OpenShift Container Platform 4.12

OpenShift 向けのサンドボックスコンテナーのサポート

OpenShift サンドボックスコンテナーガイド

Last Updated: 2023-03-25

OpenShift Container Platform 4.12 OpenShift 向けのサンドボックスコンテナーのサポート

OpenShift サンドボックスコンテナーガイド

法律上の通知

概要

OpenShift Container Platform の OpenShift サンドボックスコンテナーがサポートされることで、追加のオプションランタイムとして、Kata Container を実行するビルドインサポートが追加されます。

目次

第1章 {SANDBOXED-CONTAINERS-FIRST} {SANDBOXED-CONTAINERS-VERSION} リリースノート

1.1. 本リリースについて

これらのリリースノートは、Red Hat OpenShift Container Platform 4.12 とともに OpenShift サンドボックスコンテナー 1.3 の開発を追跡します。

この製品は、OpenShift Container Platform 4.10 の時点で完全にサポートされ、デフォルトで有効になっています。

1.2. 新機能および機能拡張

1.2.1. メトリクスリストのコンテナー ID

関連するサンドボックスコンテナーの ID を持つ **sandbox_id** が、Web コンソールの **Metrics** ページのメトリックリストに表示されるようになりました。

さらに、**kata-monitor** プロセスは、kata 固有のメトリクスに **cri_uid**、**cri_name**、および **cri_namespace** の 3 つの新しいラベルを追加するようになりました。これらのラベルにより、kata 固有のメトリックを対応する kubernetes ワークロードに関連付けることができます。

kata 固有のメトリクスの詳細については、OpenShift サンドボックスコンテナーのメトリクスについてを参照してください。

1.2.2. AWS ベアメタルでの OpenShift サンドボックスコンテナーの可用性

以前は、AWS ベアメタルでの OpenShift サンドボックスコンテナーの可用性はテクノロジープレビューでした。このリリースでは、AWS ベアメタルクラスターへの OpenShift サンドボックスコンテナーのインストールが完全にサポートされています。

1.2.3. 単一ノード OpenShift での OpenShift サンドボックスコンテナーのサポート

OpenShift サンドボックスコンテナー Operator が Red Hat Advanced Cluster Management (RHACM) によってインストールされている場合、OpenShift サンドボックスコンテナーが単一ノードの OpenShift クラスターで機能するようになりました。

1.3. 更新

KataConfig カスタムリソースを作成するときに、**kataMonitorImage** は不要になりました。この更新は、OpenShift サンドボックスコンテナー 1.3.2 で実装され、OpenShift サンドボックスコンテナー Operator のすべてのバージョン間で下位互換性があります。

1.4. バグ修正

- 以前のリリースでは、OpenShift サンドボックスコンテナーを OpenShift Container Platform 4.10 にインストールすると、コントローラーマネージャー POD が次のエラーで起動できませんでした。

```
container has runAsNonRoot and image has non-numeric user (nonroot),
        cannot verify user is non-root
```

この問題が原因で、**KataConfig** CR を作成できず、OpenShift サンドボックスコンテナーを実行できませんでした。

このリリースでは、数値のユーザー ID (499) を使用するようにマネージャーコンテナーイメージが更新されました。(KATA-1824)

- 以前は、**KataConfig** CR を作成し、**openshift-sandboxed-containers-operator** namespace で Pod のステータスを観察すると、モニター Pod の膨大な数の再起動が示されていました。モニター Pod は、**sandboxed-containers** 拡張機能のインストールの一部としてインストールされた特定の SELinux ポリシーを使用します。モニター Pod がすぐに作成されました。ただし、SELinux ポリシーはまだ利用できなかったため、Pod 作成エラーが発生し、その後 Pod が再起動されました。
このリリースでは、モニター Pod の作成時に SELinux ポリシーを使用でき、モニター Pod はすぐに **Running** 状態に移行します。(KATA-1338)

- 以前は、OpenShift サンドボックスコンテナーは、起動時にセキュリティーコンテキスト制約 (SCC) をデプロイし、Machine Config Operator (MCO) Pod では利用できなかったカスタム SELinux ポリシーを適用していました。これにより、MCO Pod が **CrashLoopBackOff** 状態に変わり、クラスターのアップグレードが失敗しました。このリリースでは、OpenShift サンドボックスコンテナーは、**KataConfig** CR の作成時に SCC をデプロイし、カスタム SELinux ポリシーの使用を強制しなくなりました。(KATA-1373)

- 以前は、OpenShift サンドボックスコンテナー Operator をアンインストールするときに、**sandboxed-containers-operator-scc** カスタムリソースが削除されませんでした。今回のリリースでは、OpenShift サンドボックスコンテナー Operator をアンインストールすると、**sandboxed-containers-operator-scc** カスタムリソースが削除されます。(KATA-1569)

1.5. 既知の問題

- OpenShift サンドボックスコンテナーを使用している場合、コンテナーレベルで SELinux Multi-Category Security (MCS) ラベルを設定すると、Pod は次のエラーで起動に失敗します。

```
Error: CreateContainer failed: EACCES: Permission denied: unknown
```

コンテナーレベルで設定された MCS ラベルは、virtiofsd には適用されません。**kata** ランタイムは、VM の作成時にコンテナーのセキュリティーコンテキストにアクセスできません。

つまり、virtiofsd が適切な SELinux ラベルで実行されず、VM 内のコンテナーに代わってホストファイルにアクセスできず、すべてのコンテナーは、VM 内のすべてのファイルにアクセスできます。

(KATA-1875)

- OpenShift サンドボックスコンテナーを使用している場合、OpenShift Container Platform クラスターの **hostPath** ボリュームからマウントされたファイルまたはディレクトリーにアクセスすると、SELinux 拒否を受け取ることがあります。これらの拒否は、特権サンドボックスコンテナーを実行している場合でも発生する可能性があります。これは、特権サンドボックスコンテナーが SELinux チェックを無効にしないためです。
ホストで SELinux ポリシーに従うことで、デフォルトでサンドボックス化されたワークロードからホストファイルシステムを完全に分離することが保証されます。これにより、**virtiofsd** デーモンまたは QEMU の潜在的なセキュリティー上の欠陥に対する保護も強化されます。

マウントされたファイルまたはディレクトリーにホスト上の特定の SELinux 要件がない場合は、代わりにローカル永続ボリュームを使用できます。ファイルは、コンテナーランタイムの SELinux ポリシーに従って、自動的に **container_file_t** に再ラベル付けされます。詳細は、ローカルボリュームを使用した永続ストレージ を参照してください。

マウントされたファイルまたはディレクトリーがホスト上で特定の SELinux ラベルを持つことが予想される場合、自動再ラベル付けはオプションではありません。代わりに、ホストでカスタム SELinux ルールを設定して、**virtiofsd** デーモンがこれらの特定のラベルにアクセスできるようにすることができます。(BZ#1904609)

- 一部の OpenShift サンドボックスコンテナー Operator Pod は、コンテナーの CPU リソース制限を使用して、Pod で使用可能な CPU の数を増やします。これらの Pod は、要求されたよりも少ない CPU を受け取る可能性があります。コンテナー内で機能が利用可能な場合は、**oc rsh <pod>** を使用して Pod にアクセスし、**lscpu** コマンドを実行することで、CPU リソースの問題を診断できます。

```
$ lscpu
```

出力例

```
CPU(s):                16
On-line CPU(s) list:   0-12,14,15
Off-line CPU(s) list:  13
```

オフライン CPU のリストは、実行ごとに予期せず変更される可能性があります。

回避策として、CPU 制限を設定するのではなく、Pod アノテーションを使用して追加の CPU をリクエストできます。Pod アノテーションを使用する CPU リクエストは、プロセッサーの割り当て方法が異なるため、この問題の影響を受けません。CPU 制限を設定するのではなく、Pod のメタデータに次の **annotation** を追加する必要があります。

```
metadata:
  annotations:
    io.katacontainers.config.hypervisor.default_vcpus: "16"
```

(KATA-1376)

- ランタイムインストールの進行状況は、**kataConfig** カスタムリソース (CR) の **status** セクションに表示されます。ただし、次の条件がすべて当てはまる場合、進行状況は表示されません。
  - ワーカーノードが定義されていません。**oc get machineconfigpool** を実行して、マシン設定プール内のワーカーノードの数を確認できます。
  - インストールするノードを選択するための **kataConfigPoolSelector** が指定されていません。

この場合、Operator はノードがコントロールプレーンとワーカーの両方のロールを持つコンバージドクラスターであると想定するため、コントロールプレーンノードでインストールが開始されます。**kataConfig** CR の **status** セクションは、インストール中に更新されません。(KATA-1017)

- OpenShift サンドボックスコンテナーで古いバージョンの Buildah ツールを使用すると、ビルドが次のエラーで失敗します。

```
process exited with error: fork/exec /bin/sh: no such file or directory

subprocess exited with status 1
```

quay.io で入手可能な Buildah の最新バージョンを使用する必要があります。

(KATA-1278)

- Web コンソールの **KataConfig** タブで、**YAML view** で **Create KataConfig** をクリックすると、**KataConfig** YAML に **spec** フィールドがありません。**Form view** に切り替えてから **YAML view** に戻ると、この問題が修正され、完全な YAML が表示されます。(KATA-1372)

