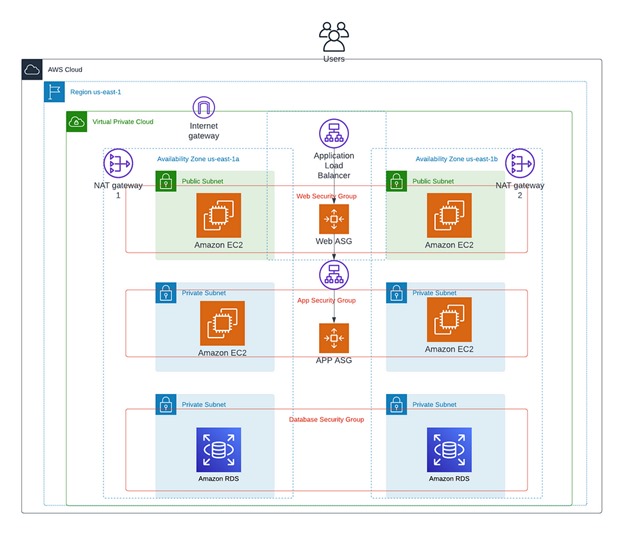
3-Tier Architecture for Web Applications in AWS



AWS provides a wide range of resources for developing and managing cloud applications, which can be customized to construct highly dependable and resilient cloud infrastructures. Suppose you are tasked with developing a three-tier architecture that is readily available for your organization’s new web application. This tutorial is extensive but comprehensive. You may want to bookmark this guide for future reference on creating web, application, and data tiers.

**What is a 3-TierArchitecture?**

A three-tier architecture comprises three layers, namely the presentation tier, the application tier, and the data tier. The presentation tier serves as the front-end, hosting the user interface, such as the website that users or clients interact with. The application tier, commonly referred to as the back-end, processes the data. Finally, the data tier is responsible for data storage and management.

**Benefits of a 3 Tier-Architecture:**

**Scalability:** Each tier can scale independently, allowing organizations to optimize their resources and minimize costs.

**Reliability:** Each tier can be replicated across multiple servers, improving application availability and reliability.

**Performance:** By dividing the application into separate layers, 3-tier architecture reduces network traffic and enhances application performance.

**Security:** Each tier can have its own security group, allowing different organizations to implement customized security measures for each layer.

**Reduced development time:** Different teams can work on different tiers simultaneously, resulting in faster deployments.

**Flexibility:** Each tier can be developed using different technologies and programming languages, enabling organizations to leverage the best tools for each layer.

