

Experiment 4

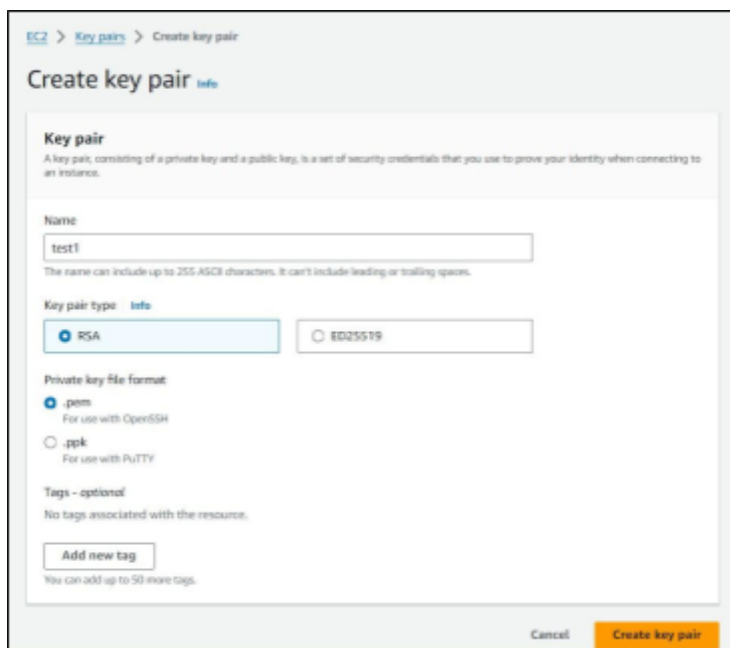
Aim:

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

Create a key pair



Name	Type	Created	Fingerprint	ID
test	rsa	2024/08/28 09:58 GMT+5:30	36:80:ad:4d:22:3d:50:d3:8f:77:bb:47:ea:3b:ac:e2:7...	key-07554c8c589...



Create key pair

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
test1
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
☒ RSA ☐ ED25519

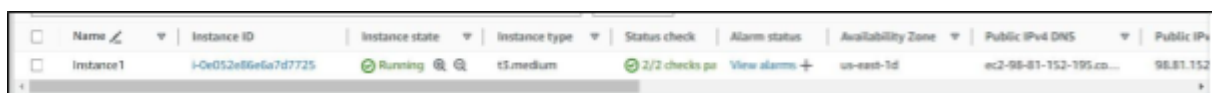
Private key file format
☒ .pem
For use with OpenSSH
☐ .ppk
For use with PuTTY

Tags - optional
No tags associated with the resource.
[Add new tag](#)
You can add up to 50 more tags.

[Cancel](#) [Create key pair](#)

The .pem file will be downloaded on your machine and will be required in the further steps.

1. Now we will create an EC2 Ubuntu instance. Select the key pair which you just created while creating this instance.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPs
Instance1	i-Oe052e86e6a7d7725	Running	t3.medium	2/2 checks pass	View alarms	us-east-1d	ec2-98-81-152-195.co...	98.81.152

Edit inbound rules info

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

Inbound rules info

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	
sg-0F0da53bad7b0975	All traffic	All	All	Custom		Delete
-	SSH	TCP	22	Anywhere...		Delete

uB X

Add rule

Cancel Preview changes Save rules

- Open git bash and go to the directory where pem file is located and use chmod to provide permissions.

```

Dell@DESKTOP-0VNTA1M MINGW64 ~/Downloads (master)
$ chmod 400 test1.pem

