

Name: Akanksha Shinde Class: D15C Roll No: 53

Experiment 4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Steps:

1. Create a key pair and from Private key format select .pem. A .pem file will be downloaded in your machine.

EC2 > Key pairs > Create key pair

Create key pair [Info](#)

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type [Info](#)
☒ RSA ☐ ED25519

Private key file format
☒ .pem
For use with OpenSSH
☐ .ppk
For use with PuTTY

Tags - optional
No tags associated with the resource.

You can add up to 50 more tags.

Now, Create AWS EC2 instance

aws

Services

Search

[Alt+S]

EC2 > Instances > Launch an instance

Launch an instance

Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Info

Name

kubernetes

Add additional tags

▼ Application and OS Images (Amazon Machine Image)

Info

An AMI is a template that contains the software configuration (operating system, application server, and

Instances (5)

Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	aws-cloud9-fir...	i-08f0cfffbb8e65871	Shutting-d...	t2.micro	–	View alarms +	us-east-1d	ec2-54-211-201-203.co...	54.211.201.203	–
<input type="checkbox"/>	master	i-04d752dc5a6094386	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1d	ec2-54-165-168-237.co...	54.165.168.237	–
<input type="checkbox"/>	kubernetes	i-0a3994ea67ef2fda5	Running	t2.micro	Initializing	View alarms +	us-east-1d	ec2-34-239-128-175.co...	34.239.128.175	–
<input type="checkbox"/>	worker-2	i-06d89cce0e15b9521	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1d	ec2-44-201-188-182.co...	44.201.188.182	–
<input type="checkbox"/>	worker-1	i-00e2c3d55f62382f0	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1d	ec2-3-83-52-234.comp...	3.83.52.234	–

2. Edit the Security Group Inbound Rules to allow SSH

Edit inbound rules

Info

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

Inbound rules

Info

Security group rule ID

sgr-0c12076fc671c17ed

Type

SSH

Protocol

TCP

Port range

22

Source

Custom

0.0.0.0/0

Description - optional

Add rule

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel

Preview changes

Save rules

3.click on the id of the ec2 instance created and click connect you will get to see this page

Connect to instance Info


Connect to your instance i-0d940ce851c06d1c6 (Exp4) using any of these options

EC2 Instance Connect

Session Manager


SSH client

EC2 serial console

**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](#).

Instance ID

 i-0d940ce851c06d1c6 (Exp4)

Connection Type


☒ **Connect using EC2 Instance Connect**

Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.

☐ **Connect using EC2 Instance Connect Endpoint**


Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.


Public IPv4 address


 44.210.19.66

Username

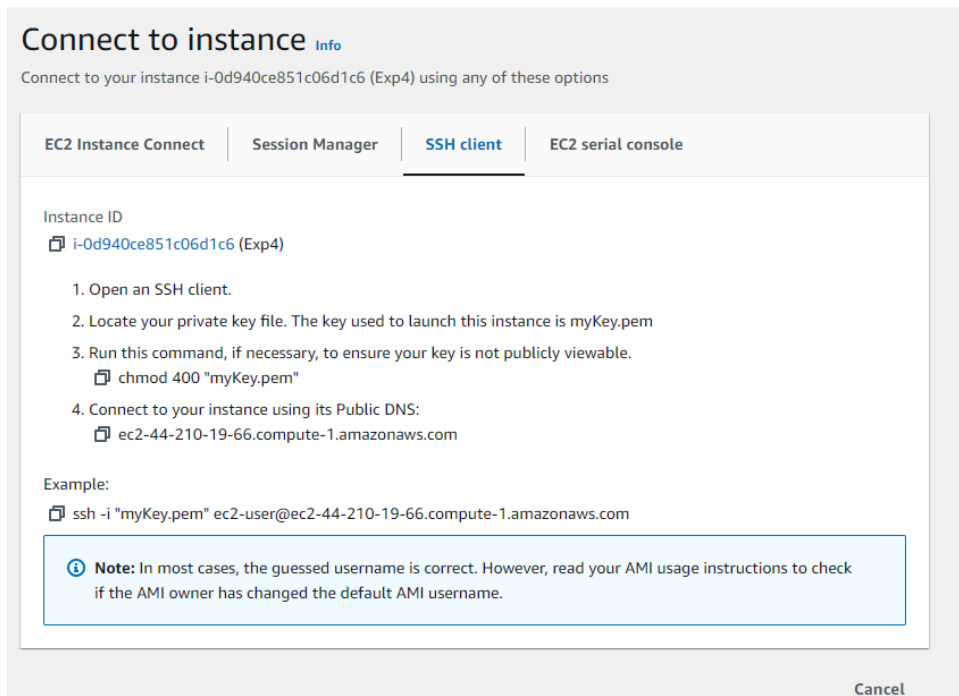
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ec2-user.

