30 May 2021 06:12 PM

Infra Optimization.

Project 1

DESCRIPTION

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

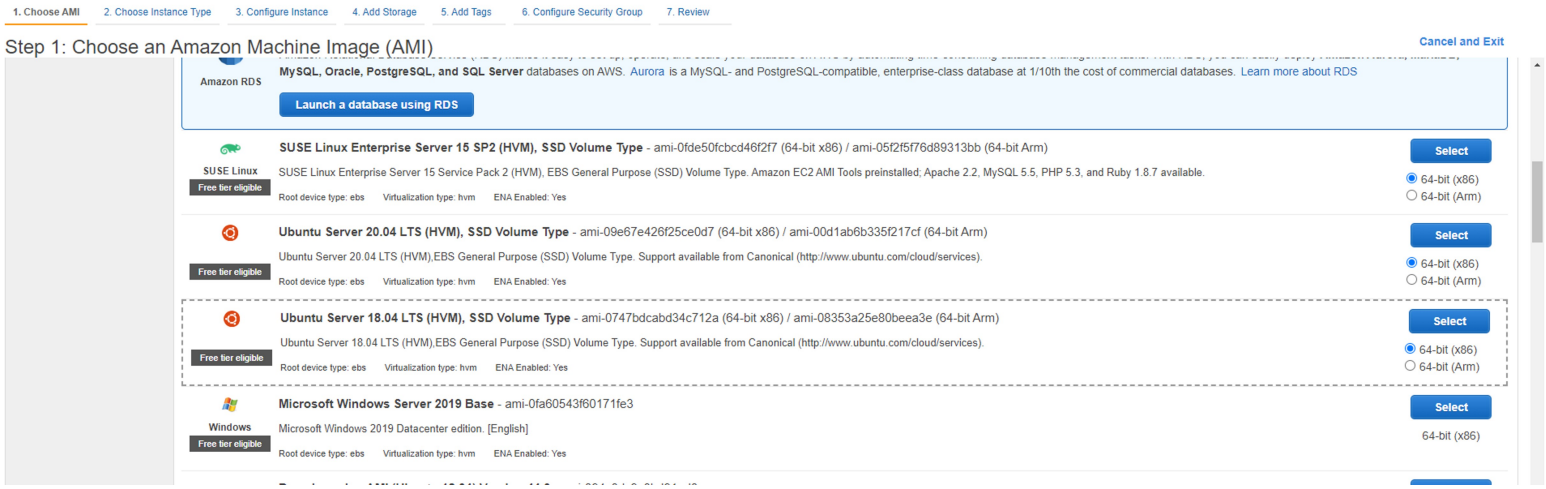
The following things to be kept in check:

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:

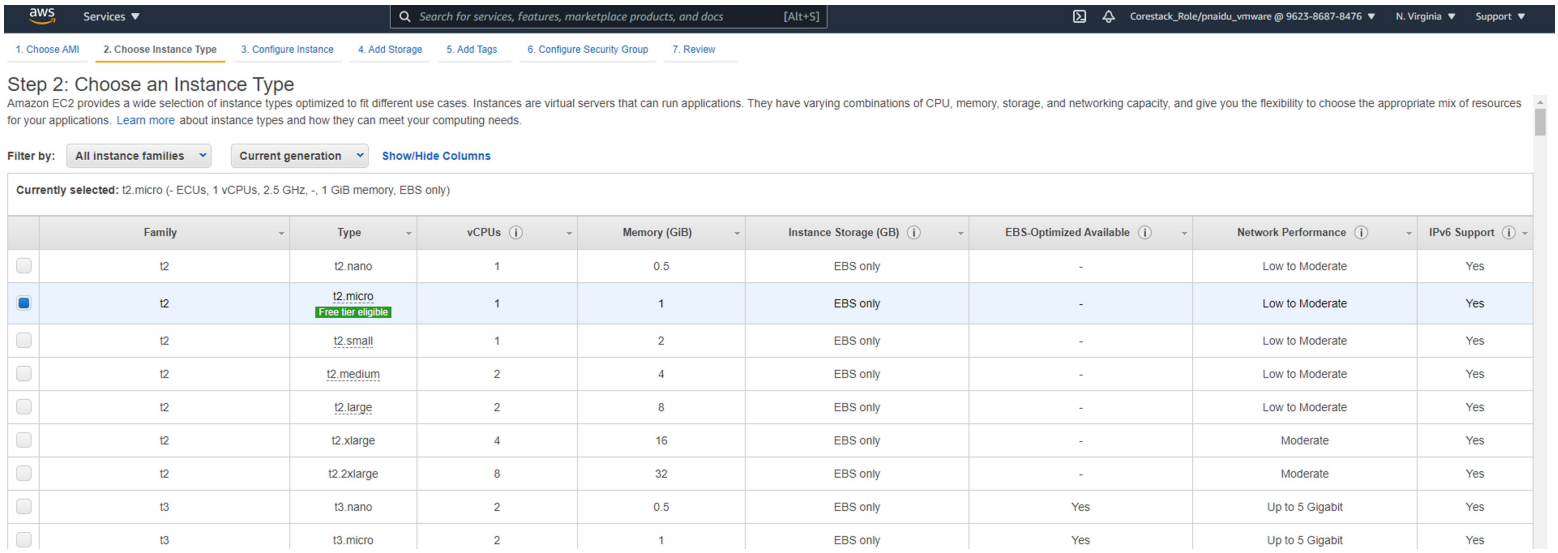
- Project and tester details
- Concepts used in the project
- Links to the GitHub repository to verify the project completion
- Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)

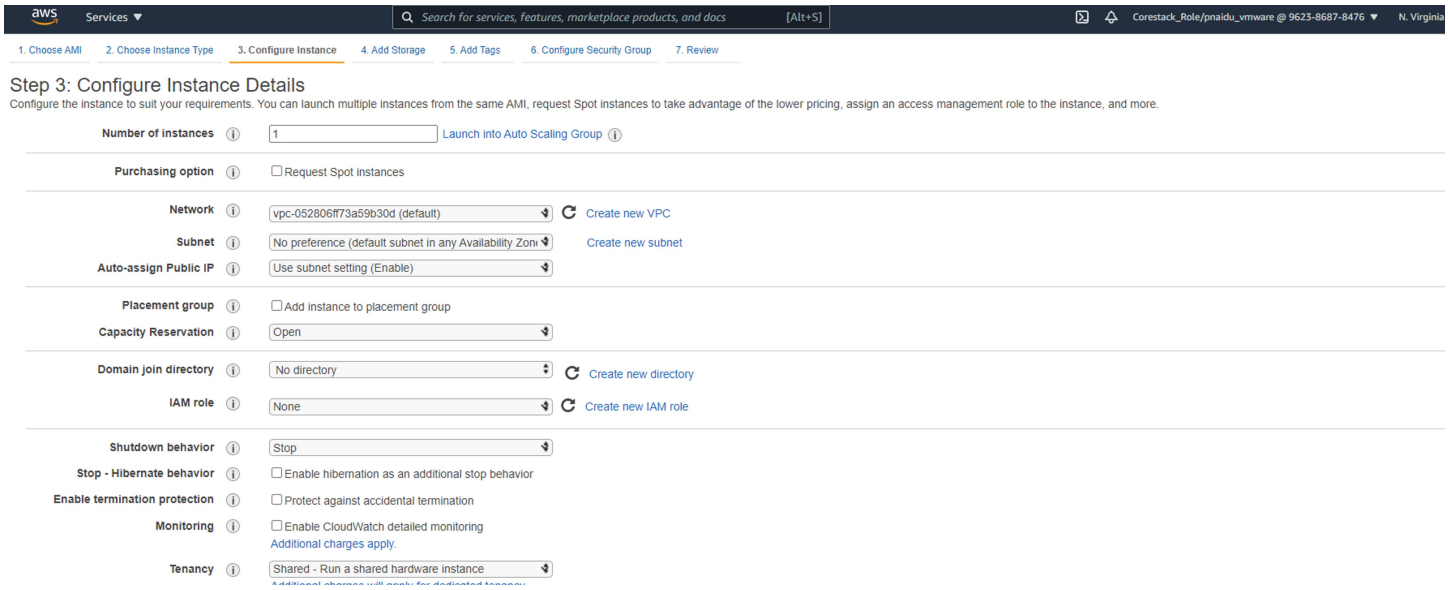
Select EC2 instance once you login to AWS Web

