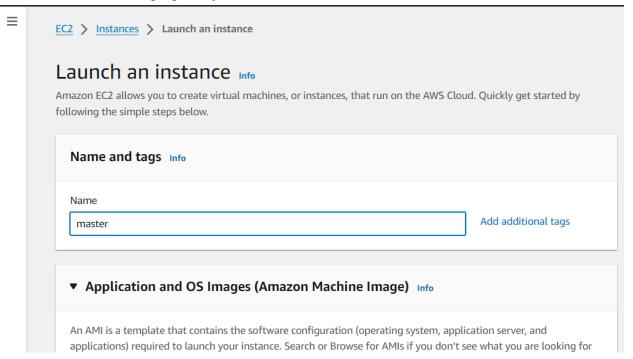
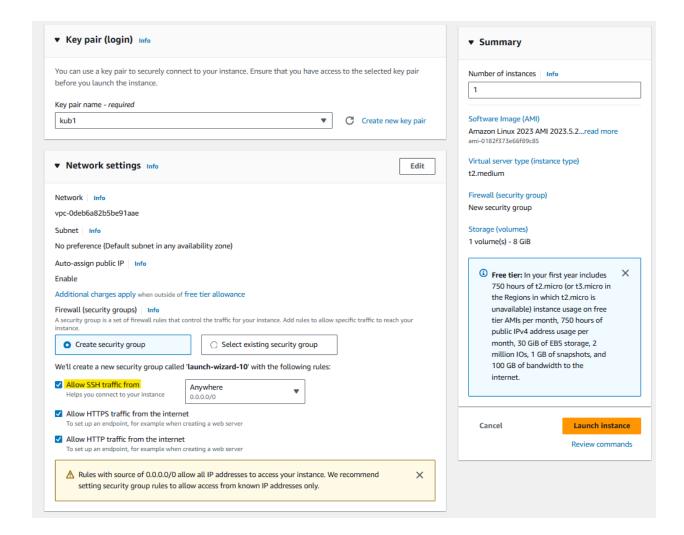
Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud

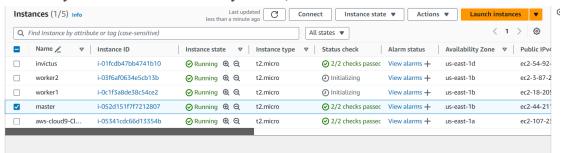
1. Create 3 EC-2 instances with all running on Amazon Linux as OS with inbound SSH allowed and the proper key



To efficient run kubernetes cluster select instance type of at least t2.medium as kubernetes recommends at least 2 vCPU to run smoothly



In this way create 3 instances namely master, worker-1 and worker-2



- 2. SSH into all 3 machines each in separate terminal
 - a. You can do it through the aws console directly

Or

b. Locate your key from the Downloads folder and open it in cmd and paste this command

ssh -i <-your-key->.pem ec2-user<ip-address of instance>

```
C:\Users\Avan\Downloads>ssh -i C:\Users\Avan\Downloads\kub1.pem ec2-user@54.85.79.186
The authenticity of host '54.85.79.186 (54.85.79.186)' can't be established.
ED25519 key fingerprint is SHA256:ffz946cqlxbNvBsPqtNcxLlfXmhW8VJhyLD4n4jStto.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.85.79.186' (ED25519) to the list of known hosts.
        ####
                     Amazon Linux 2023
      \_####\
  ~~
         \###|
           \#/
V~' '->
                     https://aws.amazon.com/linux/amazon-linux-2023
  ~~
       _/m/'
Last login: Sat Sep 14 06:03:27 2024 from 18.206.107.27
[ec2-user@ip-172-31-20-75 ~]$
```

