OpenShift Container Platform 4.6

支持

获取 OpenShift Container Platform 支持

Last Updated: 2023-02-27

OpenShift Container Platform 4.6 支持

获取 OpenShift Container Platform 支持

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本文档提供了有关从红帽获取 OpenShift Container Platform 支持的信息。文中还包含有关通过 Telemetry 和 Insights Operator 进行远程健康监控的信息。本文档还详细介绍了远程健康监控的好处。

目录

第 1 章 支持概述

红帽提供了集群管理员工具，用于为集群、监控和故障排除收集数据。

1.1. 获得支持

获取支持 ：访问红帽客户门户网站查看知识库文章、提交支持问题单以及查看其他产品文档和资源。

1.2. 远程健康监控问题

远程健康监控问题 ：OpenShift Container Platform 会收集有关集群的遥测和配置数据，并使用 Telemeter Client 和 Insights Operator 向红帽报告。红帽使用此数据了解并解决 *连接的集群* 中的问题。与连接的集群类似，您可以在 受限网络中使用远程健康监控。OpenShift Container Platform 使用以下方法收集数据和监控健康状况：

- **Telemetry** ：遥测客户端收集并每 4 分 30 秒将标值上传至红帽。红帽将此数据用于：
  - 监控集群。
  - 推出 OpenShift Container Platform 升级。
  - 改进升级体验。
- **Insights Operator** ：默认情况下，OpenShift Container Platform 安装并启用 Insight Operator，每两小时报告配置和组件故障状态。Insight Operator 有助于：
  - 主动识别潜在的集群问题。
  - 在 Red Hat OpenShift Cluster Manager 中提供解决方案和预防性操作。

您可以查看 遥测信息。

如果您启用了远程健康报告，请使用 Insights 来识别问题。您可以选择禁用远程健康报告。

1.3. 收集有关集群的数据

收集有关集群的数据 ：红帽建议在打开支持问题单时收集您的调试信息。这有助于红帽支持执行根本原因分析。集群管理员可以使用以下方法收集有关集群的数据：

- **must-gather 工具** ： 使用 **must-gather** 工具收集有关集群的信息并调试问题。
- **sosreport** ：使用 **sosreport** 工具收集配置详情、系统信息和诊断数据用于调试目的。
- **集群 ID** ：在向红帽支持提供信息时，获取集群的唯一标识符。
- **bootstrap 节点日志**: 收集 **bootkube.service journald** 单元日志和来自 bootstrap 节点的容器日志，以排除与 bootstrap 相关的问题。
- **集群节点 journal 日志** ：收集 **journald** 单元日志和独立集群节点上的 **/var/log** 中的日志，以排除与节点相关的问题。
- **网络追踪** ：从特定的 OpenShift Container Platform 集群节点或一个容器提供给红帽支持提供网络数据包跟踪，以帮助排除与网络相关的问题。
- **诊断数据** ：使用 **redhat-support-tool** 命令收集有关集群的诊断数据。

1.4. 故障排除问题

集群管理员可以监控并排除以下 OpenShift Container Platform 组件问题：

- 安装问题 ：OpenShift Container Platform 安装可完成各种阶段。您可以执行以下操作：
  - 监控安装阶段。
  - 确定在哪个阶段安装问题发生。
  - 调查多个安装问题。
  - 从失败安装中收集日志。
- 节点问题 ：集群管理员可以通过查看节点的状态、资源使用量和配置来验证和排除节点相关问题。您可以查询以下内容：
  - 节点上的 kubelet 状态。
  - 集群节点日志.
- Crio 问题 ：集群管理员可在每个集群节点上验证 CRI-O 容器运行时引擎状态。如果遇到容器运行时问题，请执行以下操作：
  - 收集 CRI-O journald 单元日志。
  - 清理 CRI-O 存储。
- 操作系统问题 ：OpenShift Container Platform 在 Red Hat Enterprise Linux CoreOS 上运行。如果遇到操作系统问题，可以调查内核崩溃过程。确保以下内容：
  - 启用 kdump。
  - 测试 kdump 配置。
  - 分析内核转储。
- 网络问题 ：要对 Open vSwitch 问题进行故障排除，集群管理员可以执行以下操作：
  - 临时配置 Open vSwitch 日志级别。
  - 永久配置 Open vSwitch 日志级别。
  - 显示 Open vSwitch 日志。
- Operator 问题 ：集群管理员可以执行以下操作来解决 Operator 问题：
  - 验证 Operator 订阅状态。
  - 检查 Operator pod 健康状况。
  - 收集 Operator 日志。
- Pod 问题 ：集群管理员可以通过查看 pod 的状态并完成以下内容来排除与 pod 相关的问题：
  - 查看 pod 和容器日志。
  - 启动具有 root 访问权限的 debug pod。

- Source-to-image 问题 ：集群管理员可以观察 S2I 阶段，以确定 S2I 进程中的故障发生位置。收集以下内容来解决 Source-to-Image(S2I)问题：
  - Source-to-Image 诊断数据。
  - 用于调查应用程序故障的应用程序诊断数据。
- 存储问题 ：当无法在新节点中挂载卷时，会发生多附加存储错误，因为失败的节点无法卸载附加的卷。集群管理员可执行以下操作解决多附加存储问题：
  - 使用 RWX 卷启用多个附件。
  - 使用 RWO 卷时,恢复或删除故障节点。
- 监控问题 ：集群管理员可按照监控故障排除页面中的步骤进行操作。如果您的用户定义的项目的指标不可用，或者 Prometheus 消耗了大量磁盘空间，请检查以下内容：
  - 调查用户定义的指标不可用的原因。
  - 确定为什么 Prometheus 消耗大量磁盘空间。
- 日志记录问题：集群管理员可以按照 OpenShift Logging 问题的故障排除页面上的步骤进行操作。检查以下内容以解决日志问题：
  - Logging Operator 的状态。
  - 日志存储的状态。
  - OpenShift Logging 警报.
  - 使用 **oc adm must-gather** 命令有关 OpenShift 日志记录环境的信息。
- OpenShift CLI(oc)问题 ：通过增加日志级别来判断 OpenShift CLI(oc)问题。

第 2 章 管理集群资源

您可以在 OpenShift Container Platform 中应用全局配置选项。Operator 在集群中应用这些配置设置。

2.1. 与集群资源交互

您可以使用 OpenShift Container Platform 中的 OpenShift CLI(**oc**)工具与集群资源交互。运行 **oc api-resources** 命令后看到的集群资源可以被编辑。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您可以访问 web 控制台或已安装了 **oc** CLI 工具。

流程

1. 要查看应用了哪些配置 Operator，请运行以下命令：

```
$ oc api-resources -o name | grep config.openshift.io
```

2. 要查看您可以配置的集群资源，请运行以下命令：

```
$ oc explain <resource_name>.config.openshift.io
```

3. 要查看集群中的自定义资源定义(CRD)对象配置，请运行以下命令：

```
$ oc get <resource_name>.config -o yaml
```

4. 要编辑集群资源配置，请运行以下命令：

```
$ oc edit <resource_name>.config -o yaml
```

第 3 章 获取支持

3.1. 获取支持

如果您在执行本文档所述的某个流程或 OpenShift Container Platform 时遇到问题，请访问 红帽客户门户网站。通过红帽客户门户网站：

- 搜索或者浏览红帽知识库，了解与红帽产品相关的文章和解决方案。
- 提交问题单给红帽支持。
- 访问其他产品文档。

要识别集群中的问题，您可以在 OpenShift Cluster Manager 中使用 Insights。Insights 提供了问题的详细信息，并在有可用的情况下，提供了如何解决问题的信息。

如果您对本文档有任何改进建议，或发现了任何错误，请为相关文档组件提交 JIRA 问题。请提供具体详情，如章节名称和 OpenShift Container Platform 版本。

3.2. 关于红帽知识库

红帽知识库提供丰富的内容以帮助您最大程度地利用红帽的产品和技术。红帽知识库包括文章、产品文档和视频，概述了安装、配置和使用红帽产品的最佳实践。另外，您还可以搜索已知问题的解决方案，其提供简洁的根原因描述和补救措施。

3.3. 搜索红帽知识库

如果出现 OpenShift Container Platform 问题，您可以先进行搜索，以确定红帽知识库中是否已存在相关的解决方案。

先决条件

- 您有红帽客户门户网站帐户。

流程

1. 登录到 红帽客户门户网站。
2. 在主红帽客户门户网站搜索字段中，输入与问题相关的关键字和字符串，包括：
   - OpenShift Container Platform 组件（如 **etcd**）
   - 相关步骤（比如 **安装**）
   - 警告、错误消息和其他与输出与特定的问题相关
3. 点 **Search**。
4. 选择 **OpenShift Container Platform** 产品过滤器。
5. 在内容类型过滤中选择 **Knowledgebase**。

3.4. 提交支持问题单

先决条件

- 已安装 OpenShift CLI（**oc**）。
- 您有红帽客户门户网站帐户。
- 您可以访问 OpenShift Cluster Manager。

流程

1. 登录到 红帽客户门户网站 并选择 **SUPPORT CASES → Open a case**。
2. 为您的问题选择适当的类别（如 **Defect / Bug**）、产品(**OpenShift Container Platform**)和产品版本（如果还没有自动填充则为**4.6** ）。
3. 查看推荐的红帽知识库解决方案列表，它们可能会与您要报告的问题相关。如果建议的文章没有解决这个问题，请点 **Continue**。
4. 输入一个简洁但描述性的问题概述，以及问题症状的详细信息，以及您预期的结果。
5. 查看更新的推荐红帽知识库解决方案列表，它们可能会与您要报告的问题相关。这个列表的范围会缩小，因为您在创建问题单的过程中提供了更多信息。如果建议的文章没有解决这个问题，请点 **Continue**。
6. 请确保提供的帐户信息是正确的，如果需要，请相应调整。
7. 检查自动填充的 OpenShift Container Platform 集群 ID 是否正确。如果不正确，请手动提供集群 ID。
   - 使用 OpenShift Container Platform Web 控制台手动获得集群 ID。
     a. 导航到 **Home → Dashboards → Overview**。
     b. 该值包括在 **Details** 中的 **Cluster ID** 项中。
   - 另外，也可以通过 OpenShift Container Platform Web 控制台直接创建新的支持问题单，并自动填充集群 ID。
     a. 从工具栏导航至 （？） **help → Open Support Case**。
     b. **Cluster ID** 的值会被自动填充 。
   - 要使用 OpenShift CLI（**oc**）获取集群 ID，请运行以下命令：

     ```
     $ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
     ```

8. 完成以下提示的问题，点 **Continue**:
   - 您在哪里遇到了这个问题？什么环境？
   - 这个行为在什么时候发生？发生频率？重复发生？是否只在特定时间发生？
   - 请提供这个问题对您的业务的影响及与时间相关的信息？
9. 上传相关的诊断数据文件并点击 **Continue**。建议您将使用 **oc adm must-gather** 命令收集的数据作为起点，并提供这个命令没有收集的与您的具体问题相关的其他数据。
10. 输入相关问题单管理详情，点 **Continue**。

11. 预览问题单详情，点 Submit。

3.5. 其他资源

- 有关在集群中识别问题的详情，请参阅使用 Insights 识别集群中的问题。

第 4 章 通过连接集群进行远程健康监控

4.1. 关于远程健康监控

OpenShift Container Platform 会收集有关集群的遥测和配置数据，并使用 Telemeter Client 和 Insights Operator 向红帽报告。提供给红帽的数据可实现本文档概述的好处。

通过 Telemetry 和 Insights Operator 向红帽报告数据的集群被称为*连接的集群 (connected cluster)*。

Telemetry 是红帽用来描述 OpenShift Container Platform Telemeter 客户端向红帽发送的信息的术语。轻量级属性从连接的集群发送到红帽，以便启用订阅管理自动化、监控集群的健康状态、提供支持以及改进客户体验。

Insights Operator 收集 OpenShift Container Platform 配置数据并将其发送到红帽。这些数据用于生成有关集群可能潜在存在的问题的分析报告。这些分析结果会通过 console.redhat.com/openshift 提供给集群管理员。

本文档中提供了有关这两个工具的更多信息。

Telemetry 和 Insights Operator 的优点

Telemetry 和 Insights Operator 为最终用户提供以下优点：

- **增强了识别和解决问题的能力**。对于一些事件，最终用户可能会认为是正常的，但从更广泛深入的角度来说，红帽会对这些事件的影响有不同的评估。有些问题可以被更快地识别并解决，而不需要最终用户创建一个支持问题单或一个 JIRA 问题。
- **高级的版本管理**。OpenShift Container Platform 提供了 **candidate**、**fast** 和 **stable** 发行频道，供您选择一个最佳的更新策略。版本从 **fast** 到 **stable** 的过程取决于更新的速度以及升级过程中的事件。通过连接的集群提供的信息，红帽可以将发行版本质量提高到 **stable** 频道，并对在 **fast** 频道中发现的问题做出更快反应。
- **有针对性地对新功能的开发进行优先级排序**。通过收集的数据，可以了解哪些 OpenShift Container Platform 的功能被用户广泛使用。通过这些信息，红帽可以专注于开发对客户有严重影响的新功能。
- **更好的支持体验**。在红帽客户门户网站上创建支持问题单时，可以为连接的集群提供集群 ID。这可让红帽通过使用连接的信息，简化用户的支持体验。本文档提供有关改进的支持体验的更多信息。
- **预测分析**。通过从连接集群收集到的信息，可以在 console.redhat.com/openshift 中显示与您的集群相关的 insights 信息。红帽正在积极应用深入学习、机器学习及智能自动化，以帮助识别 OpenShift Container Platform 集群潜在的问题。

4.1.1. 关于 Telemetry

Telemetry 会向红帽发送一组精选的集群监控指标子集。Telemeter 客户端每 4 分 30 秒获取一次指标值，并将数据上传到红帽。本文档中描述了这些指标。

红帽使用这一数据流来实时监控集群，必要时将对影响客户的问题做出反应。它同时还有助于红帽向客户推出 OpenShift Container Platform 升级，以便最大程度降低服务影响，持续改进升级体验。

这类调试信息将提供给红帽支持和工程团队，其访问限制等同于访问通过问题单报告的数据。红帽利用所有连接集群信息来帮助改进 OpenShift Container Platform，提高其易用性。

其他资源

- 有关 更新或升级集群的更多信息，请参阅 OpenShift Container Platform 更新文档。

4.1.1.1. Telemetry 收集的信息

Telemetry 收集以下信息：

- 安装期间生成的唯一随机标识符
- 版本信息，包括 OpenShift Container Platform 集群版本并安装了用于决定更新版本可用性的更新详情
- 更新信息，包括每个集群可用的更新数、用于更新的频道和镜像存储库、更新进度信息以及更新中发生的错误数
- 部署 OpenShift Container Platform 的供应商平台的名称及数据中心位置
- 有关集群、机器类型和机器的大小信息，包括 CPU 内核数和每个机器所使用的 RAM 量
- etcd 成员数和存储在 etcd 集群中的对象数量
- 在集群中安装的 OpenShift Container Platform 框架组件及其状况和状态
- 有关组件、功能和扩展的使用情况信息
- 有关技术预览和不受支持配置的使用详情
- 有关降级软件的信息
- 标记为 **NotReady** 的节点的信息
- 为降级 Operator 列出为 "related objects" 的所有命名空间的事件
- 帮助红帽支持为客户提供有用支持的配置详情。这包括云基础架构级别的节点配置、主机名、IP 地址、Kubernetes pod 名称、命名空间和服务。
- 有关证书的有效性的信息

Telemetry 不会收集任何身份识别的信息,如用户名或密码。红帽不会收集个人信息。如果红帽发现个人信息被意外地收到，红帽会删除这些信息。有关红帽隐私实践的更多信息，请参考红帽隐私声明。

其他资源

- 如需了解如何列出 Telemetry 中从 Prometheus 收集的属性的详细信息，请参阅显示 Telemetry 收集的数据。
- 如需 Telemetry 从 Prometheus 收集的属性列表，请参阅 上游 cluster-monitoring-operator 源代码。
- 在默认情况下，Telemetry 会被安装并启用。如果您需要选择不使用远程健康报告，请参阅不使用远程健康报告。

4.1.2. 关于 Insights Operator

Insights Operator 会定期收集配置和组件故障状态信息，默每两小时向红帽报告这些数据。这些信息可让红帽评估配置，它提供了比 Telemetry 报告更深入的数据。

OpenShift Container Platform 用户可以在 Red Hat Hybrid Cloud Console 上的 Insights Advisor 服务中显示每个集群的报告。如果发现了任何问题，Insights 会提供更详细的信息，并在可能的情况下提供如何解决相关问题的步骤。

Insights Operator 不会收集任何身份识别信息，如用户名、密码或证书。如需有关 Red Hat Insights 数据收集和控制的信息，请参阅 Red Hat Insights 数据和应用程序安全性 。

红帽使用所有连接的集群信息以实现：

- 识别潜在的集群问题，并在 Red Hat Hybrid Cloud Console 上的 Insights Advisor 服务中提供解决方案和防止动作
- 通过为产品和支持团队提供聚合和重要信息来改进 OpenShift Container Platform
- 使 OpenShift Container Platform 更直观

其他资源

- Insights Operator 被默认安装并启用。如果您需要选择不使用远程健康报告，请参阅不使用远程健康报告。

4.1.2.1. Insights Operator 收集的信息

Insights Operator 收集以下信息：

- 有关集群及其组件的常规信息，以识别与您所使用的具体 OpenShift Container Platform 版本和环境的相关问题
- 集群的配置文件（如容器镜像仓库的配置）用于识别设置参数中的问题
- 集群组件中发生的错误
- 正在运行的更新的进度信息，以及组件升级的状态
- 有关 OpenShift Container Platform 部署平台（如 Amazon Web Services）以及集群所在区域的详情
- 如果 Operator 报告了一个问题，则会收集 **openshift-*** 和 **kube-*** 项目中 OpenShift Container Platform 核心 pod 的信息。这包括状态、资源、安全上下文、卷信息等。

其他资源

- 如需了解如何查看 Insights Operator 收集的数据，请参阅 Insights Operator 收集的数据。
- 用户可以查看 Insights Operator 的源代码，并对代码进行贡献。如需 Insights Operator 收集的项目列表，请参阅 Insights Operator 上游项目。

4.1.3. 了解 Telemetry 和 Insights Operator 数据流

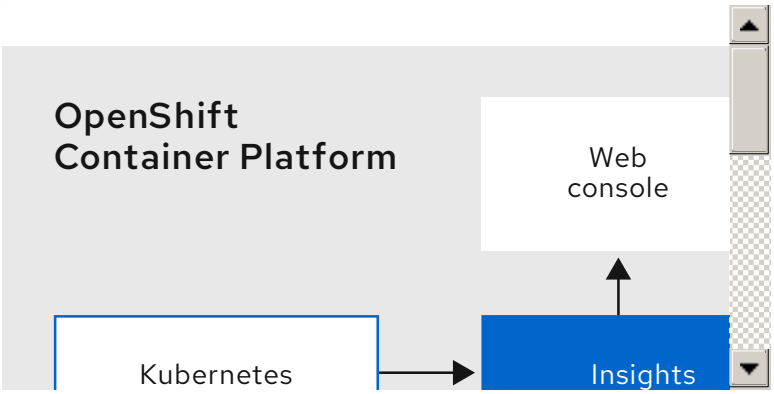
Telemeter Client 从 Prometheus API 收集所选的时间序列数据。时间序列数据每 4 分 30 秒上传到 api.openshift.com 进行处理。

Insights Operator 从 Kubernetes API 和 Prometheus API 中收集所选的数据并进行存档。该归档会每两小时上传到 console.redhat.com 进行处理。Insights Operator 还从 console.redhat.com 下载最新的 Insights 分析。这用于填充 OpenShift Container Platform Web 控制台的 **Overview** 页面中包含的 **Insights status**。

所有与红帽的通信都使用传输层安全（TLS）和 mutual 证书验证通过加密频道进行。所有数据在传输及非活跃的情况下都会被加密。

对处理客户数据的系统是通过多因素验证和严格的授权控制来控制的。访问权限的设置是基于需要的，仅限于针对需要的操作。

telemetry 和 Insights Operator 数据流



其它资源

- 如需有关 OpenShift Container Platform 监控堆栈的更多信息，请参阅监控概述。
- 如需有关配置防火墙，以及为 Telemetry 和 Insights 启用端点的详情，请参阅配置防火墙

