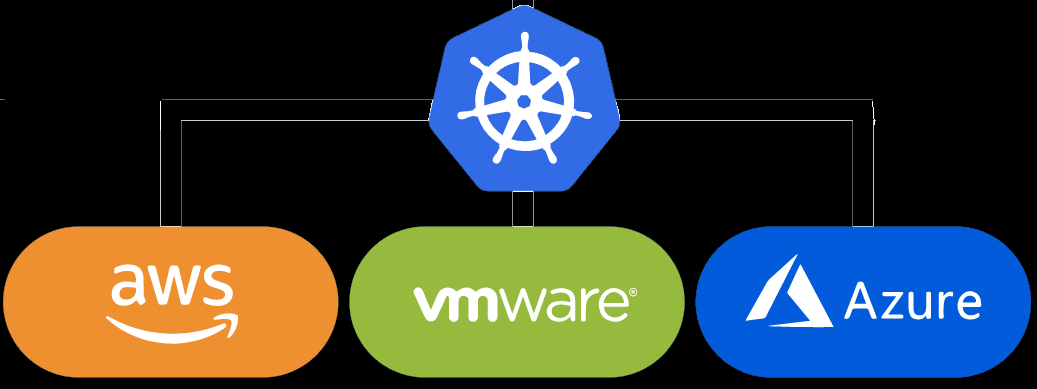
**Creating multi-node and multi-cloud Kubernetes cluster**



**Kubernetes:**

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.



**Amazon Web Service :**

We can define AWS (Amazon Web Services) as a secured cloud services platform that offers compute power, database storage, content delivery, and various other functionalities. To be more specific, it is a large bundle of cloud-based services.



**Microsoft Azure**

Azure is a public cloud computing platform — with solutions including **Infrastructure as a Service** (IaaS), **Platform as a Service** (PaaS), and **Software as a Service** (SaaS) that can be used for services such as analytics, virtual computing, storage, networking, and much more. It can be used to replace or supplement your on-premise servers.



— — — — — — — — — — — — — — — — — — — — — — — — — — — — —

In this article, we are going to configure the Kubernetes multi-node cluster. Where we have configured the master in AWS, One slave in AWS, One slave in Microsoft azure, Two slaves in Local virtual instance.

First, launch two basic EC-2 instances in AWS one for master and one for slave.

Open the Linux terminal, Now we are going the configure the Kubernetes master.

Kubernetes uses container technology, Therefore we are installing docker.

yum install docker -y

Now, start and enable the docker service.

systemctl start docker  
systemctl enable docker

Changing the docker driver from cgroupfs to systems:

vim /etc/docker/daemon.json{  
 **"exec-opts": ["native.cgroupdriver=systemd"**]  
}

To apply this driver we need to restart the docker service

systmctl docker restart

To install Kubernetes into the system first we need to configure the k8s repository.

vim /etc/yum.repos.d/k8s.repo[kubernetes]  
name=Kubernetes  
baseurl=[https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch](https://packages.cloud.google.com/yum/repos/kubernetes-el7-/$basearch)  
enabled=1  
gpgcheck=1  
repo\_gpgcheck=1  
gpgkey=<https://packages.cloud.google.com/yum/doc/yum-key.gpg> <https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg>

Installing the required packages to run the master node

kubeadm - Kubeadm is a tool built to provide kubeadm init and kubeadm join as best-practice “fast paths” for creating Kubernetes clusters.

kubectl — You can use **kubectl** to deploy applications, inspect and manage cluster resources, and view logs

kubelet — The **kubelet** is the primary “node agent” that runs on each node. It can register the node with the apiserver using one of the hostnames.

yum install kubeadm kubectl kubelet -y

Starting and enabling the kubelet service.

systemctl enable kubelet --now

To set up the Kubernetes cluster. We need to pull docker images using kubeadm. It pulls images of the config files.

kubeadm config images pull

Installing “iproute-tc” which will help us to manage the traffic on the cluster

yum install iproute-tc

Now, lets change IP tables /etc/sysctl.d/k8s.conf

vim /etc/sysctl.f/k8s.confnet.bridge.bridge-nf-call-ip6tables = 1  
net.bridge.bridge-nf-call-iptables = 1

The important step: Initializing Master

kubeadm init --control-plane-endpoint "PUBLICIP:PORT" --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem

**pod-network-cidr= IP range for pods inside the slave nodes**

**Control plane endpoint = assign the cluster with a public IP with port**

**ignore-preflight-errors= Ignoring the unwanted CPU errors and memory errors**

Now the master is initialized.

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

Run the above commands.

