

### **Advanced DevOps Experiment:3**

**Aim:** To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

#### **Theory:**

To understand Kubernetes Cluster Architecture and how to install and spin up a Kubernetes cluster on Linux machines or cloud platforms, it's essential to grasp the fundamental components and design principles of Kubernetes.

##### Overview of Kubernetes

Kubernetes is an open-source container orchestration platform developed by Google, designed to automate the deployment, scaling, and management of containerized applications. It provides a robust infrastructure that supports microservices architecture, offering features such as self-healing, scaling, and zero-downtime deployments. Kubernetes can run on various environments, including public clouds (like AWS and Azure), private clouds, and bare metal servers.

##### Kubernetes Architecture

Kubernetes architecture is primarily composed of two main components: the Control Plane and the Data Plane.

##### Control Plane

The Control Plane manages the overall state of the Kubernetes cluster and includes several key components:

- kube-apiserver: The API server acts as the gateway for all interactions with the cluster, processing REST requests and managing the state of the cluster.
- etcd: A distributed key-value store that holds the configuration data and state of the cluster, ensuring consistency and availability.
- kube-scheduler: Responsible for assigning Pods to worker nodes based on resource availability and other constraints.
- kube-controller-manager: Manages controllers that regulate the state of the cluster, ensuring that the desired state matches the actual state.
- cloud-controller-manager (optional): Integrates with cloud provider APIs to manage resources specific to the cloud environment.

##### Data Plane

The Data Plane consists of the worker nodes that run the containerized applications. Each worker node includes:

- kubelet: An agent that ensures containers are running in Pods. It communicates with the Control Plane to receive instructions.
- kube-proxy: Maintains network rules and facilitates communication between Pods and services.
- Container Runtime: Software responsible for running containers, such as Docker or containerd.

## Core Concepts

Key concepts in Kubernetes include:

- Pods: The smallest deployable units in Kubernetes, which can contain one or more containers.
- Services: Abstracts a set of Pods, providing a stable network endpoint for accessing them.
- Deployments: Define the desired state for Pods and manage their lifecycle, including scaling and updates.

## Installing and Spinning Up a Kubernetes Cluster

To install and set up a Kubernetes cluster, follow these general steps:

1. Choose an Environment: Decide whether to deploy on local machines or a cloud platform. For cloud platforms, services like Google Kubernetes Engine (GKE), Amazon EKS, or Azure AKS can simplify the process.
2. Install Prerequisites: Ensure that you have the necessary tools installed, such as kubectl (the command-line tool for interacting with the cluster) and a container runtime.
3. Set Up the Control Plane: This can be done using tools like kubeadm, which helps bootstrap the cluster by initializing the Control Plane components.
4. Join Worker Nodes: Once the Control Plane is set up, you can join worker nodes to the cluster using the token generated during the initialization.
5. Deploy Applications: After the cluster is up and running, you can deploy your applications using YAML configuration files that define the desired state of your Pods and Services.

## Best Practices

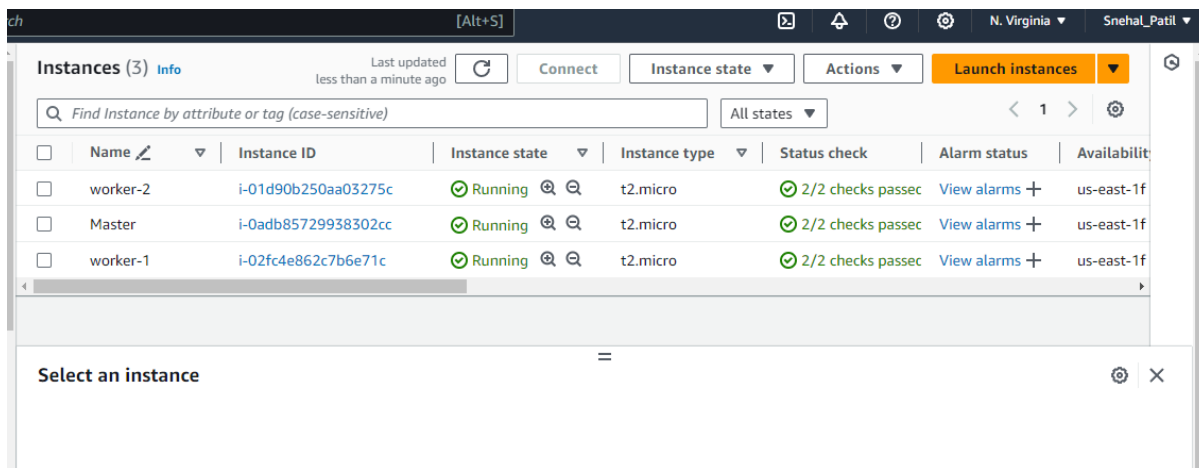
When setting up a Kubernetes cluster, consider the following best practices:

- Resource Management: Define resource requests and limits for Pods to ensure efficient utilization of cluster resources.
- High Availability: Use multiple Control Plane nodes to avoid single points of failure.
- Networking: Implement network policies to secure communication between Pods and manage external access.
- Monitoring and Logging: Integrate monitoring tools and logging solutions to keep track of cluster performance and troubleshoot issues.

By understanding the architecture and following the installation steps and best practices, you can effectively manage a Kubernetes cluster, enabling efficient deployment and scaling of containerized applications.

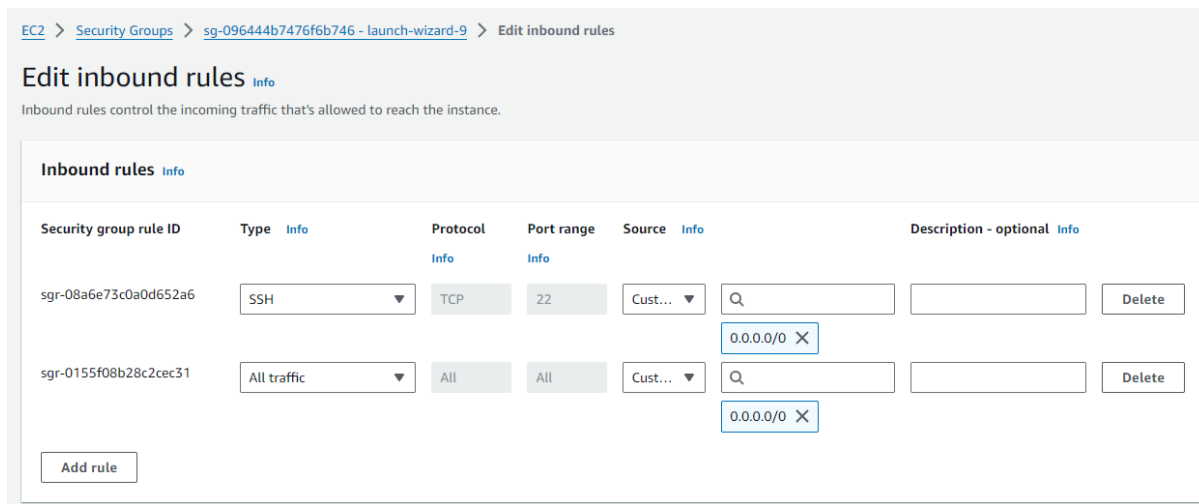
## Steps:

1. Create 3 EC2 Ubuntu Instances on AWS.



2. Now click on connect to instance, then click on SSH client.

Now copy the ssh from the example and paste it on command prompt.






## Connect to instance [Info](#)

Connect to your instance i-043b8a223eab93c13 (master) using any of these options


[EC2 Instance Connect](#)[Session Manager](#)[SSH client](#)[EC2 serial console](#)

Instance ID

 [i-043b8a223eab93c13](#) (master)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Snehalkey.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
 `chmod 400 "Snehalkey.pem"`
4. Connect to your instance using its Public DNS:  
 `ec2-34-229-230-179.compute-1.amazonaws.com`

Example:

 `ssh -i "Snehalkey.pem" ubuntu@ec2-34-229-230-179.compute-1.amazonaws.com`

### Prerequisites:

Ensure the following requirements are met before starting the Kubernetes cluster setup:

- **Ubuntu OS:** Ubuntu Xenial or later is recommended.
- **sudo Privileges:** Administrative access to execute commands.
- **Instance Type:** Use t2.medium or higher for adequate resources.

