Workload Availability for Red Hat OpenShift 24.2

补救、隔离和维护

工作负载可用性补救、隔离和维护

Last Updated: 2024-05-16

Workload Availability for Red Hat OpenShift 24.2 补救、隔离和维护

工作负载可用性补救、隔离和维护

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

有关工作负载可用性 Operator 及其用法的信息

目录

前言

为 RED HAT OPENSHIFT 文档提供有关工作负载可用性的反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。要做到这一点：

1. 进入 JIRA 网站。
2. 在 **Summary** 字段中输入描述性标题。
3. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
4. 在 **Reporter** 字段中输入您的用户名。
5. 在 **Affects Version/s** 字段中输入受影响的版本。
6. 点对话框底部的 **Create**。

第 1 章 关于节点补救、隔离和维护

硬件是 imperfect，软件包含 bug。当节点级别的故障（如内核挂起或网络接口控制器(NIC)）失败时，集群所需的工作不会减少，并且受影响节点的工作负载需要在哪里重启。但是，一些工作负载（如 ReadWriteOnce (RWO) 卷和 StatefulSets）可能需要最少的语义。

影响这些工作负载的风险、损坏或两者的故障。在启动恢复工作负载（称为 **补救** 和理想情况）之前，确保节点达到安全状态（称为 **隔离** ）。

并不总是依赖于管理员干预来确认节点和工作负载的真正状态。为了便于实现此类干预，Red Hat OpenShift 为自动化故障检测、隔离和修复提供了多个组件。

1.1. 自助服务修复

Self Node Remediation Operator 是一个 Red Hat OpenShift 附加组件 Operator，它实现了隔离的外部系统，补救重启不健康的节点并删除资源，如 Pod 和 VolumeAttachments。重启可确保工作负载被隔离，资源删除会加快重新调度受影响工作负载。与其他外部系统不同，自助节点修复不需要任何管理界面，如智能平台管理接口 (IPMI) 或用于节点置备的 API。

失败的检测系统可以使用自助服务修复，如 Machine Health Check 或 Node Health Check。

1.2. 隔离代理修复

Fence Agents Remediation (FAR) Operator 是一个 Red Hat OpenShift 附加组件 Operator，它会自动修复不健康的节点，类似于 Self Node Remediation Operator。通过使用管理界面或传统的 API，FAR 运行 fence-agent 来通过电源节点从不健康状态修复节点。

FAR 旨在为具有传统 API 端点的环境运行一组现有的上游隔离代理，例如 IPMI，用于电源循环集群节点。

1.3. 机器删除修复

Machine Deletion Remediation (MDR) Operator 是一个 Red Hat OpenShift 附加组件 Operator，它使用 Machine API 重新置备不健康的节点。MDR 可以与 NodeHealthCheck (NHC)一起工作，为 MDR 创建自定义资源(CR)，其中包含有关不健康节点的信息。

MDR 遵循节点上的注解到关联的机器对象，并确认它拥有自己的控制器。MDR 继续删除机器，然后拥有控制器重新创建替换机器。

1.4. 机器健康检查

Machine Health Check 使用 Red Hat OpenShift 内置的故障检测、隔离和修复系统，用于监控机器状态和节点状况。机器健康检查可以被配置为触发外部隔离和修复系统，如自助节点修复。

1.5. 节点健康检查

Node Health Check Operator 是一个 Red Hat OpenShift 附加组件 Operator，它实现了一个监控节点状况的故障检测系统。它没有内置的隔离或补救系统，因此必须使用提供这些功能的外部系统进行配置。默认情况下，它被配置为使用 Self Node Remediation 系统。

1.6. 节点维护

管理员面临需要中断集群的情况，例如替换驱动器、RAM 或 NIC。

在此维护之前，应该对受影响的节点进行封锁并排空。当节点被封锁时，无法将新的工作负载调度到该节点上。当节点排空时，为了避免或最小化停机时间，受影响节点上的工作负载将传送到其他节点。

虽然此维护可以使用命令行工具实现，但 Node Maintenance Operator 提供了使用自定义资源来实现此目的的声明方法。当节点存在此类资源时，Operator 会封锁并排空节点，直到资源被删除为止。

1.7. 关于工作负载可用性 OPERATOR 的指标

添加数据分析增强了对工作负载可用性 Operator 的可观察性。数据提供有关操作器的活动的指标，以及对集群的影响。这些指标提高了决策功能，启用数据驱动的优化，并增强整体系统性能。

您可以使用指标来执行这些任务：

- 访问操作器的全面跟踪数据，以监控整体系统效率。
- 访问来自跟踪数据的可操作见解，如识别故障节点，或因为 Operator 修复而停机。
- 视觉化 Operator 的补救如何真正提高系统效率。

1.7.1. 为工作负载可用性 Operator 配置指标

您可以使用 Red Hat OpenShift Web 控制台安装 Node Health Check Operator。

先决条件

- 您必须首先配置监控堆栈。如需更多信息，请参阅 配置监控堆栈。
- 您必须启用对已用项目的监控。如需更多信息，请参阅为已定义项目启用监控。

流程

1. 从现有的 **prometheus-user-workload-token** secret 创建 **prometheus-user-token** secret，如下所示：

```
existingPrometheusTokenSecret=$(kubectl get secret --namespace openshift-user-workload-monitoring | grep prometheus-user-workload-token | awk '{print $1}') (1)

kubectl get secret ${existingPrometheusTokenSecret} --namespace=openshift-user-workload-monitoring -o yaml | \
   sed '/namespace:
.*==/d;/ca.crt:/d;/serviceCa.crt/d;/creationTimestamp:/d;/resourceVersion:/d;/uid:/d;/annotations/d;/kubernetes.io/d;' | \
   sed 's/namespace: .*/namespace: openshift-workload-availability/' | \ (2)
   sed 's/name: .*/name: prometheus-user-workload-token/' | \ (3)
   sed 's/type: .*/type: Opaque/' | \
   > prom-token.yaml

kubectl apply -f prom-token.yaml
```

(1) 在下一步中创建的 Metric ServiceMonitor 需要 prometheus-user-token。

(2) 确保新的 Secret 的命名空间是安装 NHC Operator 的位置，如 openshift-workload-availability。

3 只有在启用了 User Worload Prometheus 提取时，prometheus-user-workload-token 才存在。

2. 按如下方式创建 ServiceMonitor：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: node-healthcheck-metrics-monitor
  namespace: openshift-workload-availability 1
  labels:
    app.kubernetes.io/component: controller-manager
spec:
  endpoints:
  - interval: 30s
    port: https
    scheme: https
    authorization:
      type: Bearer
      credentials:
        name: prometheus-user-workload-token
        key: token
    tlsConfig:
      ca:
        configMap:
          name: nhc-serving-certs-ca-bundle
          key: service-ca.crt
      serverName: node-healthcheck-controller-manager-metrics-service.openshift-workload-
availability.svc 2
  selector:
    matchLabels:
      app.kubernetes.io/component: controller-manager
      app.kubernetes.io/name: node-healthcheck-operator
      app.kubernetes.io/instance: metrics
```

1 指定要配置指标的命名空间，如 **openshift-workload-availability**。

2 serverName 必须包含安装 Operator 的同一命名空间。在示例中，**openshift-workload-availability** 放置在 metrics 服务名称后，并在 filetype 扩展之前放置。

验证

要确认配置是否成功，OCP Web UI 中的 **Observe** > **Targets** 选项卡显示 **Endpoint Up**。

1.7.2. 工作负载可用性 Operator 的指标示例

以下是来自不同工作负载可用性 Operator 的指标示例。

指标包括以下指示器的信息：

- Operator 可用性：显示每个 Operator 是否正在运行的时间。
- 节点补救计数 ：显示同一节点的补救数量，以及所有节点。

- 节点补救持续时间：显示补救停机时间或恢复时间。
- 节点补救量 ：显示持续补救的数量。

第 2 章 使用自节点修复

您可以使用 Self Node Remediation Operator 自动重新引导不健康的节点。此补救策略可最小化有状态应用程序和 ReadWriteOnce (RWO) 卷的停机时间，并在发生临时故障时恢复计算容量。

2.1. 关于自助服务修复 OPERATOR

Self Node Remediation Operator 在集群节点上运行，并重启被识别为不健康的节点。Operator 使用 **MachineHealthCheck** 或 **NodeHealthCheck** 控制器来检测集群中节点的健康状态。当节点识别为不健康时，**MachineHealthCheck** 或 **NodeHealthCheck** 资源会创建 **SelfNodeRemediation** 自定义资源 (CR)，这会触发 Self Node Remediation Operator。

SelfNodeRemediation CR 类似于以下 YAML 文件：

