**Project Description:**

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. Using Amazon EC2 eliminates your need to invest in hardware up front so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

**Problem Statement:**

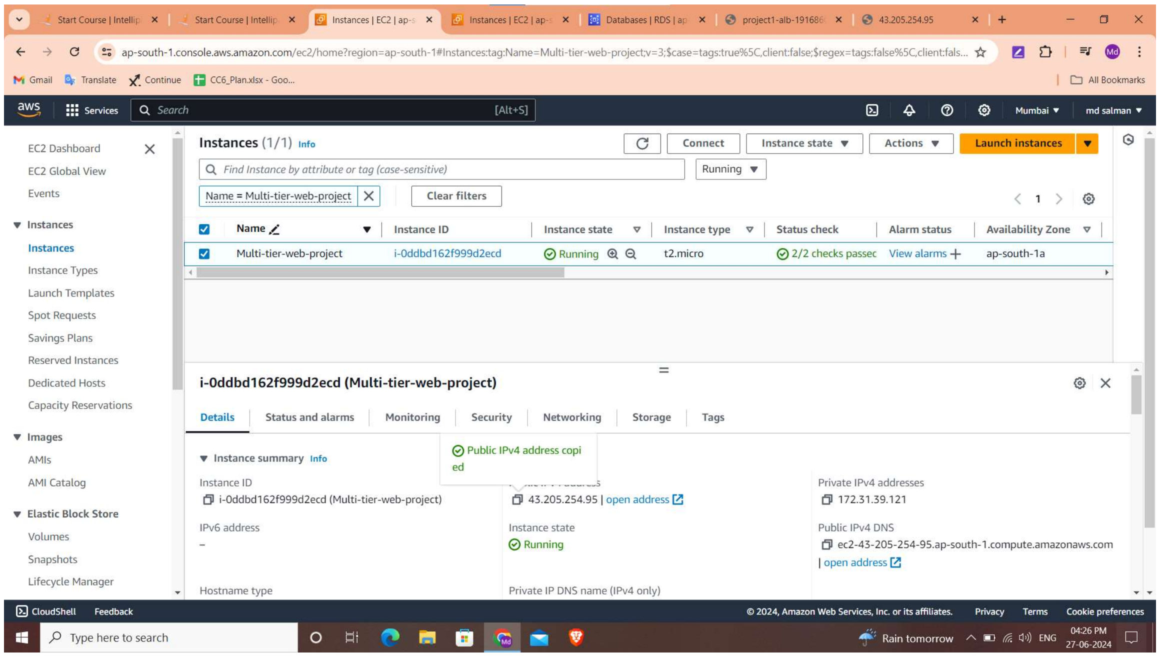
Company ABC wants to move their product to AWS. They have the following things set up right now:

1. MySQL DB
2. Website (PHP)

The company wants high availability on this product, therefore wants Auto Scaling to be enabled on this website.

# Multi-tier-web-Application.

Step1:The first thing we,re gonna do is create one ec2 Instance (OS) Ubuntu. In that we,re gonna host the frontend of the website.



Instances (1/1)

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone |
|---|---|---|---|---|---|---|
| Multi-tier-web-project | i-0ddbd162f999d2ecd | Running | t2.micro | 2/2 checks passec | View alarms + | ap-south-1a |

i-0ddbd162f999d2ecd (Multi-tier-web-project)

Instance summary

Public IPv4 address copied

Instance ID: i-0ddbd162f999d2ecd (Multi-tier-web-project)

