OpenShift Container Platform 4.9

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: 2023-05-23

OpenShift Container Platform 4.9 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} 1.1 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 4.9 とともに OpenShift サンドボックスコンテナー 1.1 の開発を追跡します。

この製品は現在テクノロジープレビューとして提供されています。OpenShift サンドボックスコンテナーは、実稼働環境での使用を目的としていません。詳細は、テクノロジープレビューの機能に関する Red Hat カスタマーポータルの サポート範囲 を参照してください。

1.2. 新機能および機能拡張

1.2.1. FIPS 互換性

OpenShift サンドボックスコンテナーでは、FIPS モードが自動的に有効になっています。FIPS モードでインストールされた OpenShift Container Platform クラスターにデプロイされた OpenShift サンドボックスコンテナーは、クラスターの FIPS サポートをテイントしません。詳細は、コンプライアンスとリスク管理について を参照してください。

1.2.2. must-gather でのリソース収集

OpenShift サンドボックスコンテナー Operator には、must-gather イメージが含まれるようになり、診断目的で、この Operator および基盤となるランタイムコンポーネントに固有のカスタムリソースとログファイルを収集できます。詳細は、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 を参照してください。

1.2.3. 非接続環境

非接続環境に OpenShift サンドボックスコンテナー Operator をインストールできるようになりました。詳細は、OpenShift サンドボックスコンテナーワークロードをデプロイするための追加リソース を参照してください。

1.3. バグ修正

- 以前のリリースでは、OpenShift サンドボックス化されたコンテナー上で Fedora を実行する場合には、デフォルトで OpenShift Container Platform がコンテナーに割り当てていないファイルのアクセスパーミッションを、パッケージによっては変更する必要がありました。今回のリリースでは、これらの権限がデフォルトで付与されています。(BZ#1915377)

- 以前は、OpenShift Container Platform Web コンソールで **kataConfgPoolSelector** に値を追加すると、空の値で **Scheduling.nodeSelector** に入力されました。その結果、値が **kata** の **RuntimeClass** オブジェクトを使用する Pod は、Kata Containers ランタイムがインストールされていないノードにスケジュールされる可能性がありました。このリリースでは、**kataConfigPoolSelector** で定義されているものと同じラベルが付いたノードのみが Kata Containers ランタイムをインストールします。(BZ#2019384)

- 以前のリリースには、Operator Hub の OpenShift サンドボックスコンテナー Operator の詳細ページにフィールドがありませんでした。今回のリリースでは、これらのフィールドが追加されています。(BZ#2019383)

- 以前のリリースでは、複数の **KataConfig** カスタムリソースを作成すると、OpenShift Container Platform Web コンソールから複数のカスタムリソース作成に失敗した旨を通知する

エラーなしに失敗していました。今回のリリースでは、ユーザーが複数のカスタムリソースを作成しようとするとエラーが発生します。(BZ#2019381)

- 以前のリリースでは、OpenShift Container Platform Web コンソールの Operator Hub に Operator のアイコンが表示されない場合がありました。今回のリリースでは、アイコンは常に表示されます。(BZ#9019380)

1.4. 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に、OpenShift Container Platform クラスターの **hostPath** ボリュームからマウントされているファイルまたはディレクトリーへのアクセスを SELinux が拒否することがありました。特権サンドボックスコンテナーは SELinux チェックを無効にしないため、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、このように拒否される可能性があります。
ホストで SELinux ポリシーに準拠すると、ホストファイルシステムとサンドボックスのワークロードを完全に分離して、**virtiofsd** または QEMU で発生する可能性のあるセキュリティーの脆弱性に対してより強力に保護します。

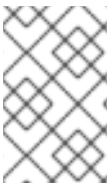
  マウントされたファイルまたはディレクトリーにホスト上の特定の SELinux 要件がない場合は、代わりにローカル永続ボリュームを使用できます。ファイルは、コンテナーランタイムの SELinux ポリシーに従って、自動的に **container_file_t** に再ラベル付けされます。詳細は ローカルボリュームを使用した永続ストレージ を参照してください。

