OpenShift Container Platform 4.3

Backup and restore

Backing up and restoring your OpenShift Container Platform cluster

Last Updated: 2020-10-22

OpenShift Container Platform 4.3 Backup and restore

Backing up and restoring your OpenShift Container Platform cluster

Legal Notice

Abstract

This document provides instructions for backing up your cluster's data and for recovering from various disaster scenarios.

Table of Contents

CHAPTER 1. BACKING UP ETCD . . . 3

1.1. BACKING UP ETCD DATA 3

CHAPTER 2. REPLACING A FAILED MASTER HOST . . . 5

2.1. REMOVING A FAILED MASTER HOST FROM THE ETCD CLUSTER 5

2.2. ADDING THE MEMBER BACK TO THE CLUSTER 6

2.2.1. Adding a master host back to the etcd cluster 6

2.2.2. Generating etcd certificates and adding the member to the cluster 8

CHAPTER 3. DISASTER RECOVERY . . . 12

3.1. ABOUT DISASTER RECOVERY 12

3.2. RECOVERING FROM LOST MASTER HOSTS 12

3.2.1. Recovering from lost master hosts 13

3.3. RESTORING TO A PREVIOUS CLUSTER STATE 19

3.3.1. Restoring to a previous cluster state 19

3.4. RECOVERING FROM EXPIRED CONTROL PLANE CERTIFICATES 22

3.4.1. Recovering from expired control plane certificates 22

CHAPTER 1. BACKING UP ETCD

etcd is the key-value store for OpenShift Container Platform, which persists the state of all resource objects.

Back up your cluster's etcd data regularly and store in a secure location ideally outside the OpenShift Container Platform environment. Do not take an etcd backup before the first certificate rotation completes, which occurs 24 hours after installation, otherwise the backup will contain expired certificates. It is also recommended to take etcd backups during non-peak usage hours, as it is a blocking action.

Once you have an etcd backup, you can recover from lost master hosts and restore to a previous cluster state.

You can perform the etcd data backup process on any master host that has connectivity to the etcd cluster, where the proper certificates are provided.

1.1. BACKING UP ETCD DATA

Follow these steps to back up etcd data by creating an etcd snapshot and backing up static Kubernetes API server resources. This backup can be saved and used at a later time if you need to restore etcd.

You should only save a backup from a single master host. You do not need a backup from each master host in the cluster.

NOTE

If you are taking an etcd backup on OpenShift Container Platform 4.3.0 or 4.3.1, then this procedure generates a single file that contains the etcd snapshot and static Kubernetes API server resources. When restoring, the **etcd-snapshot-restore.sh** script is backward compatible to accept this single file.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have checked whether the cluster-wide proxy is enabled.

  ### TIP

  You can check whether the proxy is enabled by reviewing the output of **oc get proxy cluster -o yaml**. The proxy is enabled if the **httpProxy**, **httpsProxy**, and **noProxy** fields have values set.

Procedure

1. Start a debug session for a master node:

   ```
   $ oc debug node/<node_name>
   ```

2. Change your root directory to the host:

   ```
   sh-4.2# chroot /host
   ```

3. Change to the /**home/core** directory:

```
sh-4.4# cd /home/core
```

4. If the cluster-wide proxy is enabled, be sure that you have exported the **NO_PROXY**, **HTTP_PROXY**, and **HTTPS_PROXY** environment variables.

5. Run the **etcd-snapshot-backup.sh** script and pass in the location to save the backup to.

```
sh-4.4# /usr/local/bin/etcd-snapshot-backup.sh /home/core/assets/backup
```

In this example, two files are created in the **/home/core/assets/backup/** directory on the master host:

- **snapshot_<datetimestamp>.db**: This file is the etcd snapshot.

- **static_kuberesources_<datetimestamp>.tar.gz**: This file contains the static Kubernetes API server resources. If etcd encryption is enabled, it also contains the encryption keys for the etcd snapshot.

