Create AWS Account & Kubernetes Cluster Setup

This guide walks you through creating an AWS account and setting up a Kubernetes cluster with a Master and Worker node connection.

- Create AWS Account
  Note: Practicing Kubernetes will cost you some amount
- EC2 > Ubuntu OS > t2.medium instance type or higher
- Create new security group
  Name: Kubernetes.SG
  Description: A brief description for the security group (mandatory)
  - Ensure that all instances are in the same Security Group
  - Expose port 22 in the Security Group to allow SSH access to manage the instance.
  - Expose port 6443 in the Security Group to allow worker nodes to join the cluster.
- Select this created security group while creating instances.
- Create Instances.
  - Number of Instance: 2 (Master Node , Worker Node)

Execute Below Commands on Both Nodes

Disable Swap: Required for Kubernetes to function correctly.

```
sudo swapoff -a
```

Load Necessary Kernel Modules: Required for Kubernetes networking.

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
```

Set Sysctl Parameters: Helps with networking.

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
```

```
net.ipv4.ip_forward                 = 1
EOF

sudo sysctl --system
lsmod | grep br_netfilter
lsmod | grep overlay
```

Install Containerd:

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
sudo apt-get install -y containerd.io

containerd config default | sed -e 's/SystemdCgroup = false/SystemdCgroup = true/' -e 's/sandbox_image = "registry.k8s.io\/pause:3.6"/sandbox_image = "registry.k8s.io\/pause:3.9"/' | sudo tee /etc/containerd/config.toml

sudo systemctl restart containerd
sudo systemctl status containerd
```

Install Kubernetes Components:

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

Execute ONLY on the "Master" Node

Initialize the Cluster:

sudo kubeadm init

Set Up Local kubeconfig:

mkdir -p "$HOME"/.kube

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

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

Install a Network Plugin (Calico):

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

Generate Join Command:

kubeadm token create --print-join-command

Note : Copy this generated token for next command.

Execute on ALL of your Worker Nodes

Perform pre-flight checks:

sudo kubeadm reset pre-flight checks

Paste the join command you got from the master node and append --v=5 at the end:

sudo <paste-join-command-here> --v=5

Verify Cluster Connection

On Master Node:

kubectl get nodes

On Worker Node:

sudo crictl pods

Note : You will get output similar in snip attached in post 3.

Congratulations! You have created your first Kubernetes cluster

Important : Terminate Instances and associated resource once you are done with practice to avoid getting over bill.