### Commands to Execute on Both Master and Worker Nodes:

1. After the instances have been created, copy the text given in the example part of each of the three instances into git bash.

```
Windows@DESKTOP-I925HTO MINGW64 ~/Downloads
$ ssh -i "Snehalkey.pem" ubuntu@ec2-44-204-227-4.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-227-4.compute-1.amazonaws.com (44.204.227.4)' can't be established.
ED25519 key fingerprint is SHA256:IGnMIHNbyAzruaz0ospRsy1CcVXZPSJ1iGisJHkwTqE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-227-4.compute-1.amazonaws.com' (ED25519)
to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Wed Sep 18 09:48:04 UTC 2024

System load:  0.0                Processes:            114
Usage of /:   22.8% of 6.71GB    Users logged in:     0
Memory usage: 5%                IPv4 address for enX0: 172.31.87.198
Swap usage:   0%
```

```
ubuntu@ip-172-31-87-198:~$ sudo su
root@ip-172-31-87-198:/home/ubuntu# Yum install docker -y
Command 'Yum' not found, did you mean:
  command 'gum' from snap gum (0.13.0)
  command 'uum' from deb freewnn-jserver (1.1.1~a021+cvs20130302-7build1)
  command 'num' from deb quickcal (2.4-1)
  command 'zum' from deb perforate (1.2-5.3)
  command 'sum' from deb coreutils (9.4-2ubuntu2)
See 'snap info <snapname>' for additional versions.
```

---

Run the following commands on both the master and worker nodes to prepare them for kubeadm.

### 1. Disable Swap:

- Command: `sudo swapoff -a`
- Explanation: Disables swap memory, which is required by Kubernetes for proper functioning.

### 2. Create Configuration File for Kernel Modules:

- Commands:

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

```
overlay
```

```
br_netfilter
```

```
EOF
```

- Explanation: Loads necessary kernel modules for Kubernetes networking.

```
root@ip-172-31-87-198:/home/ubuntu# sudo swapoff -a
root@ip-172-31-87-198:/home/ubuntu# cat <<EOF | sudo tee /etc/modules-load.d/k8s
.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
overlay
br_netfilter
```

---

### Configure Sysctl Parameters:

- Command:

```
bash
```

Copy code

```
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
```

- Explanation: Sets necessary network parameters for Kubernetes. These settings enable IP forwarding and ensure iptables rules are applied to bridge traffic.

```

root@ip-172-31-87-198:/home/ubuntu# 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
>
> ^C
root@ip-172-31-87-198:/home/ubuntu# 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
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1

```

---

### Apply Sysctl Parameters:

- Command: `sudo sysctl --system`
- Explanation: Applies the sysctl parameters without rebooting the system.

```

root@ip-172-31-87-198:/home/ubuntu# sudo sysctl --system
* Applying /usr/lib/sysctl.d/10-apparmor.conf ...
* Applying /etc/sysctl.d/10-console-messages.conf ...
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
* Applying /etc/sysctl.d/10-map-count.conf ...
* Applying /etc/sysctl.d/10-network-security.conf ...
* Applying /etc/sysctl.d/10-ptrace.conf ...

