OpenShift Container Platform 4.1

Disaster recovery

Recovering your OpenShift Container Platform 4.1 cluster

Last Updated: 2019-08-28

OpenShift Container Platform 4.1 Disaster recovery

Recovering your OpenShift Container Platform 4.1 cluster

Legal Notice

Abstract

This document provides instructions for how to recover your OpenShift Container Platform 4.1 cluster from various disaster scenarios.

Table of Contents

CHAPTER 1. BACKING UP ETCD

1.1. BACKING UP ETCD DATA

Follow these steps to back up etcd data by creating a snapshot. This snapshot can be saved and used at a later time if you need to restore etcd.

Prerequisites

- SSH access to a master host.

Procedure

1. Access a master host as the root user.
2. Run the **etcd-snapshot-backup.sh** script and pass in the location to save the etcd snapshot to.

   ```
   $ sudo /usr/local/bin/etcd-snapshot-backup.sh ./assets/backup/snapshot.db
   ```

   In this example, the snapshot is saved to **./assets/backup/snapshot.db** on the master host.

CHAPTER 2. RECOVERING FROM LOST MASTER HOSTS

This document describes the process to recover from a complete loss of a master host. This includes situations where a majority of master hosts have been lost, leading to etcd quorum loss and the cluster going offline.

At a high level, the procedure is to:

1. Restore etcd quorum on a remaining master host.
2. Create new master hosts.
3. Correct DNS and load balancer entries.
4. Grow etcd to full membership.

If the majority of master hosts have been lost, you will need a backed up etcd snapshot to restore etcd quorum on the remaining master host.

2.1. RECOVERING FROM LOST MASTER HOSTS

Follow these steps to recover from the loss of one or more master hosts.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- SSH access to a remaining master host.
- A backed up etcd snapshot, if you are recovering a loss of a majority of masters.

Procedure

1. Restore etcd quorum on the remaining master.

NOTE

This step is only necessary if you have had a majority of your masters fail. You can skip this step if you have a majority of your masters still available.

   a. Copy the etcd snapshot file to the remaining master host.
   This procedure assumes that you have copied a snapshot file called **snapshot.db** to the **/home/core/** directory of your master host.

   b. Access the remaining master host.

   c. Set the **INITIAL_CLUSTER** variable to the list of members in the format of **<name>=<url>**. This variable will be passed to the restore script, and in this procedure, it is assumed that there is only a single member at this time.

```
[core@ip-10-0-143-125 ~]$ export INITIAL_CLUSTER="etcd-member-ip-10-0-143-
125.ec2.internal=https://etcd-0.clustername.devcluster.openshift.com:2380"
```

   d. Run the **etcd-snapshot-restore.sh** script.

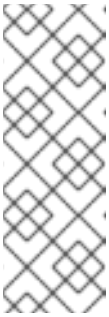
Pass in two parameters to the **etcd-snapshot-restore.sh** script: the path to the backed up etcd snapshot file and list of members, which is defined by the **INITIAL_CLUSTER** variable.

```
[core@ip-10-0-143-125 ~]$ sudo /usr/local/bin/etcd-snapshot-restore.sh
/home/core/snapshot.db $INITIAL_CLUSTER
Creating asset directory ./assets
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
Backing up /etc/kubernetes/manifests/etcd-member.yaml to ./assets/backup/
Stopping all static pods..
..stopping kube-scheduler-pod.yaml
..stopping kube-controller-manager-pod.yaml
..stopping kube-apiserver-pod.yaml
..stopping etcd-member.yaml
Stopping etcd..
Waiting for etcd-member to stop
Stopping kubelet..
Stopping all containers..
bd44e4bc942276eb1a6d4b48ecd9f5fe95570f54aa9c6b16939fa2d9b679e1ea
d88defb9da5ae623592b81619e3690faeb4fa645440e71c029812cb960ff586f
3920ced20723064a379739c4a586f909497a7b6705a5b3cf367d9b930f23a5f1
d470f7a2d962c90f3a21bcc021970bde96bc8908f317ec70f1c21720b322c25c
Backing up etcd data-dir..
Removing etcd data-dir /var/lib/etcd
Restoring etcd member etcd-member-ip-10-0-143-125.ec2.internal from snapshot..
2019-05-15 19:03:34.647589 I | pkg/netutil: resolving etcd-
0.clustername.devcluster.openshift.com:2380 to 10.0.143.125:2380
2019-05-15 19:03:34.883545 I | mvcc: restore compact to 361491
2019-05-15 19:03:34.915679 I | etcdserver/membership: added member
cbe982c74cbb42f [https://etcd-0.clustername.devcluster.openshift.com:2380] to cluster
807ae3bffc8d69ca
Starting static pods..
..starting kube-scheduler-pod.yaml
..starting kube-controller-manager-pod.yaml
..starting kube-apiserver-pod.yaml
..starting etcd-member.yaml
Starting kubelet..
```

