Module 7: Kubernetes Assignment -1

Tasks To Be Performed:

1. Deploy a Kubernetes cluster for 3 nodes

2. Create a NGINX deployment of 3 replicas

SOLUTION:

To deploy a Kubernetes cluster with three nodes on local Ubuntu EC2 instances in AWS and create an NGINX deployment with three replicas, follow these steps:

Prerequisites

AWS CLI: Install and configure the AWS CLI.

kubectl: Install kubectl, the Kubernetes command-line tool.

kubeadm, kubelet, kubectl: Install these Kubernetes components on your Ubuntu instances.

Docker: Ensure Docker is installed on all your instances

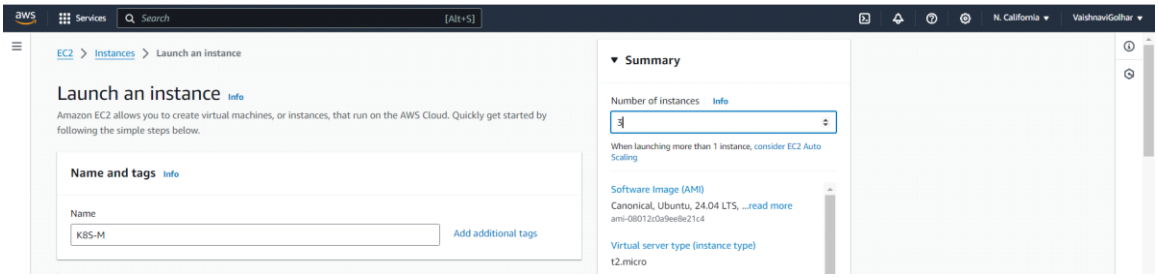
Task 1: Deploy a Kubernetes Cluster with 3 Nodes

