**CAPESTONE PROJECT-1**

**POST GRADUATE PROGRAM IN DEVOPS**

**NAME: TATA SAIKRISHNA**

**MAIL:dcgm.crmr@gmail.com**

**Mobile:7382801892**

**Registration ID:731272**

**https://github.com/tatasaikrishna/devops-capstone-project.git**

**Infra Optimization.**

Course-end 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:**

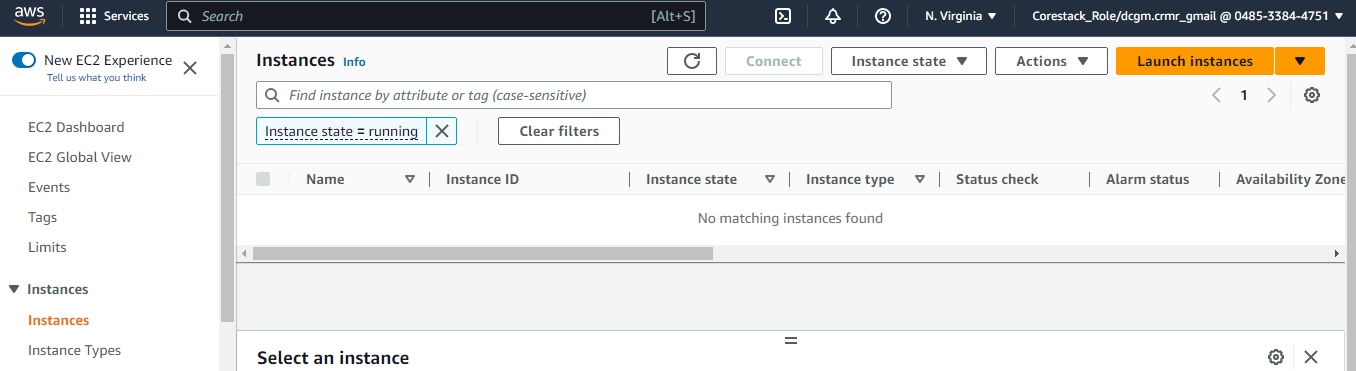
1. You need to document the steps and write the algorithms in them.
2. The submission of your GitHub repository link is mandatory. In order to track your tasks, you need to share the link of the repository.
3. Document the step-by-step process starting from creating test cases, then executing them, and 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)

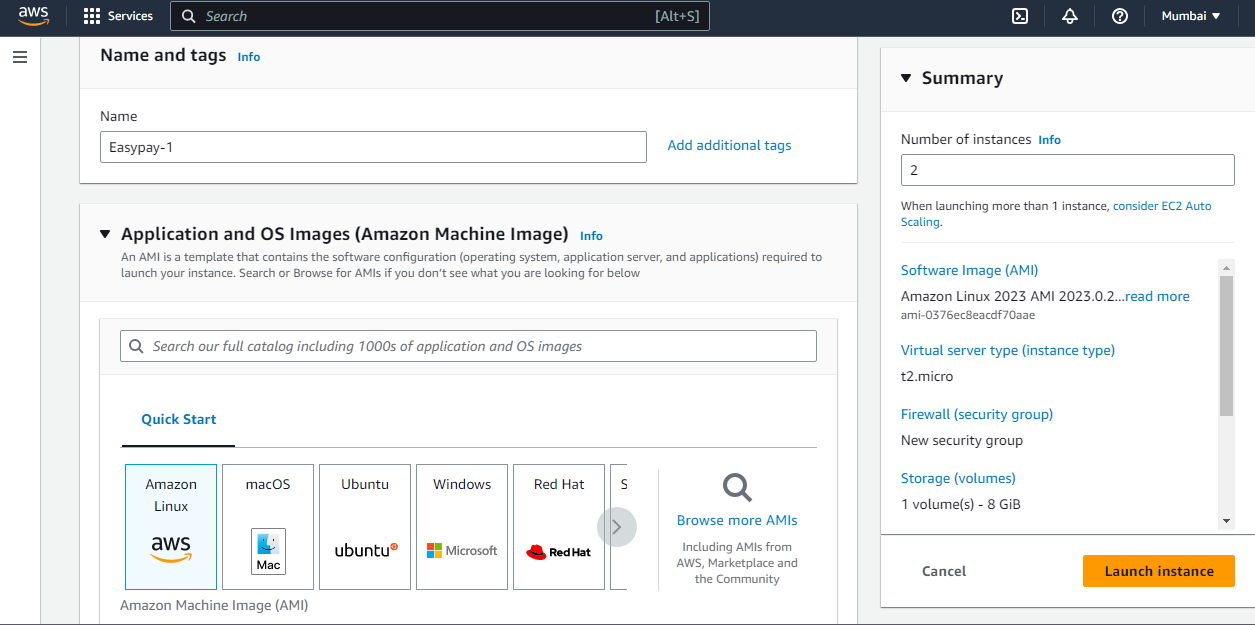
**1.Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)**

