。

以下は例になります。

```
openshift_service_catalog_image="docker-registry.default.example.com/openshift/ose-service-
catalog:${version}"
openshift_service_catalog_image_prefix="docker-registry-default.example.com/openshift/ose-"
openshift_service_catalog_image_version="v3.9.30"
```

4.22.1. OpenShift Ansible Broker の設定

OpenShift Ansible Broker (OAB) は、インストール時にデフォルトで有効になります。

OAB をインストールしない場合は、インベントリーファイルで **ansible_service_broker_install** パラメーター値を **false** に設定します。

```
ansible_service_broker_install=false
```

表4.10 サービスブローカーのカスタマイズ変数

変数	目的
openshift_service_catalog_image_prefix	サービスカタログコンポートイメージの接頭辞を指定します。

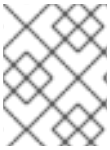
4.22.1.1. OpenShift Ansible Broker 用の永続ストレージの設定

OAB は、残りの OpenShift Container Platform クラスターが使用する etcd とは別に独自の etcd インスタンスをデプロイします。OAB の etcd インスタンスが機能するためには、永続ボリューム (PV) を使用する個別のストレージが必要です。使用可能な PV がない場合、etcd は PV の条件が満たされるまで待機します。OAB アプリケーションは、etcd インスタンスが使用可能になるまで **CrashLoop** 状態になります。

一部の Ansible Playbook Bundle (APB) でも、デプロイに専用の PV が必要になります。たとえば、APB の各データベースには 2 つのプランがあります。 開発プランは一時的なストレージを使用し、PV を必要としませんが、実稼働プランは永続的であり、PV を必要とします。

APB	PV が必要？
postgresql-apb	必要 (ただし実稼働プランの場合のみ必要)
mysql-apb	必要 (ただし実稼働プランの場合のみ必要)
mariadb-apb	必要 (ただし実稼働プランの場合のみ必要)
mediawiki-apb	Yes

OAB の永続ストレージを設定するには、以下の手順を実行します。



注記

以下の例では、NFS ホストを使用して必要な PV を提供しています。 ただし、他の永続ストレージプロバイダー を代わりに使用することもできます。

1. インベントリーファイルの **[OSEv3:children]** セクションに **nfs** を追加して、 **[nfs]** グループを有効にします。

```
[OSEv3:children]
masters
```

```
nodes
nfs
```

2. **[nfs]** グループセクションを追加し、NFS ホストになるシステムのホスト名を追加します。

```
[nfs]
master1.example.com
```

3. **[OSEv3:vars]** セクションに以下を追加します。

```
# nfs_directory must conform to DNS-1123 subdomain must consist of lower case
# alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character

openshift_hosted_etcd_storage_kind=nfs
openshift_hosted_etcd_storage_nfs_options="*(rw,root_squash,sync,no_wdelay)"
openshift_hosted_etcd_storage_nfs_directory=/opt/osev3-etcd (1)
openshift_hosted_etcd_storage_volume_name=etcd-vol2 (2)
openshift_hosted_etcd_storage_access_modes=["ReadWriteOnce"]
openshift_hosted_etcd_storage_volume_size=1G
openshift_hosted_etcd_storage_labels={'storage': 'etcd'}
```

(1) (2) NFS ボリュームが **[nfs]** グループ内のホストの **<nfs_directory>/<volume_name>** パスに作成されます。たとえば、これらのオプションを使用した場合、ボリュームパスは **/opt/osev3-etcd/etcd-vol2** になります。

これらの設定は、クラスターのインストール時に OAB の etcd インスタンスに割り当てられる永続ボリュームを作成します。

4.22.1.2. ローカルの APB 開発用の OpenShift Ansible Broker の設定

OpenShift Container レジストリーと OAB を組み合わせて APB 開発 を行うには、OAB がアクセスできるイメージのホワイトリストを定義する必要があります。ホワイトリストが定義されていない場合、ブローカーは APB を無視し、使用可能な APB がユーザーに表示されません。

デフォルトでは、ホワイトリストは空になっており、クラスター管理者がブローカーを設定するまでユーザーが APB イメージをブローカーに追加できないようになっています。**-apb** で終了するすべてのイメージをホワイトリストに入れるには、以下の手順を実行します。

1. インベントリーファイルの **[OSEv3:vars]** セクションに以下を追加します。

```
ansible_service_broker_local_registry_whitelist=['.*-apb$']
```

4.22.2. テンプレートサービスブローカーの設定

テンプレートサービスブローカー (TSB) は、インストール時にデフォルトで有効になります。

TSB をインストールしない場合は、**template_service_broker_install** パラメーターの値を **false** に設定します。

```
template_service_broker_install=false
```

TSB を設定するには、テンプレートとイメージストリームをサービスカタログに読み込めるように 1 つ以上のプロジェクトをブローカーのソースの namespace として定義する必要があります。インベント

リーファイルの **[OSEv3:vars]** セクションの以下の箇所を変更して、ソースプロジェクトを設定します。

```
openshift_template_service_broker_namespaces=['openshift','myproject']
```

表4.11 テンプレートサービスブローカーのカスタマイズ変数

変数	目的
template_service_broker_prefix	テンプレートサービスブローカーのコンポーネントイメージの接頭辞を指定します。
ansible_service_broker_image_prefix	Ansible サービスブローカーのコンポーネントイメージの接頭辞を指定します。

4.23. WEB コンソールのカスタマイズの設定

以下の Ansible 変数は、Web コンソールをカスタマイズするためのマスター設定オプションを設定します。これらのカスタマイズオプションの詳細については、Web コンソールのカスタマイズ を参照してください。

表4.12 Web コンソールのカスタマイズ変数

変数	目的
openshift_web_console_install	Web コンソールをインストールするかどうかを決定します。**true** または **false** に設定できます。デフォルトは **true** です。
openshift_web_console_prefix	Web コンソールイメージの接頭辞を指定します。
openshift_master_logout_url	Web コンソールの設定で **clusterInfo.logoutPublicURL** を設定します。詳細は、ログアウト URL の変更 を参照してください。値の例: **https://example.com/logout**
openshift_web_console_extension_script_urls	Web コンソールの設定で **extensions.scriptURLs** を設定します。詳細は、 拡張スクリプトとスタイルシートの読み込み を参照してください。値の例: **['https://example.com/scripts/menu-customization.js','https://example.com/scripts/nav-customization.js']**
openshift_web_console_extension_stylesheet_urls	Web コンソール設定で **extensions.stylesheetURLs** を設定します。詳細は、 拡張スクリプトとスタイルシートの読み込み を参照してください。値の例: **['https://example.com/styles/logo.css','https://example.com/styles/custom-styles.css']**

変数	目的
openshift_master_oauth_template	マスター設定で OAuth テンプレートを設定します。詳細については、ログインページのカスタマイズ を参照してください。値の例: **['/path/to/login-template.html']**
openshift_master_metrics_public_url	マスター設定で **metricsPublicURL** を設定します。詳細は、メトリクスのパブリック URL の設定 を参照してください。値の例: **https://hawkular-metrics.example.com/hawkular/metrics**
openshift_master_logging_public_url	マスター設定で **loggingPublicURL** を設定します。詳細は、Kibana を参照してください。値の例: **https://kibana.example.com**
openshift_web_console_inactivity_timeout_minutes	アクティブでない状態が一定期間続いた後にユーザーを自動的にログアウトするように Web コンソールを設定します。5 以上の整数を指定する必要があります。0 では、この昨日は無効になります。デフォルトは 0 (無効) です。
openshift_web_console_cluster_resource_overrides_enabled	クラスターがオーバーコミット対応に設定されているかどうかを示すブール値。**true** の場合、リソースの上限の編集時に、Web コンソールには CPU 要求、CPU 制限およびメモリー要求のフィールドは表示されません。 これらの値は、クラスターリソースのオーバーライド設定で設定する必要があるためです。
openshift_web_console_enable_context_selector	2 つのコンソール間で簡単に切り換えられるように、Web コンソールおよび管理コンソールのタイトルでコンテキストセレクターを有効にします。両方のコンソールがインストールされている場合、デフォルトは **true** に設定されます。

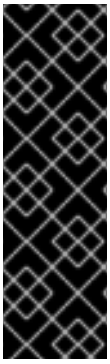
4.24. クラスターコンソールの設定

クラスターコンソールは Web コンソールのような Web インターフェイスですが、主に管理タスクに重点が置かれます。クラスターコンソールは Web コンソールと同じ共通の OpenShift Container Platform リソースの多くをサポートしますが、これにより、クラスターについてのメトリクスを表示したり、ノード、永続ボリューム、クラスターロール、およびカスタムリソース定義などの cluster スコープのリソースを管理したりすることができます。以下の変数は、クラスターコンソールをカスタマイズするために使用できます。

表4.13 クラスターコンソールのカスタマイズ変数

変数	目的
openshift_console_install	クラスターコンソールをインストールするかどうかを決定します。**true** または **false** に設定できます。デフォルトは **true** です。
openshift_console_hostname	クラスターコンソールのホスト名を設定します。デフォルトを **console.<openshift_master_default_subdomain>** に設定します。この変数を変更する場合は、ホスト名がルーター経由でアクセスできることを確認してください。
openshift_console_cert	クラスターコンソールルートに使用するオプションの証明書です。これは、カスタムホスト名を使用している場合にのみ必要です。
openshift_console_key	クラスターコンソールルートに使用するオプションのキーです。これは、カスタムホスト名を使用している場合にのみ必要です。
openshift_console_ca	クラスターコンソールルートに使用するオプションの CA です。これは、カスタムホスト名を使用している場合にのみ必要です。
openshift_base_path	クラスターコンソールのオプションのベースパスです。設定される場合、**/console/** のようにスラッシュで開始し、終了する必要があります。デフォルトで / に設定されます (ベースパスなし)。
openshift_console_auth_ca_file	OAuth サーバーに接続するために使用するオプションの CA ファイルです。デフォルトで **/var/run/secrets/kubernetes.io/serviceaccount/ca.crt** に設定されます。通常、この値は変更する必要はありません。CA ファイルが含まれる ConfigMap を作成し、これを **openshift-console** namespace の **console** Deployment に、指定した場所と同じ場所にマウントする必要があります。

4.25. OPERATOR LIFECYCLE MANAGER の設定



重要

Operator Framework 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。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポートについての詳細は、https://access.redhat.com/support/offerings/techpreview/ を参照してください。

テクノロジープレビューの Operator Framework には Operator Lifecycle Manager (OLM) が含まれます。以下の変数をインベントリーファイルに設定して、クラスターインストール時に OLM をオプションでインストールできます。

