OpenShift Sandboxed Containers 1.5

OpenShift Sandboxed Containers ユーザーガイド

OpenShift Container Platform の場合

Last Updated: 2024-04-06

OpenShift Sandboxed Containers 1.5 OpenShift Sandboxed Containers ユーザーガイド

OpenShift Container Platform の場合

法律上の通知

概要

OpenShift Sandboxed Containers operator とその使用法に関する情報

目次

はじめに

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。この取り組みは膨大な作業を要するため、これらの変更による更新は可能な範囲で段階的に行われます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Jira で **Create Issue** フォームを送信することで、フィードバックを提供したり、エラーを報告したりできます。Jira の課題は、Red Hat Hybrid Cloud Infrastructure Jira プロジェクトに作成されるため、フィードバックの進行状況を追跡できます。

1. Jira にログインしていることを確認してください。Jira アカウントをお持ちでない場合は、アカウントを作成してフィードバックを送信してください。
2. **Create issue** をクリックします。
    a. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
3. **Create** をクリックします。

Red Hat ドキュメントに関するご意見やご感想をお寄せください。

第1章 OPENSHIFT SANDBOXED CONTAINERS について

Red Hat OpenShift Container Platform の OpenShift sandboxed containers のサポートは、追加のオプションのランタイムとして Kata コンテナーを実行するための組み込みサポートを提供します。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 Sandboxed Containers (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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers でより堅牢で安全な方法で管理されます。OpenShift Sandboxed Containers 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。

1.1. OPENSHIFT SANDBOXED CONTAINERS がサポートするプラットフォーム

OpenShift Sandboxed Containers は、ベアメタルサーバーまたは Amazon Web Services (AWS) ベアメタルインスタンスに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 Sandboxed Containers 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 sandboxed containers 1.5 は、サポートされている Red Hat OpenShift Container Platform の全バージョンと互換性があります。詳細は、Red Hat OpenShift Container Platform ライフサイクルポリシー を参照してください。

1.2. OPENSHIFT SANDBOXED CONTAINERS 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 Sandboxed Containers 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化を図ります。

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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers Operator は、クラスター上でSandboxed Containers のライフサイクルを管理してタスクを実行します。OpenShift Sandboxed Containers Operator を使用して、Sandboxed Containers 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Sandboxed Containers 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Sandboxed Containers 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。

ピア Pod

OpenShift Sandboxed Containers のピア Pod は、標準 Pod の概念を拡張します。ピア Pod 内のワーカーノード自体に仮想マシンが作成される標準の Sandboxed Containers とは異なり、サポートされているハイパーバイザーまたはクラウドプロバイダー API を使用して、リモートハイパーバイザー経由で仮想マシンが作成されます。ピア Pod はワーカーノード上で通常の Pod として機能し、対応する VM が別の場所で実行されます。VM のリモートの場所はユーザーに対して透過的であり、Pod 仕様のランタイムクラスによって指定されます。ピア Pod 設計により、ネストされた仮想化の必要性が回避されます。

1.3. OPENSHIFT SANDBOXED CONTAINERS のワークロード管理

OpenShift Sandboxed Containers 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。

1.3.1. OpenShift Sandboxed Containers のビルディングブロック

OpenShift Sandboxed Containers Operator は、Kata Containers 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 Sandboxed Containers 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。

1.3.2. RHCOS 拡張機能

OpenShift Sandboxed Containers Operator は Red Hat Enterprise Linux CoreOS (RHCOS) 拡張機能の概念に基づいています。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ためのメカニズムです。OpenShift Sandboxed Containers Operator はこのメカニズムを使用して、Sandboxed Containers をクラスターにデプロイします。

Sandboxed Containers 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

1.3.3. OpenShift Virtualization および OpenShift sandboxed containers

OpenShift Virtualization を使用してクラスターで OpenShift Sandboxed Containers を使用できます。

OpenShift Virtualization と OpenShift Sandboxed Containers 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

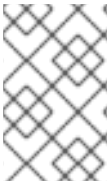
1.4. コンプライアンスおよびリスク管理について

OpenShift Container Platform は FIPS 用に設計されています。FIPS モードでブートされた Red Hat Enterprise Linux (RHEL) または Red Hat Enterprise Linux CoreOS (RHCOS) を実行する場合、OpenShift Container Platform コアコンポーネントは、**x86_64**、**ppc64le**、および **s390x** アーキテクチャーのみで、FIPS 140-2/140-3 検証のために NIST に提出された RHEL 暗号化ライブラリーを使用します。

NIST 検証プログラムの詳細は、暗号化モジュール検証プログラム を参照してください。検証のために提出された RHEL 暗号化ライブラリーの個別バージョンの最新の NIST ステータスについては、政府の標準規格 を参照してください。

OpenShift Sandboxed Containers 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 Sandboxed Containers 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。



注記

OpenShift Sandboxed Containers の FIPS コンプライアンスは、**kata** ランタイムクラスにのみ適用されます。新しいピア Pod ランタイムクラス **kata-remote-cc** はまだ完全にはサポートされておらず、FIPS コンプライアンスについてはテストされていません。

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。

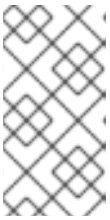
第2章 OPENSHIFT SANDBOXED CONTAINERS ワークロードのデプロイ

Web コンソールまたは OpenShift CLI (**oc**) のいずれかを使用して OpenShift Sandboxed Containers 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。

2.1. 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、またはアシステッド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 Sandboxed Containers 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 Sandboxed Containers 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。

2.1.1. 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デフォルト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。

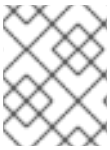
デフォルトのメモリーサイズ

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ック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ック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

2.1.2. クラスターノードが OpenShift Sandboxed Containers を実行する資格があるかどうかの確認

OpenShift Sandboxed Containers を実行する前に、クラスター内のノードが Kata Containers を実行する資格があるかどうかを確認してください。クラスターノードによっては、Sandboxed Containers 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**) がインストールされている。
- **cluster-admin** 権限を持つユーザーとしてログインしている。
- 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は、NFD のインストールを 参照してください。

2.2. WEB コンソールを使用した OPENSHIFT SANDBOXED CONTAINERS ワークロードのデプロイ

Web コンソールから OpenShift Sandboxed Containers のワークロードをデプロイできます。まず、OpenShift Sandboxed Containers 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。Sandboxed Containers 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。

2.2.1. Web コンソールを使用した OpenShift Sandboxed Containers Operator のインストール

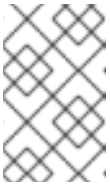
OpenShift Container Platform Web コンソールから OpenShift サンドボックスコンテナー Operator をインストールできます。

前提条件

- OpenShift Container Platform 4.15 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** 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 Sandboxed Containers Operator を **openshift-sandboxed-containers-operator** 以外の namespace にインストールしようとすると、インストールに失敗します。

   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 Sandboxed Containers への自動更新が有効になります。
6. **Install** をクリックします。

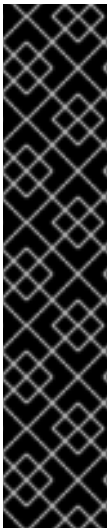
これで、OpenShift Sandboxed Containers Operator がクラスターにインストールされました。

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 Sandboxed Containers Operator がインストール済みの Operator リストに表示されていることを確認します。

2.2.2. Web コンソールでの KataConfig カスタムリソースの作成

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。



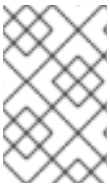
重要

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が有効である。
- SSD ではなくハードディスクドライブにデプロイしている。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイしている。
- CPU とネットワークが遅い。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator をインストールしている。



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. Operator のリストから OpenShift Sandboxed Containers Operator 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、ピア Pod には該当しない): **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 Sandboxed Containers 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。
   - **EnablePeerPods** (ピア Pod の場合): ピア Pod を有効にし、パブリッククラウド環境で OpenShift Sandboxed Containers を使用するには、このチェックボックスをオンにします。

- **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 Sandboxed Containers 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 Sandboxed Containers は、**kata** をプライマリーランタイムとしてではなく、クラスター上の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. status フィールドを監視します。
   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。

   status フィールドには conditions および kataNodes サブフィールドがあります。kataNodes サブフィールドはノード一覧のコレクションです。各ノードは、**kata** インストールの特定状態のノードを一覧表示します。

   kataNodes の下のすべてのワーカーが **installs** としてリストされ、理由を指定せずに **InProgress** の条件が **False** の場合は、**kata** がクラスターにインストールされていることを示します。詳細は、「移行のインストールとアンインストール」を参照してください。

2.2.3. Web コンソールを使用した Sandboxed Containers へのワークロードのデプロイ

OpenShift Sandboxed Containers は、Kata をプライマリー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 Sandboxed Containers 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 4.15 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 Sandboxed Containers Operator をインストールしている。
- **KataConfig** カスタムリソース (CR) を作成している。

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 オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
spec:
  runtimeClassName: kata
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

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。

2.3. 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

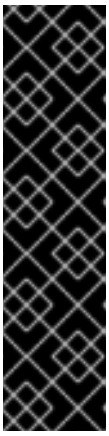
CLI を使用して、OpenShift Sandboxed Containers のワークロードをデプロイできます。まず、OpenShift Sandboxed Containers Operator をインストールしてから、**KataConfig** カスタムリソースを作成する必要があります。Sandboxed Containers にワークロードをデプロイする準備ができたら、ワークロード YAML ファイルに **runtimeClassName** として **kata** を追加する必要があります。

2.3.1. CLI を使用したSandboxed Containers 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 4.15 がインストールされている。
- IBM Z または IBM® LinuxONE へのインストールには、OpenShift Container Platform 4.14 以降がインストールされている。



重要

IBM Z でのピア Pod を使用した OpenShift Sandboxed Containers ワークロードのデプロイは、テクノロジープレビューとしてのみ提供されています。テクノロジープレビュー機能は、Red Hat 製品サポート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場合があります。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。

- 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 Sandboxed Containers カタログにサブスクライブしている。

注記

OpenShift Sandboxed Containers カタログにサブスクライブすると、**openshift-sandboxed-containers-operator** namespace の OpenShift Sandboxed Containers Operator にアクセスできるようになります。

