



OpenShift Container Platform 4.10

OpenShift 向けのサンドボックスコンテナのサ ポート

OpenShift サンドボックスコンテナガイド

OpenShift Container Platform 4.10 OpenShift 向けのサンドボックスコンテナのサポート

OpenShift サンドボックスコンテナガイド

法律上の通知

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

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

目次

第1章 {SANDBOXED-CONTAINERS-FIRST} {SANDBOXED-CONTAINERS-VERSION} リリースノート	3
1.1. このリリースについて	3
1.2. 新機能および拡張機能	3
1.3. バグ修正	4
1.4. 既知の問題	4
1.5. エラータの非同期更新	6
第2章 OPENSIFT SANDBOXED CONTAINERS について	8
2.1. OPENSIFT SANDBOXED CONTAINERS がサポートするプラットフォーム	9
2.2. OPENSIFT SANDBOXED CONTAINERS の一般的な用語	9
2.3. OPENSIFT サンドボックスコンテナのワークロード管理	10
2.4. コンプライアンスおよびリスク管理について	11
第3章 OPENSIFT SANDBOXED CONTAINERS ワークロードのデプロイ	12
3.1. 前提条件	12
3.2. WEB コンソールを使用した OPENSIFT SANDBOXED CONTAINERS ワークロードのデプロイ	16
3.3. CLI を使用した OPENSIFT サンドボックスコンテナワークロードのデプロイ	20
3.4. 関連情報	26
第4章 OPENSIFT サンドボックスコンテナのモニタリング	27
4.1. OPENSIFT SANDBOXED CONTAINERS のメトリックについて	27
4.2. OPENSIFT SANDBOXED CONTAINERS のメトリクスの表示	27
4.3. OPENSIFT SANDBOXED CONTAINERS ダッシュボードの表示	28
4.4. 関連情報	29
第5章 OPENSIFT SANDBOXED CONTAINERS のアンインストール	30
5.1. WEB コンソールを使用した OPENSIFT SANDBOXED CONTAINERS のアンインストール	30
5.2. CLI を使用した OPENSIFT サンドボックスコンテナのアンインストール	33
第6章 OPENSIFT SANDBOXED CONTAINERS のアップグレード	37
6.1. OPENSIFT サンドボックスコンテナリソースのアップグレード	37
6.2. OPENSIFT サンドボックスコンテナ OPERATOR のアップグレード	37
6.3. OPENSIFT SANDBOXED CONTAINERS モニター POD のアップグレード	37
第7章 OPENSIFT SANDBOXED CONTAINERS データの収集	40
7.1. RED HAT サポート用の OPENSIFT SANDBOXED CONTAINERS データの収集	40
7.2. OPENSIFT SANDBOXED CONTAINERS のログデータについて	41
7.3. OPENSIFT SANDBOXED CONTAINERS のデバッグログの有効化	41
7.4. 関連情報	43

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

1.1. このリリースについて

これらのリリースノートは、Red Hat OpenShift Container Platform 4.10 とともに OpenShift サンドボックスコンテナ 1.2 の開発を追跡します。

この製品は、以前は OpenShift Container Platform 4.9 でテクノロジープレビュー製品としてリリースされていましたが、OpenShift Container Platform 4.10 でデフォルトで有効になり、一般提供されています。

1.2. 新機能および拡張機能

今回のリリースでは、OpenShift サンドボックスコンテナに次の機能が追加されています。

1.2.1. Kata 固有の指標とダッシュボード

OpenShift サンドボックスコンテナ Operator は、**osc-monitor** デモンセットをデプロイするようになりました。これにより、ハイパーバイザーやゲスト OS インスタンスに関するメトリックなど、サンドボックス化されたコンテナで実行されているワークロードに固有のメトリックの収集が可能になります。さらに、事前設定されたダッシュボードは、クラスターで有効になっている軽量 VM の総数、VM ごとの CPU とメモリーの消費量など、OpenShift サンドボックスコンテナコンポーネントに関する情報をもとに理解できるようになります。すべてのメトリックとダッシュボードは、Web コンソールで利用できます。詳細は、[OpenShift サンドボックスコンテナのモニタリング](#) を参照してください。