**Launch EC2 instances**

**Click on launch instance**

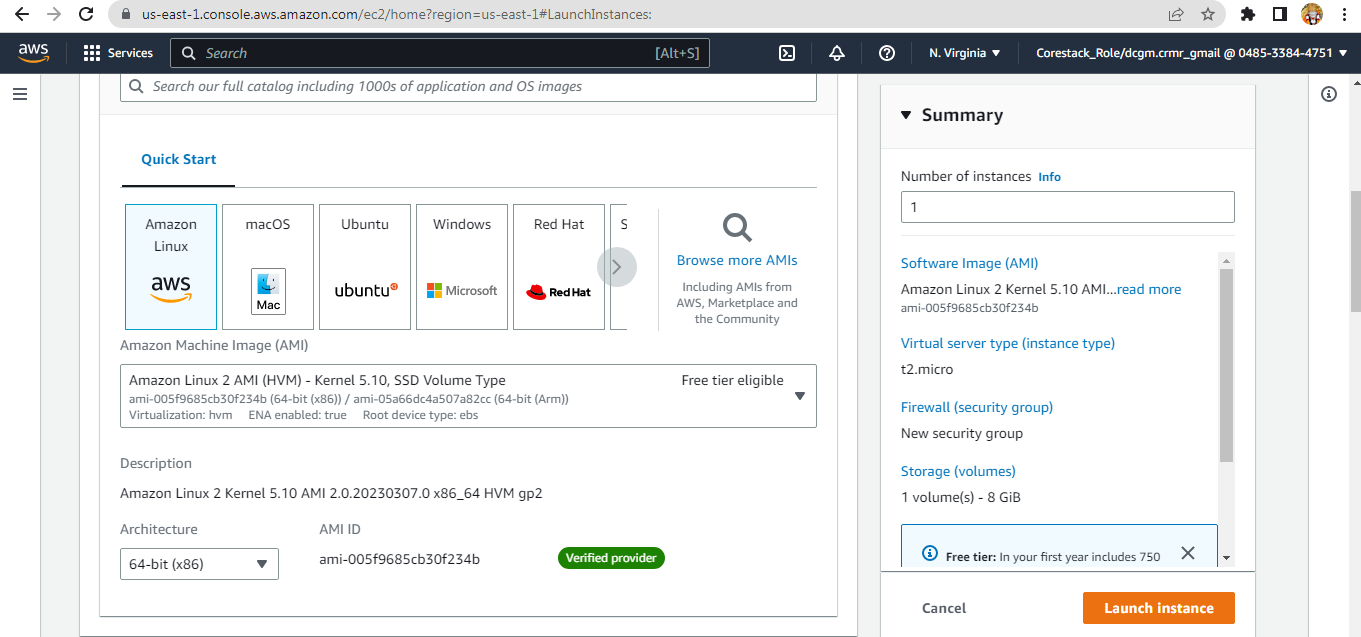


**Give instance Name: Easypay-1 & Number of Instances:2**

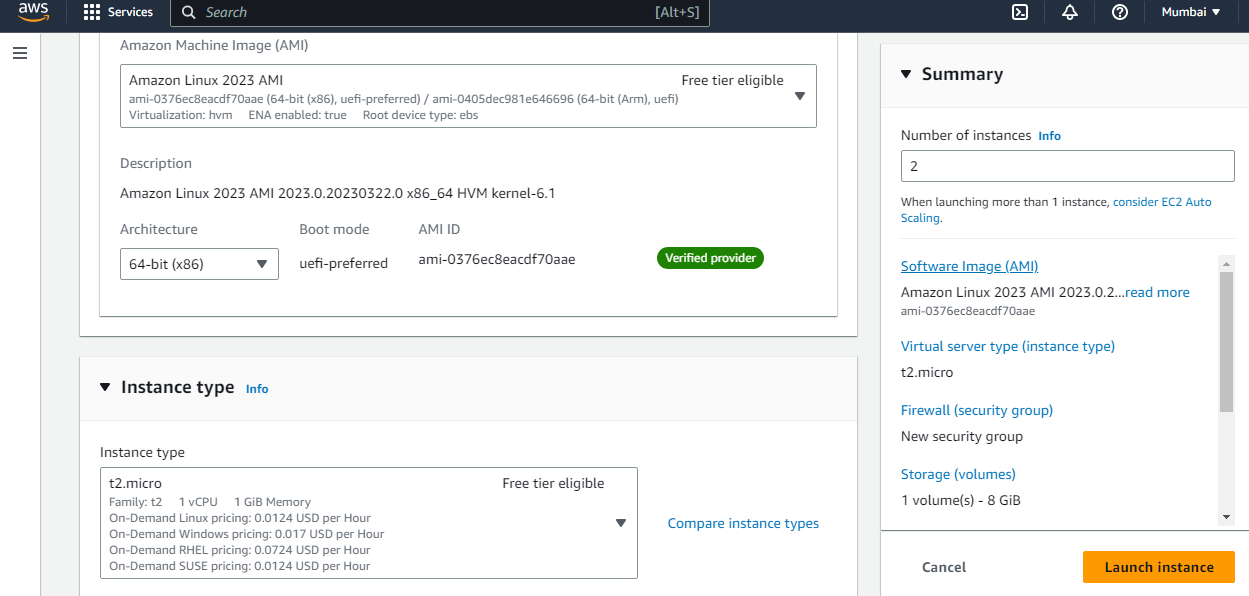


**Number of instances 2**

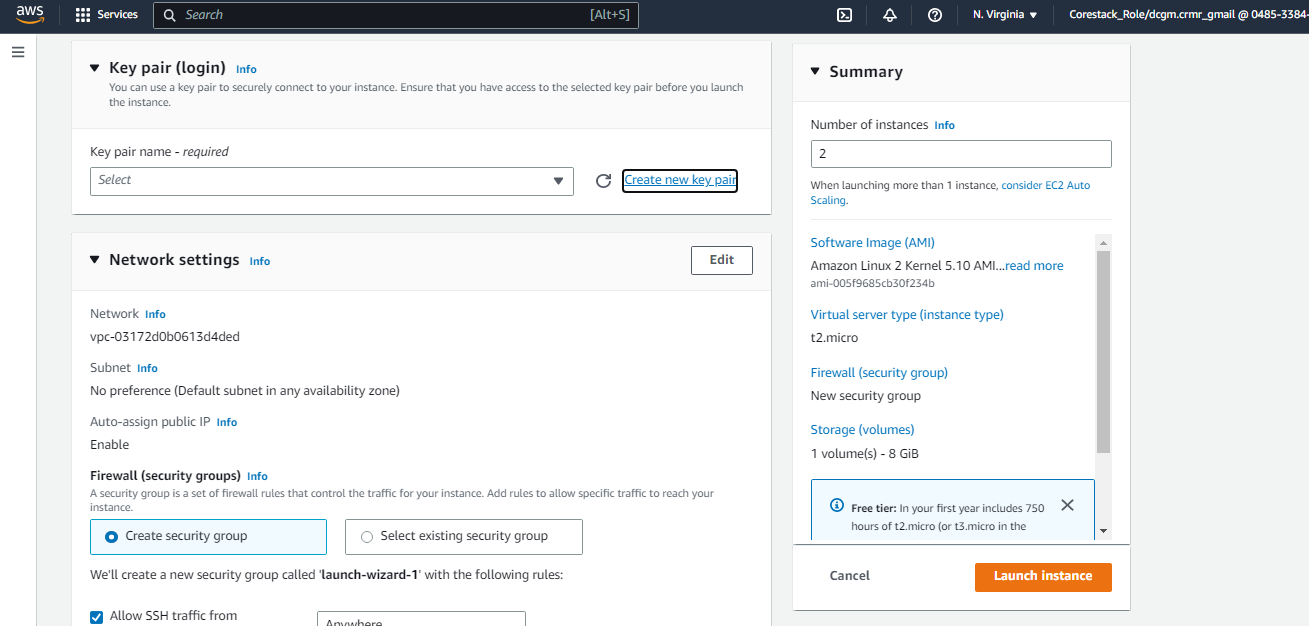
**Select AMI: Amazon Linux**

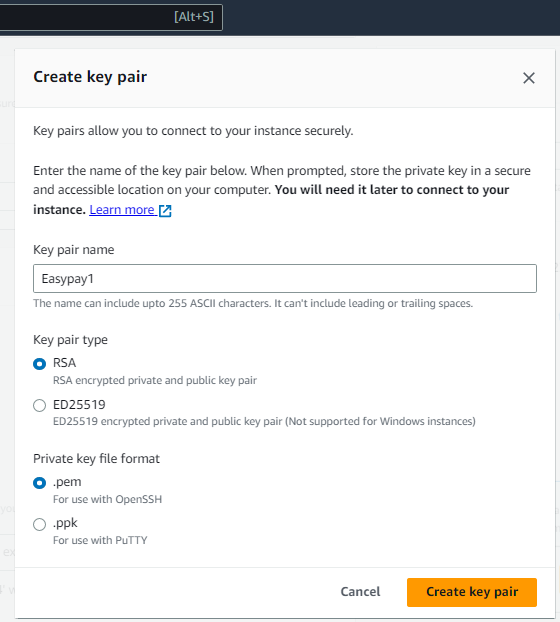