- Web コンソールの **KataConfig** タブに、**KataConfig** CR がすでに存在するかどうかにかかわらず、**404: Not found** エラーメッセージが表示されます。既存の **KataConfig** CR にアクセスするには、**Home** > **Search** に移動します。**Resources** リストから、**KataConfig** を選択します。(KATA-1605)

- OpenShift サンドボックスコンテナーをアップグレードしても、既存の **KataConfig** CR は自動的に更新されません。その結果、以前のデプロイメントのモニター Pod は再起動されず、古い **kataMonitor** イメージで引き続き実行されます。
次のコマンドを使用して、**kataMonitor** イメージをアップグレードします。

```
$ oc patch kataconfig example-kataconfig --type merge --patch '{"spec":
{"kataMonitorImage":"registry.redhat.io/openshift-sandboxed-containers/osc-monitor-
rhel8:1.3.0"}}'
```

Web コンソールで **KataConfig** YAML を編集して、**kataMonitor** イメージをアップグレードすることもできます。

(KATA-1650)

1.6. エラータの非同期更新

OpenShift sandboxed containers 4.12 のセキュリティー、バグ修正、拡張機能の更新は、Red Hat Network 経由で非同期エラータとして発表されます。OpenShift Container Platform 4.12 のすべてのエラータは Red Hat カスタマーポータルから入手できます。非同期エラータのより詳細については、OpenShift Container Platform ライフサイクル を参照してください。

Red Hat カスタマーポータルのユーザーは、Red Hat Subscription Management (RHSM) のアカウント設定でエラータの通知を有効にすることができます。エラータの通知を有効にすると、登録しているシステムに関連するエラータが新たに発表されるたびに、メールで通知が送信されます。

注記

OpenShift Container Platform のエラータ通知メールを生成させるには、Red Hat カスタマーポータルのユーザーアカウントでシステムが登録されており、OpenShift Container Platform エンタイトルメントを使用している必要があります。

以下のセクションは、これからも継続して更新され、今後の OpenShiftOpenShift sandboxed containers 1.3バージョンの非同期リリースで発表されるエラータの拡張機能およびバグ修正に関する情報を追加していきます。

1.6.1. RHSA-2022:6072 - OpenShift サンドボックスコンテナー 1.3.0 イメージのリリース、バグ修正、機能強化のアドバイザリー

発行日: 2022-08-17

OpenShift サンドボックスコンテナーリリース 1.3.0 が利用可能になりました。このアドバイザリーには、機能強化とバグ修正を含む OpenShift サンドボックスコンテナーの更新が含まれています。

更新に含まれるバグ修正のリストは、RHSA-2022:6072 アドバイザリーに記載されています。

1.6.2. RHSA-2022:7058 - OpenShift サンドボックスコンテナー 1.3.1 セキュリティー修正およびバグ修正アドバイザリー

発行日: 2022-10-19

OpenShift サンドボックスコンテナーリリース 1.3.1 が利用可能になりました。このアドバイザリーには、セキュリティー修正とバグ修正を含む OpenShift サンドボックスコンテナーの更新が含まれています。

更新に含まれるバグ修正のリストは、RHSA-2022:7058 アドバイザリーに記載されています。

1.6.3. RHBA-2023:0390 - OpenShift サンドボックスコンテナー 1.3.2 バグ修正アドバイザリー

発行日: 2023-01-24

OpenShift サンドボックスコンテナーリリース 1.3.2 が利用可能になりました。このアドバイザリーには、バグ修正を含む OpenShift サンドボックスコンテナーの更新が含まれています。

この更新に含まれるバグ修正の一覧は、RHBA-2023:0390 アドバイザリーにまとめられています。

1.6.4. RHBA-2023:0485 - OpenShift サンドボックスコンテナー 1.3.3 バグ修正アドバイザリー

発行日: 2023-01-30

OpenShift サンドボックスコンテナーリリース 1.3.3 が利用可能になりました。このアドバイザリーには、OpenShift サンドボックスコンテナーの更新と、バグ修正およびコンテナーの更新が含まれています。

更新に含まれるバグ修正のリストは、RHBA-2023:0485 アドバイザリーに記載されています。

第2章 OPENSHIFT サンドボックスコンテナーについて

OpenShift Container Platform の OpenShift サンドボックスコンテナーがサポートされることで、追加のオプションランタイムとして、Kata Container を実行するビルドインサポートが追加されます。新しいランタイムは、専用の仮想マシン (VM) でコンテナーをサポートし、ワークロードの分離を改善します。これは、次のタスクを実行する場合に特に役立ちます。

特権または信頼できないワークロードを実行する

OpenShift サンドボックスコンテナー (OSC) を使用すると、特権コンテナーを実行してクラスターノードを危険にさらすことなく、特定の特権を必要とするワークロードを安全に実行できます。特別な権限を必要とするワークロードには、次のものがあります。

- たとえば、低レベルのネットワーク機能にアクセスするために、CRI-O などの標準コンテナーランタイムによって付与されるデフォルトの機能を超えて、カーネルからの特別な機能を必要とするワークロード。
- 特定の物理デバイスにアクセスする場合など、ルート権限の昇格が必要なワークロード。OpenShift サンドボックスコンテナーを使用すると、特定のデバイスのみを VM に渡すことができるため、ワークロードがシステムの残りの部分にアクセスしたり、設定を誤ったりすることはありません。
- **set-uid** ルートバイナリーをインストールまたは使用するためのワークロード。これらのバイナリーは特別な権限を付与するため、セキュリティーリスクが生じる可能性があります。OpenShift サンドボックスコンテナーを使用すると、追加の権限は仮想マシンに制限され、クラスターノードへの特別なアクセスは許可されません。

一部のワークロードでは、特にクラスターノードを設定するための特権が必要になる場合があります。このようなワークロードは、仮想マシンで実行すると機能しなくなるため、引き続き特権コンテナーを使用する必要があります。

各ワークロードのカーネルを確実に分離する

OpenShift サンドボックスコンテナーは、カスタムカーネルチューニング (**sysctl**、スケジューラーの変更、キャッシュチューニングなど) とカスタムカーネルモジュールの作成 (**out of tree** や特別な引数など) を必要とするワークロードをサポートします。

テナント全体で同じワークロードを共有する

OpenShift サンドボックスコンテナーを使用すると、同じ OpenShift クラスターを共有するさまざまな組織の複数のユーザー (テナント) をサポートできます。このシステムでは、コンテナーネットワーク機能 (CNF) やエンタープライズアプリケーションなど、複数のベンダーのサードパーティーワークロードを実行することもできます。たとえば、サードパーティーの CNF は、カスタム設定がパケットチューニングや他のアプリケーションによって設定された **sysctl** 変数に干渉することを望まない場合があります。完全に分離されたカーネル内で実行すると、ノイジーネイバー設定の問題を防ぐのに役立ちます。

ソフトウェアのテストに適した分離とサンドボックスがあることを確認する

OpenShift サンドボックスコンテナーを使用して、既知の脆弱性を持つコンテナー化されたワークロードを実行したり、レガシーアプリケーションの問題を処理したりできます。この分離により、管理者は Pod に対する管理制御を開発者に付与することもできます。これは、開発者が、管理者が通常許可する設定を超えて設定をテストまたは検証したい場合に役立ちます。たとえば、管理者は、安全かつ確実にカーネルパケットフィルタリング (eBPF) を開発者に委譲できます。カーネルパケットフィルタリングには **CAP_ADMIN** または **CAP_BPF** 権限が必要なため、標準の CRI-O 設定では許可されません。これにより、コンテナーホストワーカーノード上のすべてのプロセスへのアクセスが許可されるためです。同様に、管理者は SystemTap などの侵入型ツールへのアクセスを許可したり、開発中にカスタムカーネルモジュールのロードをサポートしたりできます。

仮想マシン境界を使用して、デフォルトのリソースに含まれるようにする

デフォルトでは、CPU、メモリー、ストレージ、ネットワークなどのリソースは、OpenShift サンドボックスコンテナーでより堅牢で安全な方法で管理されます。OpenShift サンドボックスコンテナーは仮想マシンにデプロイされるため、分離とセキュリティーのレイヤーを追加することで、リソースへのアクセスをよりきめ細かく制御できます。たとえば、誤ったコンテナーは、仮想マシンで使用できる以上のメモリーを割り当てることができません。逆に、ネットワークカードまたはディスクへの専用アクセスが必要なコンテナーは、他のデバイスにアクセスすることなく、そのデバイスを完全に制御できます。

2.1. OPENSHIFT サンドボックスコンテナーがサポートするプラットフォーム

ベアメタルサーバーまたはアマゾンウェブサービス (AWS) ベアメタルインスタンスに OpenShift サンドボックスコンテナーをインストールできます。他のクラウドプロバイダーが提供するベアメタルインスタンスはサポートされません。

Red Hat Enterprise Linux CoreOS (RHCOS) は、OpenShift サンドボックスコンテナーで唯一サポートされているオペレーティングシステムです。OpenShift サンドボックスコンテナー 1.3 は、Red Hat Enterprise Linux CoreOS (RHCOS) 8.6 で実行されます。