1.2.2. 拡張ロギング

管理者は、OpenShift サンドボックスコンテナランタイムコンポーネントの拡張ログを収集できるようになりました。CRI-O ログレベルが **debug** に設定されている場合、拡張ログを使用できます。これらのログは、must-gather ツールによって収集されるか、ノードジャーナルで表示できます。詳細は [OpenShift サンドボックスコンテナのデバッグログの有効化](#) を参照してください。

1.2.3. OpenShift サンドボックスコンテナを実行するためのノードの適格性確認

管理者は、OpenShift サンドボックスコンテナを実行するためのクラスターノードの適格性を確認できるようになりました。この機能は、Node Feature Discovery (NFD) Operator を使用してノード機能を検出します。適格なノードは **feature.node.kubernetes.io/runtime.kata** でラベル付けされ、OpenShift サンドボックスコンテナ Operator はこのラベルを使用して、インストールの候補ノードを選択します。

管理者は、この機能を使用するために NFD Operator をデプロイし、特定の **NodeFeatureDiscovery** カスタムリソースを作成して、**KataConfig** カスタムリソースの作成時に **checkNodeEligibility** を有効にする必要があります。詳細は、[OpenShift サンドボックスコンテナを実行するためのクラスターノードの適格性確認](#) を参照してください。

1.2.4. OpenShift サンドボックスコンテナの OpenShift 仮想化との互換性

VM が正しく設定されている場合、ユーザーは OpenShift Virtualization を使用してクラスターで OpenShift サンドボックスコンテナを実行できるようになりました。詳細は、[OpenShift Virtualization での OpenShift サンドボックスコンテナの使用](#) を参照してください。

1.2.5. AWS ベアメタルでの OpenShift サンドボックスコンテナの可用性 (テクノロジープレビュー)

AWS ベアメタルクラスターに OpenShift サンドボックスコンテナをインストールできるようになりました。この機能はテクノロジープレビューであるため、完全にはサポートされていません。詳細は、[OpenShift サンドボックスコンテナについて](#) を参照してください。

1.3. バグ修正

- 以前のリリースでは、カーネルモジュールが見つからないため、サンドボックス化されたコンテナでループデバイスを使用できませんでした。今回のリリースでは、これらのカーネルモジュールがパッケージに含まれています。(KATA-1334)
- 以前は、**Kata** ランタイムがインストールされているノードを追跡するために Operator によって作成された **MachineConfigPool (MCP)** オブジェクトは、**KataConfig** カスタムリソース (CR) の削除時に自動的に削除されませんでした。今回のリリースでは、**KataConfig** CR を削除すると、**kata-oc MCP** オブジェクトが削除されます。(KATA-1184)
- 以前のリリースでは、**kataConfigPoolSelector** フィールドを作成して変更した場合に、OpenShift サンドボックスコンテナ Operator には変更が適用されませんでした。今回のリリースでは、Operator はカスタムリソース定義の **kataConfigPoolSelector** フィールドへの変更に作用し、変更に合わせて、ノードへのランタイムのインストールを適応させます。(KATA-1190)
- 以前のリリースは、**Sourcelmage** フィールドが Web コンソールに表示され、フィールドを使用してもインストールには影響しませんでした。今回のリリースでは、Web コンソールから **KataConfig** CR を作成するときに、未使用の **Sourcelmage** フィールドが表示されなくなりました。(KATA-1015)

1.4. 既知の問題

- OpenShift サンドボックスコンテナを使用している場合、OpenShift Container Platform クラスターの **hostPath** ボリュームからマウントされたファイルまたはディレクトリーにアクセスすると、SELinux 拒否を受け取ることがあります。特権 Sandboxed Container は SELinux チェックを無効にしないため、特権 Sandboxed Container を実行している場合でも、このように拒否される可能性があります。

ホストで SELinux ポリシーに準拠すると、ホストファイルシステムとサンドボックスのワークロードを完全に分離して、**virtiofsd** デーモンまたは QEMU で発生する可能性のあるセキュリティの脆弱性に対してより強力的に保護します。

