**AWS SYSOPS**

**Case Study Documentation**

**Prepared by**

**Balaji Srinivasan**

**Business requirement:**

* A New Enterprise wanted to go with modern IT solution for their new branch They are new, not sure of scale and wanted to get away from Capex cost
* To reduce the capex cost and scale as the grow, they have decided to go with full cloud-based solution and have decided with Amazon

**Prerequisites:**

* Aws Console login
* Terraform
* Ansible

**Solution:**

**Part 1:**

Create infrastructure with fully automated solution using native tools and also opensource tools to ensure best practice

**Step 1: CloudFormation**

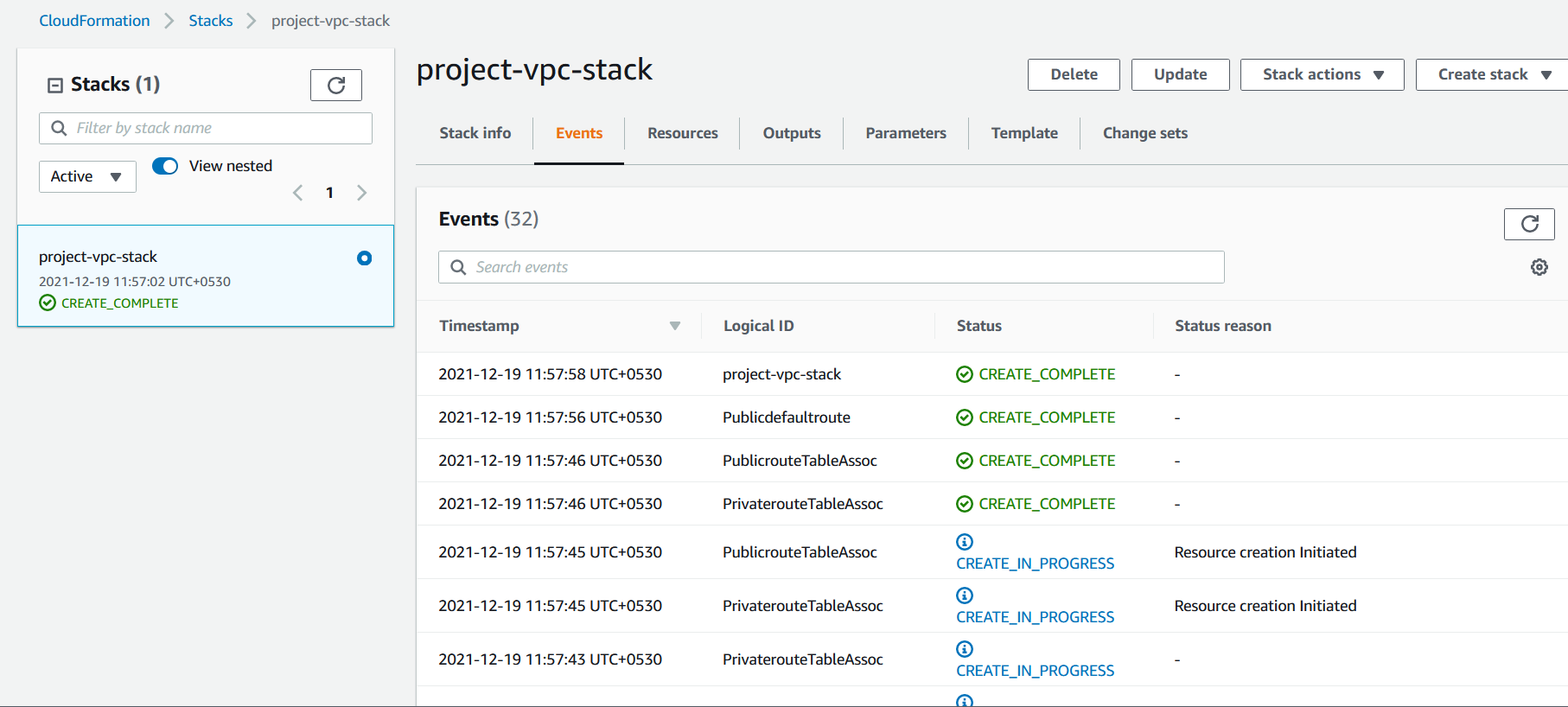
Create VPC and subnets, Internet Gateway and routing table using CloudFormation.

* 1. Login into AWS Credential’s using your login account
  2. Install Microsoft visual studio code using the link <https://code.visualstudio.com/Download>