**Select Instance type: t2.macro**

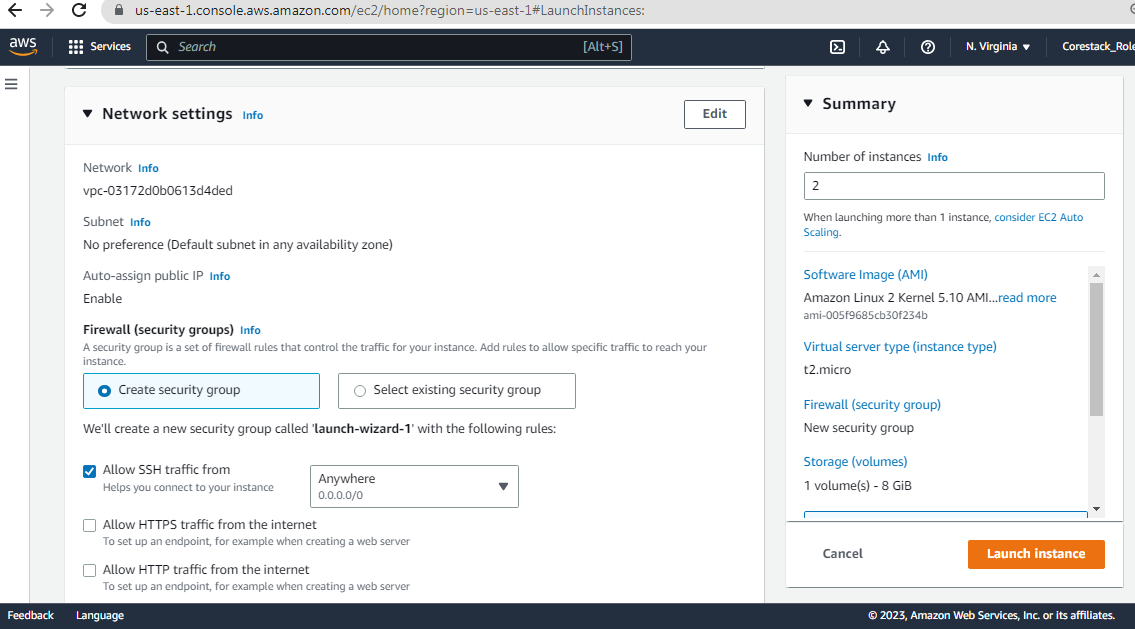


**Click on Create new key pair**

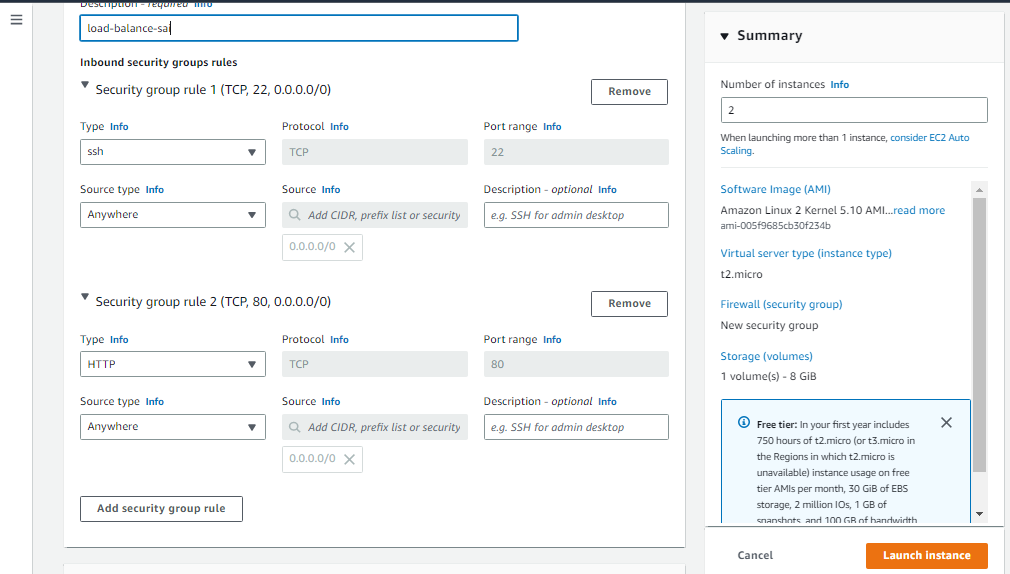




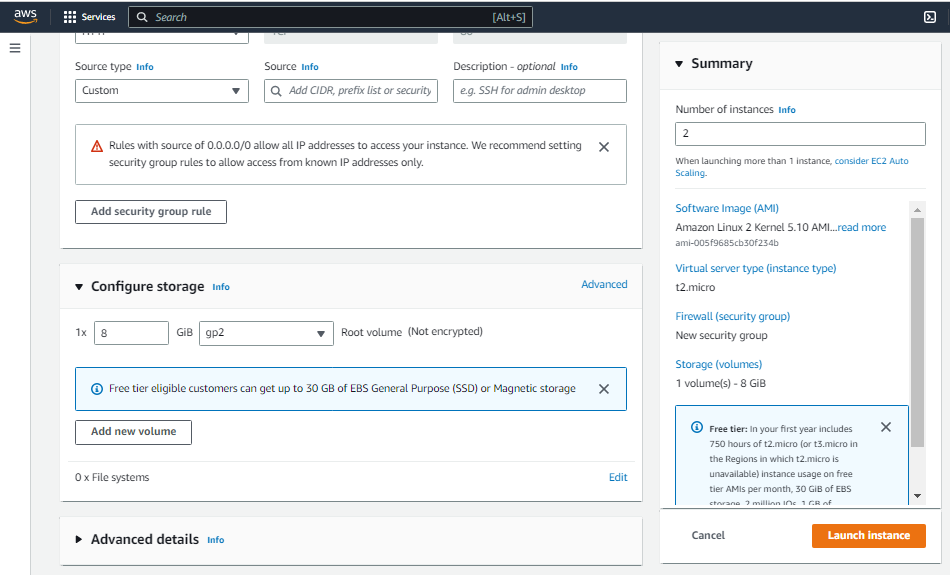
**Create Security Group**



**Give Security group name & Enable SSH and HTTP**



**Configure Storage**



In advance setting

Give Bootstrap Script Code

#!/bin/bash

sudo su

yum update -y

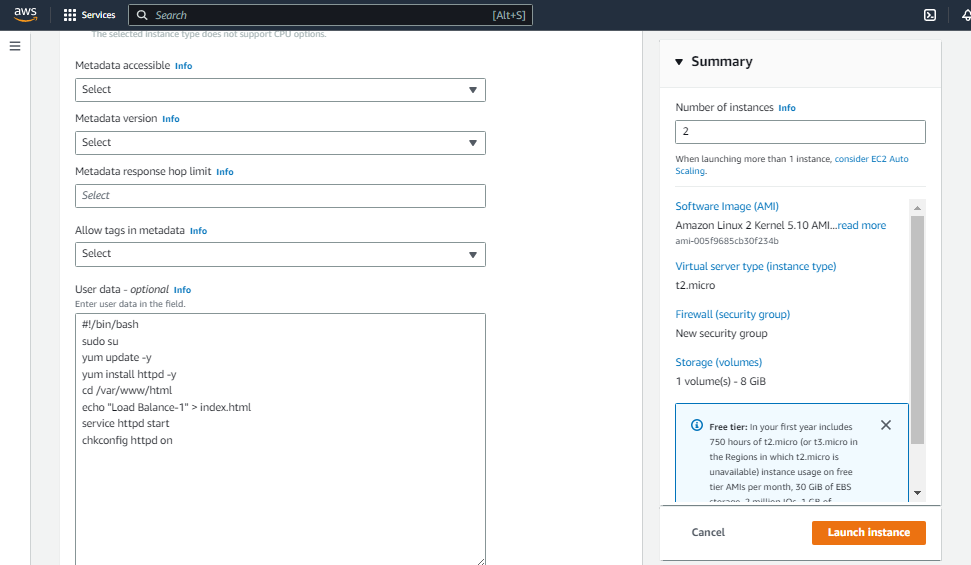