Once the **etcd-snapshot-restore.sh** script completes, your cluster should now have a single member etcd cluster, and API services will begin restarting. This might take up to 15 minutes.

In a terminal that has access to the cluster, run the following command to verify that it is ready:

```
$ oc get nodes -l node-role.kubernetes.io/master
NAME                                         STATUS   ROLES    AGE   VERSION
ip-10-0-143-125.us-east-2.compute.internal   Ready    master   46m
v1.13.4+db7b699c3
```



NOTE

Be sure that all old etcd members being replaced are shut down. Otherwise, they might try to connect to the new cluster and will report errors like the following in the logs:

```
2019-05-20 15:33:17.648445 E | rafthttp: request cluster ID mismatch (got
9f5f9f05e4d43b7f want 807ae3bffc8d69ca)
```

2. Create new master hosts.
   If your cluster has its Machine API enabled and functional, then when the OpenShift **machine-api** Operator is restored, it will create the new masters. If you do not have the **machine-api** Operator enabled, you must create new masters using the same methods that were used to originally create them.

   You will also need to approve the certificates signing requests (CSRs) for these new master hosts. Two pending CSRs are generated for each machine that was added to the cluster.

   a. In a terminal that has access to the cluster, run the following commands to approve the CSRs:

      i. Get the list of current CSRs.

      ```
      $ oc get csr
      ```

      ii. Review the details of a CSR to verify it is valid.

      ```
      $ oc describe csr <csr_name> 1
      ```

      1 **<csr_name>** is the name of a CSR from the list of current CSRs.

      iii. Approve each valid CSR.

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

      Be sure to approve both the pending client and server CSR for each master that was added to the cluster.

   b. In a terminal that has access to the cluster, run the following command to verify that your masters are ready:

   ```
   $ oc get nodes -l node-role.kubernetes.io/master
   NAME                                         STATUS   ROLES    AGE   VERSION
   ip-10-0-143-125.us-east-2.compute.internal   Ready    master   50m
   v1.13.4+db7b699c3
   ip-10-0-156-255.us-east-2.compute.internal   Ready    master   92s   v1.13.4+db7b699c3
   ip-10-0-162-178.us-east-2.compute.internal   Ready    master   70s   v1.13.4+db7b699c3
   ```

3. Correct the DNS entries.

   a. From the AWS console, review the etcd-0, etcd-1, and etcd-2 Route 53 records in the private DNS zone, and if necessary, update the value to the appropriate new private IP address. See Editing Records in the AWS documentation for instructions.

You can obtain the private IP address of an instance by running the following command in a terminal that has access to the cluster.

```
$ oc get node ip-10-0-143-125.us-east-2.compute.internal -o
jsonpath='{.status.addresses[?(@.type=="InternalIP")].address}{"\n"}'
10.0.143.125
```

4. Update load balancer entries.
   If you are using a cluster-managed load balancer, the entries will automatically be updated for you. If you are not, be sure to update your load balancer with the current addresses of your master hosts.

