Red Hat Advanced Cluster Management for Kubernetes 2.12

Add-ons

Add-ons

Last Updated: 2024-11-05

Red Hat Advanced Cluster Management for Kubernetes 2.12 Add-ons

Add-ons

Legal Notice

Abstract

Read more to learn how to use add-ons for your cluster.

Table of Contents

CHAPTER 1. MANAGED CLUSTER ADVANCED CONFIGURATION

With Red Hat Advanced Cluster Management for Kubernetes klusterlet add-ons, you can further configure your managed clusters to improve performance and add functionality to your applications. See the following enablement options:

- Enabling klusterlet add-ons on clusters for cluster management
- Configuring nodeSelectors and tolerations for klusterlet add-ons
- Enabling cluster-wide proxy on existing cluster add-ons

1.1. ENABLING KLUSTERLET ADD-ONS ON CLUSTERS FOR CLUSTER MANAGEMENT

After you install Red Hat Advanced Cluster Management for Kubernetes and then create or import clusters with multicluster engine operator you can enable the klusterlet add-ons for those managed clusters. The klusterlet add-ons are not enabled by default if you created or imported clusters unless you create or import with the Red Hat Advanced Cluster Management console. See the following available klusterlet add-ons:

- application-manager
- cert-policy-controller
- config-policy-controller
- iam-policy-controller
- governance-policy-framework
- search-collector

Complete the following steps to enable the klusterlet add-ons for the managed clusters after Red Hat Advanced Cluster Management is installed:

1. Create a YAML file that is similar to the following **KlusterletAddonConfig**, with the **spec** value that represents the add-ons:

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
  iamPolicyController:
    enabled: true
  policyController: 1
```

```
    enabled: true
  searchCollector:
    enabled: true
```

1 The **policy-controller** add-on is divided into two add-ons: The **governance-policy-framework** and the **config-policy-controller**. As a result, the **policyController** controls the **governance-policy-framework** and the **config-policy-controller** **managedClusterAddons**.

2. Save the file as **klusterlet-addon-config.yaml**.
3. Apply the YAML by running the following command on the hub cluster:

```
oc apply -f klusterlet-addon-config.yaml
```

4. To verify whether the enabled **managedClusterAddons** are created after the **KlusterletAddonConfig** is created, run the following command:

```
oc get managedclusteraddons -n <cluster namespace>
```

1.2. CONFIGURING NODESELECTORS AND TOLERATIONS FOR KLUSTERLET ADD-ONS

In Red Hat Advanced Cluster Management, you can configure nodeSelector and tolerations for the following klusterlet add-ons:

- application-manager
- cert-policy-controller
- cluster-proxy
- config-policy-controller
- governance-policy-framework
- hypershift-addon
- iam-policy-controller
- managed-serviceaccount
- observability-controller
- search-collector
- submariner
- volsync
- work-manager

Complete the following steps:

1. Use the **AddonDeploymentConfig** API to create a configuration to specify the **nodeSelector** and **tolerations** on a certain namespace on the hub cluster.

2. Create a file named **addondeploymentconfig.yaml** that is based on the following template:

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: config-name (1)
  namespace: config-name-space (2)
spec:
  nodePlacement:
    nodeSelector: node-selector (3)
    tolerations: tolerations (4)
```

(1) Replace **config-name** with the name of the **AddonDeploymentConfig** that you just created.

(2) Replace **config-namespace** with the namespace of the **AddonDeploymentConfig** that you just created.

(3) Replace **node-selector** with your node selector.

(4) Replace **tolerations** with your tolerations.

A completed **AddOnDeployment** file might resemble the following example:

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: deploy-config
  namespace: open-cluster-management-hub
spec:
  nodePlacement:
    nodeSelector:
      "node-dedicated": "acm-addon"
    tolerations:
      - effect: NoSchedule
        key: node-dedicated
        value: acm-addon
        operator: Equal
```

3. Run the following command to apply the file that you created:

```
oc apply -f addondeploymentconfig
```

4. Use the configuration that you created as the global default configuration for your add-on by running the following command:

```
oc patch clustermanagementaddons <addon-name> --type='json' -p='[{"op":"add",
"path":"/spec/supportedConfigs", "value":[{"group":"addon.open-cluster-
management.io","resource":"addondeploymentconfigs", "defaultConfig":{"name":"deploy-
config","namespace":"open-cluster-management-hub"}}]}]'
```

- Replace **addon-name** with your add-on name.
- Replace **config-name** with the name of the **AddonDeploymentConfig** that you just created.
- Replace **config-namespace** with the namespace of the **AddonDeploymentConfig** that you just created.

The **nodeSelector** and **tolerations** that you specified are applied to all of your add-on on each of the managed clusters.

You can also override the global default **AddonDeploymentConfig** configuration for your add-on on a certain managed cluster by using following steps:

1. Use the **AddonDeploymentConfig** API to create another configuration to specify the **nodeSelector** and **tolerations** on the hub cluster.
2. Link the new configuration that you created to your add-on **ManagedClusterAddon** on a managed cluster.

