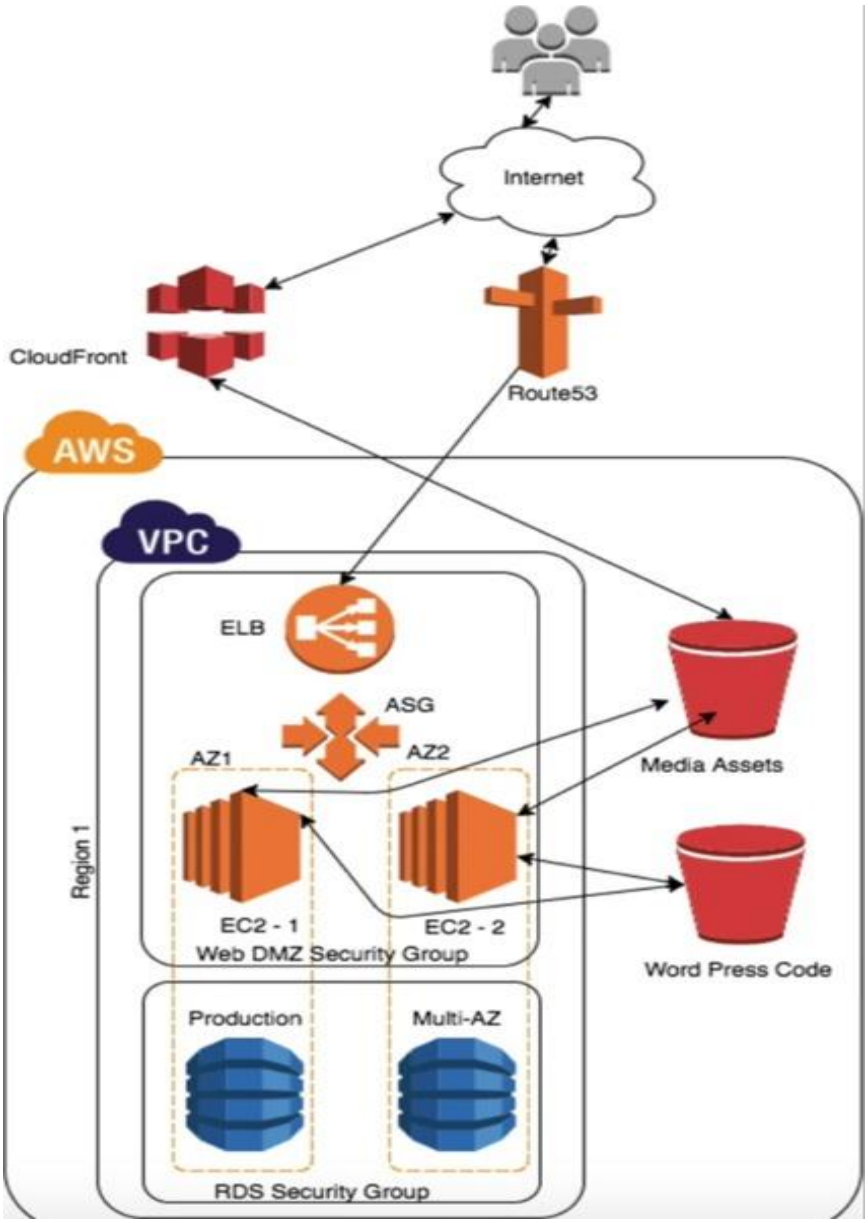
COURSE: DEVOPS

PROJECT NAME: Fault-Tolerant WordPress Site on AWS



NAME: THELLA NAVEEN

EMAIL: naveenthella3@gmail.com

Fault-Tolerant WordPress Site on AWS

Project Overview: Developed a fault-tolerant WordPress site hosted on AWS, ensuring high availability and scalability. Utilized various AWS services to build a resilient infrastructure capable of handling traffic spikes and minimizing downtime.

