



OpenShift Container Platform 4.12

OpenShift 的沙盒容器的支持

OpenShift 沙盒容器指南

法律通告

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.

摘要

OpenShift 沙盒容器支持 OpenShift Container Platform 为用户提供对将 Kata Containers 作为额外可选运行时运行的内置支持。

目录

第 1 章 {SANDBOXED-CONTAINERS-FIRST} {SANDBOXED-CONTAINERS-VERSION} 发行注记	3
1.1. 关于此版本	3
1.2. 新功能及功能增强	3
1.3. 更新	3
1.4. 程序错误修复	3
1.5. 已知问题	4
1.6. 异步勘误更新	5
第 2 章 了解 OPENSIFT 沙盒容器	7
2.1. OPENSIFT 沙盒容器支持的平台	7
2.2. OPENSIFT 沙盒容器常用术语	8
2.3. OPENSIFT 沙盒容器工作负载管理	8
2.4. 了解合规性及风险管理	9
第 3 章 部署 OPENSIFT 沙盒容器工作负载	10
3.1. 先决条件	10
3.2. 使用 WEB 控制台部署 OPENSIFT 沙盒容器工作负载	14
3.3. 使用 CLI 部署 OPENSIFT 沙盒容器工作负载	17
3.4. 其他资源	23
第 4 章 监控 OPENSIFT 沙盒容器	24
4.1. 关于 OPENSIFT 沙盒容器指标	24
4.2. 查看 OPENSIFT 沙盒容器的指标	24
4.3. 查看 OPENSIFT 沙盒容器仪表盘	25
4.4. 其他资源	25
第 5 章 卸载 OPENSIFT 沙盒容器	26
5.1. 使用 WEB 控制台卸载 OPENSIFT 沙盒容器	26
5.2. 使用 CLI 卸载 OPENSIFT 沙盒容器	29
第 6 章 升级 OPENSIFT 沙盒容器	33
6.1. 升级 OPENSIFT 沙盒容器资源	33
6.2. 升级 OPENSIFT 沙盒容器 OPERATOR	33
6.3. 升级 OPENSIFT 沙盒容器监控 POD	33
第 7 章 收集 OPENSIFT 沙盒容器数据	35
7.1. 为红帽支持收集 OPENSIFT 沙盒容器数据	35
7.2. 关于 OPENSIFT 沙盒容器日志数据	36
7.3. 为 OPENSIFT 沙盒容器启用调试日志	36
7.4. 其他资源	38

第 1 章 {SANDBOXED-CONTAINERS-FIRST} {SANDBOXED-CONTAINERS-VERSION} 发行注记

1.1. 关于此版本

本发行注记介绍了 OpenShift 沙盒容器 1.3 和 Red Hat OpenShift Container Platform 4.12 的开发。

从 OpenShift Container Platform 4.10 开始，此产品被完全支持并启用。

1.2. 新功能及功能增强

1.2.1. 指标列表中的容器 ID

带有相关沙盒容器的 ID 的 `sandbox_id` 现在会出现在 web 控制台的 **Metrics** 页面的指标列表中。

另外，`kata-monitor` 进程现在为 kata-specific metrics 添加三个新标签：`cri_uid`、`cri_name` 和 `cri_namespace`。这些标签启用特定于 kata 的指标，以便与对应的 kubernetes 工作负载相关。

如需有关 kata 特定指标的更多信息，请参阅[关于 OpenShift 沙盒容器指标](#)。

1.2.2. OpenShift 沙盒容器在 AWS 裸机上可用性

在以前的版本中，AWS 裸机上的 OpenShift 沙盒容器可用性为技术预览。在这个版本中，完全支持在 AWS 裸机集群中安装 OpenShift 沙盒容器。

1.2.3. 支持单节点 OpenShift 上的 OpenShift 沙盒容器

当 OpenShift 沙盒容器 Operator 由 Red Hat Advanced Cluster Management (RHACM) 安装时，OpenShift 沙盒容器现在可以在单节点 OpenShift 集群上工作。

1.3. 更新

在创建 **KataConfig** 自定义资源时，不再需要 `kataMonitorImage`。这个版本使用 OpenShift 沙盒容器 1.3.2 实现，在所有 OpenShift 沙盒容器 Operator 版本间向后兼容。

1.4. 程序错误修复

- 在以前的版本中，当在 OpenShift Container Platform 4.10 上安装 OpenShift 沙盒容器时，控制器管理器 POD 无法启动并显示以下错误：

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

因此，无法创建 **KataConfig** CR，OpenShift 沙盒容器就无法运行。

在这个版本中，管理器容器镜像已更新为使用数字用户 id (499)。(KATA-1824)

- 在以前的版本中，当创建 **KataConfig** CR 并在 `openshift-sandboxed-containers-operator` 命名空间中观察 pod 状态时，会显示大量监控 pod 的重启。monitor pod 使用作为 `sandboxed-containers` 扩展安装的一部分安装的特定 SELinux 策略。monitor pod 会立即创建。但是，SELinux 策略还不可用，这会导致 pod 创建错误，然后是 pod 重启。

在这个版本中，在创建 monitor pod 时 SELinux 策略可用，而 monitor pod 会立即过渡到 **Running** 状态。(KATA-1338)

- 在以前的版本中，OpenShift 沙盒容器在启动时部署了安全性上下文约束 (SCC)，它会强制使用 Machine Config Operator (MCO) pod 上不可用的自定义 SELinux 策略。这会导致 MCO pod 更改为 **CrashLoopBackOff** 状态和集群升级失败。在这个版本中，OpenShift 沙盒容器在创建 **KataConfig** CR 时部署 SCC，不再使用自定义 SELinux 策略强制执行。(KATA-1373)
- 在以前的版本中，当卸载 OpenShift 沙盒容器 Operator 时，**sandboxed-containers-operator-scc** 自定义资源不会被删除。在这个版本中，在卸载 OpenShift 沙盒容器 Operator 时，**sandboxed-containers-operator-scc** 自定义资源会被删除。(KATA-1569)

1.5. 已知问题

- 在使用 OpenShift 沙盒容器时，如果您在容器级别上设置了 SELinux Multi-Category Security (MCS) 标签，pod 无法启动并显示以下错误：

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

在容器级别设置的 MCS 标签不会应用到 virtiofsd。**kata** 运行时在创建虚拟机时无法访问容器的安全上下文。

这意味着 virtiofsd 没有使用适当的 SELinux 标签运行，且无法代表虚拟机中的容器访问主机文件；所有容器都可以访问虚拟机中的所有文件。

(KATA-1875)

- 如果使用 OpenShift 沙盒容器，您可能会在访问 OpenShift Container Platform 集群中从 **hostPath** 卷挂载的文件或目录时收到 SELinux 拒绝。即使运行特权沙盒容器，这些拒绝也会发生，因为特权沙盒容器不会禁用 SELinux 检查。
在主机中遵循 SELinux 策略可保证主机文件系统默认与沙盒工作负载完全隔离。这还对 **virtiofsd** 守护进程或 QEMU 中潜在的安全漏洞提供更强的保护。

如果挂载的文件或目录在主机上没有特定的 SELinux 要求，您可以使用本地持久性卷作为替代方案。文件会自动重新标记为 **container_file_t**，遵循 SELinux 容器运行时。如需更多信息，请参阅[使用本地卷的持久性存储](#)。

挂载文件或目录时，自动重新标记不是选项，则主机上应该具有特定的 SELinux 标签。相反，您可以在主机上设置自定义 SELinux 规则，以允许 **virtiofsd** 守护进程访问这些特定标签。