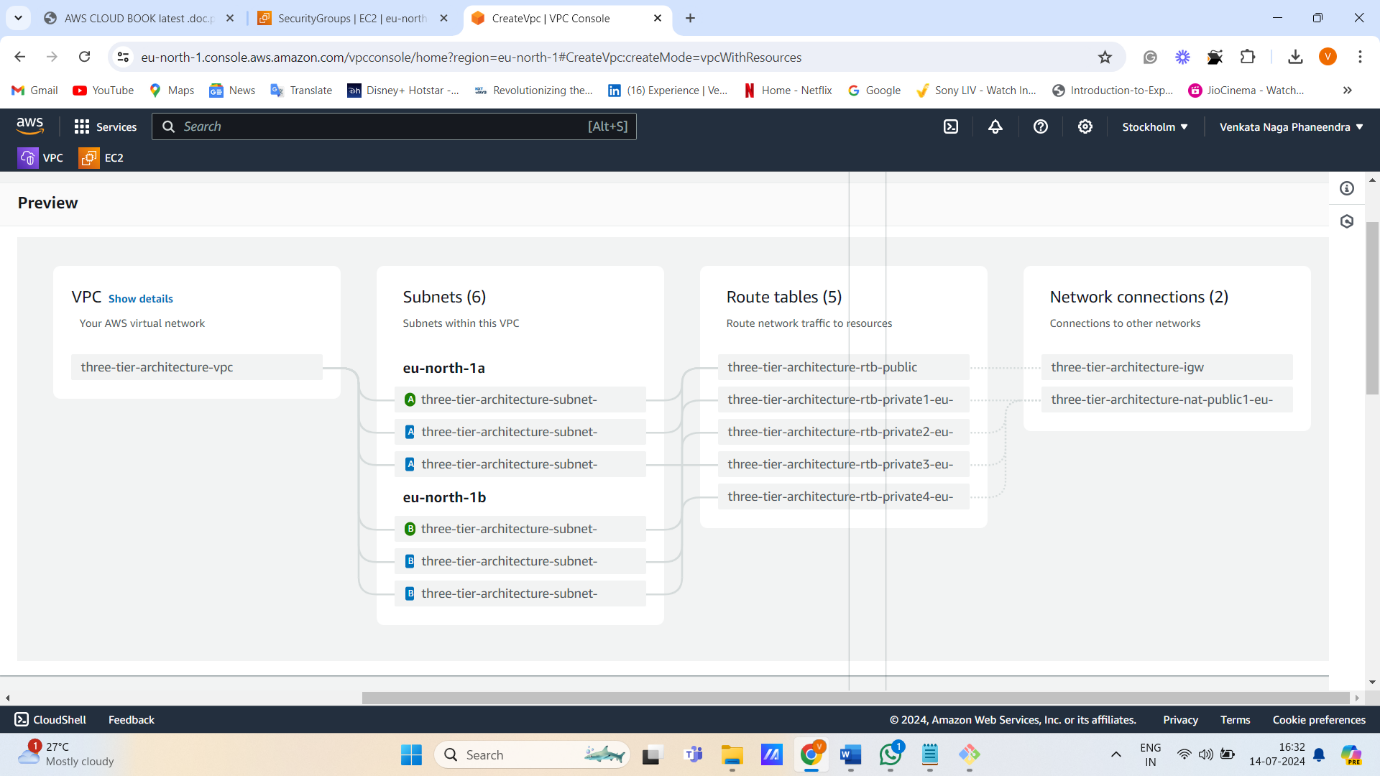
**PART – 1: WEB TIER OR USER INTERFACE**

* Create a VPC with multiple public and private subnets, availability zones, and more

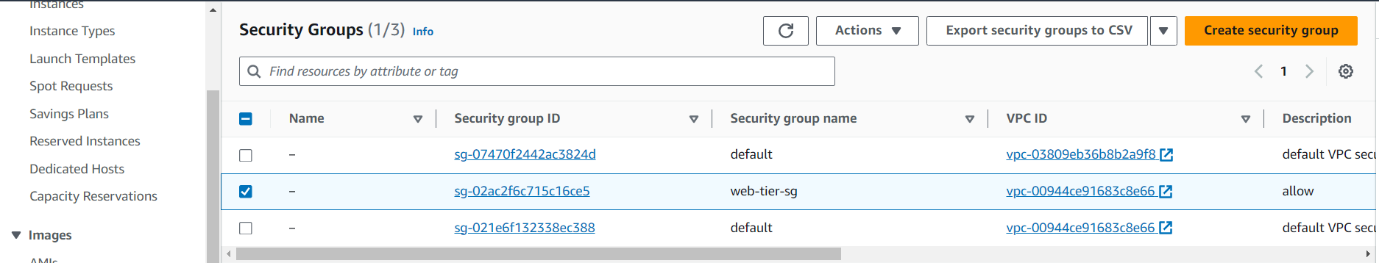
**.** 2 Availability zones

**.** 2 Public subnets (Internet – gateway)

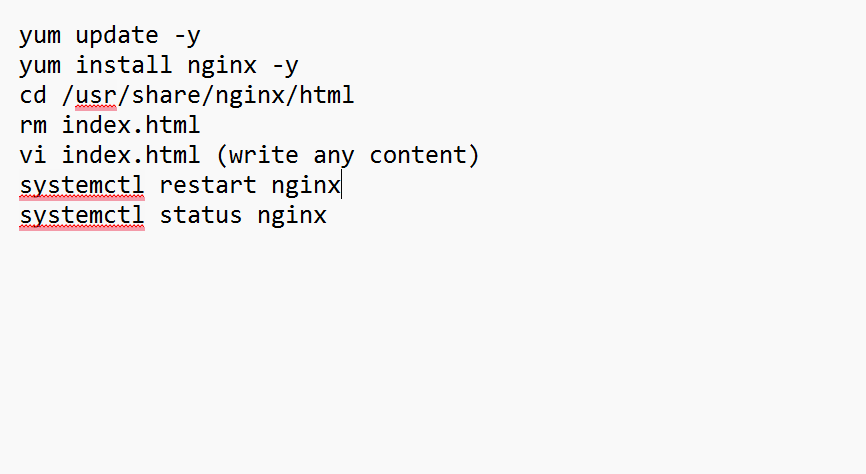
**.** 4 Private subnets (NAT – gateway)