手順

1. OpenShift Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers Operator にサブスクライブします。

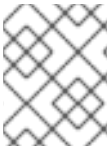
a. 次のマニフェストを含む **Subscription** オブジェクト YAML ファイルを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: stable
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.5.2
```

b. **Subscription** オブジェクトを作成します。

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

これで、OpenShift Sandboxed Containers 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.v1.5.2   openshift-sandboxed-containers-operator  1.5.2    1.5.1
Succeeded
```

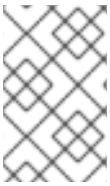
関連情報

- CLI を使用した OperatorHub からのインストール

2.3.2. CLI を使用した KataConfig カスタムリソースの作成

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 Sandboxed Containers 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。
- CRI-O ランタイムが正しいランタイムハンドラーで設定されていることを確認してください。
- 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。**kata** を特定のノードにのみ **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、**KataConfig** CR の作成時にそのラベルを定義できます。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator をインストールしている。

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が有効である。
- SSD ではなくハードディスクドライブにデプロイしている。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイしている。
- CPU とネットワークが遅い。

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 Sandboxed Containers 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 cluster-kataconfig.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 Sandboxed Containers は、**kata** をプライマリーランタイムとしてではなく、クラスター上の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"
```

  kataNodes の下のすべてのワーカーが **installs** としてリストされ、理由を指定せずに **InProgress** の条件が **False** の場合は、**kata** がクラスターにインストールされていることを示します。詳細は、「移行のインストールとアンインストール」を参照してください。

関連情報

- ノードでラベルを更新する方法について

2.3.3. CLI を使用した Sandboxed Containers へのワークロードのデプロイ

OpenShift Sandboxed Containers は、Kata をプライマリー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 Sandboxed Containers 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 4.15 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 Sandboxed Containers Operator をインストールしている。
- **KataConfig** カスタムリソース (CR) を作成している。

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: hello-openshift
  labels:
    app: hello-openshift
spec:
  runtimeClassName: kata
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsNonRoot: true
        runAsUser: 1001
        capabilities:
          drop:
            - ALL
        seccompProfile:
          type: RuntimeDefault
```

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 Sandboxed Containers で実行されています。

2.4. インストールおよびアンインストールの移行

次の表は、2 つのワーカーノードを持つクラスターのインストールとアンインストールにおける主な状態遷移を示しています。

表2.1 インストールの状態遷移

Status	説明

Status	説明
Initial installation **KataConfig** インスタンスが作成されて両方のワーカーでインストールが開始されると、ステータスは 1 ~ 2 秒間、このように表示されます。	conditions: message: Performing initial installation of kata on cluster reason: Installing status: 'True' type: InProgress kataNodes: nodeCount: 0 readyNodeCount: 0
インストール 数秒以内にステータスが変わります。	kataNodes: nodeCount: 2 readyNodeCount: 0 waitingToInstall: - worker-0 - worker-1
Installing (Worker-1 インストール開始) 短期間、ステータスが変化し、一方のノードが **kata** のインストールを開始し、他方のノードが待機状態であることを示します。これは、常に 1 つのノードだけが使用不能になる可能性があるためです。両方のノードが最終的に **kata** を受信するため、**nodeCount** は 2 のままですが、どちらのノードもまだその状態に達していないため、**readyNodeCount** は現在 0 です。	kataNodes: installing: - worker-1 nodeCount: 2 readyNodeCount: 0 waitingToInstall: - worker-0
Installing (Worker-1 がインストールされ、Worker-0 のインストールが開始されました) しばらくすると、**worker-1** がインストールを完了し、ステータスを変更します。**readyNodeCount** が 1 に更新され、**worker-1** が **kata** ワークロードを実行する準備ができたことを示します。kata runtimeClass が作成されるまでは、**kata** ワークロードをスケジュールして実行することはできません。インストールプロセスの最後にのみ、**kata runtimeClass** が作成されます。	kataNodes: installed: - worker-1 installing: - worker-0 nodeCount: 2 readyNodeCount: 1

Status	説明

Installed
 	
インストールされると、両方のワーカーがインストール済みとしてリストされ、理由を指定せずに **InProgress** 条件が **False** に移行し、クラスターへの **kata** のインストールが成功したことを示します。
	conditions:
	  message: ""
	  reason: ""
	  status: 'False'
	  type: InProgress
	kataNodes:
	  installed:
	  - worker-0
	  - worker-1
	  nodeCount: 2
	  readyNodeCount: 2

表2.2 アンインストールの移行

Status	説明

Initial uninstall
 	
kata が両方のワーカーにインストールされている場合、**KataConfig** を削除してクラスターから **kata** を削除すると、インストールプロセスと同様に、両方のワーカーが数秒間一時的に待機状態になります。
	conditions:
	  message: Removing kata from cluster
	  reason: Uninstalling
	  status: 'True'
	  type: InProgress
	kataNodes:
	  nodeCount: 0
	  readyNodeCount: 0
	  waitingToUninstall:
	  - worker-0
	  - worker-1

Uninstalling
 	
しばらくすると、ワーカーがアンインストールを開始します。
	kataNodes:
	  nodeCount: 0
	  readyNodeCount: 0
	  uninstalling:
	  - worker-1
	  waitingToUninstall:
	  - worker-0

Uninstalling
 	
worker-1 が完了すると、worker-0 はアンインストールを開始します。
	kataNodes:
	  nodeCount: 0
	  readyNodeCount: 0
	  uninstalling:
	  - worker-0

注記

reason フィールドでは、次のことも報告できます。

- **Failed**: これは、ノードが移行を完了できない場合に報告されます。**status** は **True** と報告し、**message** は **Node <node_name> Degraded: <error_message_from_the_node>** です。
- **BlockedByExistingKataPods** : これは、**kata** のアンインストール中に **kata** ランタイムを使用するクラスター上で実行している Pod がある場合に報告されます。**status** フィールドは **False** で、**message** は **Existing pods using "kata" RuntimeClass found.Please delete the pods manually for KataConfig deletion to proceed** です。クラスターコントロールプレーンとの通信が失敗した場合は、**Failed to list kata pods: <error_message>** のような技術的なエラーメッセージが報告される場合もあります。

2.5. 関連情報

- OpenShift Sandboxed Containers 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。
- 制限されたネットワーク上で切断されたクラスターを使用する場合、OperatorHub にアクセスするには 、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 Sandboxed Containers Operator を取得できます。

第3章 ピア POD を使用した OPENSHIFT SANDBOXED CONTAINERS 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 Sandboxed Containers 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 Sandboxed Containers をインストールしてピア Pod を有効にする前に、次の要件を満たす必要があります。

- OpenShift Container Platform 4.15 が AWS または Azure にインストールされている。
- IBM Z または IBM® LinuxONE へのインストールには、OpenShift Container Platform 4.14 以降がインストールされている。

重要

IBM Z でのピア Pod を使用した OpenShift Sandboxed Containers ワークロードのデプロイは、テクノロジープレビューとしてのみ提供されています。テクノロジープレビュー機能は、Red Hat 製品サポート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場合があります。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。

- 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。

3.1.1. OpenShift Sandboxed Containers のピア Pod リソース要件について

ピア Pod は、次の 2 つの場所でリソースを使用します。

- ワーカーノード。ワーカーノードは、メタデータ、Kata shim リソース (**containerd-shim-kata-v2**)、remote-hypervisor リソース (**cloud-api-adaptor**) リソース、およびワーカーノードとピア Pod 仮想マシン (VM) 間のトンネル設定を保存します。
- クラウドインスタンス。これは、クラウド内で実行されている実際のピア Pod VM です。

Kubernetes ワーカーノードで使用される CPU およびメモリーリソースは、ピア Pod の作成に使用される RuntimeClass (**kata-remote**) 定義に含まれる Pod オーバーヘッド によって処理されます。

クラウド内で実行されているピア Pod VM の合計数は、Kubernetes ノード拡張リソースとして定義されます。この制限はノードごとであり、**peerpodConfig** カスタムリソース (CR) の **limit** 属性によって設定されます。**peerpodconfig-openshift** という名前の **peerpodConfig** CR は、**kataConfig** CR を作成してピア Pod を有効にするときに作成され、**openshift-sandboxed-containers-operator** namespace に配置されます。

次の **peerpodConfig** CR の例は、デフォルトの **spec** 値を示しています。

```
apiVersion: confidentialcontainers.org/v1alpha1
kind: PeerPodConfig
metadata:
  name: peerpodconfig-openshift
  namespace: openshift-sandboxed-containers-operator
spec:
  cloudSecretName: peer-pods-secret
  configMapName: peer-pods-cm
  limit: "10" (1)
  nodeSelector:
    node-role.kubernetes.io/kata-oc: ""
```

1 デフォルトの制限は、ノードごとに 10 VM です。

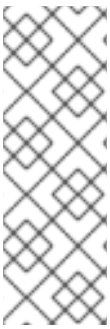
拡張リソースの名前は **kata.peerpods.io/vm** で、Kubernetes スケジューラーが容量の追跡とアカウンティングを処理できるようにします。

ご使用の環境の要件に基づいて、ノードごとの制限を編集できます。詳細は、ピア Pod のノードごとの VM 制限の変更 を 参照してください。

mutating Webhook により、拡張リソース **kata.peerpods.io/vm** が Pod 仕様に追加されます。また、リソース固有のエントリーが存在する場合は、Pod 仕様から削除されます。こうすることで、Kubernetes スケジューラーがこれらの拡張リソースを考慮できるようになり、リソースが利用可能な場合にのみピア Pod がスケジュールされるようになります。

mutating Webhook は、次のように Kubernetes Pod を変更します。

- mutating Webhook は、**TARGET_RUNTIME_CLASS** 環境変数で指定された **RuntimeClassName** の想定値であるか、Pod をチェックします。Pod 仕様の値が **TARGET_RUNTIME_CLASS** の値と一致しない場合、Webhook は Pod を変更せずに終了します。
- **RuntimeClassName の** 値が一致する場合、Webhook は Pod 仕様に次の変更を加えます。
  1. この Webhook は、Pod 内のすべてのコンテナーおよび初期化コンテナーの **resources** フィールドからすべてのリソース仕様を削除します。
  2. Webhook は、Pod 内の最初のコンテナーのリソースフィールドを変更して、拡張リソース (**kata.peerpods.io/vm**) を仕様に追加します。拡張リソース **kata.peerpods.io/vm** は Kubernetes スケジューラーによってアカウンティング目的で使用されます。



