Assignment 4:

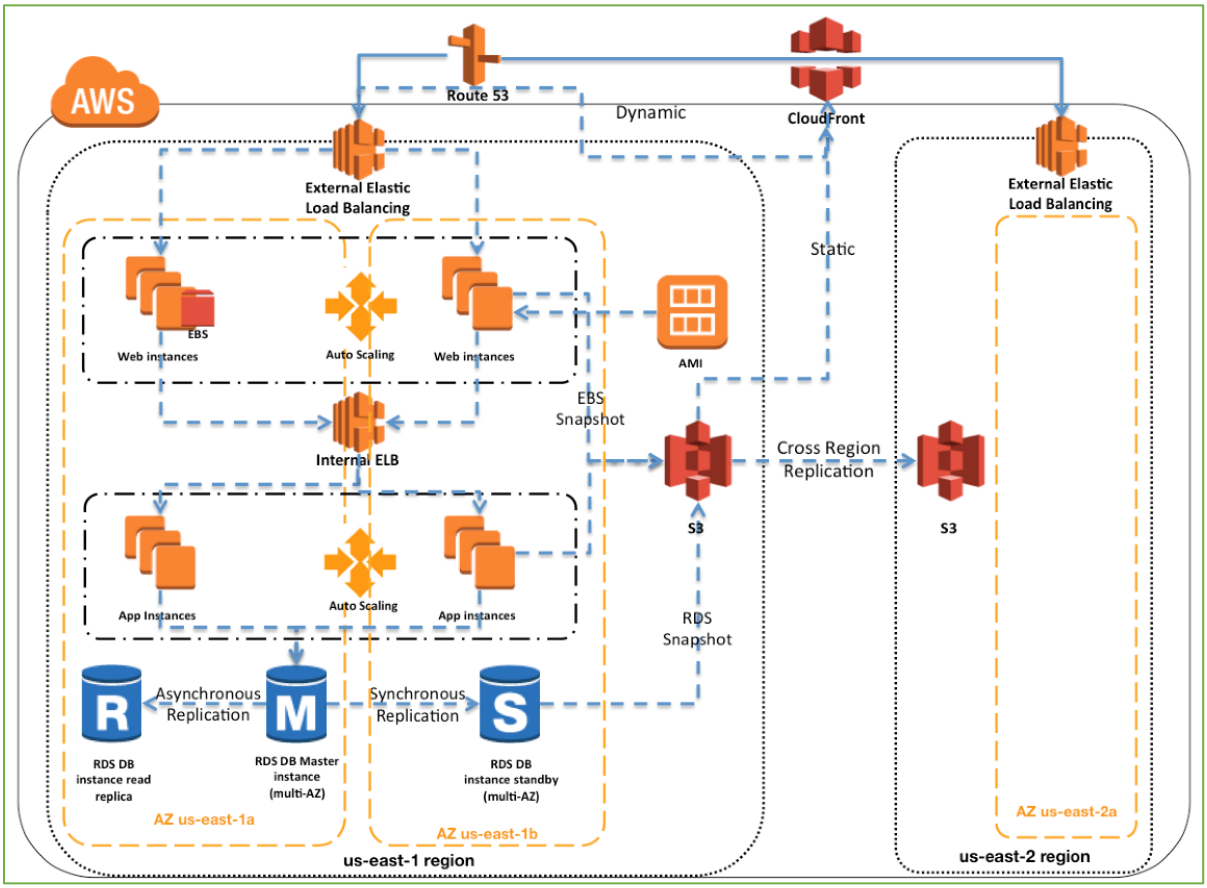
Set-up highly available, fault-tolerant, elastic and scalable architecture using cloud services to deploy web application which reactively scale-in or scale-out based on the demand.

Below is the brief description about fault-tolerant and scalable architecture.

High Availability & Fault Tolerance Architecture

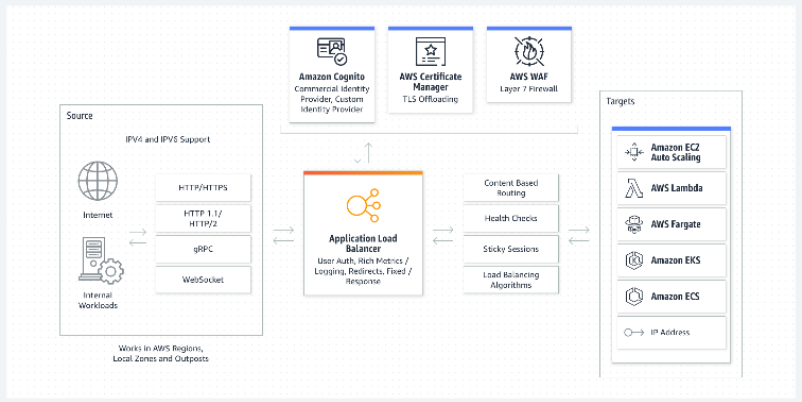
1. Amazon Web Services provides services and infrastructure to build reliable, fault-tolerant, and highly available systems in the cloud.
2. Fault-tolerance defines the ability for a system to remain in operation even if some of the components used to build the system fail.
3. Most of the higher-level services, such as S3, SimpleDB, SQS, and ELB, have been built with fault tolerance and high availability in mind.
4. Services that provide basic infrastructure, such as EC2 and EBS, provide specific features, such as availability zones, elastic IP addresses, and snapshots, that a fault-tolerant and highly available system must take advantage of and use correctly.