NOTE

If etcd encryption is enabled, it is recommended to store this second file separately from the etcd snapshot for security reasons. However, this file is required in order to restore from the etcd snapshot.

Keep in mind that etcd encryption only encrypts values, not keys. This means that resource types, namespaces, and object names are unencrypted.

CHAPTER 2. REPLACING A FAILED MASTER HOST

This document describes the process to replace a single etcd member. This procedure assumes that there is still an etcd quorum in the cluster.

NOTE

If you have lost the majority of your master hosts, leading to etcd quorum loss, then you must follow the disaster recovery procedure to recover from lost master hosts instead of this procedure.

If the control plane certificates are not valid on the member being replaced, then you must follow the procedure to recover from expired control plane certificates instead of this procedure.

To replace a single master host:

- Remove the member from the etcd cluster .
- If the etcd certificates for the master host are valid, then add the member back to the etcd cluster.
- If there are no etcd certificates for the master host or they are no longer valid, then generate etcd certificates and add the member to the etcd cluster.

2.1. REMOVING A FAILED MASTER HOST FROM THE ETCD CLUSTER

Follow these steps to remove a failed master host from the etcd cluster.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have SSH access to an active master host.

Procedure

1. View the list of Pods associated with etcd.
   In a terminal that has access to the cluster, run the following command:

```
$ oc get pods -n openshift-etcd
NAME                                               READY   STATUS    RESTARTS   AGE
etcd-member-ip-10-0-128-73.us-east-2.compute.internal    2/2     Running   0          15h
etcd-member-ip-10-0-147-172.us-east-2.compute.internal   2/2     Running   7          122m
etcd-member-ip-10-0-171-108.us-east-2.compute.internal   2/2     Running   0          15h
```

2. Access an active master host.

3. Run the **etcd-member-remove.sh** script and pass in the name of the etcd member to remove:

```
[core@ip-10-0-128-73 ~]$ sudo -E /usr/local/bin/etcd-member-remove.sh etcd-member-ip-10-0-147-172.us-east-2.compute.internal
Downloading etcdctl binary..
etcdctl version: 3.3.10
```

```
API version: 3.3
etcd client certs already backed up and available ./assets/backup/
Member 23e4736df4451b32 removed from cluster 6e25bab1bb556673
etcd member etcd-member-ip-10-0-147-172.us-east-2.compute.internal with
23e4736df4451b32 successfully removed..
```

4. Verify that the etcd member has been successfully removed from the cluster:

   a. Connect to the running etcd container:

   ```
   [core@ip-10-0-128-73 ~] 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 the variables needed for connecting to etcd:

   ```
   sh-4.3# 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 removed member is no longer listed:

   ```
   sh-4.3#  etcdctl member list -w table

   +------------------+---------+------------------------------------------------+--------------------------------------
   ------------------------------------+--------------------------------+
   |        ID        | STATUS  |                      NAME                      |                              PEER ADDRS
   |        CLIENT ADDRS        |
   +------------------+---------+------------------------------------------------+--------------------------------------
   ------------------------------------+--------------------------------+
   | 29e461db6be4eaaa | started | etcd-member-ip-10-0-128-73.us-east-2.compute.internal
   | https://etcd-2.clustername.devcluster.openshift.com:2380 | https://10.0.128.73:2379 |
   |  cbe982c74cbb42f | started |  etcd-member-ip-10-0-171-108.us-east-2.compute.internal
   | https://etcd-1.clustername.devcluster.openshift.com:2380 |   https://10.0.171.108:2379 |
   +------------------+---------+------------------------------------------------+--------------------------------------
   ------------------------------------+--------------------------------+
   ```

2.2. ADDING THE MEMBER BACK TO THE CLUSTER

After you have removed the member from the etcd cluster, use one of the following procedures to add the member to the cluster:

- If the etcd certificates for the master host are valid, then add the member back to the etcd cluster.

- If there are no etcd certificates for the master host or they are no longer valid, then generate etcd certificates and add the member to the etcd cluster.

