Workload Availability for Red Hat OpenShift 24.1

Remediation, fencing, and maintenance

Workload Availability remediation, fencing, and maintenance

Last Updated: 2024-04-25

Workload Availability for Red Hat OpenShift 24.1 Remediation, fencing, and maintenance

Workload Availability remediation, fencing, and maintenance

Abstract

Information about workload availability operators and their usage

Table of Contents

PREFACE

PROVIDING FEEDBACK ON WORKLOAD AVAILABILITY FOR RED HAT OPENSHIFT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it. To do so:

1. Go to the JIRA website.
2. Enter a descriptive title in the **Summary** field.
3. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
4. Enter your username in the **Reporter** field.
5. Enter the affected versions in the **Affects Version/s** field.
6. Click **Create** at the bottom of the dialog.

CHAPTER 1. ABOUT NODE REMEDIATION, FENCING, AND MAINTENANCE

Hardware is imperfect and software contains bugs. When node-level failures, such as the kernel hangs or network interface controllers (NICs) fail, the work required from the cluster does not decrease, and workloads from affected nodes need to be restarted somewhere. However, some workloads, such as ReadWriteOnce (RWO) volumes and StatefulSets, might require at-most-one semantics.

Failures affecting these workloads risk data loss, corruption, or both. It is important to ensure that the node reaches a safe state, known as **fencing** before initiating recovery of the workload, known as **remediation** and ideally, recovery of the node also.

It is not always practical to depend on administrator intervention to confirm the true status of the nodes and workloads. To facilitate such intervention, Red Hat OpenShift provides multiple components for the automation of failure detection, fencing and remediation.

1.1. SELF NODE REMEDIATION

The Self Node Remediation Operator is a Red Hat OpenShift add-on Operator that implements an external system of fencing and remediation that reboots unhealthy nodes and deletes resources, such as Pods and VolumeAttachments. The reboot ensures that the workloads are fenced, and the resource deletion accelerates the rescheduling of affected workloads. Unlike other external systems, Self Node Remediation does not require any management interface, like, for example, Intelligent Platform Management Interface (IPMI) or an API for node provisioning.

Self Node Remediation can be used by failure detection systems, like Machine Health Check or Node Health Check.

1.2. FENCE AGENTS REMEDIATION

The Fence Agents Remediation (FAR) Operator is a Red Hat OpenShift add-on operator that automatically remediates unhealthy nodes, similar to the Self Node Remediation Operator. Using a management interface or traditional API, FAR runs a fence-agent to remediate a node from an unhealthy state by power-cycling the node.

FAR is designed to run an existing set of upstream fencing agents for environments with a traditional API end-point, for example, IPMI, for power cycling cluster nodes.

1.3. MACHINE DELETION REMEDIATION

The Machine Deletion Remediation (MDR) Operator is a Red Hat OpenShift add-on Operator that uses the Machine API to reprovision unhealthy nodes. MDR works with NodeHealthCheck (NHC) to create a Custom Resource (CR) for MDR with information about the unhealthy node.

MDR follows the annotation on the node to the associated machine object and confirms that it has an owning controller. MDR proceeds to delete the machine, and then the owning controller recreates a replacement machine.

1.4. MACHINE HEALTH CHECK

Machine Health Check utilizes a Red Hat OpenShift built-in failure detection, fencing and remediation system, which monitors the status of machines and the conditions of nodes. Machine Health Checks can be configured to trigger external fencing and remediation systems, like Self Node Remediation.

1.5. NODE HEALTH CHECK

The Node Health Check Operator is a Red Hat OpenShift add-on Operator that implements a failure detection system that monitors node conditions. It does not have a built-in fencing or remediation system and so must be configured with an external system that provides these features. By default, it is configured to utilize the Self Node Remediation system.

1.6. NODE MAINTENANCE

Administrators face situations where they need to interrupt the cluster, for example, replace a drive, RAM, or a NIC.

In advance of this maintenance, affected nodes should be cordoned and drained. When a node is cordoned, new workloads cannot be scheduled on that node. When a node is drained, to avoid or minimize downtime, workloads on the affected node are transferred to other nodes.

While this maintenance can be achieved using command line tools, the Node Maintenance Operator offers a declarative approach to achieve this by using a custom resource. When such a resource exists for a node, the Operator cordons and drains the node until the resource is deleted.

CHAPTER 2. USING SELF NODE REMEDIATION

You can use the Self Node Remediation Operator to automatically reboot unhealthy nodes. This remediation strategy minimizes downtime for stateful applications and ReadWriteOnce (RWO) volumes, and restores compute capacity if transient failures occur.

2.1. ABOUT THE SELF NODE REMEDIATION OPERATOR

The Self Node Remediation Operator runs on the cluster nodes and reboots nodes that are identified as unhealthy. The Operator uses the **MachineHealthCheck** or **NodeHealthCheck** controller to detect the health of a node in the cluster. When a node is identified as unhealthy, the **MachineHealthCheck** or the **NodeHealthCheck** resource creates the **SelfNodeRemediation** custom resource (CR), which triggers the Self Node Remediation Operator.