4.1.4. 有关如何使用远程健康监控数据的更多详情

Telemetry 收集的信息和Insights Operator 收集的信息中提供了与启用健康检查健康相关的数据收集的信息。

如本文档前面部分所述，红帽会收集您使用红帽产品的数据，如提供支持和升级、优化性能或配置、减小服务影响、识别和补救威胁、故障排除、改进提供和用户体验、响应问题、根据情况提供账单目的。

集合保护

红帽采用一些技术和机构措施来保护遥测数据和配置数据。

共享

红帽可以在红帽内部通过 Telemetry 和 Insights Operator 共享收集的数据，以提升您的用户体验。红帽可能会以汇总的方式与业务合作伙伴共享遥测和配置数据，该表格可帮助合作伙伴更好地了解其业务及其客户对红帽产品的使用，或者确保成功整合这些合作伙伴支持的产品。

第三方

红帽可能会与某些第三方合作，协助收集、分析和存储遥测和配置数据。

用户控制/启用和禁用遥测和配置数据收集

您可以按照不使用远程健康报告中的说明禁用 OpenShift Container Platform Telemetry 和 Insights Operator。

4.2. 显示远程健康监控收集的数据

作为管理员，您可以查看 Telemetry 和 Insights Operator 收集的指标。

4.2.1. 显示 Telemetry 收集的数据

您可以查看 Telemetry 收集的集群和组件的时间序列数据。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 您必须使用具有 **cluster-admin** 角色或 **cluster-monitoring-view** 角色的用户登录 集群。

流程

1. 找到在 OpenShift Container Platform 集群中运行的 Prometheus 服务的 URL：

```
$ oc get route prometheus-k8s -n openshift-monitoring -o jsonpath="{.spec.host}"
```

2. 访问此 URL。
3. 在 **Expression** 中输入查询条件并点 **Execute**：

```
{__name__=~"cluster:usage:.*|count:up0|count:up1|cluster_version|cluster_version_available_u
pdates|cluster_operator_up|cluster_operator_conditions|cluster_version_payload|cluster_install
er|cluster_infrastructure_provider|cluster_feature_set|instance:etcd_object_counts:sum|ALERT
S|code:apiserver_request_total:rate:sum|cluster:capacity_cpu_cores:sum|cluster:capacity_mem
ory_bytes:sum|cluster:cpu_usage_cores:sum|cluster:memory_usage_bytes:sum|openshift:cpu_
usage_cores:sum|openshift:memory_usage_bytes:sum|workload:cpu_usage_cores:sum|worklo
ad:memory_usage_bytes:sum|cluster:virt_platform_nodes:sum|cluster:node_instance_type_cou
nt:sum|cnv:vmi_status_running:count|node_role_os_version_machine:cpu_capacity_cores:sum
node_role_os_version_machine:cpu_capacity_sockets:sum|subscription_sync_total|csv_succee
ded|csv_abnormal|ceph_cluster_total_bytes|ceph_cluster_total_used_raw_bytes|ceph_health_s
tatus|job:ceph_osd_metadata:count|job:kube_pv:count|job:ceph_pools_iops:total|job:ceph_pool
s_iops_bytes:total|job:ceph_versions_running:count|job:noobaa_total_unhealthy_buckets:sum|j
b:noobaa_bucket_count:sum|job:noobaa_total_object_count:sum|noobaa_accounts_num|noob
aa_total_usage|console_url|cluster:network_attachment_definition_instances:max|cluster:netwo
rk_attachment_definition_enabled_instance_up:max|insightsclient_request_send_total|cam_app
_workload_migrations|cluster:apiserver_current_inflight_requests:sum:max_over_time:2m|clust
er:telemetry_selected_series:count",alertstate=~"firing|"}
```

此查询复制 Telemetry 对正在运行的 OpenShift Container Platform 集群的 Prometheus 服务所做的请求，并返回由 Telemetry 收集的完整时间序列集。

4.2.2. 显示 Insights Operator 收集的数据

您可以查看 Insights Operator 收集的数据。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 为 Insights Operator 查找当前正在运行的 pod 的名称：

```
$ INSIGHTS_OPERATOR_POD=$(oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers  --field-selector=status.phase=Running)
```

2. 复制 Insights Operator 收集的最近数据存档：

```
$ oc cp openshift-insights/$INSIGHTS_OPERATOR_POD:/var/lib/insights-operator ./insights-data
```

Insights Operator 最近存档可在 **insights-data** 目录中找到。

4.3. 不使用远程健康报告功能

您可以选择不报告集群健康和使用情况数据。

要选择不使用远程健康报告，您必须：

1. 修改全局集群 pull secret，以禁用远程健康报告。
2. 更新集群，以使用这个修改后的 pull secret。

4.3.1. 禁用远程健康报告的后果

在 OpenShift Container Platform 中，用户可以选择不报告使用信息。但是，红帽通过连接的集群可加快对问题的反应速度，为客户提供更好支持，同时更好地了解产品升级对集群的影响。连接的集群还可帮助简化订阅和权利过程，并让 Red Hat OpenShift Cluster Manager 服务提供集群及其订阅状态概述。

红帽强烈建议，即使需要在生产环境集群中禁用这个功能，在预生产环境集群和测试集群中启用健康和使用情况报告功能，。这样红帽便可在您的环境中参与对 OpenShift Container Platform 质量的审核，并对产品问题做出更快反应。

选择不使用连接的集群的一些后果包括：

- 如果没有提交问题单，红帽将无法监控产品升级是否成功，以及您的集群的健康状况。
- 红帽将无法使用配置数据来更好地分类客户问题单，无法识别客户认为比较重要的配置。
- Red Hat OpenShift Cluster Manager 将无法显示您的集群数据，包括健康和使用情况信息。
- 没有自动使用情况报告功能，您必须通过 console.redhat.com 手动输入您的订阅授权信息。

即使在受限网络中，Telemetry 和 Insights 数据仍可通过正确配置代理来报告。

4.3.2. 修改全局集群 pull secret，以禁用远程健康报告

您可以修改现有全局集群 pull secret，以禁用远程健康报告。该操作将同时禁用 Telemetry 和 Insights Operator。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 把全局集群 pull secret 下载到本地文件系统。

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. 在文本编辑器中编辑所下载的 **.dockerconfigjson** 文件。
3. 删除 **cloud.openshift.com** JSON 条目，如：

```
"cloud.openshift.com":{"auth":"<hash>","email":"<email_address>"}
```

4. 保存该文件。

现在，您可以更新集群，使用修改后的 pull secret。

4.3.3. 更新全局集群 pull secret

您可以通过替换当前的 pull secret 或附加新的 pull secret 来更新集群的全局 pull secret。

当用户使用单独的 registry 存储镜像而不使用安装过程中的 registry时，需要这个过程。

警告

集群资源必须调整为新的 pull secret，这样可暂时限制集群的可用性。

警告

更新全局 pull secret 会导致在 Machine Config Operator（MCO）同步更改时节点重启。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 可选：要将新的 pull secret 附加到现有 pull secret 中，请完成以下步骤：
   a. 输入以下命令下载 pull secret：

```
$ oc get secret/pull-secret -n openshift-config --template='{{index .data
".dockerconfigjson" | base64decode}}' ><pull_secret_location> 1
```

   1 提供 pull secret 文件的路径。

   b. 输入以下命令来添加新 pull secret：

```
$ oc registry login --registry="<registry>" \ 1
--auth-basic="<username>:<password>" \ 2
--to=<pull_secret_location> 3
```

1. 提供新的 registry。您可以在同一个 registry 中包含多个软件仓库，例如：**--registry="<registry/my-namespace/my-repository>"**。
2. 提供新 registry 的凭据。
3. 提供 pull secret 文件的路径。

另外，您可以对 pull secret 文件执行手动更新。

2. 输入以下命令为您的集群更新全局 pull secret：

```
$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=
<pull_secret_location> 1
```

1. 提供新 pull secret 文件的路径。

该更新将推广至所有节点，可能需要一些时间，具体取决于集群大小。在这段时间中，节点会排空(drain)，pod 将在剩余节点上重新调度。

4.4. 使用 INSIGHTS 发现集群中的问题

Insights 会反复分析 Insights Operator 发送的数据。OpenShift Container Platform 用户可以在 Red Hat Hybrid Cloud Console 上的 Insights Advisor 服务中显示报告。

4.4.1. 显示集群中的潜在问题

本节论述了如何在 OpenShift Cluster Manager 中显示 Insights 报告。

请注意，Insights 会重复分析您的集群并显示最新结果。这些结果可能会改变，如您解决了一个问题，或发现了一个新问题时。

先决条件

- 集群在 OpenShift Cluster Manager 中注册。
- 启用了远程健康报告（这是默认设置）。
- 登录到 OpenShift Cluster Manager。

流程

1. 点左侧面板中的 **Cluster** 菜单。
2. 点集群名称显示集群详情。
3. 打开集群的 **Insights** 标签页。
   根据结果，标签会显示以下其中之一：

- **Your cluster passed all health checks**，如果 Insights 没有发现任何问题。
- Insights 检测到的问题列表，按照风险进行优先级排序（低、中、重要和严重）。
- **No health checks to display**，如果 Insights 还没有分析集群。这个分析会在集群安装并连接到互联网后立即开始。

4. 如果这个标签中显示了问题，点条目前面的 > 图标查看详情。
根据具体问题，详情中可能还可会包含到红帽知识库文章的链接。如需了解解决这个问题的信息，点 **How to remediate this issue**。

4.5. 使用 INSIGHTS OPERATOR

Insights Operator 会定期收集配置和组件故障状态信息，默每两小时向红帽报告这些数据。这些信息可让红帽评估配置，它提供了比 Telemetry 报告更深入的数据。OpenShift Container Platform 用户可以在 Red Hat Hybrid Cloud Console 上的 Insights Advisor 服务中显示报告。

其它资源

- Insights Operator 被默认安装并启用。如果您需要选择不使用远程健康报告，请参阅不使用远程健康报告。
- 有关使用 Insights Advisor 发现集群中的问题的更多信息，请参阅使用 Insights 识别集群中的问题。

4.5.1. 下载 Insights Operator 存档

Insights Operator 将收集的数据存储在集群的 **openshift-insights** 命名空间中的存档中。您可以下载并查看 Insights Operator 收集的数据。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 为 Insights Operator 查找正在运行的 pod 的名称：

```
$ oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers  --field-selector=status.phase=Running
```

2. 复制 Insights Operator 收集的最近数据存档：

```
$ oc cp openshift-insights/<insights_operator_pod_name>:/var/lib/insights-operator ./insights-data
```
(1)

(1) 将 **<insights_operator_pod_name>** 替换为上一命令中的 pod 名称输出。

Insights Operator 最近存档可在 **insights-data** 目录中找到。

4.5.2. 查看 Insights Operator 收集持续时间

您可以查看 Insights Operator 花时间来收集存档中包含的信息。这有助于您了解 Insights Operator 资源使用情况和 Insights Advisor 的问题。

先决条件

- Insights Operator 归档的最新副本。

流程

1. 在归档中，打开 **/insights-operator/gathers.json**。
   文件包含 Insights Operator 收集操作列表：

```
{
  "name": "clusterconfig/authentication",
  "duration_in_ms": 730, (1)
  "records_count": 1,
  "errors": null,
  "panic": null
}
```

(1) **duration_in_ms** 是每个收集操作的时间（毫秒）。

2. 检查每个收集操作是否有异常情况。

4.6. 在受限网络中使用远程健康报告

您可以手动收集和上传 Insights Operator 存档，以便从受限网络中诊断问题。

要在受限网络中使用 Insights Operator，您必须：

- 创建 Insights Operator 归档的副本。
- 将 Insights Operator 存档上传到 console.redhat.com。

4.6.1. 复制 Insights Operator 存档

您必须创建 Insights Operator 数据存储的副本，以上传到 cloud.redhat.com。

先决条件

- 以 **cluster-admin** 用户身份登录 OpenShift Container Platform。

流程

1. 查找当前运行的 Insights Operator pod 的名称：

```
$ INSIGHTS_OPERATOR_POD=$(oc get pods --namespace=openshift-insights -o custom-columns=:metadata.name --no-headers  --field-selector=status.phase=Running)
```

2. 从 Insights Operator 容器复制最近的数据存档：

```
$ oc cp openshift-insights/$INSIGHTS_OPERATOR_POD:/var/lib/insights-operator ./insights-data
```

-

Insights Operator 最近存档可在 **insights-data** 目录中找到。

4.6.2. 上传 Insights Operator 存档

您可以将 Insights Operator 存档手动上传到 console.redhat.com，以诊断潜在的问题。

先决条件

- 以 **cluster-admin** 用户身份登录 OpenShift Container Platform。
- 您有一个没有互联网访问限制的工作站。
- 您已创建了 Insights Operator 归档的副本。

流程

1. 下载 **dockerconfig.json** 文件：

```
$ oc extract secret/pull-secret -n openshift-config --to=.
```

2. 复制来自 **dockerconfig.json** 文件的 **"cloud.openshift.com" "auth"** 令牌：

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<your_token>",
      "email": "asd@redhat.com"
    }
}
```

3. 将存档上传到 console.redhat.com ：

```
$ curl -v -H "User-Agent: insights-operator/one10time200gather184a34f6a168926d93c330
cluster/<cluster_id>" -H "Authorization: Bearer <your_token>" -F
"upload=@<path_to_archive>; type=application/vnd.redhat.openshift.periodic+tar"
https://console.redhat.com/api/ingress/v1/upload
```

其中 ***<cluster_id>*** 是集群 ID，***<your_token>*** 是来自 pull secret 的令牌，***<path_to_archive>*** 是 Insights Operator 归档的路径。

如果操作成功，该命令会返回 **"request_id"** 和 **"account_number"**：

输出示例

```
* Connection #0 to host console.redhat.com left intact
{"request_id":"393a7cf1093e434ea8dd4ab3eb28884c","upload":
{"account_number":"6274079"}}%
```

验证步骤

1. 登录到 https://console.redhat.com/openshift。
2. 点左侧面板中的 **Cluster** 菜单。

3. 要显示集群详情，点集群名称。

4. 打开集群的 **Insights Advisor** 选项卡。
   如果上传成功，标签会显示以下之一：

   - 如果 Insights Advisor 没有发现任何问题，代表**您的集群已通过了所有建议**。
   - Insights Advisor 检测到的问题列表，按风险级别排列（低、中、重要和严重）。

第 5 章 收集集群数据

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。

建议您提供：

- 使用 **oc adm must-gather** 命令收集的数据
- 唯一的集群 ID

5.1. 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:v4.9.0
  ```

- 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
...
```

5.1.1. 为红帽支持收集您的集群数据

您可使用 **oc adm must-gather** CLI 命令收集有关您的集群的调试信息。

先决条件

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

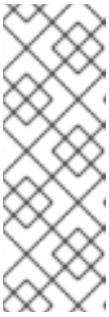
流程

1. 进入要存储 **must-gather** 数据的目录。
2. 运行 **oc adm must-gather** 命令：

```
$ oc adm must-gather
```

注意

如果此命令失败，例如，您无法在集群中调度 pod，则使用 **oc adm inspect** 命令来收集特定资源的信息。请联络红帽支持以获取推荐收集的资源信息。



注意

如果集群使用受限网络，则需要执行额外的步骤。如果您镜像的容器镜像仓库有一个信任的 CA，您必须首先将这个信任的 CA 添加到集群中。对于受限网络中的所有集群，在使用 **oc adm must-gather** 命令前，需要导入默认的 **must-gather** 镜像作为一个镜像流。

```
$ oc import-image is/must-gather -n openshift
```

3. 从工作目录中刚刚创建的 **must-gather** 目录创建一个压缩文件。例如，在使用 Linux 操作系统的计算机上运行以下命令：

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

1 务必将 **must-gather-local.5421342344627712289/** 替换为实际目录名称。

4. 在红帽客户门户中为您的问题单附上压缩文件。

5.1.2. 收集有关特定功能的数据

您可通过将 **oc adm must-gather** CLI 命令与 **--image** 或 **--image-stream** 参数结合使用来收集有关特定功能的调试信息。**must-gather** 工具支持多个镜像，这样您便可通过运行单个命令收集多个功能的数据。

表 5.1. 支持的 must-gather 镜像

Image	用途
registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.5.8	OpenShift Virtualization 的数据收集。
registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8	OpenShift Serverless 的数据收集。

Image	用途
registry.redhat.io/openshift-service-mesh/istio-must-gather-rhel8	Red Hat OpenShift Service Mesh 的数据收集。
registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7	MTC 的数据收集。
registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.6	Red Hat OpenShift Container Storage 的数据收集。
registry.redhat.io/openshift4/ose-cluster-logging-operator	Red Hat OpenShift 集群日志记录的数据收集。
registry.redhat.io/openshift4/ose-local-storage-mustgather-rhel8	Local Storage Operator 的数据收集。

注意

要收集除特定功能数据外的默认 **must-gather** 数据，请添加 **--image-stream=openshift/must-gather** 参数。

先决条件

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

流程

1. 进入存储 **must-gather** 数据的目录。
2. 使用一个或多个 **--image** 或 **--image-stream** 参数运行 **oc adm must-gather** 命令。例如，使用以下命令可收集默认集群数据和 OpenShift Virtualization 特定信息：

```
$ oc adm must-gather \
 --image-stream=openshift/must-gather \ 1
 --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.5.8 2
```

1 默认 OpenShift Container Platform **must-gather** 镜像

2 OpenShift Virtualization 的 must-gather 镜像

您可以将 **must-gather** 工具与额外参数搭配使用，以收集集群中与集群日志记录和 Cluster Logging Operator 特别相关的数据。对于集群日志记录，运行以下命令：

```
$ oc adm must-gather --image=$(oc -n openshift-logging get deployment.apps/cluster-logging-operator \
 -o jsonpath='{.spec.template.spec.containers[?(@.name == "cluster-logging-
```

```
operator")].image}')
```

例 5.1. 集群日志记录 的 must-gather 输出示例

```
├── cluster-logging
│   ├── clo
│   │   ├── cluster-logging-operator-74dd5994f-6ttgt
│   │   ├── clusterlogforwarder_cr
│   │   ├── cr
│   │   ├── csv
│   │   ├── deployment
│   │   └── logforwarding_cr
│   ├── collector
│   │   ├── fluentd-2tr64
│   ├── curator
│   │   └── curator-1596028500-zkz4s
│   ├── eo
│   │   ├── csv
│   │   ├── deployment
│   │   └── elasticsearch-operator-7dc7d97b9d-jb4r4
│   ├── es
│   │   ├── cluster-elasticsearch
│   │   │   ├── aliases
│   │   │   ├── health
│   │   │   ├── indices
│   │   │   ├── latest_documents.json
│   │   │   ├── nodes
│   │   │   ├── nodes_stats.json
│   │   │   └── thread_pool
│   │   ├── cr
│   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   │   └── logs
│   │       ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   ├── install
│   │   ├── co_logs
│   │   ├── install_plan
│   │   ├── olmo_logs
│   │   └── subscription
│   └── kibana
│       ├── cr
│       ├── kibana-9d69668d4-2rkvz
├── cluster-scoped-resources
│   └── core
│       ├── nodes
│       │   ├── ip-10-0-146-180.eu-west-1.compute.internal.yaml
│       └── persistentvolumes
│           ├── pvc-0a8d65d9-54aa-4c44-9ecc-33d9381e41c1.yaml
├── event-filter.html
├── gather-debug.log
└── namespaces
    ├── openshift-logging
    │   ├── apps
    │   │   ├── daemonsets.yaml
    │   │   ├── deployments.yaml
    │   │   ├── replicasets.yaml
    │   │   └── statefulsets.yaml
```

