

**Table of Contents**

* [Kubeadm Cert management](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#kubeadm-cert-management)
  + [Kubeadm command for renew](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#kubeadm-command-for-renew)
* [Config and certs backup](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#config-and-certs-backup)
* [Certificate Expiry Check](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#certificate-expiry-check)
* [Implementation of manual certificate renewal](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#implementation-of-manual-certificate-renewal)
  + [Renew single certificate](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#renew-single-certificate)
  + [Renew all certificates](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#renew-all-certificates)
* [Post implementation steps](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#post-implementation-steps)
* [Managing Kubelet certificate renewal](https://txconsole.com/certificate-renewal-manually-in-kubernetes/#managing-kubelet-certificate-renewal)

Certificate renewal for client certificates becomes usual process for a long running Kubernetes clusters. This blog is an implementation of the process to perform manual cert upgrade. There might be multiple business reasons to perform manual cert rotation due to stringent business requirements.  
For the Kubernetes clusters installed with kubeadm, all the client certificates generated by kubeadm expire after 1 year. As a standard procedure, [manual renewal](https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-certs/#manual-certificate-renewal) is not usually required as cert rotation gets performed automatically when we perform Kubeadm upgrade.

For scenarios to perform manual certificate renewal, during the last month or week of expiry period, we can use Kubeadm commands to verify the certs and renew them for the cluster. The operations team can connect to control plane node and access the kubeadm utility to manage the certificates.

**Kubeadm Cert management**

Kubeadm is equipped with cert management and renewal commands.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [root@kmaster ~]# kubeadm certs  Commands related to handling kubernetes certificates    Usage:    kubeadm certs [command]    Available Commands:    certificate-key  Generate certificate keys    check-expiration Check certificates expiration for a Kubernetes cluster    generate-csr     Generate keys and certificate signing requests    renew            Renew certificates for a Kubernetes cluster |

**Kubeadm command for renew**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | Usage:    kubeadm certs renew [flags]    kubeadm certs renew [command]    Available Commands:    admin.conf               Renew the certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself    all                      Renew all available certificates    apiserver                Renew the certificate for serving the Kubernetes API    apiserver-etcd-client    Renew the certificate the apiserver uses to access etcd    apiserver-kubelet-client Renew the certificate for the API server to connect to kubelet    controller-manager.conf  Renew the certificate embedded in the kubeconfig file for the controller manager to use    etcd-healthcheck-client  Renew the certificate for liveness probes to healthcheck etcd    etcd-peer                Renew the certificate for etcd nodes to communicate with each other    etcd-server              Renew the certificate for serving etcd    front-proxy-client       Renew the certificate for the front proxy client    scheduler.conf           Renew the certificate embedded in the kubeconfig file for the scheduler manager to use |

**Config and certs backup**

For precautionary measure, it’s advised to perform backup for Kubernetes configs and certificates.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | # Backup certs  mkdir -p $HOME/k8scluster-old-certs/pki  /bin/cp -p /etc/kubernetes/pki/\*.\* $HOME/k8scluster-old-certs/pki  ls -l $HOME/k8scluster-old-certs/pki/    #Backup configs  /bin/cp -p /etc/kubernetes/\*.conf $HOME/k8scluster-old-certs  ls -ltr $HOME/k8scluster-old-certs    #Backup local config  mkdir -p $HOME/k8scluster-old-certs/.kube  /bin/cp -p ~/.kube/config $HOME/k8scluster-old-certs/.kube/.  ls -l $HOME/k8scluster-old-certs/.kube/. |

**Certificate Expiry Check**

We can check the cert expiry period with follow command :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [root@kmaster ~]# kubeadm certs check-expiration  [check-expiration] Reading configuration from the cluster...    CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE AUTHORITY   EXTERNALLY MANAGED  admin.conf                 Nov 19, 2022 14:53 UTC   344d                                    no  apiserver                  Nov 19, 2022 14:53 UTC   344d            ca                      no  apiserver-etcd-client      Nov 19, 2022 14:53 UTC   344d            etcd-ca                 no  apiserver-kubelet-client   Nov 19, 2022 14:53 UTC   344d            ca                      no  controller-manager.conf    Nov 19, 2022 14:53 UTC   344d                                    no  etcd-healthcheck-client    Nov 19, 2022 14:53 UTC   344d            etcd-ca                 no  etcd-peer                  Nov 19, 2022 14:53 UTC   344d            etcd-ca                 no  etcd-server                Nov 19, 2022 14:53 UTC   344d            etcd-ca                 no  front-proxy-client         Nov 19, 2022 14:53 UTC   344d            front-proxy-ca          no  scheduler.conf             Nov 19, 2022 14:53 UTC   344d                                    no    CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED  ca                      Nov 17, 2031 14:53 UTC   9y              no  etcd-ca                 Nov 17, 2031 14:53 UTC   9y              no  front-proxy-ca          Nov 17, 2031 14:53 UTC   9y              no |

The above command shows the expiration/residual time for the client certificates. These certificates are located at /etc/kubernetes/pki on the control plane nodes of Kubernetes cluster.

