**Kubernetes配置高可用etcd服务(V1.28.2)**

**1.Configure the kubelet to be a service manager for etcd.**

|  |
| --- |
| Bash cat << EOF > /etc/systemd/system/kubelet.service.d/kubelet.conf # Replace "systemd" with the cgroup driver of your container runtime. The default value in the kubelet is "cgroupfs". # Replace the value of "containerRuntimeEndpoint" for a different container runtime if needed. # apiVersion: kubelet.config.k8s.io/v1beta1 kind: KubeletConfiguration authentication:  anonymous:  enabled: false  webhook:  enabled: false authorization:  mode: AlwaysAllow cgroupDriver: systemd address: ${HOST} containerRuntimeEndpoint: unix:///var/run/containerd/containerd.sock staticPodPath: /etc/kubernetes/manifests EOF cat << EOF > /etc/systemd/system/kubelet.service.d/20-etcd-service-manager.conf [Service] ExecStart=/usr/bin/kubelet --config=/etc/systemd/system/kubelet.service.d/kubelet.conf Restart=always EOF |

|  |
| --- |
| Bash systemctl daemon-reload systemctl restart kubelet |

**2.Create configuration files for kubeadm.**

Generate one kubeadm configuration file for each host that will have an etcd member running on it using the following script.

|  |
| --- |
| Bash *# Update HOST0, HOST1 and HOST2 with the IPs of your hosts* *export HOST0=192.168.135.149 export HOST1=192.168.135.150 export HOST2=192.168.135.147  # Update NAME0, NAME1 and NAME2 with the hostnames of your hosts* *export NAME0="k8-etcd-01"* *export NAME1="k8-etcd-02"* *export NAME2="k8-etcd-03" # Create temp directories to store files that will end up on other hosts* *mkdir -p /tmp/****${****HOST0****}****/ /tmp/****${****HOST1****}****/ /tmp/****${****HOST2****}****/  HOSTS=(****${****HOST0****}******${****HOST1****}******${****HOST2****}****)* *NAMES=(****${****NAME0****}******${****NAME1****}******${****NAME2****}****)* ***for*** *i in "****${****!HOSTS[@]****}****";*  ***do***  *HOST=****${****HOSTS[$i]****}***  *NAME=****${****NAMES[$i]****}***  *cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml*  *---*  *apiVersion: "kubeadm.k8s.io/v1beta3"*  *kind: InitConfiguration*  *nodeRegistration:  name: ${NAME}*  *localAPIEndpoint:  advertiseAddress: ${HOST}*  *---*  *apiVersion: "kubeadm.k8s.io/v1beta3"*  *kind: ClusterConfiguration*  *etcd:  local:  serverCertSANs:  - "${HOST}"  peerCertSANs:  - "${HOST}"  extraArgs:  initial-cluster: ${NAMES[0]}=https://${HOSTS[0]}:2380,${NAMES[1]}=https://${HOSTS[1]}:2380,${NAMES[2]}=https://${HOSTS[2]}:2380  initial-cluster-state: new  name: ${NAME}  listen-peer-urls: https://${HOST}:2380  listen-client-urls: https://${HOST}:2379  advertise-client-urls: https://${HOST}:2379  initial-advertise-peer-urls: https://${HOST}:2380*  *imageRepository: registry.aliyuncs.com/google\_containers*  *kubernetesVersion: v1.28.2*  *EOF* ***done*** |

**3.Delete all certs**

|  |
| --- |
| Bash #delete these files and director  *export HOST0=192.168.135.149 export HOST1=192.168.135.150 export HOST2=192.168.135.147  HOSTS=(****${****HOST0****}******${****HOST1****}******${****HOST2****}****)* ***for*** *i in "****${****!HOSTS[@]****}****";*  ***do***  *sudo rm -rf* /etc/kubernetes/pki  *sudo rm -rf /tmp/****${****HOSTS[$i]****}/***   done |

**4.Generate the certificate authority.**

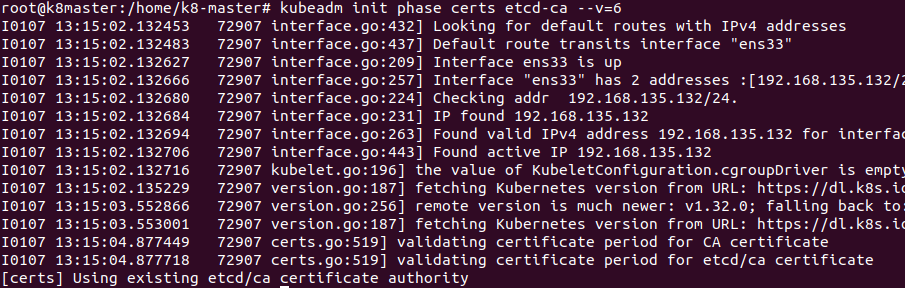
If you already have a CA then the only action that is copying the CA's crt and key file to /etc/kubernetes/pki/etcd/ca.crt and /etc/kubernetes/pki/etcd/ca.key. After those files have been copied, proceed to the next step, "Create certificates for each member".