(BZ#1904609)

- 一些 OpenShift 沙盒容器 Operator pod 使用容器 CPU 资源限制来增加 pod 的可用 CPU 数量。这些 pod 可能会收到比请求的 CPU 少。如果功能在容器内可用，您可以使用 **oc rsh <pod>** 访问 pod 并运行 **lscpu** 命令诊断 CPU 资源问题：

```
$ lscpu
```

输出示例

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

离线 CPU 列表可能会不可预测地从 run 改为 run。

作为临时解决方案，您可以使用 pod 注解来请求额外 CPU 而不是设置 CPU 限值。使用 pod 注解的 CPU 请求不受此问题的影响，因为处理器分配方法不同。以下注解必须添加到 pod 的元数据中，而不是设置 CPU 限制：

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

(KATA-1376)

- 运行时安装的进度显示在 **kataConfig** 自定义资源 (CR) 的 **status** 部分中。但是，如果所有以下条件都满足，则不会显示进度：
 - 没有定义 worker 节点。您可以运行 **oc get machineconfigpool** 来检查机器配置池中的 worker 节点数量。
 - 没有指定 **kataConfigPoolSelector** 来选择要安装的节点。

在这种情况下，安装会在 control plane 节点上启动，因为 Operator 假设节点具有 control plane 和 worker 角色。**kataConfig** CR 的 **status** 部分在安装过程中不会更新。(KATA-1017)

- 当在 OpenShift 沙盒容器中使用 Buildah 工具的旧版本时，构建会失败并显示以下错误：

```
process exited with error: fork/exec /bin/sh: no such file or directory
subprocess exited with status 1
```

您必须使用最新版本的 Buildah，位于 quay.io。

(KATA-1278)

- 在 web 控制台中的 **KataConfig** 选项卡中，如果您在 YAML 视图中点 **Create KataConfig**，则 **KataConfig** YAML 缺少 **spec** 字段。切换到 **Form** 视图，然后返回到 **YAML** 视图来解决这个问题，并显示完整的 YAML。(KATA-1372)
- 在 web 控制台的 **KataConfig** 选项卡中，如果一个 **KataConfig** CR 已存在，会出现一个 **404: Not found** 错误消息。要访问现有的 **KataConfig** CR，请转至 **Home > Search**。在 **Resources** 列表中，选择 **KataConfig**。(KATA-1605)
- 升级 OpenShift 沙盒容器不会自动更新现有的 **KataConfig** CR。因此，监控来自以前部署的 pod 不会被重启，并继续使用过时的 **kataMonitor** 镜像运行。使用以下命令升级 **kataMonitor** 镜像：

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

您还可以通过在 web 控制台中编辑 **KataConfig** YAML 来升级 **kataMonitor** 镜像。

(KATA-1650)

1.6. 异步勘误更新

OpenShift 沙盒容器 4.12 的安全更新、程序错误修正、功能增强更新将会通过红帽网络以异步勘误的形式发布。所有的 OpenShift Container Platform 4.12 勘误都可以通过[红帽客户门户网站](#)获得。如需有关异步勘误的更多信息，请参阅 [OpenShift Container Platform 生命周期](#)。

红帽客户门户网站的用户可以在红帽订阅管理（RHSM）帐户设置中启用勘误通知功能。当勘误通知被启用后，用户会在有与其注册的系统相关的勘误发行时接收到电子邮件通知。



注意

用户的红帽客户门户网站账户需要有注册的系统，以及使用 OpenShift Container Platform 的权限才可以接收到 OpenShift Container Platform 的勘误通知。

本节的内容将会持续更新，以提供以后发行的与 OpenShift 沙盒容器 1.3 相关的异步勘误信息。

1.6.1. RHSA-2022:6072 - OpenShift 沙盒容器 1.3.0 镜像发行版本、程序错误修正和功能增强公告

发布于：2022 年 8 月 17 日

OpenShift 沙盒容器发行版本 1.3.0 现已正式发布。此公告包含带有改进和程序错误修复的 OpenShift 沙盒容器的更新。

其程序错误修正列表包括在 [RHSA-2022:6072](#) 公告中。

1.6.2. RHSA-2022:7058 - OpenShift 沙盒容器 1.3.1 安全和程序错误修复公告

发布于：2022 年 10 月 19 日

OpenShift 沙盒容器发行版本 1.3.1 现已正式发布。此公告包含带有安全修复和程序错误修复的 OpenShift 沙盒容器的更新。

其程序错误修正列表包括在 [RHSA-2022:7058](#) 公告中。

1.6.3. RHBA-2023:0390 - OpenShift 沙盒容器 1.3.2 程序错误修复更新

发布于：2023 年 1 月 24 日

OpenShift 沙盒容器发行版本 1.3.2 现已正式发布。此公告包含 OpenShift 沙盒容器的更新，并包括了相关的程序漏洞修复。

其程序错误修正列表包括在 [RHBA-2023:0390](#) 公告中。

1.6.4. RHBA-2023:0485 - OpenShift 沙盒容器 1.3.3 程序错误修复更新

发布于：2023 年 1 月 30 日

OpenShift 沙盒容器发行版本 1.3.3 现已正式发布。此公告包含 OpenShift 沙盒容器的更新，包括程序错误修正和容器更新。

其程序错误修正列表包括在 [RHBA-2023:0485](#) 公告中。

第 2 章 了解 OPENSIFT 沙盒容器

OpenShift 沙盒容器支持 OpenShift Container Platform 为您提供了将 Kata Containers 作为额外可选运行时运行的内置支持。新的运行时支持专用虚拟机 (VM) 中的容器，从而改进了工作负载隔离。这对执行以下任务特别有用：

运行特权或不受信任的工作负载

OpenShift 沙盒容器 (OSC) 使得可以安全地运行需要特定特权的工作负载，而无需通过运行特权容器来破坏集群节点的风险。需要特殊权限的工作负载包括：

- 需要内核的特殊功能的工作负载，除了标准容器运行时（如 CRI-O）授予的默认功能外，例如访问低级别网络功能。
- 需要提高 root 特权的工作负载，例如访问特定物理设备。使用 OpenShift 沙盒容器时，只能将特定的设备传递给虚拟机，确保工作负载无法访问或错误配置系统的其余部分。
- 用于安装或使用 **set-uid** root 二进制文件的工作负载。这些二进制文件授予特殊权限，因此可能会造成安全风险。使用 OpenShift 沙盒容器时，对虚拟机有额外的权限，不授予对集群节点的特殊访问权限。

有些工作负载可能需要专门用于配置集群节点的权限。此类工作负载应该仍然使用特权容器，因为在虚拟机上运行可能会阻止它们正常工作。

确保每个工作负载的内核隔离

OpenShift 沙盒容器支持需要自定义内核调整（如 **sysctl**、调度程序更改或缓存调整）以及创建自定义内核模块（如 **out of tree** 或特殊参数）的工作负载。

在租户间共享相同的工作负载

OpenShift 沙盒容器允许您支持来自共享同一 OpenShift 集群的不同组织的多个用户（租户）。该系统还允许您从多个供应商运行第三方工作负载，如容器网络功能 (CNF) 和企业应用程序。例如，第三方 CNF 可能不希望其自定义设置与数据包调整或由其他应用程序设置的 **sysctl** 变量干扰。在完全隔离的内核内运行有助于防止“邻居噪音”配置问题。