Public IPv4 address: 43.205.254.95 | open address

Private IPv4 addresses: 172.31.39.121

IPv6 address: –

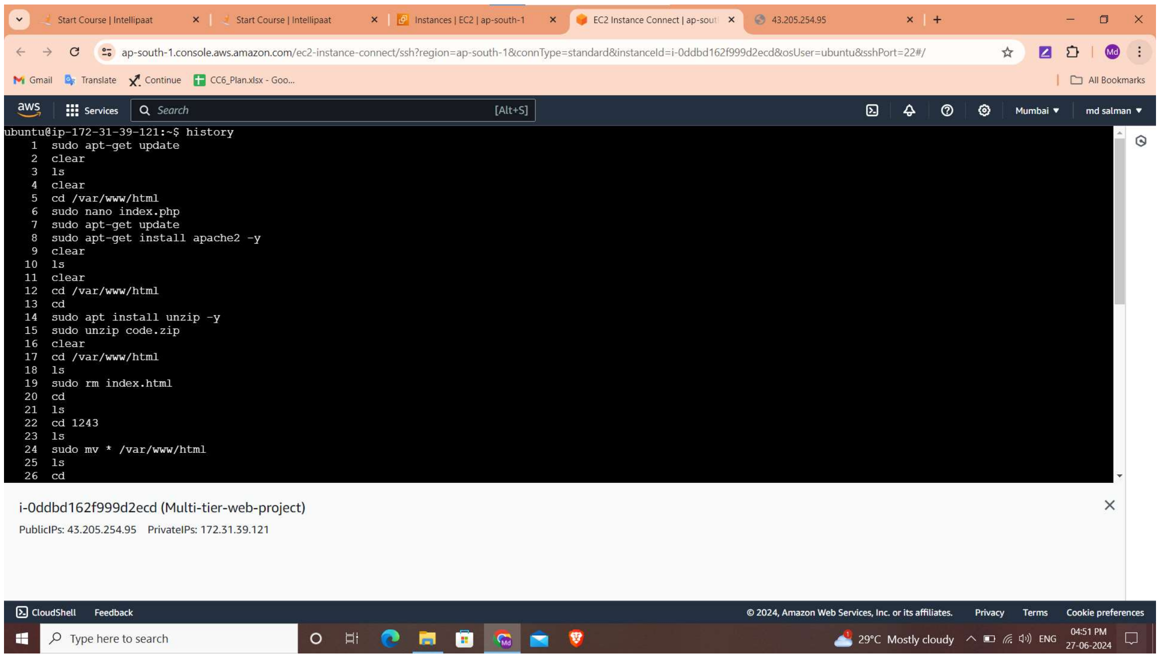
Instance state: Running

Public IPv4 DNS: ec2-43-205-254-95.ap-south-1.compute.amazonaws.com | open address

Hostname type

Private IP DNS name (IPv4 only)

Step2:Now we,re gonna copy the code in the server from our local, then we,ll unzip it and we,ll check it on local host is website working well.

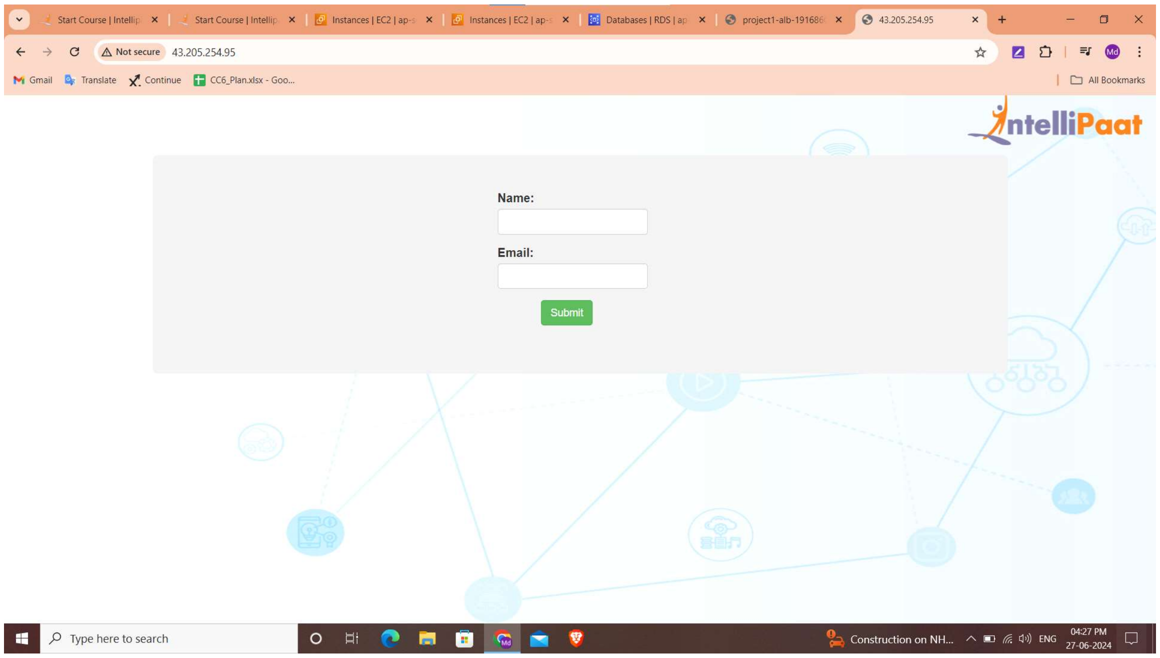


```
ubuntu@ip-172-31-39-121:~$ history
    1  sudo apt-get update
    2  clear
    3  ls
    4  clear
    5  cd /var/www/html
    6  sudo nano index.php
    7  sudo apt-get update
    8  sudo apt-get install apache2 -y
    9  clear
   10  ls
   11  clear
   12  cd /var/www/html
   13  cd
   14  sudo apt install unzip -y
   15  sudo unzip code.zip
   16  clear
   17  cd /var/www/html
   18  ls
   19  sudo rm index.html
   20  cd
   21  ls
   22  cd 1243
   23  ls
   24  sudo mv * /var/www/html
   25  ls
   26  cd
```

