Name:Aditya Dubey Div:D15C Roll no:10

EXPERIMENT NO. 3

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

1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed
   To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z
worker-2	i-03e737ce896f3e8ef	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b
master	i-0650be926fe6acc7d	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b
worker-1	i-00f42f05b3b3762ed	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b

Set up Docker

Kubernetes requires a CRI-compliant container engine runtime such as Docker, containerd, or CRI-O. This article shows you how to deploy Kubernetes using Docker.

Install Docker on each server node by executing the steps below:

1. Update the package list:

```
sudo apt update
```

```
ubuntu@ip-172-31-92-255:~$ sudo apt-get update
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 [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [351 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [502 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [123 kB]
```

2. Install Docker with the following command:

```
sudo apt install docker.io -y
```

```
root@ip-172-31-92-237:/home/ubuntu# sudo apt-get install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docke
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 130 not upgraded.
Need to get 76.8 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubun
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubu
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.1
```

- cat <<EOF | sudo tee /etc/docker/daemon.json
  {
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
  "max-size": "100m"
  },
  "storage-driver": "overlay2"
  }
  EOF

```
root@ip-172-31-92-237:/home/ubuntu# cd /etc/docker
root@ip-172-31-92-237:/etc/docker# cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
EOF
{
"exec-opts": ["native.cgroupdriver=systemd"],
"log-driver": "json-file",
"log-opts": {
"max-size": "100m"
},
"storage-driver": "overlay2"
}
root@ip-172-31-92-237:/etc/docker#
```

- sudo systemctl enable docker
- sudo systemctl daemon-reload

```
root@ip-172-31-92-237:/home/ubuntu# sudo systemctl start docker
root@ip-172-31-92-237:/home/ubuntu# sudo systemctl enable docker
root@ip-172-31-92-237:/home/ubuntu#
```

Install Kubernetes

Setting up Kubernetes on an Ubuntu system involves adding the Kubernetes repository to the APT sources list and installing the relevant tools. Follow the steps below to install Kubernetes on all the nodes in your cluster.

Step 1: Add Kubernetes Signing Key

Since Kubernetes comes from a non-standard repository, download the signing key to ensure the software is authentic.

On each node, use the curl command to download the key and store it in a safe place (default is */etc/apt/keyrings/*:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
root@ip-172-31-92-237:/etc/docker# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/
ring.gpg
root@ip-172-31-92-237:/etc/docker# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:
etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@ip-172-31-92-237:/etc/docker# sudo apt update
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 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  InRelease [1186 B]
Err:7 https://packages.cloud.google.com/apt kubernetes-xenial Release
  404  Not Found [IP: 172.253.122.100 443]
Get:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.30/deb  Packages [9318 B]
Reading package lists... Done
E: The repository 'http://apt.kubernetes.io kubernetes-xenial Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
root@ip-172-31-92-237:/etc/docker# sudo apt install kubeadm kubelet kubectl
Reading package lists... Done
Building dependency tree... Done
```

Step 2: Add Software Repositories

Kubernetes is not included in the default Ubuntu repositories. To add the Kubernetes repository to your list, enter this command on each node:

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
root@master-node:/etc/docker# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' | sudo tee /etc/
pt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /
root@master-node:/etc/docker# sudo apt update
```

Ensure all packages are up to date:

```
sudo apt update
```

Step 3: Install Kubernetes Tools

Each Kubernetes deployment consists of three separate tools:

- Kubeadm. A tool that initializes a Kubernetes cluster by fast-tracking the setup using community-sourced best practices.
- Kubelet. The work package that runs on every node and starts containers. The tool gives you command-line access to clusters.
- Kubectl. The command-line interface for interacting with clusters.