注記

または、テクノロジープレビューの Operator Framework をクラスターのインストール後にインストールできます。個別の方法については、Installing Operator Lifecycle Manager using Ansible を参照してください。

1. **[OSEv3:vars]** セクションに **openshift_enable_olm** 変数を追加し、これを **true** に設定します。

```
openshift_enable_olm=true
```

2. **[OSEv3:vars]** セクションに **openshift_additional_registry_credentials** 変数を追加し、Operator コンテナーのプルに必要な認証情報を設定します。

```
openshift_additional_registry_credentials=
[{'host':'registry.connect.redhat.com','user':'<your_user_name>','password':'<your_password>','test_image':'mongodb/enterprise-operator:0.3.2'}]
```

user および **password** を、Red Hat カスタマーポータル (https://access.redhat.com) へのログインに使用する認証情報に追加します。

test_image は、指定した認証情報をテストするために使用されるイメージを表します。

クラスターのインストールが正常に完了した後は、テクノロジープレビューフェーズで OLM をクラスター管理者として使用するための追加の手順については、Launching your first Operator を参照してください。

第5章 インベントリーファイルの例

5.1. 概要

独自のインベントリーファイルの設定 の基本を理解したら、高可用性のために 複数マスターを使用 することを含め、各種の環境トポロジーを記述する以下のインベントリーサンプルを確認できます。要件に一致するサンプルを選択し、これを環境に合わせて変更し、インストールの実行 時にインベントリーファイルとして使用できます。

重要

以下のインベントリーのサンプルでは、**[nodes]** グループにホストごとに **openshift_node_group_name** を設定する際にノードグループのデフォルトセットを使用します。独自のカスタムノードグループの定義を定義し、使用するには、インベントリーファイルに **openshift_node_groups** 変数も設定する必要があります。詳細は、Defining Node Groups and Host Mappings を参照してください。

5.2. 単一マスターの例

単一マスターと複数ノード、単一または複数の etcd ホストを含む環境を設定できます。

注記

インストール後の単一マスタークラスターから複数マスターへの移行はサポートされていません。

5.2.1. 単一マスター、単一 etcd および複数ノード

以下の表は、単一マスター (同じホストに静的 Pod として実行されている単一 etcd インスタンスがある)、ユーザーアプリケーションをホストする 2 つのノード、**専用インフラストラクチャー** をホストする node-role.kubernetes.io/infra=true ラベル付きの 2 つのノードの環境の例を示しています。

ホスト名	インストールするインフラストラクチャー/ロール
master.example.com	マスター、etcd、ノード
node1.example.com	コンピュートノード
node2.example.com	
infra-node1.example.com	インフラストラクチャーノード
infra-node2.example.com	

これらのサンプルホストは、以下のサンプルインベントリーファイルの **[masters]**、**[etcd]**、および **[nodes]** セクションに記載されています。

単一マスター、単一 etcd、および複数ノードのインベントリーファイル

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
```

```
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a password
ansible_ssh_user=root

# If ansible_ssh_user is not root, ansible_become must be set to true
#ansible_become=true

openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
master.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

ノードホストラベルの設定 を参照し、OpenShift Container Platform 3.9 以降のデフォルトノードセレクター要件とノードラベルに関する考慮事項を確認してください。

この例を使用するには、お使いの環境と仕様に合わせてファイルを変更し、これを **/etc/ansible/hosts** として保存します。

5.2.2. 単一マスター、複数 etcd、および複数ノード

以下の表は、単一マスター、3 つの etcd ホスト、ユーザーアプリケーションをホストする 2 つのノード、**専用インフラストラクチャー** をホストする node-role.kubernetes.io/infra=true ラベル付きの 2 つのノードの環境の例を示しています。

ホスト名	インストールするインフラストラクチャー/ロール
master.example.com	マスターおよびノード

ホスト名	インストールするインフラストラクチャー/ロール
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	コンピュートノード
node2.example.com	
infra-node1.example.com	専用インフラストラクチャーノード
infra-node2.example.com	

これらのサンプルホストは、以下のサンプルインベントリーファイルの **[masters]**、**[nodes]**、および **[etcd]** セクションに記載されています。

単一マスター、複数 etcd、および複数ノードのインベントリーファイル

```
# Create an OSEv3 group that contains the masters, nodes, and etcd groups
[OSEv3:children]
masters
nodes
etcd

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
master.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# host group for nodes, includes region info
[nodes]
master.example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
```

```
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

ノードホストラベルの設定 を参照し、OpenShift Container Platform 3.9 以降のデフォルトノードセレクター要件とノードラベルに関する考慮事項を確認してください。

この例を使用するには、お使いの環境と仕様に合わせてファイルを変更し、これを **/etc/ansible/hosts** として保存します。

5.3. 複数マスターの例

複数マスター、複数 etcd ホスト、複数ノードを含む環境を設定できます。高可用性 (HA) 対応複数マスター を設定すると、クラスターに単一障害点が設定されないようにすることができます。

注記

インストール後の単一マスタークラスターから複数マスターへの移行はサポートされていません。

複数マスターを設定する際には、クラスターインストールプロセスで **ネイティブ** 高可用性 (HA) メソッドがサポートされます。この方法は、OpenShift Container Platform に組み込まれているネイティブ HA マスター機能を活用するもので、 任意のロードバランシングソリューションと組み合わせことができます。

ホストがインベントリーファイルの **[lb]** セクションに定義されている場合、Ansible はロードバランシングソリューションとして HAProxy を自動的にインストールし、設定します。ホストが定義されていない場合、ユーザーが選択した外部のロードバランシングソリューションを事前に定義しており、マスター API (ポート 8443) をすべてのマスターホストで分散することが想定されます。

注記

この HAProxy ロードバランサーは、API サーバーの HA モードを実証することを意図したものであり、実稼働環境での使用には推奨されません。クラウドプロバイダーにデプロイする場合は、クラウドネイティブの TCP ベースのロードバランサーをデプロイするか、または高可用性ロードバランサーを提供するための他の手順を実行することを推奨します。

HAProxy ロードバランサーは、トラフィックを API サーバーへの負荷分散のためにのみ使用され、ユーザーアプリケーショントラフィックを負荷分散しません。

外部のロードバランシングソリューションを使用する場合は、以下が必要になります。

- SSL パススルー対応に設定された、事前に作成されたロードバランサーの仮想 IP (VIP)
- **openshift_master_api_port** 値 (デフォルトは 8443) で指定されたポートでリッスンし、そのポートですべてのマスターホストにプロキシー送信する VIP。
- DNS に登録されている VIP のドメイン名。

- このドメイン名は、OpenShift Container Platform インストーラーで **openshift_master_cluster_public_hostname** と **openshift_master_cluster_hostname** の両方の値になります。

詳細については、External Load Balancer Integrations example in Github を参照してください。高可用性マスターアーキテクチャーの詳細については、Kubernetes Infrastructure を参照してください。

注記

現時点で、クラスターインストールプロセスはアクティブ/パッシブ設定の複数の HAProxy ロードバランサーをサポートしていません。インストール後の修正については、ロードバランサー管理ドキュメント を参照してください。

複数マスターを設定するには、複数 etcd を持つ複数マスター を参照してください。

5.3.1. 外部のクラスター化された etcd を含む、ネイティブ HA を使用した複数マスター

以下の表は、 ネイティブ HA 方法を使用する 3 つの **マスター**、1 つの HAProxy ロードバランサー、3 つの etcd ホスト、ユーザーアプリケーションをホストする 2 つの ノード、**専用インフラストラクチャー** をホストする node-role.kubernetes.io/infra=true ラベル付きの 2 つのノードの環境の例を示しています。

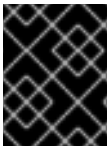
ホスト名	インストールするインフラストラクチャー/ロール
master1.example.com	マスター (クラスター化、ネイティブ HA を使用) およびノード
master2.example.com	
master3.example.com	
lb.example.com	API マスターエンドポイントの負荷分散のみの HAProxy
etcd1.example.com	etcd
etcd2.example.com	
etcd3.example.com	
node1.example.com	コンピュートノード
node2.example.com	
infra-node1.example.com	専用インフラストラクチャーノード
infra-node2.example.com	

これらのサンプルホストは、以下のサンプルインベントリーファイルの **[masters]**、**[etcd]**、**[lb]** および **[nodes]** セクションに記載されています。

HAProxy インベントリーファイルを使用する複数マスター

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# Native high availbility cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# apply updated node defaults
openshift_node_groups=[{'name': 'node-config-all-in-one', 'labels': ['node-
role.kubernetes.io/master=true', 'node-role.kubernetes.io/infra=true', 'node-
role.kubernetes.io/compute=true'], 'edits':}]

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
```

```
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```



重要

ノードホストラベルの設定 を参照し、OpenShift Container Platform 3.9 以降のデフォルトノードセレクター要件とノードラベルに関する考慮事項を確認してください。

この例を使用するには、お使いの環境と仕様に合わせてファイルを変更し、これを **/etc/ansible/hosts** として保存します。

5.3.2. 同一の場所に配置されたクラスター化された etcd を含む、ネイティブ HA を使用した複数マスター

以下の表は、ネイティブ HA 方法を使用する 3 つの **マスター** (各ホストに静的 Pod として実行される etcd がある)、1 つの HAProxy ロードバランサー、ユーザーアプリケーションをホストする 2 つの ノード、**専用インフラストラクチャー** をホストする node-role.kubernetes.io/infra=true ラベル付きの 2 つのノードの環境の例を示しています。

ホスト名	インストールするインフラストラクチャー/ロール
master1.example.com	各ホストに静的 Pod として実行されている etcd があるマスター (ネイティブ HA を使用するクラスター化されたマスター) とノード
master2.example.com	
master3.example.com	
lb.example.com	API マスターエンドポイントの負荷分散のみの HAProxy
node1.example.com	コンピュートノード
node2.example.com	
infra-node1.example.com	専用インフラストラクチャーノード
infra-node2.example.com	

これらのサンプルホストは、以下のサンプルインベントリーファイルの **[masters]**、**[etcd]**、**[lb]** および **[nodes]** セクションに記載されています。

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb
```

```
# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind':
'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
master1.example.com
master2.example.com
master3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master'
node1.example.com openshift_node_group_name='node-config-compute'
node2.example.com openshift_node_group_name='node-config-compute'
infra-node1.example.com openshift_node_group_name='node-config-infra'
infra-node2.example.com openshift_node_group_name='node-config-infra'
```