  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 がこれらの特定のラベルにアクセスできるようにします。(BZ#1904609)

1.5. エラータの非同期更新

OpenShift サンドボックスコンテナー 4.9 のセキュリティー、バグ修正、および拡張機能の更新は、Red Hat Network を通じて非同期エラータとして発表されます。OpenShift Container Platform 4.9 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 OpenShift sandboxed containers 1.1.0 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。

1.5.1. RHEA-2021:3941 - OpenShift サンドボックスコンテナー 1.1.0 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: 2021-10-21

OpenShift サンドボックスコンテナーリリース 1.1.0 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一覧は、RHEA-2021:3941 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。これは、以下のタスクを実行するユーザーに特に便利です。

- 特権または信頼できないワークロードを実行する。
- 各ワークロードのカーネルを確実に分離する。
- テナント全体で同じワークロードを共有する。
- 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。
- 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。

OpenShift サンドボックスコンテナーには、さまざまなユースケースに対応するために実行するワークロードのタイプから選択する機能も含まれます。

OpenShift サンドボックスコンテナー Operator を使用して、インストール、削除、更新、ステータスの監視などのタスクを実行できます。

サンドボックスコンテナーは、ベアメタルでのみサポートされます。

Red Hat Enterprise Linux CoreOS (RHCOS) は、OpenShift サンドボックスコンテナー 1.0.0 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。

2.1. 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。これは、サンドボックスコンテナーソフトウェアおよびステータスの監視などの操作を処理します。

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。

RHCOS 拡張機能

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。

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。

2.2. 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ています。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. 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。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

2.4. 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。

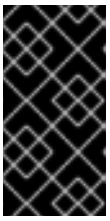
第3章 OPENSHIFT サンドボックスコンテナーワークロードのデプロイ

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) ワーカーを備えた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。



重要

  - 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。

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。

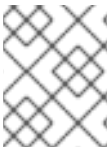
デフォルトのメモリーサイズ

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	Value
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

3.2. 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 4.9 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。

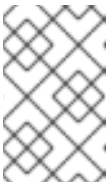
手順

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. 利用可能な **Update Channel** オプションから **preview-1.1** 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

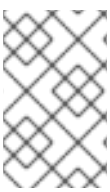
1. Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. OpenShift サンドボックスコンテナー Operator が in オペレーターリストにリストされていることを確認します。

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。

前提条件

- クラスターに OpenShift Container Platform 4.9 を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。



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、**YAML view** 経由で **KataConfig** CR を設定することを選択します。
5. 次のマニフェストをコピーして **YAML view** に貼り付けます。

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
```

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めます。

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  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。

6. **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。

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 4.9 を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

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

- クラスターに OpenShift Container Platform 4.9 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。

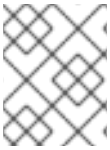
a. 次のマニフェストを含む **Subscription** オブジェクト YAML ファイルを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: "preview-1.1"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.1.0
```

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  1.1.0    1.0.2
Succeeded
```

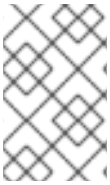
関連情報

- 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 4.9 を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 Operator をインストールしました。

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
```

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:
  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。

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 4.9 を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 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、制限されたネットワークでの 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切断されたクラスターを使用する場合に、Red Hat が提供する 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アンインストール

4.1. 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 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。

4.1.1. OpenShift サンドボックスコンテナーリソースの削除

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、まず OpenShift サンドボックスコンテナーカスタムリソース **KataConfig** を削除する必要があります。これにより、**kata** ランタイムと関連リソースがクラスターから削除され、アンインストールされます。

前提条件

- クラスターに OpenShift Container Platform 4.9 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。
  - 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。
  - 以下のコマンドを実行して、**kata** を **runtimeClassName** として使用する Pod が実行されていないことを確認します。

