OpenShift Container Platform 4.18

サポート

OpenShift Container Platform のサポート

Last Updated: 2025-12-06

OpenShift Container Platform 4.18 サポート

OpenShift Container Platform のサポート

Legal Notice

Abstract

このドキュメントでは、OpenShift Container Platform に関する Red Hat サポートを得る方法に関する情報を提供します。また、Telemetry および Insights Operator を使用したリモートヘルスモニタリングに関する情報も含まれます。また、リモートヘルスモニタリングの利点も説明します。

Table of Contents

第1章 サポートの概要

Red Hat は、クラスターのデータを収集し、モニタリングとトラブルシューティングを行うためのツールをクラスター管理者に提供します。

1.1. サポートの利用

サポートの利用: Red Hat カスタマーポータルにアクセスして、ナレッジベースの記事の確認、サポートケースの作成、追加の製品ドキュメントおよびリソースの確認を行ってください。

1.2. リモートヘルスモニタリングの問題

リモートヘルスモニタリングの問題: OpenShift Container Platform はクラスターの Telemetry および設定データを収集し、Telemeter Client および Insights Operator を使用してこのデータを Red Hat に報告します。Red Hat はこのデータを使用して、**接続されたクラスター** での問題を理解し、解決します。接続されたクラスターと同様に、ネットワークが制限された環境で、リモートヘルスモニタリングを使用できます。OpenShift Container Platform は以下を使用してデータを収集して正常性を監視します。

- **Telemetry**: Telemetry クライアントは、4 分 30 秒ごとにメトリクス値を収集し、Red Hat にアップロードします。Red Hat はこのデータを使用して以下を行います。
    - クラスターの監視。
    - OpenShift Container Platform のアップグレードのロールアウト。
    - アップグレードエクスペリエンスの向上。
- **Insights Operator**: OpenShift Container Platform は、デフォルトで Insights Operator をインストールして有効にします。この Operator は、2 時間ごとに設定とコンポーネントの障害ステータスを報告します。Insights Operator は次のことに役立ちます。
    - 発生する可能性のあるクラスターの問題を事前に特定する。
    - Red Hat OpenShift Cluster Manager でソリューションと予防措置を提供する。

Telemetry 情報を確認 できます。

リモートヘルスレポートを有効にしている場合は、Insights を使用してクラスターの問題を特定 します。必要に応じて、リモートヘルスレポートを無効にできます。

1.3. クラスターに関するデータの収集

クラスターに関するデータの収集: Red Hat は、サポートケースの作成時にデバッグ情報を収集することを推奨します。デバッグ情報があると、Red Hat サポートが根本原因を分析するのに役立ちます。クラスター管理者は、以下を使用してクラスターに関するデータを収集できます。

- **must-gather ツール**: **must-gather** ツールを使用してクラスターの情報を収集し、問題のデバッグを行います。
- **sosreport**: **sosreport** ツールを使用して、デバッグ目的で設定の詳細、システム情報、および診断データを収集します。
- **Cluster ID**: Red Hat サポートに情報を提供する際に、クラスターの一意 ID を取得します。

- **ブートストラップノードのジャーナルログ: bootkube.service** の **journald** ユニットログと、ブートストラップノードからコンテナーログを収集し、ブートストラップ関連の問題をトラブルシューティングします。
- **クラスターノードのジャーナルログ**: ノード関連の問題のトラブルシューティングに、各クラスターの **/var/log** にあるログと、**journald** ユニットログを収集します。
- **ネットワークトレース:** Red Hat サポートがネットワーク関連の問題をトラブルシューティングできるように、固有の OpenShift Container Platform クラスターノードまたはコンテナーからネットワークパケットトレースを提供します。

1.4. 問題のトラブルシューティング

クラスター管理者は、以下の OpenShift Container Platform コンポーネントの問題を監視し、トラブルシューティングできます。

- インストールの問題: OpenShift Container Platform のインストールは段階を追って進められます。以下を実行できます。
  - インストールステージの監視。
  - インストールのどの段階で発生するかの判断。
  - 複数のインストールの問題調査。
  - 失敗したインストールからのログ収集。
- ノードの問題: クラスター管理者は、ノードのステータス、リソースの使用状況、および設定を確認して、ノード関連の問題を検証およびトラブルシューティングできます。以下に対してクエリーを実行できます。
  - ノード上の kubelet のステータス。
  - クラスターノードジャーナルログ。
- Crio の問題: クラスター管理者は、各クラスターノードで CRI-O コンテナーランタイムエンジンのステータスを確認できます。コンテナーランタイムの問題が発生した場合には、以下を実行します。
  - CRI-O journald ユニットログを収集します。
  - CRI-O ストレージをクリーンアップします。
- オペレーティングシステムの問題: OpenShift Container Platform は Red Hat Enterprise Linux CoreOS で実行されます。オペレーティングシステムの問題が発生した場合は、カーネルクラッシュの手順を調査してください。以下の点を行うようにしてください。
  - kdump が有効である。
  - kdump 設定をテストする。
  - コアダンプを分析する。
- ネットワークの問題: クラスター管理者は以下を実行して、Open vSwitch の問題をトラブルシューティングできます。
  - Open vSwitch のログレベルを一時的に設定する。

    - Open vSwitch のログレベルを永続的に設定する。
    - Open vSwitch のログを表示する。
- Operator の問題: クラスター管理者は以下を実行して、Operator の問題を解決できます。
    - Operator サブスクリプションのステータスを確認する。
    - Operator Pod の正常性を確認する。
    - Operator ログを収集する。
- Pod の問題: クラスター管理者は、Pod のステータスを確認して以下を実行し、Pod 関連の問題のトラブルシューティングを行うことができます。
    - Pod およびコンテナーのログを確認する。
    - root アクセスでデバッグ Pod を起動する。
- Source-to-Image の問題: クラスター管理者は S2I ステージを確認し、S2I プロセスのどこで障害が発生したかを判断できます。Source-to-Image(S2I) の問題を解決するには、以下を収集します。
    - Source-to-Image 診断データ。
    - アプリケーションの障害を調査するためのアプリケーション診断データ。
- ストレージの問題: 障害のあるノードがアタッチしたボリュームをアンマウントできないことが原因で、新しいノードにボリュームをマウントできない場合、マルチアタッチストレージエラーが発生します。クラスター管理者は、以下を実行して、複数アタッチされているストレージの問題を解決できます。
    - RWX ボリュームを使用して、複数割り当てを有効にします。
    - RWO ボリュームの使用時に障害が発生したノードを回復するか、削除します。
- モニタリングの問題: クラスター管理者は、モニタリングに関するトラブルシューティングページの手順を実行してください。ユーザー定義プロジェクトのメトリクスが利用できない場合や、Prometheus が大量のディスク領域を消費している場合は、以下を確認します。
    - ユーザー定義のメトリクスが利用できない理由を調べる。
    - Prometheus が大量のディスク領域を消費している理由を特定する。
- OpenShift CLI (**oc**) の問題: ログレベルを増やすことで OpenShift CLI (**oc**) の問題を調査します。

第2章 クラスターリソースの管理

OpenShift Container Platform でグローバル設定オプションを適用できます。Operator はこれらの設定をクラスター全体に適用します。

2.1. クラスターリソースの操作

OpenShift Container Platform の OpenShift CLI (**oc**) ツールを使用してクラスターリソースを操作できます。**oc api-resources** コマンドの実行後に表示されるクラスターリソースを編集できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- Web コンソールにアクセスできるか、**oc** CLI ツールがインストールされている。

手順

1. 適用された設定 Operator を確認するには、以下のコマンドを実行します。

```
$ oc api-resources -o name | grep config.openshift.io
```

2. 設定可能なクラスターリソースを表示するには、以下のコマンドを実行します。

```
$ oc explain <resource_name>.config.openshift.io
```

3. クラスターのカスタムリソース定義 (CRD) オブジェクトの設定を表示するには、以下のコマンドを実行します。

```
$ oc get <resource_name>.config -o yaml
```

4. クラスターリソース設定を編集するには、以下のコマンドを実行します。

```
$ oc edit <resource_name>.config -o yaml
```

第3章 サポート

3.1. サポート

このドキュメントで説明されている手順、または OpenShift Container Platform 全般で問題が発生した場合は、Red Hat カスタマーポータル にアクセスしてください。

カスタマーポータルでは、次のことができます。

- Red Hat 製品に関するアーティクルおよびソリューションを対象とした Red Hat ナレッジベースの検索またはブラウズ。
- Red Hat サポートに対するサポートケースの送信。
- その他の製品ドキュメントへのアクセス。

クラスターの問題を特定するには、OpenShift Cluster Manager で Insights を使用できます。Insights により、問題の詳細と、利用可能な場合は問題の解決方法に関する情報が提供されます。

このドキュメントを改善するための提案がある場合、またはエラーを見つけた場合は、最も関連性の高いドキュメントコンポーネントについて Jira 課題 を送信してください。セクション名や OpenShift Container Platform バージョンなどの具体的な情報を提供してください。

3.2. RED HAT ナレッジベースについて

Red Hat ナレッジベース は、お客様が Red Hat の製品やテクノロジーを最大限に活用できるようにするための豊富なコンテンツを提供します。Red Hat ナレッジベースは、Red Hat 製品のインストール、設定、および使用に関する記事、製品ドキュメント、および動画で構成されています。さらに、既知の問題に対する解決策を検索でき、それぞれに根本原因の簡潔な説明と修復手順が記載されています。

3.3. RED HAT ナレッジベースの検索

OpenShift Container Platform の問題が発生した場合には、初期検索を実行して、解決策を Red Hat ナレッジベース内ですでに見つけることができるかどうかを確認できます。

前提条件

- Red Hat カスタマーポータルのアカウントがある。

手順

1. Red Hat カスタマーポータル にログインします。
2. **Search** をクリックします。
3. 検索フィールドに、問題に関連する次のようなキーワードと文字列を入力します。
   - OpenShift Container Platform コンポーネント (**etcd** など)
   - 関連する手順 (**installation** など)
   - 明示的な失敗に関連する警告、エラーメッセージ、およびその他の出力
4. **Enter** キーをクリックします。

5. オプション: **OpenShift Container Platform** 製品フィルターを選択します。
6. オプション: **Documentation** コンテンツタイプフィルターを選択します。

3.4. サポートケースの作成

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat カスタマーポータルのアカウントがある。
- Red Hat の Standard または Premium サブスクリプションがある。

手順

1. Red Hat カスタマーポータルの Customer Support ページ にログインします。
2. **Get support** をクリックします。
3. **Customer Support** ページの **Cases** タブで、以下を行います。
   a. オプション: 必要に応じて、事前に入力されたアカウントと所有者の詳細を変更します。
   b. 問題に該当するカテゴリー (**Bug**、**Defect** など) を選択し、**Continue** をクリックします。
4. 以下の情報を入力します。
   a. **Summary** フィールドには、問題の簡潔で説明的な概要と、確認されている現象および予想される動作の詳細情報を入力します。
   b. **Product** ドロップダウンメニューから **OpenShift Container Platform** を選択します。
   c. **Version** ドロップダウンから **4.18** を選択します。
5. Red Hat ナレッジベースで推奨されるソリューション一覧を確認してください。この一覧に上げられているソリューションは、報告しようとしている問題に適用される可能性があります。提案されている記事が問題に対応していない場合は、**Continue** をクリックします。
6. 報告している問題に対する一致に基づいて推奨される Red Hat ナレッジベースソリューションの一覧が更新されることを確認してください。ケース作成プロセスでより多くの情報を提供すると、このリストの絞り込みが行われます。提案されている記事が問題に対応していない場合は、**Continue** をクリックします。
7. アカウント情報が予想通りに表示されていることを確認し、そうでない場合は適宜修正します。
8. 自動入力された OpenShift Container Platform クラスター ID が正しいことを確認します。正しくない場合は、クラスター ID を手動で取得します。
   - OpenShift Container Platform Web コンソールを使用してクラスター ID を手動で取得するには、以下を実行します。
     a. **Home → Overview** に移動します。

b. **Details** セクションの **Cluster ID** フィールドで値を見つけます。

- または、OpenShift Container Platform Web コンソールで新規サポートケースを作成し、クラスター ID を自動的に入力することができます。

a. ツールバーから、**(?)Help → Open Support Case** に移動します。

b. **Cluster ID** 値が自動的に入力されます。

- OpenShift CLI (**oc**) を使用してクラスター ID を取得するには、以下のコマンドを実行します。

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
```

9. プロンプトが表示されたら、以下の質問に回答し、**Continue** をクリックします。
   - What are you experiencing? What are you expecting to happen?
   - Define the value or impact to you or the business.
   - Where are you experiencing this behavior? What environment?
   - When does this behavior occur? Frequency? 繰り返し発生するか。At certain times?
10. 関連する診断データファイルをアップロードし、**Continue** をクリックします。まずは、**oc adm must-gather** コマンドを使用して収集されるデータと、そのコマンドによって収集されない問題に固有のデータを含めることが推奨されます。
11. 関連するケース管理の詳細情報を入力し、**Continue** をクリックします。
12. ケースの詳細をプレビューし、**Submit** をクリックします。

3.5. 関連情報

- クラスターの問題を特定する方法の詳細は、Insights を使用したクラスターの問題の特定 を参照してください。

第4章 接続クラスターを使用したリモートヘルスモニタリング

4.1. リモートヘルスモニタリングについて

OpenShift Container Platform は、クラスターに関する Telemetry および設定データを収集し、Telemeter Client および Insights Operator を使用してこれを Red Hat にレポートします。Red Hat に提供されるデータは、このドキュメントで説明されている利点を提供します。

Telemetry および Insights Operator 経由でデータを Red Hat にレポートするクラスターは **接続クラスター (connected cluster)** と見なされます。

Telemetry は、OpenShift Container Platform Telemeter Client によって Red Hat に送信される情報を表すために Red Hat が使用する用語です。サブスクリプション管理の自動化、クラスターの健全性の監視、サポートの支援、カスタマークスペリエンスの向上を可能にするために、軽量の属性データが接続クラスターから Red Hat に送信されます。

Insights Operator は OpenShift Container Platform 設定データを収集し、これを Red Hat に送信します。データは、クラスターがさらされる可能性のある問題に関する洞察を生み出すために使用されます。これらの洞察は、OpenShift Cluster Manager でクラスター管理者に伝達されます。

これらの 2 つのプロセスの詳細は、このドキュメントを参照してください。

Telemetry および Insights Operator の利点

ユーザーにとって、Telemetry および Insights Operator には次のような利点があります。

- **問題の特定および解決の強化**。エンドユーザーには正常と思われるイベントも、Red Hat は多くのお客様を含む全体的な視点で観察できます。この視点により、一部の問題はより迅速に特定され、エンドユーザーがサポートケースを作成したり、Jira 問題 を作成しなくても解決することが可能です。

- **高度なリリース管理**。OpenShift Container Platform は **candidate**、**fast**、および **stable** リリースチャネルを提供し、これにより更新ストラテジーを選択することができます。リリースの **fast** から **stable** に移行できるかどうかは、更新の成功率やアップグレード時に確認されるイベントに依存します。接続クラスターから提供される情報により、Red Hat は **stable** チャネルのリリースの品質を高め、**fast** チャネルで見つかった問題に対してより迅速に対応することができます。

- **ターゲットが絞られた新機能の優先付け**。収集されるデータは、最も使用される OpenShift Container Platform の領域に関する洞察を提供します。この情報により、Red Hat はお客様に最も大きな影響を与える新機能の開発に重点的に取り組むことができます。

- **効率的なサポートエクスペリエンス**。Red Hat カスタマーポータル でサポートチケットを作成する際に、接続クラスターのクラスター ID を指定できます。これにより、Red Hat は接続された情報を使用してクラスター固有の効率化されたサポートエクスペリエンスを提供できます。このドキュメントは、強化されたサポートエクスペリエンスの詳細情報を提供しています。

- **予測分析**。OpenShift Cluster Manager に表示されるお客様のクラスターに関する分析情報は、接続クラスターから収集された情報によって実現されています。Red Hat は、OpenShift Container Platform クラスターがさらされる問題を特定するのに役立つディープラーニング (深層学習)、機械学習、および人工知能の自動化の適用に取り組んでいます。

4.1.1. Telemetry について

Telemetry は厳選されたクラスターモニタリングメトリクスのサブセットを Red Hat に送信します。Telemeter Client はメトリクスの値を 4 分 30 秒ごとに取得し、データを Red Hat にアップロードします。これらのメトリクスは、このドキュメントで説明しています。

このデータのストリームは、Red Hat によってリアルタイムでクラスターをモニターし、お客様に影響を与える問題に随時対応するために使用されます。またこれにより、Red Hat がサービスへの影響を最小限に抑えつつつアップグレードエクスペリエンスの継続的な改善に向けた OpenShift Container Platform のアップグレードのデプロイメントを可能にします。

このデバッグ情報は、サポートケースでレポートされるデータへのアクセスと同じ制限が適用された状態で Red Hat サポートおよびエンジニアリングチームが利用できます。接続クラスターのすべての情報は、OpenShift Container Platform をより使用しやすく、より直感的に使用できるようにするために Red Hat によって使用されます。

関連情報

- クラスターの更新またはアップグレードの詳細は、OpenShift Container Platform の更新に関するドキュメント を参照してください。

4.1.1.1. Telemetry で収集される情報

以下の情報は、Telemetry によって収集されます。

4.1.1.1.1. システム情報

- OpenShift Container Platform クラスターのバージョン情報、および更新バージョンの可用性を特定するために使用されるインストールの更新の詳細を含むバージョン情報
- クラスターごとに利用可能な更新の数、更新に使用されるチャネルおよびイメージリポジトリー、更新の進捗情報、および更新で発生するエラーの数などの更新情報
- インストール時に生成される一意でランダムな識別子
- クラウドインフラストラクチャーレベルのノード設定、ホスト名、IP アドレス、Kubernetes Pod 名、namespace、およびサービスなど、Red Hat サポートがお客様にとって有用なサポートを提供するのに役立つ設定の詳細
- クラスターにインストールされている OpenShift Container Platform フレームワークコンポーネントおよびそれらの状態とステータス
- 動作が低下した Operator の "関連オブジェクト" として一覧表示されるすべての namespace のイベント
- 動作が低下したソフトウェアに関する情報
- 証明書の有効性に関する情報
- OpenShift Container Platform がデプロイされているプラットフォームの名前およびデータセンターの場所

4.1.1.1.2. サイジング情報

- CPU コアの数およびそれぞれに使用される RAM の容量を含む、クラスター、マシンタイプ、およびマシンに関するサイジング情報
- etcd メンバーの数および etcd クラスターに保存されるオブジェクトの数

- ビルドストラテジータイプ別のアプリケーションビルドの数

4.1.1.1.3. 使用情報

- コンポーネント、機能および拡張機能に関する使用率の情報
- テクノロジープレビューおよびサポート対象外の設定に関する使用率の詳細

Telemetry は、ユーザー名やパスワードなどの識別情報を収集しません。Red Hat は、意図的な個人情報の収集は行いません。誤って個人情報を受信したことが明らかになった場合、Red Hat はその情報を削除します。Telemetry データが個人データに該当する場合は、Red Hat プライバシーステートメントを参照し、Red Hat のプライバシー方針を確認してください。

関連情報

- Telemetry が OpenShift Container Platform で Prometheus から収集する属性を一覧表示する方法の詳細は、Telemetry によって収集されるデータの表示 を参照してください。
- Telemetry が Prometheus から収集する属性のリストは、アップストリームの cluster-monitoring-operator ソースコード を参照してください。
- Telemetry はデフォルトでインストールされ、有効にされます。リモートヘルスレポートをオプトアウトする必要がある場合は、リモートヘルスレポート を参照してください。

4.1.2. Insights Operator について

Insights Operator は設定およびコンポーネントの障害ステータスを定期的に収集し、デフォルトで 2 時間ごとにそのデータを Red Hat に報告します。この情報により、Red Hat は設定や Telemetry で報告されるデータよりも深層度の高いデータを評価できます。

OpenShift Container Platform のユーザーは、Red Hat Hybrid Cloud Console の Insights Advisor サービスで各クラスターのレポートを表示できます。問題が特定されると、Insights は詳細を提供します。利用可能な場合は、問題の解決方法に関する手順が提供されます。

Insights Operator は、ユーザー名、パスワード、または証明書などの識別情報を収集しません。Red Hat Insights のデータ収集とコントロールの詳細は、Red Hat Insights のデータおよびアプリケーションセキュリティー を参照してください。

Red Hat は、接続されたすべてのクラスター情報を使用して、以下を実行します。

- Red Hat Hybrid Cloud Console の Insights Advisor サービスで、潜在的なクラスターの問題を特定し、解決策と予防措置を提供します。
- 集計される情報および重要な情報を製品およびサポートチームに提供し、OpenShift Container Platform の強化を図ります。
- OpenShift Container Platform の直感的な使用方法

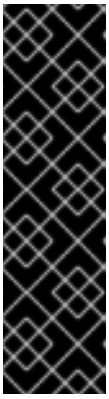
関連情報

- Insights Operator はデフォルトでインストールされ、有効にされます。リモートヘルスレポートをオプトアウトする必要がある場合は、リモートヘルスレポート を参照してください。

4.1.2.1. Insights Operator によって収集される情報

Insights Operator によって収集される情報は次のとおりです。

- OpenShift Container Platform のバージョンと環境に固有の問題を特定するための、クラスターとそのコンポーネントに関する一般情報。
- 誤った設定や設定するパラメーターに固有の問題の判別に使用するクラスターのイメージレジストリー設定などの設定ファイル。
- クラスターコンポーネントで発生するエラー。
- 実行中の更新の進捗情報、およびコンポーネントのアップグレードのステータス。
- OpenShift Container Platform がデプロイされているプラットフォームとクラスターが配置されているリージョンの詳細
- 秘密の Secure Hash Algorithm (SHA) 値に変換されたクラスターのワークロード情報。これにより、Red Hat は機密情報を開示することなく、ワークロードのセキュリティーとバージョンの脆弱性を評価できます。
- ランタイムの種類、名前、バージョンなど、オペレーティングシステムとランタイム環境に関するワークロード情報。これらは、必要に応じて **InsightsRuntimeExtractor** フィーチャーゲートを通じて有効にできます。このデータにより、Red Hat は OpenShift Container Platform コンテナーの使用方法をより的確に把握し、最適な使用率を実現するための投資判断を積極的に支援できるようになります。



重要

InsightsRuntimeExtractor 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 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

- Operator が問題を報告する場合、**openshift-*** および **kube-*** プロジェクトのコア OpenShift Container Platform Pod に関する情報が収集されます。これには、状態、リソース、セキュリティーコンテキスト、ボリューム情報などが含まれます。

関連情報

- Insights Operator によって収集されるデータを確認する方法の詳細は、Insights Operator によって収集されるデータの表示 を参照してください。
- What data is being collected by the Insights Operator in OpenShift?
- フィーチャーゲートを使用した機能の有効化
- Insights Operator のソースコードは、レビューおよびコントリビュートが可能です。Insights Operator によって収集される項目のリストは、Insights Operator のアップストリームプロジェクト を参照してください。

4.1.3. Telemetry および Insights Operator データフローについて

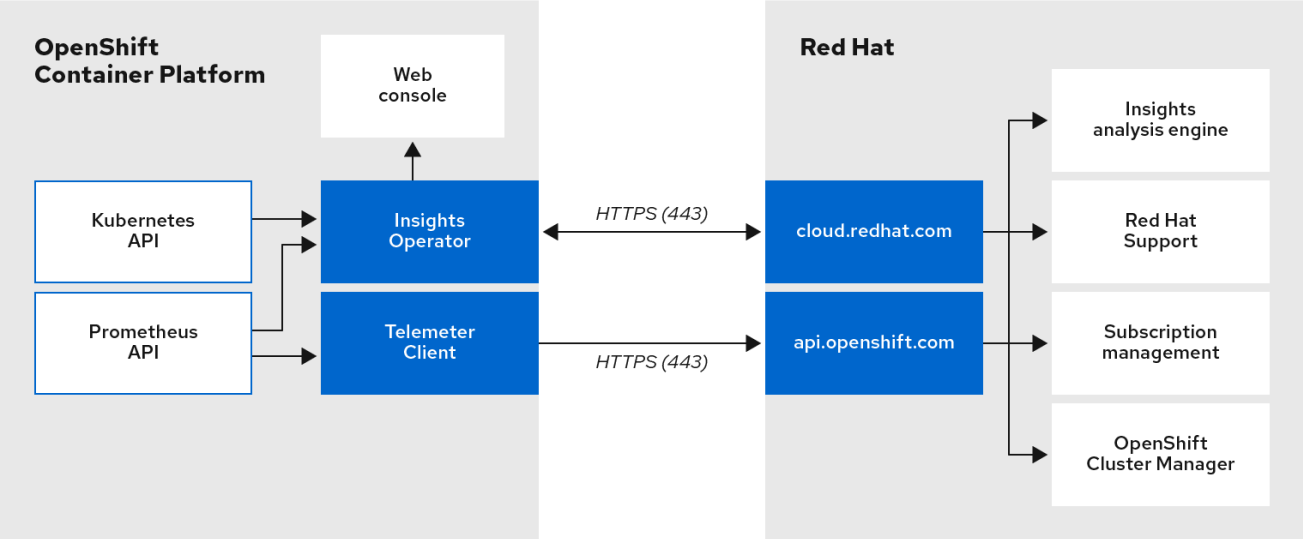
Telemeter Client は、Prometheus API から選択した時系列データを収集します。時系列データは、処理するために 4 分 30 秒ごとに api.openshift.com にアップロードされます。

Insights Operator は、選択したデータを Kubernetes API および Prometheus API からアーカイブに収集します。アーカイブは、処理のために 2 時間ごとに OpenShift Cluster Manager にアップロードされます。Insights Operator は、OpenShift Cluster Manager から最新の Insights 分析もダウンロードします。これは、OpenShift Container Platform Web コンソールの **Overview** ページに含まれる **Insights status** ポップアップを設定するために使用されます。

Red Hat との通信はすべて、Transport Layer Security (TLS) および相互証明書認証を使用して、暗号化されたチャネル上で行われます。すべてのデータは移動中および停止中に暗号化されます。

顧客データを処理するシステムへのアクセスは、マルチファクター認証と厳格な認証制御によって制御されます。アクセスは、知る必要がある場合に限って付与され、必要な操作に制限されます。

Telemetry および Insights Operator データフロー



関連情報

- OpenShift Container Platform モニタリングスタックの詳細は、OpenShift Container Platform モニタリングについて を参照してください。
- ファイアウォールを設定し、Telemetry および Insights のエンドポイントを有効にする方法の詳細は、ファイアウォールの設定 を参照してください。

4.1.4. リモートヘルスモニタリングデータの使用方法に関する追加情報

リモートヘルスモニタリングを有効にするために収集される情報の詳細は、Information collected by Telemetry および Information collected by the Insights Operator を参照してください。

このドキュメントで前述したとおり、Red Hat は、サポートおよびアップグレードの提供、パフォーマンス/設定の最適化、サービスへの影響の最小化、脅威の特定および修復、トラブルシューティング、オファリングおよびユーザーエクスペリエンスの強化、問題への対応および請求を目的として (該当する場合)、お客様の Red Hat 製品使用データを収集します。

収集における対策

Red Hat は、Telemetry および設定データを保護するために設計された技術的および組織的な対策を採用しています。

共有

Red Hat は、ユーザーエクスペリエンスの向上に向けて、Telemetry および Insights Operator で収集されるデータを内部で共有する場合があります。Red Hat は、以下の目的で Red Hat のビジネスパートナーと、お客様を特定しない集約された形式で Telemetry および設定データを共有する場合があります。つまり、パートナーが市場およびお客様の Red Hat のオファリングの使用をより良く理解できるように支援することを目的とするか、それらのパートナーと共同でサポートしている製品の統合を効果的に行うことを目的としています。

サードパーティー

Red Hat は、Telemetry および設定データの収集、分析、および保管を支援するために、特定のサードパーティーと連携する場合があります。

ユーザーコントロール/Telemetry および設定データ収集の有効化および無効化

リモートヘルスレポート の手順に従って、OpenShift Container Platform Telemetry と Insights Operator を無効にできます。

4.2. リモートヘルスモニタリングによって収集されるデータの表示

管理者は、Telemetry と Insights Operator によって収集されたメトリクスを確認できます。

4.2.1. Telemetry によって収集されるデータの表示

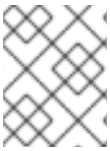
Telemetry でキャプチャーされるクラスターとコンポーネントの時系列データを表示することができます。

前提条件

- OpenShift Container Platform CLI (**oc**) をインストールしている。
- **cluster-admin** ロールまたは **cluster-monitoring-view** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. クラスターにログインします。
2. 次のコマンドを実行すると、クラスターの Prometheus サービスにクエリーが実行され、Telemetry によってキャプチャーされた時系列データの完全なセットが返されます。



注記

次の例には、OpenShift Container Platform on AWS に固有の値がいくつか含まれています。

```
$ curl -G -k -H "Authorization: Bearer $(oc whoami -t)" \
https://$(oc get route prometheus-k8s-federate -n \
openshift-monitoring -o jsonpath="{.spec.host}")/federate \
--data-urlencode 'match[]={__name__=~"cluster:usage:.*"}' \
--data-urlencode 'match[]={__name__="count:up0"}' \
```

```
--data-urlencode 'match[]={__name__="count:up1"}' \
--data-urlencode 'match[]={__name__="cluster_version"}' \
--data-urlencode 'match[]={__name__="cluster_version_available_updates"}' \
--data-urlencode 'match[]={__name__="cluster_version_capability"}' \
--data-urlencode 'match[]={__name__="cluster_operator_up"}' \
--data-urlencode 'match[]={__name__="cluster_operator_conditions"}' \
--data-urlencode 'match[]={__name__="cluster_version_payload"}' \
--data-urlencode 'match[]={__name__="cluster_installer"}' \
--data-urlencode 'match[]={__name__="cluster_infrastructure_provider"}' \
--data-urlencode 'match[]={__name__="cluster_feature_set"}' \
--data-urlencode 'match[]={__name__="instance:etcd_object_counts:sum"}' \
--data-urlencode 'match[]={__name__="ALERTS",alertstate="firing"}' \
--data-urlencode 'match[]={__name__="code:apiserver_request_total:rate:sum"}' \
--data-urlencode 'match[]={__name__="cluster:capacity_cpu_cores:sum"}' \
--data-urlencode 'match[]={__name__="cluster:capacity_memory_bytes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:cpu_usage_cores:sum"}' \
--data-urlencode 'match[]={__name__="cluster:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="openshift:cpu_usage_cores:sum"}' \
--data-urlencode 'match[]={__name__="openshift:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="workload:cpu_usage_cores:sum"}' \
--data-urlencode 'match[]={__name__="workload:memory_usage_bytes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:virt_platform_nodes:sum"}' \
--data-urlencode 'match[]={__name__="cluster:node_instance_type_count:sum"}' \
--data-urlencode 'match[]={__name__="cnv:vmi_status_running:count"}' \
--data-urlencode 'match[]={__name__="cluster:vmi_request_cpu_cores:sum"}' \
--data-urlencode 'match[]={__name__="node_role_os_version_machine:cpu_capacity_cores:sum"}' \
--data-urlencode 'match[]=
{__name__="node_role_os_version_machine:cpu_capacity_sockets:sum"}' \
--data-urlencode 'match[]={__name__="subscription_sync_total"}' \
--data-urlencode 'match[]={__name__="olm_resolution_duration_seconds"}' \
--data-urlencode 'match[]={__name__="csv_succeeded"}' \
--data-urlencode 'match[]={__name__="csv_abnormal"}' \
--data-urlencode 'match[]=
{__name__="cluster:kube_persistentvolumeclaim_resource_requests_storage_bytes:provisioner:sum"}'
\
--data-urlencode 'match[]={__name__="cluster:kubelet_volume_stats_used_bytes:provisioner:sum"}'
\
--data-urlencode 'match[]={__name__="ceph_cluster_total_bytes"}' \
--data-urlencode 'match[]={__name__="ceph_cluster_total_used_raw_bytes"}' \
--data-urlencode 'match[]={__name__="ceph_health_status"}' \
--data-urlencode 'match[]={__name__="odf_system_raw_capacity_total_bytes"}' \
--data-urlencode 'match[]={__name__="odf_system_raw_capacity_used_bytes"}' \
--data-urlencode 'match[]={__name__="odf_system_health_status"}' \
--data-urlencode 'match[]={__name__="job:ceph_osd_metadata:count"}' \
--data-urlencode 'match[]={__name__="job:kube_pv:count"}' \
--data-urlencode 'match[]={__name__="job:odf_system_pvs:count"}' \
--data-urlencode 'match[]={__name__="job:ceph_pools_iops:total"}' \
--data-urlencode 'match[]={__name__="job:ceph_pools_iops_bytes:total"}' \
--data-urlencode 'match[]={__name__="job:ceph_versions_running:count"}' \
--data-urlencode 'match[]={__name__="job:noobaa_total_unhealthy_buckets:sum"}' \
--data-urlencode 'match[]={__name__="job:noobaa_bucket_count:sum"}' \
--data-urlencode 'match[]={__name__="job:noobaa_total_object_count:sum"}' \
--data-urlencode 'match[]={__name__="odf_system_bucket_count", system_type="OCS",
system_vendor="Red Hat"}' \
--data-urlencode 'match[]={__name__="odf_system_objects_total", system_type="OCS",
system_vendor="Red Hat"}' \
```

```
--data-urlencode 'match[]={__name__="noobaa_accounts_num"}' \
--data-urlencode 'match[]={__name__="noobaa_total_usage"}' \
--data-urlencode 'match[]={__name__="console_url"}' \
--data-urlencode 'match[]={__name__="cluster:ovnkube_master_egress_routing_via_host:max"}' \
--data-urlencode 'match[]={__name__="cluster:network_attachment_definition_instances:max"}' \
--data-urlencode 'match[]=
{__name__="cluster:network_attachment_definition_enabled_instance_up:max"}' \
--data-urlencode 'match[]={__name__="cluster:ingress_controller_aws_nlb_active:sum"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:min"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:max"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:avg"}' \
--data-urlencode 'match[]={__name__="cluster:route_metrics_controller_routes_per_shard:median"}'
\
--data-urlencode 'match[]={__name__="cluster:openshift_route_info:tls_termination:sum"}' \
--data-urlencode 'match[]={__name__="insightsclient_request_send_total"}' \
--data-urlencode 'match[]={__name__="cam_app_workload_migrations"}' \
--data-urlencode 'match[]=
{__name__="cluster:apiserver_current_inflight_requests:sum:max_over_time:2m"}' \
--data-urlencode 'match[]={__name__="cluster:alertmanager_integrations:max"}' \
--data-urlencode 'match[]={__name__="cluster:telemetry_selected_series:count"}' \
--data-urlencode 'match[]={__name__="openshift:prometheus_tsdb_head_series:sum"}' \
--data-urlencode 'match[]=
{__name__="openshift:prometheus_tsdb_head_samples_appended_total:sum"}' \
--data-urlencode 'match[]={__name__="monitoring:container_memory_working_set_bytes:sum"}' \
--data-urlencode 'match[]={__name__="namespace_job:scrape_series_added:topk3_sum1h"}' \
--data-urlencode 'match[]=
{__name__="namespace_job:scrape_samples_post_metric_relabeling:topk3"}' \
--data-urlencode 'match[]={__name__="monitoring:haproxy_server_http_responses_total:sum"}' \
--data-urlencode 'match[]={__name__="rhmi_status"}' \
--data-urlencode 'match[]={__name__="status:upgrading:version:rhoam_state:max"}' \
--data-urlencode 'match[]={__name__="state:rhoam_critical_alerts:max"}' \
--data-urlencode 'match[]={__name__="state:rhoam_warning_alerts:max"}' \
--data-urlencode 'match[]={__name__="rhoam_7d_slo_percentile:max"}' \
--data-urlencode 'match[]={__name__="rhoam_7d_slo_remaining_error_budget:max"}' \
--data-urlencode 'match[]={__name__="cluster_legacy_scheduler_policy"}' \
--data-urlencode 'match[]={__name__="cluster_master_schedulable"}' \
--data-urlencode 'match[]={__name__="che_workspace_status"}' \
--data-urlencode 'match[]={__name__="che_workspace_started_total"}' \
--data-urlencode 'match[]={__name__="che_workspace_failure_total"}' \
--data-urlencode 'match[]={__name__="che_workspace_start_time_seconds_sum"}' \
--data-urlencode 'match[]={__name__="che_workspace_start_time_seconds_count"}' \
--data-urlencode 'match[]={__name__="cco_credentials_mode"}' \
--data-urlencode 'match[]={__name__="cluster:kube_persistentvolume_plugin_type_counts:sum"}' \
--data-urlencode 'match[]={__name__="visual_web_terminal_sessions_total"}' \
--data-urlencode 'match[]={__name__="acm_managed_cluster_info"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_vcenter_info:sum"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_esxi_version_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:vsphere_node_hw_version_total:sum"}' \
--data-urlencode 'match[]={__name__="openshift:build_by_strategy:sum"}' \
--data-urlencode 'match[]={__name__="rhods_aggregate_availability"}' \
--data-urlencode 'match[]={__name__="rhods_total_users"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_disk_wal_fsync_duration_seconds:histogram_quantile",quantile="0.99"}' \
--data-urlencode 'match[]={__name__="instance:etcd_mvcc_db_total_size_in_bytes:sum"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_network_peer_round_trip_time_seconds:histogram_quantile",quantile="0.99
```

