



School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: Pratik.M.Rebari **PRN:** 20220802183

Title of Practicle : 6. Deploying multi-container on Kubernetes

1. Launch an instance with installation of docker and Kubernetes

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images (selected), AMIs, AMI Catalog, and Elastic Block Store. The main area displays 'Instances (1/6) Info' with a table of instances. The table columns are Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. The instances listed are Docker-4 (terminated), K8s2 (running, t2.medium, 2/2 checks passed), and K8s (terminated, t2.xlarge). Below the table, the details for instance i-0168458e459df4e56 (K8s2) are shown, including its public and private IP addresses, instance state (Running), and DNS name (ec2-18-212-83-219.compute-1.amazonaws.com).

2. User data for installation

```
#!/bin/bash

sudo apt-get update -y

sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common -y

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) \
```



School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: Pratik.M.Rebari **PRN:** 20220802183

Title of Practicle : 6. Deploying multi-container on Kubernetes

stable"

```
sudo apt-get install docker-ce docker-ce-cli containerd.io -y
```

```
sudo usermod -a -G docker ubuntu
```

```
# Initialize Kubernetes cluster
```

```
sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --ignore-preflight-errors=NumCPU || (echo 'Failed to initialize cluster' && exit 1)
```

```
# Setup kubectl
```

```
mkdir -p $HOME/.kube ; sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config || echo 'Failed to copy admin.conf' ; sudo chown $(id -u):$(id -g) $HOME/.kube/config || echo 'Failed to change ownership of config'
```

```
# Install Calico CNI
```

```
kubectl apply -f  
https://raw.githubusercontent.com/projectcalico/calico/v3.28.3/manifests/calico.yaml
```

```
# Save join command with full permissions
```

```
sudo kubeadm token create --print-join-command
```



School of Computer Science, Engineering and Applications(SCSEA)
B.Tech FIY (CCSA)
Subject : Cloud Automation & Devops (P)

Name of the Student: Pratik.M.Rebari **PRN:** 20220802183

Title of Practicle : 6. Deploying multi-container on Kubernetes

3. Now create multi-container.yml file

```
ubuntu@ip-172-31-26-39:~$ cat multi-container.yml
apiVersion: v1
kind: Pod
metadata:
  name: two-containers
spec:
  restartPolicy: OnFailure
  initContainers:
    - name: populate-content
      image: debian:stable-slim
      command: ["/bin/sh", "-c"]
      args:
        - mkdir -p /pod-data && echo "Hello from the init container" > /pod-data/index.html
      volumeMounts:
        - name: shared-data
          mountPath: /pod-data
  containers:
    - name: nginx-container
      image: nginx:stable-alpine
      ports:
        - containerPort: 80
      volumeMounts:
        - name: shared-data
          mountPath: /usr/share/nginx/html:ro
  volumes:
    - name: shared-data
      emptyDir: {}

~
~"multi-container.yml" 27L, 651B
```

27,0-1 All

4. Then apply it

```
ubuntu@ip-172-31-26-39:~$ kubectl apply -f multi-container.yml
pod/two-containers created
ubuntu@ip-172-31-26-39:~$ kubectl get pods -w
```

5. Now check the pods

```
ubuntu@ip-172-31-26-39:~$ kubectl get pods -w
NAME           READY   STATUS    RESTARTS   AGE
two-containers  1/1     Running   0          23s

^Cubuntu@ip-172-31-26-39:~$ kubectl port-forward pod/two-containers 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
^[[A^Cubuntu@ip-172-31-26-39:~$ ^C
ubuntu@ip-172-31-26-39:~$ kubectl port-forward pod/two-containers 8080:80
```