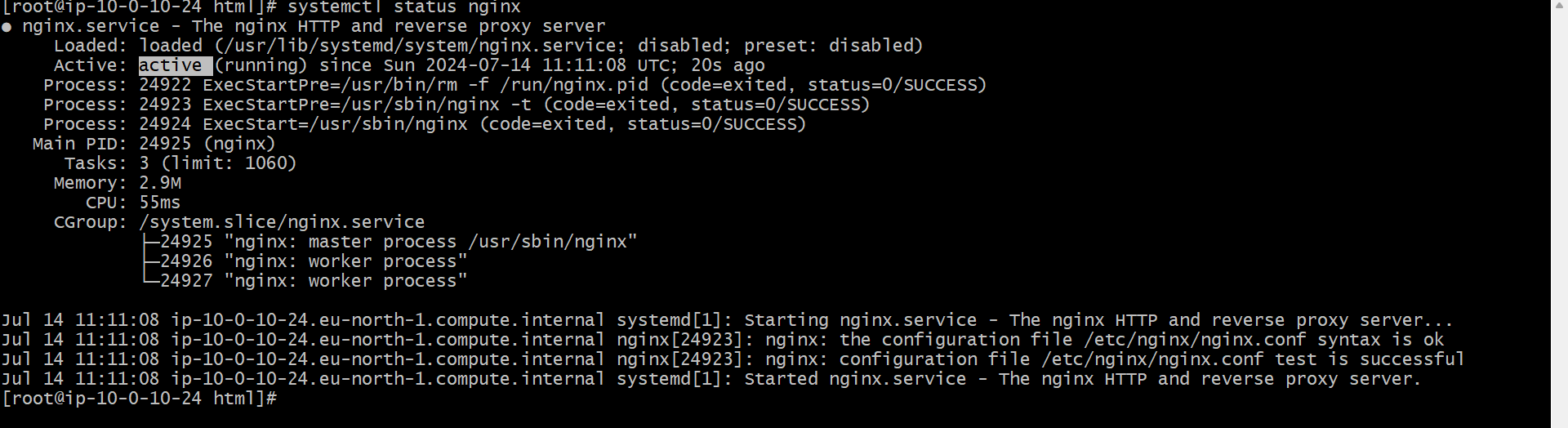
* Create a security group by giving inbound rules as SSH (port:22) and HTTP (port:80). This is for web tier (user interface)



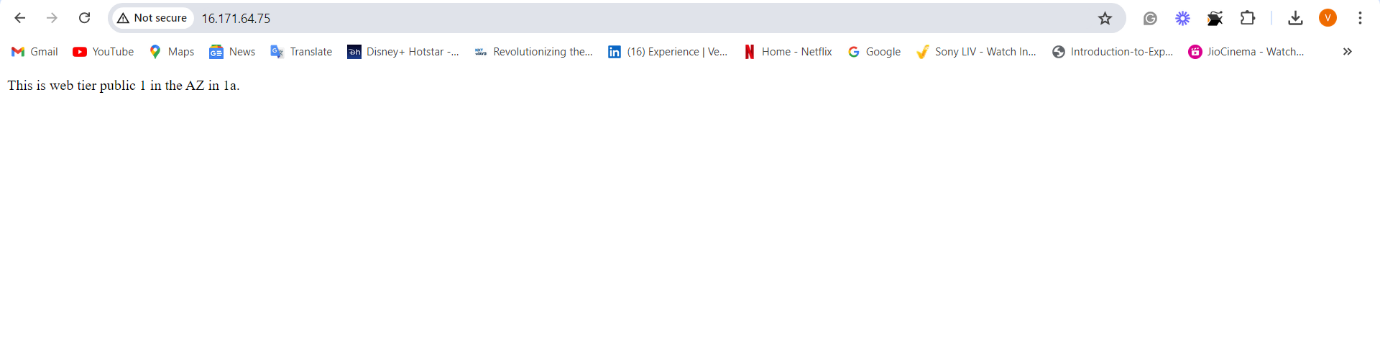
* Launch 2 instances in each availability zone by giving existing vpc & subnets, existing security group.
* After launching instances connect to the server using ‘Git-bash’ and run the following commands to connect to the web browser (nginx) of port: 80.



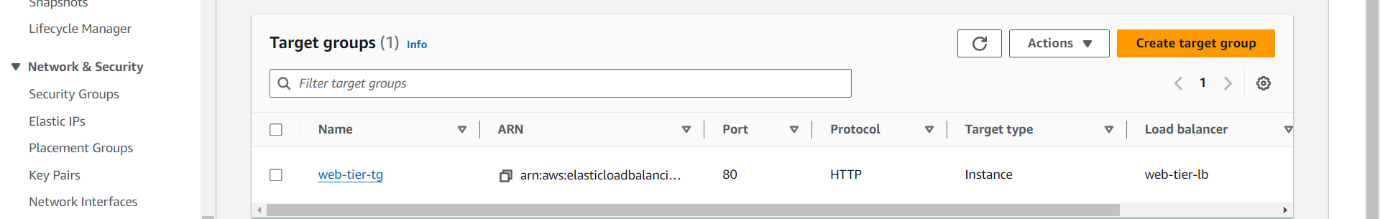
After connecting to the nginx browser, we have to check the status of the browser.



