# Kubeadm cluster set up on AWS with AWS cloud-provider and AWS LoadBalancer

# Prequisites:

# VPC

# Subnet

# Internet Gateway

# Route Table

# IAM Master Role

# IAM Worker Role

# EC2 Instances for Master and Worker

# For creation for VPC, Subnet, Internet Gateway, Route Table, IAM Master Role, IAM Worker and EC2 Instances for Master and Worker Role please go through the below link

# <https://itnext.io/kubernetes-part-2-a-cluster-set-up-on-aws-with-aws-cloud-provider-and-aws-loadbalancer-f02c3509f2c2#969d>

# Kubernetes installation:

# Step-1: Update packages list and installed packages

# apt update && apt -y upgrade

# Step-2: Add Docker and Kubernetes repositories:

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

# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

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

# echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" > /etc/apt/sources.list.d/kubernetes.list

# apt update

# apt install -y docker-ce kubelet kubeadm kubectl

# Hostname:

# Check a hostname now:

# hostname

# Get it as a fully qualified domain name (FQDN):

# curl http://169.254.169.254/latest/meta-data/local-hostname

# Set the hostname as FQDN:

# hostnamectl set-hostname ip-10-0-0-9.eu-west-3.compute.internal

# Check now:

# Hostname

# D:\Naresh\snipping\Capture.JPG

# Repeat Kubernetes installation and Hostname on Worker Machine also.

# Cluster Set up:

# Master Machine:

# Create a “/etc/kubernetes/aws.yml” file

# vi /etc/kubernetes/aws.yml

# ---

# apiVersion: kubeadm.k8s.io/v1beta2

# kind: ClusterConfiguration

# networking:

# serviceSubnet: "10.100.0.0/16"

# podSubnet: "10.244.0.0/16"

# apiServer:

# extraArgs:

# cloud-provider: "aws"

# controllerManager:

# extraArgs:

# cloud-provider: "aws"

# D:\Naresh\snipping\Capture.JPG

# Initialize cluster using this config:

# kubeadm init --config /etc/kubernetes/aws.yml

# Note:

# Here we get kubeadm join command to connect nodes to master

# D:\Naresh\snipping\Capture.JPG

# To start using your cluster, you need to run the following as a regular user:

# mkdir -p $HOME/.kube

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

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

# Check Whether Master node created or not:

# kubectl get nodes -o wide

# D:\Naresh\snipping\Capture.JPG

# Flannel CNI installation:

# From the Master node execute:

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

# D:\Naresh\snipping\Capture.JPG

# Again check Nodes:

# kubectl get nodes -o wide

# D:\Naresh\snipping\Capture.JPG

# You can get your cluster-info using the config view:

# kubeadm config view

# D:\Naresh\snipping\Capture.JPG

# Attaching the Worker Node:

# On the Worker node create a "/etc/kubernetes/node.yml" file with the JoinConfiguration:

# vi /etc/kubernetes/node.yml

# ---

# apiVersion: kubeadm.k8s.io/v1beta1

# kind: JoinConfiguration

# discovery:

# bootstrapToken:

# token: "whx7dn.yevc787z5zkj9tzk"

# apiServerEndpoint: "10.0.0.157:6443"

# caCertHashes:

# - "sha256:5d2a885e85f7be51566b46e969835c9b358212aca89c5cfd6bee3454ae0ddeb2"

# nodeRegistration:

# name: ip-10-0-0-41.ec2.internal

# kubeletExtraArgs:

# cloud-provider: aws

# With in this file we need to modify:

# - token, apiServerEndpoint and caCertHashes (These details we get from kubeadm join command)

# - nodeRegistration name (hostname of Worker machine)

# D:\Naresh\snipping\Capture.JPG

# Join this node to the cluster:

# kubeadm join --config /etc/kubernetes/node.yml

# D:\Naresh\snipping\Capture.JPG

# Go back to the Master, check nodes one more time:

# kubectl get nodes

# D:\Naresh\snipping\Capture.JPG

# Deploy NGINX with kubeadm:

# Create "nginx.yml" file:

# vi nginx.yml

# kind: Service

# apiVersion: v1

# metadata:

# name: hello

# spec:

# type: LoadBalancer

# selector:

# app: hello

# ports:

# - name: http

# protocol: TCP

# # ELB's port

# port: 80

# ---

# apiVersion: apps/v1

# kind: Deployment

# metadata:

# name: hello

# spec:

# replicas: 1

# selector:

# matchLabels:

# app: hello

# template:

# metadata:

# labels:

# app: hello

# spec:

# containers:

# - name: hello

# image: nginx

# Apply it:

# kubectl apply -f nginx.yml

# D:\Naresh\snipping\Capture.JPG

# Check Deployments and Services:

# kubectl get deploy -o wide

# kubectl get svc

# D:\Naresh\snipping\Capture.JPG

# Goto LoadBalancer in AWS and check whether nodes comes into Inservice or not:

# D:\Naresh\snipping\Capture.JPG

# Check out with LoadBalancer DNS name:

# a2731548efc66491fb3ad0735e34ed10-1086833560.us-east-1.elb.amazonaws.com

# 

# Get svc’s, pods and deployments:

# kubectl get svc

# kubectl get pods

# kubectl get deployments

# D:\Naresh\snipping\Capture.JPG

# Delete svc’s, pods and deployments:

# kubectl delete svc hello

# kubectl delete pods hello-57495d49b5-blcd7

# kubectl delete deployments hello