The **SelfNodeRemediation** CR resembles the following YAML file:

```
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) Specifies the remediation strategy for the nodes.

(2) Displays the last error that occurred during remediation. When remediation succeeds or if no errors occur, the field is left empty.

The Self Node Remediation Operator minimizes downtime for stateful applications and restores compute capacity if transient failures occur. You can use this Operator regardless of the management interface, such as IPMI or an API to provision a node, and regardless of the cluster installation type, such as installer-provisioned infrastructure or user-provisioned infrastructure.

2.1.1. About watchdog devices

Watchdog devices can be any of the following:

- Independently powered hardware devices
- Hardware devices that share power with the hosts they control
- Virtual devices implemented in software, or **softdog**

Hardware watchdog and **softdog** devices have electronic or software timers, respectively. These watchdog devices are used to ensure that the machine enters a safe state when an error condition is detected. The cluster is required to repeatedly reset the watchdog timer to prove that it is in a healthy state. This timer might elapse due to fault conditions, such as deadlocks, CPU starvation, and loss of network or disk access. If the timer expires, the watchdog device assumes that a fault has occurred and the device triggers a forced reset of the node.

Hardware watchdog devices are more reliable than **softdog** devices.

2.1.1.1. Understanding Self Node Remediation Operator behavior with watchdog devices

The Self Node Remediation Operator determines the remediation strategy based on the watchdog devices that are present.

If a hardware watchdog device is configured and available, the Operator uses it for remediation. If a hardware watchdog device is not configured, the Operator enables and uses a **softdog** device for remediation.

If neither watchdog devices are supported, either by the system or by the configuration, the Operator remediates nodes by using software reboot.

Additional resources

Configuring a watchdog device for the virtual machine

2.2. CONTROL PLANE FENCING

In earlier releases, you could enable Self Node Remediation and Node Health Check on worker nodes. In the event of node failure, you can now also follow remediation strategies on control plane nodes.

Self Node Remediation occurs in two primary scenarios.

- API Server Connectivity
    - In this scenario, the control plane node to be remediated is not isolated. It can be directly connected to the API Server, or it can be indirectly connected to the API Server through worker nodes or control-plane nodes, that are directly connected to the API Server.
    - When there is API Server Connectivity, the control plane node is remediated only if the Node Health Check Operator has created a **SelfNodeRemediation** custom resource (CR) for the node.
- No API Server Connectivity
    - In this scenario, the control plane node to be remediated is isolated from the API Server. The node cannot connect directly or indirectly to the API Server.
    - When there is no API Server Connectivity, the control plane node will be remediated as outlined with these steps:
        - Check the status of the control plane node with the majority of the peer worker nodes. If the majority of the peer worker nodes cannot be reached, the node will be analyzed further.
            - Self-diagnose the status of the control plane node
                - If self diagnostics passed, no action will be taken.
                - If self diagnostics failed, the node will be fenced and remediated.
                - The self diagnostics currently supported are checking the **kubelet** service status, and checking endpoint availability using **opt in** configuration.
        - If the node did not manage to communicate to most of its worker peers, check the connectivity of the control plane node with other control plane nodes. If the node can communicate with any other control plane peer, no action will be taken. Otherwise, the

node will be fenced and remediated.

2.3. INSTALLING THE SELF NODE REMEDIATION OPERATOR BY USING THE WEB CONSOLE

You can use the Red Hat OpenShift web console to install the Self Node Remediation Operator.

NOTE

The Node Health Check Operator also installs the Self Node Remediation Operator as a default remediation provider.

Prerequisites

- Log in as a user with **cluster-admin** privileges.

Procedure

1. In the Red Hat OpenShift web console, navigate to **Operators** → **OperatorHub**.
2. Select the Self Node Remediation Operator from the list of available Operators, and then click **Install**.
3. Keep the default selection of **Installation mode** and **namespace** to ensure that the Operator is installed to the **openshift-workload-availability** namespace.
4. Click **Install**.

Verification

To confirm that the installation is successful:

1. Navigate to the **Operators** → **Installed Operators** page.
2. Check that the Operator is installed in the **openshift-workload-availability** namespace and its status is **Succeeded**.

If the Operator is not installed successfully:

1. Navigate to the **Operators** → **Installed Operators** page and inspect the **Status** column for any errors or failures.
2. Navigate to the **Workloads** → **Pods** page and check the logs of the **self-node-remediation-controller-manager** pod and **self-node-remediation-ds** pods in the **openshift-workload-availability** project for any reported issues.

2.4. INSTALLING THE SELF NODE REMEDIATION OPERATOR BY USING THE CLI

You can use the OpenShift CLI (**oc**) to install the Self Node Remediation Operator.

You can install the Self Node Remediation Operator in your own namespace or in the **openshift-workload-availability** namespace.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **Namespace** custom resource (CR) for the Self Node Remediation Operator:

   a. Define the **Namespace** CR and save the YAML file, for example, **workload-availability-namespace.yaml**:

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

   b. To create the **Namespace** CR, run the following command:

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

2. Create an **OperatorGroup** CR:

   a. Define the **OperatorGroup** CR and save the YAML file, for example, **workload-availability-operator-group.yaml**:

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

   b. To create the **OperatorGroup** CR, run the following command:

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

3. Create a **Subscription** CR:

   a. Define the **Subscription** CR and save the YAML file, for example, **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. Specify the **Namespace** where you want to install the Self Node Remediation Operator. To install the Self Node Remediation Operator in the **openshift-workload-availability** namespace, specify **openshift-workload-availability** in the **Subscription** CR.

2. Set the approval strategy to Manual in case your specified version is superseded by a later version in the catalog. This plan prevents an automatic upgrade to a later version and requires manual approval before the starting CSV can complete the installation.

b. To create the **Subscription** CR, run the following command:

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

Verification

1. Verify that the installation succeeded by inspecting the CSV resource:

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

Example output

```
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. Verify that the Self Node Remediation Operator is up and running:

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

Example output

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

3. Verify that the Self Node Remediation Operator created the **SelfNodeRemediationConfig** CR:

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

Example output

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

4. Verify that each self node remediation pod is scheduled and running on each worker node and control plane node:

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

Example output

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

2.5. CONFIGURING THE SELF NODE REMEDIATION OPERATOR

The Self Node Remediation Operator creates the **SelfNodeRemediationConfig** CR and the **SelfNodeRemediationTemplate** Custom Resource Definition (CRD).

NOTE

To avoid unexpected reboots of a specific node, the Node Maintenance Operator places the node in maintenance mode and automatically adds a node selector that prevents the SNR daemonset from running on the specific node.

2.5.1. Understanding the Self Node Remediation Operator configuration

The Self Node Remediation Operator creates the **SelfNodeRemediationConfig** CR with the name **self-node-remediation-config**. The CR is created in the namespace of the Self Node Remediation Operator.

A change in the **SelfNodeRemediationConfig** CR re-creates the Self Node Remediation daemon set.

The **SelfNodeRemediationConfig** CR resembles the following YAML file:

```
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 Specify the time duration that the Operator waits before recovering affected workloads running on an unhealthy node. Starting replacement pods while they are still running on the failed node can lead to data corruption and a violation of run-once semantics. The time duration must be equal to

or greater than the minimum value calculated by the Operator using the values in the **ApiServerTimeout**, **ApiCheckInterval**, **maxApiErrorThreshold**, **peerDialTimeout**, and **peerRequestTimeout** fields. In the logs, you can reference the **calculated minTimeToAssumeNodeRebooted is: [value]** value to see the minimum value calculated by the Operator. Specifying a value that is lower than the minimum value calculated prevents the Operator from functioning.

2 Specify the file path of the watchdog device in the nodes. If you enter an incorrect path to the watchdog device, the Self Node Remediation Operator automatically detects the softdog device path.

If a watchdog device is unavailable, the **SelfNodeRemediationConfig** CR uses a software reboot.

3 Specify if you want to enable software reboot of the unhealthy nodes. By default, the value of **isSoftwareRebootEnabled** is set to **true**. To disable the software reboot, set the parameter value to **false**.

4 Specify the timeout duration to check connectivity with each API server. When this duration elapses, the Operator starts remediation. The timeout duration must be greater than or equal to 10 milliseconds.

5 Specify the frequency to check connectivity with each API server. The timeout duration must be greater than or equal to 1 second.

6 Specify a threshold value. After reaching this threshold, the node starts contacting its peers. The threshold value must be greater than or equal to 1 second.

7 Specify the duration of the timeout for the peer to connect the API server. The timeout duration must be greater than or equal to 10 milliseconds.

8 Specify the duration of the timeout for establishing connection with the peer. The timeout duration must be greater than or equal to 10 milliseconds.

9 Specify the duration of the timeout to get a response from the peer. The timeout duration must be greater than or equal to 10 milliseconds.

10 Specify the frequency to update peer information such as IP address. The timeout duration must be greater than or equal to 10 seconds.

11 Specify an optional value to change the port that Self Node Remediation agents use for internal communication. The value must be greater than 0. The default value is port 30001.

12 Specify custom toleration Self Node Remediation agents that are running on the DaemonSets to support remediation for different types of nodes. You can configure the following fields:

- **effect**: The effect indicates the taint effect to match. If this field is empty, all taint effects are matched. When specified, allowed values are **NoSchedule**, **PreferNoSchedule** and **NoExecute**.

- **key**: The key is the taint key that the toleration applies to. If this field is empty, all taint keys are matched. If the key is empty, the **operator** field must be **Exists**. This combination means to match all values and all keys.

- **operator**: The operator represents a key's relationship to the value. Valid operators are **Exists** and **Equal**. The default is **Equal**. **Exists** is equivalent to a wildcard for a value, so that a pod can tolerate all taints of a particular category.

- **value**: The taint value the toleration matches to. If the operator is **Exists**, the value should be empty, otherwise it is just a regular string.
- **tolerationSeconds**: The period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (that is, do not evict). Zero and negative values will be treated as 0 (that is evict immediately) by the system.
- Custom toleration allows you to add a toleration to the Self Node Remediation agent pod. For more information, see Using tolerations to control OpenShift Logging pod placement .

NOTE

You can edit the **self-node-remediation-config** CR that is created by the Self Node Remediation Operator. However, when you try to create a new CR for the Self Node Remediation Operator, the following message is displayed in the logs:

```
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. Understanding the Self Node Remediation Template configuration

The Self Node Remediation Operator also creates the **SelfNodeRemediationTemplate** Custom Resource Definition (CRD). This CRD defines the remediation strategy for the nodes. The following remediation strategies are available:

Automatic

This remediation strategy simplifies the remediation process by letting the Self Node Remediation Operator decide on the most suitable remediation strategy for the cluster. This strategy checks if the **OutOfServiceTaint** strategy is available on the cluster. If the **OutOfServiceTaint** strategy is available, the Operator selects the **OutOfServiceTaint** strategy. If the **OutOfServiceTaint** strategy is not available, the Operator selects the **ResourceDeletion** strategy. **Automatic** is the default remediation strategy.

ResourceDeletion

This remediation strategy removes the pods on the node, rather than the removal of the node object. This strategy recovers workloads faster.

OutOfServiceTaint

This remediation strategy implicitly causes the removal of the pods and associated volume attachments on the node, rather than the removal of the node object. It achieves this by placing the **OutOfServiceTaint** strategy on the node. This strategy recovers workloads faster. This strategy has been supported on technology preview since OpenShift Container Platform version 4.13, and on general availability since OpenShift Container Platform version 4.15.

The Self Node Remediation Operator creates the **SelfNodeRemediationTemplate** CR for the strategy **self-node-remediation-resource-deletion-template**, which the **ResourceDeletion** remediation strategy uses.

The **SelfNodeRemediationTemplate** CR resembles the following YAML file:

–

```
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. Specifies the type of remediation template based on the remediation strategy. Replace **<remediation_object>** with either **resource** or **node**; for example, **self-node-remediation-resource-deletion-template**.

2. Specifies the remediation strategy. The default remediation strategy is **Automatic**.

2.5.3. Troubleshooting the Self Node Remediation Operator

2.5.3.1. General troubleshooting

Issue

You want to troubleshoot issues with the Self Node Remediation Operator.

Resolution

Check the Operator logs.

2.5.3.2. Checking the daemon set

Issue

The Self Node Remediation Operator is installed but the daemon set is not available.

Resolution

Check the Operator logs for errors or warnings.

2.5.3.3. Unsuccessful remediation

Issue

An unhealthy node was not remediated.

Resolution

Verify that the **SelfNodeRemediation** CR was created by running the following command:

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

If the **MachineHealthCheck** controller did not create the **SelfNodeRemediation** CR when the node turned unhealthy, check the logs of the **MachineHealthCheck** controller. Additionally, ensure that the **MachineHealthCheck** CR includes the required specification to use the remediation template.

If the **SelfNodeRemediation** CR was created, ensure that its name matches the unhealthy node or the machine object.

2.5.3.4. Daemon set and other Self Node Remediation Operator resources exist even after uninstalling the Operator

Issue

The Self Node Remediation Operator resources, such as the daemon set, configuration CR, and the remediation template CR, exist even after after uninstalling the Operator.

Resolution

To remove the Self Node Remediation Operator resources, delete the resources by running the following commands for each resource type:

```
$ 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. Gathering data about the Self Node Remediation Operator

To collect debugging information about the Self Node Remediation Operator, use the **must-gather** tool. For information about the **must-gather** image for the Self Node Remediation Operator, see Gathering data about specific features.

2.5.5. Additional resources

- Using Operator Lifecycle Manager on restricted networks .
- Deleting Operators from a cluster

CHAPTER 3. USING FENCE AGENTS REMEDIATION

You can use the Fence Agents Remediation Operator to automatically remediate unhealthy nodes, similar to the Self Node Remediation Operator. Using a management interface or traditional API, this Operator runs a fence-agent to remediate a node from an unhealthy state by power-cycling the node.

3.1. ABOUT THE FENCE AGENTS REMEDIATION OPERATOR

The Fence Agents Remediation (FAR) Operator uses external tools to *fence* unhealthy nodes. These tools are a set of fence agents, where each fence agent can be used for different environments to fence a node, and using a traditional Application Programming Interface (API) call that reboots a node. By doing so, FAR can minimize downtime for stateful applications, restores compute capacity if transient failures occur, and increases the availability of workloads.

FAR not only fences a node when it becomes unhealthy, it also tries to *remediate* the node from being unhealthy to healthy. It adds a taint to evict stateless pods, fences the node with a fence agent, and after a reboot, it completes the remediation with resource deletion to remove any remaining workloads (mostly stateful workloads). Adding the taint and deleting the workloads accelerates the workload rescheduling.

The Operator watches for new or deleted custom resources (CRs) called **FenceAgentsRemediation** which trigger a fence agent to remediate a node, based on the CR's name. FAR uses the **NodeHealthCheck** controller to detect the health of a node in the cluster. When a node is identified as unhealthy, the **NodeHealthCheck** resource creates the **FenceAgentsRemediation** CR, based on the **FenceAgentsRemediationTemplate** CR, which then triggers the Fence Agents Remediation Operator.

FAR uses a fence agent to fence a Kubernetes node. Generally, fencing is the process of taking unresponsive/unhealthy computers into a safe state, and isolating the computer. Fence agent is a software code that uses a management interface to perform fencing, mostly power-based fencing which enables power-cycling, reset, or turning off the computer. An example fence agent is **fence_ipmilan** which is used for Intelligent Platform Management Interface (IPMI) environments.

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

(1) The node-name should match the name of the unhealthy cluster node.

The Operator includes a set of fence agents, that are also available in the Red Hat High Availability Add-On, which use a management interface, such as IPMI or an API, to provision/reboot a node for bare metal servers, virtual machines, and cloud platforms.

3.2. INSTALLING THE FENCE AGENTS REMEDIATION OPERATOR BY USING THE WEB CONSOLE

You can use the Red Hat OpenShift web console to install the Fence Agents Remediation Operator.

Prerequisites

- Log in as a user with **cluster-admin** privileges.

Procedure

1. In the Red Hat OpenShift web console, navigate to **Operators** → **OperatorHub**.
2. Select the Fence Agents Remediation Operator, or FAR, from the list of available Operators, and then click **Install**.
3. Keep the default selection of **Installation mode** and **namespace** to ensure that the Operator is installed to the **openshift-workload-availability** namespace.
4. Click **Install**.

Verification

To confirm that the installation is successful:

1. Navigate to the **Operators** → **Installed Operators** page.
2. Check that the Operator is installed in the **openshift-workload-availability** namespace and its status is **Succeeded**.