注記

mutating Webhook は、OpenShift Container Platform の特定のシステム namespace が変更されないように除外します。これらのシステム namespace でピア Pod が作成された場合、Pod の仕様に拡張リソースが含まれていない限り、Kubernetes 拡張リソースを使用したリソースアカウンティングは機能しません。

ベストプラクティスとして、特定の namespace でのみピア Pod の作成を許可するクラスター全体のポリシーを定義します。

3.1.1.1. ノードごとのピア Pod VM 制限の変更

ノードごとのピア Pod VM の制限を変更するには、**peerpodConfig** カスタムリソース (CR) を編集します。

手順

1. 次のコマンドを実行して、現在の制限を確認します。

```
$ oc get peerpodconfig peerpodconfig-openshift -n openshift-sandboxed-containers-operator
\
-o jsonpath='{.spec.limit}{"\n"}'
```

2. 次のコマンドを実行して、**peerpodConfig** CR の **limit** 属性を変更します。

```
$ oc patch peerpodconfig peerpodconfig-openshift -n openshift-sandboxed-containers-
operator \
--type merge --patch '{"spec":{"limit":"<value>"}}' 1
```

1 <value> は、定義する制限に置き換えます。

3.1.2. AWS を使用したピア Pod の前提条件

AWS を使用してピア Pod を作成している場合は、次の要件を確認する必要があります。

- OpenShift Container Platform クラスターを AWS にインストールし、ワーカーノードを最低でも 1 つ用意する。
- **AWS_ACCESS_KEY_ID** および **AWS_SECRET_ACCESS_KEY** 認証情報にアクセスできる。これらは、クラスターの同じ Virtual Private Cloud (VPC) 内に追加のクラウドインスタンスを作成するために使用されます。
- AWS CLI ツールをインストールして設定しておく。
- ポート 15150 および 9000 で内部クラスター通信を有効にする。これらのポートは、AWS ウェブコンソールまたは CLI を使用して有効にできます。

3.1.2.1. AWS のポート 15150 および 9000 の有効化

手順

1. インスタンス ID を取得します。

```
$ INSTANCE_ID=$(oc get nodes -l 'node-role.kubernetes.io/worker' -o
jsonpath='{.items[0].spec.providerID}' | sed 's#[^ ]*/##g')
```

2. AWS リージョンを取得します。

```
$ AWS_REGION=$(oc get infrastructure/cluster -o
jsonpath='{.status.platformStatus.aws.region}')
```

3. セキュリティーグループを取得します。

```
$ SG=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
'Reservations[*].Instances[*].SecurityGroups[*].GroupId' --output text --region
${AWS_REGION})
```

4. ピア Pod シムを承認し、kata-agent 通信にアクセスすることを許可します。以下のコマンドを実行します。

```
$ aws ec2 authorize-security-group-ingress --group-id ${SG} --protocol tcp --port 15150 --source-group ${SG} --region ${AWS_REGION}
```

5. ピア Pod トンネルを設定します。以下のコマンドを実行します。

```
$ aws ec2 authorize-security-group-ingress --group-id ${SG} --protocol tcp --port 9000 --source-group ${SG} --region ${AWS_REGION}
```

これでポートが有効になりました。

3.1.3. Azure を使用するピア Pod の前提条件

Microsoft Azure を使用してピア Pod を作成している場合は、次の要件を確認する必要があります。

- OpenShift Container Platform クラスターを Azure にインストールし、ワーカーノードを最低でも 1 つ用意する。
- 次の認証情報とサブスクリプションの情報にアクセスできる。
  - **AZURE_SUBSCRIPTION_ID**
  - **AZURE_CLIENT_ID**
  - **AZURE_CLIENT_SECRET**
  - **AZURE_TENANT_ID**

  これらは、クラスターの同じ Virtual Private Cloud (VPC) 内に追加のクラウドインスタンスを作成するために使用されます。
- Azure CLI ツールをインストールして設定しておく。

3.1.4. RHEL KVM で IBM Z または IBM(R) LinuxONE を使用するピア Pod の前提条件

OpenShift CLI (**oc**) を使用して、OpenShift Sandboxed Containers Operator を IBM Z にインストールできます。

注記

本書は IBM Z のみを参照しますが、これに含まれるすべての情報は IBM® LinuxONE にも適用されます。

IBM Z を使用してピア Pod を作成している場合は、次の要件を確認する必要があります。

- OpenShift Container Platform クラスターは、少なくとも 1 つのコンピュートノードを使用して IBM Z にインストールしておく。
- IBM Z KVM ホストに次のツールがインストールされている。
  - libvirt
  - podman

- git
- tar
- virt-customize

- **oc** CLI ツールをインストールして設定しておく。

非実稼働環境でピア Pod をテストする場合は、次の要件を満たす必要があります。

- OpenShift Container Platform クラスターはバージョン 4.14 以降である。
- マルチノード OpenShift Container Platform クラスターに、3 つのコントロールノードと 2 つのコンピュートノードがセットアップされている。
- クラスターノードとピア Pod は、単一の IBM Z KVM ホスト論理パーティション (LPAR) 上で実行している。
- ピア Pod 仮想マシン (VM) とクラスターノードが同じサブネットに含まれており、ノードとピア Pod VM 間の接続を可能にするようにクラスターが設定されている。

3.2. WEB コンソールでピア POD を使用した OPENSHIFT SANDBOXED CONTAINERS ワークロードのデプロイ

Web コンソールから OpenShift Sandboxed Containers のワークロードをデプロイできます。まず、OpenShift Sandboxed Containers Operator をインストールし、次にシークレットオブジェクト、VM イメージ、およびピア Pod ConfigMap を作成する必要があります。シークレットオブジェクトと ConfigMap は一意です。クラウドプロバイダーに応じて指定します。最後に、**KataConfig** カスタムリソース (CR) を作成する必要があります。Sandboxed Containers にワークロードをデプロイする準備ができたら、**kata-remote** 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。

3.2.1. Web コンソールを使用した OpenShift Sandboxed Containers 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 4.15 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** 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 Sandboxed Containers Operator を **openshift-sandboxed-containers-operator** 以外の namespace にインストールしようとすると、インストールに失敗します。

   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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers Operator がインストール済みの Operator リストに表示されていることを確認します。

3.2.2. Web コンソールを使用した AWS のピア Pod パラメーターの設定

AWS 上のピア Pod を使用して OpenShift Sandboxed Containers をデプロイするには、シークレットオブジェクトと **ConfigMap** を作成する必要があります。

シークレットオブジェクトを作成した後も、ピア Pod を使用して OpenShift Sandboxed Containers をデプロイするために **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 4.15 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 Sandboxed Containers Operator をインストールしている。

3.2.2.1. Web コンソールを使用した AWS のシークレットオブジェクトの作成

AWS アクセスキーを指定してシークレットオブジェクトでネットワークを設定します。シークレットオブジェクトは、Pod 仮想マシンイメージの作成に使用され、ピア Pod によって使用されます。

AWS のシークレットオブジェクトを作成するときは、特定の環境値を設定する必要があります。シークレットオブジェクトを作成する前に、これらの値の一部を取得できます。CLI を使用してこれらの値を取得する必要があります。詳細は、CLI を使用した AWS のシークレットオブジェクトの作成 を参照

してください。

さらに、AWS Web コンソールで次の値を生成し、保存する必要があります。

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

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. Operator のリストから OpenShift Sandboxed Containers Operator を選択します。
3. 右上隅にあるインポートアイコン (+) をクリックします。
4. **Import YAML** ウィンドウに、次の YAML マニフェストを貼り付けます。

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AWS_ACCESS_KEY_ID: "<enter value>" 1
  AWS_SECRET_ACCESS_KEY: "<enter value>" 2
  AWS_REGION: "<enter value>" 3
  AWS_SUBNET_ID: "<enter value>" 4
  AWS_VPC_ID: "<enter value>" 5
  AWS_SG_IDS: "<enter value>" 6
```

1. 開始する前に準備した **AWS_ACCESS_KEY_ID** 値を入力します。
2. 開始する前に準備した **AWS_SECRET_ACCESS_KEY** 値を入力します。
3. 取得した **AWS_REGION** 値を入力します。
4. 取得した **AWS_SUBNET_ID** 値を入力します。
5. 取得した **AWS_VPC_ID** 値を入力します。
6. 取得した **AWS_SG_IDS** 値を入力します。

5. **Create** をクリックします。

シークレットオブジェクトが作成されます。**Workloads → Secrets** の下に表示されていることが確認できます。

3.2.2.2. Web コンソールを使用した AWS 用のピア Pod ConfigMap の作成

Web コンソールを使用して、AWS のピア Pod **ConfigMap** を作成できます。

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. Operator のリストから OpenShift Sandboxed Containers Operator を選択します。
3. 右上隅にあるインポートアイコン (+) をクリックします。
4. **Import YAML** ウィンドウに、次の YAML マニフェストを貼り付けます。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "aws"
  VXLAN_PORT: "9000"
  PODVM_INSTANCE_TYPE: "t3.medium" (1)
  PODVM_INSTANCE_TYPES: "t2.small,t2.medium,t3.large" (2)
  PROXY_TIMEOUT: "5m"
```

(1) ワークロードでタイプが定義されていない場合に使用されるデフォルトのインスタンスタイプを定義します。

(2) Pod の作成時に指定できるすべてのインスタンスタイプを一覧表示します。これにより、必要なメモリーと CPU が少ないワークロードには小さなインスタンスを定義したり、より大きなワークロードには大きなインスタンスを定義したりできます。

5. **Create** をクリックします。

ConfigMap オブジェクトが作成されます。**Workloads → ConfigMaps** の下に表示されていることが確認できます。

KataConfig CR を作成すると、AWS 上のピア Pod を使用して OpenShift Sandboxed Containers を実行できます。

3.2.3. Web コンソールを使用した Azure のピア Pod パラメーターの設定

Microsoft Azure 上のピア Pod を使用して OpenShift Sandboxed Containers をデプロイするには、シークレットオブジェクトと **ConfigMap** を作成する必要があります。

