Red Hat Advanced Cluster Management for Kubernetes 2.11

Troubleshooting

Troubleshooting

Last Updated: 2024-07-23

Red Hat Advanced Cluster Management for Kubernetes 2.11 Troubleshooting

Troubleshooting

Legal Notice

Abstract

View a list of troubleshooting topics for your cluster. You can also use the must-gather command to collect logs.

Table of Contents

CHAPTER 1. TROUBLESHOOTING

Before using the Troubleshooting guide, you can run the **oc adm must-gather** command to gather details, logs, and take steps in debugging issues. For more details, see Running the must-gather command to troubleshoot.

Additionally, check your role-based access. See Role-based access control for details.

1.1. DOCUMENTED TROUBLESHOOTING

View the list of troubleshooting topics for Red Hat Advanced Cluster Management for Kubernetes:

Installation

To view the main documentation for the installing tasks, see Installing and upgrading.

- Troubleshooting installation status stuck in installing or pending
- Troubleshooting reinstallation failure
- Troubleshooting ocm-controller errors after Red Hat Advanced Cluster Management upgrade

Backup and restore

To view the main documentation for backup and restore, see Backup and restore.

- Troubleshooting restore status finishes with errors

Cluster management

To view the main documentation about managing your clusters, see The multicluster engine operator cluster lifecycle overview.

- Troubleshooting an offline cluster
- Troubleshooting a managed cluster import failure
- Troubleshooting cluster with pending import status
- Troubleshooting imported clusters offline after certificate change
- Troubleshooting cluster status changing from offline to available
- Troubleshooting cluster creation on VMware vSphere
- Troubleshooting cluster in console with pending or failed status
- Troubleshooting OpenShift Container Platform version 3.11 cluster import failure
- Troubleshooting Klusterlet with degraded conditions
- Troubleshooting Object storage channel secret
- Namespace remains after deleting a cluster
- Auto-import-secret-exists error when importing a cluster

- Troubleshooting the cinder Container Storage Interface (CSI) driver for VolSync

multicluster global hub

To view the main documentation about the multicluster global hub, see multicluster global hub .

- Troubleshooting with the *must-gather* command
- Troubleshooting by accessing the PostgreSQL database
- Troubleshooting by using the database dump and restore

Application management

To view the main documentation about application management, see Managing applications.

- Troubleshooting application Kubernetes deployment version
- Troubleshooting local cluster not selected

Governance

- Troubleshooting multiline YAML parsing

To view the security guide, see Risk and compliance.

Console observability

Console observability includes Search, along with header and navigation function. To view the observability guide, see Observability in the console .

- Troubleshooting grafana
- Troubleshooting observability
- Troubleshooting OpenShift monitoring services
- Troubleshooting metrics-collector
- Troubleshooting PostgreSQL shared memory error

Submariner networking and service discovery

This section lists the Submariner troubleshooting procedures that can occur when using Submariner with Red Hat Advanced Cluster Management or multicluster engine operator. For general Submariner troubleshooting information, see Troubleshooting in the Submariner documentation.

To view the main documentation for the Submariner networking service and service discovery, see Submariner multicluster networking and service discovery .

- Troubleshooting Submariner not connecting after installation - general information
- Troubleshooting Submariner add-on status is degraded

1.2. RUNNING THE MUST-GATHER COMMAND TO TROUBLESHOOT

To get started with troubleshooting, learn about the troubleshooting scenarios for users to run the **must-gather** command to debug the issues, then see the procedures to start using the command.

Required access: Cluster administrator

1.2.1. Must-gather scenarios

- **Scenario one:** Use the Documented troubleshooting section to see if a solution to your problem is documented. The guide is organized by the major functions of the product.
  With this scenario, you check the guide to see if your solution is in the documentation. For instance, for trouble with creating a cluster, you might find a solution in the *Manage cluster* section.

- **Scenario two:** If your problem is not documented with steps to resolve, run the **must-gather** command and use the output to debug the issue.

- **Scenario three:** If you cannot debug the issue using your output from the **must-gather** command, then share your output with Red Hat Support.

1.2.2. Must-gather procedure

See the following procedure to start using the **must-gather** command:

1. Learn about the **must-gather** command and install the prerequisites that you need at Gathering data about your cluster in the Red Hat OpenShift Container Platform documentation.

2. Log in to your cluster. Add the Red Hat Advanced Cluster Management for Kubernetes image that is used for gathering data and the directory. Run the following command, where you insert the image and the directory for the output:

   ```
   oc adm must-gather --image=registry.redhat.io/rhacm2/acm-must-gather-rhel9:v2.11 --dest-dir=<directory>
   ```

3. For the usual use-case, you should run the **must-gather** while you are logged into your *hub* cluster.
   Note: If you want to check your managed clusters, find the **gather-managed.log** file that is located in the **cluster-scoped-resources** directory:

   ```
   <your-directory>/cluster-scoped-resources/gather-managed.log>
   ```

   Check for managed clusters that are not set **True** for the JOINED and AVAILABLE column. You can run the **must-gather** command on those clusters that are not connected with **True** status.

4. Go to your specified directory to see your output, which is organized in the following levels:

   - Two peer levels: **cluster-scoped-resources** and **namespace** resources.

   - Sub-level for each: API group for the custom resource definitions for both cluster-scope and namespace-scoped resources.

   - Next level for each: YAML file sorted by **kind**.

1.2.3. Must-gather in a disconnected environment

Complete the following steps to run the **must-gather** command in a disconnected environment:

1. In a disconnected environment, mirror the Red Hat operator catalog images into their mirror registry. For more information, see Install in disconnected network environments .

2. Run the following commands to collect all of the information, replacing **<2.x>** with the supported version for both **<acm-must-gather>**, for example **2.10**, and **<multicluster-engine/must-gather>**, for example **2.5**.

```
REGISTRY=<internal.repo.address:port>
IMAGE1=$REGISTRY/rhacm2/acm-must-gather-rhel9:v<2.x>
oc adm must-gather --image=$IMAGE1 --dest-dir=<directory>
```

If you experience issues with one of the currently supported releases, or the product documentation, go to Red Hat Support where you can troubleshoot further, view Knowledgebase articles, connect with the Support Team, or open a case. You must log in with your Red Hat credentials.

1.2.4. Must-gather for a hosted cluster

If you experience issues with hosted control plane clusters, you can run the **must-gather** command to gather information to help you with troubleshooting.

1.2.4.1. About the must-gather command for hosted clusters

The command generates output for the managed cluster and the hosted cluster.

- Data from the multicluster engine operator hub cluster:
  - Cluster-scoped resources: These resources are node definitions of the management cluster.
  - The **hypershift-dump** compressed file: This file is useful if you need to share the content with other people.
  - Namespaced resources: These resources include all of the objects from the relevant namespaces, such as config maps, services, events, and logs.
  - Network logs: These logs include the OVN northbound and southbound databases and the status for each one.
  - Hosted clusters: This level of output involves all of the resources inside of the hosted cluster.
- Data from the hosted cluster:
  - Cluster-scoped resources: These resources include all of the cluster-wide objects, such as nodes and CRDs.
  - Namespaced resources: These resources include all of the objects from the relevant namespaces, such as config maps, services, events, and logs.

Although the output does not contain any secret objects from the cluster, it can contain references to the names of the secrets.

1.2.4.2. Prerequisites

To gather information by running the must-gather command, you must meet the following prerequisites:

- You must ensure that the **kubeconfig** file is loaded and is pointing to the multicluster engine operator hub cluster.
- You must have cluster-admin access to the multicluster engine operator hub cluster.
- You must have the name value for the **HostedCluster** resource and the namespace where the custom resource is deployed.

1.2.4.3. Entering the must-gather command for hosted clusters

1. Enter the following command to collect information about the hosted cluster. In the command, the **hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE** parameter is optional; if you do not include it, the command runs as though the hosted cluster is in the default namespace, which is **clusters**.

```
oc adm must-gather --image=quay.io/stolostron/backplane-must-gather:SNAPSHOTNAME
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME
```

2. To save the results of the command to a compressed file, include the **--dest-dir=NAME** parameter, replacing **NAME** with the name of the directory where you want to save the results:

```
oc adm must-gather --image=quay.io/stolostron/backplane-must-gather:SNAPSHOTNAME
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME --dest-dir=NAME ; tar -cvzf NAME.tgz NAME
```

1.2.4.4. Entering the must-gather command in a disconnected environment

Complete the following steps to run the **must-gather** command in a disconnected environment:

1. In a disconnected environment, mirror the Red Hat operator catalog images into their mirror registry. For more information, see Install in disconnected network environments .
2. Run the following command to extract logs, which reference the image from their mirror registry:

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-
rhel8@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8

oc adm must-gather --image=$IMAGE /usr/bin/gather hosted-cluster-
namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME --dest-dir=./data
```

1.2.4.5. Additional resources

- For more information about troubleshooting hosted control planes, see Troubleshooting hosted control planes in the OpenShift Container Platform documentation.

1.3. TROUBLESHOOTING INSTALLATION STATUS STUCK IN INSTALLING OR PENDING

When installing Red Hat Advanced Cluster Management, the **MultiClusterHub** remains in **Installing** phase, or multiple pods maintain a **Pending** status.

1.3.1. Symptom: Stuck in Pending status

More than ten minutes passed since you installed **MultiClusterHub** and one or more components from the **status.components** field of the **MultiClusterHub** resource report **ProgressDeadlineExceeded**. Resource constraints on the cluster might be the issue.

Check the pods in the namespace where **Multiclusterhub** was installed. You might see **Pending** with a status similar to the following:

```
reason: Unschedulable
message: '0/6 nodes are available: 3 Insufficient cpu, 3 node(s) had taint {node-
role.kubernetes.io/master:
    }, that the pod didn't tolerate.'
```

In this case, the worker nodes resources are not sufficient in the cluster to run the product.

1.3.2. Resolving the problem: Adjust worker node sizing

If you have this problem, then your cluster needs to be updated with either larger or more worker nodes. See Sizing your cluster for guidelines on sizing your cluster.

1.4. TROUBLESHOOTING REINSTALLATION FAILURE

When reinstalling Red Hat Advanced Cluster Management for Kubernetes, the pods do not start.

1.4.1. Symptom: Reinstallation failure

If your pods do not start after you install Red Hat Advanced Cluster Management, it is likely that Red Hat Advanced Cluster Management was previously installed, and not all of the pieces were removed before you attempted this installation.

In this case, the pods do not start after completing the installation process.

1.4.2. Resolving the problem: Reinstallation failure

If you have this problem, complete the following steps:

1. Run the uninstallation process to remove the current components by following the steps in Uninstalling.
2. Install the Helm CLI binary version 3.2.0, or later, by following the instructions at Installing Helm.
3. Ensure that your Red Hat OpenShift Container Platform CLI is configured to run **oc** commands. See Getting started with the OpenShift CLI in the OpenShift Container Platform documentation for more information about how to configure the **oc** commands.
4. Copy the following script into a file:

```
#!/bin/bash
ACM_NAMESPACE=<namespace>
oc delete mch --all -n $ACM_NAMESPACE
```

```
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete clusterimageset --all
oc delete clusterrole multiclusterengines.multicluster.openshift.io-v1-admin
multiclusterengines.multicluster.openshift.io-v1-crdview
multiclusterengines.multicluster.openshift.io-v1-edit
multiclusterengines.multicluster.openshift.io-v1-view open-cluster-
management:addons:application-manager open-cluster-management:admin-aggregate open-
cluster-management:cert-policy-controller-hub open-cluster-management:cluster-manager-
admin-aggregate open-cluster-management:config-policy-controller-hub open-cluster-
management:edit-aggregate open-cluster-management:iam-policy-controller-hub open-
cluster-management:policy-framework-hub open-cluster-management:view-aggregate
oc delete crd klusterletaddonconfigs.agent.open-cluster-management.io
placementbindings.policy.open-cluster-management.io policies.policy.open-cluster-
management.io userpreferences.console.open-cluster-management.io
discoveredclusters.discovery.open-cluster-management.io discoveryconfigs.discovery.open-
cluster-management.io
oc delete mutatingwebhookconfiguration ocm-mutating-webhook
managedclustermutators.admission.cluster.open-cluster-management.io multicluster-
observability-operator
oc delete validatingwebhookconfiguration
channels.apps.open.cluster.management.webhook.validator application-webhook-validator
multiclusterhub-operator-validating-webhook ocm-validating-webhook multicluster-
observability-operator multiclusterengines.multicluster.openshift.io
```

Replace **<namespace>** in the script with the name of the namespace where Red Hat Advanced Cluster Management was installed. Ensure that you specify the correct namespace, as the namespace is cleaned out and deleted.

5. Run the script to remove the artifacts from the previous installation.

6. Run the installation. See Installing while connected online .

1.5. TROUBLESHOOTING OCM-CONTROLLER ERRORS AFTER RED HAT ADVANCED CLUSTER MANAGEMENT UPGRADE

After you upgrade from 2.7.x to 2.8.x and then to 2.9.0, the **ocm-controller** of the **multicluster-engine** namespace crashes.

1.5.1. Symptom: Troubleshooting ocm-controller errors after Red Hat Advanced Cluster Management upgrade

After you attempt to list **ManagedClusterSet** and **ManagedClusterSetBinding** custom resource definitions, the following error message appears:

```
Error from server: request to convert CR from an invalid group/version: cluster.open-cluster-
management.io/v1beta1
```

The previous message indicates that the migration of **ManagedClusterSets** and **ManagedClusterSetBindings** custom resource definitions from **v1beta1** to **v1beta2** failed.

1.5.2. Resolving the problem: Troubleshooting ocm-controller errors after Red Hat Advanced Cluster Management upgrade

To resolve this error, you must initiate the API migration manually. Completet the following steps:

1. Revert the **cluster-manager** to a previous release.

   a. Pause the **multiclusterengine** with the following command:

   ```
   oc annotate mce multiclusterengine pause=true
   ```

   b. Run the following commands to replace the image of the **cluster-manager** deployment with a previous version:

   ```
   oc patch deployment cluster-manager -n multicluster-engine -p \  '{"spec":{"template":
   {"spec":{"containers":[{"name":"registration-
   operator","image":"registry.redhat.io/multicluster-engine/registration-operator-
   rhel8@sha256:35999c3a1022d908b6fe30aa9b85878e666392dbbd685e9f3edcb83e3336d
   19f"}]}}}}'
   export ORIGIN_REGISTRATION_IMAGE=$(oc get clustermanager cluster-manager -o
   jsonpath='{.spec.registrationImagePullSpec}')
   ```

   c. Replace the registration image reference in the **ClusterManager** resource with a previous version. Run the following command:

   ```
   oc patch clustermanager cluster-manager --type='json' -p='[{"op": "replace", "path":
   "/spec/registrationImagePullSpec", "value": "registry.redhat.io/multicluster-
   engine/registration-
   rhel8@sha256:a3c22aa4326859d75986bf24322068f0aff2103cccc06e1001faaf79b939051
   5"}]'
   ```

2. Run the following commands to revert the **ManagedClusterSets** and **ManagedClusterSetBindings** custom resource definitions to a previous release:

   ```
   oc annotate crds managedclustersets.cluster.open-cluster-management.io operator.open-
   cluster-management.io/version-
   oc annotate crds  managedclustersetbindings.cluster.open-cluster-management.io
   operator.open-cluster-management.io/version-
   ```

3. Restart the **cluster-manager** and wait for the custom resource definitions to be recreated. Run the following commands:

   ```
   oc -n multicluster-engine delete pods -l app=cluster-manager
   oc wait crds managedclustersets.cluster.open-cluster-management.io --for=jsonpath="
   {.metadata.annotations['operator\.open-cluster-management\.io/version']}"="2.3.3" --
   timeout=120s
   oc wait crds managedclustersetbindings.cluster.open-cluster-management.io --
   for=jsonpath="{.metadata.annotations['operator\.open-cluster-
   management\.io/version']}"="2.3.3" --timeout=120s
   ```

4. Start the storage version migration with the following commands:

   ```
   oc patch StorageVersionMigration managedclustersets.cluster.open-cluster-management.io -
   -type='json' -p='[{"op":"replace", "path":"/spec/resource/version", "value":"v1beta1"}]'
   oc patch StorageVersionMigration managedclustersets.cluster.open-cluster-management.io -
   -type='json' --subresource status -p='[{"op":"remove", "path":"/status/conditions"}]'
   oc patch StorageVersionMigration managedclustersetbindings.cluster.open-cluster-
   ```

```
management.io --type='json' -p='[{"op":"replace", "path":"/spec/resource/version",
"value":"v1beta1"}]'
oc patch StorageVersionMigration managedclustersetbindings.cluster.open-cluster-
management.io --type='json' --subresource status -p='[{"op":"remove",
"path":"/status/conditions"}]'
```

5. Run the following command to wait for the migration to complete:

```
oc wait storageversionmigration managedclustersets.cluster.open-cluster-management.io --
for=condition=Succeeded --timeout=120s
oc wait storageversionmigration managedclustersetbindings.cluster.open-cluster-
management.io --for=condition=Succeeded --timeout=120s
```

6. Restore the **cluster-manager** back to Red Hat Advanced Cluster Management 2.11. It might take several minutes. Run the following command:

```
oc annotate mce multiclusterengine pause-
oc patch clustermanager cluster-manager --type='json' -p='[{"op": "replace", "path":
"/spec/registrationImagePullSpec", "value": "'$ORIGIN_REGISTRATION_IMAGE'"}]'
```

1.5.2.1. Verification

To verify that Red Hat Advanced Cluster Management is recovered run the following commands:

```
oc get managedclusterset
oc get managedclustersetbinding -A
```

After running the commands, the **ManagedClusterSets** and **ManagedClusterSetBindings** resources are listed without error messages.

1.6. TROUBLESHOOTING AN OFFLINE CLUSTER

There are a few common causes for a cluster showing an offline status.

1.6.1. Symptom: Cluster status is offline

After you complete the procedure for creating a cluster, you cannot access it from the Red Hat Advanced Cluster Management console, and it shows a status of **offline**.

1.6.2. Resolving the problem: Cluster status is offline

1. Determine if the managed cluster is available. You can check this in the *Clusters* area of the Red Hat Advanced Cluster Management console.
   If it is not available, try restarting the managed cluster.

2. If the managed cluster status is still offline, complete the following steps:

   a. Run the **oc get managedcluster <cluster_name> -o yaml** command on the hub cluster. Replace **<cluster_name>** with the name of your cluster.

   b. Find the **status.conditions** section.

c. Check the messages for **type: ManagedClusterConditionAvailable** and resolve any problems.

1.7. TROUBLESHOOTING A MANAGED CLUSTER IMPORT FAILURE

If your cluster import fails, there are a few steps that you can take to determine why the cluster import failed.

1.7.1. Symptom: Imported cluster not available

After you complete the procedure for importing a cluster, you cannot access it from the Red Hat Advanced Cluster Management for Kubernetes console.

1.7.2. Resolving the problem: Imported cluster not available

There can be a few reasons why an imported cluster is not available after an attempt to import it. If the cluster import fails, complete the following steps, until you find the reason for the failed import:

1. On the Red Hat Advanced Cluster Management hub cluster, run the following command to ensure that the Red Hat Advanced Cluster Management import controller is running.

   ```
   kubectl -n multicluster-engine get pods -l app=managedcluster-import-controller-v2
   ```

   You should see two pods that are running. If either of the pods is not running, run the following command to view the log to determine the reason:

   ```
   kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1
   ```

2. On the Red Hat Advanced Cluster Management hub cluster, run the following command to determine if the managed cluster import secret was generated successfully by the Red Hat Advanced Cluster Management import controller:

   ```
   kubectl -n <managed_cluster_name> get secrets <managed_cluster_name>-import
   ```

   If the import secret does not exist, run the following command to view the log entries for the import controller and determine why it was not created:

   ```
   kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1 | grep importconfig-controller
   ```

3. On the Red Hat Advanced Cluster Management hub cluster, if your managed cluster is **local-cluster**, provisioned by Hive, or has an auto-import secret, run the following command to check the import status of the managed cluster.

   ```
   kubectl get managedcluster <managed_cluster_name> -o=jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}{"\t"}{.message}{"\n"}{end}' | grep ManagedClusterImportSucceeded
   ```

   If the condition **ManagedClusterImportSucceeded** is not **true**, the result of the command indicates the reason for the failure.

4. Check the Klusterlet status of the managed cluster for a degraded condition. See Troubleshooting Klusterlet with degraded conditions to find the reason that the Klusterlet is degraded.

1.8. TROUBLESHOOTING CLUSTER WITH PENDING IMPORT STATUS

If you receive *Pending import* continually on the console of your cluster, follow the procedure to troubleshoot the problem.

1.8.1. Symptom: Cluster with pending import status

After importing a cluster by using the Red Hat Advanced Cluster Management console, the cluster appears in the console with a status of *Pending import*.

1.8.2. Identifying the problem: Cluster with pending import status

1. Run the following command on the managed cluster to view the Kubernetes pod names that are having the issue:

```
kubectl get pod -n open-cluster-management-agent | grep klusterlet-registration-agent
```

2. Run the following command on the managed cluster to find the log entry for the error:

```
kubectl logs <registration_agent_pod> -n open-cluster-management-agent
```

   Replace *registration_agent_pod* with the pod name that you identified in step 1.

3. Search the returned results for text that indicates there was a networking connectivity problem. Example includes: **no such host**.

1.8.3. Resolving the problem: Cluster with pending import status

1. Retrieve the port number that is having the problem by entering the following command on the hub cluster:

```
oc get infrastructure cluster -o yaml | grep apiServerURL
```

2. Ensure that the hostname from the managed cluster can be resolved, and that outbound connectivity to the host and port is occurring.
   If the communication cannot be established by the managed cluster, the cluster import is not complete. The cluster status for the managed cluster is *Pending import*.

1.9. TROUBLESHOOTING CLUSTER WITH ALREADY EXISTS ERROR

If you are unable to import an OpenShift Container Platform cluster into Red Hat Advanced Cluster Management **MultiClusterHub** and receive an **AlreadyExists** error, follow the procedure to troubleshoot the problem.

1.9.1. Symptom: Already exists error log when importing OpenShift Container Platform cluster

An error log shows up when importing an OpenShift Container Platform cluster into Red Hat Advanced Cluster Management **MultiClusterHub**:

```
error log:
Warning: apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+,
unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition
Error from server (AlreadyExists): error when creating "STDIN":
customresourcedefinitions.apiextensions.k8s.io "klusterlets.operator.open-cluster-management.io"
already exists
The cluster cannot be imported because its Klusterlet CRD already exists.
Either the cluster was already imported, or it was not detached completely during a previous detach
process.
Detach the existing cluster before trying the import again."
```

1.9.2. Identifying the problem: Already exists when importing OpenShift Container Platform cluster

Check if there are any Red Hat Advanced Cluster Management-related resources on the cluster that you want to import to new the Red Hat Advanced Cluster Management **MultiClusterHub** by running the following commands:

```
oc get all -n open-cluster-management-agent
oc get all -n open-cluster-management-agent-addon
```

1.9.3. Resolving the problem: Already exists when importing OpenShift Container Platform cluster

Remove the **klusterlet** custom resource by using the following command:

```
oc get klusterlet | grep klusterlet | awk '{print $1}' | xargs oc patch klusterlet --type=merge -p
'{"metadata":{"finalizers": []}}'
```

Run the following commands to remove pre-existing resources:

```
oc delete namespaces open-cluster-management-agent open-cluster-management-agent-addon --
wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc delete crds --wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc patch crds --type=merge -
p '{"metadata":{"finalizers": []}}'
```

1.10. TROUBLESHOOTING CLUSTER CREATION ON VMWARE VSPHERE

If you experience a problem when creating a Red Hat OpenShift Container Platform cluster on VMware vSphere, see the following troubleshooting information to see if one of them addresses your problem.

Note: Sometimes when the cluster creation process fails on VMware vSphere, the link is not enabled for you to view the logs. If this happens, you can identify the problem by viewing the log of the **hive-controllers** pod. The **hive-controllers** log is in the **hive** namespace.

1.10.1. Managed cluster creation fails with certificate IP SAN error

1.10.1.1. Symptom: Managed cluster creation fails with certificate IP SAN error

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails with an error message that indicates a certificate IP SAN error.

1.10.1.2. Identifying the problem: Managed cluster creation fails with certificate IP SAN error

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
time="2020-08-07T15:27:55Z" level=error msg="Error: error setting up new vSphere SOAP client:
Post https://147.1.1.1/sdk: x509: cannot validate certificate for xx.xx.xx.xx because it doesn't contain
any IP SANs"
time="2020-08-07T15:27:55Z" level=error
```

1.10.1.3. Resolving the problem: Managed cluster creation fails with certificate IP SAN error

Use the VMware vCenter server fully-qualified host name instead of the IP address in the credential. You can also update the VMware vCenter CA certificate to contain the IP SAN.

1.10.2. Managed cluster creation fails with unknown certificate authority

1.10.2.1. Symptom: Managed cluster creation fails with unknown certificate authority

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because the certificate is signed by an unknown authority.

1.10.2.2. Identifying the problem: Managed cluster creation fails with unknown certificate authority

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
Error: error setting up new vSphere SOAP client: Post https://vspherehost.com/sdk: x509: certificate
signed by unknown authority"
```

1.10.2.3. Resolving the problem: Managed cluster creation fails with unknown certificate authority

Ensure you entered the correct certificate from the certificate authority when creating the credential.

1.10.3. Managed cluster creation fails with expired certificate

1.10.3.1. Symptom: Managed cluster creation fails with expired certificate

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because the certificate is expired or is not yet valid.

1.10.3.2. Identifying the problem: Managed cluster creation fails with expired certificate

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
x509: certificate has expired or is not yet valid
```

1.10.3.3. Resolving the problem: Managed cluster creation fails with expired certificate

Ensure that the time on your ESXi hosts is synchronized.

1.10.4. Managed cluster creation fails with insufficient privilege for tagging

1.10.4.1. Symptom: Managed cluster creation fails with insufficient privilege for tagging

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is insufficient privilege to use tagging.

1.10.4.2. Identifying the problem: Managed cluster creation fails with insufficient privilege for tagging

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
time="2020-08-07T19:41:58Z" level=debug msg="vsphere_tag_category.category: Creating..."
time="2020-08-07T19:41:58Z" level=error
time="2020-08-07T19:41:58Z" level=error msg="Error: could not create category: POST
https://vspherehost.com/rest/com/vmware/cis/tagging/category: 403 Forbidden"
time="2020-08-07T19:41:58Z" level=error
time="2020-08-07T19:41:58Z" level=error msg="  on ../tmp/openshift-install-436877649/main.tf line
54, in resource \"vsphere_tag_category\" \"category\":"
time="2020-08-07T19:41:58Z" level=error msg="  54: resource \"vsphere_tag_category\" \"category\"
{"
```

1.10.4.3. Resolving the problem: Managed cluster creation fails with insufficient privilege for tagging

Ensure that your VMware vCenter required account privileges are correct. See Image registry removed during information for more information.

1.10.5. Managed cluster creation fails with invalid dnsVIP

1.10.5.1. Symptom: Managed cluster creation fails with invalid dnsVIP

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an invalid dnsVIP.

1.10.5.2. Identifying the problem: Managed cluster creation fails with invalid dnsVIP

If you see the following message when trying to deploy a new managed cluster with VMware vSphere, it is because you have an older OpenShift Container Platform release image that does not support VMware Installer Provisioned Infrastructure (IPI):

