

EXPERIMENT 3

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

1. Create 3 instances and name them.

Master

Worker-1

worker-2

Select a Key pair.Allow SSH

Instances (3)
[Info](#)

Last updated
less than a minute ago

Connect

Instance state ▾

Actions ▾

Launch instances

All states ▾

< 1 >

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability
<input type="checkbox"/>	worker-2	i-0aa9f89768e3a199c	Running	t2.micro	Initializing	View alarms +	us-east-1b
<input type="checkbox"/>	worker-1	i-05c2b3ba79c1d85ab	Running	t2.micro	Initializing	View alarms +	us-east-1b
<input type="checkbox"/>	master	i-0bc867eb06b7964b7	Running	t2.micro	Initializing	View alarms +	us-east-1b

- 2.Connect the instances and open the terminal to run commands to install docker and kubernets.

```

      #_
    ~\ _ #####_ Amazon Linux 2023
  ~~ \_#####\_
  ~~   \###|
  ~~     \#/ https://aws.amazon.com/linux/amazon-linux-2023
  ~~       v~' '->
      ~~~
        ~~-.-.
          /_/_/
            /m/'
[ec2-user@ip-172-31-80-190 ~]$
```

- 3.Install Docker for all 3 instances.Repeat all the steps for all three instances.

```

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-172-31-80-190 ec2-user]# sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
   Active: active (running) since Fri 2024-09-13 04:23:51 UTC; 25s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Main PID: 29009 (dockerd)
      Tasks: 7
     Memory: 29.6M
        CPU: 330ms
    CGroup: /system.slice/docker.service
            └─29009 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nfile=32768:65536

Sep 13 04:23:50 ip-172-31-80-190.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Sep 13 04:23:50 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:50.998251424Z" level=info msg="Starting up"
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:51.068745150Z" level=info msg="Loading containers: start."
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:51.526546118Z" level=info msg="Loading containers: done."
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:51.555765635Z" level=info msg="Docker daemon" commit=b08a51
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:51.556129762Z" level=info msg="Daemon has completed initial
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal dockerd[29009]: time="2024-09-13T04:23:51.598513281Z" level=info msg="API listen on /run/docker.sock"
Sep 13 04:23:51 ip-172-31-80-190.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-20/20 (END)

```

4. Install kubernetes using intsaall kubeadms and get code from there.

1. Set SELinux to `permissive` mode:

These instructions are for Kubernetes 1.31.

```

# Linux in permissive mode (effectively disabling it)
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

```

```

# This overwrites any existing configuration in /etc/yum.repos.d/.
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/
enabled=1
gpgcheck=1
gpgkey=https://pkgs.k8s.io/core:/stable:/v1.31/rpm/repodata/repomd.xml
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF

```

3. Install kubelet, kubeadm and kubectl:

```

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

```

4. (Optional) Enable the kubelet service before running kubeadm:

```

sudo systemctl enable --now kubelet

```

```

Installing      : kubeadm-1.31.1-150500.1.1.x86_64      8/9
Installing      : kubectl-1.31.1-150500.1.1.x86_64      9/9
Running scriptlet: kubectl-1.31.1-150500.1.1.x86_64      9/9
Verifying       : contrack-tools-1.4.6-2.amzn2023.0.2.x86_64 1/9
Verifying       : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64 2/9
Verifying       : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64 3/9
Verifying       : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64 4/9
Verifying       : cri-tools-1.31.1-150500.1.1.x86_64      5/9
Verifying       : kubeadm-1.31.1-150500.1.1.x86_64      6/9
Verifying       : kubectl-1.31.1-150500.1.1.x86_64      7/9
Verifying       : kubelet-1.31.1-150500.1.1.x86_64      8/9
Verifying       : kubernetes-cni-1.5.1-150500.1.1.x86_64 9/9

Installed:
  contrack-tools-1.4.6-2.amzn2023.0.2.x86_64
  kubeadm-1.31.1-150500.1.1.x86_64
  kubelet-1.31.1-150500.1.1.x86_64
  libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
  cri-tools-1.31.1-150500.1.1.x86_64
  kubectl-1.31.1-150500.1.1.x86_64
  kubernetes-cni-1.5.1-150500.1.1.x86_64
  libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64

Complete!
[root@ip-172-31-83-1 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-83-1 ec2-user]#