yum install httpd -y

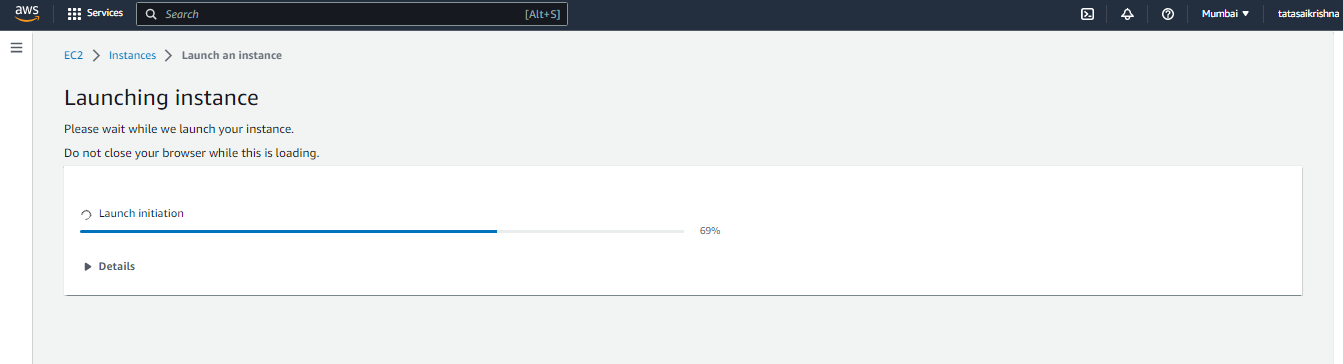
cd /var/www/html

echo "Load Balance-1" > index.html

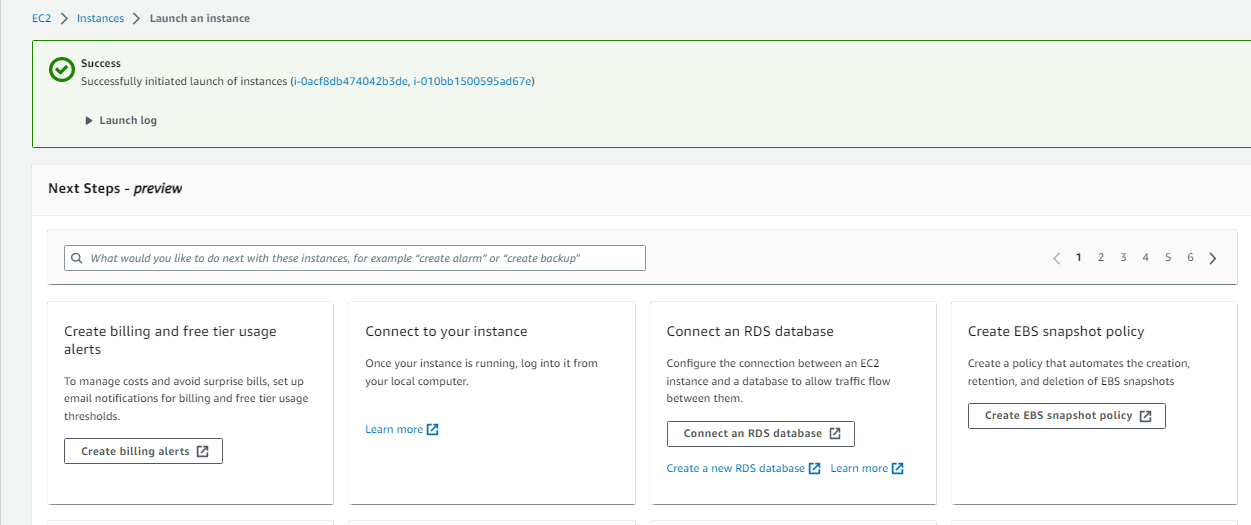
service httpd start

chkconfig httpd on

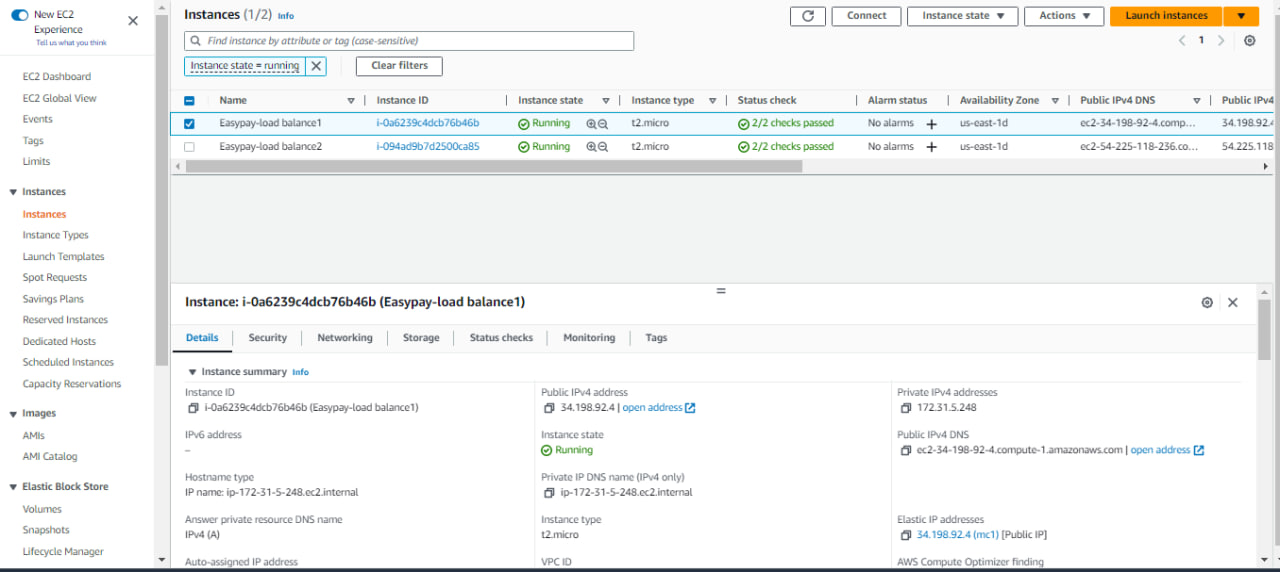




**Sucessfully Lanunc Instance**



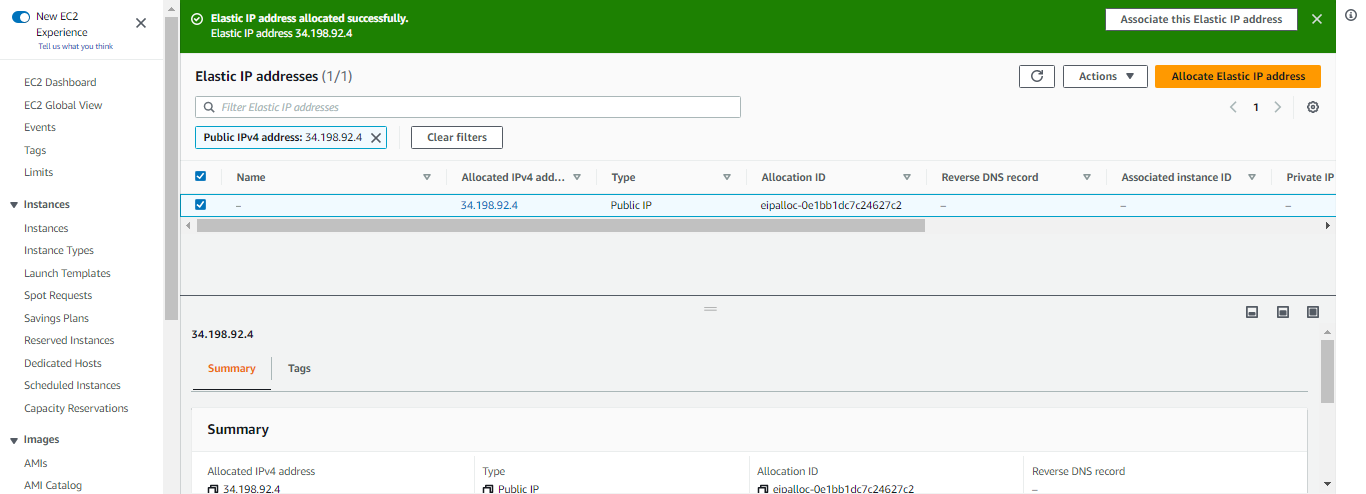
**Two instances created Easypay-1 & Easypay-2**

