Experiment No. 3

Objective: To understand the Kubernetes Cluster Architecture and to set up a Kubernetes Cluster on Linux Machines/Cloud.

Procedure:

1. **Instance Setup:**
   - Create three EC2 instances using Amazon Linux as the operating system.
   - Ensure that SSH traffic is allowed from any source.
   - For optimal performance, choose an instance type of at least `t2.medium`, as Kubernetes recommends a minimum of 2 vCPUs.

Name	Instance ID	Instance state	Instance type
kube-master	i-00aa79ac09d7462c0	Running	t2.medium
kube-worker1	i-0bab86cd3fbfcb40a	Running	t2.medium
kube-worker2	i-00dcfd302ffd80dda	Running	t2.medium

2. **SSH Access:**
   - SSH into each of the three machines using separate terminal windows: `ssh -i <keyname>.pem ubuntu@<public_ip_address>`

```
quantum@machine ~/Downloads ssh -i "ec-2-ubuntu.pem" ec2-user@ec2-3-88-111-183.compute-1.amazonaws.com
The authenticity of host 'ec2-3-88-111-183.compute-1.amazonaws.com (3.88.111.183)' can't be established.
ED25519 key fingerprint is SHA256:pQu+xs9foYbY3de1twjZcVVAOzmGwGv6PHmVruF/Q1s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-88-111-183.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

3. **Docker Installation and Configuration:**
   - On all three machines, install Docker with the command: `sudo yum install docker -y`

   - Configure Docker to use `systemd` as the cgroup driver by creating and editing the `daemon.json` file:
     - Change directory to `/etc/docker`

- Use the command `cat <<EOF | sudo tee /etc/docker/daemon.json` followed by the JSON configuration details and end with `EOF`

- Enable and restart Docker: `sudo systemctl enable docker sudo systemctl daemon-reload sudo systemctl restart docker docker -v`

```
[ec2-user@ip-172-31-92-18 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:09:56 ago on Wed Sep 11 15:19:39 2024.
Dependencies resolved.
================================================================================
 Package                                                    Architecture
================================================================================
Installing:
 docker                                                     x86_64
Installing dependencies:
 containerd                                                 x86_64
 iptables-libs                                              x86_64
 iptables-nft                                               x86_64
 libcgroup                                                  x86_64
 libnetfilter_conntrack                                     x86_64
 libnfnetlink                                               x86_64
 libnftnl                                                   x86_64
 pigz                                                       x86_64
 runc                                                       x86_64

Transaction Summary
```

4. **Kubernetes Installation:**
   - Disable SELinux before configuring `kubelet`: `sudo setenforce 0 sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config`

   - Add the Kubernetes repository and install Kubernetes components:
     - Use the command `cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo` followed by the repository configuration details and end with `EOF sudo yum update sudo yum install -y kubelet kubeadm kubectl 0--disableexcludes=kubernetes`

- Configure networking for bridging: `sudo swapoff -a echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf sudo sysctl -p`

```
[ec2-user@ip-172-31-81-63 docker]$ sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
docker -v

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-81-63 docker]$
```

```
[ec2-user@ip-172-31-81-63 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-81-63 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

```
[ec2-user@ip-172-31-81-63 docker]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:01:34 ago on Wed Sep 11 15:39:05 2024.
Dependencies resolved.
================================================================================================================================================
 Package                                                 Architecture                                 Version
================================================================================================================================================
Installing:
 kubeadm                                                 x86_64                                       1.30.4-150500.1.1
 kubectl                                                 x86_64                                       1.30.4-150500.1.1
 kubelet                                                 x86_64                                       1.30.4-150500.1.1
Installing dependencies:
 conntrack-tools                                         x86_64                                       1.4.6-2.amzn2023.0.2
 cri-tools                                               x86_64                                       1.30.1-150500.1.1
 kubernetes-cni                                          x86_64                                       1.4.0-150500.1.1
 libnetfilter_cthelper                                   x86_64                                       1.0.0-21.amzn2023.0.2
 libnetfilter_cttimeout                                  x86_64                                       1.0.0-19.amzn2023.0.2
 libnetfilter_queue                                      x86_64                                       1.0.5-2.amzn2023.0.2
 socat                                                   x86_64                                       1.7.4.2-1.amzn2023.0.2

Transaction Summary
================================================================================================================================================
Install  10 Packages
```

5. **Master Node Setup:**
   - Initialize the Kubernetes master node (perform only on the master machine): `sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all`

   - After initialization, set up the Kubernetes configuration on the master node: `mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config`

- Save the generated join command from the output for worker nodes. This command is unique and specific to your cluster setup: `kubeadm join 172.31.91.120:6443 --token r8j60r.n1j6h0klbewvoka5\--discovery-token-ca-cert-hashsha256 :dd8426260174d673303aef17717f740772fcf7ee782245bc653eecf4a13 05da7`

```
ubuntu@ip-172-31-28-117:~$ sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
        --discovery-token-ca-cert-hash sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.396793ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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.
```