```
failed to fetch Master Machines: failed to load asset \\\"Install Config\\\": invalid \\\"install-
config.yaml\\\" file: platform.vsphere.dnsVIP: Invalid value: \\\"\\\": \\\"\\\" is not a valid IP
```

1.10.5.3. Resolving the problem: Managed cluster creation fails with invalid dnsVIP

Select a release image from a later version of OpenShift Container Platform that supports VMware Installer Provisioned Infrastructure.

1.10.6. Managed cluster creation fails with incorrect network type

1.10.6.1. Symptom: Managed cluster creation fails with incorrect network type

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an incorrect network type specified.

1.10.6.2. Identifying the problem: Managed cluster creation fails with incorrect network type

If you see the following message when trying to deploy a new managed cluster with VMware vSphere, it is because you have an older OpenShift Container Platform image that does not support VMware Installer Provisioned Infrastructure (IPI):

```
time="2020-08-11T14:31:38-04:00" level=debug msg="vsphereprivate_import_ova.import: Creating..."
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error msg="Error: rpc error: code = Unavailable desc = transport is closing"
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=fatal msg="failed to fetch Cluster: failed to generate asset \"Cluster\": failed to create cluster: failed to apply Terraform: failed to complete the change"
```

1.10.6.3. Resolving the problem: Managed cluster creation fails with incorrect network type

Select a valid VMware vSphere network type for the specified VMware cluster.

1.10.7. Managed cluster creation fails with an error processing disk changes

1.10.7.1. Symptom: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an error when processing disk changes.

1.10.7.2. Identifying the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

A message similar to the following is displayed in the logs:

```
ERROR
ERROR Error: error reconfiguring virtual machine: error processing disk changes post-clone: disk.0: ServerFaultCode: NoPermission: RESOURCE (vm-71:2000), ACTION (queryAssociatedProfile): RESOURCE (vm-71), ACTION (PolicyIDByVirtualDisk)
```

1.10.7.3. Resolving the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

Use the VMware vSphere client to give the user **All privileges** for *Profile-driven Storage Privileges*.

1.11. MANAGED CLUSTER CREATION FAILS ON RED HAT OPENSTACK PLATFORM WITH UNKNOWN AUTHORITY ERROR

If you experience a problem when creating a Red Hat OpenShift Container Platform cluster on Red Hat OpenStack Platform, see the following troubleshooting information to see if one of them addresses your problem.

1.11.1. Symptom: Managed cluster creation fails with unknown authority error

After creating a new Red Hat OpenShift Container Platform cluster on Red Hat OpenStack Platform using self-signed certificates, the cluster fails with an error message that indicates an unknown authority error.

1.11.2. Identifying the problem: Managed cluster creation fails with unknown authority error

The deployment of the managed cluster fails and returns the following error message:

x509: certificate signed by unknown authority

1.11.3. Resolving the problem: Managed cluster creation fails with unknown authority error

Verify that the following files are configured correctly:

1. The **clouds.yaml** file must specify the path to the **ca.crt** file in the **cacert** parameter. The **cacert** parameter is passed to the OpenShift installer when generating the ignition shim. See the following example:

```
clouds:
  openstack:
    cacert: "/etc/pki/ca-trust/source/anchors/ca.crt"
```

2. The **certificatesSecretRef** paremeter must reference a secret with a file name matching the **ca.crt** file. See the following example:

```
spec:
  baseDomain: dev09.red-chesterfield.com
  clusterName: txue-osspoke
  platform:
    openstack:
      cloud: openstack
      credentialsSecretRef:
        name: txue-osspoke-openstack-creds
      certificatesSecretRef:
        name: txue-osspoke-openstack-certificatebundle
```

   To create a secret with a matching file name, run the following command:

```
oc create secret generic txue-osspoke-openstack-certificatebundle --from-
file=ca.crt=ca.crt.pem -n $CLUSTERNAME
```

3. The size of the **ca.cert** file must be less than 63 thousand bytes.

1.12. TROUBLESHOOTING OPENSHIFT CONTAINER PLATFORM VERSION 3.11 CLUSTER IMPORT FAILURE

1.12.1. Symptom: OpenShift Container Platform version 3.11 cluster import failure

After you attempt to import a Red Hat OpenShift Container Platform version 3.11 cluster, the import fails with a log message that resembles the following content:

```
customresourcedefinition.apiextensions.k8s.io/klusterlets.operator.open-cluster-management.io
configured
clusterrole.rbac.authorization.k8s.io/klusterlet configured
clusterrole.rbac.authorization.k8s.io/open-cluster-management:klusterlet-admin-aggregate-clusterrole
configured
clusterrolebinding.rbac.authorization.k8s.io/klusterlet configured
namespace/open-cluster-management-agent configured
secret/open-cluster-management-image-pull-credentials unchanged
serviceaccount/klusterlet configured
deployment.apps/klusterlet unchanged
klusterlet.operator.open-cluster-management.io/klusterlet configured
Error from server (BadRequest): error when creating "STDIN": Secret in version "v1" cannot be
handled as a Secret:
v1.Secret.ObjectMeta:
v1.ObjectMeta.TypeMeta: Kind: Data: decode base64: illegal base64 data at input byte 1313, error
found in #10 byte of ...|dhruy45="},"kind":"|..., bigger context
...|tye56u56u568yuo7i67i67i67o556574i"},"kind":"Secret","metadata":{"annotations":{"kube|...
```

1.12.2. Identifying the problem: OpenShift Container Platform version 3.11 cluster import failure

This often occurs because the installed version of the **kubectl** command-line tool is 1.11, or earlier. Run the following command to see which version of the **kubectl** command-line tool you are running:

```
kubectl version
```

If the returned data lists version 1.11, or earlier, complete one of the fixes in *Resolving the problem: OpenShift Container Platform version 3.11 cluster import failure*.

1.12.3. Resolving the problem: OpenShift Container Platform version 3.11 cluster import failure

You can resolve this issue by completing one of the following procedures:

- Install the latest version of the **kubectl** command-line tool.
  1. Download the latest version of the **kubectl** tool from Install and Set Up kubectl in the Kubernetes documentation.
  2. Import the cluster again after upgrading your **kubectl** tool.
- Run a file that contains the import command.
  1. Start the procedure in Importing a managed cluster with the CLI.

2. When you create the command to import your cluster, copy that command into a YAML file named **import.yaml**.

3. Run the following command to import the cluster again from the file:

```
oc apply -f import.yaml
```

1.13. TROUBLESHOOTING IMPORTED CLUSTERS OFFLINE AFTER CERTIFICATE CHANGE

Installing a custom **apiserver** certificate is supported, but one or more clusters that were imported before you changed the certificate information are in **offline** status.

1.13.1. Symptom: Clusters offline after certificate change

After you complete the procedure for updating a certificate secret, one or more of your clusters that were online now display **offline** status in the console.

1.13.2. Identifying the problem: Clusters offline after certificate change

After updating the information for a custom API server certificate, clusters that were imported and running before the new certificate are now in an **offline** state.

The errors that indicate that the certificate is the problem are found in the logs for the pods in the **open-cluster-management-agent** namespace of the offline managed cluster. The following examples are similar to the errors that are displayed in the logs:

See the following **work-agent** log:

```
E0917 03:04:05.874759       1 manifestwork_controller.go:179] Reconcile work test-1-klusterlet-addon-workmgr fails with err: Failed to update work status with err Get "https://api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:05.874887       1 base_controller.go:231] "ManifestWorkAgent" controller failed to sync "test-1-klusterlet-addon-workmgr", err: Failed to update work status with err Get "api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:37.245859       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156: Failed to watch *v1.ManifestWork: failed to list *v1.ManifestWork: Get "api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks?resourceVersion=607424": x509: certificate signed by unknown authority
```

See the following **registration-agent** log:

```
I0917 02:27:41.525026       1 event.go:282] Event(v1.ObjectReference{Kind:"Namespace", Namespace:"open-cluster-management-agent", Name:"open-cluster-management-agent", UID:"", APIVersion:"v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason: 'ManagedClusterAvailableConditionUpdated' update managed cluster "test-1" available condition to "True", due to "Managed cluster is available"
E0917 02:58:26.315984       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156: Failed to watch *v1beta1.CertificateSigningRequest: Get "https://api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true"": x509: certificate
```

