Kubernetes installation with Meshnet-CNI and k8s-topo

\*Important Note\*

Ensure you have swapoff disabled

$ sudo swapoff -a

To permanently disable swapoff upon system start, run:

$ sudo sed -i '/ swap / s/^\(.\*\)$/#\1/g' /etc/fstab

Why is disabling swap off important?

Support for swap is non-trivial. Guaranteed pods should never require swap. Burstable pods should have their requests met without requiring swap. BestEffort pods have no guarantee. The kubelet right now lacks the smarts to provide the right amount of predictable behavior here across pods.As of today, the scheduler does not understand swap configuration and usage to provide smarts for swap support.[8]

Installation of kubernetes(local host)

First lets install all the necessary packages needed for kubernetes

$ sudo apt-get update

$ sudo apt-get install -y apt-transport-https ca-certificates curl

Download the Google Cloud public signing key:

$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg

Add the Kubernetes apt repository:

$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

Install kubectl, kubelet, and kubeadm:

$ sudo apt-get install kubeadm kubelet kubectl

Once all of our necessary packages are installed we can begin to develop our cluster.

Initialize the cluster and its control plane by running the following:

$ kubeadm init

To make kubectl work for your non-root user, run these commands.:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Note:

To create a proxy to your cluster run:

$ kubectl proxy --port=8080 &

Kubernetes Cheat Sheet

https://github.com/dennyzhang/cheatsheet-kubernetes-A4

Once you have initialized your cluster and set up the configuration files we can run the following to see the pods we have at the moment.

$ kubectl get pods --all-namespaces

You should notice the namespace kube-system which is essentially the namespace for the essential control plane components. You might also notice that the coredns are not yet running. This is because we currently do not have a Container Network Interface(CNI) deployed inside our cluster. As described before we will be working with Meshnet-CNI. This works well with our topology builder k8s-topo.

#### Installing Meshnet-CNI

Retrieve the git repository from github by running the following:

$ cd $HOME

$ git clone https://github.com/networkop/meshnet-cni.git && cd meshnet-cni

Now we need to navigate to our manifests .yml file. This will give our cluster the object information to deploy the plugin. It will create a namespace for the meshnet interface, a resource definition for the network topology, privileges to meshnet to work with other resources and a daemonset with the plugin and configuration files.

$ kubectl apply -f /manifests/base/meshnet.yml

Installing k8s-topo and testing

$ cd $HOME

git clone https://github.com/networkop/k8s-topo.git && cd k8s-topo

Deploy k8s-topo pod

$kubectl create -f manifest.yml

Log into the k8s-topo pod.

$ kubectl exec --stdin --tty <pod name> -- /bin/sh

Create a random 20-node network topology( at this point you should be inside the k8s-topo pod.

/k8s-topo # ./examples/builder/builder 20 0

Total number of links generated: 19

Create the topology inside K8s

/k8s-topo # k8s-topo --create examples/builder/random.yml

Optionally, you can generate a D3.js network topology graph

/k8s-topo # k8s-topo --graph examples/builder/random.yml

View the generated topology graph at *http://<any\_k8s\_cluster\_node\_ip>:32000*

Verify that the topology has been deployed (from the master node)

/k8s-topo # kubectl get pods -o wide | grep qrtr

qrtr-1 1/1 Running 0 11s 10.233.65.231 node3 <none>

qrtr-10 1/1 Running 0 11s 10.233.65.234 node3 <none>

qrtr-11 1/1 Running 0 10s 10.233.66.246 node4 <none>

Destroy the topology and exit the pod by running:

/k8s-topo # k8s-topo --destroy examples/builder/random.yml  
/k8s-topo # exit