Example:

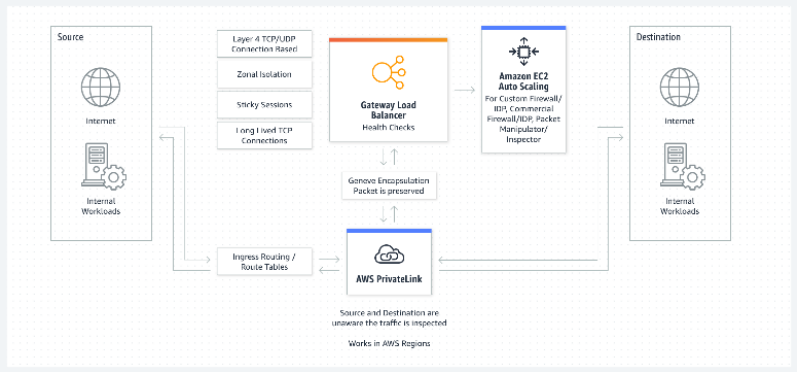


Load balancer can be configured in different ways based on requirements.

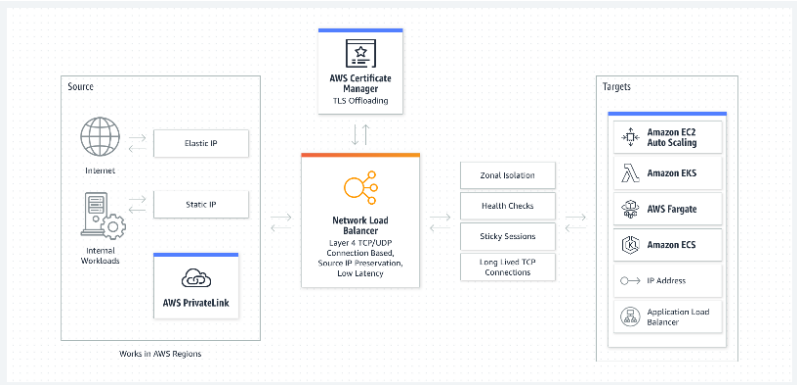
Application load Balancer



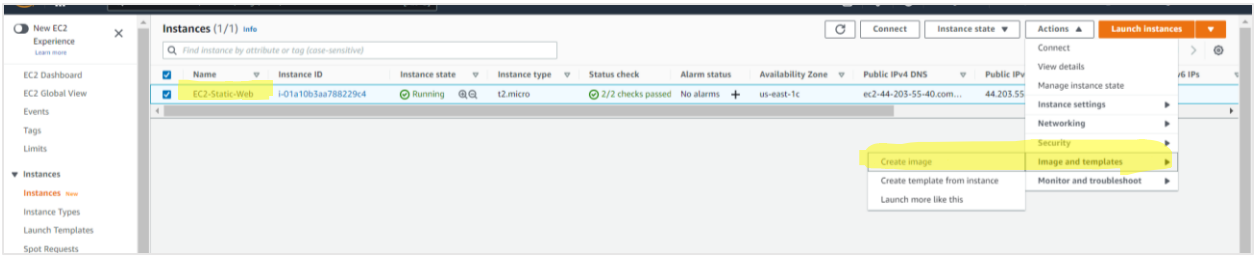
Gateway load balancer



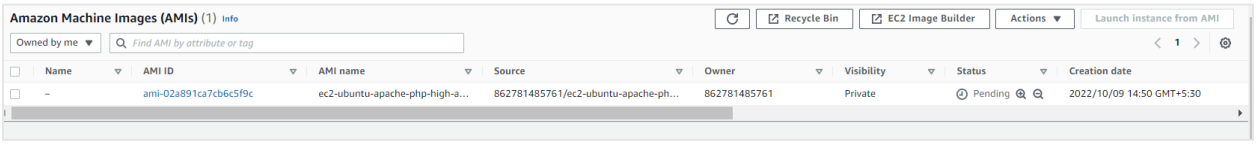
Network load balancer.



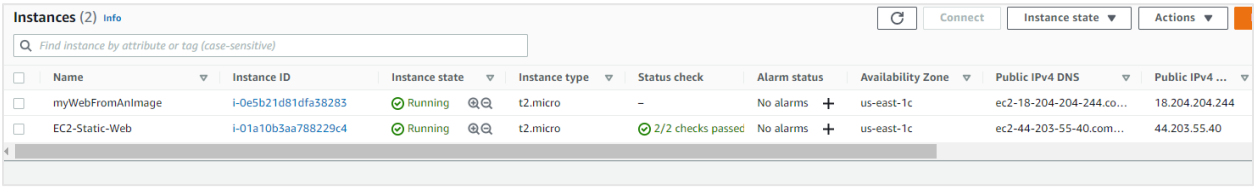
1. Create EC2 instance. Install Apache2 and other required software.
2. Check if the installed software services are running fine and default apache2 page displays.
3. Check if you have installed mysql and its running fine and will be able to connect.
4. Run a sample static web app on EC2
5. Create an AMI from existing EC2. Amazon Machine Image (AMI) provides a Template that can be used to define the service instances.