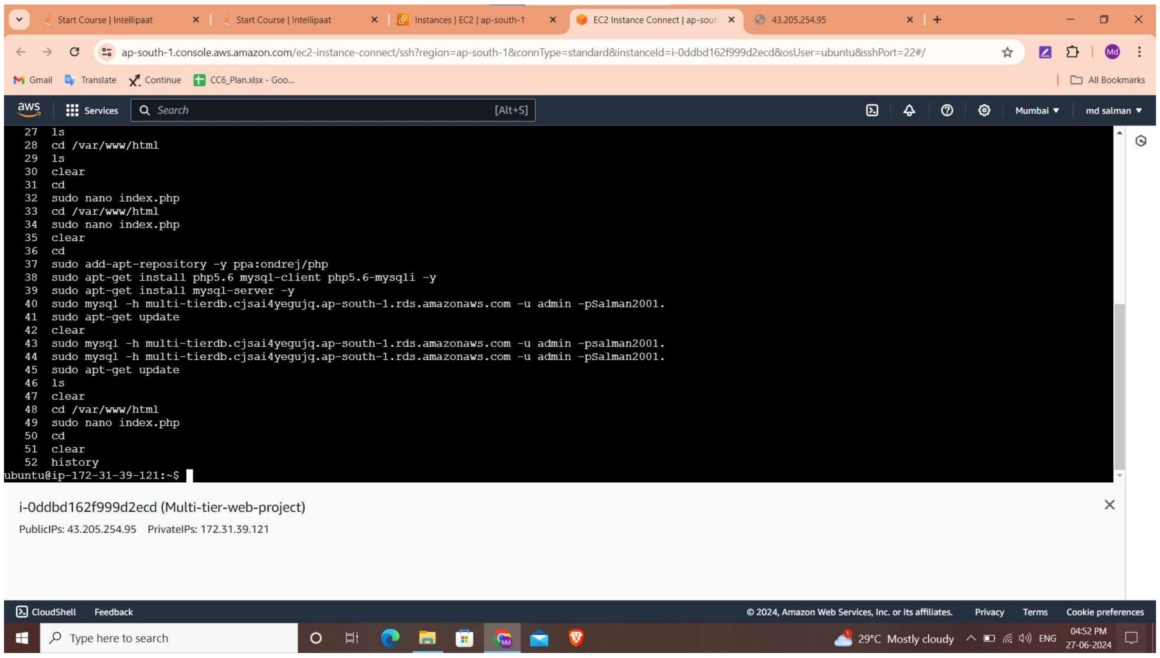
i-0ddbd162f999d2ecd (Multi-tier-web-project)

PublicIPs: 43.205.254.95 PrivateIPs: 172.31.39.121

Step3: Checking the PHP website in our local.



Step4:Now we,ll install all the PHP and Mysql repository and dependency in our server.

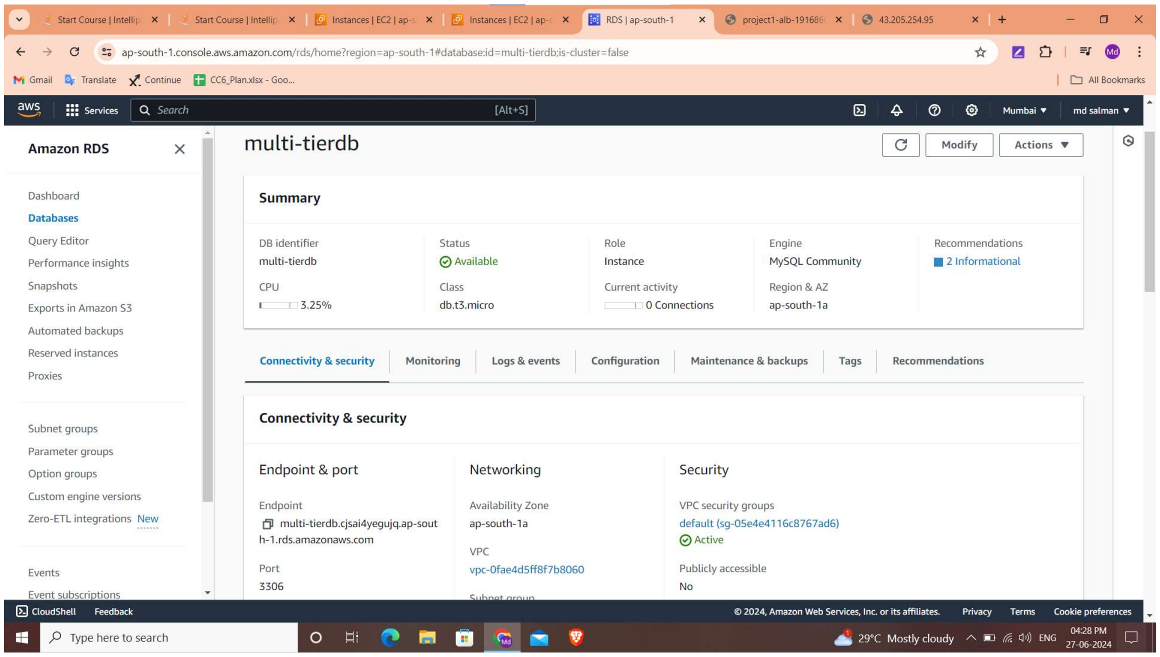


```
   27  ls
   28  cd /var/www/html
   29  ls
   30  clear
   31  cd
   32  sudo nano index.php
   33  cd /var/www/html
   34  sudo nano index.php
   35  clear
   36  cd
   37  sudo add-apt-repository -y ppa:ondrej/php
   38  sudo apt-get install php5.6 mysql-client php5.6-mysqli -y
   39  sudo apt-get install mysql-server -y
   40  sudo mysql -h multi-tierdb.cjsai4yegujq.ap-south-1.rds.amazonaws.com -u admin -pSalman2001.
   41  sudo apt-get update
   42  clear
   43  sudo mysql -h multi-tierdb.cjsai4yegujq.ap-south-1.rds.amazonaws.com -u admin -psalman2001.
   44  sudo mysql -h multi-tierdb.cjsai4yegujq.ap-south-1.rds.amazonaws.com -u admin -pSalman2001.
   45  sudo apt-get update
   46  ls
   47  clear
   48  cd /var/www/html
   49  sudo nano index.php
   50  cd
   51  clear
   52  history
ubuntu@ip-172-31-39-121:~$
```