OpenShift サンドボックスコンテナー 1.3 は、OpenShift Container Platform 4.11 と互換性があります。

2.2. OPENSHIFT サンドボックスコンテナーの一般的な用語

以下の用語は、本書全体で使用されています。

サンドボックス

サンドボックスとは、プログラムが実行可能な分離された環境のことです。サンドボックスでは、ホストマシンやオペレーティングシステムに悪影響を及ぼすことなく、テストされていないプログラムまたは信頼できないプログラムを実行できます。
OpenShift サンドボックスコンテナーのコンテキストでは、仮想化を使用して異なるカーネルでワークロードを実行し、同じホストで実行される複数のワークロードとの間の対話を強化することでサンドボックスを実現します。

Pod

Pod は Kubernetes および OpenShift Container Platform から継承されるコンストラクトです。Pod とは、コンテナーのデプロイが可能なリソースを表します。コンテナーは Pod 内で実行され、Pod を使用して複数のコンテナー間で共有できるリソースを指定します。
OpenShift サンドボックスコンテナーのコンテキストでは、Pod が仮想マシンとして実装されます。同じ仮想マシンにある同じ Pod でコンテナーを複数実行できます。

OpenShift サンドボックスコンテナー Operator

Operator は、人間のオペレーターがシステムで実行できるアクション、つまり操作を自動化するソフトウェアコンポーネントです。
OpenShift サンドボックスコンテナー Operator は、クラスター上でサンドボックスコンテナーのライフサイクルを管理してタスクを実行します。OpenShift サンドボックスコンテナー Operator を使用して、サンドボックスコンテナーのインストールと削除、ソフトウェア更新、ステータス監視などのタスクを実行できます。

Kata Container

Kata Container は OpenShift サンドボックスコンテナーの構築に使用されるコアアップストリームプロジェクトです。OpenShift サンドボックスコンテナーは Kata Container と OpenShift Container Platform を統合します。

KataConfig

KataConfig オブジェクトはサンドボックスコンテナーの設定を表します。ソフトウェアのデプロイ先のノードなど、クラスターの状態に関する情報を保存します。

ランタイムクラス

RuntimeClass オブジェクトは、指定のワークロード実行に使用可能なランタイムを記述します。**kata** という名前のランタイムクラスは、OpenShift のサンドボックスコンテナー Operator によってインストールされ、デプロイされます。ランタイムクラスには、ランタイムが Pod オーバーヘッド など、動作に必要なリソースを記述するランタイムに関する情報が含まれます。

2.3. OPENSHIFT サンドボックスコンテナーのワークロード管理

OpenShift サンドボックスコンテナーは、ワークロードの管理と割り当てを強化するための次の機能を提供します。

2.3.1. OpenShift サンドボックスコンテナーのビルディングブロック

OpenShift サンドボックス化されたコンテナー Operator は、Kata Container からのコンポーネントをすべてカプセル化します。インストール、ライフサイクル、設定タスクを管理します。

OpenShift サンドボックスコンテナー Operator は、2 つのコンテナーイメージとして Operator バンドル形式 でパッケージ化されます。バンドルイメージにはメタデータが含まれ、Operator で OLM が利用できるようにする必要があります。2 つ目のコンテナーイメージには、**KataConfig** リソースを監視および管理するための実際のコントローラーが含まれています。

2.3.2. RHCOS 拡張機能

OpenShift サンドボックスコンテナー Operator は Red Hat Enterprise Linux CoreOS (RHCOS) 拡張機能の概念に基づいています。Red Hat Enterprise Linux CoreOS (RHCOS) 拡張機能は、オプションの OpenShift Container Platform ソフトウェアをインストールするメカニズムです。OpenShift サンドボックスコンテナー Operator はこのメカニズムを使用して、サンドボックスコンテナーをクラスターにデプロイします。

サンドボックスコンテナーの RHCOS 拡張には、Kata、QEMU、およびその依存関係の RPM が含まれます。これらは、Machine Config Operator が提供する **MachineConfig** リソースを使用して有効にできます。

関連情報

- 拡張機能の RHCOS への追加

2.3.3. 仮想化および OpenShift サンドボックスコンテナー

OpenShift Virtualization を使用してクラスターで OpenShift サンドボックスコンテナーを使用できます。

OpenShift Virtualization と OpenShift サンドボックスコンテナーを同時に実行するには、仮想マシンがノードの再起動をブロックしないように、仮想マシンの移行を有効にする必要があります。仮想マシンで次のパラメーターを設定します。

- ストレージクラスとして **ocs-storagecluster-ceph-rbd** を使用します。
- 仮想マシンで **evictionStrategy** パラメーターを **LiveMigrate** に設定します。

関連情報

- 仮想マシンのローカルストレージの設定
- 仮想マシンのエビクションストラテジーの設定

2.4. コンプライアンスおよびリスク管理について

OpenShift サンドボックスコンテナーは、FIPS 対応クラスターで使用できます。

FIPS モードで実行している場合、OpenShift サンドボックスコンテナーコンポーネント、仮想マシン、および VM イメージは、FIPS に準拠するように調整されます。

FIPS コンプライアンスは、安全な環境で必要とされる最も重要なコンポーネントの 1 つであり、サポートされている暗号化技術のみがノード上で許可されるようにします。

重要

FIPS 検証済み/進行中のモジュール (Modules in Process) 暗号ライブラリーの使用は、**x86_64** アーキテクチャーの OpenShift Container Platform デプロイメントでのみサポートされています。

OpenShift Container Platform コンプライアンスフレームワークについての Red Hat のアプローチについては、OpenShift セキュリティーガイド のリスク管理および規制対応の章を参照してください。

第3章 OPENSHIFT サンドボックスコンテナーワークロードのデプロイ

Web コンソールまたは OpenShift CLI (**oc**) のいずれかを使用して OpenShift サンドボックスコンテナー Operator をインストールできます。OpenShift サンドボックスコンテナー Operator をインストールする前に、OpenShift Container Platform クラスターを準備する必要があります。

3.1. 前提条件

OpenShift サンドボックスコンテナーをインストールする前に、OpenShift Container Platform クラスターが以下の要件を満たしていることを確認してください。

- クラスターは、Red Hat Enterprise Linux CoreOS (RHCOS) ワーカーを使用してオンプレミスのベアメタルインフラストラクチャーにインストールする必要があります。ユーザーによってプロビジョニングされる、インストーラーでプロビジョニングされる、または支援付きインストーラーによるインストールなどのインストール方法を使用してクラスターをデプロイできます。

  注記

  - OpenShift サンドボックスコンテナーは RHCOS ワーカーノードのみをサポートします。RHEL ノードはサポートされていません。
  - ネストされた仮想化はサポートされていません。

- Amazon Web Services (AWS) ベアメタルインスタンスに OpenShift サンドボックスコンテナーをインストールできます。他のクラウドプロバイダーが提供するベアメタルインスタンスはサポートされません。

3.1.1. OpenShift サンドボックスコンテナーのリソース要件

OpenShift サンドボックスコンテナーを使用すると、ユーザーはサンドボックスランタイム (Kata) 内の OpenShift Container Platform クラスターでワークロードを実行できます。各 Pod は仮想マシン (VM) で表されます。各仮想マシンは QEMU プロセスで実行され、コンテナーワークロードおよびこれらのコンテナーで実行されているプロセスを管理するためのスーパーバイザーとして機能する **kata-agent** プロセスをホストします。2 つのプロセスを追加すると、オーバーヘッドがさらに増加します。

- **containerd-shim-kata-v2**。これは Pod との通信に使用されます。
- **virtiofsd**。これはゲストの代わりにホストファイルシステムのアクセスを処理します。

各仮想マシンには、デフォルトのメモリー容量が設定されます。コンテナーでメモリーが明示的に要求された場合に、メモリーが追加で仮想マシンにホットプラグされます。

メモリーリソースなしで実行されているコンテナーは、仮想マシンによって使用される合計メモリーが既定の割り当てに達するまで、空きメモリーを消費します。ゲストやその I/O バッファーもメモリーを消費します。

コンテナーに特定のメモリー量が指定されている場合には、コンテナーが起動する前に、メモリーが仮想マシンにホットプラグされます。

メモリー制限が指定されている場合には、上限より多くメモリーが消費された場合に、ワークロードが終了します。メモリー制限が指定されていない場合、仮想マシンで実行されているカーネルがメモリー不足になる可能性があります。カーネルがメモリー不足になると、仮想マシン上の他のプロセスが終了

する可能性があります。

デフォルトのメモリーサイズ

以下の表は、リソース割り当てのデフォルト値を示しています。

リソース	値
デフォルトで仮想マシンに割り当てられるメモリー	2Gi
起動時のゲスト Linux カーネルのメモリー使用量	~110Mi
QEMU プロセスで使用されるメモリー (仮想マシンメモリーを除く)	~30Mi
virtiofsd プロセスで使用されるメモリー (VM I/O バッファーを除く)	~10Mi
containerd-shim-kata-v2 プロセスで使用されるメモリー	~20Mi
Fedora で **dnf install** を実行した後のファイルバッファーのキャッシュデータ	~300Mi* [1]

ファイルバッファーが表示され、このバッファーは以下の複数の場所に考慮されます。

- ファイルバッファーキャッシュとして表示されるゲスト。
- 許可されたユーザー空間ファイルの I/O 操作をマッピングする **virtiofsd** デーモン。
- ゲストメモリーとして使用される QEMU プロセス。