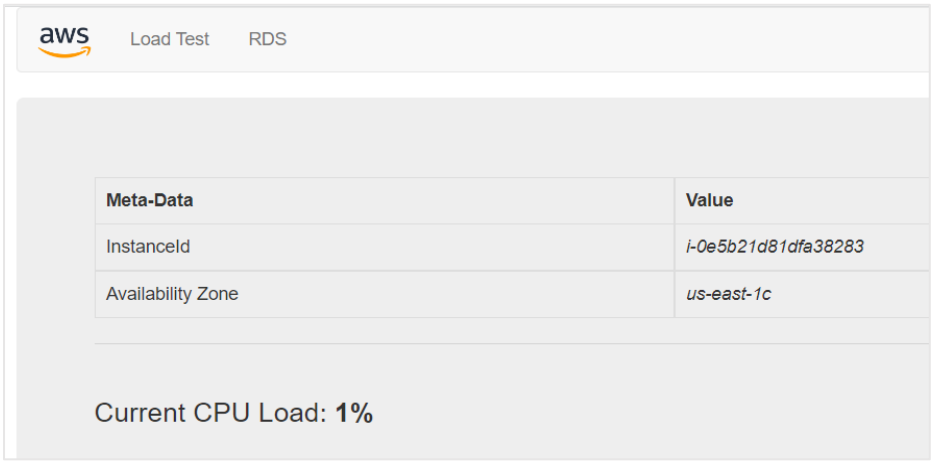
Below is the list of AMI created.



Create an instance from the saved AMI



Access the web application to test if the created instance is running with webserver.



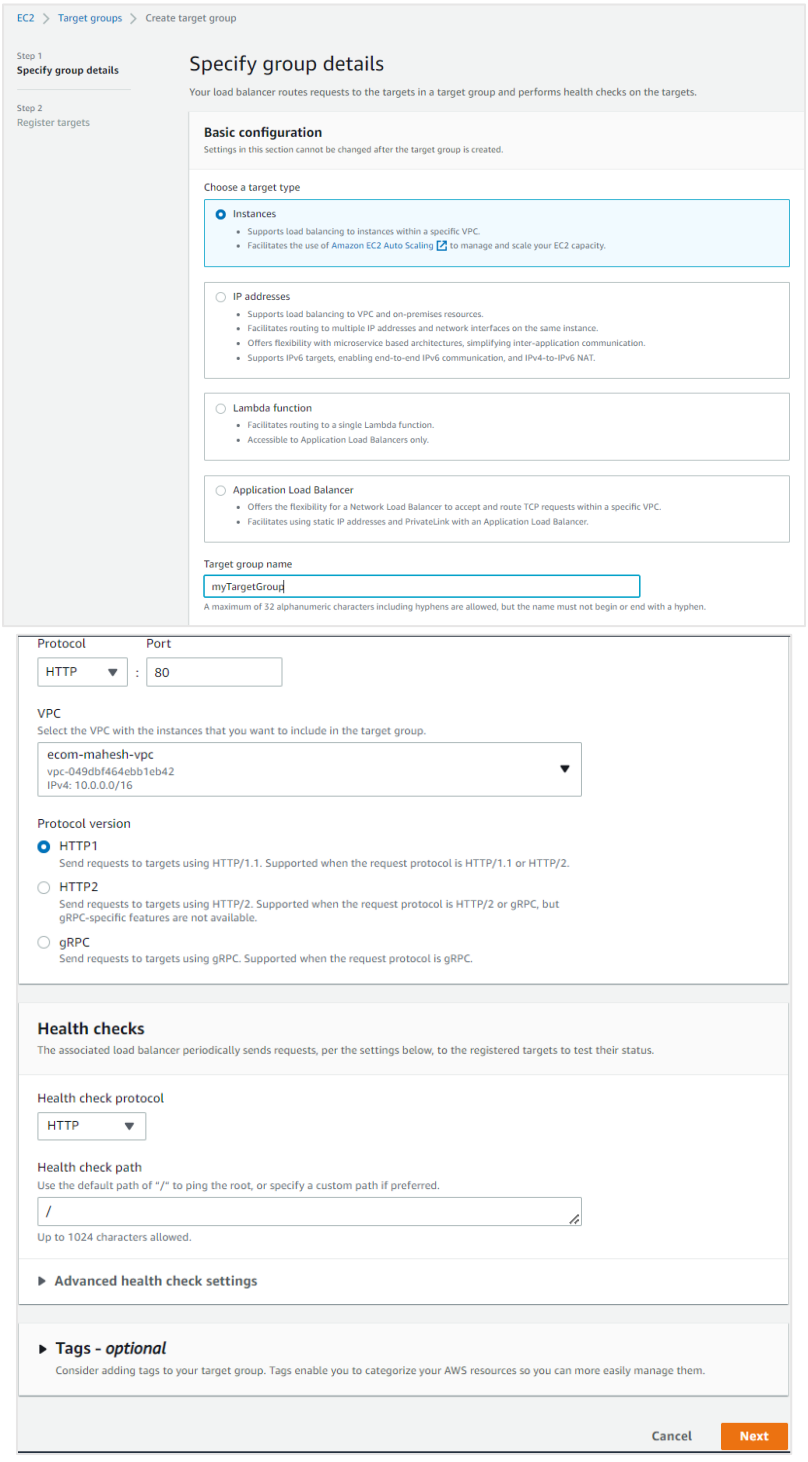
Note: *Now we use the AMI and start with creation of target group, ELB, Launch config and finally auto scaling group.*

