Kubernetes is an open source system for automating deployment, scaling and management of containerized applications.

To install Kubernetes on a CentOS 7 server in our Cloud Playground. Below, you will find a list of the commands used in this lesson.

\*Note in this lesson we are using 3 unit servers as this meets the minimum requirements for the Kubernetes installation. Use of a smaller size server (less than 2 cpus) will result in errors during installation.

1. The first thing that we are going to do is use SSH to log in to all machines. Once we have logged in, we need to elevate privileges using sudo.
2. sudo su
3. Disable SELinux.
4. setenforce 0
5. sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
6. Enable the br\_netfilter module for cluster communication.
7. modprobe br\_netfilter
8. echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
9. Disable swap to prevent memory allocation issues.
10. swapoff -a
11. vim /etc/fstab.orig -> Comment out the swap line
12. Install the Docker prerequisites.
13. yum install -y yum-utils device-mapper-persistent-data lvm2
14. Add the Docker repo and install Docker.
15. yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
16. yum install -y docker-ce
17. Conigure the Docker Cgroup Driver to systemd, enable and start Docker
18. sed -i '/^ExecStart/ s/$/ --exec-opt native.cgroupdriver=systemd/' /usr/lib/systemd/system/docker.service
19. systemctl daemon-reload
20. systemctl enable docker --now
21. Add the Kubernetes repo.
22. cat <<EOF > /etc/yum.repos.d/kubernetes.repo
23. [kubernetes]
24. name=Kubernetes
25. baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64
26. enabled=1
27. gpgcheck=0
28. repo\_gpgcheck=0
29. gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
30. https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
31. EOF
32. Install Kubernetes.
33. yum install -y kubelet kubeadm kubectl
34. Enable Kubernetes. The kubelet service will not start until you run kubeadm init.
35. systemctl enable kubelet

\**Note: Complete the following section on the MASTER ONLY!*

1. Initialize the cluster using the IP range for Flannel.
2. kubeadm init --pod-network-cidr=10.244.0.0/16
3. Copy the kubeadmin join command.
4. Exit sudo and run the following:
5. mkdir -p $HOME/.kube
6. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
7. sudo chown $(id -u):$(id -g) $HOME/.kube/config
8. Deploy Flannel.
9. kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
10. Check the cluster state.
11. kubectl get pods --all-namespaces

\**Note: Complete the following steps on the NODES ONLY!*

1. Run the join command that you copied earlier (this command needs to be run as sudo), then check your nodes from the master.

kubectl get nodes

Master & Nodes

Kubernates Master has

ETCD – Key value store of the cluster

API Server – This is where all of the instructions are send

Scheduler – This check the nodes schedule

Controller manager

On Nodes

Proxy

Kubelet

Container runtime

Kubectl get nodes

Kubectl get pods –all-namespaces

\*\*Kubectl create namespace podexample

\*\*kubectl create –f ./pod-example.yaml

\*\*kubectl --namespace=podexample get pods

\*\*kubectl --namespace=podexample delete pod examplepod

Network in Kubernetes

Create the file – replicas-example.yaml (replica set)

Kubectl create –f ./replicas-example.yaml

Kubectl get pods

Kubectl describe rs/frontend

Kubectl describe pod <pod name>

Kubectl rs/frontend --replicas=4

Kubectl rs/frontend –replicas=1

Kubectl delete rs/frontend

======

Cat ./replicas-example.yaml

apiVerison: apps/v1

kind: ReplicaSet

metadata:

name: frontend

labels:

app: nginx

tier: frontend

spec:

replicas: 2

selector:

matchLabels:

tier: frontend

matchExpressions:

- {key: tier, operator: In, values: [frontend]}

template:

metadata:

labels:

app: nginx

tier: frontend

spec:

cotnainers:

- name: nginx

image: darealmc/nginx-k8s:v1

ports:

- containerPort: 80:

Services:

Kubectl create –f ./replicas-example.yaml ( using the replica set create Pod)

Kubectl get pods

Kubectl describe pod < pod name>

Kubectl create –f ./service-example.yaml (to create service)

Kubectl describe service <service name>

Kubectl scale rs/frontend --replicas=4

Kubectl describe service <service name>

Deployment

Kubectl get deployment

**deployexample.yaml:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: example-deployment

labels:

app: nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: darealmc/nginx-k8s:v1

ports:

- containerPort: 80

Check out the official Kubernetes site for documentation and more info!

[http://kubernetes.io](http://kubernetes.io/)

Containers wrap SW in independent portable packages making it easy to quickly run software in a variety of environments

**Kubernetes Master**

Docker (it will be kubernetes runtime)

kubeadm(it will allow us to build kubernetes env quickly)

kubelet(an agent to run container on each node)

kubectl(command line tool)

Control plane (series of services which form kubernetes cluster

**Kube Node 1**

Docker , kubeadm, kubelet & kubectl

Docker – Container runtime

***Docker install***

**First, just update the apt package index:**

sudo apt-get update

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

Output

**The following NEW packages will be installed:**

**apt-transport-https**

**Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 apt-transport-https all 1.6.12 [1692 B]**

**Fetched 1692 B in 0s (0 B/s)**

**Selecting previously unselected package apt-transport-https.**

**(Reading database ... 195885 files and directories currently installed.)**

**Preparing to unpack .../apt-transport-https\_1.6.12\_all.deb ...**

**Unpacking apt-transport-https (1.6.12) ...**

**Setting up apt-transport-https (1.6.12) ...**

**Then Install packages to allow apt to use a repository over HTTPS:**

**Add the Docker repository GPG Key**

apt-key is a program that is used to manage a keyring of gpg keys for secure apt.  
gpg is the tool used in secure apt to sign files and check their signatures