确保正确隔离和沙盒测试软件

您可以使用 OpenShift 沙盒容器来运行具有已知漏洞的容器化工作负载，或者处理传统应用程序中的问题。通过这种隔离，管理员可以为开发人员提供对 pod 的管理控制，这在开发人员想要测试或验证管理员通常授权的配置时很有用。例如，管理员可以安全地将内核数据包过滤 (eBPF) 委派给开发人员。内核数据包过滤需要 **CAP_ADMIN** 或 **CAP_BPF** 权限，因此不允许在标准 CRI-O 配置下，因为这会授予容器主机 worker 节点上的每个进程的访问权限。类似地，管理员可以授予对 SystemTap 等入侵工具的访问权限，或者支持在开发期间加载自定义内核模块。

确保通过虚拟机边界的默认资源控制

默认情况下，CPU、内存、存储或网络等资源以 OpenShift 沙盒容器中的更加强大和更安全的方式进行管理。由于 OpenShift 沙盒容器部署到虚拟机上，因此额外的隔离层和安全性可为资源提供更精细的访问控制。例如，错误容器将无法分配超过虚拟机可用内存更多的内存。相反，需要专用访问网卡或磁盘的容器可以完全控制该设备，而无需访问其他设备。

2.1. OPENSIFT 沙盒容器支持的平台

您可以在裸机服务器或 Amazon Web Services (AWS) 裸机实例上安装 OpenShift 沙盒容器。不支持由其他云提供商提供的裸机实例。

Red Hat Enterprise Linux CoreOS (RHCOS) 是 OpenShift 沙盒容器唯一支持的操作系统。OpenShift 沙盒容器 1.3 在 Red Hat Enterprise Linux CoreOS (RHCOS) 8.6 上运行。

OpenShift 沙盒容器 1.3 与 OpenShift Container Platform 4.11 兼容。

2.2. OPENSIFT 沙盒容器常用术语

以下是整个文档中所使用的术语：

Sandbox

沙盒 (sandbox) 是一种隔离的环境，程序可以在其中运行。在沙盒中，您可以运行未经测试或不受信任的程序，而不影响到主机机器或操作系统。

在 OpenShift 沙盒容器环境中，沙盒通过使用虚拟化在不同的内核中运行工作负载来实现，从而增强了对在同一主机上运行的多个工作负载之间的交互的控制。

Pod

pod 是继承自 Kubernetes 和 OpenShift Container Platform 的构造。它代表了可以部署容器的资源。容器在 pod 内运行，pod 用于指定可以在多个容器之间共享的资源。

在 OpenShift 沙盒容器上下文中，pod 被实施为一个虚拟机。多个容器可以在同一虚拟机上在同一 pod 中运行。

OpenShift 沙盒容器 Operator

Operator 是一个软件组件，可自动执行一般需要人工在系统上执行的操作。

OpenShift 沙盒容器 Operator 的任务是管理集群上沙盒容器的生命周期。您可以使用 OpenShift 沙盒容器 Operator 来执行任务，如安装和删除沙盒容器、软件更新和状态监控。

Kata 容器

Kata 容器是一个上游核心项目，用于构建 OpenShift 沙盒容器。OpenShift 沙盒容器将 Kata 容器与 OpenShift Container Platform 集成。

KataConfig

KataConfig 对象代表沙盒容器的配置。它们存储有关集群状态的信息，如部署软件的节点。

运行时类

RuntimeClass 对象用于描述可以使用哪个运行时来运行给定工作负载。OpenShift 沙盒容器 Operator 安装和部署了名为 **kata** 的运行时类。运行时类包含有关运行时的信息，用于描述运行时需要运行的资源，如 [pod 开销](#)。

2.3. OPENSIFT 沙盒容器工作负载管理

OpenShift 沙盒容器提供以下功能以增强工作负载管理和分配：

2.3.1. OpenShift 沙盒容器构建块

OpenShift 沙盒容器 Operator 封装了来自 Kata 容器的所有组件。它管理安装、生命周期和配置任务。

OpenShift 沙盒容器 Operator 以 [Operator 捆绑包格式](#) 打包为两个容器镜像。捆绑包镜像包含元数据，这是使 operator OLM 就绪所必需的。第二个容器镜像包含监控和管理 **KataConfig** 资源的实际控制器。

2.3.2. RHCOS 扩展

OpenShift 沙盒容器 Operator 基于 Red Hat Enterprise Linux CoreOS (RHCOS) 扩展概念。Red Hat Enterprise Linux CoreOS (RHCOS) 扩展是安装可选 OpenShift Container Platform 软件的一种机制。OpenShift 沙盒容器 Operator 使用此机制在集群中部署沙盒容器。

沙盒容器 RHCOS 扩展包含用于 Kata、QEMU 及其依赖项的 RPM。您可以使用 Machine Config Operator 提供的 **MachineConfig** 资源启用它们。

其他资源

- [为 RHCOS 添加扩展](#)

2.3.3. 虚拟化和 OpenShift 沙盒容器

您可以在带有 OpenShift Virtualization 的集群上使用 OpenShift 沙盒容器。

要同时运行 OpenShift Virtualization 和 OpenShift 沙盒容器，您必须启用虚拟机迁移的虚拟机，以便不阻止节点重启。在虚拟机上配置以下参数：

- 使用 **ocs-storagecluster-ceph-rbd** 作为存储类。
- 在虚拟机中将 **evictionStrategy** 参数设置为 **LiveMigrate**。

其他资源

- [为虚拟机配置本地存储](#)
- [配置虚拟机驱逐策略](#)

2.4. 了解合规性及风险管理

OpenShift 沙盒容器可以在启用了 FIPS 的集群中使用。

在 FIPS 模式下运行时，OpenShift 沙盒容器组件、虚拟机和虚拟机镜像会根据 FIPS 进行调整。

FIPS 合规性是高安全性环境中所需的最重要的组件之一，可确保节点上只允许使用支持的加密技术。



重要

只有在 **x86_64** 架构中的 OpenShift Container Platform 部署支持 FIPS 验证的/Modules in Process 加密库。

要了解红帽对 OpenShift Container Platform 合规框架的观点，请参阅 [OpenShift 安全性指南手册](#) 中的“风险管理和法规就绪状态”一章。

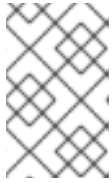
第 3 章 部署 OPENSIFT 沙盒容器工作负载

您可以使用 Web 控制台或 OpenShift CLI (**oc**) 安装 OpenShift 沙盒容器 Operator。安装 OpenShift 沙盒容器 Operator 之前，您必须准备 OpenShift Container Platform 集群。

3.1. 先决条件

在安装 OpenShift 沙盒容器前，请确保 OpenShift Container Platform 集群满足以下要求：

- 集群必须使用 Red Hat Enterprise Linux CoreOS (RHCOS) worker 在内部裸机基础架构上安装。您可以使用任何安装方法，包括用户置备、安装程序置备或协助的安装程序来部署集群。



注意

- OpenShift 沙盒容器仅支持 RHCOS worker 节点。不支持 RHEL 节点。
 - 不支持嵌套虚拟化。
- 您可以在 Amazon Web Services (AWS) 裸机实例上安装 OpenShift 沙盒容器。不支持由其他云提供商提供的裸机实例。