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

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

- Machine Config Operator (MCO) Pod が **CrashLoopBackOff** 状態に遷移し、Pod の **openshift.io/scc** アノテーションにデフォルト値の **hostmount-anyuid** ではなく **sandboxed-containers-operator-scc** が表示される問題が発生する可能性があります。

この場合は、**sandboxed-containers-operator-scc** SCC の **seLinuxOptions** ストラテジーを一時的に制限の少ない **RunAsAny** に変更し、承認プロセスが **hostmount-anyuid** SCC よりも優先しないようにします。

1. 以下のコマンドを実行して **seLinuxOptions** ストラテジーを変更します。

```
$ oc patch scc sandboxed-containers-operator-scc --type=merge --patch
'{"seLinuxContext":{"type": "RunAsAny"}}'
```

2. 以下のコマンドを実行して MCO Pod を再起動します。

```
$ oc scale deployments/machine-config-operator -n openshift-machine-config-operator --
replicas=0
```

```
$ oc scale deployments/machine-config-operator -n openshift-machine-config-operator --
replicas=1
```

3. 以下のコマンドを実行して、**sandboxed-containers-operator-scc** の **seLinuxOptions** ストラテジーを **MustRunAs** の元の値に戻します。

```
$ oc patch scc sandboxed-containers-operator-scc --type=merge --patch
'{"seLinuxContext":{"type": "MustRunAs"}}'
```

4. 以下のコマンドを実行して、**hostmount-anyuid** SCC が MCO Pod に適用されていることを確認します。

```
$ oc get pods -n openshift-machine-config-operator -l k8s-app=machine-config-operator -
o yaml | grep scc
openshift.io/scc: hostmount-anyuid
```

([BZ#2057545](#))

- OpenShift サンドボックスコンテナ Operator Pod が、コンテナ CPU リソースの制限を使用して、Pod に利用できる CPU の数量を増やした場合に、CPU の数が要求した数よりも少なくなる可能性があります。機能がコンテナ内で使用可能な場合、**oc rsh <pod>** を使用して **lscpu** コマンドを実行して、CPU リソースを診断できます。

```
$ lscpu
```

出力例

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

使用可能なオフライン CPU のリストは実行ごとに変更され、予測できません。

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

```

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

```

(KATA-1376)

- ランタイムインストールの進行状況は、**kataConfig** CR の **status** セクションに表示されます。ただし、次の条件がすべて当てはまる場合、進行状況は表示されません。
 - クラスタには、メンバーのないマシン設定プール **ワーカー** があります (**machinecount=0**)。
 - インストールするノードを選択するための **kataConfigPoolSelector** が指定されていません。

この場合、ノードがマスターロールとワーカーロールの両方を持つコンバインドクラスタであると Operator が想定するため、マスターノードでインストールが開始されません。**kataConfig** CR の **status** セクションは、インストール中に更新されません。(KATA-1017)

- **KataConfig** CR を作成し、**openshift-sandboxed-containers-operator** namespace で Pod のステータスを確認すると、モニター Pod が何度も再起動されていることが分かります。モニター Pod は、**sandboxed-containers** 拡張機能のインストールの一部としてインストールされる特定の SELinux ポリシーを使用します。モニター Pod はすぐに作成されますが、SELinux ポリシーはまだ利用できないため、Pod 作成エラーが発生し、Pod が再起動されます。拡張機能が正常にインストールされると、SELinux ポリシーが使用可能になり、モニター Pod は **Running** 状態に移行します。これは、OpenShift サンドボックスコンテナの Operator 機能には影響しません。(KATA-1338)

1.5. エラータの非同期更新

OpenShift サンドボックスコンテナ 1.2 のセキュリティ、バグ修正、および機能強化の更新は、Red Hat Network を通じて非同期エラータとして発表されます。OpenShift Container Platform 4.10 のすべてのエラータは [Red Hat カスタマーポータルから入手できます](#)。非同期エラータについては、[OpenShift Container Platform ライフサイクル](#) を参照してください。

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



注記

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

以下のセクションは、これからも継続して更新され、今後の OpenShift sandboxed containers 1.2 の非同期リリースで発表されるエラータの拡張機能およびバグ修正に関する情報を追加していきます。