```
│   ├── batch
│   │   ├── cronjobs.yaml
│   │   └── jobs.yaml
│   ├── core
│   │   ├── configmaps.yaml
│   │   ├── endpoints.yaml
│   │   ├── events
│   │   │   ├── curator-1596021300-wn2ks.162634ebf0055a94.yaml
│   │   │   ├── curator.162638330681bee2.yaml
│   │   │   ├── elasticsearch-im-app-1596020400-gm6nl.1626341a296c16a1.yaml
│   │   │   ├── elasticsearch-im-audit-1596020400-9l9n4.1626341a2af81bbd.yaml
│   │   │   ├── elasticsearch-im-infra-1596020400-v98tk.1626341a2d821069.yaml
│   │   │   ├── elasticsearch-im-app-1596020400-cc5vc.1626341a3019b238.yaml
│   │   │   ├── elasticsearch-im-audit-1596020400-s8d5s.1626341a31f7b315.yaml
│   │   │   └── elasticsearch-im-infra-1596020400-7mgv8.1626341a35ea59ed.yaml
│   │   ├── events.yaml
│   │   ├── persistentvolumeclaims.yaml
│   │   ├── pods.yaml
│   │   ├── replicationcontrollers.yaml
│   │   ├── secrets.yaml
│   │   └── services.yaml
│   ├── openshift-logging.yaml
│   ├── pods
│   │   ├── cluster-logging-operator-74dd5994f-6ttgt
│   │   │   ├── cluster-logging-operator
│   │   │   │   └── cluster-logging-operator
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   └── cluster-logging-operator-74dd5994f-6ttgt.yaml
│   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff
│   │   │   ├── cluster-logging-operator-registry
│   │   │   │   └── cluster-logging-operator-registry
│   │   │   │       └── logs
│   │   │   │           ├── current.log
│   │   │   │           ├── previous.insecure.log
│   │   │   │           └── previous.log
│   │   │   ├── cluster-logging-operator-registry-6df49d7d4-mxxff.yaml
│   │   │   └── mutate-csv-and-generate-sqlite-db
│   │   │       └── mutate-csv-and-generate-sqlite-db
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   ├── curator-1596028500-zkz4s
│   │   ├── elasticsearch-cdm-lp8l38m0-1-794d6dd989-4jxms
│   │   ├── elasticsearch-im-app-1596030300-bpgcx
│   │   │   ├── elasticsearch-im-app-1596030300-bpgcx.yaml
│   │   │   └── indexmanagement
│   │   │       └── indexmanagement
│   │   │           └── logs
│   │   │               ├── current.log
│   │   │               ├── previous.insecure.log
│   │   │               └── previous.log
│   │   └── fluentd-2tr64
```

```
        ├── fluentd
        │   └── fluentd
        │       └── logs
        │           ├── current.log
        │           ├── previous.insecure.log
        │           └── previous.log
        ├── fluentd-2tr64.yaml
        └── fluentd-init
            └── fluentd-init
                └── logs
                    ├── current.log
                    ├── previous.insecure.log
                    └── previous.log
    ├── kibana-9d69668d4-2rkvz
    │   ├── kibana
    │   │   └── kibana
    │   │       └── logs
    │   │           ├── current.log
    │   │           ├── previous.insecure.log
    │   │           └── previous.log
    │   ├── kibana-9d69668d4-2rkvz.yaml
    │   └── kibana-proxy
    │       └── kibana-proxy
    │           └── logs
    │               ├── current.log
    │               ├── previous.insecure.log
    │               └── previous.log
    └── route.openshift.io
        └── routes.yaml
└── openshift-operators-redhat
    ├── ...
```

3. 从工作目录中刚刚创建的 **must-gather** 目录创建一个压缩文件。例如，在使用 Linux 操作系统的计算机上运行以下命令：

```
$ tar cvaf must-gather.tar.gz must-gather.local.5421342344627712289/ 1
```

1 务必将 **must-gather-local.5421342344627712289/** 替换为实际目录名称。

4. 在红帽客户门户中为您的问题单附上压缩文件。

5.1.3. 收集审计日志

您可以收集审计日志，它们是与安全相关的按时间排序的记录，记录各个用户、管理员或其他系统组件影响系统的一系列活动。您可以收集以下的审计日志：

- etcd 服务器
- Kubernetes API 服务器
- OpenShift OAuth API 服务器
- OpenShift API 服务器

流程

1. 使用 **-- /usr/bin/gather_audit_logs** 标志运行 **oc adm must-gather** 命令：

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

2. 从工作目录中刚刚创建的 **must-gather** 目录创建一个压缩文件。例如，在使用 Linux 操作系统的计算机上运行以下命令：

```
$ tar cvaf must-gather.tar.gz must-gather.local.472290403699006248 1
```

1 将 **must-gather-local.472290403699006248** 替换为实际目录名称。

3. 在红帽客户门户中为您的问题单附上压缩文件。

5.2. 获取集群 ID

在向红帽支持提供信息时，提供集群的唯一标识符会很有帮助。您可以使用 OpenShift Container Platform Web 控制台自动填充集群 ID。您还可以使用 web 控制台或 OpenShift CLI（**oc**）手工获取集群 ID。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 访问安装的 web 控制台或 OpenShift CLI（**oc**）。

流程

- 使用 Web 控制台开支持问题单并自动填充集群 ID：
  a. 从工具栏导航至 （？） **help → Open Support Case**。
  b. **Cluster ID** 的值会被自动填充 。
- 使用 web 控制台手动获取集群 ID：
  a. 导航到 **Home → Dashboards → Overview**。
  b. 该值包括在 **Details** 中的 **Cluster ID** 项中。
- 要使用 OpenShift CLI（**oc**）获取集群 ID，请运行以下命令：

```
$ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
```

5.3. 关于 SOSREPORT

sosreport 是一个从 Red Hat Enterprise Linux((RHEL)和 Red Hat Enterprise Linux CoreOS(RHCOS)系统收集配置详情、系统信息和诊断数据的工具。**sosreport** 提供了一种标准的方法来收集有关节点的诊断信息,然后可以提供给红帽支持以进行诊断。

在一些情况下，红帽支持可能会要求您为特定 OpenShift Container Platform 节点收集 **sosreport** 归档。例如，有时可能需要查看系统日志或其他没有包括在 **oc adm must-gather** 输出的针对于节点的特定数据。

5.4. 为 OPENSHIFT CONTAINER PLATFORM 集群节点生成 SOSREPORT 归档

为 OpenShift Container Platform 4.6 集群节点生成 **sosreport** 的建议方法是通过 debug pod。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您需要有到主机的 SSH 访问权限。
- 已安装 OpenShift CLI（**oc**）。
- 您有红帽标准订阅或高级订阅。
- 您有红帽客户门户网站帐户。
- 您已有一个红帽支持问题单 ID。

流程

1. 获取集群节点列表：

```
$ oc get nodes
```

2. 在目标节点上进入一个 debug 会话。此步骤被实例化为一个名为 **<node_name>-debug** 的 debug pod:

```
$ oc debug node/my-cluster-node
```

要在目标节点上进入带有 **NoExecute** 效果的 debug 会话，请向 dummy 命名空间添加一个容限，并在 dummy 命名空间中启动 debug pod：

```
$ oc new-project dummy
```

```
$ oc patch namespace dummy --type=merge -p '{"metadata": {"annotations": {
"scheduler.alpha.kubernetes.io/defaultTolerations": "[{\"operator\": \"Exists\"}]"}}}'
```

```
$ oc debug node/my-cluster-node
```

3. 将 **/host** 设置为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

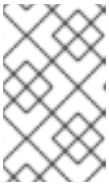
```
# chroot /host
```

注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

4. 启动 **toolbox** 容器，其中包括运行 **sosreport** 所需的二进制文件和插件：

```
# toolbox
```



注意

如果一个已存在的 **toolbox** pod 已在运行，则 **toolbox** 命令会输出 **'toolbox-' already exists.Trying to start...**使用 **podman rm toolbox-** 删除正在运行的 toolbox容器，并生成新的 toolbox 容器以避免 **sosreport** 插件出现问题。

5. 收集 **sosreport** 归档。

   a. 运行 **sosreport** 命令并启用 **crio.all** 和 **crio.logs** CRI-O 容器引擎 **sosreport** 插件：

```
# sosreport -k crio.all=on -k crio.logs=on (1)
```

   (1) **-k** 可让您在默认值之外定义 **sosreport** 插件参数。

   b. 提示后按 **Enter** 键继续。

   c. 提供红帽支持问题单 ID。**sosreport** 将 ID 添加到存档的文件名中。

   d. **sosreport** 输出提供了归档的位置和 checksum。以下示例输出引用支持问题单 ID **01234567**:

```
Your sosreport has been generated and saved in:
  /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz (1)

The checksum is: 382ffc167510fd71b4f12a4f40b97a4e
```

   (1) **sosreport** 归档的文件路径在 **chroot** 环境之外，因为 toolbox 容器会在 **/host** 挂载主机的根目录。

6. 使用以下方法之一为红帽支持提供 **sosreport** 归档以供分析。

   - 将文件直接从 OpenShift Container Platform 集群上传到现有红帽支持问题单。

     a. 在 toolbox 容器内，运行 **redhat-support-tool** 将存档直接附加到现有红帽支持问题单中。这个示例使用问题单 ID **01234567**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-sosreport.tar.xz
(1)
```

1 toolbox 容器将主机的根目录挂载到 **/host**。当指定要通过 **redhat-support-tool** 命令上传的文件时，使用 toolbox 容器的根目录（包括 **/host/**）的绝对路径。

- 将文件上传到现有红帽支持问题单中。

a. 运行 **oc debug node/<node_name>** 命令调整 **sosreport** 归档，并将输出重定向到文件中。此命令假设您已退出以前的 **oc debug** 会话：

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz' > /tmp/sosreport-my-cluster-node-01234567-2020-05-28-eyjknxt.tar.xz 1
```

1 debug 容器将主机的根目录挂载到 **/host**。在指定用于连接的目标文件时，引用 debug 容器的根目录的绝对路径，包括 **/host**。

注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 **scp** 从集群节点传输 **sosreport** 归档，这会使节点出现 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以通过运行 **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** 从节点复制 **sosreport** 归档文件。

b. 进入 https://access.redhat.com/support/cases/ 中的现有支持问题单。

c. 选择 **Attach files** 并按提示上传该文件。

5.5. 查询 BOOTSTRAP 节点日志

如果遇到与 bootstrap 相关的问题，可以从 bootstrap 节点收集 **bootkube.service** **journald** 单元日志和容器日志。

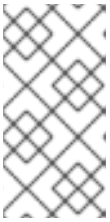
先决条件

- 具有到 bootstrap 节点的 SSH 访问权限。
- 具有 bootstrap 节点的完全限定域名。

流程

1. 在 OpenShift Container Platform 安装过程中，查询 bootstrap 节点的 **bootkube.service** **journald** 单元日志。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



注意

bootstrap 节点上的 **bootkube.service** 日志会输出 etcd **connection refused** 错误，这表示 bootstrap 服务器无法在 control plane 节点（也称为 master 节点）上连接到 etcd。在各个 control plane 节点上启动 etcd 且节点已加入集群后，这个错误应该会停止。

2. 使用 bootstrap 节点上的 **podman** 从 bootstrap 节点容器收集日志。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

```
$ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done'
```

5.6. 查询集群节点 JOURNAL 日志

您可以在独立集群节点的 **/var/log** 中收集 **journald** 单元日志和其他日志。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。

流程

1. 查询 OpenShift Container Platform 集群节点的 **kubelet journald** 单元日志。以下示例仅查询 control plane 节点（也称为 master 节点）：

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 根据情况替换 **kubelet** 以查询其他单元日志。

2. 从集群节点上 **/var/log/** 下的特定子目录收集日志。

   a. 获取 **/var/log/** 子目录中所含的日志列表。以下示例列出所有 control plane 节点上的 **/var/log/openshift-apiserver/** 中的文件：

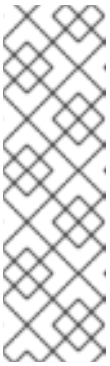
```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

   b. 检查 **/var/log/** 子目录中的特定日志。以下示例输出来自所有 control plane 节点的 **/var/log/openshift-apiserver/audit.log** 内容：

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

   c. 如果 API 无法正常工作，请使用 SSH 来查看每个节点上的日志。以下示例是 **/var/log/openshift-apiserver/audit.log** 的尾部的内容：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

5.7. 从 OPENSHIFT CONTAINER PLATFORM 节点或容器收集网络追踪（TRACE）

在调查与网络相关的 OpenShift Container Platform 问题时，红帽可能会从特定的 OpenShift Container Platform 集群节点或从特定容器请求网络数据包追踪。在 OpenShift Container Platform 中捕获网络 trace 的建议方法是通过 debug pod。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您有红帽标准订阅或高级订阅。
- 您有红帽客户门户网站帐户。
- 您已有一个红帽支持问题单 ID。
- 您需要有到主机的 SSH 访问权限。

流程

1. 获取集群节点列表：

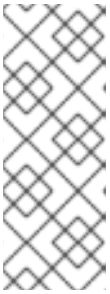
```
$ oc get nodes
```

2. 在目标节点上进入一个 debug 会话。此步骤被实例化为一个名为 **<node_name>-debug** 的 debug pod:

```
$ oc debug node/my-cluster-node
```

3. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

```
# chroot /host
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

4. 在 **chroot** 环境控制台中获取节点接口名称：

```
# ip ad
```

5. 启动 **toolbox** 容器，其中包括运行 **sosreport** 所需的二进制文件和插件：

```
# toolbox
```

注意

如果一个已存在的 **toolbox** pod 已在运行，则 **toolbox** 命令会输出 **'toolbox-' already exists.Trying to start...**要避免 **tcpdump** 出现问题，请使用 **podman rm toolbox-** 删除正在运行的 toolbox 容器，并生成新 toolbox 容器。

6. 在集群节点中启动 **tcpdump** 会话，并将输出重定向到捕获文件中。这个示例使用 **ens5** 作为接口名称：

```
$ tcpdump -nn -s 0 -i ens5 -w /host/var/tmp/my-cluster-node_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap 1
```

1 **tcpdump** 捕获文件路径在 **chroot** 环境之外，因为 toolbox 容器会在 **/host** 中挂载主机的根目录。

7. 如果节点上的特定容器需要 **tcpdump** 捕获，请按照以下步骤操作。

   a. 确定目标容器 ID。**chroot host** 命令先于这一步中的 **crictl** 命令，因为 toolbox 容器在 **/host** 中挂载主机的根目录：

   ```
   # chroot /host crictl ps
   ```

   b. 确定容器的进程 ID。在本例中，容器 ID 是 **a7fe32346b120**:

   ```
   # chroot /host crictl inspect --output yaml a7fe32346b120 | grep 'pid' | awk '{print $2}'
   ```

   c. 在容器上启动 **tcpdump** 会话，并将输出重定向到捕获文件中。本例使用 **49628** 作为容器的进程 ID，**ens5** 是接口名称。**nsenter** 命令进入目标进程的命名空间并在命名空间中运行命令。因为本例中的目标进程是一个容器的进程 ID，**tcpdump** 命令从主机在容器的命名空间中运行：

   ```
   # nsenter -n -t 49628 -- tcpdump -nn -i ens5 -w /host/var/tmp/my-cluster-node-my-container_$(date +%d_%m_%Y-%H_%M_%S-%Z).pcap.pcap 1
   ```

1 **tcpdump** 捕获文件路径在 **chroot** 环境之外，因为 toolbox 容器会在 **/host** 中挂载主机的根目录。

8. 使用以下方法之一向红帽支持提供 **tcpdump** 捕获文件进行分析。

- 将文件直接从 OpenShift Container Platform 集群上传到现有红帽支持问题单。

a. 在 toolbox 容器内，运行 **redhat-support-tool** 将该文件直接附加到现有红帽支持问题单中。这个示例使用问题单 ID **01234567**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-tcpdump-capture-file.pcap 1
```

1 toolbox 容器将主机的根目录挂载到 **/host**。当指定要通过 **redhat-support-tool** 命令上传的文件时，使用 toolbox 容器的根目录（包括 **/host/**）的绝对路径。

- 将文件上传到现有红帽支持问题单中。

a. 运行 **oc debug node/<node_name>** 命令调整 **sosreport** 归档，并将输出重定向到文件中。此命令假设您已退出以前的 **oc debug** 会话：

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-tcpdump-capture-file.pcap' > /tmp/my-tcpdump-capture-file.pcap 1
```

1 debug 容器将主机的根目录挂载到 **/host**。在指定用于连接的目标文件时，引用 debug 容器的根目录的绝对路径，包括 **/host**。

注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 **scp** 从集群节点传输 **tcpdump** 捕获文件，这会使节点出现 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以运行 **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** 从一个节点复制 **tcpdump** 捕获文件。

b. 进入 https://access.redhat.com/support/cases/ 中的现有支持问题单。

c. 选择 **Attach files** 并按提示上传该文件。

5.8. 为红帽支持提供诊断数据

在解决 OpenShift Container Platform 问题时，红帽可能会要求您将诊断数据上传到支持问题单中。文件可以通过红帽客户门户网站或使用 **redhat-support-tool** 命令直接从 OpenShift Container Platform 集群上传到支持问题单。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 您需要有到主机的 SSH 访问权限。
- 已安装 OpenShift CLI（**oc**）。
- 您有红帽标准订阅或高级订阅。
- 您有红帽客户门户网站帐户。
- 您已有一个红帽支持问题单 ID。

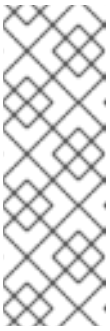
流程

- 通过红帽客户门户网站将诊断数据上传到现有红帽支持问题单中。

1. 使用 **oc debug node/<node_name>** 命令调整一个 OpenShift Container Platform 节点中包含的诊断文件，并将输出重定向到文件中。以下示例将 debug 容器中的 **/host/var/tmp/my-diagnostic-data.tar.gz** 复制到 **/var/tmp/my-diagnostic-data.tar.gz**:

```
$ oc debug node/my-cluster-node -- bash -c 'cat /host/var/tmp/my-diagnostic-data.tar.gz'
> /var/tmp/my-diagnostic-data.tar.gz 1
```

1 debug 容器将主机的根目录挂载到 **/host**。在指定用于连接的目标文件时，引用 debug 容器的根目录的绝对路径，包括 **/host**。



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 **scp** 从集群节点传输文件，这会使节点出现 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **scp core@<node>.<cluster_name>.<base_domain>:<file_path> <local_path>** 从一个节点复制诊断文件。

2. 进入 https://access.redhat.com/support/cases/ 中的现有支持问题单。
3. 选择 **Attach files** 并按提示上传该文件。

- 将诊断数据直接从 OpenShift Container Platform 集群上传到现有红帽支持问题单中。

1. 获取集群节点列表：

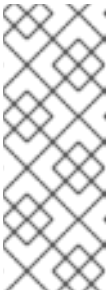
```
$ oc get nodes
```

2. 在目标节点上进入一个 debug 会话。此步骤被实例化为一个名为 **<node_name>-debug** 的 debug pod:

```
$ oc debug node/my-cluster-node
```

3. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

```
# chroot /host
```

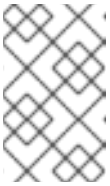


注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

4. 启动 **toolbox** 容器，其中包含运行 **redhat-support-tool** 所需的二进制文件：

```
# toolbox
```



注意

如果一个已存在的 **toolbox** pod 已在运行，则 **toolbox** 命令会输出 **'toolbox-' already exists.Trying to start...**.使用 **podman rm toolbox-** 删除正在运行的 toolbox 容器，并生成新的 toolbox 容器以避免出现问题。

a. 运行 **redhat-support-tool** 将 debug pod 的文件直接附加到现有的红帽支持问题单中。这个示例使用支持问题单 ID '01234567' 和示例文件路径 **/host/var/tmp/my-diagnostic-data.tar.gz**:

```
# redhat-support-tool addattachment -c 01234567 /host/var/tmp/my-diagnostic-data.tar.gz (1)
```

(1) toolbox 容器将主机的根目录挂载到 **/host**。当指定要通过 **redhat-support-tool** 命令上传的文件时，使用 toolbox 容器的根目录（包括 **/host/**）的绝对路径。

第 6 章 集群规格总结

6.1. 通过 CLUSTERVERSION 总结集群规格

您可以通过查询 **clusterversion** 资源来获取一个 OpenShift Container Platform 集群规格的总结数据。

先决条件

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

流程

1. 查询集群版本、可用性、运行时间以及常规状态：