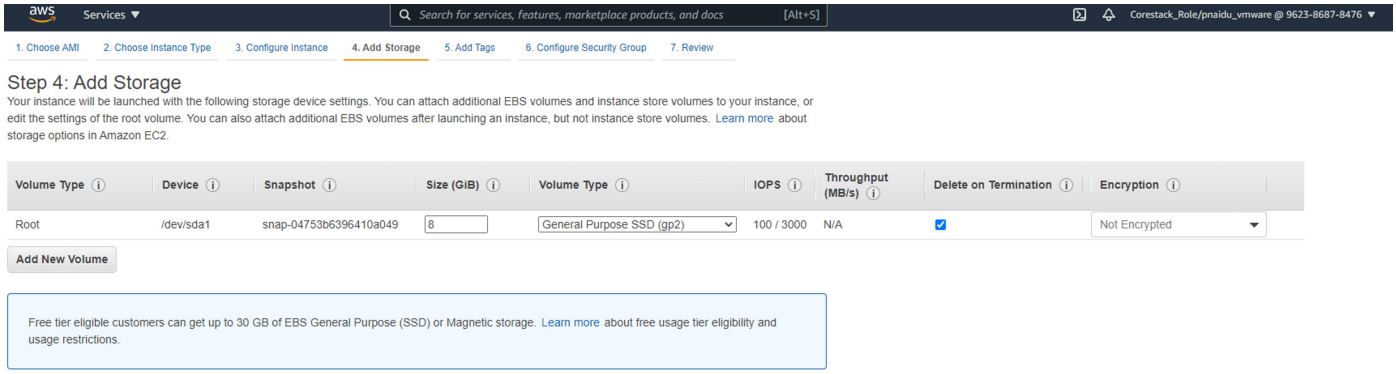


Initially selected instance type as t2.micro but later changed to t2.medium post deployment for all the three nodes to be compatible with Kubernetes

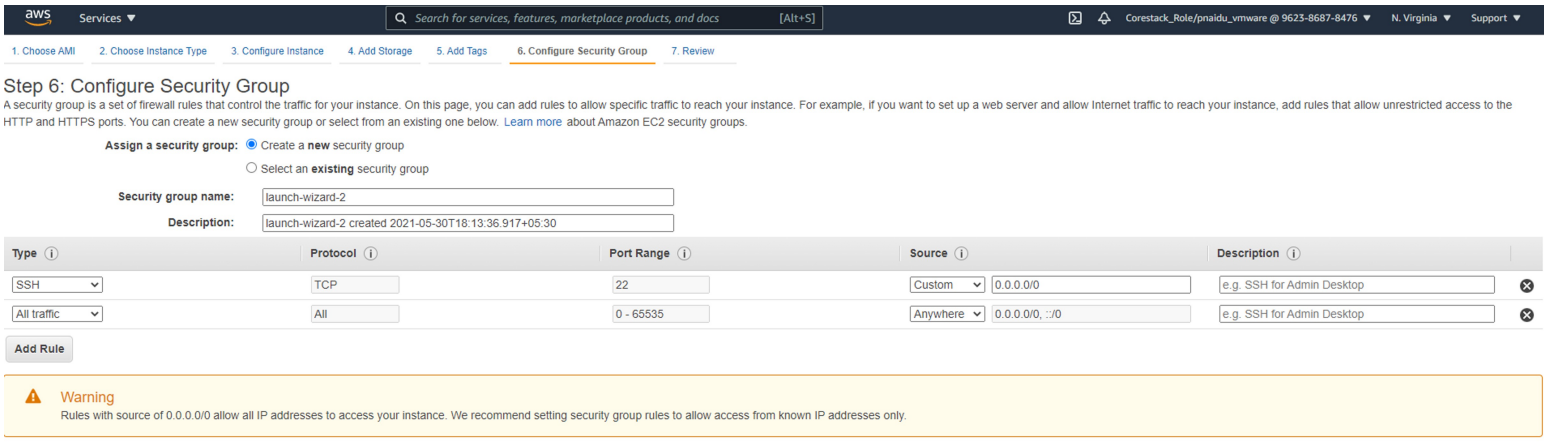


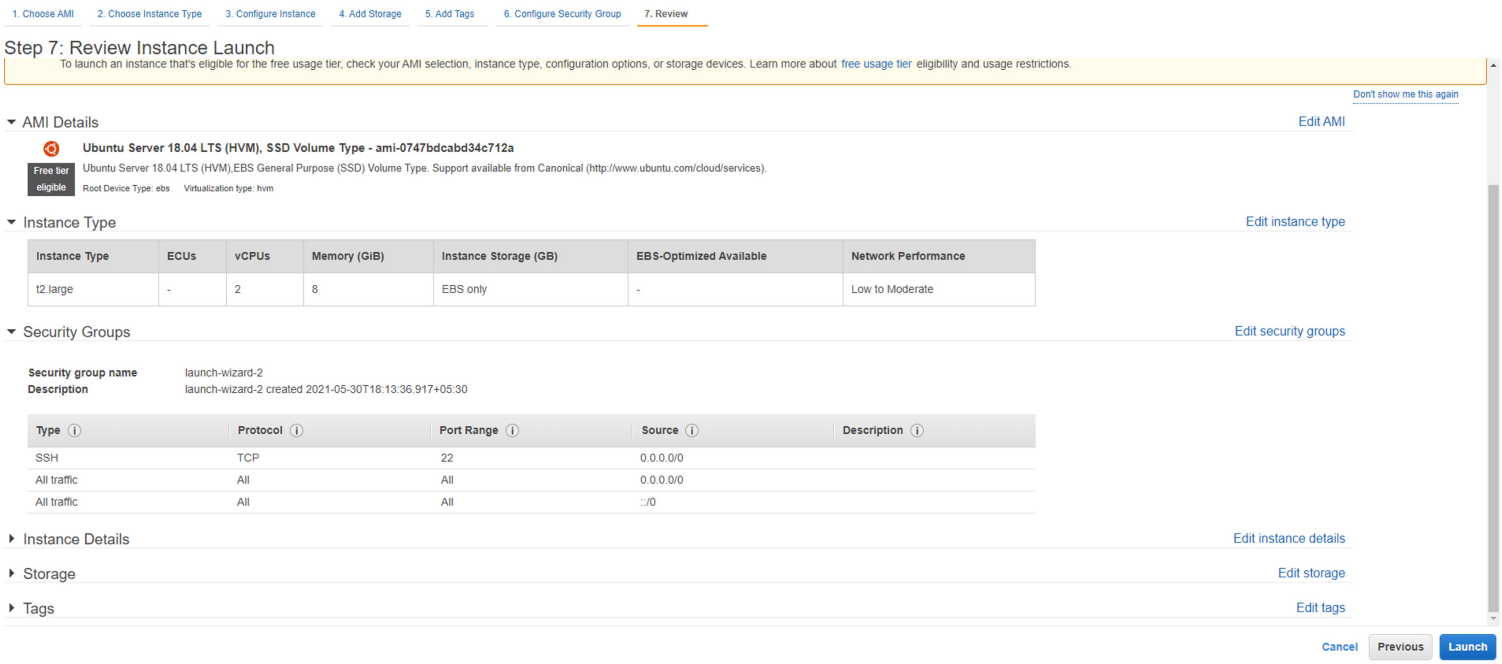
Selected Default VPC and Auto Assign Public IP