```

5. check the repositories

```

[root@ip-172-31-83-1 ec2-user]# yum repolist
repo id                                repo name
amazonlinux                            Amazon Linux 2023 repository
kernel-livepatch                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                             Kubernetes
[root@ip-172-31-83-1 ec2-user]#

```

6. run command kubeadm

```

[root@ip-172-31-83-1 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[WARNING FileExisting-socat]: socat not found in system path
[WARNING FileExisting-tc]: tc not found in system path
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR NumCPU]: the number of available CPUs 1 is less than the required 2
[ERROR Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
[preflight] If you know what you are doing, you can make a check non-fatal with '--ignore-preflight-errors=
To see the stack trace of this error execute with --v=5 or higher
[root@ip-172-31-83-1 ec2-user]#

```

7. check repo

```

[root@ip-172-31-14-85 ec2-user]# yum repolist
repo id                                repo name
amazonlinux                            Amazon Linux 2023 repository
kernel-livepatch                       Amazon Linux 2023 Kernel Livepatch repository
kubernetes                             Kubernetes
[root@ip-172-31-14-85 ec2-user]#

```

8. Now we will be initializing the kubeadm. For that “kubeadm init” command has to be used. It may show errors but those can be ignored by using --ignore-preflighterrors=all

```

[root@ip-172-31-14-85 ec2-user]# kubeadm init --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[WARNING NumCPU]: the number of available CPUs 1 is less than the required 2
[WARNING Mem]: the system RAM (949 MB) is less than the minimum 1700 MB
[WARNING FileExisting-socat]: socat not found in system path
[WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action beforehand using 'kubeadm config images pull'
W0914 15:50:31.271160 29520 checks.go:846] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is inconsistent with that used by kubeadm. It is recommended to use "registry.k8s.io/pause:3.10" as the CRI 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-172-31-14-85.ap-southeast-2.compute.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.14.85]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-14-85.ap-southeast-2.compute.internal localhost] and IPs [172.31.14.85 127.0.0.1 ::1]

```

```

85 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-14-85.ap-southeast-2.compute.internal localhost] and IPs [172.31.14.85
127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "super-admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
[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 518.648244ms
[api-check] Waiting for a healthy API server. This can take up to 4m0s

```

```

[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests"
[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 518.648244ms
[api-check] Waiting for a healthy API server. This can take up to 4m0s
[api-check] The API server is healthy after 10.001658622s
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config" in namespace kube-system with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node ip-172-31-14-85.ap-southeast-2.compute.internal as control-plane by adding the labels: [node-role.k
ubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node ip-172-31-14-85.ap-southeast-2.compute.internal as control-plane by adding the taints [node-role.k
ubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 6lysht.48enn4gmnhof6ex8
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate cred
entials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

```

9. On successful initialization we need to copy and paste the following commands on the master machine itself:

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

```

10. Next copy and paste the join link in the worker nodes so that the worker nodes can join the cluster.

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

```

kubeadm join 172.31.14.85:6443 --token 6lysht.48enn4gmnhof6ex8 \
--discovery-token-ca-cert-hash sha256:461819c971fe032e04a78e18fde8e28755825e8468d468a2c86d88c52dba4945

```

11. After performing join commands on the worker nodes, we will get following output:

```
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.
```

12. Once again when you run `kubectl get nodes` you will now see all 3 nodes have joined the cluster.

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-85-89.ec2.internal	NotReady	control-plane	119s	v1.26.0
ip-172-31-89-46.ec2.internal	NotReady	<none>	19s	v1.26.0
ip-172-31-94-70.ec2.internal	NotReady	<none>	12s	v1.26.0

Conclusion: This experiment successfully demonstrated the creation of a Kubernetes cluster and the successful addition of all three nodes using various commands