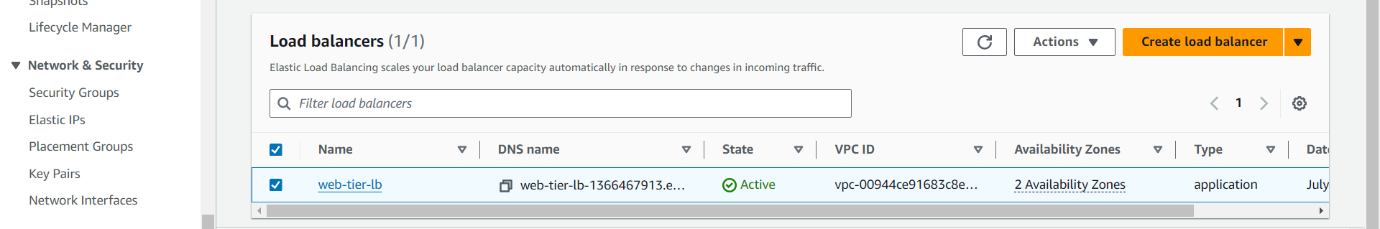
Copy and paste the public ip of both public instances in the browser with port:80, to see the information in the index.html of the nginx browser.



* Create a Target group by giving existing vpc, registering targets as instances, and include, include as pending below.

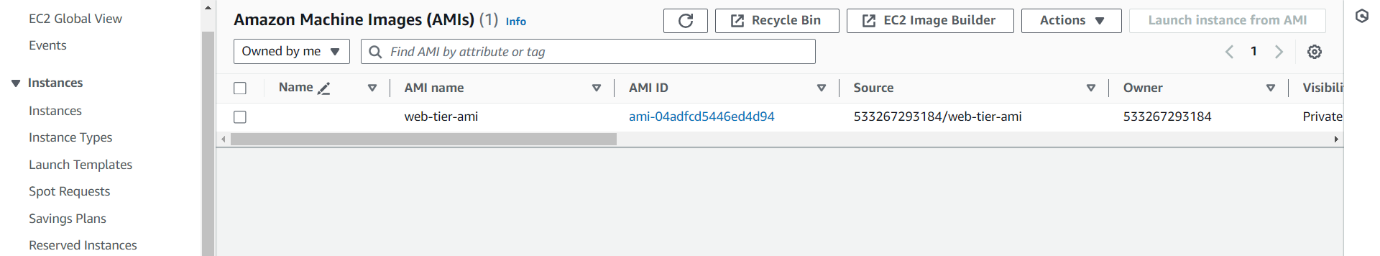


* Create an Application Load balancer by mapping subnets, attaching security groups, and target group

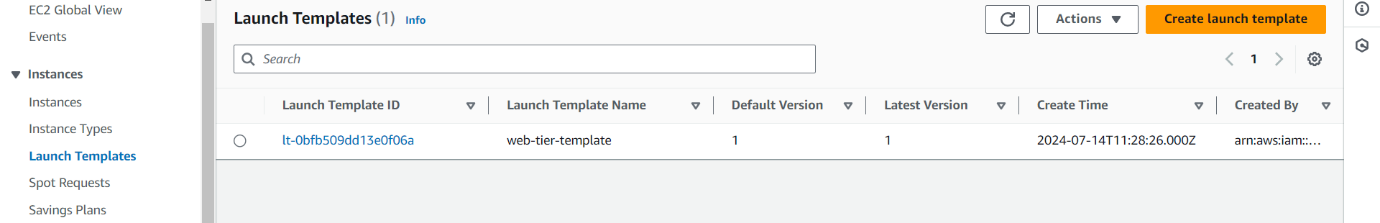


Copy DNS and paste it in the browser, to see load is equally distributed or not, if the load is equally distributed, then when we refresh the browser, we see pieces of information of index.html of nginx in each server.

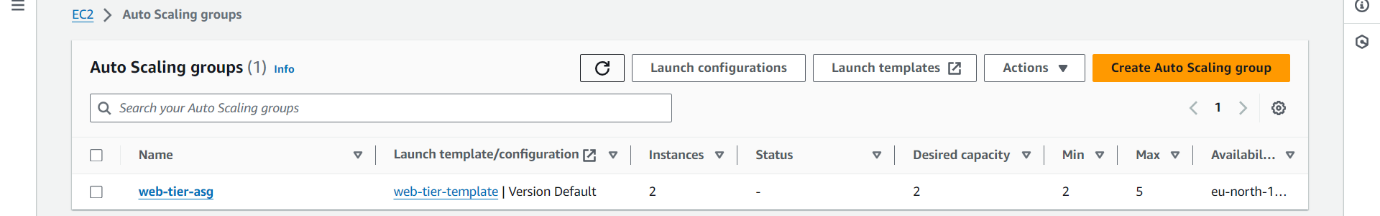
* Create an Amazon Machine Image (AMI) by selecting an instance