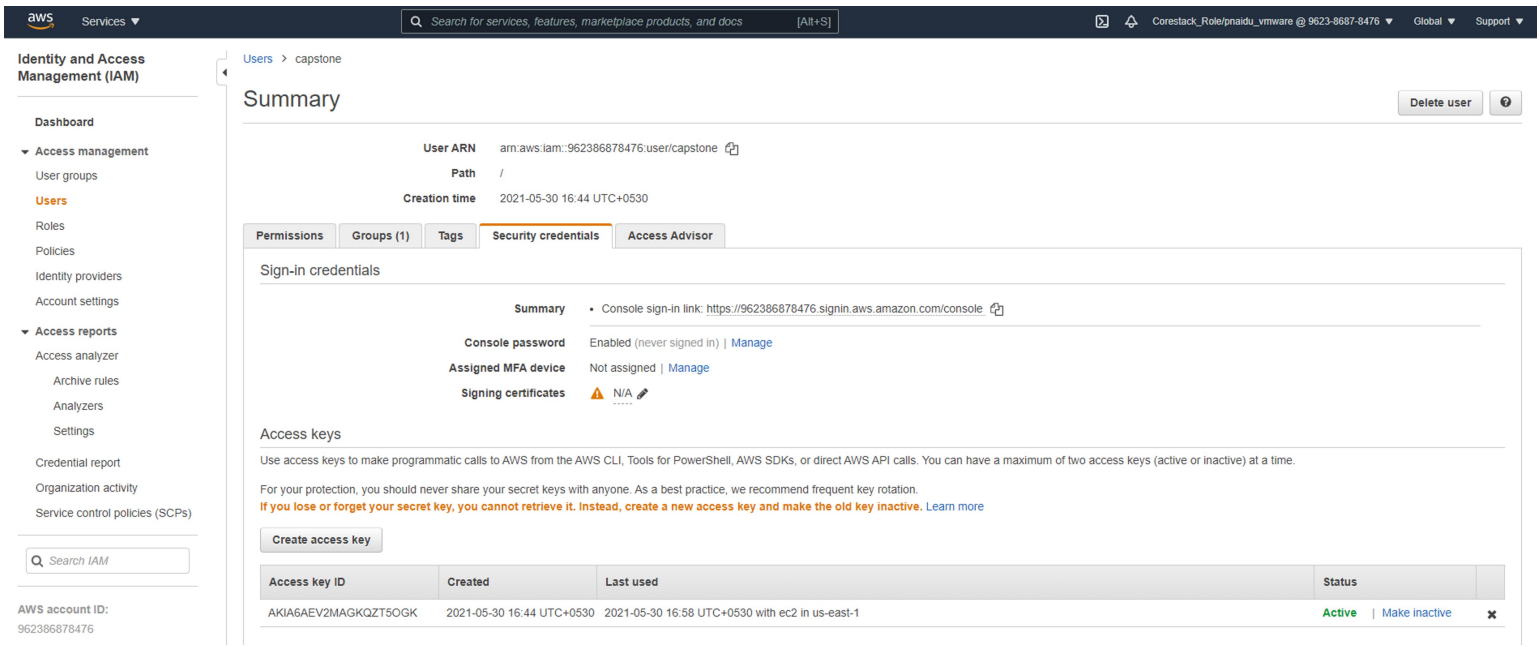
Configure Security Group for all traffic for LAB purpose only





2. Create the Access key and the secret key for the root user or IAM user for the AWS CLI configuration

3.



4. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet

Created a PEM for SSH and got the Private Key to be used for SSH from local workstation

Used the below mentioned commands to install Ansible on all the EC2 instances

sudo apt-get install -f

sudo apt-get install software-properties-common

sudo apt-add-repository ppa:ansible/ansible

sudo apt-get update

sudo apt-get install ansible

```
ubuntu@ip-172-31-54-80:~$ ansible --version
ansible 2.9.22
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ubuntu/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Feb 27 2021, 15:10:58) [GCC 7.5.0]
```

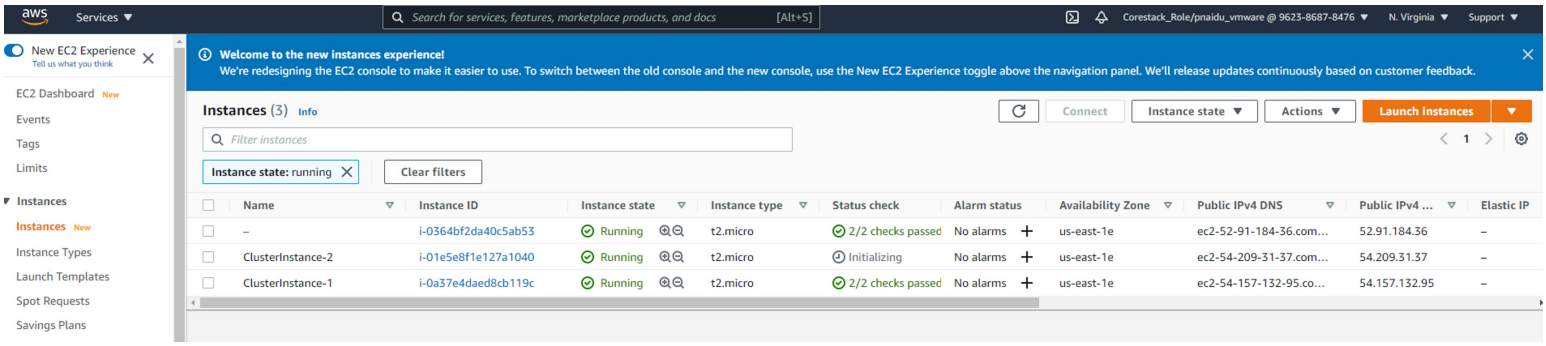
Post the installation of Anisble on the master EC2 node created a playbook for ansible to automate the EC2 instance deployment using the below mentioned playbook yaml

```
- hosts: localhost
  tasks:
      - ec2:
            aws_access_key: AKIA6AEV2MAGKQZT5OGK
            aws_secret_key: 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
            key_name: capstone
```

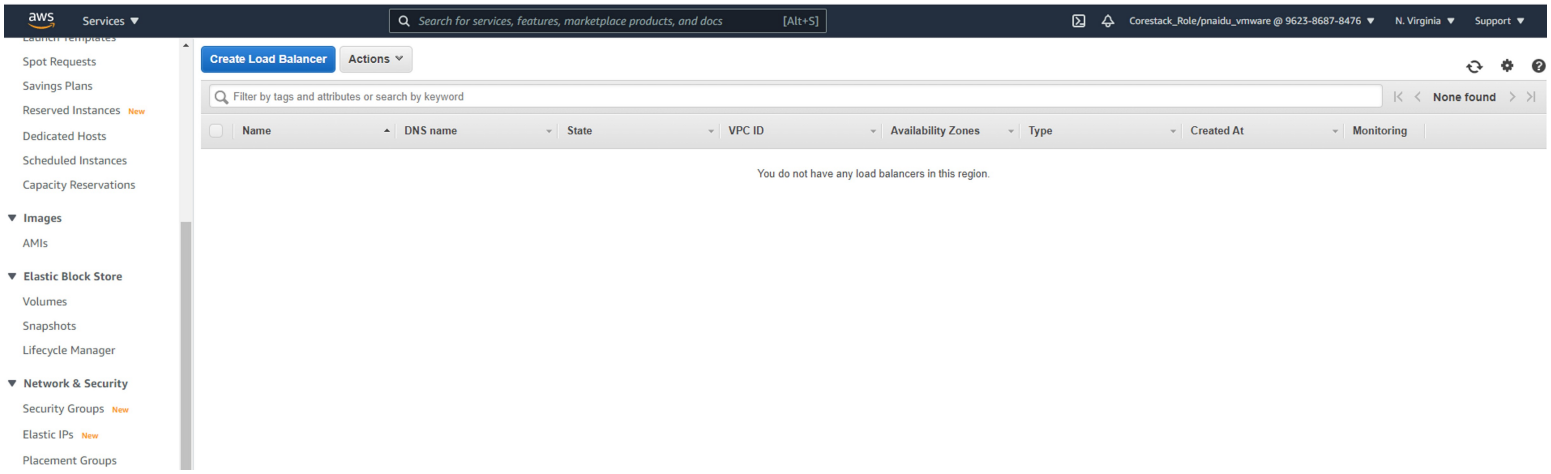
```
group: sg-0ebc95f6e95b48772
instance_type: t2.micro
image: ami-0747bdcabd34c712a
wait: yes
wait_timeout: 500
count: 1
instance_tags:
    Name: ClusterInstance-1
monitoring: yes
region: us-east-1
vpc_subnet_id: subnet-0cdbd2a7867fcb2ae
assign_public_ip: yes
```

```
PLAY [localhost] ***************************************************************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************************************************
ok: [localhost]

TASK [ec2] *********************************************************************************************************************************************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@ip-172-31-54-80:~$
```