注記

メモリー使用量の合計は、メモリー使用率メトリクスによって適切に考慮され、そのメモリーを 1 回だけカウントします。

Pod のオーバーヘッド では、ノード上の Pod が使用するシステムリソースの量を記述します。以下のように、**oc describe runtimeclass kata** を使用して、Kata ランタイムクラスの現在の Pod オーバーヘッドを取得できます。

例

```
$ oc describe runtimeclass kata
```

出力例

```
kind: RuntimeClass
apiVersion: node.k8s.io/v1
metadata:
  name: kata
```

```
overhead:
  podFixed:
    memory: "500Mi"
    cpu: "500m"
```

RuntimeClass の **spec.overhead** フィールドを変更して、Pod のオーバーヘッドを変更できます。たとえば、コンテナーに対する設定が QEMU プロセスおよびゲストカーネルデータでメモリー 350Mi 以上を消費する場合に、**RuntimeClass** のオーバーヘッドをニーズに合わせて変更できます。

注記

Red Hat では、指定のデフォルトオーバーヘッド値がサポートされます。デフォルトのオーバーヘッド値の変更はサポートされておらず、値を変更すると技術的な問題が発生する可能性があります。

ゲストで種類にかかわらず、ファイルシステム I/O を実行すると、ファイルバッファーがゲストカーネルに割り当てられます。ファイルバッファーは、**virtiofsd** プロセスだけでなく、ホスト上の QEMU プロセスでもマッピングされます。

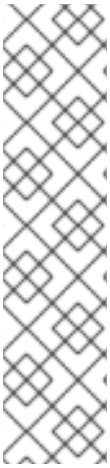
たとえば、ゲストでファイルバッファーキャッシュ 300Mi を使用すると、QEMU と **virtiofsd** の両方が、追加で 300Mi を使用するように見えます。ただし、3 つのケースすべてで同じメモリーが使用されています。つまり、メモリーの合計使用量は 300Mi のみで、このメモリー量が 3 つの異なる場所にマッピングされています。これは、メモリー使用量メトリクスの報告時に適切に考慮されます。

関連情報

- ユーザーによってプロビジョニングされるクラスターのベアメタルへのインストール

3.1.2. クラスターノードが OpenShift サンドボックスコンテナーを実行する資格があるかどうかを確認する

OpenShift サンドボックスコンテナーを実行する前に、クラスター内のノードが Kata コンテナーを実行する資格があるかどうかを確認できます。一部のクラスターノードは、サンドボックスコンテナーの最小要件に準拠していない可能性があります。ノードが不適格である最も一般的な理由は、ノードで仮想化がサポートされていないことです。サンドボックス化されたワークロードを不適格なノードで実行しようとすると、エラーが発生します。Node Feature Discovery (NFD) Operator と **NodeFeatureDiscovery** リソースを使用して、ノードの適格性を自動的に確認できます。



注記

適格であることがわかっている選択したワーカーノードのみに Kata ランタイムをインストールする場合は、選択したノードに **feature.node.kubernetes.io/runtime.kata=true** ラベルを適用し、**KataConfig** リソースで **checkNodeEligibility: true** を設定します。

または、Kata ランタイムをすべてのワーカーノードにインストールするには、**KataConfig** リソースで **checkNodeEligibility: false** を設定します。

どちらのシナリオでも、**NodeFeatureDiscovery** リソースを作成する必要はありません。ノードが Kata コンテナーを実行する資格があることが確実な場合にのみ、**feature.node.kubernetes.io/runtime.kata=true** ラベルを手動で適用する必要があります。

次の手順では、**feature.node.kubernetes.io/runtime.kata=true** ラベルをすべての適格なノードに適用し、ノードの適格性を確認するように **KataConfig** リソースを設定します。

前提条件

- OpenShift CLI (**oc**) をインストールしている。
- **cluster-admin** 権限を持つユーザーとしてログインすること。
- Node Feature Discovery (NFD) Operator をインストールします。

手順

1. **NodeFeatureDiscovery** リソースを作成して、Kata コンテナーの実行に適したノード機能を検出します。

   a. 次の YAML を **nfd.yaml** ファイルに保存します。

```
apiVersion: nfd.openshift.io/v1
kind: NodeFeatureDiscovery
metadata:
  name: nfd-kata
  namespace: openshift-nfd
spec:
  operand:
    image: quay.io/openshift/origin-node-feature-discovery:4.10
    imagePullPolicy: Always
    servicePort: 12000
  workerConfig:
    configData: |
      sources:
        custom:
          - name: "feature.node.kubernetes.io/runtime.kata"
            matchOn:
              - cpuId: ["SSE4", "VMX"]
                loadedKMod: ["kvm", "kvm_intel"]
              - cpuId: ["SSE4", "SVM"]
                loadedKMod: ["kvm", "kvm_amd"]
```

   b. **NodeFeatureDiscovery** カスタムリソース (CR) を作成します。

```
$ oc create -f nfd.yaml
```

   出力例

```
nodefeaturediscovery.nfd.openshift.io/nfd-kata created
```

   feature.node.kubernetes.io/runtime.kata=true ラベルが、資格のあるすべてのワーカーノードに適用されます。

2. **KataConfig** リソースで **checkNodeEligibility** フィールドを **true** に設定して、機能を有効にします。次に例を示します。

   a. 次の YAML を **kata-config.yaml** ファイルに保存します。

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
```

```
  name: example-kataconfig
spec:
  checkNodeEligibility: true
```

b. **KataConfig** CR を作成します。

```
$ oc create -f kata-config.yaml
```

出力例

```
kataconfig.kataconfiguration.openshift.io/example-kataconfig created
```

検証

- クラスター内の適格なノードに正しいラベルが適用されていることを確認します。

```
$ oc get nodes --selector='feature.node.kubernetes.io/runtime.kata=true'
```

出力例