3.1.1. OpenShift 沙盒容器的资源要求

OpenShift 沙盒容器允许用户在沙盒运行时 (Kata) 中的 OpenShift Container Platform 集群中运行工作负载。每个 pod 由一个虚拟机 (VM) 表示。每个虚拟机都在 QEMU 进程中运行，并托管一个 **kata-agent** 进程，它充当管理容器工作负载的监管程序，以及这些容器中运行的进程。两个额外的进程会增加开销：

- **containerd-shim-kata-v2** 用于与 pod 通信。
- **virtiofsd** 代表客户机处理主机文件系统访问。

每个虚拟机都配置有默认内存量。对于明确请求内存的容器，额外的内存会被热插到虚拟机中。

在没有内存资源的情况下运行的容器会消耗可用内存，直到虚拟机使用的总内存达到默认分配。客户机及其 I/O 缓冲区也消耗内存。

如果容器被授予特定数量的内存，那么该内存会在容器启动前热插到虚拟机中。

当指定内存限制时，如果消耗的内存超过限制，工作负载将被终止。如果没有指定内存限制，则虚拟机中运行的内核可能会耗尽内存。如果内核内存不足，它可能会终止虚拟机上的其他进程。

默认内存大小

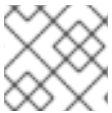
下表列出了资源分配的一些默认值。

资源	值
默认为虚拟机分配的内存	2Gi
启动时客户机 Linux 内核内存使用	~110Mi
QEMU 进程使用的内存 (虚拟机内存除外)	~30Mi

资源	值
virtiofsd 进程使用的内存（虚拟机 I/O 缓冲区除外）	~10Mi
containerd-shim-kata-v2 进程使用的内存	~20Mi
在 Fedora 上运行 dnf install 后的文件缓冲区缓存数据	~300Mi* [1]

文件缓冲区会出现并在多个位置考虑：

- 在客户机中它被显示为文件缓冲缓存。
- 在映射允许的用户空间文件 I/O 操作的 **virtiofsd** 守护进程中。
- 在 QEMU 进程中作为客户机内存。



注意

内存使用率指标正确考虑内存用量总量，该指标仅计算该内存一次。

[Pod 开销](#)描述了节点上 pod 使用的系统资源量。您可以使用 **oc describe runtimeclass kata** 获取 Kata 运行时的当前 pod 开销，如下所示。

示例

```
$ oc describe runtimeclass kata
```

输出示例

```
kind: RuntimeClass
apiVersion: node.k8s.io/v1
metadata:
  name: kata
overhead:
  podFixed:
    memory: "500Mi"
    cpu: "500m"
```

您可以通过更改 **RuntimeClass** 的 **spec.overhead** 字段来更改 pod 开销。例如，如果您为容器运行的配置消耗 QEMU 进程和客户机内核数据的 350Mi 内存，您可以更改 **RuntimeClass** 开销来满足您的需要。



注意

红帽支持指定的默认开销值。不支持更改默认开销值，这可能会导致技术问题。

在客户机中执行任何类型的文件系统 I/O 时，将在客户机内核中分配文件缓冲区。文件缓冲区也在主机上的 QEMU 进程以及 **virtiofsd** 进程中映射。

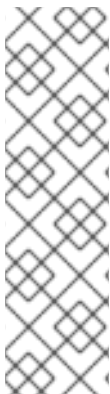
例如，如果您在客户机中使用 300Mi 文件缓冲区缓存，QEMU 和 **virtiofsd** 都显示使用 300Mi 额外内存。但是，所有三种情况下都使用相同的内存。换句话说，内存使用的总量仅为 300Mi，这个值被映射在三个不同的位置。报告内存使用率指标时，会正确计算。

其他资源

- [在裸机上安装用户置备的集群](#)

3.1.2. 检查集群节点是否有资格运行 OpenShift 沙盒容器

在运行 OpenShift 沙盒容器前，您可以检查集群中的节点是否有资格运行 Kata 容器。有些集群节点可能不符合沙盒容器的最低要求。节点不合格的最常见原因是节点上缺少虚拟化支持。如果您试图在不符合节点上运行沙盒工作负载，则会出现错误。您可以使用 Node Feature Discovery (NFD) Operator 和 **NodeFeatureDiscovery** 资源自动检查节点资格。



注意

如果您只想在符合条件的所选 worker 节点上安装 Kata 运行时，请将 **feature.node.kubernetes.io/runtime.kata=true** 标签应用到所选节点，并在 **KataConfig** 资源中设置 **checkNodeEligibility: true**。

另外，要在所有 worker 节点上安装 Kata 运行时，在 **KataConfig** 资源中设置 **checkNodeEligibility: false**。

在这两种场景中，您不需要创建 **NodeFeatureDiscovery** 资源。如果您确定节点有资格运行 Kata 容器，则应该只应用 **feature.node.kubernetes.io/runtime.kata=true** 标签。

以下流程将 **feature.node.kubernetes.io/runtime.kata=true** 标签应用到所有有资格的节点，并将 **KataConfig** 资源配置为检查节点资格。

先决条件

- 安装 OpenShift CLI (**oc**)。
- 以具有 **cluster-admin** 权限的用户身份登录。
- 安装 Node Feature Discovery (NFD) Operator。

流程

1. 创建 **NodeFeatureDiscovery** 资源来检测适合运行 Kata 容器的节点功能：
 - a. 将以下 YAML 保存到 **nfd.yaml** 文件中：

```
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. 创建 **NodeFeatureDiscovery** 自定义资源(CR) :

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

输出示例

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

feature.node.kubernetes.io/runtime.kata=true 标签应用到所有合格的 worker 节点。

2. 在 **KataConfig** 资源中将 **checkNodeEligibility** 字段设置为 **true** 来启用这个功能, 例如 :

- a. 将以下 YAML 保存到 **kata-config.yaml** 文件中 :