```
apiVersion: self-node-remediation.medik8s.io/v1alpha1
kind: SelfNodeRemediation
metadata:
  name: selfnoderemediation-sample
  namespace: openshift-workload-availability
spec:
  remediationStrategy: <remediation_strategy> 1
status:
  lastError: <last_error_message> 2
```

1 指定节点的补救策略。

2 显示补救过程中发生的最后错误。当补救成功或没有发生错误时，字段会留空。

Self Node Remediation Operator 最小化有状态应用程序的停机时间，并在出现临时故障时恢复计算容量。无论 IPMI 或 API 等管理界面如何置备节点，都可使用此 Operator 来置备节点，无论集群安装类型是什么，如安装程序置备的基础架构或用户置备的基础架构。

2.1.1. 关于 watchdog 设备

watchdog 设备可以是以下任意一种：

- 独立电源的硬件设备
- 与它们控制的主机共享电源的硬件设备
- 软件或 **softdog**中实施的虚拟设备

硬件 watchdog 和 **softdog** 设备分别具有电子计时器和软件计时器。这些 watchdog 设备用于确保在检测到错误条件时机器进入安全状态。集群需要重复重置 watchdog 定时器以证明它处于健康状态。此计时器可能会因为出现错误条件而造成问题，如死锁、CPU 不足以及网络或磁盘访问的丢失。如果计时器过期，watchdog 设备会假设发生了错误，设备会触发强制重置节点。

硬件 watchdog 设备比 **softdog** 设备更可靠。

2.1.1.1. 了解 watchdog 设备的自助服务修复 Operator 行为

Self Node Remediation Operator 根据存在的 watchdog 设备决定补救策略。

如果配置了硬件 watchdog 设备并可用，Operator 会使用它进行补救。如果没有配置硬件 watchdog 设备，Operator 会启用并使用 **softdog** 设备进行补救。

如果既不支持 watchdog 设备，无论是系统或配置，Operator 都会使用软件重启来修复节点。

其他资源

为虚拟机配置 watchdog 设备

2.2. CONTROL PLANE 隔离

在早期版本中，您可以在 worker 节点上启用自节点修复和 Node Health Check。如果节点失败，您现在可以在 control plane 节点上遵循补救策略。

在两种主要场景中进行自助服务修复。

- API 服务器连接
  - 在这种情况下，要修复的 control plane 节点不会被隔离。它可以直接连接到 API 服务器，或者可以通过 worker 节点或 control-plane 节点间接连接到 API 服务器，这些节点直接连接到 API 服务器。
  - 当有 API 服务器连接时，只有在 Node Health Check Operator 为节点创建了 **SelfNodeRemediation** 自定义资源(CR)时，才会修复 control plane 节点。
- 没有 API 服务器连接
  - 在这种情况下，要修复的 control plane 节点与 API 服务器隔离。节点无法直接连接或间接连接到 API 服务器。
  - 如果没有 API 服务器连接，则 control plane 节点将按照以下步骤进行修复：
    - 使用大多数对等 worker 节点检查 control plane 节点的状态。如果无法访问大多数对等 worker 节点，则会进一步分析该节点。
      - 自我诊断 control plane 节点的状态
        - 如果通过自我诊断，则不会执行任何操作。
        - 如果自我诊断失败，则该节点将被隔离并修复。
        - 目前支持的自诊断是使用 **opt in** 配置检查 **kubelet** 服务状态，以及检查端点的可用性。
    - 如果节点没有管理与大多数 worker 对等点通信，请检查 control plane 节点与其他 control plane 节点的连接。如果节点可以与任何其他 control plane peer 通信，则不会执行任何操作。否则，节点将被隔离并修复。

2.3. 使用 WEB 控制台安装 SELF NODE REMEDIATION OPERATOR

您可以使用 Red Hat OpenShift Web 控制台安装 Self Node Remediation Operator。

注意

Node Health Check Operator 还将 Self Node Remediation Operator 安装为默认的补救提供程序。

先决条件

- 以具有 **cluster-admin** 特权的用户身份登录。

流程

1. 在 Red Hat OpenShift Web 控制台中，导航到 **Operators → OperatorHub**。
2. 从可用的 Operator 列表中选择 Self Node Remediation Operator，然后点 **Install**。
3. 保留安装模式和命名空间的默认选择，以确保将 Operator 安装到 **openshift-workload-availability** 命名空间中。
4. **点 Install。**

验证

确认安装成功：

1. **导航到 Operators → Installed Operators 页面。**
2. 检查 Operator **是否安装在 openshift-workload-availability 命名空间中，其状态是否为 Succeeded。**

如果 Operator 没有成功安装：

1. **导航到 Operators → Installed Operators 页面，并检查 Status 列中是否有任何错误或故障。**
2. **导航到 Workloads → Pods 页面，检查 openshift-workload-availability 项目中的 self-node-remediation-controller-manager pod 和 self-node-remediation-ds pod 的日志。**

2.4. 使用 CLI 安装自助服务 OPERATOR

您可以使用 OpenShift CLI(oc)安装 Self Node Remediation Operator。

您可以在自己的命名空间中或 openshift-workload-availability 命名空间中安装 Self Node Remediation Operator。

先决条件

- **安装 OpenShift CLI（oc）。**
- **以具有 cluster-admin 特权的用户身份登录。**

流程

1. 为 Self Node Remediation Operator 创建 **Namespace 自定义资源(CR)：**
   a. **定义 Namespace CR 并保存 YAML 文件，如workload-availability-namespace.yaml ：**

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-workload-availability
```

   b. **要创建 Namespace CR，请运行以下命令：**

```
$ oc create -f workload-availability-namespace.yaml
```

2. 创建 OperatorGroup CR：

   a. 定义 OperatorGroup CR 并保存 YAML 文件，如workload-availability-operator-group.yaml ：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: workload-availability-operator-group
  namespace: openshift-workload-availability
```

   b. 要创建 OperatorGroup CR，请运行以下命令：

```
$ oc create -f workload-availability-operator-group.yaml
```

3. 创建一个 Subscription CR：

   a. 定义 Subscription CR 并保存 YAML 文件，如self-node-remediation-subscription.yaml ：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
   name: self-node-remediation-operator
   namespace: openshift-workload-availability (1)
spec:
   channel: stable
   installPlanApproval: Manual (2)
   name: self-node-remediation-operator
   source: redhat-operators
   sourceNamespace: openshift-marketplace
   package: self-node-remediation
```

   (1) 指定要安装 Self Node Remediation Operator 的命名空间。要在 openshift-workload-availability 命名空间中安装 Self Node Remediation Operator，请在Subscription CR 中指定 openshift-workload-availability。

   (2) 如果您的指定版本被目录中的后续版本取代，则将批准策略设置为 Manual。此计划阻止自动升级到更新的版本，且需要在启动 CSV 可以完成安装前手动批准。

   b. 要创建 Subscription CR，请运行以下命令：

```
$ oc create -f self-node-remediation-subscription.yaml
```

验证

1. 检查 CSV 资源来验证安装是否成功：

```
$ oc get csv -n openshift-workload-availability
```

输出示例

```
NAME                         DISPLAY                       VERSION   REPLACES   PHASE
self-node-remediation.v0.8.0      Self Node Remediation Operator   v.0.8.0   self-node-
remediation.v0.7.1          Succeeded
```

2. 验证 Self Node Remediation Operator 是否正在运行：

```
$ oc get deployment -n openshift-workload-availability
```

输出示例

```
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
self-node-remediation-controller-manager    1/1     1            1           28h
```

3. 验证 Self Node Remediation Operator 是否已创建 SelfNodeRemediationConfig CR:

```
$ oc get selfnoderemediationconfig -n openshift-workload-availability
```

输出示例

```
NAME                         AGE
self-node-remediation-config   28h
```

4. 验证每个自节点补救 pod 是否已调度并在每个 worker 节点和 control plane 节点上运行：

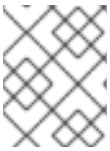
```
$ oc get daemonset -n openshift-workload-availability
```

输出示例

```
NAME                 DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE
SELECTOR  AGE
self-node-remediation-ds  6       6       6      6           6          <none>         28h
```

2.5. 配置自节点修复 OPERATOR

Self Node Remediation Operator 创建 SelfNodeRemediationConfig CR 和 SelfNodeRemediationTemplate 自定义资源定义(CRD)。



注意

为了避免意外重启特定节点，Node Maintenance Operator 会将节点置于维护模式，并自动添加阻止 SNR daemonset 在特定节点中运行的节点选择器。

2.5.1. 了解 Self Node Remediation Operator 配置

Self Node Remediation Operator 创建了名为 self-node-remediation-config 的 SelfNodeRemediationConfig CR。CR 在 Self Node Remediation Operator 的命名空间中创建。

SelfNodeRemediationConfig CR 的更改重新创建 Self Node Remediation 守护进程集。

SelfNodeRemediationConfig CR 类似于以下 YAML 文件：

```
apiVersion: self-node-remediation.medik8s.io/v1alpha1
kind: SelfNodeRemediationConfig
metadata:
  name: self-node-remediation-config
  namespace: openshift-workload-availability
spec:
  safeTimeToAssumeNodeRebootedSeconds: 180 (1)
  watchdogFilePath: /dev/watchdog (2)
  isSoftwareRebootEnabled: true (3)
  apiServerTimeout: 15s (4)
  apiCheckInterval: 5s (5)
  maxApiErrorThreshold: 3 (6)
  peerApiServerTimeout: 5s (7)
  peerDialTimeout: 5s (8)
  peerRequestTimeout: 5s (9)
  peerUpdateInterval: 15m (10)
  hostPort: 30001 (11)
  customDsTolerations: (12)
    - effect: NoSchedule
      key: node-role.kubernetes.io.infra
      operator: Equal
      value: "value1"
      tolerationSeconds: 3600
```

(1) 指定 Operator 在恢复不健康节点上运行的受影响工作负载前等待的时间。在仍然在故障节点上运行时启动替换 pod 可能会导致数据崩溃并违反运行一次语义。时间持续时间必须等于或大于 Operator 计算的最小值，使用 **ApiServerTimeout**,**ApiCheckInterval**,**maxApiErrorThreshold**,**peerDialTimeout**, 和 **peerRequestTimeout** 字段的值。在日志中，您可以引用 计算的 **minTimeToAssumeNodeRebooted is: [value]** 值来查看 Operator 计算的最小值。指定低于计算的最小值的值可防止 Operator 无法正常工作。

(2) 指定节点中 watchdog 设备的文件路径。如果您为 watchdog 设备输入了一个错误的路径，则 Self Node Remediation Operator 会自动检测到 softdog 设备路径。

如果 watchdog 设备不可用，则 **SelfNodeRemediationConfig** CR 将使用软件重启。

(3) 指定是否启用不健康节点的软件重启。默认情况下，**SoftwareRebootEnabled** 的值设置为 **true**。要禁用软件重启，请将参数设置为 **false**。

(4) 指定检查每个 API 服务器的连接的超时持续时间。此超过了此持续时间，Operator 会启动补救。超时持续时间必须大于或等于 10 毫秒。

(5) 指定检查每个 API 服务器的连接的频率。超时持续时间必须大于或等于 1 秒。

(6) 指定一个阈值。达到这个阈值后，节点开始联系其同级服务器。阈值必须大于或等于 1 秒。

(7) 指定对等对等服务器连接 API 服务器的超时时间。超时持续时间必须大于或等于 10 毫秒。

(8) 指定与对等连接建立超时的持续时间。超时持续时间必须大于或等于 10 毫秒。

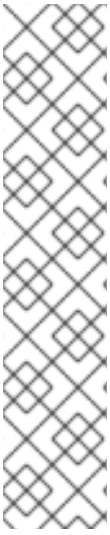
(9) 指定超时从对等点获得响应的时长。超时持续时间必须大于或等于 10 毫秒。