```
"}' \
--data-urlencode 'match[]={__name__="instance:etcd_mvcc_db_total_size_in_use_in_bytes:sum"}' \
--data-urlencode 'match[]=
{__name__="instance:etcd_disk_backend_commit_duration_seconds:histogram_quantile",quantile="0.9
9"}' \
--data-urlencode 'match[]={__name__="appsvcs:cores_by_product:sum"}' \
--data-urlencode 'match[]={__name__="nto_custom_profiles:count"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_configmap"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_secret"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_mount_failures_total"}' \
--data-urlencode 'match[]={__name__="openshift_csi_share_mount_requests_total"}' \
--data-urlencode 'match[]={__name__="cluster:velero_backup_total:max"}' \
--data-urlencode 'match[]={__name__="cluster:velero_restore_total:max"}' \
--data-urlencode 'match[]={__name__="eo_es_storage_info"}' \
--data-urlencode 'match[]={__name__="eo_es_redundancy_policy_info"}' \
--data-urlencode 'match[]={__name__="eo_es_defined_delete_namespaces_total"}' \
--data-urlencode 'match[]={__name__="eo_es_misconfigured_memory_resources_info"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_data_nodes_total:max"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_documents_created_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:eo_es_documents_deleted_total:sum"}' \
--data-urlencode 'match[]={__name__="pod:eo_es_shards_total:max"}' \
--data-urlencode 'match[]={__name__="eo_es_cluster_management_state_info"}' \
--data-urlencode 'match[]={__name__="imageregistry:imagestreamtags_count:sum"}' \
--data-urlencode 'match[]={__name__="imageregistry:operations_count:sum"}' \
--data-urlencode 'match[]={__name__="log_logging_info"}' \
--data-urlencode 'match[]={__name__="log_collector_error_count_total"}' \
--data-urlencode 'match[]={__name__="log_forwarder_pipeline_info"}' \
--data-urlencode 'match[]={__name__="log_forwarder_input_info"}' \
--data-urlencode 'match[]={__name__="log_forwarder_output_info"}' \
--data-urlencode 'match[]={__name__="cluster:log_collected_bytes_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:log_logged_bytes_total:sum"}' \
--data-urlencode 'match[]={__name__="cluster:kata_monitor_running_shim_count:sum"}' \
--data-urlencode 'match[]={__name__="platform:hypershift_hostedclusters:max"}' \
--data-urlencode 'match[]={__name__="platform:hypershift_nodepools:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_unhealthy_bucket_claims:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_buckets_claims:max"}' \
--data-urlencode 'match[]=
{__name__="namespace:noobaa_unhealthy_namespace_resources:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_namespace_resources:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_unhealthy_namespace_buckets:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_namespace_buckets:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_accounts:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_usage:max"}' \
--data-urlencode 'match[]={__name__="namespace:noobaa_system_health_status:max"}' \
--data-urlencode 'match[]={__name__="ocs_advanced_feature_usage"}' \
--data-urlencode 'match[]={__name__="os_image_url_override:sum"}' \
--data-urlencode 'match[]={__name__="openshift:openshift_network_operator_ipsec_state:info"}'
```

4.2.2. Insights Operator によって収集されるデータの表示

Insights Operator で収集されるデータを確認することができます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Insights Operator の現在実行中の Pod の名前を検索します。

```
$ INSIGHTS_OPERATOR_POD=$(oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers  --field-selector=status.phase=Running)
```

2. Insights Operator で収集される最近のデータアーカイブをコピーします。

```
$ oc cp openshift-insights/$INSIGHTS_OPERATOR_POD:/var/lib/insights-operator ./insights-data
```

最近の Insights Operator アーカイブが **insights-data** ディレクトリーで利用可能になります。

4.3. リモートヘルスレポート

クラスターの健全性と使用状況データのレポートを **オプトイン** (有効化) または **オプトアウト** (無効化) することができます。

4.3.1. リモートヘルスレポートの有効化

組織がリモートヘルスレポートを無効にしている場合は、この機能を再度有効にできます。OpenShift Container Platform Web コンソールの **Overview** ページの **Status** タイルに表示される **Insights not available** というメッセージから、リモートヘルスレポートが無効になっていることを確認できます。

リモートヘルスレポートを有効にするには、新しい認可トークンを使用してグローバルクラスタープルシークレットを変更する必要があります。リモートヘルスレポートを有効にすると、Insights Operator と Telemetry の両方が有効になります。

4.3.2. グローバルクラスタープルシークレットの変更によるリモートヘルスレポートの有効化

既存のグローバルクラスタープルシークレットを変更して、リモートヘルスレポートを有効にすることができます。リモートヘルスモニタリングを無効にしている場合は、Red Hat OpenShift Cluster Manager から **console.openshift.com** アクセストークンを使用して新しいプルシークレットをダウンロードする必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Cluster Manager へのアクセス。

手順

1. Red Hat Hybrid Cloud Console の Downloads ページに移動します。
2. **Tokens → Pull secret** から、**Download** ボタンをクリックします。
   pull-secret ファイルには、JSON 形式の **cloud.openshift.com** アクセストークンが含まれています。

```
{
  "auths": {
```

```
    "cloud.openshift.com": {
      "auth": "<your_token>",
      "email": "<email_address>"
    }
  }
}
```

3. グローバルクラスタープルシークレットをローカルファイルシステムにダウンロードします。

```
$ oc get secret/pull-secret -n openshift-config \
  --template='{{index .data ".dockerconfigjson" | base64decode}}' \
  > pull-secret
```

4. プルシークレットのバックアップコピーを作成します。

```
$ cp pull-secret pull-secret-backup
```

5. テキストエディターで **pull-secret** ファイルを開きます。

6. 先ほどダウンロードした **pull-secret** ファイルの **cloud.openshift.com** JSON エントリーを **auths** ファイルに追加します。

7. ファイルを保存します。

8. 次のコマンドを実行して、クラスター内のシークレットを更新します。

```
$ oc set data secret/pull-secret -n openshift-config \
  --from-file=.dockerconfigjson=pull-secret
```

シークレットが更新され、クラスターがレポートを開始するまで、数分間待つ必要がある場合があります。

検証

1. OpenShift Container Platform Web コンソールから検証チェックを行うには、次の手順を実行します。

   a. OpenShift Container Platform Web コンソールの **Overview** ページに移動します。

   b. **Status** タイルの **Insights** セクションを確認します。このセクションに、検出された問題の件数が表示されます。

2. OpenShift CLI (**oc**) から検証チェックを行うには、次のコマンドを入力し、**status** パラメーターの値が **false** になっていることを確認します。

```
$ oc get co insights -o jsonpath='{.status.conditions[?(@.type=="Disabled")]}'
```

4.3.3. リモートヘルスレポートを無効にした場合の影響

OpenShift Container Platform では、お客様は使用状況情報のレポートを無効にできます。

リモートヘルスレポートを無効にする前に、接続クラスターの次の利点をご確認ください。

- Red Hat による問題への対応が迅速化し、カスタマーサポートが強化されます。

- 製品のアップグレードがクラスターに与える影響を Red Hat が把握できます。
- 接続クラスターは、サブスクリプションとエンタイトルメントのプロセスを簡素化するのに役立ちます。
- 接続クラスターにより、OpenShift Cluster Manager サービスがクラスターの概要とサブスクリプションステータスを提供できるようになります。

注記

実稼働前、テスト、および実稼働クラスターでは、健全性および使用状況のレポートを有効にしておくことを検討してください。これにより、Red Hat がお客様の環境における OpenShift Container Platform の適格性評価に参加し、製品の問題に対してより迅速に対応できるようになります。

接続クラスターでリモートヘルスレポートを無効にした場合の結果を以下に示します。

- サポートケースが作成されていない場合、Red Hat が製品のアップグレードの成功やクラスターの健全性を確認することができなくなります。
- Red Hat が、設定データを使用してカスタマーサポートケースの優先順位付けを改善することや、お客様にとって重要な設定を特定することができなくなります。
- OpenShift Cluster Manager が、健全性や使用状況の情報など、クラスターに関するデータを表示できなくなります。
- 使用状況の自動レポート機能を利用できないため、お客様が **console.redhat.com** Web コンソールにサブスクリプション情報を手動で入力する必要があります。

ネットワークが制限された環境でも、プロキシーの適切な設定により Telemetry と Insights のデータは収集されます。

4.3.4. リモートヘルスレポートの無効化

既存のグローバルクラスタープルシークレットを変更して、リモートヘルスレポートを無効にすることができます。この設定により、Telemetry と Insights Operator の両方が無効になります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. グローバルクラスタープルシークレットをローカルファイルシステムにダウンロードします。

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. テキストエディターで、ダウンロードした **.dockerconfigjson** ファイルを編集し、**cloud.openshift.com** JSON エントリーを削除します。

```
"cloud.openshift.com":{"auth":"<hash>","email":"<email_address>"}
```

3. ファイルを保存します。

4. クラスターのシークレットを更新します。詳細は、「グローバルクラスタープルシークレットの更新」を参照してください。
   クラスター内のシークレットが更新されるまで、数分間待つ必要がある場合があります。

4.3.5. 非接続クラスターの登録

リモートヘルスレポートの無効化によるクラスターへの影響を回避するために、非接続の OpenShift Container Platform クラスターを Red Hat Hybrid Cloud Console に登録してください。詳細は、「リモートヘルスレポートを無効にした場合の影響」を参照してください。

重要

お客様は非接続クラスターを登録することで、引き続きサブスクリプションの使用状況を Red Hat に報告できます。そうすることで、Red Hat はサブスクリプションに関連する正確な使用状況と容量の傾向を返せるようになります。その結果、お客様はその返された情報を使用して、すべてのリソースに対するサブスクリプションの割り当てをより適切に管理できるようになります。

前提条件

- OpenShift Container Platform Web コンソールに **cluster-admin** ロールとしてログインしている。
- Red Hat Hybrid Cloud Console にログインできる。

手順

1. Red Hat Hybrid Cloud Console の Register disconnected cluster Web ページに移動します。
2. オプション: Red Hat Hybrid Cloud Console のホームページから **Register disconnected cluster** Web ページにアクセスするには、ナビゲーションメニュー項目の **Cluster List** に移動し、**Register cluster** ボタンを選択します。
3. **Register disconnected cluster** ページの指定されたフィールドにクラスターの詳細を入力します。
4. このページの **Subscription settings** セクションから、ご使用の Red Hat サブスクリプションオファリングに適用するサブスクリプション設定を選択します。
5. 非接続クラスターを登録するには、**Register cluster** ボタンを選択します。

- サブスクリプションサービスでのサブスクリプションデータの表示方法 (サブスクリプションサービスのスタートガイド)

4.3.6. グローバルクラスタープルシークレットの更新

現在のプルシークレットを置き換えるか、新しいプルシークレットを追加することで、クラスターのグローバルプルシークレットを更新できます。

この手順は、インストール時に使用したレジストリーとは別のレジストリーにイメージを保存する必要がある場合に使用してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. オプション: 既存のプルシークレットに新しいプルシークレットを追加するには、以下の手順を実行します。

   a. 以下のコマンドを入力してプルシークレットをダウンロードします。

   ```
   $ oc get secret/pull-secret -n openshift-config --template='{{index .data
   ".dockerconfigjson" | base64decode}}' > <pull_secret_location> 1
   ```

   1 **<pull_secret_location>**: プルシークレットファイルへのパスを指定します。

   b. 以下のコマンドを実行して、新しいプルシークレットを追加します。

   ```
   $ oc registry login --registry="<registry>" \ 1
   --auth-basic="<username>:<password>" \ 2
   --to=<pull_secret_location> 3
   ```

   1 **<registry>**: 新しいレジストリーを指定します。同じレジストリー内に複数のリポジトリーを指定できます (例: **--registry="<registry/my-namespace/my-repository>**)。

   2 **<username>:<password>**: 新しいレジストリーの認証情報を指定します。

   3 **<pull_secret_location>**: プルシークレットファイルへのパスを指定します。

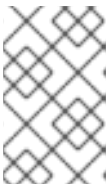
   プルシークレットファイルを手動で更新することもできます。

2. 以下のコマンドを実行して、クラスターのグローバルプルシークレットを更新します。

   ```
   $ oc set data secret/pull-secret -n openshift-config \
     --from-file=.dockerconfigjson=<pull_secret_location> 1
   ```

   1 **<pull_secret_location>**: 新しいプルシークレットファイルへのパスを指定します。

   この更新はすべてのノードにロールアウトされます。これには、クラスターのサイズに応じて多少時間がかかる場合があります。



注記

OpenShift Container Platform 4.7.4 の時点で、グローバルプルシークレットへの変更によってノードの drain (Pod の退避) の実行または再起動がトリガーされなくなりました。

関連情報

- クラスター所有権の移動

4.4. INSIGHTS を使用したクラスターの問題の特定

Insights は、Insights Operator から送信されるデータを繰り返し分析します。このデータには、Deployment Validation Operator (DVO) からのワークロード推奨事項も含まれます。OpenShift

Container Platform のユーザーは、Red Hat Hybrid Cloud Console の Insights Advisor サービスで結果を表示できます。

4.4.1. OpenShift Container Platform の Red Hat Insights Advisor について

Insights Advisor サービスを使用すると、OpenShift Container Platform クラスターの健全性を評価および監視できます。個々のクラスターとインフラストラクチャー全体のどちらに懸念があるかにかかわらず、サービスの可用性、フォールトトレランス、パフォーマンス、またはセキュリティーに影響を及ぼす可能性のある問題にさらされていることを認識することが重要です。

クラスターに Deployment Validation Operator (DVO) がインストールされている場合は、クラスターの健全性の問題につながる可能性がある設定を持つワークロードも推奨事項で示されます。

Insights による分析の結果は、Red Hat Hybrid Cloud Console の Insights Advisor サービスで確認できます。Red Hat Hybrid Cloud Console では、次のアクションを実行できます。

- 特定の推奨事項の影響を受けるクラスターとワークロードを表示する。
- 強力なフィルタリング機能を使用して、結果をその推奨事項だけに絞り込む。
- 個々の推奨事項とそれが示すリスクの詳細を確認し、個々のクラスターに合わせた解決策を確認する。
- 結果を他の関係者と共有する。

関連情報

- Red Hat Insights for OpenShift のワークフローで Deployment Validation Operator を使用する

4.4.2. Insights Advisor の推奨事項について

Insights Advisor サービスは、クラスターとワークロードのサービス可用性、フォールトトレランス、パフォーマンス、またはセキュリティーに悪影響を与える可能性のあるさまざまなクラスターの状態とコンポーネント設定に関する情報をまとめて提供します。この情報セットは、Insights Advisor サービスでは推奨事項と呼ばれます。クラスターの推奨事項には次の情報が含まれます。

- **Name:** 推奨事項の簡単な説明
- **Added:** 推奨事項が Insights Advisor サービスのアーカイブに公開されたタイミング
- **Category:** 問題がサービスの可用性、フォールトトレランス、パフォーマンス、またはセキュリティーに悪影響を及ぼす可能性があるかどうか
- **Total risk:** 状態がクラスターまたはワークロードに悪影響を与える **可能性** と、悪影響が発生した場合のシステム運用への **影響度** から算出された値
- **Clusters:** 推奨事項が検出されたクラスターのリスト
- **Description:** クラスターへの影響を含む、問題の簡単な概要

4.4.3. クラスターの潜在的な問題の表示

このセクションでは、OpenShift Cluster Manager の **Insights Advisor** に Insights レポートを表示する方法を説明します。

Insights はクラスターを繰り返し分析し、最新の結果を表示することに注意してください。問題を修正した場合や新しい問題が検出された場合などは、これらの結果が変化する可能性があります。

前提条件

- クラスターが OpenShift Cluster Manager に登録されている。
- リモートヘルスレポートが有効になっている (デフォルト)。
- OpenShift Cluster Manager にログインしている。

手順

1. OpenShift Cluster Manager で、**Advisor → Recommendations** に移動します。
   結果に応じて、Insights Advisor サービスは次のいずれかを表示します。
   - Insights で問題が特定されなかった場合は、**No matching recommendations found** が表示されます。
   - Insights が検出した問題のリストで、リスク (低、中、重要、および重大) ごとにグループ化されています。
   - Insights によるクラスター分析が行われていない場合は、**No clusters yet** が表示されます。分析は、クラスターがインストールされて登録され、インターネットに接続された直後に開始します。
2. 問題が表示された場合は、エントリーの前にある > アイコンをクリックして詳細を確認してください。
   問題によっては、Red Hat が提供する関連情報へのリンクがあります。

4.4.4. Insights Advisor サービスの推奨事項をすべて表示する

Recommendations ビューはデフォルトで、クラスターで検出された推奨事項のみを表示します。ただし、Advisor サービスのアーカイブにあるすべての推奨事項を表示することもできます。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- クラスターが Red Hat Hybrid Cloud Console に 登録 されています。
- OpenShift Cluster Manager にログインしている。

手順

1. OpenShift Cluster Manager で、**Advisor → Recommendations** に移動します。
2. **Clusters Impacted** フィルターおよび **Status** フィルターの横にある **X** アイコンをクリックします。
   これで、クラスターの潜在的な推奨事項をすべて参照できます。

4.4.5. Advisor の推奨事項のフィルター

場合によっては、Insights Advisor サービスによって多数の推奨事項が返されることがあります。最も重要な推奨事項に焦点を当てるために、Advisor の推奨事項 リストにフィルターを適用して、優先度の低い推奨事項を除外できます。

デフォルトでは、フィルターは 1 つ以上のクラスターに影響を与える有効な推奨事項のみを表示するように設定されています。Insights ライブラリー内のすべての推奨事項または無効化された推奨事項を表示するには、フィルターをカスタマイズできます。

フィルターを適用するには、フィルタータイプを選択し、ドロップダウンリストで使用できるオプションに基づき値を設定します。推奨事項のリストには、複数のフィルターを適用できます。

次のフィルタータイプを設定できます。

- **Name:** 名前で推奨事項を検索します。
- **Total risk:** クラスターに対する悪影響の可能性と重大度を示す値として、**Critical**、**Important**、**Moderate**、**Low** から 1 つ以上選択します。
- **Impact:** クラスター操作の継続性に対する潜在的な影響を示す値を、**Critical**、**High**、**Medium**、**Low** から 1 つ以上選択します。
- **Likelihood:** 推奨事項が実行された場合にクラスターに悪影響を及ぼす可能性を示す値を、**Critical**、**High**、**Medium**、**Low** から 1 つ以上選択します。
- **Category:** 注目するカテゴリーを、**Service Availability**、**Performance**、**Fault Tolerance**、**Security**、**Best Practice** から 1 つ以上選択します。
- **Status:** ラジオボタンをクリックして、有効な推奨事項 (デフォルト)、無効な推奨事項、またはすべての推奨事項を表示します。
- **Clusters impacted:** 現在 1 つ以上のクラスターに影響を与えている推奨事項、影響を与えていない推奨事項、またはすべての推奨事項を表示するようにフィルターを設定します。
- **Risk of change:** 解決策の実装がクラスター操作に及ぼす可能性のあるリスクを示す値を、**High**、**Moderate**、**Low**、**Very low** から 1 つ以上選択します。

4.4.5.1. Insights Advisor サービスの推奨事項のフィルタリング

OpenShift Container Platform クラスターマネージャーは、推奨事項リストに表示される推奨事項をフィルターできます。フィルターを適用すると、報告される推奨事項の数を減らし、最も優先度の高い推奨事項に集中できます。

次の手順は、**Category** フィルターの設定方法および削除方法を示していますが、この手順は任意のフィルタータイプおよびそれぞれの値にも適用できます。

前提条件

Hybrid Cloud Console の OpenShift Cluster Manager にログインしている。

手順

1. **OpenShift** > **Advisor** > **Recommendations** に移動します。
2. メインのフィルタータイプドロップダウンリストで、**Category** フィルタータイプを選択します。

3. フィルター値のドロップダウンリストを展開し、表示する推奨事項の各カテゴリー横にあるチェックボックスを選択します。不要なカテゴリーのチェックボックスはオフのままにします。
4. オプション: フィルターを追加して、リストをさらに絞り込みます。

選択したカテゴリーの推奨事項のみがリストに表示されます。

検証

- フィルターを適用した後、更新された推奨事項リストを表示できます。適用されたフィルターは、デフォルトのフィルターの隣に追加されます。

4.4.5.2. Insights Advisor サービスの推奨事項からフィルターを削除する

推奨事項のリストには、複数のフィルターを適用できます。準備が完了したフィルターは、個別に削除することも、完全にリセットすることもできます。

フィルターを個別に削除する

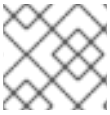
- デフォルトのフィルターを含め、各フィルターの横にある **X** アイコンをクリックすると、フィルターを個別に削除できます。

デフォルト以外のフィルターをすべて削除する

- **Reset filters** をクリックすると、適用したフィルターのみが削除され、デフォルトのフィルターはそのまま残ります。

4.4.6. Insights Advisor サービスの推奨事項を無効にする

クラスターに影響を与える特定の推奨事項を無効にして、それらがレポートに表示されないようにできます。単一のクラスターまたはすべてのクラスターの推奨を無効にできます。



注記

すべてのクラスターの推奨を無効にすると、今後のクラスターにも適用されます。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- クラスターが OpenShift Cluster Manager に登録されている。
- OpenShift Cluster Manager にログインしている。

手順

1. OpenShift Cluster Manager で、**Advisor → Recommendations** に移動します。
2. オプション: 必要に応じて、**Clusters Impacted** および **Status** フィルターを使用します。
3. 次のいずれかの方法でアラートを無効にします。
    - アラートを無効にするには、以下を実行します。

a. アラートの Options メニュー  をクリックし、**Disable recommendation** をクリックします。

b. 理由を入力し、**Save** をクリックします。

- アラートを無効にする前に、そのアラートの影響を受けるクラスターを表示するには、以下を実行します。

a. 無効にする推奨事項の名前をクリックします。その推奨事項のページに移動します。

b. **Affected clusters** セクションで、クラスターのリストを確認します。

c. **Actions → Disable recommendation** をクリックして、すべてのクラスターのアラートを無効にします。

d. 理由を入力し、**Save** をクリックします。

4.4.7. 以前に無効にした Insights Advisor サービスの推奨事項を有効にする

ある推奨事項をすべてのクラスターに対して無効にすると、その推奨事項は Insights Advisor サービスに表示されなくなります。この動作は変更できます。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- クラスターが OpenShift Cluster Manager に登録されている。
- OpenShift Cluster Manager にログインしている。

手順

1. OpenShift Cluster Manager で、**Advisor → Recommendations** に移動します。
2. 無効になっている推奨事項から、表示する推奨事項をフィルタリングします。
   a. **Status** ドロップダウンメニューから **Status** を選択します。
   b. **Filter by status** ドロップダウンメニューから、**Disabled** を選択します。
   c. オプション: **Clusters impacted** フィルターをクリアします。
3. 有効にする推奨事項を特定します。
4. Options メニュー  をクリックし、**Enable recommendation** をクリックします。

4.4.8. ワークロードに関する Insights Advisor サービスの推奨事項について

Red Hat Insights for OpenShift の Advisor サービスを使用すると、クラスターだけでなくワークロードにも影響する推奨事項に関する情報を表示および管理できます。Advisor サービスは、デプロイメント検証を活用し、OpenShift クラスターの管理者がデプロイメントポリシーに対する実行時の違反をすべて把握できるように支援します。ワークロードの推奨事項は、Red Hat Hybrid Cloud Console の OpenShift > Advisor > Workloads で確認できます。詳細は、次の関連情報を参照してください。

- Information about Kubernetes workloads
- Boost your cluster operations with Deployment Validation and Insights Advisor for Workloads
- クラスター内の namespace のワークロード推奨事項を特定する
- クラスター内の namespace のワークロード推奨事項を表示する
- クラスター内のワークロード推奨事項からオブジェクトを除外する

4.4.9. Web コンソールでの Insights ステータスの表示

Insights はクラスターを繰り返し分析し、OpenShift Container Platform Web コンソールでクラスターの特定された潜在的な問題のステータスを表示することができます。このステータスには、さまざまなカテゴリーの問題の数が表示され、詳細は OpenShift Cluster Manager のレポートへのリンクが表示されます。

前提条件

- クラスターが OpenShift Cluster Manager に 登録されている。
- リモートヘルスレポートが有効になっている (デフォルト)。
- OpenShift Container Platform Web コンソールにログインしている。

手順

1. OpenShift Container Platform Web コンソールで、**Home → Overview** に移動します。
2. **Status** カードの **Insights** をクリックします。
   ポップアップウィンドウには、リスクごとにグループ化された潜在的な問題がリスト表示されます。詳細を表示するには、個々のカテゴリーをクリックするか、**View all recommendations in Insights Advisor** を表示します。

4.5. INSIGHTS OPERATOR の使用

Insights Operator は設定およびコンポーネントの障害ステータスを定期的に収集し、デフォルトで 2 時間ごとにそのデータを Red Hat に報告します。この情報により、Red Hat は設定や Telemetry で報告されるデータよりも深層度の高いデータを評価できます。OpenShift Container Platform のユーザーは、Red Hat Hybrid Cloud Console の Insights Advisor サービスにレポートを表示できます。

関連情報

- Insights Operator はデフォルトでインストールされ、有効にされます。リモートヘルスレポートをオプトアウトする必要がある場合は、リモートヘルスレポート を参照してください。
- Insights Advisor サービスを使用してクラスターの問題を特定する方法の詳細は、Insights を使用したクラスターの問題の特定 を参照してください。

4.5.1. Insights Operator の設定

Insights Operator 設定は、デフォルトの Operator 設定と、**openshift-insights** namespace の **insights-config ConfigMap** オブジェクト、または **openshift-config** namespace のサポートシークレットのいずれかに保存されている設定を組み合わせたものです。

ConfigMap オブジェクトかサポートシークレットが存在する場合、含まれる属性値によってデフォルトの Operator 設定値がオーバーライドされます。**ConfigMap** オブジェクトとサポートシークレットが**両方とも** 存在する場合、Operator は **ConfigMap** オブジェクトを読み取ります。

ConfigMap オブジェクトはデフォルトでは存在しないため、OpenShift Container Platform クラスター管理者が作成する必要があります。

ConfigMap オブジェクトの設定の構造

この **insights-config ConfigMap** オブジェクトの例 (**config.yaml** 設定) は、標準の YAML 形式を使用した設定オプションを示しています。

```yaml
data:
  config.yaml: |
    dataReporting:
      uploadEndpoint: https://console.redhat.com/api/ingress/v1/upload,
      storagePath: /var/lib/insights-operator,
      downloadEndpoint: https://console.redhat.com/api/insights-results-aggregator/v2/cluster/%s/reports,
      conditionalGathererEndpoint: https://console.redhat.com/api/gathering/gathering_rules,
    sca:
      disable: false
      endpoint: https://api.openshift.com/api/accounts_mgmt/v1/certificates,
      interval: 8h
    alerting:
      disabled: false

```

設定可能な属性とデフォルト値

次の表は、使用可能な設定属性を示しています。

注記

insights-config ConfigMap オブジェクトは、標準の YAML 形式に準拠しています。子の値が親属性の下にあり、2 つのスペースでインデントされています。**Obfuscation** 属性では、親属性の子として箇条書きで値を入力します。

表4.1 Insights Operator の設定可能な属性

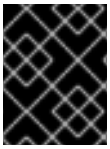
Attribute name	説明	値のタイプ	デフォルト値
alerting: disabled: false	クラスターの Prometheus インスタンスへの Insights Operator アラートを無効にします。	Boolean	**false**
clusterTransfer: endpoint: <url>	クラスター転送データを確認およびダウンロードするためのエンドポイント。	URL	https://api.openshift.com/api/accounts_mgmt/v1/cluster_transfers/
clusterTransfer: interval: 1h0m0s	利用可能なクラスター転送をチェックする頻度を設定します。	時間間隔	**24h**

Attribute name	説明	値のタイプ	デフォルト値
dataReporting: interval: 30m0s	データの収集とアップロードの頻度を設定します。	時間間隔	**2h**
dataReporting: uploadEndpoint: <url>	アップロードエンドポイントを設定します。	URL	https://console.redhat.com/api/ingress/v1/upload
dataReporting: storagePath: <path>	アーカイブされたデータが保存されるパスを設定します。	ファイルパス	/var/lib/insights-operator
dataReporting: downloadEndpoint: <url>	最新の Insights 分析をダウンロードするためのエンドポイントを指定します。	URL	https://console.redhat.com/api/ingress/v1/download
dataReporting: conditionalGathererEndpoint: <url>	条件付き収集ルール定義を提供するためのエンドポイントを設定します。	URL	https://console.redhat.com/api/gathering/gathering_rules
dataReporting: obfuscation: - networking	IP アドレスとクラスタードメイン名のグローバル難読化を有効にします。	String	該当なし
dataReporting: obfuscation: - workload_names	Data Validation Operator データの難読化を有効にします。クラスターリソースのリソース ID はアーカイブファイルでのみ表示され、リソース名は表示されません。	String	該当なし
proxy: httpProxy: http://example.com, httpsProxy: http://example.com, noProxy: test.org	Insights Operator のカスタムプロキシーを設定します。	URL	デフォルトなし

Attribute name	説明	値のタイプ	デフォルト値
`sca:` `  interval: 8h0m0s`	Simple Content Access (SCA) エンタイトルメントをダウンロードする頻度を指定します。	時間間隔	**2h**
`sca:` `  endpoint: <url>`	Simple Content Access (SCA) エンタイトルメントをダウンロードするためのエンドポイントを指定します。	URL	https://api.openshift.com/api/accounts_mgmt/v1/entitlement_certificates
`sca:` `  disabled: false`	Simple Content Access エンタイトルメントのダウンロードを無効にします。	Boolean	**false**

4.5.1.1. insights-config ConfigMap オブジェクトの作成

この手順では、Insights Operator によるカスタム設定を行うための **Insights-config ConfigMap** オブジェクトを作成する方法を説明します。



重要

デフォルトの Insights Operator 設定を変更する前に、Red Hat サポートに相談することを推奨します。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **Create ConfigMap** をクリックします。
3. **Configure via: YAML view** を選択し、次のように設定を入力します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: insights-config
  namespace: openshift-insights
```

```
data:
  config.yaml: |
    dataReporting:
      obfuscation:
        - networking
        - workload_names
    sca:
      disabled: false
      interval: 2h
    alerting:
      disabled: false
binaryData: {}
immutable: false
```

4. オプション: **Form view** を選択し、フォームで必要な情報を入力します。
5. **ConfigMap Name** フィールドに **insights-config** と入力します。
6. **Key** フィールドに **config.yaml** と入力します。
7. **Value** フィールドで、ファイルを探してフィールドにドラッグアンドドロップするか、設定パラメーターを手動で入力します。
8. **Create** をクリックすると、**ConfigMap** オブジェクトと設定情報が表示されます。

4.5.2. Insights Operator アラートについて

Insights Operator は、Prometheus モニタリングシステムを介して Alertmanager にアラートを宣言します。これらのアラートは、以下のいずれかの方法を使用して、OpenShift Container Platform Web コンソールのアラート UI で表示できます。

- **Administrator** パースペクティブで、**Observe → Alerting** をクリックします。
- **Developer** パースペクティブで、**Observe →** <project_name> **→ Alerts** タブをクリックします。

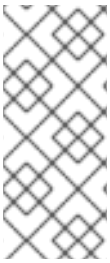
現在、Insights Operator は、条件が満たされたときに次のアラートを送信します。

表4.2 Insights Operator アラート

アラート	説明
InsightsDisabled	Insights Operator が無効になっています。
SimpleContentAccessNotAvailable	Red Hat Subscription Management で、Simple Content Access が有効になっていません。
InsightsRecommendationActive	Insights に、クラスターに関するアクティブな推奨事項があります。

4.5.2.1. Insights Operator アラートの無効化

Insights Operator がクラスター Prometheus インスタンスにアラートを送信しないようにするには、**insights-config ConfigMap** オブジェクトを作成または編集します。



注記

以前は、クラスター管理者は、**openshift-config** namespace の **サポートシークレット** を使用して Insights Operator 設定を作成または編集していました。Red Hat Insights が、Operator を設定するための **ConfigMap** オブジェクトの作成をサポートするようになりました。両方とも存在する場合、Operator はサポートシークレットよりも config map 設定を優先します。

insights-config ConfigMap オブジェクトが存在しない場合は、カスタム設定を初めて追加するときに作成する必要があります。**ConfigMap** オブジェクト内の設定は、**config/pod.yaml** ファイルで定義されているデフォルト設定よりも優先されることに注意してください。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- OpenShift Container Platform Web コンソールに **cluster-admin** としてログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。
3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. ファイル内で、**alerting** 属性を **disabled: true** に設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    alerting:
      disabled: true
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。
7. **config.yaml** の **alerting** 属性の値が **disabled: true** に設定されていることを確認します。

変更を保存すると、Insights Operator はクラスターの Prometheus インスタンスにアラートを送信しなくなります。

4.5.2.2. Insights Operator アラートの有効化

アラートを無効にすると、Insights Operator はクラスター Prometheus インスタンスにアラートを送信しなくなります。アラートは再度有効にできます。

注記

以前は、クラスター管理者は、**openshift-config** namespace の **サポートシークレット** を使用して Insights Operator 設定を作成または編集していました。Red Hat Insights が、Operator を設定するための **ConfigMap** オブジェクトの作成をサポートするようになりました。両方とも存在する場合、Operator はサポートシークレットよりも config map 設定を優先します。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- OpenShift Container Platform Web コンソールに **cluster-admin** としてログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。
3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. ファイル内で、**alerting** 属性を **disabled: false** に設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    alerting:
      disabled: false
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。
7. **config.yaml** の **alerting** 属性の値が **disabled: false** に設定されていることを確認します。

変更を保存すると、Insights Operator はクラスター Prometheus インスタンスにアラートを再度送信します。

4.5.3. Insights Operator アーカイブのダウンロード

Insights Operator は、収集したデータをクラスターの **openshift-insights** namespace にあるアーカイブに保存します。Insights Operator によって収集されたデータをダウンロードして確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Insights Operator の実行中の Pod の名前を見つけます。

```
$ oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers  --field-selector=status.phase=Running
```

2. Insights Operator で収集される最近のデータアーカイブをコピーします。

```
$ oc cp openshift-insights/<insights_operator_pod_name>:/var/lib/insights-operator ./insights-data (1)
```

(1) **<insights_operator_pod_name>** を、前のコマンドから出力された Pod 名に置き換えます。

最近の Insights Operator アーカイブが **insights-data** ディレクトリーで利用可能になります。

4.5.4. Insights Operator の収集操作の実行

Insights Operator データ収集操作をオンデマンドで実行できます。次の手順では、OpenShift Web コンソールまたは CLI を使用して収集操作のデフォルトのリストを実行する方法を説明します。オンデマンド収集機能をカスタマイズして、選択した収集操作を除外できます。デフォルトのリストからの収集操作を無効にすると、クラスターに効果的な推奨事項を提供する Insights Advisor の機能が低下します。クラスター内で Insights Operator の収集操作を無効にしていた場合は、この手順がそれらのパラメーターをオーバーライドします。