   If your load balancing is managed by AWS, see Register or Deregister Targets by IP Address in the AWS documentation for instructions on updating load balancer entries.

5. Grow etcd to full membership.

   a. Set up a temporary etcd certificate signer service on your master where you have restored etcd.

      i. Access the original master, and log in to your cluster as a **cluster-admin** user using the following command.

```
[core@ip-10-0-143-125 ~]$ oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
Password:
Login successful.
```

      ii. Obtain the pull specification for the **kube-etcd-signer-server** image.

```
[core@ip-10-0-143-125 ~]$ export KUBE_ETCD_SIGNER_SERVER=$(sudo oc
adm release info --image-for kube-etcd-signer-server --registry-
config=/var/lib/kubelet/config.json)
```

      iii. Run the **tokenize-signer.sh** script.
      Be sure to pass in the **-E** flag to **sudo** so that environment variables are properly passed to the script.

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/tokenize-signer.sh ip-10-0-143-125
(1)
Populating template /usr/local/share/openshift-recovery/template/kube-etcd-cert-
signer.yaml.template
Populating template ./assets/tmp/kube-etcd-cert-signer.yaml.stage1
Tokenized template now ready: ./assets/manifests/kube-etcd-cert-signer.yaml
```

      (1) The host name of the original master you just restored, where the signer should be deployed.

      iv. Create the signer Pod using the file that was generated.

```
[core@ip-10-0-143-125 ~]$ oc create -f assets/manifests/kube-etcd-cert-signer.yaml
pod/etcd-signer created
```

v. Verify that the signer is listening on this master node.

```
[core@ip-10-0-143-125 ~]$ ss -ltn | grep 9943
LISTEN   0         128                       *:9943                    *:*
```

b. Add the new master hosts to the etcd cluster.

i. Access one of the new master hosts, and log in to your cluster as a **cluster-admin** user using the following command.

```
[core@ip-10-0-156-255 ~]$ oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
Password:
Login successful.
```

ii. Export two environment variables that are required by the **etcd-member-recover.sh** script.

```
[core@ip-10-0-156-255 ~]$ export SETUP_ETCD_ENVIRONMENT=$(sudo oc adm
release info --image-for setup-etcd-environment --registry-
config=/var/lib/kubelet/config.json)
```

```
[core@ip-10-0-156-255 ~]$ export KUBE_CLIENT_AGENT=$(sudo oc adm release
info --image-for kube-client-agent --registry-config=/var/lib/kubelet/config.json)
```

iii. Run the **etcd-member-recover.sh** script.
Be sure to pass in the **-E** flag to **sudo** so that environment variables are properly passed to the script.

```
[core@ip-10-0-156-255 ~]$ sudo -E /usr/local/bin/etcd-member-recover.sh
10.0.143.125 (1)
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
etcd-member.yaml found in ./assets/backup/
etcd.conf backup upready exists ./assets/backup/etcd.conf
Trying to backup etcd client certs..
etcd client certs already backed up and available ./assets/backup/
Stopping etcd..
Waiting for etcd-member to stop
etcd data-dir backup found ./assets/backup/etcd..
etcd TLS certificate backups found in ./assets/backup..
Removing etcd certs..
Populating template /usr/local/share/openshift-recovery/template/etcd-generate-
certs.yaml.template
Populating template ./assets/tmp/etcd-generate-certs.stage1
Populating template ./assets/tmp/etcd-generate-certs.stage2
Starting etcd client cert recovery agent..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Stopping cert recover..
```

```
Waiting for generate-certs to stop
Patching etcd-member manifest..
Updating etcd membership..
Member 249a4b9a790b3719 added to cluster 807ae3bffc8d69ca

ETCD_NAME="etcd-member-ip-10-0-156-255.ec2.internal"
ETCD_INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-
0.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-156-
255.ec2.internal=https://etcd-1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://etcd-
1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"
Starting etcd..
```

1 The IP address of the original master, where the signer server is running.

iv. Verify that the new master host has been added to the etcd member list.

A. Access the original master and connect to the running etcd container.