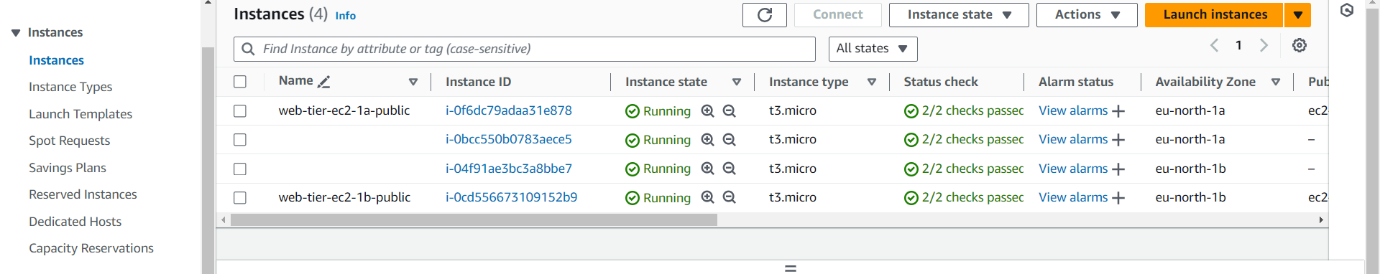
* Create a Launch Template by not including any subnets and attach AMI (Which is created before)



* Create Auto Scaling Group by attaching the launch template, and existing load balancer, configure group size, scaling (min, max desired capacity), and average CPU utilization’s target value

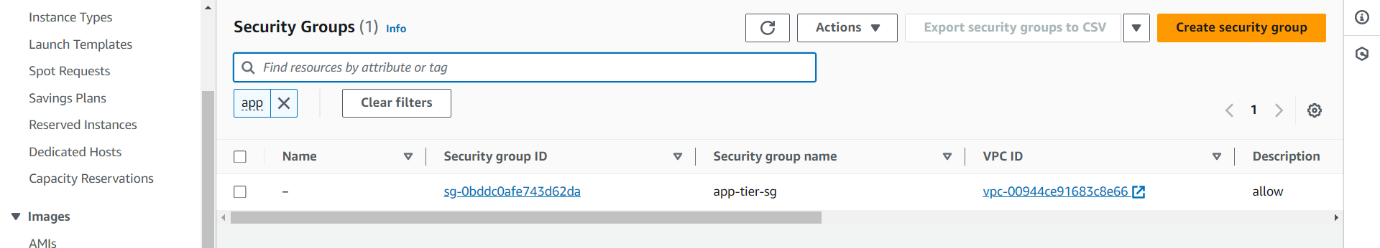


After creating ASG, we can see that additionally two more instances were created because of auto-scaling.