(10) 指定更新对等信息的频率，如 IP 地址。超时持续时间必须大于或等于 10 秒。

11 指定可选值，以更改 Self Node Remediation 代理用于内部通信的端口。该值必须大于 0。默认值为端口 30001。

12 指定在 DaemonSet 上运行的自定义容限自助修复代理，以支持对不同类型的节点的补救。您可以配置以下字段：

- effect: effect 表示污点效果要匹配。如果此字段为空，则所有污点效果都会匹配。指定后，允许的值为 NoSchedule、PreferNoSchedule 和 NoExecute。
- Key ：键是容限应用到的污点键。如果此字段为空，则所有污点键都会匹配。如果键为空，operator 字段必须是 Exists。这种组合意味着匹配所有值和所有键。
- operator ：运算符表示键与值的关系。有效的运算符为 Exists 和 Equal。默认值为 Equal。exists 等同于值的通配符，以便 pod 可以容忍特定类别的所有污点。
- 值 ：容限匹配的污点值。如果运算符是 Exists，则该值应为空，否则它只是一个常规字符串。
- TolerationSeconds ：容限的期间（它必须是 NoExecute，否则此字段将被忽略）容许污点。默认情况下，它没有被设置，这意味着容许污点（即不驱除）。系统会将零值和负值视为 0 （即立即驱除）。
- 通过自定义容限，您可以为 Self Node Remediation 代理 pod 添加容限。如需更多信息，请参阅使用容忍度来控制 OpenShift Logging pod 放置。



注意

您可以编辑由 Self Node Remediation Operator 创建的 self-node-remediation-config CR。但是，当您尝试为 Self Node Remediation Operator 创建新 CR 时，日志中会显示以下信息：

```
controllers.SelfNodeRemediationConfig
ignoring selfnoderemediationconfig CRs that are not named 'self-node-remediation-config'
or not in the namespace of the operator:
'openshift-workload-availability' {"selfnoderemediationconfig": "openshift-workload-availability/selfnoderemediationconfig-copy"}
```

2.5.2. 了解自助节点修复模板配置

Self Node Remediation Operator 还创建 SelfNodeRemediationTemplate 自定义资源定义(CRD)。此 CRD 为节点定义补救策略。可用的补救策略如下：

自动

此补救策略通过让 Self Node Remediation Operator 决定集群最合适的补救策略简化了补救过程。此策略检查集群中是否有 OutOfServiceTaint 策略。如果 OutOfServiceTaint 策略可用，Operator 会选择 OutOfServiceTaint 策略。如果 OutOfServiceTaint 策略不可用，Operator 会选择 ResourceDeletion 策略。自动 是默认的补救策略。

ResourceDeletion

此补救策略移除节点上的 pod，而不是移除节点对象。此策略可以更快地恢复工作负载。

OutOfServiceTaint

此补救策略隐式会导致在节点上删除 pod 和关联的卷附加，而不是移除节点对象。它通过将 OutOfServiceTaint 策略放在节点上来实现此目的。此策略可以更快地恢复工作负载。自 OpenShift Container Platform 版本 4.13 起，此策略在技术预览上被支持，从 OpenShift Container Platform 版本 4.15 开始正式发布。

Self Node Remediation Operator 为策略 self-node-remediation-resource-deletion-template 创建 SelfNodeRemediationTemplate CR，其 ResourceDeletion 补救策略使用。

SelfNodeRemediationTemplate CR 类似于以下 YAML 文件：

```
apiVersion: self-node-remediation.medik8s.io/v1alpha1
kind: SelfNodeRemediationTemplate
metadata:
  creationTimestamp: "2022-03-02T08:02:40Z"
  name: self-node-remediation-<remediation_object>-deletion-template 1
  namespace: openshift-workload-availability
spec:
  template:
    spec:
      remediationStrategy: <remediation_strategy> 2
```

1

根据补救策略指定补救模板的类型。将 <remediation_object> 替换为 resource 或 node; 例如 self-node-remediation-resource-deletion-template。

2

指定补救策略。默认补救策略是 Automatic。

2.5.3. 对自节点修复 Operator 进行故障排除

2.5.3.1. 常规故障排除

问题

您需要使用自助服务修复 Operator 排除问题。

解决方案

检查 Operator 日志。

2.5.3.2. 检查守护进程集

问题

已安装 Self Node Remediation Operator，但守护进程集不可用。

解决方案

检查 Operator 日志中的错误或警告。

2.5.3.3. 失败的补救

问题

一个不健康的节点没有被修复。

解决方案

运行以下命令验证 selfNodeRemediation CR 是否已创建：

```
$ oc get snr -A
```

当节点处于不健康状态时，如果 MachineHealthCheck 控制器没有创建 SelfNodeRemediation CR，请检查 MachineHealthCheck 控制器的日志。此外，请确保 MachineHealthCheck CR 包含使用补救模板所需的规范。

如果创建了 SelfNodeRemediation CR，请确保其名称与不健康的节点或机器对象匹配。

2.5.3.4. 即使在卸载了 Operator 后，守护进程集和其他自节点修复 Operator 资源也存在

问题

即使卸载 Operator 后，也会存在 Self Node Remediation Operator 资源，如守护进程集、配置 CR 和补救模板 CR。

解决方案

要删除 Self Node Remediation Operator 资源，请运行以下命令来删除每种资源类型的资源：

```
$ oc delete ds <self-node-remediation-ds> -n <namespace>
```

```
$ oc delete snrc <self-node-remediation-config> -n <namespace>
```

```
$ oc delete snrt <self-node-remediation-template> -n <namespace>
```

2.5.4. 收集自节点修复 Operator 的数据

要收集有关自助服务修复 Operator 的调试信息，请使用 must-gather 工具。有关 Self Node Remediation Operator 的 must-gather 镜像的详情，请参考 收集有关特定功能的数据。

2.5.5. 其他资源

- 在受限网络中使用 Operator Lifecycle Manager。
- 从集群中删除 Operator

第 3 章 使用隔离代理修复

您可以使用 Fence Agents Remediation Operator 自动修复不健康的节点，类似于 Self Node Remediation Operator。通过使用管理界面或传统的 API，此 Operator 运行一个 fence-agent，以通过电源节点从不健康状态修复节点。

3.1. 关于隔离代理修复 OPERATOR

Fence Agents Remediation (FAR) Operator 使用外部工具 *隔离* 不健康的节点。这些工具是一组隔离代理，每个隔离代理可用于隔离节点的不同环境，并使用重启节点的传统应用程序编程接口(API)调用。通过这样做，FAR 可以最小化有状态应用程序的停机时间，在发生临时故障时恢复计算容量，并增加工作负载的可用性。

FAR 仅在节点变得不健康时隔离节点，它还会尝试 *修复* 该节点不健康。它添加一个污点来驱除无状态 pod，使用隔离代理隔离节点，并在重启后完成补救，使用资源删除来删除任何剩余的工作负载（主要有状态工作负载）。添加污点和删除工作负载可加快工作负载重新调度。

Operator 监视是否有新的或删除的自定义资源(CR)，名为 FenceAgentsRemediation，它根据 CR 的名称触发隔离代理修复节点。FAR 使用 NodeHealthCheck 控制器来检测集群中节点的健康状况。当节点被识别为不健康时，NodeHealthCheck 资源会根据 FenceAgentsRemediation Template CR 创建 FenceAgentsRemediation CR，然后触发 Fence Agents Remediation Operator。

FAR 使用隔离代理隔离 Kubernetes 节点。通常，隔离是将无响应/释放计算机进入安全状态并隔离计算机的过程。隔离代理是一个软件代码，它使用管理界面执行隔离，主要基于电源隔离，启用电源循环、重置或关闭计算机。隔离代理示例是 fence_ipmilan，用于智能平台管理接口(IPMI)环境。

```
apiVersion: fence-agents-remediation.medik8s.io/v1alpha1
kind: FenceAgentsRemediation
metadata:
  name: node-name (1)
  namespace: openshift-workload-availability
spec:
  remediationStrategy: <remediation_strategy> (2)
```

(1) node-name 应该与不健康集群节点的名称匹配。

(2) 指定节点的补救策略。有关可用的补救策略的更多信息，请参阅了解 Fence Agents Remediation Template 配置 主题。

Operator 包括了一组隔离代理，它们也可以在红帽高可用性附加组件中可用，该插件使用 IPMI 或 API 等管理界面来为裸机服务器、虚拟机和云平台置备/重新引导节点。

3.2. 使用 WEB 控制台安装 FENCE AGENTS REMEDIATION OPERATOR

您可以使用 Red Hat OpenShift Web 控制台安装 Fence Agents Remediation Operator。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 在 Red Hat OpenShift Web 控制台中，导航到 Operators → OperatorHub。
2. 从可用的 Operator 列表中选择 Fence Agents Remediation Operator 或 FAR，然后点 Install。
3. 保留安装模式和命名空间的默认选择，以确保将 Operator 安装到 openshift-workload-availability 命名空间中。
4. 点 Install。

验证

确认安装成功：

1. 导航到 Operators → Installed Operators 页面。
2. 检查 Operator 是否安装在 openshift-workload-availability 命名空间中，其状态是否为 Succeeded。

如果 Operator 没有成功安装：

1. 导航到 Operators → Installed Operators 页面，并检查 Status 列中是否有任何错误或故障。

2. 进入到 Workloads → Pods 页面，检查 fence-agents-remediation-controller-manager pod 的日志是否有报告的问题。

3.3. 使用 CLI 安装隔离代理修复 OPERATOR

您可以使用 OpenShift CLI (oc)安装 Fence Agents Remediation Operator。

您可以在自己的命名空间中或 openshift-workload-availability 命名空间中安装 Fence Agents Remediation Operator。

先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 为 Fence Agents Remediation Operator 创建 Namespace 自定义资源(CR)：

   a. 定义 Namespace CR 并保存 YAML 文件，如 workload-availability-namespace.yaml：

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-workload-availability
   ```

   b. 要创建 Namespace CR，请运行以下命令：

   ```
   $ oc create -f workload-availability-namespace.yaml
   ```

2. 创建 OperatorGroup CR：

   a. 定义 OperatorGroup CR 并保存 YAML 文件，如 workload-availability-operator-group.yaml ：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: workload-availability-operator-group
  namespace: openshift-workload-availability
```

   b. 要创建 OperatorGroup CR，请运行以下命令：

```
$ oc create -f workload-availability-operator-group.yaml
```

