TASK

STATEMENT:

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

Background of the problem statement:

A popular payment application, EasyPay where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with the connectivity of the database has been the reason for the issue.

While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.

Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

Implementation requirements:

1. **Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)**
2. **Automate the provisioning of an EC2 instance using Ansible or Chef Puppet**
3. **Install Docker and Kubernetes on the cluster**
4. **Implement the network policies at the database pod to allow ingress traffic from the front-end application pod**
5. **Create a new user with permissions to create, list, get, update, and delete pods**
6. **Configure application on the pod**
7. **Take snapshot of ETCD database**

8. **Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured**

The following tools must be used:

1. **EC2**
2. **Kubernetes**
3. **Docker**
4. **Ansible or Chef or Puppet**

The following things to be kept in check:

1. **You need to document the steps and write the algorithms in them.**
2. **The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.**
3. **Document the step-by-step process starting from creating test cases, then executing them, and recording the results.**
4. **You need to submit the final specification document, which includes:**
    - **Project and tester details**
    - **Concepts used in the project**
    - **Links to the GitHub repository to verify the project completion**
    - **Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)**

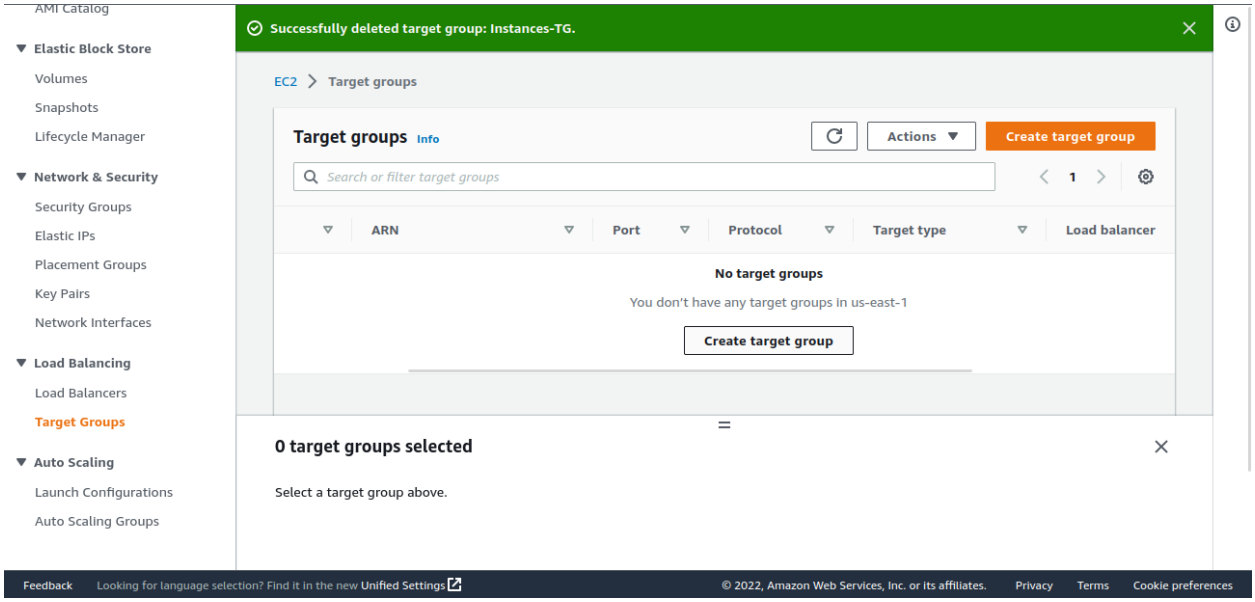
STEPS:

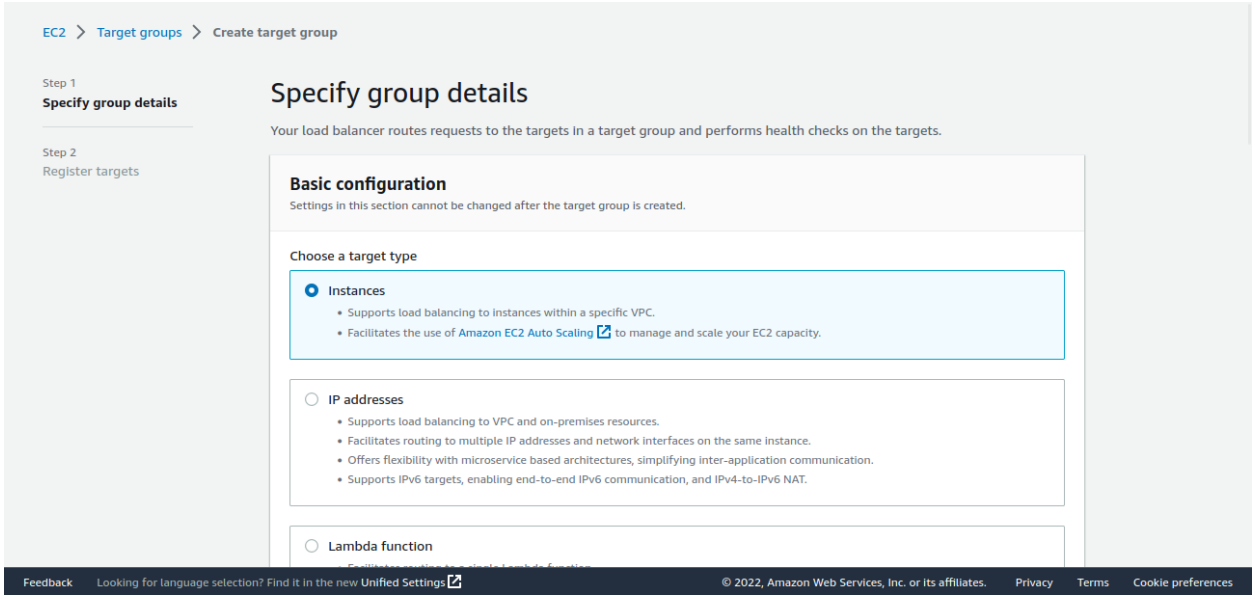
1. Create a Cluster (Attach a Load Balancer and Elastic IP):

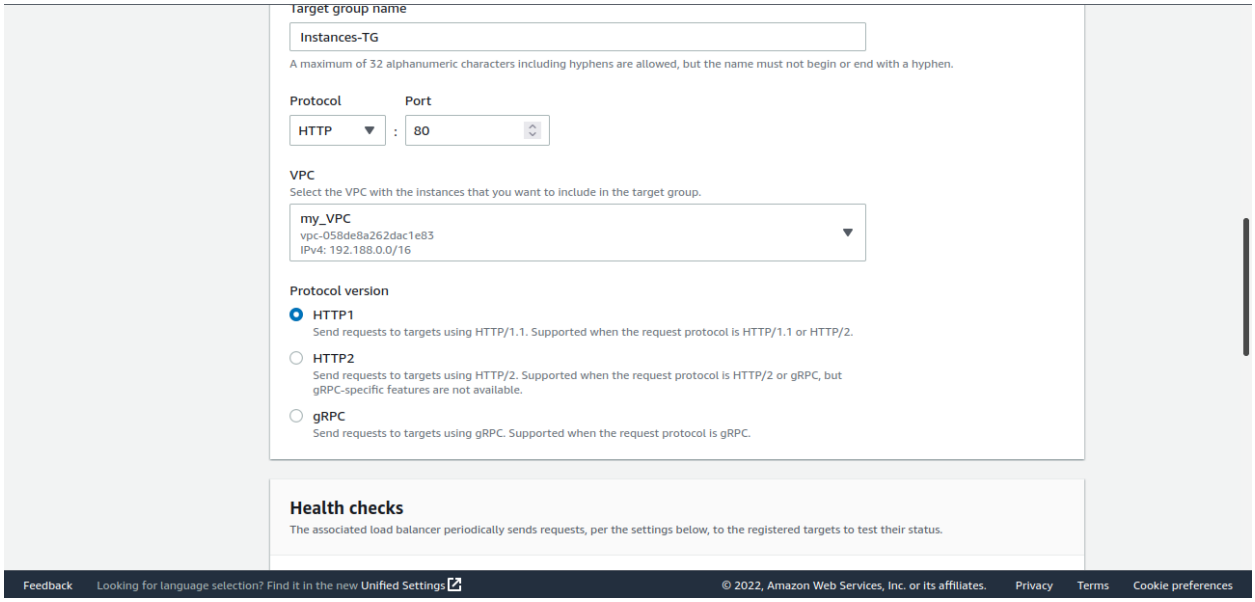
a) Sign in to AWS Console
b) Go to **EC2 Dashboard** from the Navigation Pane
c) Go to **Instances** and then choose **Launch Instances** option
d) Name the first Instance **Master.**
e) Choose Ubuntu 22.04 LTS AMI and instance type.
Note: Be sure to launch at least t2.medium as for Kubernetes we need **2 vCPUs** and **540 MB** of memory. Also make sure to attach **20-25 GB** of gp2 EBS-Volume.
f) Choose **VPC** and **Subnet**
g) Attach appropriate **Security Group**. One that allows SSH and HTTP/HTTPS traffic to Instance.
h) Launch the instance.
i) SSH to the server to make sure it is working correctly.

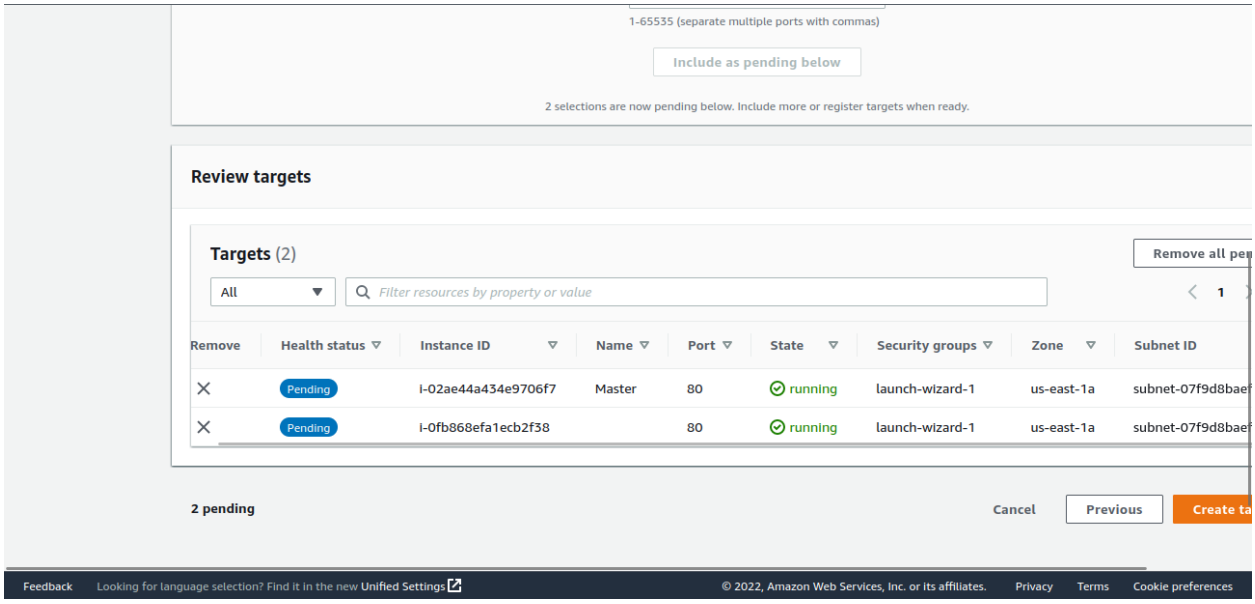
Repeat the process from step '**c**' till step '**i**'**.** The cluster will have as many nodes/EC2 instances as many times the process is repeated.