With this you can continue your commands through local terminal

3. From now on, until mentioned, perform these steps on all 3 machines.

Install Docker

sudo yum install docker -y

```
[ec2-user@ip-172-31-31-212 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:33:43 ago on Thu Sep 12 13:11:13 2024.
Dependencies resolved.
                                                                                             Architecture
Installing:
                                                                                             x86_64
                                                                                                                                                                25.0.6-1.amzn2023.0.2
docker
Installing dependencies:
containerd
iptables-libs
                                                                                             x86_64
x86_64
x86_64
x86_64
x86_64
                                                                                                                                                                1.7.20-1.amzn2023.0.1
                                                                                                                                                                1.8.8-3.amzn2023.0.2
1.8.8-3.amzn2023.0.2
  iptables-nft
 libegroup
libnetfilter_conntrack
libnfnetlink
libnftnl
                                                                                                                                                                3.0-1.amzn2023.0.1
1.0.8-2.amzn2023.0.2
1.0.1-19.amzn2023.0.2
                                                                                                                                                                1.2.2-2.amzn2023.0.2
 pigz
runc
                                                                                                                                                                2.5-1.amzn2023.0.3
                                                                                                                                                                1.1.13-1.amzn2023.0.1
Transaction Summary
```

Then, configure cgroup in a daemon.json file by using following commands

• cd /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</li>
```

```
[ec2-user@ip-172-31-20-75 ~]$ cd /etc/docker
[ec2-user@ip-172-31-20-75 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"
[ec2-user@ip-172-31-20-75 docker]$ ls
daemon.json kubectl
```

- sudo systemctl enable docker
- sudo systemctl daemon-reload
- sudo systemctl restart docker
- docker -v

```
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-31-212 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-31-212 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-31-212 docker]$ ■
```

4. Install Kubernetes on all 3 machines

SELinux needs to be disabled before configuring kubelet

- sudo setenforce 0
- sudo sed -i 's/\SELINUX=enforcing\\$/SELINUX=permissive/' /etc/selinux/config

```
[ec2-user@ip-172-31-26-2 docker]$ sudo setenforce 0
[ec2-user@ip-172-31-26-2 docker]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-26-2 docker]$
```

Add kubernetes repository (paste in terminal)

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo [kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.30/rpm/repodata/repomd.xml.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

Type following commands to install set of kubernetes packages:

- sudo yum update
- sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

Package	Architecture	Version	
Installing:			
kubeadm	x86_64	1.30.5-150500.1.1	
kubectl	x86_64	1.30.5-150500.1.1	
kubelet	x86_64	1.30.5-150500.1.1	
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	
Fransaction Summary			
Total download size: 53 M Installed size: 292 M Downloading Packages: (1/9): libnetfilter_cttimeout-1.0.0-19.am (2/9): libnetfilter_cthelper-1.0.0-21.amz (3/9): libnetfilter_queue-1.0.5-2.amzn023. (4/9): contrack-tools-1.4.6-2.amzn023.0 (5/9): kubeadm-1.30.5-150500.1.1.x86_64.r (6/9): kubectl-1.30.5-150500.1.1.x86_64.r (7/9): cri-tools-1.30.1-150500.1.1.x86_64.r (8/9): kubernetes-cni-1.4.0-150500.1.1.x86_(9/9): kubelet-1.30.5-150500.1.1.x86_64.r	n2023.0.2.x86 64.rpm 3.0.2.x86_64.rpm .2.x86_64.rpm pm rpm .rpm 6.64.rpm		
Fingerprint: DE15 B144 86CD 377B 9E87 6E	<pre><isv:kubernetes@build.opensuse.org>" 1A 2346 54DA 9A29 6436 table:/v1.30/rpm/repodata/repomd.xml.key</isv:kubernetes@build.opensuse.org></pre>		

After installing Kubernetes, we need to configure internet options to allow bridging.

- sudo swapoff -a
- echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
- sudo sysctl -p

5. Perform this ONLY on the Master machine

Initialize kubernetes by typing below command

• sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=all

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy
Your Kubernetes control-plane has initialized successfully!
To start using your cluster, you need to run the following as a regular user:
 mkdir -p $HOME/.kube
 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
 sudo chown $(id -u):$(id -g) $HOME/.kube/config
Alternatively, if you are the root user, you can run:
 export KUBECONFIG=/etc/kubernetes/admin.conf
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
 https://kubernetes.io/docs/concepts/cluster-administration/addons/
Then you can join any number of worker nodes by running the following on each as root:
kubeadm join 172.31.26.2:6443 --token 6cjsz0.8ei243v0zn9k7erg \
         -discovery-token-ca-cert-hash sha256:abd917ec30e12c5616bf647a3d174bef3d271e92c30b8f2f7768cfb3181341d4
[ec2-user@ip-172-31-26-2 docker]$
```

Copy the mkdir and chown commands from the top and execute them

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Copy this join link and save it in clipboard (copy from your output as it different for each master instance)

Example:

```
kubeadm join 172.31.20.75:6443 --token 66kg9u.2bc0kze31hrwbzvr \
--discovery-token-ca-cert-hash
sha256:5e478da328b199e17d9b5da68e78bc9a6daab2043b05860552f4c184a7b3cb66
```

Then, add a common networking plugin called flamel file as mentioned in the code.