```
NAME                    STATUS   ROLES    AGE     VERSION
compute-3.example.com   Ready    worker   4h38m   v1.25.0
compute-2.example.com   Ready    worker   4h35m   v1.25.0
```

関連情報

- Node Feature Discovery (NFD) Operator のインストールの詳細については、NFD のインストール を参照してください。

3.2. WEB コンソールを使用した OPENSHIFT サンドボックスコンテナーワークロードのデプロイ

Web コンソールから OpenShift サンドボックスコンテナーのワークロードをデプロイできます。まず、OpenShift サンドボックスコンテナー Operator をインストールしてから、**KataConfig** カスタムリソース (CR) を作成する必要があります。サンドボックスコンテナーにワークロードをデプロイする準備ができたら、ワークロード YAML ファイルに **kata** を **runtimeClassName** として手動で追加する必要があります。

3.2.1. Web コンソールを使用した OpenShift サンドボックスコンテナー Operator のインストール

OpenShift Container Platform Web コンソールから OpenShift サンドボックスコンテナー Operator をインストールできます。

前提条件

- OpenShift Container Platform 4.12 がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → OperatorHub** に移動します。
2. **Filter by keyword** フィールドに **OpenShift sandboxed containers** と入力します。
3. **OpenShift sandboxed containers** タイルを選択します。
4. Operator についての情報を確認してから、**Install** をクリックします。
5. **Install Operator** ページで以下を行います。
   a. 選択可能な **Update Channel** オプションの一覧から **stable-1.3** を選択します。
   b. **Installed Namespace** で **Operator recommend Namespace** が選択されていることを確認します。これにより、Operator が必須の **openshift-sandboxed-containers-operator** namespace にインストールされます。この namespace がまだ存在しない場合は、自動的に作成されます。

注記

OpenShift サンドボックスコンテナー Operator を **openshift-sandboxed-containers-operator** 以外の namespace にインストールしようとすると、インストールが失敗します。

   c. **Approval Strategy** で **Automatic** が選択されていることを確認します。**Automatic** がデフォルト値であり、新しい z-stream リリースが利用可能になると、OpenShift サンドボックスコンテナーへの自動更新が有効になります。
6. **Install** をクリックします。

これで、OpenShift サンドボックスコンテナー Operator がクラスターにインストールされました。

検証

1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. OpenShift サンドボックスコンテナー Operator が in オペレーターリストにリストされていることを確認します。

3.2.2. Web コンソールで KataConfig カスタムリソースを作成する

クラスターノードに **kata** を **RuntimeClass** としてインストールできるようにするには、1 つの **KataConfig** カスタムリソース (CR) を作成する必要があります。

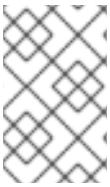
重要

KataConfig CR を作成すると、ワーカーノードが自動的に再起動します。再起動には 10 分から 60 分以上かかる場合があります。再起動時間を妨げる要因は次のとおりです。

- より多くのワーカーノードを持つ大規模な OpenShift Container Platform デプロイメント。
- BIOS および診断ユーティリティーのアクティベーション。
- SSD ではなくハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードへのデプロイメント。
- 遅い CPU とネットワーク。

前提条件

- クラスターに OpenShift Container Platform 4.12 をインストールしました。
- **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 サンドボックスコンテナー Operator をインストールしました。



注記

Kata は、デフォルトですべてのワーカーノードにインストールされます。特定のノードにのみ **kata** を **RuntimeClass** としてインストールする場合は、それらのノードにラベルを追加し、作成時に **KataConfig** CR でラベルを定義できます。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. オペレーターのリストから OpenShift サンドボックスコンテナーオペレーターを選択します。
3. **KataConfig** タブで、**Create KataConfig** をクリックします。
4. **Create KataConfig** ページで、次の詳細を入力します。
   - Name: **KataConfig** リソースの名前を入力します。デフォルトでは、名前は **example-kataconfig** として定義されています。
   - Labels (オプション): 関連する識別属性を **KataConfig** リソースに入力します。各ラベルはキーと値のペアを表します。
   - **checkNodeEligibility** (オプション): **kata** を **RuntimeClass** として実行するノードの適格性を、Node Feature Discovery Operator (NFD) を使用して検出するには、このチェックボックスを選択します。詳細は、「クラスターノードが OpenShift サンドボックスコンテナーを実行する資格があるかどうかを確認する」を参照してください。
   - **kataConfigPoolSelector**: デフォルトでは、**kata** はすべてのノードに **RuntimeClass** としてインストールされます。選択したノードにのみ **kata** を **RuntimeClass** としてインストールする場合は、**matchExpression** を追加する必要があります。
     a. **kataConfigPoolSelector** エリアを展開します。

b. **kataConfigPoolSelector** で、matchExpressions を展開します。これは、ラベルセレクターの要件のリストです。

c. **Add matchExpressions** をクリックします。

d. **key** フィールドに、セレクターの適用先のラベルキーを追加します。

e. **operator** フィールドに、ラベル値に対するキーの関係を追加します。有効な演算子は、**In**、**NotIn**、**Exists**、**DoesNotExist** です。

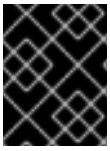
f. **values** エリアを展開し、**Add value** をクリックします。

g. **Value** フィールドで、**true** または **false** を **key** ラベル値として入力します。

- **logLevel**: **kata** を **RuntimeClass** として実行しているノードに対して、取得するログデータのレベルを定義します。詳細は、「OpenShift サンドボックスコンテナーデータの収集」を参照してください。

5. **Create** をクリックします。

新しい **KataConfig** CR が作成され、ワーカーノードに **kata** を **RuntimeClass** としてインストールし始めます。**kata** のインストールが完了し、ワーカーノードが再起動するのを待ってから、次の手順に進みます。



重要

OpenShift サンドボックスコンテナーは、Kata を主なランタイムとしてではなく、クラスターでセカンダリーオプションのランタイムとしてのみインストールします。

検証

1. **KataConfig** タブで、新しい **KataConfig** CR を選択します。
2. **KataConfig** ページで、**YAML** タブを選択します。
3. ステータスの **installationStatus** フィールドをモニターします。
   更新があるたびにメッセージが表示されます。**リロード** をクリックして、更新された **KataConfig** CR を表示します。

   Completed nodes の値がワーカーまたはラベル付けされたノードの数と等しくなると、インストールは完了です。ステータスには、インストールが完了したノードの一覧も含まれます。

3.2.3. Web コンソールを使用してサンドボックスコンテナーにワークロードをデプロイする

OpenShift サンドボックスコンテナーは、Kata を主なランタイムとしてではなく、クラスターでセカンダリーオプションのランタイムとしてインストールします。

Pod テンプレート化されたワークロードをサンドボックスコンテナーにデプロイするには、**kata** を **runtimeClassName** としてワークロード YAML ファイルに手動で追加する必要があります。

前提条件

- クラスターに OpenShift Container Platform 4.12 をインストールしました。
- **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 サンドボックスコンテナー Operator をインストールしました。
- **KataConfig** カスタムリソース (CR) を作成しました。

手順

1. Web コンソールの **Administrator** パースペクティブから、**Workloads** をデプロイメントし、作成するワークロードのタイプを選択します。
2. ワークロードページで、をクリックしてワークロードを作成します。
3. ワークロードの YAML ファイルで、コンテナーがリストされている **spec** フィールドに、**runtimeClassName: kata** を追加します。

   Pod の例

```
apiVersion: v1
kind: Pod
metadata:
  name: example
  labels:
    app: httpd
  namespace: openshift-sandboxed-containers-operator
spec:
  runtimeClassName: kata
  containers:
    - name: httpd
      image: 'image-registry.openshift-image-registry.svc:5000/openshift/httpd:latest'
      ports:
        - containerPort: 8080
```

   デプロイメントの例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example
  namespace: openshift-sandboxed-containers-operator
spec:
  selector:
    matchLabels:
      app: httpd
  replicas: 3
  template:
    metadata:
      labels:
        app: httpd
    spec:
      runtimeClassName: kata
      containers:
        - name: httpd
          image: >-
            image-registry.openshift-image-registry.svc:5000/openshift/httpd:latest
          ports:
            - containerPort: 8080
```

4. **Save** をクリックします。

OpenShift Container Platform はワークロードを作成し、スケジューリングを開始します。

3.3. CLI を使用した OPENSHIFT サンドボックスコンテナーワークロードのデプロイ

CLI を使用して、OpenShift サンドボックスコンテナーのワークロードをデプロイできます。まず、OpenShift サンドボックスコンテナー Operator をインストールしてから、**KataConfig** カスタムリソースを作成する必要があります。サンドボックスコンテナーにワークロードをデプロイする準備ができたら、ワークロード YAML ファイルに **runtimeClassName** として **kata** を追加する必要があります。

3.3.1. CLI を使用したサンドボックスコンテナー Operator のインストール

OpenShift Container Platform CLI から OpenShift サンドボックスコンテナー Operator をインストールできます。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift サンドボックスコンテナーカタログにサブスクライブしている。

注記

OpenShift サンドボックスコンテナーカタログにサブスクライブすると、**openshift-sandboxed-containers-operator** namespace の OpenShift サンドボックスコンテナー Operator にアクセスできるようになります。

手順

1. OpenShift サンドボックスコンテナー Operator の **Namespace** オブジェクトを作成します。
   a. 次のマニフェストを含む **Namespace** オブジェクト YAML ファイルを作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-sandboxed-containers-operator
```

   b. **Namespace** オブジェクトを作成します。

```
$ oc create -f Namespace.yaml
```

2. OpenShift サンドボックスコンテナー Operator の **OperatorGroup** オブジェクトを作成します。
   a. 次のマニフェストを含む **OperatorGroup** オブジェクト YAML ファイルを作成します。

```
apiVersion: operators.coreos.com/v1
```

```
kind: OperatorGroup
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  targetNamespaces:
  - openshift-sandboxed-containers-operator
```

b. **OperatorGroup** オブジェクトを作成します。

```
$ oc create -f OperatorGroup.yaml
```

3. **Subscription** オブジェクトを作成して、**Namespace** を OpenShift サンドボックスコンテナー Operator にサブスクライブします。

a. 次のマニフェストを含む **Subscription** オブジェクト YAML ファイルを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: "stable-1.3"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.3.2
```

b. **Subscription** オブジェクトを作成します。

```
$ oc create -f Subscription.yaml
```

これで、OpenShift サンドボックスコンテナー Operator がクラスターにインストールされました。

注記

上記のオブジェクトファイル名はすべて提案です。他の名前を使用してオブジェクト YAML ファイルを作成できます。

検証

- Operator が正常にインストールされていることを確認します。

```
$ oc get csv -n openshift-sandboxed-containers-operator
```

出力例

```
NAME                                DISPLAY                                  VERSION  REPLACES     PHASE
openshift-sandboxed-containers   openshift-sandboxed-containers-operator  1.3.2    1.3.1
Succeeded
```

関連情報

- CLI を使用した OperatorHub からのインストール

3.3.2. CLI を使用して KataConfig カスタムリソースを作成する

kata を **RuntimeClass** としてノードにインストールするには、1 つの **KataConfig** カスタムリソース (CR) を作成する必要があります。**KataConfig** CR を作成すると、OpenShift サンドボックス化されたコンテナー Operator がトリガーされ、以下が実行されます。

- QEMU および **kata-containers** など、必要な RHCOS 拡張を RHCOS ノードにインストールします。
- ランタイム CRI-O が正しい **kata** ランタイムハンドラーで設定されていることを確認します。
- デフォルト設定で **kata** という名前の **RuntimeClass** CR を作成します。これにより、ユーザーは、**RuntimeClassName** フィールドで CR を参照することにより、**kata** をランタイムとして使用するようにワークロードを設定できます。この CR は、ランタイムのリソースオーバーヘッドも指定します。

注記

Kata は、デフォルトですべてのワーカーノードにインストールされます。特定のノードにのみ **kata** を **RuntimeClass** としてインストールする場合は、それらのノードにラベルを追加し、作成時に **KataConfig** CR でラベルを定義できます。

前提条件

- クラスターに OpenShift Container Platform 4.12 をインストールしました。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift サンドボックスコンテナー Operator をインストールしました。

重要

KataConfig CR を作成すると、ワーカーノードが自動的に再起動します。再起動には 10 分から 60 分以上かかる場合があります。再起動時間を妨げる要因は次のとおりです。

- より多くのワーカーノードを持つ大規模な OpenShift Container Platform デプロイメント。
- BIOS および診断ユーティリティーのアクティベーション。
- SSD ではなくハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードへのデプロイメント。
- 遅い CPU とネットワーク。