```
oc -n <managed-cluster> patch managedclusteraddons <addon-name> --type='json' -
p='[{"op":"add", "path":"/spec/configs", "value":[

{"group":"addon.open-cluster-
management.io","resource":"addondeploymentconfigs","namespace":"<config-
namespace>","name":"<config-name>"}
]}]'
```

- Replace **managed-cluster** with your managed cluster name
- Replace **addon-name** with your add-on name
- Replace **config-namespace** with the namespace of the **AddonDeploymentConfig** that you just created
- Replace **config-name** with the name of the **AddonDeploymentConfig** that you just created
  The new configuration that you referenced in the add-on **ManagedClusterAddon** overrides the global default configuration that you previously defined in the **ClusterManagementAddon** add-on.

3. To make sure your content is deployed to the correct nodes, complete the steps in Optional: Configuring the klusterlet to run on specific nodes.

1.3. ENABLING CLUSTER-WIDE PROXY ON EXISTING CLUSTER ADD-ONS

You can configure the **KlusterletAddonConfig** in the cluster namespace to add the proxy environment variables to all the klusterlet add-on pods of the managed Red Hat OpenShift Container Platform clusters. Complete the following steps to configure the **KlusterletAddonConfig** to add the three environment variables to the pods of the klusterlet add-ons:

1. Edit the **KlusterletAddonConfig** file that is in the namespace of the cluster that needs the proxy. You can use the console to find the resource, or you can edit from the terminal with the following command:

```
oc -n <my-cluster-name> edit klusterletaddonconfig <my-cluster-name>
```

Note: If you are working with only one cluster, you do not need **<my-cluster-name>** at the end of your command. See the following command:

```
oc -n <my-cluster-name> edit klusterletaddonconfig
```

2. Edit the **.spec.proxyConfig** section of the file so it resembles the following example. The **spec.proxyConfig** is an optional section:

```
spec
  proxyConfig:
    httpProxy: "<proxy_not_secure>" 1
    httpsProxy: "<proxy_secure>" 2
    noProxy: "<no_proxy>" 3
```

1. Replace **proxy_not_secure** with the address of the proxy server for **http** requests. For example, use **http://192.168.123.145:3128**.

2. Replace **proxy_secure** with the address of the proxy server for **https** requests. For example, use **https://192.168.123.145:3128**.

3. Replace **no_proxy** with a comma delimited list of IP addresses, hostnames, and domain names where traffic is not routed through the proxy. For example, use **.cluster.local,.svc,10.128.0.0/14,example.com**.

If the OpenShift Container Platform cluster is created with cluster wide proxy configured on the hub cluster, the cluster wide proxy configuration values are added to the pods of the klusterlet add-ons as environment variables when the following conditions are met:

- The **.spec.policyController.proxyPolicy** in the **addon** section is enabled and set to **OCPGlobalProxy**.
- The **.spec.applicationManager.proxyPolicy** is enabled and set to **CustomProxy**.
  Note: The default value of **proxyPolicy** in the **addon** section is **Disabled**.

  See the following examples of **proxyPolicy** entries:

```
apiVersion: agent.open-cluster-management.io/v1
   kind: KlusterletAddonConfig
   metadata:
     name: clusterName
     namespace: clusterName
   spec:
     proxyConfig:
       httpProxy: http://pxuser:12345@10.0.81.15:3128
       httpsProxy: http://pxuser:12345@10.0.81.15:3128
       noProxy: .cluster.local,.svc,10.128.0.0/14, example.com
     applicationManager:
       enabled: true
       proxyPolicy: CustomProxy
     policyController:
       enabled: true
       proxyPolicy: OCPGlobalProxy
```

```
    searchCollector:
      enabled: true
      proxyPolicy: Disabled
    certPolicyController:
      enabled: true
      proxyPolicy: Disabled
    iamPolicyController:
      enabled: true
      proxyPolicy: Disabled
```

Important: Global proxy settings do not impact alert forwarding. To set up alert forwarding for Red Hat Advanced Cluster Management hub clusters with a cluster-wide proxy, see Forwarding alerts for more details.