If the Operator is not installed successfully:

1. Navigate to the **Operators** → **Installed Operators** page and inspect the **Status** column for any errors or failures.
2. Navigate to the **Workloads** → **Pods** page and check the log of the **fence-agents-remediation-controller-manager** pod for any reported issues.

3.3. INSTALLING THE FENCE AGENTS REMEDIATION OPERATOR BY USING THE CLI

You can use the OpenShift CLI (**oc**) to install the Fence Agents Remediation Operator.

You can install the Fence Agents Remediation Operator in your own namespace or in the **openshift-workload-availability** namespace.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **Namespace** custom resource (CR) for the Fence Agents Remediation Operator:
    a. Define the **Namespace** CR and save the YAML file, for example, **workload-availability-namespace.yaml**:

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

    b. To create the **Namespace** CR, run the following command:

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

2. Create an **OperatorGroup** CR:

   a. Define the **OperatorGroup** CR and save the YAML file, for example, **workload-availability-operator-group.yaml**:

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

   b. To create the **OperatorGroup** CR, run the following command:

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

3. Create a **Subscription** CR:

   a. Define the **Subscription** CR and save the YAML file, for example, **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) Specify the **Namespace** where you want to install the Fence Agents Remediation Operator, for example, the **openshift-workload-availability** outlined earlier in this procedure. You can install the **Subscription** CR for the Fence Agents Remediation Operator in the **openshift-workload-availability** namespace where there is already a matching **OperatorGroup** CR.

   b. To create the **Subscription** CR, run the following command:

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

Verification

1. Verify that the installation succeeded by inspecting the CSV resource:

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

   #### Example output

```
NAME                          DISPLAY                         VERSION   REPLACES   PHASE
fence-agents-remediation.v0.3.0      Fence Agents Remediation Operator   0.3.0    fence-
agents-remediation.v0.2.1          Succeeded
```

2. Verify that the Fence Agents Remediation Operator is up and running:

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

Example output

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

3.4. CONFIGURING THE FENCE AGENTS REMEDIATION OPERATOR

You can use the Fence Agents Remediation Operator to create the **FenceAgentsRemediationTemplate** Custom Resource (CR), which is used by the Node Health Check Operator (NHC). This CR defines the fence agent to be used in the cluster with all the required parameters for remediating the nodes. There may be many **FenceAgentsRemediationTemplate** CRs, at most one for each fence agent, and when NHC is being used it can choose the **FenceAgentsRemediationTemplate** as the remediationTemplate to be used for power-cycling the node.

NOTE

In the current release, there might be many **FenceAgentsRemediationTemplate** CRs, but at most one for each fence agent. This is a known limitation that will be addressed in a future release.

The **FenceAgentsRemediationTemplate** CR resembles the following YAML file:

```
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. Displays the name of the fence agent to be executed, for example, **fence_ipmilan**.
2. Displays the node-specific parameters for executing the fence agent, for example, **ipport**.
3. Displays the cluster-wide parameters for executing the fence agent, for example, **username**.
4. Displays the number of times to retry the fence agent command in case of failure. The default number of attempts is 5.
5. Displays the interval between retries in seconds. The default is 5 seconds.
6. Displays the timeout for the fence agent command in seconds. The default is 60 seconds.

3.5. TROUBLESHOOTING THE FENCE AGENTS REMEDIATION OPERATOR

3.5.1. General troubleshooting

Issue

You want to troubleshoot issues with the Fence Agents Remediation Operator.

Resolution

Check the Operator logs.

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

3.5.2. Unsuccessful remediation

Issue

An unhealthy node was not remediated.

Resolution

Verify that the **FenceAgentsRemediation** CR was created by running the following command:

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

If the **NodeHealthCheck** controller did not create the **FenceAgentsRemediation** CR when the node turned unhealthy, check the logs of the **NodeHealthCheck** controller. Additionally, ensure that the **NodeHealthCheck** CR includes the required specification to use the remediation template.

If the **FenceAgentsRemediation** CR was created, ensure that its name matches the unhealthy node object.

3.5.3. Fence Agents Remediation Operator resources exist after uninstalling the Operator

Issue

The Fence Agents Remediation Operator resources, such as the remediation CR and the remediation template CR, exist after uninstalling the Operator.

Resolution

To remove the Fence Agents Remediation Operator resources, you can delete the resources by selecting the "Delete all operand instances for this operator" checkbox before uninstalling. This checkbox feature is only available in Red Hat OpenShift since version 4.13. For all versions of Red Hat OpenShift, you can delete the resources by running the following relevant command for each resource type:

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

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

The remediation CR **far** must be created and deleted by the same entity, for example, NHC. If the remediation CR **far** is still present, it is deleted, together with the FAR operator.

The remediation template CR **fartemplate** only exists if you use FAR with NHC. When the FAR operator is deleted using the web console, the remediation template CR **fartemplate** is also deleted.

3.6. GATHERING DATA ABOUT THE FENCE AGENTS REMEDIATION OPERATOR

To collect debugging information about the Fence Agents Remediation Operator, use the **must-gather** tool. For information about the **must-gather** image for the Fence Agents Remediation Operator, see Gathering data about specific features .

3.7. ADDITIONAL RESOURCES

- Using Operator Lifecycle Manager on restricted networks .
- Deleting Operators from a cluster

CHAPTER 4. USING MACHINE DELETION REMEDIATION

You can use the Machine Deletion Remediation Operator to reprovision unhealthy nodes using the Machine API. You can use the Machine Deletion Remediation Operator in conjunction with the Node Health Check Operator.

4.1. ABOUT THE MACHINE DELETION REMEDIATION OPERATOR

The Machine Deletion Remediation (MDR) operator works with the **NodeHealthCheck** controller, to reprovision unhealthy nodes using the Machine API. MDR follows the annotation on the node to the associated machine object, confirms that it has an owning controller (for example, **MachineSetController**), and deletes it. Once the machine CR is deleted, the owning controller creates a replacement.

The prerequisites for MDR include:

- a Machine API-based cluster that is able to programmatically destroy and create cluster nodes,
- nodes that are associated with machines, and
- declaratively managed machines.

You can then modify the **NodeHealthCheck** CR to use MDR as its remediator. An example MDR template object and **NodeHealthCheck** configuration are provided in the documentation.

The MDR process works as follows:

- the Node Health Check Operator detects an unhealthy node and creates a MDR CR.
- the MDR Operator watches for the MDR CR associated with the unhealthy node and deletes it, if the machine has an owning controller.
- when the node is healthy again, the MDR CR is deleted by the **NodeHealthCheck** controller.

4.2. INSTALLING THE MACHINE DELETION REMEDIATION OPERATOR BY USING THE WEB CONSOLE

You can use the Red Hat OpenShift web console to install the Machine Deletion Remediation Operator.

Prerequisites

- Log in as a user with **cluster-admin** privileges.

Procedure

1. In the Red Hat OpenShift web console, navigate to **Operators → OperatorHub**.
2. Select the Machine Deletion Remediation Operator, or MDR, from the list of available Operators, and then click **Install**.
3. Keep the default selection of **Installation mode** and **namespace** to ensure that the Operator is installed to the **openshift-workload-availability** namespace.
4. Click **Install**.

Verification

To confirm that the installation is successful:

1. Navigate to the **Operators → Installed Operators** page.
2. Check that the Operator is installed in the **openshift-workload-availability** namespace and its status is **Succeeded**.

If the Operator is not installed successfully:

1. Navigate to the **Operators → Installed Operators** page and inspect the **Status** column for any errors or failures.
2. Navigate to the **Workloads → Pods** page and check the log of the pod in the **openshift-workload-availability** project for any reported issues.

4.3. INSTALLING THE MACHINE DELETION REMEDIATION OPERATOR BY USING THE CLI

You can use the OpenShift CLI (**oc**) to install the Machine Deletion Remediation Operator.

You can install the Machine Deletion Remediation Operator in your own namespace or in the **openshift-workload-availability** namespace.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **Namespace** custom resource (CR) for the Machine Deletion Remediation Operator:
   a. Define the **Namespace** CR and save the YAML file, for example, **workload-availability-namespace.yaml**:

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

   b. To create the **Namespace** CR, run the following command:

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

2. Create an **OperatorGroup** CR:
   a. Define the **OperatorGroup** CR and save the YAML file, for example, **workload-availability-operator-group.yaml**:

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

b. To create the **OperatorGroup** CR, run the following command:

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

3. Create a **Subscription** CR:

a. Define the **Subscription** CR and save the YAML file, for example, **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 Specify the **Namespace** where you want to install the Machine Deletion Remediation Operator. When installing the Machine Deletion Remediation Operator in the **openshift-workload-availability Subscription** CR, the **Namespace** and **OperatorGroup** CRs will already exist.

b. To create the **Subscription** CR, run the following command:

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

Verification

1. Verify that the installation succeeded by inspecting the CSV resource:

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

Example output