 ec2-user



 **Note:** In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Now click on the SSH client and copy the third command and the command below the example in notepad.



Go to git bash terminal and go to downloads where your .pem file has been downloaded and paste the copied commands.

```
Akanksha Shinde@AkankshaShinde MINGW64 ~/downloads
$ chmod 400 "myKey.pem"

Akanksha Shinde@AkankshaShinde MINGW64 ~/downloads
$ ssh -i "myKey.pem" ec2-user@ec2-44-210-19-66.compute-1.amazonaws.com
The authenticity of host 'ec2-44-210-19-66.compute-1.amazonaws.com (44.210.19.66)' can't be established.
ED25519 key fingerprint is SHA256:i+yKgHfjRavbh4Is9J7Spd+rR/laVV+otA8HG1EvaTM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added 'ec2-44-210-19-66.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
##### Amazon Linux 2023
#####
\###|
\#/
V~' -> https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-172-31-75-104 ~]$
[ec2-user@ip-172-31-75-104 ~]$
```

4. Install Docker

To install the docker run this command “sudo yum install docker -y”

```
[ec2-user@ip-172-31-75-104 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:12:45 ago on Sat Sep 14 11:27:03 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-libse                         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
runc                                   x86_64            1.1.13-1.amzn2023.0.1  amazonlinux      3.2 M
=====
Transaction Summary
=====
Install 10 Packages
```

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

First perform this : `cd /etc/docker`

Then run this : `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
```

After EOF automatically the code will get completed as you hit enter after typing till EOF.

```
complete:
[ec2-user@ip-172-31-75-104 ~]$ cd /etc/docker
[ec2-user@ip-172-31-75-104 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-172-31-75-104 docker]$ |
```

After this now, s run the following command to enable and start docker and also to load the daemon.json file.

```
sudo systemctl enable docker
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

```
[ec2-user@ip-172-31-75-104 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-172-31-75-104 docker]$ sudo systemctl daemon-reload
[ec2-user@ip-172-31-75-104 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-75-104 docker]$
```

Now check the version of the docker which will tell that whether you performed all tasks correctly or not. If yes then it will show the version else the otherwise.

```
[ec2-user@ip-172-31-75-104 docker]$ sudo systemctl restart docker
[ec2-user@ip-172-31-75-104 docker]$ docker -v
Docker version 25.0.5, build 5dc9bcc
[ec2-user@ip-172-31-75-104 docker]$
```

5. Install Kubernetes

To install kubernetes disable the SELinux for doing so run the command

1. `sudo setenforce 0`
2. `sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config`

```
[ec2-user@ip-172-31-75-104 ~]$ sudo setenforce 0
[ec2-user@ip-172-31-75-104 ~]$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
[ec2-user@ip-172-31-75-104 ~]$
```

Add kubernetes using the below commands -

```
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
```

```
[ec2-user@ip-172-31-75-104 ~]$ 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
[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
[ec2-user@ip-172-31-75-104 ~]$ |
```