i-0ddbd162f999d2ecd (Multi-tier-web-project)

PublicIPs: 43.205.254.95 PrivateIPs: 172.31.39.121

Step5:In this step we,re gonna create RDS DB engine is MYSQL and then through database endpoint in web-server we,ll login as mysql.


ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1#database:id=multi-tierdb;is-cluster=false

Amazon RDS

- Dashboard
- Databases
- Query Editor
- Performance insights
- Snapshots
- Exports in Amazon S3
- Automated backups
- Reserved instances
- Proxies
- Subnet groups
- Parameter groups
- Option groups
- Custom engine versions
- Zero-ETL integrations New
- Events
- Event subscriptions

multi-tierdb

Summary

| DB identifier | Status | Role | Engine | Recommendations |
|---|---|---|---|---|
| multi-tierdb | Available | Instance | MySQL Community | 2 Informational |
| CPU | Class | Current activity | Region & AZ | |
| 3.25% | db.t3.micro | 0 Connections | ap-south-1a | |

Connectivity & security | Monitoring | Logs & events | Configuration | Maintenance & backups | Tags | Recommendations

Connectivity & security

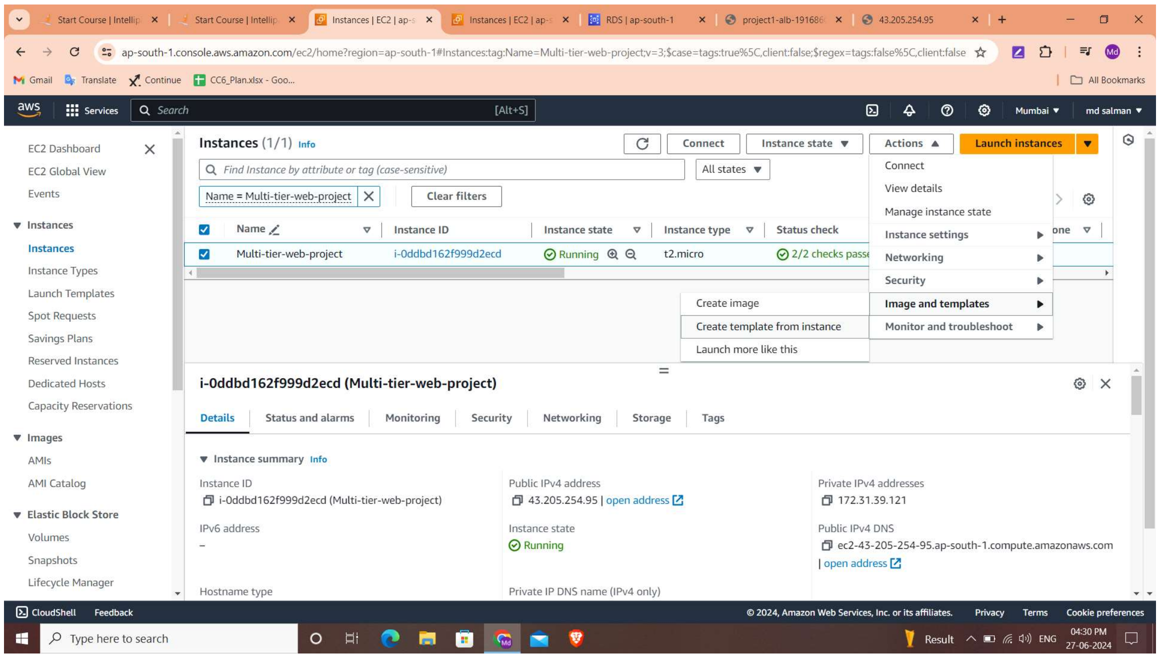
Endpoint & port
Endpoint: multi-tierdb.cjsai4yegujq.ap-south-1.rds.amazonaws.com
Port: 3306

Networking
Availability Zone: ap-south-1a
VPC: vpc-0fae4d5ff8f7b8060

Security
VPC security groups: default (sg-05e4e4116c8767ad6) Active
Publicly accessible: No

© 2024, Amazon Web Services, Inc. or its affiliates.