シークレットオブジェクトを作成した後も、ピア Pod を使用して OpenShift Sandboxed Containers をデプロイするために **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 4.15 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 Sandboxed Containers Operator をインストールしている。

3.2.3.1. Web コンソールを使用した Azure のシークレットオブジェクトの作成

Azure アクセスキーを設定し、シークレットオブジェクトでネットワークを設定します。シークレットオブジェクトは、Pod 仮想マシンイメージの作成に使用され、ピア Pod によって使用されます。

Azure のシークレットオブジェクトを作成するときは、特定の環境値を設定する必要があります。これらの値は、シークレットオブジェクトを作成する前に取得できます。CLI を使用してこれらの値を取得する必要があります。詳細は、CLI を使用した Azure のシークレットオブジェクトの作成 を参照してください。

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. Operator のリストから OpenShift Sandboxed Containers Operator を選択します。
3. 右上隅にあるインポートアイコン (+) をクリックします。
4. **Import YAML** ウィンドウに、次の YAML マニフェストを貼り付けます。

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AZURE_CLIENT_ID: "<enter value>" (1)
  AZURE_CLIENT_SECRET: "<enter value>" (2)
  AZURE_TENANT_ID: "<enter value>" (3)
  AZURE_SUBSCRIPTION_ID: "<enter value>" (4)
  AZURE_REGION: "<enter value>" (5)
  AZURE_RESOURCE_GROUP: "<enter value>" (6)
```

(1) RBAC ファイルで生成した **AZURE_CLIENT_ID** 値を入力します。

(2) RBAC ファイルで生成した **AZURE_CLIENT_SECRET** 値を入力します。

(3) RBAC ファイルで生成した **AZURE_TENANT_ID** 値を入力します。

(4) 取得した **AZURE_SUBSCRIPTION_ID** 値を入力します。

(5) 取得した **AZURE_REGION** 値を入力します。

(6) 取得した **AZURE_RESOURCE_GROUP** 値を入力します。

5. **Create** をクリックします。

シークレットオブジェクトが作成されます。**Workloads → Secrets** の下に表示されていることが確認できます。

3.2.3.2. Web コンソールを使用した Azure のピア Pod ConfigMap の作成

Web コンソールを使用して Azure のピア Pod **ConfigMap** を作成できます。

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. Operator のリストから OpenShift Sandboxed Containers Operator を選択します。
3. 右上隅にあるインポートアイコン (+) をクリックします。
4. **Import YAML** ウィンドウに、次の YAML マニフェストを貼り付けます。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "azure"
  VXLAN_PORT: "9000"
  AZURE_INSTANCE_SIZE: "Standard_B2als_v2" (1)
  AZURE_INSTANCE_SIZES:
"Standard_DC2as_v5,Standard_DC4as_v5,Standard_DC8as_v5" (2)
  AZURE_SUBNET_ID: "<enter value>" (3)
  AZURE_NSG_ID: "<enter value>" (4)
  PROXY_TIMEOUT: "5m"
  DISABLECVM: "true"
```

(1) インスタンスがワークロードに定義されていない場合に使用されるデフォルトのインスタンスサイズを定義します。

(2) Pod の作成時に指定できるすべてのインスタンスサイズを一覧表示します。これにより、必要なメモリーと CPU が少ないワークロードには小さなインスタンスを定義したり、より大きなワークロードには大きなインスタンスを定義したりできます。

(3) 取得した **AZURE_SUBNET_ID** 値を入力します。

(4) 取得した **AZURE_NSG_ID** 値を入力します。

5. **Create** をクリックします。

ConfigMap` オブジェクトが作成されます。**Workloads → ConfigMaps** の下に表示されていることが確認できます。

3.2.3.3. Web コンソールを使用した Azure の SSH キーシークレットオブジェクトの作成

Azure でピア Pod を使用するには、SSH キーシークレットオブジェクトを作成する必要があります。オブジェクトの作成用の SSH 鍵がない場合は、CLI を使用して SSH 鍵を生成する必要があります。詳細は、以下を参照してください。

手順

1. Web コンソールの **Administrator** パースペクティブから、**Workloads → Secrets** に移動します。
2. **Secrets** ウィンドウの左上隅で、**openshift-sandboxed-containers-operator** プロジェクトにいることを確認します。
3. **Create** をクリックし、リストから **Key/value secret** を選択します。
4. **Secret name** フィールドに **ssh-key-secret** と入力します。
5. **Key** フィールドに **id_rsa.pub** と入力します。
6. **Value** フィールドに、公開 SSH 鍵を貼り付けます。
7. **Create** をクリックします。

SSH 鍵のシークレットオブジェクトが作成されます。**Workloads → Secrets** の下に表示されていることが確認できます。

KataConfig CR を作成すると、Azure 上のピア Pod を使用して OpenShift Sandboxed Containers を実行できます。

3.2.4. Web コンソールでの KataConfig カスタムリソースの作成

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

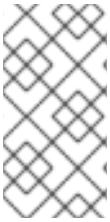
重要

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が有効である。
- SSD ではなくハードディスクドライブにデプロイしている。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイしている。
- CPU とネットワークが遅い。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator をインストールしている。



注記

ピア Pod の 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。**kata-remote**を 特定のノードにのみ **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. Operator のリストから OpenShift Sandboxed Containers Operator 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、ピア Pod には該当しない): **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 Sandboxed Containers 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。
   - **EnablePeerPods** (ピア Pod の場合): ピア Pod を有効にし、パブリッククラウド環境で OpenShift Sandboxed Containers を使用するには、このチェックボックスをオンにします。
   - **kataConfigPoolSelector**: デフォルトでは、**kata-remote** 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-remote** 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。

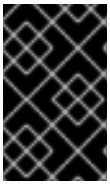
- **logLevel**:**kata-remote** を **RuntimeClass** として実行しているノードに対して取得するログデータのレベルを定義します。詳細は、「OpenShift Sandboxed Containers データの収集」を参照してください。

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 **RuntimeClass** として **kata-remote** をインストールし始めます。インストールが完了してワーカーノードが再起動するまで待ってから、次の手順に進みます。

注記

CR を実行すると、VM イメージが作成されます。イメージの作成はクラウドプロバイダーにより行われ、追加のリソースを使用する場合があります。



重要

OpenShift Sandboxed Containers は、**kata-remote** をプライマリーランタイムとしてではなく、クラスター上の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. **status** フィールドを監視します。
   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。

   status フィールドには **conditions** および **kataNodes** サブフィールドがあります。**kataNodes** サブフィールドはノード一覧のコレクションです。各ノードは、**kata** インストールの特定状態のノードを一覧表示します。

   kataNodes の下のすべてのワーカーが **installs** としてリストされ、理由を指定せずに **InProgress** の条件が **False** の場合は、**kata** がクラスターにインストールされていることを示します。詳細は、「移行のインストールとアンインストール」を参照してください。

3.2.5. Web コンソールを使用したピア Pod を含むワークロードの Sandboxed Containers へのデプロイ

OpenShift Sandboxed Containers は、Kata をプライマリー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。

Sandboxed Containers 内のピア Pod を使用して Pod テンプレート化されたワークロードをデプロイするには、**kata-remote** 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。

また、YAML ファイルにアノテーションを追加して、**ConfigMap** で以前に定義したデフォルトのインスタンスサイズまたはタイプを使用してワークロードをデプロイするかどうかも定義する必要があります。インスタンスサイズまたはインスタンスタイプの使用は、クラウドプロバイダーによって異なります。インスタンスのサイズまたはタイプを手動で定義しない場合は、アノテーションを追加して、使用可能なメモリーに基づいて自動インスタンスのサイズまたはタイプの使用を定義する必要があります。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator をインストールしている。
- クラウドプロバイダーに固有のシークレットオブジェクトとピア Pod の **ConfigMap** を作成している。
- **KataConfig** カスタムリソース (CR) を作成している。

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-remote** を追加します。
4. ワークロードの YAML ファイルにアノテーションを追加して、デフォルトのインスタンスサイズまたはタイプを使用するか、自動インスタンスサイズまたはタイプを使用するかを定義します。インスタンスサイズは特定のクラウドプロバイダーに使用され、インスタンスタイプは他のクラウドプロバイダーに使用されます。
   - 特定のインスタンスサイズタイプについては、次のアノテーションを追加します。

     ```
     io.katacontainers.config.hypervisor.machine_type: <instance type/instance size>
     ```

     ワークロードが使用するインスタンスのサイズまたはタイプを定義します。これらのデフォルトのサイズまたはタイプは、ピア Pod の **ConfigMap** を作成するときに事前に定義してあります。そのうちの 1 つを選択してください。
   - 自動インスタンスのサイズまたはタイプについては、次のアノテーションを追加します。

     ```
     io.katacontainers.config.hypervisor.default_vcpus: <vcpus>
     io.katacontainers.config.hypervisor.default_memory: <memory>
     ```

     ワークロードが使用できるメモリーの量を定義します。ワークロードは、使用可能なメモリーの量に基づいて、自動インスタンスサイズまたはタイプで実行されます。

Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
  annotations:
    io.katacontainers.config.hypervisor.machine_type: Standard_DC4as_v5 1
spec:
  runtimeClassName: kata-remote
```