```

---

### Install CRI-O Runtime:

- **Commands:**

```

bash
Copy code
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --
dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-
o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service

```

- **Explanation:** Installs and configures CRI-O, a container runtime compatible with Kubernetes.

```

root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https ca-c
ertificates gpg
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [
126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packag
es [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-
en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Compon
ents [3871 kB]

```

---

## Add Kubernetes APT Repository and Install Packages:

- Commands:**

```

bash
Copy code
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 -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"
sudo apt-get update -y
sudo apt-get install -y jq

sudo systemctl enable --now kubelet
sudo systemctl start kubelet

```

- Explanation:** Adds the Kubernetes APT repository, installs Kubernetes tools (kubelet, kubectl, kubeadm), and ensures the kubelet service is running.

```

root@ip-172-31-87-198:/home/ubuntu# sudo curl -fsSL https://pkgs.k8s.io/addons:/
cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyring
s/cri-o-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.i
o/addons:/cri-o:/prerelease:/main/deb/ " | sudo tee /etc/apt/sources.list.d/cri
-o.list
deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addo
ns:/cri-o:/prerelease:/main/deb/ /

```

---

```
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y cri-o
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [12
6 kB]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
-o:/prerelease:/main/deb InRelease [1206 B]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
-o:/prerelease:/main/deb Packages [2454 B]
Fetched 130 kB in 0s (282 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Recommended packages:
  kubernetes-cni
The following NEW packages will be installed:
  cri-o
0 upgraded, 1 newly installed, 0 to remove and 130 not upgraded.
Need to get 19.9 MB of archives.
After this operation, 76.5 MB of additional disk space will be used.
Get:1 https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb cri-o 1.32.0~dev-
2.1 [19.9 MB]
```

---

```
root@ip-172-31-87-198:/home/ubuntu# sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
root@ip-172-31-87-198:/home/ubuntu# echo "CRI runtime installed successfully"
CRI runtime installed successfully
root@ip-172-31-87-198:/home/ubuntu# 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
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io
/core:/stable:/v1.29/deb/ /
root@ip-172-31-87-198:/home/ubuntu# sudo apt-get update -y
sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-*" kubeadm="1.29.0-*"

sudo apt-get update -y
sudo apt-get install -y jq
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri
```

---

```
0 upgraded, 0 newly installed, 0 to remove and 200 not upgraded.
root@ip-172-31-87-198:/home/ubuntu# sudo systemctl enable --now kubelet
sudo systemctl start kubelet
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm config images pull
I0918 09:59:24.968795 3577 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm init
I0918 09:59:41.619600 3915 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
```

---



## Commands to execute on master node:

### 1. Pull Kubernetes Control Plane Images:

- Command: `sudo kubeadm config images pull`
- Explanation: Downloads the required container images for the Kubernetes Control Plane components.

```
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm config images pull
I0918 09:59:24.968795    3577 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[config/images] Pulled registry.k8s.io/kube-apiserver:v1.29.9
[config/images] Pulled registry.k8s.io/kube-controller-manager:v1.29.9
[config/images] Pulled registry.k8s.io/kube-scheduler:v1.29.9
[config/images] Pulled registry.k8s.io/kube-proxy:v1.29.9
[config/images] Pulled registry.k8s.io/coredns/coredns:v1.11.1
[config/images] Pulled registry.k8s.io/pause:3.9
[config/images] Pulled registry.k8s.io/etcd:3.5.10-0
```

### 2. Initialize the Kubernetes Cluster:

- Command: `sudo kubeadm init`
- Explanation: Initializes the Kubernetes Control Plane on the Master node

```
root@ip-172-31-87-198:/home/ubuntu# sudo kubeadm init
I0918 09:59:41.619600    3915 version.go:256] remote version is much newer: v1.3
1.0; falling back to: stable-1.29
[init] Using Kubernetes version: v1.29.9
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your inte
rnet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
images pull'
W0918 09:59:41.935262    3915 checks.go:835] detected that the sandbox image "re
gistry.k8s.io/pause:3.10" of the container runtime is inconsistent with that use
d by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CR
I sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-87-198 kuberne
tes.kubernetes.default.kubernetes.default.svc.kubernetes.default.svc.cluster.loc
al] and IP addresses [10.0.0.1 172.31.87.198]
```

## Configure kubectl:

- Commands:  
  
`mkdir -p "$HOME"/.kube`  
`sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config`  
`sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config`
- Explanation: Sets up kubectl on the Master node to interact with the cluster.

```
root@ip-172-31-87-198:/home/ubuntu# mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config
```