To connect the nodes of the master and slave we use a flannel. Flannels act as a DHCP server as well as a router in the cluster. It will create a nating between the pods running in the cluster. The flannel works on the underlying network.

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

The above command will download and apply the flannel.

**Generating Token:**

The token is created at the time of initializing the master

kubeadm token create --print-join-command

This will print the token which we have to run on the slaves.

Now, The master is ready.

**Slaves Configuration:**

We are going the launch one slave in AWS, Second in Azure, third and fourth in local systems

yum install iproute-tc #Installing iproute-tcyum install docker -y #Install Dockervim /etc/docker/daemon.json #Changing the driver{  
 **"exec-opts": ["native.cgroupdriver=systemd"**]  
}systemctl restart docker #Restart docker  
systemctl enable docker --now #enable Docker#Kubernetes Repository  
vim /etc/yum.repos.d/k8s.repo [kubernetes]  
name=Kubernetes  
baseurl=[https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch](https://packages.cloud.google.com/yum/repos/kubernetes-el7-/$basearch)  
enabled=1  
gpgcheck=1  
repo\_gpgcheck=1  
gpgkey=<https://packages.cloud.google.com/yum/doc/yum-key.gpg> <https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg>#Installing the required packages  
yum install kubeadm kubectl kubelet -y#Enabling kubelet service  
systemctl enable kubelet --now#Configure the iptables /etc/sysctl.d/k8s.confvim /etc/sysctl.d/k8s.conf  
net.bridge.bridge-nf-call-ip6tables = 1  
net.bridge.bridge-nf-call-iptables = 1sysctl --system

The above command I explained in detail while creating the master, they have the same meaning as in the master.

First, let's connect the local systems.

Disable swap in the local systems

swapoff -a

Now, go to the master and print the token using the join command.

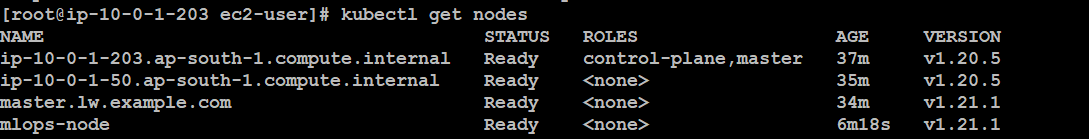
kubeadm token create --print-join-command

https://miro.medium.com/max/1400/1*evE5K5javlPsJJZS0kfjDw.png

Copy the above-printed token and run it on all the slaves

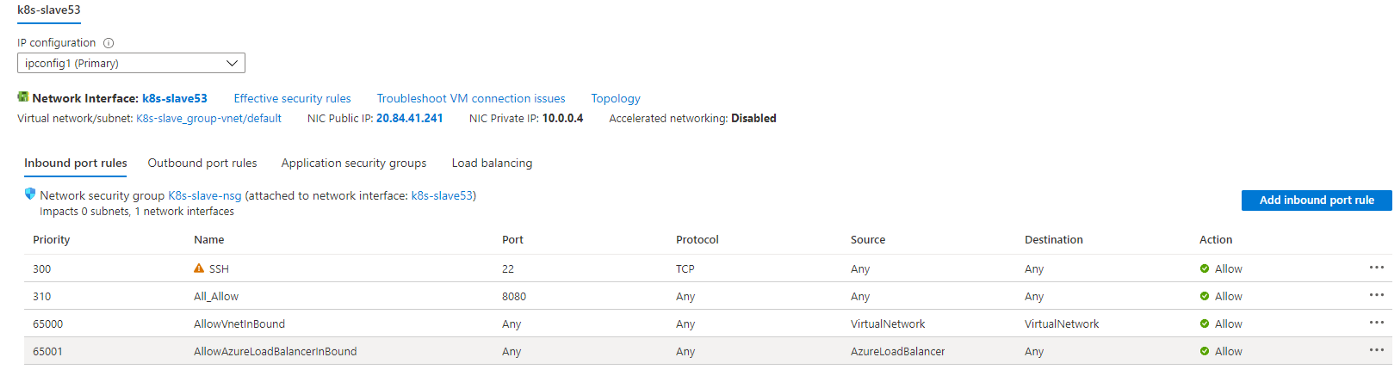
Check the nodes using

kubectl get nodes



One slave is connected through AWS, master.lw.example.com and mlops-node are from the local systems.

In the azure instance the inbound and outbound rule are as follow:





Successfully created multi-node and multi-cloud Kubernetes cluster.

**⭐Keep Learning !! Keep Sharing !!⭐**

Thank you for visiting my article.