```
signed by unknown authority
E0917 02:58:26.598343       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManagedCluster: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true": x509: certificate
signed by unknown authority
E0917 02:58:27.613963       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManagedCluster: failed to list *v1.ManagedCluster: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true"": x509: certificate
signed by unknown authority
```

1.13.3. Resolving the problem: Clusters offline after certificate change

If your managed cluster is the **local-cluster**, or your managed cluster was created by using Red Hat Advanced Cluster Management for Kubernetes, you must wait 10 minutes or longer to reimport your managed cluster.

To reimport your managed cluster immediately, you can delete your managed cluster import secret on the hub cluster and reimport it by using Red Hat Advanced Cluster Management. Run the following command:

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

Replace **<cluster_name>** with the name of the managed cluster that you want to import.

If you want to reimport a managed cluster that was imported by using Red Hat Advanced Cluster Management, complete the following steps to import the managed cluster again:

1. On the hub cluster, recreate the managed cluster import secret by running the following command:

   ```
   oc delete secret -n <cluster_name> <cluster_name>-import
   ```

   Replace **<cluster_name>** with the name of the managed cluster that you want to import.

2. On the hub cluster, expose the managed cluster import secret to a YAML file by running the following command:

   ```
   oc get secret -n <cluster_name> <cluster_name>-import -ojsonpath='{.data.import\.yaml}' |
   base64 --decode  > import.yaml
   ```

   Replace **<cluster_name>** with the name of the managed cluster that you want to import.

3. On the managed cluster, apply the **import.yaml** file by running the following command:

   ```
   oc apply -f import.yaml
   ```

Note: The previous steps do not detach the managed cluster from the hub cluster. The steps update the required manifests with current settings on the managed cluster, including the new certificate information.

1.14. NAMESPACE REMAINS AFTER DELETING A CLUSTER

When you remove a managed cluster, the namespace is normally removed as part of the cluster removal process. In rare cases, the namespace remains with some artifacts in it. In that case, you must manually remove the namespace.

1.14.1. Symptom: Namespace remains after deleting a cluster

After removing a managed cluster, the namespace is not removed.

1.14.2. Resolving the problem: Namespace remains after deleting a cluster

Complete the following steps to remove the namespace manually:

1. Run the following command to produce a list of the resources that remain in the <cluster_name> namespace:

   ```
   oc api-resources --verbs=list --namespaced -o name | grep -E
   '^secrets|^serviceaccounts|^managedclusteraddons|^roles|^rolebindings|^manifestworks|^lease
   |^managedclusterinfo|^appliedmanifestworks'|^clusteroauths' | xargs -n 1 oc get --show-kind -
   -ignore-not-found -n <cluster_name>
   ```

   Replace **cluster_name** with the name of the namespace for the cluster that you attempted to remove.

2. Delete each identified resource on the list that does not have a status of **Delete** by entering the following command to edit the list:

   ```
   oc edit <resource_kind> <resource_name> -n <namespace>
   ```

   Replace **resource_kind** with the kind of the resource. Replace **resource_name** with the name of the resource. Replace **namespace** with the name of the namespace of the resource.

3. Locate the **finalizer** attribute in the in the metadata.

4. Delete the non-Kubernetes finalizers by using the vi editor **dd** command.

5. Save the list and exit the **vi** editor by entering the **:wq** command.

6. Delete the namespace by entering the following command:

   ```
   oc delete ns <cluster-name>
   ```

   Replace **cluster-name** with the name of the namespace that you are trying to delete.

1.15. AUTO-IMPORT-SECRET-EXISTS ERROR WHEN IMPORTING A CLUSTER

Your cluster import fails with an error message that reads: auto import secret exists.

1.15.1. Symptom: Auto import secret exists error when importing a cluster

When importing a hive cluster for management, an **auto-import-secret already exists** error is displayed.

1.15.2. Resolving the problem: Auto-import-secret-exists error when importing a cluster

This problem occurs when you attempt to import a cluster that was previously managed by Red Hat Advanced Cluster Management. When this happens, the secrets conflict when you try to reimport the cluster.

To work around this problem, complete the following steps:

1. To manually delete the existing **auto-import-secret**, run the following command on the hub cluster:

   ```
   oc delete secret auto-import-secret -n <cluster-namespace>
   ```

   Replace **cluster-namespace** with the namespace of your cluster.

2. Import your cluster again by using the procedure in Cluster import introduction .

1.16. TROUBLESHOOTING THE CINDER CONTAINER STORAGE INTERFACE (CSI) DRIVER FOR VOLSYNC

If you use VolSync or use a default setting in a cinder Container Storage Interface (CSI) driver, you might encounter errors for the PVC that is in use.

1.16.1. Symptom: Volumesnapshot error state

You can configure a VolSync **ReplicationSource** or **ReplicationDestination** to use snapshots. Also, you can configure the **storageclass** and **volumesnapshotclass** in the **ReplicationSource** and **ReplicationDestination**. There is a parameter on the cinder **volumesnapshotclass** called **force-create** with a default value of **false**. This **force-create** parameter on the **volumesnapshotclass** means cinder does not allow the **volumesnapshot** to be taken of a PVC in use. As a result, the **volumesnapshot** is in an error state.

1.16.2. Resolving the problem: Setting the parameter to true

1. Create a new **volumesnapshotclass** for the cinder CSI driver.

2. Change the paramater, **force-create**, to **true**. See the following sample YAML:

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   deletionPolicy: Delete
   driver: cinder.csi.openstack.org
   kind: VolumeSnapshotClass
   metadata:
     annotations:
       snapshot.storage.kubernetes.io/is-default-class: 'true'
     name: standard-csi
   parameters:
     force-create: 'true'
   ```

1.17. TROUBLESHOOTING WITH THE *MUST-GATHER* COMMAND

1.17.1. Symptom: Errors with multicluster global hub

You might experience various errors with multicluster global hub. You can run the **must-gather** command for troubleshooting issues with multicluster global hub.

1.17.2. Resolving the problem: Running the *must-gather* command for dubugging

Run the **must-gather** command to gather details, logs, and take steps in debugging issues. This debugging information is also useful when you open a support request. The **oc adm must-gather** CLI command collects the information from your cluster that is often needed for debugging issues, including:

- Resource definitions
- Service logs

1.17.2.1. Prerequisites

You must meet the following prerequisites to run the **must-gather** command:

- Access to the global hub and managed hub clusters as a user with the **cluster-admin** role.
- The OpenShift Container Platform CLI (oc) installed.

1.17.2.2. Running the must-gather command

Complete the following procedure to collect information by using the must-gather command:

1. Learn about the **must-gather** command and install the prerequisites that you need by reading the Gathering data about your cluster in the OpenShift Container Platform documentation.
2. Log in to your global hub cluster. For the typical use case, run the following command while you are logged into your global hub cluster:

   ```
   oc adm must-gather --image=quay.io/stolostron/must-gather:SNAPSHOTNAME
   ```

   If you want to check your managed hub clusters, run the **must-gather** command on those clusters.
3. Optional: If you want to save the results in a the **SOMENAME** directory, you can run the following command instead of the one in the previous step:

   ```
   oc adm must-gather --image=quay.io/stolostron/must-gather:SNAPSHOTNAME --dest-dir=
   <SOMENAME> ; tar -cvzf <SOMENAME>.tgz <SOMENAME>
   ```

   You can specify a different name for the directory.

   Note: The command includes the required additions to create a gzipped tarball file.

The following information is collected from the **must-gather** command:

- Two peer levels: **cluster-scoped-resources** and **namespaces** resources.
- Sub-level for each: API group for the custom resource definitions for both cluster-scope and namespace-scoped resources.

- Next level for each: YAML file sorted by kind.
- For the global hub cluster, you can check the **PostgresCluster** and **Kafka** in the **namespaces** resources.
- For the global hub cluster, you can check the multicluster global hub related pods and logs in **pods** of **namespaces** resources.
- For the managed hub cluster, you can check the multicluster global hub agent pods and logs in **pods** of **namespaces** resources.

1.18. TROUBLESHOOTING BY ACCESSING THE POSTGRESQL DATABASE

1.18.1. Symptom: Errors with multicluster global hub

You might experience various errors with multicluster global hub. You can access the provisioned PostgreSQL database to view messages that might be helpful for troubleshooting issues with multicluster global hub.

1.18.2. Resolving the problem: Accessing the PostgresSQL database

> There are two ways to access the provisioned PostgreSQL database.

- Using the **ClusterIP** service

```
oc exec -it multicluster-global-hub-postgres-0 -c multicluster-global-hub-postgres -n multicluster-global-hub -- psql -U postgres -d hoh

# Or access the database installed by crunchy operator
oc exec -it $(kubectl get pods -n multicluster-global-hub -l postgres-operator.crunchydata.com/role=master -o jsonpath='{.items..metadata.name}') -c database -n multicluster-global-hub -- psql -U postgres -d hoh -c "SELECT 1"
```

- **LoadBalancer**
  - Expose the service type to **LoadBalancer** provisioned by default:

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: Service
metadata:
  name: multicluster-global-hub-postgres-lb
  namespace: multicluster-global-hub
spec:
  ports:
  - name: postgres
    port: 5432
    protocol: TCP
    targetPort: 5432
  selector:
    name: multicluster-global-hub-postgres
  type: LoadBalancer
EOF
```

Run the following command to get your the credentials:

```
# Host
oc get svc postgres-ha -ojsonpath='{.status.loadBalancer.ingress[0].hostname}'

# Password
oc get secrets -n multicluster-global-hub postgres-pguser-postgres -o go-template='{{index (.data) "password" | base64decode}}'
```

- Expose the service type to **LoadBalancer** provisioned by crunchy operator:

```
oc patch postgrescluster postgres -n multicluster-global-hub -p '{"spec":{"service":{"type":"LoadBalancer"}}}'  --type merge
```

Run the following command to get your the credentials:

```
# Host
oc get svc -n multicluster-global-hub postgres-ha -ojsonpath='{.status.loadBalancer.ingress[0].hostname}'

# Username
oc get secrets -n multicluster-global-hub postgres-pguser-postgres -o go-template='{{index (.data) "user" | base64decode}}'

# Password
oc get secrets -n multicluster-global-hub postgres-pguser-postgres -o go-template='{{index (.data) "password" | base64decode}}'

# Database
oc get secrets -n multicluster-global-hub postgres-pguser-postgres -o go-template='{{index (.data) "dbname" | base64decode}}'
```

1.19. TROUBLESHOOTING BY USING THE DATABASE DUMP AND RESTORE

In a production environment, back up your PostgreSQL database regularly as a database management task. The backup can also be used for debugging the multicluster global hub.

1.19.1. Symptom: Errors with multicluster global hub

You might experience various errors with multicluster global hub. You can use the database dump and restore for troubleshooting issues with multicluster global hub.

1.19.2. Resolving the problem: Dumping the output of the database for dubugging

Sometimes you need to dump the output in the multicluster global hub database to debug a problem. The PostgreSQL database provides the **pg_dump** command line tool to dump the contents of the database. To dump data from localhost database server, run the following command:

```
pg_dump hoh > hoh.sql
```

To dump the multicluster global hub database located on a remote server with compressed format, use the command-line options to control the connection details, as shown in the following example:

```
pg_dump -h my.host.com -p 5432 -U postgres -F t hoh -f hoh-$(date +%d-%m-%y_%H-%M).tar
```

1.19.3. Resolving the problem: Restore database from dump

To restore a PostgreSQL database, you can use the **psql** or **pg_restore** command line tools. The **psql** tool is used to restore plain text files created by **pg_dump**:

```
psql -h another.host.com -p 5432 -U postgres -d hoh < hoh.sql
```

The **pg_restore** tool is used to restore a PostgreSQL database from an archive created by **pg_dump** in one of the non-plain-text formats (custom, tar, or directory):

```
pg_restore -h another.host.com -p 5432 -U postgres -d hoh hoh-$(date +%d-%m-%y_%H-%M).tar
```

1.20. TROUBLESHOOTING CLUSTER STATUS CHANGING FROM OFFLINE TO AVAILABLE

The status of the managed cluster alternates between **offline** and **available** without any manual change to the environment or cluster.

1.20.1. Symptom: Cluster status changing from offline to available

When the network that connects the managed cluster to the hub cluster is unstable, the status of the managed cluster that is reported by the hub cluster cycles between **offline** and **available**.

The connection between the hub cluster and managed cluster is maintained through a lease that is validated at the **leaseDurationSeconds** interval value. If the lease is not validated within five consecutive attempts of the **leaseDurationSeconds** value, then the cluster is marked **offline**.

For example, the cluster is marked **offline** after five minutes with a **leaseDurationSeconds** interval of **60 seconds**. This configuration can be inadequate for reasons such as connectivity issues or latency, causing instability.

1.20.2. Resolving the problem: Cluster status changing from offline to available

The five validation attempts is default and cannot be changed, but you can change the **leaseDurationSeconds** interval.

Determine the amount of time, in minutes, that you want the cluster to be marked as **offline**, then multiply that value by 60 to convert to seconds. Then divide by the default five number of attempts. The result is your **leaseDurationSeconds** value.

1. Edit your **ManagedCluster** specification on the hub cluster by entering the following command, but replace **cluster-name** with the name of your managed cluster:

   ```
   oc edit managedcluster <cluster-name>
   ```

2. Increase the value of **leaseDurationSeconds** in your **ManagedCluster** specification, as seen in the following sample YAML:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: <cluster-name>
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

3. Save and apply the file.

1.21. TROUBLESHOOTING CLUSTER IN CONSOLE WITH PENDING OR FAILED STATUS

If you observe *Pending* status or *Failed* status in the console for a cluster you created, follow the procedure to troubleshoot the problem.

1.21.1. Symptom: Cluster in console with pending or failed status

After creating a new cluster by using the Red Hat Advanced Cluster Management for Kubernetes console, the cluster does not progress beyond the status of *Pending* or displays *Failed* status.

1.21.2. Identifying the problem: Cluster in console with pending or failed status

If the cluster displays *Failed* status, navigate to the details page for the cluster and follow the link to the logs provided. If no logs are found or the cluster displays *Pending* status, continue with the following procedure to check for logs:

- Procedure 1

  1. Run the following command on the hub cluster to view the names of the Kubernetes pods that were created in the namespace for the new cluster:

     ```
     oc get pod -n <new_cluster_name>
     ```

     Replace ***new_cluster_name*** with the name of the cluster that you created.

  2. If no pod that contains the string **provision** in the name is listed, continue with Procedure 2. If there is a pod with **provision** in the title, run the following command on the hub cluster to view the logs of that pod:

     ```
     oc logs <new_cluster_name_provision_pod_name> -n <new_cluster_name> -c hive
     ```

     Replace **new_cluster_name_provision_pod_name** with the name of the cluster that you created, followed by the pod name that contains **provision**.

  3. Search for errors in the logs that might explain the cause of the problem.

- Procedure 2
  If there is not a pod with **provision** in its name, the problem occurred earlier in the process. Complete the following procedure to view the logs:

  1. Run the following command on the hub cluster:

     ```
     oc describe clusterdeployments -n <new_cluster_name>
     ```

Replace **new_cluster_name** with the name of the cluster that you created. For more information about cluster installation logs, see Gathering installation logs in the Red Hat OpenShift documentation.

2. See if there is additional information about the problem in the *Status.Conditions.Message* and *Status.Conditions.Reason* entries of the resource.

1.21.3. Resolving the problem: Cluster in console with pending or failed status

After you identify the errors in the logs, determine how to resolve the errors before you destroy the cluster and create it again.

The following example provides a possible log error of selecting an unsupported zone, and the actions that are required to resolve it:

```
No subnets provided for zones
```

When you created your cluster, you selected one or more zones within a region that are not supported. Complete one of the following actions when you recreate your cluster to resolve the issue:

- Select a different zone within the region.
- Omit the zone that does not provide the support, if you have other zones listed.
- Select a different region for your cluster.

After determining the issues from the log, destroy the cluster and recreate it.

See Cluster creation introduction for more information about creating a cluster.

1.22. TROUBLESHOOTING GRAFANA

When you query some time-consuming metrics in the Grafana explorer, you might encounter a **Gateway Time-out** error.

1.22.1. Symptom: Grafana explorer gateway timeout

If you hit the **Gateway Time-out** error when you query some time-consuming metrics in the Grafana explorer, it is possible that the timeout is caused by the Grafana in the **open-cluster-management-observability** namespace.

1.22.2. Resolving the problem: Configure the Grafana

If you have this problem, complete the following steps:

1. Verify that the default configuration of Grafana has expected timeout settings:
   a. To verify that the default timeout setting of Grafana, run the following command:

```
oc get secret grafana-config -n open-cluster-management-observability -o jsonpath="
{.data.grafana\.ini}" | base64 -d | grep dataproxy -A 4
```

   The following timeout settings should be displayed:

```
[dataproxy]
timeout = 300
dial_timeout = 30
keep_alive_seconds = 300
```

b. To verify the default data source query timeout for Grafana, run the following command:

```
oc get secret/grafana-datasources -n open-cluster-management-observability -o
jsonpath="{.data.datasources\.yaml}" | base64 -d | grep queryTimeout
```

The following timeout settings should be displayed:

```
queryTimeout: 300s
```

2. If you verified the default configuration of Grafana has expected timeout settings, then you can configure the Grafana in the **open-cluster-management-observability** namespace by running the following command:

```
oc annotate route grafana -n open-cluster-management-observability --overwrite
haproxy.router.openshift.io/timeout=300s
```

Refresh the Grafana page and try to query the metrics again. The **Gateway Time-out** error is no longer displayed.

1.23. TROUBLESHOOTING LOCAL CLUSTER NOT SELECTED WITH PLACEMENT RULE

The managed clusters are selected with a placement rule, but the **local-cluster**, which is a hub cluster that is also managed, is not selected. The placement rule user is not granted permission to get the **managedcluster** resources in the **local-cluster** namespace.

1.23.1. Symptom: Troubleshooting local cluster not selected as a managed cluster

All managed clusters are selected with a placement rule, but the **local-cluster** is not. The placement rule user is not granted permission to get the **managedcluster** resources in the **local-cluster** namespace.

1.23.2. Resolving the problem: Troubleshooting local cluster not selected as a managed cluster

Deprecated: PlacementRule

To resolve this issue, you need to grant the **managedcluster** administrative permission in the **local-cluster** namespace. Complete the following steps:

1. Confirm that the list of managed clusters does include **local-cluster**, and that the placement rule **decisions** list does not display the **local-cluster**. Run the following command and view the results:

```
% oc get managedclusters
```

See in the sample output that **local-cluster** is joined, but it is not in the YAML for **PlacementRule**:

```
NAME          HUB ACCEPTED   MANAGED CLUSTER URLS   JOINED   AVAILABLE
AGE
local-cluster   true                          True     True       56d
cluster1        true                          True     True       16h
```

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  name: all-ready-clusters
  namespace: default
spec:
  clusterSelector: {}
status:
  decisions:
  - clusterName: cluster1
    clusterNamespace: cluster1
```

2. Create a **Role** in your YAML file to grant the **managedcluster** administrative permission in the **local-cluster** namespace. See the following example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: managedcluster-admin-user-zisis
  namespace: local-cluster
rules:
- apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - managedclusters
  verbs:
  - get
```

3. Create a **RoleBinding** resource to grant the placement rule user access to the **local-cluster** namespace. See the following example:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: managedcluster-admin-user-zisis
  namespace: local-cluster
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: managedcluster-admin-user-zisis
  namespace: local-cluster
subjects:
- kind: User
  name: zisis
  apiGroup: rbac.authorization.k8s.io
```

1.24. TROUBLESHOOTING APPLICATION KUBERNETES DEPLOYMENT VERSION

A managed cluster with a deprecated Kubernetes **apiVersion** might not be supported. See the Kubernetes issue for more details about the deprecated API version.