Step6:In this step I created AMI because I want to add autoscaling on my product. So I created AMI Template by using this below PHP web-server


ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:tag:Name=Multi-tier-web-project;v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false

Instances (1/1)

Name = Multi-tier-web-project

| Name | Instance ID | Instance state | Instance type | Status check |
|---|---|---|---|---|
| Multi-tier-web-project | i-0ddbd162f999d2ecd | Running | t2.micro | 2/2 checks passed |

Actions: Connect, View details, Manage instance state, Instance settings, Networking, Security, Image and templates, Monitor and troubleshoot
Image and templates: Create image, Create template from instance, Launch more like this

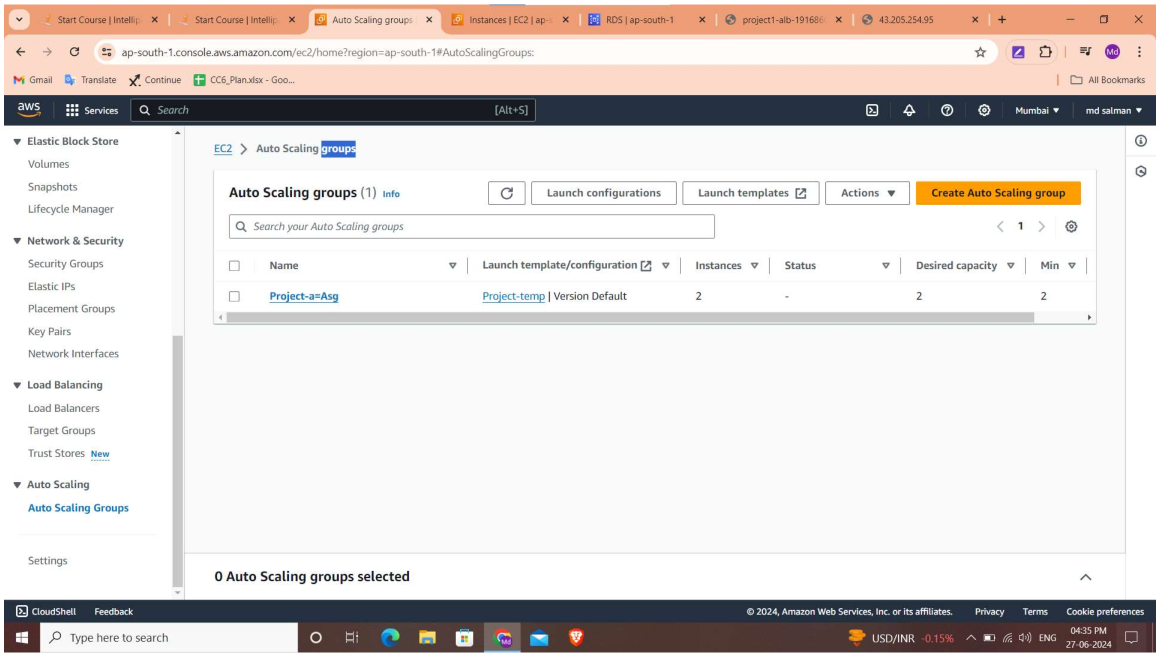
i-0ddbd162f999d2ecd (Multi-tier-web-project)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID: i-0ddbd162f999d2ecd (Multi-tier-web-project)
Public IPv4 address: 43.205.254.95 | open address
Private IPv4 addresses: 172.31.39.121
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-43-205-254-95.ap-south-1.compute.amazonaws.com | open address

Step7: Using that AMI template I created Autoscaling group.