```
containers:
  - name: hello-openshift
    image: quay.io/openshift/origin-hello-openshift
    ports:
      - containerPort: 8888
    securityContext:
      privileged: false
      allowPrivilegeEscalation: false
      runAsNonRoot: true
      runAsUser: 1001
      capabilities:
        drop:
          - ALL
      seccompProfile:
        type: RuntimeDefault
```

1 この例では、Azure を使用してピア Pod に事前に定義されたインスタンスサイズを使用します。AWS を使用するピア Pod はインスタンスタイプを使用します。

5. **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 でピア POD を使用した OPENSHIFT SANDBOXED CONTAINERS ワークロードのデプロイ

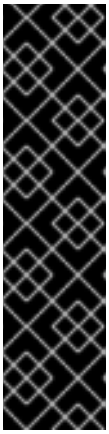
CLI を使用して、OpenShift Sandboxed Containers のワークロードをデプロイできます。まず、OpenShift Sandboxed Containers Operator をインストールしてから、**KataConfig** カスタムリソースを作成する必要があります。Sandboxed Containers にワークロードをデプロイする準備ができたら、**kata-remote** を **runtimeClassName** としてワークロード YAML ファイルに追加する必要があります。

3.3.1. CLI を使用したSandboxed Containers 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 4.15 がインストールされている。
- IBM Z または IBM® LinuxONE へのインストールには、OpenShift Container Platform 4.14 以降がインストールされている。

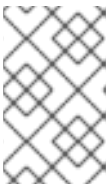


重要

IBM Z でのピア Pod を使用した OpenShift Sandboxed Containers ワークロードのデプロイは、テクノロジープレビューとしてのみ提供されています。テクノロジープレビュー機能は、Red Hat 製品サポート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場合があります。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。

- 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 Sandboxed Containers カタログにサブスクライブしている。



注記

OpenShift Sandboxed Containers カタログにサブスクライブすると、**openshift-sandboxed-containers-operator** namespace の OpenShift Sandboxed Containers Operator にアクセスできるようになります。

手順

1. OpenShift Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers Operator にサブスクライブします。

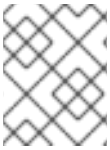
   a. 次のマニフェストを含む **Subscription** オブジェクト YAML ファイルを作成します。

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: openshift-sandboxed-containers-operator
  namespace: openshift-sandboxed-containers-operator
spec:
  channel: stable
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.5.2
```

   b. **Subscription** オブジェクトを作成します。

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

これで、OpenShift Sandboxed Containers 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  1.5.2    1.5.1
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 を使用した AWS のピア Pod の設定

AWS で使用するピア Pod を設定するには、シークレットオブジェクト、AWS イメージ VM (AMI)、およびピア Pod の **ConfigMap** を作成する必要があります。

AWS のピア Pod を設定した後に、ピア Pod を使用して OpenShift Sandboxed Containers をデプロイするには、**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 4.15 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 Sandboxed Containers Operator をインストールしている。

3.3.2.1. CLI を使用した AWS のシークレットオブジェクトの作成

AWS アクセスキーを指定してシークレットオブジェクトでネットワークを設定します。シークレットオブジェクトは、Pod 仮想マシンイメージの作成に使用され、ピア Pod によって使用されます。

AWS のシークレットオブジェクトを作成するときは、特定の環境値を設定する必要があります。シークレットオブジェクトを作成する前に、これらの値の一部を取得できます。ただし、次の値を準備しておく必要があります。

- **AWS_ACCESS_KEY_ID**
- **AWS_SECRET_ACCESS_KEY**

これらの値は、AWS コンソールで生成できます。

手順

1. シークレットオブジェクトに必要なパラメーター値を収集します。それぞれの値を必ず書き留めてください。
   a. インスタンス ID を取得します。

   ```
   $ INSTANCE_ID=$(oc get nodes -l 'node-role.kubernetes.io/worker' -o
   jsonpath='{.items[0].spec.providerID}' | sed 's#[^ ]*/##g')
   ```

   この値はシークレットオブジェクト自体には必要ありませんが、シークレットオブジェクトの他の値を取得するために使用されます。

   b. AWS リージョンを取得します。

   ```
   $ AWS_REGION=$(oc get infrastructure/cluster -o
   jsonpath='{.status.platformStatus.aws.region}') && echo "AWS_REGION:
   \"$AWS_REGION\""
   ```

   c. AWS サブネット ID を取得します。

   ```
   $ AWS_SUBNET_ID=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --
   query 'Reservations[*].Instances[*].SubnetId' --region ${AWS_REGION} --output text) &&
   echo "AWS_SUBNET_ID: \"$AWS_SUBNET_ID\""
   ```

   d. AWS VPC ID を取得します。

```
$ AWS_VPC_ID=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
'Reservations[*].Instances[*].VpcId' --region ${AWS_REGION} --output text) && echo
"AWS_VPC_ID: \"$AWS_VPC_ID\""
```

e. AWS セキュリティーグループ ID を取得します。

```
$ AWS_SG_IDS=$(aws ec2 describe-instances --instance-ids ${INSTANCE_ID} --query
'Reservations[*].Instances[*].SecurityGroups[*].GroupId' --region ${AWS_REGION} --
output text)
&& echo "AWS_SG_IDS: \"$AWS_SG_IDS\""
```

2. 次のマニフェストで YAML ファイルを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  AWS_ACCESS_KEY_ID: "<enter value>" 1
  AWS_SECRET_ACCESS_KEY: "<enter value>" 2
  AWS_REGION: "<enter value>" 3
  AWS_SUBNET_ID: "<enter value>" 4
  AWS_VPC_ID: "<enter value>" 5
  AWS_SG_IDS: "<enter value>" 6
```

1. 開始する前に準備した **AWS_ACCESS_KEY_ID** 値を入力します。
2. 開始する前に準備した **AWS_SECRET_ACCESS_KEY** 値を入力します。
3. 取得した **AWS_REGION** 値を入力します。
4. 取得した **AWS_SUBNET_ID** 値を入力します。
5. 取得した **AWS_VPC_ID** 値を入力します。
6. 取得した **AWS_SG_IDS** 値を入力します。

3. シークレットオブジェクトを適用します。

```
$ oc apply -f peer-pods-secret.yaml
```

シークレットオブジェクトが適用されました。

3.3.2.2. CLI を使用した AWS 用のピア Pod ConfigMap の作成

AWS の **ConfigMap** を作成するときは、AMI ID を設定する必要があります。この値は、**ConfigMap** を作成する前に取得できます。

手順

1. 次のマニフェストで YAML ファイルを作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "aws"
  VXLAN_PORT: "9000"
  PODVM_INSTANCE_TYPE: "t3.medium" (1)
  PODVM_INSTANCE_TYPES: "t2.small,t2.medium,t3.large" (2)
  PROXY_TIMEOUT: "5m"
```

1. ワークロードでタイプが定義されていない場合に使用されるデフォルトのインスタンスタイプを定義します。
2. Pod の作成時に指定できるすべてのインスタンスタイプを一覧表示します。これにより、必要なメモリーと CPU が少ないワークロードには小さなインスタンスを定義したり、より大きなワークロードには大きなインスタンスを定義したりできます。

2. **ConfigMap** を適用します。

```
$ oc apply -f peer-pods-cm.yaml
```

ConfigMap が適用されます。**KataConfig** CR を作成すると、AWS 上のピア Pod を使用して OpenShift Sandboxed Containers を実行できます。

3.3.3. CLI を使用した Azure のピア Pod の設定

Microsoft Azure で使用するピア Pod を設定するには、シークレットオブジェクト、Azure イメージ VM、ピア Pod **ConfigMap**、および SSH 鍵のシークレットオブジェクトを作成する必要があります。

Azure のピア Pod を設定した後、ピア Pod を使用して OpenShift Sandboxed Containers をデプロイするには、**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 4.15 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 Sandboxed Containers Operator をインストールしている。
- Azure CLI ツールをインストールして設定しておく。

3.3.3.1. CLI を使用した Azure のシークレットオブジェクトの作成

Azure アクセスキーを設定し、シークレットオブジェクトでネットワークを設定します。シークレットオブジェクトは、Pod 仮想マシンイメージの作成に使用され、ピア Pod によって使用されます。

Azure のシークレットオブジェクトを作成するときは、特定の環境値を設定する必要があります。これらの値は、シークレットオブジェクトを作成する前に取得できます。ロールベースのアクセス制御 (RBAC) ファイルも作成する必要があります。このファイルでは以下の値が生成されます。

- **AZURE_CLIENT_ID**
- **AZURE_CLIENT_SECRET**
- **AZURE_TENANT_ID**

手順

1. Azure サブスクリプション ID を取得します。

```
$ AZURE_SUBSCRIPTION_ID=$(az account list --query "[?isDefault].id" -o tsv) && echo
"AZURE_SUBSCRIPTION_ID: \"$AZURE_SUBSCRIPTION_ID\""
```

2. RBAC コンテンツを生成します。これにより、クライアント ID、クライアントシークレット、およびテナント ID が生成されます。

```
$ az ad sp create-for-rbac --role Contributor --scopes
/subscriptions/$AZURE_SUBSCRIPTION_ID --query "{ client_id: appId, client_secret:
password, tenant_id: tenant }
```

   以下の出力が表示されます。

```
{
  "client_id": `AZURE_CLIENT_ID`,
  "client_secret": `AZURE_CLIENT_SECRET`,
  "tenant_id": `AZURE_TENANT_ID`
}
```

3. シークレットオブジェクトで使用するために、RBAC 出力の値を保存します。
4. シークレットオブジェクトの追加のパラメーター値を収集します。それぞれの値を必ず書き留めてください。
   a. リソースグループを取得します。

```
$ AZURE_RESOURCE_GROUP=$(oc get infrastructure/cluster -o
jsonpath='{.status.platformStatus.azure.resourceGroupName}') && echo
"AZURE_RESOURCE_GROUP: \"$AZURE_RESOURCE_GROUP\""
```

   b. Azure リージョンを取得します。

```
$ AZURE_REGION=$(az group show --resource-group
${AZURE_RESOURCE_GROUP} --query "{Location:location}" --output tsv) && echo
"AZURE_REGION: \"$AZURE_REGION\""
```

5. 次のマニフェストで YAML ファイルを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
```

```
AZURE_CLIENT_ID: "<enter value>" 1
AZURE_CLIENT_SECRET: "<enter value>" 2
AZURE_TENANT_ID: "<enter value>" 3
AZURE_SUBSCRIPTION_ID: "<enter value>" 4
AZURE_REGION: "<enter value>" 5
AZURE_RESOURCE_GROUP: "<enter value>" 6
```

1. RBAC ファイルで生成した **AZURE_CLIENT_ID** 値を入力します。
2. RBAC ファイルで生成した **AZURE_CLIENT_SECRET** 値を入力します。
3. RBAC ファイルで生成した **AZURE_TENANT_ID** 値を入力します。
4. 取得した **AZURE_SUBSCRIPTION_ID** 値を入力します。
5. 取得した **AZURE_REGION** 値を入力します。
6. 取得した **AZURE_RESOURCE_GROUP** 値を入力します。

6. シークレットオブジェクトを適用します。