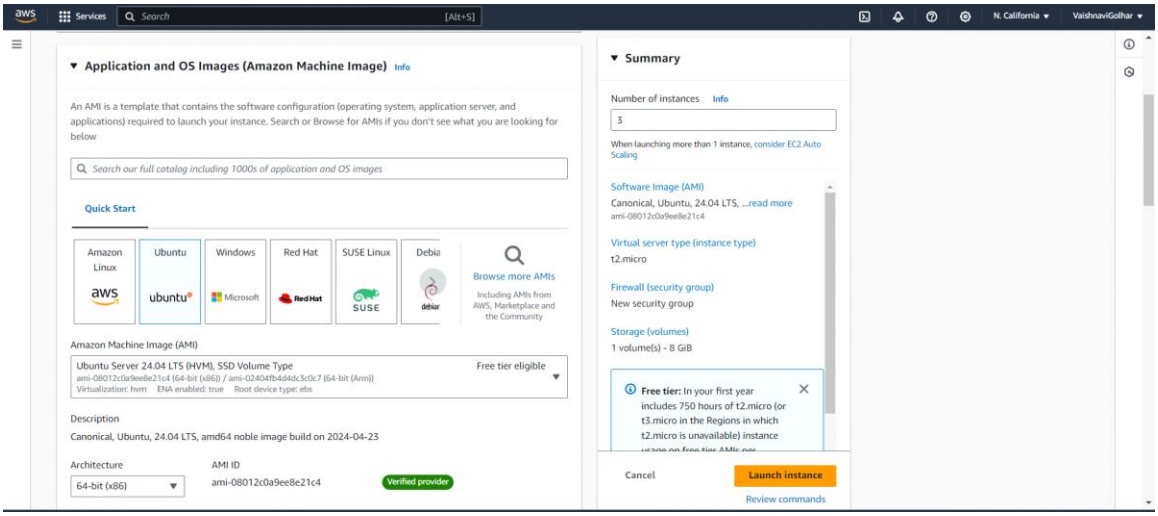
Step 1: Set Up AWS EC2 Instances

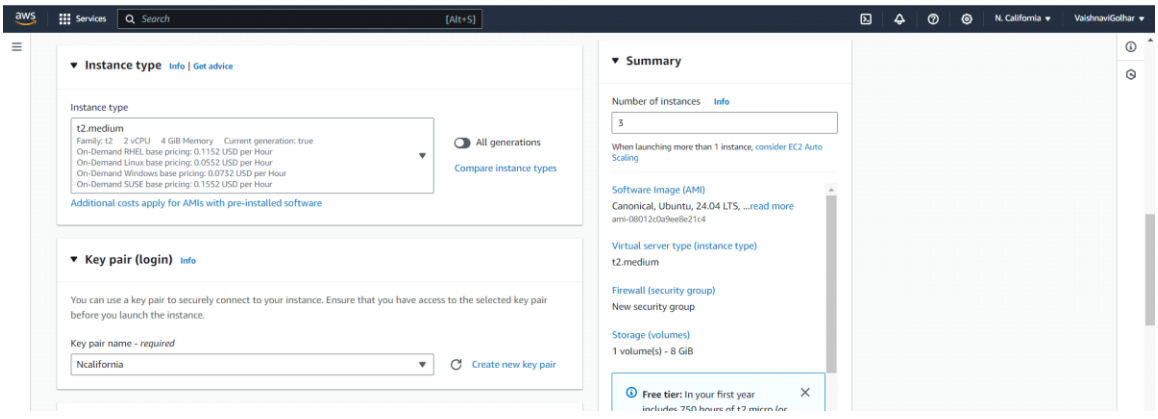
A} Launch 3 Ubuntu EC2 Instances:

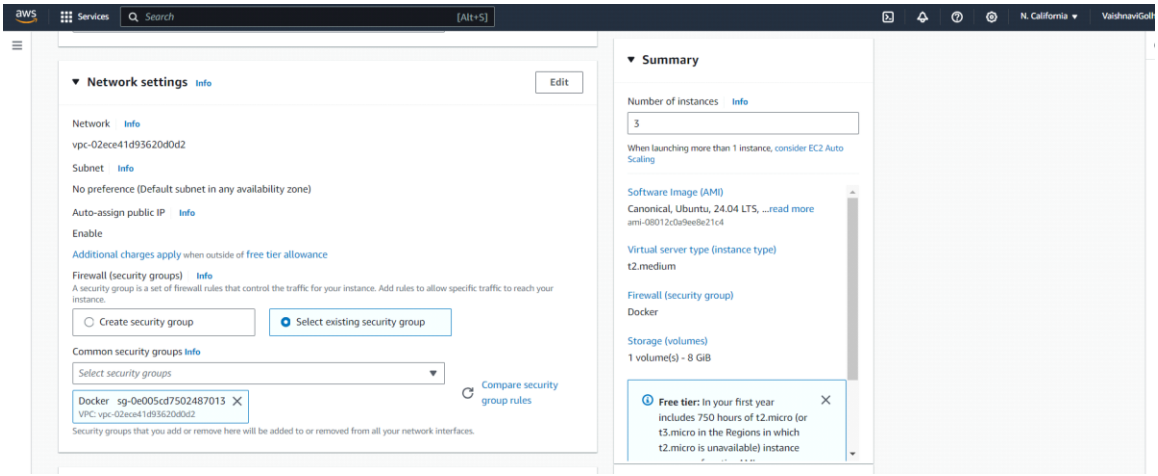
Ensure they are in the same VPC and subnet for network communication.