1.3 Go to cloud formation and create stack (Project-vpc-stack) using below script.

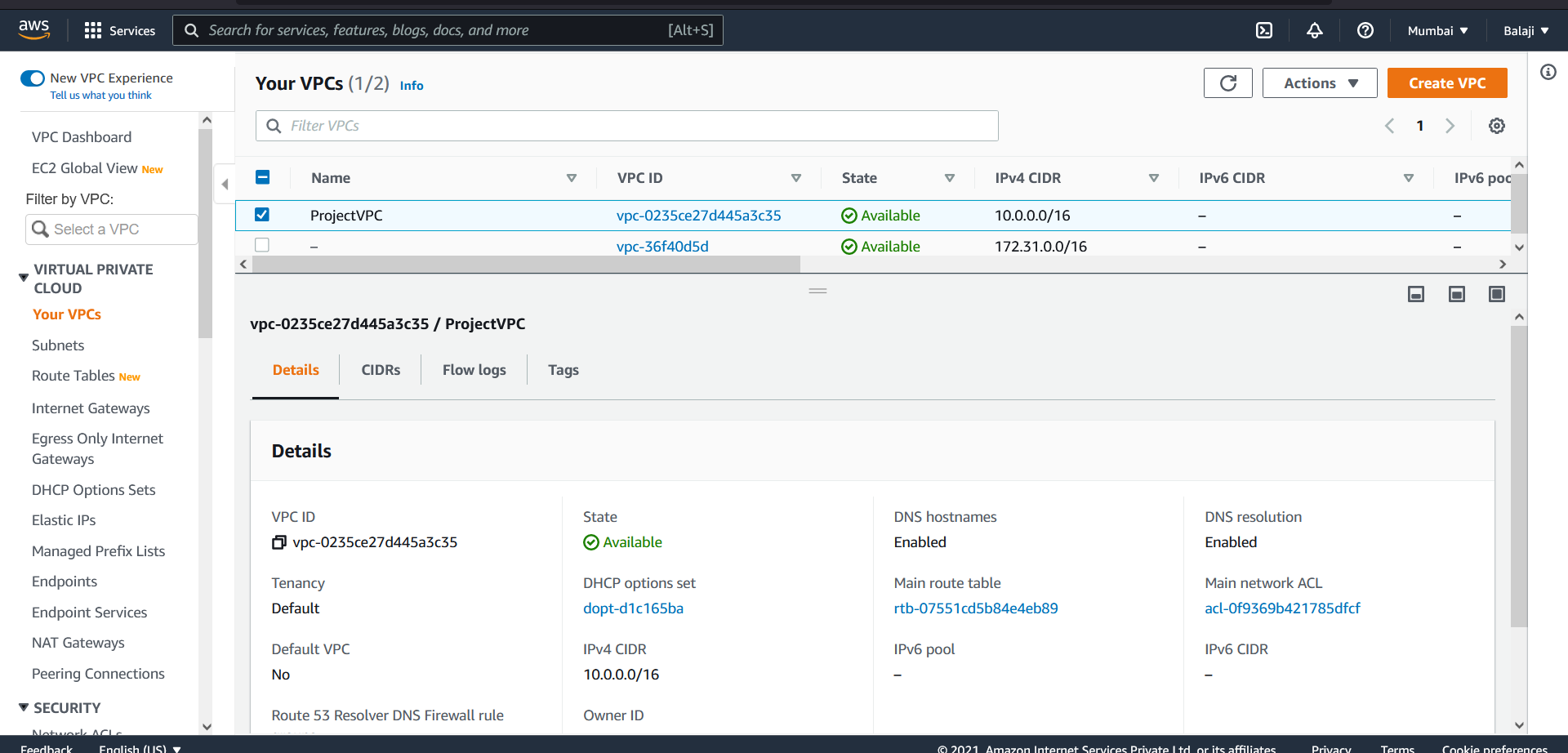


1.4 Once Stack create completes; the below screen shot will be appeared:

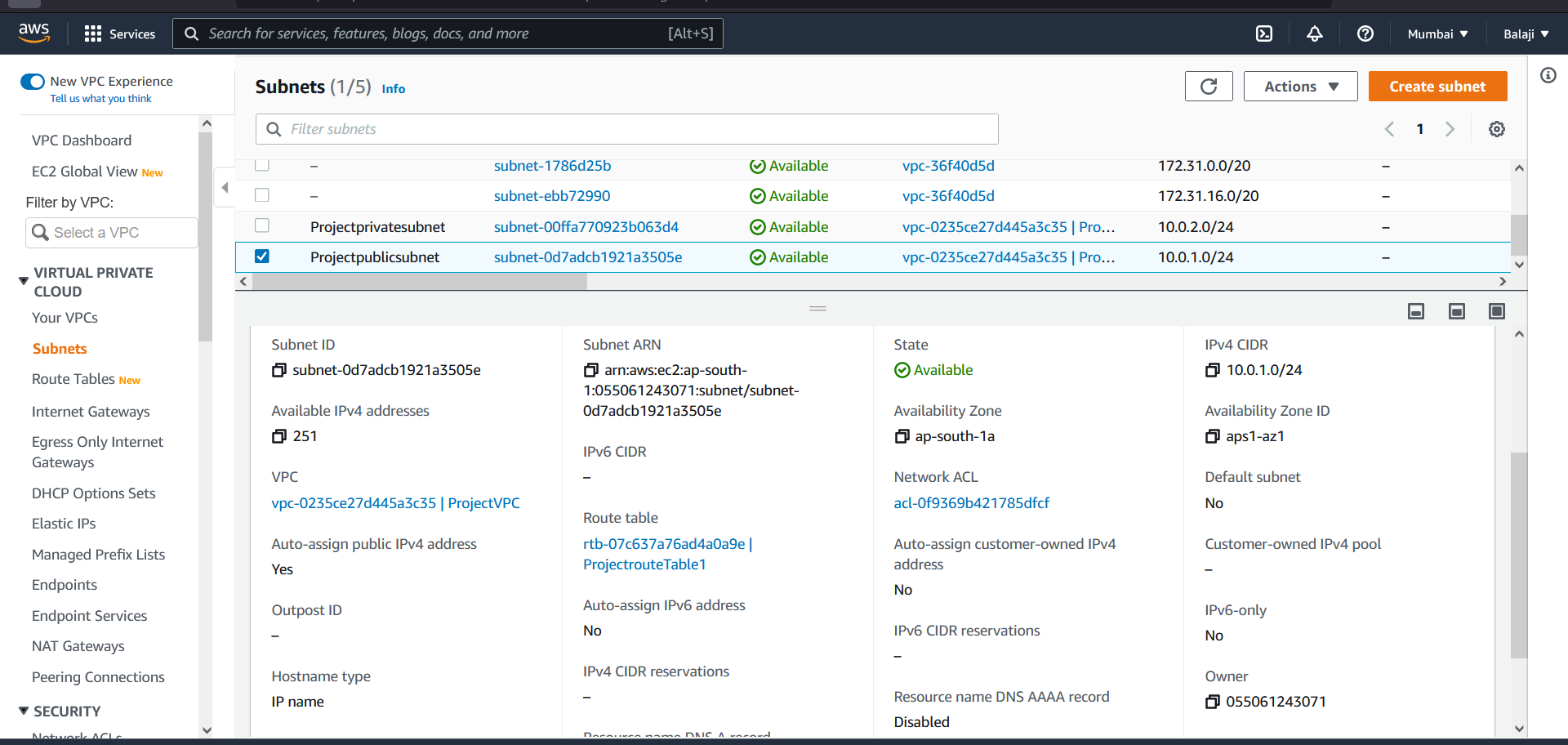


1.5 Now VPC and their resources will be reflected in VPC dashboard

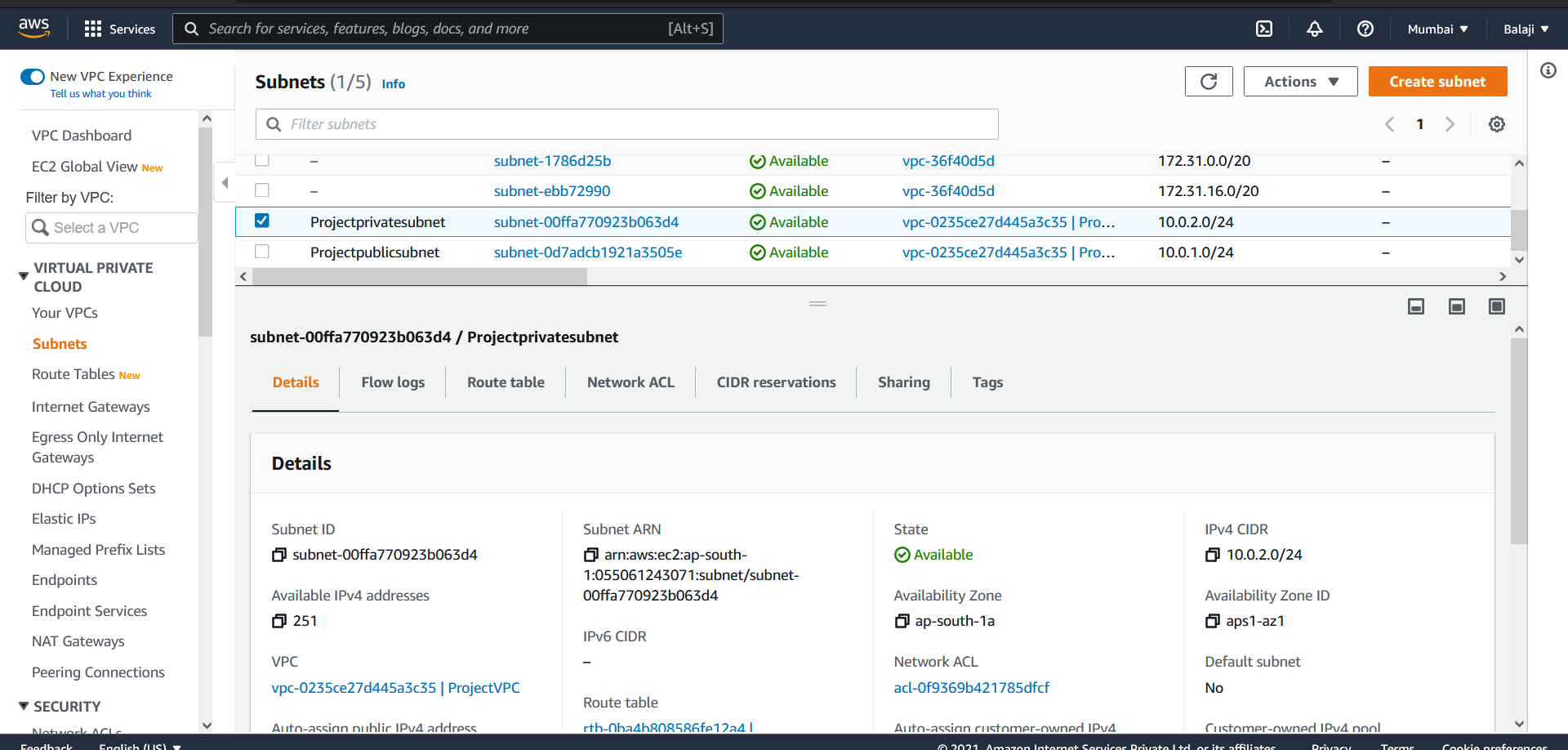
VPC:



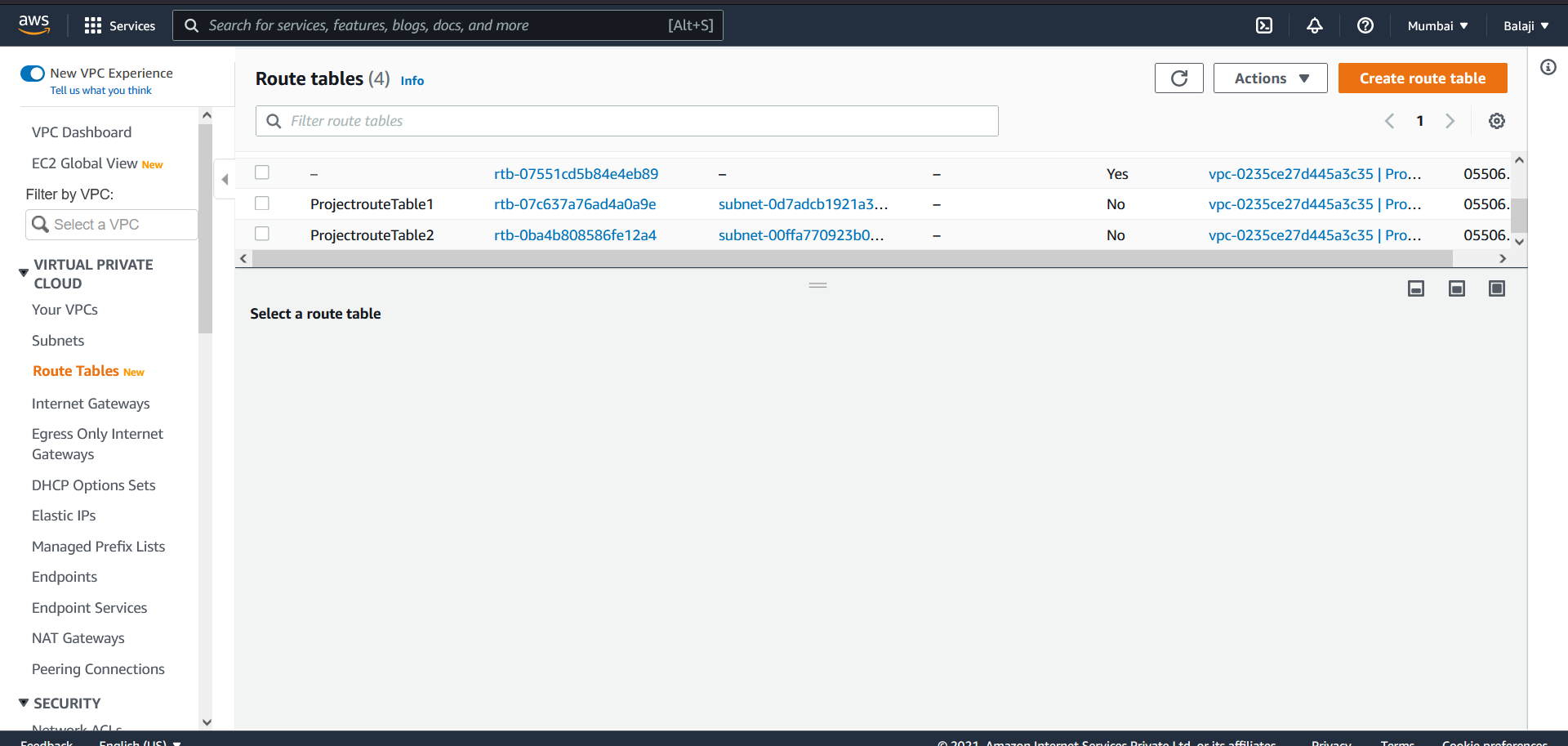
Public Subnet:



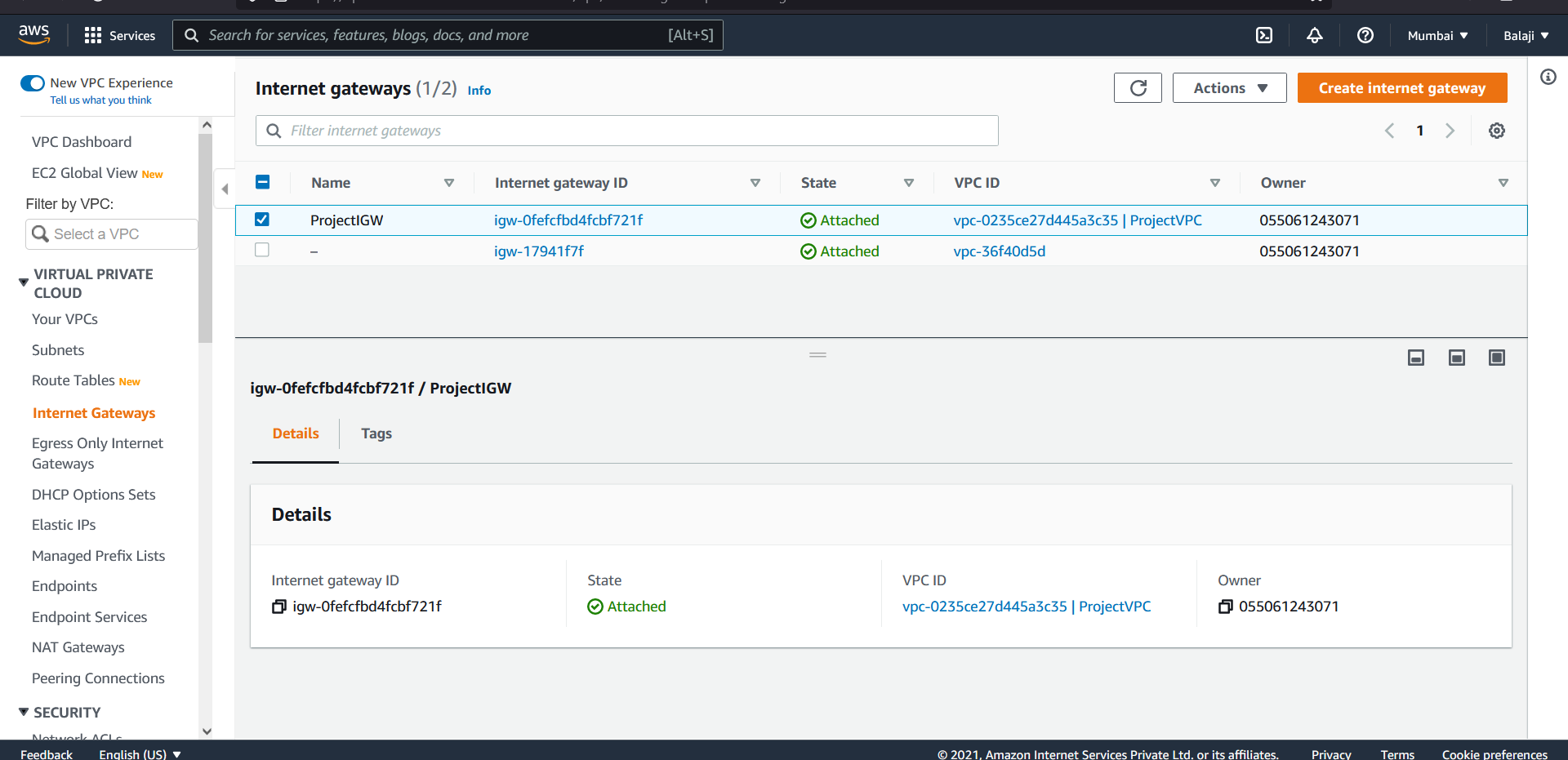
Private Subnet:



Routing Table:



Internet Gateway:



**Step 2: Terraform**

Create Autoscaling group (2 linux instance -ubuntu 18.04) in public subnet frontend with classic (layer 4) load balancer and one linux (ubuntu 18.04) in private subnet using Terraform.