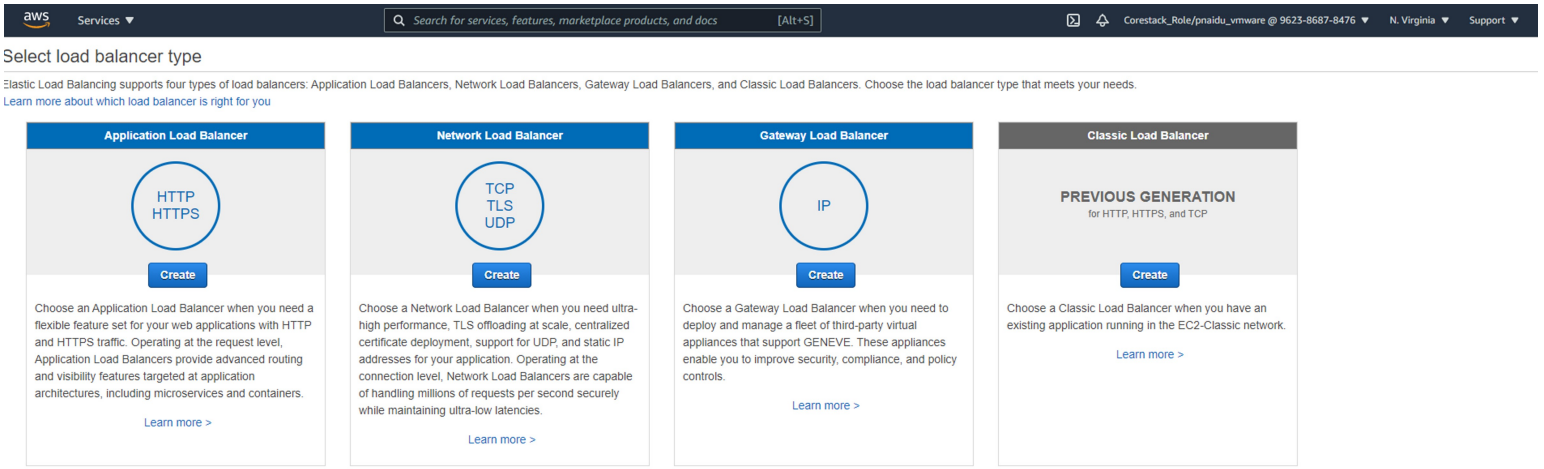
Playbook execution completed successfully for 2 more EC2 instances with Name ClusterInstance-1 and ClusterInstance-2 respectively