3. 创建一个 Subscription CR：

   a. 定义 Subscription CR 并保存 YAML 文件，如 fence-agents-remediation-subscription.yaml ：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
   name: fence-agents-remediation-subscription
   namespace: openshift-workload-availability (1)
spec:
   channel: stable
   name: fence-agents-remediation
   source: redhat-operators
   sourceNamespace: openshift-marketplace
   package: fence-agents-remediation
```

   (1) 指定您要安装 Fence Agents Remediation Operator 的命名空间，例如此流程前面概述的 openshift-workload-availability。您可以在 openshift-workload-availability 命名空间中为 Fence Agents Remediation Operator 安装 Subscription CR，其中已有匹配的 OperatorGroup CR。

   b. 要创建 Subscription CR，请运行以下命令：

```
$ oc create -f fence-agents-remediation-subscription.yaml
```

验证

1. 检查 CSV 资源来验证安装是否成功：

```
$ oc get csv -n openshift-workload-availability
```

输出示例

```
NAME                                  DISPLAY                          VERSION   REPLACES   PHASE
fence-agents-remediation.v0.4.0       Fence Agents Remediation Operator   0.4.0   fence-agents-remediation.v0.3.0   Succeeded
```

2. 验证 Fence Agents Remediation Operator 是否正在运行：

```
$ oc get deployment -n openshift-workload-availability
```

输出示例

```
NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
fence-agents-remediation-controller-manager   2/2     2            2           110m
```

3.4. 配置隔离代理修复 OPERATOR

您可以使用 Fence Agents Remediation Operator 创建 FenceAgentsRemediationTemplate 自定义资源(CR)，由 Node Health Check Operator (NHC)使用。此 CR 定义集群中要使用的隔离代理，以及修复节点所需的所有参数。每个隔离代理可能会有许多 FenceAgentsRemediationTemplate CR，对于每个隔离代理来说，当 NHC 被使用时，可以选择 FenceAgentsRemediationTemplate 作为用于电源循环节点的 remediationTemplate。

注意

在当前发行版本中，每个隔离代理可能有很多 FenceAgentsRemediationTemplate CR，但大多数 CR。这是一个已知的限制，将在以后的发行版本中解决。

FenceAgentsRemediationTemplate CR 类似于以下 YAML 文件：

```
apiVersion: fence-agents-remediation.medik8s.io/v1alpha1
kind: FenceAgentsRemediationTemplate
metadata:
  name: fence-agents-remediation-template-fence-ipmilan
  namespace: openshift-workload-availability
spec:
  template:
    spec:
      agent: fence_ipmilan (1)
      nodeparameters: (2)
        --ipport:
          master-0-0: '6230'
          master-0-1: '6231'
          master-0-2: '6232'
          worker-0-0: '6233'
          worker-0-1: '6234'
          worker-0-2: '6235'
      sharedparameters: (3)
        '--action': reboot
        '--ip': 192.168.123.1
        '--lanplus': ''
        '--password': password
        '--username': admin
      retryCount: '5' (4)
      retryInterval: '5' (5)
      timeout: '60' (6)
```

(1) 显示要执行的隔离代理的名称，如 fence_ipmilan。

(2) 显示执行隔离代理的节点特定参数，例如 ipport。

(3) 显示执行隔离代理的集群范围参数，例如 用户名。

(4)

显示失败时重试隔离代理命令的次数。默认尝试次数为 5。

5

显示重试间隔（以秒为单位）。默认值为 5 秒。

6

显示隔离代理命令的超时时间（以秒为单位）。默认为 60 秒。

3.4.1. 了解隔离代理修复模板配置

Fence Agents Remediation Operator 还会创建 FenceAgentsRemediationTemplate 自定义资源定义(CRD)。此 CRD 为旨在更快地恢复工作负载的节点定义补救策略。可用的补救策略如下：

ResourceDeletion

此补救策略移除节点上的 pod，而不是移除节点对象。此策略可以更快地恢复工作负载。

OutOfServiceTaint

此补救策略隐式会导致在节点上删除 pod 和关联的卷附加，而不是移除节点对象。它通过将 OutOfServiceTaint 污点放在节点上来实现此目的。OutOfServiceTaint 策略也代表一个非正常节点关闭。当节点关闭且没有检测到时，会发生非正常节点关闭，而不是触发操作系统关闭。自 OpenShift Container Platform 版本 4.13 起，此策略在技术预览上被支持，从 OpenShift Container Platform 版本 4.15 开始正式发布。

Fence Agents Remediation Operator 为策略 fence-agents-remediation-resource-deletion-template 创建 FenceAgentsRemediationTemplate CR，其使用 ResourceDeletion 补救策略。

FenceAgentsRemediationTemplate CR 类似于以下 YAML 文件：

```
apiVersion: fence-agents-remediation.medik8s.io/v1alpha1
kind: FenceAgentsRemediationTemplate
metadata:
  name: fence-agents-remediation-<remediation_object>-deletion-template 1
  namespace: openshift-workload-availability
spec:
  template:
    spec:
      remediationStrategy: <remediation_strategy> 2
```

1

根据补救策略指定补救模板的类型。将 < remediation_object > 替换为 resource 或 node; 例如 fence-agents-remediation-resource-deletion-template。

2

指定补救策略。补救策略可以是 ResourceDeletion 或 OutOfServiceTaint。

3.5. 对隔离代理修复 OPERATOR 进行故障排除

3.5.1. 常规故障排除

问题

您希望排除 Fence Agents Remediation Operator 的问题。

解决方案

检查 Operator 日志。

```
$ oc logs <fence-agents-remediation-controller-manager-name> -c manager -n
<namespace-name>
```

3.5.2. 失败的补救

问题

一个不健康的节点没有被修复。

解决方案

运行以下命令验证 FenceAgentsRemediation CR 是否已创建：

```
$ oc get far -A
```

如果 NodeHealthCheck 控制器在节点关闭不健康时没有创建 FenceAgentsRemediation CR，请检查 NodeHealthCheck 控制器的日志。另外，请确保 NodeHealthCheck CR 包含使用补救模板所需的规格。

如果创建了 FenceAgentsRemediation CR，请确保其名称与不健康的节点对象匹配。

3.5.3. 卸载 Operator 后存在 fence Agents Remediation Operator 资源

问题

卸载 Operator 后，存在 Fence Agents Remediation Operator 资源，如补救 CR 和补救模板 CR。

解决方案

要删除 Fence Agents Remediation Operator 资源，您可以在卸载前选择 "Delete all operand instance for this operator" 复选框来删除资源。自版本 4.13 起，此复选框功能仅适用于 Red Hat OpenShift。对于所有版本的 Red Hat OpenShift，您可以通过为每个资源类型运行以下相关命令来删除资源：

