Openshift Master Node Recovery Steps

High Level Steps:

1. Backing up etcd
2. Recovering from lost Master hosts (Restoring to a previous cluster state)

Applies to Version:

1. Openshift V4.X

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.

We back up OS cluster’s etcd data by performing a single invocation of the backup script on a master host. Do not take a backup for each master host.

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

*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.

*Procedure*

* 1. Start a debug session for a master node:

$ oc debug node/<node\_name>

* 1. Change your root directory to the host:

sh-4.2# chroot /host

* 1. If the cluster-wide proxy is enabled, be sure that you have exported the NO\_PROXY, HTTP\_PROXY, and HTTPS\_PROXY environment variables.
  2. Run the cluster-backup.sh script and pass in the location to save the backup to.

**/usr/local/bin/cluster-backup.sh /home/core/assets/backup**

**1bf371f1b5a483927cd01bb593b0e12cff406eb8d7d0acf4ab079c36a0abd3f7**

**etcdctl version: 3.3.18**

**API version: 3.3**

**found latest kube-apiserver-pod: /etc/kubernetes/static-pod-resources/kube-apiserver-pod-7**

**found latest kube-controller-manager-pod: /etc/kubernetes/static-pod-resources/kube-controller-manager-pod-8**

**found latest kube-scheduler-pod: /etc/kubernetes/static-pod-resources/kube-scheduler-pod-6**

**found latest etcd-pod: /etc/kubernetes/static-pod-resources/etcd-pod-2**

**Snapshot saved at /home/core/assets/backup/snapshot\_2020-03-18\_220218.db**

**snapshot db and kube resources are successfully saved to /home/core/assets/backup**

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.

You can use a saved etcd backup to restore back to a previous cluster state. You use the etcd backup to restore a single master host. Then the etcd cluster Operator handles scaling to the remaining master hosts.

*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 the resources for the static Pods, which were 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.

*Procedure*

1. Select a master host to use as the recovery host. This is the host that you will run the restore operation on. Establish SSH connectivity to each of the master nodes, including the recovery host. The Kubernetes API server will become inaccessible once the restore process has started, so you cannot access the master nodes. For this reason, it is recommended to establish SSH connectivity to each master host in a separate terminal. If you do not complete this step, you will not be able to access the master hosts to complete the restore procedure, and you will be unable to recover your cluster from this state.
2. Copy the etcd backup directory to the recovery master host. This procedure assumes that you copied the backup directory containing the etcd snapshot and the resources for the static Pods to the /home/core/ directory of your recovery master host.
3. Stop the static Pods on all other master nodes. It is not required to manually stop the Pods on the recovery host. The recovery script will stop the Pods on the recovery host.
4. Access a master host that is not the recovery host.
5. Move the existing etcd Pod file out of the kubelet manifest directory:

[core@ip-10-0-154-194 ~]$ sudo mv /etc/kubernetes/manifests/etcd-pod.yaml /tmp

Verify that the etcd Pods are stopped.

[core@ip-10-0-154-194 ~]$ sudo crictl ps | grep etcd

The output of this command should be empty. If it is not empty, wait a few minutes and check again.

1. Move the existing Kubernetes API server Pod file out of the kubelet manifest directory:

[core@ip-10-0-154-194 ~]$ sudo mv /etc/kubernetes/manifests/kube-apiserver-pod.yaml /tmp

1. Verify that the Kubernetes API server Pods are stopped.

[core@ip-10-0-154-194 ~]$ sudo crictl ps | grep kube-apiserver

The output of this command should be empty. If it is not empty, wait a few minutes and check again.

1. Move the etcd data directory to a different location:

[core@ip-10-0-154-194 ~]$ sudo mv /var/lib/etcd/ /tmp

1. Repeat this step on each of the other master hosts that is not the recovery host.

Access the recovery master host.