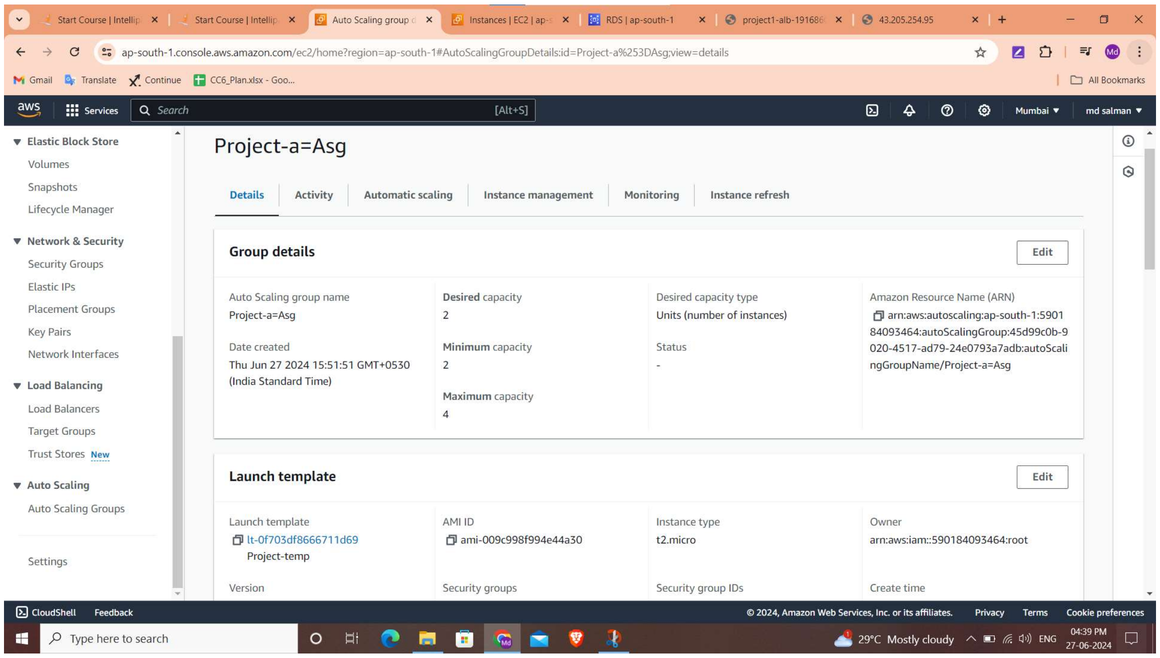

EC2 > Auto Scaling groups

Auto Scaling groups (1) Info

| | Name | Launch template/configuration | Instances | Status | Desired capacity | Min |
|---|---|---|---|---|---|---|
| ☐ | Project-a=Asg | Project-temp \| Version Default | 2 | - | 2 | 2 |

0 Auto Scaling groups selected

Step8:In Autoscaling I created 2 desired capacity and 2 Minimum and 2 Maximum capacity as per your requirement.


Project-a=Asg

Group details

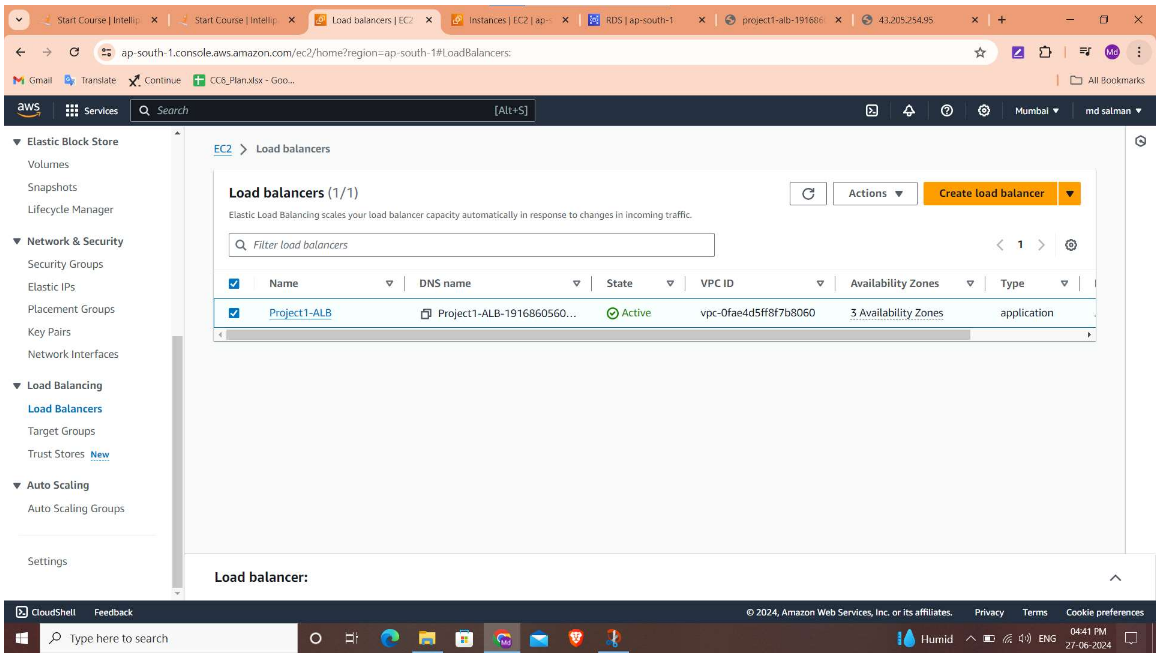
| | | | |
|---|---|---|---|
| Auto Scaling group name<br>Project-a=Asg | Desired capacity<br>2 | Desired capacity type<br>Units (number of instances) | Amazon Resource Name (ARN)<br>arn:aws:autoscaling:ap-south-1:590184093464:autoScalingGroup:45d99c0b-9020-4517-ad79-24e0793a7adb:autoScalingGroupName/Project-a=Asg |
| Date created<br>Thu Jun 27 2024 15:51:51 GMT+0530 (India Standard Time) | Minimum capacity<br>2 | Status<br>- | |
| | Maximum capacity<br>4 | | |

Launch template

| | | | |
|---|---|---|---|
| Launch template<br>lt-0f703df8666711d69<br>Project-temp | AMI ID<br>ami-009c998f994e44a30 | Instance type<br>t2.micro | Owner<br>arn:aws:iam::590184093464:root |
| Version | Security groups | Security group IDs | Create time |

Step9: In order to achieve high availability I created Loadbalancer which is application

loadbalancer.