Execute the following commands on each server node to install the Kubernetes tools:

1. Run the install command:

```
sudo apt install kubeadm kubelet kubectl
```

```
root@ip-172-31-92-237:/etc/docker# sudo apt install kubeadm kubelet kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 130 not upgraded.
Need to get 93.5 MB of archives.
After this operation, 341 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

2. Mark the packages as held back to prevent automatic installation, upgrade, or removal:

```
sudo apt-mark hold kubeadm kubelet kubectl
```

```
root@ip-172-31-92-237:/etc/docker# sudo apt-mark hold kubeadm kubelet kubectl
kubeadm set on hold.
kubelet set on hold.
kubectl set on hold.
```

> Note: The process presented in this tutorial prevents APT from automatically updating Kubernetes. For instructions on how to update, please see the official developers' instructions.

3. Verify the installation with:

```
kubeadm version
```

```
root@master-node:/etc/docker# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"30", GitVersion:"v1.30.5", GitCommit:"74e84a90c725047b1328ff3d589fedb1cb7a
"2024-09-12T00:17:07Z", GoVersion:"go1.22.6", Compiler:"gc", Platform:"linux/amd64"}
```

The output of the `version` command shows basic deployment information.

> Note: BMC offers balanced and affordable server instances well suited for containerized services deployment. To simplify and streamline the process, deploy Kubernetes clusters on BMC using our Rancher solution.

Deploy Kubernetes

With the necessary tools installed, proceed to deploy the cluster. Follow the steps below to make the necessary system adjustments, initialize the cluster, and join worker nodes.

Step 1: Prepare for Kubernetes Deployment

This section shows you how to prepare the servers for a Kubernetes deployment. Execute the steps below on each server node:

1. Disable all swap spaces with the swapoff command:

```
sudo swapoff -a
```

Then use the sed command below to make the necessary adjustments to the */etc/fstab* file:

```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

```
root@ip-172-31-92-237:/home/ubuntu# sudo swapoff -a
root@ip-172-31-92-237:/home/ubuntu# sed -i '/ swap / s/^/#/' /etc/fstab
```

2. Load the required containerd modules. Start by opening the containerd configuration file in a text editor, such as nano:

```
sudo nano /etc/modules-load.d/containerd.conf
```

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/modules-load.d/containerd.conf
root@ip-172-31-92-237:/etc/docker#
```

3. Add the following two lines to the file:

```
overlay
br_netfilter
```

```
  GNU nano 7.2
overlay
br_netfilter
```

Save the file and exit.

4. Next, use the modprobe command to add the modules:

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

```
root@ip-172-31-92-237:/etc/docker# sudo modprobe overlay
root@ip-172-31-92-237:/etc/docker# sudo modprobe br_netfilter
```

5. Open the kubernetes.conf file to configure Kubernetes networking:

```
sudo nano /etc/sysctl.d/kubernetes.conf
```

```
root@ip-172-31-92-255:/etc/docker# sudo nano /etc/sysctl.d/kubernetes.conf
root@ip-172-31-92-255:/etc/docker#
```

6. Add the following lines to the file:

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
```

Save the file and exit.

7. Reload the configuration by typing:

```
sudo sysctl --system
```

```
root@ip-172-31-92-237:/etc/docker# 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 ...
* Applying /etc/sysctl.d/10-zeropage.conf ...
* Applying /etc/sysctl.d/50-cloudimg-settings.conf ...
* Applying /usr/lib/sysctl.d/50-pid-max.conf ...
* Applying /etc/sysctl.d/99-cloudimg-ipv6.conf ...
* Applying /usr/lib/sysctl.d/99-protect-links.conf ...
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/kubernetes.conf ...
* Applying /etc/sysctl.conf ...
kernel.apparmor_restrict_unprivileged_userns = 1
kernel.printk = 4 4 1 7
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
kernel.kptr_restrict = 1
kernel.sysrq = 176
```

Step 2: Assign Unique Hostname for Each Server Node

1. Decide which server will be the master node. Then, enter the command on that node to name it accordingly:

```
sudo hostnamectl set-hostname master-node
```

```
net.ipv4.ip_forward = 1
root@ip-172-31-92-237:/etc/docker# sudo hostnamectl set-hostname master-node
root@ip-172-31-92-237:/etc/docker#
```

2. Next, set the hostname on the first worker node by entering the following command:

```
sudo hostnamectl set-hostname worker01 and worker02
```

```
root@ip-172-31-86-115:/etc/docker# sudo hostnamectl set-hostname worker01
root@ip-172-31-86-115:/etc/docker#
```

```
net.ipv4.ip_forward = 1
root@ip-172-31-92-255:/etc/docker# sudo hostnamectl set-hostname worker02
root@ip-172-31-92-255:/etc/docker#
```

3. Edit the hosts file on each node by adding the IP addresses and hostnames of the servers that will be part of the cluster.