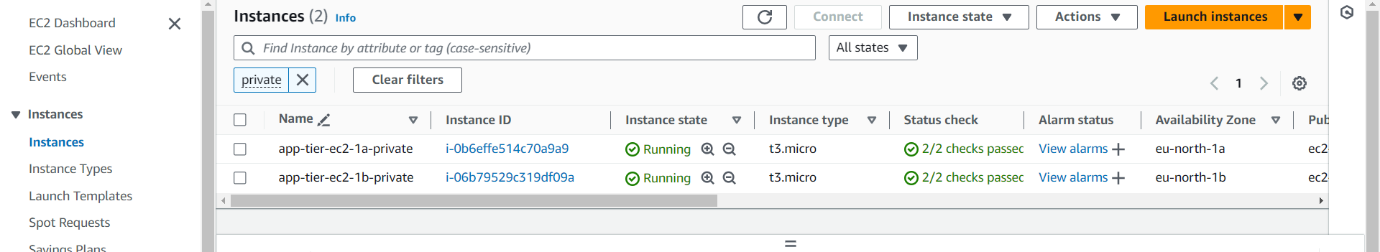


**PART – 2: APPLICATION TIER OR BUSINESS LOGIC TIER**

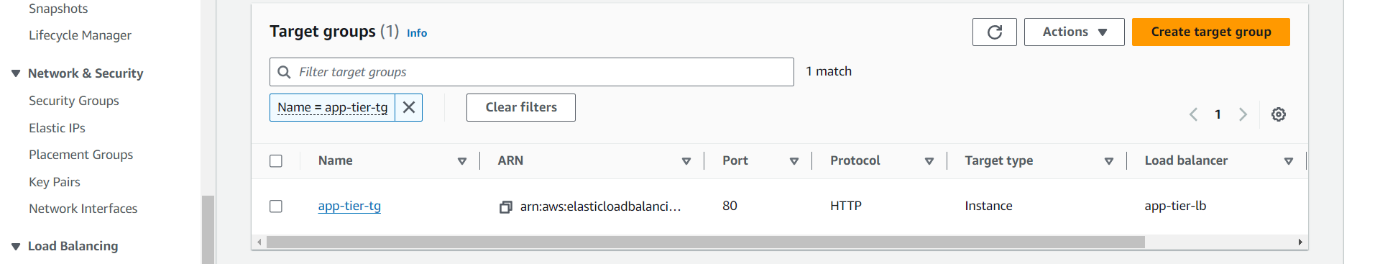
* Create a security group separately for this app tier



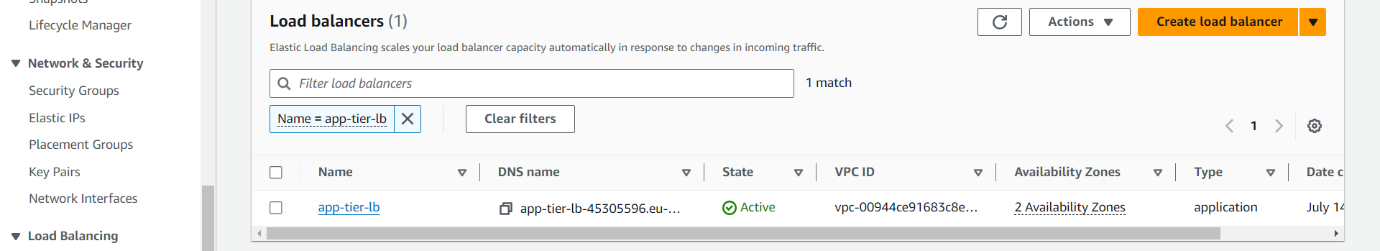
* Launch 2 instances by attaching 2 private subnets in the different availability zones, and existing security group.



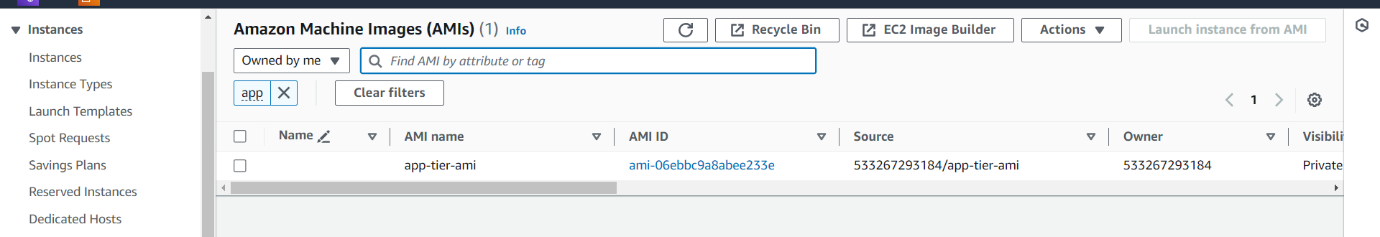
* Create a Target group by giving existing vpc, registering targets as instances, and include, include as pending below for application tier



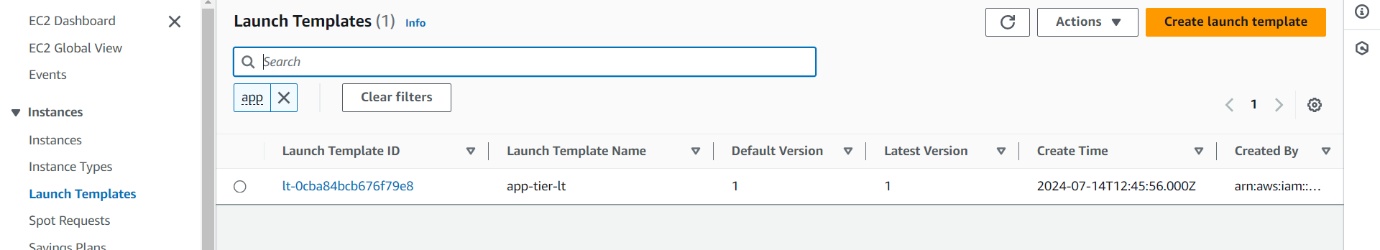
* Create an Application Load balancer by mapping subnets, attaching security groups, and target group