**From command line**

*find /etc/kubernetes/pki/ -type f -name "\*.crt" -print|egrep -v 'ca.crt$'|xargs -L 1 -t -i bash -c 'openssl x509 -noout -text -in {}|grep After'*

**Note:** kubelet.conf is not included in the list above because kubeadm configures kubelet for automatic certificate renewal with rotatable certificates under /var/lib/kubelet/pki.

**Note:**Certs renew uses the existing certificates as the authoritative source for attributes (Common Name, Organization, SAN, etc.) instead of the kubeadm-config ConfigMap. It is strongly recommended to keep them both in sync.

**Implementation of manual certificate renewal**

We can renew the certificates manually at any time with the kubeadm certs renew command. This command performs the renewal using CA certificate and key stored in /etc/kubernetes/pki.

For an HA Kubernetes cluster, kubeadm certs renew command needs to be executed on all the control-plane nodes.

**Renew single certificate**

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@kmaster kubernetes]# kubeadm certs renew admin.conf  [renew] Reading configuration from the cluster...    certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself renewed |

**Result of above step**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@kmaster kubernetes]# kubeadm certs check-expiration  [check-expiration] Reading configuration from the cluster...    CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE AUTHORITY   EXTERNALLY MANAGED  admin.conf                 Dec 10, 2022 05:53 UTC   364d                                    no  ....... |

**Renew all certificates**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@kmaster kubernetes]# kubeadm certs renew all  [renew] Reading configuration from the cluster...    certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself renewed  certificate for serving the Kubernetes API renewed  certificate the apiserver uses to access etcd renewed  certificate for the API server to connect to kubelet renewed  certificate embedded in the kubeconfig file for the controller manager to use renewed  certificate for liveness probes to healthcheck etcd renewed  certificate for etcd nodes to communicate with each other renewed  certificate for serving etcd renewed  certificate for the front proxy client renewed  certificate embedded in the kubeconfig file for the scheduler manager to use renewed    Done renewing certificates. You must restart the kube-apiserver, kube-controller-manager, kube-scheduler and etcd, so that they can use the new certificates. |

**Result of above step**

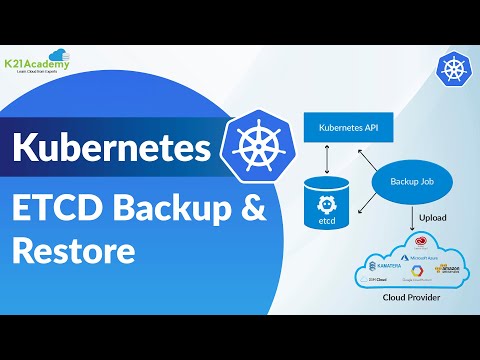
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [root@kmaster kubernetes]# kubeadm certs check-expiration  [check-expiration] Reading configuration from the cluster...    CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE AUTHORITY   EXTERNALLY MANAGED  admin.conf                 Dec 10, 2022 05:56 UTC   364d                                    no  apiserver                  Dec 10, 2022 05:56 UTC   364d            ca                      no  apiserver-etcd-client      Dec 10, 2022 05:56 UTC   364d            etcd-ca                 no  apiserver-kubelet-client   Dec 10, 2022 05:56 UTC   364d            ca                      no  controller-manager.conf    Dec 10, 2022 05:56 UTC   364d                                    no  etcd-healthcheck-client    Dec 10, 2022 05:56 UTC   364d            etcd-ca                 no  etcd-peer                  Dec 10, 2022 05:56 UTC   364d            etcd-ca                 no  etcd-server                Dec 10, 2022 05:56 UTC   364d            etcd-ca                 no  front-proxy-client         Dec 10, 2022 05:56 UTC   364d            front-proxy-ca          no  scheduler.conf             Dec 10, 2022 05:56 UTC   364d                                    no    CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED  ca                      Nov 17, 2031 14:53 UTC   9y              no  etcd-ca                 Nov 17, 2031 14:53 UTC   9y              no  front-proxy-ca          Nov 17, 2031 14:53 UTC   9y              no |

**Post implementation steps**

After running the kubeadm renew command we should restart the control plane Pods. Dynamic certificate reload is currently not supported for all components and certificates. Since,Static Pods are managed by the local kubelet and not by the API Server, thus kubectl cannot be used to delete and restart them.To restart a static Pod we can temporarily remove its manifest file from /etc/kubernetes/manifests/ and wait for 20 seconds.The kubelet will terminate the Pod if it’s no longer in the manifest directory.We can then move the file back and after another fileCheckFrequency period, the kubelet will recreate the Pod and the certificate renewal for the component can complete.