```
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. CONFIGURING THE MACHINE DELETION REMEDIATION OPERATOR

You can use the Machine Deletion Remediation Operator, with the Node Health Check Operator, to create the **MachineDeletionRemediationTemplate** Custom Resource (CR). This CR defines the remediation strategy for the nodes.

The **MachineDeletionRemediationTemplate** CR resembles the following YAML file:

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

4.5. TROUBLESHOOTING THE MACHINE DELETION REMEDIATION OPERATOR

4.5.1. General troubleshooting

Issue

You want to troubleshoot issues with the Machine Deletion Remediation Operator.

Resolution

Check the Operator logs.

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

4.5.2. Unsuccessful remediation

Issue

An unhealthy node was not remediated.

Resolution

Verify that the **MachineDeletionRemediation** CR was created by running the following command:

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

If the **NodeHealthCheck** controller did not create the **MachineDeletionRemediation** CR when the node turned unhealthy, check the logs of the **NodeHealthCheck** controller. Additionally, ensure that the **NodeHealthCheck** CR includes the required specification to use the remediation template.

If the **MachineDeletionRemediation** CR was created, ensure that its name matches the unhealthy node object.

4.5.3. Machine Deletion Remediation Operator resources exist even after uninstalling the Operator

Issue

The Machine Deletion Remediation Operator resources, such as the remediation CR and the remediation template CR, exist even after uninstalling the Operator.

Resolution

To remove the Machine Deletion Remediation Operator resources, you can delete the resources by selecting the **Delete all operand instances for this operator**checkbox before uninstalling. This checkbox feature is only available in Red Hat OpenShift since version 4.13. For all versions of Red Hat OpenShift, you can delete the resources by running the following relevant command for each resource type:

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

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

The remediation CR **mdr** must be created and deleted by the same entity, for example, NHC. If the remediation CR **mdr** is still present, it is deleted, together with the MDR operator.

The remediation template CR **mdrt** only exists if you use MDR with NHC. When the MDR operator is deleted using the web console, the remediation template CR **mdrt** is also deleted.

4.6. GATHERING DATA ABOUT THE MACHINE DELETION REMEDIATION OPERATOR

To collect debugging information about the Machine Deletion Remediation Operator, use the **must-gather** tool. For information about the **must-gather** image for the Machine Deletion Remediation Operator, see Gathering data about specific features .

4.7. ADDITIONAL RESOURCES

- Using Operator Lifecycle Manager on restricted networks .
- Deleting Operators from a cluster

CHAPTER 5. REMEDIATING NODES WITH MACHINE HEALTH CHECKS

Machine health checks automatically repair unhealthy machines in a particular machine pool.

5.1. ABOUT MACHINE HEALTH CHECKS

NOTE

You can only apply a machine health check to control plane machines on clusters that use control plane machine sets.

To monitor machine health, create a resource to define the configuration for a controller. Set a condition to check, such as staying in the **NotReady** status for five minutes or displaying a permanent condition in the node-problem-detector, and a label for the set of machines to monitor.

The controller that observes a **MachineHealthCheck** resource checks for the defined condition. If a machine fails the health check, the machine is automatically deleted and one is created to take its place. When a machine is deleted, you see a **machine deleted** event.

To limit disruptive impact of the machine deletion, the controller drains and deletes only one node at a time. If there are more unhealthy machines than the **maxUnhealthy** threshold allows for in the targeted pool of machines, remediation stops and therefore enables manual intervention.

NOTE

Consider the timeouts carefully, accounting for workloads and requirements.

- Long timeouts can result in long periods of downtime for the workload on the unhealthy machine.
- Too short timeouts can result in a remediation loop. For example, the timeout for checking the **NotReady** status must be long enough to allow the machine to complete the startup process.

To stop the check, remove the resource.

5.1.1. Limitations when deploying machine health checks

There are limitations to consider before deploying a machine health check:

- Only machines owned by a machine set are remediated by a machine health check.
- If the node for a machine is removed from the cluster, a machine health check considers the machine to be unhealthy and remediates it immediately.
- If the corresponding node for a machine does not join the cluster after the **nodeStartupTimeout**, the machine is remediated.
- A machine is remediated immediately if the **Machine** resource phase is **Failed**.

5.2. CONFIGURING MACHINE HEALTH CHECKS TO USE THE SELF NODE REMEDIATION OPERATOR

Use the following procedure to configure the worker or control-plane machine health checks to use the Self Node Remediation Operator as a remediation provider.

NOTE

To use the Self Node Remediation Operator as a remediation provider for machine health checks, a machine must have an associated node in the cluster.

Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **SelfNodeRemediationTemplate** CR:

   a. Define the **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) Specifies the remediation strategy. The default remediation strategy is **Automatic**.

   b. To create the **SelfNodeRemediationTemplate** CR, run the following command:

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

2. Create or update the **MachineHealthCheck** CR to point to the **SelfNodeRemediationTemplate** CR:

   a. Define or update the **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. Selects whether the machine health check is for **worker** or **control-plane** nodes. The label can also be user-defined.
2. Specifies the details for the remediation template.

b. To create a **MachineHealthCheck** CR, run the following command:

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

c. To update a **MachineHealthCheck** CR, run the following command:

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

CHAPTER 6. REMEDIATING NODES WITH NODE HEALTH CHECKS

You can use the Node Health Check Operator to identify unhealthy nodes. The Operator uses the Self Node Remediation Operator to remediate the unhealthy nodes.

For more information on the Self Node Remediation Operator, see the Using Self Node Remediation chapter.

NOTE

Due to the existence of preinstalled machine health checks on Red Hat OpenShift Service on AWS (ROSA) clusters, the Node Health Check Operator is unable to function in such an environment.

