

Name : Nidhi Rijhwani

Subject : Adv Devops Exp No 04

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

Theory:

Kubernetes, originally developed by Google, is an open-source container orchestration platform. It automates the deployment, scaling, and management of containerized applications, ensuring high availability and fault tolerance. Kubernetes is now the industry standard for container orchestration and is governed by the Cloud Native Computing Foundation (CNCF), with contributions from major cloud and software providers like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat.

Kubernetes Deployment: Is a resource in Kubernetes that provides declarative updates for Pods and ReplicaSets. With a Deployment, you can define how many replicas of a pod should run, roll out new versions of an application, and roll back to previous versions if necessary. It ensures that the desired number of pod replicas are running at all times.

Necessary Requirements:

- **EC2 Instance:** The experiment required launching a t2.medium EC2 instance with 2 CPUs, as Kubernetes demands sufficient resources for effective functioning.
- **Minimum Requirements:**
 - Instance Type: t2.medium
 - CPUs: 2
 - Memory: Adequate for container orchestration

Step 1: Log in to your AWS Academy/personal account and launch a new Ec2 Instance. Select Ubuntu as AMI and t2.medium as Instance Type, create a key of type RSA with .pem extension, and move the downloaded key to the new folder.

Note: A minimum of 2 CPUs are required so Please select t2.medium and do not forget to stop the instance after the experiment because it is not available in the free tier.

Step 2: After creating the instance click on Connect the instance and navigate to SSH Client.

Step 3: Now open the folder in the terminal where our .pem key is stored and paste the Example : command (starting with ssh -i) in the terminal. (ssh -i "Master_Ec2_Key.pem" ubuntu@ec2-54-196-129-215.compute-1.amazonaws.com)

Step 4: Run the below commands to install and setup Docker.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
```

```
sudo tee /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
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
```

```
sudo mkdir -p /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
```

Step 5: Run the below command to install Kubernetes.

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key | sudo gpg --dearmor
-o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ ' |
sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

```
sudo systemctl enable --now kubelet
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

```
sudo systemctl restart containerd
sudo systemctl enable containerd
sudo systemctl status containerd
sudo apt-get install -y socat
```

Step 6: Initialize the Kubecluster

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

Step 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
```

Step 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.

The image shows two screenshots from the AWS Management Console. The top screenshot displays the 'Create new EC2 instance' wizard. The 'Summary' tab is selected, showing the configuration for an Ubuntu Server 24.04 LTS (HVM) instance. The instance type is 't2.medium', and the software image is 'Canonical, Ubuntu, 24.04, amd64_...'. The instance is set to be 'Free tier eligible'. A 'Free tier' notification box is visible, stating: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, ...'. The 'Launch instance' button is highlighted.

The bottom screenshot shows the 'Instances (1)' page. The instance 'Exp4' with ID 'i-04a49a928f6d2dbf1' is in the 'Running' state. The 'Connect to instance' page is open, showing the 'SSH client' tab. The instance ID 'i-04a49a928f6d2dbf1 (Exp4)' is displayed. The steps to connect via SSH are listed:

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is exp4key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 "exp4key.pem"`
4. Connect to your instance using its Public DNS:
`ec2-52-91-240-34.compute-1.amazonaws.com`

An example command is provided: `ssh -i "exp4key.pem" ubuntu@ec2-52-91-240-34.compute-1.amazonaws.com`. A note at the bottom states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

Microsoft Windows [Version 10.0.22000.2057]
(c) Microsoft Corporation. All rights reserved.