If you do not already have a CA then run this command on $HOST0 (where you generated the configuration files for kubeadm).

|  |
| --- |
| Plain Text kubeadm init phase certs etcd-ca --kubernetes-version v1.28.2 |

This creates two files:

* /etc/kubernetes/pki/etcd/ca.crt
* /etc/kubernetes/pki/etcd/ca.key



**5.Create certificates for each member.(On HOST0)**

|  |
| --- |
| Bash #HOST2 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST2**}**/kubeadmcfg.yaml cp -R /etc/kubernetes/pki /tmp/**${**HOST2**}**/ *# cleanup non-reusable certificates* find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST1 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST1**}**/kubeadmcfg.yaml cp -R /etc/kubernetes/pki /tmp/**${**HOST1**}**/ find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete  #HOST0 kubeadm init phase certs etcd-server --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-peer --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml kubeadm init phase certs etcd-healthcheck-client --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml kubeadm init phase certs apiserver-etcd-client --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml *# No need to move the certs because they are for HOST0* *# clean up certs that should not be copied off this host* find /tmp/**${**HOST2**}** -name ca.key -type f -delete find /tmp/**${**HOST1**}** -name ca.key -type f -delete |

**6.Copy certificates and kubeadm configs.(ON HOST0)**

The certificates have been generated and now they must be moved to their respective hosts.

|  |
| --- |
| Bash #k8-etcd-02 USER=${NAME2} HOST=${HOST2} scp -r /tmp/${HOST}/\* ${USER}@${HOST}: ssh ${USER}@${HOST} |

|  |
| --- |
| Bash #k8-etcd-03 USER=${NAME3} HOST=${HOST3} scp -r /tmp/${HOST}/\* ${USER}@${HOST}: ssh ${USER}@${HOST} |

|  |
| --- |
| Bash #on k8-etcd-02 and k8-etcd-03 sudo -Es chown -R root:root pki mv pki /etc/kubernetes/ |

**7.Ensure all expected files exist.**

The complete list of required files on $HOST0 is:

|  |
| --- |
| Bash tree /tmp/${HOST0} /etc/kubernetes/pki  /tmp/${HOST0} └── kubeadmcfg.yaml --- /etc/kubernetes/pki ├── apiserver-etcd-client.crt ├── apiserver-etcd-client.key └── etcd  ├── ca.crt  ├── ca.key  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key |

On $HOST1:

|  |
| --- |
| Bash tree $HOME  $HOME └── kubeadmcfg.yaml --- /etc/kubernetes/pki ├── apiserver-etcd-client.crt ├── apiserver-etcd-client.key └── etcd  ├── ca.crt  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key |

On $HOST2:

|  |
| --- |
| Bash tree $HOME $HOME └── kubeadmcfg.yaml --- /etc/kubernetes/pki ├── apiserver-etcd-client.crt ├── apiserver-etcd-client.key └── etcd  ├── ca.crt  ├── healthcheck-client.crt  ├── healthcheck-client.key  ├── peer.crt  ├── peer.key  ├── server.crt  └── server.key |

**8.Create the static pod manifests.**

Now that the certificates and configs are in place it's time to create the manifests. On each host run the kubeadm command to generate a static manifest for etcd.

|  |
| --- |
| Bash kubeadm init phase etcd local --config=/tmp/**${**HOST0**}**/kubeadmcfg.yaml kubeadm init phase etcd local --config=$HOME/kubeadmcfg.yaml kubeadm init phase etcd local --config=$HOME/kubeadmcfg.yaml |

**9.Check if they are healthy.**

|  |
| --- |
| Bash #k8-etcd-01 docker run --rm -it \ --net host \ -v /etc/kubernetes:/etc/kubernetes registry.aliyuncs.com/google\_containers/etcd:3.5.9-0 etcdctl \ --cert /etc/kubernetes/pki/etcd/peer.crt **\** --key /etc/kubernetes/pki/etcd/peer.key **\** --cacert /etc/kubernetes/pki/etcd/ca.crt **\** --endpoints https://**192.168.135.151**:2379 endpoint health  #k8-etcd-02 docker run --rm -it \ --net host \ -v /etc/kubernetes:/etc/kubernetes registry.aliyuncs.com/google\_containers/etcd:3.5.9-0 etcdctl \ --cert /etc/kubernetes/pki/etcd/peer.crt **\** --key /etc/kubernetes/pki/etcd/peer.key **\** --cacert /etc/kubernetes/pki/etcd/ca.crt **\** --endpoints https://**${HOST1}**:2379 endpoint health \   #k8-etcd-03 docker run --rm -it \ --net host \ -v /etc/kubernetes:/etc/kubernetes registry.aliyuncs.com/google\_containers/etcd:3.5.9-0 etcdctl \ --cert /etc/kubernetes/pki/etcd/peer.crt **\** --key /etc/kubernetes/pki/etcd/peer.key **\** --cacert /etc/kubernetes/pki/etcd/ca.crt **\** --endpoints https://**${HOST2}**:2379 endpoint health |