```
$ oc get clusterversion
```

2. 获取集群规格、更新可用性和更新历史记录的详细概述：

```
$ oc describe clusterversion
```

第 7 章 故障排除

7.1. 安装故障排除

7.1.1. 确定安装问题在哪里发生

当对 OpenShift Container Platform 安装问题进行故障排除时，可以监控安装日志来确定在哪个阶段出现问题。然后，检索与该阶段相关的诊断数据。

OpenShift Container Platform 安装过程包括以下阶段：

1. 创建 Ignition 配置文件。
2. 启动引导计算机并开始托管要启动的 control plane 机器（也称为 master 机器）所需的远程资源。
3. control plane 机器从 bootstrap 机器获取远程资源并完成启动。
4. control plane 机器使用 bootstrap 机器组成 etcd 集群。
5. bootstrap 机器使用新的 etcd 集群启动临时 Kubernetes control plane。
6. 临时 control plane 将生产环境 control plane 调度到 control plane 机器。
7. 临时 control plane 关机，并将控制权交给生产环境 control plane。
8. bootstrap 机器将 OpenShift Container Platform 组件添加到生产环境 control plane。
9. 安装程序关闭 bootstrap 机器。
10. control plane 设置 worker 节点。
11. control plane 以一组 Operator 的形式安装其他服务。
12. 集群下载并配置日常运作所需的其余组件，包括在受支持的环境中创建 worker 机器。

7.1.2. 用户置备的基础架构安装注意事项

默认安装方法使用安装程序置备的基础架构。对于安装程序置备的基础架构的集群，OpenShift Container Platform 可以管理集群的所有方面，包括操作系统本身。若有可能，可以使用此功能来避免置备和维护集群基础架构。

您也可以在自己提供的基础架构上安装 OpenShift Container Platform 4.6。如果您使用这个安装方法，请完全遵循用户置备的基础架构安装文档中的内容。另外，在安装前请考虑以下事项：

- 查看 Red Hat Enterprise Linux（RHEL）生态系统 中的内容来决定您所使用的服务器硬件或虚拟环境可以获得的 Red Hat Enterprise Linux CoreOS（RHCOS）对其支持的级别。
- 很多虚拟化和云环境需要在客户端操作系统中安装代理。确保将这些代理作为通过守护进程集部署的容器化工作负载安装。
- 如果要启用动态存储、按需服务路由、节点主机名到 Kubernetes 主机名解析以及集群自动扩展等功能，请安装云供应商集成。

注意

不可能在混合有不同云供应商，或跨多个物理或虚拟平台的 OpenShift Container Platform 环境中启用云供应商集成。节点生命周期控制器不允许将来自现有供应商外部的节点添加到集群中，且无法指定多个云供应商集成。

- 如果要使用机器集或自动扩展来自动置备 OpenShift Container Platform 集群节点，则需要针对于供应商的 Machine API 实现。
- 检查您所选云供应商是否提供了将 Ignition 配置文件注入主机的方法作为其初始部署的一部分。如果不这样做，则需要使用 HTTP 服务器托管 Ignition 配置文件。解决 Ignition 配置文件问题的步骤会因部署了这两种方法的不同而有所不同。
- 如果要使用可选的框架组件，如嵌入式容器 registry、Elasticsearch 或 Prometheus 等可选框架组件，则需要手动置备存储。除非明确进行了配置，用户置备的基础架构安装中不定义默认存储类。
- 在高可用性 OpenShift Container Platform 环境中，需要一个负载均衡器来在所有 control plane 节点（也称为 master 节点）之间分发 API 请求。您可以使用任何满足 OpenShift Container Platform DNS 路由和端口要求的基于 TCP 的负载平衡解决方案。

7.1.3. 在 OpenShift Container Platform 安装前检查负载均衡器配置

在开始 OpenShift Container Platform 安装前，请检查您的负载均衡器配置。

先决条件

- 已根据选择配置了外部负载均衡器，准备 OpenShift Container Platform 安装。以下示例基于使用 HAProxy 为集群提供负载均衡服务的 Red Hat Enterprise Linux（RHEL）主机。
- 为准备 OpenShift Container Platform 安装已配置了 DNS 准备。
- 有到负载均衡器的 SSH 访问权限。

流程

1. 检查 **haproxy** systemd 服务是否活跃：

```
$ ssh <user_name>@<load_balancer> systemctl status haproxy
```

2. 验证负载均衡器是否在监听所需端口。以下示例引用端口 **80**、**443**、**6443** 和 **22623**。
   - 对于在 Red Hat Enterprise Linux（RHEL）6 上运行的 HAProxy 实例，请使用 **netstat** 命令验证端口状态：

```
$ ssh <user_name>@<load_balancer> netstat -nltupe | grep -E ':80|:443|:6443|:22623'
```

   - 对于在 Red Hat Enterprise Linux（RHEL）7 或 8 上运行的 HAProxy 实例，请使用 **ss** 命令验证端口状态：

```
$ ssh <user_name>@<load_balancer> ss -nltupe | grep -E ':80|:443|:6443|:22623'
```

注意

红帽建议在 Red Hat Enterprise Linux(RHEL)7 或更高版本中使用 **ss** 命令而不是 **netstat**。**SS** 由 iproute 软件包提供。有关 **ss** 命令的详情请参考 Red Hat Enterprise Linux（RHEL）7 性能调节指南。

3. 检查通配符 DNS 记录是否被解析为负载均衡器：

```
$ dig <wildcard_fqdn> @<dns_server>
```

7.1.4. 指定 OpenShift Container Platform 安装程序日志级别

默认情况下，OpenShift Container Platform 安装程序日志级别设置为 **info**。如果在诊断失败的 OpenShift Container Platform 安装时需要更详细的日志记录，您可以在重新开始安装时将 **openshift-install** 日志级别提高为 **debug**。

先决条件

- 有访问安装主机的访问权限。

流程

- 在启动安装时将安装日志级别设置为 **debug**:

```
$ ./openshift-install --dir <installation_directory> wait-for bootstrap-complete --log-level debug
1
```

1 可能的日志级别包括 **info**、**warn** 、**error** 和 **debug**。

7.1.5. openshift-install 命令问题故障排除

如果您在运行 **openshift-install** 命令时遇到问题，请检查以下内容：

- 安装过程在 Ignition 配置文件创建后 24 小时内启动。运行以下命令时创建 Ignition 文件：

```
$ ./openshift-install create ignition-configs --dir=./install_dir
```

- **install-config.yaml** 文件与安装程序位于同一个目录中。如果使用 **./openshift-install --dir** 选项声明了其他不同的安装路径，请验证 **install-config.yaml** 文件是否存在于该目录中。

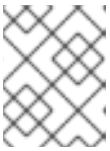
7.1.6. 监控安装进度

您可以在 OpenShift Container Platform 安装过程中监控高级别安装、bootstrap 和 control plane 日志。这提高了安装过程的可视性，并有助于识别安装失败的阶段。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。

- 您有 bootstrap 和 control plane 节点的完全限定域名（也称为 master 节点）。



注意

初始 **kubeadmin** 密码可在安装主机的 **<install_directory>/auth/kubeadmin-password** 中找到。

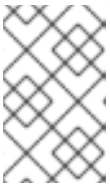
流程

1. 在安装过程中监控安装日志：

```
$ tail -f ~/<installation_directory>/.openshift_install.log
```

2. 在 bootstrap 节点引导后，监控 **bootkube.service** journald 单元日志。这可让您了解第一个 control plane 的 bootstrap。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

```
$ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
```



注意

bootstrap 节点上的 **bootkube.service** 日志会输出 etcd **connection refused** 错误，这表示 bootstrap 服务器无法在 control plane 节点上连接到 etcd。在各个 control plane 节点上启动 etcd 且节点已加入集群后，这个错误应该会停止。

3. 引导后，监控 control plane 节点上的 **kubelet.service** journald 单元日志。这可让您了解 control plane 节点代理活动。

   a. 使用 **oc** 监控日志：

```
$ oc adm node-logs --role=master -u kubelet
```

   b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
```

4. 引导后，监控 control plane 节点上的 **crio.service** journald 单元日志。这可让您了解 control plane 节点 CRI-O 容器运行时的活动。

   a. 使用 **oc** 监控日志：

```
$ oc adm node-logs --role=master -u crio
```

   b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@master-N.cluster_name.sub_domain.domain journalctl -b -f -u crio.service
```

7.1.7. 收集 bootstrap 节点诊断数据

当遇到与 bootstrap 相关的问题，可以从 bootstrap 节点收集 **bootkube.service journald** 单元日志和容器日志。

先决条件

- 具有到 bootstrap 节点的 SSH 访问权限。
- 具有 bootstrap 节点的完全限定域名。
- 如果使用 HTTP 服务器托管 Ignition 配置文件，则必须具有 HTTP 服务器的完全限定域名和端口号。还必须有到 HTTP 主机的 SSH 访问权限。

流程

1. 如果可以访问 bootstrap 节点的控制台，请监控控制台，直到节点可以提供登录提示。
2. 验证 Ignition 文件配置。
   - 如果您使用 HTTP 服务器托管 Ignition 配置文件。
     a. 验证 bootstrap 节点 Ignition 文件 URL。将 **<http_server_fqdn>** 替换为 HTTP 服务器的完全限定域名：

     ```
     $ curl -I http://<http_server_fqdn>:<port>/bootstrap.ign 1
     ```

     1 **-I** 选项只返回标头。如果指定的 URL 中有 Ignition 文件，该命令会返回 **200 OK** 状态。如果没有，该命令会返回 **404 file not found**。

     b. 要验证 bootstrap 节点是否收到 Ignition 文件，请查询服务主机上的 HTTP 服务器日志。例如，如果您使用 Apache web 服务器提供 Ignition 文件，请输入以下命令：

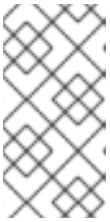
     ```
     $ grep -is 'bootstrap.ign' /var/log/httpd/access_log
     ```

     如果收到 bootstrap Ignition 文件，相关的 **HTTP GET** 日志消息将包含 **200 OK** 成功状态，这表示请求成功。

     c. 如果未收到 Ignition 文件，请检查 Ignition 文件是否存在，是否在服务主机上具有适当的文件和 Web 服务器权限。

   - 如果您使用云供应商机制将 Ignition 配置文件注入主机作为其初始部署的一部分。
     a. 查看 bootstrap 节点的控制台，以确定机制是否正确注入 bootstrap 节点 Ignition 文件。
3. 验证 bootstrap 节点分配的存储设备是否可用。
4. 验证 bootstrap 节点是否已从 DHCP 服务器分配了一个 IP 地址。
5. 从 bootstrap 节点收集 **bootkube.service** journald 单元日志。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

   ```
   $ ssh core@<bootstrap_fqdn> journalctl -b -f -u bootkube.service
   ```



注意

bootstrap 节点上的 **bootkube.service** 日志会输出 etcd **connection refused** 错误，这表示 bootstrap 服务器无法在 control plane 节点（也称为 master 节点）上连接到 etcd。在各个 control plane 节点上启动 etcd 且节点已加入集群后，这个错误应该会停止。

6. 从 bootstrap 节点容器收集日志。

   a. 使用 bootstrap 节点上的 **podman** 来收集日志。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

   ```
   $ ssh core@<bootstrap_fqdn> 'for pod in $(sudo podman ps -a -q); do sudo podman logs $pod; done'
   ```

7. 如果 bootstrap 过程失败，请验证以下内容。

   - 您可从安装主机解析 **api.<cluster_name>.<base_domain>**。
   - 负载均衡器代理端口 6443 连接到 bootstrap 和 control plane 节点。确保代理配置满足 OpenShift Container Platform 安装要求。

7.1.8. 调查 control plane 节点安装问题

如果您遇到 control plane 节点（也称为 master 节点）安装问题，请确定 control plane 节点 OpenShift Container Platform 软件定义的网络 (SDN) 和网络 Operator 状态。收集 **kubelet.service**、**crio.service** journald 单元日志和 control plane 节点容器日志，以检查 control plane 节点代理、CRI-O 容器运行时和 pod 的情况。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。
- 您有 bootstrap 和 control plane 节点的完全限定域名。
- 如果使用 HTTP 服务器托管 Ignition 配置文件，则必须具有 HTTP 服务器的完全限定域名和端口号。还必须有到 HTTP 主机的 SSH 访问权限。

注意

初始 **kubeadmin** 密码可在安装主机的 **<install_directory>/auth/kubeadmin-password** 中找到。

流程

1. 如果您可以访问 control plane 节点的控制台，请监控控制台，直到节点可以提供登录提示。在安装过程中，Ignition 日志消息输出到控制台。
2. 验证 Ignition 文件配置。
   - 如果您使用 HTTP 服务器托管 Ignition 配置文件。

a. 验证 control plane 节点 Ignition 文件 URL。将 **<http_server_fqdn>** 替换为 HTTP 服务器的完全限定域名：

```
$ curl -I http://<http_server_fqdn>:<port>/master.ign 1
```

1 **-I** 选项只返回标头。如果指定的 URL 中有 Ignition 文件，该命令会返回 **200 OK** 状态。如果没有，该命令会返回 **404 file not found**。

b. 要验证 control plane 节点是否收到 Ignition 文件，请查询服务主机上的 HTTP 服务器日志。例如，如果您使用 Apache web 服务器提供 Ignition 文件：

```
$ grep -is 'master.ign' /var/log/httpd/access_log
```

如果收到 master Ignition 文件，相关的 **HTTP GET** 日志消息将包含 **200 OK** 成功状态，表示请求成功。

c. 如果未收到 Ignition 文件，请检查是否直接存在于服务主机上。确定有适当的文件权限和网页服务器权限。

- 如果您使用云供应商机制将 Ignition 配置文件注入主机作为其初始部署的一部分。

a. 查看 control plane 节点的控制台，以确定机制是否正确注入 control plane 节点 Ignition 文件。

3. 检查分配给 control plane 节点的存储设备是否可用。

4. 验证 control plane 节点是否已从 DHCP 服务器分配了一个 IP 地址。

5. 确定 control plane 节点状态。

a. 查询 control plane 节点状态：

```
$ oc get nodes
```

b. 如果一个 control plane 节点没有处于 **Ready** 状态，检索详细的节点描述：

```
$ oc describe node <master_node>
```

注意

如果某个安装问题导致 OpenShift Container Platform API 无法运行，或者 kubelet 还没有在每个节点上运行，则无法运行 **oc** 命令：

6. 确定 OpenShift Container Platform SDN 状态。

a. 在 **openshift-sdn** 命名空间内查看 **sdn-controller**、**sdn** 和 **ovs** 守护进程集的状态：

```
$ oc get daemonsets -n openshift-sdn
```

b. 如果这些资源列为 **Not found**，查看 **openshift-sdn** 命名空间中的 pod：

```
$ oc get pods -n openshift-sdn
```

c. 检查 **openshift-sdn** 命名空间中与失败的 OpenShift Container Platform SDN pod 相关的日志：

```
$ oc logs <sdn_pod> -n openshift-sdn
```

7. 确定集群网络配置状态。

a. 查看集群的网络配置是否存在：

```
$ oc get network.config.openshift.io cluster -o yaml
```

b. 如果安装程序无法创建网络配置，请再次生成 Kubernetes 清单并查看消息输出：

```
$ ./openshift-install create manifests
```

c. 查看 **openshift-network-operator** 命名空间中的 pod 状态，以确定 Cluster Network Operator（CNO）是否在运行：

```
$ oc get pods -n openshift-network-operator
```

d. 从 **openshift-network-operator** 命名空间中收集网络 Operator pod 日志：

```
$ oc logs pod/<network_operator_pod_name> -n openshift-network-operator
```

8. 引导后，监控 control plane 节点上的 **kubelet.service** journald 单元日志。这可让您了解 control plane 节点代理活动。

a. 使用 **oc** 检索日志：

```
$ oc adm node-logs --role=master -u kubelet
```

b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service
```

注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

9. 引导后，在 control plane 节点上检索 **crio.service** journald 单元日志。这可让您了解 control plane 节点 CRI-O 容器运行时的活动。

a. 使用 **oc** 检索日志：

```
$ oc adm node-logs --role=master -u crio
```

b. 如果 API 无法正常工作，使用 SSH 来查看日志：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
crio.service
```

10. 从 control plane 节点上的 **/var/log/** 下的特定子目录收集日志。

a. 获取 **/var/log/** 子目录中所含的日志列表。以下示例列出所有 control plane 节点上的 **/var/log/openshift-apiserver/** 中的文件：

```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

b. 检查 **/var/log/** 子目录中的特定日志。以下示例输出来自所有 control plane 节点的 **/var/log/openshift-apiserver/audit.log** 内容：

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. 如果 API 无法正常工作，请使用 SSH 来查看每个节点上的日志。以下示例是 **/var/log/openshift-apiserver/audit.log** 的尾部的内容：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```

11. 使用 SSH 查看 control plane 节点容器日志。

a. 列出容器：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps -a
```

b. 使用 **crictl** 检索容器日志：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

12. 如果您遇到 control plane 节点配置问题，请验证 MCO、MCO 端点和 DNS 记录是否正常工作。Machine Config Operator（MCO）在安装过程中管理操作系统配置。同时还要检查系统时钟准确性以及证书的有效性。

a. 测试 MCO 端点是否可用。将 **<cluster_name>** 替换为适当的值：

```
$ curl https://api-int.<cluster_name>:22623/config/master
```

b. 如果端点不响应，请验证负载均衡器配置。确保端点配置为在端口 22623 上运行。

c. 验证 MCO 端点的 DNS 记录是否已配置并解析到负载均衡器。

i. 对定义的 MCO 端点名称运行 DNS 查找：

```
$ dig api-int.<cluster_name> @<dns_server>
```

ii. 在负载均衡器上对分配的 MCO IP 地址进行逆向查询：

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

d. 验证 MCO 是否直接从 bootstrap 节点运行。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/master
```

e. 系统时钟时间必须在 bootstrap、master 和 worker 节点间保持同步。检查每个节点的系统时钟参考时间和时间同步统计信息：

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

f. 检查证书的有效性：

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

7.1.9. 解决 etcd 安装问题

如果在安装过程中遇到 etcd 问题，您可以检查 etcd pod 状态并收集 etcd pod 日志。您还可以验证 etcd DNS 记录并检查 control plane 节点上的 DNS 可用性（也称为 master 节点）。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。
- 您有 control plane 节点的完全限定域名。

流程

1. 检查 etcd pod 的状态。
   a. 查看 **openshift-etcd** 命名空间中的 pod 状态：

```
$ oc get pods -n openshift-etcd
```

   b. 查看 **openshift-etcd-operator** 命名空间中的 pod 状态：

```
$ oc get pods -n openshift-etcd-operator
```

2. 如果以上命令中列出的任何 pod 没有显示 **Running** 或 **Completed** 状态，请为 pod 收集诊断信息。
   a. 检查 pod 的事件：

```
$ oc describe pod/<pod_name> -n <namespace>
```

   b. 检查 pod 的日志：

```
$ oc logs pod/<pod_name> -n <namespace>
```

c. 如果 pod 有多个容器，以上命令会创建一个错误并在错误消息中提供容器名称。检查每个容器的日志：

```
$ oc logs pod/<pod_name> -c <container_name> -n <namespace>
```

3. 如果 API 无法正常工作，请使用 SSH 来查看每个 control plane 节点上的 etcd pod 和容器日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值。

a. 列出每个 control plane 节点上的 etcd pod：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods --
name=etcd-
```

b. 对于没有显示 **Ready** 状态的 pod，详细检查 pod 状态。将 **<pod_id>** 替换为上一命令输出中列出的 pod ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
<pod_id>
```

c. 列出与 pod 相关的容器：

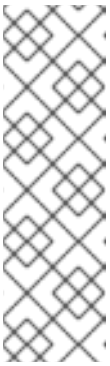
```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps | grep
'<pod_id>'
```

d. 对于没有显示 **Ready** 状态的容器，请详细检查容器状态。将 **<container_id>** 替换为上一命令输出中列出的容器 ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

e. 检查任何未显示 **Ready** 状态的容器的日志。将 **<container_id>** 替换为上一命令输出中列出的容器 ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

4. 验证 control plane 节点的主和从属 DNS 服务器连接。

7.1.10. 调查 control plane 节点 kubelet 和 API 服务器问题

要在安装过程中调查 control plane 节点（也称为 master 节点）kubelet 和 API 服务器问题，请检查 DNS、DHCP 和负载均衡器功能。另外，请确认证书没有过期。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。
- 您有 control plane 节点的完全限定域名。

流程

1. 验证 API 服务器的 DNS 记录是否将 control plane 节点上的 kubelet 定向到 **https://api-int.<cluster_name>.<base_domain>:6443**。确保记录引用负载均衡器。
2. 确保负载均衡器的端口 6443 定义引用每个 control plane 节点。
3. 检查 DHCP 是否提供了唯一的 control plane 节点主机名。
4. 检查每个 control plane 节点上的 **kubelet.service** journald 单元日志。
   a. 使用 **oc** 检索日志：