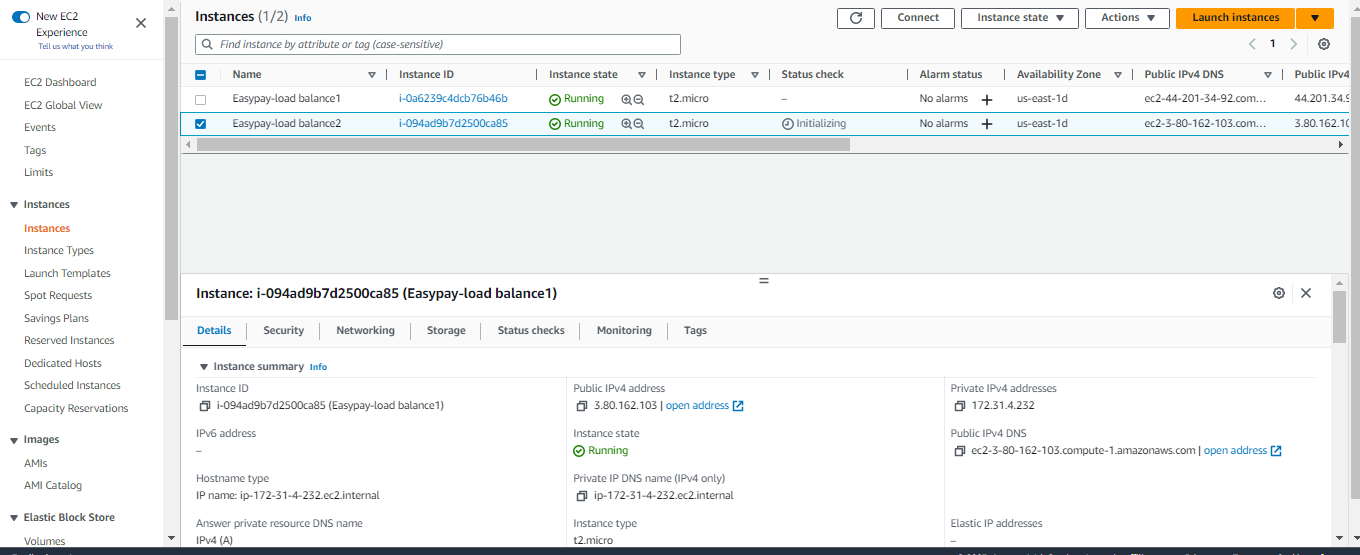


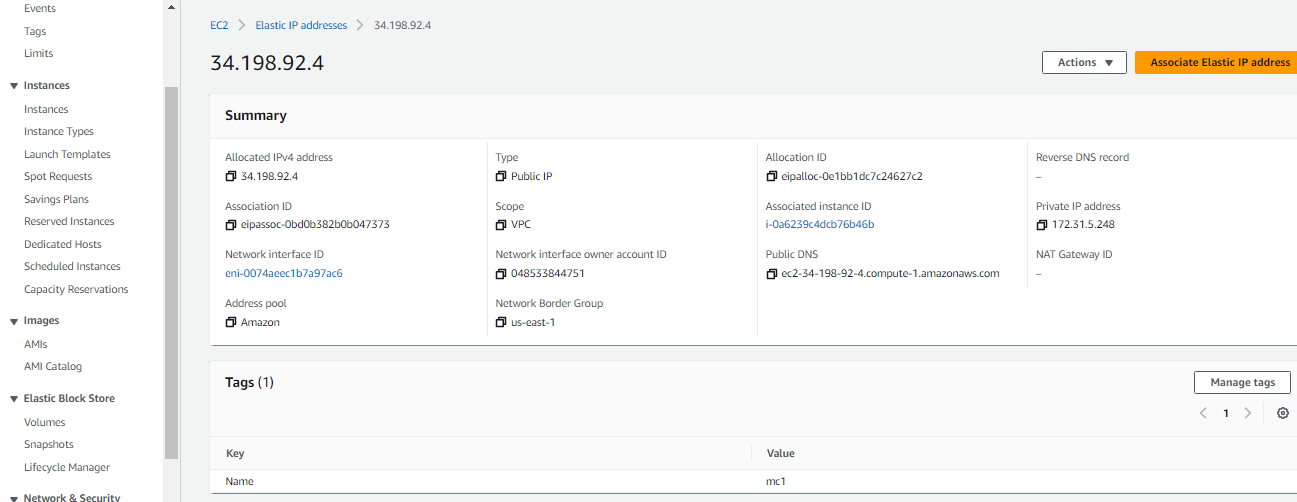
**To create Elastic IPs on 2 instances**

**Allocate Elastic IP address for 2 instances**

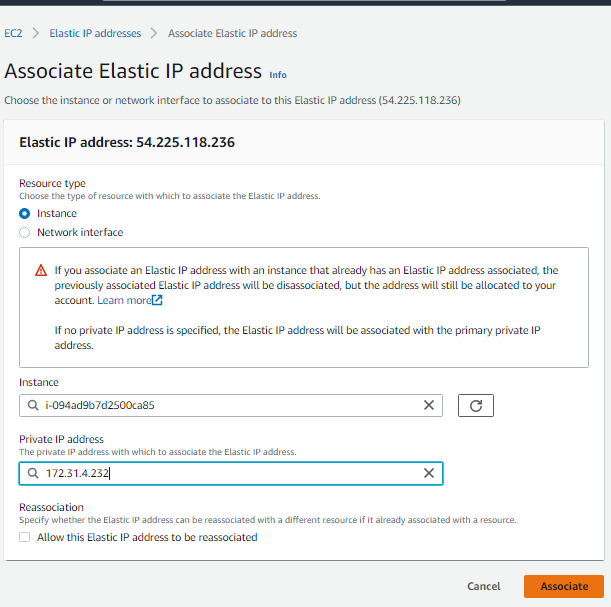
**Elastic IP: 34.198.92.4 for instance-1 Easypay-load balance1**

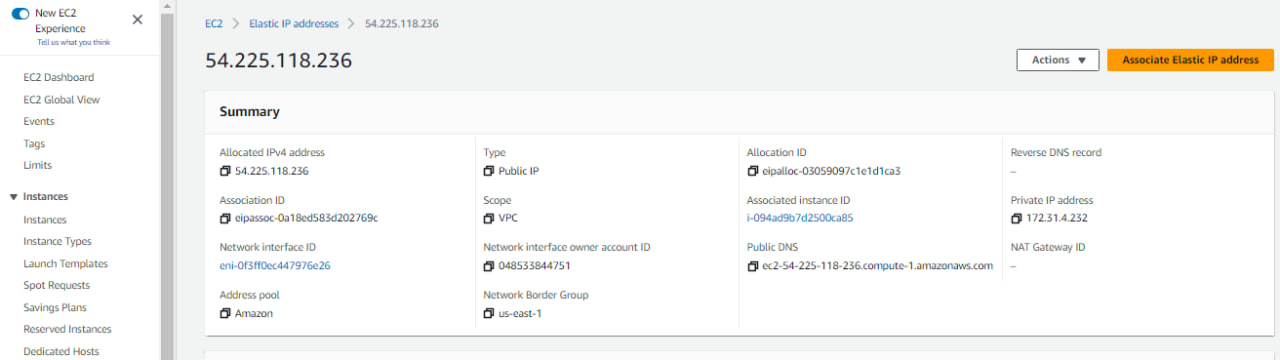
****

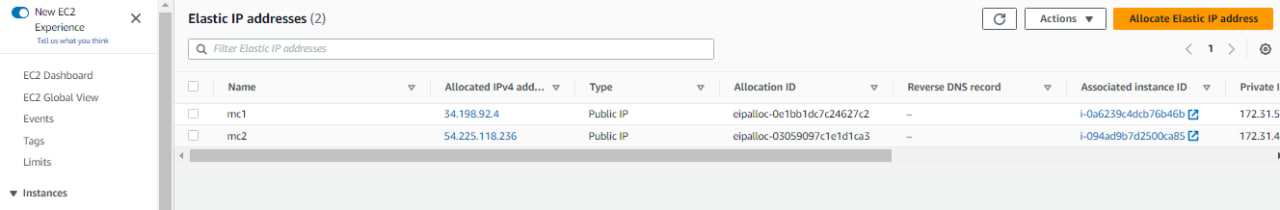
****

****

**Elastic IP: 54.225.118.236 for instance-1 Easypay-load balance2**

****

****

****

**Two Elastic IP addresses**

1. **Elastic IP: 34.198.92.4 for instance-1 Easypay-load balance1**
2. **Elastic IP: 54.225.118.236 for instance-1 Easypay-load balance2**