1.5.1. RHSA-2022:1508 - OpenShift サンドボックスコンテナ 1.2.2 バグ修正アドバイザリー

発行日: 2022-07-26

OpenShift サンドボックスコンテナリリース 1.2.2 が利用可能になりました。このアドバイザリーには、バグ修正を含む OpenShift サンドボックスコンテナの更新が含まれています。

更新に含まれるバグ修正の一覧は、[RHSA-2022:5725](#) アドバイザリーに記載されています。

1.5.2. RHSA-2022:1508 - OpenShift サンドボックスコンテナ 1.2.1 バグ修正アドバイザリー

発行日: 2022-04-21

OpenShift サンドボックスコンテナリリース 1.2.1 が利用可能になりました。このアドバイザリーには、バグ修正を含む OpenShift サンドボックスコンテナの更新が含まれています。

更新に含まれるバグ修正の一覧は、[RHSA-2022:1508](#) アドバイザリーに記載されています。

1.5.3. RHSA-2022:0855 - OpenShift サンドボックスコンテナ 1.2.0 イメージのリリース、セキュリティー更新、バグ修正、および機能強化のアドバイザリー

発行日: 2022-03-14

OpenShift サンドボックスコンテナリリース 1.2.0 が利用可能になりました。このアドバイザリーには、機能強化、セキュリティー更新、およびバグ修正を含む OpenShift サンドボックスコンテナの更新が含まれています。

更新に含まれるバグ修正の一覧は、[RHSA-2022:0855](#) アドバイザリーに記載されています。

第2章 OPENSIFT SANDBOXED CONTAINERS について

OpenShift Container Platform の OpenShift サンドボックスコンテナがサポートされることで、追加のオプションランタイムとして、Kata Container を実行するビルドインサポートが追加されます。新しいランタイムは、専用の仮想マシン (VM) でコンテナをサポートし、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は、他のデバイスにアクセスすることなく、そのデバイスを完全に制御できます。

2.1. OPENSIFT 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 サンドボックスコンテナ 1.2 は、Red Hat Enterprise Linux CoreOS (RHCOS) 8.6 で実行されます。

OpenShift サンドボックスコンテナ 1.2 は、OpenShift Container Platform 4.11 と互換性があります。



重要

AWS ベアメタルインスタンスへの OpenShift サンドボックスコンテナのインストールは、テクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

2.2. OPENSIFT 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、コンテナのデプロイが可能なりソースを表します。コンテナは Pod 内で実行され、Pod を使用して複数のコンテナ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。

2.3. OPENSIFT サンドボックスコンテナ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。

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

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

OpenShift サンドボックスコンテナ Operator は、2つのコンテナイメージとして [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つ目のコンテナイメージには、**KataConfig** リソースを監視および管理するための実際のコントローラーが含まれています。

2.3.2. RHCOS 拡張機能

OpenShift 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。Red Hat Enterprise Linux CoreOS (RHCOS) 拡張機能は、オプションの OpenShift Container Platform ソフトウェアをインストールするメカニズムです。OpenShift 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](#)

2.3.3. 仮想化および 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](#)

2.4. コンプライアンスおよびリスク管理について

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 サンドボックスコンテナコンポーネント、仮想マシン、および VM イメージは、FIPS に準拠するように調整されます。

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



重要

クラスターで FIPS モードを有効にするには、FIPS モードで動作するように設定された Red Hat Enterprise Linux (RHEL) コンピューターからインストールプログラムを実行する必要があります。RHEL での FIPS モードの設定の詳細は、[FIPS モードでのシステムのインストール](#) を参照してください。プロセス暗号化ライブラリーでの FIPS 検証済みまたはモジュールの使用は、**x86_64** アーキテクチャーでの OpenShift Container Platform デプロイメントでのみサポートされます。

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

第3章 OPENSIFT 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 サンドボックスコンテナ Operator をインストールできます。OpenShift サンドボックスコンテナ Operator をインストールする前に、OpenShift Container Platform クラスタを準備する必要があります。

3.1. 前提条件

OpenShift サンドボックスコンテナをインストールする前に、OpenShift Container Platform クラスタ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。