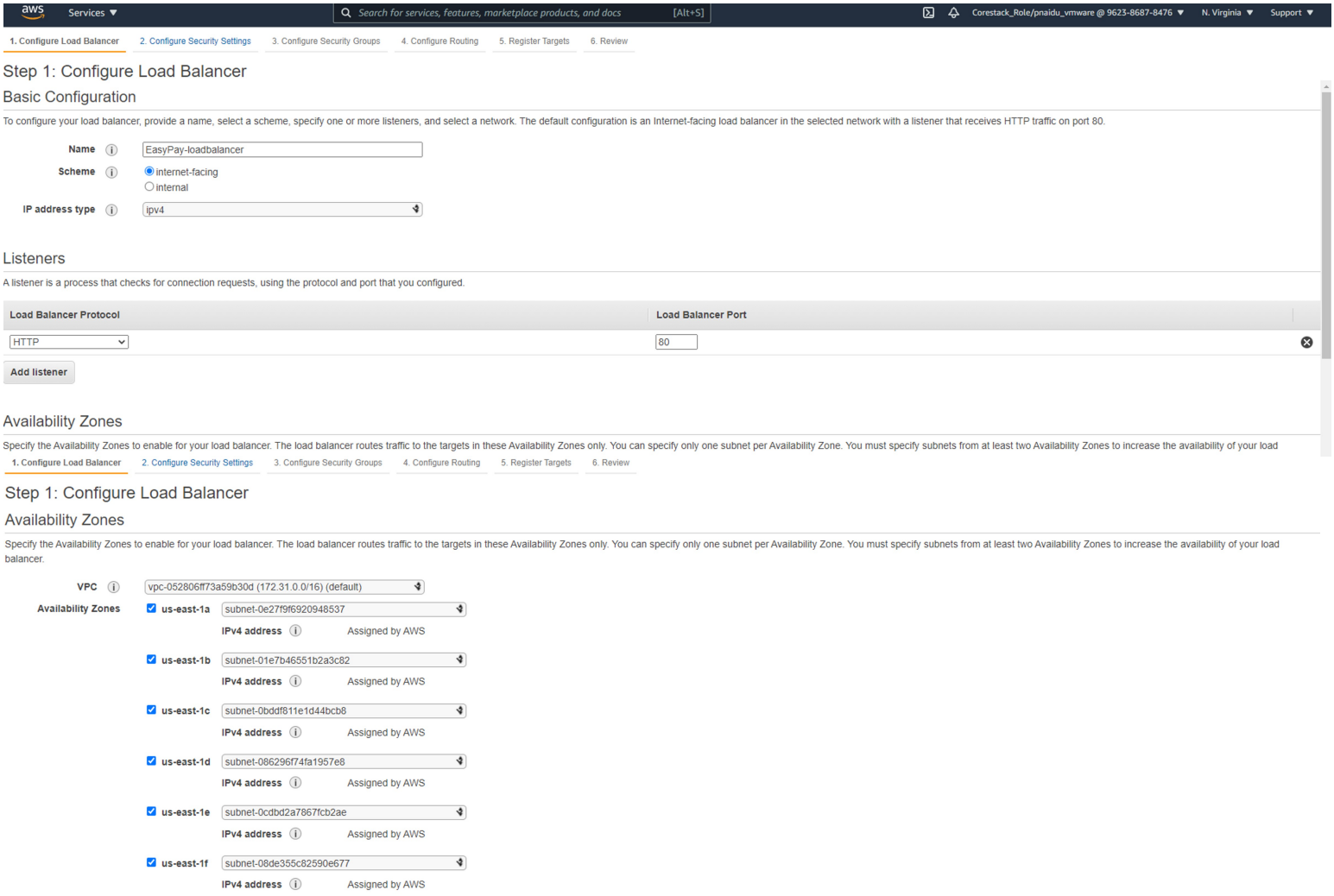


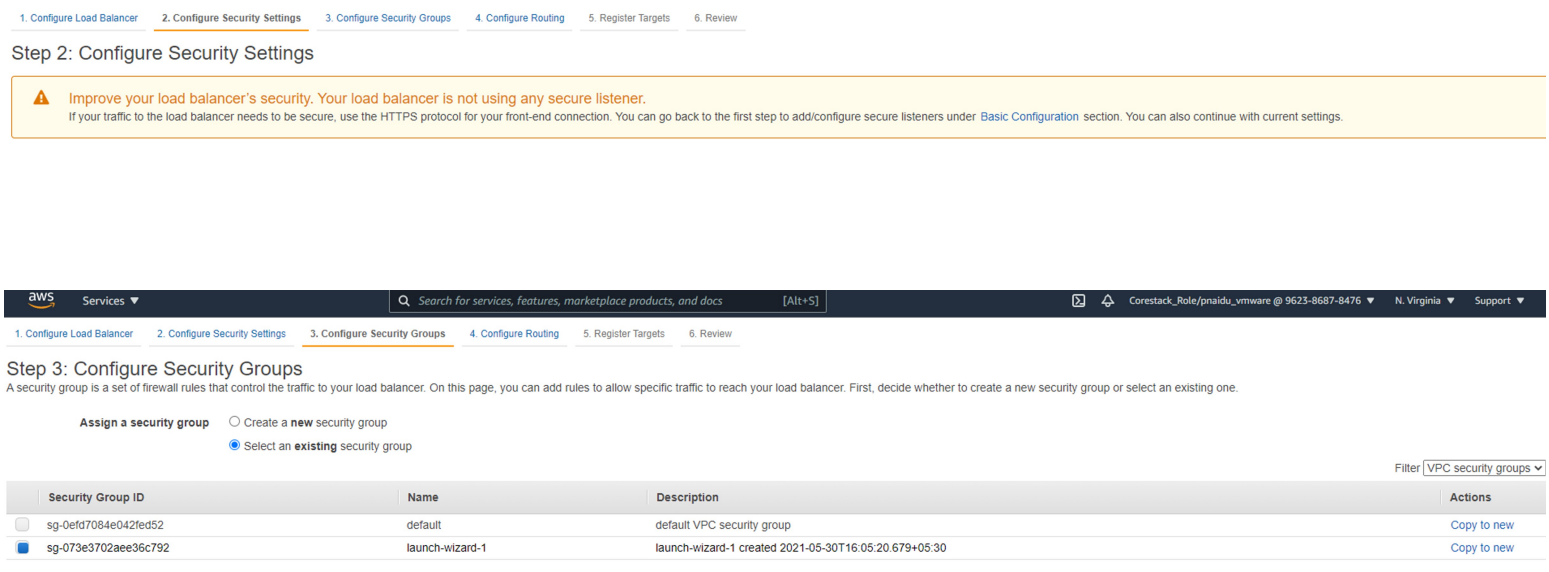
Steps to create the Load Balancer and Taget Group and Attach the instance in the Target Group



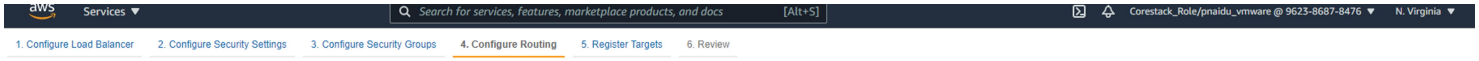
Select HTTP/HTTPS LB




Services
Search for services, features, marketplace products, and docs [Alt+S]
Corestack_Role/pnaidu_vmware @ 9623-8687-8476
N. Virginia
Support
1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review
Step 1: Configure Load Balancer
Basic Configuration
To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network. The default configuration is an Internet-facing load balancer in the selected network with a listener that receives HTTP traffic on port 80.
Name
EasyPay-loadbalancer
Scheme
internet-facing
internal
IP address type
ipv4
Listeners
A listener is a process that checks for connection requests, using the protocol and port that you configured.
Load Balancer Protocol
Load Balancer Port
HTTP
80
Add listener
Availability Zones
Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load
1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review
Step 1: Configure Load Balancer
Availability Zones
Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. You can specify only one subnet per Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.
VPC
vpc-052806ff73a59b30d (172.31.0.0/16) (default)
Availability Zones
us-east-1a
subnet-0e27f9f6920948537
IPv4 address
Assigned by AWS
us-east-1b
subnet-01e7b46551b2a3c82
IPv4 address
Assigned by AWS
us-east-1c
subnet-0bddf811e1d44bcb8
IPv4 address
Assigned by AWS
us-east-1d
subnet-086296f74fa1957e8
IPv4 address
Assigned by AWS
us-east-1e
subnet-0cdbd2a7867fcb2ae
IPv4 address
Assigned by AWS
us-east-1f
subnet-08de355c82590e677
IPv4 address
Assigned by AWS
Add-on services
Additional AWS services can be integrated with this load balancer at launch when you enable them below. 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.


1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review
Step 2: Configure Security Settings
Improve your load balancer's security. Your load balancer is not using any secure listener.
If your traffic to the load balancer needs to be secure, use the HTTPS protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under Basic Configuration section. You can also continue with current settings.

Services
Search for services, features, marketplace products, and docs [Alt+S]
Corestack_Role/pnaidu_vmware @ 9623-8687-8476
N. Virginia
Support
1. Configure Load Balancer
2. Configure Security Settings
3. Configure Security Groups
4. Configure Routing
5. Register Targets
6. Review
Step 3: Configure Security Groups
A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.
Assign a security group
Create a new security group
Select an existing security group
Filter
VPC security groups
Security Group ID
Name
Description
Actions
sg-0efd7084e042fed52
default
default VPC security group
Copy to new
sg-073e3702aee36c792
launch-wizard-1
launch-wizard-1 created 2021-05-30T16:05:20.679+05:30
Copy to new



1. Configure Load Balancer | 2. Configure Security Settings | 3. Configure Security Groups | **4. Configure Routing** | 5. Register Targets | 6. Review

Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port that you specify here. It also performs health checks on the targets using these settings. The target group you specify in this step will apply to all of the listeners configured on this load balancer. You add listeners after the load balancer is created.

Target group

Target group (i): New target group

Name (i): EasyPay-Target

Target type:
- (•) Instance
- ( ) IP
- ( ) Lambda function

Protocol (i): HTTP

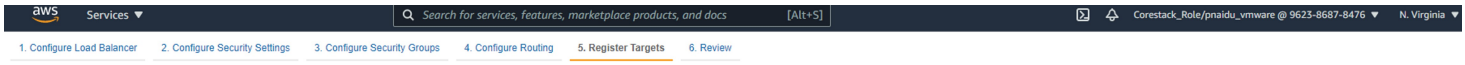
Port (i): 80

Protocol version (i):
- (•) 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

Protocol (i): HTTP

Path (i): /



1. Configure Load Balancer | 2. Configure Security Settings | 3. Configure Security Groups | 4. Configure Routing | **5. Register Targets** | 6. Review

Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Remove

Instance	Name	Port	State	Security groups	Zone
No instances available.					