```

apiVersion: kataconfiguration.openshift.io/v1
kind: KataConfig
metadata:
  name: example-kataconfig
spec:
  checkNodeEligibility: true

```

- b. 创建 **KataConfig** CR :

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

输出示例

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

验证

- 验证集群中是否应用了正确的标签 :

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

输出示例

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

其他资源

- 有关安装 Node Feature Discovery (NFD) Operator 的更多信息，请参阅[安装 NFD](#)。

3.2. 使用 WEB 控制台部署 OPENSIFT 沙盒容器工作负载

您可从 web 控制台部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建 **KataConfig** 自定义资源 (CR)。在沙盒容器中部署工作负载后，您必须手动将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

3.2.1. 使用 Web 控制台安装 OpenShift 沙盒容器 Operator

您可从 OpenShift Container Platform Web 控制台安装 OpenShift 沙盒容器 Operator。

先决条件

- 已安装 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 Web 控制台中的 **Administrator** 视角，进入到 **Operators** → **OperatorHub**。
2. 在 **Filter by keyword** 字段中，输入 **OpenShift sandboxed containers**。
3. 选择 **OpenShift sandboxed containers** 标题。
4. 阅读 Operator 信息并单击 **Install**。
5. 在 **Install Operator** 页面中：
 - a. 从可用 **Update Channel** 选项列表中选择 **stable-1.3**。
 - b. 验证为 **Installed Namespace** 选择了 **Operator recommended Namespace**。这会在 **openshift-sandboxed-containers-operator** 命名空间中安装 Operator。如果此命名空间尚不存在，则会自动创建。



注意

尝试在 **openshift-sandboxed-containers-operator** 以外的命名空间中安装 OpenShift 沙盒容器 Operator 会导致安装失败。

- c. 验证是否为 **Approval Strategy** 选择了 **Automatic**。**Automatic** 是默认值，当有新的 z-stream 发行版本可用时，自动启用对 OpenShift 沙盒容器的自动更新。
6. 点 **Install**。

OpenShift 沙盒容器 Operator 现已安装在集群中。

验证

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators** → **Installed Operators**。
2. 验证 OpenShift 沙盒容器 Operator 是否在 operator 列表中列出。

3.2.2. 在 web 控制台中创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源(CR)，以便在集群节点上启用将 **kata** 作为 **RuntimeClass**。



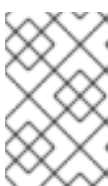
重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 OpenShift Container Platform 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。



注意

Kata 默认安装在所有 worker 节点上。如果要在特定节点上安装 **kata** 作为 **RuntimeClass**，您可以向这些节点添加标签，然后在创建时定义 **KataConfig** CR 中的标签。

流程

1. 从 Web 控制台中的 **Administrator** 视角，导航到 **Operators** → **Installed Operators**。
2. 从 Operator 列表中选择 OpenShift 沙盒容器 Operator。
3. 在 **KataConfig** 选项卡中，点 **Create KataConfig**。
4. 在 **Create KataConfig** 页面中，输入以下详情：
 - **名称**：输入 **KataConfig** 资源的名称。默认情况下，名称定义为 **example-kataconfig**。
 - **Labels**（可选）：输入任何相关的、识别到 **KataConfig** 资源的属性。每个标签代表一个键值对。
 - **checkNodeEligibility**（可选）：选择此复选框以使用 Node Feature Discovery Operator (NFD) 检测节点资格以作为 **RuntimeClass** 运行 **kata**。如需更多信息，请参阅“检查集群节点是否有资格运行 OpenShift 沙盒容器”。
 - **kataConfigPoolSelector**：默认情况下，**kata** 在所有节点上作为 **RuntimeClass** 安装。如果要在所选节点上安装 **kata** 作为 **RuntimeClass**，您必须添加一个 **matchExpression**：
 - a. 展开 **kataConfigPoolSelector** 区域。

- b. 在 **kataConfigPoolSelector** 中，展开 **matchExpressions**。这是标签选择器要求列表。
 - c. 点 **Add matchExpressions**。
 - d. 在 **key** 字段中，添加选择器应用到的标签键。
 - e. 在 **operator** 字段中，添加键与标签值的关系。有效的运算符为 **In**、**NotIn**、**Exists** 和 **DoesNotExist**。
 - f. 展开 **values** 区域，然后点 **Add value**。
 - g. 在 **Value** 字段中，为 **key** 标签值输入 **true** 或 **false**。
- **loglevel**：定义为将 **kata** 作为 **RuntimeClass** 运行的节点检索的日志数据级别。如需更多信息，请参阅“收集 OpenShift 沙盒容器数据”。

5. 点 **Create**。

新的 **KataConfig** CR 会被创建，并开始在工作节点上作为 **RuntimeClass** 安装 **kata**。等待 **kata** 安装完成，以及工作节点重启，然后继续下一步。



重要

OpenShift 沙盒容器仅将 **Kata** 安装为集群中的辅助可选运行时，而不作为主要运行时安装。

验证

1. 在 **KataConfig** 选项卡中，选择新的 **KataConfig** CR。
2. 在 **KataConfig** 页面中，选择 **YAML** 选项卡。
3. 监控状态中的 **installationStatus** 字段。
每次有更新时都会出现一条消息。点 **Reload** 查看更新的 **KataConfig** CR。

当 **Completed nodes** 的值等于工作节点或已标记的节点的数量，则代表安装已完成。该状态还包含安装完成的节点的列表。

3.2.3. 使用 Web 控制台在沙盒容器中部署工作负载

OpenShift 沙盒容器将 **Kata** 安装为集群上的辅助、可选运行时，而不是主运行时。

要在沙盒容器中部署 pod 模板工作负载，您必须手动将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

1. 从 web 控制台中的 **Administrator** 视角，展开 **Workloads** 并选择您要创建的工作负载类型。
2. 在工作负载页面中，点击以创建工作负载。
3. 在工作负载的 YAML 文件中，在列出容器的 **spec** 字段中，添加 **runtimeClassName: kata**。

Pod 示例

```

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

```

部署示例

```

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

```

4. 点击 **Save**。

OpenShift Container Platform 创建工作负载并开始调度它。

3.3. 使用 CLI 部署 OPENSIFT 沙盒容器工作负载

您可以使用 CLI 部署 OpenShift 沙盒容器工作负载。首先，您必须安装 OpenShift 沙盒容器 Operator，然后创建 **KataConfig** 自定义资源。在沙盒容器中部署工作负载后，您必须将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

3.3.1. 使用 CLI 安装 OpenShift 沙盒容器 Operator

您可以使用 OpenShift Container Platform CLI 安装 OpenShift 沙盒容器 Operator。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已订阅了 OpenShift 沙盒容器目录。



注意

订阅 OpenShift 沙盒容器目录为 **openshift-sandboxed-containers-operator** 命名空间提供了对 OpenShift 沙盒容器 Operator 的访问权限。

流程

1. 为 OpenShift 沙盒容器 Operator 创建 **Namespace** 对象。
 - a. 创建一个包含以下清单的 **Namespace** 对象 YAML 文件：

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

- b. 创建 **Namespace** 对象：

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

2. 为 OpenShift 沙盒容器 Operator 创建 **OperatorGroup** 对象。
 - a. 创建一个包含以下清单的 **OperatorGroup** 对象 YAML 文件：

```
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. 创建 **OperatorGroup** 对象：

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

3. 创建 **Subscription** 对象，以便为 OpenShift 沙盒容器 Operator 订阅 **命名空间**。

- a. 创建一个包含以下内容的 **Subscription** 对象 YAML 文件：

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

- b. 创建 **Subscription** 对象：

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

OpenShift 沙盒容器 Operator 现已安装在集群中。



注意

以上列出的所有对象文件名都是建议。您可以使用其他名称创建对象 YAML 文件。

验证

- 确保正确安装 Operator:

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

输出示例

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

其他资源

- [使用 CLI 从 OperatorHub 安装](#)

3.3.2. 使用 CLI 创建 KataConfig 自定义资源

您必须创建一个 **KataConfig** 自定义资源 (CR) 来作为 **RuntimeClass** 在节点上安装 **kata**。创建 **KataConfig** CR 会触发 OpenShift 沙盒容器 Operator 来执行以下操作：

- 在 RHCOS 节点上安装所需的 RHCOS 扩展，如 QEMU 和 **kata-containers**。
- 确保 **CRI-O** 运行时配置了正确的 **kata** 运行时处理程序。

- 使用默认配置创建一个名为 **kata** 的 **RuntimeClass** CR。这可让用户在 **RuntimeClassName** 字段中引用 CR 将工作负载配置为使用 **kata** 作为运行时。此 CR 也指定运行时的资源开销。



注意

Kata 默认安装在所有 worker 节点上。如果要在特定节点上安装 **kata** 作为 **RuntimeClass**，您可以向这些节点添加标签，然后在创建时定义 **KataConfig** CR 中的标签。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。



重要

创建 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 OpenShift Container Platform 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

流程

1. 使用以下清单创建 YAML 文件：

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

- 1** 将 'checkNodeEligibility' 设置为 **true**，以检测节点资格作为 **RuntimeClass** 运行 **kata**。如需更多信息，请参阅"检查集群节点是否有资格运行 OpenShift 沙盒容器"。