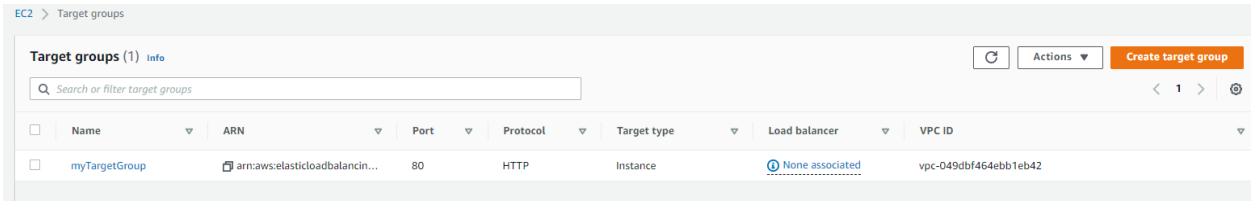
Create a VPC spanning across 2 Availability Zones

This VPC should have 2 public and 2 private subnets

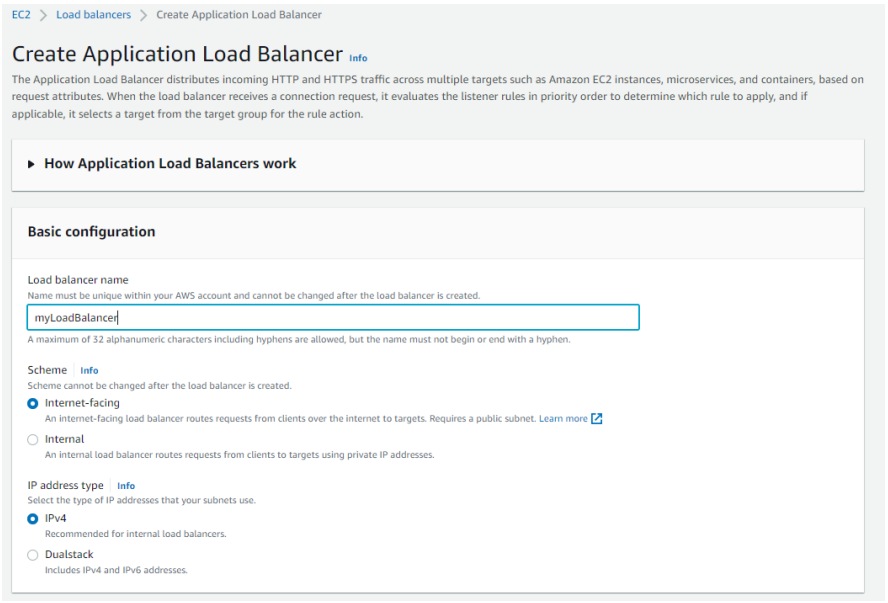
1. Click Subnet and create your Subnet with:
2. Public Subnet 1 and Public Subnet 2 valid Name & VPC.
3. Valid Subnet range which is valid IPv4 CIDR Block.
4. Repeat steps 2 & 3, with Private Subnet too.