重要

ノードホストラベルの設定 を参照し、OpenShift Container Platform 3.9 以降のデフォルトノードセレクター要件とノードラベルに関する考慮事項を確認してください。

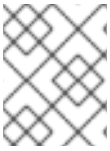
この例を使用するには、お使いの環境と仕様に合わせてファイルを変更し、これを **/etc/ansible/hosts** として保存します。

第6章 OPENSHIFT CONTAINER PLATFORM のインストール

OpenShift Container Platform クラスターをインストールするには、一連の Ansible Playbook を実行します。

重要

Ansible Playbook を **--tags** または **--check** オプションを使用して実行することを、Red Hat ではサポートしていません。



注記

OpenShift Container Platform をスタンドアロンレジストリーとして インストールするには、スタンドアロンレジストリーのインストール を参照してください。

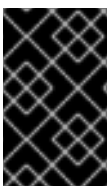
6.1. 前提条件

OpenShift Container Platform をインストールする前に、クラスターホストを準備します。

- システムおよび環境の要件 を確認します。
- クラスターが大規模な場合は、インストールタイミングについての提案について、Scaling and Performance Guide を参照してください。
- ホストを準備します。このプロセスには、コンポーネントのタイプ別にシステムおよび環境の要件を確認すること、**docker** サービスをインストールして設定すること、および Ansible バージョン 2.6 以降をインストールすることが含まれます。インストール Playbook を実行するには、Ansible をインストールする必要があります。
- 環境および OpenShift Container Plat クラスター設定を定義するには、インベントリーファイルを設定 します。初期インストールおよび以降のクラスターアップグレードのいずれもこのインベントリーファイルに基づいて実行されます。
- OpenShift Container Platform を Red Hat Enterprise Linux にインストールしている場合は、RPM またはシステムコンテナー のインストール方式を使用するかどうかを決定します。システムコンテナー方式は RHEL Atomic Host システムに必要です。

6.1.1. RPM ベースのインストーラーの実行

RPM ベースのインストーラーは、RPM パッケージでインストールされた Ansible を使用し、ローカルホストで使用可能な Playbook と設定ファイルを実行します。



重要

OpenShift Ansible Playbook は **nohup** で実行しないでください。Playbook で **nohup** を使用すると、ファイル記述子が作成され、ファイルが閉じなくなります。その結果、システムでファイルをさらに開けなくなり、Playbook が失敗します。

RPM ベースのインストーラーを実行するには、以下の手順を実行します。

1. Playbook ディレクトリーに切り替え、**prerequisites.yml** Playbook を実行します。この Playbook は必要なソフトウェアパッケージをインストールし (ある場合)、コンテナーランタイムを変更します。コンテナーランタイムを設定する必要がない限り、この Playbook はクラスターの初回のデプロイ前に 1 度のみ実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
  playbooks/prerequisites.yml
```

1 インベントリーファイルが **/etc/ansible/hosts** ディレクトリーにない場合、**-i** およびインベントリーファイルのパスを指定します。

2. Playbook ディレクトリーに切り替え、**deploy_cluster.yml** Playbook を実行してクラスターインストールを開始します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook [-i /path/to/inventory] \ 1
    playbooks/deploy_cluster.yml
```

1 インベントリーファイルが **/etc/ansible/hosts** ディレクトリーにない場合、**-i** およびインベントリーファイルのパスを指定します。

. **インストールが正常に完了した場合は、インストールを確認します。** インストールが失敗した場合は、インストールを再試行します。

6.1.2. コンテナー化インストーラーの実行

openshift3/ose-ansible イメージは、 OpenShift Container Platform インストーラーのコンテナー化バージョンです。このインストーラーイメージは、RPM ベースのインストーラーと同じ機能を提供しますが、ホストに直接インストールされるのではなく、そのすべての依存関係を提供するコンテナー化環境で実行されます。この使用にあたっての唯一の要件は、コンテナーを実行できることになります。

6.1.2.1. インストーラーをシステムコンテナーとして実行する

インストーラーイメージは、システムコンテナー として使用できます。システムコンテナーは、従来の **docker** サービスの外部に保存して実行できます。これにより、ホストでのインストールによって **docker** が再起動されることを心配することなく、ターゲットホストのいずれかからインストーラーイメージを実行することが可能になります。

Atomic CLI を使用してインストーラーを 1 回だけ実行されるシステムコンテナーとして実行するには、以下の手順を **root** ユーザーとして実行します。

1. **prerequisites.yml** Playbook を実行します。

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml \
    --set OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 ローカルホスト上にインベントリーファイルの場所を指定します。

このコマンドは、指定されるインベントリーファイルと **root** ユーザーの SSH 設定を使用して一連の前提条件タスクを実行します。

2. **deploy_cluster.yml** Playbook を実行します。

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \ 1
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
\
    --set OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 ローカルホスト上にインベントリーファイルの場所を指定します。

このコマンドは、指定されるインベントリーファイルと **root** ユーザーの SSH 設定を使用してクラスターインストールを開始します。出力のログを端末に記録し、さらに **/var/log/ansible.log** ファイルに保存します。このコマンドの初回実行時に、イメージは OSTree ストレージ (システムコンテナーは **docker** デーモンストレージではなくこのストレージを使用します) にインポートされます。後続の実行では、保存されたイメージが再利用されます。

何らかの理由でインストールが失敗した場合は、インストーラーを再実行する前に 既知の問題 に目を通し、特定の指示や回避策がないかどうか確認してください。

6.1.2.2. その他の Playbook の実行

PLAYBOOK_FILE 環境変数を使用すると、コンテナー化インストーラーで実行するその他の Playbook を指定できます。**PLAYBOOK_FILE** のデフォルト値は、メインのクラスターインストール Playbook である **/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml** ですが、これをコンテナー内の別の Playbook のパスに設定できます。

たとえば、インストールの前に プレインストールチェック Playbook を実行するには、以下のコマンドを使用します。

```
# atomic install --system \
    --storage=ostree \
    --set INVENTORY_FILE=/path/to/inventory \
    --set PLAYBOOK_FILE=/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-
install.yml \ 1
    --set OPTS="-v" \ 2
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 **PLAYBOOK_FILE** を **playbooks/** ディレクトリーで始まる Playbook のフルパスに設定します。Playbook は、RPM ベースのインストーラーと同じ場所にあります。

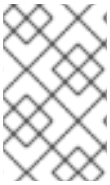
2 **OPTS** を設定して、コマンドラインオプションを **ansible-playbook** に追加します。

6.1.2.3. インストーラーをコンテナーとして実行する

インストーラーイメージは、**docker** が実行できる任意の場所で **docker** コンテナーとして実行することもできます。

警告

この方法は、設定されているホストのいずれかでインストーラーを実行するために使用しないでください。 インストーラーによってホストで **docker** が再起動され、インストールの実行が中断する可能性があるためです。



注記

この方法と上記のシステムコンテナー方法は同じイメージを使用しますが、それぞれ異なるエントリーポイントとコンテキストで実行されます。 そのため、ランタイムパラメーターは同じではありません。

インストーラーを **docker** コンテナーとして実行する場合は、少なくとも以下を指定する必要があります。

- SSH キー (Ansible がホストにアクセスできるようにするため)。
- Ansible インベントリーファイル。
- そのインベントリーに対して実行する Ansible Playbook の場所。

次に、**docker** 経由でインストールを実行する方法の例を示します。 これは、**docker** へのアクセス権限を持つ非 **root** ユーザーとして実行する必要があります。

1. まず、**prerequisites.yml** Playbook を実行します。

```
$ docker run -t -u `id -u` \ 1
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \ 2
    -v $HOME/ansible/hosts:/tmp/inventory:Z \ 3
    -e INVENTORY_FILE=/tmp/inventory \ 4
    -e PLAYBOOK_FILE=playbooks/prerequisites.yml \ 5
    -e OPTS="-v" \ 6
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

1 **-u `id -u`** は、コンテナーが現在のユーザーと同じ UID で実行されるようにします。これにより、そのユーザーがコンテナー内の SSH キーを使用できるようになります (SSH プライベートキーは所有者のみが判読できることが予想されます)。

2 **-v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z** は、SSH キー (**$HOME/.ssh/id_rsa**) をコンテナーユーザーの **$HOME/.ssh** ディレクトリーにマウントします。**/opt/app-root/src** は、コンテナー内のユーザーの **$HOME** です。SSH キーを別の場所にマウントする場合は、**-e ANSIBLE_PRIVATE_KEY_FILE=/the/mount/point** で環境変数を追加するか、**ansible_ssh_private_key_file=/the/mount/point** をインベントリーの変数として設定して、Ansible を Ansible を参照するようにインベントリーで変数としてポイントします。SSH キーは **:Z** フラグでマウントされることに注意してください。このフラグは必須で、コンテナーが、制限された SELinux コンテキストの SSH キーを読み込むことができます。つまり、元の SSH キーファイルは **system_u:object_r:container_file_t:s0:c113,c247** などのように再度ラベル付けされます。**:Z** についての詳細は、**docker-run(1)** の man ページを参照してください。予期しない結果が発生する可能性があります。たとえば **$HOME/.ssh** ディレクトリー全体をマウント

(再ラベル付け) すると、そのホストの **sshd** が、ログインする公開鍵へのアクセスをブロックします。このため、元のファイルラベルを変更しなくてもすむように SSH キーまたはディレクトリーの別のコピーを使用することをお勧めします。

3 4 **-v $HOME/ansible/hosts:/tmp/inventory:Z** と **-e INVENTORY_FILE=/tmp/inventory** は、静的 Ansible インベントリーファイルを **/tmp/inventory** としてコンテナーにマウントし、これを参照する対応する環境変数を設定します。SSH キーと同様に、既存のラベルによっては、コンテナー内を読み取れるように、**:Z** フラグを使用してインベントリーファイルの SELinux ラベルを変更しなければならない場合があります。ユーザーの **$HOME** ディレクトリー内のファイルの場合、これが必要になる可能性があります。そのため、この場合もまた、マウント前にインベントリーを専用の場所にコピーすることをお勧めします。インベントリーファイルは、**INVENTORY_URL** 環境変数を指定した場合には、Web サーバーからダウンロードすることもできます。 または **DYNAMIC_SCRIPT_URL** パラメーターを使用して、動的なインベントリーを提供する実行可能スクリプトを指定することにより動的に生成することもできます。

5 **-e PLAYBOOK_FILE=playbooks/prerequisites.yml** は、**openshift-ansible** コンテンツのトップレベルのディレクトリーからの相対パスとして実行する Playbook を指定します。この例では、前提条件の Playbook を指定します。また、RPM からの完全パスや、コンテナー内の他の Playbook ファイルへのパスを指定できます。