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

Manage apt's list of trusted keys

apt-key add <file> - add the key contained in <file> ('-' for stdin)

apt-key del <keyid> - remove the key <keyid>

apt-key export <keyid> - output the key <keyid>

apt-key exportall - output all trusted keys

apt-key update - update keys using the keyring package

apt-key net-update - update keys using the network

apt-key list - list keys

apt-key finger - list fingerprints

apt-key adv - pass advanced options to gpg (download key)

**Add the Docker repository**

sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

Reload the apt sources list

sudo apt-get update

Install Docker

sudo apt-get install -y docker-ce=18.06.1~ce~3-0~ubuntu

If not specific version then we can use - **sudo apt-get install –y docker-ce**

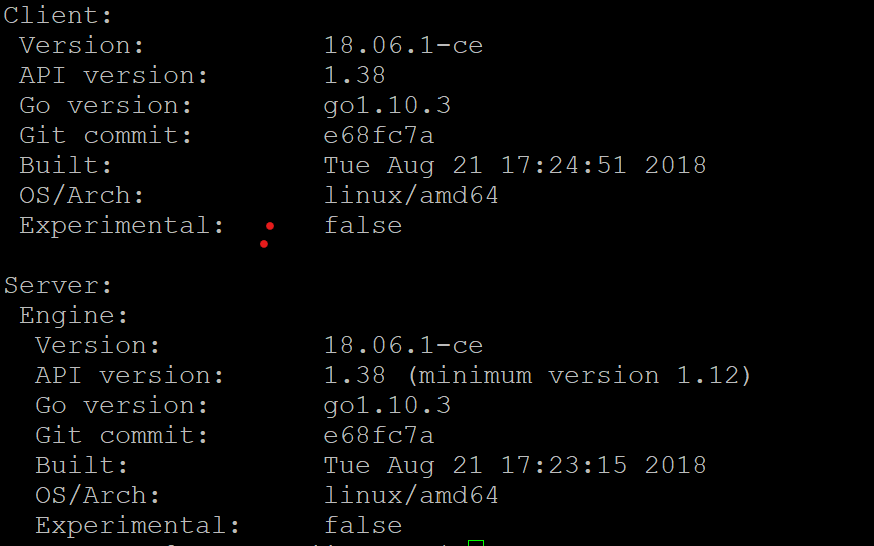
Prevent auto update for Docker package

sudo apt-mark hold docker-ce

sudo apt-mark unhold docker-ce

You can verify that docker is working by running this command:

sudo docker version



We can also run the hello world to see whether it is installed successfully

sudo docker run hello-world

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

**Install the Kubernetes components**

kubeadm – this is a tool which automates a large portion of the process of setting up a cluster

kubelet – essential component of Kubernetes that handle running container on a node.

Kubectl - command line tool to interact with cluster once it’s up. It’s used to manage the cluster

Add kubernetes repository GPG key

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

add kubernetes Repository

cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update

Install packages

sudo apt-get install -y kubelet=1.12.7-00 kubeadm=1.12.7-00 kubectl=1.12.7-00

Prevent auto update

sudo apt-mark hold kubelet kubeadm kubectl

For version

kubeadm version

**Bootstrapping the Cluster**

Use kubeadm to build the cluster. Initialize the cluster on kube master server :

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

setup kubeconfig for the local user on kube master server. This will allow you to use kubectl when logged into the master

mkdir -p $HOME/.kube

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

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

Verify that the cluster is responsive and that Kubectl is working:

kubectl version

The kubeadm init command should output a kubeadm join command containing a token and hash. Copy that command and run it with sudo on both worker nodes. It should look something like this:

sudo kubeadm join $some\_ip:6443 --token $some\_token --discovery-token-ca-cert-hash $some\_hash

Verify that all nodes have successfully joined the cluster

kubectl get nodes

You should see all three of your nodes listed. It should look something like this:

NAME STATUS ROLES AGE VERSION

wboyd1c.mylabserver.com NotReady master 5m17s v1.12.2

wboyd2c.mylabserver.com NotReady <none> 53s v1.12.2

wboyd3c.mylabserver.com NotReady <none> 31s v1.12.2

\*\*\*\* <https://www.edureka.co/blog/kubernetes-networking/> \*\*

## Configuring Networking with Flannel

flannel is a virtual network that gives a subnet to each host for use with container runtimes.

* On all three nodes, run the following:

echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf

sudo sysctl -p

* Install Flannel in the cluster by running this only on the Master node:

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml

* Verify that all the nodes now have a STATUS of Ready:

kubectl get nodes

* It is also a good idea to verify that the Flannel pods are up and running. Run this command to get a list of system pods:

kubectl get pods -n kube-system

You should have three pods with flannel in the name, and all three should have a status of Running.

## Containers and Pods