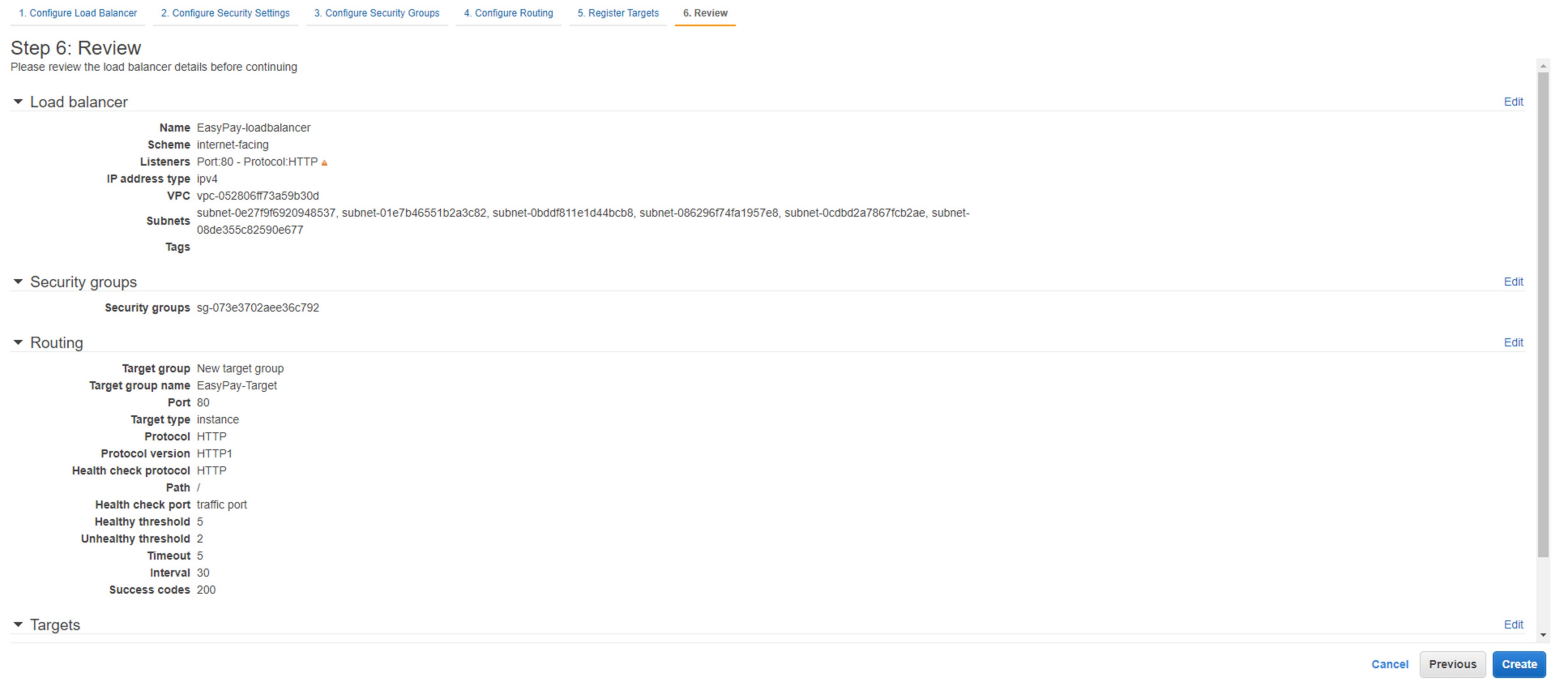
Instances

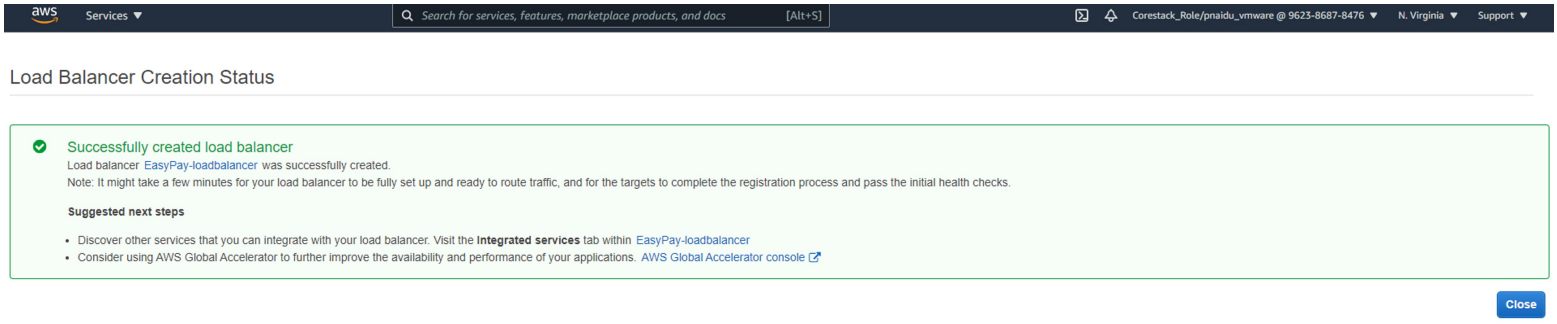
To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered on port 80

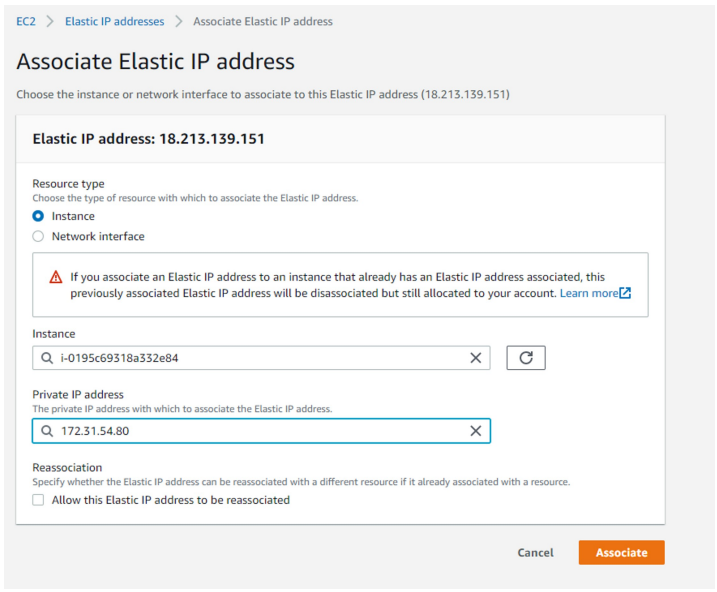
Search Instances

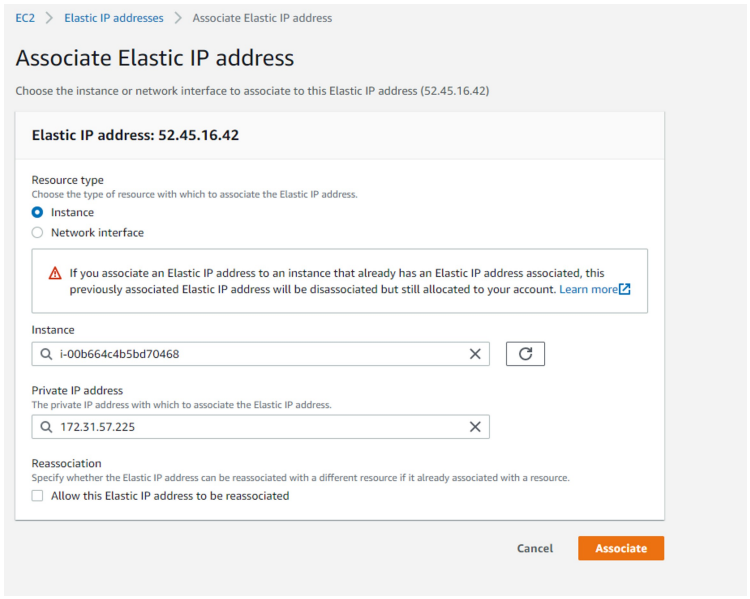
Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
i-0364bf2da40c5ab53		running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fcb2ae	172.31.48.0/20
i-01e5e8f1e127a1040	ClusterInstance-2	running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fcb2ae	172.31.48.0/20
i-0a37e4daed8cb119c	ClusterInstance-1	running	launch-wizard-1	us-east-1e	subnet-0cdbd2a7867fcb2ae	172.31.48.0/20