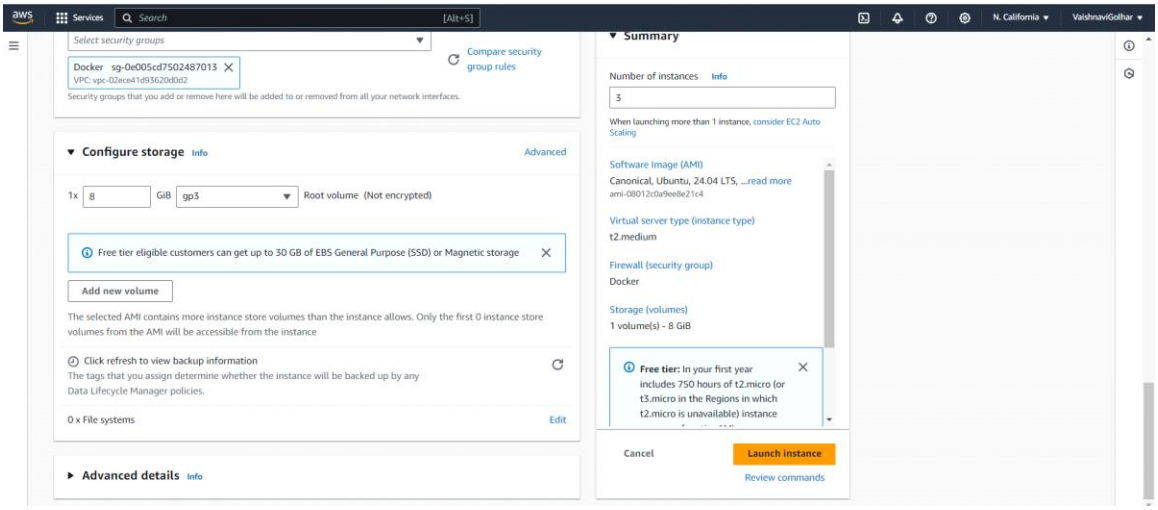
Open necessary ports in the security group (e.g., 6443 for Kubernetes API server, 10250 for kubelet API, 2379-2380 for etcd server client API, 179 for Calico networking).

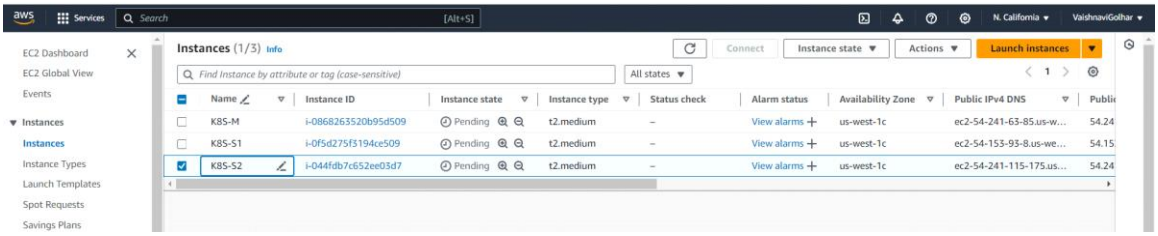

aws
Services
Search
[Alt+S]
N. California
VaishnaviGolhar
EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
K8S-M
Add additional tags
Summary
Number of instances Info
3
When launching more than 1 instance, consider EC2 Auto Scaling
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-08012c0a9ee8e21c4
Virtual server type (instance type)
t2.micro


aws
Services
Search
[Alt+S]
N. California
VaishnaviGolhar
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
Ubuntu
Windows
Red Hat
SUSE Linux
Debia
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
Free tier eligible
ami-08012c0a9ee8e21c4 (64-bit (x86)) / ami-02404fb4d4dc3c0c7 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs
Description
Canonical, Ubuntu, 24.04 LTS, amd64 noble image build on 2024-04-23
Architecture
64-bit (x86)
AMI ID
ami-08012c0a9ee8e21c4
Verified provider
Summary
Number of instances Info
3
When launching more than 1 instance, consider EC2 Auto Scaling
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-08012c0a9ee8e21c4
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
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
Cancel
Launch instance
Review commands


aws
Services
Search
[Alt+S]
N. California
VaishnaviGolhar
Instance type Info | Get advice
Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand RHEL base pricing: 0.1152 USD per Hour
On-Demand Linux base pricing: 0.0552 USD per Hour
On-Demand Windows base pricing: 0.0732 USD per Hour
On-Demand SUSE base pricing: 0.1552 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
Key pair (login) Info
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.
Key pair name - required
Ncalifornia
Create new key pair
Summary
Number of instances Info
3
When launching more than 1 instance, consider EC2 Auto Scaling
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-08012c0a9ee8e21c4
Virtual server type (instance type)
t2.medium
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year