```

- Now use this command on the terminal: `ssh -i <keyname>.pem ubuntu@<public_ip_address>` and replace

```

Dell@DESKTOP-0VNTA1M MINGW64 ~/Downloads (master)
$ ssh -i "test1.pem" ec2-user@ec2-44-204-14-37.compute-1.amazonaws.com
The authenticity of host 'ec2-44-204-14-37.compute-1.amazonaws.com (44.204.14.37)' can't be established.
ED25519 key fingerprint is SHA256:CtXhAZnv4MFbUai03z96MQzMKK6JxuN/nwIierDSazI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-204-14-37.compute-1.amazonaws.com' (ED25519)
to the list of known hosts.

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

[ec2-user@ip-172-31-81-24 ~]$

```

4. Docker installation:

We will be installing docker by using “sudo yum install docker -y”

```
[root@ip-172-31-81-24 ec2-user]# sudo yum install docker -y
Last metadata expiration check: 0:01:25 ago on Sat Sep 14 06:42:34 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-libe                        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
libcgroup                            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
libnetfilter_link                    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
Total download size: 84 M
Installed size: 317 M
Downloading Packages:
(1/10): iptables-libe-1.8.8-3.amzn2023.0.2.x86_64.rpm                4.3 MB/s | 401 kB  00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm                3.3 MB/s | 183 kB  00:00
(3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64.rpm                    1.6 MB/s | 75 kB   00:00
(4/10): libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm      1.6 MB/s | 58 kB   00:00
(5/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm                  940 kB/s | 30 kB   00:00
(6/10): libnetfilter_link-1.0.1-19.amzn2023.0.2.x86_64.rpm          2.4 MB/s | 84 kB   00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm                          2.1 MB/s | 83 kB   00:00
(8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rpm                      19 MB/s | 3.2 MB   00:00
(9/10): containerd-1.7.20-1.amzn2023.0.1.x86_64.rpm                27 MB/s | 35 MB    00:01
(10/10): docker-25.0.6-1.amzn2023.0.2.x86_64.rpm                  27 MB/s | 44 MB    00:01
Total                                                                49 MB/s | 84 MB    00:01
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing                : runc-1.1.13-1.amzn2023.0.1.x86_64      1/1
  Installing               : containerd-1.7.20-1.amzn2023.0.1.x86_64 1/10
  Installing               : containerd-1.7.20-1.amzn2023.0.1.x86_64 2/10
  Running scriptlet: containerd-1.7.20-1.amzn2023.0.1.x86_64      7/10
```

5. Then to configure cgroup in a daemon json file we will run

```
cd /etc/docker
```

```
cat <<EOF | sudo tee /etc/docker/daemon.json
```

```
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
```

```
EOF
```

```
sudo systemctl enable docker
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

```
[root@ip-172-31-81-24 ec2-user]# cd /etc/docker
cat <<EOF | sudo tee /etc/docker/daemon.json
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
EOF
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker
{
"exec-opts": ["native.cgroupdriver=systemd"]
}
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
```

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.

```
[root@ip-172-31-81-24 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service -> /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-81-24 docker]#
```