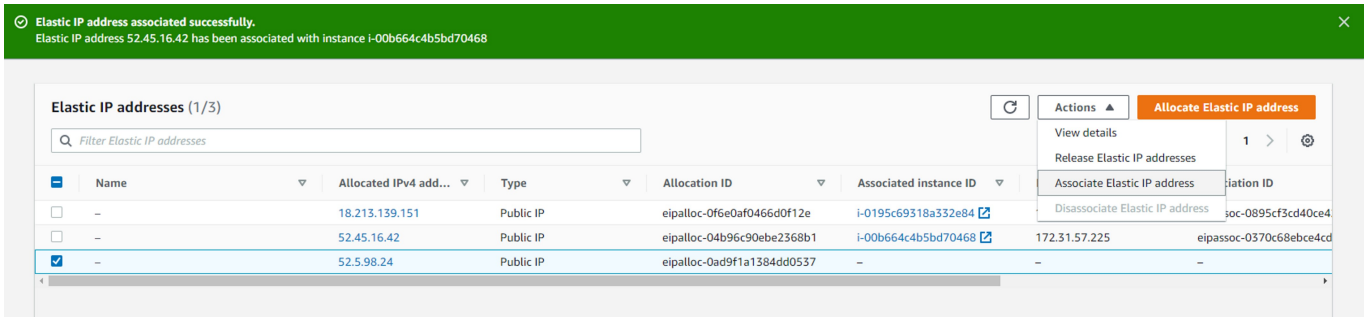


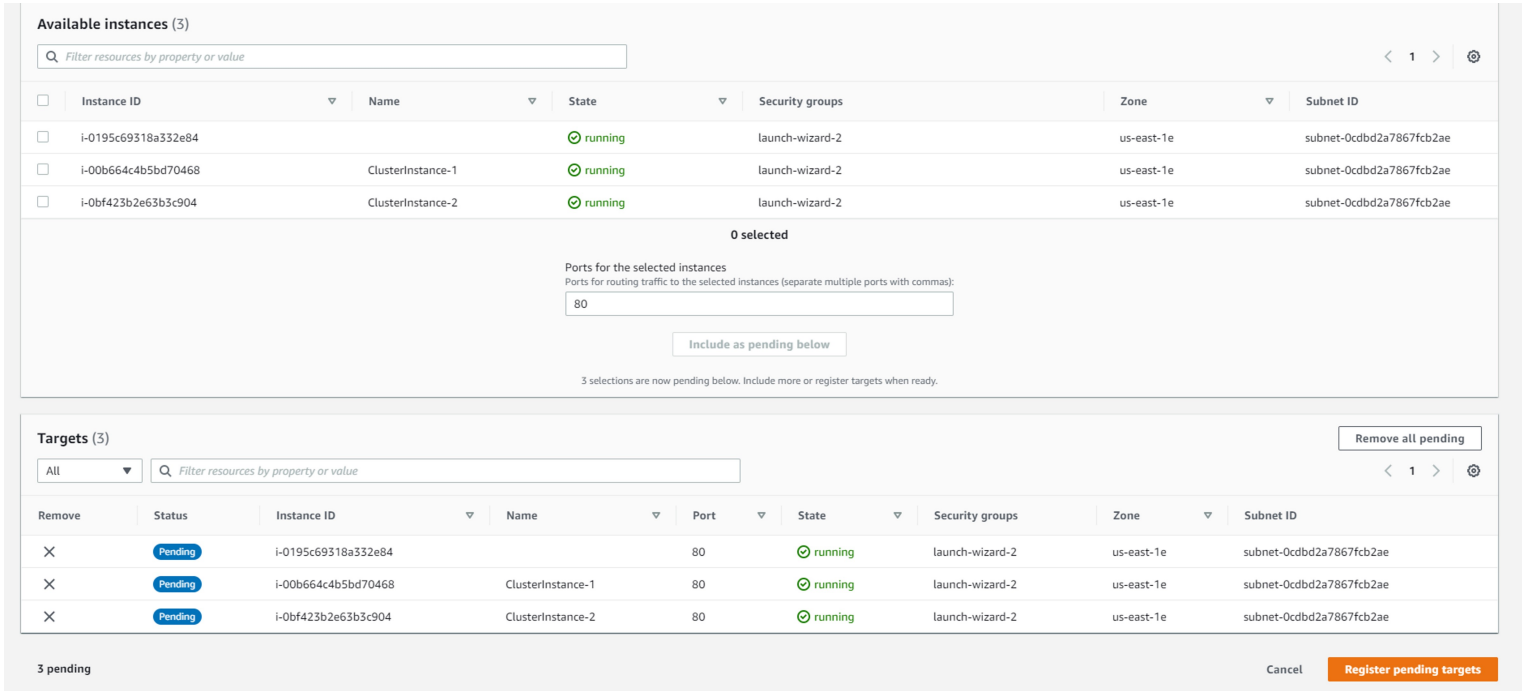


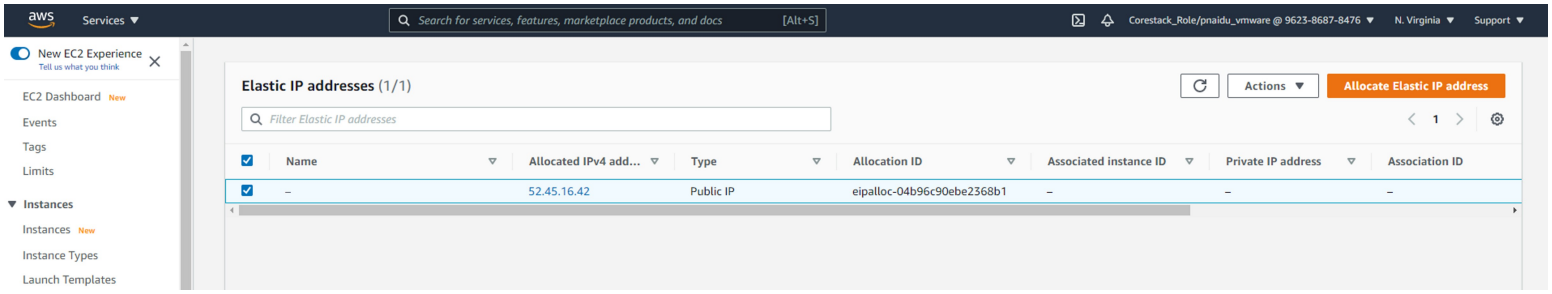
Elastic IP configuration for all the three instances




EC2 > Elastic IP addresses > Associate Elastic IP address
Associate Elastic IP address
Choose the instance or network interface to associate to this Elastic IP address (52.45.16.42)
Elastic IP address: 52.45.16.42
Resource type
Choose the type of resource with which to associate the Elastic IP address.
Instance
Network interface
If you associate an Elastic IP address to an instance that already has an Elastic IP address associated, this previously associated Elastic IP address will be disassociated but still allocated to your account. Learn more
Instance
i-00b664c4b5bd70468
Private IP address
The private IP address with which to associate the Elastic IP address.
172.31.57.225
Reassociation
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.
Allow this Elastic IP address to be reassociated
Cancel
Associate


Elastic IP address associated successfully.
Elastic IP address 52.45.16.42 has been associated with instance i-00b664c4b5bd70468
Elastic IP addresses (1/3)
Filter Elastic IP addresses
Actions
Allocate Elastic IP address
View details
Release Elastic IP addresses
Associate Elastic IP address
Disassociate Elastic IP address
Name
Allocated IPv4 add...
Type
Allocation ID
Associated instance ID
18.213.139.151 Public IP eipalloc-0f6e0af0466d0f12e i-0195c69318a332e84
52.45.16.42 Public IP eipalloc-04b96c90ebe2368b1 i-00b664c4b5bd70468 172.31.57.225 eipassoc-0370c68ebce4cd
52.5.98.24 Public IP eipalloc-0ad9f1a1384dd0537


Available instances (3)
Filter resources by property or value
Instance ID
Name
State
Security groups
Zone
Subnet ID
i-0195c69318a332e84 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
i-00b664c4b5bd70468 ClusterInstance-1 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
i-0bf423b2e63b3c904 ClusterInstance-2 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
0 selected
Ports for the selected instances
Ports for routing traffic to the selected instances (separate multiple ports with commas):
80
Include as pending below
3 selections are now pending below. Include more or register targets when ready.
Targets (3)
Remove all pending
All
Filter resources by property or value
Remove
Status
Instance ID
Name
Port
State
Security groups
Zone
Subnet ID
Pending i-0195c69318a332e84 80 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
Pending i-00b664c4b5bd70468 ClusterInstance-1 80 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
Pending i-0bf423b2e63b3c904 ClusterInstance-2 80 running launch-wizard-2 us-east-1e subnet-0cdbd2a7867fcb2ae
3 pending
Cancel
Register pending targets



Install Docker and Kubernetes on the cluster

completed the installation of Docker on all the three instances first.

ubuntu@ip-172-31-54-80:~$ history
1 sudo apt-get update
2 sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
3 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
4 sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
5 sudo apt-get install docker-ce
6 docker –version
7 sudo docker --version

Configured Master node with Kubeadm init and joined the rest of the nodes to the master

sudo kubeadm init
mkdir -p $HOME/.kube
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl get nodes