2.2.1. Adding a master host back to the etcd cluster

Follow these steps to add a master host back to the etcd cluster. This procedure assumes that you previously removed the master host from the cluster and that its etcd dependencies, such as TLS certificates and DNS, are valid.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have SSH access to the master host to add to the etcd cluster.
- You have the IP address of an existing active etcd member.

Procedure

1. Access the master host to add to the etcd cluster.

   > **IMPORTANT**
   >
   > You must run this procedure on the master host that is being added to the etcd cluster.

2. Run the **etcd-member-add.sh** script and pass in two parameters:

   - the IP address of an existing etcd member
   - the name of the etcd member to add

```
[core@ip-10-0-147-172 ~]$ sudo -E /usr/local/bin/etcd-member-add.sh \
10.0.128.73 \ 1
etcd-member-ip-10-0-147-172.us-east-2.compute.internal 2

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
Stopping etcd..
Waiting for etcd-member to stop
etcd data-dir backup found ./assets/backup/etcd..
Updating etcd membership..
Removing etcd data_dir /var/lib/etcd..

ETCD_NAME="etcd-member-ip-10-0-147-172.us-east-2.compute.internal"
ETCD_INITIAL_CLUSTER="etcd-member-ip-10-0-147-172.us-east-
2.compute.internal=https://etcd-1.clustername.devcluster.openshift.com:2380,etcd-member-
ip-10-0-171-108.us-east-2.compute.internal=https://etcd-
2.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-128-73.us-east-
2.compute.internal=https://etcd-0.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://etcd-
1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"'
Member  1e42c7070decd39 added to cluster 6e25bab1bb556673
Starting etcd..
```

1 The IP address of an active etcd member. This is *not* the IP address of the member that you are adding.

2 The name of the etcd member to add.

3. Verify that the etcd member has been successfully added to the etcd cluster:

   a. Connect to the running etcd container:

   ```
   [core@ip-10-0-147-172 ~] 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 the variables needed for connecting to etcd:

   ```
   sh-4.3# 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-4.3#  etcdctl member list -w table

   +------------------+---------+------------------------------------------------+-------------------------------------
   ------------------------------------+-----------------------------+
   |        ID        | STATUS  |                      NAME                      |                       PEER ADDRS
   |       CLIENT ADDRS        |
   +------------------+---------+------------------------------------------------+-------------------------------------
   ------------------------------------+-----------------------------+
   | 29e461db6be4eaaa | started | etcd-member-ip-10-0-128-73.us-east-2.compute.internal
   | https://etcd-2.clustername.devcluster.openshift.com:2380 | https://10.0.128.73:2379 |
   |  cbe982c74cbb42f | started | etcd-member-ip-10-0-147-172.us-east-2.compute.internal |
   https://etcd-0.clustername.devcluster.openshift.com:2380 | https://10.0.147.172:2379 |
   | a752f80bcb0da3e8 | started |   etcd-member-ip-10-0-171-108.us-east-
   2.compute.internal | https://etcd-1.clustername.devcluster.openshift.com:2380 |
   https://10.0.171.108:2379 |
   +------------------+---------+------------------------------------------------+-------------------------------------
   ------------------------------------+-----------------------------+
   ```

   It may take up to 10 minutes for the new member to start.

   d. In the etcd container, execute **etcdctl endpoint health** and verify that the new member is healthy:

   ```
   sh-4.3# etcdctl endpoint health --cluster
   https://10.0.128.73:2379 is healthy: successfully committed proposal: took = 4.5576ms
   https://10.0.147.172:2379 is healthy: successfully committed proposal: took = 5.1521ms
   https://10.0.171.108:2379 is healthy: successfully committed proposal: took = 4.2631ms
   ```

4. Verify that the new member is in the list of Pods associated with etcd and that its status is **Running**.
   In a terminal that has access to the cluster, run the following command:

   ```
   $ oc get pods -n openshift-etcd
   NAME                                                    READY   STATUS    RESTARTS   AGE
   etcd-member-ip-10-0-128-73.us-east-2.compute.internal    2/2     Running   0          15h
   etcd-member-ip-10-0-147-172.us-east-2.compute.internal   2/2     Running   7          122m
   etcd-member-ip-10-0-171-108.us-east-2.compute.internal   2/2     Running   0          15h
   ```

2.2.2. Generating etcd certificates and adding the member to the cluster

If the node is new or the etcd certificates on the node are no longer valid, you must generate the etcd certificates before you can add the member to the etcd cluster.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have SSH access to the new master host to add to the etcd cluster.
- You have SSH access to the one of the healthy master hosts.
- You have the IP address of one of the healthy master hosts.

Procedure

1. Set up a temporary etcd certificate signer service on one of the healthy master nodes.

    a. Access one of the healthy master nodes and log in to your cluster as a **cluster-admin** user using the following command.

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

    b. 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)
    ```

    c. 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 ❶
    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
    ```

    ❶ The host name of the healthy master, where the signer should be deployed.

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

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

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

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

2. Add the new master host to the etcd cluster.

   a. Access the new master host to be added to the cluster, and log in to your cluster as a **cluster-admin** user using the following command.

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

   b. 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 machine-config-operator --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)
   ```