重要

DataGather カスタムリソースはテクノロジープレビューのみの機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

注記

クラスターでテクノロジープレビューを有効にすると、Insights Operator は個々の Pod で収集操作を実行します。これは、Insights Operator のテクノロジープレビュー機能セットの一部であり、新しいデータ収集機能をサポートします。

4.5.4.1. Insights Operator の収集期間の表示

Insights Operator がアーカイブに含まれる情報を収集する際にかかる時間を表示できます。これは、Insights Operator のリソースの使用状況と Insights Advisor の問題を理解する上で役立ちます。

前提条件

- Insights Operator アーカイブの最新のコピー。

手順

1. アーカイブから **/insights-operator/gathers.json** を開きます。
   このファイルには、Insights Operator 収集操作のリストが含まれています。

```
{
  "name": "clusterconfig/authentication",
  "duration_in_ms": 730, (1)
  "records_count": 1,
  "errors": null,
  "panic": null
}
```

(1) **duration_in_ms** は、各収集操作にかかるミリ秒単位の時間です。

2. 各収集操作に異常がないか検査します。

4.5.4.2. Web コンソールから Insights Operator の収集操作を実行する

データを収集するには、OpenShift Container Platform Web コンソールを使用して Insights Operator の収集操作を実行できます。

前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。

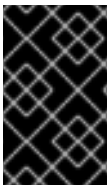
手順

1. コンソールで、**Administration → CustomResourceDefinitions** を選択します。
2. **CustomResourceDefinitions** ページの **Search by name** フィールドで **DataGather** リソース定義を見つけ、クリックします。
3. **CustomResourceDefinition details** ページで、**Instances** タブをクリックします。
4. **Create DataGather** をクリックします。
5. 新しい **DataGather** 操作を作成するには、次の設定ファイルを編集し、変更を保存します。

```
apiVersion: insights.openshift.io/v1alpha1
kind: DataGather
metadata:
  name: <your_data_gather> (1)
spec:
 gatherers: (2)
   - name: workloads
     state: Disabled
```

(1) **metadata** で、**<your_data_gather>** を収集操作用の一意の名前に置き換えます。

(2) **gatherers** で、無効にする収集操作をそれぞれ指定します。例では、無効化されたデータ収集操作は **workloads** のみで、他のすべてのデフォルト操作が実行されるように設定されています。**spec** パラメーターが空の場合は、デフォルトの収集操作がすべて実行されます。



重要

収集操作の名前に **periodic-gathering-** の接頭辞は追加しないでください。この文字列は他の管理操作用に予約されており、意図した収集操作に影響を及ぼす可能性があります。

検証

1. コンソールで **Workloads → Pods** を選択します。
2. Pods ページで **Project** プルダウンメニューに移動し、**Show default projects** を選択します。
3. **Project** プルダウンメニューから **openshift-insights** プロジェクトを選択します。
4. **openshift-insights** プロジェクトの Pod のリストで、新しい収集オペレーションに選択した名前の接頭辞が付いていることを確認します。完了すると、Insights Operator は処理のためにデータを Red Hat に自動的にアップロードします。

4.5.4.3. OpenShift CLI から Insights Operator 収集操作を実行する

OpenShift Container Platform コマンドラインインターフェイスを使用して、Insights Operator の収集操作を実行できます。

前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform にログインしている。

手順

- 次のコマンドを入力して、収集操作を実行します。

```
$ oc apply -f <your_datagather_definition>.yaml
```

<your_datagather_definition>.yaml を、以下のパラメーターが含まれる設定ファイルに置き換えます。

```
apiVersion: insights.openshift.io/v1alpha1
kind: DataGather
metadata:
  name: <your_data_gather> (1)
spec:
 gatherers: (2)
   - name: workloads
     state: Disabled
```

(1) **metadata** で、**<your_data_gather>** を収集操作用の一意の名前に置き換えます。

(2) **gatherers** で、無効にする収集操作をそれぞれ指定します。例では、無効化されたデータ収集操作は **workloads** のみで、他のすべてのデフォルト操作が実行されるように設定されています。**spec** パラメーターが空の場合は、デフォルトの収集操作がすべて実行されます。

重要

収集操作の名前に **periodic-gathering-** の接頭辞は追加しないでください。この文字列は他の管理操作用に予約されており、意図した収集操作に影響を及ぼす可能性があります。

検証

- **openshift-insights** プロジェクトの Pod のリストで、新しい収集オペレーションに選択した名前の接頭辞が付いていることを確認します。完了すると、Insights Operator は処理のためにデータを Red Hat に自動的にアップロードします。

関連情報

- Insights Operator Gathered Data GitHub repository

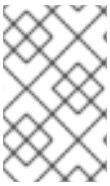
4.5.4.4. Insights Operator の収集操作の無効化

Insights Operator の収集操作を無効にすることができます。収集操作を無効にすると、Insights Operator が Insights クラスターレポートを収集して Red Hat に送信しなくなるため、組織のプライバシーを高めることができます。これにより、クラスター転送などの Red Hat との通信を必要とする他のコア機能に影響を与えることなく、クラスターの Insights 分析と推奨事項が無効になります。Insights Operator アーカイブの **/insights-operator/gathers.json** ファイルから、クラスターに対して試行された収集操作のリストを表示できます。一部の収集操作は、特定の条件が満たされた場合にのみ発生し、最新のアーカイブには表示されない可能性があることに注意してください。

重要

InsightsDataGather カスタムリソースは、テクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。



注記

クラスターでテクノロジープレビューを有効にすると、Insights Operator は個々の Pod で収集操作を実行します。これは、Insights Operator のテクノロジープレビュー機能セットの一部であり、新しいデータ収集機能をサポートします。

前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。

手順

1. **Administration → CustomResourceDefinitions** に移動します。

2. **CustomResourceDefinitions** ページで、**Search by name** フィールドを使用して **InsightsDataGather** リソース定義を見つけてクリックします。

3. **CustomResourceDefinition details** ページで、**Instances** タブをクリックします。

4. **cluster** をクリックし、**YAML** タブをクリックします。

5. **InsightsDataGather** 設定ファイルに対して次のいずれかの編集を実行して、収集操作を無効にします。

   a. すべての収集操作を無効にするには、**disabledGatherers** キーの下に **all** を入力します。

```
apiVersion: config.openshift.io/v1alpha1
kind: InsightsDataGather
metadata:
....

spec: (1)
  gatherConfig:
    disabledGatherers:
      - all (2)
```

(1) **spec** パラメーターは、収集設定を指定します。

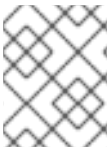
(2) **all** 値は、すべての収集操作を無効にします。

   b. 個々の収集操作を無効にするには、**disabledGatherers** キーの下に値を入力します。

```
spec:
  gatherConfig:
    disabledGatherers:
      - clusterconfig/container_images (1)
      - clusterconfig/host_subnets
      - workloads/workload_info
```

(1) 個別収集操作の例

6. **Save** をクリックします。
   変更を保存すると、Insights Operator の収集設定が更新され、操作は行われなくなります。

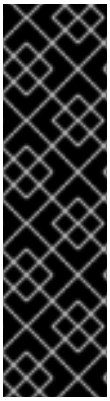


注記

収集操作を無効にすると、クラスターに効果的な推奨事項を提供する Insights Advisor サービスの機能が低下します。

4.5.4.5. Insights Operator の収集操作の有効化

収集操作が無効になっている場合は、Insights Operator の収集操作を有効にできます。



重要

InsightsDataGather カスタムリソースは、テクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。

手順

1. **Administration** → **CustomResourceDefinitions** に移動します。
2. **CustomResourceDefinitions** ページで、**Search by name** フィールドを使用して **InsightsDataGather** リソース定義を見つけてクリックします。
3. **CustomResourceDefinition details** ページで、**Instances** タブをクリックします。
4. **cluster** をクリックし、**YAML** タブをクリックします。
5. 次のいずれかの編集を実行して、収集操作を有効にします。
   - 無効になっているすべての収集操作を有効にするには、**gatherConfig** スタンザを削除します。

```
apiVersion: config.openshift.io/v1alpha1
kind: InsightsDataGather
metadata:
....

spec:
  gatherConfig: 1
    disabledGatherers: all
```

1 すべての収集操作を有効にするには、**gatherConfig** スタンザを削除します。

   - 個々の収集操作を有効にするには、**disabledGatherers** キーの下の値を削除します。

```
spec:
  gatherConfig:
    disabledGatherers:
      - clusterconfig/container_images 1
      - clusterconfig/host_subnets
      - workloads/workload_info
```

1 1 つ以上の収集操作を削除します。

6. **Save** をクリックします。
   変更を保存すると、Insights Operator の収集設定が更新され、その影響を受けた収集操作が開始します。

注記

収集操作を無効にすると、クラスターに効果的な推奨事項を提供する Insights Advisor の機能が低下します。

4.5.5. Deployment Validation Operator のデータの難読化

デフォルトでは、Deployment Validation Operator (DVO) をインストールすると、OpenShift Container Platform の Insights Operator によって収集および処理されるデータに、リソースの名前と一意の識別子 (UID) が追加されます。クラスター管理者の場合は、Insights Operator を設定して、Deployment Validation Operator (DVO) からのデータを難読化することができます。たとえば、アーカイブファイル内のワークロード名を難読化して、Red Hat に送信することができます。

リソースの名前を難読化するには、次の手順で説明するように、**insights-config ConfigMap** オブジェクトの **obfuscation** 属性を手動で設定し、**workload_names** 値を含める必要があります。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- "cluster-admin" ロールを使用して OpenShift Container Platform Web コンソールにログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。
- クラスターがセルフマネージドであり、Deployment Validation Operator がインストールされている。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。
3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. ファイル内で、**workload_names** 値を使用して **obfuscation** 属性を設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    dataReporting:
      obfuscation:
        - workload_names
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。

7. **config.yaml** の **obfuscation** 属性の値が **- workload_names** に設定されていることを確認します。

4.6. 限定的なネットワーク環境でのリモートヘルスレポートの使用

Insights Operator アーカイブを手動で収集し、アップロードして限定的なネットワーク環境から問題を診断できます。

ネットワークが制限された環境で Insights Operator を使用するには、以下を行う必要があります。

- Insights Operator アーカイブのコピーを作成します。
- Insights Operator アーカイブを console.redhat.com にアップロードします。

さらに、アップロード前に Insights Operator データを 難読化 することを選択できます。

4.6.1. Insights Operator の収集操作の実行

Insights Operator アーカイブを作成するには、収集操作を実行する必要があります。

前提条件

- **cluster-admin** として OpenShift Container Platform にログインしている。

手順

1. 以下のテンプレートを使用して、**gather-job.yaml** という名前のファイルを作成します。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: insights-operator-job
  annotations:
    config.openshift.io/inject-proxy: insights-operator
spec:
  backoffLimit: 6
  ttlSecondsAfterFinished: 600
  template:
    spec:
      restartPolicy: OnFailure
      serviceAccountName: operator
      nodeSelector:
        beta.kubernetes.io/os: linux
        node-role.kubernetes.io/master: ""
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
        operator: Exists
      - effect: NoExecute
        key: node.kubernetes.io/unreachable
        operator: Exists
        tolerationSeconds: 900
      - effect: NoExecute
        key: node.kubernetes.io/not-ready
        operator: Exists
```

```
      tolerationSeconds: 900
    volumes:
    - name: snapshots
      emptyDir: {}
    - name: service-ca-bundle
      configMap:
        name: service-ca-bundle
        optional: true
    initContainers:
    - name: insights-operator
      image: quay.io/openshift/origin-insights-operator:latest
      terminationMessagePolicy: FallbackToLogsOnError
      volumeMounts:
      - name: snapshots
        mountPath: /var/lib/insights-operator
      - name: service-ca-bundle
        mountPath: /var/run/configmaps/service-ca-bundle
        readOnly: true
      ports:
      - containerPort: 8443
        name: https
      resources:
        requests:
          cpu: 10m
          memory: 70Mi
      args:
      - gather
      - -v=4
      - --config=/etc/insights-operator/server.yaml
    containers:
      - name: sleepy
        image: quay.io/openshift/origin-base:latest
        args:
          - /bin/sh
          - -c
          - sleep 10m
        volumeMounts: [{name: snapshots, mountPath: /var/lib/insights-operator}]
```

2. **insights-operator** イメージバージョンをコピーします。

```
$ oc get -n openshift-insights deployment insights-operator -o yaml
```

出力例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: insights-operator
  namespace: openshift-insights
# ...
spec:
  template:
# ...
    spec:
      containers:
```

```
    - args:
# ...
      image: registry.ci.openshift.org/ocp/4.15-2023-10-12-
212500@sha256:a0aa581400805ad0... 1
# ...
```

1 **insights-operator** イメージのバージョンを指定します。

3. **gather-job.yaml** でイメージのバージョンを貼り付けます。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: insights-operator-job
# ...
spec:
# ...
  template:
    spec:
    initContainers:
    - name: insights-operator
      image: image: registry.ci.openshift.org/ocp/4.15-2023-10-12-
212500@sha256:a0aa581400805ad0... 1
      terminationMessagePolicy: FallbackToLogsOnError
      volumeMounts:
```

1 既存の値を **insights-operator** イメージのバージョンに置き換えます。

4. 収集ジョブを作成します。

```
$ oc apply -n openshift-insights -f gather-job.yaml
```

5. ジョブ Pod の名前を見つけます。

```
$ oc describe -n openshift-insights job/insights-operator-job
```

出力例

```
Name:             insights-operator-job
Namespace:        openshift-insights
# ...
Events:
  Type    Reason            Age    From            Message
  ----    ------            ----   ----            -------
  Normal  SuccessfulCreate  7m18s  job-controller  Created pod: insights-operator-job-
<your_job>
```

ここでは、以下のようになります。

insights-operator-job-<your_job> は Pod の名前です。

6. 操作が完了したことを確認します。

```
$ oc logs -n openshift-insights insights-operator-job-<your_job> insights-operator
```

出力例

```
I0407 11:55:38.192084       1 diskrecorder.go:34] Wrote 108 records to disk in 33ms
```

7. 作成したアーカイブを保存します。

```
$ oc cp openshift-insights/insights-operator-job-<your_job>:/var/lib/insights-operator
./insights-data
```

8. ジョブをクリーンアップします。

```
$ oc delete -n openshift-insights job insights-operator-job
```

4.6.2. Insights Operator アーカイブのアップロード

Insights Operator アーカイブを console.redhat.com に手動でアップロードし、潜在的な問題を診断できます。

前提条件

- **cluster-admin** として OpenShift Container Platform にログインしている。
- 制限なくインターネットアクセスができるワークステーションがある。
- Insights Operator アーカイブのコピーを作成している。

手順

1. **dockerconfig.json** ファイルをダウンロードします。

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. **dockerconfig.json** ファイルから **"cloud.openshift.com"** の **"auth"** トークンをコピーします。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<your_token>",
      "email": "asd@redhat.com"
    }
}
```

3. console.redhat.com にアーカイブをアップロードします。

```
$ curl -v -H "User-Agent: insights-operator/one10time200gather184a34f6a168926d93c330
cluster/<cluster_id>" -H "Authorization: Bearer <your_token>" -F
"upload=@<path_to_archive>; type=application/vnd.redhat.openshift.periodic+tar"
https://console.redhat.com/api/ingress/v1/upload
```

ここで、**<cluster_id>** はクラスター ID、**<your_token>** はプルシークレットからのトークン、**<path_to_archive>** は Insights Operator アーカイブへのパスに置き換えます。

操作に成功すると、コマンドは **"request_id"** と **"account_number"** を返します。

出力例

```
* Connection #0 to host console.redhat.com left intact
{"request_id":"393a7cf1093e434ea8dd4ab3eb28884c","upload":
{"account_number":"6274079"}}%
```

検証手順

1. https://console.redhat.com/openshift にログインします。
2. 左側のペインの **Cluster List** メニューをクリックします。
3. クラスターの詳細を表示するには、クラスターの名前をクリックします。
4. クラスターの **Insights Advisor** タブを開きます。
   アップロードに成功すると、タブには以下のいずれかが表示されます。
   - **Your cluster passed all recommendations** Insights Advisor で何も問題が特定されなかった場合。
   - Insights Advisor が検出した問題。リスク (低、中、重要および重大) 別に優先度が付けられます。

4.6.3. Insights Operator データの難読化の有効化

難読化を有効にして、機密性が高く、識別可能な IPv4 アドレスとクラスターベースドメインをマスクし、Insights Operator が console.redhat.com に送信できるようにします。

警告

この機能は利用可能ですが、Red Hat ではサポートサービスをより効果的に行えるように、難読化を無効のままにすることを推奨します。

難読化は、識別されていない値をクラスター IPv4 アドレスに割り当て、メモリーに保持される変換テーブルを使用して、データを console.redhat.com にアップロードする前に、Insights Operator アーカイブ全体で IP アドレスを難読化バージョンに変更します。

クラスターベースドメインでは、難読化により、ベースドメインがハードコードされたサブ文字列に変更されます。たとえば、**cluster-api.openshift.example.com** は **cluster-api.<CLUSTER_BASE_DOMAIN>** になります。

以下の手順では、**openshift-config** namespace で **support** シークレットを使用して難読化を有効にします。

前提条件

- OpenShift Container Platform Web コンソールに **cluster-admin** としてログインしている。

手順

1. **Workloads** → **Secrets** に移動します。
2. **openshift-config** プロジェクトを選択します。
3. **Search by name** フィールドを使用して **support** シークレットを検索します。存在しない場合には、**Create** → **Key/value secret** をクリックして作成します。
4. Options メニュー をクリックしてから **Edit Secret** をクリックします。
5. **Add Key/Value** をクリックします。
6. 値が **true** の **enableGlobalObfuscation** という名前のキーを作成し、**Save** をクリックします。
7. **Workloads** → **Pods** に移動します。
8. **openshift-insights** プロジェクトを選択します。
9. **insights-operator** Pod を検索します。
10. **insights-operator** Pod を再起動するには、Options メニュー をクリックしてから **Delete Pod** をクリックします。

検証

1. **Workloads** → **Secrets** に移動します。
2. **openshift-insights** プロジェクトを選択します。
3. **Search by name** フィールドを使用して **obfuscation-translation-table** シークレットを検索します。

obfuscation-translation-table シークレットが存在する場合は、難読化が有効になって機能します。

または、Insights Operator アーカイブの **/insights-operator/gathers.json** で **"is_global_obfuscation_enabled": true** の値を確認できます。

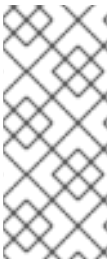
関連情報

- Insights Operator アーカイブのダウンロード方法の詳細は、Insights Operator によって収集されるデータの表示 を参照してください。

4.7. INSIGHTS OPERATOR を使用した SIMPLE CONTENT ACCESS エンタイトルメントのインポート

Insights Operator は、OpenShift Cluster Manager から Simple Content Access エンタイトルメントを定期的にインポートして **openshift-config-managed** namespace の **etc-pki-entitlement** シークレット

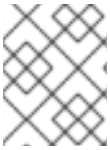
に保存します。Simple Content Access は Red Hat のサブスクリプションツールの機能で、エンタイトルメントツールの動作を簡素化します。この機能は、サブスクリプションツールを設定する複雑さを伴わずに、Red Hat のサブスクリプションが提供するコンテンツを簡単に利用できます。



注記

以前は、クラスター管理者は、**openshift-config** namespace の **サポートシークレット** を使用して Insights Operator 設定を作成または編集していました。Red Hat Insights が、Operator を設定するための **ConfigMap** オブジェクトの作成をサポートするようになりました。両方とも存在する場合、Operator はサポートシークレットよりも config map 設定を優先します。

Insights Operator は 8 時間ごとに Simple Content Access エンタイトルメントをインポートしますが、**openshift-insights** namespace の **insights-config ConfigMap** オブジェクトを使用すると、インポートを設定または無効にすることができます。



注記

インポートを機能させるには、Red Hat Subscription Management で Simple Content Access を有効にする必要があります。

関連情報

- Simple Content Access の詳細は、Red Hat Subscription Central ドキュメントの Simple Content Access を参照してください。
- OpenShift Container Platform のビルドで Simple Content Access エンタイトルメントを使用する場合の詳細は、ビルドでの Red Hat サブスクリプションの使用 を参照してください。

4.7.1. Simple Content Access のインポート間隔の設定

openshift-insights namespace の **insights-config ConfigMap** オブジェクトを使用して、Insights Operator が Simple Content Access (SCA) エンタイトルメントをインポートする頻度を設定できます。エンタイトルメントのインポートは通常 8 時間ごとに行われますが、**insights-config ConfigMap** オブジェクトで Simple Content Access の設定を更新すると、SCA のこの間隔を短くすることができます。

この手順では、インポート間隔を 2 時間 (2h) に更新する方法を説明します。時間 (h) または時間と分を指定できます (例: 2h30m)。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。

3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. コンテンツを 2 時間ごとにインポートするには、ファイルの **sca** 属性を **interval: 2h** に設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    sca:
      interval: 2h
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。
7. **config.yaml** の **sca** 属性の値が **interval: 2h** に設定されていることを確認します。

4.7.2. Simple Content Access インポートの無効化

openshift-insights namespace の **insights-config ConfigMap** オブジェクトを使用して、Simple Content Access エンタイトルメントのインポートを無効にできます。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- OpenShift Container Platform Web コンソールに **cluster-admin** としてログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。
3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. ファイルで、**sca** 属性を **disabled: true** に設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    sca:
      disabled: true
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。

7. **config.yaml** の **sca** 属性の値が **disabled: true** に設定されていることを確認します。

4.7.3. 無効にしていたシンプルコンテンツアクセスインポートの有効化

シンプルコンテンツアクセスエンタイトルメントのインポートが無効になっている場合、Insights Operator はシンプルコンテンツアクセスエンタイトルメントをインポートしません。この動作は変更できます。

前提条件

- リモートヘルスレポートが有効になっている (デフォルト)。
- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインしている。
- **insights-config ConfigMap** オブジェクトが、**openshift-insights** namespace に存在する。

手順

1. **Workloads → ConfigMaps** に移動し、**Project: openshift-insights** を選択します。
2. **insights-config ConfigMap** オブジェクトをクリックして開きます。
3. **Actions** をクリックし、**Edit ConfigMap** を選択します。
4. **YAML view** のラジオボタンをクリックします。
5. ファイル内で、**sca** 属性を **disabled: false** に設定します。

```
apiVersion: v1
kind: ConfigMap
# ...
data:
  config.yaml: |
    sca:
      disabled: false
# ...
```

6. **Save** をクリックします。**insights-config** config-map の詳細ページが開きます。
7. **config.yaml** の **sca** 属性の値が **disabled: false** に設定されていることを確認します。

第5章 クラスターに関するデータの収集

サポートケースを作成する際、ご使用のクラスターに関するデバッグ情報を Red Hat サポートに提供していただくと Red Hat のサポートに役立ちます。

以下を提供することが推奨されます。

- **oc adm must-gather** コマンドを使用して収集されるデータ
- 一意のクラスター ID

5.1. MUST-GATHER ツールについて

oc adm must-gather CLI コマンドは、以下のような問題のデバッグに必要となる可能性のあるクラスターからの情報を収集します。

- リソース定義
- サービスログ

デフォルトで、**oc adm must-gather** コマンドはデフォルトのプラグインイメージを使用し、**./must-gather.local** に書き込みを行います。

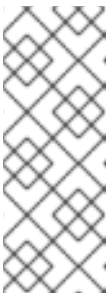
または、以下のセクションで説明されているように、適切な引数を指定してコマンドを実行すると、特定の情報を収集できます。

- 1 つ以上の特定の機能に関連するデータを収集するには、以下のセクションに示すように、イメージと共に **--image** 引数を使用します。
  以下に例を示します。

```
$ oc adm must-gather \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.18.17
```

  - 監査ログを収集するには、以下のセクションで説明されているように **--/usr/bin/gather_audit_logs** 引数を使用します。
    以下に例を示します。

```
$ oc adm must-gather -- /usr/bin/gather_audit_logs
```



注記

- ファイルのサイズを小さくするために、監査ログはデフォルトの情報セットの一部として収集されません。
- Windows オペレーティングシステムでは、**oc rsync** コマンドで使用するために **cwRsync** クライアントをインストールし、**PATH** 変数に追加します。

oc adm must-gather を実行すると、ランダムな名前を持つ新規 Pod がクラスターの新規プロジェクトに作成されます。データはその Pod 上で収集され、現在の作業ディレクトリー内の **must-gather.local** で始まる新しいディレクトリーに保存されます。

以下に例を示します。

```
NAMESPACE                  NAME             READY   STATUS      RESTARTS      AGE
...
openshift-must-gather-5drcj    must-gather-bklx4    2/2     Running     0          72s
openshift-must-gather-5drcj    must-gather-s8sdh    2/2     Running     0          72s
...
```

任意で、**--run-namespace** オプションを使用して、特定の namespace で **oc adm must-gather** コマンドを実行できます。

以下に例を示します。

```
$ oc adm must-gather --run-namespace <namespace> \
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.18.17
```

5.1.1. Red Hat サポート用のクラスターに関するデータの収集

oc adm must-gather CLI コマンドを使用して、クラスターに関するデバッグ情報を収集できます。

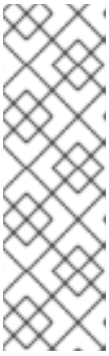
セルフマネージドのホステッドクラスターをデバッグするための情報を収集する場合は、「Hosted Control Plane のトラブルシューティング用の情報収集」を参照してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform CLI (**oc**) がインストールされている。

手順

1. **must-gather** データを保存するディレクトリーに移動します。



注記

クラスターが非接続環境にある場合は、追加の手順を実行する必要があります。ミラーレジストリーに信頼される CA がある場合、まず信頼される CA をクラスターに追加する必要があります。非接続環境のすべてのクラスターに対して、デフォルトの **must-gather** イメージをイメージストリームとしてインポートする必要があります。

```
$ oc import-image is/must-gather -n openshift
```

2. **oc adm must-gather** コマンドを実行します。

```
$ oc adm must-gather
```

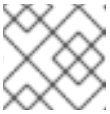
重要

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

注記

このコマンドは、デフォルトでランダムなコントロールプレーンノードを選択するため、Pod は **NotReady** および **SchedulingDisabled** 状態のコントロールプレーンノードにスケジュールされる場合があります。

a. このコマンドが失敗する場合 (クラスターで Pod をスケジュールできない場合など)、**oc adm inspect** コマンドを使用して、特定リソースに関する情報を収集します。



注記

収集する推奨リソースは、Red Hat サポートにお問い合わせください。

3. 作業ディレクトリーに作成された **must-gather** ディレクトリーから圧縮ファイルを作成します。固有の must-gather データの日付とクラスター ID を必ず提供してください。クラスター ID を確認する方法の詳細は、How to find the cluster-id or name on OpenShift cluster を参照してください。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
<must_gather_local_dir> 1
```

1 **<must_gather_local_dir>** は、実際のディレクトリー名に置き換えます。

4. Red Hat カスタマーポータルの **カスタマーサポート** ページ で、圧縮ファイルをサポートケースに添付します。

関連情報

- Hosted Control Plane のトラブルシューティング用の情報収集

5.1.2. must-gather フラグ

次の表にリストされているフラグは、**oc adm must-gather** コマンドで使用できます。

表5.1 oc adm must-gather の OpenShift Container Platform フラグ

フラグ	コマンドの例	説明
--all-images	**oc adm must-gather --all-images=false**	クラスター上の、**operators.openshift.io/must-gather-image** でアノテーションが付けられたすべての Operator のデフォルトイメージを使用して、**must-gather** データを収集します。
--dest-dir	**oc adm must-gather --dest-dir='<directory_name>'**	収集されたデータが書き込まれるローカルマシン上の特定のディレクトリーを設定します。

フラグ	コマンドの例	説明
--host-network	**oc adm must-gather --host-network=false**	**must-gather** Pod を **hostNetwork: true** として実行します。特定のコマンドとイメージでホストレベルのデータをキャプチャーする必要がある場合に関連します。
--image	**oc adm must-gather --image=[<plugin_image>]**	実行する **must-gather** プラグインイメージを指定します。指定されていない場合は、OpenShift Container Platform のデフォルトの **must-gather** イメージが使用されます。
--image-stream	**oc adm must-gather --image-stream=[<image_stream>]**	実行する **must-gather** プラグインイメージを含む namespace または name:tag 値を使用して、`<image_stream>` を指定します。
--node-name	**oc adm must-gather --node-name='<node>'**	使用する特定のノードを設定します。指定しない場合は、デフォルトでランダムマスターが使用されます。
--node-selector	**oc adm must-gather --node-selector='<node_selector_name>'**	使用する特定のノードセレクターを設定します。クラスターノードのセットで同時にデータをキャプチャーする必要があるコマンドとイメージを指定する場合にのみ関連します。
--run-namespace	**oc adm must-gather --run-namespace='<namespace>'**	**must-gather** Pod を実行する既存の特権 namespace。指定しない場合は、一時的な namespace が生成されます。
--since	**oc adm must-gather --since=<time>**	指定された期間よりも新しいログのみを返します。デフォルトはすべてのログです。プラグインが推奨されますが、このサポートには必要はありません。**since-time** または **since** を 1 つだけ使用できます。
--since-time	**oc adm must-gather --since-time='<date_and_time>'**	(RFC3339) 形式で表現された特定の日時以降のログのみを返します。デフォルトはすべてのログです。プラグインが推奨されますが、このサポートには必要はありません。**since-time** または **since** を 1 つだけ使用できます。
--source-dir	**oc adm must-gather --source-dir='/<directory_name>/'**	収集したデータをコピーする Pod 上の特定のディレクトリーを設定します。

フラグ	コマンドの例	説明
--timeout	**oc adm must-gather --timeout='<time>'**	タイムアウトする前にデータを収集する時間の長さ。秒、分、時間で表されます (例: 3s、5m、2h)。指定する時間は 0 より大きい必要があります。指定されていない場合はデフォルトで 10 分になります。
--volume-percentage	**oc adm must-gather --volume-percentage=<percent>**	**must-gather** に使用できる Pod の割り当てボリュームの最大パーセンテージを指定します。この制限を超えると、**must-gather** は収集を停止しますが、収集されたデータはコピーし続けます。指定されていない場合はデフォルトで 30% になります。

5.1.3. 特定の機能に関するデータ収集

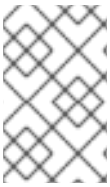
oc adm must-gather CLI コマンドを **--image** または **--image-stream** 引数と共に使用して、特定に機能に関するデバッグ情報を収集できます。**must-gather** ツールは複数のイメージをサポートするため、単一のコマンドを実行して複数の機能に関するデータを収集できます。

表5.2 サポート対象の must-gather イメージ

イメージ	目的
registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.18.17	OpenShift Virtualization のデータ収集。
registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8	OpenShift Serverless のデータ収集。
registry.redhat.io/openshift-service-mesh/istio-must-gather-rhel8:<installed_version_service_mesh>	Red Hat OpenShift Service Mesh のデータ収集。
registry.redhat.io/multicluster-engine/must-gather-rhel8	Hosted Control Plane のデータ収集。
registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v<installed_version_migration_toolkit>	Migration Toolkit for Containers のデータ収集。
registry.redhat.io/odf4/odf-must-gather-rhel9:v<installed_version_ODF>	Red Hat OpenShift Data Foundation のデータ収集。
registry.redhat.io/openshift-logging/cluster-logging-rhel9-operator:v<installed_version_logging>	ロギング用のデータ収集。
quay.io/netobserv/must-gather	Network Observability Operator のデータ収集。

イメージ	目的
registry.redhat.io/openshift4/ose-local-storage-mustgather-rhel9:v<installed_version_LSO>	ローカルストレージ Operator のデータ収集。
registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:v<installed_version_sandboxed_containers>	OpenShift サンドボックスコンテナーのデータ収集。
registry.redhat.io/workload-availability/node-healthcheck-must-gather-rhel8:v<installed_version_NHC>	Red Hat Workload Availability Operator のデータ収集。これには、Self Node Remediation (SNR) Operator、Fence Agents Remediation (FAR) Operator、Machine Deletion Remediation (MDR) Operator、Node Health Check (NHC) Operator、および Node Maintenance Operator (NMO) が含まれます。 NHC Operator バージョンが **0.9.0 より前** の場合は、このイメージを使用します。 詳細は、修復、フェンシング、およびメンテナンス (Workload Availability for Red Hat OpenShift ドキュメント) の各 Operator の「データの収集」セクションを参照してください。
registry.redhat.io/workload-availability/node-healthcheck-must-gather-rhel9:v<installed_version_NHC>	Red Hat Workload Availability Operator のデータ収集。これには、Self Node Remediation (SNR) Operator、Fence Agents Remediation (FAR) Operator、Machine Deletion Remediation (MDR) Operator、Node Health Check (NHC) Operator、および Node Maintenance Operator (NMO) が含まれます。 NHC Operator バージョンが **0.9.0 以降** の場合は、このイメージを使用します。 詳細は、修復、フェンシング、およびメンテナンス (Workload Availability for Red Hat OpenShift ドキュメント) の各 Operator の「データの収集」セクションを参照してください。
registry.redhat.io/numaresources/numaresources-must-gather-rhel9:v<installed-version-nro>	NUMA Resources Operator (NRO) のデータ収集。
registry.redhat.io/openshift4/ptp-must-gather-rhel8:v<installed-version-ptp>	PTP Operator のデータ収集。
registry.redhat.io/openshift-gitops-1/must-gather-rhel8:v<installed_version_GitOps>	Red Hat OpenShift GitOps のデータ収集。

イメージ	目的
registry.redhat.io/openshift4/ose-secrets-store-csi-mustgather-rhel9:v<installed_version_secret_store>	Secrets Store CSI Driver Operator のデータ収集。
registry.redhat.io/lvms4/lvms-must-gather-rhel9:v<installed_version_LVMS>	LVM Operator のデータ収集。
registry.redhat.io/compliance/openshift-compliance-must-gather-rhel8:<digest-version>	Compliance Operator のデータ収集。



注記

OpenShift Container Platform コンポーネントのイメージの最新バージョンを確認するには、Red Hat カスタマーポータルの OpenShift Operator ライフサイクル Web ページを参照してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform CLI (**oc**) がインストールされている。

手順

1. **must-gather** データを保存するディレクトリーに移動します。
2. **oc adm must-gather** コマンドを 1 つまたは複数の **--image** または **--image-stream** 引数と共に実行します。

注記

- 特定の機能データに加えてデフォルトの **must-gather** データを収集するには、**--image-stream=openshift/must-gather** 引数を追加します。
- Custom Metrics Autoscaler に関するデータの収集は、以下の関連情報セクションを参照してください。

たとえば、以下のコマンドは、デフォルトのクラスターデータと OpenShift Virtualization に固有の情報の両方を収集します。

```
$ oc adm must-gather \
  --image-stream=openshift/must-gather \ 1
  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel9:v4.18.17 2
```

1 デフォルトの OpenShift Container Platform **must-gather** イメージ

2 OpenShift Virtualization の must-gather イメージ

must-gather ツールを追加の引数と共に使用し、OpenShift Logging およびクラスター内の Red Hat OpenShift Logging Operator に関連するデータを収集できます。OpenShift Logging の場合、以下のコマンドを実行します。

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator \
  -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-operator")].image}')
```

