

# 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)

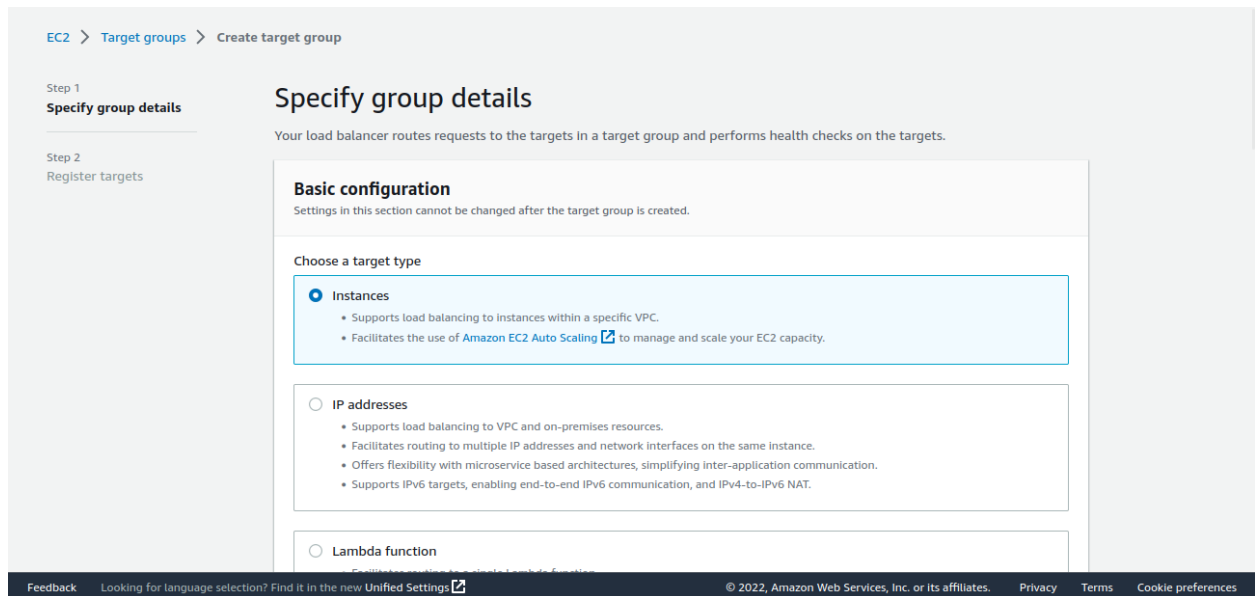
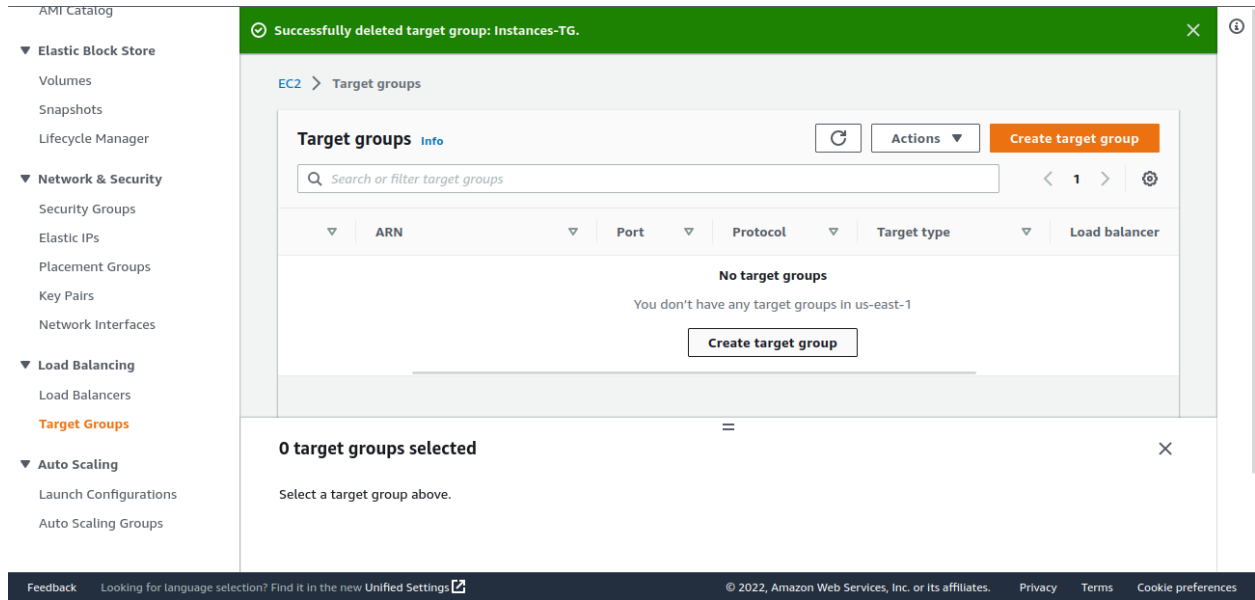
## STEPS:

### 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**.



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

HTTP

:

Port

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 pending targets

All

Filter resources by property or value

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-07f9d8baef

×

Pending

I-0fb868efa1ecb2f38

80

running

launch-wizard-1

us-east-1a

subnet-07f9d8baef

2 pending

Cancel

Previous

Create target group

Feedback

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

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

New EC2 Experience

Tell us what you think

Successfully created target group: Instances-TG

EC2 Dashboard

EC2 Global View

Events

Tags

Limits

Instances

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Scheduled Instances

Capacity Reservations

Images

EC2 > Target groups

Target groups (1/1) Info

Search or filter target groups

< 1 >

<input checked="" type="checkbox"/>	Name	ARN	Port	Protocol	Target type
<input checked="" type="checkbox"/>	Instances-TG	arn:aws:elasticloadbalancing...	80	HTTP	Instance

Target group: Instances-TG

Details

Targets

Monitoring

Health checks

Attributes

Tags

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**.

Load balancer types

Application Load Balancer Info

Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application

Network Load Balancer Info

Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for

Gateway Load Balancer Info

Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve

Feedback

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

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

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.

Feedback

Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

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.

☒ us-east-1a (use1-az6)

Subnet

subnet-07f9d8baef520c38c

Sub1

IPv4 settings

Assigned by AWS

☒ us-east-1b (use1-az1)

Subnet

Feedback

Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Assigned by AWS

### Security groups [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer.

Security groups

Select up to 5 security groups

Create new security group [↗](#)

ALB-SG sg-0d5ab93de04825d49 ✕

VPC: vpc-058de8a262dac1e83

### Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

Remove

Feedback Looking for language selection? Find it in the new Unified Settings [↗](#) © 2022, Amazon Web Services, Inc. or its affiliates. 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

▼ **Add-on services - optional**

Additional AWS services can be integrated with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.

AWS Global Accelerator [Info](#)

☒ Create an accelerator to get static IP addresses and improve the performance and availability of your applications. [Additional charges apply ↗](#)

Accelerator name

Enter a name for your accelerator. Once created, you can use the Global Accelerator console to manage this accelerator.

accelerator

The accelerator name can have up to 64 characters. Valid characters: a-z, A-Z, 0-9, . and - (hyphen).

► **Tags - optional**

Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

### Summary

Review and confirm your configurations. [Estimate cost ↗](#)

Basic configuration [Edit](#) Security groups [Edit](#) Network mapping [Edit](#) Listeners and routing [Edit](#)

Feedback Looking for language selection? Find it in the new Unified Settings [↗](#) © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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
```

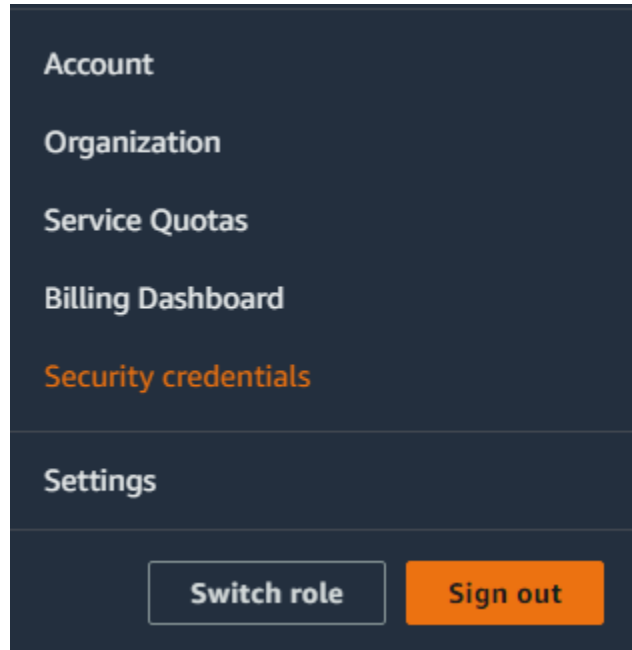
- c) After installation run the following commands:

```
aws configure
```

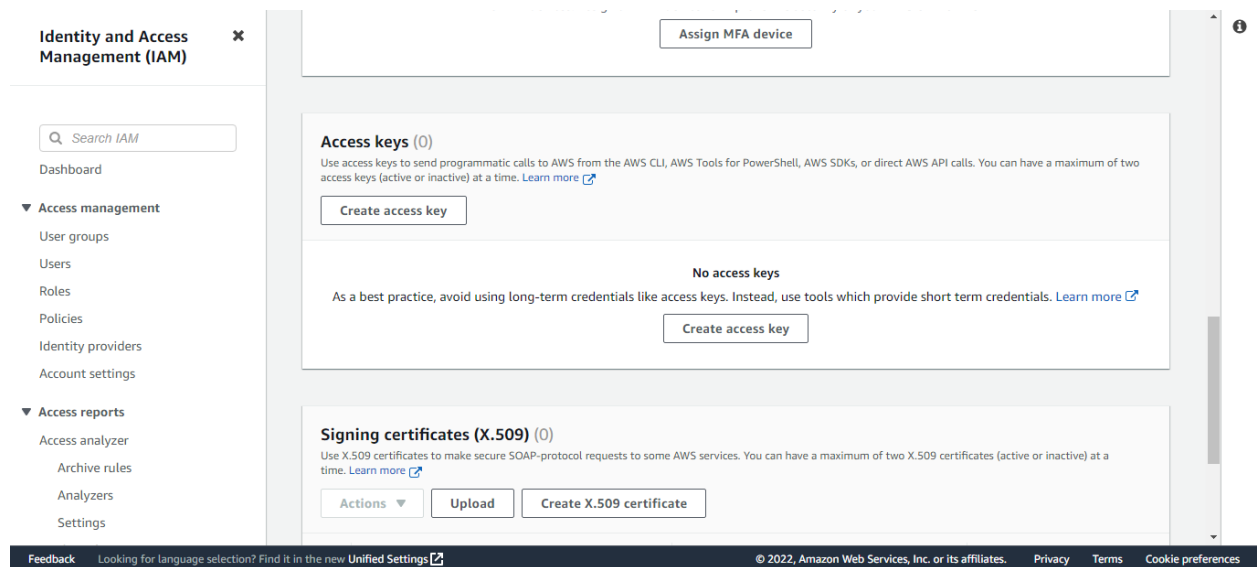
- d) This will prompt you to enter **AWS ACCESS KEY** and **AWS SECRET ACCESS KEY**. Generate these from your account by following the steps below:

## GENERATING AWS ACCESS & SECRET ACCESS KEY:

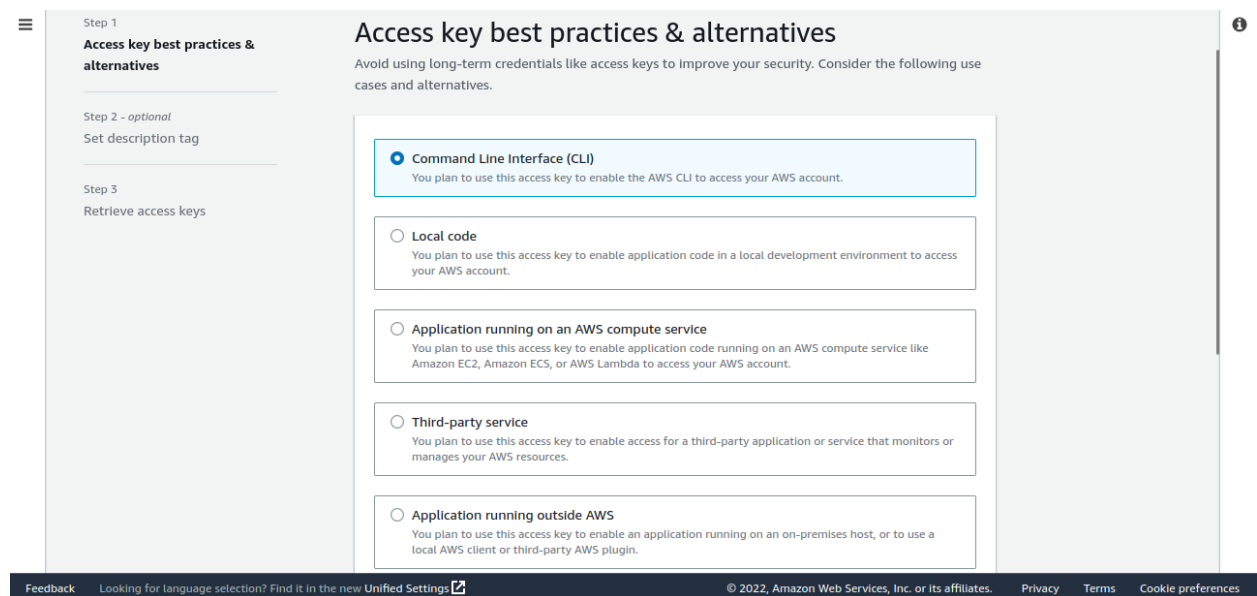
- Go into Security Credentials by clicking on your account name



- Scroll Down and select **Create Key**



- Choose the Command Line Interface option



- Optionally tag the key and create the access key

The screenshot shows the 'Set description tag - optional' step in the AWS IAM 'Create access key' wizard. The left sidebar shows the progress: Step 1 (Access key best practices & alternatives), Step 2 - optional (Set description tag), and Step 3 (Retrieve access keys). The main content area has the title 'Set description tag - optional' and a subtitle 'The description for this access key will be attached to this user as a tag and shown alongside the access key.' Below this is a text input field for the 'Description tag value' with a placeholder 'Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.' A note below the field states: 'Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ : / \* + - @'. At the bottom right are three buttons: 'Cancel', 'Previous', and 'Create access key'.

The screenshot shows the 'Retrieve access keys' step in the AWS IAM 'Create access key' wizard. A green banner at the top states: 'Access key created. This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' The left sidebar shows the progress: Step 1 (Access key best practices & alternatives), Step 2 - optional (Set description tag), and Step 3 (Retrieve access keys). The main content area has the title 'Retrieve access keys'. Below this is a section titled 'Access key' with a subtitle 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' Below this is a table with two columns: 'Access key' and 'Secret access key'. The 'Access key' column contains the value 'AKIAVMERSI2BRIHDQP5K'. The 'Secret access key' column contains a masked value '\*\*\*\*\*' with a 'Show' link. Below the table is a section titled 'Access key best practices' with a list of three items: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', and 'Enable least-privilege permissions.' At the bottom are three buttons: 'Cancel', 'Previous', and 'Create 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.