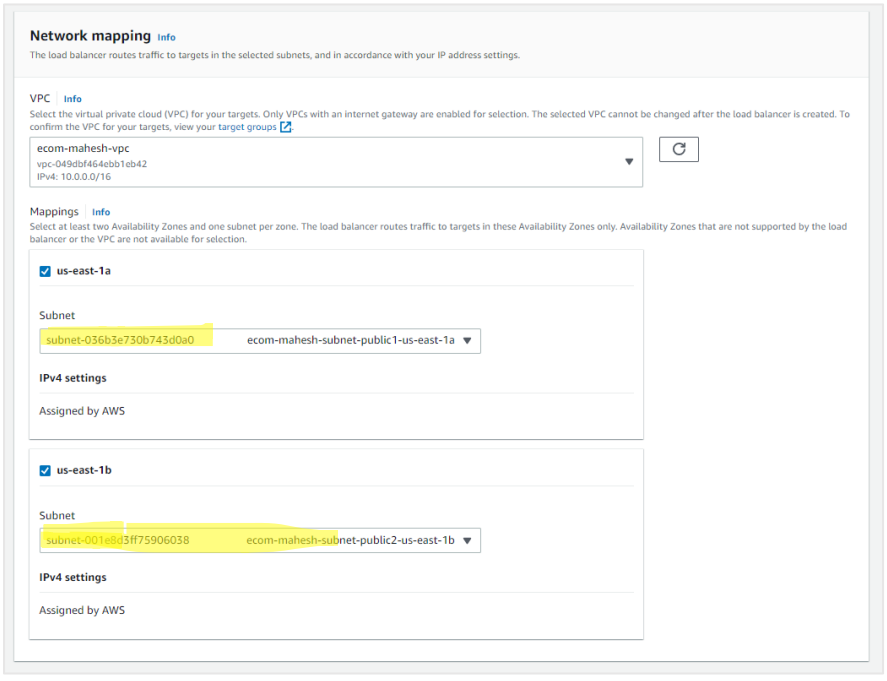
Create a Target group for the ELB

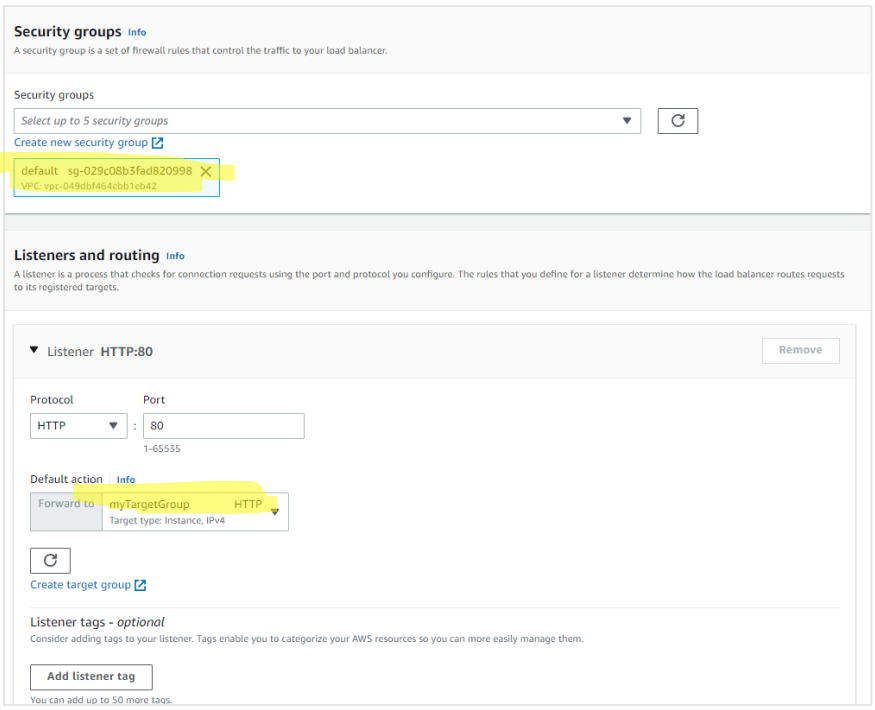


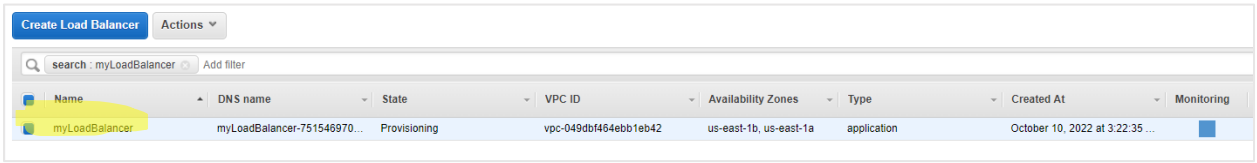


Create Load Balancer ELB, associate with Target Group.