例5.1 OpenShift Logging の must-gather の出力例

```
├── cluster-logging
│   ├── clo
│   │   ├── cluster-logging-operator-74dd5994f-6ttgt
│   │   ├── clusterlogforwarder_cr
│   │   ├── cr
│   │   ├── csv
│   │   ├── deployment
│   │   └── logforwarding_cr
│   ├── collector
│   │   ├── fluentd-2tr64
│   ├── eo
│   │   ├── csv
│   │   ├── deployment
│   │   └── elasticsearch-operator-7dc7d97b9d-jb4r4
│   ├── es
│   │   ├── cluster-elasticsearch
│   │   │   ├── aliases
│   │   │   ├── health
│   │   │   ├── indices
│   │   │   ├── latest_documents.json
│   │   │   ├── nodes
│   │   │   ├── nodes_stats.json
│   │   │   └── thread_pool
│   │   ├── cr
│   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   │   └── logs
│   │       ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   ├── install
│   │   ├── co_logs
│   │   ├── install_plan
│   │   ├── olmo_logs
│   │   └── subscription
│   └── kibana
│       ├── cr
│       ├── kibana-9d69668d4-2rkvz
├── cluster-scoped-resources
│   └── core
│       ├── nodes
│       │   ├── ip-10-0-146-180.eu-west-1.compute.internal.yaml
│       └── persistentvolumes
│           ├── pvc-0a8d65d9-54aa-4c44-9ecc-33d9381e41c1.yaml
```

```
├── event-filter.html
├── gather-debug.log
└── namespaces
    ├── openshift-logging
    │   ├── apps
    │   │   ├── daemonsets.yaml
    │   │   ├── deployments.yaml
    │   │   ├── replicasets.yaml
    │   │   └── statefulsets.yaml
    │   ├── batch
    │   │   ├── cronjobs.yaml
    │   │   └── jobs.yaml
    │   ├── core
    │   │   ├── configmaps.yaml
    │   │   ├── endpoints.yaml
    │   │   ├── events
    │   │   │   ├── elasticsearch-im-app-1596020400-gm6nl.1626341a296c16a1.yaml
    │   │   │   ├── elasticsearch-im-audit-1596020400-9l9n4.1626341a2af81bbd.yaml
    │   │   │   ├── elasticsearch-im-infra-1596020400-v98tk.1626341a2d821069.yaml
    │   │   │   ├── elasticsearch-im-app-1596020400-cc5vc.1626341a3019b238.yaml
    │   │   │   ├── elasticsearch-im-audit-1596020400-s8d5s.1626341a31f7b315.yaml
    │   │   │   └── elasticsearch-im-infra-1596020400-7mgv8.1626341a35ea59ed.yaml
    │   │   ├── events.yaml
    │   │   ├── persistentvolumeclaims.yaml
    │   │   ├── pods.yaml
    │   │   ├── replicationcontrollers.yaml
    │   │   ├── secrets.yaml
    │   │   └── services.yaml
    │   ├── openshift-logging.yaml
    │   ├── pods
    │   │   ├── cluster-logging-operator-74dd5994f-6ttgt
    │   │   │   ├── cluster-logging-operator
    │   │   │   │   └── cluster-logging-operator
    │   │   │   │       └── logs
    │   │   │   │           ├── current.log
    │   │   │   │           ├── previous.insecure.log
    │   │   │   │           └── previous.log
    │   │   │   └── cluster-logging-operator-74dd5994f-6ttgt.yaml
    │   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff
    │   │   │   ├── cluster-logging-operator-registry
    │   │   │   │   └── cluster-logging-operator-registry
    │   │   │   │       └── logs
    │   │   │   │           ├── current.log
    │   │   │   │           ├── previous.insecure.log
    │   │   │   │           └── previous.log
    │   │   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff.yaml
    │   │   │   └── mutate-csv-and-generate-sqlite-db
    │   │   │       └── mutate-csv-and-generate-sqlite-db
    │   │   │           └── logs
    │   │   │               ├── current.log
    │   │   │               ├── previous.insecure.log
    │   │   │               └── previous.log
    │   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
    │   │   ├── elasticsearch-im-app-1596030300-bpgcx
    │   │   │   ├── elasticsearch-im-app-1596030300-bpgcx.yaml
    │   │   │   └── indexmanagement
```

```
│   │   │   └── indexmanagement
│   │   │       └── logs
│   │   │           ├── current.log
│   │   │           ├── previous.insecure.log
│   │   │           └── previous.log
│   │   ├── fluentd-2tr64
│   │   │   ├── fluentd
│   │   │   │   └── fluentd
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   ├── fluentd-2tr64.yaml
│   │   │   └── fluentd-init
│   │   │       └── fluentd-init
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   ├── kibana-9d69668d4-2rkvz
│   │   │   ├── kibana
│   │   │   │   └── kibana
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   ├── kibana-9d69668d4-2rkvz.yaml
│   │   │   └── kibana-proxy
│   │   │       └── kibana-proxy
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   └── route.openshift.io
│   │       └── routes.yaml
│   └── openshift-operators-redhat
│       ├── ...
```

3. **oc adm must-gather** コマンドを 1 つまたは複数の **--image** または **--image-stream** 引数と共に実行します。たとえば、以下のコマンドは、デフォルトのクラスターデータと KubeVirt に固有の情報の両方を収集します。

```
$ oc adm must-gather \
 --image-stream=openshift/must-gather \ 1
 --image=quay.io/kubevirt/must-gather 2
```

1 デフォルトの OpenShift Container Platform **must-gather** イメージ

2 KubeVirt の must-gather イメージ

4. 作業ディレクトリーに作成された **must-gather** ディレクトリーから圧縮ファイルを作成します。固有の must-gather データの日付とクラスター ID を必ず提供してください。クラスター ID を確認する方法の詳細は、How to find the cluster-id or name on OpenShift cluster を参照し

てください。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
<must_gather_local_dir> 1
```

1 **<must_gather_local_dir>** は、実際のディレクトリー名に置き換えます。

5. Red Hat カスタマーポータルの **カスタマーサポート** ページ で、圧縮ファイルをサポートケースに添付します。

関連情報

- Custom Metrics Autoscaler の デバッグデータの収集
- Red Hat OpenShift Container Platform のライフサイクルポリシー

5.1.4. ネットワークログの収集

クラスター内のすべてのノードでネットワークログを収集できます。

手順

1. **-- gather_network_logs** を指定して **oc adm must-gather** コマンドを実行します。

```
$ oc adm must-gather -- gather_network_logs
```

注記

デフォルトでは、**must-gather** ツールはクラスター内のすべてのノードから OVN **nbdb** および **sbdb** データベースを収集します。OVN **nbdb** データベースの OVN-Kubernetes トランザクションを含む追加のログを含めるには、**--gather_network_logs** オプションを追加します。

2. 作業ディレクトリーに作成された **must-gather** ディレクトリーから圧縮ファイルを作成します。固有の must-gather データの日付とクラスター ID を必ず提供してください。クラスター ID を確認する方法の詳細は、How to find the cluster-id or name on OpenShift cluster を参照してください。たとえば、Linux オペレーティングシステムを使用するコンピューターで以下のコマンドを実行します。

```
$ tar cvaf must-gather-`date +"%m-%d-%Y-%H-%M-%S"`-<cluster_id>.tar.gz
<must_gather_local_dir>1
```

1 **<must_gather_local_dir>** は、実際のディレクトリー名に置き換えます。

3. Red Hat カスタマーポータルの **カスタマーサポート** ページ で、圧縮ファイルをサポートケースに添付します。

5.1.5. must-gather ストレージ制限の変更

oc adm must-gather コマンドを使用してデータを収集する場合、情報のデフォルトの最大ストレージ

は、コンテナーのストレージ容量の 30% です。30% の制限に達すると、コンテナーが強制終了し、収集プロセスが停止します。すでに収集された情報は、ローカルストレージにダウンロードされます。must-gather コマンドを再度実行するには、ストレージ容量がより大きなコンテナーを使用するか、最大ボリュームの割合を調整する必要があります。

コンテナーがストレージ制限に達すると、次の例のようなエラーメッセージが生成されます。

出力例

```
Disk usage exceeds the volume percentage of 30% for mounted directory. Exiting...
```

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

- **volume-percentage** フラグを指定して **oc adm must-gather** コマンドを実行します。新しい値として 100 を超える値を指定することはできません。

```
$ oc adm must-gather --volume-percentage <storage_percentage>
```

5.2. クラスター ID の取得

Red Hat サポートに情報を提供する際には、クラスターに固有の識別子を提供していただくと役に立ちます。OpenShift Container Platform Web コンソールを使用してクラスター ID を自動入力できます。Web コンソールまたは OpenShift CLI (**oc**) を使用してクラスター ID を手動で取得することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- Web コンソールまたはインストールされている OpenShift CLI (**oc**) にアクセスできる。

手順

- Web コンソールを使用してサポートケースを開き、クラスター ID の自動入力を行うには、以下を実行します。
  a. ツールバーから、**(?)Help** に移動し、リストから **Share Feedback** を選択します。
  b. **Tell us about your experience** ウィンドウで **Open a support case** をクリックします。
- Web コンソールを使用してクラスター ID を手動で取得するには、以下を実行します。
  a. **Home → Overview** に移動します。
  b. 値は **Details** セクションの **Cluster ID** フィールドで利用できます。
- OpenShift CLI (**oc**) を使用してクラスター ID を取得するには、以下のコマンドを実行します。

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
```

5.3. SOSREPORT について

sosreport は、設定の詳細、システム情報、および診断データを Red Hat Enterprise Linux (RHEL) および Red Hat Enterprise Linux CoreOS (RHCOS) システムから収集するツールです。**sosreport** は、ノードに関連する診断情報を収集するための標準化した方法を提供します。この情報は、問題の診断のために Red Hat サポートに提供できます。

サポートによっては、Red Hat サポートは特定の OpenShift Container Platform ノードの **sosreport** アーカイブを収集するよう依頼する場合があります。たとえば、**oc adm must-gather** の出力に含まれないシステムログまたは他のノード固有のデータを確認する必要がある場合があります。

5.4. OPENSHIFT CONTAINER PLATFORM クラスターノードの SOSREPORT アーカイブの生成

OpenShift Container Platform 4.18 クラスターノードの **sosreport** を生成する際に推奨される方法は、デバッグ Pod を使用することです。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ホストへの SSH アクセスがあること。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat の Standard または Premium サブスクリプションがある。
- Red Hat カスタマーポータルのアカウントがある。
- 既存の Red Hat サポートケース ID がある。

手順

1. クラスターノードのリストを取得します。

```
$ oc get nodes
```

2. ターゲットノードのデバッグセッションに入ります。この手順は、**<node_name>-debug** というデバッグ Pod をインスタンス化します。

```
$ oc debug node/my-cluster-node
```

NoExecute エフェクトで taint が付けられたターゲットノードで、デバッグセッションに入るには、ダミー namespace に toleration を追加して、そのダミー namespace でデバッグ Pod を起動します。

```
$ oc new-project dummy
```

```
$ oc patch namespace dummy --type=merge -p '{"metadata": {"annotations": {
"scheduler.alpha.kubernetes.io/defaultTolerations": "[{\"operator\": \"Exists\"}]"}}}'
```

```
$ oc debug node/my-cluster-node
```

3. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

4. **sosreport** を実行するために必要なバイナリーおよびプラグインが含まれる **toolbox** コンテナーを起動します。

```
# toolbox
```

注記

既存の **toolbox** Pod がすでに実行されている場合、**toolbox** コマンドは以下を出力します。**'toolbox-' already exists. Trying to start…**.**podman rm toolbox-** で実行中の toolbox コンテナーを削除して、**sosreport** プラグインの問題を回避するために、新規の toolbox コンテナーを生成します。

5. **sosreport** アーカイブを収集します。

   a. **sos report** コマンドを実行して、**crio** および **podman** で必要なトラブルシューティングデータを収集します。

```
# sos report -k crio.all=on -k crio.logs=on  -k podman.all=on -k podman.logs=on (1)
```

   (1) **-k** により、デフォルト以外の **sosreport** プラグインパラメーターを定義できます。

   b. オプション: ノードからの OVN-Kubernetes ネットワーク設定に関する情報をレポートに含めるには、次のコマンドを実行します。

```
# sos report --all-logs
```

   c. プロンプトが表示されたら **Enter** を押して続行します。

   d. Red Hat サポートケース ID を指定します。**sosreport** は ID をアーカイブのファイル名に追加します。

   e. **sosreport** 出力は、アーカイブの場所とチェックサムを提供します。以下の出力参照例は、ケース ID **01234567** を参照します。

```
Your sosreport has been generated and saved in:
  /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz 1

The checksum is: 382ffc167510fd71b4f12a4f40b97a4e
```

1 toolbox コンテナーはホストの root ディレクトリーを **/host** にマウントするため、**sosreport** アーカイブのファイルパスは **chroot** 環境外にあります。

6. 以下の方法のいずれかを使用して、解析のために **sosreport** アーカイブを Red Hat サポートに提供します。
   - 既存の Red Hat サポートケースにファイルをアップロードします。
     a. **oc debug node/<node_name>** コマンドを実行して **sosreport** アーカイブを連結し、出力をファイルにリダイレクトします。このコマンドは、直前の **oc debug** セッションを終了していることを前提としています。

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz' > /tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz 1
```

1 デバッグコンテナーは、ホストの root ディレクトリーを **/host** にマウントします。連結のためにターゲットファイルを指定する際に、デバッグコンテナーの root ディレクトリー (**/host** を含む) から絶対パスを参照します。

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。**scp** を使用してクラスターノードから **sosreport** アーカイブを転送することは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この状態では、**scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** を実行して、ノードから **sosreport** アーカイブをコピーすることができます。

     b. Red Hat カスタマーポータルの Customer Support ページ にある既存のサポートケースに移動します。

     c. **Attach files** を選択し、プロンプトに従ってファイルをアップロードします。

5.5. ブートストラップノードのジャーナルログのクエリー

ブートストラップ関連の問題が発生した場合、ブートストラップノードから **bootkube.service** の **journald** ユニットログおよびコンテナーログを収集できます。

前提条件

- ブートストラップノードへの SSH アクセスがある。

- ブートストラップノードの完全修飾ドメイン名がある。

手順

1. OpenShift Container Platform のインストール時にブートストラップに対して **bootkube.service** の **journald** ユニットログのクエリーを実行。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```

注記

ブートストラップノードの **bootkube.service** のログは etcd の **connection refused** エラーを出力し、ブートストラップサーバーがコントロールプレーンノードの etcd に接続できないことを示します。etcd が各コントロールプレーンノードで起動し、ノードがクラスターに参加した後には、エラーは発生しなくなるはずです。

2. ブートストラップノードで **podman** を使用してブートストラップノードのコンテナーからログを収集します。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs
$pod; done'
```

5.6. クラスターノードジャーナルログのクエリー

個別のクラスターノードの **/var/log** 内で **journald** ユニットログおよびその他のログを収集できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- API サービスが機能している。
- ホストへの SSH アクセスがあること。

手順

1. OpenShift Container Platform クラスターノードに対して、**kubelet** の **journald** ユニットログのクエリーを実行します。以下の例では、コントロールプレーンノードのみがクエリーされます。

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 クエリーを実行して他のユニットログを取得するために、**kubelet** を適宜置き換えます。

2. クラスターノードの **/var/log/** の下にある特定のサブディレクトリーからログを収集します。

a. **/var/log/** サブディレクトリー内に含まれるログの一覧を取得します。以下の例では、すべてのコントロールプレーンノードの **/var/log/openshift-apiserver/** にあるファイルをリスト表示します。

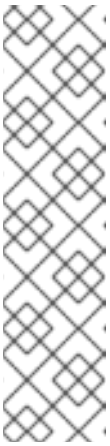
```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

b. **/var/log/** サブディレクトリー内の特定ログを確認します。以下の例は、すべてのコントロールプレーンノードから **/var/log/openshift-apiserver/audit.log** コンテンツを出力します。

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. API が機能しない場合は、代わりに SSH を使用して各ノードのログを確認します。次の例では、**/var/log/openshift-apiserver/audit.log** の末尾を表示 (tail) します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

5.7. ネットワークトレースメソッド

パケットキャプチャーレコードの形式でネットワークトレースを収集すると、Red Hat がネットワークの問題のトラブルシューティングをサポートできます。

OpenShift Container Platform では、ネットワークトレースの実行方法として 2 種類サポートします。以下の表を確認し、ニーズに合った方法を選択します。

表5.3 サポート対象のネットワークトレース収集の方法

メソッド	利点および機能
ホストのネットワークトレースの収集	1 つ以上のノードで同時に指定する期間で、パケットキャプチャーを実行します。パケットキャプチャーファイルは、指定した期間に達すると、ノードからクライアントマシンに転送されます。 特定のアクションが原因でネットワーク通信に問題を発生される理由をトラブルシューティングでいます。パケットキャプチャーを実行し、問題を発生させるアクションを実行してログで問題を診断します。

メソッド	利点および機能
OpenShift Container Platform ノードまたはコンテナーからのネットワークトレースの収集	1 つのノードまたは 1 つのコンテナーでパケットキャプチャーを実行します。パケットキャプチャーの期間を制御できるように **tcpdump** コマンドを対話的に実行します。 パケットキャプチャーを手動で開始し、ネットワーク通信の問題をトリガーしてから、パケットキャプチャーを手動で停止できます。 この方法では、**cat** コマンドおよびシェルのリダイレクトを使用して、パケットキャプチャーデータをノードまたはコンテナーからクライアントマシンにコピーします。

5.8. ホストのネットワークトレースの収集

ネットワーク関連の問題のトラブルシューティングは、ネットワーク通信を追跡して複数のノードで同時にパケットをキャプチャーすることで簡素化されます。

oc adm must-gather コマンドおよび **registry.redhat.io/openshift4/network-tools-rhel8** コンテナーイメージの組み合わせを使用して、ノードからパケットキャプチャーを収集できます。パケットキャプチャーの分析は、ネットワーク通信の問題のトラブルシューティングに役立ちます。

oc adm must-gather コマンドは、特定のノードの Pod で **tcpdump** コマンドの実行に使用されます。**tcpdump** コマンドは、Pod でキャプチャーされたパケットを記録します。**tcpdump** コマンドを終了すると、**oc adm must-gather** コマンドは、Pod からクライアントマシンにキャプチャーされたパケットが含まれるファイルを転送します。

ヒント

以下の手順で使用するコマンド例は、**tcpdump** コマンドを使用してパケットキャプチャーを実行する方法を示しています。ただし、**--image** 引数で指定したコンテナーイメージでコマンドを実行すると、複数のノードから同時にトラブルシューティング情報を収集できます。

前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform にログインしている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下のコマンドを実行して、一部のノードでホストネットワークからパケットキャプチャーを実行します。

```
$ oc adm must-gather \
   --dest-dir /tmp/captures \// <.>
   --source-dir '/tmp/tcpdump/' \// <.>
   --image registry.redhat.io/openshift4/network-tools-rhel8:latest \// <.>
   --node-selector 'node-role.kubernetes.io/worker' \// <.>
   --host-network=true \// <.>
   --timeout 30s \// <.>
```

```
-- \
tcpdump -i any \/ <.>
-w /tmp/tcpdump/%Y-%m-%dT%H:%M:%S.pcap -W 1 -G 300
```

<.> **--dest-dir** 引数では、**oc adm must-gather** の実行時に、クライアントマシンの **/tmp/captures** と相対パスにあるディレクトリーに、キャプチャーしたパケットを保存することを指定します。書き込み可能な任意のディレクトリーを指定できます。<.> **oc adm must-gather** が開始するデバッグ Pod で **tcpdump** が実行される場合に、**--source-dir** 引数は、パケットキャプチャーが Pod の **/tmp/tcpdump** ディレクトリーに一時的に保存されることを指定します。<.> **--image** 引数は、**tcpdump** コマンドを含むコンテナーイメージを指定します。<.> **--node-selector** 引数とサンプル値は、ワーカーノードでパケットキャプチャーを実行するように指定します。別の方法としては、代わりに **--node-name** 引数を指定して、1 つのノードでパケットキャプチャーを実行できます。**--node-selector** と **--node-name** 引数の両方を省略すると、すべてのノードでパケットキャプチャーが実行されます。<.> ノードのネットワークインターフェイスでパケットキャプチャーが実行されるように、**--host-network=true** 引数が必要です。<.> **--timeout** 引数と値は、デバッグ Pod を 30 秒間実行するように指定します。 **--timeout** 引数と期間を指定しない場合、デバッグ Pod は 10 分間実行されます。<.> **tcpdump** コマンドの **-i any** 引数は、すべてのネットワークインターフェイスでパケットをキャプチャーするように指定します。また、ネットワークインターフェイス名を指定することもできます。

2. ネットワークトレースがパケットをキャプチャーしている間に、ネットワーク通信の問題を発生させる、Web アプリケーションにアクセスするなど、特定のアクションを実行します。

3. **oc adm must-gather** で Pod からクライアントマシンに転送したパケットキャプチャーファイルを確認します。

```
tmp/captures
├──── event-filter.html
├──── ip-10-0-192-217-ec2-internal 1
│     └──── registry-redhat-io-openshift4-network-tools-rhel8-sha256-bca...
│           └──── 2022-01-13T19:31:31.pcap
├──── ip-10-0-201-178-ec2-internal 2
│     └──── registry-redhat-io-openshift4-network-tools-rhel8-sha256-bca...
│           └──── 2022-01-13T19:31:30.pcap
├──── ip-...
└──── timestamp
```

1 2 パケットのキャプチャーは、ホスト名、コンテナー、ファイル名を識別するディレクトリーに保存されます。**--node-selector** 引数を指定しなかった場合には、ホスト名のディレクトリーレベルは存在しません。

5.9. OPENSHIFT CONTAINER PLATFORM ノードまたはコンテナーからのネットワークトレースの収集

ネットワーク関連の OpenShift Container Platform の潜在的な問題を調査する際に、Red Hat サポートは特定の OpenShift Container Platform クラスターノードまたは特定のコンテナーからネットワークパケットトレースを要求する可能性があります。OpenShift Container Platform でネットワークトレースをキャプチャーする方法として、デバッグ Pod を使用できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

- OpenShift CLI (**oc**) がインストールされている。
- 既存の Red Hat サポートケース ID がある。
- Red Hat の Standard または Premium サブスクリプションがある。
- Red Hat カスタマーポータルのアカウントがある。
- ホストへの SSH アクセスがあること。

手順

1. クラスターノードのリストを取得します。

```
$ oc get nodes
```

2. ターゲットノードのデバッグセッションに入ります。この手順は、**<node_name>-debug** というデバッグ Pod をインスタンス化します。

```
$ oc debug node/my-cluster-node
```

3. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

4. **chroot** 環境コンソール内から、ノードのインターフェイス名を取得します。

```
# ip ad
```

5. **sosreport** を実行するために必要なバイナリーおよびプラグインが含まれる **toolbox** コンテナーを起動します。

```
# toolbox
```

注記

既存の **toolbox** Pod がすでに実行されている場合、**toolbox** コマンドは以下を出力します。**'toolbox-' already exists. Trying to start…**。**tcpdump** の問題を回避するために、**podman rm toolbox-** を使用して実行中のツールボックスコンテナーを削除し、新しいツールボックスコンテナーを生成してください。

6. クラスターノードで **tcpdump** セッションを開始し、出力をキャプチャーファイルにリダイレクトします。この例では、**ens5** をインターフェイス名として使用します。

```
$ tcpdump -nn -s 0 -i ens5 -w /host/var/tmp/my-cluster-node_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap
```
(1)

(1) toolbox コンテナーはホストの root ディレクトリーを **/host** にマウントするため、**tcpdump** キャプチャーファイルのパスは **chroot** 環境外にあります。

7. ノード上の特定コンテナーに **tcpdump** キャプチャーが必要な場合は、以下の手順に従います。

   a. ターゲットコンテナー ID を確認します。toolbox コンテナーはホストの root ディレクトリーを **/host** にマウントするため、この手順では、**chroot host** コマンドが **crictl** コマンドの前に実行されます。

```
# chroot /host crictl ps
```

   b. コンテナーのプロセス ID を確認します。この例では、コンテナー ID は **a7fe32346b120** です。

```
# chroot /host crictl inspect --output yaml a7fe32346b120 | grep 'pid' | awk '{print $2}'
```

   c. コンテナーで **tcpdump** セッションを開始し、出力をキャプチャーファイルにリダイレクトします。この例では、**49628** をコンテナーのプロセス ID として使用し、**ens5** をインターフェイス名として使用します。**nsenter** コマンドはターゲットプロセスの namespace に入り、その namespace でコマンドを実行します。この例ではターゲットプロセスがコンテナーのプロセス ID であるため、**tcpdump** コマンドはホストからコンテナーの namespace で実行されます。

```
# nsenter -n -t 49628 -- tcpdump -nn -i ens5 -w /host/var/tmp/my-cluster-node-my-container_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap
```
(1)

(1) toolbox コンテナーはホストの root ディレクトリーを **/host** にマウントするため、**tcpdump** キャプチャーファイルのパスは **chroot** 環境外にあります。

8. 以下の方法のいずれかを使用して、分析用に **tcpdump** キャプチャーファイルを Red Hat サポートに提供します。

   - 既存の Red Hat サポートケースにファイルをアップロードします。

     a. **oc debug node/<node_name>** コマンドを実行して **sosreport** アーカイブを連結し、出力をファイルにリダイレクトします。このコマンドは、直前の **oc debug** セッションを終了していることを前提としています。

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-tcpdump-capture-file.pcap' > /tmp/my-tcpdump-capture-file.pcap
```
(1)

(1) デバッグコンテナーは、ホストの root ディレクトリーを **/host** にマウントします。連結のためにターゲットファイルを指定する際に、デバッグコンテナーの root ディレクトリー (**/host** を含む) から絶対パスを参照します。

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。**scp** を使用してクラスターノードから **tcpdump** キャプチャーファイルを転送することは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この状態では、**scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** を実行して、ノードから **tcpdump** キャプチャーファイルをコピーすることができます。

b. Red Hat カスタマーポータルの Customer Support ページ にある既存のサポートケースに移動します。

c. **Attach files** を選択し、プロンプトに従ってファイルをアップロードします。

5.10. RED HAT サポートへの診断データの提供

OpenShift Container Platform の問題を調査する際に、Red Hat サポートは診断データをサポートケースにアップロードするよう依頼する可能性があります。Red Hat カスタマーポータルから、ファイルをサポートケースにアップロードできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- Red Hat の Standard または Premium サブスクリプションがある。
- Red Hat カスタマーポータルのアカウントがある。
- 既存の Red Hat サポートケース ID がある。

手順

- Red Hat カスタマーポータルから既存の Red Hat サポートケースに診断データをアップロードします。

  a. **oc debug node/<node_name>** コマンドを使用して OpenShift Container Platform ノードで組み込まれている診断ファイルを連結し、出力をファイルにリダイレクトします。以下の例では、**/host/var/tmp/my-diagnostic-data.tar.gz** をデバッグコンテナーから **/var/tmp/my-diagnostic-data.tar.gz** にコピーします。

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-diagnostic-data.tar.gz' > /var/tmp/my-diagnostic-data.tar.gz
```
(1)

(1) デバッグコンテナーは、ホストの root ディレクトリーを **/host** にマウントします。連結のためにターゲットファイルを指定する際に、デバッグコンテナーの root ディレクトリー (**/host** を含む) から絶対パスを参照します。

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。**scp** を使用してクラスターノードからファイルを転送することは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この状態では、**scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** を実行してノードから診断ファイルをコピーできます。

b. Red Hat カスタマーポータルの Customer Support ページ にある既存のサポートケースに移動します。

c. **Attach files** を選択し、プロンプトに従ってファイルをアップロードします。

5.11. TOOLBOX について

toolbox は、Red Hat Enterprise Linux CoreOS (RHCOS) システムでコンテナーを起動するツールです。このツールは、主に **sosreport** などのコマンドの実行に必要なバイナリーおよびプラグインを含むコンテナーを起動するために使用されます。

toolbox コンテナーの主な目的は、診断情報を収集し、これを Red Hat サポートに提供することにあります。ただし、追加の診断ツールが必要な場合は、RPM パッケージを追加するか、標準のサポートツールイメージの代替イメージを実行することができます。

5.11.1. toolbox コンテナーへのパッケージのインストール

デフォルトでは、**toolbox** コマンドを実行すると、**registry.redhat.io/rhel9/support-tools:latest** イメージを使用してコンテナーが起動します。このイメージには、最も頻繁に使用されるサポートツールが含まれます。イメージの一部ではないサポートツールを必要とするノード固有のデータを収集する必要がある場合は、追加のパッケージをインストールできます。

前提条件

- **oc debug node/<node_name>** コマンドでノードにアクセスしている。
- root 権限を持つユーザーとしてシステムにアクセスできる。

手順

1. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

2. toolbox コンテナーを起動します。

```
# toolbox
```

3. **wget** などの追加のパッケージをインストールします。

```
# dnf install -y <package_name>
```

5.11.2. toolbox を使用した代替イメージの起動

デフォルトでは、**toolbox** コマンドを実行すると、**registry.redhat.io/rhel9/support-tools:latest** イメージを使用してコンテナーが起動します。

注記

.toolboxrc ファイルを作成し、実行するイメージを指定して代替イメージを起動できます。ただし、**registry.redhat.io/rhel8/support-tools:latest** など、古いバージョンの **support-tools** イメージの実行は、OpenShift Container Platform 4.18 ではサポートされていません。

前提条件

- **oc debug node/<node_name>** コマンドでノードにアクセスしている。
- root 権限を持つユーザーとしてシステムにアクセスできる。

手順

1. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

2. オプション: デフォルトのイメージの代わりに代替イメージを使用する必要がある場合は、root ユーザー ID のホームディレクトリーに **.toolboxrc** ファイルを作成し、イメージのメタデータを指定します。

```
REGISTRY=quay.io                  1
IMAGE=fedora/fedora:latest        2
TOOLBOX_NAME=toolbox-fedora-latest 3
```

1. オプション: 代替コンテナーレジストリーを指定します。
2. 開始する代替イメージを指定します。
3. オプション: toolbox コンテナーの代替名を指定します。

3. 次のコマンドを入力して toolbox コンテナーを起動します。

```
# toolbox
```

注記

既存の **toolbox** Pod がすでに実行されている場合、**toolbox** コマンドは以下を出力します。**'toolbox-' already exists. Trying to start…**。**sosreport** プラグインの問題を回避するには、**podman rm toolbox-** を使用して実行中の toolbox コンテナーを削除し、新しい toolbox コンテナーを生成してください。

第6章 クラスター仕様の要約

6.1. クラスターバージョンオブジェクトを使用してクラスター仕様を要約する

クエリーを実行して **clusterversion** リソースを取得することにより、OpenShift Container Platform クラスター仕様の要約を取得できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. クラスターバージョン、可用性、アップタイム、および一般的なステータスのクエリーを実行します。

```
$ oc get clusterversion
```

出力例

```
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.13.8    True        False         8h      Cluster version is 4.13.8
```

2. クラスター仕様の詳細な要約、更新の可用性、および更新履歴を取得します。

```
$ oc describe clusterversion
```

出力例

```
Name:         version
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  config.openshift.io/v1
Kind:         ClusterVersion
# ...
    Image:    quay.io/openshift-release-dev/ocp-
release@sha256:a956488d295fe5a59c8663a4d9992b9b5d0950f510a7387dbbfb8d20fc5970ce

    URL:      https://access.redhat.com/errata/RHSA-2023:4456
    Version:  4.13.8
  History:
    Completion Time:    2023-08-17T13:20:21Z
    Image:              quay.io/openshift-release-dev/ocp-
release@sha256:a956488d295fe5a59c8663a4d9992b9b5d0950f510a7387dbbfb8d20fc5970ce

    Started Time:       2023-08-17T12:59:45Z
    State:              Completed
```

```
    Verified:          false
    Version:           4.13.8
# ...
```

第7章 トラブルシューティング

7.1. インストールのトラブルシューティング

7.1.1. インストールの問題が発生する場所の判別

OpenShift Container Platform のインストールの問題のトラブルシューティング時に、インストールログを監視して、問題が発生した段階を判別できます。次に、その段階に関連する診断データを取得します。

OpenShift Container Platform インストールは以下の段階に従って実行されます。

1. Ignition 設定ファイルが作成されます。
2. ブートストラップマシンが起動し、コントロールプレーンマシンの起動に必要なリモートリソースのホスティングを開始します。
3. コントロールプレーンマシンは、ブートストラップマシンからリモートリソースをフェッチし、起動を終了します。
4. コントロールプレーンマシンはブートストラップマシンを使用して、etcd クラスターを作成します。
5. ブートストラップマシンは、新規 etcd クラスターを使用して一時的な Kubernetes コントロールプレーンを起動します。
6. 一時的なコントロールプレーンは、実稼働コントロールプレーンをコントロールプレーンマシンにスケジュールします。
7. 一時的なコントロールプレーンはシャットダウンし、コントロールを実稼働コントロールプレーンに渡します。
8. ブートストラップマシンは OpenShift Container Platform コンポーネントを実稼働コントロールプレーンに追加します。
9. インストールプログラムはブートストラップマシンをシャットダウンします。
10. コントロールプレーンはワーカーノードをセットアップします。
11. コントロールプレーンは一連の Operator の形式で追加のサービスをインストールします。
12. クラスターはサポートされる環境でのワーカーマシンの作成など、日常の操作に必要な残りのコンポーネントをダウンロードし、設定します。

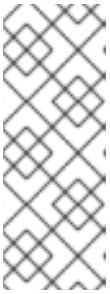
7.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーのインストールに関する考慮事項

デフォルトのインストール方法は、インストーラーでプロビジョニングされるインフラストラクチャーです。インストーラーでプロビジョニングされるインフラストラクチャークラスターの場合、OpenShift Container Platform は、オペレーティングシステム自体を含むクラスターのすべての側面を管理します。可能な場合は、この機能を使用してクラスターインフラストラクチャーのプロビジョニングと保守の手間を省くようにしてください。

OpenShift Container Platform 4.18 は、ユーザーが独自に提供するインフラストラクチャーにインストールすることもできます。このインストール方法を使用する場合は、ユーザーによってプロビジョニ

ングされるインフラストラクチャーのインストールドキュメントに注意深く従ってください。また、インストール前に以下の考慮事項を確認してください。

- Red Hat Enterprise Linux (RHEL) Ecosystem を確認し、選択したサーバーハードウェアまたは仮想化テクノロジー向けに提供されている Red Hat Enterprise Linux CoreOS (RHCOS) サポートのレベルを判別します。
- 多くの仮想化環境およびクラウド環境では、ゲストオペレーティングシステムにエージェントをインストールする必要があります。これらのエージェントがデーモンセット経由でデプロイされるコンテナー化されたワークロードとしてインストールされていることを確認します。
- 動的ストレージ、オンデマンドサービスルーティング、ノードホスト名の Kubernetes ホスト名への解決、クラスターの自動スケーリングなどの機能を有効にする場合は、クラウドプロバイダーの統合をインストールします。



注記

異なるクラウドプロバイダーのリソースを組み合わせた OpenShift Container Platform 環境でのクラウドプロバイダーの統合を有効にしたり、複数の物理または仮想プラットフォームにまたがるクラウドプロバイダーの統合を有効にすることはできません。ノードライフサイクルコントローラーでは、既存プロバイダーの外部にあるノードをクラスターに追加することはできず、複数のクラウドプロバイダーの統合を指定することはできません。

- マシンセットまたは自動スケーリングを使用して OpenShift Container Platform クラスターノードを自動的にプロビジョニングする必要がある場合、プロバイダー固有のマシン API 実装が必要です。
- 選択したクラウドプロバイダーが、初期デプロイメントの一部として Ignition 設定ファイルをホストに挿入する方法を提供するかどうかを確認します。提供しない場合は、HTTP サーバーを使用して Ignition 設定ファイルをホストする必要があります。Ignition 設定ファイルの問題のトラブルシューティングを行う手順は、これらの 2 つの方法のどちらをデプロイするかによって異なります。
- 組み込みコンテナーレジストリー、Elasticsearch、Prometheus などのオプションのフレームワークコンポーネントを利用する必要がある場合は、ストレージを手動でプロビジョニングする必要があります。デフォルトのストレージクラスは、明示的に設定されない限り、ユーザーによってプロビジョニングされるインフラストラクチャーのインストールでは定義されません。
- ロードバランサーは、可用性の高い OpenShift Container Platform 環境にあるすべてのコントロールプレーンノードに API 要求を分散するために必要です。OpenShift Container Platform DNS ルーティングおよびポートの要件を満たす TCP ベースの負荷分散ソリューションを使用できます。

7.1.3. OpenShift Container Platform インストール前のロードバランサー設定の確認

OpenShift Container Platform インストールを開始する前に、ロードバランサーの設定を確認してください。

前提条件

- OpenShift Container Platform インストールの準備のために、選択した外部ロードバランサーを設定している。以下の例では、HAProxy を使用した Red Hat Enterprise Linux (RHEL) ホストに基づいて、負荷分散サービスをクラスターに提供します。

- OpenShift Container Platform インストールの準備のために DNS を設定している。
- ロードバランサーへの SSH アクセスがある。

手順

1. **haproxy** systemd サービスがアクティブであることを確認します。