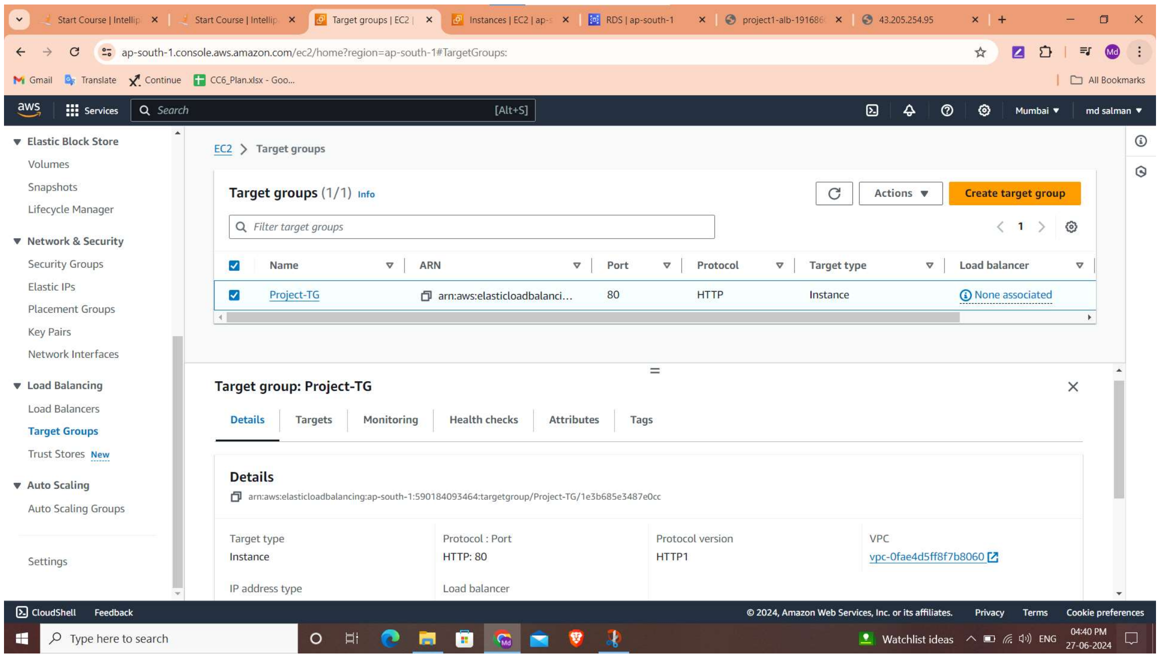
**Load balancers** (1/1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

| Name | DNS name | State | VPC ID | Availability Zones | Type |
|---|---|---|---|---|---|
| Project1-ALB | Project1-ALB-1916860560... | Active | vpc-0fae4d5ff8f7b8060 | 3 Availability Zones | application |

Load balancer:

Step10: This the target group for our Instances. we,re gonna add include them as pending below.



**Target groups** (1/1) Info

| Name | ARN | Port | Protocol | Target type | Load balancer |
|---|---|---|---|---|---|
| Project-TG | arn:aws:elasticloadbalanci... | 80 | HTTP | Instance | None associated |

Target group: Project-TG

Details

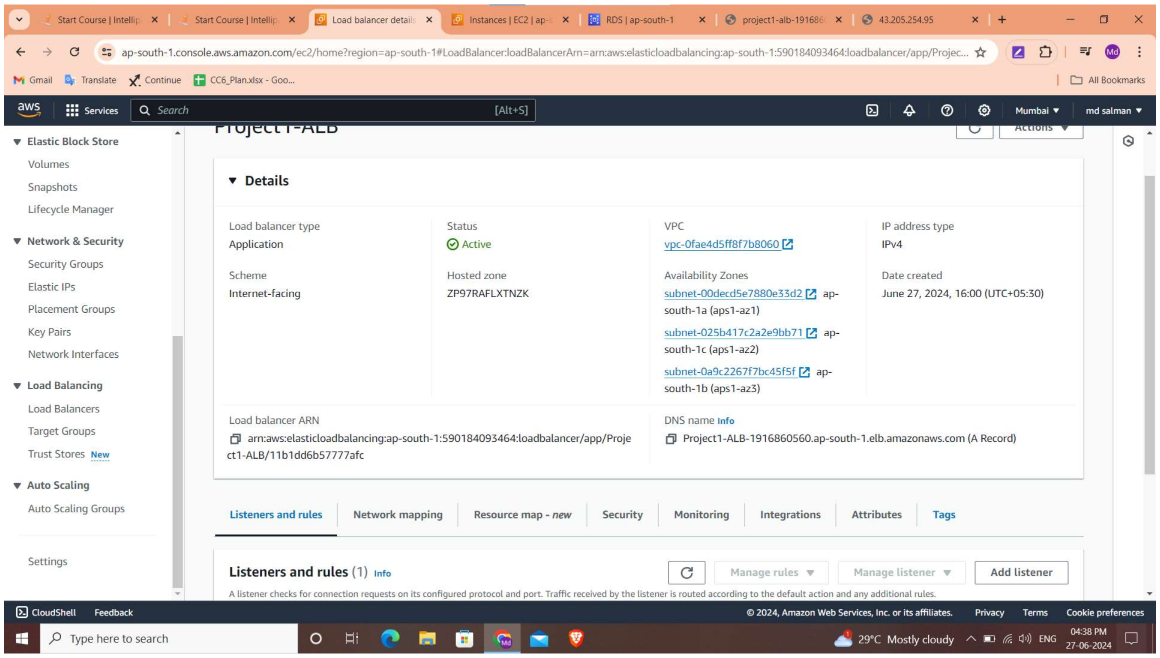
arn:aws:elasticloadbalancing:ap-south-1:590184093464:targetgroup/Project-TG/1e3b685e3487e0cc