```
ubuntu@ip-192-188-4-156:~$ aws configure
AWS Access Key ID [None]: AKIAVMERSI2BRIHDQPSK
AWS Secret Access Key [None]: Bd6QH31QfJNsiKXiYQldPHTXpm2JHgw86ChS50I6
Default region name [None]: us-east-1
Default output format [None]:
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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**

```
- name: Starting Provisioning
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: 'AKIAVMERSI2BRIHDQPSK'
    aws_secret_key: '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'
    key_name: "{{ keypair }}"
    group: launch-wizard-1
    instance_type: "{{ instance_type }}"
    image: "{{ ami }}"
    wait: true
    region: "{{ region }}"
    vpc_subnet_id: subnet-07f9d8baef520c38c
    count: 1

:wg
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

Feedback

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

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

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

```
ubuntu@ip-192-188-4-156:~$ vi launcher.yaml
ubuntu@ip-192-188-4-156:~$ ansible-playbook launcher.yaml
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Starting Provisioning] *****
TASK [Create an ec2 instance] *****
changed: [localhost]

PLAY RECAP *****
localhost      : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#) © 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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***

```
ubuntu@ip-192-188-4-156:~$ sudo snap install docker
docker 20.10.17 from Canonical/ installed
ubuntu@ip-192-188-4-156:~$ sudo snap install microk8s --classic
microk8s (1.25/stable) v1.25.4 from Canonical/ installed
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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***