手順

1. 次のマニフェストで YAML ファイルを作成します。
   -

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  checkNodeEligibility: false (1)
  logLevel: info
```

(1) **kata** を **RuntimeClass** として実行するノードの適格性を検出するには、`checkNodeEligibility` を **true** に設定します。詳細は、「クラスターノードが OpenShift サンドボックスコンテナーを実行する資格があるかどうかを確認する」を参照してください。

2. (オプション) 選択したノードにのみ **kata** を **RuntimeClass** としてインストールする場合は、マニフェストにラベルを含む YAML ファイルを作成します。

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  checkNodeEligibility: false
  logLevel: info
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' (1)
```

(1) **kataConfigPoolSelector** のラベルは単一値のみをサポートします。**nodeSelector** 構文はサポートされていません。

3. **KataConfig** リソースを作成します。

```
$ oc create -f <file name>.yaml
```

新しい **KataConfig** CR が作成され、ワーカーノードに **kata** を **RuntimeClass** としてインストールし始めます。**kata** のインストールが完了し、ワーカーノードが再起動するのを待ってから、次の手順に進みます。

重要

OpenShift サンドボックスコンテナーは、Kata を主なランタイムとしてではなく、クラスターでセカンダリーオプションのランタイムとしてのみインストールします。

検証

- インストールの進捗を監視します。

```
$ watch "oc describe kataconfig | sed -n /^Status:/,/^Events/p"
```

Is In Progressの値が **false** と表示されたら、インストールは完了です。

関連情報

- ノードでラベルを更新する方法について

3.3.3. CLI を使用してサンドボックスコンテナーにワークロードをデプロイする

OpenShift サンドボックスコンテナーは、Kata を主なランタイムとしてではなく、クラスターでセカンダリーオプションのランタイムとしてインストールします。

Pod テンプレート化されたワークロードをサンドボックスコンテナーにデプロイするには、ワークロード YAML ファイルに **runtimeClassName** として **kata** を追加する必要があります。

前提条件

- クラスターに OpenShift Container Platform 4.12 をインストールしました。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift サンドボックスコンテナー Operator をインストールしました。
- **KataConfig** カスタムリソース (CR) を作成しました。

手順

- 任意の Pod テンプレートオブジェクトに **runtimeClassName: kata** を追加します。
  - **Pod** オブジェクト
  - **ReplicaSet** オブジェクト
  - **ReplicationController** オブジェクト
  - **StatefulSet** オブジェクト
  - **Deployment** オブジェクト
  - **DeploymentConfig** オブジェクト

Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
 name: mypod
spec:
  runtimeClassName: kata
```

Deployment オブジェクトの例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mypod
  labels:
    app: mypod
```

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mypod
  template:
    metadata:
      labels:
        app: mypod
    spec:
      runtimeClassName: kata
      containers:
      - name: mypod
        image: myImage
```

OpenShift Container Platform はワークロードを作成し、スケジューリングを開始します。

検証

- Pod テンプレートオブジェクトの **runtimeClassName** フィールドを調べます。**runtimeClassName** が **kata** の場合、ワークロードは OpenShift サンドボックスコンテナーで実行されています。

3.4. 関連情報

- OpenShift サンドボックスコンテナー Operator は、制限されたネットワーク環境でサポートされます。詳細は、ネットワークが制限された環境での Operator Lifecycle Manager の使用 を参照してください。
- ネットワークが制限された環境で非接続クラスターを使用する場合に、OperatorHub にアクセスできるようにするには、Operator Lifecycle Manager でプロキシーサポート を設定する必要があります。プロキシーを使用すると、クラスターは OpenShift サンドボックスコンテナー Operator を取得できます。

第4章 OPENSHIFT サンドボックスコンテナーのモニタリング

OpenShift Container Platform Web コンソールを使用して、サンドボックス化されたワークロードおよびノードのヘルスステータスに関連するメトリクスを監視できます。

OpenShift サンドボックスコンテナーには、Web コンソールで使用できる事前設定済みのダッシュボードがあり、管理者は Prometheus を介して生のメトリックにアクセスしてクエリーを実行することもできます。

4.1. OPENSHIFT サンドボックスコンテナーのメトリクスについて

OpenShift サンドボックスコンテナーメトリックにより、管理者はサンドボックスコンテナーの実行状況を監視できます。これらのメトリクスは、Web コンソールのメトリクス UI でクエリーできます。

OpenShift サンドボックスコンテナーのメトリックは、次のカテゴリーで収集されます。

Kata エージェントの指標

カタエージェントメトリックには、サンドボックスコンテナーに埋め込まれた VM で実行されているカタエージェントプロセスに関する情報が表示されます。これらのメトリックには、**/proc/<pid>/io、stat、status** からのデータが含まれます。

Kata ゲスト OS メトリクス

Kata ゲスト OS メトリクスには、サンドボックスコンテナーで実行されているゲスト OS からのデータが表示されます。これらのメトリクスには、**/proc/[stats, diskstats, meminfo, vmstats]** および **/proc/net/dev** からのデータが含まれます。

ハイパーバイザーの指標

ハイパーバイザーメトリックには、サンドボックスコンテナーに埋め込まれた VM を実行しているハイパーバイザーに関するデータが表示されます。これらのメトリックには、主に **/proc/<pid>/[io, stat, status]** からのデータが含まれます。

Kata モニターのメトリクス

Kata モニターは、メトリックデータを収集し、Prometheus で利用できるようにするプロセスです。kata モニターメトリックには、kata-monitor プロセス自体のリソース使用状況に関する詳細情報が表示されます。これらのメトリクスには、Prometheus データコレクションからのカウンターも含まれます。

Kata containerd shim v2 メトリクス

Kata containerd shim v2 メトリクスには、kata shim プロセスに関する詳細情報が表示されます。これらのメトリクスには、**/proc/<pid>/[io, stat, status]** からのデータと詳細なリソース使用メトリクスが含まれます。

4.2. OPENSHIFT サンドボックスコンテナーのメトリクスの表示

Web コンソールの **Metrics** ページで、OpenShift サンドボックスコンテナーのメトリックにアクセスできます。

前提条件

- OpenShift Container Platform 4.12 がインストールされている。
- OpenShift サンドボックスコンテナーがインストールされている。
- **cluster-admin** ロールまたはすべてのプロジェクトの表示パーミッション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. Web コンソールの **Administrator** パースペクティブから、**Observe → Metrics** に移動します。
2. 入力フィールドに、監視するメトリクスのクエリーを入力します。以下に例を示します。
   kata 関連のメトリクスはすべて **kata** で始まります。**kata** と入力すると、使用可能なすべての kata メトリクスのリストが表示されます。

クエリーのメトリクスがページに視覚化されます。

関連情報

- メトリクスを表示するための PromQL クエリーの作成の詳細については、メトリクスのクエリー を参照してください。

4.3. OPENSHIFT サンドボックスコンテナーダッシュボードの表示

Web コンソールの **Dashboards** ページで、OpenShift サンドボックスコンテナーダッシュボードにアクセスできます。

前提条件

- OpenShift Container Platform 4.12 がインストールされている。
- OpenShift サンドボックスコンテナーがインストールされている。
- **cluster-admin** ロールまたはすべてのプロジェクトの表示パーミッション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. Web コンソールの **Administrator** パースペクティブから、**Observe → Dashboards** に移動します。
2. **Dashboard** ドロップダウンリストから、**Sandboxed Containers** ダッシュボードを選択します。
3. 必要に応じて、**Time Range** 一覧でグラフの時間範囲を選択します。
   - 事前定義済みの期間を選択します。
   - **時間範囲** 一覧で **カスタムの時間範囲** を選択して、カスタムの時間範囲を設定します。
     a. 表示するデータの日付と時刻の範囲を定義します。
     b. **Save** をクリックして、カスタムの時間範囲を保存します。
4. オプション: **Refresh Interval** を選択します。

ページにダッシュボードが表示され、Kata ゲスト OS カテゴリーの次のメトリックが表示されます。

実行中の仮想マシンの数
クラスターで実行されているサンドボックスコンテナーの総数を表示します。

CPU 使用率 (仮想マシンあたり)
個々のサンドボックスコンテナーの CPU 使用率を表示します。

メモリー使用量 (仮想マシンあたり)

個々のサンドボックスコンテナーのメモリー使用量を表示します。

特定の項目についての詳細情報を表示するには、ダッシュボードの各グラフにカーソルを合わせます。

4.4. 関連情報

- サポートのためのデータ収集について詳しくは、クラスターに関するデータの収集 を参照してください。

第5章 OPENSHIFT サンドボックスコンテナーのアンインストール

OpenShift Container Platform Web コンソールまたは OpenShift CLI (**oc**) のいずれかを使用して、OpenShift サンドボックス化コンテナーをアンインストールできます。両方の手順を以下で説明します。

5.1. WEB コンソールを使用した OPENSHIFT サンドボックスコンテナーのアンインストール

OpenShift Container Platform Web コンソールを使用して、関連する OpenShift サンドボックスコンテナーの Pod、リソース、および namespace を削除します。

5.1.1. Web コンソールを使用した OpenShift サンドボックスコンテナー Pod の削除

OpenShift サンドボックスコンテナーをアンインストールするには、最初に **kata** を **runtimeClass** として使用する実行中のすべての Pod を削除する必要があります。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **kata** を **runtimeClass** として使用する Pod のリストがあります。

手順

1. **Administrator** パースペクティブから、**Workloads → Pods** に移動します。
2. **Search by name** フィールドを使用して、削除する Pod を検索します。
3. Pod 名をクリックして開きます。
4. **Details** ページで、**Runtime class** に **kata** が表示されていることを確認します。
5. **Actions** メニューをクリックし、**Delete Pod** を選択します。
6. 確認ウィンドウで **Delete** をクリックします。

関連情報

OpenShift CLI から、**kata** を **runtimeClass** として使用する実行中の Pod のリストを取得できます。詳細については、OpenShift サンドボックスコンテナー Pod の削除 を参照してください。

5.1.2. Web コンソールを使用して KataConfig カスタムリソースを削除する

KataConfig カスタムリソース (CR) を削除すると、クラスターから **kata** ランタイムとその関連リソースが削除され、アンインストールされます。

重要

KataConfig CR を削除すると、ワーカーノードが自動的に再起動します。再起動には 10 分から 60 分以上かかる場合があります。再起動時間を妨げる要因は次のとおりです。

- より多くのワーカーノードを持つ大規模な OpenShift Container Platform デプロイメント。
- BIOS および診断ユーティリティーのアクティベーション。
- SSD ではなくハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードへのデプロイメント。
- 遅い CPU とネットワーク。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- **kata** を **runtimeClassName** として使用する実行中の Pod がない。

手順

1. **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. **Search by name** フィールドを使用して、OpenShift サンドボックスコンテナー Operator を検索します。
3. Operator をクリックして開き、**KataConfig** タブを選択します。
4. **KataConfig** リソースの**オプション** メニュー **をクリックし、KataConfig の削除** を選択します。
5. 確認ウィンドウで **Delete** をクリックします。

kata ランタイムとリソースがアンインストールされ、ワーカーノードが再起動されるまで待ってから、次の手順に進みます。

5.1.3. Web コンソールを使用した OpenShift サンドボックスコンテナー Operator のインストール

OpenShift サンドボックスコンテナーの削除 Operator は、その Operator のカタログサブスクリプション、Operator グループ、およびクラスターサービスバージョン (CSV) を削除します。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. **Search by name** フィールドを使用して、OpenShift サンドボックスコンテナー Operator を検索します。
3. Operator の **オプション** メニュー をクリックし、**Operator のアンインストール** を選択します。
4. 確認ウィンドウで **Uninstall** をクリックします。

5.1.4. Web コンソールを使用して OpenShift サンドボックスコンテナーの namespace を削除する

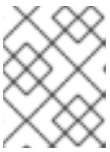
上記のコマンドを実行すると、インストールプロセス前の状態にクラスターが復元されます。**openshift-sandboxed-containers-operator** namespace を削除することで、Operator への namespace アクセスを取り消すことができるようになりました。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. **Administrator** パースペクティブから、**Administration → Namespaces** に移動します。
2. **Search by name** フィールドを使用して **openshift-sandboxed-containers-operator** namespace を検索します。
3. namespace の **オプション** メニュー をクリックし、**Delete Namespace** を選択します。



注記

Delete Namespace オプションが選択できない場合には、namespace を削除するパーミッションがありません。

4. **Delete Namespace** ペインで、**openshift-sandboxed-containers-operator** と入力し、**Delete** をクリックします。
5. **Delete** をクリックします。

5.1.5. Web コンソールを使用して KataConfig カスタムリソース定義を削除する

KataConfig カスタムリソース定義 (CRD) を使用すると、**KataConfig** CR を定義できます。アンインストールプロセスを完了するには、クラスターから **KataConfig** CRD を削除します。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- クラスターから **KataConfig** CR を削除しました。
- クラスターから OpenShift サンドボックスコンテナー Operator を削除しました。

手順

1. **Administrator** パースペクティブから、**Administration → CustomResourceDefinitions** に移動します。
2. **Search by name** フィールドを使用して **KataConfig** を検索します。
3. **KataConfig** CRD の **Options** メニュー  をクリックし、**Delete CustomResourceDefinition** を選択します。
4. 確認ウィンドウで **Delete** をクリックします。
5. **KataConfig** CRD がリストから消えるまで待ちます。これには数分の時間がかかる場合があります。

5.2. CLI を使用した OPENSHIFT サンドボックスコンテナーのアンインストール

OpenShift Container Platform コマンドラインインターフェイス (CLI) を使用して OpenShift サンドボックスコンテナーをアンインストールできます。以下の手順を記載順に実行してください。

5.2.1. CLI を使用した OpenShift サンドボックスコンテナー Pod の削除

OpenShift サンドボックスコンテナーをアンインストールするには、最初に **kata** を **runtimeClass** として使用する実行中のすべての Pod を削除する必要があります。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- コマンドライン JSON プロセッサー (**jq**) 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、**kata** を **runtimeClass** として使用する Pod を検索します。