Step 2: Install Docker and Kubernetes Components

A} Install Docker:

sudo apt-get update

sudo apt-get install -y docker.io

sudo systemctl enable docker

sudo systemctl start docker

B} Install kubeadm, kubelet, and kubectl:

Run the Script on All Nodes:

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

sudo mkdir -p /etc/apt/keyrings

sudo chmod -R 755 /etc/apt/keyrings

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo systemctl enable --now kubelet

C} Initialize the Master Node:

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

D} Join Worker Nodes to the Cluster:

kubeadm token create --print-join-command

sudo kubeadm join <master-node-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>

kubectl get nodes

Command given on both worker node

sudo kubeadm join 172.31.4.255:6443 --token rc1zsf.macmfqdbio6s2l33 \

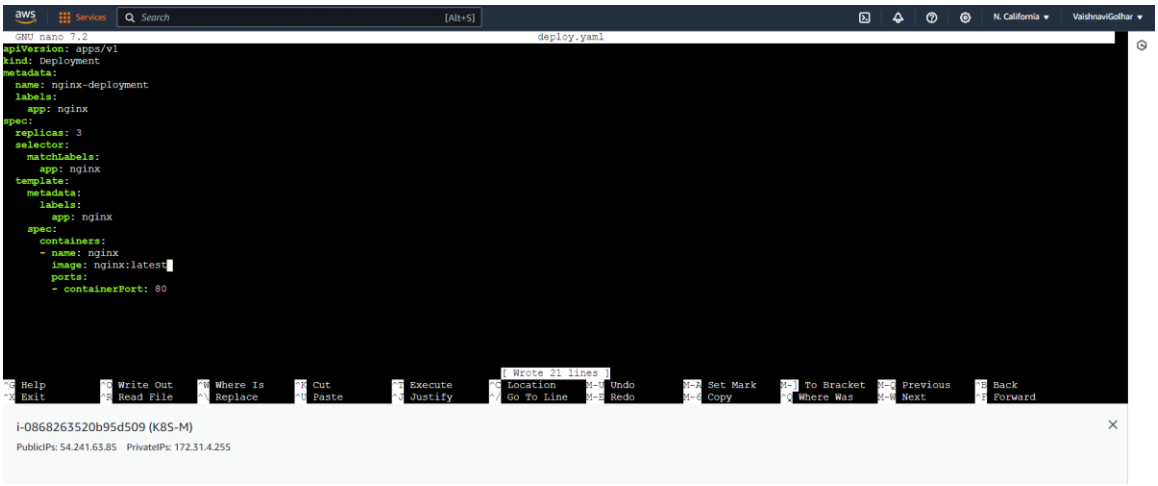
--discovery-token-ca-cert-hash sha256:65dcebc23f17ea902081b886fb835b57b7d018e277afcfb2c79582e141efdfea

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-14-55:~$
```

i-0f5d275f3194ce509 (K8S-S1)

PublicIPs: 54.153.93.8 PrivateIPs: 172.31.14.55

Task 2: Create a NGINX deployment of 3 replicas




```
ubuntu@ip-172-31-4-255:~$ sudo nano deploy.yaml
ubuntu@ip-172-31-4-255:~$ kubectl apply -f deploy.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-4-255:~$
```

i-0868263520b95d509 (K8S-M)

PublicIPs: 54.241.63.85 PrivateIPs: 172.31.4.255

```
ubuntu@ip-172-31-4-255:~$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           2m21s
ubuntu@ip-172-31-4-255:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7c79c4bf97-chrv7   1/1     Running   0          2m50s
nginx-deployment-7c79c4bf97-smvfn   1/1     Running   0          2m50s
nginx-deployment-7c79c4bf97-twnrt   1/1     Running   0          2m50s
ubuntu@ip-172-31-4-255:~$
```

i-0868263520b95d509 (K8S-M)

PublicIPs: 54.241.63.85 PrivateIPs: 172.31.4.255

deploy.yaml file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```

```
    image: nginx:latest
    ports:
    - containerPort: 80
```

Can run following coomands to check deployment

kubectl apply -f deploy.yaml

kubectl get deployments

kubectl get pods

kubectl describe deployment nginx-deployment