```
[core@ip-10-0-143-125 ~] id=$(sudo crictl ps --name etcd-member | awk
'FNR==2{ print $1}') && sudo crictl exec -it $id /bin/sh
```

B. In the etcd container, export variables needed for connecting to etcd.

```
sh-4.2# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find
/etc/ssl/ -name *peer*key)
```

C. In the etcd container, execute **etcdctl member list** and verify that the new member is listed.

```
sh-4.2#  etcdctl member list -w table

+------------------+---------+----------------------------------------+----------------------
----------------------------------------+--------------------------+
|        ID        | STATUS  |                NAME                    |                      PEER
ADDRS                      |      CLIENT ADDRS        |
+------------------+---------+----------------------------------------+----------------------
----------------------------------------+--------------------------+
|  cbe982c74cbb42f | started |  etcd-member-ip-10-0-156-255.ec2.internal |
https://etcd-0.clustername.devcluster.openshift.com:2380 |
https://10.0.156.255:2379 |
| 249a4b9a790b3719 | started | etcd-member-ip-10-0-143-125.ec2.internal |
https://etcd-1.clustername.devcluster.openshift.com:2380 |
https://10.0.143.125:2379 |
+------------------+---------+----------------------------------------+----------------------
----------------------------------------+--------------------------+
```

Note that it may take up to 10 minutes for the new member to start.

v. Repeat these steps to add your other new master host until you have achieved full etcd membership.

c. After all members are restored, remove the signer Pod because it is no longer needed. In a terminal that has access to the cluster, run the following command:

```
$ oc delete pod -n openshift-config etcd-signer
```

CHAPTER 3. RESTORING BACK TO A PREVIOUS CLUSTER STATE

In order to restore the cluster to a previous state, you must have previously backed up etcd data by creating a snapshot. You will use this snapshot to restore the cluster state.

3.1. RESTORING BACK TO A PREVIOUS CLUSTER STATE

You can use a saved etcd snapshot to restore back to a previous cluster state.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- SSH access to master hosts.
- A backed up etcd snapshot.

Procedure

1. Prepare each master host in your cluster to be restored.
   You should run the restore script on all of your master hosts within a short period of time so that the cluster members come up at about the same time and form a quorum. For this reason, it is recommended to stage each master host in a separate terminal, so that the restore script can then be started quickly on each.

   a. Copy the etcd snapshot file to a master host.
      This procedure assumes that you have copied a snapshot file called **snapshot.db** to the /**home**/**core**/ directory of your master host.

   b. Access the master host.

   c. Set the **INITIAL_CLUSTER** variable to the list of members in the format of **<name>=<url>**. This variable will be passed to the restore script and must be exactly the same for each member.

      ```
      [core@ip-10-0-143-125 ~]$ export INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-0.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-35-108.ec2.internal=https://etcd-1.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-10-16.ec2.internal=https://etcd-2.clustername.devcluster.openshift.com:2380"
      ```

   d. Repeat these steps on your other master hosts, each in a separate terminal.

2. Run the restore script on all of your master hosts.

   a. Start the **etcd-snapshot-restore.sh** script on your first master host. Pass in two parameters: the path to the snapshot file and list of members, which is defined by the **INITIAL_CLUSTER** variable.

      ```
      [core@ip-10-0-143-125 ~]$ sudo /usr/local/bin/etcd-snapshot-restore.sh /home/core/snapshot.db $INITIAL_CLUSTER
      Creating asset directory ./assets
      Downloading etcdctl binary..
      ```