```
ubuntu@ip-192-188-4-156:~$ sudo microk8s start
ubuntu@ip-192-188-4-156:~$ sudo microk8s status
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    ha-cluster      # (core) Configure high availability on the current node
    helm            # (core) Helm - the package manager for Kubernetes
    helm3           # (core) Helm 3 - the package manager for Kubernetes
  disabled:
    cert-manager    # (core) Cloud native certificate management
    community       # (core) The community addons repository
    dashboard       # (core) The Kubernetes dashboard
    dns             # (core) CoreDNS
    gpu             # (core) Automatic enablement of Nvidia CUDA
    host-access     # (core) Allow Pods connecting to Host services smoothly
    hostpath-storage # (core) Storage class; allocates storage from host directory
    ingress         # (core) Ingress controller for external access
    kube-ovn        # (core) An advanced network fabric for Kubernetes
    mayastor        # (core) OpenEBS MayaStor
    metallb         # (core) Loadbalancer for your Kubernetes cluster
    metrics-server  # (core) K8s Metrics Server for API access to service metrics
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

Feedback

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

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- 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***

```
ubuntu@ip-192-188-4-156:~$ sudo microk8s add-node
From the node you wish to join to this cluster, run the following:
microk8s join 192.188.4.156:25000/db7a1d89d8f2257d7ed22d8bb1f8ad08/eedf87ecdcc5

Use the '--worker' flag to join a node as a worker not running the control plane, eg:
microk8s join 192.188.4.156:25000/db7a1d89d8f2257d7ed22d8bb1f8ad08/eedf87ecdcc5 --worker

If the node you are adding is not reachable through the default interface you can use one of the following:
microk8s join 192.188.4.156:25000/db7a1d89d8f2257d7ed22d8bb1f8ad08/eedf87ecdcc5
microk8s join 172.17.0.1:25000/db7a1d89d8f2257d7ed22d8bb1f8ad08/eedf87ecdcc5
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

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

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- 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

```
ubuntu@ip-192-188-4-96:~$ sudo microk8s join 192.188.4.156:25000/c9d15ed8fd1530bc4b13623ec3a5b85e/eedf87ecdcc5 --worker
Contacting cluster at 192.188.4.156

The node has joined the cluster and will appear in the nodes list in a few seconds.

This worker node gets automatically configured with the API server endpoints.
If the API servers are behind a loadbalancer please set the '--refresh-interval' to '0s' in:
/var/snap/microk8s/current/args/apiserver-proxy
and replace the API server endpoints with the one provided by the loadbalancer in:
/var/snap/microk8s/current/args/traefik/provider.yaml
ubuntu@ip-192-188-4-96:~$
```

i-06ec000d02feb4507

PublicIPs: 54.87.254.144 PrivateIPs: 192.188.4.96

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

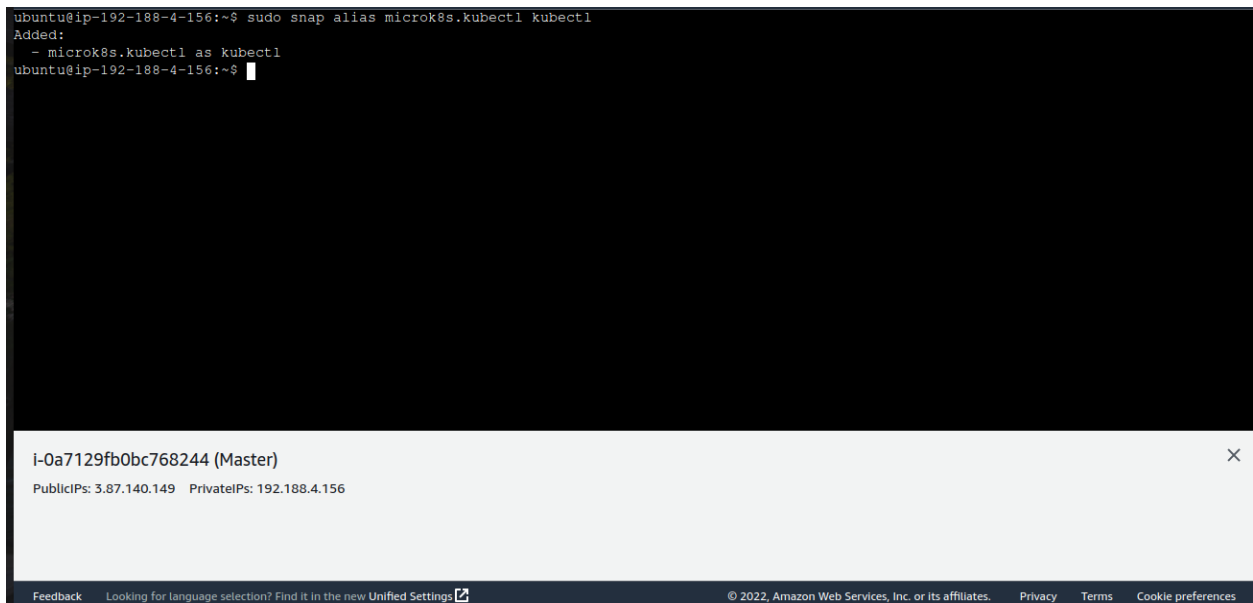
© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 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>***