   ```
   $ oc adm node-logs --role=master -u kubelet
   ```

   b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值：

   ```
   $ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u kubelet.service
   ```

   注意

   运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

5. 检查 control plane 节点 kubelet 日志中的证书过期信息。
   a. 使用 **oc** 检索日志：

   ```
   $ oc adm node-logs --role=master -u kubelet | grep -is 'x509: certificate has expired'
   ```

   b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service  | grep -is 'x509: certificate has expired'
```

7.1.11. 调查 worker 节点安装问题

如果遇到 worker 节点安装问题，可以查看 worker 节点的状态。收集 **kubelet.service**、**crio.service** journald 单元日志和 worker 节点容器日志，以检查 worker 节点代理、CRI-O 容器运行时和 pod 的情况。另外，还可以检查 Ignition 文件和 Machine API Operator 是否正常工作。如果 worker 节点安装后配置失败，检查 Machine Config Operator（MCO）和 DNS 是否正常工作。您还可以验证 bootstrap、master 和 worker 节点之间的系统时钟是否同步，并验证证书。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。
- 您有 bootstrap 和 worker 节点的完全限定域名。
- 如果使用 HTTP 服务器托管 Ignition 配置文件，则必须具有 HTTP 服务器的完全限定域名和端口号。还必须有到 HTTP 主机的 SSH 访问权限。

注意

初始 **kubeadmin** 密码可在安装主机的 **<install_directory>/auth/kubeadmin-password** 中找到。

流程

1. 如果可以访问 worker 节点的控制台，请监控控制台，直到节点可以提供登录提示。在安装过程中，Ignition 日志消息输出到控制台。
2. 验证 Ignition 文件配置。
   - 如果您使用 HTTP 服务器托管 Ignition 配置文件。
     a. 验证 worker 节点 Ignition 文件 URL。将 **<http_server_fqdn>** 替换为 HTTP 服务器的完全限定域名：

```
$ curl -I http://<http_server_fqdn>:<port>/worker.ign 1
```

     1 **-I** 选项只返回标头。如果指定的 URL 中有 Ignition 文件，该命令会返回 **200 OK** 状态。如果没有，该命令会返回 **404 file not found**。

     b. 要验证 worker 节点是否收到 Ignition 文件，请查询 HTTP 主机上的 HTTP 服务器日志。例如，如果您使用 Apache web 服务器提供 Ignition 文件：

```
$ grep -is 'worker.ign' /var/log/httpd/access_log
```

     如果收到 worker Ignition 文件，相关的 **HTTP GET** 日志消息将包含 **200 OK** 成功状态，表示请求成功。

c. 如果未收到 Ignition 文件，请检查是否直接存在于服务主机上。确定有适当的文件权限和网页服务器权限。

- 如果您使用云供应商机制将 Ignition 配置文件注入主机作为其初始部署的一部分。

a. 查看 worker 节点的控制台，以确定机制是否正确注入 worker 节点 Ignition 文件。

3. 检查 worker 节点分配的存储设备是否可用。

4. 验证 worker 节点是否已从 DHCP 服务器分配了一个 IP 地址。

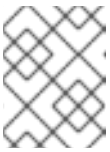
5. 确定 worker 节点状态。

a. 查询节点状态：

```
$ oc get nodes
```

b. 获取没有显示 **Ready** 状态的 worker 节点的详细节点描述：

```
$ oc describe node <worker_node>
```



注意

如果某个安装问题导致 OpenShift Container Platform API 无法运行，或者 kubelet 还没有在每个节点上运行，则无法运行 **oc** 命令。

6. 与 control plane 节点（也称为 master 节点）不同，worker 节点使用 Machine API Operator 部署和扩展。检查 Machine API Operator 的状态。

a. 查看 Machine API Operator pod 状态：

```
$ oc get pods -n openshift-machine-api
```

b. 如果 Machine API Operator pod 没有处于 **Ready** 状态，请详细列出 pod 的事件：

```
$ oc describe pod/<machine_api_operator_pod_name> -n openshift-machine-api
```

c. 检查 **machine-api-operator** 容器日志。容器在 **machine-api-operator** pod 中运行：

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c machine-api-operator
```

d. 同时检查 **kube-rbac-proxy** 容器日志。容器也会在 **machine-api-operator** pod 中运行：

```
$ oc logs pod/<machine_api_operator_pod_name> -n openshift-machine-api -c kube-rbac-proxy
```

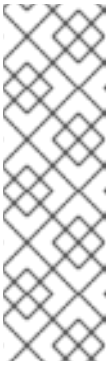
7. 引导后，监控 worker 节点上的 **kubelet.service** journald 单元日志。这可让您了解 worker 节点代理活动。

a. 使用 **oc** 检索日志：

```
$ oc adm node-logs --role=worker -u kubelet
```

b. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<worker-node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u
kubelet.service
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

8. 引导后，在 worker 节点上获取 **crio.service** journald 单元日志。这可让您了解 worker 节点 CRI-O 容器运行时的活动。

   a. 使用 **oc** 检索日志：

   ```
   $ oc adm node-logs --role=worker -u crio
   ```

   b. 如果 API 无法正常工作，使用 SSH 来查看日志：

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> journalctl -b -f -u
   crio.service
   ```

9. 从 worker 节点上的 **/var/log/** 下的特定子目录收集日志。

   a. 获取 **/var/log/** 子目录中所含的日志列表。以下示例列出所有 worker 节点上 **/var/log/sssd/** 中的文件：

   ```
   $ oc adm node-logs --role=worker --path=sssd
   ```

   b. 检查 **/var/log/** 子目录中的特定日志。以下示例输出来自所有 worker 节点的 **/var/log/sssd/audit.log** 内容：

   ```
   $ oc adm node-logs --role=worker --path=sssd/sssd.log
   ```

   c. 如果 API 无法正常工作，请使用 SSH 来查看每个节点上的日志。以下示例显示 **/var/log/sssd/sssd.log** 的尾部信息：

   ```
   $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo tail -f
   /var/log/sssd/sssd.log
   ```

10. 使用 SSH 查看 worker 节点容器日志。

    a. 列出容器：

    ```
    $ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl ps -a
    ```

b. 使用 **crictl** 检索容器日志：

```
$ ssh core@<worker-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```

11. 如果您遇到 worker 节点配置问题，检查 MCO、MCO 端点和 DNS 记录是否正常运行。Machine Config Operator（MCO）在安装过程中管理操作系统配置。同时还要检查系统时钟准确性以及证书的有效性。

a. 测试 MCO 端点是否可用。将 **<cluster_name>** 替换为适当的值：

```
$ curl https://api-int.<cluster_name>:22623/config/worker
```

b. 如果端点不响应，请验证负载均衡器配置。确保端点配置为在端口 22623 上运行。

c. 验证 MCO 端点的 DNS 记录是否已配置并解析到负载均衡器。

i. 对定义的 MCO 端点名称运行 DNS 查找：

```
$ dig api-int.<cluster_name> @<dns_server>
```

ii. 在负载均衡器上对分配的 MCO IP 地址进行逆向查询：

```
$ dig -x <load_balancer_mco_ip_address> @<dns_server>
```

d. 验证 MCO 是否直接从 bootstrap 节点运行。将 **<bootstrap_fqdn>** 替换为 bootstrap 节点的完全限定域名：

```
$ ssh core@<bootstrap_fqdn> curl https://api-int.<cluster_name>:22623/config/worker
```

e. 系统时钟时间必须在 bootstrap、master 和 worker 节点间保持同步。检查每个节点的系统时钟参考时间和时间同步统计信息：

```
$ ssh core@<node>.<cluster_name>.<base_domain> chronyc tracking
```

f. 检查证书的有效性：

```
$ openssl s_client -connect api-int.<cluster_name>:22623 | openssl x509 -noout -text
```

7.1.12. 安装后查询 Operator 状态

您可以在安装结束时检查 Operator 状态。检索不可用的 Operator 的诊断数据。检查所有列为 **Pending** 或具有错误状态的 Operator pod 的日志。验证有问题的 pod 所使用的基础镜像。

先决条件

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

流程

1. 检查安装结束时，是否所有集群 Operator 都可用。

```
$ oc get clusteroperators
```

2. 验证所有所需的证书签名请求（CSR）是否已批准。有些节点可能无法变为 **Ready** 状态，且有些集群 Operator 可能无法使用（如果待处理的 CSR）。

   a. 检查 CSR 的状态，并确保添加到集群中的每台机器都有 **Pending** 或 **Approved** 状态的客户端和服务器请求：

   ```
   $ oc get csr
   ```

   输出示例

   ```
   NAME        AGE    REQUESTOR                                                   CONDITION
   csr-8b2br   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending 1
   csr-8vnps   15m    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
   csr-bfd72   5m26s  system:node:ip-10-0-50-126.us-east-2.compute.internal         Pending 2
   csr-c57lv   5m26s  system:node:ip-10-0-95-157.us-east-2.compute.internal         Pending
   ...
   ```

   1 客户端请求 CSR。

   2 服务器请求 CSR。

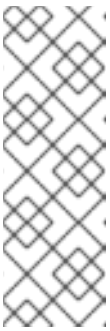
   在本例中，两台机器加入了集群。您可能会在列表中看到更多已批准的 CSR。

   b. 如果 CSR 没有获得批准，在您添加的机器的所有待处理 CSR 都处于 **Pending 状态** 后，请批准集群机器的 CSR：

   注意

   由于 CSR 会自动轮转，因此请在将机器添加到集群后一小时内批准您的 CSR。如果没有在一小时内批准它们，证书将会轮转，每个节点会存在多个证书。您必须批准所有这些证书。批准初始 CSR 后，集群的 **kube-controller-manager** 会自动批准后续的节点客户端 CSR。

   注意

   对于在未启用机器 API 的平台中运行的集群，如裸机和其他用户置备的基础架构，必须采用一种方法自动批准 kubelet 提供证书请求（CSR）。如果没有批准请求，则 **oc exec**、oc**rsh** 和 **oc logs** 命令将无法成功，因为 API 服务器连接到 kubelet 时需要服务证书。与 Kubelet 端点联系的任何操作都需要此证书批准。该方法必须监视新的 CSR，确认 CSR 由 system: **node** 或 **system:admin** 组中的 **node-bootstrapper** 服务帐户提交，并确认节点的身份。

   - 要单独批准，请对每个有效的 CSR 运行以下命令：

     ```
     $ oc adm certificate approve <csr_name> 1
     ```

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

- 要批准所有待处理的 CSR，请运行以下命令：

```
$ oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}
{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve
```

3. 查看 Operator 事件：

```
$ oc describe clusteroperator <operator_name>
```

4. 查看 Operator 命名空间中的 Operator pod 状态：

```
$ oc get pods -n <operator_namespace>
```

5. 获取没有处于 **Running** 状态的 pod 的详细描述：

```
$ oc describe pod/<operator_pod_name> -n <operator_namespace>
```

6. 检查 pod 日志：

```
$ oc logs pod/<operator_pod_name> -n <operator_namespace>
```

7. 遇到与 pod 基础镜像相关的问题时，请查看基础镜像状态。

   a. 获取有问题的 pod 使用的基础镜像的详情：

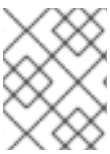
```
$ oc get pod -o "jsonpath={range .status.containerStatuses[*]}{.name}{'\t'}{.state}{'\t'}
{.image}{'\n'}{end}" <operator_pod_name> -n <operator_namespace>
```

   b. 列出基础镜像发行信息：

```
$ oc adm release info <image_path>:<tag> --commits
```

7.1.13. 从失败安装中收集日志

如果您为安装程序提供了 SSH 密钥，则可以收集有关失败安装的数据。



注意

与从正在运行的集群收集日志相比，您可以使用其他命令收集失败安装的日志。如果必须从正在运行的集群中收集日志，请使用 **oc adm must-gather** 命令。

先决条件

- 在 bootstrap 过程完成前，OpenShift Container Platform 安装会失败。bootstrap 节点正在运行，并可通过 SSH 访问。
- **ssh-agent** 进程在您的计算机上处于活跃状态，并且为 **ssh-agent** 进程和安装程序提供了相同的 SSH 密钥。

- 如果尝试在您置备的基础架构上安装集群，则必须具有 bootstrap 和 control plane 节点（也称为 master 节点）的完全限定域名。

流程

1. 生成从 bootstrap 和 control plane 机器获取安装日志的命令：

   - 如果您使用安装程序置备的基础架构，请切换到包含安装程序的目录，并运行以下命令：

     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> 1
     ```

     1 **installation_directory** 是您在运行时指定的目录 。/**openshift-install create cluster**。此目录包含安装程序创建的 OpenShift Container Platform 定义文件。

     对于安装程序置备的基础架构，安装程序会保存有关集群的信息，因此您不用指定主机名或 IP 地址。

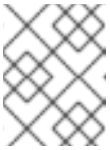
   - 如果您使用您置备的基础架构，请切换到包含安装程序的目录，并运行以下命令：

     ```
     $ ./openshift-install gather bootstrap --dir <installation_directory> \ 1
       --bootstrap <bootstrap_address> \ 2
       --master <master_1_address> \ 3
       --master <master_2_address> \ 4
       --master <master_3_address>" 5
     ```

     1 对于 **installation_directory**，请指定您在运行时指定的同一目录 。/**openshift-install create cluster**。此目录包含安装程序创建的 OpenShift Container Platform 定义文件。

     2 **<bootstrap_address>** 是集群 bootstrap 机器的完全限定域名或 IP 地址。

     3 4 5 对于集群中的每个 control plane 或 master 机器，将 **<master_*_address>** 替换为其完全限定域名或 IP 地址。

     注意

     默认集群包含三台 control plane 机器。如所示，列出所有 control plane 机器，无论您的集群使用了多少个。

     输出示例

     ```
     INFO Pulling debug logs from the bootstrap machine
     INFO Bootstrap gather logs captured here "<installation_directory>/log-bundle-
     <timestamp>.tar.gz"
     ```

     如果需要创建关安装失败的红帽支持问题单，请在问题单中附上压缩日志。

7.1.14. 其他资源

- 如需了解更多与 OpenShift Container Platform 安装类型和流程相关的详细信息，请参阅安装过程。

7.2. 验证节点健康状况

7.2.1. 查看节点状态、资源使用量和配置

查看集群节点健康状况、资源消耗统计和节点日志。另外，在单个节点上查询 **kubelet** 状态。

先决条件

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

流程

- 列出集群中所有节点的名称、状态和角色：

```
$ oc get nodes
```

- 总结集群中每个节点的 CPU 和内存使用情况：

```
$ oc adm top nodes
```

- 总结特定节点的 CPU 和内存使用情况：

```
$ oc adm top node my-node
```

7.2.2. 在节点上查询 kubelet 状态

您可以查看集群节点健康状况、资源消耗统计和节点日志。另外，您还可以在单个节点上查询 **kubelet** 状态。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

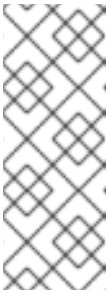
1. kubelet 通过每个节点上的 systemd 服务来管理。通过在 debug pod 中查询 **kubelet** systemd 服务来查看 kubelet 的状态。

   a. 为节点启动 debug pod:

   ```
   $ oc debug node/my-node
   ```

   b. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

   ```
   # chroot /host
   ```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 **kubelet** 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

c. 检查 **kubelet** systemd 服务是否在该节点上活跃：

```
# systemctl is-active kubelet
```

d. 输出更详细的 **kubelet.service** 状态概述：

```
# systemctl status kubelet
```

7.2.3. 查询集群节点 journal 日志

您可以在独立集群节点的 **/var/log** 中收集 **journald** 单元日志和其他日志。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。
- 您需要有到主机的 SSH 访问权限。

流程

1. 查询 OpenShift Container Platform 集群节点的 **kubelet journald** 单元日志。以下示例仅查询 control plane 节点（也称为 master 节点）：

```
$ oc adm node-logs --role=master -u kubelet 1
```

1 根据情况替换 **kubelet** 以查询其他单元日志。

2. 从集群节点上 **/var/log/** 下的特定子目录收集日志。

a. 获取 **/var/log/** 子目录中所含的日志列表。以下示例列出所有 control plane 节点上的 **/var/log/openshift-apiserver/** 中的文件：

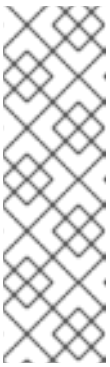
```
$ oc adm node-logs --role=master --path=openshift-apiserver
```

b. 检查 **/var/log/** 子目录中的特定日志。以下示例输出来自所有 control plane 节点的 **/var/log/openshift-apiserver/audit.log** 内容：

```
$ oc adm node-logs --role=master --path=openshift-apiserver/audit.log
```

c. 如果 API 无法正常工作，请使用 SSH 来查看每个节点上的日志。以下示例是 **/var/log/openshift-apiserver/audit.log** 的尾部的内容：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo tail -f
/var/log/openshift-apiserver/audit.log
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

7.3. CRI-O 容器运行时问题故障排除

7.3.1. 关于 CRI-O 容器运行时引擎

CRI-O 是 Kubernetes 的原生容器运行时实现，可与操作系统紧密集成来提供高效和优化的 Kubernetes 体验。CRI-O，提供用于运行、停止和重启容器的工具。

CRI-O 容器运行时引擎由在每个 OpenShift Container Platform 集群节点上使用 systemd 服务进行管理。当出现容器运行时问题时，验证每个节点上的 **crio** systemd 服务的状态。从清单容器运行时问题的节点收集 CRI-O journald 单元日志。

7.3.2. 验证 CRI-O 运行时引擎状态

您可以在每个集群节点上验证 CRI-O 容器运行时引擎状态。

先决条件

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

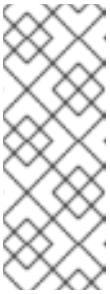
流程

1. 通过在节点上查询 debug pod 中的 **crio** systemd 服务来查看 CRI-O 状态。
   a. 为节点启动 debug pod:

   ```
   $ oc debug node/my-node
   ```

   b. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

   ```
   # chroot /host
   ```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作, **oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

c. 检查 **crio** systemd 服务在该节点上是否活跃：

```
# systemctl is-active crio
```

d. 输出更详细的 **crio.service** 状态概述：

```
# systemctl status crio.service
```

7.3.3. 收集 CRI-O journald 单元日志

如果遇到 CRI-O 问题，您可以从节点获取 CRI-O journald 单元日志。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI(**oc**)。
- 您有 control plane 或 control plane 机器的完全限定域名（也称为 master 机器）。

流程

1. 收集 CRI-O journald 单元日志。以下示例从所有 control plane 节点（在集群中）收集日志：

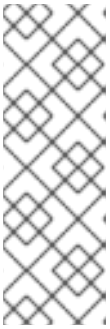
```
$ oc adm node-logs --role=master -u crio
```

2. 从特定节点收集 CRI-O journald 单元日志：

```
$ oc adm node-logs <node_name> -u crio
```

3. 如果 API 无法正常工作，使用 SSH 来查看日志。将 **<node>.<cluster_name>.<base_domain>** 替换为适当的值：

```
$ ssh core@<node>.<cluster_name>.<base_domain> journalctl -b -f -u crio.service
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作, **oc** 操作将会受到影响。在这种情况下,可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

7.3.4. 清理 CRI-O 存储

如果遇到以下问题，您可以手动清除 CRI-O 临时存储：

- 节点无法在任何 pod 上运行，并出现以下错误：

```
Failed to create pod sandbox: rpc error: code = Unknown desc = failed to mount container XXX: error recreating the missing symlinks: error reading name of symlink for XXX: open /var/lib/containers/storage/overlay/XXX/link: no such file or directory
```

- 您无法在工作节点上创建新容器，并出现 “can’t stat lower layer” 错误：

```
can't stat lower layer ... because it does not exist. Going through storage to recreate the missing symlinks.
```

- 在集群升级后或尝试重启节点时，您的节点处于 **NotReady** 状态。
- 容器运行时实施 (**crio**) 无法正常工作。
- 您无法使用 **oc debug node/<nodename>** 在节点上启动 debug shell，因为容器运行时实例 (**crio**) 无法正常工作。

按照以下步骤完全擦除 CRI-O 存储并解决错误。

先决条件：

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

流程

1. 在节点上使用 **cordon**。这是为了避免在节点处于 **Ready** 状态时调度任何工作负载。当您的 Status 部分中存在 **SchedulingDisabled** 时代表调度被禁用：