```
$ ssh <user_name>@<load_balancer> systemctl status haproxy
```

2. ロードバランサーが必要なポートでリッスンしていることを確認します。以下の例では、ポート **80**、**443**、**6443**、および **22623** を参照します。
   - Red Hat Enterprise Linux (RHEL) 6 で実行している HAProxy インスタンスの場合は、**netstat** コマンドを使用して、ポートのステータスを確認します。

```
$ ssh <user_name>@<load_balancer> netstat -nltupe | grep -E ':80|:443|:6443|:22623'
```

   - Red Hat Enterprise Linux (RHEL) 7 または 8 で実行している HAProxy インスタンスの場合、**ss** コマンドを使用して、ポートのステータスを確認します。

```
$ ssh <user_name>@<load_balancer> ss -nltupe | grep -E ':80|:443|:6443|:22623'
```

注記

Red Hat は、Red Hat Enterprise Linux (RHEL) 7 以降の **netstat** ではなく、**ss** コマンドを推奨しています。**ss** は、iproute パッケージで提供されます。**ss** コマンドの詳細は、Red Hat Enterprise Linux (RHEL) 7 パフォーマンスチューニングガイド を参照してください。

3. ワイルドカード DNS レコードがロードバランサーに解決されていることを確認します。

```
$ dig <wildcard_fqdn> @<dns_server>
```

7.1.4. OpenShift Container Platform インストーラーのログレベルの指定

デフォルトで、OpenShift Container Platform インストーラーのログレベルは **info** に設定されます。失敗した OpenShift Container Platform インストールの診断時により詳細なロギングが必要な場合は、再びインストールを開始する際に **openshift-install** ログレベルを **debug** に引き上げることができます。

前提条件

- インストールホストにアクセスできる。

手順

- インストールを開始する際に、インストールのログレベルを **debug** に設定します。

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete --log-level debug
1
```

1 使用可能なログレベルは、**info**、**warn**、**error**、および **debug** です。

7.1.5. openshift-install コマンド関連の問題のトラブルシューティング

openshift-install コマンドの実行に問題がある場合には、以下を確認してください。

- インストールは Ignition 設定ファイルの作成から 24 時間以内に開始されている。Ignition ファイルは以下のコマンドの実行時に作成されている。

```
$ ./openshift-install create ignition-configs --dir=./install_dir
```

- **install-config.yaml** ファイルはインストーラーと同じディレクトリーにある。代替インストールパスが **./openshift-install --dir** オプションを使用して宣言される場合、そのディレクトリーに **install-config.yaml** ファイルが存在することを確認します。

7.1.6. インストールの進捗の監視

OpenShift Container Platform インストールの進捗として、高レベルのインストール、ブートストラップ、およびコントロールプレーンのログをモニターできます。これにより、インストールの進捗をより明確に把握できるようになり、インストールが失敗する段階を特定しやすくなります。

前提条件

- **cluster-admin** クラスターロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- ブートストラップおよびコントロールプレーンノードの完全修飾ドメイン名がある。

注記

初期の **kubeadmin** パスワードは、インストールホストの **<install_directory>/auth/kubeadmin-password** にあります。

手順

1. インストールの進捗に応じてインストールログを監視します。

```
$ tail -f ~/<installation_directory>/.openshift_install.log
```

2. 起動後にブートストラップノードで **bootkube.service** journald ユニットログを監視します。これにより、最初のコントロールプレーンのブートストラップを可視化できます。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```

注記

ブートストラップノードの **bootkube.service** のログは etcd の **connection refused** エラーを出力し、ブートストラップサーバーがコントロールプレーンノードの etcd に接続できないことを示します。etcd が各コントロールプレーンノードで起動し、ノードがクラスターに参加した後には、エラーは発生しなくなるはずです。

3. 起動後のコントロールプレーンノードで **kubelet.service** journald ユニットログを監視します。これにより、コントロールプレーンノードエージェントのアクティビティーを可視化できます。

   a. **oc** を使用してログを監視します。

   ```
   $ oc adm node-logs --role=master -u kubelet
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
   ```

4. 起動後のコントロールプレーンノードで **crio.service** journald ユニットログを監視します。これにより、コントロールプレーンノードの CRI-O コンテナーランタイムのアクティビティーを可視化できます。

   a. **oc** を使用してログを監視します。

   ```
   $ oc adm node-logs --role=master -u crio
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

   ```
   $ ssh core@master-N.cluster_name.sub_domain.domain journalctl -b -f -u crio.service
   ```

7.1.7. ブートストラップノードの診断データの収集

ブートストラップ関連の問題が発生した場合、ブートストラップノードから **bootkube.service** の **journald** ユニットログおよびコンテナーログを収集できます。

前提条件

- ブートストラップノードへの SSH アクセスがある。
- ブートストラップノードの完全修飾ドメイン名がある。
- HTTP サーバーを使用して Ignition 設定ファイルをホストする場合、HTTP サーバーの完全修飾ドメイン名およびポート番号が必要です。HTTP ホストへの SSH アクセスも必要です。

手順

1. ブートストラップノードのコンソールにアクセスできる場合は、ノードがログインプロンプトに到達するまでコンソールを監視します。

2. Ignition ファイル設定を検証します。
   - HTTP サーバーを使用して Ignition 設定ファイルをホストする場合。
     a. ブートストラップノードの Ignition ファイル URL を確認します。**<http_server_fqdn>** を HTTP サーバーの完全修飾ドメイン名に置き換えます。

```
$ curl -I http://<http_server_fqdn>:<port>/bootstrap.ign 1
```

     1 **-I** オプションはヘッダーのみを返します。Ignition ファイルが指定された URL で利用可能な場合、コマンドは **200 OK** ステータスを返します。これが利用できない場合は、コマンドは **404 file not found** を返します。

     b. Ignition ファイルがブートストラップノードで受信されたことを確認するには、提供側ホストの HTTP サーバーログのクエリーを実行します。たとえば、Apache Web サーバーを使用して Ignition ファイルを提供する場合は、以下のコマンドを入力します。

```
$ grep -is 'bootstrap.ign' /var/log/httpd/access_log
```

     ブートストラップ Ignition ファイルが受信される場合、関連付けられた **HTTP GET** ログメッセージには要求が成功したことを示す **200 OK** の成功ステータスが含まれます。

     c. Ignition ファイルが受信されていない場合には、Ignition ファイルが存在し、それらに提供側ホストの適切なファイルおよび Web サーバーパーミッションがあることを直接確認します。

   - クラウドプロバイダーのメカニズムを使用して Ignition 設定ファイルを初期デプロイメントの一部としてホストに挿入する場合。
     a. ブートストラップノードのコンソールを確認し、ブートストラップノードの Ignition ファイルを正しく挿入するメカニズムが機能しているかどうかを確認します。

3. ブートストラップノードの割り当てられたストレージデバイスの可用性を確認します。
4. ブートストラップノードに DHCP サーバーから IP アドレスが割り当てられていることを確認します。
5. ブートストラップノードから **bootkube.service** journald ユニットログを収集します。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```

注記

ブートストラップノードの **bootkube.service** のログは etcd の **connection refused** エラーを出力し、ブートストラップサーバーがコントロールプレーンノードの etcd に接続できないことを示します。etcd が各コントロールプレーンノードで起動し、ノードがクラスターに参加した後には、エラーは発生しなくなるはずです。

6. ブートストラップノードコンテナーからログを収集します。
   a. ブートストラップノードで **podman** を使用してログを収集します。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman
logs $pod; done'
```

7. ブートストラッププロセスに失敗した場合は、以下を確認します。
   - インストールホストから **api.<cluster_name>.<base_domain>** を解決できます。
   - ロードバランサーはブートストラップおよびコントロールプレーンノードへのポート 6443 接続をプロキシーします。プロキシー設定が OpenShift Container Platform のインストール要件を満たしていることを確認します。

7.1.8. コントロールプレーンノードのインストールの問題の調査

コントロールプレーンノードのインストールに問題がある場合には、コントロールプレーンノード、OpenShift Container Platform ソフトウェア定義ネットワーク (SDN)、およびネットワーク Operator のステータスを判別します。**kubelet.service**、**crio.service** journald ユニットログ、およびコントロールプレーンノードコンテナーログを収集し、コントロールプレーンノードエージェント、CRI-O コンテナーランタイム、および Pod アクティビティーを可視化します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- ブートストラップおよびコントロールプレーンノードの完全修飾ドメイン名がある。
- HTTP サーバーを使用して Ignition 設定ファイルをホストする場合、HTTP サーバーの完全修飾ドメイン名およびポート番号が必要です。HTTP ホストへの SSH アクセスも必要です。

注記

初期の **kubeadmin** パスワードは、インストールホストの **<install_directory>/auth/kubeadmin-password** にあります。

手順

1. コントロールプレーンノードのコンソールにアクセスできる場合は、ノードがログインプロンプトに到達するまでコンソールを監視します。インストール時に、Ignition ログメッセージはコンソールに出力されます。
2. Ignition ファイル設定を確認します。
   - HTTP サーバーを使用して Ignition 設定ファイルをホストする場合。
     a. コントロールプレーンノードの Ignition ファイル URL を確認します。**<http_server_fqdn>** を HTTP サーバーの完全修飾ドメイン名に置き換えます。

```
$ curl -I http://<http_server_fqdn>:<port>/master.ign 1
```

1 **-I** オプションはヘッダーのみを返します。Ignition ファイルが指定された URL で利用可能な場合、コマンドは **200 OK** ステータスを返します。これが利用できな

b. Ignition ファイルがコントロールプレーンノードで受信されたことを確認するには、提供側ホストの HTTP サーバーログのクエリーを実行します。たとえば、Apache Web サーバーを使用して Ignition ファイルを提供する場合は、以下を考慮してください。

```
$ grep -is 'master.ign' /var/log/httpd/access_log
```

マスター Ignition ファイルが受信される場合、関連付けられた **HTTP GET** ログメッセージには要求が成功したことを示す **200 OK** の成功ステータスが含まれます。

c. Ignition ファイルが受信されなかった場合、これが提供側ホストに存在することを直接確認します。適切なファイルおよび Web サーバーのパーミッションが適用されていることを確認します。

- クラウドプロバイダーのメカニズムを使用して Ignition 設定ファイルを初期デプロイメントの一部としてホストに挿入する場合。

a. コントロールプレーンノードのコンソールを確認し、コントロールプレーンノードの Ignition ファイルを正しく挿入するメカニズムが機能しているかどうかを確認します。

3. コントロールプレーンノードに割り当てられたストレージデバイスの可用性を確認します。

4. コントロールプレーンノードに DHCP サーバーから IP アドレスが割り当てられていることを確認します。

5. コントロールプレーンノードのステータスを判別します。

a. コントロールプレーンノードのステータスのクエリーを実行します。

```
$ oc get nodes
```

b. コントロールプレーンノードのいずれかが **Ready** ステータスに達していない場合は、詳細なノードの説明を取得します。

```
$ oc describe node <master_node>
```

注記

インストールの問題により OpenShift Container Platform API が実行できなくなったり、kubelet が各ノードでまだ実行されていない場合、**oc** コマンドを実行することはできません。

6. OVN-Kubernetes のステータスを確認します。

a. **openshift-ovn-kubernetes** namespace で、 **ovnkube-node** デーモンセットのステータスを確認します。

```
$ oc get daemonsets -n openshift-ovn-kubernetes
```

b. これらのリソースが **Not found** としてリストされている場合は、**openshift-ovn-kubernetes** namespace 内の Pod を確認します。

```
$ oc get pods -n openshift-ovn-kubernetes
```

c. **openshift-ovn-kubernetes** namespace 内の失敗した OpenShift Container Platform OVN-Kubernetes Pod に関連するログを確認します。

```
$ oc logs <ovn-k_pod> -n openshift-ovn-kubernetes
```

7. クラスターのネットワーク設定のステータスを確認します。

a. クラスターのネットワーク設定が存在するかどうかを確認します。

```
$ oc get network.config.openshift.io cluster -o yaml
```

b. インストーラーがネットワーク設定の作成に失敗した場合、Kubernetes マニフェストを再度生成し、メッセージの出力を確認します。

```
$ ./openshift-install create manifests
```

c. **openshift-network-operator** namespace で Pod のステータスを確認し、Cluster Network Operator (CNO) が実行されているかどうかを判別します。

```
$ oc get pods -n openshift-network-operator
```

d. **openshift-network-operator** namespace からネットワーク Operator Pod ログを収集します。

```
$ oc logs pod/<network_operator_pod_name> -n openshift-network-operator
```

8. 起動後のコントロールプレーンノードで **kubelet.service** journald ユニットログを監視します。これにより、コントロールプレーンノードエージェントのアクティビティーを可視化できます。

a. **oc** を使用してログを取得します。

```
$ oc adm node-logs --role=master -u kubelet
```

b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

9. 起動後のコントロールプレーンノードで **crio.service** journald ユニットログを取得します。これにより、コントロールプレーンノードの CRI-O コンテナーランタイムのアクティビティーを可視化できます。

   a. **oc** を使用してログを取得します。

   ```
   $ oc adm node-logs --role=master -u crio
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
   crio.service
   ```

10. コントロールプレーンノードの **/var/log/** の下にある特定のサブディレクトリーからログを収集します。

    a. **/var/log/** サブディレクトリー内に含まれるログの一覧を取得します。以下の例では、すべてのコントロールプレーンノードの **/var/log/openshift-apiserver/** にあるファイルをリスト表示します。

    ```
    $ oc adm node-logs --role=master --path=openshift-apiserver
    ```

    b. **/var/log/** サブディレクトリー内の特定ログを確認します。以下の例は、すべてのコントロールプレーンノードから **/var/log/openshift-apiserver/audit.log** コンテンツを出力します。

    ```
    $ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
    ```

    c. API が機能しない場合は、代わりに SSH を使用して各ノードのログを確認します。次の例では、**/var/log/openshift-apiserver/audit.log** の末尾を表示 (tail) します。

    ```
    $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
    /var/log/openshift-apiserver/audit.log
    ```

11. SSH を使用してコントロールプレーンノードのコンテナーログを確認します。

    a. コンテナーを一覧表示します。

    ```
    $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps -a
    ```

b. **crictl** を使用してコンテナーのログを取得します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

12. コントロールプレーンノードの設定に問題がある場合には、MCO、MCO エンドポイント、および DNS レコードが機能していることを確認します。Machine Config Operator (MCO) は、インストール時にオペレーティングシステムの設定を管理します。システムクロックの精度と証明書の有効性も確認します。

a. MCO エンドポイントが利用可能かどうかをテストします。**<cluster_name>** を適切な値に置き換えます。

```
$ curl https://api-int.<cluster_name>:22623/config/master
```

b. エンドポイントが応答しない場合は、ロードバランサーの設定を確認します。エンドポイントがポート 22623 で実行されるよう設定されていることを確認します。

c. MCO エンドポイントの DNS レコードが設定され、ロードバランサーに対して解決していることを確認します。

i. 定義された MCO エンドポイント名の DNS ルックアップを実行します。

```
$ dig api-int.<cluster_name> @<dns_server>
```

ii. ロードバランサーの割り当てられた MCO IP アドレスに対して逆引き参照を実行します。

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

d. MCO がブートストラップノードから直接機能していることを確認します。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/master
```

e. システムクロックは、ブートストラップ、マスター、およびワーカーノード間で同期される必要があります。各ノードのシステムクロックの参照時間と時刻同期の統計を確認します。

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

f. 証明書の有効性を確認します。

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

7.1.9. etcd インストールの問題の調査

インストール時に etcd の問題が発生した場合には、etcd Pod のステータスを確認し、etcd Pod ログを収集できます。etcd DNS レコードを確認し、コントロールプレーンノードで DNS の可用性を確認することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- コントロールプレーンノードの完全修飾ドメイン名がある。

手順

1. etcd Pod のステータスを確認します。
   a. **openshift-etcd** namespace の Pod のステータスを確認します。

      ```
      $ oc get pods -n openshift-etcd
      ```

   b. **openshift-etcd-operator** namespace の Pod のステータスを確認します。

      ```
      $ oc get pods -n openshift-etcd-operator
      ```

2. 直前のコマンドでリスト表示される Pod のいずれかに **Running** または **Completed** ステータスが表示されない場合は、Pod の診断情報を収集します。
   a. Pod のイベントを確認します。

      ```
      $ oc describe pod/<pod_name> -n <namespace>
      ```

   b. Pod のログを検査します。

      ```
      $ oc logs pod/<pod_name> -n <namespace>
      ```

   c. Pod に複数のコンテナーがある場合、前述のコマンドでエラーが作成され、コンテナー名はエラーメッセージに指定されます。各コンテナーのログを検査します。

      ```
      $ oc logs pod/<pod_name> -c <container_name> -n <namespace>
      ```

3. API が機能しない場合には、代わりに SSH を使用して各コントロールプレーンノードで etcd Pod およびコンテナーログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。
   a. 各コントロールプレーンノードに etcd Pod をリスト表示します。

      ```
      $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods --
      name=etcd-
      ```

   b. **Ready** ステータスが表示されない Pod について、Pod のステータスの詳細を検査します。**<pod_id>** を前述のコマンドの出力にリスト表示されている Pod の ID に置き換えます。

      ```
      $ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
      <pod_id>
      ```

   c. Pod に関連するコンテナーをリスト表示します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps | grep
'<pod_id>'
```

d. **Ready** ステータスが表示されていないコンテナーの場合は、コンテナーのステータスの詳細を検査します。**<container_id>** を前述のコマンドの出力にリスト表示されているコンテナー ID に置き換えます。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

e. **Ready** ステータスが表示されていないコンテナーのログを確認します。**<container_id>** を前述のコマンドの出力に一覧表示されているコンテナー ID に置き換えます。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

4. コントロールプレーンノードからプライマリーおよびセカンダリー DNS サーバー接続を検証します。

7.1.10. コントロールプレーンノードの kubelet および API サーバーの問題の調査

インストール時にコントロールプレーンノードの kubelet および API サーバーの問題を調査するには、DNS、DHCP、およびロードバランサーの機能を確認してください。また、証明書の有効期限が切れていないことを確認します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- コントロールプレーンノードの完全修飾ドメイン名がある。

手順

1. API サーバーの DNS レコードがコントロールプレーンノードの kubelet を **https://api-int.<cluster_name>.<base_domain>:6443** にダイレクトすることを確認します。レコードがロードバランサーを参照することを確認します。

2. ロードバランサーのポート 6443 定義が各コントロールプレーンノードを参照することを確認します。

3. DHCP によって固有のコントロールプレーンノードのホスト名が指定されていることを確認します。

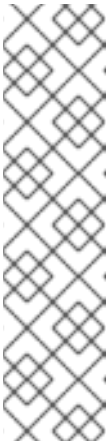
4. 各コントロールプレーンノードで **kubelet.service** journald ユニットログを検査します。

   a. **oc** を使用してログを取得します。

   ```
   $ oc adm node-logs --role=master -u kubelet
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
   ```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

5. コントロールプレーンノードの kubelet ログで証明書の有効期限のメッセージの有無を確認します。

   a. **oc** を使用してログを取得します。

   ```
   $ oc adm node-logs --role=master -u kubelet | grep -is 'x509: certificate has expired'
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service  | grep -is 'x509: certificate has expired'
   ```

7.1.11. ワーカーノードのインストールに関連する問題の調査

ワーカーノードのインストールに問題がある場合には、ワーカーノードのステータスを確認できます。**kubelet.service**、**crio.service** journald ユニットログ、およびワーカーノードコンテナーログを収集し、ワーカーノードエージェント、CRI-O コンテナーランタイム、および Pod アクティビティーを可視化します。さらに、Ignition ファイルおよびマシン API Operator の機能を確認することもできます。ワーカーノードのインストール後の設定が失敗する場合は、Machine Config Operator (MCO) および DNS 機能を確認します。また、ブートストラップ、マスター、およびワーカーノード間のシステムクロックの同期を確認し、証明書を検証することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ホストへの SSH アクセスがあること。
- ブートストラップおよびワーカーノードの完全修飾ドメイン名がある。
- HTTP サーバーを使用して Ignition 設定ファイルをホストする場合、HTTP サーバーの完全修飾ドメイン名およびポート番号が必要です。HTTP ホストへの SSH アクセスも必要です。

注記

初期の **kubeadmin** パスワードは、インストールホストの **<install_directory>/auth/kubeadmin-password** にあります。

手順

1. ワーカーノードのコンソールにアクセスできる場合は、ノードがログインプロンプトに到達するまでコンソールを監視します。インストール時に、Ignition ログメッセージはコンソールに出力されます。
2. Ignition ファイル設定を確認します。
   - HTTP サーバーを使用して Ignition 設定ファイルをホストする場合。
     a. ワーカーノードの Ignition ファイル URL を確認します。**<http_server_fqdn>** を HTTP サーバーの完全修飾ドメイン名に置き換えます。

     ```
     $ curl -I http://<http_server_fqdn>:<port>/worker.ign 1
     ```

     1 **-I** オプションはヘッダーのみを返します。Ignition ファイルが指定された URL で利用可能な場合、コマンドは **200 OK** ステータスを返します。これが利用できない場合は、コマンドは **404 file not found** を返します。

     b. Ignition ファイルがワーカーノードで受信されたことを確認するには、HTTP ホストの HTTP サーバーログのクエリーを実行します。たとえば、Apache Web サーバーを使用して Ignition ファイルを提供する場合は、以下を考慮してください。

     ```
     $ grep -is 'worker.ign' /var/log/httpd/access_log
     ```

     ワーカー Ignition ファイルが受信される場合、関連付けられた **HTTP GET** ログメッセージには要求が成功したことを示す **200 OK** の成功ステータスが含まれます。

     c. Ignition ファイルが受信されなかった場合、これが提供側ホストに存在することを直接確認します。適切なファイルおよび Web サーバーのパーミッションが適用されていることを確認します。

   - クラウドプロバイダーのメカニズムを使用して Ignition 設定ファイルを初期デプロイメントの一部としてホストに挿入する場合。
     a. ワーカーノードのコンソールを確認し、ワーカーノードの Ignition ファイルを正しく挿入するメカニズムが機能しているかどうかを確認します。

3. ワーカーノードの割り当てられたストレージデバイスの可用性を確認します。
4. ワーカーノードに DHCP サーバーから IP アドレスが割り当てられていることを確認します。
5. ワーカーノードのステータスを判別します。
   a. ノードのステータスのクエリーを実行します。

```
$ oc get nodes
```

   b. **Ready** ステータスが表示されないワーカーノードの詳細なノードの説明を取得します。

```
$ oc describe node <worker_node>
```

注記

インストールの問題により OpenShift Container Platform API が実行できなくなったり、kubelet が各ノードでまだ実行されていない場合、**oc** コマンドを実行することはできません。

6. コントロールプレーンノードとは異なり、ワーカーノードは Machine API Operator を使用してデプロイされ、スケーリングされます。Machine API Operator のステータスを確認します。
   a. Machine API Operator Pod のステータスを確認します。

```
$ oc get pods -n openshift-machine-api
```

   b. Machine API Operator Pod のステータスが **Ready** ではない場合は、Pod のイベントを詳細に作成します。

```
$ oc describe pod/<machine_api_operator_pod_name> -n openshift-machine-api
```

   c. **machine-api-operator** コンテナーログを検査します。コンテナーは **machine-api-operator** Pod 内で実行されます。

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c machine-api-operator
```

   d. また、**kube-rbac-proxy** コンテナーログも検査します。コンテナーは **machine-api-operator** Pod 内でも実行されます。

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c kube-rbac-proxy
```

7. **kubelet.service** journald ユニットログを、起動後のワーカーノードでモニターします。これにより、ワーカーノードエージェントのアクティビティーを可視化できます。
   a. **oc** を使用してログを取得します。

```
$ oc adm node-logs --role=worker -u kubelet
```

b. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<worker-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

8. 起動後のワーカーノードで **crio.service** journald ユニットログを取得します。これにより、ワーカーノードの CRI-O コンテナーランタイムのアクティビティーを可視化できます。

   a. **oc** を使用してログを取得します。

   ```
   $ oc adm node-logs --role=worker -u crio
   ```

   b. API が機能しない場合は、代わりに SSH を使用してログを確認します。

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u
   crio.service
   ```

9. ワーカーノードの **/var/log/** の下にある特定のサブディレクトリーからログを収集します。

   a. **/var/log/** サブディレクトリー内に含まれるログの一覧を取得します。以下の例は、すべてのワーカーノードの **/var/log/sssd/** にあるファイルをリスト表示します。

   ```
   $ oc adm node-logs --role=worker --path=sssd
   ```

   b. **/var/log/** サブディレクトリー内の特定ログを確認します。次の例では、すべてのワーカーノードの **/var/log/sssd/sssd.log** の内容を出力します。

   ```
   $ oc adm node-logs --role=worker --path=sssd/sssd.log
   ```

   c. API が機能しない場合は、代わりに SSH を使用して各ノードのログを確認します。次の例では、**/var/log/sssd/sssd.log** の末尾を表示 (tail) します。

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo tail -f
   /var/log/sssd/sssd.log
   ```

10. SSH を使用してワーカーノードのコンテナーログを確認します。

    a. コンテナーを一覧表示します。

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl ps -a
```

b. **crictl** を使用してコンテナーのログを取得します。

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

11. ワーカーノードの設定に問題がある場合には、MCO、MCO エンドポイント、および DNS レコードが機能していることを確認します。Machine Config Operator (MCO) は、インストール時にオペレーティングシステムの設定を管理します。システムクロックの精度と証明書の有効性も確認します。

a. MCO エンドポイントが利用可能かどうかをテストします。**<cluster_name>** を適切な値に置き換えます。

```
$ curl https://api-int.<cluster_name>:22623/config/worker
```

b. エンドポイントが応答しない場合は、ロードバランサーの設定を確認します。エンドポイントがポート 22623 で実行されるよう設定されていることを確認します。

c. MCO エンドポイントの DNS レコードが設定され、ロードバランサーに対して解決していることを確認します。

i. 定義された MCO エンドポイント名の DNS ルックアップを実行します。

```
$ dig api-int.<cluster_name> @<dns_server>
```

ii. ロードバランサーの割り当てられた MCO IP アドレスに対して逆引き参照を実行します。

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

d. MCO がブートストラップノードから直接機能していることを確認します。**<bootstrap_fqdn>** をブートストラップノードの完全修飾ドメイン名に置き換えます。

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/worker
```

e. システムクロックは、ブートストラップ、マスター、およびワーカーノード間で同期される必要があります。各ノードのシステムクロックの参照時間と時刻同期の統計を確認します。

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

f. 証明書の有効性を確認します。

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

7.1.12. インストール後の Operator ステータスのクエリー

インストールの終わりに Operator のステータスを確認できます。利用できない Operator の診断データを取得します。**Pending** と一覧表示されているか、エラーステータスのある Operator Pod のログを確認します。問題のある Pod によって使用されるベースイメージを検証します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. クラスター Operator がすべてインストールの終わりに利用可能な状態であることを確認します。

```
$ oc get clusteroperators
```

2. 必要な証明書署名要求 (CSR) がすべて承認されていることを確認します。一部のノードは **Ready** ステータスには移行さず、一部のクラスター Operator は保留中の CSR がある場合に利用できない可能性があります。

   a. CSR のステータスを確認し、クラスターに追加したそれぞれのマシンのクライアントおよびサーバー要求に **Pending** または **Approved** ステータスが表示されていることを確認します。

```
$ oc get csr
```

   出力例

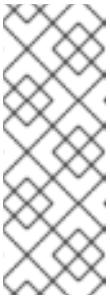
```
NAME        AGE     REQUESTOR                                                                   CONDITION
csr-8b2br   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending 1
csr-8vnps   15m     system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
csr-bfd72   5m26s   system:node:ip-10-0-50-126.us-east-2.compute.internal                        Pending 2
csr-c57lv   5m26s   system:node:ip-10-0-95-157.us-east-2.compute.internal                        Pending
...
```

   1 クライアント要求の CSR。

   2 サーバー要求の CSR。

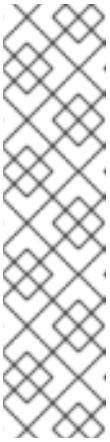
   この例では、2 つのマシンがクラスターに参加しています。このリストにはさらに多くの承認された CSR が表示される可能性があります。

   b. 追加したマシンの保留中の CSR すべてが **Pending** ステータスになった後に CSR が承認されない場合には、クラスターマシンの CSR を承認します。



注記

CSR のローテーションは自動的に実行されるため、クラスターにマシンを追加後 1 時間以内に CSR を承認してください。1 時間以内に承認しない場合には、証明書のローテーションが行われ、各ノードに 3 つ以上の証明書が存在するようになります。これらの証明書すべてを承認する必要があります。最初の CSR の承認後、後続のノードクライアント CSR はクラスターの **kube-controller-manager** によって自動的に承認されます。



注記

ベアメタルおよび他の user-provisioned infrastructure などのマシン API ではないプラットフォームで実行されているクラスターの場合、kubelet 提供証明書要求 (CSR) を自動的に承認する方法を実装する必要があります。要求が承認されない場合、API サーバーが kubelet に接続する際に提供証明書が必須であるため、**oc exec**、**oc rsh**、および **oc logs** コマンドは正常に実行できません。Kubelet エンドポイントにアクセスする操作には、この証明書の承認が必要です。この方法は新規 CSR の有無を監視し、CSR が **system:node** または **system:admin** グループの **node-bootstrapper** サービスアカウントによって提出されていることを確認し、ノードの ID を確認します。

- それらを個別に承認するには、それぞれの有効な CSR に以下のコマンドを実行します。

```
$ oc adm certificate approve <csr_name> 1
```

1 **<csr_name>** は、現行の CSR のリストからの CSR の名前です。

- すべての保留中の CSR を承認するには、以下のコマンドを実行します。

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

3. Operator イベントを表示します。

```
$ oc describe clusteroperator <operator_name>
```

4. Operator の namespace 内で Operator Pod のステータスを確認します。

```
$ oc get pods -n <operator_namespace>
```

5. **Running** ステータスを持たない Pod の詳細な説明を取得します。

```
$ oc describe pod/<operator_pod_name> -n <operator_namespace>
```

6. Pod ログを検査します。

```
$ oc logs pod/<operator_pod_name> -n <operator_namespace>
```

7. Pod ベースイメージに関連する問題が発生した場合には、ベースイメージのステータスを確認します。

a. 問題のある Pod で使用されるベースイメージの詳細を取得します。

```
$ oc get pod -o "jsonpath={range .status.containerStatuses[*]}{.name}{'\t'}{.state}{'\t'}
{.image}{'\n'}{end}" <operator_pod_name> -n <operator_namespace>
```

b. ベースイメージのリリース情報をリスト表示します。

```
$ oc adm release info <image_path>:<tag> --commits
```

7.1.13. 失敗したインストールのログの収集

SSH キーをインストールプログラムに指定している場合は、失敗したインストールに関するデータを収集できます。

注記

実行中のクラスターからログを収集する場合とは異なるコマンドを使用して失敗したインストールに関するログを収集します。実行中のクラスターからログを収集する必要がある場合は、**oc adm must-gather** コマンドを使用します。

前提条件

- OpenShift Container Platform のインストールがブートストラッププロセスの終了前に失敗している。ブートストラップノードは実行中で、SSH でアクセスできる。
- **ssh-agent** プロセスはコンピューター上でアクティブであり、**ssh-agent** プロセスとインストールプログラムの両方に同じ SSH キーを提供している。
- 独自にプロビジョニングしたインフラストラクチャーにクラスターのインストールを試行した場合は、ブートストラップおよびコントロールプレーンノードの完全修飾ドメイン名がある。

手順

1. ブートストラップおよびコントロールプレーンマシンからインストールログを収集するために必要なコマンドを生成します。

   - installer-provisioned infrastructure を使用する場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> 1
     ```

     1 **installation_directory** は、**./openshift-install create cluster** を実行した際に指定したディレクトリーです。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

     installer-provisioned infrastructure の場合、インストールプログラムは、ホスト名または IP アドレスを指定しなくてもよいようにクラスターに関する情報を保存します。

   - 各自でプロビジョニングしたインフラストラクチャーを使用した場合は、インストールプログラムが含まれるディレクトリーに切り替え、以下のコマンドを実行します。

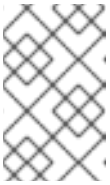
     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
       --bootstrap <bootstrap_address> \ 2
     ```

```
--master <master_1_address> \ 3
--master <master_2_address> \ 4
--master <master_3_address> 5
```

1 **installation_directory** には、**./openshift-install create cluster** を実行した際に指定したのと同じディレクトリーを指定します。このディレクトリーには、インストールプログラムが作成する OpenShift Container Platform 定義ファイルが含まれます。

2 **<bootstrap_address>** は、クラスターのブートストラップマシンの完全修飾ドメイン名または IP アドレスです。

3 4 5 クラスター内のそれぞれのコントロールプレーン (またはマスター) マシンで、**<master_*_address>** をその完全修飾ドメイン名または IP アドレスに置き換えます。



注記

デフォルトクラスターには 3 つのコントロールプレーンマシンが含まれます。クラスターが使用する数にかかわらず、表示されるようにすべてのコントロールプレーンマシンをリスト表示します。

出力例

```
INFO Pulling debug logs from the bootstrap machine
INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
<timestamp>.tar.gz"
```

インストールの失敗に関する Red Hat サポートケースを作成する場合は、圧縮したログをケースに含めるようにしてください。

7.1.14. 関連情報

- OpenShift Container Platform のインストールタイプおよびプロセスに関する詳細は、インストールプロセス を参照してください。

7.2. ノードの健全性の確認

7.2.1. ノードのステータス、リソースの使用状況および設定の確認

クラスターノードの健全性ステータス、リソース消費統計、およびノードログを確認します。さらに、個別ノードの **kubelet** ステータスのクエリーを実行します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

- クラスターのすべてのノードの名前、ステータスおよびロールをリスト表示します。

```
$ oc get nodes
```

- クラスター内の各ノードの CPU およびメモリーの使用状況を要約します。

```
$ oc adm top nodes
```

- 特定のノードの CPU およびメモリーの使用状況を要約します。

```
$ oc adm top node my-node
```

7.2.2. ノードにおける kubelet ステータスのクエリー

クラスターノードの健全性ステータス、リソース消費統計、およびノードログを確認できます。さらに、個別のノードで **kubelet** ステータスをクエリーできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. kubelet は各ノードの systemd サービスを使用して管理されます。デバッグ Pod 内で **kubelet** systemd サービスをクエリーし、kubelet のステータスを確認します。

   a. ノードのデバッグ Pod を起動します。

```
$ oc debug node/my-node
```

注記

コントロールプレーンノードで **oc debug** を実行している場合は、**/etc/kubernetes/static-pod-resources/kube-apiserver-certs/secrets/node-kubeconfigs** ディレクトリーに管理用 **kubeconfig** ファイルがあります。

   b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、**kubelet** がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

c. **kubelet** systemd サービスがノードでアクティブかどうかを確認します。

```
# systemctl is-active kubelet
```

d. より詳細な **kubelet.service** ステータスの要約を出力します。

```
# systemctl status kubelet
```

7.2.3. クラスターノードジャーナルログのクエリー

個別のクラスターノードの **/var/log** 内で **journald** ユニットログおよびその他のログを収集できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- API サービスが機能している。
- ホストへの SSH アクセスがあること。

手順

1. OpenShift Container Platform クラスターノードに対して、**kubelet** の **journald** ユニットログのクエリーを実行します。以下の例では、コントロールプレーンノードのみがクエリーされます。

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 クエリーを実行して他のユニットログを取得するために、**kubelet** を適宜置き換えます。

2. クラスターノードの **/var/log/** の下にある特定のサブディレクトリーからログを収集します。

a. **/var/log/** サブディレクトリー内に含まれるログの一覧を取得します。以下の例では、すべてのコントロールプレーンノードの **/var/log/openshift-apiserver/** にあるファイルをリスト表示します。