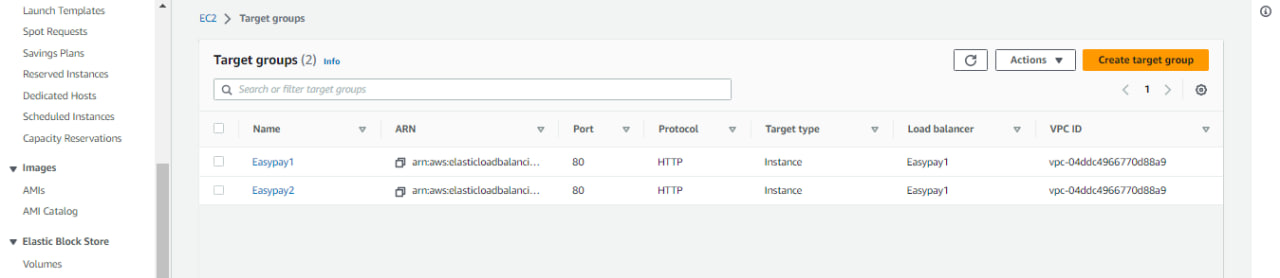
**To create a Application Load Balancer**

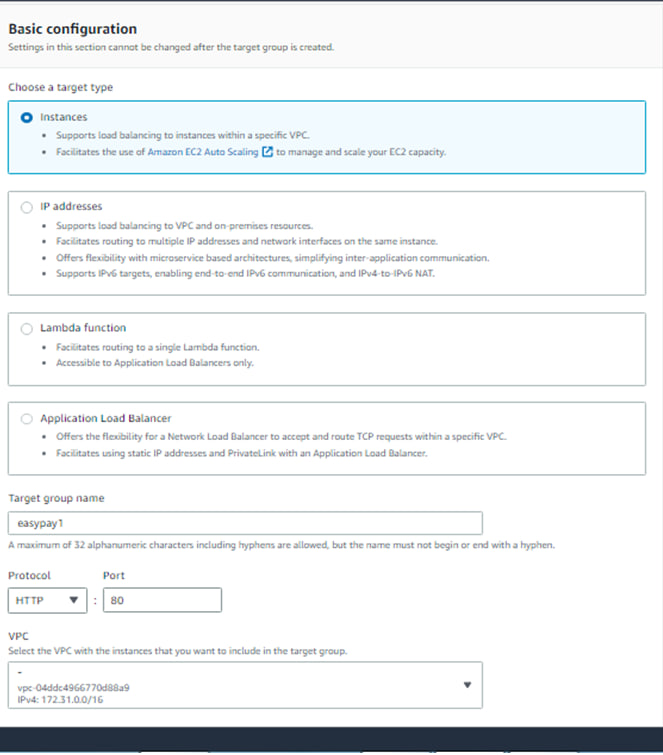
**First create target group**

**Configure your target group**

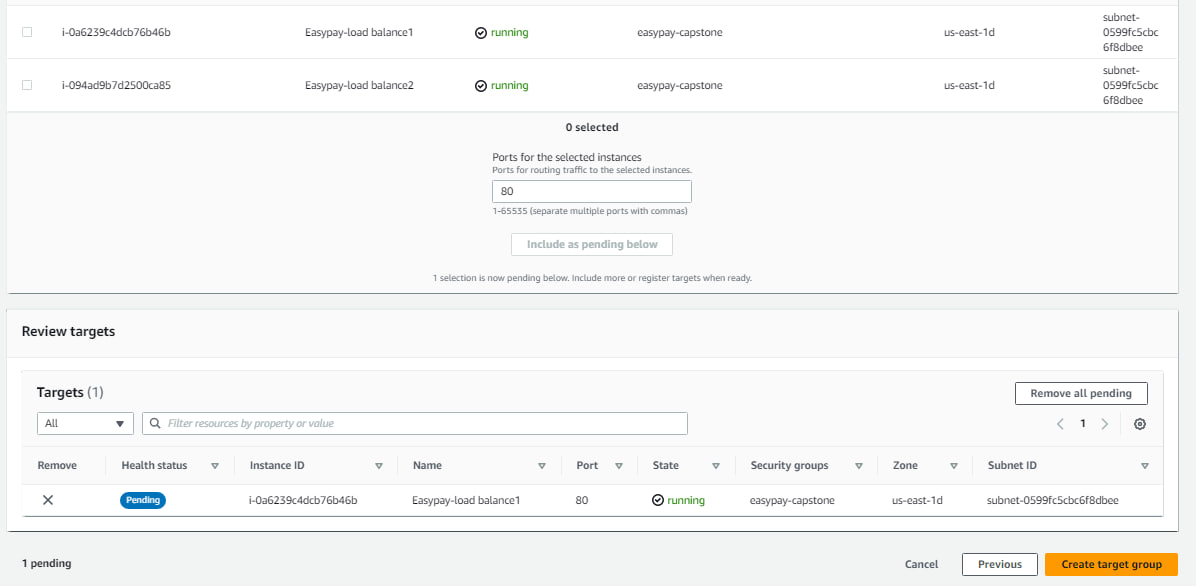
* + In the navigation pane, under **Load Balancing**, choose **Target Groups**.
  + Choose **Create target group**.
  + Under **Basic configuration**, keep the **Target type** as instance.
  + For **Target group name**, enter a name for the new target group.
  + Keep the default protocol (**HTTP**) and port (**80**).
  + Select the **VPC** containing your instances. Keep the protocol version as **HTTP1**.
  + For **Health checks**, keep the default settings.
  + Choose Next.
  + On the **Register targets** page, complete the following steps. This is an optional step for creating the load balancer. However, you must register this target if you want to test your load balancer and ensure that it is routing traffic to this target.
  + For **Available instances**, select one or more instances.
  + Keep the default port 80, and choose **Include as pending below**.
  + Choose **Create target group**.

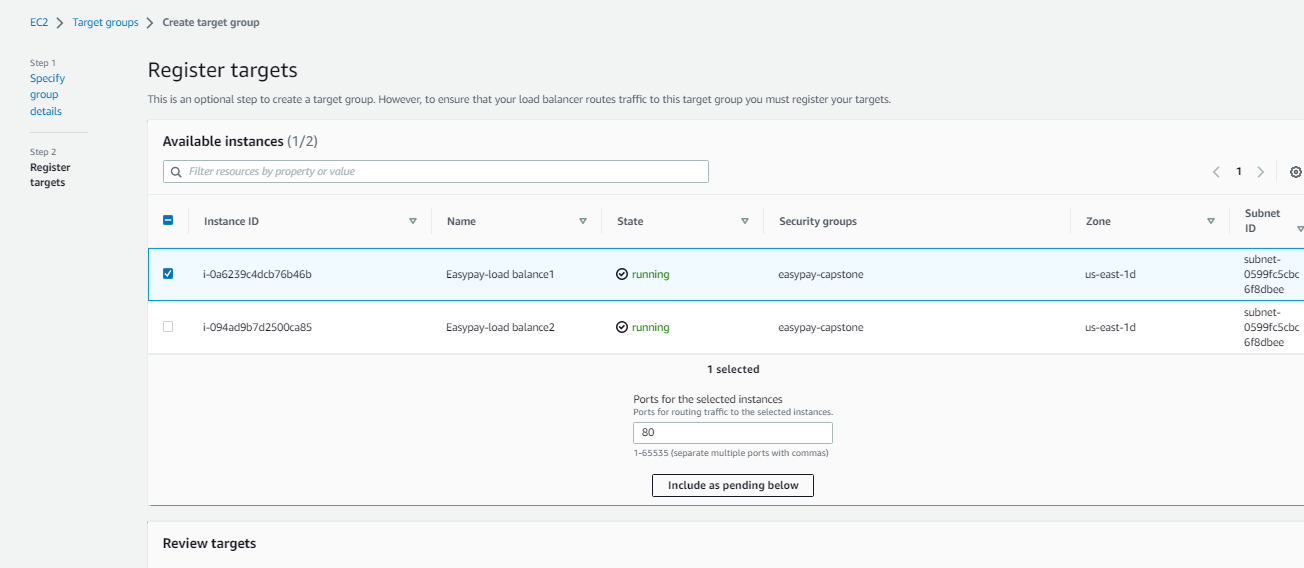
**Create target group: Easypay1 & Easypay2**