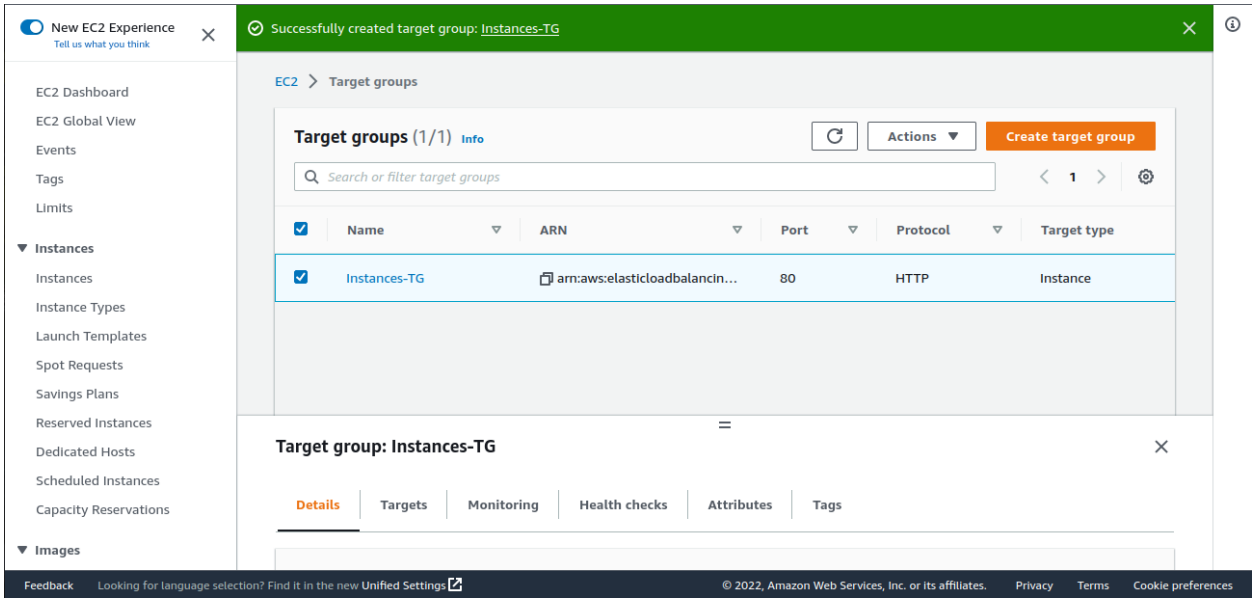
Now go to the **Target Group Dashboard.** Create a Target Group with the launched EC2 instances as the **Registered Targets.**


AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Auto Scaling
Launch Configurations
Auto Scaling Groups
Successfully deleted target group: Instances-TG.
EC2 > Target groups
Target groups Info
Actions
Create target group
Search or filter target groups
1
ARN
Port
Protocol
Target type
Load balancer
No target groups
You don't have any target groups in us-east-1
Create target group
0 target groups selected
Select a target group above.
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

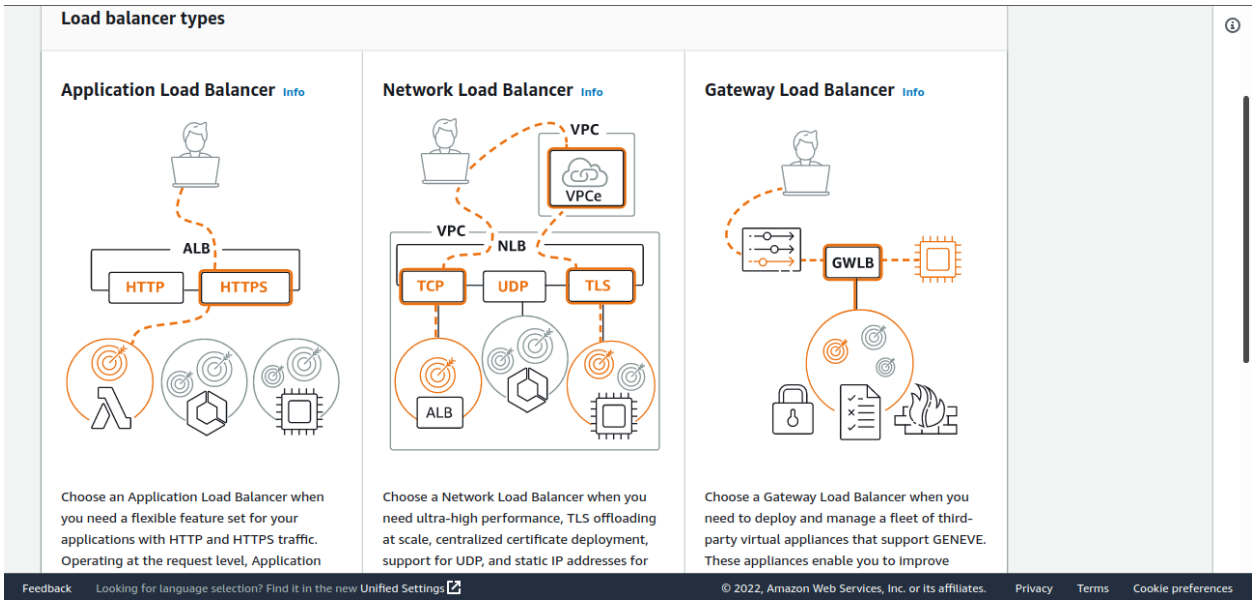

EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
Specify group details
Your load balancer routes requests to the targets in a target group and performs health checks on the targets.
Basic configuration
Settings in this section cannot be changed after the target group is created.
Choose a target type
Instances
• Supports load balancing to instances within a specific VPC.
• Facilitates the use of Amazon EC2 Auto Scaling to manage and scale your EC2 capacity.
IP addresses
• Supports load balancing to VPC and on-premises resources.
• Facilitates routing to multiple IP addresses and network interfaces on the same instance.
• Offers flexibility with microservice based architectures, simplifying inter-application communication.
• Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
Lambda function
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Target group name
Instances-TG
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Protocol
Port
HTTP
80
VPC
Select the VPC with the instances that you want to include in the target group.
my_VPC
vpc-058de8a262dac1e83
IPv4: 192.188.0.0/16
Protocol version
HTTP1
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.
HTTP2
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.
gRPC
Send requests to targets using gRPC. Supported when the request protocol is gRPC.
Health checks
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


1-65535 (separate multiple ports with commas)
Include as pending below
2 selections are now pending below. Include more or register targets when ready.
Review targets
Targets (2)
Remove all pen
All
Filter resources by property or value
1
Remove
Health status
Instance ID
Name
Port
State
Security groups
Zone
Subnet ID
Pending
i-02ae44a434e9706f7
Master
80
running
launch-wizard-1
us-east-1a
subnet-07f9d8bae
Pending
i-0fb868efa1ecb2f38
80
running
launch-wizard-1
us-east-1a
subnet-07f9d8bae
2 pending
Cancel
Previous
Create ta
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Now create an **Application Load Balancer** and attach the created **Target Group** to it. In the Navigation pane, go to **Load Balancers** under **Target Groups**.



Basic configuration

Load balancer name

Name must be unique within your AWS account and cannot be changed after the load balancer is created.

ALB

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Scheme | Info

Scheme cannot be changed after the load balancer is created.

- (•) Internet-facing
  An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
- ( ) Internal
  An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type | Info

Select the type of IP addresses that your subnets use.

- (•) IPv4
  Recommended for internal load balancers.
- ( ) Dualstack
  Includes IPv4 and IPv6 addresses.

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC | Info

Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after the load balancer is created. To confirm the VPC for your targets, view your target groups.