Now, To install kubelet ,kubeadm, kubectl run the following command

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

```
[ec2-user@ip-172-31-75-104 ~]$ sudo yum update
Kubernetes
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-75-104 ~]$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Last metadata expiration check: 0:00:21 ago on Sat Sep 14 12:05:54 2024.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
kubeadm	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubectl	x86_64	1.30.5-150500.1.1	kubernetes	10 M
kubelet	x86_64	1.30.5-150500.1.1	kubernetes	17 M
Installing dependencies:				
conntrack-tools	x86_64	1.4.6-2.amzn2023.0.2	amazonlinux	208 k
cri-tools	x86_64	1.30.1-150500.1.1	kubernetes	8.6 M
kubernetes-cni	x86_64	1.4.0-150500.1.1	kubernetes	6.7 M
libnetfilter_cthelper	x86_64	1.0.0-21.amzn2023.0.2	amazonlinux	24 k
libnetfilter_cttimeout	x86_64	1.0.0-19.amzn2023.0.2	amazonlinux	24 k
libnetfilter_queue	x86_64	1.0.5-2.amzn2023.0.2	amazonlinux	30 k
Transaction Summary				
Install 9 Packages				

```
Installed:
  conntrack-tools-1.4.6-2.amzn2023.0.2.x86_64    cri-tools-1.30.1-150500.1.1.x86_64    kubeadm-1.30.5-150500.1.1.x86_64    kubectl-1.30.5-150500.1.1.x86_64
  kubelet-1.30.5-150500.1.1.x86_64    kubernetes-cni-1.4.0-150500.1.1.x86_64    libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64    libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
  libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-75-104 ~]$
```

Do some configurations to allow bridging.

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

```
[ec2-user@ip-172-31-75-104 ~]$ sudo swapoff -a
[ec2-user@ip-172-31-75-104 ~]$ echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[ec2-user@ip-172-31-75-104 ~]$ sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[ec2-user@ip-172-31-75-104 ~]$
```

6. Initialize the Kubecluster

Run the following command

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

If you get any warning regarding the CPU or ram space just then only run the following command -

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
```

```
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 172.31.75.104:6443 --token 5l2kvw.26ysk1yk3vx3cdm9 \
--discovery-token-ca-cert-hash sha256:7ad09cacafa9be4798dff17aef5271313eb82ca7ea0b85a8c47
d0f671768fd56
[ec2-user@ip-172-31-75-104 ~]$ |
```

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

```
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
```

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

```
kubectl apply -f
```

```
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.y
ml
```



```
[ec2-user@ip-172-31-75-104 ~]$ 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-cfg created
daemonset.apps/kube-flannel-ds created
[ec2-user@ip-172-31-75-104 ~]$
```

7. Now that the cluster is up and running, we can deploy our nginx server on this cluster.

Apply this deployment file using this command to create a deployment

`kubectl apply -f https://k8s.io/examples/application/deployment.yaml`

```
[ec2-user@ip-172-31-75-104 ~]$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[ec2-user@ip-172-31-75-104 ~]$
```

Run ‘kubectl get pods’ to verify if the deployment was properly created and the pod is working correctly.

```
[ec2-user@ip-172-31-75-104 ~]$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-d556bf558-8idl1f  0/1     Pending   0           18s
```

8. Lastly, port forward the deployment to your localhost so that you can view it.

`kubectl port-forward $POD_NAME 8080:80`

```
[ec2-user@ip-172-31-75-104 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

9. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

`curl --head http://127.0.0.1:8080`

If the response is 200 OK and you can see the Nginx server name, your deployment was successful. We have successfully deployed our Nginx server on our EC2 instance.

Conclusion:

An EC2 instance was launched and I enabled SSH access by modifying the inbound rules. Docker and Kubernetes were then installed, and internet bridging was configured. After setting up the cluster, we integrated the Flannel networking plugin. With the cluster running, we deployed an Nginx server and confirmed its successful deployment. I verified the Nginx deployment using Kubernetes commands and ensured it was accessible through the configured port. The entire setup demonstrated successful

integration of Docker, Kubernetes, and networking components within the EC2 environment.