6.1. ABOUT THE NODE HEALTH CHECK OPERATOR

The Node Health Check Operator detects the health of the nodes in a cluster. The **NodeHealthCheck** controller creates the **NodeHealthCheck** custom resource (CR), which defines a set of criteria and thresholds to determine the health of a node.

The Node Health Check Operator also installs the Self Node Remediation Operator as a default remediation provider.

When the Node Health Check Operator detects an unhealthy node, it creates a remediation CR that triggers the remediation provider. For example, the controller creates the **SelfNodeRemediation** CR, which triggers the Self Node Remediation Operator to remediate the unhealthy node.

The **NodeHealthCheck** CR resembles the following YAML file, with self-node-remediation as the remediation provider:

```
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 Specifies the amount of healthy nodes(in percentage or number) required for a remediation provider to concurrently remediate nodes in the targeted pool. If the number of healthy nodes equals to or exceeds the limit set by **minHealthy**, remediation occurs. The default value is 51%.

2 Prevents any new remediation from starting, while allowing any ongoing remediations to persist. The default value is empty. However, you can enter an array of strings that identify the cause of pausing the remediation. For example, **pause-test-cluster**.

NOTE

During the upgrade process, nodes in the cluster might become temporarily unavailable and get identified as unhealthy. In the case of worker nodes, when the Operator detects that the cluster is upgrading, it stops remediating new unhealthy nodes to prevent such nodes from rebooting.

3 Specifies a remediation template from the remediation provider. For example, from the Self Node Remediation Operator. **remediationTemplate** is mutually exclusive with **escalatingRemediations**.

4 Specifies a list of **RemediationTemplates** with order and timeout fields. To obtain a healthy node, use this field to sequence and configure multiple remediations. This strategy increases the likelihood of obtaining a healthy node, instead of depending on a single remediation that might not be successful. The **order** field determines the order in which the remediations are invoked (lower order = earlier invocation). The **timeout** field determines when the next remediation is invoked. **escalatingRemediations** is mutually exclusive with **remediationTemplate**.

5 Specifies a selector that matches labels or expressions that you want to check. Avoid selecting both control-plane and worker nodes in one CR.

6 Specifies a list of the conditions that determine whether a node is considered unhealthy.

7 8 Specifies the timeout duration for a node condition. If a condition is met for the duration of the timeout, the node will be remediated. Long timeouts can result in long periods of downtime for a workload on an unhealthy node.

The **NodeHealthCheck** CR resembles the following YAML file, with metal3 as the remediation provider:

```
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
```

NOTE

The **matchExpressions** are examples only; you must map your machine groups based on your specific needs.

The **Metal3RemediationTemplate** resembles the following YAML file, with metal3 as the remediation provider:

```
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
```

NOTE

In addition to creating a **NodeHealthCheck** CR, you must also create the **Metal3RemediationTemplate**.

6.1.1. Understanding the Node Health Check Operator workflow

When a node is identified as unhealthy, the Node Health Check Operator checks how many other nodes are unhealthy. If the number of healthy nodes exceeds the amount that is specified in the **minHealthy** field of the **NodeHealthCheck** CR, the controller creates a remediation CR from the details that are provided in the external remediation template by the remediation provider. After remediation, the kubelet updates the node's health status.

When the node turns healthy, the controller deletes the external remediation template.

6.1.2. About how node health checks prevent conflicts with machine health checks

When both, node health checks and machine health checks are deployed, the node health check avoids conflict with the machine health check.

> **NOTE**
>
> Red Hat OpenShift deploys **machine-api-termination-handler** as the default **MachineHealthCheck** resource.

The following list summarizes the system behavior when node health checks and machine health checks are deployed:

- If only the default machine health check exists, the node health check continues to identify unhealthy nodes. However, the node health check ignores unhealthy nodes in a Terminating state. The default machine health check handles the unhealthy nodes with a Terminating state.

  Example log message

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

- If the default machine health check is modified (for example, the **unhealthyConditions** is **Ready**), or if additional machine health checks are created, the node health check is disabled.

  Example log message

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

- When, again, only the default machine health check exists, the node health check is re-enabled.

  Example log message

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

6.2. CONTROL PLANE FENCING

In earlier releases, you could enable Self Node Remediation and Node Health Check on worker nodes. In the event of node failure, you can now also follow remediation strategies on control plane nodes.

Do not use the same **NodeHealthCheck** CR for worker nodes and control plane nodes. Grouping worker nodes and control plane nodes together can result in incorrect evaluation of the minimum healthy node count, and cause unexpected or missing remediations. This is because of the way the Node Health Check Operator handles control plane nodes. You should group the control plane nodes in their own group and the worker nodes in their own group. If required, you can also create multiple groups of worker nodes.

Considerations for remediation strategies:

- Avoid Node Health Check configurations that involve multiple configurations overlapping the same nodes because they can result in unexpected behavior. This suggestion applies to both worker and control plane nodes.

- The Node Health Check Operator implements a hardcoded limitation of remediating a maximum of one control plane node at a time. Multiple control plane nodes should not be remediated at the same time.

6.3. INSTALLING THE NODE HEALTH CHECK OPERATOR BY USING THE WEB CONSOLE

You can use the Red Hat OpenShift web console to install the Node Health Check Operator.

Prerequisites

- Log in as a user with **cluster-admin** privileges.

Procedure

1. In the Red Hat OpenShift web console, navigate to **Operators** → **OperatorHub**.
2. Select the Node Health Check Operator, then click **Install**.
3. Keep the default selection of **Installation mode** and **namespace** to ensure that the Operator will be installed to the **openshift-workload-availability** namespace.
4. Ensure that the **Console plug-in** is set to **Enable**.
5. Click **Install**.

Verification

To confirm that the installation is successful:

1. Navigate to the **Operators** → **Installed Operators** page.
2. Check that the Operator is installed in the **openshift-workload-availability** namespace and that its status is **Succeeded**.

If the Operator is not installed successfully:

1. Navigate to the **Operators** → **Installed Operators** page and inspect the **Status** column for any errors or failures.
2. Navigate to the **Workloads** → **Pods** page and check the logs in any pods in the **openshift-workload-availability** project that are reporting issues.

6.4. INSTALLING THE NODE HEALTH CHECK OPERATOR BY USING THE CLI

You can use the OpenShift CLI (**oc**) to install the Node Health Check Operator.

You can install the Node Health Check Operator in your own namespace or in the **openshift-workload-availability** namespace.

Prerequisites

- Install the OpenShift CLI (**oc**).

- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **Namespace** custom resource (CR) for the Node Health Check Operator:

   a. Define the **Namespace** CR and save the YAML file, for example, **node-health-check-namespace.yaml**:

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

   b. To create the **Namespace** CR, run the following command:

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

2. Create an **OperatorGroup** CR:

   a. Define the **OperatorGroup** CR and save the YAML file, for example, **workload-availability-operator-group.yaml**:

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

   b. To create the **OperatorGroup** CR, run the following command:

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

3. Create a **Subscription** CR:

   a. Define the **Subscription** CR and save the YAML file, for example, **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) Specify the **Namespace** where you want to install the Node Health Check Operator. To install the Node Health Check Operator in the **openshift-workload-availability** namespace, specify **openshift-workload-availability** in the **Subscription** CR.

2 Specify the channel name for your subscription. To upgrade to the latest version of the Node Health Check Operator, you must manually change the channel name for

3 Set the approval strategy to Manual in case your specified version is superseded by a later version in the catalog. This plan prevents an automatic upgrade to a later version and requires manual approval before the starting CSV can complete the installation.

b. To create the **Subscription** CR, run the following command:

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

Verification

1. Verify that the installation succeeded by inspecting the CSV resource:

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

Example output