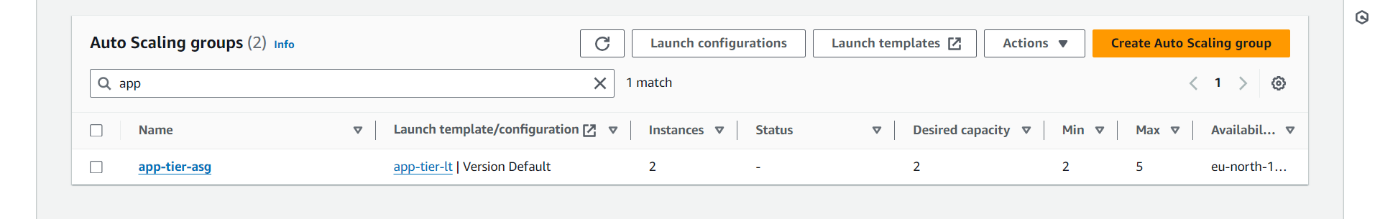
* Create an Amazon Machine Image (AMI) by selecting an instance



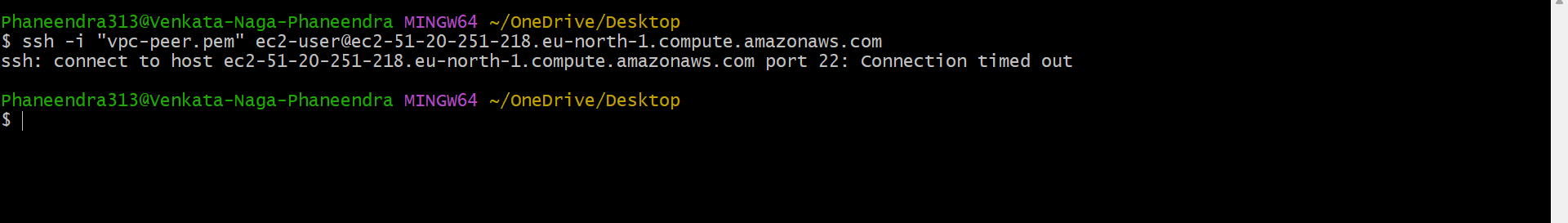
* Create a Launch Template by not including any subnets and attach AMI (Which is created before)



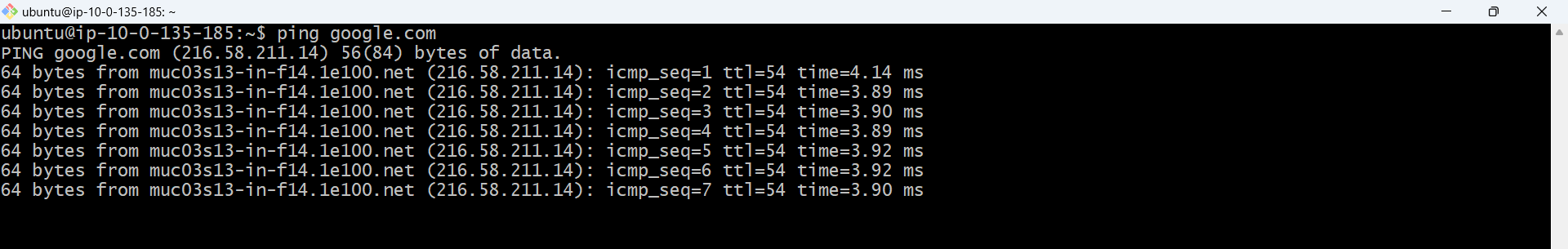
* Create Auto Scaling Group by attaching the launch template, and existing load balancer, configure group size, scaling (min, max desired capacity), and average CPU utilization’s target value



When I tried to SSH into the application tier EC2 instance running, the connection timed out, this is exactly what we want here.

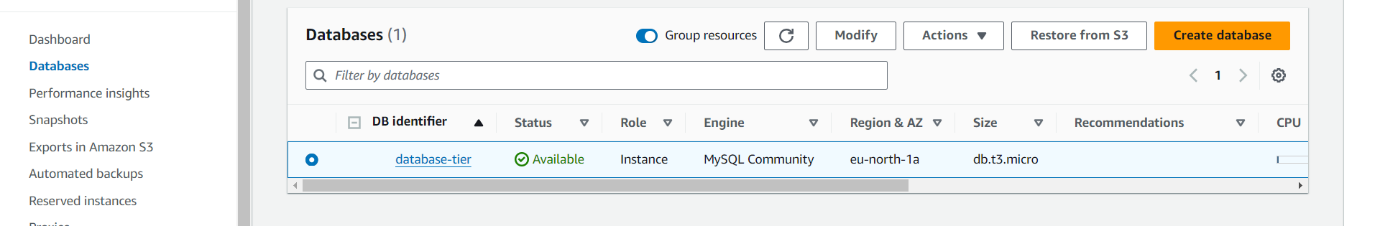


We still need to check if our tier-1 servers interact with our tier-2 servers. To test this, you will need to log into your tier-1 EC2 instances via SSH and run a ping command to a private IP address of our tier-2 servers. Below you can see a successful ping.