```
$ oc adm cordon <nodename>
```

2. 以 cluster-admin 用户身份排空节点：

```
$ oc adm drain <nodename> --ignore-daemonsets --delete-local-data
```

注意

pod 或 pod 模板的 **terminationGracePeriodSeconds** 属性控制恰当终止周期。此属性默认值 30 秒，但可以根据需要为每个应用程序自定义。如果设置为 90 秒以上，pod 可能会标记为 **SIGKILL**，且无法成功终止。

3. 当节点返回时，通过 SSH 或控制台连接节点。然后连接到 root 用户：

```
$ ssh core@node1.example.com
$ sudo -i
```

4. 手动停止 kubelet：

```
# systemctl stop kubelet
```

5. 停止容器和 pod：

```
# crictl rmp -fa
```

6. 手动停止 crio 服务：

```
# systemctl stop crio
```

7. 运行这些命令后，您可以完全擦除临时存储：

```
# crio wipe -f
```

8. 启动 crio 和 kubelet 服务：

```
# systemctl start crio
# systemctl start kubelet
```

9. 如果 crio 和 kubelet 服务启动，且节点处于 **Ready** 状态时，代表清理操作已正常工作：

```
$ oc get nodes
```

输出示例

```
NAME         STATUS                     ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1  Ready, SchedulingDisabled   master  133m   v1.22.0-
rc.0+75ee307
```

10. 将节点标记为可以调度。当状态中不再有 **SchedulingDisabled** 时代表启用了调度：

```
$ oc adm uncordon <nodename>
```

输出示例

```
NAME         STATUS      ROLES    AGE    VERSION
ci-ln-tkbxyft-f76d1-nvwhr-master-1   Ready          master   133m   v1.22.0-rc.0+75ee307
```

7.4. 网络问题故障排除

7.4.1. 如何选择网络接口

对于在裸机上安装，或在具有多个网络接口控制器（NIC）的虚拟机中安装时，OpenShift Container Platform 用于与 Kubernetes API 服务器通信的 NIC 由节点引导时 systemd 运行的 **nodeip-configuration.service** 服务单元决定。该服务会逐个检查节点上的每个网络接口，当第一个配置了可以用于 API 服务器的 IP 地址的子网的网络接口被选择用来进行 OpenShift Container Platform 通讯。

在 **nodeip-configuration.service** 服务确定正确的 NIC 后，该服务会创建 **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** 文件。**20-nodenet.conf** 文件将 **KUBELET_NODE_IP** 环境变量设置为服务所选的 IP 地址。

当 kubelet 服务启动时，它会从 **20-nodenet.conf** 文件中读取环境变量的值，并将 IP 地址设置为 **--node-ip** kubelet 命令行参数。因此，kubelet 服务使用所选 IP 地址作为节点 IP 地址。

如果在安装后重新配置了硬件或网络，**nodeip-configuration.service** 服务可能会在重启后选择不同的 NIC。在某些情况下，**oc get nodes -o wide** 命令的输出中的 **INTERNAL-IP** 列中可能会看到选择了一个不同的 NIC。

如果因为选择了一个不同的 NIC 而导致网络通信中断或配置错误，使用一个策略来覆盖选择的过程以明确设置正确的 IP 地址。以下列表确定了高级步骤和注意事项：

- 创建一个 shell 脚本，以确定用于 OpenShift Container Platform 通信的 IP 地址。让脚本创建一个自定义单元文件，如 **/etc/systemd/system/kubelet.service.d/98-nodenet-override.conf**。使用自定义单元文件 **98-nodenet-override.conf**，将 **KUBELET_NODE_IP** 环境变量设置为 IP 地址。

- 不要覆盖 **/etc/systemd/system/kubelet.service.d/20-nodenet.conf** 文件。指定同一目录路径中的数值较高的文件名，如 **98-nodenet-override.conf**。这样做的目标是使自定义单元文件在 **20-nodenet.conf** 之后运行，并覆盖环境变量的值。

- 使用 shell 脚本创建一个机器配置对象，作为 base64 编码字符串，并使用 Machine Config Operator 将脚本部署到文件系统路径（如 **/usr/local/bin/override-node-ip.sh** ）的节点。

- 确保 **systemctl daemon-reload** 在 shell 脚本运行后运行。最简单的方法是在机器配置中指定 **ExecStart=systemctl daemon-reload**，如下例所示。

用于覆盖 kubelet 网络接口的机器配置示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 98-nodenet-override
spec:
  config:
    ignition:
      version: 3.1.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,<encoded_script>
        mode: 0755
        overwrite: true
```

```
      path: /usr/local/bin/override-node-ip.sh
  systemd:
    units:
    - contents: |
        [Unit]
        Description=Override node IP detection
        Wants=network-online.target
        Before=kubelet.service
        After=network-online.target
        [Service]
        Type=oneshot
        ExecStart=/usr/local/bin/override-node-ip.sh
        ExecStart=systemctl daemon-reload
        [Install]
        WantedBy=multi-user.target
      enabled: true
      name: nodenet-override.service
```

7.5. TROUBLESHOOTING OPERATOR 的问题

Operator 是一种打包、部署和管理 OpenShift Container Platform 应用程序的方法。它可以被看作是软件厂商的工程团队的扩展，可以在 OpenShift Container Platform 监控软件的运行情况，并根据软件的当前状态实时做出决策。Operator 被设计为用来无缝地处理升级过程，并对出现的错误自动进行响应，而且不会采取"捷径"（如跳过软件备份过程来节省时间）。

OpenShift Container Platform 4.6 包括一组默认 Operator，这是集群正常工作所需的。这些默认 Operator 由 Cluster Version Operator（CVO）管理。

作为集群管理员，您可使用 OpenShift Container Platform Web 控制台或 CLI 安装来自 OperatorHub 的应用程序 Operator。然后，您可将 Operator 订阅至一个或多个命名空间，供集群上的开发人员使用。应用程序 Operator 由 Operator Lifecycle Manager（OLM）进行管理。

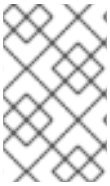
如果遇到 Operator 问题，请验证 Operator 订阅状态。检查集群中的 Operator pod 健康状况，并收集 Operator 日志以进行诊断。

7.5.1. operator 订阅状况类型

订阅可报告以下状况类型：

表 7.1. 订阅状况类型

状况	描述
CatalogSourcesUnhealthy	用于解析的一个或多个目录源不健康。
InstallPlanMissing	缺少订阅的安装计划。
InstallPlanPending	订阅的安装计划正在安装中。
InstallPlanFailed	订阅的安装计划失败。



注意

默认 OpenShift Container Platform 集群 Operator 由 Cluster Version Operator（CVO）管理，它们没有 **Subscription** 对象。应用程序 Operator 由 Operator Lifecycle Manager（OLM）管理，它们具有 **Subscription** 对象。

7.5.2. 使用 CLI 查看 Operator 订阅状态

您可以使用 CLI 查看 Operator 订阅状态。

先决条件

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

流程

1. 列出 Operator 订阅：

```
$ oc get subs -n <operator_namespace>
```

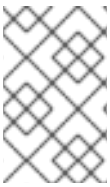
2. 使用 **oc describe** 命令检查 **Subscription** 资源：

```
$ oc describe sub <subscription_name> -n <operator_namespace>
```

3. 在命令输出中，找到 Operator 订阅状况类型的 **Conditions** 部分。在以下示例中，**CatalogSourcesUnhealthy** 条件类型具有 **false** 状态，因为所有可用目录源都健康：

输出示例

```
Conditions:
  Last Transition Time:  2019-07-29T13:42:57Z
  Message:               all available catalogsources are healthy
  Reason:                AllCatalogSourcesHealthy
  Status:                False
  Type:                  CatalogSourcesUnhealthy
```



注意

默认 OpenShift Container Platform 集群 Operator 由 Cluster Version Operator（CVO）管理，它们没有 **Subscription** 对象。应用程序 Operator 由 Operator Lifecycle Manager（OLM）管理，它们具有 **Subscription** 对象。

7.5.3. 使用 CLI 查看 Operator 目录源状态

您可以使用 CLI 查看 Operator 目录源的状态。

先决条件

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

流程

1. 列出命名空间中的目录源。例如，您可以检查 **openshift-marketplace** 命名空间，该命名空间用于集群范围的目录源：

```
$ oc get catalogsources -n openshift-marketplace
```

输出示例

```
NAME                  DISPLAY               TYPE   PUBLISHER   AGE
certified-operators   Certified Operators   grpc   Red Hat     55m
community-operators   Community Operators   grpc   Red Hat     55m
example-catalog       Example Catalog       grpc   Example Org 2m25s
redhat-marketplace    Red Hat Marketplace   grpc   Red Hat     55m
redhat-operators      Red Hat Operators     grpc   Red Hat     55m
```

2. 使用 **oc describe** 命令获取有关目录源的详情和状态：

```
$ oc describe catalogsource example-catalog -n openshift-marketplace
```

输出示例

```
Name:         example-catalog
Namespace:    openshift-marketplace
...
Status:
  Connection State:
    Address:              example-catalog.openshift-marketplace.svc:50051
    Last Connect:         2021-09-09T17:07:35Z
    Last Observed State:  TRANSIENT_FAILURE
  Registry Service:
    Created At:         2021-09-09T17:05:45Z
    Port:               50051
    Protocol:           grpc
    Service Name:       example-catalog
    Service Namespace:  openshift-marketplace
```

在上例的输出中，最后观察到的状态是 **TRANSIENT_FAILURE**。此状态表示目录源建立连接时出现问题。

3. 列出创建目录源的命名空间中的 pod：

```
$ oc get pods -n openshift-marketplace
```

输出示例

```
NAME                                  READY   STATUS             RESTARTS   AGE
certified-operators-cv9nn             1/1     Running            0          36m
community-operators-6v8lp             1/1     Running            0          36m
marketplace-operator-86bfc75f9b-jkgbc 1/1     Running            0          42m
example-catalog-bwt8z                 0/1     ImagePullBackOff   0          3m55s
redhat-marketplace-57p8c              1/1     Running            0          36m
redhat-operators-smxx8                1/1     Running            0          36m
```

在命名空间中创建目录源时，会在该命名空间中为目录源创建一个 pod。在前面的示例中，**example-catalog-bwt8z** pod 的状态是 **ImagePullBackOff**。此状态表示拉取目录源的索引镜像存在问题。

4. 使用 **oc describe** 命令检查 pod 以获取更多详细信息：

```
$ oc describe pod example-catalog-bwt8z -n openshift-marketplace
```

输出示例

```
Name:         example-catalog-bwt8z
Namespace:    openshift-marketplace
Priority:     0
Node:         ci-ln-jyryyg2-f76d1-ggdbq-worker-b-vsxjd/10.0.128.2
...
Events:
  Type     Reason          Age                From               Message
  ----     ------          ----               ----               -------
  Normal   Scheduled       48s                default-scheduler  Successfully assigned openshift-marketplace/example-catalog-bwt8z to ci-ln-jyryyf2-f76d1-fgdbq-worker-b-vsxjd
  Normal   AddedInterface  47s                multus             Add eth0 [10.131.0.40/23] from openshift-sdn
  Normal   BackOff         20s (x2 over 46s)  kubelet            Back-off pulling image "quay.io/example-org/example-catalog:v1"
  Warning  Failed          20s (x2 over 46s)  kubelet            Error: ImagePullBackOff
  Normal   Pulling         8s (x3 over 47s)   kubelet            Pulling image "quay.io/example-org/example-catalog:v1"
  Warning  Failed          8s (x3 over 47s)   kubelet            Failed to pull image "quay.io/example-org/example-catalog:v1": rpc error: code = Unknown desc = reading manifest v1 in quay.io/example-org/example-catalog: unauthorized: access to the requested resource is not authorized
  Warning  Failed          8s (x3 over 47s)   kubelet            Error: ErrImagePull
```

在前面的示例输出中，错误消息表示目录源的索引镜像因为授权问题而无法成功拉取。例如，索引镜像可能存储在需要登录凭证的 registry 中。

其它资源

- Operator Lifecycle Manager 概念和资源 → Catalog 源
- gRPC 文档：连接状态

7.5.4. 查询 Operator pod 状态

您可以列出集群中的 Operator pod 及其状态。您还可以收集详细的 Operator pod 概述。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 列出集群中运行的 Operator。输出包括 Operator 版本、可用性和运行时间信息：

```
$ oc get clusteroperators
```

2. 列出在 Operator 命名空间中运行的 Operator pod，以及 pod 状态、重启和年龄：

```
$ oc get pod -n <operator_namespace>
```

3. 输出详细的 Operator pod 概述：

```
$ oc describe pod <operator_pod_name> -n <operator_namespace>
```

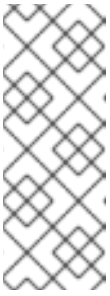
4. 如果 Operator 问题特定于某个节点，则在该节点上查询 Operator 容器状态。

   a. 为节点启动 debug pod：

```
$ oc debug node/my-node
```

   b. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

```
# chroot /host
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作, **oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

   c. 列出节点容器的详细信息，包括状态和关联的 pod ID:

```
# crictl ps
```

   d. 列出节点上特定 Operator 容器的信息。以下示例列出了 **network-operator** 容器的信息：

```
# crictl ps --name network-operator
```

   e. 退出 debug shell。

7.5.5. 收集 Operator 日志

如果遇到 Operator 问题，您可以从 Operator pod 日志中收集详细诊断信息。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI(**oc**)。
- 您有 control plane 或 control plane 机器的完全限定域名（也称为 master 机器）。

流程

1. 列出在 Operator 命名空间中运行的 Operator pod，以及 pod 状态、重启和年龄：

```
$ oc get pods -n <operator_namespace>
```

2. 检查 Operator pod 的日志：

```
$ oc logs pod/<pod_name> -n <operator_namespace>
```

如果 Operator pod 具有多个容器，则上述命令将会产生一个错误，其中包含每个容器的名称。从独立容器查询日志：

```
$ oc logs pod/<operator_pod_name> -c <container_name> -n <operator_namespace>
```

3. 如果 API 无法正常工作，请使用 SSH 来查看每个 control plane 节点上的 Operator pod 和容器日志。将 **<master-node>.<cluster_name>.<base_domain>** 替换为适当的值。

   a. 列出每个 control plane 节点上的 pod：

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl pods
```

   b. 对于任何未显示 **Ready** 状态的 Operator pod，详细检查 Pod 的状态。将 **<operator_pod_id>** 替换为上一命令输出中列出的 Operator pod ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspectp
<operator_pod_id>
```

   c. 列出与 Operator pod 相关的容器：

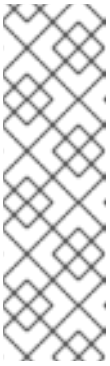
```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl ps --pod=
<operator_pod_id>
```

   d. 对于任何未显示 **Ready** 状态的 Operator 容器，请详细检查容器的状态。将 **<container_id>** 替换为上一命令输出中列出的容器 ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl inspect
<container_id>
```

   e. 检查任何未显示 **Ready** 状态的 Operator 容器的日志。将 **<container_id>** 替换为上一命令输出中列出的容器 ID:

```
$ ssh core@<master-node>.<cluster_name>.<base_domain> sudo crictl logs -f
<container_id>
```



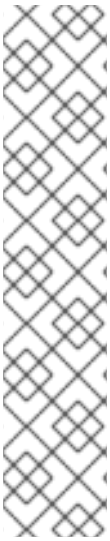
注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。在尝试通过 SSH 收集诊断数据前，请运行 **oc adm must gather** 和其他 **oc** 命令看它们是否可以提供足够的数据。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作，**oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

7.5.6. 禁用 Machine Config Operator 自动重新引导

当 Machine Config Operator 进行配置更改时，Red Hat Enterprise Linux CoreOS（RHCOS）必须重启才能使更改生效。无论配置更改是自动的（如 **kube-apiserver-to-kubelet-signer** CA 被轮转）或手动的（如更新容器镜像仓库 或 SSH 密钥），RHCOS 节点都会自动重启，除非被暂停。

为了避免不必要的中断，您可以修改机器配置池（MCP）以防止在 Operator 更改机器配置后自动重启。



注意

暂停 MCP 可防止 MCO 在关联的节点上应用任何配置更改。暂停 MCP 还可以防止任何自动轮转的证书被推送到关联的节点，包括自动轮转 **kube-apiserver-to-kubelet-signer** CA 证书。如果在 **kube-apiserver-to-kubelet-signer** CA 证书过期且 MCO 尝试自动续订证书时，MCP 会暂停，但不会在暂停的 MCP 中跨节点应用新证书。这会导致多个 **oc** 命令失败，包括但不限于 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。在暂停 MCP 时应该非常小心，需要仔细考虑 **kube-apiserver-to-kubelet-signer** CA 证书过期的问题，且仅在短时间内暂停。

新 CA 证书会在自安装日期起的 292 天生成，并在自该日期起的 365 天后删除。要确定下一个自动 CA 证书轮转，请参阅 Red Hat OpenShift 4 中的了解 CA 证书自动续订 。

kube-apiserver-to-kubelet-signer CA 轮转不会造成在 OpenShift Container Platform 版本 4.7 及更高版本中的意外的节点重启问题。

7.5.6.1. 使用控制台禁用 Machine Config Operator 自动重新引导

为了避免对 Machine Config Operator（MCO）所做的更改造成不必要的中断，您可以使用 OpenShift Container Platform Web 控制台修改机器配置池（MCP），以防止 MCO 在那个池中对节点进行任何更改。这会防止任何通常属于 MCO 更新过程一部分的重启。

注意

暂停 MCP 可防止 MCO 在关联的节点上应用任何配置更改。暂停 MCP 还可以防止任何自动轮转的证书被推送到关联的节点，包括自动轮转 **kube-apiserver-to-kubelet-signer** CA 证书。如果在 **kube-apiserver-to-kubelet-signer** CA 证书过期且 MCO 尝试自动续订证书时，MCP 会暂停，但不会在暂停的 MCP 中跨节点应用新证书。这会导致多个 **oc** 命令失败，包括但不限于 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。在暂停 MCP 时应该非常小心，需要仔细考虑 **kube-apiserver-to-kubelet-signer** CA 证书过期的问题，且仅在短时间内暂停。

新 CA 证书会在自安装日期起的 292 天生成，并在自该日期起的 365 天后删除。要确定下一个自动 CA 证书轮转，请参阅 Red Hat OpenShift 4 中的了解 CA 证书自动续订 。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

流程

要暂停或取消暂停自动 MCO 更新重新引导：

- 暂停自动引导过程：

  1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
  2. 点 **Compute → MachineConfigPools**。
  3. 在 **MachineConfigPools** 页面中，点击 **master** 或 **worker**，具体取决于您要暂停重新引导的节点。
  4. 在 **master** 或 **worker** 页面中，点 **YAML**。
  5. 在 YAML 中，将 **spec.paused** 字段更新为 **true**。

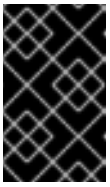
     MachineConfigPool 对象示例

     ```
     apiVersion: machineconfiguration.openshift.io/v1
     kind: MachineConfigPool
      ...
     spec:
      ...
       paused: true (1)
     ```

     (1) 将 **spec.paused** 字段更新为 **true** 以暂停重新引导。

  6. 要验证 MCP 是否已暂停，请返回到 **MachineConfigPools** 页面。
     在 **MachineConfigPools** 页面中，您修改的 MCP 报告了 **Paused** 列中为 **True**。

     如果 MCP 在暂停时有待处理的变化，**Updated** 列为 **False**，**Updating** 为 **False**。当 **Updated** 为 **True** 且 **Updating** 为 **False** 时，代表没有待处理的更改。

     重要

     如果有尚未进行的更改（**Updated** 和 **Updating** 字段都是 **False**）,建议您尽快调度一个维护窗口用于重启。使用以下步骤取消暂停自动引导过程，以应用上一次重启后排队的更改。

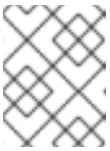
- 取消暂停自动引导过程：

  1. 以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform web 控制台。
  2. 点 **Compute → MachineConfigPools**。
  3. 在 **MachineConfigPools** 页面中，点击 **master** 或 **worker**，具体取决于您要暂停重新引导的节点。
  4. 在 **master** 或 **worker** 页面中，点 **YAML**。
  5. 在 YAML 中，将 **spec.paused** 字段更新为 **false**。