```
$ oc delete far <fence-agents-remediation> -n <namespace>
```

```
$ oc delete fartemplate <fence-agents-remediation-template> -n <namespace>
```

补救 CR 目前必须由同一实体创建和删除，如 NHC。如果补救 CR 仍然存在，它将与 FAR operator 一起删除。

只有在 NHC 中使用 FAR 时，补救模板 CR fartemplate 才存在。使用 Web 控制台删除 FAR Operator 时，也会删除补救模板 CR fartemplate。

3.6. 收集 FENCE AGENTS REMEDIATION OPERATOR 的数据

要收集有关 Fence Agents Remediation Operator 的调试信息，请使用 must-gather 工具。有关 Fence Agents Remediation Operator 的 must-gather 镜像的详情，请参考 收集有关特定功能的数据。

3.7. FENCE AGENTS REMEDIATION OPERATOR 支持的代理

本节论述了 Fence Agents Remediation Operator 当前支持的代理。

大多数支持的代理可由节点的硬件专有和使用情况分组，如下所示：

1. 裸机

2. **虚拟化**
3. **Intel**
4. **HP**
5. **IBM**
6. **VMware**
7. **Cisco**
8. **APC**
9. **Dell**
10. **其他**

表 3.1. baremetal - 建议使用 Redfish 管理界面，除非不支持。

Agent	描述
fence_redfish	一个 I/O 隔离代理，可与支持 **Redfish** API 的 Out-of-Band 控制器一起使用。
fence_ipmilan[a]	一个 I/O 隔离代理，可用于 **IPMI** 控制的计算机。
[a] 此描述也适用于代理 **fence_ilo3**、**fence_ilo 4**、**fence_ilo 5**、**fence_imm**、**fence_idrac**、**fence_ipmilanplus**。	

表 3.2. 虚拟化

Agent	描述
fence_rhevm	可与 RHEV-M REST API 结合使用的 I/O 隔离代理来隔离 **虚拟机**。

Agent	描述
fence_virt [a]	可与 **虚拟机** 一起使用的 I/O 隔离代理。

[a] 这个描述也适用于代理 **fence_xvm**。

表 3.3. Intel

Agent	描述
fence_amt_ws	可用于 **Intel** AMT (WS)的 I/O 隔离代理。
fence_intelmodular	可与 **Intel** Modular 设备一起使用的 I/O 隔离代理（在 Intel MFSYS25 上测试，也应该使用 MFSYS35）。

表 3.4. HP - iLO 管理界面或 BladeSystem 的代理。

Agent	描述
fence_ilo [a]	一个 I/O 隔离代理，可用于带有 Integrated Light Out (**iLO**) PCI 卡的 HP 服务器。
fence_ilo_ssh [b]	可用于连接到 **iLO** 设备的隔离代理。它通过 ssh 登录到设备并重启指定的 outlet。
fence_ilo_moonshot	可用于 HP Moonshot **iLO** 的 I/O 隔离代理。
fence_ilo_mp	可用于 HP **iLO** MP 的 I/O 隔离代理。
fence_hpblade	可用于 HP **BladeSystem** 和 HP Integrity Superdome X 的 I/O 隔离代理。

[a] 此描述也适用于代理 **fence_ilo2**。

[b] 此描述也适用于代理 **fence_ilo3_ssh**、**fence_ilo4_ssh** 和 **fence_ilo5_ssh**。

表 3.5. IBM

Agent	描述
fence_bladecenter	可用于 **IBM** Bladecenters 的 I/O 隔离代理，带有包含 telnet 支持的最新固件。
fence_ibmblade	可与 **IBM** BladeCenter 机箱一起使用的 I/O 隔离代理。
fence_ipdu	可与 **IBM** iPDU 网络电源开关一起使用的 I/O 隔离代理。
fence_rsa	可与 **IBM** RSA II 管理界面一起使用的 I/O 隔离代理。

表 3.6. VMware

Agent	描述
fence_vmware_rest	一个 I/O 隔离代理，可与 **VMware** API 一起使用来隔离虚拟机。
fence_vmware_soap	一个 I/O 隔离代理，可与具有 SOAP API v4.1+ 的 **VMWare** 产品管理的虚拟机一起使用。

表 3.7. Cisco

Agent	描述
fence_cisco_mds	一个 I/O 隔离代理，可与启用了 SNMP 的设备的任何 **Cisco** MDS 9000 系列一起使用。
fence_cisco_ucs	可与 **Cisco** UCS 一起使用的 I/O 隔离代理来隔离机器。

表 3.8. APC

Agent	描述
fence_apc	可与 **APC** 网络电源开关一起使用的 I/O 隔离代理。
fence_apc_snmp	可与 **APC** 网络电源开关或 Tripplite PDU 设备一起使用的 I/O 隔离代理。

表 3.9. Dell

Agent	描述
fence_drac5	可与 **Dell** Remote Access Card v5 或 CMC (DRAC)一起使用的 I/O 隔离代理。

表 3.10. 其他 - 不使用上表中列出的使用情况的代理。

Agent	描述
fence_brocade	可与 Brocade FC 交换机一起使用的 I/O 隔离代理。
fence_compute	用于告知 Nova 计算节点已停机并重新调度标记实例的资源。
fence_eaton_snmp	一个 I/O 隔离代理，可与 Eaton 网络电源开关一起使用。
fence_emerson	可用于 MPX 和 MPH2 管理的机架 PDU 的 I/O 隔离代理。
fence_eps	一个 I/O 隔离代理，可与 ePowerSwitch 8M+ 电源开关一起使用，以隔离连接的机器。

Agent	描述
fence_evacuate	用于重新调度标记实例的资源。
fence_heuristics_ping	可与 ping-heuristics 一起使用的资源来控制在同一隔离级别执行另一个隔离代理。
fence_ifmib	可与任何 SNMP IF-MIB 功能设备一起使用的 I/O 隔离代理。
fence_kdump	可与 kdump 崩溃恢复服务一起使用的 I/O 隔离代理。
fence_mpath	一个 I/O 隔离代理，可用于 SCSI-3 持久保留来控制访问多路径设备。
fence_rsb	可用于 Fujitsu-Siemens RSB 管理界面的 I/O 隔离代理。
fence_sbd	一个 I/O 隔离代理，可用于 sbd （共享存储）的环境。
fence_scsi	一个 I/O 隔离代理，可用于 SCSI-3 持久保留来控制对共享存储设备的访问。
fence_wti	一个 I/O 隔离代理，可与 WTI 网络电源交换机(NPS)一起使用。

3.8. 其他资源

- **在受限网络中使用 Operator Lifecycle Manager。**
- **从集群中删除 Operator**

第 4 章 使用机器删除修复

您可以使用 Machine Deletion Remediation Operator 使用 Machine API 重新置备不健康的节点。您可以将 Machine Deletion Remediation Operator 与 Node Health Check Operator 搭配使用。

4.1. 关于 MACHINE DELETION REMEDIATION OPERATOR

Machine Deletion Remediation (MDR)操作器可用于 NodeHealthCheck 控制器，以使用 Machine API 重新置备不健康的节点。MDR 遵循节点上的注解到关联的机器对象，确认它拥有自己的控制器（如 MachineSetController），并删除它。删除机器 CR 后，拥有的控制器会创建一个替换。

MDR 的先决条件包括：

- 基于 Machine API 的集群，可以通过编程方式销毁并创建集群节点，
- 与机器关联的节点，以及
- 声明性管理的机器。

然后，您可以修改 NodeHealthCheck CR，以使用 MDR 作为其补救器。文档中提供了 MDR 模板对象和 NodeHealthCheck 配置示例。

MDR 进程按如下方式工作：

- Node Health Check Operator 检测到不健康的节点并创建 MDR CR。
- MDR Operator 会监视与不健康节点关联的 MDR CR，并在机器有自己的控制器时删除它。
- 当节点再次处于健康状态时，NodeHealthCheck 控制器会删除 MDR CR。

4.2. 使用 WEB 控制台安装 MACHINE DELETION REMEDIATION OPERATOR

您可以使用 Red Hat OpenShift Web 控制台安装 Machine Deletion Remediation Operator。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 在 Red Hat OpenShift Web 控制台中，导航到 Operators → OperatorHub。
2. 从可用的 Operator 列表中选择 Machine Deletion Remediation Operator 或 MDR，然后点 Install。
3. 保留安装模式和命名空间的默认选择，以确保将 Operator 安装到 openshift-workload-availability 命名空间中。
4. 点 Install。

验证

确认安装成功：

1. 导航到 Operators → Installed Operators 页面。
2. 检查 Operator 是否安装在 openshift-workload-availability 命名空间中，其状态是否为 Succeeded。

如果 Operator 没有成功安装：

1. 导航到 Operators → Installed Operators 页面，并检查 Status 列中是否有任何错误或故障。
2. 导航到 Workloads → Pods 页面，检查 openshift-workload-availability 项目中的 pod 日志以了解报告的问题。

4.3. 使用 CLI 安装 MACHINE DELETION REMEDIATION OPERATOR

您可以使用 OpenShift CLI (oc)安装 Machine Deletion Remediation Operator。

您可以在自己的命名空间中或 openshift-workload-availability 命名空间中安装 Machine Deletion Remediation Operator。

先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 为 Machine Deletion Remediation Operator 创建 Namespace 自定义资源(CR)：
   a. 定义 Namespace CR 并保存 YAML 文件，如 workload-availability-namespace.yaml ：

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-workload-availability
```

   b. 要创建 Namespace CR，请运行以下命令：

```
$ oc create -f workload-availability-namespace.yaml
```

2. 创建 OperatorGroup CR：
   a. 定义 OperatorGroup CR 并保存 YAML 文件，如 workload-availability-operator-group.yaml ：

```
apiVersion: operators.coreos.com/v1
```

```
kind: OperatorGroup
metadata:
  name: workload-availability-operator-group
  namespace: openshift-workload-availability
```

b. 要创建 OperatorGroup CR，请运行以下命令：

```
$ oc create -f workload-availability-operator-group.yaml
```

3. 创建一个 Subscription CR：

a. 定义 Subscription CR 并保存 YAML 文件，如 machine-deletion-remediation-subscription.yaml ：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
   name: machine-deletion-remediation-operator
   namespace: openshift-workload-availability (1)
spec:
   channel: stable
   name: machine-deletion-remediation-operator
   source: redhat-operators
   sourceNamespace: openshift-marketplace
   package: machine-deletion-remediation
```

(1) 指定您要安装 Machine Deletion Remediation Operator 的命名空间。在 openshift-workload-availability Subscription CR 中安装 Machine Deletion Remediation Operator 时，Namespace 和 OperatorGroup CR 已存在。

b. 要创建 Subscription CR，请运行以下命令：

```
$ oc create -f machine-deletion-remediation-subscription.yaml
```

验证

1. 检查 CSV 资源来验证安装是否成功：

```
$ oc get csv -n openshift-workload-availability
```

输出示例

```
NAME                                    DISPLAY                          VERSION   REPLACES   PHASE
machine-deletion-remediation.v0.3.0     Machine Deletion Remediation Operator   0.3.0
machine-deletion-remediation.v0.2.1                                      Succeeded
```

4.4. 配置 MACHINE DELETION REMEDIATION OPERATOR

您可以使用 Machine Deletion Remediation Operator 和 Node Health Check Operator 创建 MachineDeletionRemediationTemplate 自定义资源(CR)。此 CR 为节点定义补救策略。

MachineDeletionRemediationTemplate CR 类似于以下 YAML 文件：

```
apiVersion: machine-deletion-remediation.medik8s.io/v1alpha1
kind: MachineDeletionRemediationTemplate
metadata:
  name: machinedeletionremediationtemplate-sample
  namespace: openshift-workload-availability
spec:
  template:
    spec: {}
```

4.5. 对 MACHINE DELETION REMEDIATION OPERATOR 进行故障排除

4.5.1. 常规故障排除

问题

您需要使用 Machine Deletion Remediation Operator 排除问题。

解决方案

检查 Operator 日志。

```
$ oc logs <machine-deletion-remediation-controller-manager-name> -c manager -n
<namespace-name>
```

4.5.2. 失败的补救

问题

一个不健康的节点没有被修复。

解决方案

运行以下命令验证 MachineDeletionRemediation CR 是否已创建：

```
$ oc get mdr -A
```

如果 NodeHealthCheck 控制器在节点不健康时没有创建 MachineDeletionRemediation CR，请检查 NodeHealthCheck 控制器的日志。另外，请确保 NodeHealthCheck CR 包含使用补救模板所需的规格。

如果创建了 MachineDeletionRemediation CR，请确保其名称与不健康的节点对象匹配。

4.5.3. 即使卸载 Operator 后也会存在机器删除修复 Operator 资源

问题

Machine Deletion Remediation Operator 资源（如补救 CR 和补救模板 CR ）在卸载 Operator 后也存在。

解决方案

要删除 Machine Deletion Remediation Operator 资源，您可以在卸载前选择 Delete all operand instances for this operator 复选框来删除资源。自版本 4.13 起，此复选框功能仅适用于 Red Hat OpenShift。对于所有版本的 Red Hat OpenShift，您可以通过为每个资源类型运行以下相关命令来删除资源：