1.24.1. Symptom: Application deployment version

If one or more of your application resources in the Subscription YAML file uses the deprecated API, you might receive an error similar to the following error:

```
failed to install release: unable to build kubernetes objects from release manifest: unable to recognize
"": no matches for
kind "Deployment" in version "extensions/v1beta1"
```

Or with new Kubernetes API version in your YAML file named **old.yaml** for instance, you might receive the following error:

```
error: unable to recognize "old.yaml": no matches for kind "Deployment" in version
"deployment/v1beta1"
```

1.24.2. Resolving the problem: Application deployment version

1. Update the **apiVersion** in the resource. For example, if the error displays for *Deployment* kind in the subscription YAML file, you need to update the **apiVersion** from **extensions/v1beta1** to **apps/v1**.
   See the following example:

   ```
   apiVersion: apps/v1
   kind: Deployment
   ```

2. Verify the available versions by running the following command on the managed cluster:

   ```
   kubectl explain <resource>
   ```

3. Check for **VERSION**.

1.25. TROUBLESHOOTING KLUSTERLET WITH DEGRADED CONDITIONS

The Klusterlet degraded conditions can help to diagnose the status of Klusterlet agents on managed cluster. If a Klusterlet is in the degraded condition, the Klusterlet agents on managed cluster might have errors that need to be troubleshooted. See the following information for Klusterlet degraded conditions that are set to **True**.

1.25.1. Symptom: Klusterlet is in the degraded condition

After deploying a Klusterlet on managed cluster, the **KlusterletRegistrationDegraded** or **KlusterletWorkDegraded** condition displays a status of *True*.

1.25.2. Identifying the problem: Klusterlet is in the degraded condition

1. Run the following command on the managed cluster to view the Klusterlet status:

```
kubectl get klusterlets klusterlet -oyaml
```

2. Check **KlusterletRegistrationDegraded** or **KlusterletWorkDegraded** to see if the condition is set to **True**. Proceed to *Resolving the problem* for any degraded conditions that are listed.

1.25.3. Resolving the problem: Klusterlet is in the degraded condition

See the following list of degraded statuses and how you can attempt to resolve those issues:

- If the **KlusterletRegistrationDegraded** condition with a status of *True* and the condition reason is: *BootStrapSecretMissing*, you need create a bootstrap secret on **open-cluster-management-agent** namespace.
- If the **KlusterletRegistrationDegraded** condition displays *True* and the condition reason is a *BootstrapSecretError*, or *BootstrapSecretUnauthorized*, then the current bootstrap secret is invalid. Delete the current bootstrap secret and recreate a valid bootstrap secret on **open-cluster-management-agent** namespace.
- If the **KlusterletRegistrationDegraded** and **KlusterletWorkDegraded** displays *True* and the condition reason is *HubKubeConfigSecretMissing*, delete the Klusterlet and recreate it.
- If the **KlusterletRegistrationDegraded** and **KlusterletWorkDegraded** displays *True* and the condition reason is: *ClusterNameMissing*, *KubeConfigMissing*, *HubConfigSecretError*, or *HubConfigSecretUnauthorized*, delete the hub cluster kubeconfig secret from **open-cluster-management-agent** namespace. The registration agent will bootstrap again to get a new hub cluster kubeconfig secret.
- If the **KlusterletRegistrationDegraded** displays *True* and the condition reason is *GetRegistrationDeploymentFailed*, or *UnavailableRegistrationPod*, you can check the condition message to get the problem details and attempt to resolve.
- If the **KlusterletWorkDegraded** displays *True* and the condition reason is *GetWorkDeploymentFailed* ,or *UnavailableWorkPod*, you can check the condition message to get the problem details and attempt to resolve.

1.26. TROUBLESHOOTING OBJECT STORAGE CHANNEL SECRET

If you change the **SecretAccessKey**, the subscription of an Object storage channel cannot pick up the updated secret automatically and you receive an error.

1.26.1. Symptom: Object storage channel secret

The subscription of an Object storage channel cannot pick up the updated secret automatically. This prevents the subscription operator from reconciliation and deploys resources from Object storage to the managed cluster.

1.26.2. Resolving the problem: Object storage channel secret

You need to manually input the credentials to create a secret, then refer to the secret within a channel.

1. Annotate the subscription CR in order to generate a reconcile single to subscription operator. See the following **data** specification:

```
apiVersion: apps.open-cluster-management.io/v1
```

```
kind: Channel
metadata:
  name: deva
  namespace: ch-obj
  labels:
    name: obj-sub
spec:
  type: ObjectBucket
  pathname: http://ec2-100-26-232-156.compute-1.amazonaws.com:9000/deva
  sourceNamespaces:
    - default
  secretRef:
    name: dev
---
apiVersion: v1
kind: Secret
metadata:
  name: dev
  namespace: ch-obj
  labels:
    name: obj-sub
data:
  AccessKeyID: YWRtaW4=
  SecretAccessKey: cGFzc3dvcmRhZG1pbg==
```

2. Run **oc annotate** to test:

```
oc annotate appsub -n <subscription-namespace> <subscription-name> test=true
```

After you run the command, you can go to the Application console to verify that the resource is deployed to the managed cluster. Or you can log in to the managed cluster to see if the application resource is created at the given namespace.

1.27. TROUBLESHOOTING OBSERVABILITY

After you install the observability component, the component might be stuck and an **Installing** status is displayed.

1.27.1. Symptom: MultiClusterObservability resource status stuck

If the observability status is stuck in an **Installing** status after you install and create the Observability custom resource definition (CRD), it is possible that there is no value defined for the **spec:storageConfig:storageClass** parameter. Alternatively, the observability component automatically finds the default **storageClass**, but if there is no value for the storage, the component remains stuck with the **Installing** status.

1.27.2. Resolving the problem: MultiClusterObservability resource status stuck

If you have this problem, complete the following steps:

1. Verify that the observability components are installed:
   a. To verify that the **multicluster-observability-operator**, run the following command:

```
kubectl get pods -n open-cluster-management|grep observability
```

b. To verify that the appropriate CRDs are present, run the following command:

```
kubectl get crd|grep observ
```

The following CRDs must be displayed before you enable the component:

```
multiclusterobservabilities.observability.open-cluster-management.io
observabilityaddons.observability.open-cluster-management.io
observatoria.core.observatorium.io
```

2. If you create your own storageClass for a Bare Metal cluster, see Persistent storage using NFS .

3. To ensure that the observability component can find the default storageClass, update the **storageClass** parameter in the **multicluster-observability-operator** custom resource definition. Your parameter might resemble the following value:

```
storageclass.kubernetes.io/is-default-class: "true"
```

The observability component status is updated to a *Ready* status when the installation is complete. If the installation fails to complete, the *Fail* status is displayed.

1.28. TROUBLESHOOTING OPENSHIFT MONITORING SERVICE

Observability service in a managed cluster needs to scrape metrics from the OpenShift Container Platform monitoring stack. The **metrics-collector** is not installed if the OpenShift Container Platform monitoring stack is not ready.

1.28.1. Symptom: OpenShift monitoring service is not ready

The **endpoint-observability-operator-x** pod checks if the **prometheus-k8s** service is available in the **openshift-monitoring** namespace. If the service is not present in the **openshift-monitoring** namespace, then the **metrics-collector** is not deployed. You might receive the following error message: **Failed to get prometheus resource**.

1.28.2. Resolving the problem: OpenShift monitoring service is not ready

If you have this problem, complete the following steps:

1. Log in to your OpenShift Container Platform cluster.

2. Access the **openshift-monitoring** namespace to verify that the **prometheus-k8s** service is available.

3. Restart **endpoint-observability-operator-x** pod in the **open-cluster-management-addon-observability** namespace of the managed cluster.

1.29. TROUBLESHOOTING METRICS-COLLECTOR

When the **observability-client-ca-certificate** secret is not refreshed in the managed cluster, you might receive an internal server error.

1.29.1. Symptom: metrics-collector cannot verify observability-client-ca-certificate

There might be a managed cluster, where the metrics are unavailable. If this is the case, you might receive the following error from the **metrics-collector** deployment:

```
error: response status code is 500 Internal Server Error, response body is x509: certificate signed by unknown authority (possibly because of "crypto/rsa: verification error" while trying to verify candidate authority certificate "observability-client-ca-certificate")
```

1.29.2. Resolving the problem: metrics-collector cannot verify observability-client-ca-certificate

If you have this problem, complete the following steps:

1. Log in to your managed cluster.
2. Delete the secret named, **observability-controller-open-cluster-management.io-observability-signer-client-cert** that is in the **open-cluster-management-addon-observability** namespace. Run the following command:

   ```
   oc delete secret observability-controller-open-cluster-management.io-observability-signer-client-cert -n open-cluster-management-addon-observability
   ```

   Note: The **observability-controller-open-cluster-management.io-observability-signer-client-cert** is automatically recreated with new certificates.

The **metrics-collector** deployment is recreated and the **observability-controller-open-cluster-management.io-observability-signer-client-cert** secret is updated.

1.30. TROUBLESHOOTING POSTGRESQL SHARED MEMORY ERROR

If you have a large environment, you might encounter a PostgreSQL shared memory error that impacts search results and the topology view for applications.