```
$ oc apply -f peer-pods-secret.yaml
```

シークレットオブジェクトが適用されました。

3.3.3.2. CLI を使用した Azure のピア Pod ConfigMap の作成

Azure の **ConfigMap** を作成するときは、特定の設定値を指定する必要があります。これらの値は、**ConfigMap** を作成する前に取得できます。

手順

1. Azure ピア Pod **ConfigMap** の設定値を収集します。それぞれの値を必ず書き留めてください。
   a. Azure VNet 名を取得します。

   ```
   $ AZURE_VNET_NAME=$(az network vnet list --resource-group
   ${AZURE_RESOURCE_GROUP} --query "[].{Name:name}" --output tsv)
   ```

   この値は **ConfigMap** には必要ありませんが、Azure サブネット ID を取得するために使用されます。

   b. Azure サブネット ID を取得します。

   ```
   $ AZURE_SUBNET_ID=$(az network vnet subnet list --resource-group
   ${AZURE_RESOURCE_GROUP} --vnet-name $AZURE_VNET_NAME --query "[].{Id:id}
   | [? contains(Id, 'worker')]" --output tsv) && echo "AZURE_SUBNET_ID:
   \"$AZURE_SUBNET_ID\""
   ```

   c. Azure ネットワークセキュリティーグループ (NSG) ID を取得します。

```
$ AZURE_NSG_ID=$(az network nsg list --resource-group
${AZURE_RESOURCE_GROUP} --query "[].{Id:id}" --output tsv) && echo
"AZURE_NSG_ID: \"$AZURE_NSG_ID\""
```

2. 次のマニフェストで YAML ファイルを作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "azure"
  VXLAN_PORT: "9000"
  AZURE_INSTANCE_SIZE: "Standard_B2als_v2" (1)
  AZURE_INSTANCE_SIZES:
"Standard_DC2as_v5,Standard_DC4as_v5,Standard_DC8as_v5" (2)
  AZURE_SUBNET_ID: "<enter value>" (3)
  AZURE_NSG_ID: "<enter value>" (4)
  PROXY_TIMEOUT: "5m"
  DISABLECVM: "true"
```

(1) インスタンスがワークロードに定義されていない場合に使用されるデフォルトのインスタンスサイズを定義します。

(2) Pod の作成時に指定できるすべてのインスタンスサイズを一覧表示します。これにより、必要なメモリーと CPU が少ないワークロードには小さなインスタンスを定義したり、より大きなワークロードには大きなインスタンスを定義したりできます。

(3) 取得した **AZURE_SUBNET_ID** 値を入力します。

(4) 取得した **AZURE_NSG_ID** 値を入力します。

3. **ConfigMap** を適用します。

```
$ oc apply -f peer-pods-cm.yaml
```

ConfigMap がデプロイされました。

3.3.3.3. CLI を使用した Azure の SSH 鍵のシークレットオブジェクト作成

Azure でピア Pod を使用するには、SSH 鍵を生成し、SSH 鍵のシークレットオブジェクトを作成する必要があります。

手順

1. SSH 鍵を生成します。

```
$ ssh-keygen -f ./id_rsa -N ""
```

2. SSH シークレットオブジェクトを作成します。

```
$ oc create secret generic ssh-key-secret -n openshift-sandboxed-containers-operator --
from-file=id_rsa.pub=./id_rsa.pub --from-file=id_rsa=./id_rsa
```

SSH 鍵が作成され、SSH 鍵のシークレットオブジェクトが作成されます。**KataConfig** CR を作成すると、Azure 上のピア Pod を使用して OpenShift Sandboxed Containers を実行できます。

3.3.4. CLI を使用した IBM Z のピア Pod のセットアップ

IBM Z 上で実行している OpenShift Container Platform クラスターで使用するピア Pod をセットアップするには、シークレットオブジェクト、RHEL KVM イメージ仮想マシン、およびピア Pod **ConfigMap** を作成する必要があります。これらは、libvirt と KVM ホストとの間の通信に必要な認証情報を保持します。

IBM Z のピア Pod を設定した後に、ピア Pod を使用して OpenShift Sandboxed Containers をデプロイするには、**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

- OpenShift Container Platform 4.14 以降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。
- **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 Sandboxed Containers Operator をインストールしている。
- KVM ホストに libvirt がインストールされており、管理者権限が割り当てられている。

3.3.4.1. KVM ホストでの libvirt のセットアップ

KVM ホスト上に libvirt をセットアップする必要があります。IBM Z 上のピア Pod は、クラウド API アダプターの libvirt プロバイダーを使用して仮想マシンを作成および管理します。

手順

1. IBM Z KVM ホストにログインし、libvirt がストレージ管理に使用するシェル変数を設定します。
   a. 次のコマンドを実行して、libvirt プールの名前を設定します。

   ```
   $ export LIBVIRT_POOL=<name_of_libvirt_pool_to_create>
   ```

   b. 次のコマンドを実行して、libvirt プールの名前を設定します。

   ```
   $ export LIBVIRT_VOL_NAME=<name_of_libvirt_volume_to_create>
   ```

   c. 次のコマンドを実行して、デフォルトのストレージプールの場所のパスを設定します。

   ```
   $ export LIBVIRT_POOL_DIRECTORY=<name_of_target_directory> 1
   ```

   1 libvirt に読み取りおよび書き込みアクセス許可があることを確認するには、libvirt のストレージディレクトリーのサブディレクトリーを使用します。デフォルトは **/var/lib/libvirt/images/** です。

2. 次のコマンドを実行して、libvirt プールを作成します。

```
$ virsh pool-define-as $LIBVIRT_POOL --type dir --target "$LIBVIRT_POOL_DIRECTORY"
```

3. 次のコマンドを実行して、libvirt プールを開始します。

```
$ virsh pool-start $LIBVIRT_POOL
```

4. 次のコマンドを実行して、プールの libvirt ボリュームを作成します。

```
$ virsh -c qemu:///system \
  vol-create-as --pool $LIBVIRT_POOL \
  --name $LIBVIRT_VOL_NAME \
  --capacity 20G \
  --allocation 2G \
  --prealloc-metadata \
  --format qcow2
```

3.3.4.2. IBM Z 用のピア PodVM イメージの作成

IBM Z 上でピア Pod を使用して OpenShift Sandboxed Containers を実行するには、まず、libvirt プロバイダーがピア Pod VM を起動する KVM ゲストを作成する必要があります。

イメージが作成されたら、前の手順で作成したボリュームにイメージをアップロードします。

前提条件

- IBM z15 以降、または IBM® LinuxONE III 以降。
- KVM を備えた RHEL 9 以降で実行している少なくとも 1 つの LPAR。

手順

1. システムの RHEL **ORG_ID** および **ACTIVATION_KEY** シェル変数を設定します。
   a. サブスクライブされた RHEL システムを使用する場合は、次のコマンドを実行して、組織 ID と Red Hat Subscription Management (RHSM) のアクティベーションキーを保持するファイルにシェル環境変数を設定します。

```
$ export ORG_ID=$(cat ~/.rh_subscription/orgid)

$ export ACTIVATION_KEY=$(cat ~/.rh_subscription/activation_key)
```

   b. サブスクライブされていない RHEL システムを使用する場合は、次のコマンドを実行して適切なサブスクリプション値を設定します。

```
$ export ORG_ID=<RHEL_ORGID_VALUE>

$ export ACTIVATION_KEY=<RHEL_ACVTIVATION_KEY>
```

2. IBM Z システムにログインし、以下の手順を実行します。
   a. **s390x** RHEL KVM ゲストイメージを Red Hat カスタマーポータル から libvirt ストレージディレクトリーにダウンロードして、libvirt に正しいアクセスを許可します。デフォルトの

ディレクトリーは **/var/lib/libvirt/images** です。このイメージは、関連するバイナリーを含むピア Pod VM イメージを生成するために使用されます。

b. 次のコマンドを実行して、ダウンロードしたイメージの **IMAGE_URL** シェル環境変数を設定します。

```
$ export IMAGE_URL=<location_of_downloaded_KVM_guest_image> 1
```

1 前の手順でダウンロードした KVM ゲストイメージのパスを入力します。

c. 次のコマンドを実行して、ゲスト KVM イメージを登録します。

```
$ export REGISTER_CMD="subscription-manager register --org=${ORG_ID} \
  --activationkey=${ACTIVATION_KEY}"
```

d. 次のコマンドを実行して、ゲスト KVM イメージをカスタマイズします。

```
$ virt-customize -v -x -a ${IMAGE_URL} --run-command "${REGISTER_CMD}"
```

e. 次のコマンドを実行して、イメージのチェックサムを設定します。

```
$ export IMAGE_CHECKSUM=$(sha256sum ${IMAGE_URL} | awk '{ print $1 }')
```

3.3.4.2.1. ピア Pod VM QCOW2 イメージの構築

IBM Z 上でピア Pod を使用して OpenShift Sandboxed Containers を実行するには、ピア Pod VM QCOW2 イメージを構築する必要があります。

手順

1. 次のコマンドを実行して、cloud-api-adaptor リポジトリーのクローンをビルドワークステーションに作成します。

```
$ git clone --single-branch https://github.com/confidential-containers/cloud-api-adaptor.git
```

2. 次のコマンドを実行して、**podvm** ディレクトリーに移動します。

```
$ cd cloud-api-adaptor && git checkout 8577093
```

3. 最終的な QCOW2 イメージの生成元となるビルダーイメージを作成します。

a. サブスクライブされた RHEL システムを使用する場合は、次のコマンドを実行します。

```
$ podman build -t podvm_builder_rhel_s390x \
  --build-arg ARCH="s390x" \
  --build-arg GO_VERSION="1.21.3" \
  --build-arg PROTOC_VERSION="25.1" \
  --build-arg PACKER_VERSION="v1.9.4" \
  --build-arg RUST_VERSION="1.72.0" \
  --build-arg YQ_VERSION="v4.35.1" \
  --build-arg
```

```
YQ_CHECKSUM="sha256:4e6324d08630e7df733894a11830412a43703682d65a76f1fc925aac08268a45" \
  -f podvm/Dockerfile.podvm_builder.rhel .
```

b. サブスクライブされていない RHEL システムを使用する場合は、次のコマンドを実行します。

```
$ podman build -t podvm_builder_rhel_s390x \
  --build-arg ORG_ID=$ORG_ID \
  --build-arg ACTIVATION_KEY=$ACTIVATION_KEY \
  --build-arg ARCH="s390x" \
  --build-arg GO_VERSION="1.21.3" \
  --build-arg PROTOC_VERSION="25.1" \
  --build-arg PACKER_VERSION="v1.9.4" \
  --build-arg RUST_VERSION="1.72.0" \
  --build-arg YQ_VERSION="v4.35.1" \
  --build-arg YQ_CHECKSUM="sha256:4e6324d08630e7df733894a11830412a43703682d65a76f1fc925aac08268a45" \
  -f podvm/Dockerfile.podvm_builder.rhel .
```

