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

Theory:

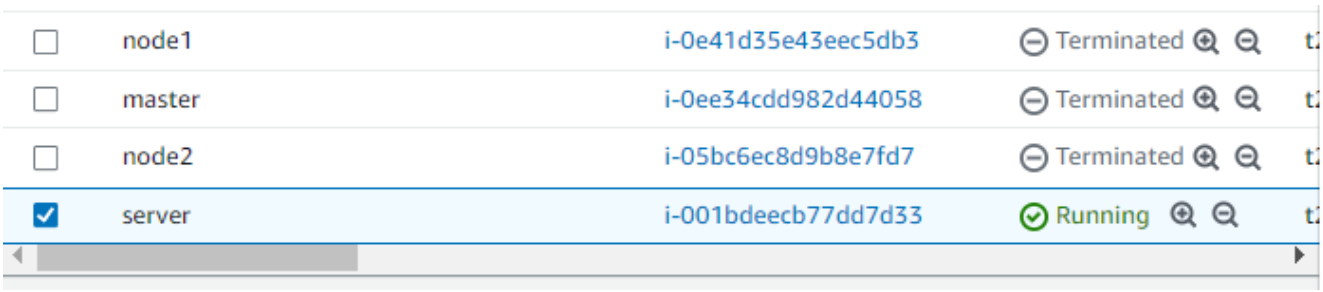
Originally developed by Google, Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. In fact, Kubernetes has established itself as the defacto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes Deployment

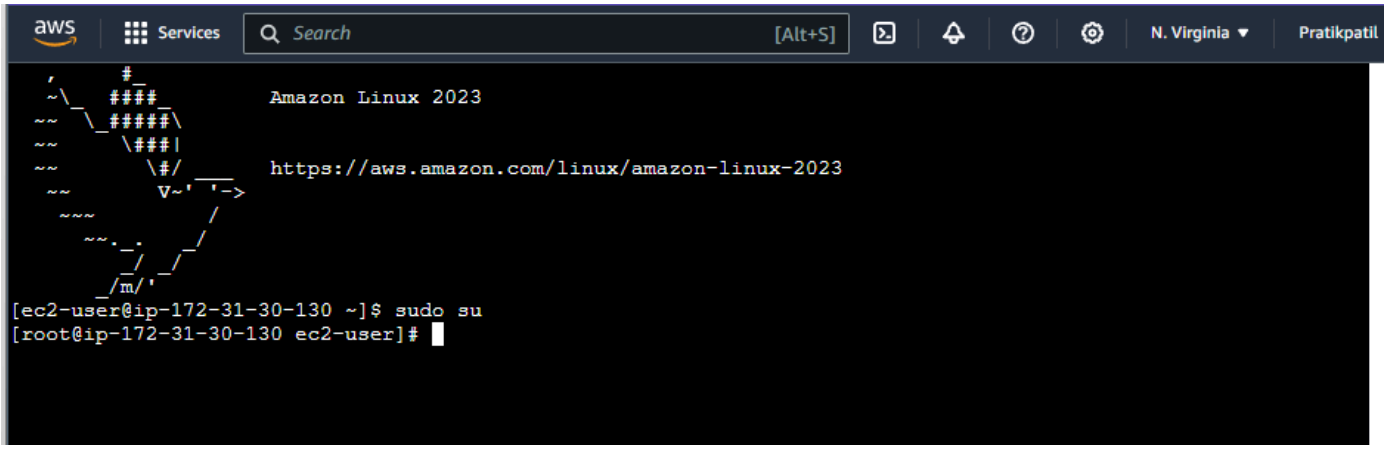
A Kubernetes Deployment is used to tell Kubernetes how to create or modify instances of the pods that hold a containerized application. Deployments can scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary.

Steps:

1. Create an EC2 Ubuntu Instance on AWS.



2. SSH into the machine



3. Install Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install -y docker-ce
```

```
ec2-user@ip-172-31-30-130 ~]$ sudo su
root@ip-172-31-30-130 ec2-user]# yum install docker -y
ast metadata expiration check: 0:04:22 ago on Sat Sep 14 13:00:06 2024.
ependencies resolved.
================================================================================
Package                 Arch    Version                    Repository    Size
================================================================================
nstalling:
docker                  x86_64  25.0.6-1.amzn2023.0.2     amazonlinux   44 M
nstalling 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
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
libnfnetlink            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

ransaction Summary
================================================================================
nstall  10 Packages