6 **-e OPTS="-v"** は、コンテナーで実行される **ansible-playbook** コマンドに任意のコマンドラインオプションを提供します。この例では、**-v** を指定して詳細度を上げることができます。

2. 次に、**deploy_cluster.yml** playbook を実行してクラスターインストールを開始します。

```
$ docker run -t -u `id -u` \
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z \
    -v $HOME/ansible/hosts:/tmp/inventory:Z \
    -e INVENTORY_FILE=/tmp/inventory \
    -e PLAYBOOK_FILE=playbooks/deploy_cluster.yml \
    -e OPTS="-v" \
    registry.redhat.io/openshift3/ose-ansible:v3.11
```

6.1.2.4. OpenStack インストール Playbook の実行

OpenShift Container Platform を既存の OpenStack インストールにインストールするには、OpenStack Playbook を使用します。詳細の前提条件を含む Playbook についての詳細は、OpenStack Provisioning readme ファイル を参照してください。

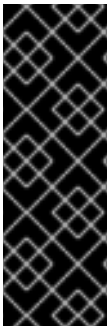
Playbook を実行するには、以下のコマンドを実行します。

```
$ ansible-playbook --user openshift \
  -i openshift-ansible/playbooks/openstack/inventory.py \
  -i inventory \
  openshift-ansible/playbooks/openstack/openshift-cluster/provision_install.yml
```

6.1.3. インストール Playbook について

インストーラーはモジュール化された Playbook を使用します。そのため、管理者は必要に応じて特定のコンポーネントをインストールできます。ロールと Playbook を分けることで、アドホックな管理タスクをより適切にターゲット設定できます。 その結果、インストール時の制御レベルが強化され、時間の節約が可能になります。

メインのインストール Playbook である **/usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml**は、一連の個別コンポーネント Playbook を特定の順序で実行します。実行の最後に、インストーラーから完了したフェーズが報告されます。 インストールが失敗した場合は、そのフェーズが失敗したかについて Ansible の実行エラーと共に画面に表示されます。



重要

RHEL Atomic Host は OpenShift Container Platform サービスをシステムコンテナーとして実行するためにサポートされていますが、このインストール方式では RHEL Atomic Host で利用できない Ansible を使用します。そのため、RPM ベースのインストーラーは RHEL 7 システムから実行される必要があります。インストールを開始するホストは OpenShift Container Platform クラスターに組み込まれる必要はありませんが、組み込みは可能です。または、インストーラーのコンテナー化バージョン を、RHEL Atomic Host システムから実行できるシステムコンテナーとして使用することもできます。

6.2. インストールの再試行

Ansible インストーラーが失敗する場合でも、OpenShift Container Platform をインストールできます。

1. 既知の問題 を確認し、特定の指示または回避策を確認します。
2. インストールのエラーに対応します。
3. アンインストール、再インストール、またはインストール再試行が必要かどうかを判断します。
   - SDN 設定を変更しておらず、新規証明書を生成する場合は、インストールを再試行します。
   - SDN 設定を変更し、新規証明書を生成している場合、またはインストーラーが再び失敗する場合、クリーンなオペレーティングシステムインストールで起動し直すか、または アンインストール し、再度インストールする必要があります。
   - 仮想マシンを使用する場合、新規イメージから起動するか、または アンインストール を実行して再度インストールします。
   - ベアメタルマシンを使用する場合、再度 アンインストール およびインストールを実行します。
4. インストールを再試行します。
   - **deploy_cluster.yml** Playbook を再び実行できます。
   - 残りのそれぞれのインストール Playbook を実行できます。
     残りの Playbook のみを実行する必要がある場合、失敗したフェーズの Playbook から実行し、その後に残りの Playbook を順番に実行して開始します。以下のコマンドでそれぞれの Playbook を実行します。

     ```
     # ansible-playbook [-i /path/to/inventory] <playbook_file_location>
     ```

     以下の表は、Playbook が実行される順序で Playbook を一覧表示しています。

     表6.1 個別コンポーネント Playbook の実行順序

Playbook 名	ファイルの場所
Health Check	**/usr/share/ansible/openshift-ansible/playbooks/openshift-checks/pre-install.yml**
Node Bootstrap	**/usr/share/ansible/openshift-ansible/playbooks/openshift-node/bootstrap.yml**
etcd Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-etcd/config.yml**
NFS Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-nfs/config.yml**
Load Balancer Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-loadbalancer/config.yml**
Master Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-master/config.yml**
Master Additional Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-master/additional_config.yml**
Node Join	**/usr/share/ansible/openshift-ansible/playbooks/openshift-node/join.yml**
GlusterFS Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-glusterfs/config.yml**
Hosted Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-hosted/config.yml**
Monitoring Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-monitoring/config.yml**
Web Console Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-web-console/config.yml**
Admin Console Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-console/config.yml**
Metrics Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml**
metrics-server	**/usr/share/ansible/openshift-ansible/playbooks/metrics-server/config.yml**
Logging Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-logging/config.yml**

Playbook 名	ファイルの場所
Availability Monitoring Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-monitor-availability/config.yml**
Service Catalog Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-service-catalog/config.yml**
Management Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-management/config.yml**
Descheduler Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-descheduler/config.yml**
Node Problem Detector Install	**/usr/share/ansible/openshift-ansible/playbooks/openshift-node-problem-detector/config.yml**
Operator Lifecycle Manager (OLM) Install (テクノロジープレビュー)	**/usr/share/ansible/openshift-ansible/playbooks/olm/config.yml**

6.3. インストールの検証

インストールが完了したら、次の手順を実行します。

1. マスターが起動しており、ノードが登録されており、**Ready** ステータスで報告されていることを確認します。**マスターホストで** 以下のコマンドを root で実行します。

```
# oc get nodes
NAME                STATUS    ROLES     AGE       VERSION
master.example.com  Ready     master    7h        v1.9.1+a0ce1bc657
node1.example.com   Ready     compute   7h        v1.9.1+a0ce1bc657
node2.example.com   Ready     compute   7h        v1.9.1+a0ce1bc657
```

2. Web コンソールが正常にインストールされているか確認するには、マスターホスト名と Web コンソールのポート番号を使用して Web ブラウザーで Web コンソールにアクセスします。たとえば、ホスト名が **master.openshift.com** で、デフォルトポート **8443** を使用するマスターホストの場合、Web コンソール URL は **https://master.openshift.com:8443/console** になります。

複数 etcd ホストの確認
複数 etcd ホストをインストールした場合は、以下の手順を実行します。

1. まず、 **etcdctl** コマンドを提供する **etcd** パッケージがインストールされていることを確認します。

```
# yum install etcd
```

2. マスターホストで etcd クラスターの正常性を確認します。 以下で実際の etcd ホストの FQDN の置き換えを実行します。

```
# etcdctl -C \

https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:237
9 \
    --ca-file=/etc/origin/master/master.etcd-ca.crt \
    --cert-file=/etc/origin/master/master.etcd-client.crt \
    --key-file=/etc/origin/master/master.etcd-client.key cluster-health
```

3. メンバーリストが正しいことも確認します。

```
# etcdctl -C \

https://etcd1.example.com:2379,https://etcd2.example.com:2379,https://etcd3.example.com:237
9 \
    --ca-file=/etc/origin/master/master.etcd-ca.crt \
    --cert-file=/etc/origin/master/master.etcd-client.crt \
    --key-file=/etc/origin/master/master.etcd-client.key member list
```

HAProxy を使用する複数マスターの確認
HAProxy をロードバランサーとして使用して複数のマスターをインストールしている場合、以下の URL を開き、HAProxy のステータスを確認します。

```
http://<lb_hostname>:9000 (1)
```

(1) インベントリーファイルの **[lb]** セクションに一覧表示されているロードバランサーのホスト名を指定します。

HAProxy の設定に関するドキュメント を参照してインストールを検証できます。

6.4. ビルドのオプションでのセキュリティー保護

docker build の実行は特権付きのプロセスのため、コンテナーにはマルチテナント環境で許可される以上のノードに対するアクセスがある場合があります。ユーザーを信頼しない場合、インストール後により多くのセキュアなオプションを使用できます。クラスターで Docker ビルドを無効にし、ユーザーに対してクラスター外でイメージをビルドするように要求できます。このオプションのプロセスについての詳細は、Securing Builds by Strategy を参照してください。

6.5. 既知の問題

- 複数マスタークラスターでフェイルオーバーが発生すると、コントローラーマネージャーの過剰修正が生じ、結果として予定よりも多くの pod がシステムで実行される可能性があります。ただし、これは一時的なイベントであり、後にシステムによって修正されます。詳細については、https://github.com/kubernetes/kubernetes/issues/10030 を参照してください。
- 既知の問題により、インストールの実行後、NFS ボリュームがいずれかのコンポーネント用にプロビジョニングされている場合、それらのコンポーネントが NFS ボリュームにデプロイされるかどうかにかかわらず、以下のディレクトリーが作成される可能性があります。
  - **/exports/logging-es**
  - **/exports/logging-es-ops/**
  - **/exports/metrics/**

- /exports/prometheus
- /exports/prometheus-alertbuffer/
- /exports/prometheus-alertmanager/
  インストール後にこれらのディレクトリーを随時削除することができます。

6.6. 次のステップ

これで OpenShift Container Platform インスタンスが機能し、以下を実行できるようになります。

- 統合コンテナーイメージレジストリー をデプロイします。
- ルーター をデプロイします。

第7章 非接続インストール

データセンターの一部が、プロキシーサーバー経由でもインターネットにアクセスできないことがよくあります。このような環境でも OpenShift Container Platform をインストールできますが、必要なソフトウェアおよびイメージをダウンロードし、これらを非接続環境で利用できる状態にする必要があります。

インストールコンポーネントがノードホストで利用可能な状態で、標準的なインストール手順に従って OpenShift Container Platform をインストールします。

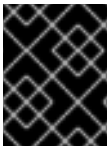
OpenShift Container Platform をインストールしたら、プルした S2I ビルダーイメージをクラスターで利用可能にする必要があります。