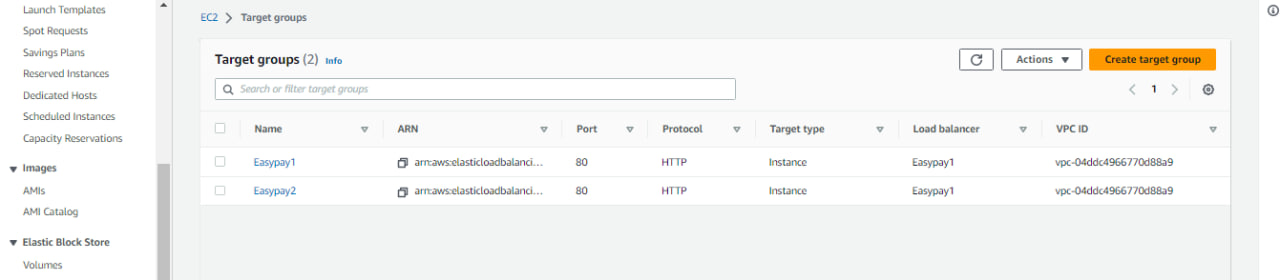
****

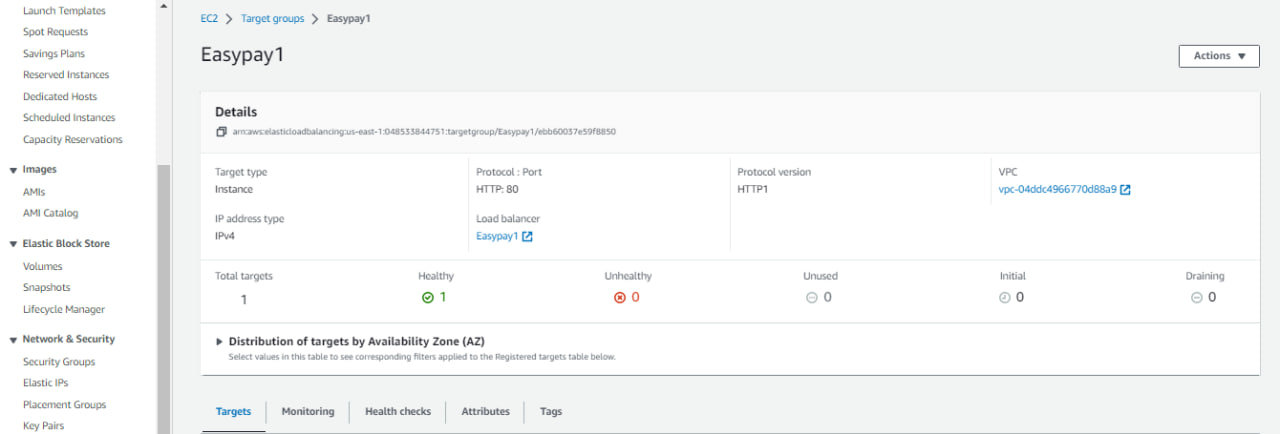
****

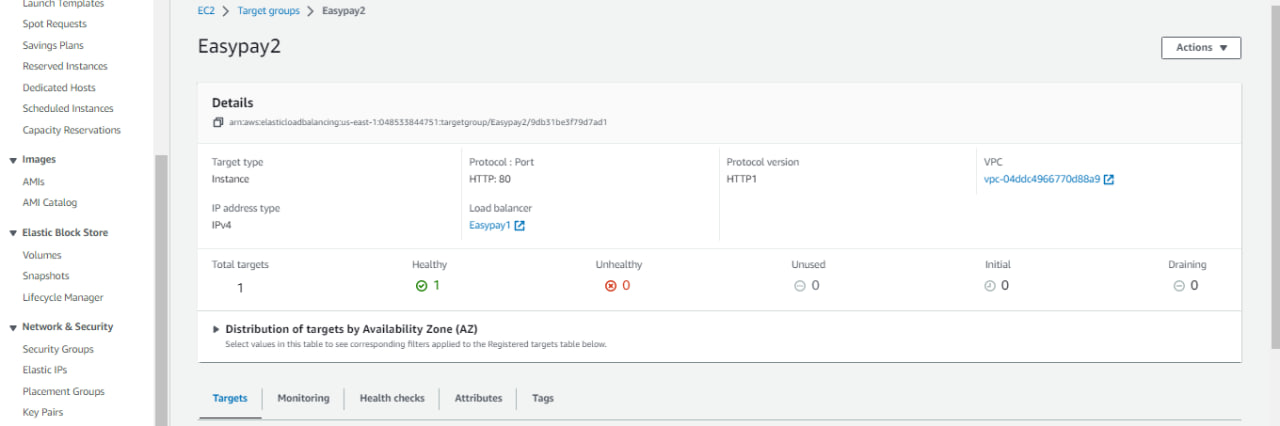
Keep the default port 80, and choose **Include as pending below**

****

****

****

****

****

**To create a Application Load Balancer**

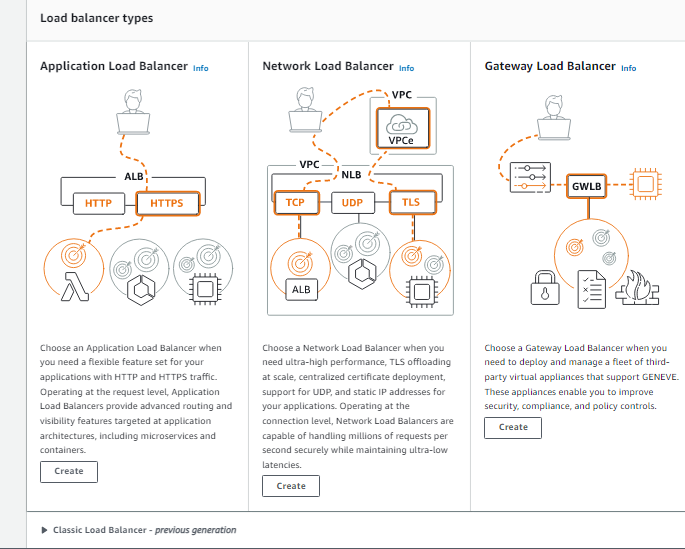
Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

On the navigation bar, choose a Region for your load balancer. Be sure to choose the same Region that you used for your EC2 instances.

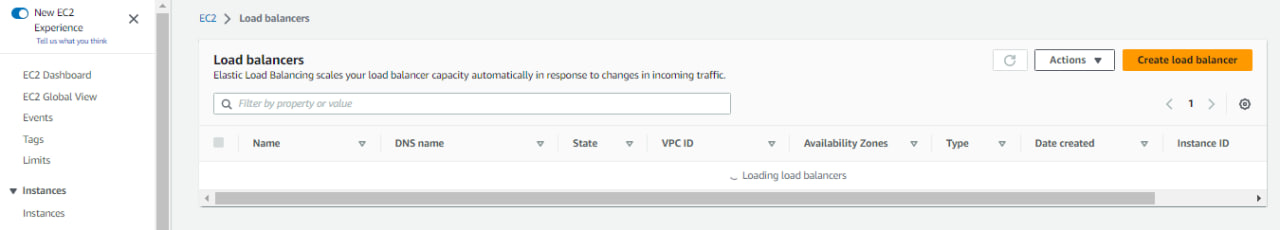
In the navigation pane, under **Load Balancing**, choose **Load Balancers**.