4. 次のコマンドを実行して、ピア Pod を実行するために必要なバイナリーを含む中間イメージパッケージを生成します。

```
$ podman build -t podvm_binaries_rhel_s390x \
  --build-arg BUILDER_IMG="podvm_builder_rhel_s390x:latest" \
  --build-arg ARCH=s390x \
  -f podvm/Dockerfile.podvm_binaries.rhel .
```

注記

このプロセスにはかなりの時間がかかることが予想されます。

5. 次のコマンドを実行して、バイナリーを抽出し、ピア Pod QCOW2 イメージを構築します。

```
$ podman build -t podvm_rhel_s390x \
  --build-arg ARCH=s390x \
  --build-arg CLOUD_PROVIDER=libvirt \
  --build-arg BUILDER_IMG="localhost/podvm_builder_rhel_s390x:latest" \
  --build-arg BINARIES_IMG="localhost/podvm_binaries_rhel_s390x:latest" \
  -v ${IMAGE_URL}:/tmp/rhel.qcow2:Z \
  --build-arg IMAGE_URL="/tmp/rhel.qcow2" \
  --build-arg IMAGE_CHECKSUM=${IMAGE_CHECKSUM} \
  -f podvm/Dockerfile.podvm.rhel .
```

6. 次のコマンドを実行して、ピア Pod QCOW2 イメージを選択したディレクトリーに抽出します。

```
$ export IMAGE_OUTPUT_DIR=<image_output_directory> 1

$ mkdir -p $IMAGE_OUTPUT_DIR
```

```
$ podman save podvm_rhel_s390x | tar -xO --no-wildcards-match-slash '*.tar' | tar -x -C
${IMAGE_OUTPUT_DIR}
```

1. 最終的な QCOW イメージを抽出する **image_output_directory** を入力します。

7. ピア Pod QCOW2 イメージを libvirt ボリュームにアップロードします。

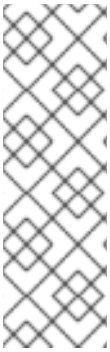
```
$ virsh -c qemu:///system vol-upload \
  --vol $ LIBVIRT_VOL_NAME \
  $IMAGE_OUTPUT_DIR/podvm-*.qcow2 \
  --pool $LIBVIRT_POOL --sparse
```

3.3.4.3. ピア Pod 認証情報の RHEL シークレットの作成

IBM Z のシークレットオブジェクトを作成するときは、特定の環境値を設定する必要があります。シークレットオブジェクトを作成する前に、これらの値の一部を取得できます。ただし、次の値を準備しておく必要があります。

- **LIBVIRT_POOL**
- **LIBVIRT_VOL_NAME**
- **LIBVIRT_URI**

LIBVIRT_URI は、libvirt ネットワークのデフォルトゲートウェイ IP アドレスです。この値を取得するには、libvirt ネットワーク設定を確認してください。



注記

libvirt インストールでデフォルトのブリッジ仮想ネットワークを使用している場合は、次のコマンドを実行して **LIBVIRT_URI** を取得できます。

```
$ virtint=$(bridge_line=$(virsh net-info default | grep Bridge);  echo
"${bridge_line//Bridge:/}" | tr -d [:blank:])

$ LIBVIRT_URI=$( ip -4 addr show $virtint | grep -oP '(?<=inet\s)\d+(\.\d+){3}')
```

手順

1. 次のマニフェストを使用して YAML ファイル **peer-pods-secret.yaml** を作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: peer-pods-secret
  namespace: openshift-sandboxed-containers-operator
type: Opaque
stringData:
  CLOUD_PROVIDER: "libvirt" 1
  LIBVIRT_URI: "<libvirt_gateway_uri>" 2
  LIBVIRT_POOL: "<libvirt_pool>" 3
  LIBVIRT_VOL_NAME: "<libvirt_volume>" 4
```

```
-
```

1. クラウドプロバイダーとして **libvirt** を入力します。
2. 取得した **libvirt_gateway_uri** 値を入力します。
3. 取得した **libvirt_pool** 値を入力します。
4. 取得した **libvirt_volume** 値を入力します。

2. シークレットオブジェクトを作成します。

```
$ oc apply -f peer-pods-secret.yaml
```

シークレットオブジェクトが適用されました。

3.3.4.4. CLI を使用した IBM Z のピア Pod ConfigMap の作成

IBM Z の **ConfigMap** を作成するときは、libvirt プロバイダーを使用する必要があります。

手順

1. 次のマニフェストを使用して YAML ファイル **peer-pods-cm.yaml** を作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: peer-pods-cm
  namespace: openshift-sandboxed-containers-operator
data:
  CLOUD_PROVIDER: "libvirt"
  PROXY_TIMEOUT: "15m"
```

2. **ConfigMap** を適用します。

```
$ oc apply -f peer-pods-cm.yaml
```

ConfigMap が適用されます。

3.3.4.5. CLI を使用した IBM Z の SSH キーのシークレットオブジェクトの作成

IBM Z でピア Pod を使用するには、SSH 鍵ピアを生成し、SSH 鍵のシークレットオブジェクトを作成する必要があります。

手順

1. SSH 鍵を生成します。

```
$ ssh-keygen -f ./id_rsa -N ""
```

2. SSH 公開キーを KVM ホストにコピーします。

```
$ ssh-copy-id -i ./id_rsa.pub <KVM_HOST_ADDRESS> 1
```

1 KVM ホストの IP アドレスを入力します。

3. シークレットオブジェクトを作成します。

```
$ oc create secret generic ssh-key-secret \
    -n openshift-sandboxed-containers-operator \
    --from-file=id_rsa.pub=./id_rsa.pub \
    --from-file=id_rsa=./id_rsa
```

4. SSH キーを削除します。

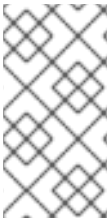
```
$ shred –remove id_rsa.pub id_rsa
```

シークレットオブジェクトが作成されます。**KataConfig** CR を作成すると、IBM Z 上のピア Pod を使用して OpenShift Sandboxed Containers を実行できます。

3.3.5. CLI を使用した KataConfig カスタムリソースの作成

kata-remote をノードに **RuntimeClass** としてインストールするには、**KataConfig** カスタムリソース (CR) を 1 つ作成する必要があります。**KataConfig** CR を作成すると、OpenShift Sandboxed Containers 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。
- CRI-O ランタイムが正しいランタイムハンドラーで設定されていることを確認してください。
- デフォルト設定で **kata-remote** という名前の **RuntimeClass** CR を作成します。これにより、**RuntimeClassName** フィールドの CR を参照して、**kata-remote** 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るようになります。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。

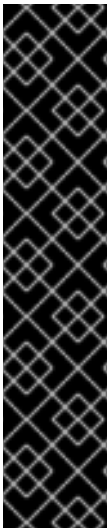


注記

ピア Pod の 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。**kata-remote** を特定のノードにのみ **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 4.15 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 Sandboxed Containers Operator をインストールしている。



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が有効である。
- SSD ではなくハードディスクドライブにデプロイしている。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイしている。
- CPU とネットワークが遅い。

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:
  enablePeerPods: true
  logLevel: info
```

2. (オプション) **kata-remote** を選択したノードにのみ **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。

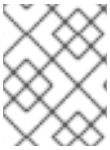
```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  enablePeerPods: true
  logLevel: info
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' (1)
```

(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。

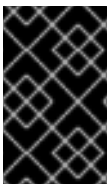
```
$ oc create -f cluster-kataconfig.yaml
```

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



注記

CR を実行すると、VM イメージが作成されます。イメージの作成はクラウドプロバイダーにより行われ、追加のリソースを使用する場合があります。



重要

OpenShift Sandboxed Containers は、**kata-remote** をプライマリーランタイムとしてではなく、クラスター上の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"
```

kataNodes の下のすべてのワーカーが **installs** としてリストされ、理由を指定せずに **InProgress** の条件が **False** の場合は、**kata** がクラスターにインストールされていることを示します。詳細は、「移行のインストールとアンインストール」を参照してください。

関連情報

- ノードでラベルを更新する方法について

3.3.6. CLI を使用したSandboxed Containers 内のピア Pod を含むワークロードのデプロイ

OpenShift Sandboxed Containers は、Kata をプライマリー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。

Sandboxed Containers 内のピア Pod を使用して Pod テンプレート化されたワークロードをデプロイするには、**kata-remote** を **runtimeClassName** としてワークロード YAML ファイルに追加する必要があります。

また、YAML ファイルにアノテーションを追加して、**ConfigMap** で以前に定義したデフォルトのインスタンスサイズまたはタイプを使用してワークロードをデプロイするかどうかも定義する必要があります。インスタンスサイズまたはインスタンスタイプの使用は、クラウドプロバイダーによって異なります。インスタンスのサイズまたはタイプを手動で定義しない場合は、アノテーションを追加して、使用可能なメモリーに基づいて自動インスタンスのサイズまたはタイプの使用を定義する必要があります。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator をインストールしている。
- クラウドプロバイダーに固有のシークレットオブジェクトとピア Pod の **ConfigMap** を作成している。
- **KataConfig** カスタムリソース (CR) を作成している。

手順

1. **runtimeClassName: kata-remote** を Pod でテンプレート化されたオブジェクトに追加します。
   - **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
2. Pod のテンプレート化されたオブジェクトにアノテーションを追加し、特定のインスタンスサイズまたはタイプを使用するか、自動インスタンスサイズまたはタイプを使用するかを定義します。インスタンスサイズは特定のクラウドプロバイダーに使用され、インスタンスタイプは他のクラウドプロバイダーに使用されます。
   - 特定のインスタンスのサイズまたはタイプについては、次のアノテーションを追加します。