**Managing Kubelet certificate renewal**

Kubeadm configures a kubelet with automatic rotation of client certificates by using the /var/lib/kubelet/pki/kubelet-client-current.pem symlink specified in /etc/kubernetes/kubelet.conf. In case of failure in rotation process, we might see errors such as x509: certificate has expired or is not yet valid in kube-apiserver logs. To manage the manual renewal of kubelet certificates we can follow [Kubernetes doc](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/#kubelet-client-cert).



## Pre Requisite

Make sure you have a K8s cluster deployed already.

**Learn** [How To Setup A Three Node Kubernetes Cluster For CKA](https://k21academy.com/docker-kubernetes/three-node-kubernetes-cluster/)

## Installing and Placing Etcd Binaries

Users mostly interact with etcd by putting or getting the value of a key. We do that by using **etcdctl**, a command line tool for interacting with etcd server. In this section, we are downloading the etcd binaries so that we have the **etcdctl** tool with us to interact.

**1)**  Create a temporary directory for the ETCD binaries.

$ mkdir -p /tmp/etcd && cd /tmp/etcd



$ curl -s https://api.github.com/repos/etcd-io/etcd/releases/latest | grep browser\_download\_url | grep linux-amd64 | cut -d '"' -f 4 | wget -qi -

**3)** Unzip the compressed binaries:

$ tar xvf \*.tar.gz

## Output Screenshot

$ cd etcd-\*/

$ mv etcd\* /usr/local/bin/

$ cd ~

$ rm -rf /tmp/etcd

## Find K8s Manifest Location

In Cluster we can check manifest default location with the help of the kubelet config file.

# cat /var/lib/kubelet/config.yaml

With this Manifest location, you can check the Kubernetes static pods location and find Api-server and ETCD pod location then under these pods you can check certificate file and data-dir location.

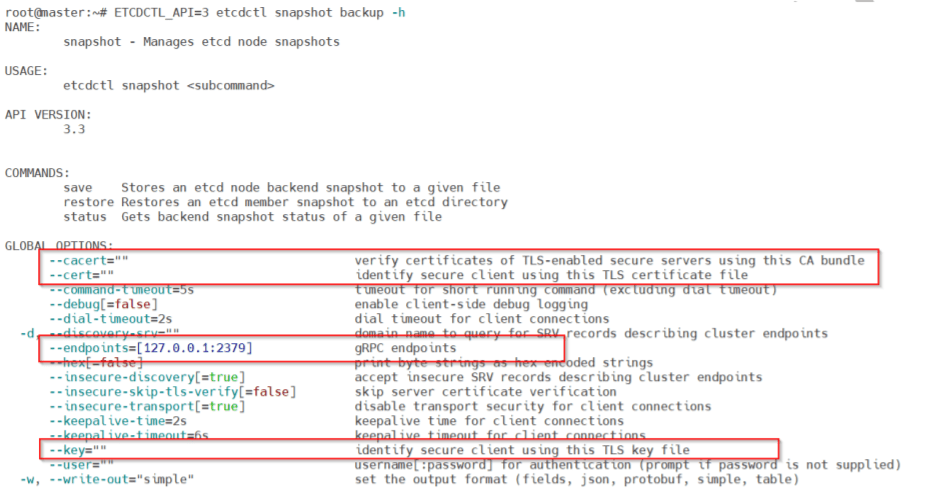
## How to backup the Etcd & Restore it

The etcd server is the only stateful component of the Kubernetes cluster. Kuberenetes stores all API objects and settings on the etcd server.  
Backing up this storage is enough to restore the Kubernetes cluster’s state completely.