MachineConfigPool 对象示例

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfigPool
 ...
spec:
 ...
  paused: false (1)
```

(1) 将 **spec.paused** 字段更新为 **false** 以允许重启。



注意

通过取消暂停 MCP，MCO 应用所有暂停的更改，根据需要重启 Red Hat Enterprise Linux CoreOS（RHCOS）。

6. 要验证 MCP 是否已暂停，请返回到 **MachineConfigPools** 页面。
在 **MachineConfigPools** 页面中，您修改的 MCP 报告 **Paused** 列为 **False**。

   如果 MCP 正在应用任何待处理的更改，**Updated** 列为 **False**，**Updating** 列为 **True**。当 **Updated** 为 **True** 且 **Updating** 为 **False** 时，不会再进行任何更改。

7.5.6.2. 使用 CLI 禁用 Machine Config Operator 自动重新引导

为了避免对 Machine Config Operator（MCO）所做的更改造成不必要的中断，您可以使用 OpenShift CLI（oc）来修改机器配置池（MCP），以防止 MCO 在那个池中对节点进行任何更改。这会防止任何通常属于 MCO 更新过程一部分的重启。

注意

暂停 MCP 可防止 MCO 在关联的节点上应用任何配置更改。暂停 MCP 还可以防止任何自动轮转的证书被推送到关联的节点，包括自动轮转 **kube-apiserver-to-kubelet-signer** CA 证书。如果在 **kube-apiserver-to-kubelet-signer** CA 证书过期且 MCO 尝试自动续订证书时，MCP 会暂停，但不会在暂停的 MCP 中跨节点应用新证书。这会导致多个 **oc** 命令失败，包括但不限于 **oc debug**、**oc logs**、**oc exec** 和 **oc attach**。在暂停 MCP 时应该非常小心，需要仔细考虑 **kube-apiserver-to-kubelet-signer** CA 证书过期的问题，且仅在短时间内暂停。

新 CA 证书会在自安装日期起的 292 天生成，并在自该日期起的 365 天后删除。要确定下一个自动 CA 证书轮转，请参阅 Red Hat OpenShift 4 中的了解 CA 证书自动续订 。

先决条件

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

流程

要暂停或取消暂停自动 MCO 更新重新引导：

- 暂停自动引导过程：
  1. 更新 **MachineConfigPool** 自定义资源，将 **spec.paused** 字段设置为 **true**。

Control plane（master）节点

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/master
```

Worker 节点

```
$ oc patch --type=merge --patch='{"spec":{"paused":true}}' machineconfigpool/worker
```

2. 验证 MCP 是否已暂停：

Control plane（master）节点

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

Worker 节点

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

输出示例

```
true
```

spec.paused 字段为 **true**，MCP 暂停。

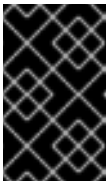
3. 确定 MCP 是否有待处理的更改：

```
# oc get machineconfigpool
```

输出示例

```
NAME     CONFIG                                             UPDATED   UPDATING
master   rendered-master-33cf0a1254318755d7b48002c597bf91   True      False
worker   rendered-worker-e405a5bdb0db1295acea08bcca33fa60   False     False
```

如果 **UPDATED** 列是 **False**，**UPDATING** 为 **False**，则有待处理的更改。当 **UPDATED** 为 **True** 且 **UPDATING** 为 **False** 时，没有待处理的更改。在上例中，worker 节点有待处理的变化。control plane 节点（也称为 master 节点）没有任何待处理的更改。



重要

如果有尚未进行的更改（ **Updated** 和 **Updating** 字段都是 **False**）,建议您尽快调度一个维护窗口用于重启。使用以下步骤取消暂停自动引导过程，以应用上一次重启后排队的更改。

- 取消暂停自动引导过程：

1. 更新 **MachineConfigPool** 自定义资源，将 **spec.paused** 字段设置为 **false**。

Control plane（master）节点

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/master
```

Worker 节点

```
$ oc patch --type=merge --patch='{"spec":{"paused":false}}' machineconfigpool/worker
```

注意

通过取消暂停 MCP，MCO 应用所有暂停的更改，根据需要重启 Red Hat Enterprise Linux CoreOS（RHCOS）。

2. 验证 MCP 是否已取消暂停：

Control plane（master）节点

```
$ oc get machineconfigpool/master --template='{{.spec.paused}}'
```

Worker 节点

```
$ oc get machineconfigpool/worker --template='{{.spec.paused}}'
```

输出示例

```
false
```

spec.paused 字段为 **false**，MCP 被取消暂停。

3. 确定 MCP 是否有待处理的更改：

```
$ oc get machineconfigpool
```

输出示例

```
NAME     CONFIG                                UPDATED  UPDATING
master   rendered-master-546383f80705bd5aeaba93   True     False
worker   rendered-worker-b4c51bb33ccaae6fc4a6a5   False    True
```

如果 MCP 正在应用任何待处理的更改，**UPDATED** 列为 **False**，**UPDATING** 列为 **True**。当 **UPDATED** 为 **True** 且 **UPDATING** 为 **False** 时，没有进一步的更改。在上例中，MCO 正在更新 worker 节点。

7.5.7. 刷新失败的订阅

在 Operator Lifecycle Manager（OLM）中，如果您订阅的是引用网络中无法访问的镜像的 Operator，您可以在 **openshift-marketplace** 命名空间中找到带有以下错误的作业：

输出示例

```
ImagePullBackOff for
Back-off pulling image "example.com/openshift4/ose-elasticsearch-operator-
bundle@sha256:6d2587129c846ec28d384540322b40b05833e7e00b25cca584e004af9a1d292e"
```

输出示例

```
rpc error: code = Unknown desc = error pinging docker registry example.com: Get
"https://example.com/v2/": dial tcp: lookup example.com on 10.0.0.1:53: no such host
```

因此，订阅会处于这个失败状态，Operator 无法安装或升级。

您可以通过删除订阅、集群服务版本（CSV）及其他相关对象来刷新失败的订阅。重新创建订阅后，OLM 会重新安装 Operator 的正确版本。

先决条件

- 您有一个失败的订阅，无法拉取不能访问的捆绑包镜像。
- 已确认可以访问正确的捆绑包镜像。

流程

1. 从安装 Operator 的命名空间中获取 **Subscription** 和 **ClusterServiceVersion** 对象的名称：

```
$ oc get sub,csv -n <namespace>
```

输出示例

```
NAME                                         PACKAGE                  SOURCE             CHANNEL
subscription.operators.coreos.com/elasticsearch-operator   elasticsearch-operator   redhat-
operators   5.0

NAME                                                  DISPLAY                  VERSION
REPLACES   PHASE
clusterserviceversion.operators.coreos.com/elasticsearch-operator.5.0.0-65   OpenShift
Elasticsearch Operator   5.0.0-65              Succeeded
```

2. 删除订阅：

```
$ oc delete subscription <subscription_name> -n <namespace>
```

3. 删除集群服务版本：

```
$ oc delete csv <csv_name> -n <namespace>
```

4. 在 **openshift-marketplace** 命名空间中获取所有失败的作业的名称和相关配置映射：

```
$ oc get job,configmap -n openshift-marketplace
```

输出示例

```
NAME                                                COMPLETIONS   DURATION   AGE
job.batch/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   1/1
26s        9m30s

NAME                                                DATA   AGE
configmap/1de9443b6324e629ddf31fed0a853a121275806170e34c926d69e53a7fcbccb   3
9m30s
```

5. 删除作业：

```
$ oc delete job <job_name> -n openshift-marketplace
```

这样可确保尝试拉取无法访问的镜像的 Pod 不会被重新创建。

6. 删除配置映射：

```
$ oc delete configmap <configmap_name> -n openshift-marketplace
```

7. 在 Web 控制台中使用 OperatorHub 重新安装 Operator。

验证

- 检查是否已成功重新安装 Operator:

```
$ oc get sub,csv,installplan -n <namespace>
```

7.6. 检查 POD 问题

OpenShift Container Platform 利用 Kubernetes 的 pod 概念，它是共同部署在同一主机上的一个或多个容器。pod 是可在 OpenShift Container Platform 4.6 上定义、部署和管理的最小计算单元。

在定义了 pod 后，它将分配到节点上运行，直到容器退出，或直到它被删除为止。根据策略和退出代码，Pod 可在退出或保留后删除，以便访问其日志。

首先要检查 pod 出现问题时 pod 的状态。如果发生 pod 故障，请观察 pod 的错误状态以识别特定镜像、容器或 pod 网络问题。根据错误状态集中诊断数据收集。查看 pod 事件消息以及 pod 和容器日志信息。通过访问命令行中运行的 pod，或根据 Pod 的部署配置启动具有 root 访问权限的调试 pod 来动态诊断问题。

7.6.1. 了解 pod 错误状态

pod 失败返回显式错误状态，可在 **oc get pods** 输出的 **status** 字段中观察到。Pod 错误状态会涵盖镜像、容器和容器网络相关的故障。

下表提供了 pod 错误状态及其描述列表。

表 7.2. Pod 错误状态

Pod 错误状态	描述
ErrImagePull	通用镜像检索错误。
ErrImagePullBackOff	镜像检索失败。
ErrInvalidImageName	指定镜像名称无效。
ErrImageInspect	镜像检查没有成功。

Pod 错误状态	描述
ErrImageNeverPull	**PullPolicy** 设置为 **NeverPullImage**，目标镜像没有本地存在。
ErrRegistryUnavailable	当尝试从 registry 检索镜像时，会出现 HTTP 错误。
ErrContainerNotFound	指定容器在声明的 pod 中不存在或未由 kubelet 管理。
ErrRunInitContainer	容器初始化失败。
ErrRunContainer	pod 的容器都没有成功启动。
ErrKillContainer	没有 pod 的容器被成功终止。
ErrCrashLoopBackOff	容器已终止。kubelet 将不会试图重启它。
ErrVerifyNonRoot	容器或镜像尝试使用 root 权限运行。
ErrCreatePodSandbox	Pod 沙盒创建没有成功。
ErrConfigPodSandbox	Pod 沙盒配置没有获得。
ErrKillPodSandbox	pod 沙箱没有成功停止。
ErrSetupNetwork	网络初始化失败。
ErrTeardownNetwork	网络终止失败。

7.6.2. 检查 pod 状态

您可以查询 pod 状态和错误状态。您还可以查询 pod 的相关部署配置，并查看基础镜像的可用性。

先决条件

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

- 已安装了 **skopeo**。

流程

1. 切换到项目：

```
$ oc project <project_name>
```

2. 列出在命名空间中运行的 pod，以及 pod 状态、错误状态、重启和年龄：

```
$ oc get pods
```

3. 确定命名空间是否由部署配置管理：

```
$ oc status
```

如果命名空间由部署配置管理，输出包括部署配置名称和基础镜像引用。

4. 检查以上命令输出中引用的基础镜像：

```
$ skopeo inspect docker://<image_reference>
```

5. 如果基础镜像引用不正确，请更新部署配置中的引用：

```
$ oc edit deployment/my-deployment
```

6. 当部署配置退出时，配置将自动重新部署。在部署过程中的 Watch pod 的状态，以确定这个问题是否已解决：

```
$ oc get pods -w
```

7. 检查命名空间中的事件，以了解与 pod 失败相关的诊断信息：

```
$ oc get events
```

7.6.3. 检查 pod 和容器日志

您可以检查 pod 和容器日志，以查看与显式 pod 失败相关的警告和错误消息。根据策略和退出代码，pod 和容器日志在 pod 终止后仍然可用。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 查询特定 pod 的日志：

```
$ oc logs <pod_name>
```

2. 查询 pod 中特定容器的日志：

```
$ oc logs <pod_name> -c <container_name>
```

由前面的 **oc logs** 命令所获得的日志由发送到 pod 或容器中的 stdout 的信息组成。

3. 检查 pod 中的 **/var/log/** 中包含的日志。

a. 列出 pod 中 **/var/log** 中所含的日志文件和子目录：

```
$ oc exec <pod_name> ls -alh /var/log
```

b. 查询 pod 中 **/var/log** 中所含的特定日志文件：

```
$ oc exec <pod_name> cat /var/log/<path_to_log>
```

c. 列出特定容器内 **/var/log** 中含有的日志文件和子目录：

```
$ oc exec <pod_name> -c <container_name> ls /var/log
```

d. 查询特定容器中的 **/var/log** 中所含的特定日志文件：

```
$ oc exec <pod_name> -c <container_name> cat /var/log/<path_to_log>
```

7.6.4. 访问运行的 pod

您可以通过在 pod 中打开 shell，或通过端口转发获取网络访问，来动态查看正在运行的 pod。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 切换到包含您要访问的 pod 的项目。这是必要的，因为 **oc rsh** 命令不支持使用 **-n** 选项指定命名空间：

```
$ oc project <namespace>
```

2. 启动到 pod 的远程 shell:

```
$ oc rsh <pod_name> 1
```

1 如果 pod 有多个容器，除非使用 **-c <container_name>** 指定了一个容器，否则 **oc rsh** 会默认使用第一个容器。

3. 启动至 pod 中的特定容器中的一个远程 shell ：

```
$ oc rsh -c <container_name> pod/<pod_name>
```

4. 创建一个端口转发会话到 pod 上的端口：

```
$ oc port-forward <pod_name> <host_port>:<pod_port>
```
(1)

(1) 输入 **Ctrl+C** 来取消端口转发会话。

7.6.5. 启动具有 root 访问权限的 debug pod

您可以基于一个有问题的 pod 部署或部署配置，启动具有根访问权限的 debug pod。pod 用户通常使用非 root 权限运行，但运行具有临时 root 特权的 pod 进行故障排除时在调查问题时很有用：

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 根据一个部署启动具有 root 访问权限的 debug pod。
   a. 获取项目部署名称：

```
$ oc get deployment -n <project_name>
```

   b. 根据部署启动带有 root 权限的 debug pod:

```
$ oc debug deployment/my-deployment --as-root -n <project_name>
```

2. 根据部署配置启动具有 root 访问权限的 debug pod。
   a. 获取项目的部署配置名称：

```
$ oc get deploymentconfigs -n <project_name>
```

   b. 根据部署配置，使用 root 权限启动 debug pod:

```
$ oc debug deploymentconfig/my-deployment-configuration --as-root -n <project_name>
```

注意

您可以将 **-- <command>** 附加到前面的 **oc debug** 命令中，以便在 debug pod 中运行单个命令，而不是运行交互式 shell。

7.6.6. 将文件复制到 pod 和容器，或从 pod 和容器中复制

您可以将文件复制到 pod 或从 pod 复制，以测试配置更改或收集诊断信息。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 将文件复制到 pod:

```
$ oc cp <local_path> <pod_name>:/<path> -c <container_name>
```
(1)

(1) 如果没有指定 **-c** 选项，则会选择 pod 中的第一个容器。

2. 从 pod 复制文件：

```
$ oc cp <pod_name>:/<path>  -c <container_name><local_path>
```
(1)

(1) 如果没有指定 **-c** 选项，则会选择 pod 中的第一个容器。

注意

要使 **oc cp** 正常工作，容器内必须有 **tar**。

7.7. 对 SOURCE-TO-IMAGE 进行故障排除

7.7.1. Source-to-Image 故障排除策略

Source-to-Image (S2I) 是一种用于构建可重复生成的 Docker 格式容器镜像的工具。它通过将应用程序源代码注入容器镜像，并汇编新镜像来生成可随时运行的镜像。新镜像融合了基础镜像（构建器）和构建的源。

要确定 S2I 进程中的故障发生位置，您可以观察与以下 S2I 阶段相关的 pod 状态：

1. **在构建配置阶段**，构建 pod 用于从基础镜像和应用程序源代码创建应用程序容器镜像。
2. **在部署配置阶段**，部署 pod 用于从构建配置阶段构建的应用程序容器镜像中部署应用程序 pod。部署 pod 还会部署其他资源，如服务和路由。部署配置在构建配置成功后开始。
3. **在部署 pod 启动应用程序 pod 后**，应用程序故障可能会在运行的应用程序 pod 中发生。例如，即使应用程序 pod 处于 **Running** 状态，应用程序的行为也可能不会如预期。在这种情况下，您可以访问正在运行的应用程序 pod，以调查 pod 中的应用程序故障。

当对 S2I 问题进行故障排除时，请按照这个策略操作：

1. 监控构建、部署和应用程序 pod 状态
2. 确定发生问题 S2I 进程阶段

3. 查看与失败阶段对应的日志

7.7.2. 收集 Source-to-Image 诊断数据

S2I 工具按顺序运行构建 pod 和部署 pod。部署 pod 负责根据构建阶段创建的应用程序容器镜像部署应用程序 pod。观察构建、部署和应用程序 pod 状态，以确定 S2I 进程中的故障发生位置。然后，重点收集诊断数据。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- API 服务仍然可以正常工作。
- 已安装 OpenShift CLI（**oc**）。

流程

1. 在整个 S2I 过程中监控 pod 状态，以确定在哪个阶段发生故障：

```
$ oc get pods -w
```
(1)

(1) 使用 **-w** 来监控 pod 是否有变化，直到您使用 **Ctrl+C** 退出命令。

2. 检查 pod 失败的日志以找出错误。

- **如果构建 pod 失败**，请检查构建 pod 的日志：

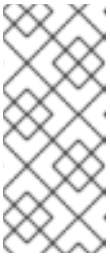
```
$ oc logs -f pod/<application_name>-<build_number>-build
```

注意

另外，您可以使用 **oc logs -f bc/<application_name>** 来查看构建配置的日志。构建配置的日志包括来自构建 pod 的日志。

- **如果部署 pod 失败**，请查看部署 pod 的日志：

```
$ oc logs -f pod/<application_name>-<build_number>-deploy
```



注意

另外，您可以使用 **oc logs -f bc/<application_name>** 来查看部署配置的日志。此操作会从部署 pod 输出日志，直到部署 pod 成功完成为止。如果在部署 pod 完成后运行，命令会输出来自应用程序 pod 的日志。部署 pod 完成后，仍可通过运行 **oc logs -f pod/<application_name>-<build_number>-deploy** 来访问其日志。

- **如果应用程序 pod 失败，或者应用程序没有如预期在正在运行的应用程序 pod 中发生**，请查看应用程序 pod 的日志：

```
$ oc logs -f pod/<application_name>-<build_number>-<random_string>
```

7.7.3. 收集应用程序诊断数据以调查应用程序失败

应用程序故障可在运行的应用程序 pod 中发生。在这些情况下，您可以使用以下策略检索诊断信息：

- 检查与应用程序 pod 相关的事件。
- 查看应用程序 pod 的日志，包括不是由 OpenShift Container Platform 日志框架收集的特定应用程序日志文件。
- 以互动方式测试应用程序功能，并在应用程序容器中运行诊断工具。

先决条件

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

流程

1. 列出与特定应用程序 pod 相关的事件。以下示例检索名为 **my-app-1-akdlg** 的应用程序 pod 的事件:

```
$ oc describe pod/my-app-1-akdlg
```

2. 检查应用程序 pod 的日志：

```
$ oc logs -f pod/my-app-1-akdlg
```

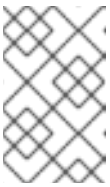
3. 在正在运行的应用程序 pod 中查询特定日志。发送到 stdout 的日志由 OpenShift Container Platform 日志记录框架收集，并包含在上一命令的输出中。以下查询只适用于没有发送到 stdout 的日志。

   a. 如果应用程序日志可以在 pod 内不需要 root 权限的情况下就可以进行访问，则按如下方式处理日志文件：

```
$ oc exec my-app-1-akdlg -- cat /var/log/my-application.log
```

   b. 如果需要 root 访问权限才能查看应用程序日志，您可以启动具有 root 权限的 debug 容器，然后从容器内查看日志文件。从项目的 **DeploymentConfig** 对象启动 debug 容器。pod 用户通常使用非 root 权限运行，但运行具有临时 root 特权的 pod 进行故障排除时在调查问题时很有用：

```
$ oc debug dc/my-deployment-configuration --as-root -- cat /var/log/my-application.log
```



注意

如果您运行不使用 **-- <command>** 的**oc debug dc/<deployment_configuration> --as-root**，则可以获得 debug pod 内带有 root 权限的一个互动式 shell 。

4. 以互动方式测试应用程序功能,并在带有互动 shell 的应用程序容器中运行诊断工具。

   a. 在应用程序容器上启动一个交互式 shell:

```
$ oc exec -it my-app-1-akdlg /bin/bash
```

b. 在 shell 中以互动方式测试应用程序功能。例如，您可以运行容器的入口点命令并观察结果。然后，在更新源代码并通过 S2I 进程重建应用程序容器前，直接从命令行测试更改。

c. 运行容器中的诊断二进制文件。

注意

运行一些诊断二进制文件需要 root 权限。在这些情况下，您可以通过运行 **oc debug dc/<deployment_configuration> --as-root**，根据有问题的 pod 的 **DeploymentConfig** 对象启动一个带有 root 访问权限的 debug pod。然后，您可以从 debug pod 中以 root 用户身份运行诊断二进制文件。

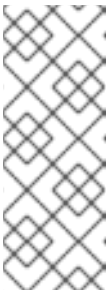
5. 如果容器中没有诊断二进制文件，您可以使用 **nsenter** 在容器的命名空间中运行主机的诊断二进制文件。以下实例在一个容器的命名空间中运行 **ip ad**，使用主机的 **ip** 二进制代码。

a. 在目标节点上进入一个 debug 会话。此步骤被实例化为一个名为 **<node_name>-debug** 的 debug pod:

```
$ oc debug node/my-cluster-node
```

b. 将 **/host** 设为 debug shell 中的根目录。debug pod 在 pod 中的 **/host** 中挂载主机的 root 文件系统。将根目录改为 **/host**，您可以运行主机可执行路径中包含的二进制文件：

```
# chroot /host
```



注意

运行 Red Hat Enterprise Linux CoreOS（RHCOS）的 OpenShift Container Platform 4.6 集群节点不可变，它依赖于 Operator 来应用集群更改。不建议使用 SSH 访问集群节点，节点将会标记为 *accessed* 污点。但是，如果 OpenShift Container Platform API 不可用，或 kubelet 在目标节点上无法正常工作, **oc** 操作将会受到影响。在这种情况下，可以使用 **ssh core@<node>.<cluster_name>.<base_domain>** 来访问节点。

c. 确定目标容器 ID:

```
# crictl ps
```

d. 确定容器的进程 ID。在本例中，目标容器 ID 是**7fe32346b120**:

```
# crictl inspect a7fe32346b120 --output yaml | grep 'pid:' | awk '{print $2}'
```

e. 在容器命名空间中运行 **ip ad**，使用主机的 **ip** 二进制代码。本例使用 **31150** 作为容器的进程 ID。**nsenter** 命令输入目标进程的命名空间并在命名空间中运行命令。因为本例中的目标进程是一个容器的进程 ID，所以 **ip ad** 命令在主机的容器命名空间中运行：

```
# nsenter -n -t 31150 -- ip ad
```

注意

只有在使用特权容器（如 debug 节点）时，才能在容器的命名空间中运行主机的诊断二进制代码。

7.7.4. 其他资源

- 如需有关S2I 构建策略的更多详情，请参阅 Source-to-Image（S2I）构建。

7.8. 存储问题故障排除

7.8.1. 解决多附件错误

当节点崩溃或立即关闭时,预期会从节点卸载附加的 ReadWriteOnce(RWO)卷,以便被调度到另一节点上的 pod 使用。

但是,不可能在新节点中挂载,因为失败的节点无法卸载附加的卷。

报告了一个 multi-attach 错误：

输出示例

```
Unable to attach or mount volumes: unmounted volumes=[sso-mysql-pvol], unattached volumes=
[sso-mysql-pvol default-token-x4rzc]: timed out waiting for the condition
Multi-Attach error for volume "pvc-8837384d-69d7-40b2-b2e6-5df86943eef9" Volume is already used
by pod(s) sso-mysql-1-ns6b4
```

流程

要解决 multi-attach 问题,请使用以下解决方案之一：

- 使用 RWX 卷启用多个附件。
  对于大多数存储解决方案,您可以使用 ReadWriteMany(RWX)卷以防止多附加错误。

- 使用 RWO 卷时,恢复或删除故障节点。
  对于不支持 RWX 的存储,如 VMware vSphere,必须改为使用 RWO 卷。但是, RWO 卷无法挂载到多个节点上。

  如果您遇到带有 RWO 卷的多附件错误消息，请强制在关闭或崩溃的节点上删除 pod，以避免关键工作负载中的数据丢失，例如在附加动态持久性卷时。

  ```
  $ oc delete pod <old_pod> --force=true --grace-period=0s
  ```

  该命令会在 6 分钟后删除处于关闭或崩溃的节点上的卷。

7.9. WINDOWS 容器工作负载问题故障排除

7.9.1. 没有安装 Windows Machine Config Operator

如果已完成了安装 Windows Machine Config Operator（WMCO）的过程，但 Operator 一直处于 **InstallWaiting** 阶段，问题可能是由网络问题造成的。

WMCO 要求使用 OVN-Kubernetes 配置 OpenShift Container Platform 集群和混合网络。WMCO 在没有混合网络的情况下无法完成安装过程。这是管理多个操作系统（OS）和 OS 变体的节点所必需的。这必须在集群安装过程中完成。

如需更多信息，请参阅配置混合网络。

7.9.2. 检查为什么 Windows 机器没有成为计算节点

Windows 机器没有成为计算节点的原因有很多。调查此问题的最佳方法是收集 Windows Machine Config Operator（WMCO）日志。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。

流程

- 运行以下命令来收集 WMCO 日志：

```
$ oc logs -f $(oc get pods -o jsonpath={.items[0].metadata.name} -n openshift-windows-machine-config-operator) -n openshift-windows-machine-config-operator
```

7.9.3. 访问 Windows 节点

Windows 节点无法使用 **oc debug node** 命令访问；命令需要在该节点上运行特权 pod，但 Windows 还不支持这个 pod。可以使用 SSH 或 Remote Desktop Protocol（RDP）访问 Windows 节点。两种方法都需要一个 SSH 堡垒。

7.9.3.1. 使用 SSH 访问 Windows 节点

您可以使用 SSH 访问 Windows 节点。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装了 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。
- 您已将 **cloud-private-key** secret 中使用的密钥以及创建集群时使用的密钥添加到 ssh-agent。为安全起见，请记住在使用后从 ssh-agent 中删除密钥。
- 已使用 **ssh-bastion** pod 连接到 Windows 节点。

流程

- 运行以下命令来访问 Windows 节点：

```
$ ssh -t -o StrictHostKeyChecking=no -o ProxyCommand='ssh -A -o StrictHostKeyChecking=no \
    -o ServerAliveInterval=30 -W %h:%p core@$(oc get service --all-namespaces -l run=ssh-
```

```
bastion \
    -o go-template="{{ with (index (index .items 0).status.loadBalancer.ingress 0) }}{{ or
.hostname .ip }}{{end}}")' <username>@<windows_node_internal_ip> 1 2
```

1 指定云供应商用户名，如 Amazon Web Services(AWS)的 **Administrator** 或 Microsoft Azure 的 **capi**。

2 指定节点的内部 IP 地址，可通过运行以下命令来发现该地址：

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\
(@.type==\"InternalIP\")].address}
```

7.9.3.2. 使用 RDP 访问 Windows 节点

您可以使用 Remote Desktop Protocol（RDP）访问 Windows 节点。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。
- 您已将 **cloud-private-key** secret 中使用的密钥以及创建集群时使用的密钥添加到 ssh-agent。为安全起见，请记住在使用后从 ssh-agent 中删除密钥。
- 已使用 **ssh-bastion** pod 连接到 Windows 节点。

流程

1. 运行以下命令设定 SSH tunnel：

```
$ ssh -L 2020:<windows_node_internal_ip>:3389 \ 1
    core@$(oc get service --all-namespaces -l run=ssh-bastion -o go-template="{{ with (index
(index .items 0).status.loadBalancer.ingress 0) }}{{ or .hostname .ip }}{{end}}")
```

1 指定节点的内部 IP 地址，可通过运行以下命令来发现该地址：

```
$ oc get nodes <node_name> -o jsonpath={.status.addresses[?\
(@.type==\"InternalIP\")].address}
```

2. 在生成的 shell 中 SSH 到 Windows 节点，并运行以下命令为用户创建密码：

```
C:\> net user <username> * 1
```

1 指定云供应商用户名，如 AWS 的 **Administrator** 或 Azure 的 **capi**。

现在您可以使用 RDP 客户端在 **localhost:2020** 远程访问 Windows 节点。

7.9.4. 为 Windows 容器收集 Kubernetes 节点日志

Windows 容器日志记录与 Linux 容器日志记录不同，Windows 工作负载的 Kubernetes 节点日志默认流传输至 **C:\var\logs** 目录。因此，您必须从该目录中收集 Windows 节点日志。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。

流程

1. 要在 **C:\var\logs** 中的所有目录中查看日志，请运行以下命令：

```
$ oc adm node-logs -l kubernetes.io/os=windows --path= \
  /ip-10-0-138-252.us-east-2.compute.internal containers \
  /ip-10-0-138-252.us-east-2.compute.internal hybrid-overlay \
  /ip-10-0-138-252.us-east-2.compute.internal kube-proxy \
  /ip-10-0-138-252.us-east-2.compute.internal kubelet \
  /ip-10-0-138-252.us-east-2.compute.internal pods
```

2. 您现在可以使用同样的命令列出目录中的文件并查看单个日志文件。例如，要查看 kubelet 日志，请运行以下命令：

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=/kubelet/kubelet.log
```

7.9.5. 收集 Windows 应用程序事件日志

kubelet **logs** 端点上的 **Get-WinEvent shim** 可用于从 Windows 机器收集应用程序事件日志。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。

流程

- 要查看所有应用程序日志记录到 Windows 机器上的事件日志的日志，请运行：

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal
```

使用 **oc adm must-gather** 收集日志时执行同样的命令。

还可以通过使用 **-u** 标志指定相应服务来收集来自事件日志的其它 Windows 应用程序日志。例如，您可以运行以下命令来收集 docker runtime 服务的日志：

```
$ oc adm node-logs -l kubernetes.io/os=windows --path=journal -u docker
```

7.9.6. 为 Windows 容器收集 Docker 日志

Windows Docker 服务不会将日志流传输到 stdout，而是记录到 Windows 的事件日志中。您可以查看 Docker 事件日志，以调查您认为由 Windows Docker 服务造成的问题。

先决条件

- 已使用 Operator Lifecycle Manager（OLM）安装 Windows Machine Config Operator（WMCO）。
- 您已创建了 Windows 机器集。

流程

1. SSH 到 Windows 节点并输入 PowerShell:

```
C:\> powershell
```

2. 运行以下命令来查看 Docker 日志：

```
C:\> Get-EventLog -LogName Application -Source Docker
```

7.9.7. 其他资源

- Windows 容器的故障排除
- 排除主机和容器镜像不匹配的问题
- 用于 Windows 故障排除的 Docker
- Windows 中的常见 Kubernetes 问题

7.10. 调查监控问题

OpenShift Container Platform 包括一个预配置、预安装和自我更新的监控堆栈，用于监控核心平台组件。在 OpenShift Container Platform 4.6 中，集群管理员可以选择性地为用户定义的项目启用监控。

如果您自己的指标不可用，或者 Prometheus 消耗了大量磁盘空间，则可按照以下步骤操作。

7.10.1. 检查为什么用户定义的指标不可用

通过 **ServiceMonitor** 资源，您可以确定如何使用用户定义的项目中的服务公开的指标。如果您创建了 **ServiceMonitor** 资源，但无法在 Metrics UI 中看到任何对应的指标，请按该流程中所述的步骤操作。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您已为用户定义的工作负载启用并配置了监控。
- 您已创建了 **user-workload-monitoring-config ConfigMap** 对象。
- 您已创建了 **ServiceMonitor** 资源。

流程

1. 在服务和 **ServiceMonitor** 资源配置中检查对应的标签是否匹配。

   a. 获取服务中定义的标签。以下示例在 **ns1** 项目中查询 **prometheus-example-app** 服务：

   ```
   $ oc -n ns1 get service prometheus-example-app -o yaml
   ```

   输出示例

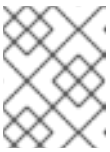
   ```
     labels:
       app: prometheus-example-app
   ```

   b. 检查 **ServiceMonitor** 资源配置中的 **matchLabels app** 标签是否与上一步中的标签输出匹配：

   ```
   $ oc -n ns1 get servicemonitor prometheus-example-monitor -o yaml
   ```

   输出示例

   ```
   spec:
     endpoints:
     - interval: 30s
       port: web
       scheme: http
     selector:
       matchLabels:
         app: prometheus-example-app
   ```

   注意

   您可以作为具有项目查看权限的开发者检查服务和 **ServiceMonitor** 资源标签。

2. 在 **openshift-user-workload-monitoring** 项目中检查 Prometheus Operator 的日志。

   a. 列出 **openshift-user-workload-monitoring** 项目中的 Pod：

   ```
   $ oc -n openshift-user-workload-monitoring get pods
   ```

   输出示例

   ```
   NAME                                   READY   STATUS    RESTARTS   AGE
   prometheus-operator-776fcbbd56-2nbfm   2/2     Running   0          132m
   prometheus-user-workload-0             5/5     Running   1          132m
   prometheus-user-workload-1             5/5     Running   1          132m
   thanos-ruler-user-workload-0           3/3     Running   0          132m
   thanos-ruler-user-workload-1           3/3     Running   0          132m
   ```

   b. 从 **prometheus-operator** Pod 中的 **prometheus-operator** 容器获取日志。在以下示例中，Pod 名为 **prometheus-operator-776fcbbd56-2nbfm**：

```
$ oc -n openshift-user-workload-monitoring logs prometheus-operator-776fcbbd56-2nbfm -c prometheus-operator
```

如果服务监控器出现问题，日志可能包含类似本例的错误：

```
level=warn ts=2020-08-10T11:48:20.906739623Z caller=operator.go:1829 component=prometheusoperator msg="skipping servicemonitor" error="it accesses file system via bearer token file which Prometheus specification prohibits" servicemonitor=eagle/eagle namespace=openshift-user-workload-monitoring prometheus=user-workload
```

3. 直接在 Prometheus UI 中查看项目的目标状态。

   a. 建立端口转发到 **openshift-user-workload-monitoring** 项目中的 Prometheus 实例：

   ```
   $ oc port-forward -n openshift-user-workload-monitoring pod/prometheus-user-workload-0 9090
   ```

   b. 在 Web 浏览器中打开 http://localhost:9090/targets，并在 Prometheus UI 中直接查看项目的目标状态。检查与目标相关的错误消息。

4. 在 **openshift-user-workload-monitoring** 项目中为 Prometheus Operator 配置 debug 级别的日志记录。

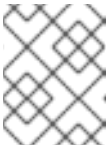
   a. 在 **openshift-user-workload-monitoring** 项目中编辑 **user-workload-monitoring-config ConfigMap** 对象：

   ```
   $ oc -n openshift-user-workload-monitoring edit configmap user-workload-monitoring-config
   ```

   b. 在 **data/config.yaml** 下为 **prometheusOperator** 添加 **logLevel: debug**，将日志级别设置为 **debug**：

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: user-workload-monitoring-config
     namespace: openshift-user-workload-monitoring
   data:
     config.yaml: |
       prometheusOperator:
         logLevel: debug
   ```

   c. 保存文件以使改变生效。

   注意

   在应用日志级别更改时，**openshift-user-workload-monitoring** 项目中的 **prometheus-operator** 会自动重启。

   d. 确认 **debug** 日志级别已应用到 **openshift-user-workload-monitoring** 项目中的 **prometheus-operator** 部署：

```
$ oc -n openshift-user-workload-monitoring get deploy prometheus-operator -o yaml |
grep "log-level"
```

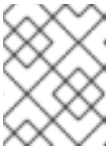
输出示例

```
        - --log-level=debug
```

Debug 级别日志记录将显示 Prometheus Operator 发出的所有调用。

e. 检查 **prometheus-operator** Pod 是否正在运行：

```
$ oc -n openshift-user-workload-monitoring get pods
```



注意

如果配置映射中包含了一个未识别的 Prometheus Operator **loglevel** 值，则 **prometheus-operator** Pod 可能无法成功重启。

f. 查看 debug 日志，以了解 Prometheus Operator 是否在使用 **ServiceMonitor** 资源。查看日志中的其他相关错误。

其他资源

- 创建用户定义的工作负载监控配置映射
- 如需有关如何创建 ServiceMonitor 或 PodMonitor 的详细信息，请参阅指定如何监控服务

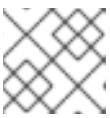
7.10.2. 确定为什么 Prometheus 消耗大量磁盘空间

开发人员可以使用键值对的形式为指标定义属性。潜在的键值对数量与属性的可能值数量对应。具有无限数量可能值的属性被称为未绑定属性。例如，**customer_id** 属性不绑定，因为它有无限多个可能的值。

每个分配的键值对都有唯一的时间序列。在标签中使用许多未绑定属性可导致所创建的时间序列数量出现指数增加。这可能会影响 Prometheus 性能，并消耗大量磁盘空间。

当 Prometheus 消耗大量磁盘时，您可以使用以下方法：

- **检查正在收集的提取示例数量**。
- **检查 Prometheus UI 中的时间序列数据库（TSDB）状态**以了解有关哪些标签创建最多时间序列的更多信息。这需要集群管理员特权。
- **要减少创建的唯一时间序列数量**，您可以减少分配给用户定义的指标的未绑定属性数量



注意

使用绑定到一组有限可能值的属性可减少潜在的键-值对组合数量。

- **对可在用户定义的项目中提取的示例数量实施限制**。这需要集群管理员特权。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。

- 已安装 OpenShift CLI(**oc**)。

流程

1. 在 **Administrator** 视角中，导航到 **Monitoring → Metrics**。
2. 在 **Expression** 字段中运行以下 Prometheus Query Language (PromQL) 查询。这会返回具有最高提取示例数的十个指标：

```
topk(10,count by (job)({__name__=~".+"}))
```

3. 如果指标的提取示例数大于预期，请检查分配给指标的未绑定标签值数量。
   - **如果指标与用户定义的项目相关**，请查看分配给您的工作负载的指标键-值对。它们通过应用程序级别的 Prometheus 客户端库实施。尝试限制标签中引用的未绑定属性数量。
   - **如果指标与 OpenShift Container Platform 核心项目相关**，请在红帽客户门户网站上创建一个红帽支持问题单。
4. 检查 Prometheus UI 中的 TSDB 状态。
   a. 在 **Administrator** 视角中，导航至 **Networking → Routes**。
   b. 选择 **Project** 列表中的 **openshift-monitoring** 项目。
   c. 选择 **prometheus-k8s** 行中的 URL 来打开 Prometheus UI 的登录页面。
   d. 选择 **Log in with OpenShift** 来使用您的 OpenShift Container Platform 凭证进行登录。
   e. 在 Prometheus UI 中，进入到 **Status → TSDB Status**。

其它资源

- 如需有关如何设置提取示例限制和创建相关警报规则的详细信息，请参阅为用户定义的项目设置提取示例限制

7.11. 诊断 OPENSHIFT CLI（OC）问题

7.11.1. 了解 OpenShift CLI（oc）日志级别

借助 OpenShift CLI（**oc**），您可以从终端创建应用程序并管理 OpenShift Container Platform 项目。

如果出现特定于 **oc** 命令的问题，将 **oc** 日志级别提高为输出 API 请求、API 响应以及命令生成的 **curl** 请求详情。这提供了特定的 **oc** 命令的底层操作信息，以帮助您了解故障的本质。

oc 日志级别范围从 1 到 10。下表提供了 **oc** 日志级别列表及其描述。

表 7.3. OpenShift CLI（oc）日志级别

日志级别	描述
1 到 5	没有额外的日志记录到 stderr。

日志级别	描述
6	为 stderr 记录 API 请求。
7	将 API 请求和标头记录到 stderr。
8	记录 API 请求、标头和正文，以及 API 响应标头和正文到 stderr。
9	记录日志 API 请求、标头和正文、API 响应标头和正文，以及 **curl** 请求到 stderr。
10	记录日志 API 请求、标头和正文、API 响应标头和正文，以及 **curl** 请求到 stderr。记录的信息会更详细。

7.11.2. 指定 OpenShift CLI（oc）日志级别

您可以通过提高命令的日志级别来调查 OpenShift CLI（**oc**）问题。

先决条件

- 已安装 OpenShift CLI（**oc**）。

流程

1. 在运行 **oc** 命令时指定 **oc** 日志级别：

```
$ oc <options> --loglevel <log_level>
```

2. OpenShift Container Platform 用户的当前会话令牌通常包含在记录的 **curl** 请求中。您还可以手动获取当前用户的会话令牌，以便在测试 **oc** 命令的底层进程的各个方面时使用：

```
$ oc whoami -t
```