2. (可选) 如果只在所选节点上安装 **kata** 作为 **RuntimeClass**，请创建一个包含清单中的标签的 YAML 文件：

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



```

metadata:
  name: cluster-kataconfig
spec:
  checkNodeEligibility: false
  logLevel: info
  kataConfigPoolSelector:
    matchLabels:
      <label_key>: '<label_value>' ❶

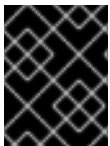
```

❶ **kataConfigPoolSelector** 中的标签只支持单个值；不支持 **nodeSelector** 语法。

3. 创建 **KataConfig** 资源：

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

新的 **KataConfig** CR 会被创建，并开始在工作节点上作为 **RuntimeClass** 安装 **kata**。等待 **kata** 安装完成，以及工作节点重启，然后继续下一步。



重要

OpenShift 沙盒容器仅将 Kata 安装为集群中的辅助可选运行时，而不作为主要运行时安装。

验证

- 监控安装进度：

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

当 **Is In Progress** 的值显示为 **false** 后，安装就已完成。

其他资源

- [了解如何更新节点上的标签](#)

3.3.3. 使用 CLI 在沙盒容器中部署工作负载

OpenShift 沙盒容器将 Kata 安装为集群上的辅助、可选运行时，而不是主运行时。

要在沙盒容器中部署 pod 模板工作负载，您必须将 **kata** 作为 **runtimeClassName** 添加到工作负载 YAML 文件中。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift 沙盒容器 Operator。
- 您已创建了 **KataConfig** 自定义资源 (CR)。

流程

- 将 **runtimeClassName: kata** 添加到任何 pod 模板对象中：
 - **Pod** 对象
 - **ReplicaSet** 对象
 - **ReplicationController** 对象
 - **StatefulSet** 对象
 - **Deployment** 对象
 - **deploymentConfig** 对象

Pod 对象示例

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

Deployment 对象示例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mypod
  labels:
    app: mypod
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mypod
  template:
    metadata:
      labels:
        app: mypod
    spec:
      runtimeClassName: kata
      containers:
        - name: mypod
          image: myImage
```

OpenShift Container Platform 创建工作负载并开始调度它。

验证

- 检查 pod 模板对象上的 **runtimeClassName** 字段。如果 **runtimeClassName** 是 **kata**，则工作负载在 OpenShift 沙盒容器中运行。

3.4. 其他资源

- OpenShift 沙盒容器 Operator 在受限网络环境中被支持。如需更多信息，请参阅[在受限网络中使用 Operator Lifecycle Manager](#)。
- 当在受限网络中使用断开连接的集群时，您必须在 [Operator Lifecycle Manager](#) 中配置代理支持，以访问 OperatorHub。使用代理允许集群获取 OpenShift 沙盒容器 Operator。

第 4 章 监控 OPENSIFT 沙盒容器

您可以使用 OpenShift Container Platform Web 控制台监控与沙盒工作负载和节点的健康状态相关的指标。

OpenShift 沙盒容器在 web 控制台中有一个预先配置的仪表板，管理员还可以通过 Prometheus 访问和查询原始指标。

4.1. 关于 OPENSIFT 沙盒容器指标

OpenShift 沙盒容器指标让管理员能够监控沙盒容器的运行方式。您可以在 web 控制台中的 Metrics UI 中查询这些指标。

OpenShift 沙盒容器指标为以下类别收集：

Kata 代理指标

Kata 代理指标显示有关嵌入在沙盒容器中运行的 kata 代理进程的信息。这些指标包括 `/proc/<pid>/[io, stat, status]` 中的数据。

Kata 客户机操作系统指标

Kata 客户机操作系统指标显示沙盒容器中运行的客户机操作系统中的数据。这些指标包括 `/proc/[stats, diskstats, meminfo, vmstats]` 和 `/proc/net/dev` 中的数据。

hypervisor 指标

hypervisor 指标显示有关运行嵌入在沙盒容器中虚拟机的虚拟机监控程序的数据。这些指标主要包括 `/proc/<pid>/[io, stat, status]` 中的数据。

Kata 监控指标

Kata 监控器是收集指标数据并提供给 Prometheus 的进程。kata 监控指标显示有关 kata-monitor 进程本身的资源使用情况的详细信息。这些指标还包括 Prometheus 数据收集的计数器。

Kata containerd shim v2 指标

Kata containerd shim v2 指标显示有关 kata shim 进程的详细信息。这些指标包括来自 `/proc/<pid>/[io, stat, status]` 和详细的资源使用量指标的数据。

4.2. 查看 OPENSIFT 沙盒容器的指标

您可以在 web 控制台的 **Metrics** 页面中访问 OpenShift 沙盒容器的指标。

先决条件

- 已安装 OpenShift Container Platform 4.12。
- 已安装 OpenShift 沙盒容器。
- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Observe → Metrics**。
2. 在输入字段中，输入您要观察到的指标的查询。
所有与 kata 相关的指标都以 **kata** 开头。键入 **kata** 将显示含有所有可用 kata 指标的列表。

在页面中会视觉化查询的指标。

其他资源

- 有关创建 PromQL 查询来查看指标的更多信息，请参阅 [查询指标](#)。

4.3. 查看 OPENSIFT 沙盒容器仪表板

您可以在 web 控制台的 **Dashboards** 页面中访问 OpenShift 沙盒容器仪表板。

先决条件

- 已安装 OpenShift Container Platform 4.12。
- 已安装 OpenShift 沙盒容器。
- 您可以使用具有 **cluster-admin** 角色或所有项目的查看权限的用户访问集群。

流程

1. 从 web 控制台中的 **Administrator** 视角，进入到 **Observe → Dashboards**。
2. 从 **Dashboard** 下拉列表中，选择 **Sandboxed Containers** 仪表板。
3. 可选：在 **Time Range** 列表中为图形选择一个时间范围。
 - 选择预定义的时间段。
 - 通过选择 **Time Range** 列表中的 **Custom** 时间范围来设置自定义时间范围。
 - a. 定义您要查看的数据的日期和时间范围。
 - b. 单击 **Save** 以保存自定义时间范围。
4. 可选：选择一个 **Refresh Interval**。

仪表板会出现在页面中，其中包含来自 Kata 客户机操作系统类别的以下指标：

正在运行的虚拟机数量

显示集群中运行的沙盒容器总数。

CPU 使用率（每个虚拟机）

显示每个沙盒容器的 CPU 使用量。

内存用量（每个虚拟机）

显示每个沙盒容器的内存用量。

将鼠标悬停在仪表板中的每个图形上，以显示具体项目的详细信息。

4.4. 其他资源

- 有关收集支持数据的更多信息，请参阅 [收集集群数据](#)。

第 5 章 卸载 OPENSIFT 沙盒容器

您可以使用 OpenShift Container Platform Web 控制台或 OpenShift CLI (**oc**) 卸载 OpenShift 沙盒容器。下面解释这两个程序。

5.1. 使用 WEB 控制台卸载 OPENSIFT 沙盒容器

使用 OpenShift Container Platform Web 控制台删除相关的 OpenShift 沙盒容器 pod、资源和命名空间。

5.1.1. 使用 Web 控制台删除 OpenShift 沙盒容器 pod

要卸载 OpenShift 沙盒容器，您必须首先删除所有使用 **kata** 作为 **runtimeClass** 的 pod。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您有一个使用 **kata** 作为 **runtimeClass** 的 pod 列表。

流程

1. 从 **Administrator** 视角中，进入到 **Workloads** → **Pods**。
2. 使用 **Search by name** 字段搜索您要删除的 pod。
3. 点 pod 名称打开它。
4. 在 **Details** 页面中，检查已针对 **Runtime** 类显示 **kata**。
5. 点 **Actions** 菜单，再选择 **Delete Pod**。
6. 在确认窗口中点击 **Delete**。

其他资源

您可以从 OpenShift CLI 检索使用 **kata** 作为 **runtimeClass** 的运行 pod 的列表。详情请参阅 [删除 OpenShift 沙盒容器 pod](#)。

5.1.2. 使用 Web 控制台删除 KataConfig 自定义资源

删除 **KataConfig** 自定义资源 (CR) 会从集群中移除并卸载 **kata** 运行时及其相关资源。



重要

删除 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 OpenShift Container Platform 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您没有任何正在运行的 pod 使用 **kata** 作为 **runtimeClass**。

流程

1. 从 **Administrator** 视角中，进入到 **Operators → Installed Operators**。
2. 使用 **Search by name** 字段搜索 OpenShift 沙盒容器 Operator。
3. 点 Operator 打开它，然后选择 **KataConfig** 选项卡。
4. 点 **KataConfig** 资源的 **Options** 菜单 ，然后选择 **Delete KataConfig**。
5. 在确认窗口中点击 **Delete**。

等待 **kata** 运行时和资源卸载，并使 worker 节点重启，然后继续下一步。

5.1.3. 使用 Web 控制台删除 OpenShift 沙盒容器 Operator

删除 OpenShift 沙盒容器 Operator 会删除 Operator 的目录订阅、Operator 组和集群服务版本 (CSV)。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 **Administrator** 视角中，进入到 **Operators → Installed Operators**。
2. 使用 **Search by name** 字段搜索 OpenShift 沙盒容器 Operator。

3. 点击 Operator 的 **Options** 菜单  并选择 **Uninstall Operator**。
4. 在确认窗口中点 **Uninstall**。

5.1.4. 使用 Web 控制台删除 OpenShift 沙盒容器命名空间

运行上述命令后，集群将恢复到安装过程之前的状态。现在，您可以通过删除 **openshift-sandboxed-containers-operator** 命名空间来撤销对 Operator 的命名空间访问。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 **Administrator** 视角中，进入到 **Administration** → **Namespaces**。
2. 使用 **Search by name** 字段搜索 **openshift-sandboxed-containers-operator** 命名空间。

3. 点命名空间的 **Options** 菜单  并选择 **Delete Namespace**。



注意

如果 **Delete Namespace** 选项不可用，代表您没有删除命名空间的权限。

4. 在 **Delete Namespace** 窗格中，输入 **openshift-sandboxed-containers-operator** 并点 **Delete**。
5. 点 **Delete**。

5.1.5. 使用 Web 控制台删除 KataConfig 自定义资源定义


KataConfig 自定义资源定义 (CRD) 可让您定义 **KataConfig** CR。要完成卸载过程，请从集群中删除 **KataConfig** CRD。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已从集群中删除 **KataConfig** CR。
- 您已从集群中移除了 OpenShift 沙盒容器 Operator。

流程

1. 从 **Administrator** 视角，进入到 **Administration** → **CustomResourceDefinitions**。

2. 使用 **Search by name** 字段搜索 **KataConfig**。
3. 点 **KataConfig** CRD  的 **Options** 菜单，然后选择 **Delete CustomResourceDefinition**。
4. 在确认窗口中点击 **Delete**。
5. 等待 **KataConfig** CRD 会从列表中消失。这可能需要几分钟。

5.2. 使用 CLI 卸载 OPENSIFT 沙盒容器

您可以使用 OpenShift Container Platform [命令行界面\(CLI\)](#) 卸载 OpenShift 沙盒容器。按照显示它们的顺序按照以下步骤操作。

5.2.1. 使用 CLI 删除 OpenShift 沙盒容器 pod

要卸载 OpenShift 沙盒容器，您必须首先删除所有使用 **kata** 作为 **runtimeClass** 的 pod。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 已安装命令行 JSON 处理器 (**jq**)。

流程

1. 运行以下命令，搜索使用 **kata** 作为 **runtimeClass** 的 pod：

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

2. 要删除每个 pod，请运行以下命令：

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

5.2.2. 使用 CLI 删除 KataConfig 自定义资源

从集群中删除并卸载 **kata** 运行时及其所有相关资源，如 CRI-O 配置和 **RuntimeClass**。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。



重要

删除 **KataConfig** CR 会自动重启 worker 节点。重启可能需要 10 到 60 分钟。妨碍重启时间的因素如下：

- 带有更多 worker 节点的大型 OpenShift Container Platform 部署。
- 激活 BIOS 和 Diagnostics 实用程序。
- 在硬盘而不是 SSD 上部署。
- 在物理节点上部署，如裸机，而不是在虚拟节点上部署。
- CPU 和网络较慢。

流程

1. 运行以下命令来删除 **KataConfig** 自定义资源：

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

OpenShift 沙盒容器 Operator 会删除最初为在集群中启用运行时创建的所有资源。



重要

在删除过程中，CLI 会停止响应，直到所有 worker 节点重启为止。等待进程完成，然后执行验证或继续进行后续步骤。

验证

- 要验证 **KataConfig** 自定义资源是否已删除，请运行以下命令：

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

输出示例