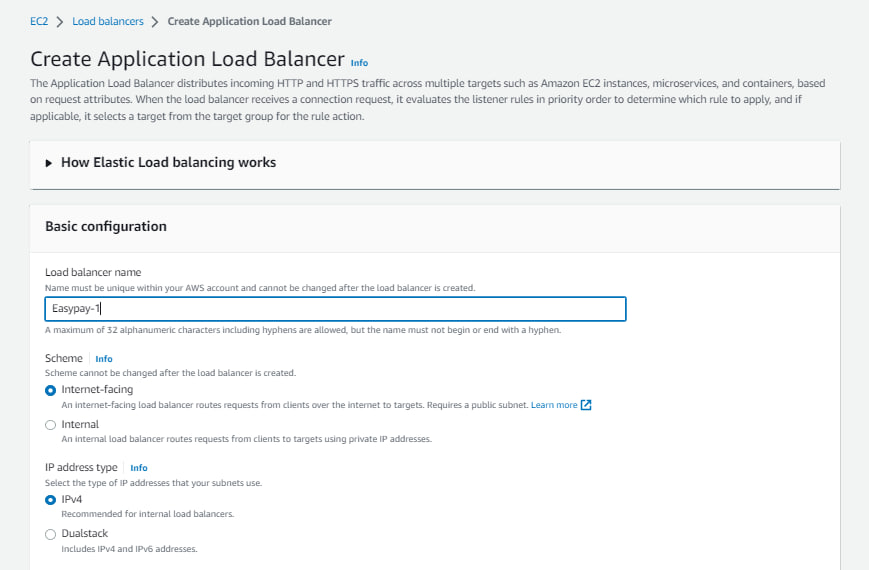
Choose **Create Load Balancer**.

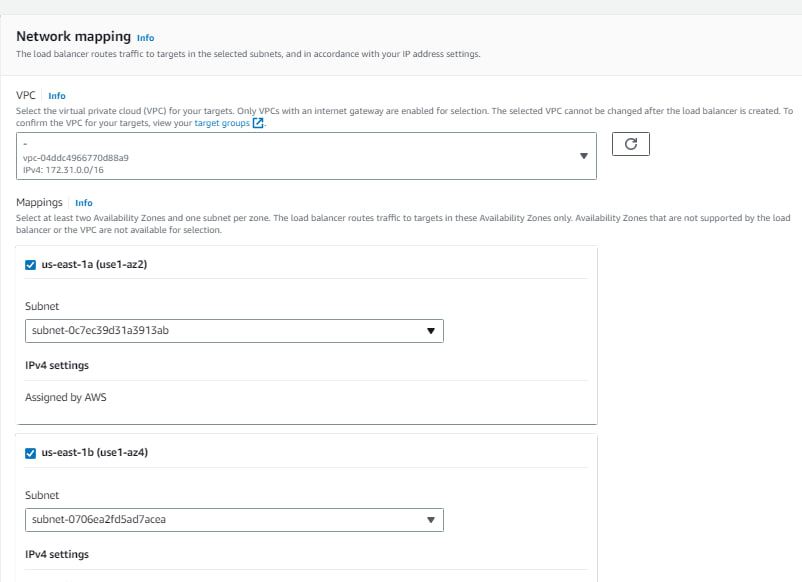
For **Application Load Balancer**, choose **Create**.

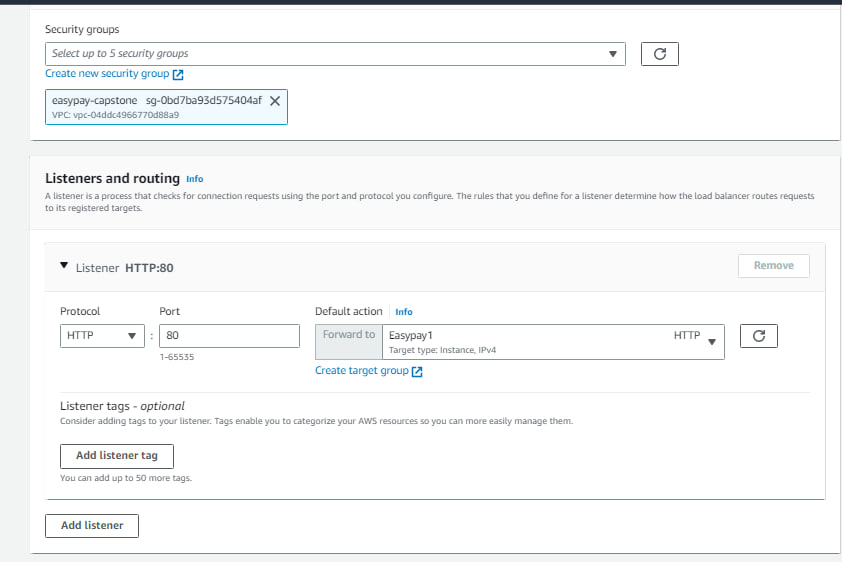


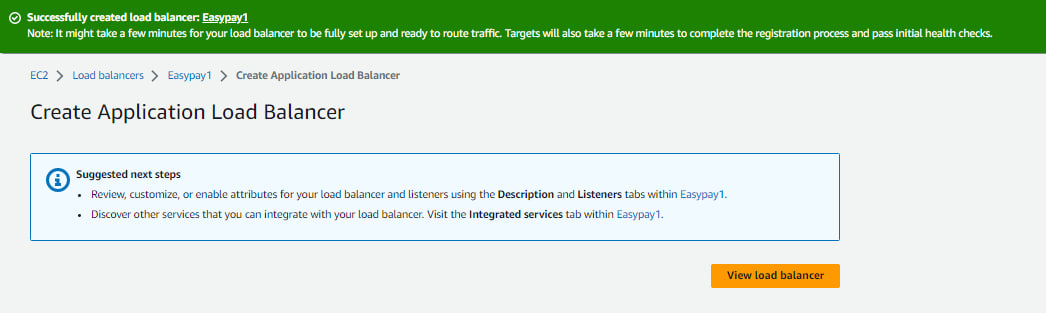
**Create Load Balancer easypay-capstone**

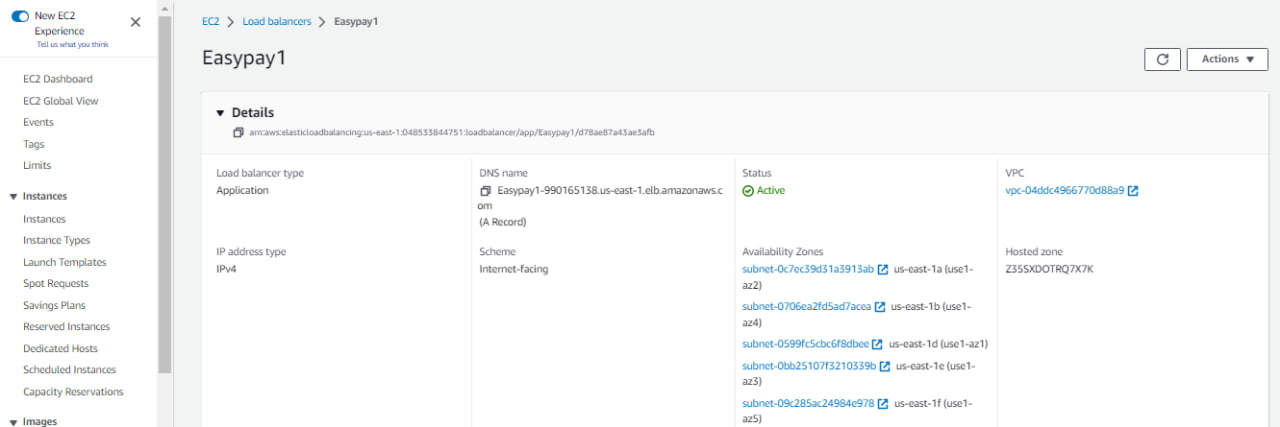












**Copy public IP and check in the Browser**