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/hosts
root@ip-172-31-92-237:/etc/docker#
```

```
  GNU nano 7.2
127.0.0.1 localhost
3.89.108.169 master-node
44.202.47.176 worker01
34.238.119.65 worker02
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

4. Restart the terminal application to apply the hostname change.

Step 3: Initialize Kubernetes on Master Node

Once you finish setting up hostnames on cluster nodes, switch to the master node and follow the steps to initialize Kubernetes on it:

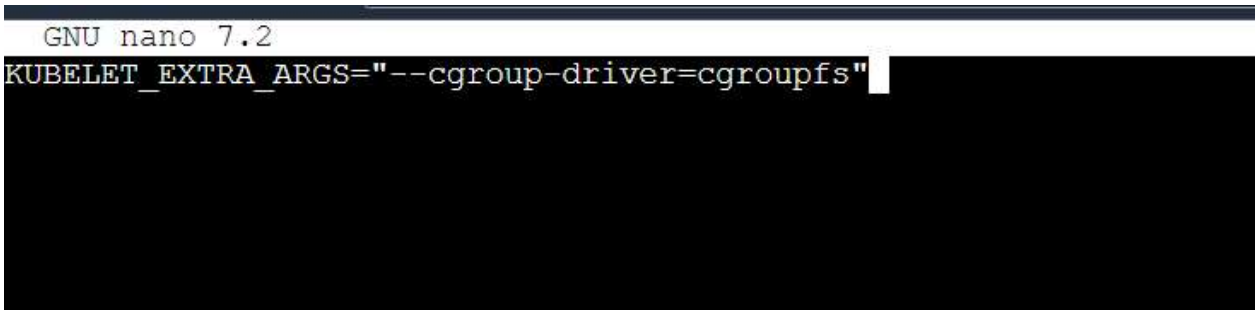
1. Open the kubelet file in a text editor.

```
sudo nano /etc/default/kubelet
```

```
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/hosts
root@ip-172-31-92-237:/etc/docker# sudo nano /etc/default/kubelet
root@ip-172-31-92-237:/etc/docker#
```

2. Add the following line to the file:

```
KUBELET_EXTRA_ARGS="--cgroup-driver=cgroupfs"
```



Save and exit.

3. Reload the configuration and restart the kubelet:

```
sudo systemctl daemon-reload && sudo systemctl restart kubelet
```

```
root@ip-172-31-92-237:/# sudo systemctl daemon-reload && sudo systemctl restart kubelet
root@ip-172-31-92-237:/#
```

6. Open the kubeadm configuration file:

```
sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

```
root@ip-172-31-92-237:/# sudo nano /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
root@ip-172-31-92-237:/#
```

7. Add the following line to the file:

```
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
```

```
  GNU nano 7.2                                  /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf *
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernet
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variable dy
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced from
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

Save the file and exit.

9. Reload the configuration and restart the kubelet:

```
sudo systemctl daemon-reload && sudo systemctl restart kubelet
```

```
root@ip-172-31-92-237:/# sudo systemctl daemon-reload && sudo systemctl restart kubelet
root@ip-172-31-92-237:/#
```

10. Finally, initialize the cluster by typing:

```
sudo kubeadm init --control-plane-endpoint=master-node
--upload-certs —-ignore-preflight-errors=all
```

```
root@master-node:/etc/docker# sudo kubeadm init --control-plane-endpoint=master-node --upload-certs --ignore-preflight-errors=all
I0914 09:48:13.230610    9946 version.go:256] remote version is much newer: v1.31.0; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.4
[preflight] Running pre-flight checks
        [WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
        [WARNING Mem]: the system RAM (957 MB) is less than the minimum 1700 MB
        [WARNING Port-6443]: Port 6443 is in use
        [WARNING Port-10259]: Port 10259 is in use
        [WARNING Port-10257]: Port 10257 is in use
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml]: /etc/kubernetes/manifests/kube-apiserver.yaml already
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-controller-manager.yaml]: /etc/kubernetes/manifests/kube-controller-ma
        [WARNING FileAvailable--etc-kubernetes-manifests-kube-scheduler.yaml]: /etc/kubernetes/manifests/kube-scheduler.yaml already
        [WARNING FileAvailable--etc-kubernetes-manifests-etcd.yaml]: /etc/kubernetes/manifests/etcd.yaml already exists
        [WARNING Port-10250]: Port 10250 is in use
        [WARNING Port-2379]: Port 2379 is in use
        [WARNING Port-2380]: Port 2380 is in use
```