```
[root@ip-172-31-81-24 docker]# sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
Dependencies resolved.
Package Architecture Version Repository Size
Installing:
kubeadm x86_64 1.31.1-150500.1.1 kubernetes 11 M
kubectl x86_64 1.31.1-150500.1.1 kubernetes 11 M
kubernet x86_64 1.31.1-150500.1.1 kubernetes 15 M
Installing dependencies:
cri-tools x86_64 1.4.6-2.amzn2023.0.2 amazonlinux 208 k
kubernetes-cni x86_64 1.5.1-150500.1.1 kubernetes 7.1 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
Total download size: 51 M
Installed size: 269 M
Downloading Packages:
(1/9): libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64.rpm 490 kB/s | 24 kB 00:00
(2/9): libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64.rpm 450 kB/s | 24 kB 00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64.rpm 1.2 MB/s | 30 kB 00:00
(4/9): cri-tools-1.4.6-2.amzn2023.0.2.x86_64.rpm 2.2 MB/s | 208 kB 00:00
(5/9): kubernetes-cni-1.5.1-150500.1.1.x86_64.rpm 25 MB/s | 6.5 MB 00:00
(6/9): kubeadm-1.31.1-150500.1.1.x86_64.rpm 27 MB/s | 11 MB 00:00
(7/9): kubectl-1.31.1-150500.1.1.x86_64.rpm 23 MB/s | 11 MB 00:00
(8/9): kubelet-1.31.1-150500.1.1.x86_64.rpm 35 MB/s | 15 MB 00:00
(9/9): kubernetes-cni-1.5.1-150500.1.1.x86_64.rpm 25 MB/s | 7.1 MB 00:00
Total
Kubernetes
Importing GPG key 0x0A296436:
Userid : "io:kubernetes:085 Project <io:kubernetes@build.opensuse.org>"
Fingerprint: DE15 B144 86CD 3778 9E87 6E1A 2346 54DA 9A29 6436
From : https://pkgs.k8s.io/core/stable/v1.31/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : kubernetes-cni-1.5.1-150500.1.1.x86_64
Installing : cri-tools-1.31.1-150500.1.1.x86_64
Installing : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Installing : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
Installing : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
Running scriptlet: contrack-tools-1.4.6-2.amzn2023.0.2.x86_64
Installing : kubelet-1.31.1-150500.1.1.x86_64
Running scriptlet: kubeadm-1.31.1-150500.1.1.x86_64
Installing : kubectl-1.31.1-150500.1.1.x86_64
Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64
Verifying : contrack-tools-1.4.6-2.amzn2023.0.2.x86_64
Verifying : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
Verifying : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
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:
contrack-tools-1.4.6-2.amzn2023.0.2.x86_64 cri-tools-1.31.1-150500.1.1.x86_64 kubeadm-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_cthelper-1.0.0-21.amzn2023.0.2.x86_64
libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Complete!
[root@ip-172-31-81-24 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service -> /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-81-24 docker]#
```

```
(6/9): kubeadm-1.31.1-150500.1.1.x86_64.rpm
(7/9): kubectl-1.31.1-150500.1.1.x86_64.rpm
(8/9): kubelet-1.31.1-150500.1.1.x86_64.rpm
(9/9): kubernetes-cni-1.5.1-150500.1.1.x86_64.rpm
Total
Kubernetes
Importing GPG key 0x0A296436:
Userid : "io:kubernetes:085 Project <io:kubernetes@build.opensuse.org>"
Fingerprint: DE15 B144 86CD 3778 9E87 6E1A 2346 54DA 9A29 6436
From : https://pkgs.k8s.io/core/stable/v1.31/rpm/repodata/repomd.xml.key
Key imported successfully
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing : kubernetes-cni-1.5.1-150500.1.1.x86_64
Installing : cri-tools-1.31.1-150500.1.1.x86_64
Installing : libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Installing : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
Installing : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
Running scriptlet: contrack-tools-1.4.6-2.amzn2023.0.2.x86_64
Installing : kubelet-1.31.1-150500.1.1.x86_64
Running scriptlet: kubeadm-1.31.1-150500.1.1.x86_64
Installing : kubectl-1.31.1-150500.1.1.x86_64
Running scriptlet: kubelet-1.31.1-150500.1.1.x86_64
Verifying : contrack-tools-1.4.6-2.amzn2023.0.2.x86_64
Verifying : libnetfilter_cthelper-1.0.0-21.amzn2023.0.2.x86_64
Verifying : libnetfilter_cttimeout-1.0.0-19.amzn2023.0.2.x86_64
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:
contrack-tools-1.4.6-2.amzn2023.0.2.x86_64 cri-tools-1.31.1-150500.1.1.x86_64 kubeadm-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_cthelper-1.0.0-21.amzn2023.0.2.x86_64
libnetfilter_queue-1.0.5-2.amzn2023.0.2.x86_64
Complete!
[root@ip-172-31-81-24 docker]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service -> /usr/lib/systemd/system/kubelet.service.
[root@ip-172-31-81-24 docker]#
```

- 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