#following steps :

Step 1: Create VPC for the AWS Project

Step 2: Create a role in IAM for S3 full access

Step 3: Create Security Groups

--- web server SG

--- RDS security groups

Step 4: Create buckets in S3 in same region where security groups were created

--- mys3bucketawsproject

--- mys3buketawsprojet2

Step 5: Create cloud front Distribution

----Origin as mys3bucketawsproject.s3.ap-south-1.amazonaws.com and update new permission

Step 6: Create RDS instance with MYSQL multi AZ

Step 7: Create Elastic Load Balancer

Step 8: Create a record set in the Route 53 and map the naked domain to ELB created in step 7 (my domain is dtx6l2ip2uoip.cloudfront.net)

Step 9: Launch EC2 instance and just associate with the role created in the above step 2

Step 10: SSH into instance and run the scripts (installing the required software and patches)

```
#!/bin/bash
Yum update –y
Yum install httpd php php-mysql stress –y
cd /etc/httpd/conf
cp  httpd.conf  httpdconfbackup.conf
```

rm –rf httpd.conf

wget httpd://s3-eu-west-1.amazon.com/ <bucketname>/httpd.conf

cd /var/www/html/

echo “welcome world” > naveen.html

wget htpps://wordpress.org/latest.tar.gz

tar –xzf latest.tar.gz

cp –r wordpress/* /var/www/html/

rm –rf wordpress

rm –rf latest/tar.gz

chmod –R 755 wp-content

chown _R apache.apache wp-content

service httpd start

chkconfig httpd on

Step 11: Add the EC2 instance to the load balancer & log into domain name and configure wordpress website

Step 12: Create wp-content.php and copy the content from wordpress website and paste that

Step13: Try to access S3 bucket from EC2 instance

Step14: Copy all the contents from the html folder to the S3 bucket

-Enter the cmd aws s3 cp –recursive /var/www/html/ s3://mys3bucketawsproject/

Step 15: delete the html folder and generate a real time failover

Step 16: recreate the html directory & now copy back the content from s3 bucket

Step 17: update the permissions

chmod –R 755 wp-content

chown _R apache.apache wp-content

Step 18: check out the directory wp-content/uploads/ and see that the media files that we have uploads on wordpress website appear

Step 19: Automate the media files to move to s3 bucket ---> mys3buketawsprojet2

-So that cloud front will serve the media files

Step 20: Enter the cmd aws s3 cp --recursive /var/html/wp-content/uploads/ s3:// mys3buketawsprojet2/

Step 21: add some media files via wordpress website and find new uploads file in the wp-content/uploads/ directory

-We need to now sync the s3 bucket to with the wp-content directory

-Enter the cmd aws s3 sync –delete /var/www/html/wp-content/uploads/ s3:// mys3buketawsprojet2/

Note: we use –delete to ensure that all our directory wp-content is completely in sync with inserts/deletes that happen

Step 22: now how do we ensure to deliver the media content from cloud front ==> htaccess files

--Download the htaccess file to /var/www/html/ folder and rename it as .htaccess

-Now update the cloudfront url in the .htaccess file

-Nano .htaccess (update the cloud front URL)

Step 23: restart your httpd sever (service httpd restart)

Step 24: now try to go to your wordpress website and check out if the images are from the cloudfront or not

Step 25: setting up the crontab job

===> */2 * * * * root aws s3 sync –delete /var/www/html/ s3:// mys3buketawsprojet2/

===> */2 * * * * root aws s3 sync –delete /var/www/html/wp-content/uploads/ s3:// mys3bucketawsproject/

Step 26: try to upload a new file the media content of the WordPress server and now see that thumbnails will take little while the image as crontab sync will happen there by will deliver content

Step 27 : so we are ready to create AMI (go to console issue a request to new AMI)

Step 28: launch new EC2 instance with AMI and put this small bootstrap script and enable the role on instance

#!/bin/bash

yum update –y

aws s3 sync –delete s3://mys3bucketawsproject/ /var/www/html/

Step 29: Create Launch configuaration & then Auto scalling groups

Note: Before you perform above step, ensure that there are no Active EC2 instance

Step 30: verify the new instance EC2 instances, verify ELB & then check your web server

Step 31: simulate the scalling by stressing out the cpu (stress –cpu 150) to verify the scalling