1. Run the restore script on the recovery master host and pass in the path to the etcd backup directory:

[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/cluster-restore.sh /home/core/backup

...stopping kube-scheduler-pod.yaml

...stopping kube-controller-manager-pod.yaml

...stopping etcd-pod.yaml

...stopping kube-apiserver-pod.yaml

Waiting for container etcd to stop

.complete

Waiting for container etcdctl to stop

.............................complete

Waiting for container etcd-metrics to stop

complete

Waiting for container kube-controller-manager to stop

complete

Waiting for container kube-apiserver to stop

..........................................................................................complete

Waiting for container kube-scheduler to stop

complete

Moving etcd data-dir /var/lib/etcd/member to /var/lib/etcd-backup

starting restore-etcd static pod

starting kube-apiserver-pod.yaml

static-pod-resources/kube-apiserver-pod-7/kube-apiserver-pod.yaml

starting kube-controller-manager-pod.yaml

static-pod-resources/kube-controller-manager-pod-7/kube-controller-manager-pod.yaml

starting kube-scheduler-pod.yaml

static-pod-resources/kube-scheduler-pod-8/kube-scheduler-pod.yaml

1. Restart the kubelet service on all master hosts.

From the recovery host, run the following command:

[core@ip-10-0-143-125 ~]$ sudo systemctl restart kubelet.service

Repeat this step on all other master hosts.

1. Verify that the single member control plane has started successfully.

From the recovery host, verify that the etcd container is running.

[core@ip-10-0-143-125 ~]$ sudo crictl ps | grep etcd

3ad41b7908e32 36f86e2eeaaffe662df0d21041eb22b8198e0e58abeeae8c743c3e6e977e8009 About a minute ago Running etcd 0 7c05f8af362f0

1. From the recovery host, verify that the etcd Pod is running.

[core@ip-10-0-143-125 ~]$ oc get pods -n openshift-etcd | grep etcd

NAME READY STATUS RESTARTS AGE

etcd-ip-10-0-143-125.ec2.internal 1/1 Running 1 2m47s

If the status is Pending, or the output lists more than one running etcd Pod, wait a few minutes and check again.

Force etcd redeployment.

1. In a terminal that has access to the cluster as a cluster-admin user, run the following command:

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

The forceRedeploymentReason value must be unique, which is why a timestamp is appended.

When the etcd cluster Operator performs a redeployment, the existing nodes are started with new Pods similar to the initial bootstrap scale up.

1. Verify all nodes are updated to the latest revision.

In a terminal that has access to the cluster as a cluster-admin user, run the following command:

$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'

Review the NodeInstallerProgressing status condition for etcd to verify that all nodes are at the latest revision. The output shows AllNodesAtLatestRevision upon successful update:

AllNodesAtLatestRevision

3 nodes are at revision 3

If the output shows a message such as 2 nodes are at revision 3; 1 nodes are at revision 4, this means that the update is still in progress. Wait a few minutes and try again.

After etcd is redeployed, force new rollouts for the control plane. The Kubernetes API server will reinstall itself on the other nodes because the kubelet is connected to API servers using an internal load balancer.

In a terminal that has access to the cluster as a cluster-admin user, run the following commands.

Update the kubeapiserver:

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

Verify all nodes are updated to the latest revision.

$ oc get kubeapiserver -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'

Review the NodeInstallerProgressing status condition to verify that all nodes are at the latest revision. The output shows AllNodesAtLatestRevision upon successful update:

AllNodesAtLatestRevision

3 nodes are at revision 3

Update the kubecontrollermanager:

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

Verify all nodes are updated to the latest revision.

$ oc get kubecontrollermanager -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'

Review the NodeInstallerProgressing status condition to verify that all nodes are at the latest revision. The output shows AllNodesAtLatestRevision upon successful update:

AllNodesAtLatestRevision

3 nodes are at revision 3

Update the kubescheduler:

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

Verify all nodes are updated to the latest revision.

$ oc get kubescheduler -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'

Review the NodeInstallerProgressing status condition to verify that all nodes are at the latest revision. The output shows AllNodesAtLatestRevision upon successful update:

AllNodesAtLatestRevision

3 nodes are at revision 3

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

In a terminal that has access to the cluster as a cluster-admin user, run the following command:

$ oc get pods -n openshift-etcd | grep etcd

etcd-ip-10-0-143-125.ec2.internal 2/2 Running 0 9h

etcd-ip-10-0-154-194.ec2.internal 2/2 Running 0 9h

etcd-ip-10-0-173-171.ec2.internal 2/2 Running 0 9h

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.