重要

AWS ベアメタルインスタンスへの OpenShift サンドボックスコンテナのインストールは、テクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

3.1.1. OpenShift Sandboxed Containers のリソース要件

OpenShift サンドボックスコンテナを使用すると、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、コンテナワークロードおよびこれらのコンテナ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 バッファもメモリーを消費します。

コンテナ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のキャッシュデータ	~300Mi* [1]

ファイルバッファが表示され、このバッファは以下の複数の場所に考慮されます。

- ファイルバッファ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。

Example

```
$ 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のベアメタルへのインストール](#)

3.1.2. クラスタノードが 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、クラスタ内のノードが Kata コンテナ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、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:
              - cpuld: ["SSE4", "VMX"]
                loadedKMod: ["kvm", "kvm_intel"]
              - cpuld: ["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	v1.23.3+e419edf
compute-2.example.com	Ready	worker	4h35m	v1.23.3+e419edf

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。

3.2. WEB コンソールを使用した OPENSIFT 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。

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 Operator をインストールできます。

前提条件

- OpenShift Container Platform 4.10 がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Operators** → **OperatorHub** に移動します。
2. **Filter by keyword** フィールドに **OpenShift sandboxed containers** と入力します。
3. **OpenShift sandboxed containers** タイルを選択します。
4. Operator についての情報を確認してから、**Install** をクリックします。
5. **Install Operator** ページで以下を行います。
 - a. 選択可能な **Update Channel** オプションの一覧から **stable-1.2** 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のアクティベーション。
- SSD ではなく、ハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイします。
- 遅い CPU またはネットワーク。

前提条件

- クラスターに OpenShift Container Platform 4.10 をインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift 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、**YAML view** 経由で **KataConfig** CR を設定することを選択します。
5. 次のマニフェストをコピーして **YAML view** に貼り付けます。

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

```

name: cluster-kataconfig
spec:
  kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-
  rhel8:1.2.0

```

選択したノードにのみ **kata** を **RuntimeClass** としてインストールする場合は、マニフェストにラベルを含めます。

```

apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-
  rhel8:1.2.0
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' ❶

```

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

6. **Create** をクリックします。

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



重要

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

検証

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

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

3.2.3. 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

- クラスタに OpenShift Container Platform 4.10 をインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスタにアクセスできる。
- 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。

3.3. CLI を使用した OPENSIFT サンドボックスコンテナ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

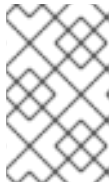
作成する必要があります。Sandboxed Containers にワークロードをデプロイする準備ができたなら、ワークロード YAML ファイルに **runtimeClassName** として **kata** を追加する必要があります。

3.3.1. CLI を使用した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 Operator をインストールできます。

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。
- **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-1.2"
  installPlanApproval: Automatic
  name: sandboxed-containers-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: sandboxed-containers-operator.v1.2.2
```

- 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 1.2.2 1.2.1
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 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 サンドボックス化されたコンテナ Operator がトリガーされ、以下が実行されます。

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



注記

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

前提条件

- クラスタに OpenShift Container Platform 4.10 をインストールされている。
- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスタにアクセスできる。
- 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のアクティベーション。
- SSD ではなく、ハードドライブへのデプロイメント。
- 仮想ノードではなく、ベアメタルなどの物理ノードにデプロイします。
- 遅い CPU またはネットワーク。

手順

1. 次のマニフェストで YAML ファイルを作成します。

```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-rhel8:1.2.0
```

- (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。

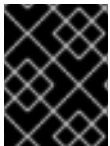
```
apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: cluster-kataconfig
spec:
  kataMonitorImage: registry.redhat.io/openshift-sandboxed-containers/osc-monitor-
  rhel8:1.2.0
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' ❶
```

- ❶ **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。

- KataConfig** リソースを作成します。

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

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



重要

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

検証

- インストールの進捗を監視します。

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

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

関連情報

- [ノードでラベルを更新する方法について](#)

3.3.3. CLI を使用した 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 4.10 をインストールされている。

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift 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 サンドボックスコンテナで実行されています。

