Many guides about installing Docker, Kubernetes and deploy Kubernetes Dashboard and access it remotely, yet each guide I found required serious extra work and configuration. Should you be someone like me, unhappy with other guides, try this one and let me know how I did.

**Prerequisites:**

This guide assumes that you have a minimal configuration like this:



We will use only one master and one worker. In a virtual environment, each server must have at least 2 virtual CPUs.

In this guide, master and worker(s) are Ubuntu server(s). IPs are fixed, either at server level or reserved at the DHCP level.

The terminal has the *ssh* feature installed/available.

**Installation and configuration**

* On the **master** server
1. Install Docker distribution version, and check version number

sudo apt-get update

sudo apt-get install docker.io

docker -v

1. Enable Docker, start Docker and check status

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl status docker

Docker status should display the service is active. Quit with **q** .

 

1. Add repository for Kubernetes

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt-get update

1. Install Kubernetes and verify version

sudo apt-get install kubeadm kubelet kubectl

kubeadm version

1. Disable swap on the Ubuntu server on every node.

Why? Kubernetes is a distributed system that is designed for scalability in a deterministic way. Running large number of containers on several machines require predictability and consistency. Disabling swap is the right approach. Therefore, the kubelet is not designed to handle swap feature. Usually the swap file is /swap.img

sudo swapoff -v /swap.img

sudo rm /swap.img

sudo vi /etc/fstab

Comment out in /etc/fstab the swap related line.

1. Initialize Kubernetes on the master node

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

1. Check status with

sudo systemctl status kubelet



**If kubeadm fails, check the drivers,**

sudo docker info | grep -i cgroup

and set the same in /etc/systemd/system/kubelet.service.d/10-kubeadm.conf : *Environment="KUBELET\_KUBECONFIG\_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf* ***--cgroup-driver=cgroupfs****"*

If the service is active, save the join string displayed at the execution end because it is needed to join the workers. The join string will look like this :

kubeadm join 192.168.0.14:6443 --token j5c1cn.b6ub4u5r8umwb1bp --discovery-token-ca-cert-hash sha256:a44849c4d31cf517d7f58ea8d5051bdc24561aae642f3bc76a61a9d077b4728e

1. Create configuration for the cluster

mkdir -p $HOME/.kube

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

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

1. Deploy Pod Network to Cluster allowing communication between different nodes in the cluster. We use the flannel virtual network.

sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

1. Allow the process to complete and verify everything is running and communicating. All must be running.

kubectl get pods --all-namespaces



Go to each worker now and follow these steps.

* On each worker
1. Install Docker distribution version, and check version number

sudo apt-get update

sudo apt-get install docker.io

docker -v

1. Enable Docker, start Docker and check status

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl status docker

Docker status should display the service is active. Quit with **q** .

 

1. Install curl

sudo apt install apt-transport-https curl

1. Add repository for Kubernetes

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add

sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt-get update

1. Install Kubernetes and verify version

sudo apt-get install kubeadm kubelet kubectl

kubeadm version

1. Disable swap on the Ubuntu server on every node.

Why? Kubernetes is a distributed system that is designed for scalability in a deterministic way. Running large number of containers on several machines require predictability and consistency. Disabling swap is the right approach. Therefore, the kubelet is not designed to handle swap feature. Usually the swap file is /swap.img

sudo swapoff -v /swap.img

sudo rm /swap.img

sudo vi /etc/fstab

Comment out in /etc/fstab the swap related line.

1. Join worker node to cluster with the string backed up before

sudo kubeadm join 192.168.0.14:6443 *--token j5c1cn.b6ub4u5r8umwb1bp --discovery-token-ca-cert-hash sha256:a44849c4d31cf517d7f58ea8d5051bdc24561aae642f3bc76a61a9d077b4728e*

Return to the master server

* On the **master** server

We will deploy Kubernetes Dashboard, a web-based Kubernetes user interface allowing deploying containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources. Dashboard provides also an overview of applications running on your cluster, as well as for creating or modifying individual Kubernetes resources (such as Deployments, Jobs, DaemonSets, etc). For example, you can scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard. Dashboard also provides information on the state of Kubernetes resources in your cluster and on any errors that may have occurred.

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml

Allow a few moment and check the deployment

kubectl get pods --all-namespaces

All must be running

To access Dashboard using the kubectl command-line tool start the proxy in background

kubectl proxy

^Z

bg

Kubectl will make Dashboard available at http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

By design, the UI can only be accessed from the machine where the command is executed. In order to access it from a terminal in a web browser, few additional steps:

For the Dashboard authentication, create a Dashboard service account in the default namespace

kubectl create serviceaccount dashboard-admin-sa

Bind the service account to the cluster admin role

kubectl create clusterrolebinding dashboard-admin-sa --clusterrole=cluster-admin --serviceaccount=default:dashboard-admin-sa

At the service account creation Kubernetes created also the secrets

kubectl get secrets

In the returned list look for a secret like *dashboard-admin-sa-token-?????* And get the associated token

kubectl describe secret dashboard-admin-sa-token-?????

Save the token for authentication. Now go to the terminal and create a tunnel to the Dashboard deployment.

* On the terminal

On the terminal computer we ssh must be available

Open a command line terminal (cmd on Windows or term on Linux) and create a tunnel with the following command:

ssh -L 8001:127.0.0.1:8001 -N -f -l username ip\_address\_of\_the\_master

After that, in a browser, with the url [http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/#/overview?namespace=default](http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https%3Akubernetes-dashboard%3A/proxy/#/overview?namespace=default) you should be able to display the dashboard. Authenticate with the toke saved before, and you are done.