- Deploy the Flannel networking plugin to enable pod communication: `kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml`

- Check the status of the pods to ensure they are running: `kubectl get pods --all-namespaces`

```
ubuntu@ip-172-31-27-176:~$ kubectl get nodes
NAME               STATUS     ROLES           AGE     VERSION
ip-172-31-18-135   NotReady   <none>          88s     v1.31.1
ip-172-31-27-176   NotReady   control-plane   10m     v1.31.1
ip-172-31-28-117   NotReady   <none>          2m58s   v1.31.1
```

```
ubuntu@ip-172-31-27-176:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

sudo systemctl status kubelet

```
ubuntu@ip-172-31-27-176:~$ sudo systemctl status kubelet
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; preset: enabled)
    Drop-In: /usr/lib/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: active (running) since Mon 2024-09-16 15:40:01 UTC; 11min ago
       Docs: https://kubernetes.io/docs/
   Main PID: 5989 (kubelet)
      Tasks: 10 (limit: 4676)
     Memory: 32.6M (peak: 33.2M)
        CPU: 10.705s
     CGroup: /system.slice/kubelet.service
             └─5989 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/

Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497458    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497516    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497569    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497620    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497669    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:29 ip-172-31-27-176 kubelet[5989]: I0916 15:51:29.497719    5989 reconciler_common.go:245] "operationExecutor.VerifyControllerAttachedVolume s
Sep 16 15:51:31 ip-172-31-27-176 kubelet[5989]: E0916 15:51:31.605091    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
Sep 16 15:51:32 ip-172-31-27-176 kubelet[5989]: I0916 15:51:32.366237    5989 scope.go:117] "RemoveContainer" containerID="f44f06967c5b3e567e07841a7b4352ae
Sep 16 15:51:36 ip-172-31-27-176 kubelet[5989]: E0916 15:51:36.606675    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
Sep 16 15:51:41 ip-172-31-27-176 kubelet[5989]: E0916 15:51:41.608404    5989 kubelet.go:2902] "Container runtime network not ready" networkReady="NetworkR
```

Now Run command kubectl get nodes -o wide we can see Status is ready.

```
ubuntu@ip-172-31-27-176:~$ ubuntu@ip-172-31-27-176:~$ kubectl get nodes -o wide
NAME              STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-18-135  Ready    <none>          6m19s   v1.31.1   172.31.18.135   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-27-176  Ready    control-plane   15m     v1.31.1   172.31.27.176   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
ip-172-31-28-117  Ready    <none>          7m49s   v1.31.1   172.31.28.117   <none>        Ubuntu 24.04 LTS   6.8.0-1012-aws   containerd://1.7.12
```

6. **Worker Node Setup:**
    - On each worker node, install the required package and configure `kubelet`: `sudo yum install iproute-tc -y sudo systemctl enable kubelet sudo systemctl restart kubelet`

    - Join the worker nodes to the Kubernetes cluster using the join command from the master node: `kubeadm join 172.31.91.120:6443 --token r8j60r.n1j6h0klbewvoka5\--discovery-token-ca-cert-hashsha256:dd8426260174d673303aef17717f740772fcf7ee782245bc653eecf4a1305da7`

Node 1

```
ubuntu@ip-172-31-28-117:~$ sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
        --discovery-token-ca-cert-hash sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6

[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 501.396793ms
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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.

ubuntu@ip-172-31-28-117:~$
```

Node 2

```
ubuntu@ip-172-31-18-135:~$ sudo kubeadm join 172.31.27.176:6443 --token ttay2x.n0sqeukjai8sgfg3 \
        --discovery-token-ca-cert-hash sha256:d6fc5fb7e984c83e2807780047fec6c4f2acfe9da9184ecc028d77157608fbb6
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-check] Waiting for a healthy kubelet at http://127.0.0.1:10248/healthz. This can take up to 4m0s
[kubelet-check] The kubelet is healthy after 1.001003808s
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap

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.

ubuntu@ip-172-31-18-135:~$
```

7. **Verify Node Status:**
    - On the master node, verify that the worker nodes have successfully joined the cluster by running: `watch kubectl get nodes`

Or run kubectl get nodes

```
ubuntu@ip-172-31-27-176:~$ kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-18-135   Ready    Node2           24m   v1.31.1
ip-172-31-27-176   Ready    control-plane   33m   v1.31.1
ip-172-31-28-117   Ready    Node1           25m   v1.31.1
ubuntu@ip-172-31-27-176:~$
```

Conclusion:

Setting up the Kubernetes cluster involved several challenges. Network configuration issues initially hindered the deployment of the Flannel plugin, requiring open ports and a functional Kubernetes API server. Disabling SELinux and adjusting firewall rules were essential for proper communication between

components. Worker nodes experienced difficulties with the kubelet service, which needed to be correctly configured and restarted. Additionally, accurate copying of the join command, including the token and discovery-token-ca-cert-hash, was crucial for integrating worker nodes into the cluster. These issues underscored the need for precise configuration and troubleshooting to achieve a stable Kubernetes setup.