```
$ oc delete mdr <machine-deletion-remediation> -n <namespace>
```

```
$ oc delete mdrt <machine-deletion-remediation-template> -n <namespace>
```

补救 CR mdr 必须被同一实体创建和删除，如 NHC。如果补救 CR mdr 仍然存在，它将与 MDR operator 一起删除。

只有在 NHC 中使用 MDR 时，补救模板 CR mdrt 才存在。使用 Web 控制台删除 MDR Operator 时，也会删除补救模板 CR mdrt。

4.6. 收集 MACHINE DELETION REMEDIATION OPERATOR 的数据

要收集有关 Machine Deletion Remediation Operator 的调试信息，请使用 must-gather 工具。有关 Machine Deletion Remediation Operator 的 must-gather 镜像的详情，请参考 收集有关特定功能的数据。

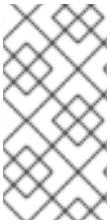
4.7. 其他资源

- 在受限网络中使用 Operator Lifecycle Manager。
- 从集群中删除 Operator

第 5 章 使用机器健康检查修复节点

机器健康检查自动修复特定机器池中不健康的机器。

5.1. 关于机器健康检查



注意

您只能对使用 control plane 机器集的集群中的 control plane 机器应用机器健康检查。

要监控机器的健康状况，创建资源来定义控制器的配置。设置要检查的条件（例如，处于 NotReady 状态达到五分钟或 node-problem-detector 中显示了持久性状况），以及用于要监控的机器集合的标签。

监控 MachineHealthCheck 资源的控制器会检查定义的条件。如果机器无法进行健康检查，则会自动删除机器并创建一个机器来代替它。删除机器之后，您会看到机器被删除事件。

为限制删除机器造成的破坏性影响，控制器一次仅清空并删除一个节点。如果目标机器池中不健康的机器池中不健康的机器数量大于 maxUnhealthy 的值，则补救会停止，需要启用手动干预。

注意

请根据工作负载和要求仔细考虑超时。

- 超时时间较长可能会导致不健康的机器上的工作负载长时间停机。
- 超时时间太短可能会导致补救循环。例如，检查 NotReady 状态的超时时间必须足够长，以便机器能够完成启动过程。

要停止检查，请删除资源。

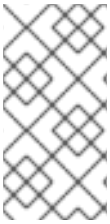
5.1.1. 部署机器健康检查时的限制

部署机器健康检查前需要考虑以下限制：

- 只有机器集拥有的机器才可以由机器健康检查修复。
- 如果机器的节点从集群中移除，机器健康检查会认为机器不健康，并立即修复机器。
- 如果机器对应的节点在 nodeStartupTimeout 之后没有加入集群，则会修复机器。
- 如果 Machine 资源阶段为 Failed，则会立即修复机器。

5.2. 配置机器健康检查以使用 SELF NODE REMEDIATION OPERATOR

使用以下步骤将 worker 或 control-plane 机器健康检查配置为使用 Self Node Remediation Operator 作为补救供应商。



注意

要使用 Self Node Remediation Operator 作为机器健康检查的补救供应商，机器必须在集群中有一个关联的节点。

先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 创建 SelfNodeRemediationTemplate CR：
   a. 定义 SelfNodeRemediationTemplate CR：

```
apiVersion: self-node-remediation.medik8s.io/v1alpha1
```

```
kind: SelfNodeRemediationTemplate
metadata:
  namespace: openshift-machine-api
  name: selfnoderemediationtemplate-sample
spec:
  template:
    spec:
      remediationStrategy: Automatic (1)
```

(1) 指定补救策略。默认补救策略是 Automatic。

b. 要创建 SelfNodeRemediationTemplate CR，请运行以下命令：

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

2. 创建或更新 MachineHealthCheck CR 以指向 SelfNodeRemediationTemplate CR：

a. 定义或更新 MachineHealthCheck CR：

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineHealthCheck
metadata:
  name: machine-health-check
  namespace: openshift-machine-api
spec:
  selector:
    matchLabels: (1)
      machine.openshift.io/cluster-api-machine-role: "worker"
      machine.openshift.io/cluster-api-machine-type: "worker"
  unhealthyConditions:
  - type:    "Ready"
    timeout: "300s"
    status: "False"
  - type:    "Ready"
    timeout: "300s"
    status: "Unknown"
  maxUnhealthy: "40%"
  nodeStartupTimeout: "10m"
  remediationTemplate: (2)
    kind: SelfNodeRemediationTemplate
    apiVersion: self-node-remediation.medik8s.io/v1alpha1
    name: selfnoderemediationtemplate-sample
```

(1)

2 指定补救模板的详情。

b. 要创建 MachineHealthCheck CR，请运行以下命令：

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

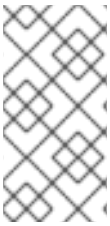
c. 要更新 MachineHealthCheck CR，请运行以下命令：

```
$ oc apply -f <mhc-name>.yaml
```

第 6 章 使用节点健康检查修复节点

您可以使用 Node Health Check Operator 来识别不健康的节点。Operator 使用 Self Node Remediation Operator 来修复不健康的节点。

如需有关 Self Node Remediation Operator 的更多信息，请参阅使用自节点修复 章节。



注意

由于 Red Hat OpenShift Service on AWS (ROSA)集群上存在预安装的机器健康检查，Node Health Check Operator 无法在此环境中正常工作。

6.1. 关于 NODE HEALTH CHECK OPERATOR

Node Health Check Operator 检测到集群中的节点的健康状态。NodeHealthCheck 控制器创建 NodeHealthCheck 自定义资源(CR)，它定义了一组条件和阈值来确定节点的健康状态。

Node Health Check Operator 还将 Self Node Remediation Operator 安装为默认的补救提供程序。

当 Node Health Check Operator 检测到不健康的节点时，它会创建一个补救 CR 触发补救供应商。例如，控制器创建 SelfNodeRemediation CR，它会触发 Self Node Remediation Operator 来修复不健康的节点。

NodeHealthCheck CR 类似于以下 YAML 文件，self-node-remediation 作为补救供应商：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: nodehealthcheck-sample
spec:
  minHealthy: 51% 1
  pauseRequests: 2
    - <pause-test-cluster>
  remediationTemplate: 3
    apiVersion: self-node-remediation.medik8s.io/v1alpha1
    name: self-node-remediation-resource-deletion-template
    namespace: openshift-workload-availability
    kind: SelfNodeRemediationTemplate
  escalatingRemediations: 4
    - remediationTemplate:
```

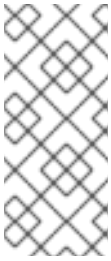
```
    apiVersion: self-node-remediation.medik8s.io/v1alpha1
    name: self-node-remediation-resource-deletion-template
    namespace: openshift-workload-availability
    kind: SelfNodeRemediationTemplate
  order: 1
  timeout: 300s
selector: 5
  matchExpressions:
   - key: node-role.kubernetes.io/worker
     operator: Exists
unhealthyConditions: 6
  - type: Ready
    status: "False"
    duration: 300s 7
  - type: Ready
    status: Unknown
    duration: 300s 8
```

1

指定补救供应商同时修复目标池中节点所需的健康节点数量（以百分比或数量）。如果健康节点的数量等于或超过 minHealthy 设定的限制，则会出现补救。默认值为 51%。

2

防止任何新的补救启动，同时允许持续补救保留。默认值为空。但是，您可以输入字符串数组来识别暂停补救的原因。例如 pause-test-cluster。



注意

在升级过程中，集群中的节点可能会临时不可用，并被识别为不健康。对于 worker 节点，当 Operator 检测到集群正在升级时，它会停止修复新的不健康节点，以防止此类节点重新引导。

3

指定补救供应商的补救模板。例如，通过 Self Node Remediation Operato。remediationTemplate 与 escalatingRemediations 相互排斥。

4

使用 order 和 timeout 字段指定 RemediationTemplates 列表。要获得健康的节点，请使用此字段序列并配置多个补救。此策略提高了获取健康节点的可能性，而不是基于可能不成功的单一补救。order 字段决定调用补救的顺序（低顺序 = 早期调用）。timeout 字段决定何时调用下一个补救。escalatingRemediations 与 remediationTemplate 相互排斥。

5

6

指定决定节点是否被视为不健康的条件列表。

7 8

指定节点状况的超时持续时间。如果在超时时间内满足了条件，则会修复该节点。超时时间较长可能会导致不健康节点上的工作负载长时间停机。

NodeHealthCheck CR 类似于以下 YAML 文件，metal3 作为补救供应商：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: nhc-worker-metal3
spec:
  minHealthy: 30%
  remediationTemplate:
    apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
    kind: Metal3RemediationTemplate
    name: metal3-remediation
    namespace: openshift-machine-api
  selector:
    matchExpressions:
    - key: node-role.kubernetes.io/worker
      operator: Exists
  unhealthyConditions:
  - duration: 300s
    status: 'False'
    type: Ready
  - duration: 300s
    status: 'Unknown'
    type: Ready
```

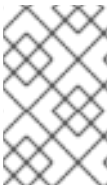
注意

matchExpressions 只是示例；您必须根据具体需求映射您的机器组。

Metal3RemediationTemplate 类似于以下 YAML 文件，metal3 作为补救供应商：

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: Metal3RemediationTemplate
metadata:
  name: metal3-remediation
```