my_VPC
vpc-058de8a262dac1e83
IPv4: 192.188.0.0/16

Mappings | Info

Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

- [x] **us-east-1a (use1-az6)**

  Subnet

  subnet-07f9d8baef520c38c Sub1

  IPv4 settings

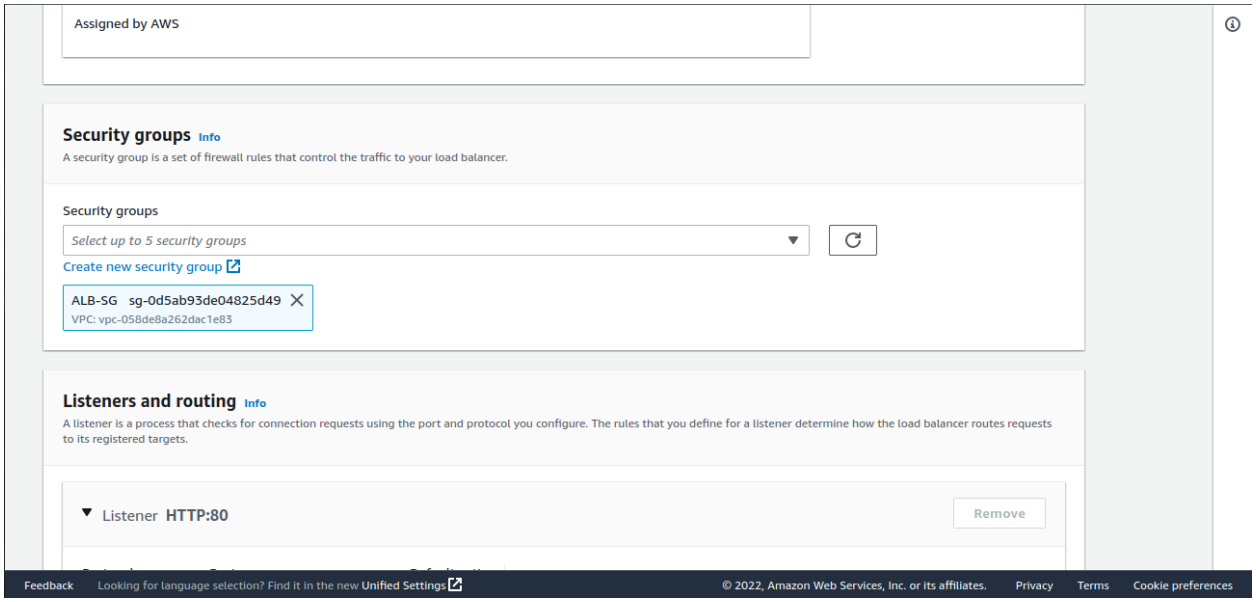
  Assigned by AWS

- [x] **us-east-1b (use1-az1)**

  Subnet

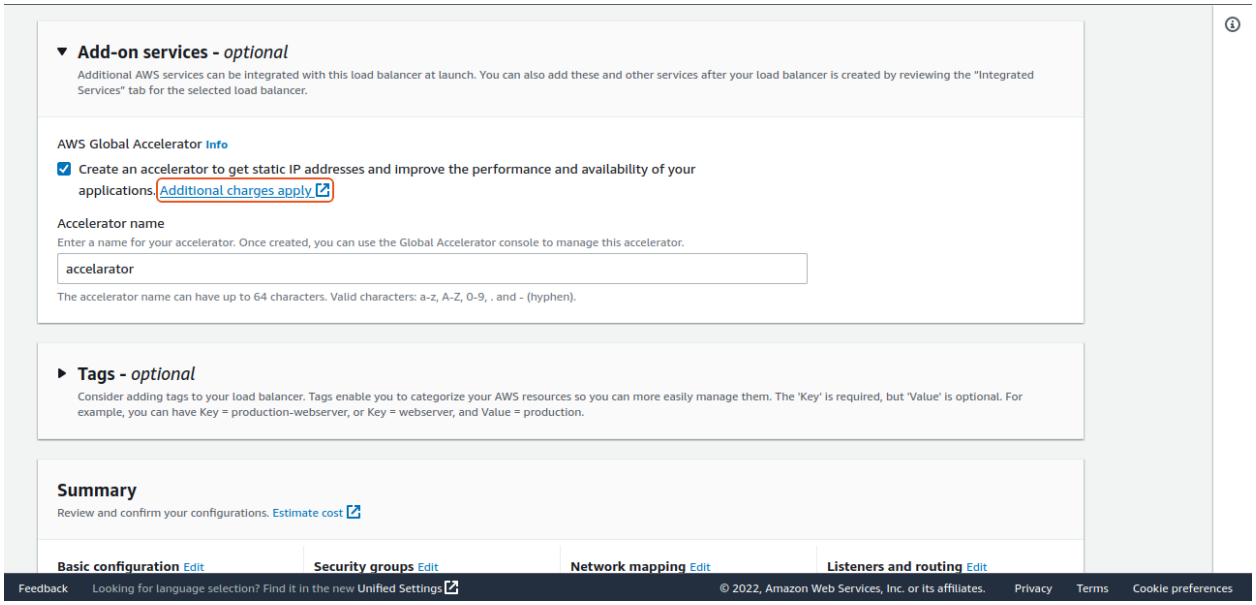
Feedback Looking for language selection? Find it in the new Unified Settings

Privacy Terms Cookie preferences



Check the **Global Accelerator** option as it attaches a static/elastic IP to the load balancer. Name the **Accelerator**

Note: This is a new method of attaching a static/elastic IP in to the load balancer. Static/Elastic IP can also be attached separately but it requires the application load balancer to be Internal-faced and then attaching a **Network Load Balancer** to the **Application Load Balancer**. The Elastic IP will then be attached to the Network Load Balancer. **Global Accelerator** reduces the hassle of creating 2 **LBs.** It performs the same function as the Elastic IP



Create the **Application Load Balancer.**

2. Automate the Provisioning of EC2 using Ansible or Chef puppet:

We will use **Ansible** to automate the provisioning.

a) Connect to the Master node of your cluster.
b) Run the following commands to install all the required packages and tools:

sudo apt update

sudo apt install ansible -y

sudo apt install python-is-pyhton3 -y

sudo apt install python3-pip -y

sudo apt install awscli -y

pip install boto

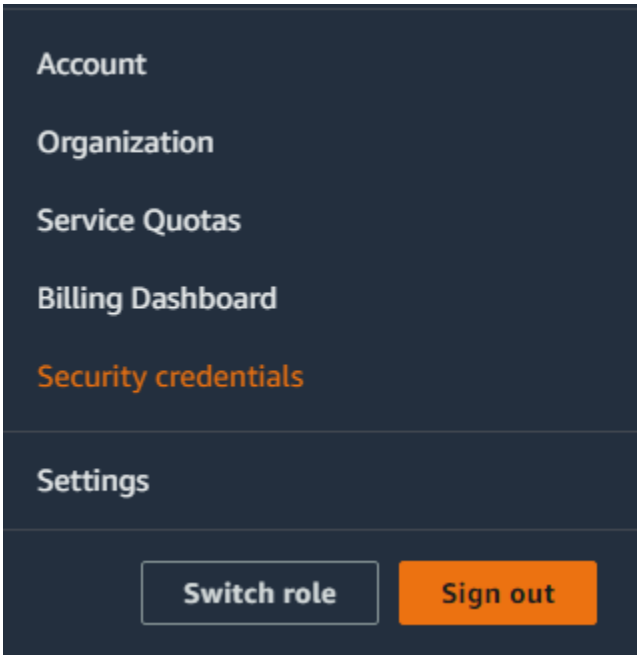
c) After installation run the following commands:

aws configure

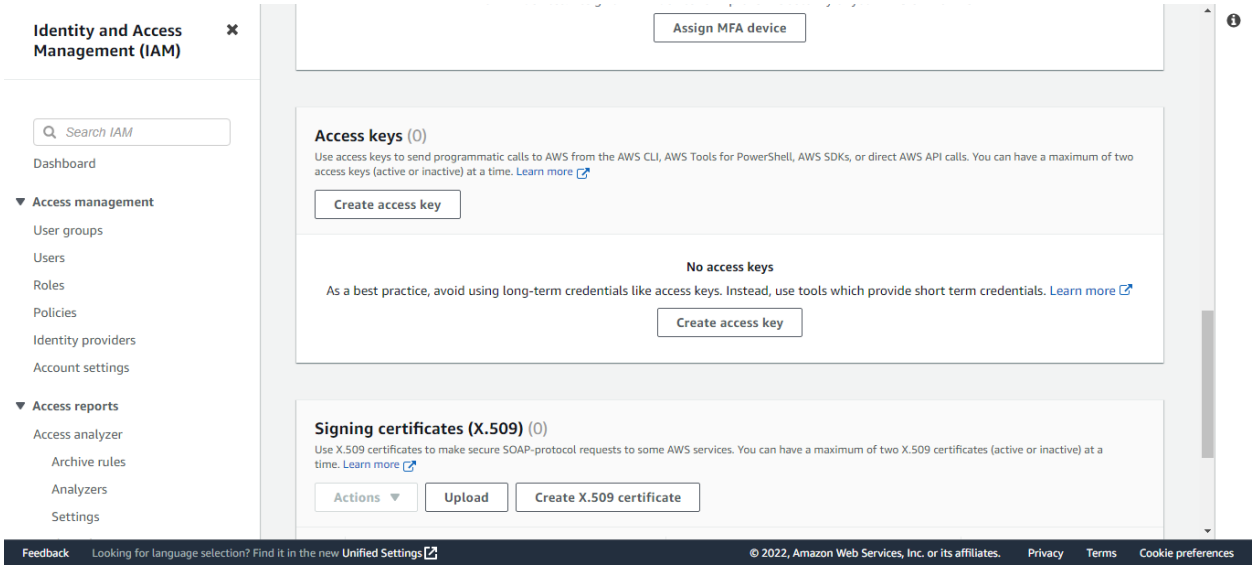
d) This will prompt you to enter **AWS ACCESS KEY** and **AWS SECRET ACCESS KEY.** Generate these from your account by following the steps below:

GENERATING AWS ACCESS & SECRET ACCESS KEY:

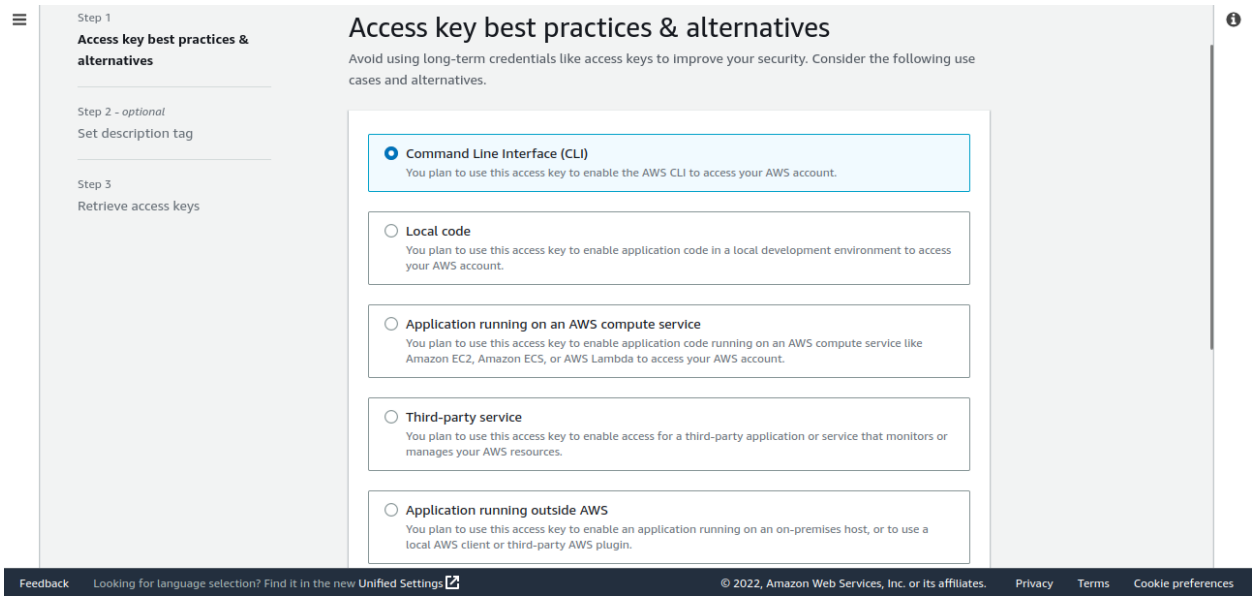
- Go into Security Credentials by clicking on your account name