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

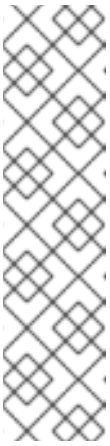
b. /var/log/ サブディレクトリー内の特定ログを確認します。以下の例は、すべてのコント

b. **/var/log/** サブディレクトリー内の特定ログを確認します。以下の例は、すべてのコントロールプレーンノードから **/var/log/openshift-apiserver/audit.log** コンテンツを出力します。

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. API が機能しない場合は、代わりに SSH を使用して各ノードのログを確認します。次の例では、**/var/log/openshift-apiserver/audit.log** の末尾を表示 (tail) します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

7.3. CRI-O コンテナーランタイムの問題のトラブルシューティング

7.3.1. CRI-O コンテナーランタイムエンジンについて

CRI-O は Kubernetes ネイティブコンテナーエンジン実装です。これはオペレーティングシステムに密接に統合し、Kubernetes の効率的で最適化されたエクスペリエンスを提供します。CRI-O コンテナーエンジンは、各 OpenShift Container Platform クラスターノードで systemd サービスとして実行されます。

コンテナーランタイムの問題が発生する場合は、各ノードの **crio** systemd サービスのステータスを確認します。コンテナーのランタイムに問題があるノードから CRI-O journald ユニットログを収集します。

7.3.2. CRI-O ランタイムエンジンのステータスの確認

各クラスターノードで CRI-O コンテナーランタイムエンジンのステータスを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

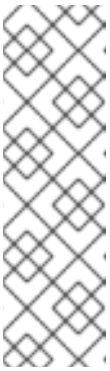
1. デバッグ Pod 内で、ノードの **crio** systemd サービスをクエリーして CRI-O ステータスを確認します。

a. ノードのデバッグ Pod を起動します。

```
$ oc debug node/my-node
```

b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

c. **crio** systemd サービスがノードでアクティブかどうかを確認します。

```
# systemctl is-active crio
```

d. より詳細な **crio.service** ステータスの要約を出力します。

```
# systemctl status crio.service
```

7.3.3. CRI-O の journald ユニットログの収集

CRI-O の問題が発生した場合には、ノードから CRI-O journald ユニットログを取得できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。
- コントロールプレーンまたはコントロールプレーンマシンの完全修飾ドメイン名がある。

手順

1. CRI-O journald ユニットログを収集します。以下の例は、クラスター内のすべてのコントロールプレーンノードからログを収集します。

```
$ oc adm node-logs --role=master -u crio
```

2. 特定のノードから CRI-O journald ユニットログを収集します。

```
$ oc adm node-logs <node_name> -u crio
```

3. API が機能しない場合は、代わりに SSH を使用してログを確認します。**<node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

```
$ ssh core@<node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

7.3.4. CRI-O ストレージの消去

以下の問題が発生した場合、CRI-O の一時ストレージを手動でクリアすることができます。

- ノードは Pod を実行できず、次のエラーが表示されます。

```
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to mount container XXX: error recreating the missing symlinks: error reading name of symlink for XXX: open /var/lib/containers/storage/overlay/XXX/link: no such file or directory
```

- 作業ノードに新しいコンテナーを作成することができず、“can’t stat lower layer” というエラーが表示される。

```
can't stat lower layer ...  because it does not exist.  Going through storage to recreate the missing symlinks.
```

- クラスターをアップグレードした後、またはノードを再起動しようとすると、ノードが **NotReady** 状態になる。
- コンテナーランタイム実装 (**crio**) が正しく動作していない。
- コンテナーランタイムインスタンス (**crio**) が動作していないため、**oc debug node/<node_name>** を使用してノード上でデバッグシェルを開始できない。

この手順で、CRI-O のストレージを完全に消去し、エラーを解消してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

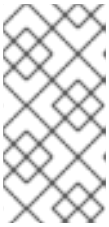
手順

1. ノードで **cordon** を使用します。これは、ノードが **Ready** 状態になった場合に、ワークロードがスケジューリングされるのを防ぐためです。Status セクションに **SchedulingDisabled** と表示されていれば、スケジューリングが無効になっていることがわかります。

```
$ oc adm cordon <node_name>
```

2. cluster-admin ユーザーとして、ノードをドレインします。

```
$ oc adm drain <node_name> --ignore-daemonsets --delete-emptydir-data
```



注記

Pod または Pod テンプレートの **terminationGracePeriodSeconds** 属性は、正常な終了期間を制御します。この属性のデフォルトは 30 秒ですが、必要に応じてアプリケーションごとにカスタマイズできます。90 秒を超えて設定すると、Pod が **SIGKILLed** とマークされ、正常に終了しない可能性があります。

3. ノードが戻ってきたら、SSH またはコンソールでノードに接続し直します。その後、root ユーザーで接続します。

```
$ ssh core@node1.example.com
$ sudo -i
```

4. kubelet を手動で停止します。

```
# systemctl stop kubelet
```

5. コンテナーや Pod を停止します。

   a. 以下のコマンドを使用して、**HostNetwork** にない Pod を停止します。これらが削除されるかどうかは、**HostNetwork** にあるネットワークプラグイン Pod に左右されるので、先に削除する必要があります。

```
.. for pod in $(crictl pods -q); do if [[ "$(crictl inspectp $pod | jq -r
.status.linux.namespaces.options.network)" != "NODE" ]]; then crictl rmp -f $pod; fi; done
```

   b. 他のすべての Pod を停止します。

```
# crictl rmp -fa
```

6. crio のサービスを手動で停止します。

```
# systemctl stop crio
```

7. これらのコマンドを実行すると、一時ストレージを完全に消去することができます。

```
# crio wipe -f
```

8. crio および kubelet サービスを起動します。

```
# systemctl start crio
# systemctl start kubelet
```

9. crio および kubelet サービスが起動しており、ノードが **Ready** のステータスになっている場合には、クリーンアップが正常に機能したことが分かります。

```
$ oc get nodes
```

出力例

```
NAME       STATUS                ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1  Ready, SchedulingDisabled   master  133m   v1.31.3
```

10. ノードをスケジューリング可能な状態にします。スケジューリングが有効になったことは、**SchedulingDisabled** のステータスがなくなったときにわかります。

```
$ oc adm uncordon <node_name>
```

出力例

```
NAME       STATUS      ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1   Ready          master   133m   v1.31.3
```

7.4. オペレーティングシステムの問題のトラブルシューティング

OpenShift Container Platform は RHCOS で実行されます。以下の手順に従って、オペレーティングシステムに関連する問題のトラブルシューティングを行うことができます。

7.4.1. カーネルクラッシュの調査

kexec-tools パッケージに含まれる **kdump** サービスは、クラッシュダンプメカニズムを提供します。このサービスを使用して、後で分析するためにシステムのメモリーの内容を保存できます。

x86_64 アーキテクチャーは一般提供 (GA) ステータスの kdump をサポートしますが、他のアーキテクチャーはテクノロジープレビュー (TP) ステータスの kdump をサポートします。

次の表は、さまざまなアーキテクチャーの kdump のサポートレベルに関する詳細を示しています。

表7.1 RHCOS での Kdump サポート

アーキテクチャー	サポートレベル
x86_64	GA
aarch64	TP
s390x	TP
ppc64le	TP

重要

表の中の、最初の 3 つのアーキテクチャーに対する Kdump サポートは、テクノロジープレビュー機能のみになります。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、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範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

7.4.1.1. kdump の有効化

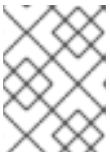
RHCOS には **kexec-tools** パッケージが同梱されていますが、**kdump** サービスを有効にするには手動で設定する必要があります。

手順

RHCOS で kdump を有効にするには、以下の手順を実行します。

1. 初回のカーネルの起動時にクラッシュカーネルのメモリーを確保するには、以下のコマンドを入力してカーネル引数を指定します。

```
# rpm-ostree kargs --append='crashkernel=256M'
```



注記

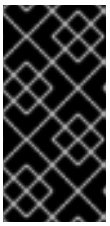
ppc64le プラットフォームの場合、**crashkernel** の推奨値は **crashkernel=2G-4G:384M,4G-16G:512M,16G-64G:1G,64G-128G:2G,128G-:4G** です。

2. オプション: デフォルトのローカルの **/var/crash** の場所ではなく、ネットワーク経由または他の場所にクラッシュダンプを書き込むには、**/etc/kdump.conf** 設定ファイルを編集します。

注記

ノードが LUKS 暗号化デバイスを使用している場合、kdump は LUKS 暗号化デバイスへのクラッシュダンプの保存をサポートしていないため、ネットワークダンプを使用する必要があります。

kdump サービスの設定の詳細は、**/etc/sysconfig/kdump**、**/etc/kdump.conf**、および **kdump.conf** manual ページのコメントを参照してください。ダンプターゲットの設定に関する詳細は、RHEL kdump documentation も参照してください。



重要

プライマリーディスクでマルチパスが有効になっている場合、ダンプターゲットは NFS サーバーまたは SSH サーバーのいずれかである必要があり、**/etc/kdump.conf** 設定ファイルからマルチパスモジュールを除外する必要があります。

3. **kdump** systemd サービスを有効にします。

```
# systemctl enable kdump.service
```

4. システムを再起動します。

```
# systemctl reboot
```

5. **kdump.service** systemd サービスが正常に開始および終了したこと、および **cat /sys/kernel/kexec_crash_loaded** コマンドが値 **1** を出力することを確認して、kdump がクラッシュカーネルをロードしたことを確認します。

7.4.1.2. Day-1 の kdump の有効化

kdump サービスは、カーネルの問題をデバッグするために、ノードごとに有効にすることが意図されています。kdump の有効化にはコストがかかり、これらのコストは kdump が有効化されたノードを追加するたびに増えるため、**kdump** サービスを必要な場合にのみ各ノードで有効にすことが推奨されます。各ノードで **kdump** サービスを有効にする場合の潜在的なコストには、以下が含まれます。

- クラッシュカーネル用にメモリーが予約されているため、利用可能な RAM が少ない。
- カーネルがコアをダンプしている間にノードが利用できなくなる。
- 追加のストレージ容量がクラッシュダンプを保存するために使用される。

kdump サービスを有効にすることに伴う不利な点やトレードオフを把握している場合には、クラスター全体で kdump を有効にすることができます。マシン固有のマシン設定はまだサポートされていませんが、Day 1 のカスタマイズとして **MachineConfig** オブジェクトで **systemd** ユニットを使用し、クラスター内のすべてのノードで kdump を有効にすることができます。**MachineConfig** オブジェクトを作成し、そのオブジェクトをクラスターのセットアップ時に Ignition が使用するマニフェストファイルのセットに挿入することができます。

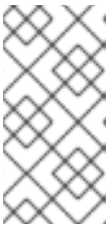
注記

Ignition 設定の使用方法の詳細は、**インストール → インストール設定** セクションの「ノードのカスタマイズ」を参照してください。

手順

クラスター全体の設定の **MachineConfig** オブジェクトを作成します。

1. kdump を設定および有効にする Butane 設定ファイル **99-worker-kdump.bu** を作成します。



注記

設定ファイルで指定する Butane のバージョン は、OpenShift Container Platform のバージョンと同じである必要があり、末尾は常に **0** です。たとえば、**4.18.0** です。Butane の詳細は、「Butane を使用したマシン設定の作成」を参照してください。

```
variant: openshift
version: 4.18.0
metadata:
  name: 99-worker-kdump (1)
  labels:
```

```
    machineconfiguration.openshift.io/role: worker (2)
openshift:
  kernel_arguments: (3)
    - crashkernel=256M
storage:
  files:
    - path: /etc/kdump.conf (4)
      mode: 0644
      overwrite: true
      contents:
        inline: |
          path /var/crash
          core_collector makedumpfile -l --message-level 7 -d 31

    - path: /etc/sysconfig/kdump (5)
      mode: 0644
      overwrite: true
      contents:
        inline: |
          KDUMP_COMMANDLINE_REMOVE="hugepages hugepagesz slub_debug quiet
log_buf_len swiotlb"
          KDUMP_COMMANDLINE_APPEND="irqpoll nr_cpus=1 reset_devices
cgroup_disable=memory mce=off numa=off udev.children-max=2 panic=10 rootflags=nofail
acpi_no_memhotplug transparent_hugepage=never nokaslr novmcoredd hest_disable" (6)
          KEXEC_ARGS="-s"
          KDUMP_IMG="vmlinuz"

systemd:
  units:
    - name: kdump.service
      enabled: true
```

(1) (2) コントロールプレーンノードの **MachineConfig** オブジェクトの作成時に、両方の場所にある **worker** を **master** に置き換えます。

(3) カーネル引数を指定して、クラッシュカーネル用にメモリーを予約します。必要に応じて、他のカーネル引数を追加できます。**ppc64le** プラットフォームの場合、**crashkernel** の推奨値は **crashkernel=2G-4G:384M,4G-16G:512M,16G-64G:1G,64G-128G:2G,128G-:4G** です。

(4) **/etc/kdump.conf** の内容をデフォルトから変更する場合は、このセクションを追加し、それに応じて **inline** サブセクションを変更します。

(5) **/etc/sysconfig/kdump** の内容をデフォルトから変更する場合は、このセクションを追加し、それに応じて **inline** サブセクションを変更します。

(6) **ppc64le** プラットフォームの場合は、**nr_cpus=1** を **maxcpus=1** に置き換えますが、これはこのプラットフォームではサポートされていません。

注記

ダンプを NFS ターゲットにエクスポートするには、一部のカーネルモジュールを明示的に設定ファイルに追加する必要があります。

/etc/kdump.conf ファイルの例

```
nfs server.example.com:/export/cores
core_collector makedumpfile -l --message-level 7 -d 31
extra_bins /sbin/mount.nfs
extra_modules nfs nfsv3 nfs_layout_nfsv41_files blocklayoutdriver nfs_layout_flexfiles
nfs_layout_nfsv41_files
```

1. Butane を使用して、ノードに配信する設定が含まれる、マシン設定 YAML ファイル (**99-worker-kdump.yaml**) を生成します。

```
$ butane 99-worker-kdump.bu -o 99-worker-kdump.yaml
```

2. クラスターのセットアップ中に、YAML ファイルを **<installation_directory>/manifests/** ディレクトリーに配置します。YAML ファイルを使用してクラスターの設定後にこの **MachineConfig** オブジェクトを作成することもできます。

```
$ oc create -f 99-worker-kdump.yaml
```

7.4.1.3. kdump 設定のテスト

kdump は、RHEL ドキュメントの Testing the kdump configuration セクションを参照してください。

7.4.1.4. コアダンプの分析

kdump は、RHEL ドキュメントの Analyzing a core dump セクションを参照してください。

注記

別の RHEL システムで vmcore 分析を実行することが推奨されます。

7.4.1.5. 関連情報

- RHEL での kdump の設定
- kdump に関する Linux カーネルドキュメント
- kdump.conf(5): 利用可能なオプションの詳細なドキュメントを含む **/etc/kdump.conf** 設定ファイルの man ページ
- kexec(8) – **kexec** パッケージの man ページ
- kexec および kdump に関する Red Hat ナレッジの記事

7.4.2. Ignition の失敗のデバッグ

マシンをプロビジョニングできない場合、Ignition は失敗し、RHCOS は緊急シェルで起動します。デバッグ情報を取得するには、次の手順を使用します。

手順

1. 次のコマンドを実行して、失敗したサービスユニットを表示します。

```
$ systemctl --failed
```

2. オプション: 個々のサービスユニットで次のコマンドを実行して、詳細を確認します。

```
$ journalctl -u <unit>.service
```

7.5. ネットワーク関連の問題のトラブルシューティング

7.5.1. ネットワークインターフェイスの選択方法

ベアメタルでのインストールや、複数のネットワークインターフェイスコントローラー (NIC) でのインストールの場合に、OpenShift Container Platform が Kubernetes API サーバーとの通信に使用する NIC は、ノードの起動時に systemd で実行される **nodeip-configuration.service** サービスユニットによって決定されます。**nodeip-configuration.service** は、デフォルトルートに関連付けられたインターフェイスから IP を選択します。

nodeip-configuration.service サービスが正しい NIC を決定すると、このサービスは **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** ファイルを作成します。**20-nodenet.conf** ファイルは、**KUBELET_NODE_IP** 環境変数を、サービスが選択した IP アドレスに設定します。

kubelet サービスの起動時に、**20-nodenet.conf** ファイルから環境変数の値を読み取り、IP アドレスを **--node-ip** kubelet コマンドライン引数に設定します。その結果、kubelet サービスは選択した IP アドレスをノード IP アドレスとして使用します。

インストール後にハードウェアまたはネットワークが再設定された場合、またはノード IP がデフォルトルートインターフェイスから取得されないネットワークレイアウトがある場合、再起動後に **nodeip-configuration.service** サービスが別の NIC を選択する可能性があります。 **oc get nodes -o wide** コマンドの出力の **INTERNAL-IP** 列を確認して、別の NIC が選択されていることを確認できる場合があります。

別の NIC が選択されているためにネットワーク通信が中断または誤って設定されている場合、次のエラーが表示されることがあります: **EtcdCertSignerControllerDegraded**。**NODEIP_HINT** 変数を含むヒントファイルを作成して、デフォルトの IP 選択ロジックをオーバーライドできます。詳細は、オプション: デフォルトのノード IP 選択ロジックのオーバーライドを参照してください。

7.5.1.1. オプション: デフォルトのノード IP 選択ロジックのオーバーライド

デフォルトの IP 選択ロジックをオーバーライドするには、**NODEIP_HINT** 変数を含むヒントファイルを作成して、デフォルトの IP 選択ロジックをオーバーライドします。ヒントファイルを作成すると、**NODEIP_HINT** 変数で指定された IP アドレスのサブネット内のインターフェイスから特定のノード IP アドレスを選択できます。

たとえば、ノードに **10.0.0.10/24** のアドレスを持つ **eth0** と **192.0.2.5/24** のアドレスを持つ **eth1** の 2 つのインターフェイスがあり、デフォルトルートが **eth0** (**10.0.0.10**) を指している場合、ノードの IP アドレスは通常、**10.0.0.10** IP アドレスを使用します。

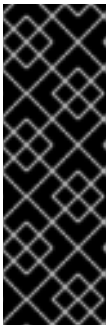
ユーザーは、別のサブネット **192.0.2.0/24** が選択されるように、サブネット内の既知の IP (たとえば、**192.0.2.1** などのサブネットゲートウェイ) を指すように **NODEIP_HINT** 変数を設定できます。その結果、**eth1** の **192.0.2.5** IP アドレスがノードに使用されます。

次の手順は、デフォルトのノード IP 選択ロジックをオーバーライドする方法を示しています。

手順

1. **/etc/default/nodeip-configuration** ファイルにヒントファイルを追加します。たとえば、以下のようになります。

```
NODEIP_HINT=192.0.2.1
```



重要

- **192.0.2.5** など、ノードの正確な IP アドレスをヒントとして使用しないでください。ノードの正確な IP アドレスを使用すると、ヒント IP アドレスを使用するノードが正しく設定できなくなります。
- ヒントファイル内の IP アドレスは、正しいサブネットを決定するためにのみ使用されます。ヒントファイルに表示されるため、トラフィックを受信しません。

2. 次のコマンドを実行して、**base-64** でエンコードされたコンテンツを生成します。

```
$ echo -n 'NODEIP_HINT=192.0.2.1' | base64 -w0
```

出力例

```
Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxxx==
```

3. クラスターをデプロイする前に、**master** ロールと **worker** ロールの両方のマシン設定マニフェストを作成して、ヒントを有効にします。

99-nodeip-hint-master.yaml

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-nodeip-hint-master
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,<encoded_content> 1
        mode: 0644
        overwrite: true
        path: /etc/default/nodeip-configuration
```

1 **<encoded_contents>** を **/etc/default/nodeip-configuration** ファイルの base64 でエンコードされたコンテンツに置き換えます (例: **Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxx==**)。コンマの後およびエンコードされたコンテンツの前にスペースを入れることはできないことに注意してください。

99-nodeip-hint-worker.yaml

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
 labels:
   machineconfiguration.openshift.io/role: worker
   name: 99-nodeip-hint-worker
spec:
 config:
   ignition:
     version: 3.2.0
   storage:
     files:
     - contents:
         source: data:text/plain;charset=utf-8;base64,<encoded_content> 1
       mode: 0644
       overwrite: true
       path: /etc/default/nodeip-configuration
```

1 **<encoded_contents>** を **/etc/default/nodeip-configuration** ファイルの base64 でエンコードされたコンテンツに置き換えます (例: **Tk9ERUlQX0hJTlQ9MTkyLjAuMCxxxx==**)。コンマの後およびエンコードされたコンテンツの前にスペースを入れることはできないことに注意してください。

4. ~/**clusterconfigs** など、クラスター設定を保存するディレクトリーにマニフェストを保存します。

5. クラスターのデプロイ

7.5.1.2. セカンダリー OVS ブリッジを使用するように OVN-Kubernetes を設定する

追加または **セカンダリー** の Open vSwitch (OVS) ブリッジ **br-ex1** を作成できます。このブリッジは、OVN-Kubernetes によって管理され、Multiple External Gateways (MEG) 実装が OpenShift Container Platform ノードの外部ゲートウェイを定義するために使用するものです。MEG は、**AdminPolicyBasedExternalRoute** カスタムリソース (CR) で定義できます。MEG 実装は、複数のゲートウェイ、等コストマルチパス (ECMP) ルート、および双方向フォワーディング検出 (BFD) 実装へのアクセスを Pod に提供します。

Multiple External Gateways (MEG) 機能の影響を受ける Pod のユースケースを考えてみてください。たとえば、ノード上の別のインターフェイス (**br-ex1** など) に Egress トラフィックを送信するとします。MEG の影響を受けない Pod の Egress トラフィックは、デフォルトの OVS **br-ex** ブリッジにルーティングされます。

重要

現在、MEG は、Egress IP、Egress ファイアウォール、Egress ルーターなどの他の Egress 機能との使用はサポートされていません。Egress IP などの Egress 機能で MEG を使用しようとすると、ルーティングとトラフィックフローの競合が発生する可能性があります。これは、OVN-Kubernetes がルーティングと送信元ネットワークアドレス変換 (SNAT) を処理する方法が原因で発生します。その結果、ルーティングに一貫性がなくなり、戻りパスが着信パスをパッチする必要がある一部の環境では接続が切断される可能性があります。

追加のブリッジは、マシン設定マニフェストファイルのインターフェイス定義で定義する必要があります。Machine Config Operator が、このマニフェストを使用して、ホスト上の **/etc/ovnk/extra_bridge** に新しいファイルを作成します。この新しいファイルには、追加の OVS ブリッジがノード用に設定するネットワークインターフェイスの名前が含まれています。

マニフェストファイルを作成して編集すると、Machine Config Operator は次の順序でタスクを完了します。

1. 選択したマシン設定プールに基づいて、単一の順序でノードをドレインします。
2. 各ノードに Ignition 設定ファイルを挿入し、各ノードが追加の **br-ex1** ブリッジネットワーク設定を受信するようにします。
3. **br-ex** の MAC アドレスが、 **br-ex** がネットワーク接続に使用するインターフェイスの MAC アドレスと一致していることを確認します。
4. 新しいインターフェイス定義を参照する **configure-ovs.sh** シェルスクリプトを実行します。
5. **br-ex** と **br-ex1** をホストノードに追加します。
6. ノードをスケジューリング対象に戻します。

注記

すべてのノードが **Ready** 状態に戻り、OVN-Kubernetes Operator が **br-ex** と **br-ex1** を検出して設定した後、Operator は各ノードに **k8s.ovn.org/l3-gateway-config** アノテーションを適用します。

追加の **br-ex1** ブリッジが役立つ状況と、デフォルトの **br-ex** ブリッジが常に必要な状況の詳細は、「ローカルネットトポロジーの設定」を参照してください。

手順

1. オプション: 次の手順を実行して、追加のブリッジ **br-ex1** が使用できるインターフェイス接続を作成します。この手順の例では、新しいボンディングとその依存インターフェイスを作成する方法を示しています。これらはすべて、マシン設定マニフェストファイルで定義されています。追加のブリッジは、**MachineConfig** オブジェクトを使用して追加のボンディングインターフェイスを形成します。

重要

追加のインターフェイスを定義するために、Kubernetes NMState Operator や **NodeNetworkConfigurationPolicy** (NNCP) マニフェストファイルを使用しないでください。

また、**bond** インターフェイスを定義するときには、追加のインターフェイスまたはサブインターフェイスが既存の **br-ex** OVN Kubernetes ネットワークデプロイメントで使用されていないことを確認してください。

a. 次のインターフェイス定義ファイルを作成します。以下のファイルは、ホストノードが定義ファイルにアクセスできるように、マシン設定マニフェストファイルに追加されます。

1番目のインターフェイス定義ファイルの例 (eno1.config)

```
[connection]
id=eno1
type=ethernet
interface-name=eno1
master=bond1
slave-type=bond
autoconnect=true
autoconnect-priority=20
```

2番目のインターフェイス定義ファイルの例 (eno2.config)

```
[connection]
id=eno2
type=ethernet
interface-name=eno2
master=bond1
slave-type=bond
autoconnect=true
autoconnect-priority=20
```

2番目のボンディングインターフェイス定義ファイルの例 (bond1.config)

```
[connection]
id=bond1
type=bond
interface-name=bond1
autoconnect=true
connection.autoconnect-slaves=1
autoconnect-priority=20

[bond]
mode=802.3ad
miimon=100
xmit_hash_policy="layer3+4"

[ipv4]
method=auto
```

b. 次のコマンドを実行して、定義ファイルを Base64 でエンコードされた文字列に変換します。

```
$ base64 <directory_path>/en01.config
```

```
$ base64 <directory_path>/eno2.config
```

```
$ base64 <directory_path>/bond1.config
```

2. 環境変数を準備します。**<machine_role>** を、**worker** などのノードロールに置き換え、**<interface_name>** を追加の **br-ex** ブリッジ名に置き換えます。

```
$ export ROLE=<machine_role>
```

3. マシン設定マニフェストファイルで各インターフェイス定義を定義します。

bond1、eno1、en02 の定義を追加したマシン設定ファイルの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-${ROLE}-sec-bridge-cni
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-bond1.conf>
        path: /etc/NetworkManager/system-connections/bond1.nmconnection
        filesystem: root
        mode: 0600
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-eno1.conf>
        path: /etc/NetworkManager/system-connections/eno1.nmconnection
        filesystem: root
        mode: 0600
      - contents:
          source: data:;base64,<base-64-encoded-contents-for-eno2.conf>
        path: /etc/NetworkManager/system-connections/eno2.nmconnection
        filesystem: root
        mode: 0600
# ...
```

4. ターミナルで次のコマンドを入力して、ネットワークプラグインを設定するためのマシン設定マニフェストファイルを作成します。

```
$ oc create -f <machine_config_file_name>
```

5. OVN-Kubernetes ネットワークプラグインを使用して **extra_bridge** ファイルを作成し、ノード

上に Open vSwitch (OVS) ブリッジ **br-ex1** を作成します。このファイルは、ホストの **/etc/ovnk/extra_bridge** パスに保存します。ファイルには、ノードのプライマリー IP アドレスを保持する **br-ex** をサポートするデフォルトのインターフェイスではなく、追加のブリッジをサポートするインターフェイスの名前を記述する必要があります。

追加のインターフェイスを参照する extra_bridge ファイル /etc/ovnk/extra_bridge の設定例

```
bond1
```

6. クラスターで再起動されるノードで **br-ex1** をホストする既存の静的インターフェイスを定義したマシン設定マニフェストファイルを作成します。

br-ex1 をホストするインターフェイスとして bond1 を定義したマシン設定ファイルの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-worker-extra-bridge
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /etc/ovnk/extra_bridge
          mode: 0420
          overwrite: true
          contents:
            source: data:text/plain;charset=utf-8,bond1
          filesystem: root
```

7. 選択したノードにマシン設定を適用します。

```
$ oc create -f <machine_config_file_name>
```

8. オプション: **/var/lib/ovnk/iface_default_hint** リソースを作成するマシン設定ファイルを作成することにより、ノードの **br-ex** 選択ロジックをオーバーライドできます。

注記

このリソースには、**br-ex** がクラスター用に選択するインターフェイスの名前がリストされます。**br-ex** は、デフォルトで、ブート順序とマシンネットワーク内の IP アドレスサブネットに基づいて、ノードのプライマリーインターフェイスを選択します。マシンネットワークの設定によっては、ホストノードのデフォルトのインターフェイスまたはボンディングを引き続き **br-ex** が選択する必要があります。

a. ホストノードに、デフォルトのインターフェイスをオーバーライドするためのマシン設定ファイルを作成します。

重要

このマシン設定ファイルは、**br-ex** 選択ロジックを変更する目的でのみ作成します。このファイルを使用してクラスター内の既存のノードの IP アドレスを変更することはサポートされていません。

デフォルトのインターフェイスをオーバーライドするマシン設定ファイルの例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: ${worker}
  name: 12-worker-br-ex-override
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - path: /var/lib/ovnk/iface_default_hint
          mode: 0420
          overwrite: true
          contents:
            source: data:text/plain;charset=utf-8,bond0 (1)
          filesystem: root
```

(1) マシン設定ファイルをノードに適用する前に、ノードに **bond0** が存在することを確認します。

b. クラスター内のすべての新しいノードに設定を適用する前に、ホストノードを再起動して、**br-ex** が目的のインターフェイスを選択し、**br-ex1** で定義した新しいインターフェイスと競合しないことを検証します。

c. クラスター内のすべての新しいノードにマシン設定ファイルを適用します。

```
$ oc create -f <machine_config_file_name>
```

検証

1. クラスター内の **exgw-ip-addresses** ラベルを持つノードの IP アドレスを特定し、ノードがデフォルトのブリッジではなく追加のブリッジを使用していることを確認します。

```
$ oc get nodes -o json | grep --color exgw-ip-addresses
```

出力例

```
"k8s.ovn.org/l3-gateway-config":
  \"exgw-ip-address\":\"172.xx.xx.yy/24\",\"next-hops\":[\"xx.xx.xx.xx\"],
```

2. ホストノードのネットワークインターフェイス名を確認して、ターゲットノードに追加のブリッジが存在することを確認します。

```
$ oc debug node/<node_name> -- chroot /host sh -c "ip a | grep mtu | grep br-ex"
```

出力例

```
Starting pod/worker-1-debug ...
To use host binaries, run `chroot /host`
# ...
5: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
6: br-ex1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
```

3. オプション: **/var/lib/ovnk/iface_default_hint** を使用する場合は、**br-ex** の MAC アドレスが、選択したプライマリーインターフェイスの MAC アドレスと一致していることを確認します。

```
$ oc debug node/<node_name> -- chroot /host sh -c "ip a | grep -A1 -E 'br-ex|bond0'
```

br-ex のプライマリーインターフェイスが bond0 であることを示す出力例

```
Starting pod/worker-1-debug ...
To use host binaries, run `chroot /host`
# ...
sh-5.1# ip a | grep -A1 -E 'br-ex|bond0'
2: bond0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
ovs-system state UP group default qlen 1000
    link/ether fa:16:3e:47:99:98 brd ff:ff:ff:ff:ff:ff
--
5: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
    link/ether fa:16:3e:47:99:98 brd ff:ff:ff:ff:ff:ff
    inet 10.xx.xx.xx/21 brd 10.xx.xx.255 scope global dynamic noprefixroute br-ex
```

関連情報

- デフォルトネットワークに外部ゲートウェイを設定する

7.5.2. Open vSwitch の問題のトラブルシューティング

Open vSwitch(OVS) の問題をトラブルシューティングするためには、より多くの情報を含むようにログレベルを設定する必要があるかもしれません。

ノードのログレベルを一時的に変更した場合、次の例のようにノード上のマシン設定デーモンからログメッセージを受信することがあるので注意が必要です。

```
E0514 12:47:17.998892    2281 daemon.go:1350] content mismatch for file /etc/systemd/system/ovs-
vswitchd.service: [Unit]
```

不一致に関連するログメッセージを回避するには、トラブルシューティングが完了した後に、ログレベルの変更を元に戻してください。

7.5.2.1. Open vSwitch のログレベルの一時的な設定

短期間のトラブルシューティングのために、Open vSwitch(OVS) のログレベルを一時的に設定することができます。以下の手順では、ノードを再起動する必要はありません。また、ノードを再起動した場合、設定の変更は保持されません。

この手順を実行してログレベルを変更した後、**ovs-vswitchd.service** のコンテンツの不一致を示すログメッセージをマシン設定デーモンから受け取ることがあります。ログメッセージが表示されないようにするには、この手順を繰り返し、ログレベルを元の値に設定してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ノードのデバッグ Pod を起動します。

```
$ oc debug node/<node_name>
```

2. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストからのルートファイルシステムをマウントします。ルートディレクトリーを **/host** に変更すると、ホストファイルシステムからのバイナリーを実行できます。

```
# chroot /host
```

3. OVS モジュールの現在の syslog レベルを表示します。

```
# ovs-appctl vlog/list
```

次の出力例では、syslog のログレベルが **info** に設定されています。

出力例

```
                console    syslog    file
                -------    ------    ------
backtrace          OFF       INFO       INFO
bfd                OFF       INFO       INFO
bond               OFF       INFO       INFO
bridge             OFF       INFO       INFO
bundle             OFF       INFO       INFO
bundles            OFF       INFO       INFO
cfm                OFF       INFO       INFO
collectors         OFF       INFO       INFO
command_line       OFF       INFO       INFO
connmgr            OFF       INFO       INFO
conntrack          OFF       INFO       INFO
conntrack_tp       OFF       INFO       INFO
coverage           OFF       INFO       INFO
ct_dpif            OFF       INFO       INFO
daemon             OFF       INFO       INFO
daemon_unix        OFF       INFO       INFO
```

```
dns_resolve       OFF       INFO       INFO
dpdk              OFF       INFO       INFO
...
```

4. **/etc/systemd/system/ovs-vswitchd.service.d/10-ovs-vswitchd-restart.conf** ファイルでログレベルを指定します。

```
Restart=always
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /var/lib/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /etc/openvswitch'
ExecStartPre=-/bin/sh -c '/usr/bin/chown -R :$${OVS_USER_ID##*:} /run/openvswitch'
ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg
ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
```

前述の例では、ログレベルは **dbg** に設定されています。**syslog:<log_level>** を **off**、**emer**、**err**、**warn**、**info**、または **dbg** に設定することで、最後の 2 行を変更します。ログレベル **off** では、すべてのログメッセージが除外されます。

5. サービスを再起動します。

```
# systemctl daemon-reload
```

```
# systemctl restart ovs-vswitchd
```

7.5.2.2. Open vSwitch のログレベルの恒久的な設定

Open vSwitch(OVS) のログレベルを長期的に変更する場合は、ログレベルを恒久的に変更することができます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下の例のような **MachineConfig** オブジェクトで、**99-change-ovs-loglevel.yaml** のようなファイルを作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master (1)
  name: 99-change-ovs-loglevel
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
      - dropins:
```

```
    - contents: |
        [Service]
          ExecStartPost=-/usr/bin/ovs-appctl vlog/set syslog:dbg  2
          ExecReload=-/usr/bin/ovs-appctl vlog/set syslog:dbg
      name: 20-ovs-vswitchd-restart.conf
    name: ovs-vswitchd.service
```

1. この手順を実行してコントロールプレーンノードを設定した後、手順を繰り返し、ロールを **worker** に設定してワーカーノードを設定します。

2. **syslog:<log_level>** の値を設定します。ログレベルは **off**、**emer**、**err**、**warn**、**info**、または **dbg** です。値を **off** に設定すると、すべてのログメッセージが除外されます。

2. マシン設定を適用します。