```
C:\Users\ACER\Downloads>ssh -i "exp4key.pem" ubuntu@ec2-52-91-240-34.compute-1.amazonaws.com
The authenticity of host 'ec2-52-91-240-34.compute-1.amazonaws.com (52.91.240.34)' can't be established.
ECDSA key fingerprint is SHA256:hQXGXhM3JrUApDQWobOui+rTzu/uzA7hY4Hs9p58oLM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-91-240-34.compute-1.amazonaws.com,52.91.240.34' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro
```

System information as of Sat Sep 21 10:54:46 UTC 2024

System load:	0.08	Processes:	115
Usage of /:	22.7% of 6.71GB	Users logged in:	0
Memory usage:	6%	IPv4 address for enX0:	172.31.87.78
Swap usage:	0%		

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: `sudo pro status`

```
9AcZ58Em+1WsVnAXdUR//bMmhyr8wL/G1Y01V3JE3TRdxs5xdYa4deGBBY/Adpsw
24jxhOJR+1sJpqIUeb999+R8euDhRHG9eF07DRu6weatUJ6suupoDTRWtr/4yGqe
dKxV3qQhNLsnaAzqW/1nA3iUB4k7kCaKZxhdhDbC1f9P37qaRW4678LCVO/col3y
Vm50dwdrNtKpMBh3ZpbB1uJvgi9mXtyBOMJ3v8RZeDzFiG8HdCtg9RvIt/AIFoHR
H3S+u79NT6i0KPzLIImDfs8T7R1pyuMc4Ufs8gggyg9v3Ae6cN3eQyxcK3w0cbBwsh
/nQnfsA6uu+9H7NhbBhBMhYnpNzYrHzCmzyXkauwRAqoCbGCNykTRwsur9gS41TQ
M8sD1jFheOJf3hODnkKU+HKjvMR011DK7zdmLdNzA1cvtZH/nCC9KPj1z8QC47S
xx+dTZSx4ONAhwbS/LN3PoKtn8LPjY9NP9uDWI+TwYquS2U+KHDrBD1sgozDbs/O
jCxcDzNmXpWQHETHU7649OXHP7UeNST1mCUCH5qdark0V1iejF6/CfTFU4MfcrG
YT90qFF93M3v018bxP+EIY2/9tiIPbrd
=0YYh
-----END PGP PUBLIC KEY BLOCK-----
ubuntu@ip-172-31-87-78:~$ /etc/apt/trusted.gpg.d/docker.gpg > /dev/null
-bash: /etc/apt/trusted.gpg.d/docker.gpg: No such file or directory
ubuntu@ip-172-31-87-78:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Repository: 'deb [arch=amd64] https://download.docker.com/linux/ubuntu noble stable'
Description:
Archive for codename: noble components: stable
More info: https://download.docker.com/linux/ubuntu
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Adding deb entry to /etc/apt/sources.list.d/archive_uri=https_download_docker_com_linux_ubuntu-noble.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri=https_download_docker_com_linux_ubuntu-noble.list
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [15.3 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
```

```
Unpacking libltdl7:amd64 (2.4.7-7build1) ...
Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../8-libslirp0_4.7.0-1ubuntu3_amd64.deb ...
Unpacking libslirp0:amd64 (4.7.0-1ubuntu3) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../9-slirp4netns_1.2.1-1build2_amd64.deb ...
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.17.1-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.22-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service
Setting up docker-compose-plugin (2.29.7-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.2) ...
Scanning processes...
Scanning linux images...
```

```

ubuntu@ip-172-31-87-78:~$ sudo mkdir -p /etc/docker
ubuntu@ip-172-31-87-78:~$ cat <<EOF | sudo tee /etc/docker/daemon.json
> {
>   "exec-opts":["native.cgroupdriver=systemd"]
> }
> EOF
{
"exec-opts":["native.cgroupdriver=systemd"]
}

```

```

ubuntu@ip-172-31-87-78:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
ubuntu@ip-172-31-87-78:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-87-78:~$ sudo systemctl restart docker

```