```
NAME                              DISPLAY                    VERSION   REPLACES PHASE
node-healthcheck-operator.v0.7.0  Node Health Check Operator  0.7.0    node-healthcheck-
operator.v0.6.1           Succeeded
```

2. Verify that the Node Health Check Operator is up and running:

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

Example output

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

6.5. CREATING A NODE HEALTH CHECK

Using the web console, you can create a node health check to identify unhealthy nodes and specify the remediation type and strategy to fix them.

Procedure

1. From the **Administrator** perspective of the Red Hat OpenShift web console, click **Compute → NodeHealthChecks → CreateNodeHealthCheck**.
2. Specify whether to configure the node health check using the **Form view** or the **YAML view**.
3. Enter a **Name** for the node health check. The name must consist of lower case, alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character.
4. Specify the **Remediator** type, and **Self node remediation** or **Other**. The Self node remediation option is part of the Self Node Remediation Operator that is installed with the Node Health Check Operator. Selecting **Other** requires an **API version**, **Kind**, **Name**, and **Namespace** to be entered, which then points to the remediation template resource of a remediator.

5. Make a **Nodes** selection by specifying the labels of the nodes you want to remediate. The selection matches labels that you want to check. If more than one label is specified, the nodes must contain each label. The default value is empty, which selects both worker and control-plane nodes.

   NOTE

   When creating a node health check with the Self Node Remediation Operator, you must select either **node-role.kubernetes.io/worker** or **node-role.kubernetes.io/control-plane** as the value.

6. Specify the minimum number of healthy nodes, using either a percentage or a number, required for a **NodeHealthCheck** to remediate nodes in the targeted pool. If the number of healthy nodes equals to or exceeds the limit set by **Min healthy**, remediation occurs. The default value is 51%.

7. Specify a list of **Unhealthy conditions** that if a node meets determines whether the node is considered unhealthy, and requires remediation. You can specify the **Type**, **Status** and **Duration**. You can also create your own custom type.

8. Click **Create** to create the node health check.

Verification

- Navigate to the **Compute → NodeHealthCheck** page and verify that the corresponding node health check is listed, and their status displayed. Once created, node health checks can be paused, modified, and deleted.

6.6. GATHERING DATA ABOUT THE NODE HEALTH CHECK OPERATOR

To collect debugging information about the Node Health Check Operator, use the **must-gather** tool. For information about the **must-gather** image for the Node Health Check Operator, see Gathering data about specific features.

6.7. ADDITIONAL RESOURCES

- Changing the update channel for an Operator
- Using Operator Lifecycle Manager on restricted networks .

CHAPTER 7. PLACING NODES IN MAINTENANCE MODE WITH NODE MAINTENANCE OPERATOR

You can use the Node Maintenance Operator to place nodes in maintenance mode by using the **oc adm** utility or **NodeMaintenance** custom resources (CRs).

7.1. ABOUT THE NODE MAINTENANCE OPERATOR

The Node Maintenance Operator watches for new or deleted **NodeMaintenance** CRs. When a new **NodeMaintenance** CR is detected, no new workloads are scheduled and the node is cordoned off from the rest of the cluster. All pods that can be evicted are evicted from the node. When a **NodeMaintenance** CR is deleted, the node that is referenced in the CR is made available for new workloads.

NOTE

Using a **NodeMaintenance** CR for node maintenance tasks achieves the same results as the **oc adm cordon** and **oc adm drain** commands using standard Red Hat OpenShift CR processing.

7.2. INSTALLING THE NODE MAINTENANCE OPERATOR

You can install the Node Maintenance Operator using the web console or the OpenShift CLI (**oc**).

NOTE

If OpenShift Virtualization version 4.10 or less is installed in your cluster, it includes an outdated version of the Node Maintenance Operator.

7.2.1. Installing the Node Maintenance Operator by using the web console

You can use the Red Hat OpenShift web console to install the Node Maintenance Operator.

Prerequisites

- Log in as a user with **cluster-admin** privileges.

Procedure

1. In the Red Hat OpenShift web console, navigate to **Operators** → **OperatorHub**.
2. Select the Node Maintenance Operator, then click **Install**.
3. Keep the default selection of **Installation mode** and **namespace** to ensure that the Operator will be installed to the **openshift-workload-availability** namespace.
4. Click **Install**.

Verification

To confirm that the installation is successful:

1. Navigate to the **Operators** → **Installed Operators** page.

2. Check that the Operator is installed in the **openshift-workload-availability** namespace and that its status is **Succeeded**.

If the Operator is not installed successfully:

1. Navigate to the **Operators → Installed Operators** page and inspect the **Status** column for any errors or failures.

2. Navigate to the **Operators → Installed Operators → Node Maintenance Operator → Details** page, and inspect the **Conditions** section for errors before pod creation.

3. Navigate to the **Workloads → Pods** page, search for the **Node Maintenance Operator** pod in the installed namespace, and check the logs in the **Logs** tab.

7.2.2. Installing the Node Maintenance Operator by using the CLI

You can use the OpenShift CLI (**oc**) to install the Node Maintenance Operator.

You can install the Node Maintenance Operator in your own namespace or in the **openshift-workload-availability** namespace.

Prerequisites

- Install the OpenShift CLI (**oc**).

- Log in as a user with **cluster-admin** privileges.

Procedure

1. Create a **Namespace** CR for the Node Maintenance Operator:

   a. Define the **Namespace** CR and save the YAML file, for example, **workload-availability-namespace.yaml**:

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

   b. To create the **Namespace** CR, run the following command:

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

2. Create an **OperatorGroup** CR:

   a. Define the **OperatorGroup** CR and save the YAML file, for example, **workload-availability-operator-group.yaml**:

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

   b. To create the **OperatorGroup** CR, run the following command:

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

3. Create a **Subscription** CR:

   a. Define the **Subscription** CR and save the YAML file, for example, **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) Specify the **Namespace** where you want to install the Node Maintenance Operator.

   IMPORTANT

   To install the Node Maintenance Operator in the **openshift-workload-availability** namespace, specify **openshift-workload-availability** in the **Subscription** CR.

   b. To create the **Subscription** CR, run the following command:

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

Verification

1. Verify that the installation succeeded by inspecting the CSV resource:

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

   Example output

   ```
   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. Verify that the Node Maintenance Operator is running:

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

   Example output

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

The Node Maintenance Operator is supported in a restricted network environment. For more information, see Using Operator Lifecycle Manager on restricted networks .

7.3. SETTING A NODE TO MAINTENANCE MODE

You can place a node into maintenance mode from the web console or from the CLI by using a **NodeMaintenance** CR.

7.3.1. Setting a node to maintenance mode by using the web console

To set a node to maintenance mode, you can create a **NodeMaintenance** custom resource (CR) by using the web console.