```
etcdctl version: 3.3.10
API version: 3.3
Backing up /etc/kubernetes/manifests/etcd-member.yaml to ./assets/backup/
Stopping all static pods..
..stopping kube-scheduler-pod.yaml
..stopping kube-controller-manager-pod.yaml
..stopping kube-apiserver-pod.yaml
..stopping etcd-member.yaml
Stopping etcd..
Waiting for etcd-member to stop
Stopping kubelet..
Stopping all containers..
bd44e4bc942276eb1a6d4b48ecd9f5fe95570f54aa9c6b16939fa2d9b679e1ea
d88defb9da5ae623592b81619e3690faeb4fa645440e71c029812cb960ff586f
3920ced20723064a379739c4a586f909497a7b6705a5b3cf367d9b930f23a5f1
d470f7a2d962c90f3a21bcc021970bde96bc8908f317ec70f1c21720b322c25c
Backing up etcd data-dir..
Removing etcd data-dir /var/lib/etcd
Restoring etcd member etcd-member-ip-10-0-143-125.ec2.internal from snapshot..
2019-05-15 19:03:34.647589 I | pkg/netutil: resolving etcd-
0.clustername.devcluster.openshift.com:2380 to 10.0.143.125:2380
2019-05-15 19:03:34.883545 I | mvcc: restore compact to 361491
2019-05-15 19:03:34.915679 I | etcdserver/membership: added member
cbe982c74cbb42f [https://etcd-0.clustername.devcluster.openshift.com:2380] to cluster
807ae3bffc8d69ca
Starting static pods..
..starting kube-scheduler-pod.yaml
..starting kube-controller-manager-pod.yaml
..starting kube-apiserver-pod.yaml
..starting etcd-member.yaml
Starting kubelet..
```

b. Once the restore starts, run the script on your other master hosts.

3. Verify that the Machine Configs have been applied.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command.

```
$ oc get machineconfigpool
NAME     CONFIG                                          UPDATED   UPDATING
master   rendered-master-50e7e00374e80b767fcc922bdfbc522b   True      False
```

When the snapshot has been applied, the **currentConfig** of the master will match the ID from when the etcd snapshot was taken. The **currentConfig** name for masters is in the format **rendered-master-<currentConfig>**.

4. Verify that all master hosts have started and joined the cluster.

a. Access a master host and connect to the running etcd container.

```
[core@ip-10-0-143-125 ~] id=$(sudo crictl ps --name etcd-member | awk 'FNR==2{ print
$1}') && sudo crictl exec -it $id /bin/sh
```

b. In the etcd container, export variables needed for connecting to etcd.

```
sh-4.2# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find /etc/ssl/ -
name *peer*key)
```

c. In the etcd container, execute **etcdctl member list** and verify that the three members show as started.

```
sh-4.2#  etcdctl member list -w table

+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
|        ID        | STATUS  |                NAME                     |                        PEER ADDRS
|      CLIENT ADDRS        |
+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
| 29e461db6be4eaaa | started | etcd-member-ip-10-0-164-170.ec2.internal | https://etcd-
2.clustername.devcluster.openshift.com:2380 | https://10.0.164.170:2379 |
|  cbe982c74cbb42f | started | etcd-member-ip-10-0-143-125.ec2.internal | https://etcd-
0.clustername.devcluster.openshift.com:2380 | https://10.0.143.125:2379 |
| a752f80bcb0da3e8 | started |   etcd-member-ip-10-0-156-2.ec2.internal | https://etcd-
1.clustername.devcluster.openshift.com:2380 |   https://10.0.156.2:2379 |
+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
```

Note that it may take up to 10 minutes for each new member to start.

CHAPTER 4. RECOVERING FROM EXPIRED CONTROL PLANE CERTIFICATES

4.1. RECOVERING FROM EXPIRED CONTROL PLANE CERTIFICATES

Follow this procedure to recover from a situation where your control plane certificates have expired.

Prerequisites

- SSH access to master hosts.

Procedure

1. Access a master host with an expired certificate as the root user.
2. Obtain the **cluster-kube-apiserver-operator** image reference for a release.