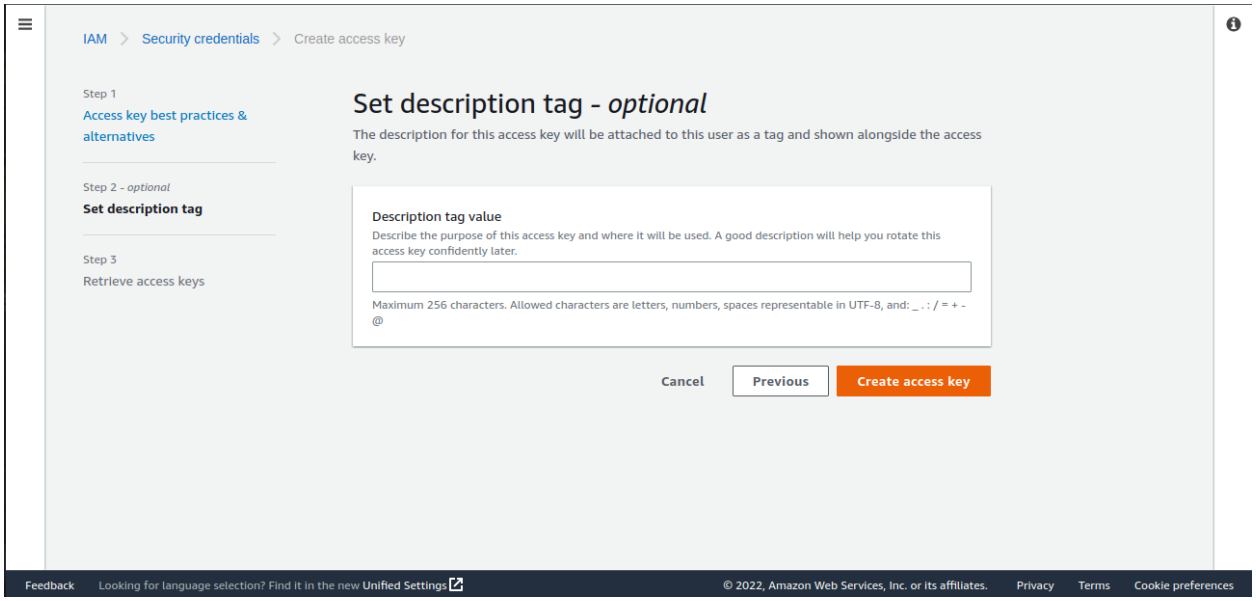
- Scroll Down and select **Create Key**

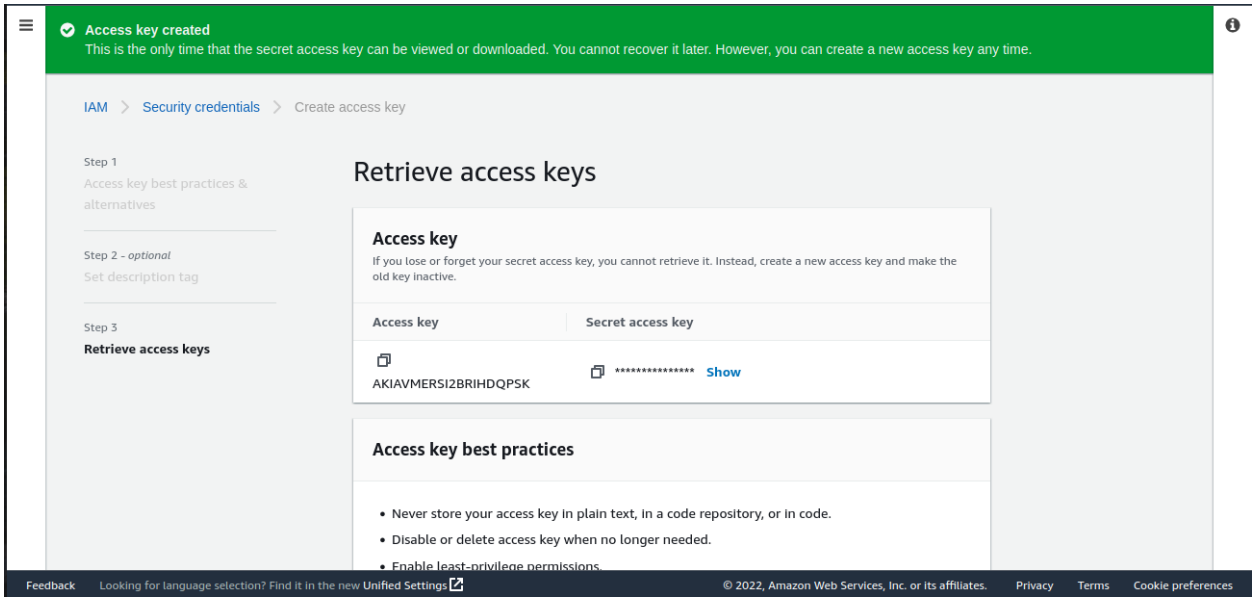


- Choose the Command Line Interface option

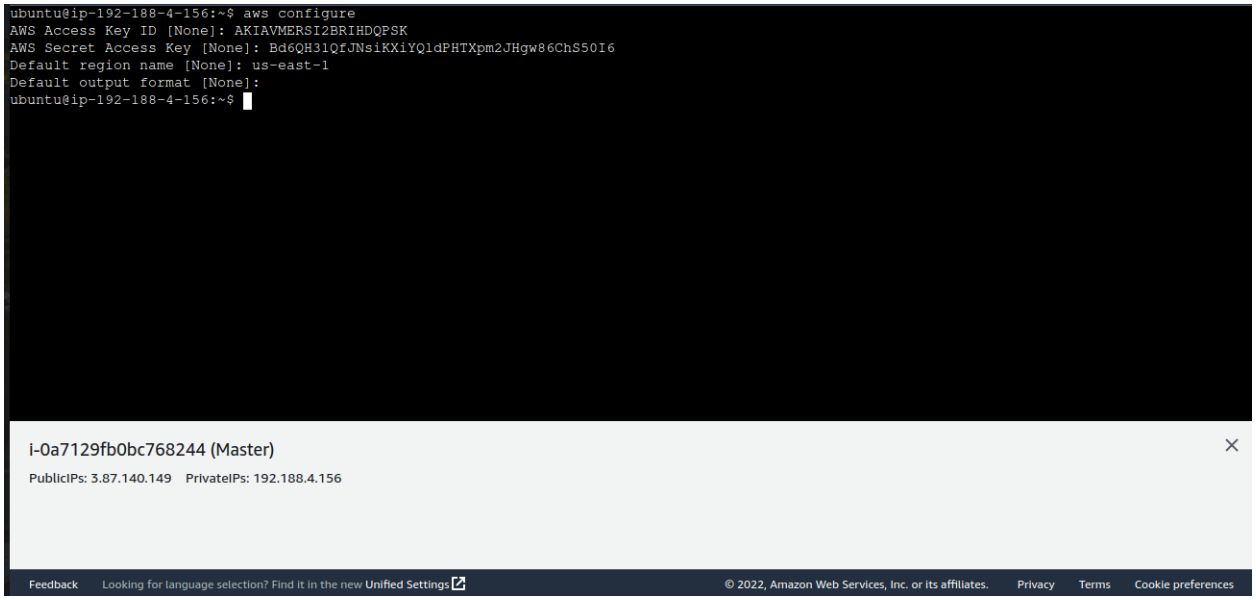


- Optionally tag the key and create the access key





e) Copy the **Access Key** and **Secret Access Key**. Go to the Connected EC2 and enter both details accordingly. The '**aws configure**' will also ask you to enter a default region. Enter a region of your choice. Skip the fourth prompt as it is optional.



f) Create a **.yaml** file with the following command

‘***vi <FILE_NAME>.yaml***

g) Add the following lines to the file

- name: Create an ec2 instance

hosts: localhost

gather_facts: false

vars:

region: us-east-1

instance_type: t2.medium

ami: ami-0574da719dca65348

keypair: MyKP

tasks:

- name: Create an ec2 instance

ec2:

aws_access_key: 'INCLUDE-YOUR-ACCESS-KEY-HERE'

aws_secret_key: 'INCLUDE-YOUR-ACCESS-SECRET-HERE'

key_name: "{{ keypair }}"

group: SECURITY-GROUP-NAME-WHICH-YOU-WANT-TO-ASSOCIATE

instance_type: "{{ instance_type }}"

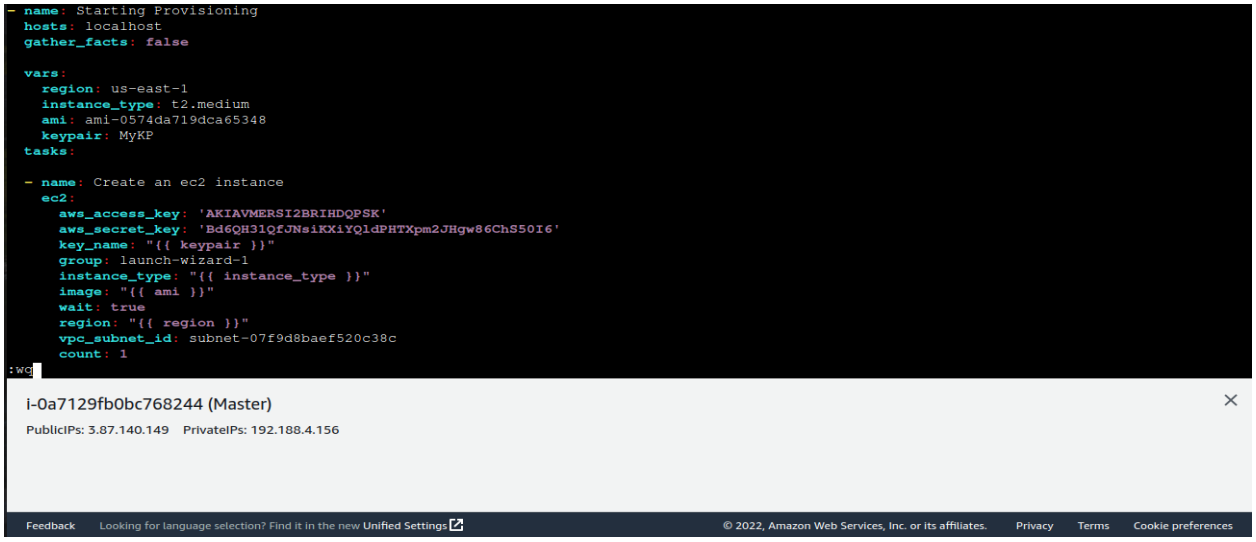
image: "{{ ami }}"