Once the operation finishes, the output displays a `kubeadm join` command at the bottom. Make a note of this command, as you will use it to join the worker nodes to the cluster.

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join master-node:6443 --token fjlof5.8fnxwt2begoiwzrf \
        --discovery-token-ca-cert-hash sha256:d79687f16bbb4c7d8c78a4c02995b1f6a906afa8aaefc31dc66695800c31aed6
root@master-node:/etc/docker# ^C
root@master-node:/etc/docker#
```

11. Create a directory for the Kubernetes cluster:

```
mkdir -p $HOME/.kube
```

```
root@master-node:/# mkdir -p $HOME/.kube
root@master-node:/#
```

12. Copy the configuration file to the directory:

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

13. Change the ownership of the directory to the current user and group using the chown command:

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

```
root@master-node:/etc/docker# cd ../..
root@master-node:/# mkdir -p $HOME/.kube
root@master-node:/# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@master-node:/# sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@master-node:/# kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
root@master-node:/# kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/master-node untainted
root@master-node:/#
```

Step 4: Deploy Pod Network to Cluster

A pod network is a way to allow communication between different nodes in the cluster. This tutorial uses the Flannel node network manager to create a pod network.

Apply the Flannel manager to the master node by executing the steps below:

1. Use kubectl to install Flannel:

```
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
```

```
root@master-node:/# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/maste
namespace/kube-flannel unchanged
clusterrole.rbac.authorization.k8s.io/flannel unchanged
clusterrolebinding.rbac.authorization.k8s.io/flannel unchanged
serviceaccount/flannel unchanged
configmap/kube-flannel-cfg unchanged
daemonset.apps/kube-flannel-ds created
```

2. Untaint the node:

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

```
root@master-node:/# kubectl taint nodes --all node-role.kubernetes.io/control-plane-
node/master-node untainted
root@master-node:/#
```

Step 5: Join Worker Node to Cluster

Repeat the following steps on each worker node to create a cluster:

1. Stop and disable AppArmor:

```
sudo systemctl stop apparmor && sudo systemctl disable apparmor
```

2. Restart containerd:

```
sudo systemctl restart containerd.service
```

```
Removed "/etc/systemd/system/sysinit.target.wants/apparmor.service".
root@worker02:/etc/docker# sudo systemctl restart containerd.service
root@worker02:/etc/docker#
```

3. Apply the kubeadm join command from Step 3 on worker nodes to connect them to the master node. Prefix the command with `sudo`:

```
sudo kubeadm join [master-node-ip]:6443 --token [token]
--discovery-token-ca-cert-hash sha256:[hash]
```

```
root@worker01:/home/ubuntu# kubeadm join master-node:6443 --token fjlof5.8fnxwt2begoiwzrf
95b1f6a906afa8aaefc31dc66695800c31aed6
```

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Replace [master-node-ip], [token], and [hash] with the values from the kubeadm join command output.

4. After a few minutes, switch to the master server and enter the following command to check the status of the nodes:

```
kubectl get nodes
```

```
Every 2.0s: kubectl get nodes

NAME                              STATUS   ROLES           AGE     VERSION
ip-172-31-81-63.ec2.internal      Ready    control-plane   29m     v1.30.4
ip-172-31-87-137.ec2.internal     Ready    <none>          5m58s   v1.30.4
ip-172-31-92-18.ec2.internal      Ready    <none>          5m53s   v1.30.4
```

The system displays the master node and the worker nodes in the cluster.