3.4. 関連情報

- OpenShift サンドボックスコンテナ Operator は、制限されたネットワーク環境でサポートされます。詳細は、[制限されたネットワークでの Operator Lifecycle Manager の使用](#) を参照してください。
- ネットワークが制限された環境で切断されたクラスターを使用する場合に、Red Hat が提供する OperatorHub にアクセスできるようにするには、[Operator Lifecycle Manager でプロキシサポート](#) を設定する必要があります。プロキシを使用すると、クラスターは OpenShift Sandboxed Containers Operator を取得できます。

第4章 OPENSIFT サンドボックスコンテナのモニタリング

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. OPENSIFT 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. OPENSIFT 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 4.10 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

- メトリクス表示の PromQL クエリーを作成する方法は、[メトリクスのクエリー](#) を参照してください。

4.3. OPENSIFT 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 4.10 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

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

第5章 OPENSIFT 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 OPENSIFT 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 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 Container Platform 4.10 がクラスターに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、[OpenShift サンドボックスコンテナ Pod の削除](#) を参照してください。

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

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

重要

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

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

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。
- **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

- OpenShift Container Platform 4.10 がクラスターに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

- OpenShift Container Platform 4.10 がクラスターに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

- OpenShift Container Platform 4.10 がクラスターに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。

- クラスタから **KataConfig** CR を削除している。
- クラスタから 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 OPENSIFT サンドボックスコンテナ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 (**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) をクラスタから削除およびアンインストールできます。デプロイのサイズにもよりますが、通常、削除には 10 ~ 40 分かかります。

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



重要

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

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

手順

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

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

OpenShift 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 (**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 Lifecycle 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。

- クラスタから **KataConfig** CR を削除している。
- クラスタから 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

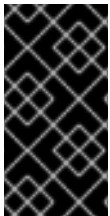
```
Unknown CR KataConfig
```


第6章 OPENSIFT SANDBOXED CONTAINERS のアップグレード

OpenShift サンドボックスコンテナ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 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 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 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 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. OPENSIFT サンドボックスコンテナ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. OPENSIFT サンドボックスコンテナ 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 サンドボックスコンテナ 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. OPENSIFT 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

- OpenShift Container Platform 4.10 がクラスターにインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

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

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

6. **Save** をクリックします。

6.3.2. CLI を使用した監視 Pod のアップグレード

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

前提条件

- OpenShift Container Platform 4.10 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。
- **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 OPENSIFT 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 OPENSIFT 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の両方の診断情報を提供してください。

7.1.1. must-gather ツールについて

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

- リソース定義
- サービスログ

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

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

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

```
$ oc adm must-gather --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.10.0
```

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

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



注記

ファイルのサイズを小さくするために、監査ログはデフォルトの情報セットの一部として収集されません。

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

以下に例を示します。

```

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

```

must-gather を使用して OpenShift Sandboxed Containers データを収集するには、OpenShift Sandboxed Containers イメージを指定する必要があります。

```
--image=registry.redhat.io/openshift-sandboxed-containers/osc-must-gather-rhel8:1.2.0
```

7.2. OPENSIFT 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. OPENSIFT SANDBOXED CONTAINERS のデバッグログの有効化

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

手順

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

```

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

```

```

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

```

- 1 変更する必要があるマシン設定プールのラベルを指定します。

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

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



注記

上記のファイル名は提案です。別の名前を使用してこのファイルを作成できます。

3. CR が作成されたことを確認します。

```
$ oc get ctrcfg
```

出力例

```

NAME      AGE
crio-debug 3m19s

```

検証

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

```
$ oc get mcp worker
```

出力例

```

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

```

2. CRI-O で **log_level** が更新されたことを確認します。
 - a. マシン設定プールのノードに対して **oc debug** セッションを開き、**chroot /host** を実行します。

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

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

b. **crio.conf** ファイルの変更を確認します。

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

出力例

```
log_level = "debug"
```

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

前提条件

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

手順

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

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

- kata ランタイムログを確認するには、次を実行します。

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

- virtiofsd ログを確認するには、次を実行します。

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

7.4. 関連情報

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