| Target type | Protocol : Port | Protocol version | VPC |
|---|---|---|---|
| Instance | HTTP: 80 | HTTP1 | vpc-0fae4d5ff8f7b8060 |

IP address type

Load balancer

Step11:Application Loadbalancer configuration.



ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LoadBalancer:loadBalancerArn=arn:aws:elasticloadbalancing:ap-south-1:590184093464:loadbalancer/app/Projec...

Project1-ALB

**Details**

| Load balancer type | Status | VPC | IP address type |
|---|---|---|---|
| Application | Active | vpc-0fae4d5ff8f7b8060 | IPv4 |

| Scheme | Hosted zone | Availability Zones | Date created |
|---|---|---|---|
| Internet-facing | ZP97RAFLXTNZK | subnet-00decd5e7880e33d2 ap-south-1a (aps1-az1)<br>subnet-025b417c2a2e9bb71 ap-south-1c (aps1-az2)<br>subnet-0a9c2267f7bc45f5f ap-south-1b (aps1-az3) | June 27, 2024, 16:00 (UTC+05:30) |

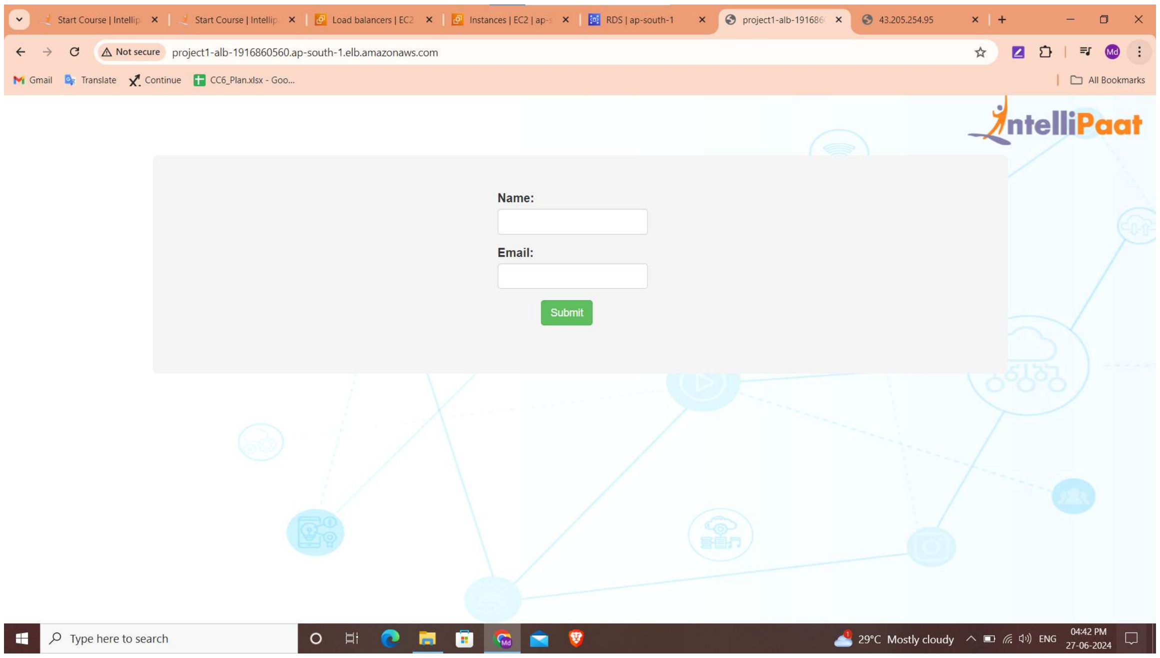
Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:590184093464:loadbalancer/app/Project1-ALB/11b1dd6b57777afc

DNS name Info
Project1-ALB-1916860560.ap-south-1.elb.amazonaws.com (A Record)

Listeners and rules (1) Info

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Step12:Copied the DNS of Application loadbalancer and checked on local server.



project1-alb-1916860560.ap-south-1.elb.amazonaws.com