2.1. Install terraform using the link <https://releases.hashicorp.com/terraform/1.0.11/terraform_1.0.11_windows_amd64.zip>

2.2 Install aws-CLI for windows using the link <https://awscli.amazonaws.com/AWSCLIV2.msi>

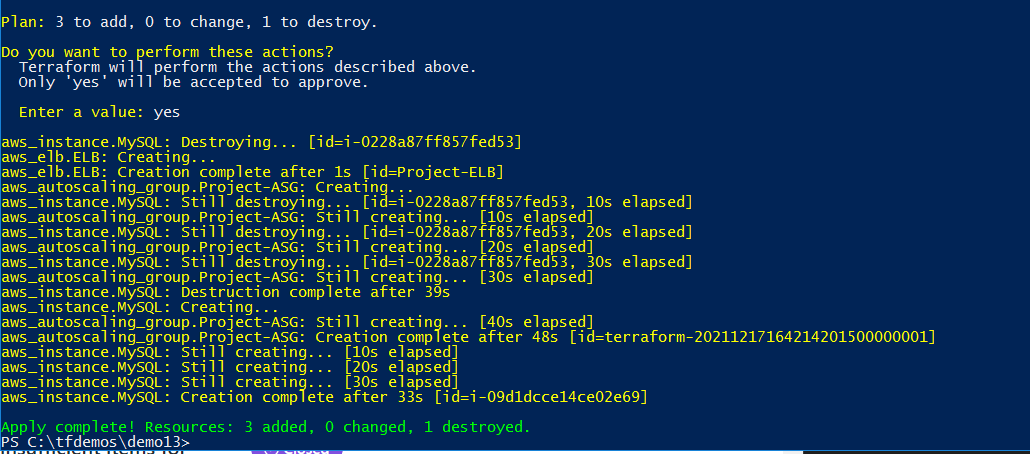
2.3 Create a folder in c: drive and save the below scripts. ( .tf file)



2.4 using windows power shell and run below commands

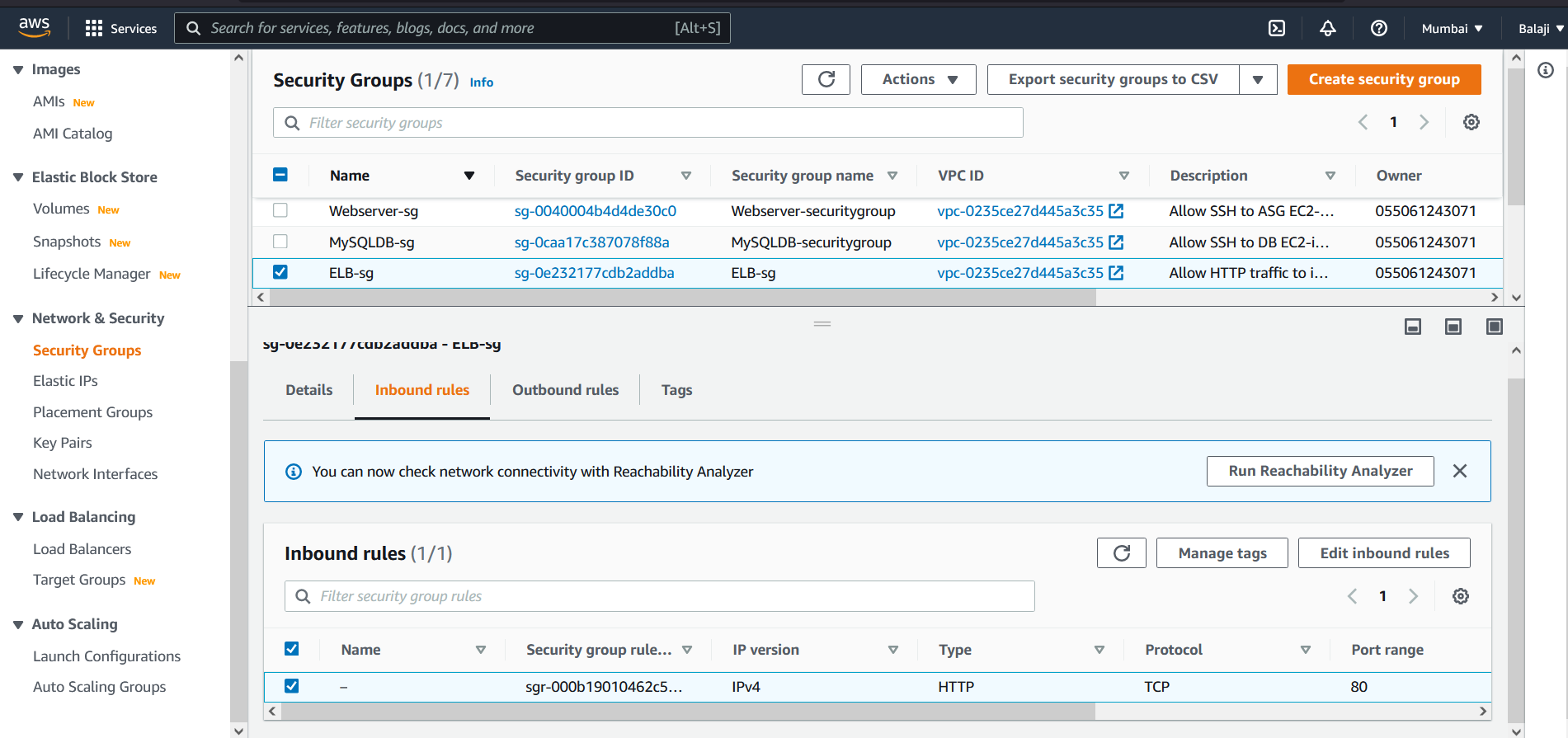
* terraform init
* terraform validate
* terraform plan
* terraform apply

Terraform output:

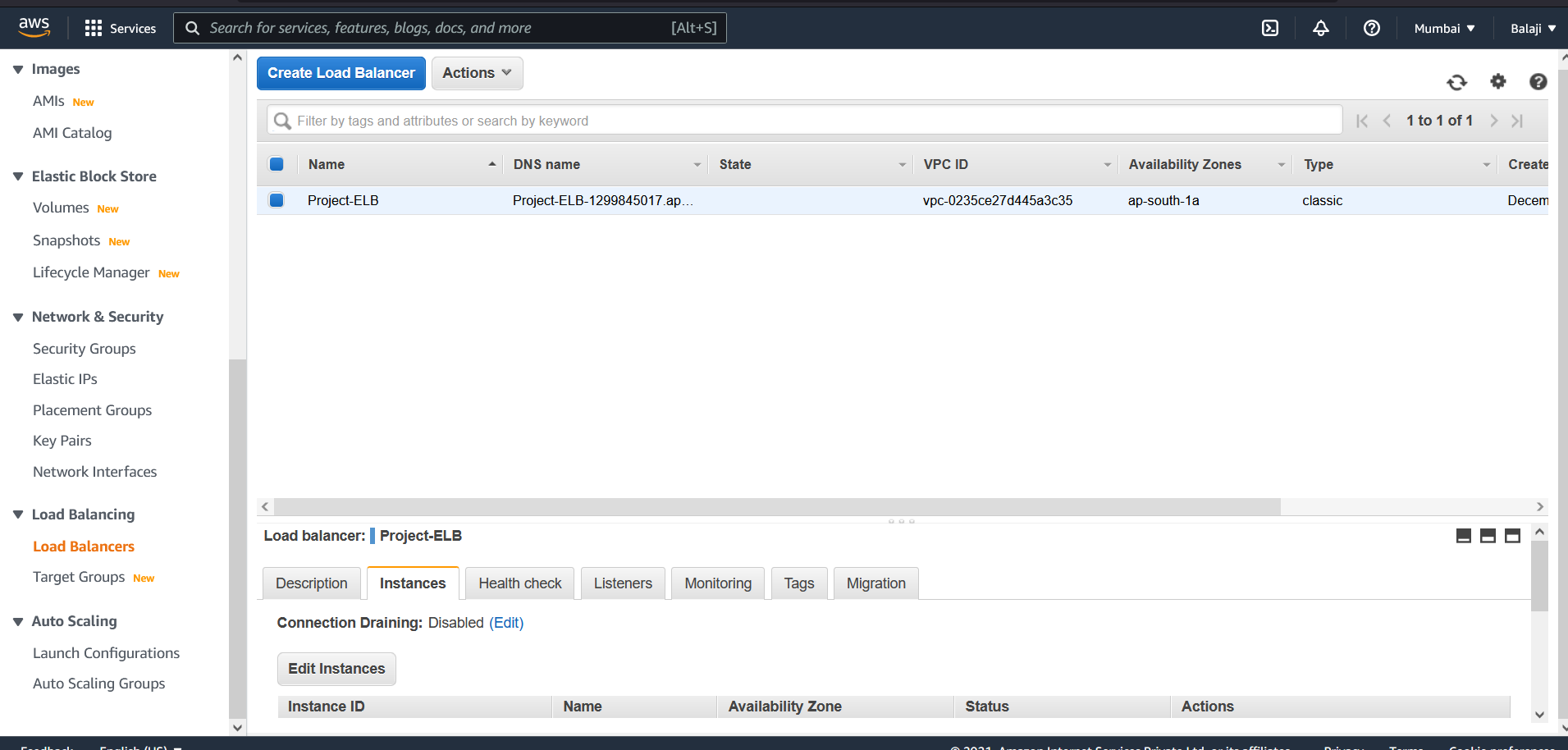


2.5 Once run above commands, resources will be created in aws

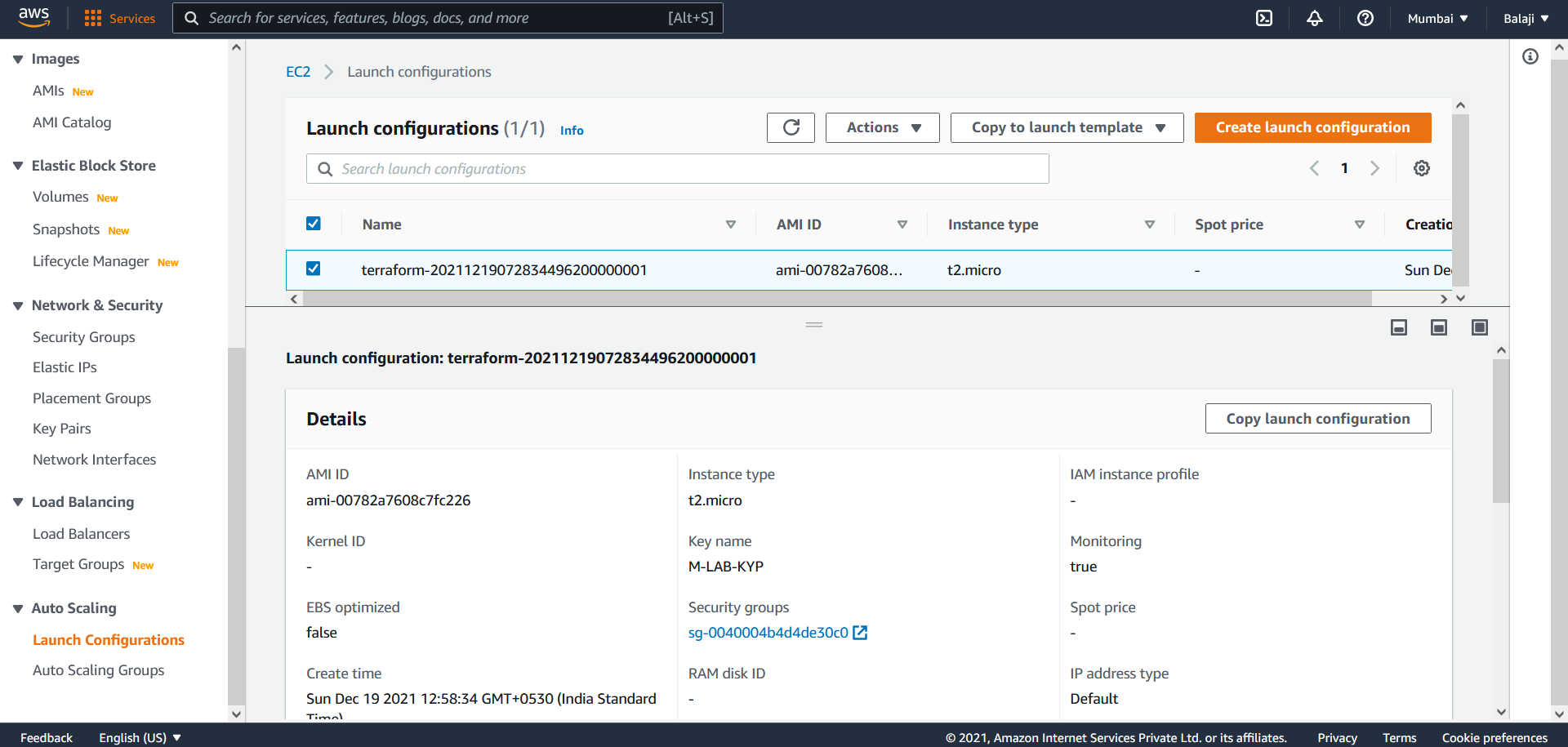
Security Groups:



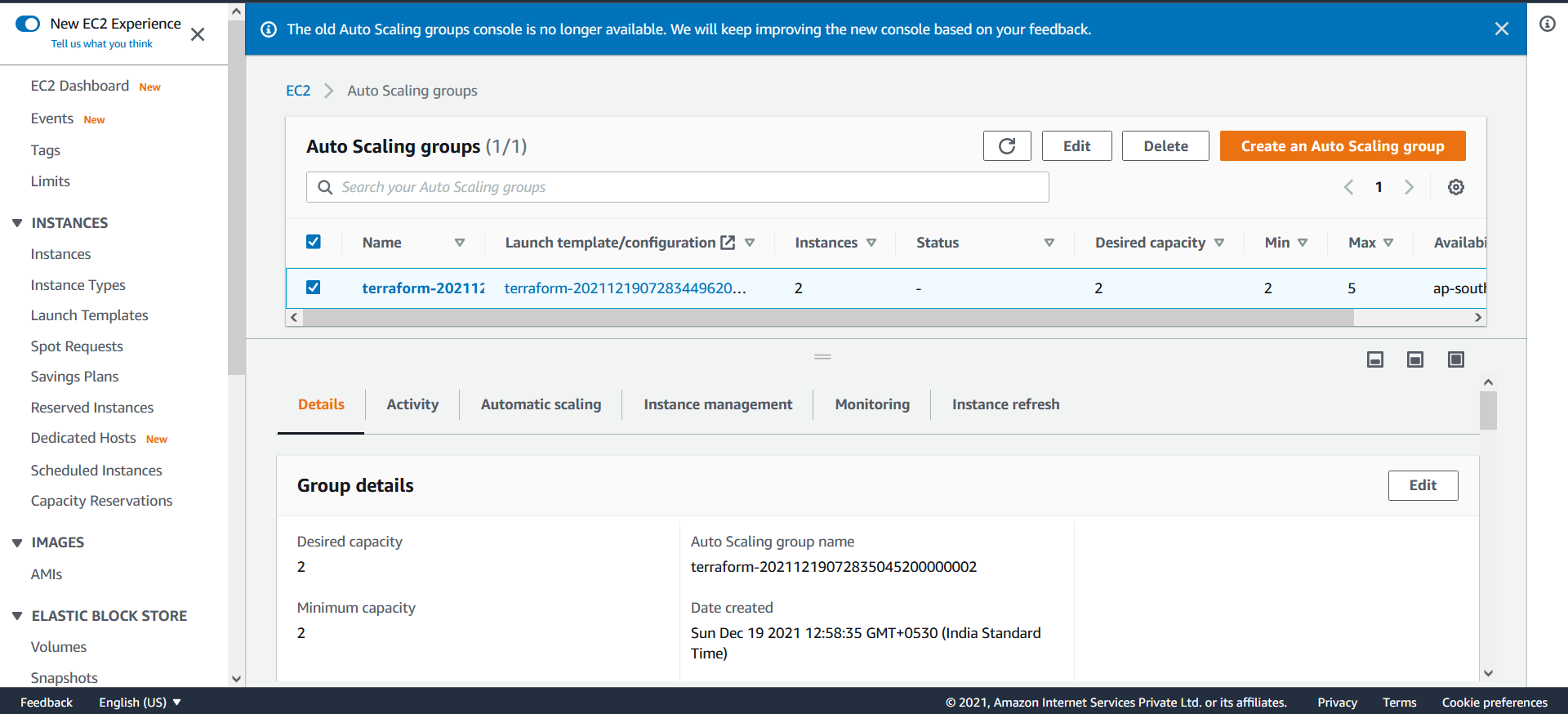
Load Balancer:



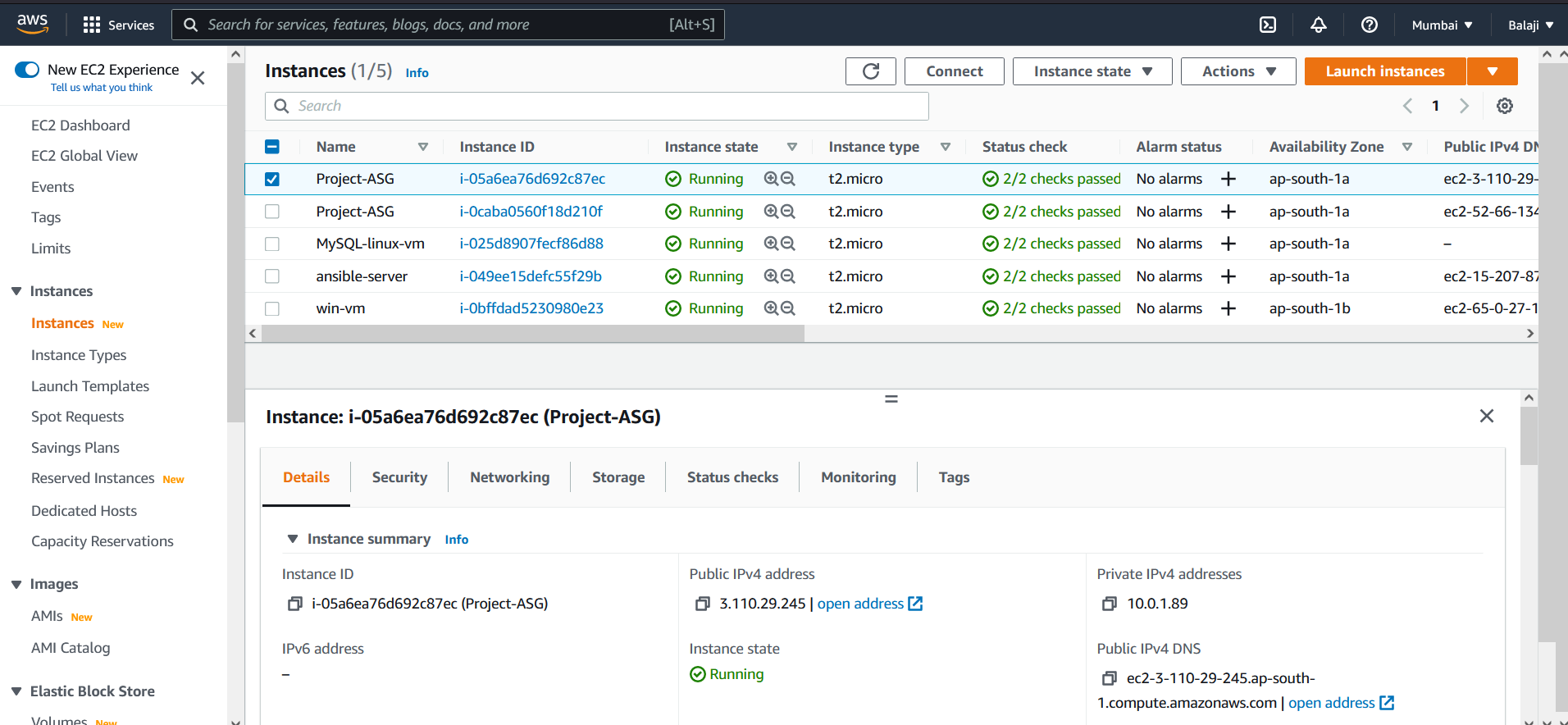
Launch Configuration:



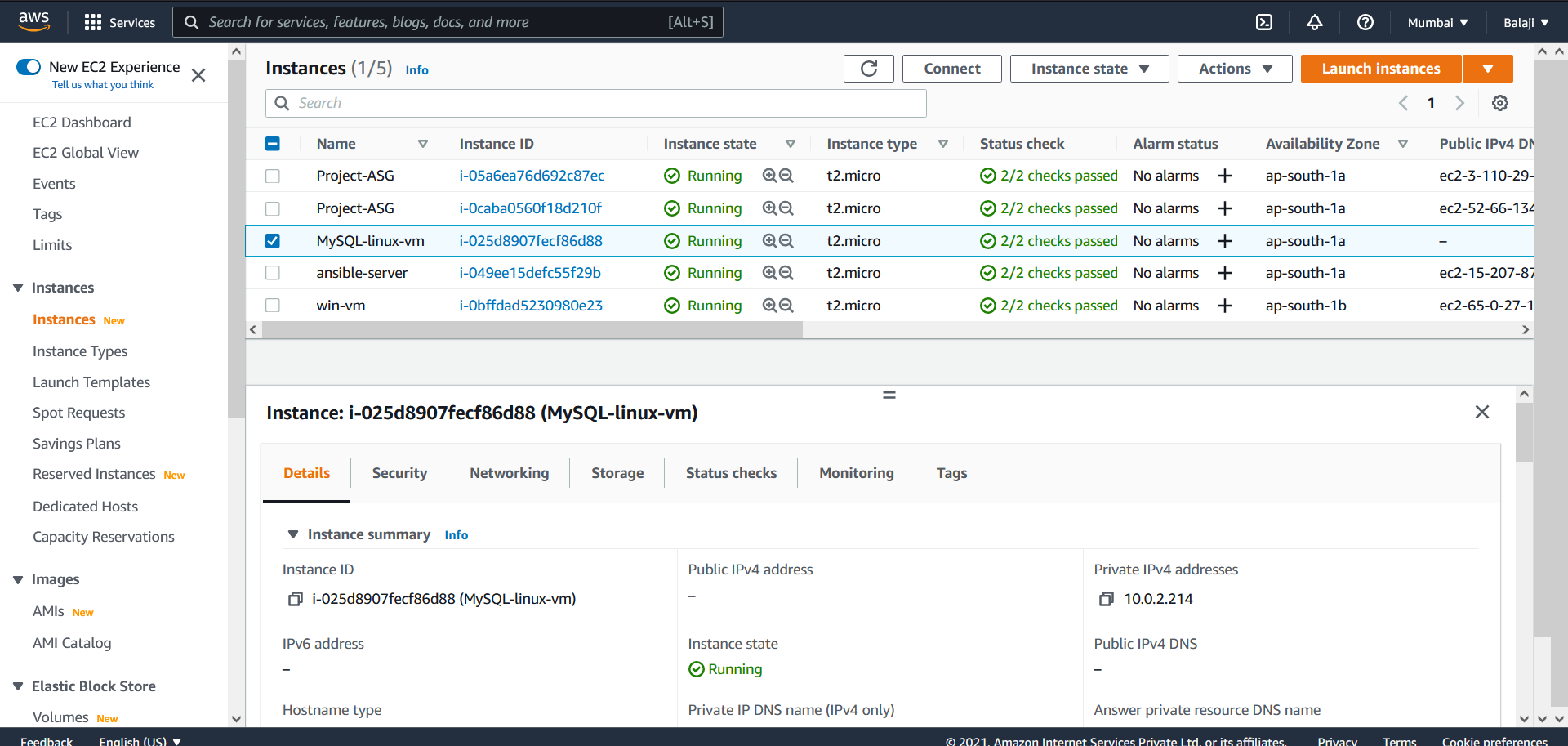
Auto scaling group:



Web cluster Instances:



MySQL Instance:



**Step 3: Ansible**

Install Apache server in autoscaling group public instance and MySQL DB in private instance

3.1 Launch new ubuntu instance and install ansible using below command

$ sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get install ansible

ansible server:

---------------------

Login as ubuntu

ssh-keygen

cat ./.ssh/id\_rsa.pub

copy the content of id\_rsa.pub

ansible node:

--------------

Login as ubuntu

mkdir .ssh

vi .ssh/authorized\_keys

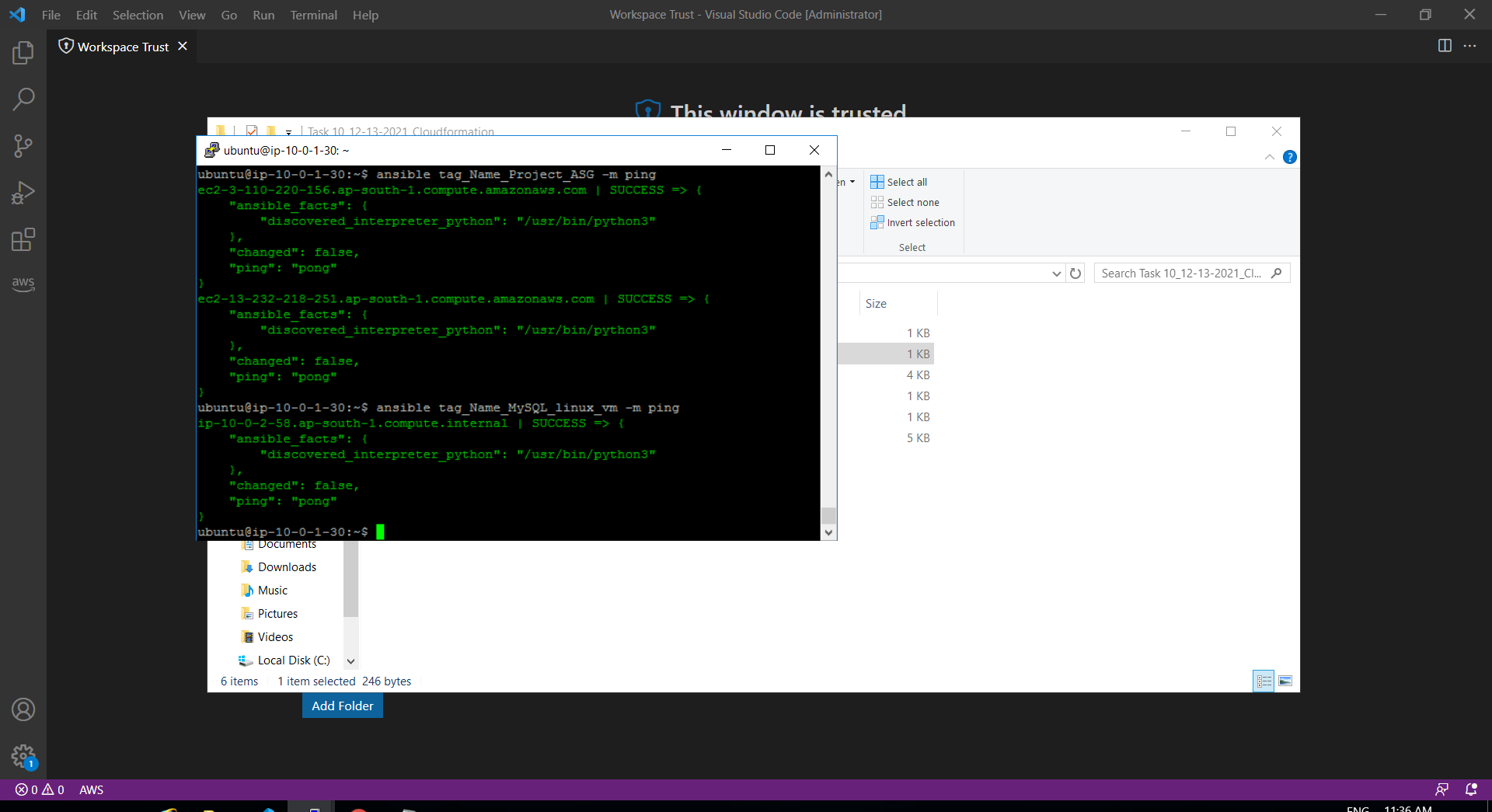
paste the content of id\_rsa.pub here

finally from ansible server execute the following command and check:

ssh ubuntu@ansible-node-privateip

3.2 Configure Dynamic inventory by using below link. And check the ping test for remote nodes

<https://devopscube.com/setup-ansible-aws-dynamic-inventory/>



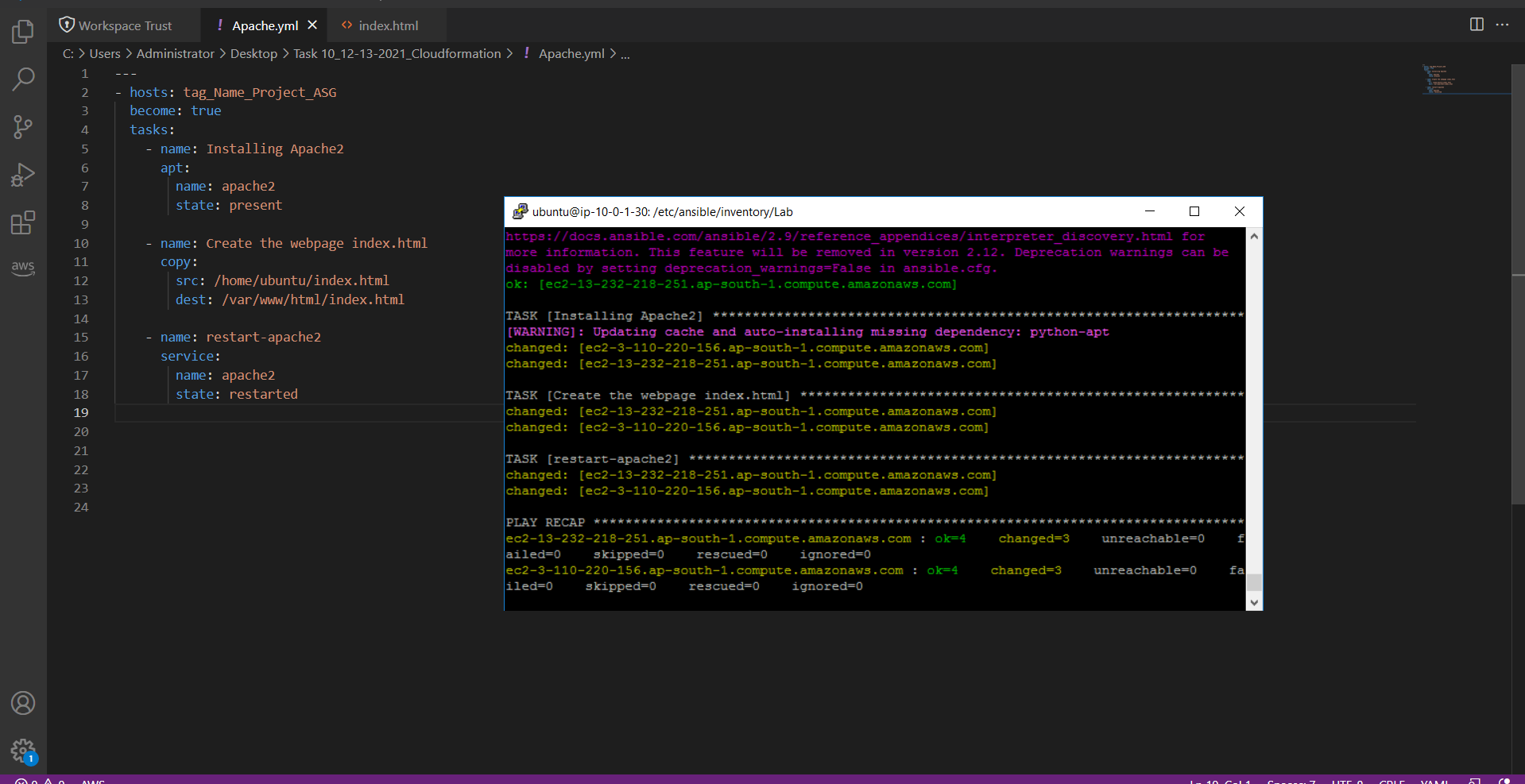
3.3 Save the below scripts in ansible server