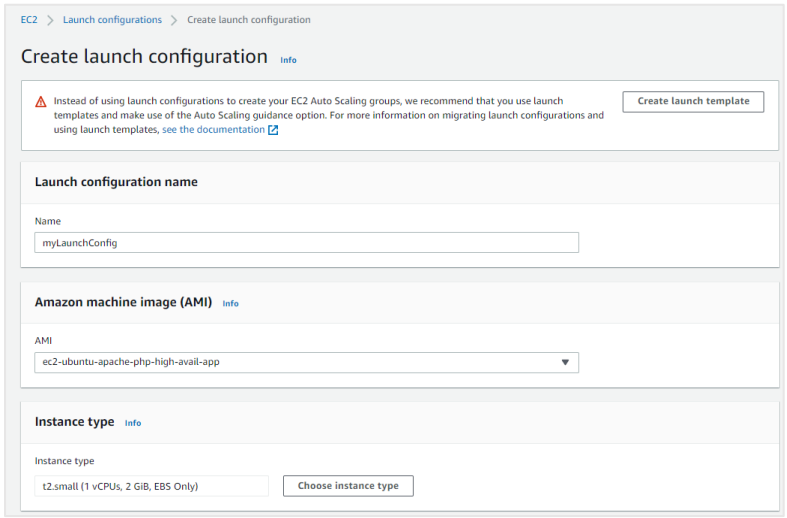


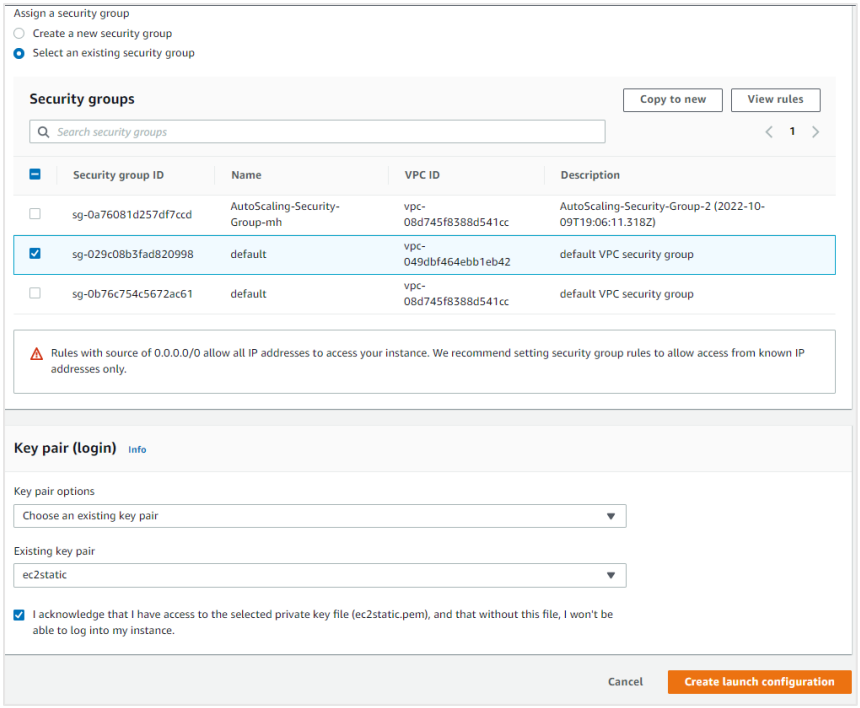


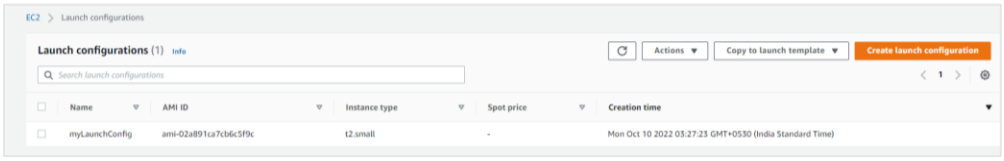




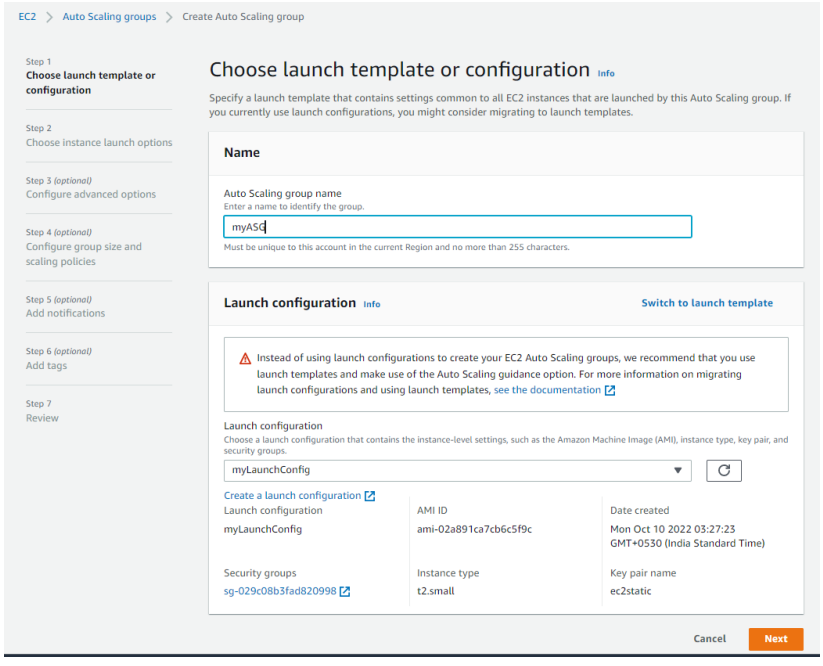
Create Launch config



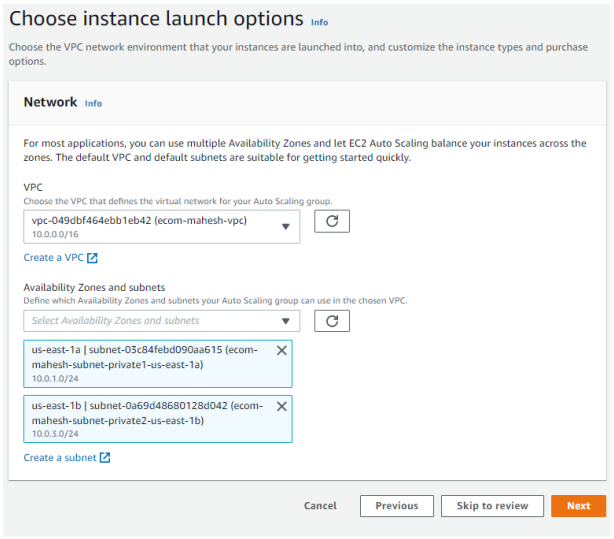




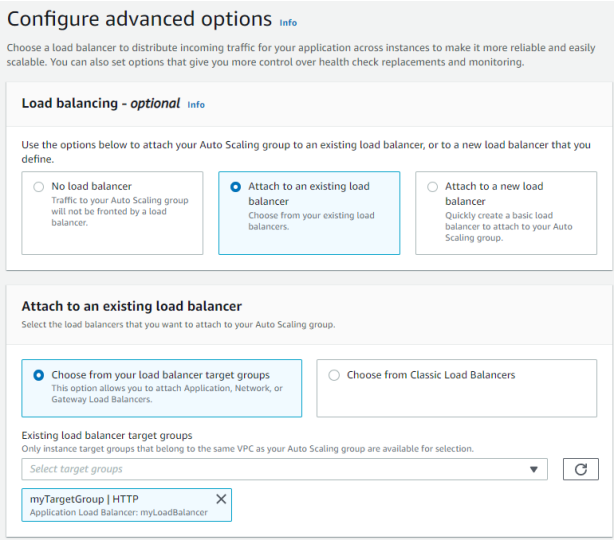
Create Autoscaling Group (ASG) using Launch config. Selected the create ASG from drop down.



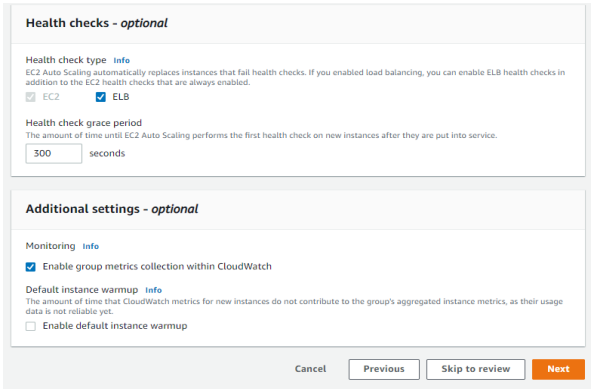
From the subnet drop down select the private subnets