```
ubuntu@ip-192-188-4-156:~$ sudo snap alias microk8s.kubectl kubectl
Added:
- microk8s.kubectl as kubectl
ubuntu@ip-192-188-4-156:~$
```



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

```
ubuntu@ip-192-188-4-156:~$ sudo kubectl apply -f NetworkPolicy.yaml
networkpolicy.networking.k8s.io/network-policy created
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

[Privacy](#)

[Terms](#)

[Cookie preferences](#)

- *Describe the policy with the following command:*

***sudo kubectl describe NetworkPolicy/<NAME\_OF\_POLICY>***

This command gives the following output:

```
ubuntu@ip-192-188-4-156:~$ sudo kubectl describe NetworkPolicy/network-policy
Name:          network-policy
Namespace:     default
Created on:    2022-12-16 07:24:37 +0000 UTC
Labels:        <none>
Annotations:   <none>
Spec:
  PodSelector:  app=db
  Allowing ingress traffic:
    To Port: 6379/TCP
    From:
      NamespaceSelector: name=Mynamespace
      From:
        PodSelector: app=frontend
  Not affecting egress traffic
  Policy Types: Ingress
ubuntu@ip-192-188-4-156:~$
```

i-Oa7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

[Privacy](#)

[Terms](#)

[Cookie preferences](#)

## 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"]***

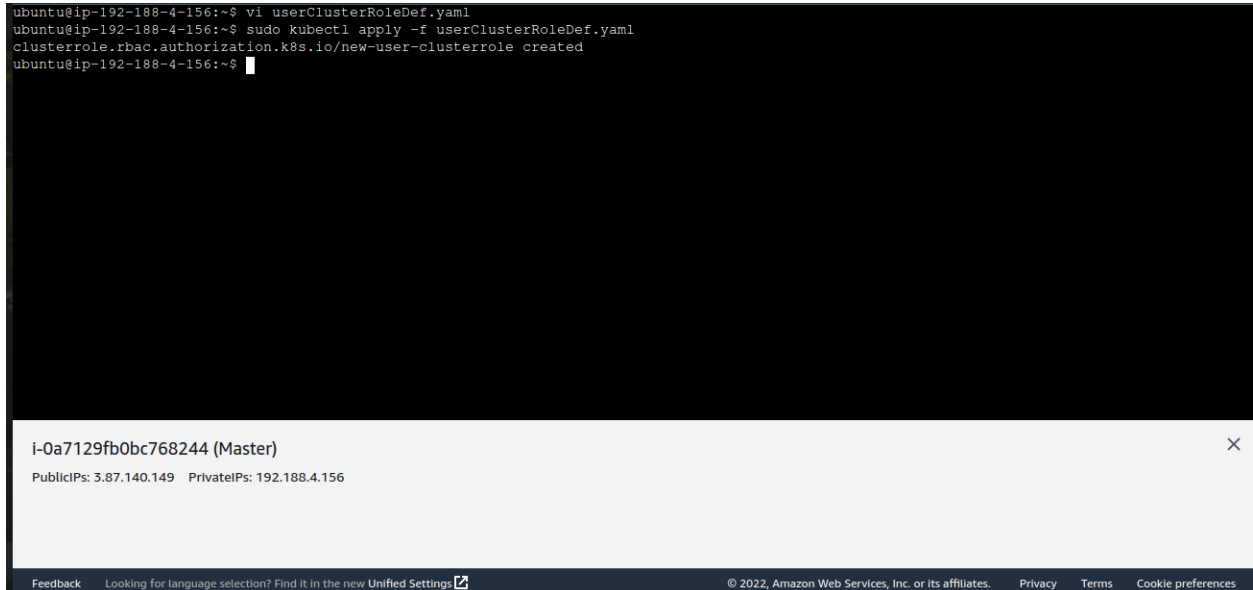
- After adding the lines to the file, save and quit from vi editor. Apply the file with the following command:

***sudo kubectl apply -f <FILE\_NAME>.yaml***



The command gives the following output:

```
ubuntu@ip-192-188-4-156:~$ vi userClusterRoleDef.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl apply -f userClusterRoleDef.yaml
clusterrole.rbac.authorization.k8s.io/new-user-clusterrole created
ubuntu@ip-192-188-4-156:~$
```



- 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

```
ubuntu@ip-192-188-4-156:~$ vi userClusterRoleBinding.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl apply -f userClusterRoleBinding.yaml
clusterrolebinding.rbac.authorization.k8s.io/new-user-clusterrolebinding created
ubuntu@ip-192-188-4-156:~$
```

I-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#) [Looking for language selection? Find it in the new Unified Settings](#) © 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## 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:

```
ubuntu@ip-192-188-4-156:~$ vi secrets.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace apply -f configMap.yaml
configmap/pod-configmap created
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace apply -f secrets.yaml
secret/pod-secret created
ubuntu@ip-192-188-4-156:~$ vi PV.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl apply -f PV.yaml
persistentvolume/persistent-volume created
ubuntu@ip-192-188-4-156:~$ vi PVC.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace apply -f PVC.yaml
persistentvolumeclaim/pv-claim created
ubuntu@ip-192-188-4-156:~$ vi Svrvice.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace apply -f Svrvice.yaml
service/mydb created
ubuntu@ip-192-188-4-156:~$ vi deployment.yaml
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace apply -f deployment.yaml
deployment.apps/myapps created
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#)
[Looking for language selection? Find it in the new Unified Settings](#)
© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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:

```
ReadOnly: false
kube-api-access-4tb65:
  Type:                      Projected (a volume that contains injected data from multiple sources)
  TokenExpirationSeconds:    3607
  ConfigMapName:             kube-root-ca.crt
  ConfigMapOptional:         <nil>
  DownwardAPI:               true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From              Message
  ----     -
Normal    Scheduled   34s   default-scheduler Successfully assigned newnamespace/myapps-7885bd97dd-7wvfg to ip-192-188-4-156
Normal    Pulling     33s   kubelet           Pulling image "ggnika007/mysql-dnsutils"
Normal    Pulled      33s   kubelet           Successfully pulled image "ggnika007/mysql-dnsutils" in 116.004566ms
Normal    Created     33s   kubelet           Created container init-mysql
Normal    Started     33s   kubelet           Started container init-mysql
Normal    Pulling     27s   kubelet           Pulling image "wordpress"
Normal    Pulled      27s   kubelet           Successfully pulled image "wordpress" in 118.928516ms
Normal    Created     27s   kubelet           Created container wordpress-container
Normal    Started     27s   kubelet           Started container wordpress-container
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

[Feedback](#)
[Looking for language selection? Find it in the new Unified Settings](#)
© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

## 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:

```
ubuntu@ip-192-188-4-156:~$ sudo kubectl -n newnamespace get hpa
NAME      REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
myapps    Deployment/myapps   0%/50%   1         10        1         6m44s
ubuntu@ip-192-188-4-156:~$
```

i-0a7129fb0bc768244 (Master)

PublicIPs: 3.87.140.149 PrivateIPs: 192.188.4.156

Feedback Looking for language selection? Find it in the new Unified Settings © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**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

- EasyPaisha 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.
- EasyPaisha gives its golden users an '**EasyPaisha Card**' which they can use like a normal credit card anywhere.
- Apart from utility bills, EasyPaisha 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 EasyPaisha app.
- You can pay for your leisure club fee from EasyPaisha as well.
- EasyPaisha 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 EasyPaisha.
- You can Donate money to registered companies.
- EasyPaisha also gives loans to needy users.