```
# RELEASE_IMAGE=<release_image> 1
```

1 An example value for **<release_image>** is **quay.io/openshift-release-dev/ocp-release:4.1.0**.

```
# KAO_IMAGE=$( oc adm release info --registry-config='/var/lib/kubelet/config.json'
"${RELEASE_IMAGE}" --image-for=cluster-kube-apiserver-operator )
```

3. Pull the **cluster-kube-apiserver-operator** image.

```
# podman pull --authfile=/var/lib/kubelet/config.json "${KAO_IMAGE}"
```

4. Create a recovery API server.

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" recovery-apiserver
create
```

5. Run the **export KUBECONFIG** command from the output of the above command, which is needed for the **oc** commands later in this procedure.

```
# export KUBECONFIG=/<path_to_recovery_kubeconfig>/admin.kubeconfig
```

6. Wait for the recovery API server to come up.

```
# until oc get namespace kube-system 2>/dev/null 1>&2; do echo 'Waiting for recovery
apiserver to come up.'; sleep 1; done
```

7. Run the **regenerate-certificates** command. It fixes the certificates in the API, overwrites the old certificates on the local drive, and restarts static Pods to pick them up.

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" regenerate-certificates
```

8. After the certificates are fixed in the API, use the following commands to force new rollouts for the control plane. It will reinstall itself on the other nodes because the kubelet is connected to API servers using an internal load balancer.

```
# oc patch kubeapiserver cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$(
date --rfc-3339=ns )"'"}}' --type=merge
```

```
# oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

```
# oc patch kubescheduler cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$(
date --rfc-3339=ns )"'"}}' --type=merge
```

9. Create a bootstrap kubeconfig with a valid user.

   a. Run the **recover-kubeconfig.sh** script and save the output to a file called **kubeconfig**.

   ```
   # recover-kubeconfig.sh > kubeconfig
   ```

   b. Copy the **kubeconfig** file to all master hosts and move it to **/etc/kubernetes/kubeconfig**.

   c. Get the CA certificate used to validate connections from the API server.

   ```
   # oc get configmap kube-apiserver-to-kubelet-client-ca -n openshift-kube-apiserver-
   operator --template='{{ index .data "ca-bundle.crt" }}' > /etc/kubernetes/ca.crt
   ```

   d. Copy the **/etc/kubernetes/ca.crt** file to all other master hosts and nodes.

10. Recover the kubelet on all masters.

    a. On a master host, stop the kubelet.

    ```
    # systemctl stop kubelet
    ```

    b. Delete stale kubelet data.

    ```
    # rm -rf /var/lib/kubelet/pki /var/lib/kubelet/kubeconfig
    ```

    c. Restart the kubelet.

    ```
    # systemctl start kubelet
    ```

    d. Repeat these steps on all other master hosts.

11. If necessary, recover the kubelet on the worker nodes.
    After the master nodes are restored, the worker nodes might restore themselves. You can verify this by running the **oc get nodes** command. If the worker nodes are not listed, then perform the following steps on each worker node.

    a. Stop the kubelet.

    ```
    # systemctl stop kubelet
    ```

b. Delete stale kubelet data.

```
# rm -rf /var/lib/kubelet/pki /var/lib/kubelet/kubeconfig
```

c. Restart the kubelet.

```
# systemctl start kubelet
```

12. Approve the pending **node-bootstrapper** certificates signing requests (CSRs).

a. Get the list of current CSRs.

```
# oc get csr
```

b. Review the details of a CSR to verify it is valid.

```
# oc describe csr <csr_name> 1
```

1 **<csr_name>** is the name of a CSR from the list of current CSRs.

c. Approve each valid CSR.

```
# oc adm certificate approve <csr_name>
```

Be sure to approve all pending **node-bootstrapper** CSRs.

13. Destroy the recovery API server because it is no longer needed.

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" recovery-apiserver
destroy
```

Wait for the control plane to restart and pick up the new certificates. This might take up to 10 minutes.