```
No KataConfig instances exist
```

5.2.3. 使用 CLI 删除 OpenShift 沙盒容器 Operator

通过删除 Operator 订阅、Operator 组、集群服务版本(CSV)和命名空间从集群中删除 OpenShift 沙盒容器 Operator。

先决条件

- 已在集群中安装了 OpenShift Container Platform 4.10。
- 已安装 OpenShift CLI(**oc**)。
- 您已安装了 readand-line JSON 处理器(**jq**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 运行以下命令，从订阅中获取 OpenShift 沙盒容器的集群服务版本 (CSV) 名称：

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

2. 运行以下命令，从 Operator Lifecycle Manager (OLM) 中删除 OpenShift 沙盒容器 Operator 订阅：

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

3. 运行以下命令，删除 OpenShift 沙盒容器的 CSV 名称：

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

4. 运行以下命令来获取 OpenShift 沙盒容器 Operator 组名称：

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

5. 运行以下命令来删除 OpenShift 沙盒容器 Operator 组名称：

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

6. 运行以下命令来删除 OpenShift 沙盒容器命名空间：

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

5.2.4. 使用 CLI 删除 KataConfig 自定义资源定义

KataConfig 自定义资源定义 (CRD) 可让您定义 **KataConfig** CR。从集群中删除 **KataConfig** CRD。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已从集群中删除 **KataConfig** CR。
- 您已从集群中移除了 OpenShift 沙盒容器 Operator。

流程

1. 运行以下命令来删除 **KataConfig** CRD：

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

验证

- 要验证 **KataConfig** CRD 是否已删除，请运行以下命令：

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

输出示例