Command:

kubectl apply -f

https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

```
[ec2-user@ip-172-31-26-2 docker]$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml namespace/kube-flannel created clusterrole.rbac.authorization.k8s.io/flannel created clusterrolebinding.rbac.authorization.k8s.io/flannel created serviceaccount/flannel created configmap/kube-flannel-orgered configmap/kube-flannel-orgered daemonset.apps/kube-flannel-ds created
```

6. Perform this ONLY on the worker machines

Paste the below command on all 2 worker machines

- sudo yum install iproute-tc -y
- sudo systemctl enable kubelet
- sudo systemctl restart kubelet

```
Repository
                                                                                 x86 64
                                                                                                                                                             5.10.0-2.amzn2023.0.5
                                                                                                                                                                                                                                                                                                                                                              455 k
                                                                                                                                                                                                                                                                         amazonlinux
ransaction Summary
install 1 Package
Cotal download size: 455 k
installed size: 928 k
lownloading Packages:
.proute-tc-5.10.0-2.amzn2023.0.5.x86_64.rpm
                                                                                                                                                                                                                                                                                                                  4.0 MB/s | 455 kB
                                                                                                                                                                                                                                                                                                                                                      00:00
                                                                                                                                                                                                                                                                                                                  2.8 MB/s | 455 kB
                                                                                                                                                                                                                                                                                                                                                      00:00
 ransaction check succeeded.
unning transaction test
ransaction test succeeded.
unning transaction
Preparing
| iproute-to-5.10.0-2.amzn2023.0.5.x86_64
Running scriptcti proute-to-5.10.0-2.amzn2023.0.5.x86_64
Verifying
| iproute-to-5.10.0-2.amzn2023.0.5.x86_64
 nstalled:
iproute-tc-5.10.0-2.amzn2023.0.5.x86_64
Complete!
|ec2-user@in-172-31-31-212 docker1$ ■
[ec2-user@ip-172-31-17-184 docker]$ sudo systemctl enable kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service - /usr/lib/systemd/system/kubelet.service.
[ec2-user@ip-172-31-17-184 docker]$ sudo systemctl restart kubelet
[ec2-user@ip-172-31-17-184 docker]$
```

Now paste the hash that yo copied in these worker note to connect to master cluster

Now we can see in the master/control node of kubernetes that worker nodes are connected by this command

• watch kubectl get nodes

(in the master node instance)

Errors faced during the execution:

- 1. In the end kubelet might not respond or the connectivity of nodes to master might not happen
- 2. You can see this error

```
[ec2-user@ip-172-31-20-75 docker] kubectl get nodes
80914 06:14:55.95691
connection refused
80914 06:14:55.95778
connection refused
80914 06:14:55.957879
connection refused
80914 06:14:55.96197
connection refused
80914 06:14:55.96107
connection refused
80914 06:14:55.96160
connection refused
80914 06:14:55.961526
connection refused
```

3. Try to restart the kubelet from worker instance and try the commands again

Conclusion:

- The experiment aimed to deploy Kubernetes on Docker by connecting a master node to two worker nodes.
- Encountered issues with misconfigured SSH inbound rules.
- Resolved the SSH issue by correctly enabling necessary access rules.
- Realized that using t2.medium or t3 instances was essential to provide sufficient resources for Kubernetes.
- Despite adjustments, the worker nodes could not join the cluster.
- Master node was successfully initialized, but worker nodes faced issues.
- Possible causes of worker node problems included:
- Misconfigurations in kubelet setup.
- Networking challenges.
- Inability of worker nodes to communicate with the master node's API server.
- Potential incorrect firewall settings.
- Missing API server certificates.
- Errors during the kubeadm join process on the worker nodes.