**Taking Snapshot and Verifying it:**

**1)** Check backup Command flag which you need to include in the command

$ ETCDCTL\_API=3 etcdctl snapshot backup -h



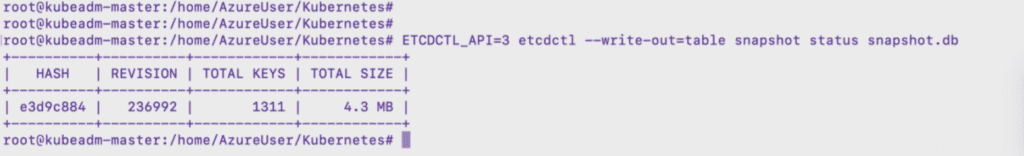
**2)**Take a snapshot of the etcd datastore using etcdctl:

$ sudo ETCDCTL\_API=3 etcdctl snapshot save snapshot.db --cacert /etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/server.crt --key /etc/kubernetes/pki/etcd/server.key



**3)** View that the snapshot was successful:

$ sudo ETCDCTL\_API=3 etcdctl snapshot status snapshot.db



**Note:** Important Note: If you are backing up and restoring the cluster do not run the status command after the backup this might temper the backup due to this restore process might fail.

**Backing-up The Certificates And Key Files:**

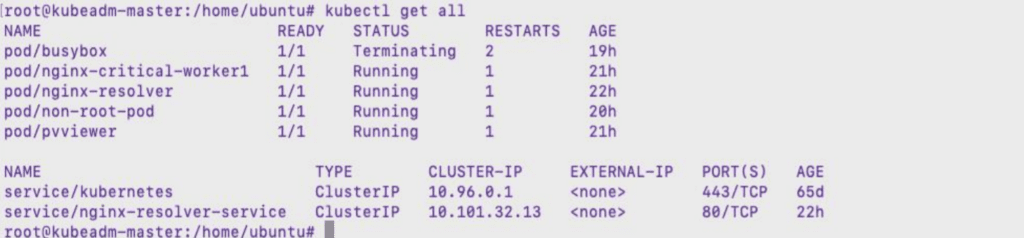
Zip up the contents of the etcd directory to save the certificates too

$ sudo tar -zcvf etcd.tar.gz /etc/kubernetes/pki/etcd

**Restoring Etcd From Snapshot & Verify:**

**1)** Check the present state of the cluster which is stored in present snapshot taken in above task:

$ kubectl get all

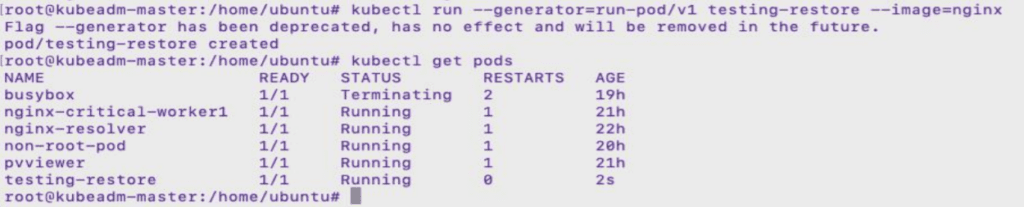


**2)** To verify now we will create a pod and as the new pod is not present in the snapshot will not be present when we restore the content back using restore command:

$ kubectl run testing-restore --image=nginx

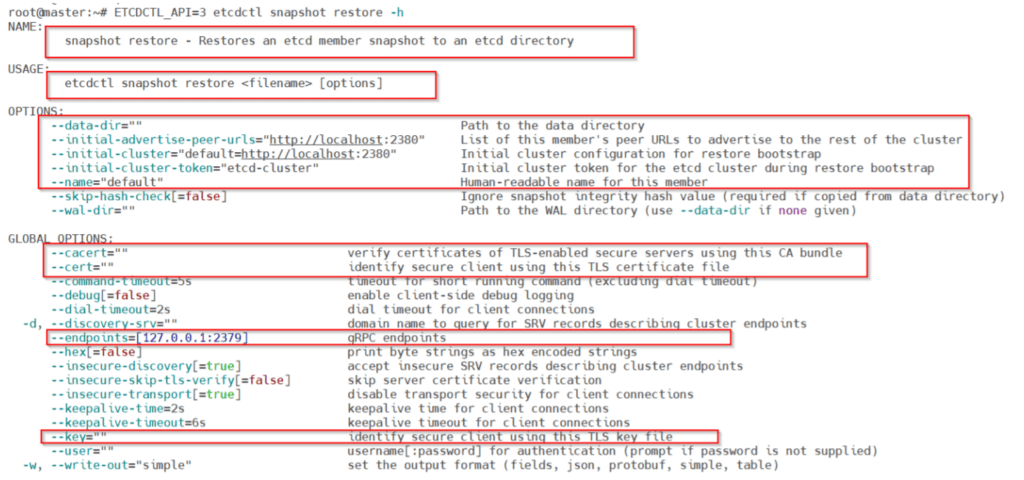
$ kubectl get pods

**Note:** In the new Kubernetes version the generator flag is deprecated so use the above updated command to create a pod.