**PART – 3: DATABASE TIER OR STORAGE TIER**

* Go to RDS, create a database by configuring engines, vpc, security group, and set up ec2 connection by connecting to public instance of any AZ.



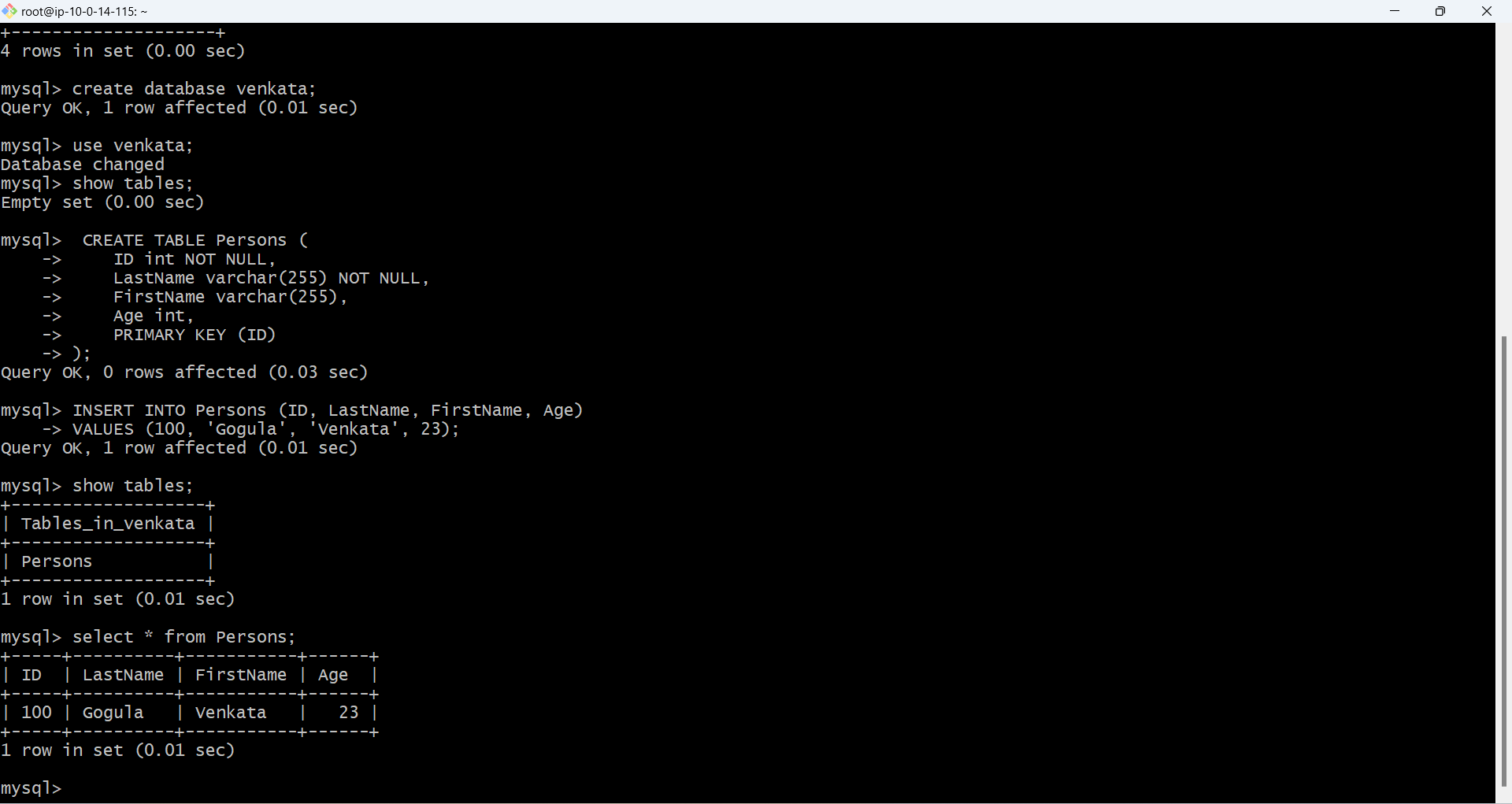
* Run the below commands, after connecting to the public instance through git-bash to install mysql in the server.



* To connect the mysql to the server and create database, create tables and read database, run the below commands



* After running the above commands, we see the database is connected to the server.



We have successfully connected the database to the web server.