7.1. 前提条件

- OpenShift Container Platform のアーキテクチャーの概要 を確認し、環境トポロジーについて計画します。
- root アクセスが可能な、インターネットにアクセスでき、110 GB 以上のディスク領域を持つ Red Hat Enterprise Linux (RHEL) 7 サーバーを取得します。このコンピューターに必要なソフトウェアリポジトリーおよびコンテナーイメージをダウンロードします。
- ミラーリングされたリポジトリーを提供するために Web サーバーを非接続環境で維持できるように計画します。インターネットに接続されたホストからこの Web サーバーに対して、ネットワーク経由か、または非接続デプロイメントで物理メディアを使用してリポジトリーをコピーします。
- ソースコントロールリポジトリーを指定します。インストール後に、ノードは Git などのソースコードリポジトリーのソースコードにアクセスする必要があります。
  OpenShift Container Platform でアプリケーションをビルドする場合、ビルドに Maven リポジトリーや Ruby アプリケーション用の Gem ファイルなどの外部の依存関係が含まれる可能性があります。
- 非接続環境内にレジストリーを指定します。オプションには以下が含まれます。
  - スタンドアロン OpenShift Container Platform レジストリー のインストール
  - コンテナーイメージレジストリーとして動作する Red Hat Satellite 6.1 サーバーの使用

7.2. 必要なソフトウェアパッケージおよびイメージの取得

OpenShift Container Platform を非接続環境にインストールする前に、必要なイメージおよびコンポーネントを取得し、それらをリポジトリーに保存します。



重要

非接続環境のクラスターと同じアーキテクチャーを持つシステムで必要なイメージおよびソフトウェアコンポーネントを取得する必要があります。

7.2.1. OpenShift Container Platform パッケージの取得

インターネット接続のある RHEL 7 サーバーで、リポジトリーを同期します。

1. リポジトリーの同期後にパッケージが削除されないように GPG キーをインポートします。
   -

```
$ rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. サーバーを Red Hat カスタマーポータルに登録します。OpenShift Container Platform サブスクリプションにアクセスできるアカウントに関連付けられている認証情報を使用する必要があります。

```
$ subscription-manager register
```

3. RHSM から最新のサブスクリプションデータをプルします。

```
$ subscription-manager refresh
```

4. OpenShift Container Platform チャンネルを提供するサブスクリプションをアタッチします。

   a. OpenShift Container Platform チャンネルを提供する利用可能なサブスクリプションプールを検索します。

```
$ subscription-manager list --available --matches '*OpenShift*'
```

   b. OpenShift Container Platform を提供するサブスクリプションのプール ID をアタッチします。

```
$ subscription-manager attach --pool=<pool_id>
$ subscription-manager repos --disable="*"
```

5. OpenShift Container Platform 3.11 で必要なリポジトリーのみを有効にします。

   - x86_64 サーバーでのクラウドインストールおよびオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.11-rpms" \
    --enable="rhel-7-server-ansible-2.9-rpms"
```

   - IBM POWER8 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-for-power-le-rpms" \
    --enable="rhel-7-for-power-le-extras-rpms" \
    --enable="rhel-7-for-power-le-optional-rpms" \
    --enable="rhel-7-server-ansible-2.9-for-power-le-rpms" \
    --enable="rhel-7-server-for-power-le-rhscl-rpms" \
    --enable="rhel-7-for-power-le-ose-3.11-rpms"
```

   - IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
# subscription-manager repos \
    --enable="rhel-7-for-power-9-rpms" \
    --enable="rhel-7-for-power-9-extras-rpms" \
    --enable="rhel-7-for-power-9-optional-rpms" \
```

```
    --enable="rhel-7-server-ansible-2.9-for-power-9-rpms" \
    --enable="rhel-7-server-for-power-9-rhscl-rpms" \
    --enable="rhel-7-for-power-9-ose-3.11-rpms"
```

注記

以前のバージョンの OpenShift Container Platform 3.11 は Ansible 2.6 のみをサポートしていました。最新バージョンの Playbook が Ansible 2.9 に対応するようになりました。Ansible 2.9 は、使用する推奨バージョンです。

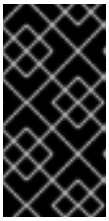
6. 必要なパッケージをインストールします。

```
$ sudo yum -y install yum-utils createrepo docker git
```

yum-utils パッケージは **reposync** ユーティリティーを提供します。 これによって yum リポジトリーをミラーリングでき、**createrepo** で使用可能な **yum** リポジトリーをディレクトリーから作成できます。

7. ソフトウェアを保存するディレクトリーをサーバーのストレージまたは、USB ドライブまたは他の外部デバイスに作成します。

```
$ mkdir -p </path/to/repos>
```



重要

このサーバーを接続されていない LAN に再接続し、これをリポジトリーサーバーとして使用する場合、ファイルをローカルに保存します。ローカルに保存できない場合は、USB で接続されたストレージを使用し、ソフトウェアを接続されていない LAN のリポジトリーサーバーに移動できるようにします。

8. パッケージを同期し、各パッケージのリポジトリーを作成します。

- x86_64 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ for repo in \
  rhel-7-server-rpms \
  rhel-7-server-extras-rpms \
  rhel-7-server-ansible-2.9-rpms \
  rhel-7-server-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=</path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1) (2) 作成したディレクトリーのパスを指定します。

- IBM POWER8 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ for repo in \
  rhel-7-for-power-le-rpms \
  rhel-7-for-power-le-extras-rpms \
```

```
  rhel-7-for-power-le-optional-rpms \
  rhel-7-server-ansible-2.9-for-power-le-rpms \
  rhel-7-server-for-power-le-rhscl-rpms \
  rhel-7-for-power-le-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=</path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1)(2) 作成したディレクトリーのパスを指定します。

- IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ for repo in \
  rhel-7-for-power-9-rpms \
  rhel-7-for-power-9-extras-rpms \
  rhel-7-for-power-9-optional-rpms \
  rhel-7-server-ansible-2.9-for-power-9-rpms \
  rhel-7-server-for-power-9-rhscl-rpms \
  rhel-7-for-power-9-ose-3.11-rpms
do
  reposync --gpgcheck -lm --repoid=${repo} --download_path=/<path/to/repos> (1)
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo} (2)
done
```

(1)(2) 作成したディレクトリーのパスを指定します。

7.2.2. イメージの取得

必要なコンテナーイメージをプルします。

1. Docker デーモンを起動します。

```
$ systemctl start docker
```

2. 必要な OpenShift Container Platform インフラストラクチャーコンポーネントイメージすべてをプルします。**<tag>** をインストールするバージョンに置き換えます。たとえば、最新バージョンの **v3.11.634** を指定します。別のマイナーバージョンを指定することもできます。コンテナー化されたインストーラーを使用する場合は、これらの必要なイメージに加えて **registry.redhat.io/openshift3/ose-ansible:v3.11** をプルします。

```
$ docker pull registry.redhat.io/openshift3/apb-base:<tag>
$ docker pull registry.redhat.io/openshift3/apb-tools:<tag>
$ docker pull registry.redhat.io/openshift3/automation-broker-apb:<tag>
$ docker pull registry.redhat.io/openshift3/csi-attacher:<tag>
$ docker pull registry.redhat.io/openshift3/csi-driver-registrar:<tag>
$ docker pull registry.redhat.io/openshift3/csi-livenessprobe:<tag>
$ docker pull registry.redhat.io/openshift3/csi-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/grafana:<tag>
$ docker pull registry.redhat.io/openshift3/kuryr-controller:<tag>
$ docker pull registry.redhat.io/openshift3/kuryr-cni:<tag>
$ docker pull registry.redhat.io/openshift3/local-storage-provisioner:<tag>
```

```
$ docker pull registry.redhat.io/openshift3/manila-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/mariadb-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki:<tag>
$ docker pull registry.redhat.io/openshift3/mediawiki-apb:<tag>
$ docker pull registry.redhat.io/openshift3/mysql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ansible-service-broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cli:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-autoscaler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-capacity:<tag>
$ docker pull registry.redhat.io/openshift3/ose-cluster-monitoring-operator:<tag>
$ docker pull registry.redhat.io/openshift3/ose-console:<tag>
$ docker pull registry.redhat.io/openshift3/ose-configmap-reloader:<tag>
$ docker pull registry.redhat.io/openshift3/ose-control-plane:<tag>
$ docker pull registry.redhat.io/openshift3/ose-deployer:<tag>
$ docker pull registry.redhat.io/openshift3/ose-descheduler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-builder:<tag>
$ docker pull registry.redhat.io/openshift3/ose-docker-registry:<tag>
$ docker pull registry.redhat.io/openshift3/ose-efs-provisioner:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-dns-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-http-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-egress-router:<tag>
$ docker pull registry.redhat.io/openshift3/ose-haproxy-router:<tag>
$ docker pull registry.redhat.io/openshift3/ose-hyperkube:<tag>
$ docker pull registry.redhat.io/openshift3/ose-hypershift:<tag>
$ docker pull registry.redhat.io/openshift3/ose-keepalived-ipfailover:<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-rbac-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-kube-state-metrics:<tag>
$ docker pull registry.redhat.io/openshift3/ose-metrics-server:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node:<tag>
$ docker pull registry.redhat.io/openshift3/ose-node-problem-detector:<tag>
$ docker pull registry.redhat.io/openshift3/ose-operator-lifecycle-manager:<tag>
$ docker pull registry.redhat.io/openshift3/ose-ovn-kubernetes:<tag>
$ docker pull registry.redhat.io/openshift3/ose-pod:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-config-reloader:<tag>
$ docker pull registry.redhat.io/openshift3/ose-prometheus-operator:<tag>
$ docker pull registry.redhat.io/openshift3/ose-recycler:<tag>
$ docker pull registry.redhat.io/openshift3/ose-service-catalog:<tag>
$ docker pull registry.redhat.io/openshift3/ose-template-service-broker:<tag>
$ docker pull registry.redhat.io/openshift3/ose-tests:<tag>
$ docker pull registry.redhat.io/openshift3/ose-web-console:<tag>
$ docker pull registry.redhat.io/openshift3/postgresql-apb:<tag>
$ docker pull registry.redhat.io/openshift3/registry-console:<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-controller:<tag>
$ docker pull registry.redhat.io/openshift3/snapshot-provisioner:<tag>
$ docker pull registry.redhat.io/rhel7/etcd:3.2.28
```

3. x86_64 サーバーのオンプレミスインストールの場合、以下のイメージをプルします。**<tag>** をインストールするバージョンに置き換えます。たとえば、最新バージョンの **v3.11.634** を指定します。別のマイナーバージョンを指定することもできます。