Step 32: then we see that ASG scales up by introducing a new EC2 instance

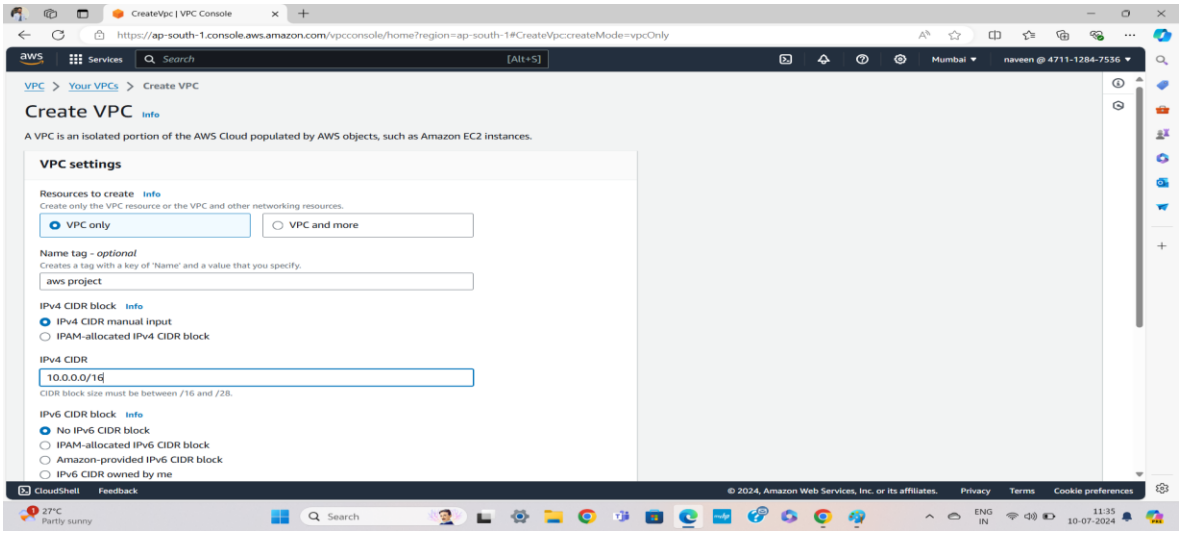
Step 33: you may also want to generate the failover of RDS instance (incase you are doing Multi AZ)

Go ahead and click on reboot the RDS instance, that will automatically cause failover

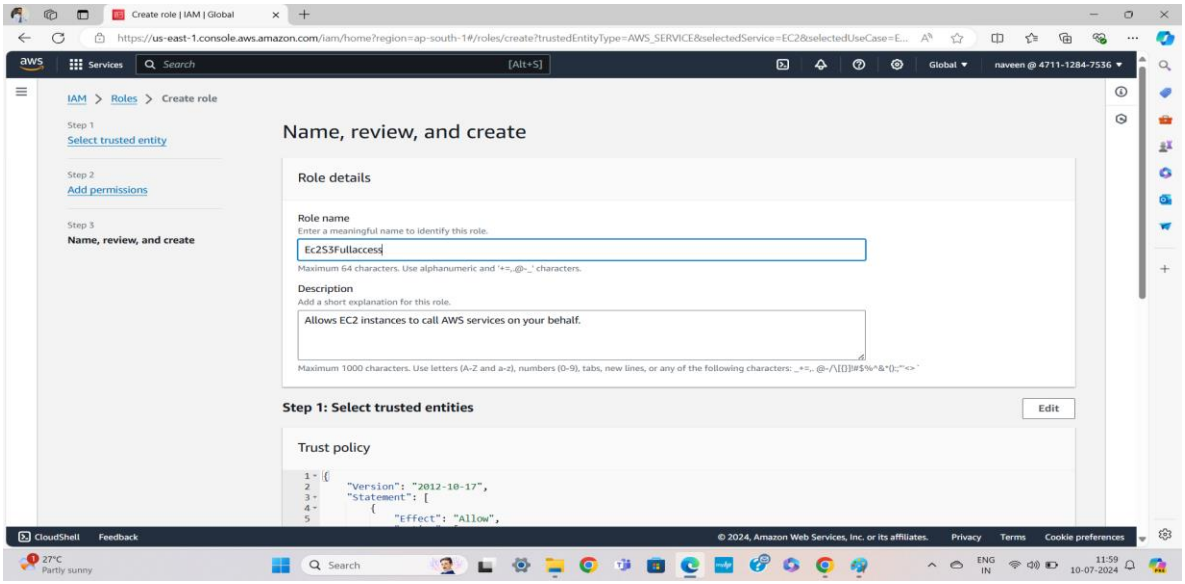
Step 34: verify that your WordPress server is still up and running in all the failover above

Step 1: Create a VPC for AWS Project and Create two public subnets and two private subnets within a VPC