```
[root@ip-172-31-81-24 docker]# sudo swapoff -a
[root@ip-172-31-81-24 docker]# echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf
net.bridge.bridge-nf-call-iptables=1
[root@ip-172-31-81-24 docker]# sudo sysctl -p
net.bridge.bridge-nf-call-iptables = 1
[root@ip-172-31-81-24 docker]#
```

8. Initializing kubecoluster:

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

```
[root@ip-172-31-81-24 docker]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[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-172-31-81-24 docker]# sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=NumCPU,Mem
[init] Using Kubernetes version: v1.31.0
```

```
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/custer-administration/addons/

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

kubeadm join 172.31.81.24:6443 --token 4a91z3.yz6rwmkf9yncyd2 \
--discovery-token-ca-cert-hash sha256:3404bd1bcdd9cf90a003673f622d1672acb4c6ce7c15c4738c80a0a1560fe70d
[root@ip-172-31-81-24 docker]#
```

9. The mkdir command that is generated after initialization has to be copy pasted in the terminal.

```
[root@ip-172-31-81-24 docker]# mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
[root@ip-172-31-81-24 docker]#
```

10. Then, add a common networking plugin called flannel:

kubectl apply -f
<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

```
[root@ip-172-31-81-24 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-cfg created
daemonset.apps/kube-flannel-ds created
[root@ip-172-31-81-24 docker]#
```

11. Apply this deployment file using this command to create a deployment

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

```
[root@ip-172-31-81-24 docker]# kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
[root@ip-172-31-81-24 docker]#
```


11. Use `kubectl get pods` to check if pod is working correctly

```
[root@ip-172-31-81-24 docker]# kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-d556bf558-8jd1f	0/1	Pending	0	18s

12. To change status from pending to running use following command: `kubectl describe pod nginx`.

```
[root@ip-172-31-16-56 ~]# kubectl describe pod nginx
Name: nginx-deployment-d556bf558-gw8v8
Namespace: default
Priority: 0
Service Account: default
Node: <none>
Labels: app=nginx
        pod-template-hash=d556bf558
Annotations: <none>
Status: Pending
IP:
IPs: <none>
Controlled By: ReplicaSet/nginx-deployment-d556bf558
Containers:
  nginx:
    Image: nginx:1.14.2
    Port: 80/TCP
    Host Port: 0/TCP
    Environment: <none>
    Mounts:
```

```

Conditions:
  Type          Status
PodScheduled   False

Volumes:
  kube-api-access-f9k9s:
    Type:          Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:  kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:    true
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type          Reason          Age           From          Message
  ----          -
Warning        FailedScheduling  114s         default-scheduler  0/1 nodes are available: 1 node(s) had untoleration taint {node-role.kubernetes.io/control-plane:NoSchedule=Exists} (node-role.kubernetes.io/control-plane:NoSchedule=Exists)
Warning        FailedScheduling  3m18s        default-scheduler  0/1 nodes are available: 1 node(s) had untoleration taint {node-role.kubernetes.io/control-plane:NoSchedule=Exists} (node-role.kubernetes.io/control-plane:NoSchedule=Exists)

```

Use the below command to remove taints

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-  
node/ip-172-31-26-174 ec2:internal:untainted
```

13. Check the pod status

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx	1/1	Running	1 (6s ago)	90s

14. port forward the deployment to your localhost so that you can view it.

```
[ec2-user@ip-172-31-26-174 ~]$ kubectl port-forward nginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
```

15. 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`

Conclusion: :Firstly I created an EC2 AWS Linux instance successfully.then installed docker and kubernetes successfully. Initialized kubernetes and execute mkdir and chown command successfully. Then I tried to deploy nginx which initially gave an error. Then I deployed (simple-pod.yml) nginx successfully and also checked by using the get pods command.