## Install Network Plugin (Calico):

- Command: `kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml`
- Explanation: Deploys Calico as the network plugin for the cluster.

```
root@ip-172-31-87-198:/home/ubuntu# mkdir -p "$HOME"/.kube
sudo cp -i /etc/kubernetes/admin.conf "$HOME"/.kube/config
sudo chown "$(id -u)": "$(id -g)" "$HOME"/.kube/config
root@ip-172-31-87-198:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
serviceaccount/calico-cni-plugin created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico
```

---

## Generating Join Command for Worker Nodes:

- Command: `kubeadm token create --print-join-command`
- Explanation: Generates the command to join worker nodes to the Kubernetes cluster.

```
root@ip-172-31-87-198:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.87.198:6443 --token xqllei.w91ed3g6i8omszpr --discovery-token-ca-cert-hash sha256:750e1245a8b1c658b62daad553dbcd27c89ac2bb81aca75f1b686c4da2838885
root@ip-172-31-87-198:/home/ubuntu# kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
ip-172-31-80-64     Ready    <none>    15s     v1.29.0
ip-172-31-81-208    Ready    <none>    49s     v1.29.0
ip-172-31-87-198    Ready    control-plane 3m53s    v1.29.0
root@ip-172-31-87-198:/home/ubuntu#
```

---

## Commands to execute on worker nodes:

### 1. Reset Pre-flight Checks:

- Command: `sudo kubeadm reset`
- Explanation: Resets the Kubernetes setup on worker nodes, useful for ensuring a clean state before joining.

```
root@ip-172-31-81-208:/home/ubuntu# sudo kubeadm reset pre-flight checks
W0918 10:01:40.885084 3738 preflight.go:56] [reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: yes
[preflight] Running pre-flight checks
W0918 10:01:46.735217 3738 removeetcdmember.go:106] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]
```

---

## 2. Join the Cluster:

- Command: `sudo kubeadm join 172.31.87.198:6443 --token xqllei.w91ed3g6i8omszpr --discovery-token-ca-cert-hash sha256:750e1245a8b1c658b62daad553dbcd27c89ac2bb81aca75f1b686c4da2838885 --v=5`
- Explanation: Joins the worker node to the Kubernetes cluster using the join command generated from the Master node.

```
root@ip-172-31-81-208:/home/ubuntu# sudo kubeadm join 172.31.87.198:6443 --token
xqllei.w91ed3g6i8omszpr --discovery-token-ca-cert-hash sha256:750e1245a8b1c658b
62daad553dbcd27c89ac2bb81aca75f1b686c4da2838885 --v=5
I0918 10:02:52.014281    3760 join.go:413] [preflight] found NodeName empty; usi
ng OS hostname as NodeName
I0918 10:02:52.015029    3760 initconfiguration.go:122] detected and using CRI s
ocket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I0918 10:02:52.015127    3760 preflight.go:93] [preflight] Running general check
s
I0918 10:02:52.015154    3760 checks.go:280] validating the existence of file /e
tc/kubernetes/kubelet.conf
I0918 10:02:52.015223    3760 checks.go:280] validating the existence of file /e
tc/kubernetes/bootstrap-kubelet.conf
I0918 10:02:52.015235    3760 checks.go:104] validating the container runtime
I0918 10:02:52.034714    3760 checks.go:639] validating whether swap is enabled
or not
I0918 10:02:52.034785    3760 checks.go:370] validating the presence of executab
le crictl
```

---

## Verify Cluster Connection

### 1. Check Nodes:

- **Command:** `kubectl get nodes`
- **Explanation:** Verifies that all nodes (Master and Worker) have successfully joined the cluster and are in a ready state.

```
~~~~~
root@ip-172-31-87-198:/home/ubuntu# kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
ip-172-31-80-64     Ready     <none>    15s     v1.29.0
ip-172-31-81-208     Ready     <none>    49s     v1.29.0
ip-172-31-87-198     Ready     control-plane 3m53s   v1.29.0
root@ip-172-31-87-198:/home/ubuntu#
```

---