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

手順

1. **kata** の値で **runtimeClassName** を使用するすべての Pod を削除します。
2. OpenShift Container Platform Web コンソールから、**Projects** 一覧より **openshift-sandboxed-containers** を選択します。
3. **Operators → Installed Operators** ページに移動します。
4. **OpenShift sandboxed containers** をクリックします。
5. **OpenShift sandboxed containers Operator** タブをクリックします。
6. **Operator Details** のスクロールダウン一覧をクリックし、**Delete KataConfig** をクリックします。
7. 確認ウィンドウで **Delete** をクリックします。

4.1.1.1. Web コンソールを使用した namespace の削除

OpenShift Container Platform Web コンソールを使用して namespace を削除できます。

前提条件

- クラスターに OpenShift Container Platform 4.9 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. **Administration → Namespaces** に移動します。
2. namespace の一覧で削除する **openshift-sandboxed-containers-operator** namespace を見つけます。
3. namespace の一覧の右端で、**Options** メニュー から **Delete Namespace** を選択します。
4. **Delete Namespace** ペインが表示されたら、フィールドに **openshift-sandboxed-containers-operator** を入力します。

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。

5. **Delete** をクリックします。

4.1.2. OpenShift サンドボックスコンテナー Operator の削除

カタログサブスクリプションを削除し、Operator への namespace アクセスを取り消すことで、OpenShift でサンドボックス化したコンテナー Operator を削除できます。

前提条件

- クラスターに OpenShift Container Platform 4.9 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. **Operators → OperatorHub** ページに移動します。
2. **OpenShift sandboxed containers** 検索して、Operator を選択します。
3. **Uninstall** をクリックします。
4. **openshift-sandboxed-containers-operator** namespace を削除します。

4.2. CLI からの KATA ランタイム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。

4.2.1. OpenShift サンドボックスコンテナーリソースの削除

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

- クラスターに OpenShift Container Platform 4.9 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

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>
```

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

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

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。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 が削除できるようになりました。

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
```

- **KataConfig** カスタムリソース定義が削除されていることを確認するには、以下のコマンドを実行します。

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

出力例

```
Unknown CR KataConfig
```

4.2.2. OpenShift サンドボックスコンテナー Operator の削除

OpenShift サンドボックスコンテナー Operator をクラスターから削除できます。

前提条件

- クラスターに OpenShift Container Platform 4.9 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

1. 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 openshift-sandboxed-containers-subscription -n openshift-sandboxed-containers-operator
```

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のクラスターサービスバージョン (CSV) 名を環境変数として設定します。

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

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
```

第5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 Operator および OpenShift サンドボックスコンテナーのアーティファクトをアップグレードして、OpenShift サンドボックスコンテナーのコンポーネントをアップグレードできます。

5.1. 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。最初のデプロイメント時に手動または自動アップグレードを選択できます。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。

追加リソース

- インストールされた Operator のアップグレード

5.2. OPENSHIFT サンドボックスコンテナーアーティファクトをアップグレードします。

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。この拡張機能は、クラスターを OpenShift Container Platform の新規リリースにアップグレードする時に、アップグレードされます。

第6章 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。

6.1. 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:v4.9.0
  ```

- 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
...
```

6.2. OPENSHIFT サンドボックスコンテナーデータの収集について

oc adm must-gather CLI コマンドを使用して、クラスターに関する情報を収集できます。次の機能とオブジェクトは、OpenShift サンドボックスコンテナーに関連付けられています。

- 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)

oc adm must-gather CLI コマンドは、次のコンポーネントログを収集します。

- QEMU ログ
- 監査ログ
- OpenShift サンドボックスコンテナーのログ
- CRI-O ログ

これらの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 が最低でも 1 つあれば、収集されます。

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-tech-preview/osc-must-gather-rhel8:1.1.0
```

6.3. 関連情報

- サポート向けのデータ収集に関する詳細は クラスターに関するデータの収集 を参照してください。