**Install on master and worker nodes**

Docker – is a software responsible for running the containers.

kubeadm – a CLI tool that will install and configure the various components of a cluster in a standard way.

kubelet – a system service/program that runs on all nodes and handles node-level operations.

kubectl – a CLI tool used for issuing commands to the cluster through its API Server.

**Steps to install on ‘AWS manage cluster-EKS’…**

1. Install awscli

#apt install python-pip (ubuntu18)

#apt install python3-pip (ubuntu20.4)

#pip install awscli

#pip3 install awscli (ubuntu20.4)

1. Add IAM user credentials in credentials

#cd ~/.aws

#vi credentials

[default]

aws\_access\_key\_id=\*\*\*\*\*\*\*\*\*\*\*\*

aws\_secret\_access\_key=\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#aws iam list-users

1. Create IAM role with permissions ‘AmazonEKSClusterpolicy’
2. Create EKS cluster under EKS by attaching above role.
3. Install ‘AWS IAM authenticator’ (<https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html>)
4. Install ‘kubectl’ (<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>)

#kubectl get service

1. Update kubernetes cluster

#aws eks --region ap-south-1 update-kubeconfig --name EKS-CLUSTER-NAME

#more /home/ubuntu/.kube/config

#export KUBECONFIG=~/.kube/config

#kubectl get service

1. Create another IAM role for node group on ‘EC2’ service with following permissions
   1. AmazonEKS\_CNI\_policy
   2. AmazonEKSworkerNodepolicy
   3. AmazonEC2ContainerRegistryReadOnly
2. Create worker node group under EKS cluster by attaching above role
3. Get the nodes list and verify nodes

#kubectl get nodes

**2. On virtual instances(‘kubeadm’)…**

**Run on master and nodes**

1. $ apt-get update && apt-get install -y docker.io
2. $ vi /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

1. systemctl enable docker

systemctl daemon-reload

systemctl restart docker

systemctl status docker

1. $ apt-get update && apt-get install -y apt-transport-https
2. $ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

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

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

EOF

1. $ apt-get update
2. $ apt-get install -y kubelet kubeadm kubectl
3. $ apt-mark hold kubelet kubeadm kubectl

**Create kubernetes cluster (Run on Master)**

1. $ kubeadm init
2. Save ‘kubadm join’

(use the command “$ kubeadm token create --print-join-command” to regenerate kubeadm token)

1. $ cp /etc/kubernetes/admin.conf $HOME/
2. $ chown $(id -u):$(id -g) $HOME/admin.conf
3. $ export KUBECONFIG=$HOME/admin.conf

**Join Worker Nodes to the Kubernetes Cluster (Run on worker node)**

1. Run ‘kubadm join’ command.

**Testing the Kubernetes Cluster(Run on Master)**

1. $ kubectl get nodes
2. $ kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

$ kubectl apply -f [https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')](https://cloud.weave.works/k8s/net?k8s-version=$(kubectl%20version%20|%20base64%20|%20tr%20-d%20'\n')) …. Deprecated

1. $ kubectl get nodes
2. kubectl get pods -n kube-system

**Run following commands if ‘kubectl apply’ fails..(master)**

mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config