```
Unknown CR KataConfig
```

第 6 章 升级 OPENSIFT 沙盒容器

OpenShift 沙盒容器组件的升级由以下三个步骤组成：

- 升级 OpenShift Container Platform 以更新 **Kata** 运行时及其依赖项。
- 升级 OpenShift 沙盒容器 Operator 以更新 Operator 订阅。
- 手动修补 **KataConfig** 自定义资源 (CR) 以更新监控 pod。

您可以在 OpenShift 沙盒容器 Operator 升级前或之后升级 OpenShift Container Platform，但有以下例外。在升级 OpenShift 沙盒容器 Operator 后，始终立即应用 **KataConfig** 补丁。



重要

如果您使用 OpenShift 沙盒容器 1.3 升级到 OpenShift Container Platform 4.11，建议的顺序是从 OpenShift 沙盒容器从 1.2 升级到 1.3，然后将 OpenShift Container Platform 从 4.10 升级到 4.11。

6.1. 升级 OPENSIFT 沙盒容器资源

OpenShift 沙盒容器资源使用 Red Hat Enterprise Linux CoreOS (RHCOS) 扩展部署到集群中。

RHCOS 扩展沙盒容器包含运行 Kata 容器所需的组件，如 Kata 容器运行时、虚拟机监控程序 QEMU 和其他依赖项。您可以通过将集群升级到 OpenShift Container Platform 的新版本来升级扩展。

有关升级 OpenShift Container Platform 的更多信息，请参阅[更新集群](#)。

6.2. 升级 OPENSIFT 沙盒容器 OPERATOR

使用 Operator Lifecycle Manager (OLM) 手动或自动升级 OpenShift 沙盒容器 Operator。在初始部署期间，选择手动或自动升级可决定将来的升级模式。对于手动升级，Web 控制台会显示集群管理员可安装的可用更新。

有关在 Operator Lifecycle Manager (OLM) 中升级 OpenShift 沙盒容器 Operator 的更多信息，请参阅[更新已安装的 Operator](#)。

6.3. 升级 OPENSIFT 沙盒容器监控 POD

升级 OpenShift 沙盒容器后，您需要更新 **KataConfig** CR 中的 monitor 镜像来升级监控 pod。否则，监控器 pod 将继续运行之前版本中的镜像。

您可以使用 Web 控制台或 CLI 执行更新。

6.3.1. 使用 Web 控制台升级 monitor pod

OpenShift Container Platform 中的 **KataConfig** YAML 文件包含监控镜像的版本号。使用正确的版本更新版本号。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 从 OpenShift Container Platform 的 **Administrator** 视角，进入到 **Operators → Installed Operators**。
2. 选择 **OpenShift 沙盒容器 Operator** 并进入 **KataConfig** 选项卡。
3. 使用 **Search by name** 字段搜索 **KataConfig** 资源。**KataConfig** 资源的默认名称为 **example-kataconfig**。
4. 选择 **KataConfig** 资源，再进入 **KataConfig** 选项卡。
5. 修改 **kataMonitorImage** 的版本号：

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

6. 点击 **Save**。

6.3.2. 使用 CLI 升级 monitor pod

您可以手动修补 **KataConfig** CR 中的 monitor 镜像，以更新 monitor pod。

先决条件

- 在集群中安装了 OpenShift Container Platform 4.12。
- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

- 在 OpenShift Container Platform CLI 中运行以下命令：

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

其中：**<kataconfig_name>** :: 指定您的 Kata 配置文件的名称，如 **example-kataconfig**。

第 7 章 收集 OPENSIFT 沙盒容器数据

当对 OpenShift 沙盒容器进行故障排除时，您可以创建一个支持问题单，并使用 **must-gather** 工具提供调试信息。

如果您是集群管理员，您还可以自行查看日志，启用更详细的日志级别。

7.1. 为红帽支持收集 OPENSIFT 沙盒容器数据

在提交问题单时同时提供您的集群信息，可以帮助红帽支持为您进行排除故障。

您可使用 **must-gather** 工具来收集有关 OpenShift Container Platform 集群的诊断信息，包括虚拟机和有关 OpenShift 沙盒容器的其他数据。

为了获得快速支持，请提供 OpenShift Container Platform 和 OpenShift 沙盒容器的诊断信息。

7.1.1. 关于 must-gather 工具

oc adm must-gather CLI 命令可收集最有助于解决问题的集群信息，包括：

- 资源定义
- 服务日志

默认情况下，**oc adm must-gather** 命令使用默认的插件镜像，并写入 **./must-gather.local**。

另外，您可以使用适当的参数运行命令来收集具体信息，如以下部分所述：

- 要收集与一个或多个特定功能相关的数据，请使用 **--image** 参数和镜像，如以下部分所述。
例如：

```
$ oc adm must-gather --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.12.0
```

- 要收集审计日志，请使用 **-- /usr/bin/gather_audit_logs** 参数，如以下部分所述。
例如：

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



注意

作为默认信息集合的一部分，不会收集审计日志来减小文件的大小。

当您运行 **oc adm must-gather** 时，集群的新项目中会创建一个带有随机名称的新 pod。在该 pod 上收集数据，并保存至以 **must-gather.local** 开头的一个新目录中。此目录在当前工作目录中创建。

例如：

```

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
...

```

要使用 **must-gather** 来收集 OpenShift 沙盒容器数据，您必须指定 OpenShift 沙盒容器镜像：

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

7.2. 关于 OPENSIFT 沙盒容器日志数据

当您收集集群的日志数据时，以下功能和对象与 OpenShift 沙盒容器相关联：

- 所有属于任何 OpenShift 沙盒容器资源的命名空间及其子对象
- 所有 OpenShift 沙盒容器自定义资源定义 (CRD)

以下 OpenShift 沙盒容器组件日志会针对使用 **kata** 运行时运行的每个 pod 收集：

- Kata 代理日志
- Kata 运行时日志
- QEMU 日志
- 审计日志
- CRI-O 日志

7.3. 为 OPENSIFT 沙盒容器启用调试日志

作为集群管理员，您可以为 OpenShift 沙盒容器收集更详细的日志级别。通过更改运行 OpenShift 沙盒容器的 worker 节点的 CRI-O 运行时中的 **log_level** 来增强日志记录。

流程

1. 使用以下清单为 **ContainerRuntimeConfig** CR 创建 YAML 文件：

```
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 为您要修改的机器配置池指定一个标签。

2. 创建 **ContainerRuntimeConfig** CR：

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



注意

以上列出的文件名是一个建议。您可以使用其他名称创建此文件。

3. 验证是否已创建 CR :

```
$ oc get ctrcfg
```

输出示例

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

验证

1. 监控机器配置池，直到所有 worker 节点的 **UPDATED** 字段都显示为 **True** :

```
$ oc get mcp worker
```

输出示例

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

2. 验证 **log_level** 是否在 CRI-O 中更新 :a. 打开到机器配置池中节点的 **oc debug** 会话，并运行 **chroot /host**。

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

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

b. 验证 **crio.conf** 文件中的更改 :

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

输出示例

```
log_level = "debug"
```

7.3.1. 查看 OpenShift 沙盒容器的调试日志

集群管理员可以使用 OpenShift 沙盒容器增强的调试日志来排除问题。每个节点的日志会输出到节点日志中。

您可以查看以下 OpenShift 沙盒容器组件的日志 :

- Kata 代理
- Kata runtime (**containerd-shim-kata-v2**)
- virtiofsd

QEMU 的日志不会打印到节点日志。但是，QEMU 故障会报告到运行时，QEMU 客户机的控制台会输出到节点日志中。您可以将这些日志与 Kata 代理日志一起查看。

先决条件

- 已安装 OpenShift CLI(**oc**)。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

- 要查看 Kata 代理日志和客户机控制台日志，请运行：

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

- 要查看 kata 运行时日志，请运行：

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

- 要查看 virtiofsd 日志，请运行：

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

7.4. 其他资源

- 有关收集支持数据的更多信息，请参阅[收集集群数据](#)。