```
$ docker pull registry.redhat.io/openshift3/ose-efs-provisioner:<tag>
```

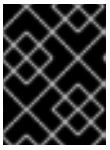
4. オプションのコンポーネントに必要な OpenShift Container Platform コンポーネントイメージすべてをプルします。**<tag>** をインストールするバージョンに置き換えます。たとえば、最新バージョンの **v3.11.634** を指定します。別のマイナーバージョンを指定することもできます。

- x86_64 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker pull registry.redhat.io/openshift3/metrics-cassandra:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-metrics:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-openshift-agent:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-heapster:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-schema-installer:<tag>
$ docker pull registry.redhat.io/openshift3/oauth-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-curator5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-elasticsearch5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-eventrouter:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-fluentd:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-kibana5:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alertmanager:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-node-exporter:<tag>
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-postgresql
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-memcached
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app-ui
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-app
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible
$ docker pull registry.redhat.io/cloudforms46/cfme-openshift-httpd
$ docker pull registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator
$ docker pull registry.redhat.io/rhgs3/rhgs-server-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-volmanager-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7
$ docker pull registry.redhat.io/rhgs3/rhgs-s3-server-rhel7
```

- IBM POWER8 または IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker pull registry.redhat.io/openshift3/metrics-cassandra:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-hawkular-openshift-agent:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-heapster:<tag>
$ docker pull registry.redhat.io/openshift3/metrics-schema-installer:<tag>
$ docker pull registry.redhat.io/openshift3/oauth-proxy:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-curator5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-elasticsearch5:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-eventrouter:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-fluentd:<tag>
$ docker pull registry.redhat.io/openshift3/ose-logging-kibana5:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alert-buffer:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-alertmanager:<tag>
$ docker pull registry.redhat.io/openshift3/prometheus-node-exporter:<tag>
```



重要

Red Hat サポートの場合、コンバージドモードのサブスクリプションが **rhgs3/** イメージに必要です。

5. OpenShift 環境で使用する Red Hat 認定の Source-to-Image (S2I) ビルダーイメージをプルします。

バージョン番号を指定して正しいタグを使用していることを確認します。イメージバージョンの互換性についての詳細は、OpenShift および Atomic プラットフォームのテスト済みの統合の S2I テーブルを参照してください。

以下のイメージをプルできます。

```
$ docker pull registry.redhat.io/jboss-amq-6/amq63-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datagrid-7/datagrid71-client-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-openshift:<tag>
$ docker pull registry.redhat.io/jboss-datavirt-6/datavirt63-driver-openshift:<tag>
$ docker pull registry.redhat.io/jboss-decisionserver-6/decisionserver64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-processserver-6/processserver64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-eap-6/eap64-openshift:<tag>
$ docker pull registry.redhat.io/jboss-eap-7/eap71-openshift:<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:<tag>
$ docker pull registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-2-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-maven-35-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-agent-nodejs-8-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-base-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-maven-rhel7:<tag>
$ docker pull registry.redhat.io/openshift3/jenkins-slave-nodejs-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/mongodb-32-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/mysql-57-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/perl-524-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/php-56-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/postgresql-95-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/python-35-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso70-openshift:<tag>
$ docker pull registry.redhat.io/rhscl/ruby-24-rhel7:<tag>
$ docker pull registry.redhat.io/redhat-openjdk-18/openjdk18-openshift:<tag>
$ docker pull registry.redhat.io/redhat-sso-7/sso71-openshift:<tag>
$ docker pull registry.redhat.io/rhscl/nodejs-6-rhel7:<tag>
$ docker pull registry.redhat.io/rhscl/mariadb-101-rhel7:<tag>
```

7.2.3. イメージのエクスポート

ご使用の環境に内部ネットワークへのアクセスがない場合で、コンテンツの移動に物理メディアが必要になる場合、イメージを圧縮されたファイルにエクスポートします。ホストがインターネットと内部ネットワークの両方に接続されている場合、以下の手順に従い、リポジトリーサーバーの準備および設定 に進みます。

1. 圧縮されたイメージを保存するディレクトリーを作成し、これに切り替えます。

```
$ mkdir </path/to/images>
$ cd </path/to/images>
```

2. OpenShift Container Platform インフラストラクチャーコンポーネントのイメージをエクスポートします。コンテナー化されたインストーラーを使用している場合は、これらの必要なイメージに加えて **registry.redhat.io/openshift3/ose-ansible:v3.11** をエクスポートします。

   - x86_64 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker save -o ose3-images.tar \
```

```
registry.redhat.io/openshift3/apb-base \
registry.redhat.io/openshift3/apb-tools \
registry.redhat.io/openshift3/automation-broker-apb \
registry.redhat.io/openshift3/csi-attacher \
registry.redhat.io/openshift3/csi-driver-registrar \
registry.redhat.io/openshift3/csi-livenessprobe \
registry.redhat.io/openshift3/csi-provisioner \
registry.redhat.io/openshift3/grafana \
registry.redhat.io/openshift3/kuryr-controller \
registry.redhat.io/openshift3/kuryr-cni \
registry.redhat.io/openshift3/local-storage-provisioner \
registry.redhat.io/openshift3/manila-provisioner \
registry.redhat.io/openshift3/mariadb-apb \
registry.redhat.io/openshift3/mediawiki \
registry.redhat.io/openshift3/mediawiki-apb \
registry.redhat.io/openshift3/mysql-apb \
registry.redhat.io/openshift3/ose-ansible-service-broker \
registry.redhat.io/openshift3/ose-cli \
registry.redhat.io/openshift3/ose-cluster-autoscaler \
registry.redhat.io/openshift3/ose-cluster-capacity \
registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
registry.redhat.io/openshift3/ose-console \
registry.redhat.io/openshift3/ose-configmap-reloader \
registry.redhat.io/openshift3/ose-control-plane \
registry.redhat.io/openshift3/ose-deployer \
registry.redhat.io/openshift3/ose-descheduler \
registry.redhat.io/openshift3/ose-docker-builder \
registry.redhat.io/openshift3/ose-docker-registry \
registry.redhat.io/openshift3/ose-efs-provisioner \
registry.redhat.io/openshift3/ose-egress-dns-proxy \
registry.redhat.io/openshift3/ose-egress-http-proxy \
registry.redhat.io/openshift3/ose-egress-router \
registry.redhat.io/openshift3/ose-haproxy-router \
registry.redhat.io/openshift3/ose-hyperkube \
registry.redhat.io/openshift3/ose-hypershift \
registry.redhat.io/openshift3/ose-keepalived-ipfailover \
registry.redhat.io/openshift3/ose-kube-rbac-proxy \
registry.redhat.io/openshift3/ose-kube-state-metrics \
registry.redhat.io/openshift3/ose-metrics-server \
registry.redhat.io/openshift3/ose-node \
registry.redhat.io/openshift3/ose-node-problem-detector \
registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
registry.redhat.io/openshift3/ose-ovn-kubernetes \
registry.redhat.io/openshift3/ose-pod \
registry.redhat.io/openshift3/ose-prometheus-config-reloader \
registry.redhat.io/openshift3/ose-prometheus-operator \
registry.redhat.io/openshift3/ose-recycler \
registry.redhat.io/openshift3/ose-service-catalog \
registry.redhat.io/openshift3/ose-template-service-broker \
registry.redhat.io/openshift3/ose-tests \
registry.redhat.io/openshift3/ose-web-console \
registry.redhat.io/openshift3/postgresql-apb \
registry.redhat.io/openshift3/registry-console \
registry.redhat.io/openshift3/snapshot-controller \
registry.redhat.io/openshift3/snapshot-provisioner \
registry.redhat.io/rhel7/etcd:3.2.28 \
```

- IBM POWER8 または IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker save -o ose3-images.tar \
    registry.redhat.io/openshift3/apb-base \
    registry.redhat.io/openshift3/apb-tools \
    registry.redhat.io/openshift3/automation-broker-apb \
    registry.redhat.io/openshift3/csi-attacher \
    registry.redhat.io/openshift3/csi-driver-registrar \
    registry.redhat.io/openshift3/csi-livenessprobe \
    registry.redhat.io/openshift3/csi-provisioner \
    registry.redhat.io/openshift3/grafana \
    registry.redhat.io/openshift3/kuryr-controller \
    registry.redhat.io/openshift3/kuryr-cni \
    registry.redhat.io/openshift3/local-storage-provisioner \
    registry.redhat.io/openshift3/manila-provisioner \
    registry.redhat.io/openshift3/mariadb-apb \
    registry.redhat.io/openshift3/mediawiki \
    registry.redhat.io/openshift3/mediawiki-apb \
    registry.redhat.io/openshift3/mysql-apb \
    registry.redhat.io/openshift3/ose-ansible-service-broker \
    registry.redhat.io/openshift3/ose-cli \
    registry.redhat.io/openshift3/ose-cluster-autoscaler \
    registry.redhat.io/openshift3/ose-cluster-capacity \
    registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
    registry.redhat.io/openshift3/ose-console \
    registry.redhat.io/openshift3/ose-configmap-reloader \
    registry.redhat.io/openshift3/ose-control-plane \
    registry.redhat.io/openshift3/ose-deployer \
    registry.redhat.io/openshift3/ose-descheduler \
    registry.redhat.io/openshift3/ose-docker-builder \
    registry.redhat.io/openshift3/ose-docker-registry \
    registry.redhat.io/openshift3/ose-egress-dns-proxy \
    registry.redhat.io/openshift3/ose-egress-http-proxy \
    registry.redhat.io/openshift3/ose-egress-router \
    registry.redhat.io/openshift3/ose-haproxy-router \
    registry.redhat.io/openshift3/ose-hyperkube \
    registry.redhat.io/openshift3/ose-hypershift \
    registry.redhat.io/openshift3/ose-keepalived-ipfailover \
    registry.redhat.io/openshift3/ose-kube-rbac-proxy \
    registry.redhat.io/openshift3/ose-kube-state-metrics \
    registry.redhat.io/openshift3/ose-metrics-server \
    registry.redhat.io/openshift3/ose-node \
    registry.redhat.io/openshift3/ose-node-problem-detector \
    registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
    registry.redhat.io/openshift3/ose-ovn-kubernetes \
    registry.redhat.io/openshift3/ose-pod \
    registry.redhat.io/openshift3/ose-prometheus-config-reloader \
    registry.redhat.io/openshift3/ose-prometheus-operator \
    registry.redhat.io/openshift3/ose-recycler \
    registry.redhat.io/openshift3/ose-service-catalog \
    registry.redhat.io/openshift3/ose-template-service-broker \
    registry.redhat.io/openshift3/ose-tests \
    registry.redhat.io/openshift3/ose-web-console \
    registry.redhat.io/openshift3/postgresql-apb \
    registry.redhat.io/openshift3/registry-console \
```