1.30.1. Symptom: PostgreSQL shared memory error

An error message resembling the following appears in the **search-api** logs: **ERROR: could not resize shared memory segment "/PostgreSQL.1083654800" to 25031264 bytes: No space left on device (SQLSTATE 53100)**

1.30.2. Resolving the problem: PostgreSQL shared memory error

To resolve the issue, update the PostgreSQL resources found in the **search-postgres** ConfigMap. Complete the following steps to update the resources:

1. Run the following command to switch to the **open-cluster-management** project:

   ```
   oc project open-cluster-management
   ```

2. Increase the **search-postgres** pod memory. The following command increases the memory to **16Gi**:

```
oc patch search -n open-cluster-management search-v2-operator --type json -p '[{"op": "add", "path": "/spec/deployments/database/resources", "value": {"limits": {"memory": "16Gi"}, "requests": {"memory": "32Mi", "cpu": "25m"}}}]'
```

3. Run the following command to prevent the search operator from overwriting your changes:

```
oc annotate search search-v2-operator search-pause=true
```

4. Run the following command to update the resources in the **search-postgres** YAML file:

```
oc edit cm search-postgres -n open-cluster-management
```

See the following example for increasing resources:

```
postgresql.conf: |-
  work_mem = '128MB' # Higher values allocate more memory
  max_parallel_workers_per_gather = '0' # Disables parallel queries
  shared_buffers = '1GB' # Higher values allocate more memory
```

Make sure to save your changes before exiting.

5. Run the following command to restart the **postgres** and **api** pod.

```
oc delete pod search-postgres-xyz search-api-xzy
```

6. To verify your changes, open the **search-postgres** YAML file and confirm that the changes you made to **postgresql.conf:** are present by running the following command:

```
oc get cm search-postgres -n open-cluster-management -o yaml
```

See Search customization and configurations for more information on adding environment variables.

1.31. TROUBLESHOOTING SUBMARINER NOT CONNECTING AFTER INSTALLATION

If Submariner does not run correctly after you configure it, complete the following steps to diagnose the issue.

1.31.1. Symptom: Submariner not connecting after installation

Your Submariner network is not communicating after installation.

1.31.2. Identifying the problem: Submariner not connecting after installation

If the network connectivity is not established after deploying Submariner, begin the troubleshooting steps. Note that it might take several minutes for the processes to complete when you deploy Submariner.

1.31.3. Resolving the problem: Submariner not connecting after installation

When Submariner does not run correctly after deployment, complete the following steps:

1. Check for the following requirements to determine whether the components of Submariner deployed correctly:
    - The **submariner-addon** pod is running in the **open-cluster-management** namespace of your hub cluster.
    - The following pods are running in the **submariner-operator** namespace of each managed cluster:
        - submariner-addon
        - submariner-gateway
        - submariner-routeagent
        - submariner-operator
        - submariner-globalnet (only if Globalnet is enabled in the ClusterSet)
        - submariner-lighthouse-agent
        - submariner-lighthouse-coredns
        - submariner-networkplugin-syncer (only if the specified CNI value is **OVNKubernetes**)
        - submariner-metrics-proxy
2. Run the **subctl diagnose all** command to check the status of the required pods, with the exception of the **submariner-addon** pods.
3. Make sure to run the **must-gather** command to collect logs that can help with debugging issues.

1.32. TROUBLESHOOTING SUBMARINER ADD-ON STATUS IS DEGRADED

After adding the Submariner add-on to the clusters in your cluster set, the status in the *Connection status*, *Agent status*, and *Gateway nodes* show unexpected status for the clusters.

1.32.1. Symptom: Submariner add-on status is degraded

You add the Submariner add-on to the clusters in your cluster set, the following status is shown in the *Gateway nodes*, *Agent status*, and *Connection status* for the clusters:

- Gateway nodes labeled
    - **Progressing**: The process to label the gateway nodes started.
    - **Nodes not labeled**: The gateway nodes are not labeled, possibly because the process to label them has not completed.
    - **Nodes not labeled**: The gateway nodes are not yet labeled, possibly because the process is waiting for another process to finish.
    - Nodes labeled: The gateway nodes have been labeled.
- Agent status

- Progressing: The installation of the Submariner agent started.
- Degraded: The Submariner agent is not running correctly, possibly because it is still in progress.

- Connection status
  - Progressing: The process to establish a connection with the Submariner add-on started.
  - Degraded: The connection is not ready. If you just installed the add-on, the process might still be in progress. If it was after the connection has already been established and running, then two clusters have lost the connection to each other. When there are multiple clusters, all clusters display a **Degraded** status if any of the clusters is in adisconnected state.

It will also show which clusters are connected, and which ones are disconnected.

1.32.2. Resolving the problem: Submariner add-on status is degraded

- The degraded status often resolves itself as the processes complete. You can see the current step of the process by clicking the status in the table. You can use that information to determine whether the process is finished and you need to take other troubleshooting steps.
- For an issue that does not resolve itself, complete the following steps to troubleshoot the problem:

1. You can use the **diagnose** command with the **subctl** utility to run some tests on the Submariner connections when the following conditions exist:
   a. The *Agent status* or *Connection status* is in a **Degraded** state. The **diagnose** command provides detailed analysis about the issue.
   b. Everything is green in console, but the networking connections are not working correctly. The **diagnose** command helps to confirm that there are no other connectivity or deployment issues outside of the console. It is considered best practice to run the **diagnostics** command after any deployment to identify issues.
   See **diagnose** in the Submariner for more information about how to run the command.
2. If a problem continues with the **Connection status**, you can start by running the **diagnose** command of the **subctl** utility tool to get a more detailed status for the connection between two Submariner clusters. The format for the command is:

   ```
   subctl diagnose all --kubeconfig <path-to-kubeconfig-file>
   ```

   Replace **path-to-kubeconfig-file** with the path to the **kubeconfig** file. See **diagnose** in the Submariner documentation for more information about the command.
3. Check the firewall settings. Sometimes a problem with the connection is caused by firewall permissions issues that prevent the clusters from communicating. This can cause the **Connection status** to show as degraded. Run the following command to check the firewall issues:

   ```
   subctl diagnose firewall inter-cluster <path-to-local-kubeconfig> <path-to-remote-cluster-kubeconfig>
   ```

   Replace **path-to-local-kubeconfig** with the path to the **kubeconfig** file of one of the clusters.

Replace **path-to-remote-kubeconfig** with the path to the **kubeconfig** file of the other cluster. you can run the **verify** command with your **subctl** utility tool to test the connection between two Submariner clusters. The basic format for the command is:

4. If a problem continues with the **Connection status**, you can run the **verify** command with your **subctl** utility tool to test the connection between two Submariner clusters. The basic format for the command is:

```
subctl verify --kubecontexts <cluster1>,<cluster2> [flags]
```

Replace **cluster1** and **cluster2** with the names of the clusters that you are testing. See **verify** in the Submariner documentation for more information about the command.

5. After the troubleshooting steps resolve the issue, use the **benchmark** command with the **subctl** tool to establish a base on which to compare when you run additional diagnostics. See **benchmark** in the Submariner documentation for additional information about the options for the command.

1.33. TROUBLESHOOTING RESTORE STATUS FINISHES WITH ERRORS

After you restore a backup, resources are restored correctly but the Red Hat Advanced Cluster Management restore resource shows a **FinishedWithErrors** status.

1.33.1. Symptom: Troubleshooting restore status finishes with errors

Red Hat Advanced Cluster Management shows a **FinishedWithErrors** status and one or more of the Velero restore resources created by the Red Hat Advanced Cluster Management restore show a **PartiallyFailed** status.

1.33.2. Resolving the problem: Troubleshooting restore status finishes with errors

If you restore from a backup that is empty, you can safely ignore the **FinishedWithErrors** status.

Red Hat Advanced Cluster Management for Kubernetes restore shows a cumulative status for all Velero restore resources. If one status is **PartiallyFailed** and the others are **Completed**, the cumulative status you see is **PartiallyFailed** to notify you that there is at least one issue.

To resolve the issue, check the status for all individual Velero restore resources with a **PartiallyFailed** status and view the logs for more details. You can get the log from the object storage directly, or download it from the OADP Operator by using the **DownloadRequest** custom resource.

To create a **DownloadRequest** from the console, complete the following steps:

1. Navigate to **Operators** > **Installed Operators** > **Create DownloadRequest**
2. Select **BackupLog** as your **Kind** and follow the console instructions to complete the **DownloadRequest** creation.

1.34. TROUBLESHOOTING MULTILINE YAML PARSING

When you want to use the **fromSecret** function to add contents of a **Secret** resource into a **Route** resource, the contents are displayed incorrectly.

1.34.1. Symptom: Troubleshooting multiline YAML parsing

When the managed cluster and hub cluster are the same cluster the certificate data is redacted, so the contents are not parsed as a template JSON string. You might receive the following error messages:

```
message: >-
        [spec.tls.caCertificate: Invalid value: "redacted ca certificate
        data": failed to parse CA certificate: data does not contain any
        valid RSA or ECDSA certificates, spec.tls.certificate: Invalid
        value: "redacted certificate data": data does not contain any valid
        RSA or ECDSA certificates, spec.tls.key: Invalid value: "": no key specified]
```

1.34.2. Resolving the problem: Troubleshooting multiline YAML parsing

Configure your certificate policy to retrieve the hub cluster and managed cluster **fromSecret** values. Use the **autoindent** function to update your certificate policy with the following content:

```
            tls:
              certificate: |
                {{ print "{{hub fromSecret "open-cluster-management" "minio-cert" "tls.crt" hub}}" |
base64dec | autoindent }}
```