     ```
     io.katacontainers.config.hypervisor.machine_type: <instance type/instance size>
     ```

     ワークロードが使用するインスタンスのサイズまたはタイプを定義します。これらのデフォルトのサイズまたはタイプは、ピア Pod の **ConfigMap** を作成するときに事前に定義してあります。そのうちの 1 つを選択してください。
   - 自動インスタンスのサイズまたはタイプについては、次のアノテーションを追加します。

     ```
     io.katacontainers.config.hypervisor.default_vcpus: <vcpus>
     io.katacontainers.config.hypervisor.default_memory: <memory>
     ```

     ワークロードが使用できるメモリーの量を定義します。ワークロードは、使用可能なメモリーの量に基づいて、自動インスタンスサイズまたはタイプで実行されます。

Pod オブジェクトの例

```
apiVersion: v1
kind: Pod
metadata:
  name: hello-openshift
  labels:
    app: hello-openshift
  annotations:
    io.katacontainers.config.hypervisor.machine_type: Standard_DC4as_v5 1
spec:
  runtimeClassName: kata-remote
  containers:
    - name: hello-openshift
      image: quay.io/openshift/origin-hello-openshift
      ports:
        - containerPort: 8888
```

```
  securityContext:
    privileged: false
    allowPrivilegeEscalation: false
    runAsNonRoot: true
    runAsUser: 1001
    capabilities:
      drop:
        - ALL
    seccompProfile:
      type: RuntimeDefault
```

1 この例では、Azure を使用してピア Pod に事前に定義されたインスタンスサイズを使用します。AWS を使用するピア Pod はインスタンスタイプを使用します。

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。

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-remote** の場合、ワークロードはピア Pod を使用して OpenShift Sandboxed Containers 上で実行されます。

3.4. 関連情報

- OpenShift Sandboxed Containers 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。
- 制限されたネットワーク上で切断されたクラスターを使用する場合、OperatorHub にアクセスするには 、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 Sandboxed Containers 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 SANDBOXED CONTAINERS のメトリックについて

OpenShift Sandboxed Containers メトリックにより、管理者はSandboxed Containers 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ック UI でクエリーできます。

OpenShift Sandboxed Containers 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、Sandboxed Containers 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、Sandboxed Containers 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。

ハイパーバイザーメトリック

ハイパーバイザーメトリックには、Sandboxed Containers 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 SANDBOXED CONTAINERS のメトリクスの表示

Web コンソールの **Metrics** ページで、OpenShift Sandboxed Containers のメトリックにアクセスできます。

前提条件

- OpenShift Container Platform 4.15 がインストールされている。
- OpenShift Sandboxed Containers 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

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

4.3. OPENSHIFT SANDBOXED CONTAINERS ダッシュボードの表示

Web コンソールの **Dashboards** ページで、OpenShift Sandboxed Containers ダッシュボードにアクセスできます。

前提条件

- OpenShift Container Platform 4.15 がインストールされている。
- OpenShift Sandboxed Containers 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** リストで **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Sandboxed Containers の総数を表示します。

CPU 使用率 (仮想マシンあたり)
個々のSandboxed Containers 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Sandboxed Containers 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 SANDBOXED CONTAINERS 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 SANDBOXED CONTAINERS 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 Sandboxed Containers 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

- クラスターに OpenShift Container Platform 4.15 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

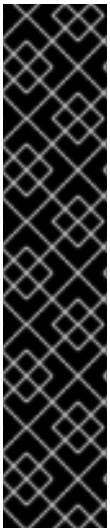
1. **Administrator** パースペクティブから、**Workloads → Pods** に移動します。
2. **Search by name** フィールドを使用して、削除する Pod を検索します。
3. Pod 名をクリックして開きます。
4. **Details** ページで、**Runtime class** に **kata** が表示されていることを確認します。
5. **Actions** メニューをクリックし、**Delete Pod** を選択します。
6. 確認ウィンドウで **Delete** をクリックします。

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。詳細は、Deleting OpenShift sandboxed containers pods 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が有効である。
- 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にデプロイしている。
- CPU とネットワークが遅い。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers 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 **Options** メニュー をクリックし、**Delete KataConfig** 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 Sandboxed Containers Operator のインストール

OpenShift Sandboxed Containers 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

- クラスターに OpenShift Container Platform 4.15 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 Sandboxed Containers Operator を検索します。
3. Operator の **Options** メニュー をクリックし、**Uninstall Operator** 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た OpenShift Sandboxed Containers の namespace の削除

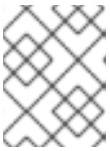
上記のコマンドを実行すると、インストールプロセス前の状態にクラスターが復元されます。**openshift-sandboxed-containers-operator** namespace を削除することで、Operator への namespace アクセスを取り消すことができるようになりました。

前提条件

- クラスターに OpenShift Container Platform 4.15 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 **Options** 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

- クラスターに OpenShift Container Platform 4.15 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 を削除している。
- クラスターから OpenShift Sandboxed Containers Operator を削除している。

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 Sandboxed Containers Pod の削除

OpenShift Sandboxed Containers 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。

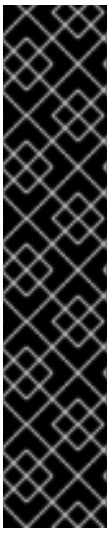
```
$ oc delete pod <pod-name>
```

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

- クラスターに OpenShift Container Platform 4.15 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。

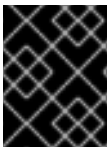
- より多くのワーカーノードを持つ大規模な OpenShift Container Platform デプロイメント。
- BIOS および診断ユーティリティーが有効である。
- SSD ではなくハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイしている。
- CPU とネットワークが遅い。

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 Sandboxed Containers 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Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 Sandboxed Containers 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 を削除している。
- クラスターから OpenShift Sandboxed Containers Operator を削除している。

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

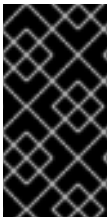
```
Unknown CR KataConfig
```

第6章 OPENSHIFT SANDBOXED CONTAINERS 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 Sandboxed Containers 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 **Sandboxed Containers** 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 Sandboxed Containers Operator を手動で設定するか、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 Sandboxed Containers Operator のアップグレードの詳細は、インストール済み Operator の更新 を参照してください。

6.3. OPENSHIFT SANDBOXED CONTAINERS モニター POD のアップグレード

OpenShift Sandboxed Containers 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

- クラスターに OpenShift Container Platform 4.15 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

- クラスターに OpenShift Container Platform 4.15 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 SANDBOXED CONTAINERS データの収集

OpenShift Sandboxed Containers 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 SANDBOXED CONTAINERS 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 ツールの使用

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/openshift-sandboxed-containers/osc-must-gather-rhel9:1.5.0
  ```

- 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
...
```

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/openshift-sandboxed-containers/osc-must-gather-rhel9:1.5.0
```

7.2. OPENSHIFT SANDBOXED CONTAINERS 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 Sandboxed Containers に関連付けられます。

- OpenShift Sandboxed Containers リソースに属するすべての namespace とその子オブジェクト
- すべての OpenShift Sandboxed Containers のカスタムリソース定義 (CRD)

次の OpenShift Sandboxed Containers 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 SANDBOXED CONTAINERS のデバッグログの有効化

クラスター管理者は、OpenShift Sandboxed Containers のより詳細なレベルのログを収集できます。**KataConfig** CR の **logLevel** フィールドを変更することで、ロギングを強化することもできます。これにより、OpenShift Sandboxed Containers を実行しているワーカーノードの CRI-O ランタイムの **log_level** が変更されます。

手順

1. 既存の **KataConfig** CR の **logLevel** フィールドを **debug** に変更します。

```
$ oc patch kataconfig <name_of_kataconfig_file> --type merge --patch '{"spec":{"logLevel":"debug"}}'
```

注記

このコマンドを実行するときは、**KataConfig** CR の名前を参照します。これは、OpenShift Sandboxed Containers のセットアップ時に CR を作成するために使用した名前です。

検証

1. すべてのワーカーノードが更新されて **UPDATED** フィールドが **True** になるまで、**kata-oc** マシン設定プールを監視します。

```
$ oc get mcp kata-oc
```

出力例

```
NAME     CONFIG              UPDATED  UPDATING  DEGRADED  MACHINECOUNT
READYMACHINECOUNT  UPDATEDMACHINECOUNT  DEGRADEDMACHINECOUNT
AGE
kata-oc  rendered-kata-oc-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 Sandboxed Containers のデバッグログの表示

クラスター管理者は、OpenShift Sandboxed Containers 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 Sandboxed Containers 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 は警告ログとエラーログのみを生成します。これらの警告とエラーは、追加の **qemuPid** フィールドとともに Kata ランタイムログと CRI-O ログの両方でノードジャーナルに出力されます。

QEMU ログの例

```
Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.587116986Z"
level=info msg="Start logging QEMU (qemuPid=2241693)" name=containerd-shim-v2 pid=2241647
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
source=virtcontainers/hypervisor subsystem=qemu

Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.607339014Z"
level=error msg="qemu-kvm: -machine q35,accel=kvm,kernel_irqchip=split,foo: Expected '=' after
parameter 'foo'" name=containerd-shim-v2 pid=2241647 qemuPid=2241693
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
source=virtcontainers/hypervisor subsystem=qemu

Mar 11 11:57:28 openshift-worker-0 kata[2241647]: time="2023-03-11T11:57:28.60890737Z"
level=info msg="Stop logging QEMU (qemuPid=2241693)" name=containerd-shim-v2 pid=2241647
sandbox=d1d4d68efc35e5ccb4331af73da459c13f46269b512774aa6bde7da34db48987
source=virtcontainers/hypervisor subsystem=qemu
```

Kata ランタイムは、QEMU が起動すると **Start logging QEMU** を出力し、QEMU が停止すると **Stop Logging QEMU** を出力します。エラーは、**qemuPid** フィールドが含まれる、これら 2 つのログメッセージの間に表示されます。QEMU からの実際のエラーメッセージは赤色で表示されます。

QEMU ゲストのコンソールはノードジャーナルにも出力されます。ゲストコンソールログを 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
```

- QEMU 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 -g "qemuPid=\d+"
```

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。