```
$ oc apply -f 99-change-ovs-loglevel.yaml
```

関連情報

- Machine Config Operator について
- マシン設定プールのステータスの確認

7.5.2.3. Open vSwitch のログの表示

Open vSwitch(OVS) のログを表示するには、以下の手順で行います。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

- 以下のコマンドのいずれかを実行します。
  - クラスター外から **oc** コマンドを使用してログを表示する。

    ```
    $ oc adm node-logs <node_name> -u ovs-vswitchd
    ```

  - クラスター内のノードにログオンした後にログを表示する。

    ```
    # journalctl -b -f -u ovs-vswitchd.service
    ```

    ノードにログオンする 1 つの方法は、**oc debug node/<node_name>** コマンドを使用することです。

7.6. OPERATOR 関連の問題のトラブルシューティング

Operator は、OpenShift Container Platform アプリケーションをパッケージ化し、デプロイし、管理する方法です。Operator はソフトウェアベンダーのエンジニアリングチームの拡張機能のように動作

し、OpenShift Container Platform 環境を監視し、その最新状態に基づいてリアルタイムの意思決定を行います。Operator はアップグレードをシームレスに実行し、障害に自動的に対応するように設計されており、時間の節約のためにソフトウェアのバックアッププロセスを省略するなどのショートカットを実行することはありません。

OpenShift Container Platform 4.18 には、クラスターが適切に機能するために必要なデフォルトの Operator セットが含まれています。これらのデフォルト Operator は Cluster Version Operator (CVO) によって管理されます。

クラスター管理者は、OpenShift Container Platform Web コンソールまたは CLI を使用して OperatorHub からアプリケーション Operator をインストールできます。その後、Operator を 1 つまたは複数の namespace にサブスクライブし、クラスター上で開発者が使用できるようにできます。アプリケーション Operator は Operator Lifecycle Manager (OLM) によって管理されます。

Operator に問題が発生した場合には、Operator Subscription のステータスを確認します。クラスター全体で Operator Pod の正常性を確認し、診断用に Operator ログを収集します。

7.6.1. Operator サブスクリプションの状態のタイプ

サブスクリプションは状態に関する以下のタイプを報告します。

表7.2 サブスクリプションの状態のタイプ

状態	説明
CatalogSourcesUnhealthy	解決に使用される一部のまたはすべてのカタログソースは正常ではありません。
InstallPlanMissing	サブスクリプションのインストール計画がありません。
InstallPlanPending	サブスクリプションのインストール計画はインストールの保留中です。
InstallPlanFailed	サブスクリプションのインストール計画が失敗しました。
ResolutionFailed	サブスクリプションの依存関係の解決に失敗しました。

注記

デフォルトの OpenShift Container Platform Cluster Operator は Cluster Version Operator (CVO) によって管理され、これらの Operator には **Subscription** オブジェクトがありません。アプリケーション Operator は Operator Lifecycle Manager (OLM) によって管理され、それらには **Subscription** オブジェクトがあります。

関連情報

- カタログの正常性要件

7.6.2. CLI を使用した Operator サブスクリプションステータスの表示

CLI を使用して Operator サブスクリプションステータスを表示できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. Operator サブスクリプションをリスト表示します。

```
$ oc get subs -n <operator_namespace>
```

2. **oc describe** コマンドを使用して、**Subscription** リソースを検査します。

```
$ oc describe sub <subscription_name> -n <operator_namespace>
```

3. コマンド出力で、**Conditions** セクションで Operator サブスクリプションの状態タイプのステータスを確認します。以下の例では、利用可能なすべてのカタログソースが正常であるため、**CatalogSourcesUnhealthy** 状態タイプのステータスは **false** になります。

出力例

```
Name:         cluster-logging
Namespace:    openshift-logging
Labels:       operators.coreos.com/cluster-logging.openshift-logging=
Annotations:  <none>
API Version:  operators.coreos.com/v1alpha1
Kind:         Subscription
# ...
Conditions:
  Last Transition Time:  2019-07-29T13:42:57Z
  Message:               all available catalogsources are healthy
  Reason:                AllCatalogSourcesHealthy
  Status:                False
  Type:                  CatalogSourcesUnhealthy
# ...
```

注記

デフォルトの OpenShift Container Platform Cluster Operator は Cluster Version Operator (CVO) によって管理され、これらの Operator には **Subscription** オブジェクトがありません。アプリケーション Operator は Operator Lifecycle Manager (OLM) によって管理され、それらには **Subscription** オブジェクトがあります。

7.6.3. CLI を使用した Operator カタログソースのステータス表示

Operator カタログソースのステータスは、CLI を使用して確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. namespace のカタログソースをリスト表示します。たとえば、クラスター全体のカタログソースに使用されている **openshift-marketplace** namespace を確認することができます。

```
$ oc get catalogsources -n openshift-marketplace
```

出力例

```
NAME                  DISPLAY               TYPE   PUBLISHER   AGE
certified-operators   Certified Operators   grpc   Red Hat     55m
community-operators   Community Operators   grpc   Red Hat     55m
example-catalog       Example Catalog       grpc   Example Org 2m25s
redhat-marketplace    Red Hat Marketplace   grpc   Red Hat     55m
redhat-operators      Red Hat Operators     grpc   Red Hat     55m
```

2. カタログソースの詳細やステータスを確認するには、**oc describe** コマンドを使用します。

```
$ oc describe catalogsource example-catalog -n openshift-marketplace
```

出力例

```
Name:         example-catalog
Namespace:    openshift-marketplace
Labels:       <none>
Annotations:  operatorframework.io/managed-by: marketplace-operator
              target.workload.openshift.io/management: {"effect": "PreferredDuringScheduling"}
API Version:  operators.coreos.com/v1alpha1
Kind:         CatalogSource
# ...
Status:
  Connection State:
    Address:              example-catalog.openshift-marketplace.svc:50051
    Last Connect:         2021-09-09T17:07:35Z
    Last Observed State:  TRANSIENT_FAILURE
  Registry Service:
    Created At:         2021-09-09T17:05:45Z
    Port:               50051
    Protocol:           grpc
    Service Name:       example-catalog
    Service Namespace:  openshift-marketplace
# ...
```

前述の出力例では、最後に観測された状態が **TRANSIENT_FAILURE** となっています。この状態は、カタログソースの接続確立に問題があることを示しています。

3. カタログソースが作成された namespace の Pod をリストアップします。

```
$ oc get pods -n openshift-marketplace
```

出力例

```
NAME                                  READY   STATUS             RESTARTS   AGE
certified-operators-cv9nn             1/1     Running            0          36m
community-operators-6v8lp             1/1     Running            0          36m
```

```
marketplace-operator-86bfc75f9b-jkgbc   1/1     Running           0          42m
example-catalog-bwt8z                   0/1     ImagePullBackOff  0          3m55s
redhat-marketplace-57p8c                1/1     Running           0          36m
redhat-operators-smxx8                  1/1     Running           0          36m
```

namespace にカタログソースを作成すると、その namespace にカタログソース用の Pod が作成されます。前述の出力例では、**example-catalog-bwt8z** Pod のステータスが **ImagePullBackOff** になっています。このステータスは、カタログソースのインデックスイメージのプルに問題があることを示しています。

4. **oc describe** コマンドを使用して、より詳細な情報を得るために Pod を検査します。

```
$ oc describe pod example-catalog-bwt8z -n openshift-marketplace
```

出力例

```
Name:         example-catalog-bwt8z
Namespace:    openshift-marketplace
Priority:     0
Node:         ci-ln-jyryyg2-f76d1-ggdbq-worker-b-vsxjd/10.0.128.2
...
Events:
  Type     Reason          Age                From               Message
  ----     ------          ----               ----               -------
  Normal   Scheduled       48s                default-scheduler  Successfully assigned openshift-marketplace/example-catalog-bwt8z to ci-ln-jyryyf2-f76d1-fgdbq-worker-b-vsxjd
  Normal   AddedInterface  47s                multus             Add eth0 [10.131.0.40/23] from openshift-sdn
  Normal   BackOff         20s (x2 over 46s)  kubelet            Back-off pulling image "quay.io/example-org/example-catalog:v1"
  Warning  Failed          20s (x2 over 46s)  kubelet            Error: ImagePullBackOff
  Normal   Pulling         8s (x3 over 47s)   kubelet            Pulling image "quay.io/example-org/example-catalog:v1"
  Warning  Failed          8s (x3 over 47s)   kubelet            Failed to pull image "quay.io/example-org/example-catalog:v1": rpc error: code = Unknown desc = reading manifest v1 in quay.io/example-org/example-catalog: unauthorized: access to the requested resource is not authorized
  Warning  Failed          8s (x3 over 47s)   kubelet            Error: ErrImagePull
```

前述の出力例では、エラーメッセージは、カタログソースのインデックスイメージが承認問題のために正常にプルできないことを示しています。例えば、インデックスイメージがログイン認証情報を必要とするレジストリーに保存されている場合があります。

関連情報

- Operator Lifecycle Manager の概念およびリソース → カタログソース
- gRPC ドキュメント:接続性の状態
- プライベートレジストリーからの Operator のイメージへのアクセス

7.6.4. Operator Pod ステータスのクエリー

クラスター内の Operator Pod およびそれらのステータスをリスト表示できます。詳細な Operator Pod の要約を収集することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. クラスターで実行されている Operators をリスト表示します。出力には、Operator バージョン、可用性、およびアップタイムの情報が含まれます。

```
$ oc get clusteroperators
```

2. Operator の namespace で実行されている Operator Pod をリスト表示し、Pod のステータス、再起動、および経過時間をリスト表示します。

```
$ oc get pod -n <operator_namespace>
```

3. 詳細な Operator Pod の要約を出力します。

```
$ oc describe pod <operator_pod_name> -n <operator_namespace>
```

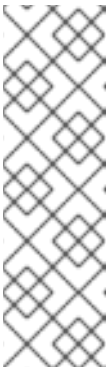
4. Operator の問題がノード固有の問題である場合、クエリーを実行してそのノードの Operator コンテナーのステータスを取得します。

   a. ノードのデバッグ Pod を起動します。

```
$ oc debug node/my-node
```

   b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```



注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

   c. 状態および関連付けられた Pod ID を含む、ノードのコンテナーの詳細をリスト表示します。

```
# crictl ps
```

d. ノード上の特定の Operator コンテナーに関する情報をリスト表示します。以下の例では、**network-operator** コンテナーに関する情報をリスト表示します。

```
# crictl ps --name network-operator
```

e. デバッグシェルを終了します。

7.6.5. Operator ログの収集

Operator の問題が発生した場合、Operator Pod ログから詳細な診断情報を収集できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。
- コントロールプレーンまたはコントロールプレーンマシンの完全修飾ドメイン名がある。

手順

1. Operator の namespace で実行されている Operator Pod、Pod のステータス、再起動、および経過時間をリスト表示します。

```
$ oc get pods -n <operator_namespace>
```

2. Operator Pod のログを確認します。

```
$ oc logs pod/<pod_name> -n <operator_namespace>
```

Operator Pod に複数のコンテナーがある場合、前述のコマンドにより各コンテナーの名前が含まれるエラーが生成されます。個別のコンテナーに対して、ログのクエリーを実行します。

```
$ oc logs pod/<operator_pod_name> -c <container_name> -n <operator_namespace>
```

3. API が機能しない場合には、代わりに SSH を使用して各コントロールプレーンノードで Operator Pod およびコンテナーログを確認します。**<master-node>.<cluster_name>.<base_domain>** を適切な値に置き換えます。

a. 各コントロールプレーンノードの Pod をリスト表示します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods
```

b. Operator Pod で **Ready** ステータスが表示されない場合は、Pod のステータスを詳細に検査します。**<operator_pod_id>** を直前のコマンドの出力にリスト表示されている Operator Pod の ID に置き換えます。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
<operator_pod_id>
```

c. Operator Pod に関連するコンテナーをリスト表示します。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps --pod=
<operator_pod_id>
```

d. **Ready** ステータスが Operator コンテナーに表示されない場合は、コンテナーのステータスを詳細に検査します。**<container_id>** を前述のコマンドの出力に一覧表示されているコンテナー ID に置き換えます。

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

e. **Ready** ステータスが表示されない Operator コンテナーのログを確認します。**<container_id>** を前述のコマンドの出力に一覧表示されているコンテナー ID に置き換えます。

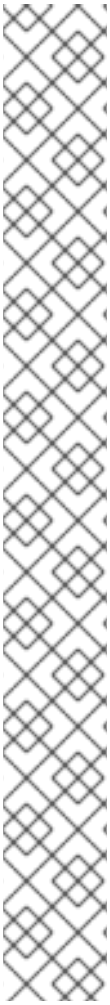
```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。SSH 経由で診断データの収集を試行する前に、**oc adm must gather** およびその他の **oc** コマンドを実行して収集されるデータが十分であるかどうかを確認してください。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

7.6.6. Machine Config Operator の自動再起動の無効化

設定変更が Machine Config Operator (MCO) によって行われる場合、Red Hat Enterprise Linux CoreOS (RHCOS) を再起動して変更を反映する必要があります。設定の変更が自動または手動であるかどうかにかかわらず、RHCOS ノードは、一時停止されない限り自動的に再起動します。



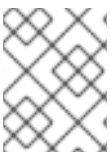
注記

- MCO が以下の変更のいずれかを検出すると、ノードの drain (Pod の退避) の実行または再起動を行わずに更新を適用します。
  - マシン設定の **spec.config.passwd.users.sshAuthorizedKeys** パラメーターの SSH キーの変更。
  - **openshift-config** namespace でのグローバルプルシークレットまたはプルシークレットへの変更。
  - Kubernetes API Server Operator による **/etc/kubernetes/kubelet-ca.crt** 認証局 (CA) の自動ローテーション。
- MCO は、**/etc/containers/registries.conf** ファイルへの変更 (**ImageDigestMirrorSet**、**ImageTagMirrorSet**、または **ImageContentSourcePolicy** オブジェクトの編集など) を検出すると、対応するノードの drain (Pod の退避) を実行し、変更を適用して、ノードをスケジューリング対象に戻します。次の変更ではノードの drain (Pod の退避) の実行は発生しません。
  - **pull-from-mirror = "digest-only"** パラメーターがミラーごとに設定されたレジストリーの追加。
  - **pull-from-mirror = "digest-only"** パラメーターがレジストリーに設定されたミラーの追加。
  - **unqualified-search-registries** へのアイテムの追加。

不要な中断を防ぐために、マシン設定プール (MCP) を変更して、Operator がマシン設定を変更した後に自動再起動を防ぐことができます。

7.6.6.1. コンソールの使用による Machine Config Operator の自動再起動の無効化

Machine Config Operator (MCO) の変更から不要な中断を防ぐには、OpenShift Container Platform Web コンソールを使用してマシン設定プール (MCP) を変更し、MCO がそのプール内のノードに変更を加えられないようにすることができます。これにより、通常 MCO 更新プロセスの一部として実行される再起動ができなくなります。



注記

Machine Config Operator の自動再起動の無効化 の 2 つ目の **NOTE** を参照してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

自動 MCO 更新の再起動の一時停止または一時停止を解除するには、以下を実行します。

- 自動再起動プロセスを一時停止します。
  1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。

2. **Compute → MachineConfigPools** をクリックします。
3. **MachineConfigPools** ページで、再起動を一時停止するノードに合わせて **master** または **worker** のいずれかをクリックします。
4. **master** または **worker** ページで、**YAML** をクリックします。
5. YAML で、**spec.paused** フィールドを **true** に更新します。

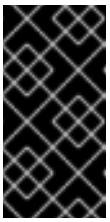
   MachineConfigPool オブジェクトのサンプル

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
# ...
spec:
# ...
  paused: true (1)
# ...
```

   (1) **spec.paused** フィールドを **true** に更新し、再起動を一時停止します。

6. MCP が一時停止されていることを確認するには、**MachineConfigPools** ページに戻ります。
   MachineConfigPools ページの **Paused** 列では、変更した MCP に対して **True** が報告されます。

   MCP が一時停止中に保留中の変更がある場合は、**Updated** 列は **False** であり、**Updating** は **False** になります。**Updated** が **True** であり、**Updating** が **False** の場合、保留中の変更はありません。



重要

保留中の変更がある場合 (**Updated** および **Updating** 列の両方が **False** の場合)、できるだけ早期に再起動のメンテナンス期間をスケジュールすることが推奨されます。自動再起動プロセスの一時停止を解除して、最後に再起動してからキューに追加された変更を適用するには、以下の手順に従います。

- 自動再起動プロセスの一時停止を解除するには、以下を実行します。

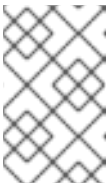
1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。
2. **Compute → MachineConfigPools** をクリックします。
3. **MachineConfigPools** ページで、再起動を一時停止するノードに合わせて **master** または **worker** のいずれかをクリックします。
4. **master** または **worker** ページで、**YAML** をクリックします。
5. YAML で、**spec.paused** フィールドを **false** に更新します。

   MachineConfigPool オブジェクトのサンプル

```
apiVersion: machineconfiguration.openshift.io/v1
```

```
kind: MachineConfigPool
# ...
spec:
# ...
  paused: false (1)
# ...
```

(1) **spec.paused** フィールドを **false** に更新し、再起動を許可します。



注記

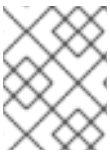
MCP の一時停止を解除すると、MCO は一時停止したすべての変更を適用し、必要に応じて Red Hat Enterprise Linux CoreOS (RHCOS) を再起動します。

6. MCP が一時停止されていることを確認するには、**MachineConfigPools** ページに戻ります。
MachineConfigPools ページの **Paused** 列では、変更した MCP に対して **False** が報告されます。

   MCP が保留中の変更を適用する場合、**Updated** 列は **False** になり、**Updating** 列は **True** になります。**Updated** が **True** であり、**Updating** が **False** の場合、追加の変更は加えられません。

7.6.6.2. CLI の使用による Machine Config Operator の自動再起動の無効化

Machine Config Operator (MCO) によって加えられる変更から生じる不要な中断を防ぐには、OpenShift CLI (oc) を使用してマシン設定プール (MCP) を変更し、MCO がそのプール内のノードに変更を加えられないようにすることができます。これにより、通常 MCO 更新プロセスの一部として実行される再起動ができなくなります。



注記

Machine Config Operator の自動再起動の無効化 の 2 つ目の **NOTE** を参照してください。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

自動 MCO 更新の再起動の一時停止または一時停止を解除するには、以下を実行します。

- 自動再起動プロセスを一時停止します。
  1. **MachineConfigPool** カスタムリソースを、**spec.paused** フィールドを **true** に設定するように更新します。

     コントロールプレーン (マスター) ノード

     ```
     $ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/master
     ```

ワーカーノード

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/worker
```

2. MCP が一時停止されていることを確認します。

コントロールプレーン (マスター) ノード

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

ワーカーノード

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

出力例

```
true
```

spec.paused フィールドは **true** であり、MCP は一時停止されます。

3. MCP に保留中の変更があるかどうかを判別します。

```
# oc get machineconfigpool
```

出力例

```
NAME     CONFIG                                             UPDATED   UPDATING
master   rendered-master-33cf0a1254318755d7b48002c597bf91   True      False
worker   rendered-worker-e405a5bdb0db1295acea08bcca33fa60   False     False
```

UPDATED 列が **False** であり、**UPDATING** が **False** の場合は、保留中の変更があります。**UPDATED** が **True** であり、**UPDATING** が **False** の場合、保留中の変更はありません。この例では、ワーカーノードに保留中の変更があります。コントロールプレーンノードには保留中の変更がありません。

重要

保留中の変更がある場合 (**Updated** および **Updating** 列の両方が **False** の場合)、できるだけ早期に再起動のメンテナンス期間をスケジュールすることが推奨されます。自動再起動プロセスの一時停止を解除して、最後に再起動してからキューに追加された変更を適用するには、以下の手順に従います。

- 自動再起動プロセスの一時停止を解除するには、以下を実行します。

1. **MachineConfigPool** カスタムリソースを、**spec.paused** フィールドを **false** に設定するように更新します。

コントロールプレーン (マスター) ノード

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/master
```

ワーカーノード

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/worker
```

注記

MCP の一時停止を解除すると、MCO は一時停止したすべての変更を適用し、必要に応じて Red Hat Enterprise Linux CoreOS (RHCOS) を再起動します。

2. MCP の一時停止が解除されていることを確認します。

コントロールプレーン (マスター) ノード

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

ワーカーノード

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

出力例

```
false
```

spec.paused フィールドは **false** であり、マシン設定プールの一時停止は解除されます。

3. MCP に保留中の変更があるかどうかを判別します。

```
$ oc get machineconfigpool
```

出力例

```
NAME     CONFIG                                UPDATED  UPDATING
master   rendered-master-546383f80705bd5aeaba93   True     False
worker   rendered-worker-b4c51bb33ccaae6fc4a6a5   False    True
```

MCP が保留中の変更を適用する場合、**UPDATED** 列は **False** で、**UPDATING** 列は **True** になります。**UPDATED** が **True** であり、**UPDATING** が **False** の場合、追加の変更は加えられません。直前の例では、MCO はワーカーノードを更新しています。

7.6.7. 障害のあるサブスクリプションの更新

Operator Lifecycle Manager (OLM) で、ネットワークでアクセスできないイメージを参照する Operator をサブスクライブする場合、以下のエラーを出して失敗した **openshift-marketplace** namespace でジョブを見つけることができます。

出力例

```
ImagePullBackOff for
Back-off pulling image "example.com/openshift4/ose-elasticsearch-operator-
bundle@sha256:6d2587129c846ec28d384540322b40b05833e7e00b25cca584e004af9a1d292e"
```

出力例

```
rpc error: code = Unknown desc = error pinging docker registry example.com: Get
"https://example.com/v2/": dial tcp: lookup example.com on 10.0.0.1:53: no such host
```

その結果、サブスクリプションはこの障害のある状態のままとなり、Operator はインストールまたはアップグレードを実行できません。

サブスクリプション、クラスターサービスバージョン (CSV) その他の関連オブジェクトを削除して、障害のあるサブスクリプションを更新できます。サブスクリプションを再作成した後に、OLM は Operator の正しいバージョンを再インストールします。

前提条件

- アクセス不可能なバンドルイメージをプルできない障害のあるサブスクリプションがある。
- 正しいバンドルイメージにアクセスできることを確認している。

手順

1. Operator がインストールされている namespace から **Subscription** および **ClusterServiceVersion** オブジェクトの名前を取得します。

   ```
   $ oc get sub,csv -n <namespace>
   ```

   出力例

   ```
   NAME                                         PACKAGE                  SOURCE             CHANNEL
   subscription.operators.coreos.com/elasticsearch-operator   elasticsearch-operator   redhat-operators   5.0

   NAME                                                  DISPLAY                  VERSION    REPLACES   PHASE
   clusterserviceversion.operators.coreos.com/elasticsearch-operator.5.0.0-65   OpenShift Elasticsearch Operator   5.0.0-65              Succeeded
   ```

2. サブスクリプションを削除します。

   ```
   $ oc delete subscription <subscription_name> -n <namespace>
   ```

3. クラスターサービスバージョンを削除します。

   ```
   $ oc delete csv <csv_name> -n <namespace>
   ```

4. **openshift-marketplace** namespace の失敗したジョブおよび関連する config map の名前を取得します。

   ```
   $ oc get job,configmap -n openshift-marketplace
   ```

   出力例

   ```
   NAME                                                  COMPLETIONS   DURATION   AGE
   ```

```
job.batch/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   1/1
26s        9m30s

NAME                                                                DATA   AGE
configmap/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   3
9m30s
```

5. ジョブを削除します。

```
$ oc delete job <job_name> -n openshift-marketplace
```

これにより、アクセスできないイメージのプルを試行する Pod は再作成されなくなります。

6. 設定マップを削除します。

```
$ oc delete configmap <configmap_name> -n openshift-marketplace
```

7. Web コンソールの OperatorHub を使用した Operator の再インストール

検証

- Operator が正常に再インストールされていることを確認します。

```
$ oc get sub,csv,installplan -n <namespace>
```

7.6.8. アンインストール失敗後の Operator の再インストール

同じ Operator の再インストールを試行する前に、Operator を正常かつ完全にアンインストールする必要があります。Operator を適切かつ完全にアンインストールできていない場合、プロジェクトや namespace などのリソースが "Terminating" ステータスでスタックし、"error resolving resource" メッセージが表示されます。以下に例を示します。

Project リソースの説明例

```
...
    message: 'Failed to delete all resource types, 1 remaining: Internal error occurred:
      error resolving resource'
...
```

これらのタイプの問題は、Operator の正常な再インストールを妨げる可能性があります。

警告

namespace を強制的に削除しても、"Terminating" 状態の問題が解決される可能性は低く、クラスターの動作が不安定または予測不能になる可能性があるため、namespace の削除を妨げている可能性のある関連リソースの特定に注力することが推奨されます。詳細は、Red Hat Knowledgebase Solution #4165791 を参照し、特に注意と警告に注目してください。

次の手順では、以前インストールされた Operator からの既存カスタムリソース定義 (CRD) が原因で関連する namespace が正常に削除されないために Operator を再インストールできない場合のトラブルシューティングを示します。

手順

1. "Terminating" 状態のままになっている Operator に関連する namespace があるかどうかを確認します。

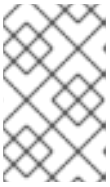
```
$ oc get namespaces
```

出力例

```
operator-ns-1                                Terminating
```

2. アンインストールの失敗後も Operator に関連する CRD があるか確認します。

```
$ oc get crds
```



注記

CRD はグローバルクラスター定義です。CRD に関連する実際のカスタムリソース (CR) インスタンスは、他の namespace にあるか、グローバルクラスターインスタンスである可能性があります。

3. Operator によって提供または管理されている CRD があり、その CRD をアンインストール後に削除する必要がある場合は、CRD を削除します。

```
$ oc delete crd <crd_name>
```

4. アンインストールした後も Operator に関連する CR インスタンスが残っているか確認し、残っている場合は CR を削除します。

   a. アンインストール後は、検索する CR のタイプの判断が困難になり、Operator が管理する CRD を把握している必要がある場合もあります。たとえば、**EtcdCluster** CRD を提供する etcd Operator のアンインストールをトラブルシューティングする場合、namespace で残りの **EtcdCluster** CR を検索できます。

```
$ oc get EtcdCluster -n <namespace_name>
```

   もしくは、すべての namespace で検索できます。

```
$ oc get EtcdCluster --all-namespaces
```

   b. 削除する必要のある CR が残っている場合は、インスタンスを削除します。

```
$ oc delete <cr_name> <cr_instance_name> -n <namespace_name>
```

5. namespace の削除が正常に解決されたことを確認します。

```
$ oc get namespace <namespace_name>
```

重要

namespace やその他の Operator リソースが正常にアンインストールされていない場合は、Red Hat サポートにお問い合わせください。

6. Web コンソールの OperatorHub を使用した Operator の再インストール

検証

- Operator が正常に再インストールされていることを確認します。

```
$ oc get sub,csv,installplan -n <namespace>
```

関連情報

- クラスターからの Operator の削除
- Operator のクラスターへの追加

7.7. POD の問題の調査

OpenShift Container Platform は、ホスト上に共にデプロイされる 1 つ以上のコンテナーである Pod の Kubernetes の概念を活用しています。Pod は、OpenShift Container Platform 4.18 で定義、デプロイ、管理できる最小のコンピュート単位です。

Pod が定義されると、そのコンテナーが終了するか、Pod が削除されるまで、Pod はノードで実行されるように割り当てられます。ポリシーと終了コードに応じて、Pod は終了後に削除されるか、ログにアクセスできるように保持されます。

Pod の問題が発生した場合には、まず Pod のステータスをチェックします。Pod の明示的な障害が発生した場合には、Pod のエラー状態をチェックして、特定のイメージ、コンテナー、または Pod ネットワークの問題を特定してください。エラー状態に基づく診断データの収集を行います。Pod イベントメッセージおよび Pod およびコンテナーのログ情報を確認します。コマンドライン上で実行中の Pod にアクセスするか、問題のある Pod のデプロイメント設定に基づいて root アクセスでデバッグ Pod を起動して問題を動的に診断します。

7.7.1. Pod のエラー状態について

Pod に障害が発生すると、明示的なエラー状態が返されます。このエラー状態は、**oc get pods** の出力の **status** フィールドで確認できます。Pod のエラー状態は、イメージ、コンテナー、およびコンテナーネットワークに関連する障害に関する状態を示します。

以下の表は、Pod のエラー状態のリストをそれらの説明を記載しています。

表7.3 Pod のエラー状態

Pod のエラー状態	説明
ErrImagePull	一般的なイメージの取得エラー。
ErrImagePullBackOff	イメージの取得に失敗し、取り消されました。

Pod のエラー状態	説明
ErrInvalidImageName	指定されたイメージ名は無効です。
ErrImageInspect	イメージの検査に失敗しました。
ErrImageNeverPull	**PullPolicy** は **NeverPullImage** に設定され、ターゲットイメージはホスト上でローカルに見つかりません。
ErrRegistryUnavailable	レジストリーからイメージの取得を試みる際に、HTTP エラーが発生しました。
ErrContainerNotFound	指定されたコンテナーが宣言された Pod 内にないか、kubelet によって管理されていません。
ErrRunInitContainer	コンテナーの初期化に失敗しました。
ErrRunContainer	Pod のコンテナーのいずれも正常に起動しませんでした。
ErrKillContainer	Pod のコンテナーのいずれも正常に強制終了されませんでした。
ErrCrashLoopBackOff	コンテナーが終了しました。kubelet は再起動を試行しません。
ErrVerifyNonRoot	コンテナーまたはイメージが root 権限で実行を試行しました。
ErrCreatePodSandbox	Pod サンドボックスの作成が成功しませんでした。
ErrConfigPodSandbox	Pod サンドボックス設定を取得できませんでした。
ErrKillPodSandbox	Pod サンドボックスは正常に停止しませんでした。
ErrSetupNetwork	ネットワークの初期化に失敗しました。
ErrTeardownNetwork	ネットワークの終了に失敗しました。

7.7.2. Pod ステータスの確認

Pod のステータスおよびエラー状態をクエリーできます。Pod に関連するデプロイメント設定をクエリーし、ベースイメージの可用性を確認することもできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- **skopeo** がインストールされている。

手順

1. プロジェクトに切り替えます。

```
$ oc project <project_name>
```

2. namespace 内で実行されている Pod、Pod のステータス、エラーの状態、再起動、および経過時間をリスト表示します。

```
$ oc get pods
```

3. namespace がデプロイメント設定で管理されているかどうかを判別します。

```
$ oc status
```

namespace がデプロイメント設定で管理される場合、出力には、デプロイメント設定名とベースイメージの参照が含まれます。

4. 前述のコマンドの出力で参照されているベースイメージを検査します。

```
$ skopeo inspect docker://<image_reference>
```

5. ベースイメージの参照が正しくない場合は、デプロイメント設定の参照を更新します。

```
$ oc edit deployment/my-deployment
```

6. デプロイメント設定が終了時に変更されると、設定が自動的に再デプロイされます。デプロイメントの進行中に Pod ステータスを確認し、問題が解決されているかどうかを判別します。

```
$ oc get pods -w
```

7. Pod の失敗に関連する診断情報は、namespace 内でイベントを確認します。

```
$ oc get events
```

7.7.3. Pod およびコンテナーログの検査

明示的な Pod の失敗に関連する警告およびエラーメッセージの有無について Pod およびコンテナーログを検査できます。ポリシーおよび終了コードによっては、Pod およびコンテナーログは Pod の終了後も利用可能のままになります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 特定 Pod のログのクエリーを実行します。

```
$ oc logs <pod_name>
```

2. クエリーを実行して、Pod 内の特定コンテナーのログを取得します。

```
$ oc logs <pod_name> -c <container_name>
```

前述の **oc logs** コマンドを使用して取得されるログは、Pod またはコンテナー内の標準出力 (stdout) に送信されるメッセージで構成されます。

3. Pod 内の **/var/log/** に含まれるログを検査します。
   a. Pod 内の **/var/log** に含まれるファイルおよびサブディレクトリーをリスト表示します。

```
$ oc exec <pod_name>  -- ls -alh /var/log
```

出力例

```
total 124K
drwxr-xr-x. 1 root root   33 Aug 11 11:23 .
drwxr-xr-x. 1 root root   28 Sep  6  2022 ..
-rw-rw----. 1 root utmp    0 Jul 10 10:31 btmp
-rw-r--r--. 1 root root  33K Jul 17 10:07 dnf.librepo.log
-rw-r--r--. 1 root root  69K Jul 17 10:07 dnf.log
-rw-r--r--. 1 root root 8.8K Jul 17 10:07 dnf.rpm.log
-rw-r--r--. 1 root root  480 Jul 17 10:07 hawkey.log
-rw-rw-r--. 1 root utmp    0 Jul 10 10:31 lastlog
drwx------. 2 root root   23 Aug 11 11:14 openshift-apiserver
drwx------. 2 root root    6 Jul 10 10:31 private
drwxr-xr-x. 1 root root   22 Mar  9 08:05 rhsm
-rw-rw-r--. 1 root utmp    0 Jul 10 10:31 wtmp
```

   b. Pod 内の **/var/log** に含まれる特定のログファイルのクエリーを実行します。

```
$ oc exec <pod_name> cat /var/log/<path_to_log>
```

出力例

```
2023-07-10T10:29:38+0000 INFO --- logging initialized ---
2023-07-10T10:29:38+0000 DDEBUG timer: config: 13 ms
2023-07-10T10:29:38+0000 DEBUG Loaded plugins: builddep, changelog, config-
manager, copr, debug, debuginfo-install, download, generate_completion_cache, groups-
manager, needs-restarting, playground, product-id, repoclosure, repodiff, repograph,
repomanage, reposync, subscription-manager, uploadprofile
2023-07-10T10:29:38+0000 INFO Updating Subscription Management repositories.
2023-07-10T10:29:38+0000 INFO Unable to read consumer identity
2023-07-10T10:29:38+0000 INFO Subscription Manager is operating in container mode.
2023-07-10T10:29:38+0000 INFO
```

c. 特定のコンテナー内の **/var/log** に含まれるログファイルおよびサブディレクトリーをリスト表示します。

```
$ oc exec <pod_name> -c <container_name> ls /var/log
```

d. 特定のコンテナー内の **/var/log** に含まれる特定のログファイルのクエリーを実行します。

```
$ oc exec <pod_name> -c <container_name> cat /var/log/<path_to_log>
```

7.7.4. 実行中の Pod へのアクセス

Pod 内でシェルを開くか、ポート転送によりネットワークアクセスを取得して、実行中の Pod を動的に確認することができます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. アクセスする Pod が含まれるプロジェクトに切り替えます。これは、**oc rsh** コマンドが **-n** namespace オプションを受け入れないために必要です。

```
$ oc project <namespace>
```

2. リモートシェルを Pod で起動します。

```
$ oc rsh <pod_name> 1
```

1 Pod に複数のコンテナーがある場合、**oc rsh** は **-c <container_name>** が指定されていない限り最初のコンテナーにデフォルト設定されます。

3. Pod 内の特定のコンテナーでリモートシェルを起動します。

```
$ oc rsh -c <container_name> pod/<pod_name>
```

4. Pod のポートへのポート転送セッションを作成します。

```
$ oc port-forward <pod_name> <host_port>:<pod_port>
```
(1)

(1) ポート転送セッションをキャンセルするには、**Ctrl+C** を入力します。

7.7.5. root アクセスでのデバッグ Pod の起動

問題のある Pod のデプロイメントまたはデプロイメント設定に基づいて、root アクセスでデバッグ Pod を起動できます。通常、Pod ユーザーは root 以外の権限で実行しますが、問題を調査するために一時的な root 権限で Pod のトラブルシューティングを実行することは役に立ちます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. デプロイメントに基づいて、root アクセスでデバッグ Pod を起動します。
   a. プロジェクトのデプロイメント名を取得します。

```
$ oc get deployment -n <project_name>
```

   b. デプロイメントに基づいて、root 権限でデバッグ Pod を起動します。

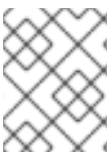
```
$ oc debug deployment/my-deployment --as-root -n <project_name>
```

2. デプロイメント設定に基づいて、root アクセスでデバッグ Pod を起動します。
   a. プロジェクトのデプロイメント設定名を取得します。

```
$ oc get deploymentconfigs -n <project_name>
```

   b. デプロイメント設定に基づいて、root 権限でデバッグ Pod を起動します。