```
$ oc get pods -A -o json | jq -r '.items[] | select(.spec.runtimeClassName ==
"kata").metadata.name'
```

2. 各 Pod を削除するには、次のコマンドを実行します。

```
$ oc delete pod <pod-name>
```

5.2.2. CLI を使用した KataConfig カスタムリソースの削除

kata ランタイムとその関連リソースすべて (CRI-O 設定や **RuntimeClass** など) をクラスターから削除およびアンインストールできます。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

重要

KataConfig CR を削除すると、ワーカーノードが自動的に再起動します。再起動には 10 分から 60 分以上かかる場合があります。再起動時間を妨げる要因は次のとおりです。

- より多くのワーカーノードを持つ大規模な OpenShift Container Platform デプロイメント。
- BIOS および診断ユーティリティーのアクティベーション。
- SSD ではなくハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードへのデプロイメント。
- 遅い CPU とネットワーク。

手順

1. 以下のコマンドを実行して **KataConfig** カスタムリソースを削除します。

```
$ oc delete kataconfig <KataConfig_CR_Name>
```

OpenShift サンドボックスコンテナー Operator は、クラスターでランタイムを有効化するために初期に作成されていたリソースをすべて削除します。

重要

削除中、CLI はすべてのワーカーノードが再起動するまで応答を停止します。プロセスが完了するまで待ってから、検証を実行するか、次の手順に進みます。

検証

- **KataConfig** カスタムリソースが削除されたことを確認するには、以下のコマンドを実行します。

```
$ oc get kataconfig <KataConfig_CR_Name>
```

出力例

```
No KataConfig instances exist
```

5.2.3. CLI を使用したサンドボックスコンテナー Operator のインストール

Operator サブスクリプション、Operator グループ、クラスターサービスバージョン (CSV)、および namespace を削除して、クラスターから OpenShift サンドボックスコンテナー Operator を削除します。

前提条件

- OpenShift Container Platform 4.10 がクラスターにインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- コマンドライン JSON プロセッサー (**jq**) をインストールしました。
- **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. 次のコマンドを実行して、サブスクリプションから OpenShift サンドボックスコンテナーのクラスターサービスバージョン (CSV) 名をフェッチします。

```
CSV_NAME=$(oc get csv -n openshift-sandboxed-containers-operator -o=custom-columns=:metadata.name)
```

2. 以下のコマンドを実行して、OpenShift サンドボックスコンテナー Operator サブスクリプションを Operator Lifecyle Manager (OLM) から削除します。

```
$ oc delete subscription sandboxed-containers-operator -n openshift-sandboxed-containers-operator
```

3. 以下のコマンドを実行して、OpenShift サンドボックスコンテナーの CSV 名を削除します。

```
$ oc delete csv ${CSV_NAME} -n openshift-sandboxed-containers-operator
```

4. 次のコマンドを実行して、OpenShift サンドボックスコンテナーの Operator グループ名を取得します。

```
$ OG_NAME=$(oc get operatorgroup -n openshift-sandboxed-containers-operator -o=jsonpath={..name})
```

5. 次のコマンドを実行して、OpenShift サンドボックスコンテナー Operator グループ名を削除します。

```
$ oc delete operatorgroup ${OG_NAME} -n openshift-sandboxed-containers-operator
```

6. 次のコマンドを実行して、OpenShift サンドボックスコンテナーの namespace を削除します。

```
$ oc delete namespace openshift-sandboxed-containers-operator
```

5.2.4. CLI を使用した KataConfig カスタムリソース定義の削除

KataConfig カスタムリソース定義 (CRD) を使用すると、**KataConfig** CR を定義できます。クラスターから **KataConfig** CRD を削除します。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- クラスターから **KataConfig** CR を削除しました。
- クラスターから OpenShift サンドボックスコンテナー Operator を削除しました。

手順

1. 次のコマンドを実行して、**KataConfig** CRD を削除します。

```
$ oc delete crd kataconfigs.kataconfiguration.openshift.io
```

検証

- **KataConfig** CRD が削除されたことを確認するには、次のコマンドを実行します。

