

Advance DevOps

Experiment 3

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

Steps:

1. We will create 3 EC2 instances. One will be the master node and the other 2 will be slave/worker nodes.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input checked="" type="checkbox"/>	master	i-057a73de5d3df0746	Running	t2.micro	2/2 checks passec	View alarms
<input type="checkbox"/>	slave node 1	i-01ea28da4403edd84	Running	t2.micro	2/2 checks passec	View alarms
<input type="checkbox"/>	slave node 2	i-0a47253bfd474144d	Running	t2.micro	2/2 checks passec	View alarms

2. After the instances have been created, we will connect them one by one.

The screenshot shows the AWS Management Console interface for configuring an EC2 instance connection. At the top, there's a navigation bar with the AWS logo, 'Services' link, a search bar, and an '[Alt+S]' shortcut. A yellow warning banner at the top states: 'Port 22 (SSH) is open to all IPv4 addresses. Port 22 (SSH) is currently open to all IPv4 addresses, indicated by 0.0.0.0/0 in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. Learn more.' Below this, the 'Instance ID' is listed as 'i-07e31cd245aca1668 (Master)'. Under 'Connection Type', there are two radio buttons: 'Connect using EC2 Instance Connect' (which is selected) and 'Connect using EC2 Instance Connect Endpoint'. The 'Public IP address' is shown as '18.212.227.56'. The 'Username' field is set to 'ubuntu'. A blue note box at the bottom states: 'Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' At the bottom right, there are 'Cancel' and 'Connect' buttons.

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sgr-03ea67416cb707df4	SSH	TCP	22	Custom	Q	Delete
					0.0.0.0/0	
	All traffic	All	All	Anywher...	Q	Delete
					0.0.0.0/0	

Add rule

Cancel

Preview changes

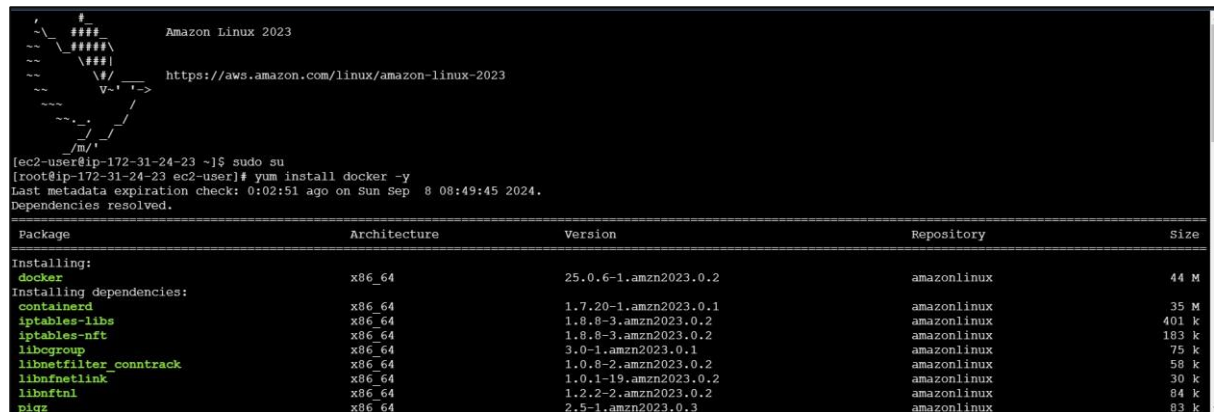
Save rules

4. Docker installation:

This step has to be performed on all the 3 instances.

The following command has to be run:

`yum install docker -y`



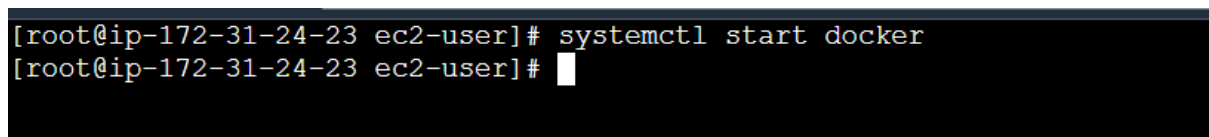
```

Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-24-23 ~]$ sudo su
[root@ip-172-31-24-23 ec2-user]# yum install docker -y
Last metadata expiration check: 0:02:51 ago on Sun Sep  8 08:49:45 2024.
Dependencies resolved.
=====
Package                                Architecture      Version            Repository          Size
-----
Installing:
docker                                x86_64            25.0.6-1.amzn2023.0.2  amazonlinux         44 M
Installing dependencies:
containerd                            x86_64            1.7.20-1.amzn2023.0.1  amazonlinux         35 M
iptables-libs                         x86_64            1.8.8-3.amzn2023.0.2  amazonlinux         401 k
iptables-nft                          x86_64            1.8.8-3.amzn2023.0.2  amazonlinux         183 k
libcgroupp                            x86_64            3.0-1.amzn2023.0.1    amazonlinux         75 k
libnetfilter_conntrack                x86_64            1.0.8-2.amzn2023.0.2  amazonlinux         58 k
libnftnl                             x86_64            1.0.1-19.amzn2023.0.2 amazonlinux         30 k
libnftnl                             x86_64            1.2.2-2.amzn2023.0.2  amazonlinux         84 k
pigz                                  x86_64            2.5-1.amzn2023.0.3    amazonlinux         83 k