```
    registry.redhat.io/openshift3/snapshot-controller \
    registry.redhat.io/openshift3/snapshot-provisioner \
    registry.redhat.io/rhel7/etcd:3.2.28 \
```

3. オプションコンポーネントのイメージを同期している場合は、それらをエクスポートします。

- x86_64 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker save -o ose3-optional-imags.tar \
    registry.redhat.io/openshift3/metrics-cassandra \
    registry.redhat.io/openshift3/metrics-hawkular-metrics \
    registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
    registry.redhat.io/openshift3/metrics-heapster \
    registry.redhat.io/openshift3/metrics-schema-installer \
    registry.redhat.io/openshift3/oauth-proxy \
    registry.redhat.io/openshift3/ose-logging-curator5 \
    registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
    registry.redhat.io/openshift3/ose-logging-eventrouter \
    registry.redhat.io/openshift3/ose-logging-fluentd \
    registry.redhat.io/openshift3/ose-logging-kibana5 \
    registry.redhat.io/openshift3/prometheus \
    registry.redhat.io/openshift3/prometheus-alertmanager \
    registry.redhat.io/openshift3/prometheus-node-exporter \
    registry.redhat.io/cloudforms46/cfme-openshift-postgresql \
    registry.redhat.io/cloudforms46/cfme-openshift-memcached \
    registry.redhat.io/cloudforms46/cfme-openshift-app-ui \
    registry.redhat.io/cloudforms46/cfme-openshift-app \
    registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible \
    registry.redhat.io/cloudforms46/cfme-openshift-httpd \
    registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator \
    registry.redhat.io/rhgs3/rhgs-server-rhel7 \
    registry.redhat.io/rhgs3/rhgs-volmanager-rhel7 \
    registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7 \
    registry.redhat.io/rhgs3/rhgs-s3-server-rhel7 \
```

- IBM POWER8 または IBM POWER9 サーバーでのオンプレミスインストールの場合は、以下のコマンドを実行します。

```
$ docker save -o ose3-optional-imags.tar \
    registry.redhat.io/openshift3/metrics-cassandra \
    registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
    registry.redhat.io/openshift3/metrics-heapster \
    registry.redhat.io/openshift3/metrics-schema-installer \
    registry.redhat.io/openshift3/oauth-proxy \
    registry.redhat.io/openshift3/ose-logging-curator5 \
    registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
    registry.redhat.io/openshift3/ose-logging-eventrouter \
    registry.redhat.io/openshift3/ose-logging-fluentd \
    registry.redhat.io/openshift3/ose-logging-kibana5 \
    registry.redhat.io/openshift3/prometheus \
    registry.redhat.io/openshift3/prometheus-alert-buffer \
    registry.redhat.io/openshift3/prometheus-alertmanager \
    registry.redhat.io/openshift3/prometheus-node-exporter \
```

4. プルした S2I ビルダーイメージをエクスポートします。たとえば、Jenkins および Tomcat イメージのみを同期している場合は、以下を実行します。

```
$ docker save -o ose3-builder-images.tar \
    registry.redhat.io/jboss-webserver-3/webserver31-tomcat7-openshift:<tag> \
    registry.redhat.io/jboss-webserver-3/webserver31-tomcat8-openshift:<tag> \
    registry.redhat.io/openshift3/jenkins-2-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-agent-maven-35-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-agent-nodejs-8-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-base-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-maven-rhel7:<tag> \
    registry.redhat.io/openshift3/jenkins-slave-nodejs-rhel7:<tag> \
```

5. 圧縮されたファイルをインターネットに接続されたホストから内部ホストにコピーします。

6. コピーしたイメージを読み込みます。

```
$ docker load -i ose3-images.tar
$ docker load -i ose3-builder-images.tar
$ docker load -i ose3-optional-images.tar
```

7.3. リポジトリーサーバーの準備および設定

インストール時および追加の更新時に、ソフトウェアをホストする Web サーバーが必要になります。RHEL 7 は Apache Web サーバーを提供します。

1. Web サーバーを準備します。

   a. 非接続環境に新規の Web サーバーをインストールする必要がある場合は、110 GB 以上の領域を持つ新規の RHEL 7 システムを LAN でインストールします。RHEL インストール時に、**Basic Web Server**オプションを選択します。

   b. OpenShift Container Platform ソフトウェアをダウンロードしており、イメージが必要なサーバーを再利用している場合、Apache をサーバーにインストールします。

```
$ sudo yum install httpd
```

2. リポジトリーファイルを Apache のルートフォルダーに配置します。

   - サーバーを再利用している場合は、以下を実行します。

```
$ mv /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

   - 新規サーバーをインストールしている場合、外部ストレージを割り当ててから、ファイルをコピーします。

```
$ cp -a /path/to/repos /var/www/html/
$ chmod -R +r /var/www/html/repos
$ restorecon -vR /var/www/html
```

3. ファイアウォールのルールを追加します。

```
$ sudo firewall-cmd --permanent --add-service=http
$ sudo firewall-cmd --reload
```

4. 変更を有効にするには、Apache を有効にしてから起動します。

```
$ systemctl enable httpd
$ systemctl start httpd
```

7.4. レジストリーの設定

非接続環境でイメージにタグを付け、そのイメージを内部レジストリーにプッシュします。

重要

以下の手順では、イメージをレジストリーに読み込む方法についての概要を示します。イメージを読み込む際に、追加の、または異なるアクションを実行する必要がある可能性があります。

1. イメージをレジストリーにプッシュする前に、それぞれのイメージに再度タグを付けます。

   - **openshift3** リポジトリーのイメージについては、イメージにメジャーおよびマイナー番号の両方のタグを付けます。たとえば、OpenShift Container Platform ノードイメージにタグを付けるには、以下を実行します。

     ```
     $ docker tag registry.redhat.io/openshift3/ose-node:<tag>
     registry.example.com/openshift3/ose-node:<tag>
     $ docker tag registry.redhat.io/openshift3/ose-node:<tag>
     registry.example.com/openshift3/ose-node:{major-tag}
     ```

   - 他のイメージについては、イメージに完全に一致するバージョン番号のタグを付けます。たとえば、etcd イメージにタグを付けるには、以下を実行します。

     ```
     $ docker tag registry.redhat.io/rhel7/etcd:3.2.28 registry.example.com/rhel7/etcd:3.2.28
     ```

2. 各イメージをレジストリーにプッシュします。たとえば、OpenShift Container Platform ノードイメージをプッシュするには、以下を実行します。

   ```
   $ docker push registry.example.com/openshift3/ose-node:<tag>
   $ docker push registry.example.com/openshift3/ose-node:{major-tag}
   ```

7.5. クラスターホストの準備

インストールファイルを準備したら、次にホストを準備します。

1. OpenShift Container Platform クラスターのホストを作成します。最新バージョンの RHEL 7 を使用し、最小インストールを実行することが推奨されます。ホストが システム要件 を満たしていることを確認します。

2. 各ノードホストで、リポジトリー定義を作成します。以下のテキストを **/etc/yum.repos.d/ose.repo** ファイルに配置します。

   ```
   [rhel-7-server-rpms]
   ```

```
name=rhel-7-server-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-rpms (1)
enabled=1
gpgcheck=0
[rhel-7-server-extras-rpms]
name=rhel-7-server-extras-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-extras-rpms (2)
enabled=1
gpgcheck=0
[rhel-7-server-ansible-2.9-rpms]
name=rhel-7-server-ansible-2.9-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ansible-2.9-rpms (3)
enabled=1
gpgcheck=0
[rhel-7-server-ose-3.11-rpms]
name=rhel-7-server-ose-3.11-rpms
baseurl=http://<server_IP>/repos/rhel-7-server-ose-3.11-rpms (4)
enabled=1
gpgcheck=0
```

(1)(2)(3)(4) **<server_IP>** を IP アドレス、またはソフトウェアリポジトリーをホストする Apache サーバーの名前に置き換えます。

3. ホストのインストールを準備します。ホストの準備 の手順に従い、**ホスト登録**のセクションの手順は省略します。

7.6. OPENSHIFT CONTAINER PLATFORM のインストール

ソフトウェア、イメージおよびホストを準備したら、標準的なインストール方法を使用して OpenShift Container Platform をインストールします。

1. 内部レジストリーを参照するように インベントリーファイルを設定 します。

- 内部レジストリーの場合:

```
oreg_url=registry.example.com/openshift3/ose-<component>:<version> (1)
openshift_examples_modify_imagestreams=true
```

(1) **ose** コンポーネントの名前およびバージョン番号の両方を指定します。

- Satellite イメージレジストリーの場合:

```
oreg_url=satellite.example.com/oreg-prod-openshift3_ose-<component>:<version> (1)
osm_etcd_image=satellite.example.com/oreg-prod-rhel7_etcd:3.2.28 (2)
openshift_examples_modify_imagestreams=true
```

(1) **ose** コンポーネントの名前およびバージョン番号の両方を指定します。

(2) etcd イメージの URL 接頭辞が Satellite サーバー上では異なる場合、**osm_etcd_image** パラメーターに etcd イメージの場所および名前を指定する必要があります。

2. インストール Playbook の実行

第8章 OPENSHIFT CONTAINER コンテナーイメージレジストリーのスタンドアロンデプロイメントのインストール

OpenShift Container Platform は、OpenShift Container レジストリー (OCR) と呼ばれる統合コンテナーイメージレジストリーを含む完全な機能を備えたエンタープライズソリューションです。また、OpenShift Container Platform を開発者向けの完全な PaaS 環境としてデプロイする代わりに、 OCR をスタンドアロンのコンテナーイメージレジストリーとしてインストールし、オンサイトまたはクラウドで実行することも可能です。

OCR のスタンドアロンデプロイメントをインストールすると、標準的な OpenShift Container Platform のインストールと同様にマスターとノードのクラスターも引き続きインストールされます。次に、コンテナーイメージレジストリーはそのクラスター上で実行されるようにデプロイされます。このスタンドアロンデプロイメントのオプションは、コンテナーイメージレジストリーは必要だが、開発者向けの Web コンソールやアプリケーションのビルドおよびデプロイツールを含む OpenShift Container Platform の完全な環境は必要ない、という管理者に役立ちます。

OCR には以下の機能があります。

- ユーザー向けのレジストリー Web コンソール Cockpit。
- デフォルトの セキュリティー保護されたトラフィック (TLS 経由で提供される)。
- グローバルな アイデンティティープロバイダー認証。
- チームが ロールベースのアクセス制御 (RBAC) 認証を通じて連携できるようにする プロジェクト namespace モデル。
- サービスを管理するための Kubernetes ベースのクラスター。
- イメージ管理を強化するための イメージストリーム というイメージの抽象化。

管理者は、スタンドアロン OCR をデプロイすることで OpenShift Container Platform の複数のクラスターに対応しているレジストリーを個別に管理できます。また、スタンドアロン OCR を使うと、セキュリティーやコンプライアンスに関する独自の要件を満たすようにレジストリーを分離することも可能です。

8.1. ハードウェアの最小要件

スタンドアロン OCR をインストールするためのハードウェア要件は以下の通りです。

- 物理または仮想システム、またはパブリックまたはプライベート IaaS で実行されるインスタンス。
- ベース OS: RHEL 7.5 以降 (RHEL 7 Extras チャンネルの最小限のインストールオプションおよび最新のパッケージ)、または、RHEL Atomic Host 7.4.5 以降。
- NetworkManager 1.0 以降。
- 2 vCPU。
- 16 GB 以上の RAM。
- **/var/** を含むファイルシステムの 15 GB 以上のハードディスク領域。
- Docker のストレージバックエンドに使用する 15 GB 以上の追加の未割り当て領域。 詳細は Docker ストレージの設定 を参照してください。

重要

OpenShift Container Platform は x86_64 or IBM POWER アーキテクチャーを使用するサーバーをサポートします。IBM POWER サーバーを使用してクラスターホストをホストする場合は、使用できるサーバーは IBM POWER のみになります。

注記

RHEL Atomic Host の **/var/** のファイルシステムのサイジング要件を満たすには、デフォルト設定を変更する必要があります。インストール時またはインストール後にこの設定を行う方法については Managing Storage in Red Hat Enterprise Linux Atomic Host を参照してください。

8.2. サポートされているシステムトポロジー

以下のシステムトポロジーはスタンドアロン OCR でサポートされています。

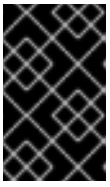
オールインワン	マスター、ノード、レジストリーの各コンポーネントを含む単一ホスト。
複数マスター (高可用性)	すべてのコンポーネント (マスター、ノード、レジストリー) がそれぞれに含まれる 3 つのホスト。 マスターはネイティブの高可用性を確保するように設定されます。

8.3. OPENSHIFT CONTAINER レジストリーのインストール

1. 最初に インストールの準備 を確認し、完全なクラスターインストールプロセスを確認します。OCR のインストールは同じプロセスを使用しますが、インベントリーファイルにいくつかの特定の設定が必要です。インストールのドキュメントには、インベントリーファイルの利用可能な Ansible 変数の総合的な一覧が記載されています。

2. ホスト準備 の手順を完了します。

3. インベントリーファイル を **/etc/ansible/hosts** ディレクトリーに作成します。



重要

スタンドアロン OCR をインストールするには、インベントリーファイルの **[OSEv3:vars]** セクションに **deployment_subtype=registry** を設定する必要があります。

以下のサポートされている複数の異なるシステムトポロジー用のインベントリーファイルのサンプルを使用します。

オールインワンのスタンドアロン OpenShift Container レジストリーインベントリーファイル

```
# Create an OSEv3 group that contains the masters and nodes groups
[OSEv3:children]
masters
nodes
etcd
```

```
# Set variables common for all OSEv3 hosts
[OSEv3:vars]
# SSH user, this user should allow ssh based auth without requiring a password
ansible_ssh_user=root

openshift_master_default_subdomain=apps.test.example.com

# If ansible_ssh_user is not root, ansible_become must be set to true
#ansible_become=true

openshift_deployment_type=openshift-enterprise
deployment_subtype=registry (1)
openshift_hosted_infra_selector="" (2)

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# host group for masters
[masters]
registry.example.com

# host group for etcd
[etcd]
registry.example.com

# host group for nodes
[nodes]
registry.example.com openshift_node_group_name='node-config-all-in-one'
```

1. **deployment_subtype=registry** を設定して、OpenShift Container Platform 環境のすべてではなく、スタンドアロン OCR がインストールされるようにします。
2. レジストリーとその Web コンソールが単一ホストでスケジュールされることを可能にします。

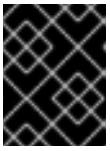
複数マスター (高可用性) スタンドアロン OpenShift Container レジストリーインベントリーファイル

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
nodes
etcd
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
deployment_subtype=registry (1)
```

```
openshift_master_default_subdomain=apps.test.example.com

# Uncomment the following to enable htpasswd authentication; defaults to
# DenyAllPasswordIdentityProvider.
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-internal.example.com
openshift_master_cluster_public_hostname=openshift-cluster.example.com

# apply updated node-config-compute group defaults
openshift_node_groups=[{'name': 'node-config-compute', 'labels': ['node-
role.kubernetes.io/compute=true'], 'edits': [{'key': 'kubeletArguments.max-pods','value':
['250']}, {'key': 'kubeletArguments.image-gc-high-threshold', 'value':['90']}, {'key':
'kubeletArguments.image-gc-low-threshold', 'value': ['80']}]}]

# enable ntp on masters to ensure proper failover
openshift_clock_enabled=true

# host group for masters
[masters]
master1.example.com
master2.example.com
master3.example.com

# host group for etcd
[etcd]
etcd1.example.com
etcd2.example.com
etcd3.example.com

# Specify load balancer host
[lb]
lb.example.com

# host group for nodes, includes region info
[nodes]
master[1:3].example.com openshift_node_group_name='node-config-master-infra'
node1.example.com       openshift_node_group_name='node-config-compute'
node2.example.com       openshift_node_group_name='node-config-compute'
```

1 **deployment_subtype=registry** を設定して、OpenShift Container Platform 環境のすべてではなく、スタンドアロン OCR がインストールされるようにします。

4. スタンドアロンの OCR をインストールします。このプロセスは、完全な クラスターインストール プロセスに似ています。



重要

Ansible Playbook を実行するホストには、ホストあたり 75MiB 以上の空きメモリーがインベントリーファイルで必要になります。

a. 新規クラスターをデプロイする前に、クラスターのディレクトリーに切り替え、**prerequisites.yml** Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \ 1
    playbooks/prerequisites.yml
```

1 インベントリーファイルが **/etc/ansible/hosts** ディレクトリーにない場合、**-i** およびインベントリーファイルのパスを指定します。

この Playbook は一回のみ実行する必要があります。

b. インストールを開始するには、Playbook ディレクトリーに切り替え、**deploy_cluster.yml** Playbook を実行します。

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook  [-i /path/to/inventory] \ 1
    playbooks/deploy_cluster.yml
```

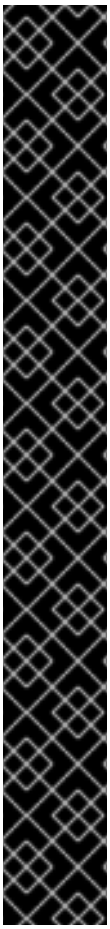
1 インベントリーファイルが **/etc/ansible/hosts** ディレクトリーにない場合、**-i** およびインベントリーファイルのパスを指定します。

第9章 OPENSHIFT CONTAINER PLATFORM のアンインストール

クラスターの OpenShift Container Platform ホストをアンインストールするには、**uninstall.yml** Playbook を実行します。この Playbook は、Ansible によってインストールされた OpenShift Container Platform コンテンツを削除します。 これには以下が含まれます。

- 設定
- コンテナー
- デフォルトのテンプレートとイメージストリーム
- イメージ
- RPM パッケージ

Playbook は、Playbook の実行時に指定するインベントリーファイルに定義されたホストのコンテンツを削除します。



重要

クラスターをアンインストールする前に、シナリオの以下の一覧を確認し、アンインストールが最適なオプションであることを確認します。

- インストールプロセスが失敗しており、このプロセスを続行する必要がある場合は、インストールを再試行 できます。インストール Playbook は、クラスターのインストールに失敗した場合に、クラスターをアンインストールする必要なくそれらを再度実行できるように設計されています。
- 失敗したインストールを最初から再開する必要がある場合は、以下のセクションで説明されているように **uninstall.yml** Playbook を実行して、クラスターで OpenShift Container Platform ホストをアンインストールできます。この Playbook は、インストールした最新バージョンの OpenShift Container Platform アセットのみをアンインストールします。
- ホスト名または証明書名を変更する必要がある場合は、**uninstall.yml** Playbook を実行し、インストールの再試行前に証明書を再作成する必要があります。インストール Playbook を再度実行しても、証明書は再作成されません。
- 以前に OpenShift Container Platform をインストールしたホストの用途を変更する必要がある場合 (概念実証のインストールなど) や、異なるマイナーバージョンまたは非同期バージョンの OpenShift Container Platform をインストールする必要がある場合は、ホストの再イメージ化を実行してから、それらを実稼働クラスターで使用する必要があります。**uninstall.yml** Playbook の実行後、一部のホストアセットは変更された状態のままになる可能性があります。

9.1. OPENSHIFT CONTAINER PLATFORM クラスターのアンインストール

クラスター内のすべてのホストで OpenShift Container Platform をアンインストールするには、Playbook ディレクトリーに切り替え、最近使用したインベントリーファイルを使用して Playbook を実行します。

```
# ansible-playbook [-i /path/to/file] \ 1
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

-

1 インベントリーファイルが **/etc/ansible/hosts** ディレクトリーにない場合、**-i** およびインベントリーファイルのパスを指定します。

9.2. ノードのアンインストール

uninstall.yml Playbook を使用してノードコンポーネントを特定のホストからアンインストールし、それ以外のホストとクラスターをそのままにしておくには、以下を実行します。

警告

特定のマスターまたは etcd ホストではなく、特定のノードホストのアンインストールを試行する場合にのみこの方法を使用します。マスターまたは etcd ホストのアンインストールでは、クラスターにさらに多くの設定の変更が必要になります。

1. ノードオブジェクトをクラスターから削除するには、ノードの削除 の手順に従います。
2. これらのホストのみを参照する別のインベントリーファイルを作成します。たとえば、1 つのノードからのみコンテンツを削除する場合は、以下を実行します。

```
[OSEv3:children]
nodes 1

[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise

[nodes]
node3.example.com openshift_node_group_name='node-config-infra' 2
```

1 アンインストールするホストに適用されるセクションのみを含めます。

2 アンインストールするホストのみを含めます。

3. Playbook ディレクトリーに切り替え、**uninstall.yml** Playbook を実行します。

```
# ansible-playbook -i /path/to/new/file \ 1
    /usr/share/ansible/openshift-ansible/playbooks/adhoc/uninstall.yml
```

1 新規インベントリーファイルへのパスを指定します。

Playbook が完了すると、すべての OpenShift Container Platform コンテンツが指定したホストから削除されます。