otal download size: 84 M
nstalled size: 317 M
ownloading Packages:
1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x 4.9 MB/s | 401 kB     00:00
2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x8 4.4 MB/s | 183 kB     00:00
3/10): libcgroup-3.0-1.amzn2023.0.1.x86_64. 2.9 MB/s |  75 kB     00:00
4/10): libnetfilter_conntrack-1.0.8-2.amzn2 1.5 MB/s |  58 kB     00:00
5/10): libnfnetlink-1.0.1-19.amzn2023.0.2.x 1.1 MB/s |  30 kB     00:00
6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64 2.2 MB/s |  84 kB     00:00
7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm   970 kB/s |  83 kB     00:00
8/10): runc-1.1.13-1.amzn2023.0.1.x86_64.rp  20 MB/s | 3.2 MB     00:00
9/10): containerd-1.7.20-1.amzn2023.0.1.x86  32 MB/s |  35 MB     00:01
10/10): docker-25.0.6-1.amzn2023.0.2.x86_64  28 MB/s |  44 MB     00:01
```

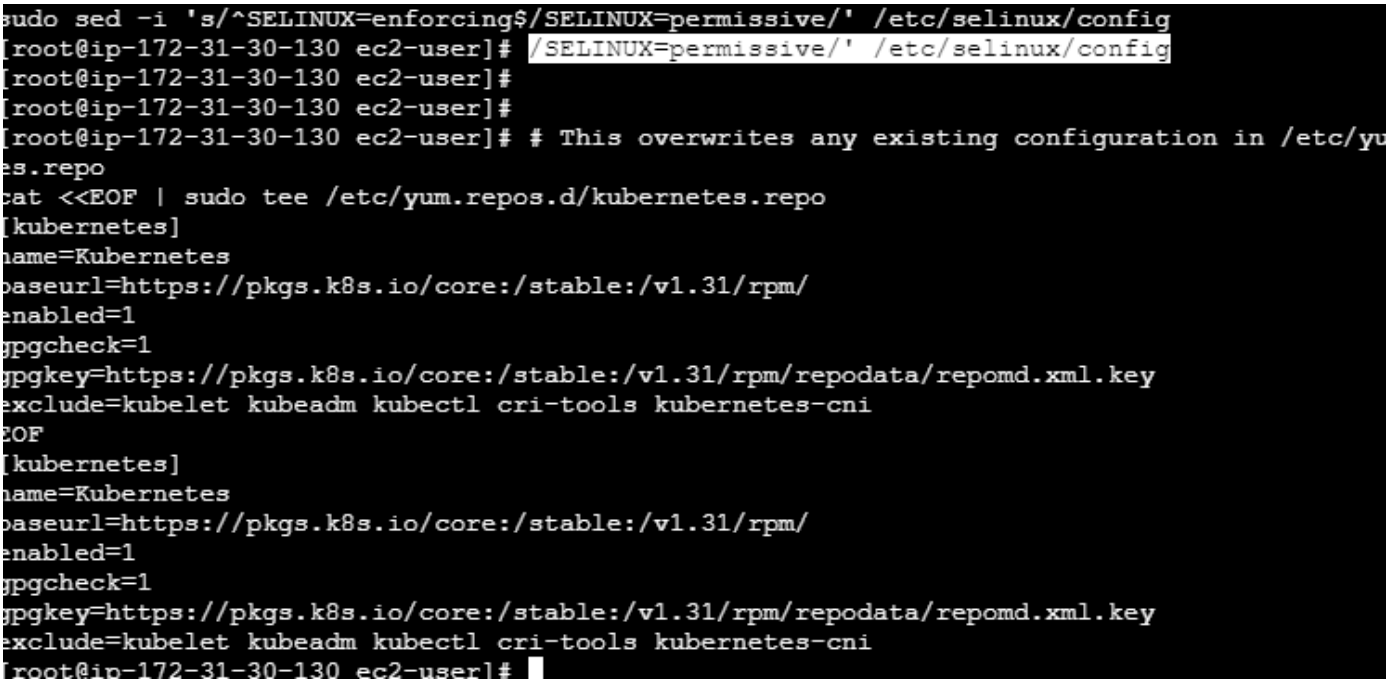
Then, configure cgroup in a daemon.json file.

```
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
```

4. Install Kubernetes