Prerequisites

- Log in as a user with **cluster-admin** privileges.
- Install the Node Maintenance Operator from the **OperatorHub**.

Procedure

1. From the **Administrator** perspective in the web console, navigate to **Operators → Installed Operators**.
2. Select the Node Maintenance Operator from the list of Operators.
3. In the **Node Maintenance** tab, click **Create NodeMaintenance**.
4. In the **Create NodeMaintenance** page, select the **Form view** or the **YAML view** to configure the **NodeMaintenance** CR.
5. To apply the **NodeMaintenance** CR that you have configured, click **Create**.

Verification

In the **Node Maintenance** tab, inspect the **Status** column and verify that its status is **Succeeded**.

7.3.2. Setting a node to maintenance mode by using the CLI

You can put a node into maintenance mode with a **NodeMaintenance** custom resource (CR). When you apply a **NodeMaintenance** CR, all allowed pods are evicted and the node is rendered unschedulable. Evicted pods are queued to be moved to another node in the cluster.

Prerequisites

- Install the Red Hat OpenShift CLI **oc**.
- Log in to the cluster as a user with **cluster-admin** privileges.

Procedure

1. Create the following **NodeMaintenance** CR, and save the file as **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) The name of the node maintenance CR.

   (2) The name of the node to be put into maintenance mode.

   (3) A plain text description of the reason for maintenance.

2. Apply the node maintenance CR by running the following command:

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

Verification

1. Check the progress of the maintenance task by running the following command:

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

   where **<node-name>** is the name of your node; for example, **node-1.example.com**

2. Check the example output:

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

7.3.3. Checking status of current NodeMaintenance CR tasks

You can check the status of current **NodeMaintenance** CR tasks.

Prerequisites

- Install the Red Hat OpenShift CLI **oc**.
- Log in as a user with **cluster-admin** privileges.

Procedure

- Check the status of current node maintenance tasks, for example the **NodeMaintenance** CR or **nm** object, by running the following command:

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

Example output

```
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. The percentage completion of draining the node.
2. The number of pods scheduled for eviction.
3. The latest eviction error, if any.
4. The last time the status was updated.
5. The total number of pods before the node entered maintenance mode.

7.4. RESUMING A NODE FROM MAINTENANCE MODE

You can resume a node from maintenance mode from the web console or from the CLI by using a **NodeMaintenance** CR. Resuming a node brings it out of maintenance mode and makes it schedulable again.

7.4.1. Resuming a node from maintenance mode by using the web console

To resume a node from maintenance mode, you can delete a **NodeMaintenance** custom resource (CR) by using the web console.

Prerequisites

- Log in as a user with **cluster-admin** privileges.
- Install the Node Maintenance Operator from the **OperatorHub**.

Procedure

1. From the **Administrator** perspective in the web console, navigate to **Operators → Installed Operators**.
2. Select the Node Maintenance Operator from the list of Operators.

3. In the **Node Maintenance** tab, select the **NodeMaintenance** CR that you want to delete.

4. Click the Options menu ⋮ at the end of the node and select **Delete NodeMaintenance**.

Verification

1. In the Red Hat OpenShift console, click **Compute → Nodes**.

2. Inspect the **Status** column of the node for which you deleted the **NodeMaintenance** CR and verify that its status is **Ready**.

7.4.2. Resuming a node from maintenance mode by using the CLI

You can resume a node from maintenance mode that was initiated with a **NodeMaintenance** CR by deleting the **NodeMaintenance** CR.

Prerequisites

- Install the Red Hat OpenShift CLI **oc**.
- Log in to the cluster as a user with **cluster-admin** privileges.

Procedure

- When your node maintenance task is complete, delete the active **NodeMaintenance** CR:

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

Example output

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

Verification

1. Check the progress of the maintenance task by running the following command:

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

where **<node-name>** is the name of your node; for example, **node-1.example.com**

2. Check the example output:

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

7.5. WORKING WITH BARE-METAL NODES

For clusters with bare-metal nodes, you can place a node into maintenance mode, and resume a node from maintenance mode, by using the web console **Actions** control.

NOTE

Clusters with bare-metal nodes can also place a node into maintenance mode, and resume a node from maintenance mode, by using the web console and CLI, as outlined. These methods, by using the web console **Actions** control, are applicable to bare-metal clusters only.

7.5.1. Maintaining bare-metal nodes

When you deploy Red Hat OpenShift on bare-metal infrastructure, you must take additional considerations into account compared to deploying on cloud infrastructure. Unlike in cloud environments, where the cluster nodes are considered ephemeral, reprovisioning a bare-metal node requires significantly more time and effort for maintenance tasks.

When a bare-metal node fails due to a kernel error or a NIC card hardware failure, workloads on the failed node need to be restarted on another node in the cluster while the problem node is repaired or replaced. Node maintenance mode allows cluster administrators to gracefully turn-off nodes, move workloads to other parts of the cluster, and ensure that workloads do not get interrupted. Detailed progress and node status details are provided during maintenance.

7.5.2. Setting a bare-metal node to maintenance mode

Set a bare-metal node to maintenance mode using the Options menu ⋮ found on each node in the **Compute → Nodes** list, or using the **Actions** control of the **Node Details** screen.

Procedure

1. From the **Administrator** perspective of the web console, click **Compute → Nodes**.
2. You can set the node to maintenance from this screen, which makes it easier to perform actions on multiple nodes, or from the **Node Details** screen, where you can view comprehensive details of the selected node:
   - Click the Options menu ⋮ at the end of the node and select **Start Maintenance**.
   - Click the node name to open the **Node Details** screen and click **Actions → Start Maintenance**.
3. Click **Start Maintenance** in the confirmation window.

The node is no longer schedulable. If it had virtual machines with the **LiveMigration** eviction strategy, then it will live migrate them. All other pods and virtual machines on the node are deleted and recreated on another node.

Verification

- Navigate to the **Compute → Nodes** page and verify that the corresponding node has a status of **Under maintenance**.

7.5.3. Resuming a bare-metal node from maintenance mode

Resume a bare-metal node from maintenance mode using the Options menu ⋮ found on each node in the **Compute → Nodes** list, or using the **Actions** control of the **Node Details** screen.

Procedure

1. From the **Administrator** perspective of the web console, click **Compute → Nodes**.
2. You can resume the node from this screen, which makes it easier to perform actions on multiple nodes, or from the **Node Details** screen, where you can view comprehensive details of the selected node:
   - Click the Options menu ⋮ at the end of the node and select **Stop Maintenance**.
   - Click the node name to open the **Node Details** screen and click **Actions → Stop Maintenance**.
3. Click **Stop Maintenance** in the confirmation window.

The node becomes schedulable. If it had virtual machine instances that were running on the node prior to maintenance, then they will not automatically migrate back to this node.

Verification

- Navigate to the **Compute → Nodes** page and verify that the corresponding node has a status of **Ready**.

7.6. GATHERING DATA ABOUT THE NODE MAINTENANCE OPERATOR

To collect debugging information about the Node Maintenance Operator, use the **must-gather** tool. For information about the **must-gather** image for the Node Maintenance Operator, see Gathering data about specific features.

7.7. ADDITIONAL RESOURCES

- Gathering data about your cluster
- Understanding how to evacuate pods on nodes
- Understanding how to mark nodes as unschedulable or schedulable