**How to Install Kubernetes Cluster on AWS : A Step-by-Step Guide**

**Prerequisites**

* Ubuntu OS (Xenial or later)
* Internet access
* t2.medium instance type or higher (Make sure it should have atleast 2 vCPUs)

**First Let’s set up the security groups for the cluster**

* Go to the **EC2 Dashboard**
* In the left menu under **Network & Security**, click on **Security Groups**.
* Click on **Create Security Group**.
* Provide the following details:
  + **Name**: (e.g., Kubernetes-Cluster-SG)
  + **Description**: A brief description for the security group (mandatory)
  + **VPC**: Select the appropriate VPC for your instances (default is acceptable)
* **Allow SSH Traffic (Port 22)**:
  + **Type**: SSH
  + **Port Range**: 22
  + **Source**: 0.0.0.0/0 (Anywhere) or your specific IP
* **Allow Kubernetes API Traffic (Port 6443)**:
  + **Type**: Custom TCP
  + **Port Range**: 6443
  + **Source**: 0.0.0.0/0 (Anywhere) or specific IP ranges
* Click on **Create Security Group** to save the settings.

**AWS Setup**

* Ensure that all instances are in the same **Security Group**
* Expose port **6443** in the **Security Group** to allow worker nodes to join the cluster
* Expose port **22** in the **Security Group** to allows SSH access to manage the instance
* When launching EC2 instances:
  + Under **Configure Security Group**, select the existing security group (Kubernetes-Cluster-SG)

*You can also add the instance in the security groups after creating the instance*

*Select instance → Networking → Select in Network Interfaces → Actions(Top Right)→ Change security groups → Select and add security groups*

**Execute on both master & worker nodes**

**Disable Swap**: Required for Kubernetes to function correctly.

sudo swapoff -a *(disables all swap spaces on the system to force the use of RAM.)*

**Load Necessary Kernel Modules**: Required for Kubernetes networking.

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

overlay

br\_netfilter

EOF

*(The command creates (or overwrites) the /etc/modules-load.d/k8s.conf file with the kernel modules overlay and br\_netfilter, ensuring they load at system startup for Kubernetes networking.)*

sudo modprobe overlay *(Loads the overlay kernel module, required for container storage.)*

sudo modprobe br\_netfilter  *(Loads the br\_netfilter kernel module, needed for Kubernetes networking.)*

**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

*(Creates a configuration file to enable packet forwarding and bridge networking for Kubernetes.)*

sudo sysctl –system *(Applies all system parameter changes from configuration files.)*

lsmod | grep br\_netfilter *(Checks if the br\_netfilter module is loaded.)*

lsmod | grep overlay *(Checks if the overlay module is loaded.)*

**Install Containerd**:

sudo apt-get update *(Updates the package lists to fetch the latest versions.)*

sudo apt-get install -y ca-certificates curl *(Installs ca-certificates and curl, which are required for secure downloads.)*

sudo install -m 0755 -d /etc/apt/keyrings *(Creates a secure directory for storing APT GPG keyrings.)*

sudo curl -fsSL [https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc](https://download.docker.com/linux/ubuntu/gpg%20-o%20/etc/apt/keyrings/docker.asc)

*(Downloads Docker’s GPG key for package verification.)*

sudo chmod a+r /etc/apt/keyrings/docker.asc *(Sets read permissions on Docker’s GPG key for all users.)*

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

*(Adds the official Docker repository to the system’s package sources.)*

sudo apt-get update *(Refreshes package lists to include Docker repository packages.)*

sudo apt-get install -y containerd.io *(Installs containerd, a container runtime needed for Kubernetes.)*

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

*(Configures containerd to use systemd as the cgroup driver and updates the sandbox image.)*

sudo systemctl restart containerd *(Restarts the containerd service to apply the new configuration.)*

sudo systemctl status containerd *(Checks the status of the containerd service.)*

**Install Kubernetes Components**:

sudo apt-get update *(Updates package lists before installing Kubernetes components.)*

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

*(Installs dependencies needed to fetch and verify Kubernetes packages.)*

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

*(Downloads and saves the GPG key for Kubernetes package verification.)*

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

*(Adds the Kubernetes package repository to the system.)*

sudo apt-get update *(Refreshes package lists to include Kubernetes repository packages.)*

sudo apt-get install -y kubelet kubeadm kubectl *(Installs Kubernetes components: kubelet, kubeadm, and kubectl.)*

sudo apt-mark hold kubelet kubeadm kubectl *(Prevents automatic updates of Kubernetes components to avoid compatibility issues.)*

**Execute only on the master node**

**Initialize the Cluster**:

sudo kubeadm init (*Initializes the Kubernetes control-plane (master node) and outputs the join command for worker nodes.)*

**Set Up Local kubeconfig**:

mkdir -p "$HOME"/.kube *(Creates the .kube directory in the user's home folder if it doesn't exist (for storing the kubeconfig).)*

sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config *(Copies the admin configuration file to the .kube directory for kubectl access.)*

sudo chown "$(id -u)":"$(id -g)" "$HOME"/.kube/config *(Changes ownership of the kubeconfig file to the current user for permission access.)*

**Install a Network Plugin (Calico)**:

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

*(Deploys the Calico network plugin to enable pod-to-pod communication within the cluster.)*

**Generate Join Command**:

kubeadm token create --print-join-command

*(Generates the command (with token and hash) for worker nodes to join the Kubernetes cluster.)*

💡

*Copy this generated token for next command.*

**Execute on all of your worker nodes**

Perform pre-flight checks:

Copy

sudo kubeadm reset pre-flight checks *(Resets the worker node, removing old cluster configurations to ensure a clean join.)*

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

💡

*Make sure to replace the IP address, token and hash key*

Copy

sudo kubeadm join <private-ip-of-control-plane>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash> --cri-socket

"unix:///run/containerd/containerd.sock" --v=5

*(Connects the worker node to the master node using the provided IP, token, hash, and container runtime interface (CRI) socket with verbose output.)*

**Verify cluster connection**

**On Master Node:**

Copy

kubectl get nodes

*((on Master Node): Lists all nodes in the cluster, showing their statuses to confirm if the worker nodes successfully joined.)*

You should see master and all worker nodes here :

A computer screen with white text

Description automatically generated

You have successfully created a Kubernetes cluster :)