```

ubuntu@ip-172-31-87-78:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb InRelease [1186 B]
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:7 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb Packages [4865 B]
Fetched 6051 B in 1s (10.5 kB/s)
Reading package lists... Done
W: https://download.docker.com/linux/ubuntu/dists/noble/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg),
ubuntu@ip-172-31-87-78:~$ sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools kubernetes-cni
The following NEW packages will be installed:
  conntrack cri-tools kubeadm kubectl kubelet kubernetes-cni
0 upgraded, 6 newly installed, 0 to remove and 139 not upgraded.
Need to get 87.4 MB of archives.
After this operation, 314 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 conntrack amd64 1:1.4.8-1ubuntu1 [37.9 kB]
Get:2 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb cri-tools 1.31.1-1.1 [15.7 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubeadm 1.31.1-1.1 [11.4 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubectl 1.31.1-1.1 [11.2 MB]
Get:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubernetes-cni 1.5.1-1.1 [33.9 MB]
Get:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.31/deb kubelet 1.31.1-1.1 [15.2 MB]
Fetched 87.4 MB in 1s (87.2 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 68007 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.8-1ubuntu1_amd64.deb ...
Unpacking conntrack (1:1.4.8-1ubuntu1) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.31.1-1.1_amd64.deb ...
Unpacking cri-tools (1.31.1-1.1) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../2-kubeadm_1.31.1-1.1_amd64.deb ...
Unpacking kubeadm (1.31.1-1.1) ...
Selecting previously unselected package kubectl.

```

```
Preparing to unpack .../4-kubernetes-cni_1.5.1-1.1_amd64.deb ...
Unpacking kubernetes-cni (1.5.1-1.1) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../5-kubelet_1.31.1-1.1_amd64.deb ...
Unpacking kubelet (1.31.1-1.1) ...
Setting up conntrack (1:1.4.8-1ubuntu1) ...
Setting up kubect1 (1.31.1-1.1) ...
Setting up cri-tools (1.31.1-1.1) ...
Setting up kubernetes-cni (1.5.1-1.1) ...
Setting up kubeadm (1.31.1-1.1) ...
Setting up kubelet (1.31.1-1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-87-78:~$ sudo apt-mark hold kubelet kubeadm kubect1
kubelet set on hold.
kubeadm set on hold.
kubect1 set on hold.
```

```
The following packages will be REMOVED:
  containerd.io docker-ce
The following NEW packages will be installed:
  containerd runc
0 upgraded, 2 newly installed, 2 to remove and 139 not upgraded.
Need to get 47.2 MB of archives.
After this operation, 53.1 MB disk space will be freed.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.12-0ubuntu4.1 [38.6 MB]
Fetched 47.2 MB in 1s (88.2 MB/s)
(Reading database ... 68064 files and directories currently installed.)
Removing docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Removing containerd.io (1.7.22-1) ...
Selecting previously unselected package runc.
(Reading database ... 68044 files and directories currently installed.)
Preparing to unpack .../runc_1.1.12-0ubuntu3.1_amd64.deb ...
Unpacking runc (1.1.12-0ubuntu3.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../containerd_1.7.12-0ubuntu4.1_amd64.deb ...
Unpacking containerd (1.7.12-0ubuntu4.1) ...
Setting up runc (1.1.12-0ubuntu3.1) ...
Setting up containerd (1.7.12-0ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-87-78:~$ █
```

```
ubuntu@ip-172-31-87-78:~$ sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/config.toml
ubuntu@ip-172-31-87-78:~$ sudo mkdir -p /etc/containerd
ubuntu@ip-172-31-87-78:~$ sudo containerd config default | sudo tee /etc/containerd/config.toml
disabled_plugins = []
imports = []
oom_score = 0
plugin_dir = ""
required_plugins = []
root = "/var/lib/containerd"
state = "/run/containerd"
temp = ""
version = 2
```

```
[cgroup]
  path = ""
```

```
[debug]
  address = ""
  format = ""
  gid = 0
  level = ""
  uid = 0
```

```
[grpc]
  address = "/run/containerd/containerd.sock"
  gid = 0
  max_recv_message_size = 16777216
  max_send_message_size = 16777216
  tcp_address = ""
  tcp_tls_ca = ""
  tcp_tls_cert = ""
  tcp_tls_key = ""
  uid = 0
```