   c. 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
   etcd-member-ip-10-0-156-255.ec2.internal 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 Specify both the IP address of the healthy master where the signer server is running, and the etcd name of the new member.

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

i. Access the healthy 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
```

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

```
sh-4.3# 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)
```

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

```
sh-4.3#  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
|
+------------------+---------+------------------------------------------+---------------------------
------------------------------------+---------------------------+
```

It may take up to 20 minutes for the new member to start.

3. After the new member is added, 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. DISASTER RECOVERY

3.1. ABOUT DISASTER RECOVERY

The disaster recovery documentation provides information for administrators on how to recover from several disaster situations that might occur with their OpenShift Container Platform cluster. As an administrator, you might need to follow one or more of the following procedures in order to return your cluster to a working state.

Recovering from lost master hosts

This solution handles situations where you have lost the majority of your master hosts, leading to etcd quorum loss and the cluster going offline. As long as you have taken an etcd backup and have at least one remaining healthy master host, you can follow this procedure to recover your cluster.
If applicable, you might also need to recover from expired control plane certificates .

> NOTE
>
> If you have a majority of your masters still available and have an etcd quorum, then follow the procedure to replace a single failed master host .

Restoring to a previous cluster state

This solution handles situations where you want to restore your cluster to a previous state, for example, if an administrator deletes something critical. As long as you have taken an etcd backup, you can follow this procedure to restore your cluster to a previous state.
If applicable, you might also need to recover from expired control plane certificates .

Recovering from expired control plane certificates

This solution handles situations where your control plane certificates have expired. For example, if you shut down your cluster before the first certificate rotation, which occurs 24 hours after installation, your certificates will not be rotated and will expire. You can follow this procedure to recover from expired control plane certificates.

3.2. 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. This procedure assumes that you have at least one healthy master host.

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 an etcd backup to restore etcd quorum on the remaining master host.

NOTE

If you have a majority of your masters still available and have an etcd quorum, then follow the procedure to replace a single failed master host.

3.2.1. Recovering from lost master hosts

Follow these steps to recover from the loss of the majority of master hosts, leading to etcd quorum loss.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- SSH access to a remaining master host.
- A backup directory containing both the etcd snapshot and static Kubernetes API server resources taken from the same backup. The file names in the directory must be in the following formats: **snapshot_<datetimestamp>.db** and **static_kuberesources_<datetimestamp>.tar.gz**.

  NOTE

  If the etcd backup was taken from OpenShift Container Platform 4.3.0 or 4.3.1, then it is a single file that contains the etcd snapshot and static Kubernetes API server resources. The **etcd-snapshot-restore.sh** script is backward compatible to accept this single file, which must be in the format of **snapshot_db_kuberesources_<datetimestamp>.tar.gz**.

Procedure

1. Restore etcd quorum on the remaining master.

   a. Copy the etcd backup directory to the remaining master host.
   This procedure assumes that you copied the **backup** directory containing the etcd snapshot and static Kubernetes API server resources 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. If the cluster-wide proxy is enabled, be sure that you have exported the **NO_PROXY**, **HTTP_PROXY**, and **HTTPS_PROXY** environment variables.

   TIP

   You can check whether the proxy is enabled by reviewing the output of **oc get proxy cluster -o yaml**. The proxy is enabled if the **httpProxy**, **httpsProxy**, and **noProxy** fields have values set.

e. Run the **etcd-snapshot-restore.sh** script.
   Pass in two parameters to the **etcd-snapshot-restore.sh** script: the path to the etcd backup directory and list of members, which is defined by the **INITIAL_CLUSTER** variable.

   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/etcd-snapshot-restore.sh
/home/core/backup $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   v1.16.2
```