```
$ oc get crd kataconfigs.kataconfiguration.openshift.io
```

出力例

```
Unknown CR KataConfig
```

第6章 OPENSHIFT サンドボックスコンテナーのアップグレード

OpenShift サンドボックスコンテナーコンポーネントのアップグレードは、次の 3 つの手順で設定されます。

- **Kata** ランタイムとその依存関係を更新するための OpenShift Container Platform のアップグレード。
- OpenShift サンドボックスコンテナー Operator をアップグレードして、Operator サブスクリプションを更新します。
- **KataConfig** カスタムリソース (CR) に手動でパッチを適用して、モニター Pod を更新します。

以下に示す 1 つの例外を除いて、OpenShift サンドボックスコンテナー Operator のアップグレードの前または後に OpenShift Container Platform をアップグレードできます。OpenShift サンドボックスコンテナー Operator をアップグレードした直後に、常に **KataConfig** パッチを適用します。

重要

OpenShift サンドボックスコンテナー 1.3 を使用して OpenShift Container Platform 4.11 にアップグレードする場合、推奨される順序は、最初に OpenShift サンドボックスコンテナーを 1.2 から 1.3 にアップグレードし、次に OpenShift Container Platform を 4.10 から 4.11 にアップグレードすることです。

6.1. OPENSHIFT サンドボックスコンテナーリソースのアップグレード

OpenShift サンドボックスコンテナーアーティファクトは、Red Hat Enterprise Linux CoreOS (RHCOS) 拡張機能を使用してクラスターにデプロイされます。

RHCOS 拡張 **サンドボックスコンテナー** には、Kata コンテナーランタイム、ハイパーバイザーの QEMU およびその他の依存関係などの Kata コンテナーの実行に必要なコンポーネントが含まれます。クラスターを OpenShift Container Platform の新しいリリースにアップグレードすることで、拡張機能をアップグレードします。

OpenShift Container Platform のアップグレードに関する詳細は、クラスターの更新 を参照してください。

6.2. OPENSHIFT サンドボックスコンテナー OPERATOR のアップグレード

Operator Lifecycle Manager (OLM) を使用して、OpenShift サンドボックスコンテナー Operator を手動で設定するか、または自動的にアップグレードできます。初期導入時に手動アップグレードか自動アップグレードかを選択することで、将来のアップグレードモードが決まります。手動アップグレードのコンテキストでは、Web コンソールに、クラスター管理者がインストールでき、利用可能な更新を表示します。

Operator Lifecycle Manager (OLM) での OpenShift サンドボックスコンテナー Operator のアップグレードの詳細については、インストール済み Operator の更新 を参照してください。

6.3. OPENSHIFT サンドボックスコンテナーモニター POD のアップグレード

OpenShift サンドボックスコンテナーをアップグレードした後、**KataConfig** CR でモニターイメージを更新して、モニター Pod をアップグレードする必要があります。それ以外の場合、モニター Pod は以前のバージョンのイメージを実行し続けます。

Web コンソールまたは CLI を使用して更新を実行できます。

6.3.1. Web コンソールを使用した監視 Pod のアップグレード

OpenShift Container Platform の **KataConfig** YAML ファイルには、モニターイメージのバージョン番号が含まれています。バージョン番号を正しいバージョンに更新します。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. OpenShift Container Platform の **Administrator** パースペクティブから、**Operators → Installed Operators** に移動します。
2. **OpenShift sandboxed containers Operator** を選択し、**KataConfig** タブに移動します。
3. **Search by name** フィールドを使用して、**KataConfig** リソースを検索します。**KataConfig** リソースのデフォルト名は **example-kataconfig** です。
4. **KataConfig** リソースを選択し、**KataConfig** タブに移動します。
5. **kataMonitorImage** のバージョン番号を変更します。

```
    checkNodeEligibility: false
    kataConfigPoolSelector: null
    kataMonitorImage: 'registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.3.0'
```

6. **Save** をクリックします。

6.3.2. CLI を使用した監視 Pod のアップグレード

KataConfig CR のモニターイメージに手動でパッチを適用して、モニター Pod を更新できます。

前提条件

- OpenShift Container Platform 4.12 をお使いのクラスターがインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- OpenShift Container Platform CLI で、以下のコマンドを実行します。

```
$ oc patch kataconfig <kataconfig_name> --type merge --patch
'{"spec":{"kataMonitorImage":"registry.redhat.io/openshift-sandboxed-containers/osc-monitor-
rhel8:1.3.0"}}'
```

<kataconfig_name>:: は、 **example-kataconfig** などの Kata 設定ファイルの名前を指定します。

第7章 OPENSHIFT サンドボックスコンテナーデータの収集

OpenShift サンドボックスコンテナーのトラブルシューティングを行う場合、サポートケースを開き、**must-gather** ツールを使用してデバッグ情報を提供できます。

クラスター管理者は、自分でログを確認して、より詳細なレベルのログを有効にすることもできます。

7.1. RED HAT サポート用の OPENSHIFT サンドボックスコンテナーデータの収集

サポートケースを作成する際、ご使用のクラスターについてのデバッグ情報を 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。

must-gather ツールを使用すると、仮想マシンおよび OpenShift Virtualization に関連する他のデータを含む、 OpenShift Container Platform クラスターについての診断情報を収集できます。

迅速なサポートのために、OpenShift Container Platform と OpenShift サンドボックスコンテナーの両方の診断情報を提供してください。

7.1.1. 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-rhel8:v4.12.0
  ```

- 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。

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 Pod で収集され、**must-gather.local** で始まる新規ディレクトリーに保存されます。このディレクトリーは、現行の作業ディレクトリーに作成されます。

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
...
```

must-gather を使用して OpenShift サンドボックスコンテナーデータを収集するには、OpenShift サンドボックスコンテナーイメージを指定する必要があります。

```
--image=registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:1.3.0
```

7.2. OPENSHIFT サンドボックスコンテナーのログデータについて

クラスターに関するログデータを収集すると、次の機能とオブジェクトが OpenShift サンドボックスコンテナーに関連付けられます。

- OpenShift サンドボックスコンテナーリソースに属するすべての名前空間とその子オブジェクト
- すべての OpenShift サンドボックスコンテナーのカスタムリソース定義 (CRD)

次の OpenShift サンドボックスコンテナーコンポーネントログは、**kata** ランタイムで実行されている Pod ごとに収集されます。

- Kata エージェントログ
- Kata ランタイムログ
- QEMU ログ
- 監査ログ
- CRI-O ログ

7.3. OPENSHIFT サンドボックスコンテナーのデバッグログを有効にする

クラスター管理者は、OpenShift サンドボックスコンテナーのより詳細なレベルのログを収集できます。OpenShift サンドボックスコンテナーを実行しているワーカーノードの CRI-O ランタイムで **log_level** を変更することにより、ロギングを強化します。

手順

1. 次のマニフェストを使用して、**ContainerRuntimeConfig** CR の YAML ファイルを作成します。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: ContainerRuntimeConfig
metadata:
```

```
 name: crio-debug
spec:
 machineConfigPoolSelector:
  matchLabels:
    pools.operator.machineconfiguration.openshift.io/worker: " 1
 containerRuntimeConfig:
    logLevel: debug
```

1 変更する必要のあるマシン設定プールのラベルを指定します。

2. **ContainerRuntimeConfig** CR を作成します。

```
$ oc create -f ctrcfg.yaml
```

注記

上記のファイル名は提案です。別の名前を使用してこのファイルを作成できます。

3. CR が作成されたことを確認します。

```
$ oc get ctrcfg
```

出力例

```
NAME          AGE
crio-debug    3m19s
```

検証

1. すべてのワーカーノードの **UPDATED** フィールドが **True** と表示されるまで、マシン設定プールをモニターします。

```
$ oc get mcp worker
```

出力例

```
NAME    CONFIG              UPDATED  UPDATING  DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT
AGE
worker  rendered-worker-169  False    True      False     3          1                  1                  0
9h
```

2. CRI-O で **log_level** が更新されたことを確認します。

   a. マシン設定プールのノードに対して **oc debug** セッションを開き、**chroot /host** を実行します。

```
$ oc debug node/<node_name>
```

```
sh-4.4# chroot /host
```

b. **crio.conf** ファイルの変更を確認します。

```
sh-4.4# crio config | egrep 'log_level
```

出力例

```
log_level = "debug"
```

7.3.1. OpenShift サンドボックスコンテナーのデバッグログの表示

クラスター管理者は、OpenShift サンドボックスコンテナーの強化されたデバッグログを使用して、問題のトラブルシューティングを行うことができます。各ノードのログは、ノードジャーナルに出力されます。

次の OpenShift サンドボックスコンテナーコンポーネントのログを確認できます。

- Kata エージェント
- Kata ランタイム (**containerd-shim-kata-v2**)
- virtiofsd

QEMU のログはノードジャーナルに出力されません。ただし、QEMU の障害はランタイムに報告され、QEMU ゲストのコンソールがノードジャーナルに出力されます。これらのログは、Kata エージェントログと一緒に表示できます。

前提条件

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

手順

- Kata エージェントのログとゲストコンソールのログを確認するには、次のコマンドを実行します。

```
$ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t kata -g “reading guest console”
```

- kata ランタイムログを確認するには、次を実行します。

```
$ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t kata
```

- virtiofsd ログを確認するには、次を実行します。

```
$ oc debug node/<nodename> -- journalctl -D /host/var/log/journal -t virtiofsd
```

7.4. 関連情報

- サポートのためのデータ収集について詳しくは、クラスターに関するデータの収集 を参照してください。