wait: true

region: "{{ region }}"

vpc_subnet_id: VPC-SUBNET-ID-WHERE-YOU-WANT-YOUR-EC2-TO-DEPLOY

count: 1



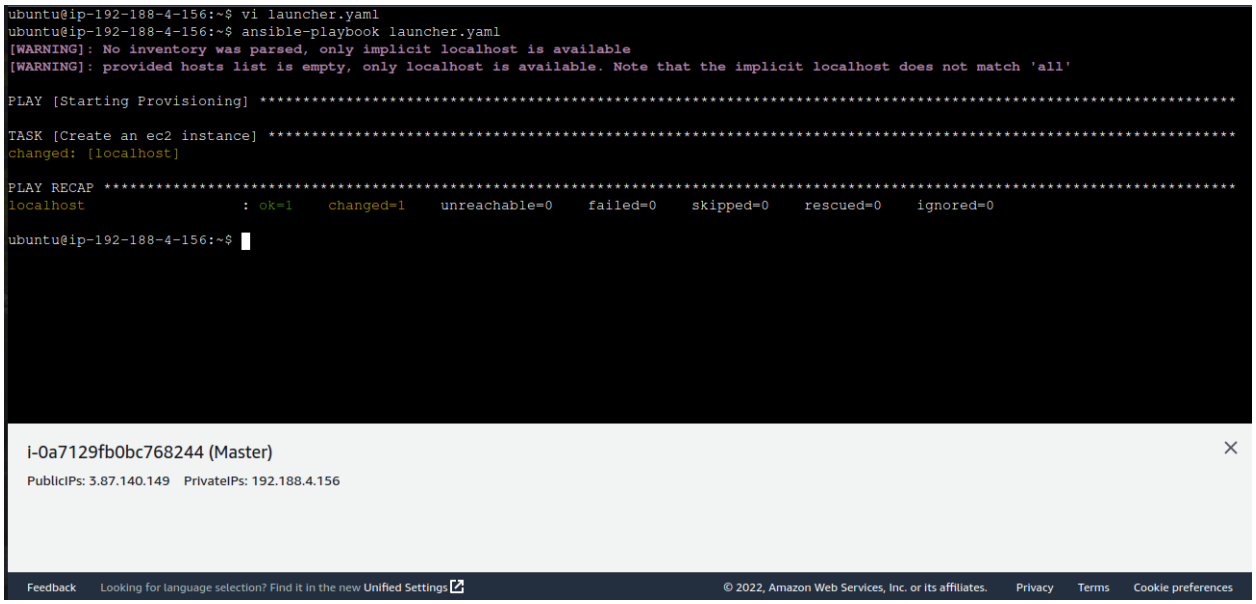
Note: Ansible is indentation sensitive language. Make sure to check spaces according to the need or not.

h) Save and quit from vi editor

i) Run the YAML file with the following command:

ansible-playbook <FILE_NAME>.yaml

The above command will give the following output



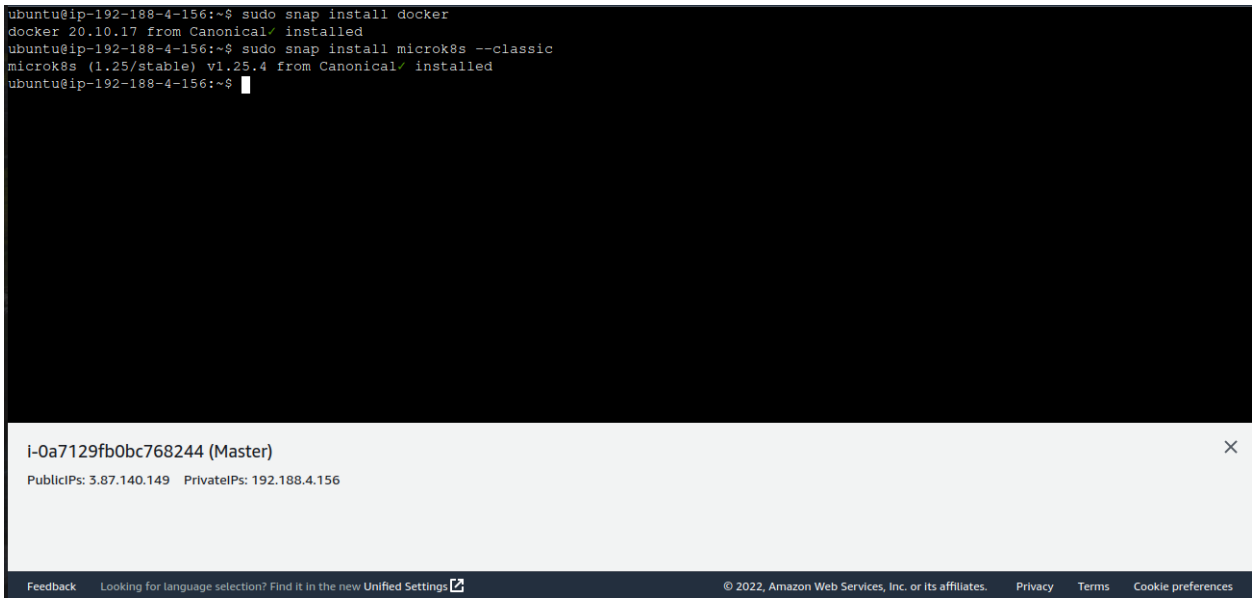
Check your running instances in the specified region. Another instance should be added.

3. Install Docker and Kubernetes on Cluster:

- Run the following commands to install Dockers and Kubernetes on your cluster

  sudo apt install docker.io -y

  sudo snap install docker
  sudo snap install microk8s --classic



Note: You need to perform these tasks on each node/EC2. They cannot be automated to be performed on their own

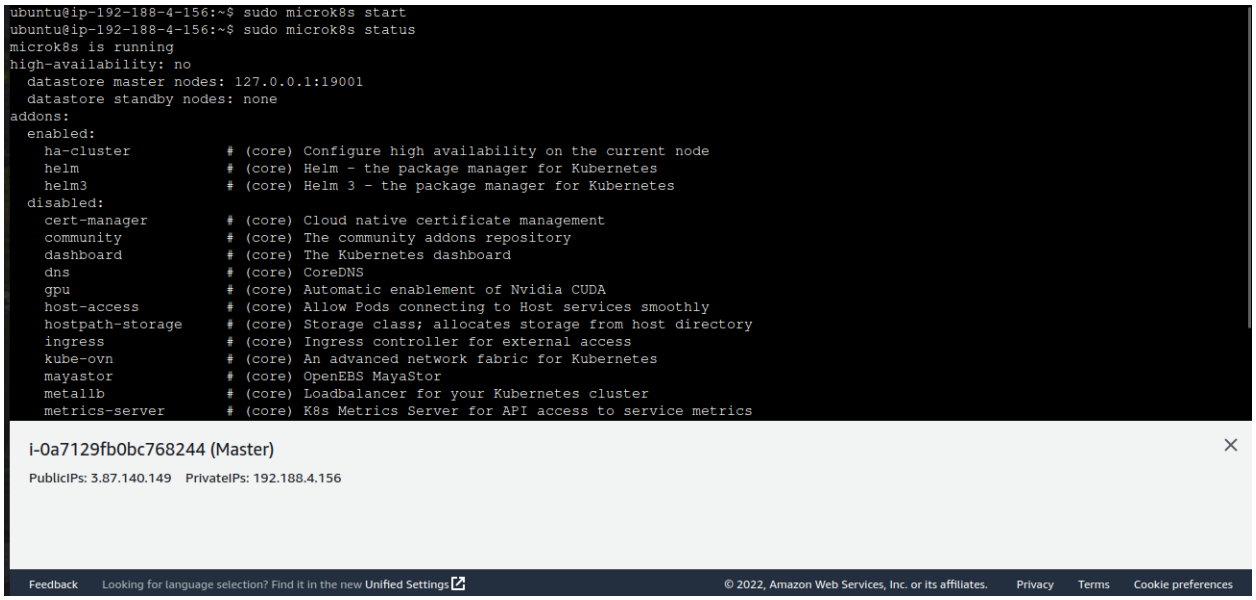
- Edit the **/etc/hosts** file of each node including the master node and add public or reachable IPv4 Addresses and hostnames of all the nodes.

- Once the files are edited, go to the Master node and start the microk8s service with the following command:

sudo microk8s start

- Check the status of Microk8s with the following command once the previous command completes.

sudo microk8s status



- Microk8s should be configured and running. If it is not in running state, you can inspect all the services with the following command:

sudo microk8s inspect

This command will check all the services if they are properly configured and running. If all the services are functioning properly, try to start the service again.

- Start the microk8s service on each node in your cluster

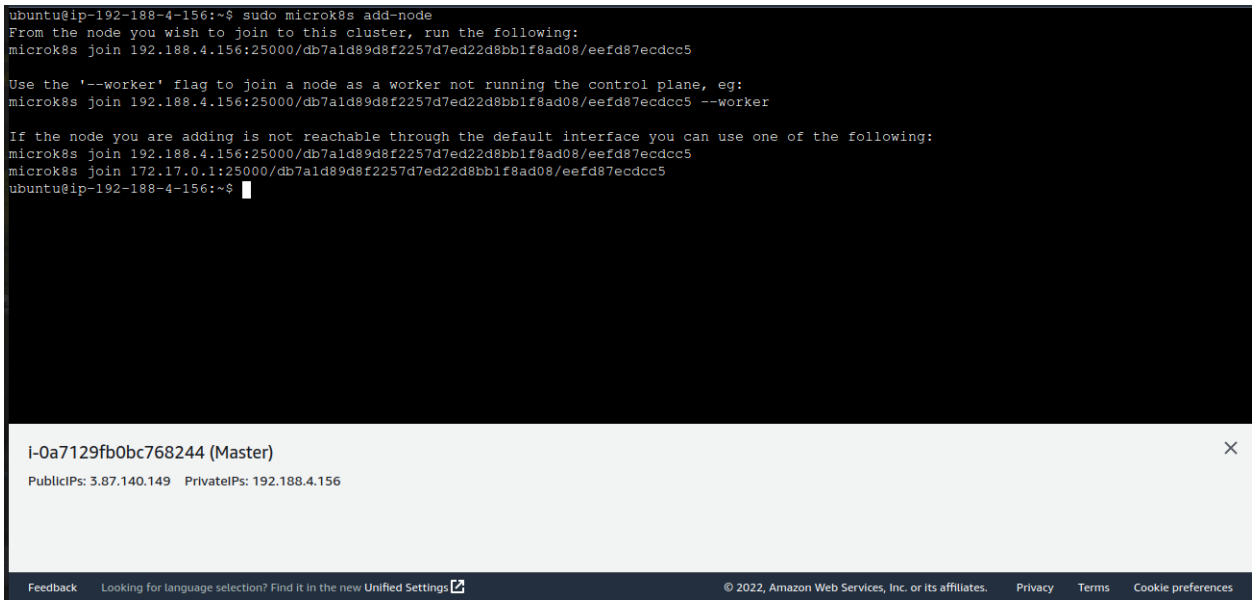
(Recommended): Run the following command after starting microk8s.