```

5. After successfully docker has been installed it has to be started on all machines by using the command “systemctl start docker”



```

[root@ip-172-31-24-23 ec2-user]# systemctl start docker
[root@ip-172-31-24-23 ec2-user]#

```

6. Kubernetes installation:

Search kubeadm installation on your browser and scroll down and select red hat-based distributions.

Debian-based distributions
Red Hat-based distributions

Without a package manager

1. Set SELinux to `permissive` mode:
These instructions are for Kubernetes 1.31.

```

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

```

```
# This overwrites any existing configuration in /etc/yum.repos.d/kubernetes.repo
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.key
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni
EOF
```

3. Install kubelet, kubeadm and kubectl:

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

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

```
sudo systemctl enable --now kubelet
```

Copy the above given steps and paste in the terminal. This will create a Kubernetes repository, install kubelet, kubeadm and kubectl and also enable the services.

```
Verifying      : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Verifying      : cri-tools-1.31.1-150500.1.1.x86_64
Verifying      : kubeadm-1.31.1-150500.1.1.x86_64
Verifying      : kubectl-1.31.1-150500.1.1.x86_64
Verifying      : kubelet-1.31.1-150500.1.1.x86_64
Verifying      : kubernetes-cni-1.5.1-150500.1.1.x86_64

Installed:
  conntrack-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-172-31-18-228 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-172-31-18-228 ec2-user]#
```

7. We can check if repository has been created by using yum repolist command

```
[root@ip-172-31-24-23 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-172-31-24-23 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-preflight-errors=all

```
[root@ip-172-31-24-23 ec2-user]# kubeadm init
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
[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-172-31-24-23 ec2-user]# ^C
[root@ip-172-31-24-23 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-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'
W0908 09:11:46.376173 28683 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-172-31-24-23.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.24.23]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key

[root@ip-172-31-82-191 ec2-user]# kubeadm init --ignore-preflight-errors=NumCPU --ignore-preflight-errors=Mem
[init] Using Kubernetes version: v1.31.0
[preflight] Running pre-flight checks
W0913 15:40:09.666593 2255 checks.go:1080] [preflight] WARNING: Couldn't create the interface used for talking to the container runtime: failed to create new CRI runtime service: validate service connection: validate CRI v1 runtime API for endpoint "unix:///var/run/containerd/containerd.sock": rpc error: code = Unavailable desc = connection error: desc = "transport: Error while dialing: dial unix /var/run/containerd/containerd.sock: connect: no such file or directory"
[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 FileContent--proc-sys-net-ipv4-ip-forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
[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-172-31-82-191 ec2-user]# ^C
[root@ip-172-31-82-191 ec2-user]# systemctl start docker
[root@ip-172-31-82-191 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 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'
W0913 15:44:18.815161 2565 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-172-31-82-191.ec2.internal kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local]
```

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 172.31.82.191:6443 --token 8450pt.tdprcovwa6lrqyo1 \
--discovery-token-ca-cert-hash sha256:b11f191f3df19a2e9112a5c19b4461bffeadd8b5be8625ad8451019aecc043c
```

11. We can check the nodes that have joined the cluster using `kubectl get nodes`. Right now, there is only one node which is the master node.

```
[root@ip-172-31-85-89 ec2-user]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-85-89.ec2.internal	NotReady	control-plane	72s	v1.26.0

12. 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 apiservert 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.
```

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

```
[root@ip-172-31-34-212 ec2-user]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-34-212.ec2.internal	Ready	control-plane	18m	v1.31.1
ip-172-31-37-229.ec2.internal	Ready	<none>	13m	v1.31.1
ip-172-31-45-98.ec2.internal	Ready	<none>	14m	v1.31.1

```
[root@ip-172-31-34-212 ec2-user]#
```

Conclusion:

This experiment enabled the creation of a Kubernetes cluster and the successful joining of all 3 nodes using various commands. Errors during initialization can be handled in two ways: 1. By ignoring the errors, or 2. By changing the instance type to `t3.medium` or `t3.large` if the issue is related to insufficient memory space or CPU resources. Also, it is to be ensured that the inbound rules and outbound rules allow all traffic or else it leads to connectivity issues between master node and worker nodes.