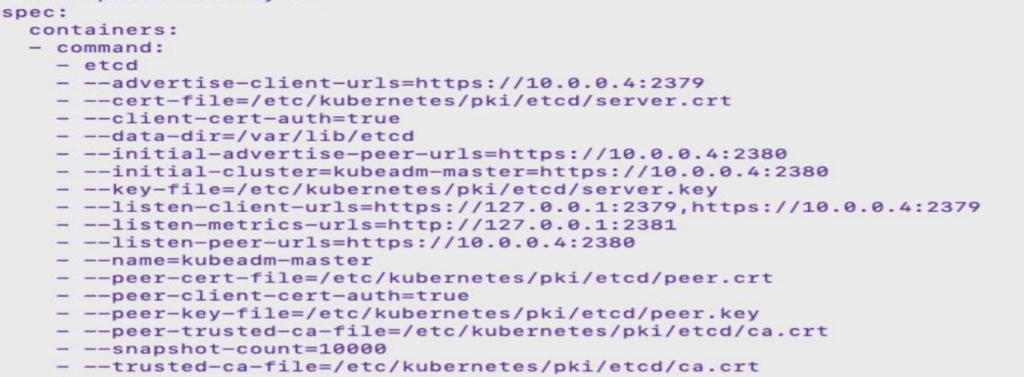
**3)** Check restore Command flag which you need to include in command

$ ETCDCTL\_API=3 etcdctl snapshot restore -h



**4)**To restore we will have to first delete the present ETCD content. So lets look into and grab all the details we need for the restore command to execute

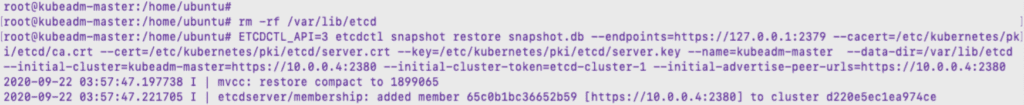
$ cat /etc/kubernetes/manifests/etcd.yaml



**5)** Will delete the present content of ETCD and execute the restore command

$ rm -rf /var/lib/etcd

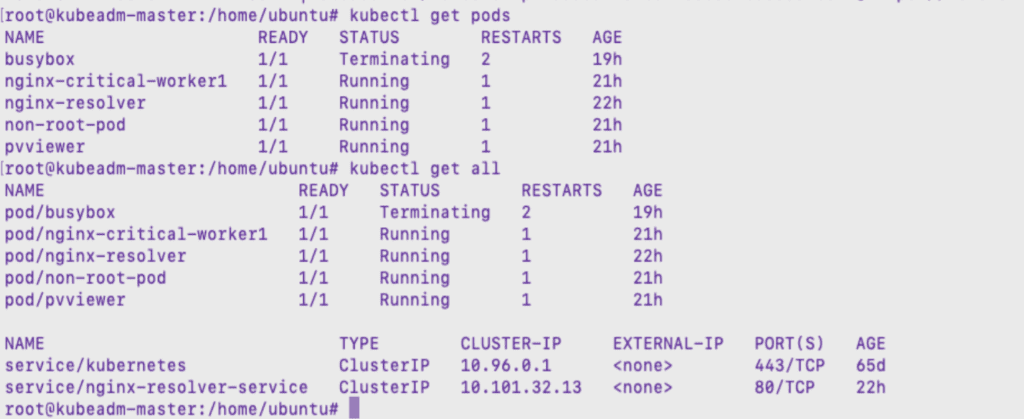
$ ETCDCTL\_API=3 etcdctl snapshot restore snapshot.db --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key --name=kubeadm-master --data-dir=/var/lib/etcd --initial-cluster=kubeadmmaster=https://10.0.0.4:2380 --initial-cluster-token=etcd-cluster-1 --initial-advertise-peerurls=https://10.0.0.4:2380



**6)** Verify that the cluster is back to status of which we had taken the snapshot

$ kubectl get pods

$ kubectl get all



Congratulations! We are now successfully done with the backup & restoration process of our ETCD cluster in Kubernetes.

## Conclusion

For a single control plane arrangement, a Kubernetes cluster with infrequent API server changes is a fantastic alternative. Backups of the etcd cluster on a regular basis will reduce the time frame for potential data loss