```
$ oc debug deploymentconfig/my-deployment-configuration --as-root -n <project_name>
```



注記

インタラクティブなシェルを実行する代わりに、**-- <command>** を前述の **oc debug** コマンドに追加し、デバッグ Pod 内で個々のコマンドを実行することができます。

7.7.6. Pod およびコンテナーへの/からのファイルのコピー

Pod に/からファイルをコピーして、設定変更をテストしたり、診断情報を収集したりできます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. ファイルを Pod にコピーします。

```
$ oc cp <local_path> <pod_name>:/<path> -c <container_name>  1
```

1 **-c** オプションが指定されていない場合、Pod の最初のコンテナーが選択されます。

2. Pod からファイルをコピーします。

```
$ oc cp <pod_name>:/<path>  -c <container_name> <local_path>  1
```

1 **-c** オプションが指定されていない場合、Pod の最初のコンテナーが選択されます。

注記

oc cp が機能するには、**tar** バイナリーがコンテナー内で利用可能である必要があります。

7.8. SOURCE-TO-IMAGE (S2I) プロセスのトラブルシューティング

7.8.1. Source-to-Image (S2I) のトラブルシューティングのストラテジー

Source-to-Image (S2I) を使用して、再現可能な Docker 形式のコンテナーイメージをビルドします。アプリケーションソースコードをコンテナーイメージに挿入し、新規イメージをアセンブルして実行可能なイメージを作成できます。新規イメージには、ベースイメージ (ビルダー) およびビルドされたソースが組み込まれています。

手順

1. S2I プロセスで障害が発生する場所を特定するには、以下の各 S2I ステージに関連する Pod の状態を確認できます。

   a. **ビルド設定の段階** で、ビルド Pod はベースイメージおよびアプリケーションのソースコードからアプリケーションコンテナーイメージを作成するために使用されます。

   b. **デプロイメント設定の段階** で、デプロイメント Pod はビルド設定段階でビルドされたアプリケーションコンテナーイメージからアプリケーション Pod をデプロイするために使用されます。デプロイメント Pod は、サービスやルートなどの他のリソースもデプロイします。デプロイメント設定は、ビルド設定が成功すると開始されます。

   c. **デプロイメント Pod のアプリケーション Pod の起動後** に、アプリケーションの障害が実行中のアプリケーション Pod 内で発生する可能性があります。たとえば、アプリケーション Pod が **Running** 状態であっても、アプリケーションは予想通りに動作しない可能性があり

ます。このシナリオでは、実行中のアプリケーション Pod にアクセスして、Pod 内のアプリケーションの障害を調査できます。

2. S2I の問題のトラブルシューティングを行う際には、以下のストラテジーに従います。

   a. ビルド、デプロイメント、およびアプリケーション Pod ステータスを監視します。

   b. 問題が発生した S2I プロセスのステージを決定します。

   c. 失敗したステージに対応するログを確認します。

7.8.2. Source-to-Image 診断データの収集

S2I ツールは、ビルド Pod とデプロイメント Pod を順番に実行します。デプロイメント Pod は、ビルドステージで作成されたアプリケーションコンテナーイメージに基づいてアプリケーション Pod をデプロイします。S2I プロセスで障害が発生する場所を判別するために、ビルド、デプロイメント、およびアプリケーション Pod のステータスを監視します。次に、これに応じて診断データを収集します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- API サービスが機能している。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. S2I プロセス全体での Pod のステータスを確認し、障害が発生するステージを判別します。

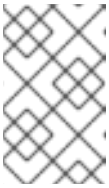
```
$ oc get pods -w 1
```

1 **-w** を使用して、**Ctrl+C** を使用してコマンドを終了するまで Pod で変更の有無を監視します。

2. 障害のある Pod のログでエラーの有無を確認します。

   - **ビルド Pod が失敗する場合**、ビルド Pod のログを確認します。

```
$ oc logs -f pod/<application_name>-<build_number>-build
```

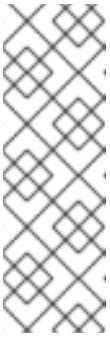


注記

または、**oc logs -f bc/<application_name>** を使用して、ビルド設定のログを確認できます。ビルド設定のログには、ビルド Pod からのログが含まれます。

   - **デプロイメント Pod が失敗する場合**、デプロイメント Pod のログを確認します。

```
$ oc logs -f pod/<application_name>-<build_number>-deploy
```



注記

または、**oc logs -f dc/<application_name>** を使用して、デプロイメント設定のログを確認できます。これにより、デプロイメント Pod が正常に実行されるまで、デプロイメント Pod からログが出力されます。デプロイメント Pod の完了後に実行すると、コマンドはアプリケーション Pod からログを出力します。デプロイメント Pod の完了後も、**oc logs -f pod/<application_name>-<build_number>-deploy** を実行してログにアクセスできます。

- **アプリケーション Pod が失敗した場合や、アプリケーションが実行中のアプリケーション Pod 内で予想通りに動作しない場合**、アプリケーション Pod のログを確認します。

```
$ oc logs -f pod/<application_name>-<build_number>-<random_string>
```

7.8.3. アプリケーションの障害を調査するためのアプリケーション診断データの収集

アプリケーションの障害は実行中のアプリケーション Pod 内で発生する可能性があります。このような状態では、以下のストラテジーを使用して診断情報を取得できます。

- アプリケーション Pod に関連するイベントを確認します。
- アプリケーション Pod からログを確認します。これには、OpenShift Logging フレームワークによって収集されないアプリケーション固有のログファイルが含まれます。
- アプリケーション機能を対話的にテストし、アプリケーションコンテナーで診断ツールを実行します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 特定のアプリケーション Pod に関連するイベントをリスト表示します。以下の例では、**my-app-1-akdlg** という名前のアプリケーション Pod のイベントを取得します。

```
$ oc describe pod/my-app-1-akdlg
```

2. アプリケーション Pod からのログを確認します。

```
$ oc logs -f pod/my-app-1-akdlg
```

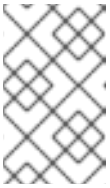
3. 実行中のアプリケーション Pod 内で特定のログを照会します。標準出力 (stdout) に送信されるログは OpenShift Logging フレームワークによって収集され、これは前述のコマンドの出力に含まれます。以下のクエリーは、標準出力 (stdout) に送信されないログにのみ必要です。

   a. Pod 内で root 権限なしにアプリケーションログにアクセスできる場合は、以下のようにログファイルを表示します。

```
$ oc exec my-app-1-akdlg -- cat /var/log/my-application.log
```

b. アプリケーションログの表示に root アクセスが必要な場合は、root 権限でデバッグコンテナーを起動し、コンテナー内でログファイルを表示できます。プロジェクトの **DeploymentConfig** オブジェクトからデバッグコンテナーを起動します。通常、Pod ユーザーは root 以外の権限で実行しますが、問題を調査するために一時的な root 権限で Pod のトラブルシューティングを実行することは役に立ちます。

```
$ oc debug dc/my-deployment-configuration --as-root -- cat /var/log/my-application.log
```



注記

oc debug dc/<deployment_configuration> --as-root を **-- <command>** を追加せずに実行する場合、デバッグ Pod 内で root アクセスでインタラクティブなシェルにアクセスできます。

4. インタラクティブなシェルでアプリケーション機能を対話的にテストし、アプリケーションコンテナーで診断ツールを実行します。

a. アプリケーションコンテナーでインタラクティブなシェルを起動します。

```
$ oc exec -it my-app-1-akdlg /bin/bash
```

b. シェルからアプリケーションの機能を対話的にテストします。たとえば、コンテナーのエントリーポイントコマンドを実行して、結果を確認することができます。次に、ソースコードを更新し、S2I プロセスでアプリケーションコンテナーを再ビルドする前に、コマンドラインから直接に変更をテストします。

c. コンテナー内で利用可能な診断バイナリーを実行します。

注記

一部の診断バイナリーを実行するには、root 権限が必要です。このような状況では、**oc debug dc/<deployment_configuration> --as-root** を実行して、問題のある Pod の **DeploymentConfig** オブジェクトに基づいて、root 権限でデバッグ Pod を起動できます。次に、デバッグ Pod 内から診断バイナリーを root として実行できます。

5. 診断バイナリーがコンテナー内で利用できない場合は、**nsenter** を使用して、コンテナーの namespace 内でホストの診断バイナリーを実行できます。以下の例では、ホストの **ip** バイナリーを使用して、コンテナーの namespace 内で **ip ad** を実行します。

a. ターゲットノードのデバッグセッションに入ります。この手順は、**<node_name>-debug** というデバッグ Pod をインスタンス化します。

```
$ oc debug node/my-cluster-node
```

b. **/host** をデバッグシェル内の root ディレクトリーとして設定します。デバッグ Pod は、Pod 内の **/host** にホストの root ファイルシステムをマウントします。root ディレクトリーを **/host** に変更すると、ホストの実行パスに含まれるバイナリーを実行できます。

```
# chroot /host
```

注記

Red Hat Enterprise Linux CoreOS (RHCOS) を実行する OpenShift Container Platform 4.18 クラスターノードは、イミュータブルです。クラスターの変更を適用するには、Operator を使用します。SSH を使用したクラスターノードへのアクセスは推奨されません。ただし、OpenShift Container Platform API が利用できない場合や、kubelet がターゲットノードで適切に機能しない場合、**oc** 操作がその影響を受けます。この場合は、代わりに **ssh core@<node>.<cluster_name>.<base_domain>** を使用してノードにアクセスできます。

c. ターゲットコンテナー ID を判別します。

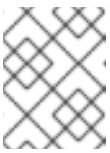
```
# crictl ps
```

d. コンテナーのプロセス ID を確認します。この例では、ターゲットコンテナー ID は **a7fe32346b120** です。

```
# crictl inspect a7fe32346b120 --output yaml | grep 'pid:' | awk '{print $2}'
```

e. ホストの **ip** バイナリーを使用して、コンテナーの namespace 内で **ip ad** を実行します。この例では、**31150** をコンテナーのプロセス ID として使用します。**nsenter** コマンドは、ターゲットプロセスの namespace を入力し、その namespace でコマンドを実行します。この例のターゲットプロセスはコンテナーのプロセス ID であるため、**ip ad** コマンドは、ホストからコンテナーの namespace で実行されます。

```
# nsenter -n -t 31150 -- ip ad
```



注記

デバッグノードなどの特権付きコンテナーを使用している場合のみ、コンテナーの namespace 内でホストの診断バイナリーを実行できます。

7.8.4. 関連情報

- S2I ビルドストラテジーの詳細は、Source-to-Image (S2I) ビルド を参照してください。

7.9. ストレージの問題のトラブルシューティング

7.9.1. 複数割り当てエラーの解決

ノードが予期せずにクラッシュまたはシャットダウンすると、割り当てられた ReadWriteOnce (RWO) ボリュームがノードからアンマウントされ、その後は別のノードでスケジュールされる Pod で使用可能になることが予想されます。

ただし、障害が発生したノードは割り当てられたボリュームをアンマウントできないため、新規ノードにマウントすることはできません。

複数割り当てのエラーが報告されます。

出力例

```
Unable to attach or mount volumes: unmounted volumes=[sso-mysql-pvol], unattached volumes=
[sso-mysql-pvol default-token-x4rzc]: timed out waiting for the condition
Multi-Attach error for volume "pvc-8837384d-69d7-40b2-b2e6-5df86943eef9" Volume is already used
by pod(s) sso-mysql-1-ns6b4
```

手順

複数割り当ての問題を解決するには、以下のソリューションのいずれかを使用します。

- RWX ボリュームを使用して、複数割り当てを有効にします。
  ほとんどのストレージソリューションでは、ReadWriteMany (RWX) ボリュームを使用して、複数割り当てエラーを防ぐことができます。

- RWO ボリュームの使用時に障害が発生したノードを回復するか、削除します。
  VMware vSphere などの RWX をサポートしないストレージの場合、RWO ボリュームが代わりに使用される必要があります。ただし、RWO ボリュームは複数のノードにマウントできません。

  複数割り当てのエラーメッセージが RWO ボリュームと共に表示される場合には、シャットダウンまたはクラッシュしたノードで Pod を強制的に削除し、動的永続ボリュームの割り当て時などの重要なワークロードでのデータ損失を回避します。

  ```
  $ oc delete pod <old_pod> --force=true --grace-period=0
  ```

  このコマンドは、シャットダウンまたはクラッシュしたノードで停止したボリュームを 6 分後に削除します。

7.10. WINDOWS コンテナーのワークロード関連の問題のトラブルシューティング

7.10.1. Windows Machine Config Operator がインストールされない

Windows Machine Config Operator (WMCO) のインストールプロセスを完了しているが、Operator が **InstallWaiting** フェーズのままである場合、問題はネットワークに関係する問題によって引き起こされている可能性があります。

WMCO では、OVN-Kubernetes を使用して OpenShift Container Platform クラスターをハイブリッドネットワークで設定する必要があります。WMCO はハイブリッドネットワークが利用可能でない状態でインストールプロセスを完了できません。これは、複数のオペレーティングシステム (OS) および OS バリアント上でノードを管理するために必要です。これは、クラスターのインストール時に完了する必要があります。

詳細は、ハイブリッドネットワークの設定 を参照してください。

7.10.2. Windows マシンがコンピュートノードにならない理由の調査

Windows マシンがコンピュートノードにならない理由には、各種の理由があります。この問題を調査する最適な方法として、Windows Machine Config Operator (WMCO) ログを収集することができます。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。

- Windows コンピュートマシンセットを作成しました。

手順

- 以下のコマンドを実行して WMCO ログを収集します。

```
$ oc logs -f deployment/windows-machine-config-operator -n openshift-windows-machine-config-operator
```

7.10.3. Windows ノードへのアクセス

Windows ノードは **oc debug node** コマンドを使用してアクセスできません。このコマンドでは、ノードで特権付き Pod を実行する必要があります。これは Windows ではまだサポートされていません。代わりとして Windows ノードは、セキュアシェル (SSH) または Remote Desktop Protocol (RDP) を使用してアクセスできます。どちらの方法にも SSH bastion が必要です。

7.10.3.1. SSH を使用した Windows ノードへのアクセス

セキュアシェル (SSH) を使用して Windows ノードにアクセスできます。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Windows コンピュートマシンセットを作成しました。
- **cloud-private-key** シークレットで使用されるキーおよび ssh-agent に対してクラスターを作成する際に使用されるキーを追加している。セキュリティー上の理由から、キーは使用後は ssh-agent から削除するようにしてください。
- **ssh-bastion** Pod を使用して Windows ノードに接続している。

手順

- 以下のコマンドを実行して Windows ノードにアクセスします。

```
$ ssh -t -o StrictHostKeyChecking=no -o ProxyCommand='ssh -A -o StrictHostKeyChecking=no \
    -o ServerAliveInterval=30 -W %h:%p core@$(oc get service --all-namespaces -l run=ssh-bastion \
    -o go-template="{{ with (index (index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")' <username>@<windows_node_internal_ip> 1 2
```

1. Amazon Web Services (AWS) の **Administrator**、または Microsoft Azure の **capi** などのクラウドプロバイダーのユーザー名を指定します。
2. 以下のコマンドを実行して検出可能な、ノードの内部 IP アドレスを指定します。

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\(@.type==\"InternalIP\"\)].address}
```

7.10.3.2. RDP を使用した Windows ノードへのアクセス

Remote Desktop Protocol (RDP) を使用して Windows ノードにアクセスできます。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Windows コンピュートマシンセットを作成しました。
- **cloud-private-key** シークレットで使用されるキーおよび ssh-agent に対してクラスターを作成する際に使用されるキーを追加している。セキュリティー上の理由から、キーは使用後は ssh-agent から削除するようにしてください。
- **ssh-bastion** Pod を使用して Windows ノードに接続している。

手順

1. 以下のコマンドを実行して SSH トンネルを設定します。

```
$ ssh -L 2020:<windows_node_internal_ip>:3389 \ 1
    core@$(oc get service --all-namespaces -l run=ssh-bastion -o go-template="{{ with (index (index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")
```

1 以下のコマンドを実行して検出可能な、ノードの内部 IP アドレスを指定します。

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\
(@.type==\"InternalIP\")].address}
```

2. 生成されるシェル内で Windows ノードに対して SSH を実行し、以下のコマンドを実行してユーザーのパスワードを作成します。

```
C:\> net user <username> * 1
```

1 AWS の **Administrator**、または Azure の **capi** などのクラウドプロバイダーのユーザー名を指定します。

RDP クライアントを使用して、**localhost:2020** で Windows ノードにリモートでアクセスできるようになりました。

7.10.4. Windows コンテナーの Kubernetes ノードログの収集

Windows コンテナーロギングは Linux コンテナーロギングとは異なる仕方で機能します。Windows ワークロードの Kubernetes ノードログは、デフォルトで **C:\var\logs** ディレクトリーにストリーミングされます。したがって、そのディレクトリーから Windows ノードのログを収集する必要があります。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。

- Windows コンピュートマシンセットを作成しました。

手順

1. **C:\var\logs** のすべてのディレクトリー下でログを表示するには、以下のコマンドを実行します。

```
$ oc adm node-logs -l kubernetes.io/os=windows --path= \
   /ip-10-0-138-252.us-east-2.compute.internal containers \
   /ip-10-0-138-252.us-east-2.compute.internal hybrid-overlay \
   /ip-10-0-138-252.us-east-2.compute.internal kube-proxy \
   /ip-10-0-138-252.us-east-2.compute.internal kubelet \
   /ip-10-0-138-252.us-east-2.compute.internal pods
```

2. 同じコマンドを使用してディレクトリー内のファイルをリスト表示し、個別のログファイルを表示できるようになりました。たとえば、kubelet ログを表示するには、以下のコマンドを実行します。

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=/kubelet/kubelet.log
```

7.10.5. Windows アプリケーションイベントログの収集

kubelet **logs** エンドポイントの **Get-WinEvent** shim は、Windows マシンからアプリケーションイベントログを収集するために使用できます。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Windows コンピュートマシンセットを作成しました。

手順

- Windows マシンのイベントログですべてのアプリケーションロギングのログを表示するには、以下を実行します。

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal
```

同じコマンドが **oc adm must-gather** でログを収集する際に実行されます。

イベントログの他の Windows アプリケーションログは、それぞれのサービスを **-u** フラグで指定して収集することもできます。たとえば、以下のコマンドを実行して containerd コンテナーランタイムサービスのログを収集できます。

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal -u containerd
```

7.10.6. Windows コンテナーの containerd ログの収集

Windows containerd コンテナーサービスは、stdout ではなく Windows イベントログにログデータをストリーミングします。containerd イベントログを表示して、Windows containerd コンテナーサービスによって発生したと思われる問題を調査できます。

前提条件

- Operator Lifecycle Manager (OLM) を使用して Windows Machine Config Operator (WMCO) をインストールしている。
- Windows コンピュートマシンセットを作成しました。

手順

- 次のコマンドを実行して、containerd ログを表示します。

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=containerd
```

7.10.7. 関連情報

- Containers on Windows troubleshooting
- Troubleshoot host and container image mismatches
- Common Kubernetes problems with Windows

7.11. モニタリング関連の問題の調査

OpenShift Container Platform には、コアプラットフォームコンポーネントのモニタリングを提供する事前に設定され、事前にインストールされた自己更新型のモニタリングスタックが含まれます。OpenShift Container Platform 4.18 では、クラスター管理者は必要に応じてユーザー定義プロジェクトのモニタリングを有効にできます。

次の問題が発生した場合は、このセクションの手順に従ってください。

- 独自のメトリクスが利用できない。
- Prometheus が大量のディスク容量を消費している。
- Prometheus に対して **KubePersistentVolumeFillingUp** アラートが発生している。

7.11.1. ユーザー定義のプロジェクトメトリクスが使用できない理由の調査

ServiceMonitor リソースを使用すると、ユーザー定義プロジェクトでサービスによって公開されるメトリクスの使用方法を判別できます。**ServiceMonitor** リソースを作成している場合で、メトリクス UI に対応するメトリクスが表示されない場合は、この手順で説明されるステップを実行します。

前提条件

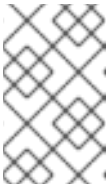
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- ユーザー定義プロジェクトのユーザー定義のプロジェクトのモニタリングを有効にして設定した。
- **ServiceMonitor** リソースを作成した。

手順

1. プロジェクトとリソースがユーザーワークロード監視から除外されていないことを確認します。次の例では、**ns1** プロジェクトを使用します。

   a. プロジェクトに **openshift.io/user-monitoring=false** ラベルが **割り当てられていない** ことを確認します。

   ```
   $ oc get namespace ns1 --show-labels | grep 'openshift.io/user-monitoring=false'
   ```

   注記

   ユーザーワークロードのプロジェクトに設定されるデフォルトのラベルは、**openshift.io/user-monitoring=true** です。ただし、ラベルは手動で適用しない限り表示されません。

   b. **ServiceMonitor** および **PodMonitor** リソースに **openshift.io/user-monitoring=false** ラベルが **割り当てられていない** ことを確認します。次の例では、**prometheus-example-monitor** サービスモニターをチェックします。

   ```
   $ oc -n ns1 get servicemonitor prometheus-example-monitor --show-labels | grep
   'openshift.io/user-monitoring=false'
   ```

   c. ラベルが割り当てられている場合は、ラベルを削除します。

   プロジェクトからラベルを削除する例

   ```
   $ oc label namespace ns1 'openshift.io/user-monitoring-'
   ```

   リソースからラベルを削除する例

   ```
   $ oc -n ns1 label servicemonitor prometheus-example-monitor 'openshift.io/user-
   monitoring-'
   ```

   出力例

   ```
   namespace/ns1 unlabeled
   ```

2. サービスと **ServiceMonitor** リソース設定の対応するラベルが一致していることを確認します。次の例では、**prometheus-example-app** サービス、**prometheus-example-monitor** サービスモニター、および **ns1** プロジェクトを使用します。

   a. サービスに定義されたラベルを取得します。

   ```
   $ oc -n ns1 get service prometheus-example-app -o yaml
   ```

   出力例

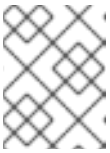
   ```
     labels:
       app: prometheus-example-app
   ```

   b. **ServiceMonitor** リソース設定の **matchLabels** 定義が、直前の手順のラベルの出力と一致することを確認します。

```
$ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
```

出力例

```
apiVersion: v1
kind: ServiceMonitor
metadata:
  name: prometheus-example-monitor
  namespace: ns1
spec:
  endpoints:
  - interval: 30s
    port: web
    scheme: http
  selector:
    matchLabels:
      app: prometheus-example-app
```



注記

プロジェクトの表示権限を持つ開発者として、サービスおよび **ServiceMonitor** リソースラベルを確認できます。

3. **openshift-user-workload-monitoring** プロジェクトで Prometheus Operator のログを調べます。

   a. **openshift-user-workload-monitoring** プロジェクトの Pod をリスト表示します。

   ```
   $ oc -n openshift-user-workload-monitoring get pods
   ```

   出力例

   ```
   NAME                                  READY   STATUS    RESTARTS   AGE
   prometheus-operator-776fcbbd56-2nbfm   2/2     Running   0          132m
   prometheus-user-workload-0             5/5     Running   1          132m
   prometheus-user-workload-1             5/5     Running   1          132m
   thanos-ruler-user-workload-0           3/3     Running   0          132m
   thanos-ruler-user-workload-1           3/3     Running   0          132m
   ```

   b. **prometheus-operator** Pod の **prometheus-operator** コンテナーからログを取得します。以下の例では、Pod は **prometheus-operator-776fcbbd56-2nbfm** になります。

   ```
   $ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-
   2nbfm -c prometheus-operator
   ```

   サービスモニターに問題がある場合、ログには以下のようなエラーが含まれる可能性があります。

   ```
   level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829
   component=prometheusoperator msg="skipping servicemonitor" error="it accesses file
   system via bearer token file which Prometheus specification prohibits"
   ```

```
servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring
prometheus=user-workload
```

4. OpenShift Container Platform Web コンソール UI の **Metrics targets** ページで、エンドポイントのターゲットステータスを確認します。

   a. OpenShift Container Platform の Web コンソールにログインし、**管理者** パースペクティブの **Observe → Targets** に移動します。

   b. リストでメトリクスのエンドポイントを探し、**Status** 列でターゲットのステータスを確認します。

   c. **Status** が **Down** の場合、エンドポイントの URL をクリックすると、そのメトリクスターゲットの **Target Details** ページで詳細情報を見ることができます。

5. **openshift-user-workload-monitoring** プロジェクトで Prometheus Operator のデバッグレベルのロギングを設定します。

   a. **openshift-user-workload-monitoring** プロジェクトで **user-workload-monitoring-config ConfigMap** オブジェクトを編集します。

```
$ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-
config
```

   b. **prometheusOperator** の **logLevel: debug** を **data/config.yaml** に追加し、ログレベルを **debug** に設定します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
    prometheusOperator:
      logLevel: debug
# ...
```

   c. 変更を適用するためにファイルを保存します。影響を受ける **prometheus-operator** Pod は自動的に再デプロイされます。

   d. **debug** ログレベルが **openshift-user-workload-monitoring** プロジェクトの **prometheus-operator** デプロイメントに適用されていることを確認します。

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

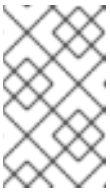
   出力例

```
        - --log-level=debug
```

   デバッグレベルのロギングでは、Prometheus Operator によって行われるすべての呼び出しが表示されます。

e. **prometheus-operator** Pod が実行されていることを確認します。

```
$ oc -n openshift-user-workload-monitoring get pods
```



注記

認識されない Prometheus Operator の **loglevel** 値が config map に含まれる場合、**prometheus-operator** Pod が正常に再起動されない可能性があります。

f. デバッグログを確認し、Prometheus Operator が **ServiceMonitor** リソースを使用しているかどうかを確認します。ログで他の関連するエラーの有無を確認します。

関連情報

- ユーザー定義プロジェクトのモニタリングの有効化
- サービスモニターまたは Pod モニターの作成方法に関する詳細は、サービスのモニター方法の指定 を参照してください。
- メトリクスターゲットに関する詳細情報の取得 を参照してください。

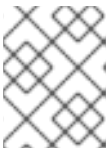
7.11.2. Prometheus が大量のディスク領域を消費している理由の特定

開発者は、キーと値のペアの形式でメトリクスの属性を定義するためにラベルを作成できます。使用できる可能性のあるキーと値のペアの数は、属性に使用できる可能性のある値の数に対応します。数が無制限の値を持つ属性は、バインドされていない属性と呼ばれます。たとえば、**customer_id** 属性は、使用できる値が無限にあるため、バインドされていない属性になります。

割り当てられるキーと値のペアにはすべて、一意の時系列があります。ラベルに多数のバインドされていない値を使用すると、作成される時系列の数が指数関数的に増加する可能性があります。これは Prometheus のパフォーマンスに影響する可能性があり、多くのディスク領域を消費する可能性があります。

Prometheus が多くのディスクを消費する場合、以下の手段を使用できます。

- どのラベルが最も多くの時系列データを作成しているか詳しく知るには、**Prometheus HTTP API を使用して時系列データベース (TSDB) のステータスを確認** します。これを実行するには、クラスター管理者権限が必要です。
- 収集されている **スクレイプサンプルの数を確認** します。
- ユーザー定義メトリクスに割り当てられるバインドされていない属性の数を減らすことで、**作成される一意の時系列の数を減らします**。



注記

使用可能な値の制限されたセットにバインドされる属性を使用すると、可能なキーと値のペアの組み合わせの数が減ります。

- ユーザー定義のプロジェクト全体で **スクレイピングできるサンプルの数に制限を適用** します。これには、クラスター管理者の権限が必要です。

前提条件

- **cluster-admin** クラスターロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **Administrator** パースペクティブで、**Observe → Metrics** に移動します。
2. **Expression** フィールドに、Prometheus Query Language (PromQL) クエリーを入力します。次のクエリー例は、ディスク領域の消費量の増加につながる可能性のある高カーディナリティメトリクスを識別するのに役立ちます。
   - 次のクエリーを実行すると、スクレイプサンプルの数が最も多いジョブを 10 個特定できます。

     ```
     topk(10, max by(namespace, job) (topk by(namespace, job) (1,
     scrape_samples_post_metric_relabeling)))
     ```

   - 次のクエリーを実行すると、過去 1 時間に最も多くの時系列データを作成したジョブを 10 個特定して、時系列のチャーンを正確に特定できます。

     ```
     topk(10, sum by(namespace, job) (sum_over_time(scrape_series_added[1h])))
     ```

3. 想定よりもサンプルのスクレイプ数が多いメトリクスに割り当てられたラベルで、値が割り当てられていないものの数を確認します。
   - **メトリクスがユーザー定義のプロジェクトに関連する場合**、ワークロードに割り当てられたメトリクスのキーと値のペアを確認します。これらのライブラリーは、アプリケーションレベルで Prometheus クライアントライブラリーを使用して実装されます。ラベルで参照されるバインドされていない属性の数の制限を試行します。
   - **メトリクスが OpenShift Container Platform のコアプロジェクトに関連する場合**、Red Hat サポートケースを Red Hat カスタマーポータル で作成してください。
4. クラスター管理者としてログインしてから、次の手順に従い Prometheus HTTP API を使用して TSDB ステータスを確認します。
   a. 次のコマンドを実行して、Prometheus API ルート URL を取得します。

      ```
      $ HOST=$(oc -n openshift-monitoring get route prometheus-k8s -
      ojsonpath='{.status.ingress[].host}')
      ```

   b. 次のコマンドを実行して認証トークンを抽出します。

      ```
      $ TOKEN=$(oc whoami -t)
      ```

   c. 次のコマンドを実行して、Prometheus の TSDB ステータスのクエリーを実行します。

      ```
      $ curl -H "Authorization: Bearer $TOKEN" -k "https://$HOST/api/v1/status/tsdb"
      ```

      出力例

      ```
      "status": "success","data":{"headStats":{"numSeries":507473,
      ```

```
"numLabelPairs":19832,"chunkCount":946298,"minTime":1712253600010,
"maxTime":1712257935346},"seriesCountByMetricName":
[{"name":"etcd_request_duration_seconds_bucket","value":51840},
{"name":"apiserver_request_sli_duration_seconds_bucket","value":47718},
...
```

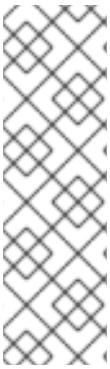
関連情報

- ユーザー定義プロジェクトのスクレイピング間隔、評価間隔、および制限適用の設定

7.11.3. Prometheus に対する KubePersistentVolumeFillingUp アラートの解決

クラスター管理者は、Prometheus に対してトリガーされている **KubePersistentVolumeFillingUp** アラートを解決できます。

openshift-monitoring プロジェクトの **prometheus-k8s-*** Pod によって要求された永続ボリューム (PV) の合計残り容量が 3% 未満になると、重大アラートが発生します。これにより、Prometheus の動作異常が発生する可能性があります。



注記

KubePersistentVolumeFillingUp アラートは 2 つあります。

- **重大アラート**: マウントされた PV の合計残り容量が 3% 未満になると、**severity="critical"** ラベルの付いたアラートがトリガーされます。
- **警告アラート**: マウントされた PV の合計空き容量が 15% 未満になり、4 日以内にいっぱいになると予想される場合、**severity="warning"** ラベルの付いたアラートがトリガーされます。

この問題に対処するには、Prometheus 時系列データベース (TSDB) のブロックを削除して、PV 用のスペースを増やすことができます。

前提条件

- **cluster-admin** クラスターロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 次のコマンドを実行して、すべての TSDB ブロックのサイズを古いものから新しいものの順にリスト表示します。

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \ (1)
-c prometheus --image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \
(2)
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') \
-- sh -c 'cd /prometheus/;du -hs $(ls -dtr */ | grep -Eo "[0-9|A-Z]{26}")'
```

(1) (2) **<prometheus_k8s_pod_name>** は、**KubePersistentVolumeFillingUp** アラートの説明に記載されている Pod に置き換えます。

出力例

```
308M    01HVKMPKQWZYWS8WVDAYQHNMW6
52M     01HVK64DTDA81799TBR9QDECEZ
102M    01HVK64DS7TRZRWF2756KHST5X
140M    01HVJS59K11FBVAPVY57K88Z11
90M     01HVH2A5Z58SKT810EM6B9AT50
152M    01HV8ZDVQMX41MKCN84S32RRZ1
354M    01HV6Q2N26BK63G4RYTST71FBF
156M    01HV664H9J9Z1FTZD73RD1563E
216M    01HTHXB60A7F239HN7S2TENPNS
104M    01HTHMGRXGS0WXA3WATRXHR36B
```

2. 削除できるブロックとその数を特定し、ブロックを削除します。次のコマンド例は、**prometheus-k8s-0** Pod から最も古い 3 つの Prometheus TSDB ブロックを削除します。

```
$ oc debug prometheus-k8s-0 -n openshift-monitoring \
-c prometheus --image=$(oc get po -n openshift-monitoring prometheus-k8s-0 \
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') \
-- sh -c 'ls -latr /prometheus/ | egrep -o "[0-9|A-Z]{26}" | head -3 | \
while read BLOCK; do rm -r /prometheus/$BLOCK; done'
```

3. 次のコマンドを実行して、マウントされた PV の使用状況を確認し、十分な空き容量があることを確認します。

```
$ oc debug <prometheus_k8s_pod_name> -n openshift-monitoring \ (1)
--image=$(oc get po -n openshift-monitoring <prometheus_k8s_pod_name> \ (2)
-o jsonpath='{.spec.containers[?(@.name=="prometheus")].image}') -- df -h /prometheus/
```

(1) (2) **<prometheus_k8s_pod_name>** は、**KubePersistentVolumeFillingUp** アラートの説明に記載されている Pod に置き換えます。

次の出力例は、**prometheus-k8s-0** Pod によって要求されるマウントされた PV に、63% の空き容量が残っていることを示しています。

出力例

```
Starting pod/prometheus-k8s-0-debug-j82w4 ...
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1p4  40G   15G  40G  37% /prometheus

Removing debug pod ...
```

7.12. OPENSHIFT CLI (OC) 関連の問題の診断

7.12.1. OpenShift CLI (oc) ログレベルについて

OpenShift CLI (**oc**) を使用すると、ターミナルからアプリケーションを作成し、OpenShift Container Platform プロジェクトを管理できます。

oc コマンド固有の問題が発生した場合は、**oc** のログレベルを引き上げ、コマンドで生成される API 要求、API 応答、および **curl** 要求の詳細を出力します。これにより、特定の **oc** コマンドの基礎となる操作の詳細ビューが得られます。これにより、障害の性質に関する洞察が得られる可能性があります。

oc ログレベルは、1 から 10 まであります。以下の表は、 **oc** ログレベルのリストとそれらの説明を示しています。

表7.4 OpenShift CLI (oc) ログレベル

ログレベル	説明
1 - 5	標準エラー (stderr) への追加のロギングはありません。
6	標準エラー (stderr) に API 要求のログを記録します。
7	標準エラー (stderr) に API 要求およびヘッダーのログを記録します。
8	標準エラー (stderr) に API 要求、ヘッダーおよび本体、ならびに API 応答ヘッダーおよび本体のログを記録します。
9	標準エラー (stderr) に API 要求、ヘッダーおよび本体、API 応答ヘッダーおよび本体、**curl** 要求のログを記録します。
10	標準エラー (stderr) に API 要求、ヘッダーおよび本体、API 応答ヘッダーおよび本体、**curl** 要求のログを詳細に記録します。

7.12.2. OpenShift CLI (oc) ログレベルの指定

コマンドのログレベルを引き上げて、OpenShift CLI (**oc**) の問題を調査できます。

通常、OpenShift Container Platform ユーザーの現行セッショントークンは、必要に応じてログに記録される **curl** 要求に含まれます。また、**oc** コマンドの基礎となるプロセスを手順ごとにテストする際に使用するために、現行ユーザーのセッショントークンを手動で取得することもできます。

前提条件

- OpenShift CLI (**oc**) がインストールされている。

手順

- **oc** コマンドの実行時に **oc** ログレベルを指定します。

```
$ oc <command> --loglevel <log_level>
```

ここでは、以下のようになります。

<command>
実行しているコマンドを指定します。

<log_level>
コマンドに適用するログレベルを指定します。

- 現行ユーザーのセッショントークンを取得するには、次のコマンドを実行します。

```
$ oc whoami -t
```

出力例

```
sha256~RCV3Qcn7H-OEfqCGVI0CvnZ6...
```