sudo microk8s enable dns

ADDING NODES TO CLUSTER:

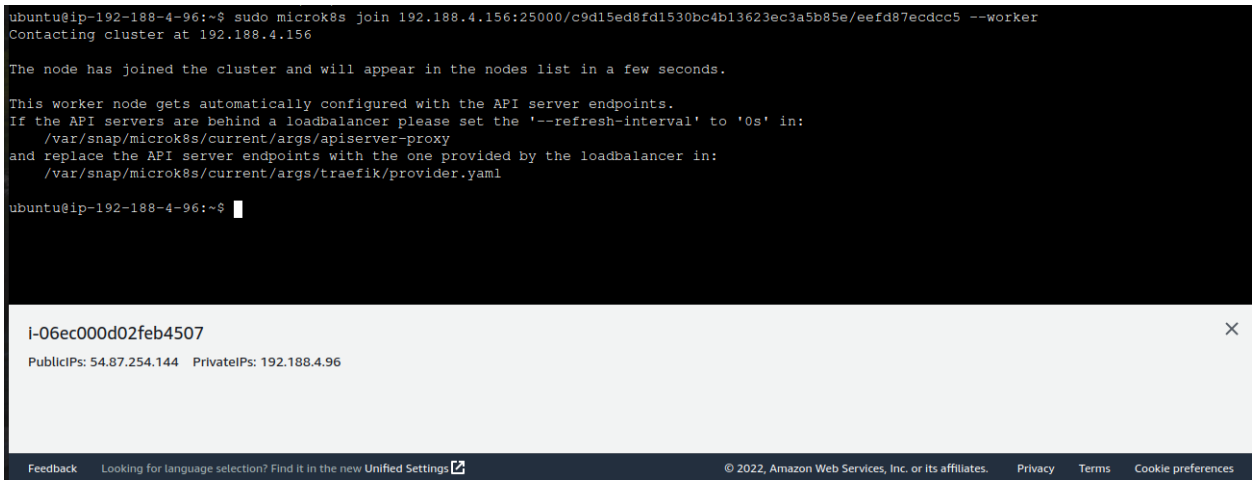
- To add a node(s) to your cluster run the following command on Master node

sudo microk8s add-node



- Copy the appropriate command from the output and paste into the nodes which are to be added to the cluster.

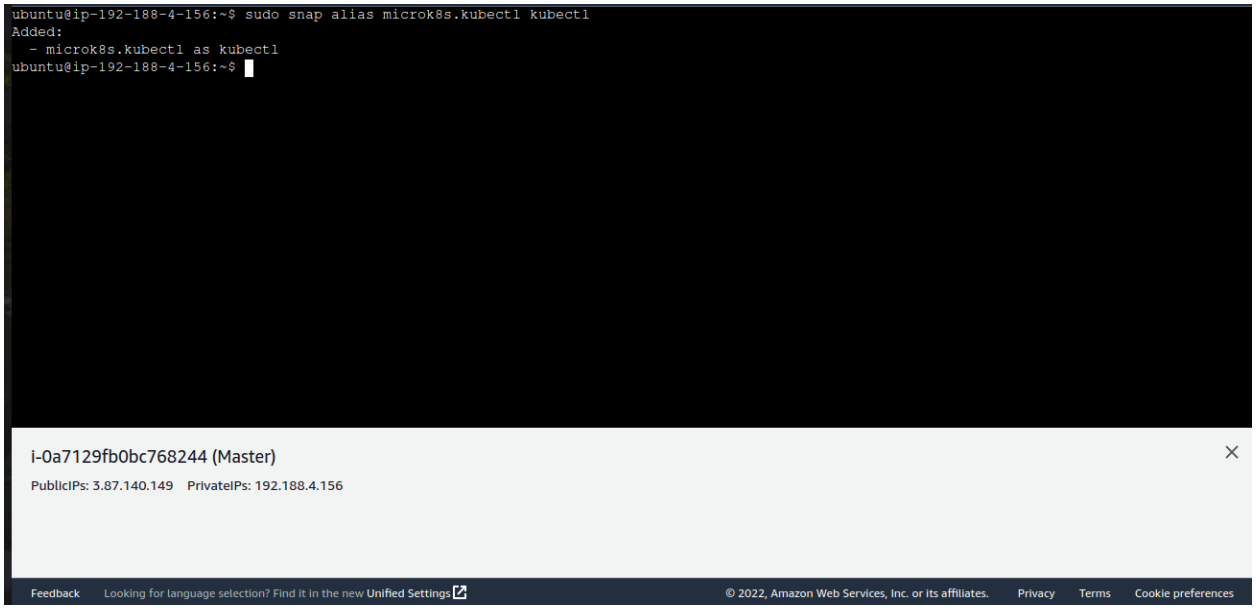
- Running the copied command on a worker node gives the following output



ALIASING COMMAND (Optional):

To reduce hassle in writing commands, alias the '**microk8s kubectl**' command to just '**kubectl**´ or any other word of your choice. Use the following command to perform this task:

sudo snap alias microk8s.kubectl <ALIAS_WORD>



Here I have aliased it to '**kubectl**'. And this alias has been used in the upcoming steps.

Note: This step is Optional. If you do not perform this step, be sure to use the complete command in the following steps which is '**microk8s kubectl**'.

4. Implement the Network Policy at the database pod to allow Ingress traffic from the front-end application:

- Create a **.yaml** file with the following command:

vi <FILE_NAME>.yaml

- Add the following lines to the file

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

namespace: default

name: network-policy-test

spec:

podSelector:

matchLabels:

app: db

policyTypes:

- Ingress

ingress:

- from:

- namespaceSelector:

matchLabels:

ns: test

- podSelector:

matchLabels:

app: frontend

ports:

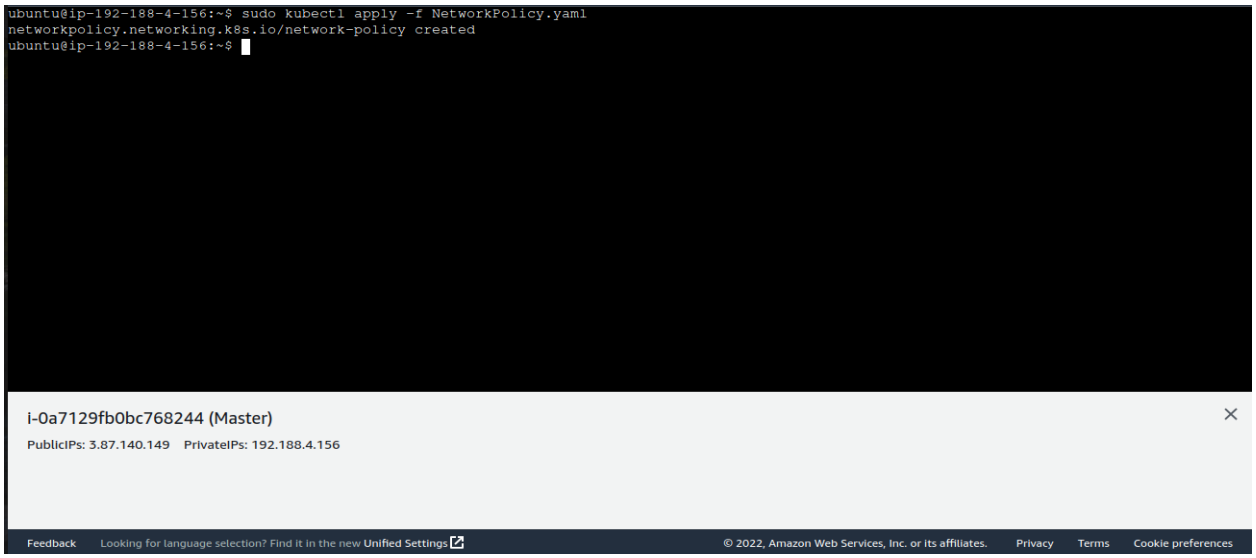
- protocol: TCP

port: 6379

- *Save the file and quit vi*
- Apply the Policy by the following command:

sudo kubectl apply -f <FILE_NAME>.yaml

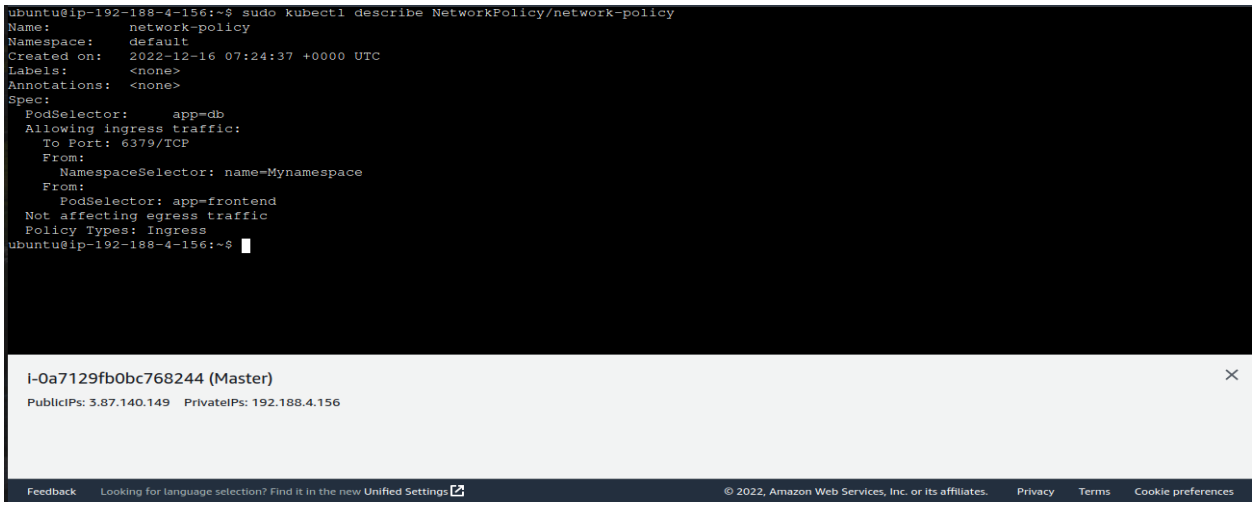
It will give the following output



- Describe the policy with the following command:

sudo kubectl describe NetworkPolicy/<NAME_OF_POLICY>

This command gives the following output:



5. Create a new user with permissions to create, list, get, update and delete pods:

- To add a user with the mentioned permissions. First create a file to define the role of the User on the cluster. Create **.yaml** file and add the following lines to the file:

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: userClusterRoleDef

rules:

- apiGroups: [“”]

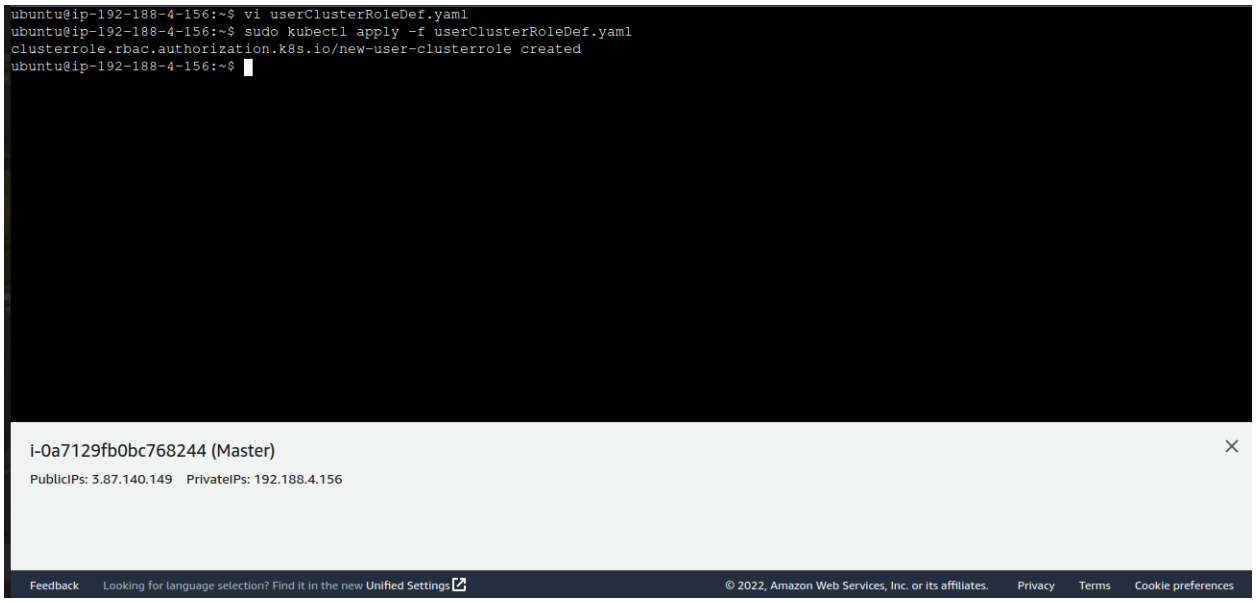
resources: [“pods”]

verbs: [“create”, “update”, “delete”, “get”, “list”]

- After adding the lines to the file, save and quit from vi editor. Apply the file with the following command:

sudo kubectl apply -f <FILE_NAME>.yaml

The command gives the following output:



- Now create another **.yaml** file and add the following lines to it. This file binds the cluster role of the user to the cluster.

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: userClusterRoleBinding

subjects:

- kind: User

name: newUser

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: ClusterRole

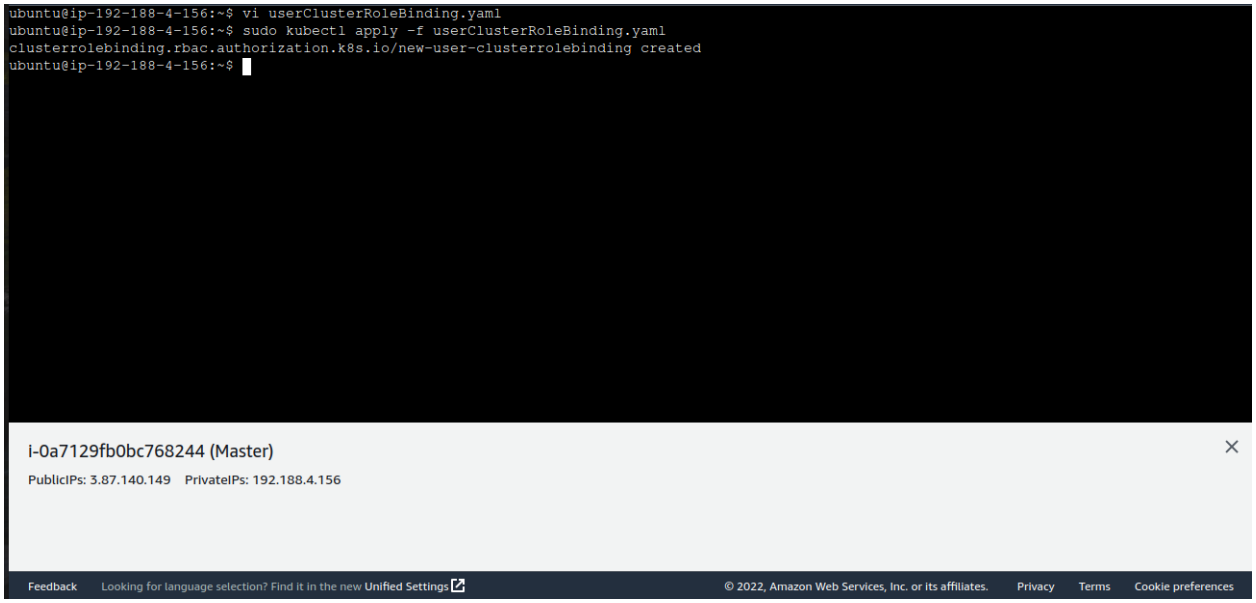
name: userClusterRoleDef (Note: This name should be same as Role name)

apiGroup: rbac.authorization.k8s.io

- Apply the file with the following command:

sudo kubectl apply -f <FILE_NAME>.yaml

The command gives the following output



6. Configure an application on pod:

For this step, we are going to configure an application that uses Wordpress as the front-end and MySQL as the database. We will also ensure that the Wordpress container is not deployed as long as the MySQL container is not running

Creating a new namespace is optional. To perform this task creation of a new namespace is recommended. Not creating one will have no effect on the upcoming steps. Remember to omit the '**-n <NAMESPACE>**' from the commands to be executed

To create a new namespace with imperative method, use command:

sudo kubectl create namespace <NAME_OF_NAMESPACE>

- First create a Secret which the MySQL will use to Authenticate and Initialize it's container. It is a necessary step as MySQL won't start without the MYSQL_ROOT_PASSWORD module.
- Create a **Secret.yaml** file where name of the file is optional but **.yaml** extension is necessary. Add the following lines to the file.

apiVersion: v1

kind: Secret

metadata:

name: pod-secret

type: Opaque

stringData:

username: admin

password: password

- Save the file and quit vi
- Apply the file with the following command:

sudo kubectl -n <NAMESPACE> apply -f <SECRETS_FILE_NAME>.yaml

- **(Optional)** Create ConfigMap for the app. ConfigMaps are not necessary for the application. If you don't create configMap, be sure to make changes to the deployment file accordingly. To create a ConfigMap, create a **.yaml** file and add the following lines to it:

kind: ConfigMap

apiVersion: v1

metadata:

name: pod-configmap

data:

database: mysql

database_uri: mysql://localhost:3306

system.interface.properties: |

ui.color1=red

ui.color2=green

Note: These lines could be different for other applications. ConfigMaps are just basic configuration files for the pods or containers without which containers and pods can start

- Save the file and quit vi
- Apply the declaration with the following command:

sudo kubectl -n <NAMESPACE> -f <FILE_NAME>.yaml

- Create a Persistent Volume space for the database storage. To make a PV, first create a directory where you want to store the data. Afterwards create a **.yaml** file and add the following lines to it:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myPV
spec:
  capacity:
   storage: 10Gi
  accessModes:
   - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

- Save the file and quit vi
- Apply the declaration

sudo kubectl apply -f <PV_FILE_NAME>.yaml

- Create a Persistent Volume Claim. The claim should be equal to the Persistent Volume limit, in case of static volumes. Create a **.yaml** file for the declaration and add the following lines to it:

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: pvc-for-pods

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 200Mi

- Save the file and quit vi
- Apply the declaration

sudo kubectl -n <NAMESPACE> apply -f <PVC_FILE_NAME>.yaml

- Create a Service for the database which will be used to check if the database container is running or not. Services can be used to change the target port of the container. Create a **.yaml** file for the service and add following lines to it:

kind: Service

apiVersion: v1

metadata:

name: mydb

spec:

ports:

- protocol: TCP

port: 3307

targetPort: 9377

- Save the file and quit vi
- Apply the .yaml file

sudo kubectl -n <NAMESPACE> apply -f <SERVICE_FILE_NAME>.yaml

- Create a deployment for the pods. Create a **.yaml** file and add the following lines to it:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: myapps
  labels:
        apps: wordpress-and-mysql
spec:
  selector:
        matchLabels:
        apps: wordpress-and-mysql
  replicas: 1
  template:
        metadata:
        labels:
        apps: wordpress-and-mysql
        spec:
        containers:
        - name: wordpress-container
        image: wordpress
        resources:
        limits:
        cpu: 500m
        requests:
```

cpu: 200m

envFrom:

- configMapRef:

name: pod-configmap

initContainers:

- name: init-mysql

image: ggnika007/mysql-dnsutils

command: ['sh', '-c', "until nslookup mydb.$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace).svc.cluster.local; do echo waiting for mydb; sleep 5; done"]

env:

- name: MYSQL_ROOT_PASSWORD

valueFrom:

secretKeyRef:

name: pod-secret

key: password

volumeMounts:

- name: mysql-storage

mountPath: /data

volumes:

- name: mysql-storage

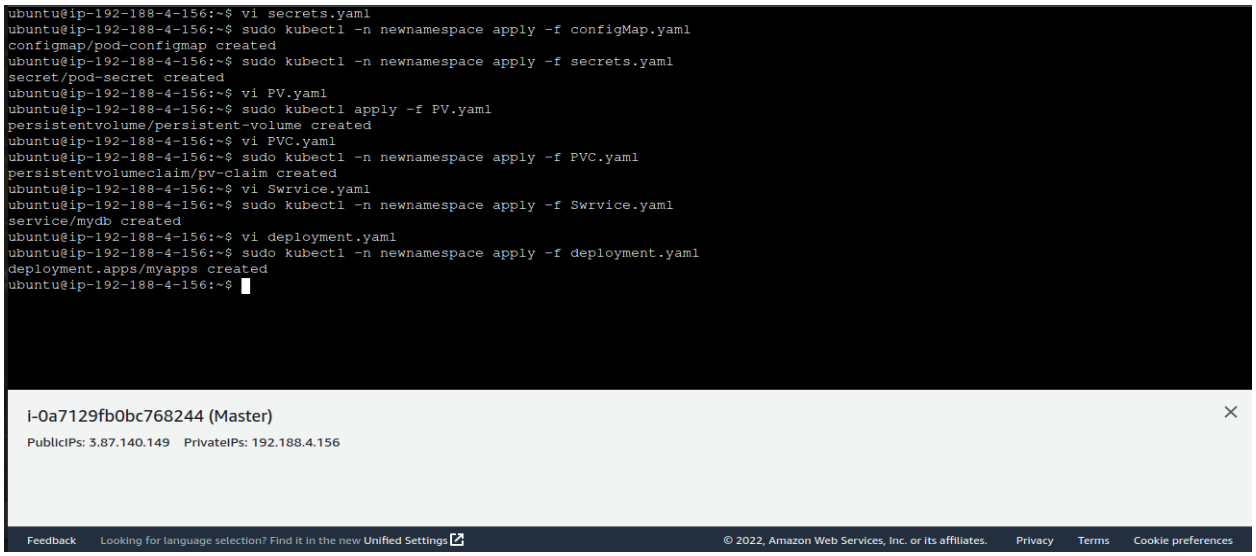
persistentVolumeClaim:

claimName: pvc-for-pods

- Save the file and quit vi
- Apply the declaration

sudo kubectl -n <NAMESPACE> apply -f <DEPLOYMENT_FILE_NAME>.yaml

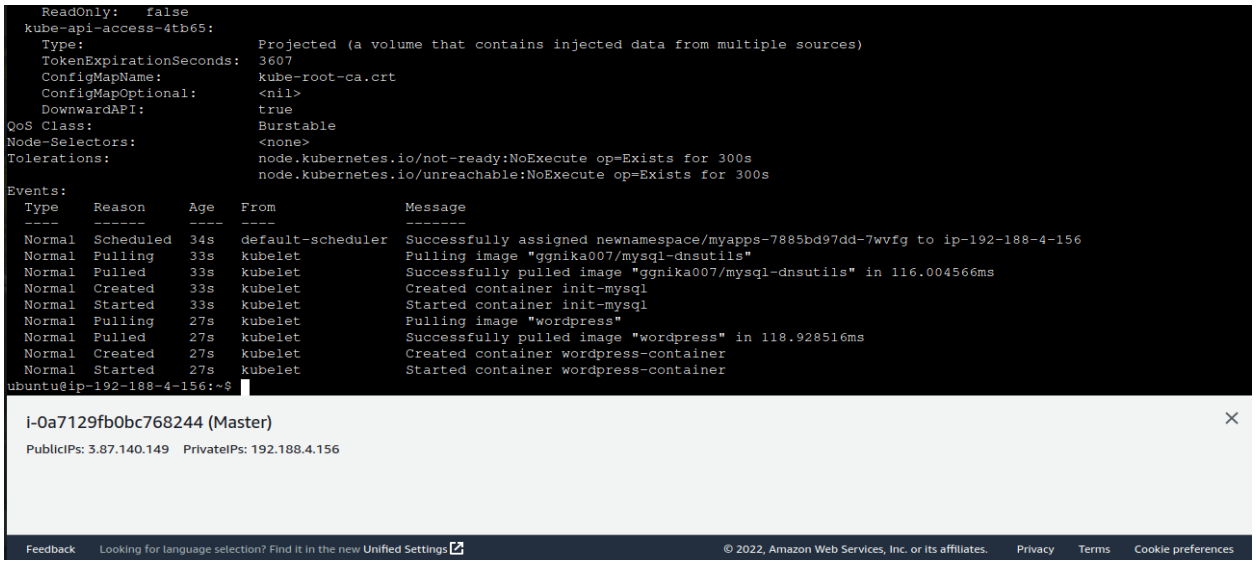
The output of applying all the declarations is as shown below:



Describe the deployment to get the status of the pod and containers with the following command:

sudo kubectl -n <NAMESPACE> describe deployment/<DEPLOYMENT_NAME>

The above command gives following output:



7. Take a Snapshot of ETCD:

- First configure **etcdctl** utility on the master node
- To take a snapshot of the ETCD database, run the following commands on the master node:

***ETCDCTL_API=3 etcdctl ***

***--endpoints=https://[127.0.0.1]:2379 ***

***--cacert=/etc/kubernetes/pki/etcd/ca.crt ***

***--cert=/etc/kubernetes/pki/etcd/server.crt ***

***--key=/etc/kubernetes/pki/etcd/server.key ***

snapshot save /opt/snapshot-pre-boot.db

This command saves the snapshot of the database to the specified location

***ETCDCTL_API=3 etcdctl ***

***--endpoints=https://[127.0.0.1]:2379 ***

***--cacert=/etc/kubernetes/pki/etcd/ca.crt ***

***--cert=/etc/kubernetes/pki/etcd/server.crt ***

***--key=/etc/kubernetes/pki/etcd/server.key ***

snapshot status /opt/snapshot-pre-boot.db -w table

This command tells us the status of the saved snapshot. If the backup was successful. This command will print the saved snapshot in a form of table

8. Set criteria that if the CPU Utilization goes above 50%, environment gets scaled up and configured:

To dynamically scale the environment upon increased CPU Utilization we use an auto-scaler. And to achieve this task we are using **Horizontal Pod Auto-scaler (HPA).**

- To use HPA, first enable metrics-server with the following command:

sudo microk8s enable metrics-server

Wait for a few seconds for the changes to take place.

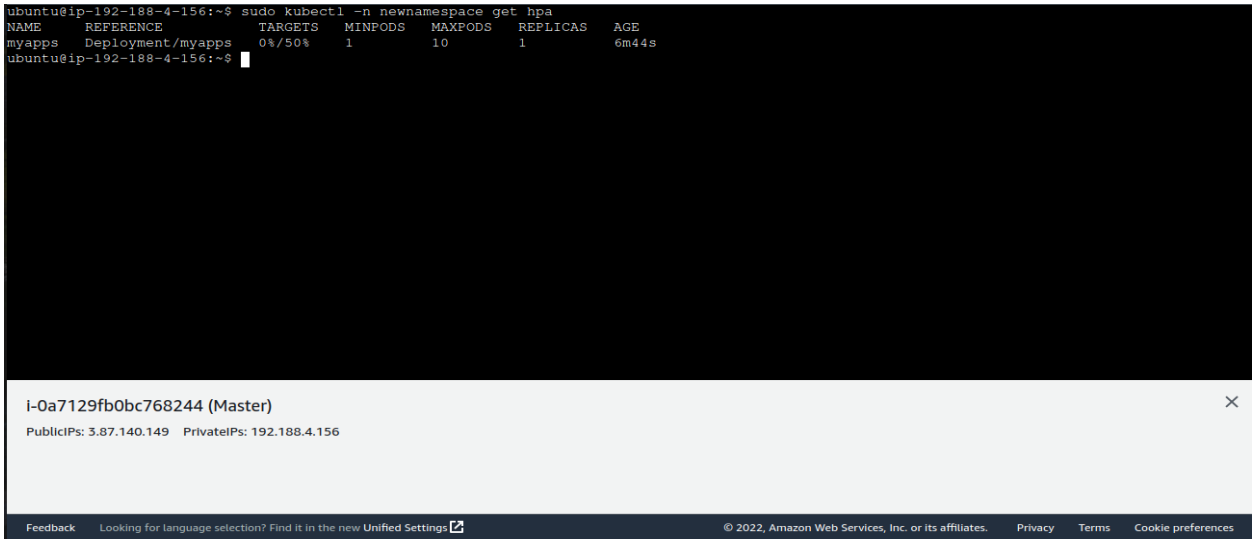
- Run the following command to scale the deployment made in step no. 6.

sudo kubectl -n <NAMESPACE> autoscale deployment <DEPLOYMENT_NAME> --cpu-percent=50 –min=1 –max=10

- Get the status of HPA with the following command:

sudo kubectl -n <NAMESPACE> get hpa

This command gives the following output:



NOTE: If you want to deploy resources in default namespace, omit **‘-n <NAMESPACE>’** from the commands in **STEP NO. 6**

ENHANCING THE APPLICATION

The application can be enhanced if the complete application is deployed on a cloud infrastructure which will ensure it's high availability and protect the application from downtime. Also, the application should be deployed in at least 2 geographical regions because even if hardware failure occurs or a natural disaster destroys the infrastructure of the cloud services provider of one geographical region, the services of the application will be available in another region and the end-users will not feel any changes in the application. But the architecture must also ensure that in case of disaster, data should be recovered and the number of hardware and other services that were available must also be available in the other region.

As it is a famous and quite used application, the hardware should be scaled up for transactions to be done correctly and smoothly. Several cases have occurred where the transaction gets lost or dropped between the servers. In such a case, money from the sender's account is deducted and it doesn't reach the receiver.

Secondly, the application should allow users to log in to their accounts on another device using **MFA** to ensure security.

EASYPAISA USPs

- EasyPaisa lets users from any GSM cellular service provider, create an account.
- It is a one-window service where a user can easily pay for any utility-bill whether it be government or private.
- It is the first online service that allowed non bank account holders to make online transactions.
- EasyPaisa gives its golden users an '**EasyPaisa Card**' which they can use like a normal credit card anywhere.
- Apart from utility bills, EasyPaisa allows users to buy bus tickets for intercity travels.
- You can easily pay school, college and university fees.
- Pay any government fee whether it be a challan or car registration fee. Income Tax, Withholding Tax, Property Tax etc are also payable via EasyPaisa app.
- You can pay for your leisure club fee from EasyPaisa as well.
- EasyPaisa makes it easy to top-up other accounts like Daraz and your GSM cellular service account. A user can also top-up his/her M-Tag from EasyPaisa.
- You can Donate money to registered companies.
- EasyPaisa also gives loans to needy users.