```
# 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
```




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

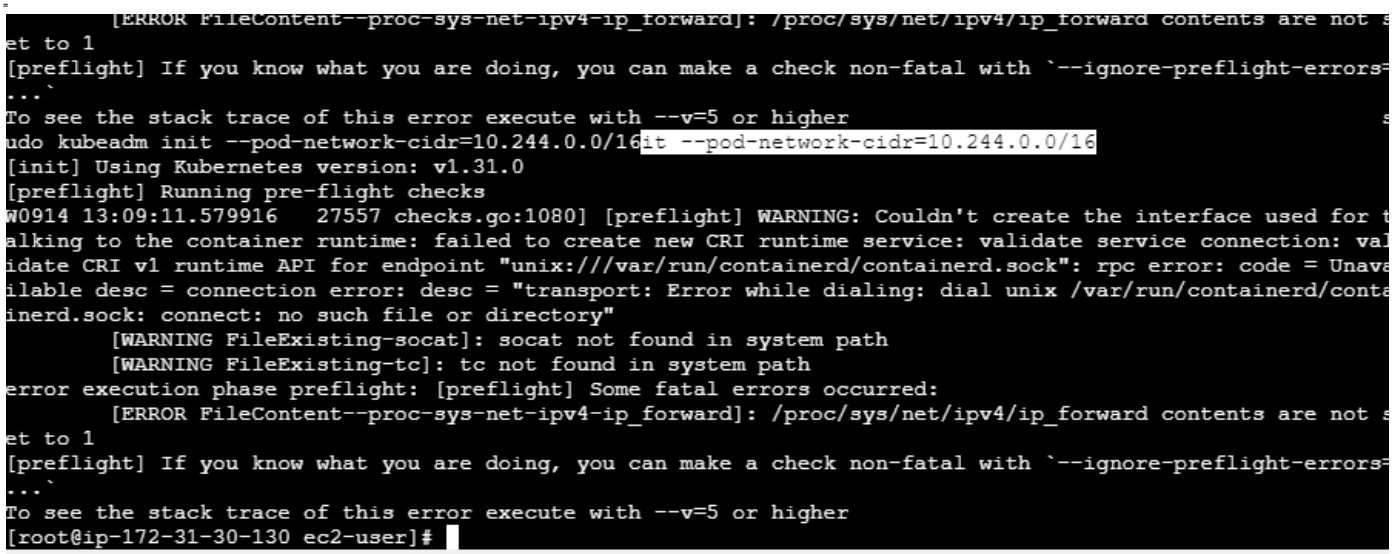
```
exclude=kubelet kubeadm kubectl cri-tools kubernetes-cni                                                    s
udo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetesubectl --disableexcludes=kubernetes
Kubernetes                                       56 kB/s | 9.4 kB     00:00
Dependencies resolved.
================================================================================
 Package                 Arch    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
 kubelet                 x86_64  1.31.1-150500.1.1         kubernetes    15 M
Installing dependencies:
 conntrack-tools         x86_64  1.4.6-2.amzn2023.0.2      amazonlinux  208 k
 cri-tools               x86_64  1.31.1-150500.1.1         kubernetes   6.9 M
 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_cthelper-1.0.0-21.amzn20 323 kB/s |  24 kB     00:00
(2/9): libnetfilter_cttimeout-1.0.0-19.amzn2 269 kB/s |  24 kB     00:00
(3/9): libnetfilter_queue-1.0.5-2.amzn2023.0 1.1 MB/s |  30 kB     00:00
(4/9): conntrack-tools-1.4.6-2.amzn2023.0.2. 1.5 MB/s | 208 kB     00:00
(5/9): kubeadm-1.31.1-150500.1.1.x86_64.rpm   24 MB/s |  11 MB     00:00
```

sudo systemctl enable --now kubelet

```
Complete!
[root@ip-172-31-30-130 ec2-user]# sudo systemctl enable --now kubelet
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /usr/lib/systemd/system/kubel
et.service.
[root@ip-172-31-30-130 ec2-user]#
```

5. Initialize the Kubecluster

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



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
```

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

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation
/ k ube-flannel.yml
```

```
ubuntu@ip-172-31-4-0:/etc/docker$ kubectl apply -f https://raw.githubusercontent.com/coreos/fla
nnel/master/Documentation/kube-flannel.yml
```

6. 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
   ```

   ```
   ubuntu@ip-172-31-4-0:/etc/docker$ kubectl apply -f https://k8s.io/examples/application/deployme
   nt.yaml
   deployment.apps/nginx-deployment created
   ```

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

   Next up, create a name alias for this pod.

   ```
   POD_NAME=$(kubectl get pods -l app=nginx -o
   jsonpath="{.items[0].metadata.name}")
   ```

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

   ```
   kubectl port-forward $POD_NAME 8080:80
   ```

8. 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
   ```

```
ubuntu@ip-172-31-4-0:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sat, 02 Oct 2021 16:07:48 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

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: open-source platform originally developed by Google, has become the industry standard for managing containerized applications. It automates the deployment, scaling, and orchestration of containers, simplifying the process of maintaining complex application environments. With support from major industry players, Kubernetes ensures reliable, scalable, and efficient application management.

By using Kubernetes Deployments, developers can efficiently manage application lifecycles, scale replica pods, roll out updates in a controlled manner, and easily revert to previous versions if needed, making it a powerful tool for modern cloud-native application management.