```
ubuntu@ip-172-31-87-78:~$ sudo systemctl restart containerd
ubuntu@ip-172-31-87-78:~$ sudo systemctl enable containerd
ubuntu@ip-172-31-87-78:~$ sudo systemctl status containerd
• containerd.service - containerd container runtime
   Loaded: loaded (/usr/lib/systemd/system/containerd.service; enabled; preset: enabled)
   Active: active (running) since Sat 2024-09-21 12:48:02 UTC; 2min 6s ago
     Docs: https://containerd.io
   Main PID: 7830 (containerd)
      Tasks: 7
     Memory: 13.2M (peak: 14.0M)
        CPU: 295ms
     CGroup: /system.slice/containerd.service
             └─7830 /usr/bin/containerd
```

```
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603681825Z" level=info msg="Start subscribing containerd event"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603713219Z" level=info msg=serving... address=/run/containerd/containerd.sock.ttrpc
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603728772Z" level=info msg="Start recovering state"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603745293Z" level=info msg=serving... address=/run/containerd/containerd.sock
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603770644Z" level=info msg="Start event monitor"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603779762Z" level=info msg="Start snapshots syncer"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603787308Z" level=info msg="Start cni network conf syncer for default"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603795250Z" level=info msg="Start streaming server"
Sep 21 12:48:02 ip-172-31-87-78 containerd[7830]: time="2024-09-21T12:48:02.603847770Z" level=info msg="containerd successfully booted in 0.027952s"
Sep 21 12:48:02 ip-172-31-87-78 systemd[1]: Started containerd.service - containerd container runtime.
```

```
ubuntu@ip-172-31-87-78:~$ sudo apt-get install -y socat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docker-buildx-plugin docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libsllp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  socat
0 upgraded, 1 newly installed, 0 to remove and 139 not upgraded.
Need to get 374 kB of archives.
After this operation, 1649 kB of additional disk space will be used.
Get:1 http://us-east-1-ec2.archive.ubuntu.com/ubuntu noble/main amd64 socat amd64 1.8.0.0-4build3 [374 kB]
Fetched 374 kB in 0s (13.2 MB/s)
```

```
ubuntu@ip-172-31-87-78:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
[init] Using Kubernetes version: v1.11.0
[preflight] Running pre-flight checks
[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'
W0021 12:58:06.857054 9434 checks.go:846] detected that the sandbox image 'registry.k8s.io/pause:3.0' of the container runtime is inconsistent with that used by kubeadm. It is recommended to use 'registry.k8s.io/pause:3.0' 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-172-31-87-78.kubernetes.kubernetes.default.kubernetes.default.svc.kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 172.31.87.78]
[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/peer' certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-87-78.localhost] and IPs [172.31.87.78 127.0.0.1 ::1]
[certs] Generating 'etcd/peer' certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-87-78.localhost] and IPs [172.31.87.78 127.0.0.1 ::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/kubeade-flags.env'
[kubelet-start] Writing kubelet configuration file to file '/var/lib/kubelet/config.yaml'

See kubeadm init --help for usage.
ubuntu@ip-172-31-87-78:~$ 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
ubuntu@ip-172-31-87-78:~$
```

```
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-87-78:~$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-87-78:~$
```

```
ubuntu@ip-172-31-87-78:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-d556bf558-2nwj8	0/1	Pending	0	6m54s
nginx-deployment-d556bf558-vbnn6	0/1	Pending	0	6m54s

```
ubuntu@ip-172-31-87-78:~$
```



```
ubuntu@ip-172-31-20-171:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 15 Sep 2024 07:59:03 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
```

Conclusion:

In this experiment, we successfully installed Kubernetes on an EC2 instance and deployed an Nginx server using Kubectl commands. During the process, we encountered two main errors: the Kubernetes pod was initially in a pending state, which was resolved by removing the control-plane taint using `kubectl taint nodes --all`, and we also faced an issue with the missing containerd runtime, which was fixed by installing and starting containerd. We used a t2.medium EC2 instance with 2 CPUs to meet the necessary resource requirements for the Kubernetes setup and deployment.