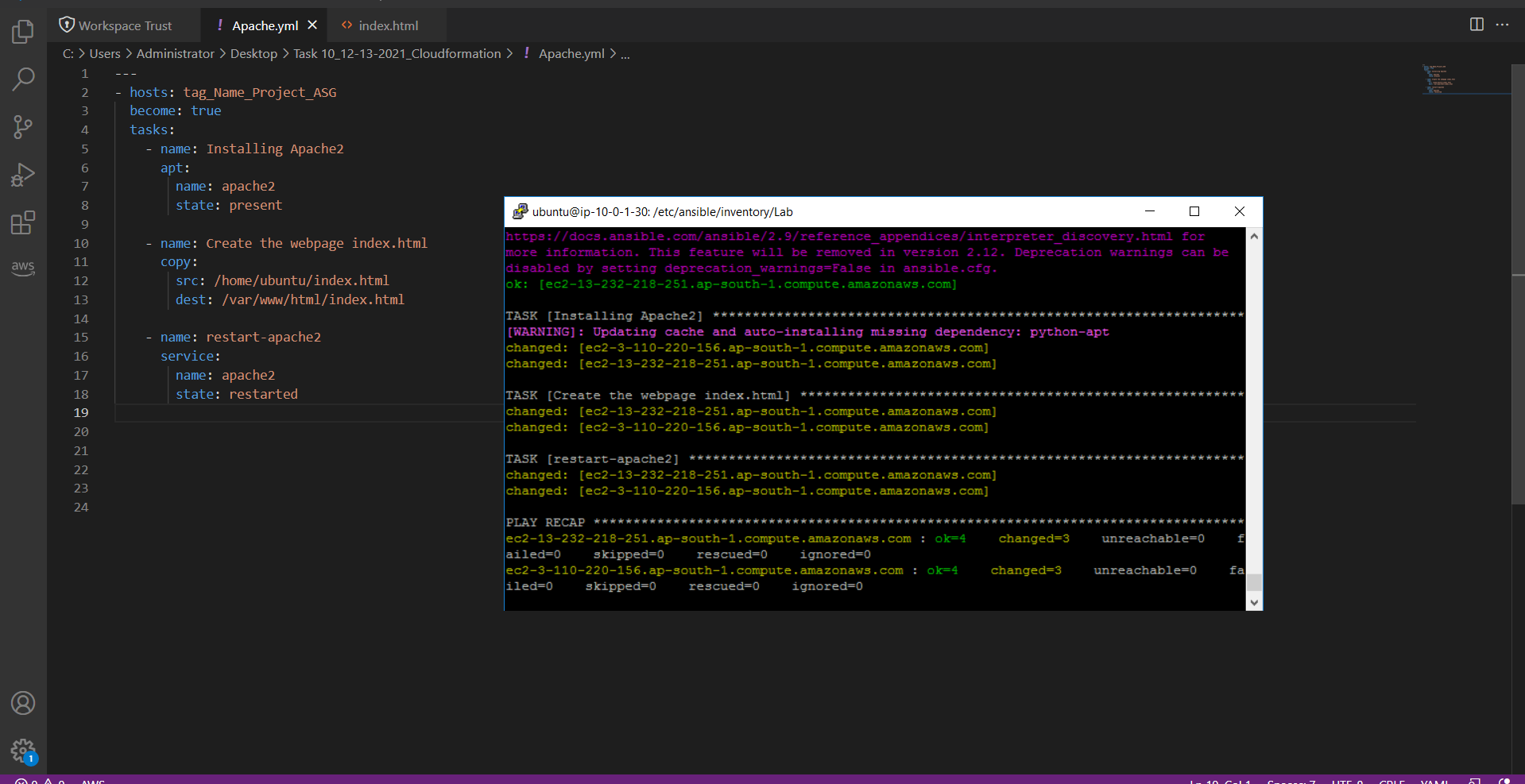


3.4 Install Apache server on ASG public instances and MySQL DB on Private instances by usung below command

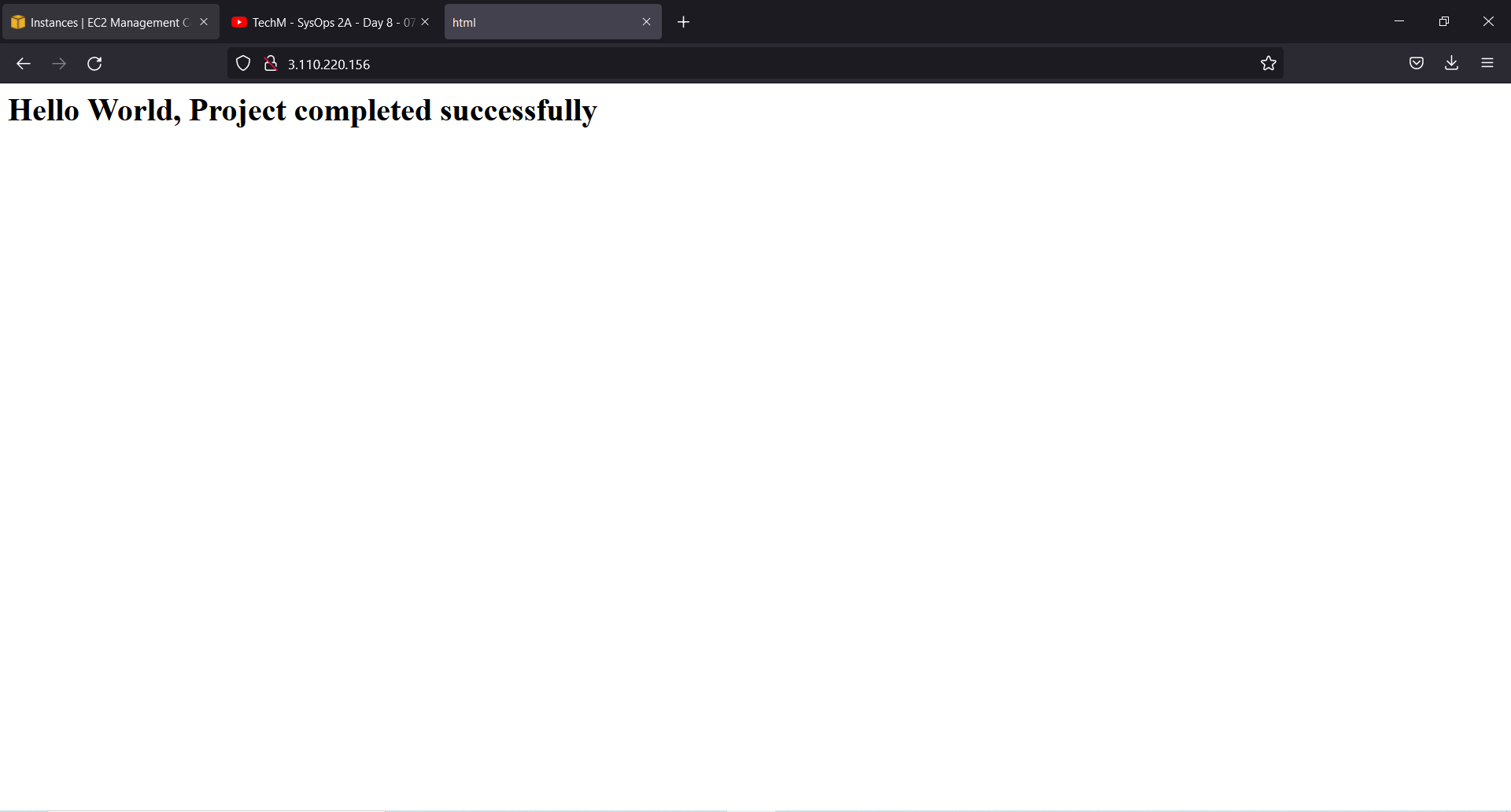
ansible-playbook Apache.yml

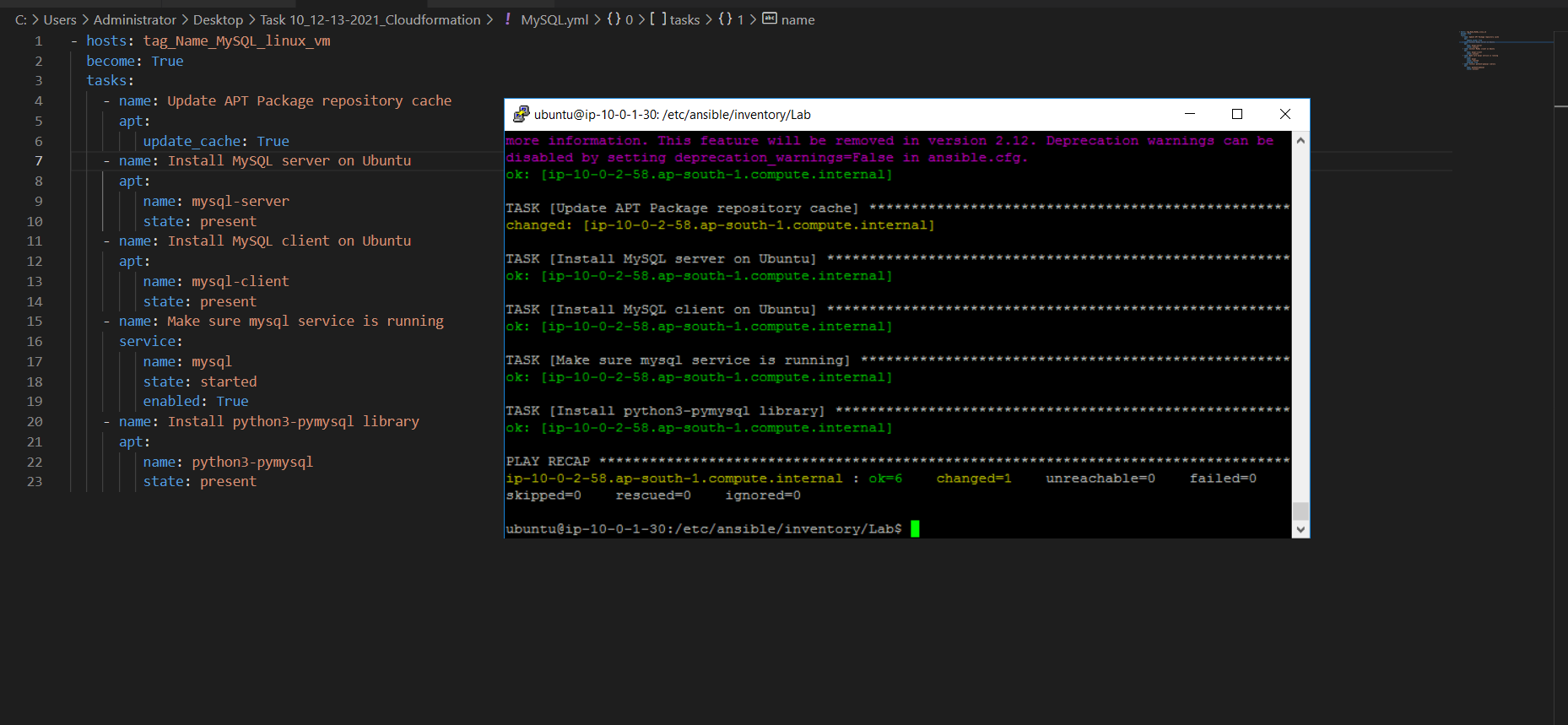
ansible-playbook MySQL.yml





3.5 check the application status



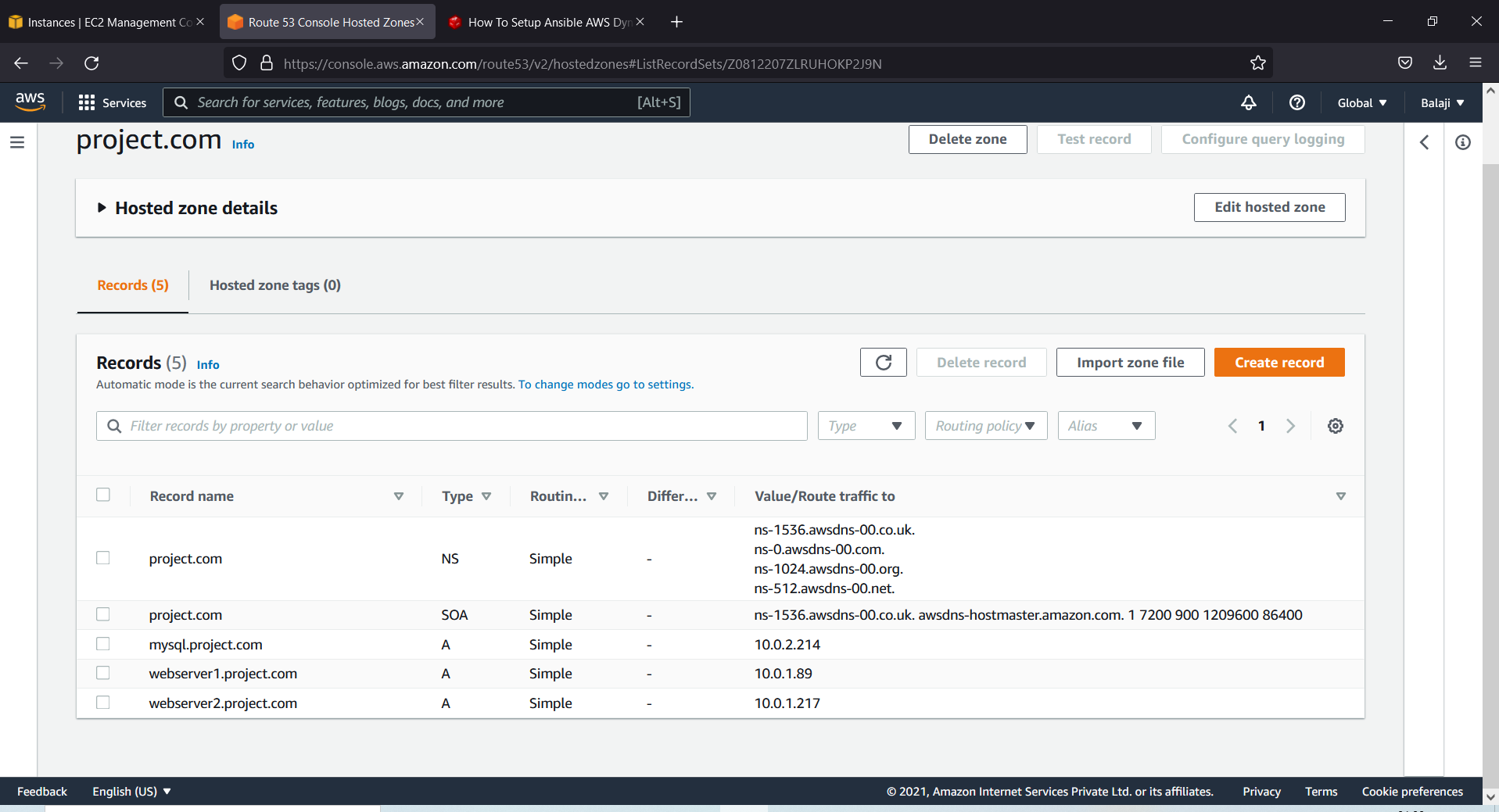


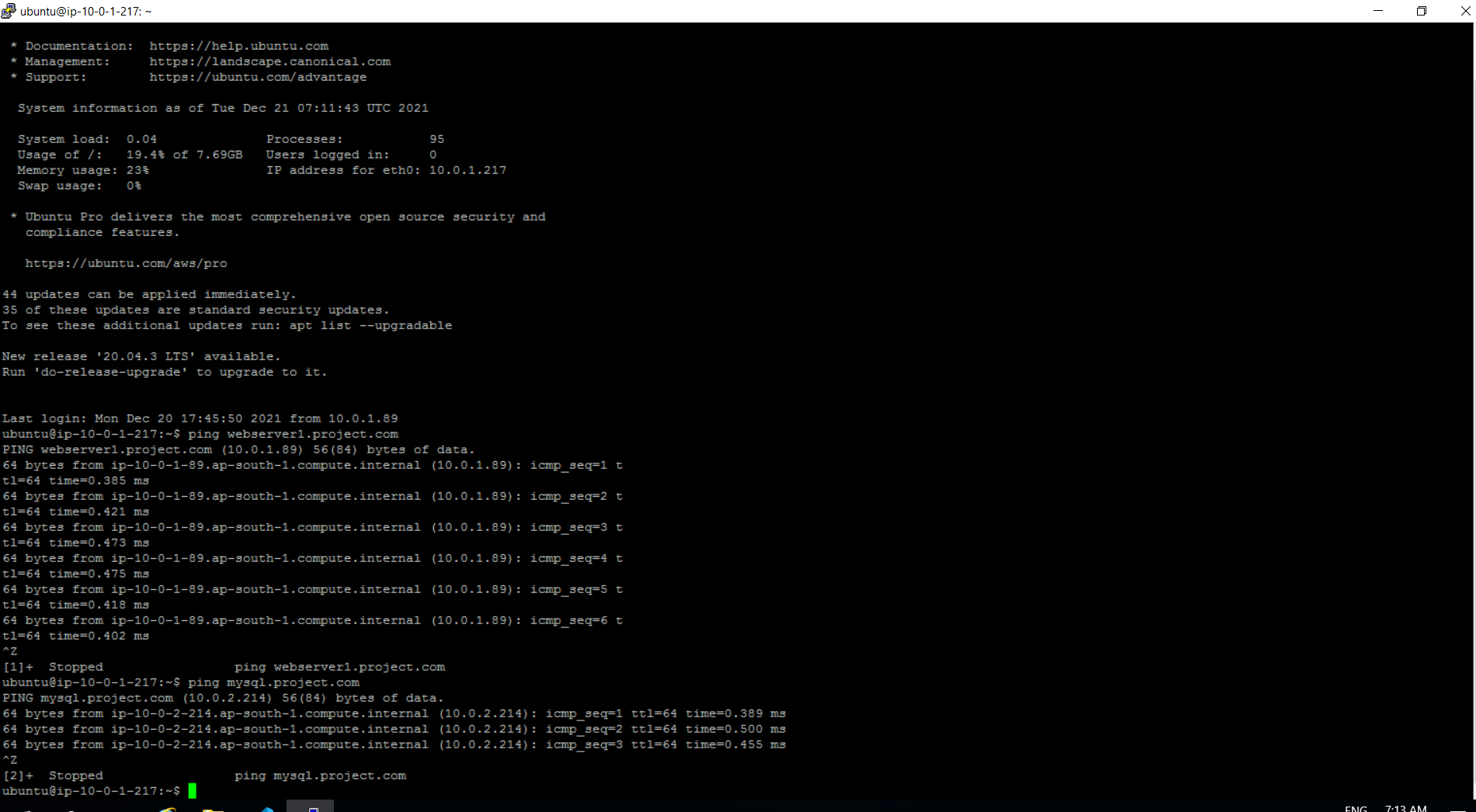
**Part 2**:

They wanted to have highly Secured internal Communication Application Management, hence decided to configure the infra manually

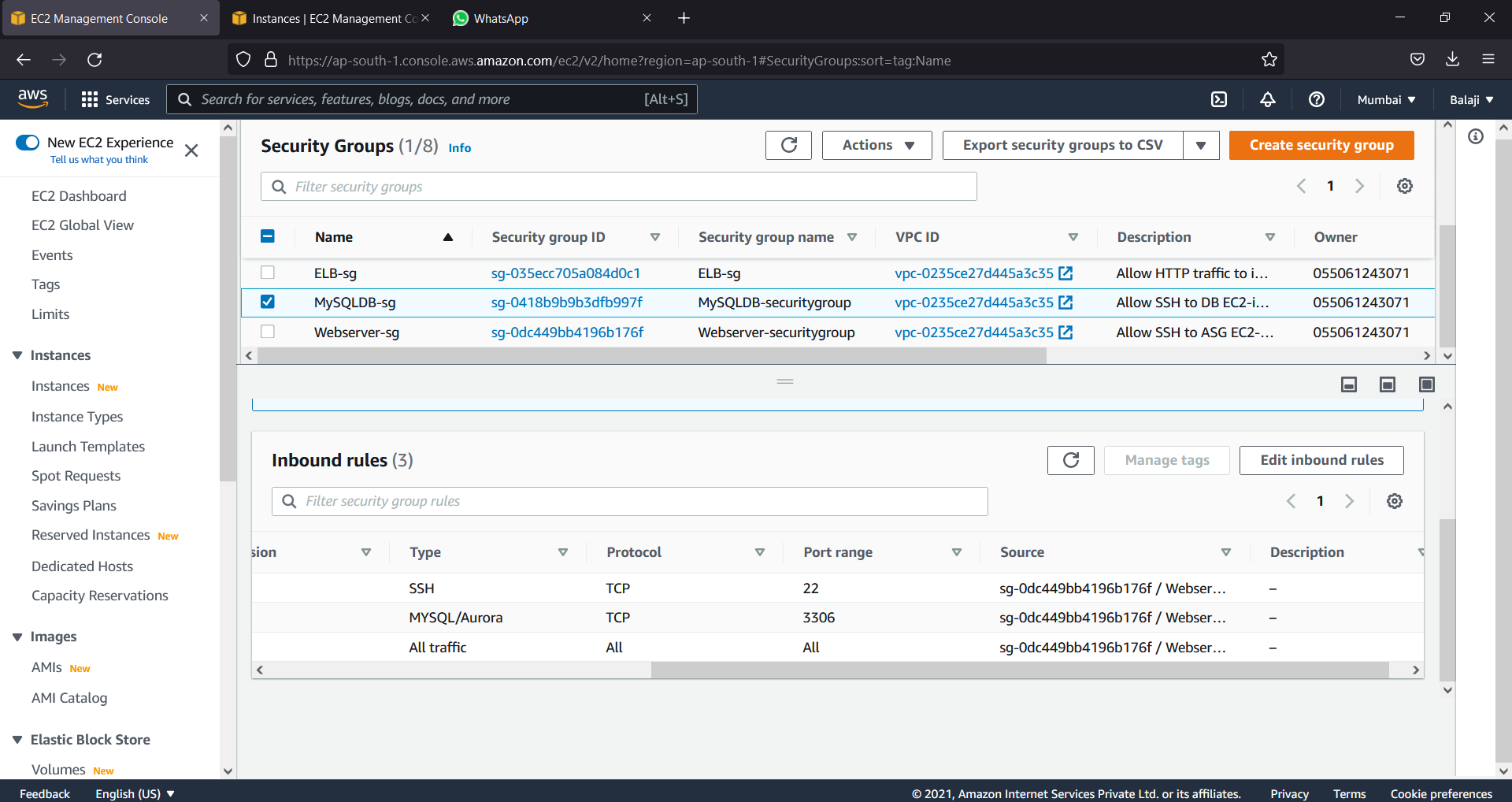
Step 1: Create Route53 Configuration for VPC internal communication

* 1. Login into aws cloud using console.
  2. Go to Route 53 and create private hosted Zone
  3. Create A name record for ASG instances and DB instance.





Step 2: Modify the security group of MySQL DB (EC2) that should be accessible only from ASG (Webserver Cluster)



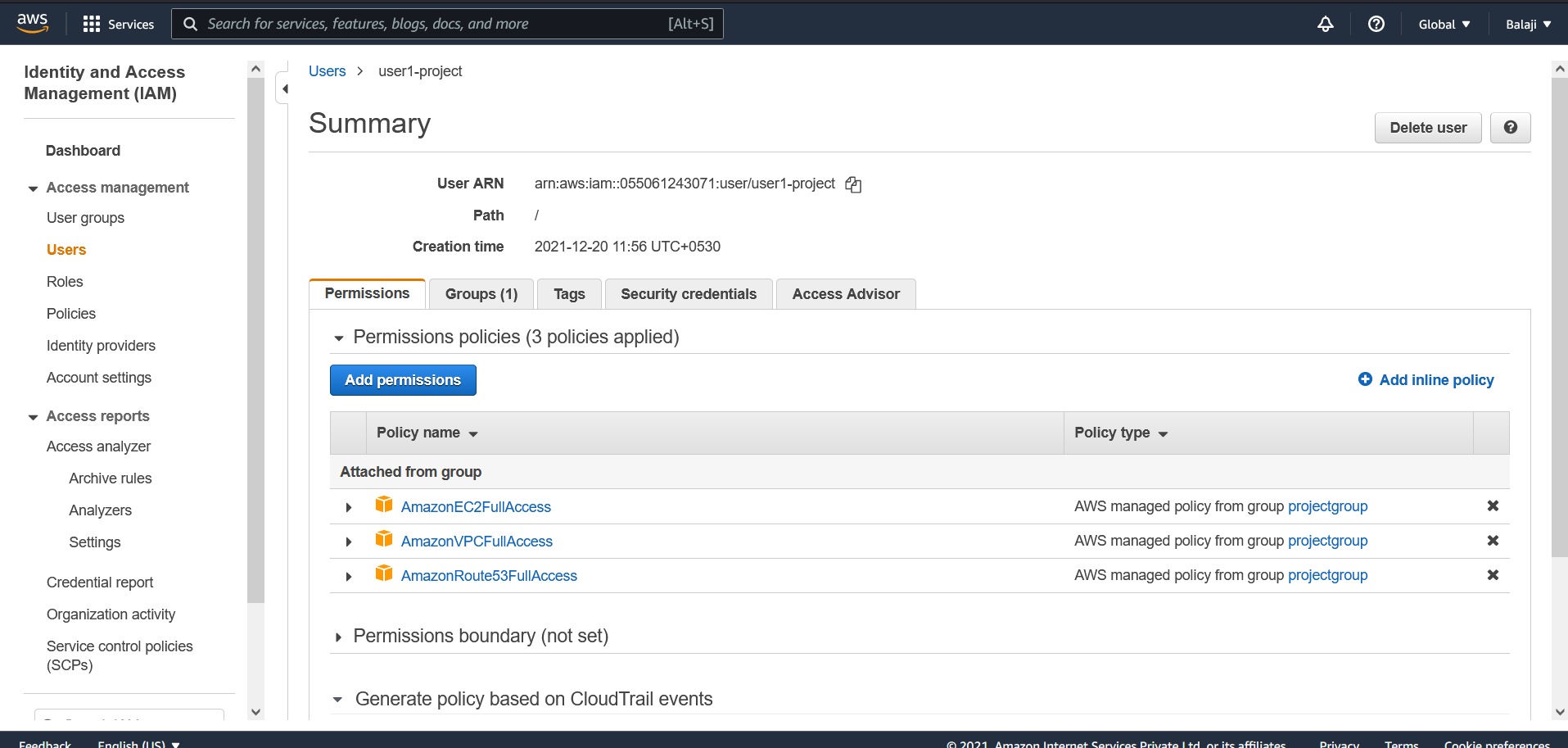
Step 3: Create user to manage application infrastructure (VPC, Route53, EC2)

3.1 Go to IAM services and create user with below policy

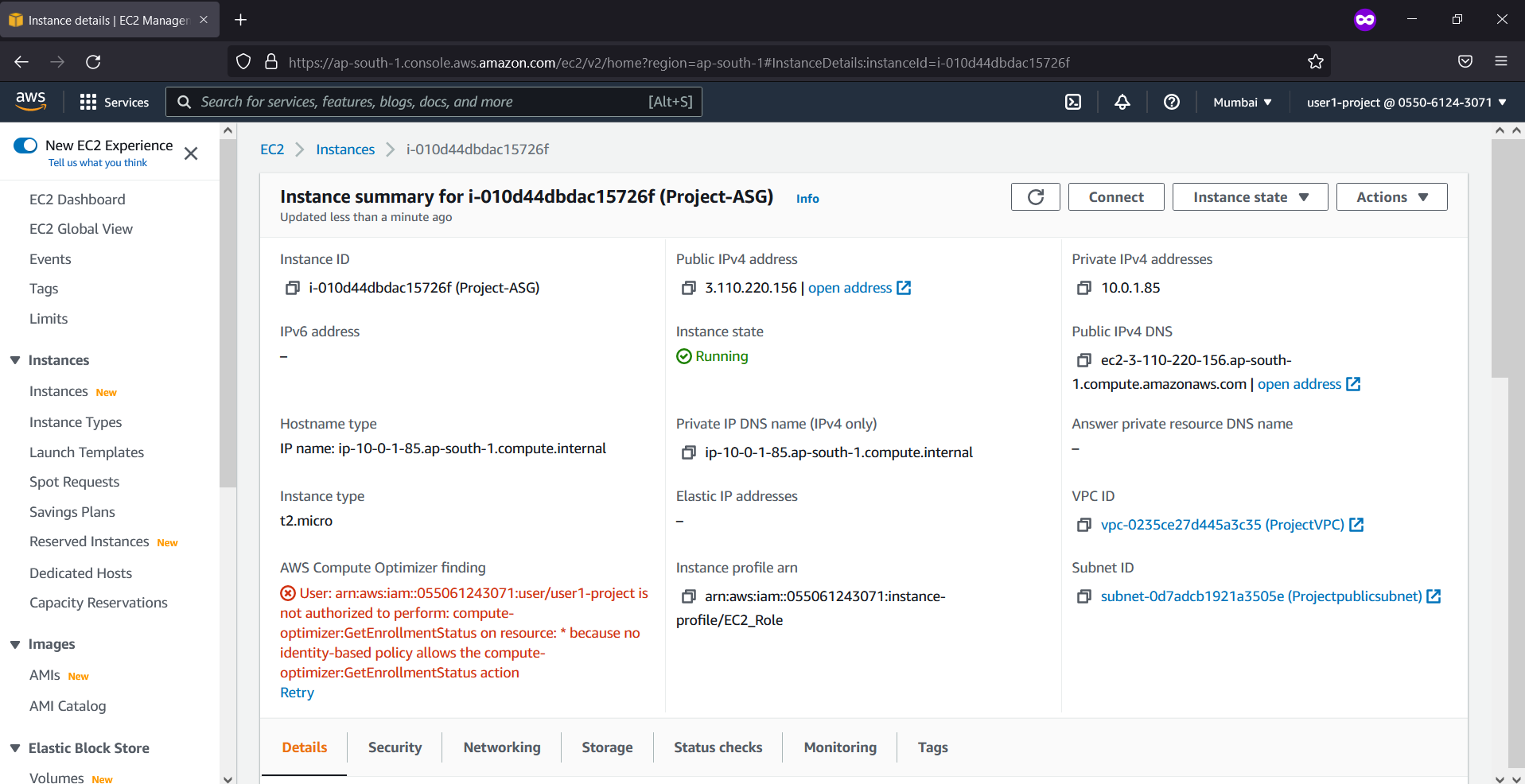
[AmazonEC2FullAccess](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonEC2FullAccess)