> **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> ❶
      ```

      ❶ **<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   v1.16.2
   ip-10-0-156-255.us-east-2.compute.internal  Ready    master   92s   v1.16.2
   ip-10-0-162-178.us-east-2.compute.internal  Ready    master   70s   v1.16.2
   ```

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 ~]$ sudo 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 ~]$ sudo 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 ~]$ sudo 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 machine-config-operator --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 etcd-member-ip-10-0-156-255.ec2.internal 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 Specify both the IP address of the original master where the signer server is running, and the etcd name of the new member.

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-4.3# 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-4.3#  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 |
+------------------+---------+------------------------------------------+------------------------
-----------------------------------------+---------------------------+
```

It may take up to 20 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
```

Note that it might take several minutes after completing this procedure for all services to be restored. For example, authentication by using **oc login** might not immediately work until the OAuth server Pods are restarted.

3.3. RESTORING TO A PREVIOUS CLUSTER STATE

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.

3.3.1. Restoring to a previous cluster state

You can use a saved etcd backup 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 backup directory containing both the etcd snapshot and static Kubernetes API server resources taken from the same backup. The file names in the directory must be in the following formats: **snapshot_<datetimestamp>.db** and **static_kuberesources_<datetimestamp>.tar.gz**.

NOTE

You must use the same etcd backup directory on all master hosts in the cluster.

NOTE

If the etcd backup was taken from OpenShift Container Platform 4.3.0 or 4.3.1, then it is a single file that contains the etcd snapshot and static Kubernetes API server resources. The **etcd-snapshot-restore.sh** script is backward compatible to accept this single file, which must be in the format of **snapshot_db_kuberesources_<datetimestamp>.tar.gz**.

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 backup directory to a master host.

This procedure assumes that you copied the **backup** directory containing the etcd snapshot and static Kubernetes API server resources 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. If the cluster-wide proxy is enabled, be sure that you have exported the **NO_PROXY**, **HTTP_PROXY**, and **HTTPS_PROXY** environment variables.

TIP

You can check whether the proxy is enabled by reviewing the output of **oc get proxy cluster -o yaml**. The proxy is enabled if the **httpProxy**, **httpsProxy**, and **noProxy** fields have values set.

e. Repeat these steps on your other master hosts, each in a separate terminal. Be sure to use the backup directory containing the same set of backup files on each master host.

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 etcd backup directory and list of members, which is defined by the **INITIAL_CLUSTER** variable.

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/etcd-snapshot-restore.sh
/home/core/backup $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-4.3# 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-4.3#  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 |
+------------------+---------+------------------------------------------+----------------------------------
---------------------------------+---------------------------+
```

It may take up to 20 minutes for each new member to start.

3.4. RECOVERING FROM EXPIRED CONTROL PLANE CERTIFICATES

3.4.1. 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:4.3.0-x86_64**. See the Repository Tags page for a list of available tags.

   ```
   # 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/kubelet-ca.crt
   ```

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

   e. Add the **machine-config-daemon-force** file to all master hosts and nodes to force the Machine Config Daemon to accept this certificate update.

   ```
   # touch /run/machine-config-daemon-force
   ```

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.