Applied the overlay network for K8

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
kubectl get pods -n kube-system

```
Preparing to unpack .../1-containerd.io_1.4.6-1_amd64.deb ...
Unpacking containerd.io (1.4.6-1) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../3-docker-ce_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../4-docker-ce-rootless-extras_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../5-docker-scan-plugin_0.7.0~ubuntu-bionic_amd64.deb ...
Unpacking docker-scan-plugin (0.7.0~ubuntu-bionic) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../6-libltdl7_2.4.6-2_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.6-2) ...
Setting up containerd.io (1.4.6-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up docker-ce-rootless-extras (5:20.10.6~3-0~ubuntu-bionic) ...
Setting up docker-scan-plugin (0.7.0~ubuntu-bionic) ...
Setting up libltdl7:amd64 (2.4.6-2) ...
Setting up docker-ce-cli (5:20.10.6~3-0~ubuntu-bionic) ...
Setting up pigz (2.4-1) ...
Setting up docker-ce (5:20.10.6~3-0~ubuntu-bionic) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Processing triggers for systemd (237-3ubuntu10.46) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-54-80:~$ docker -version
docker: '-version' is not a docker command.
See 'docker --help'
ubuntu@ip-172-31-54-80:~$ sudo docker --version
Docker version 20.10.6, build 370c289
ubuntu@ip-172-31-54-80:~$
```

```
ubuntu@ip-172-31-54-80:~$ sudo kubeadm token create --print-join-command
kubeadm join 172.31.54.80:6443 --token 1jr0n4.pdx18hyv9rxtt7vl --discovery-token-ca-cert-hash sha256:b4ee6f520fb56bd187e59df30f009458578148c33ad1176d2ea0e68f93059ffe
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME              STATUS     ROLES                  AGE     VERSION
ip-172-31-54-80   NotReady   control-plane,master   9m40s   v1.21.1
ip-172-31-57-225  NotReady   <none>                 14s     v1.21.1
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME              STATUS     ROLES                  AGE     VERSION
ip-172-31-49-129  NotReady   <none>                 17s     v1.21.1
ip-172-31-54-80   NotReady   control-plane,master   11m     v1.21.1
ip-172-31-57-225  NotReady   <none>                 2m7s    v1.21.1
ubuntu@ip-172-31-54-80:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
ubuntu@ip-172-31-54-80:~$ kubectl get pods -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-5rp7h                  0/1     Pending   0          17m
coredns-558bd4d5db-t425w                  0/1     Pending   0          17m
etcd-ip-172-31-54-80                      1/1     Running   0          18m
kube-apiserver-ip-172-31-54-80            1/1     Running   0          18m
kube-controller-manager-ip-172-31-54-80   1/1     Running   0          18m
kube-proxy-cww74                          1/1     Running   0          8m40s
kube-proxy-vsjht                          1/1     Running   0          6m50s
kube-proxy-wpxpv                          1/1     Running   0          17m
kube-scheduler-ip-172-31-54-80            1/1     Running   0          18m
weave-net-5zs9j                           2/2     Running   1          12s
weave-net-851tl                           2/2     Running   1          12s
weave-net-gkkcq                           2/2     Running   1          12s
ubuntu@ip-172-31-54-80:~$ kubectl get pods -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-5rp7h                  0/1     Pending   0          18m
coredns-558bd4d5db-t425w                  0/1     Pending   0          18m
etcd-ip-172-31-54-80                      1/1     Running   0          18m
kube-apiserver-ip-172-31-54-80            1/1     Running   0          18m
kube-controller-manager-ip-172-31-54-80   1/1     Running   0          18m
kube-proxy-cww74                          1/1     Running   0          8m45s
kube-proxy-vsjht                          1/1     Running   0          6m55s
kube-proxy-wpxpv                          1/1     Running   0          18m
kube-scheduler-ip-172-31-54-80            1/1     Running   0          18m
weave-net-5zs9j                           2/2     Running   1          17s
weave-net-851tl                           2/2     Running   1          17s
weave-net-gkkcq                           2/2     Running   1          17s
ubuntu@ip-172-31-54-80:~$ kubectl get nodes
NAME              STATUS   ROLES                  AGE     VERSION
ip-172-31-49-129  Ready    <none>                 7m2s    v1.21.1
ip-172-31-54-80   Ready    control-plane,master   18m     v1.21.1
ip-172-31-57-225  Ready    <none>                 8m52s   v1.21.1
```

Implement the network policies at the database pod to allow ingress traffic from the front-end application pod

Created below mentioned yaml to create the network policy at the database pod to allow ingress traffic from front-end application POD

```
ubuntu@ip-172-31-54-80:~$ cat networkpolicy.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
ubuntu@ip-172-31-54-80:~$ kubectl create -f networkpolicy.yaml
networkpolicy.networking.k8s.io/test-network-policy created
```

Implement the network policies at the database pod to allow ingress traffic from the front-end application pod

```
ubuntu@ip-172-31-54-80:~$ cat > userrole.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list", "create", "delete", "update"]
^C
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f userrole.yaml
role.rbac.authorization.k8s.io/pod-reader created
ubuntu@ip-172-31-54-80:~$
```

Configure application on the pod

```
ubuntu@ip-172-31-54-80:~$ cat > pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    role: db
spec:
  containers:
    - name: web
      image: nginx
      ports:
        - name: web
          containerPort: 80
          protocol: TCP^C
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f pod.yaml
pod/static-web created
ubuntu@ip-172-31-54-80:~$
```

```
ubuntu@ip-172-31-54-80:~$ ^C
ubuntu@ip-172-31-54-80:~$ kubectl create -f pod.yaml
pod/static-web created
ubuntu@ip-172-31-54-80:~$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
static-web   1/1     Running   0          38s
ubuntu@ip-172-31-54-80:~$
```

Take snapshot of ETCD database

Download and configured ETCD from GitHub

```
ubuntu@ip-172-31-54-80:~$ etcd --version
etcd Version: 3.3.13
Git SHA: 98d3084
Go Version: go1.10.8
Go OS/Arch: linux/amd64
ubuntu@ip-172-31-54-80:~$
```

Executed below mentioned command to create a backup WAL for ETCD database

```
2021-05-30 15:00:45.992243 I | ignoring v3 raft entry
2021-05-30 15:00:45.992255 I | ignoring v3 raft entry
2021-05-30 15:00:45.992270 I | ignoring v3 raft entry
2021-05-30 15:00:45.992285 I | ignoring v3 raft entry
2021-05-30 15:00:45.992299 I | ignoring v3 raft entry
2021-05-30 15:00:45.992311 I | ignoring v3 raft entry
ubuntu@ip-172-31-54-80:~$ ls
cluster.yml  etcd-backup  etcd-v3.3.13-linux-amd64  etcd-v3.3.13-linux-amd64.tar.gz  member  networkpolicy.yaml  pod.yaml  userrole.yaml
ubuntu@ip-172-31-54-80:~$ cd /backupdir/
-bash: cd: /backupdir/: Permission denied
ubuntu@ip-172-31-54-80:~$ ls
cluster.yml  etcd-backup  etcd-v3.3.13-linux-amd64  etcd-v3.3.13-linux-amd64.tar.gz  member  networkpolicy.yaml  pod.yaml  userrole.yaml
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/
member
ubuntu@ip-172-31-54-80:~$ sudo cd /backupdir/
sudo: cd: command not found
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/member
snap  wal
ubuntu@ip-172-31-54-80:~$ sudo ls /backupdir/member/wal/
0000000000000000-0000000000000000.wal
ubuntu@ip-172-31-54-80:~$ sudo etcdctl backup --data-dir /var/lib/etcd/ --backup-dir /backupdir
```

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

Below mentioned autoscaling yaml was used for the same

```
ubuntu@ip-172-31-51-16:~$ cat podautoscaler.yml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: web
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Pod
    name: static-web
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50

ubuntu@ip-172-31-51-16:~$ kubectl apply -f podautoscaler.yml
horizontalpodautoscaler.autoscaling/web created
ubuntu@ip-172-31-51-16:~$ kubectl get hpa
NAME    REFERENCE        TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
web     Pod/static-web   <unknown>/50%    1         10        0          18s
```

Complete code has been pushed to GitHub for review

complete code is pushed to GitHub for review