[AmazonVPCFullAccess](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonVPCFullAccess)

[AmazonRoute53FullAccess](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonRoute53FullAccess)

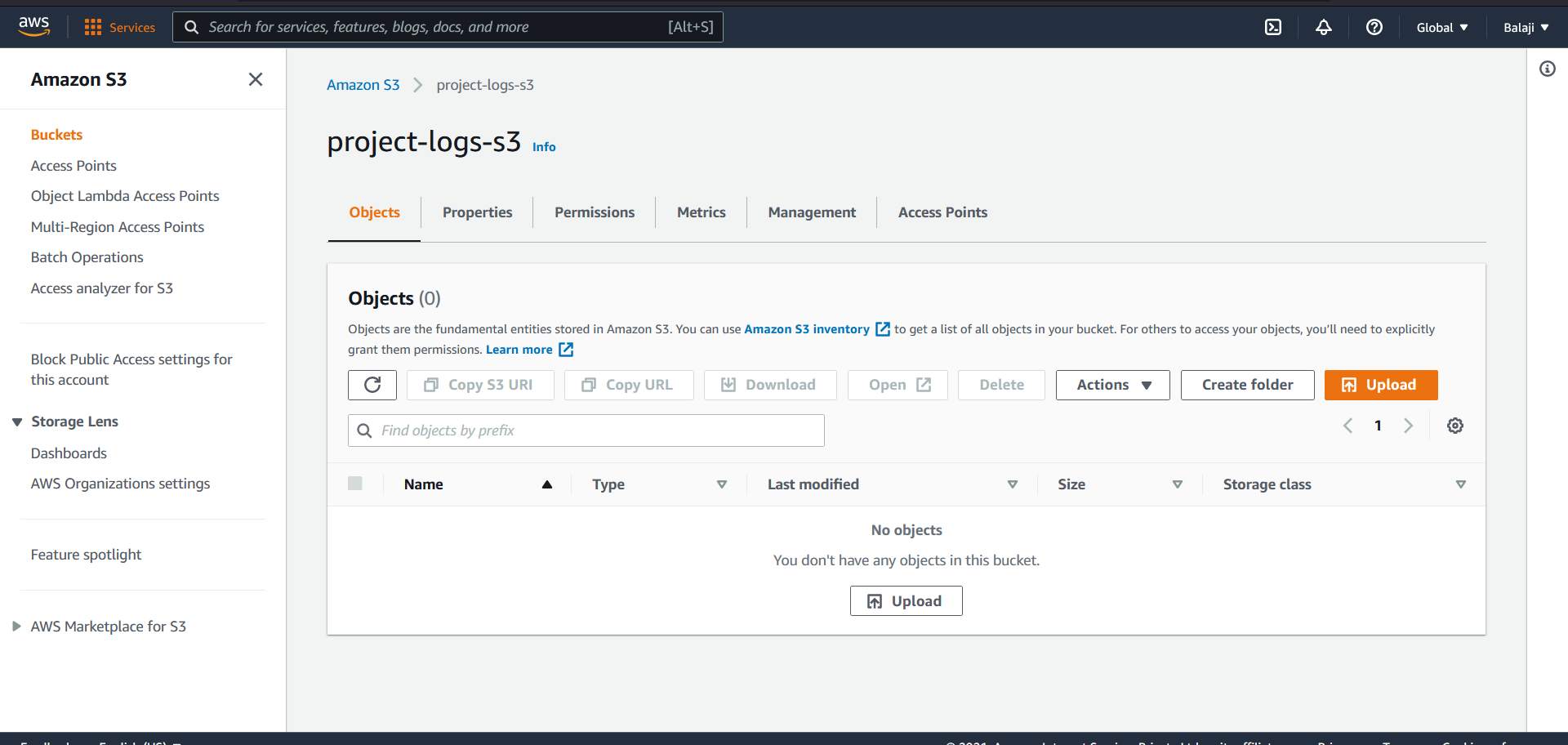


3.2 Login into user account and check the access



Step 4: Create S3 Bucket to store log files

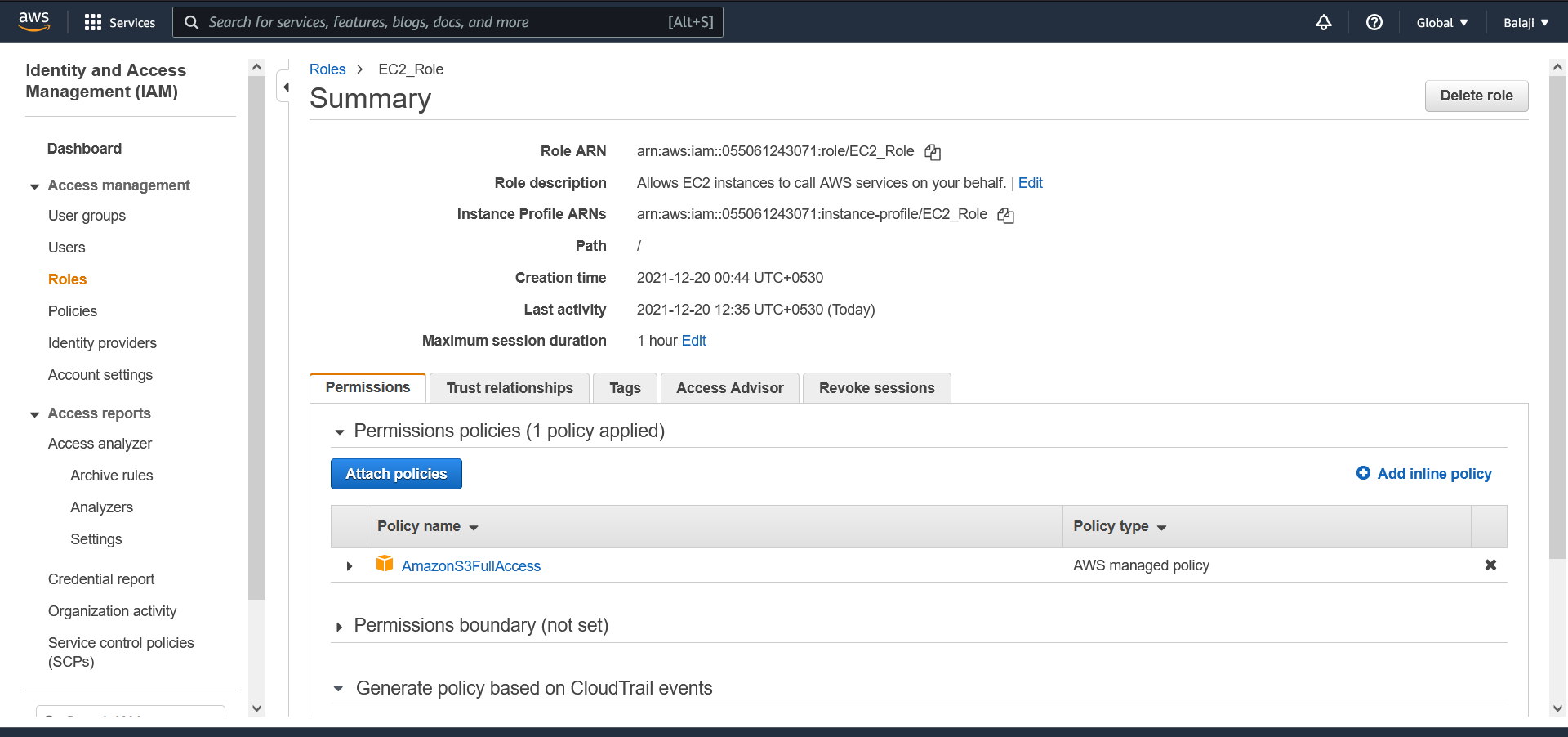
4.1 Go to S3 services and create a bucket



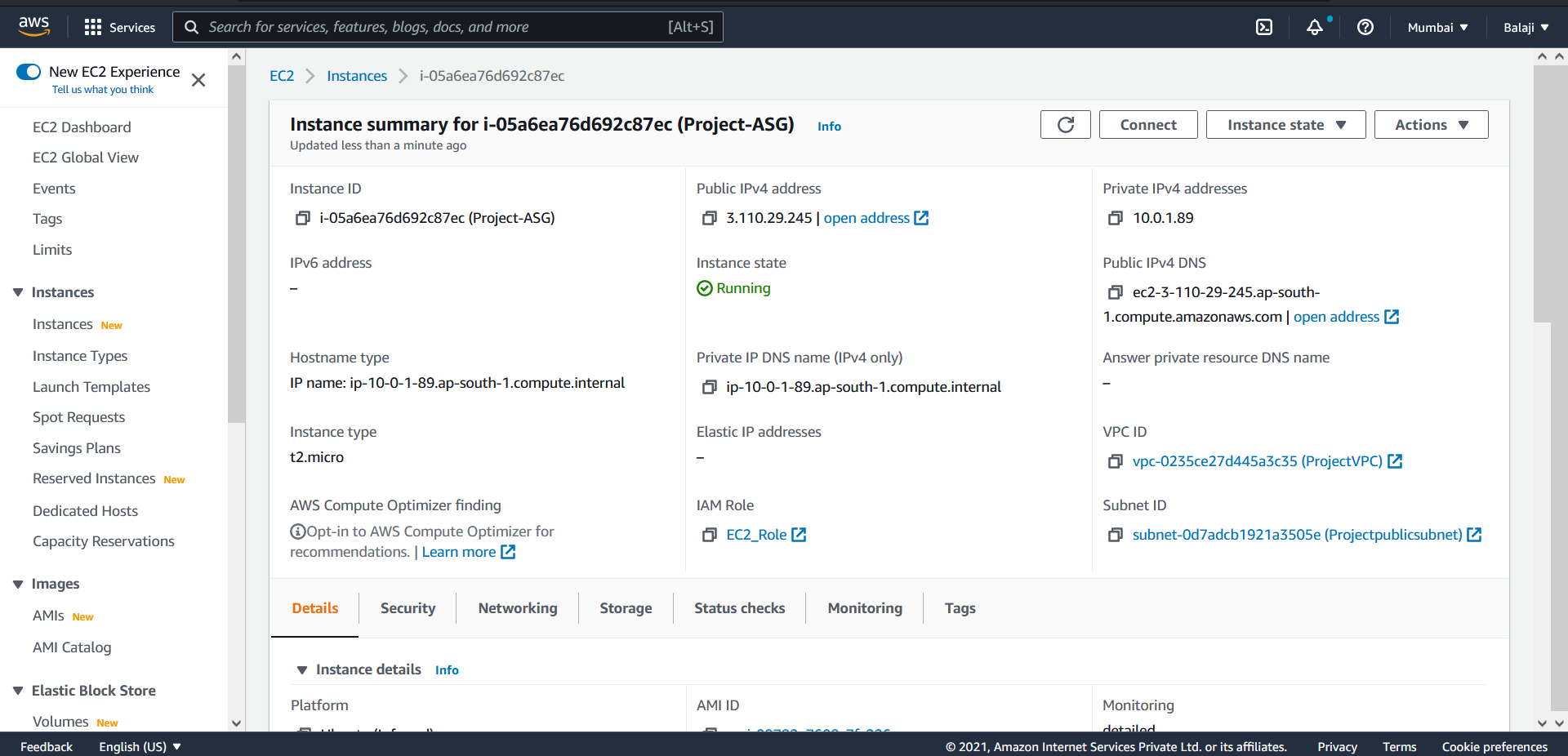
Step 5: All EC2 instances should have permission to access S3

5.1 Create the IAM role for EC2 services with below policy

[AmazonS3FullAccess](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonS3FullAccess)



5.2 Go the EC2 instances and change the ASG instance IAM role



**Part 3:**

They wanted to continuously monitor the resources for optimum usage and resources available for scalability

Step 1: Create a CloudWatch Dashboard to monitor:

* Network Traffic (Network I/O)
* EC2 instances CPU, Disk, Memory Usage
* Load balancer Traffic
  1. Go to CloudWatch service and add the dashboard with data source as metrices
  2. Add the above services in metrices