```
  namespace: openshift-machine-api
spec:
  template:
    spec:
      strategy:
        retryLimit: 1
        timeout: 5m0s
        type: Reboot
```



注意

除了创建 NodeHealthCheck CR 外，还必须创建 Metal3RemediationTemplate。

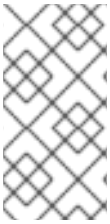
6.1.1. 了解 Node Health Check Operator 工作流

当节点标识为不健康状态时，Node Health Check Operator 会检查其他节点不健康的数量。如果健康的节点数量超过 NodeHealthCheck CR 的 minHealthy 字段中指定的数量，控制器会从补救供应商在外部补救模板中提供的详细信息创建一个补救 CR。补救后，kubelet 会更新节点的健康状况。

当节点处于健康状态时，控制器会删除外部补救模板。

6.1.2. 关于节点健康检查如何防止与机器健康检查冲突

当同时部署节点健康检查和机器健康检查时，节点健康检查会避免与机器健康检查冲突。



注意

Red Hat OpenShift 将 machine-api-termination-handler 部署为默认的 MachineHealthCheck 资源。

以下列表概述了部署节点健康检查和机器健康检查时的系统行为：

- 如果只有默认机器健康检查，则节点健康检查将继续识别不健康的节点。但是，节点健康检查会忽略处于 Terminating 状态的不健康节点。默认机器健康检查处理处于 Terminating 状态的不健康节点。

  日志消息示例

> **INFO MHCChecker ignoring unhealthy Node, it is terminating and will be handled by MHC {"NodeName": "node-1.example.com"}**

- 如果修改了默认机器健康检查（例如，unhealthyConditions 为 Ready），或者是否创建额外的机器健康检查，则节点健康检查被禁用。

日志消息示例

> INFO controllers.NodeHealthCheck disabling NHC in order to avoid conflict with custom MHCs configured in the cluster {"NodeHealthCheck": "/nhc-worker-default"}

- 当只存在默认的机器健康检查，则会重新启用节点健康检查。

日志消息示例

> INFO controllers.NodeHealthCheck re-enabling NHC, no conflicting MHC configured in the cluster {"NodeHealthCheck": "/nhc-worker-default"}

6.2. CONTROL PLANE 隔离

在早期版本中，您可以在 worker 节点上启用自节点修复和 Node Health Check。如果节点失败，您现在可以在 control plane 节点上遵循补救策略。

不要将相同的 NodeHealthCheck CR 用于 worker 节点和 control plane 节点。将 worker 节点和 control plane 节点分组在一起可能会导致评估最小健康节点数，并导致意外或缺失的补救。这是因为 Node Health Check Operator 处理 control plane 节点的方式。您应该在自己的组中对 control plane 节点进行分组，并在自己的组中对 worker 节点进行分组。如果需要，您还可以创建多个 worker 节点组。

补救策略的注意事项：

- 避免涉及多个配置的节点健康检查配置重叠同一节点，因为它们可能会导致意外行为。本建议适用于 worker 和 control plane 节点。

- Node Health Check Operator 实现了一个硬性限制，用于一次修复一个 control plane 节点。不应该同时修复多个 control plane 节点。

6.3. 使用 WEB 控制台安装 NODE HEALTH CHECK OPERATOR

您可以使用 Red Hat OpenShift Web 控制台安装 Node Health Check Operator。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 在 Red Hat OpenShift Web 控制台中，导航到 Operators → OperatorHub。
2. 选择 Node Health Check Operator，然后点 Install。
3. 保留安装模式和命名空间的 默认选择，以确保将 Operator 安装到 openshift-workload-availability 命名空间中。
4. 确保 Console 插件 被设置为 Enable。
5. 点 Install。

验证

确认安装成功：

1. 导航到 Operators → Installed Operators 页面。

2. 检查 Operator 是否安装在 openshift-workload-availability 命名空间中，其状态是否为 Succeeded。

如果 Operator 没有成功安装：

1. 导航到 Operators → Installed Operators 页面，并检查 Status 列中是否有任何错误或故障。

2. 导航到 Workloads → Pods 页面，检查 openshift-workload-availability 项目中报告问题的 pod 的日志。

6.4. 使用 CLI 安装 NODE HEALTH CHECK OPERATOR

您可以使用 OpenShift CLI （oc）安装 Node Health Check Operator。

您可以在自己的命名空间中或 openshift-workload-availability 命名空间中安装 Node Health Check Operator。

先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 为 Node Health Check Operator 创建 Namespace 自定义资源 (CR)：

   a. 定义 Namespace CR 并保存 YAML 文件，如 node-health-check-namespace.yaml：

```
apiVersion: v1
```

```
kind: Namespace
metadata:
  name: openshift-workload-availability
```

b. 要创建 Namespace CR，请运行以下命令：

```
$ oc create -f node-health-check-namespace.yaml
```

2. 创建 OperatorGroup CR：

a. 定义 OperatorGroup CR 并保存 YAML 文件，如 workload-availability-operator-group.yaml ：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: workload-availability-operator-group
  namespace: openshift-workload-availability
```

b. 要创建 OperatorGroup CR，请运行以下命令：

```
$ oc create -f workload-availability-operator-group.yaml
```

3. 创建一个 Subscription CR：

a. 定义 Subscription CR 并保存 YAML 文件，如 node-health-check-subscription.yaml ：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
   name: node-health-check-operator
   namespace: openshift-workload-availability 1
spec:
   channel: stable 2
   installPlanApproval: Manual 3
   name: node-healthcheck-operator
   source: redhat-operators
   sourceNamespace: openshift-marketplace
   package: node-healthcheck-operator
```

1

指定您要安装 Node Health Check Operator 的命名空间。要在 openshift-workload-availability 命名空间中安装 Node Health Check Operator，请在 Subscription CR 中指定 openshift-workload-availability。

2

指定订阅的频道名称。要升级到 Node Health Check Operator 的最新版本，您必须手动将订阅的频道名称从 candidate 改为 stable。

3

如果您的指定版本被目录中的后续版本取代，则将批准策略设置为 Manual。此计划阻止自动升级到更新的版本，且需要在启动 CSV 可以完成安装前手动批准。

b. 要创建 Subscription CR，请运行以下命令：

```
$ oc create -f node-health-check-subscription.yaml
```

验证

1. 检查 CSV 资源来验证安装是否成功：

```
$ oc get csv -n openshift-workload-availability
```

输出示例

```
NAME                                DISPLAY                     VERSION  REPLACES PHASE
node-healthcheck-operator.v0.8.0  Node Health Check Operator  0.8.0   node-healthcheck-operator.v0.7.0          Succeeded
```

2. 验证 Node Health Check Operator 是否正在运行：

```
$ oc get deployment -n openshift-workload-availability
```

输出示例

```
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
node-healthcheck-controller-manager         2/2     2            2           10d
```

6.5. 创建节点健康检查

使用 Web 控制台，您可以创建一个节点健康检查来标识不健康的节点，并指定补救类型和策略来修复它们。

流程

1. 从 Red Hat OpenShift Web 控制台的 Administrator 视角，点 Compute → NodeHealthChecks → CreateNodeHealthCheck。

2. 指定是否使用 Form view 或 YAML 视图配置节点健康检查。

3. 输入节点健康检查的名称。名称必须包含小写、字母数字字符、'-' 或 '.'，且必须以字母数字字符开头和结尾。

4. 指定 Remediator 类型，以及 Self node remediation 或其他。Self 节点补救选项是安装 Node Health Check Operator 的 Self Node Remediation Operator 的一部分。选择 Other 需要输入 API 版本、Kind、Name 和 Namespace，然后指向补救器的补救模板资源。

5. 通过指定要修复的 Nodes 标签来创建节点选择。选择与要检查的标签匹配。如果指定了多个标签，节点必须包含每个标签。默认值为空，用于选择 worker 和 control-plane 节点。

注意

使用 Self Node Remediation Operator 创建节点健康检查时，您必须选择 node-role.kubernetes.io/worker 或 node-role.kubernetes.io/control-plane 作为值。

6. 使用 NodeHealthCheck 来修复目标池中的节点所需的最小健康节点数量（百分比或数

字）。如果健康的节点数量等于或超过 Min healthy 设定的限制，则会出现补救。默认值为 51%。

7. 指定一个非健康条件列表，在决定一个端点被认为是非健康以及需要的补救时使用这些条件。您可以指定 Type、Status 和 Duration。您还可以创建自己的自定义类型。

8. 点 Create 创建节点健康检查。

验证

- 进入到 Compute → NodeHealthCheck 页面，再验证是否列出了对应的节点健康检查，并且显示其状态。创建后，可以暂停、修改和删除节点健康检查。

6.6. 收集 NODE HEALTH CHECK OPERATOR 的数据

要收集有关 Node Health Check Operator 的调试信息，请使用 must-gather 工具。有关 Node Health Check Operator 的 must-gather 镜像的详情，请参阅收集有关特定功能的数据。

6.7. 其他资源

- 更改 Operator 的更新频道
- 在受限网络中使用 Operator Lifecycle Manager。

第 7 章 使用 NODE MAINTENANCE OPERATOR 将节点置于维护模式

您可以使用 oc adm 实用程序或 NodeMaintenance 自定义资源(CR)来使用 Node Maintenance Operator 将节点置于维护模式。

7.1. 关于 NODE MAINTENANCE OPERATOR

Node Maintenance Operator 监视是否有新的或删除的 NodeMaintenance CR。当检测到新的 NodeMaintenance CR 时，不会调度新的工作负载，并且该节点从集群的其余部分中分离。所有可被驱除的 pod 都会从节点上驱除。删除 NodeMaintenance CR 时，CR 中引用的节点将可用于新工作负载。

注意

使用 NodeMaintenance CR 进行节点维护任务可实现与 oc adm cordon 和 oc adm drain 命令相同的结果，使用标准的 Red Hat OpenShift CR 处理。

7.2. 安装 NODE MAINTENANCE OPERATOR

您可以使用 Web 控制台或 OpenShift CLI(oc)安装 Node Maintenance Operator。

注意

如果在集群中安装了 OpenShift Virtualization 版本 4.10 或更少，它包括了一个过时的 Node Maintenance Operator 版本。

7.2.1. 使用 Web 控制台安装 Node Maintenance Operator

您可以使用 Red Hat OpenShift Web 控制台安装 Node Maintenance Operator。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 在 Red Hat OpenShift Web 控制台中，导航到 Operators → OperatorHub。

2. 选择 **Node Maintenance Operator**，然后点 **Install**。

3. 保留安装模式和命名空间的 默认选择，以确保将 **Operator** 安装到 **openshift-workload-availability** 命名空间中。

4. 点 **Install**。

验证

确认安装成功：

1. 导航到 **Operators** → **Installed Operators** 页面。

2. 检查 **Operator** 是否安装在 **openshift-workload-availability** 命名空间中，其状态是否为 **Succeeded**。

如果 **Operator** 没有成功安装：

1. 导航到 **Operators** → **Installed Operators** 页面，并检查 **Status** 列中是否有任何错误或故障。

2. 进入到 **Operators** → **Installed Operators** → **Node Maintenance Operator** → **Details** 页面，并在创建 **pod** 前检查 **Conditions** 部分是否有错误。

3. 进入到 **Workloads** → **Pods** 页面，在已安装的命名空间中搜索 **Node Maintenance Operator pod**，并在 **Logs** 选项卡中检查日志。

7.2.2. 使用 CLI 安装 Node Maintenance Operator

您可以使用 **OpenShift CLI(oc)**安装 **Node Maintenance Operator**。

您可以在自己的命名空间中或 **openshift-workload-availability** 命名空间中安装 **Node Maintenance Operator**。

先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

流程

1. 为 Node Maintenance Operator 创建一个 Namespace CR：
   a. 定义 Namespace CR 并保存 YAML 文件，如 workload-availability-namespace.yaml ：

```
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-workload-availability
```

   b. 要创建 Namespace CR，请运行以下命令：

```
$ oc create -f workload-availability-namespace.yaml
```

2. 创建 OperatorGroup CR：
   a. 定义 OperatorGroup CR 并保存 YAML 文件，如 workload-availability-operator-group.yaml ：

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: workload-availability-operator-group
  namespace: openshift-workload-availability
```

   b. 要创建 OperatorGroup CR，请运行以下命令：

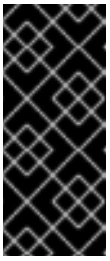
```
$ oc create -f workload-availability-operator-group.yaml
```