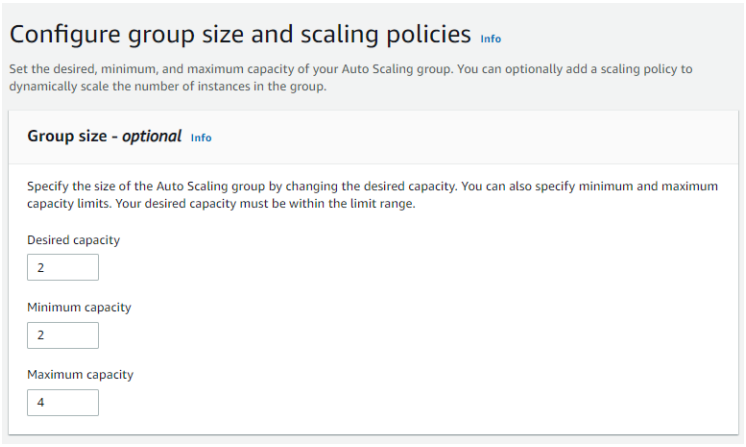
Select the existing load balancer and select the target group.

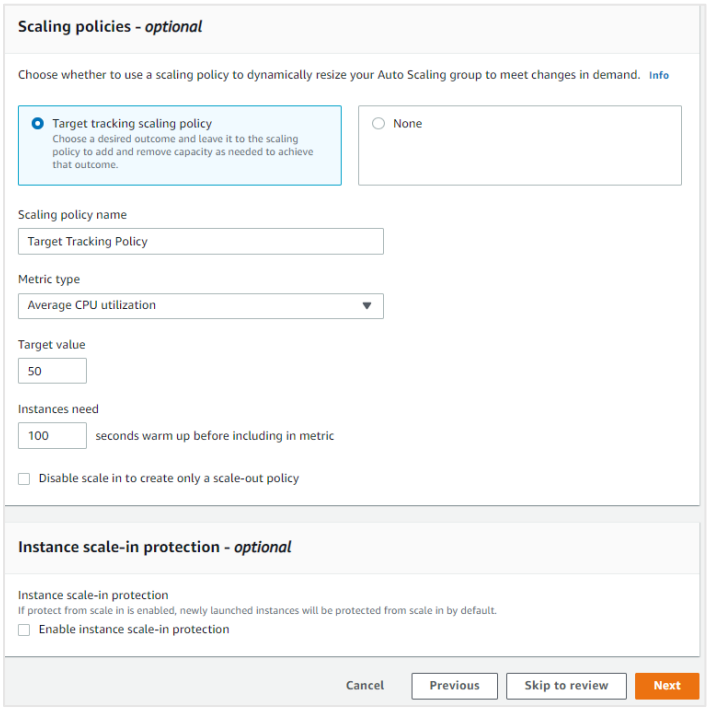


Health check gives the heart rate of the application. ELB checks the webserver is up and running every 300 seconds.

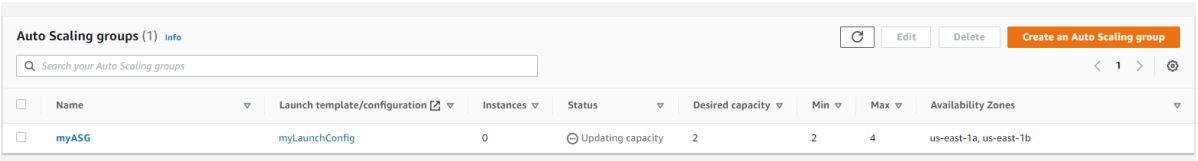


Below is the important step which define the resources group size for auto scaling based on the scaling policy.