3. 创建一个 Subscription CR：

a. 定义 Subscription CR，并保存 YAML 文件，如 node-maintenance-subscription.yaml ：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: node-maintenance-operator
  namespace: openshift-workload-availability 1
spec:
  channel: stable
  installPlanApproval: Automatic
  name: node-maintenance-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  package: node-maintenance-operator
```

1 指定您要安装 Node Maintenance Operator 的命名空间。



重要

要在 openshift-workload-availability 命名空间中安装 Node Maintenance Operator，请在 Subscription CR 中指定 openshift-workload-availability。

b. 要创建 Subscription CR，请运行以下命令：

```
$ oc create -f node-maintenance-subscription.yaml
```

验证

1. 检查 CSV 资源来验证安装是否成功：

```
$ oc get csv -n openshift-workload-availability
```

输出示例

-

```
NAME                                    DISPLAY                    VERSION   REPLACES  PHASE
node-maintenance-operator.v5.3.0   Node Maintenance Operator   5.3.0   node-
maintenance-operator.v5.2.1          Succeeded
```

2. 验证 Node Maintenance Operator 是否正在运行：

```
$ oc get deployment -n openshift-workload-availability
```

输出示例

```
NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
node-maintenance-operator-controller-manager   1/1     1            1           10d
```

受限网络环境中支持 Node Maintenance Operator。如需更多信息，请参阅在受限网络中使用 Operator Lifecycle Manager。

7.3. 将节点设置为维护模式

您可以通过 web 控制台或 CLI 使用 NodeMaintenance CR 将节点置于维护模式。

7.3.1. 使用 Web 控制台将节点设置为维护模式

要将节点设置为维护模式，您可以使用 Web 控制台创建 NodeMaintenance 自定义资源(CR)。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。
- 从 OperatorHub 安装 Node Maintenance Operator。

流程

1. 从 Web 控制台中的 Administrator 视角，导航到 Operators → Installed Operators。

2. 从 Operator 列表中选择 Node Maintenance Operator。

3. 在 Node Maintenance 选项卡中，点 Create NodeMaintenance。

4. 在 Create NodeMaintenance 页面中，选择 Form view 或 YAML 视图来 配置 NodeMaintenance CR。

5. 要应用您配置的 NodeMaintenance CR，请点击 Create。

验证

在 Node Maintenance 选项卡中，检查 Status 列并验证其状态是否为 Succeeded。

7.3.2. 使用 CLI 将节点设置为维护模式

您可以使用 NodeMaintenance 自定义资源（CR）将节点置于维护模式。应用 NodeMaintenance CR 时，所有允许的 pod 都会被驱除，且节点不可调度。被驱除的 pod 会被放入到集群中的另一节点中。

先决条件

- 安装 Red Hat OpenShift CLI oc。
- 以具有 cluster-admin 权限的用户身份登录集群。

流程

1. 创建以下 NodeMaintenance CR，并将文件保存为 nodemaintenance-cr.yaml ：

```
apiVersion: nodemaintenance.medik8s.io/v1beta1
kind: NodeMaintenance
metadata:
  name: nodemaintenance-cr 1
```

```
spec:
  nodeName: node-1.example.com 2
  reason: "NIC replacement" 3
```

1

节点维护 CR 的名称。

2

要置于维护模式的节点名称。

3

有关维护原因的纯文本描述。

2. 运行以下命令来应用节点维护 CR：

```
$ oc apply -f nodemaintenance-cr.yaml
```

验证

1. 运行以下命令，检查维护任务的进度：

```
$ oc describe node <node-name>
```

其中 <node-name> 是节点的名称，如 node-1.example.com

2. 检查输出示例：

```
Events:
  Type    Reason                  Age                From     Message
  ----    ------                  ----               ----     -------
  Normal  NodeNotSchedulable      61m                kubelet  Node node-
1.example.com status is now: NodeNotSchedulable
```

7.3.3. 检查当前 NodeMaintenance CR 任务的状态

您可以检查当前 NodeMaintenance CR 任务的状态。

先决条件

- 安装 Red Hat OpenShift CLI oc。
- 以具有 cluster-admin 特权的用户身份登录。

流程

- 运行以下命令，检查当前节点维护任务的状态，如 NodeMaintenance CR 或 nm 对象：

```
$ oc get nm -o yaml
```

输出示例

```
apiVersion: v1
items:
- apiVersion: nodemaintenance.medik8s.io/v1beta1
  kind: NodeMaintenance
  metadata:
...
  spec:
    nodeName: node-1.example.com
    reason: Node maintenance
  status:
    drainProgress: 100 (1)
    evictionPods: 3 (2)
    lastError: "Last failure message" (3)
    lastUpdate: "2022-06-23T11:43:18Z" (4)
    phase: Succeeded
    totalpods: 5 (5)
...
```

(1) 排空节点完成的百分比。

(2) 调度用于驱除的 pod 数量。

3 最新的驱除错误（若有）。

4 最后一次更新状态的时间。

5 节点进入维护模式前的 pod 总数。

7.4. 从维护模式恢复节点

您可以通过 web 控制台或使用 NodeMaintenance CR 从维护模式恢复节点。恢复节点会使节点退出维护模式，并使其可再次调度。

7.4.1. 使用 Web 控制台从维护模式恢复节点

要从维护模式恢复节点，您可以使用 Web 控制台删除 NodeMaintenance 自定义资源(CR)。

先决条件

- 以具有 cluster-admin 特权的用户身份登录。
- 从 OperatorHub 安装 Node Maintenance Operator。

流程

1. 从 Web 控制台中的 Administrator 视角，导航到 Operators → Installed Operators。
2. 从 Operator 列表中选择 Node Maintenance Operator。
3. 在 Node Maintenance 选项卡中，选择您要删除的 NodeMaintenance CR。
4. 点击节点末尾的 Options 菜单

并选择 **Delete NodeMaintenance**。

验证

1. 在 **Red Hat OpenShift** 控制台中，点 **Compute → Nodes**。
2. 检查您删除 **NodeMaintenance CR** 的节点的 **Status** 列，并验证其状态是否为 **Ready**。

7.4.2. 使用 CLI 从维护模式恢复节点

您可以通过删除 **NodeMaintenance CR**，从 **NodeMaintenance CR** 启动的维护模式恢复节点。

先决条件

- 安装 **Red Hat OpenShift CLI oc**。
- 以具有 **cluster-admin** 权限的用户身份登录集群。

流程

- 节点维护任务完成后,删除活跃的 **NodeMaintenance CR**：

```
$ oc delete -f nodemaintenance-cr.yaml
```

输出示例

```
nodemaintenance.nodemaintenance.medik8s.io "maintenance-example" deleted
```

验证

1. 运行以下命令，检查维护任务的进度：

```
$ oc describe node <node-name>
```

其中 <node-name> 是节点的名称，如 node-1.example.com

2. 检查输出示例：

```
Events:
  Type    Reason             Age                From     Message
  ----    ------             ----               ----     -------
  Normal  NodeSchedulable    2m                 kubelet  Node node-1.example.com
status is now: NodeSchedulable
```

7.5. 使用裸机节点

对于使用裸机节点的集群，您可以使用 web 控制台 Actions 控制将节点置于维护模式，并从维护模式恢复节点。

注意

具有裸机节点的集群也可以将节点置于维护模式，并使用 web 控制台和 CLI 从维护模式恢复节点。这些方法通过使用 Web 控制台 Actions 控制，仅适用于裸机集群。

7.5.1. 维护裸机节点

在裸机基础架构上部署 Red Hat OpenShift 时，与在云基础架构上部署相比，您必须考虑其他注意事项。与云环境不同，集群节点被视为临时的，重新置备裸机节点需要大量时间和精力来进行维护任务。

当因为内核错误或 NIC 卡硬件故障造成裸机节点失败时，故障节点上的工作负载需要在集群中的另一个节点上重启，同时问题节点被修复或替换。节点维护模式允许集群管理员安全关闭节点，将工作负载移到集群的其它部分，并确保工作负载不会中断。详细进度和节点状态详情会在维护过程中提供。

7.5.2. 将裸机节点设置为维护模式

使用 Compute → Nodes 列表中每个节点上

的 Options 菜单，或使用 Node Details 屏幕中的 Actions 控制，将裸机节点设置为维护模式。

流程

1. 从 Web 控制台的 Administrator 视角中，点 Compute → Nodes。
2. 您可从此屏幕将节点设置为维护，这有助于对多个虚拟机执行操作，也可通过 Node Details 屏幕进行，其中可查看所选节点的综合详情：
   - 点击节点  末尾的 Options 菜单并选择 Start Maintenance。
   - 点击节点名称以打开 Node Details 屏幕，然后点击 Actions → Start Maintenance。
3. 在确认窗口中点击 Start Maintenance。

该节点不可调度。如果已有带有 LiveMigration 驱除策略的虚拟机，则会实时迁移它们。该节点上的所有其他 pod 和虚拟机均被删除，并会在另一节点上重新创建。

验证

- 进入到 Compute → Nodes 页面，验证对应节点的状态是否为 Under Maintenance。

7.5.3. 从维护模式恢复裸机节点

使用 Compute → Nodes 列表中每个节点上  的 Options 菜单，或使用 Node Details 屏幕中的 Actions 控制，从维护模式恢复裸机节点。

流程

1. 从 Web 控制台的 Administrator 视角中，点 Compute → Nodes。

2. 您可从此屏幕恢复节点，这有助于对多个虚拟机执行操作，也可从 Node Details 屏幕，其中可查看所选节点的综合详情：

   - 点击节点  末尾的 Options 菜单并选择 Stop Maintenance。

   - 点击节点名称以打开 Node Details 屏幕，然后点击 Actions → Stop Maintenance。

3. 在确认窗口中点击 Stop Maintenance。

节点变为可调度。如果在维护之前已有该节点上运行的虚拟机实例，则它们不会自动迁移回该节点。

验证

- 进入到 Compute → Nodes 页面，验证对应节点的状态是否为 Ready。

7.6. 收集 NODE MAINTENANCE OPERATOR 的数据

要收集有关 Node Maintenance Operator 的调试信息，请使用 must-gather 工具。有关 Node Maintenance Operator 的 must-gather 镜像的信息，请参阅收集有关特定功能的数据。

7.7. 其他资源

- 收集集群数据
- 了解如何撤离节点上的 pod
- 了解如何将节点标记为不可调度或可以调度