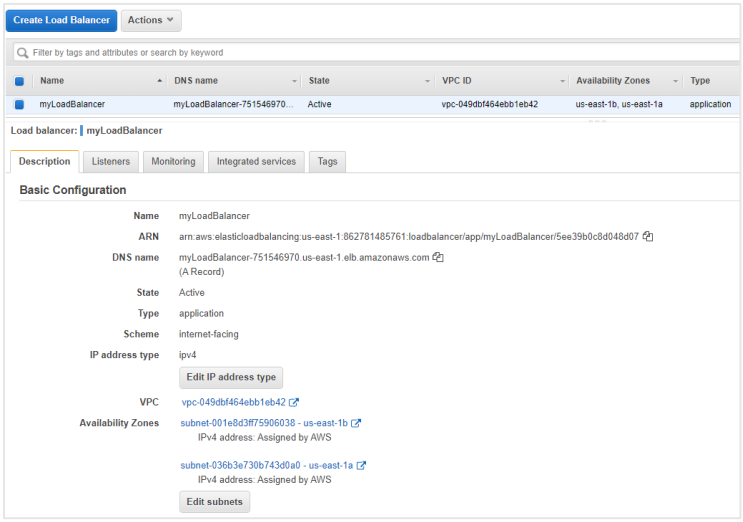




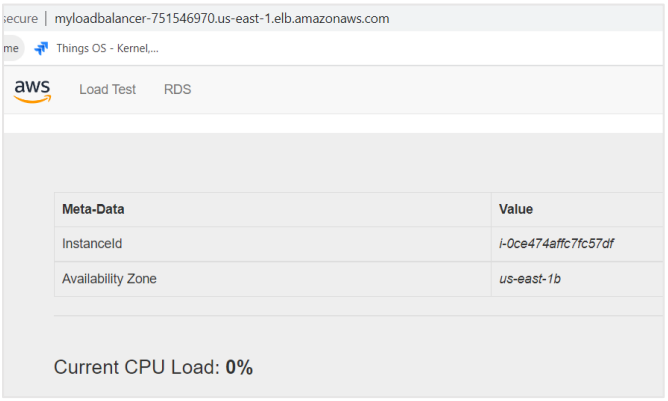
Click next and review and finally create ASG



Next open Load Balancer (ELB) description and copy the DNS name.

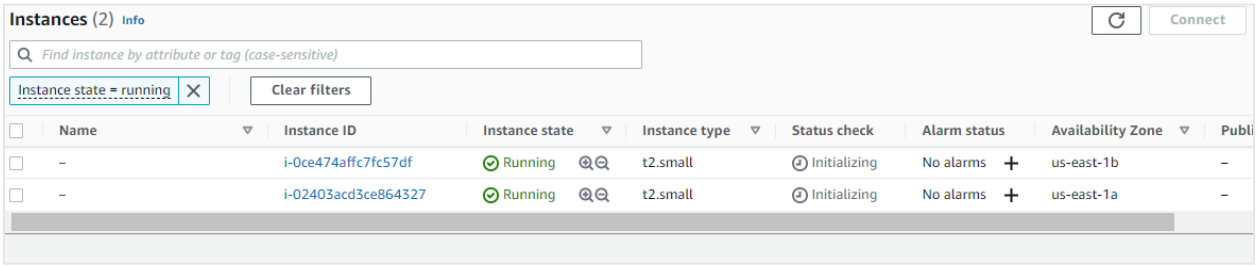


Open the same DNS in browser to verify if the webservice is running

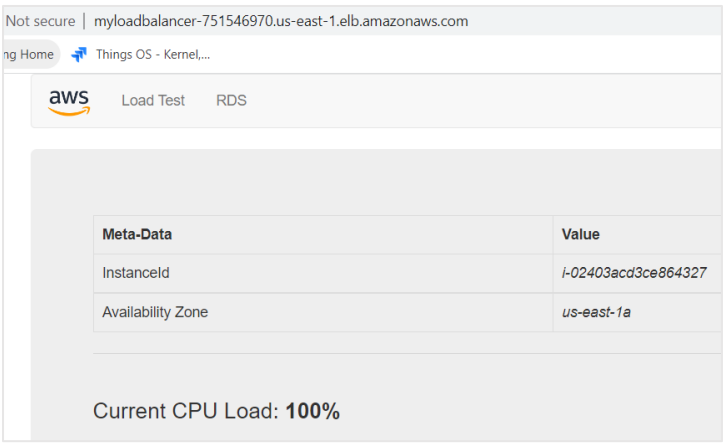


To perform the load testing on the load balancer run the application in the browser.

Keep refreshing the page and observe if we are getting different instance ID of EC2. Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones.



Once the server CPU goes above the mentioned threshold, it creates instances for the load balancing.



Once we start performing the load testing, if the CPU usage crosses the average value defined it creates new instance to serve the load of the application.