My vpc is – 10.0.0.0/16



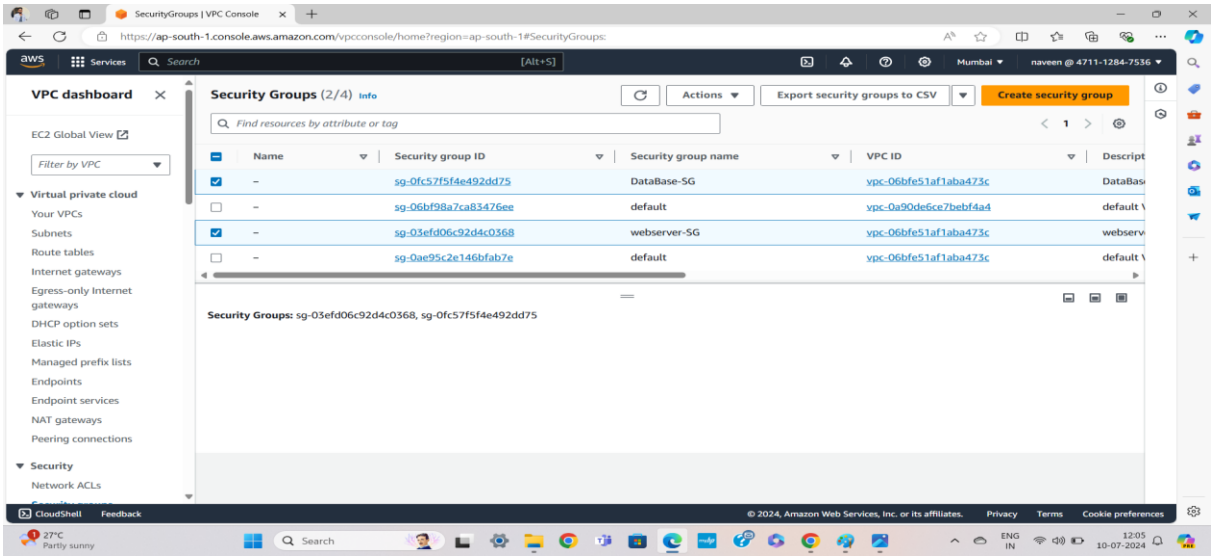
step 2 : Create a Role in IAM for S3 Full Access



Step 3: Create a Security Groups

- WebServer SG

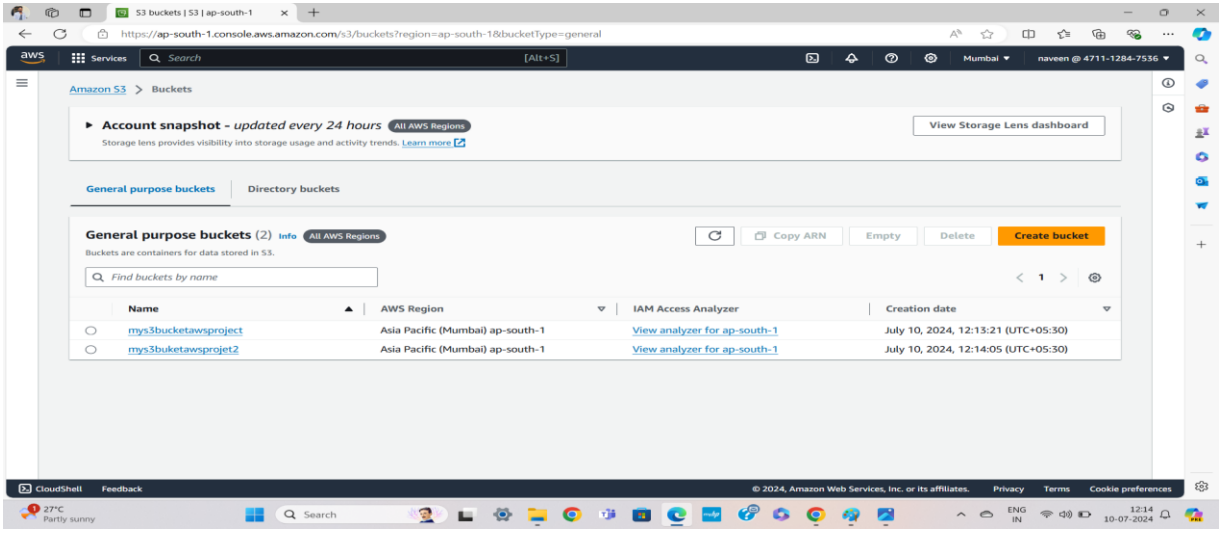
- RDS Security Groups



Step 4: Create buckets in S3 in same region where security groups were created

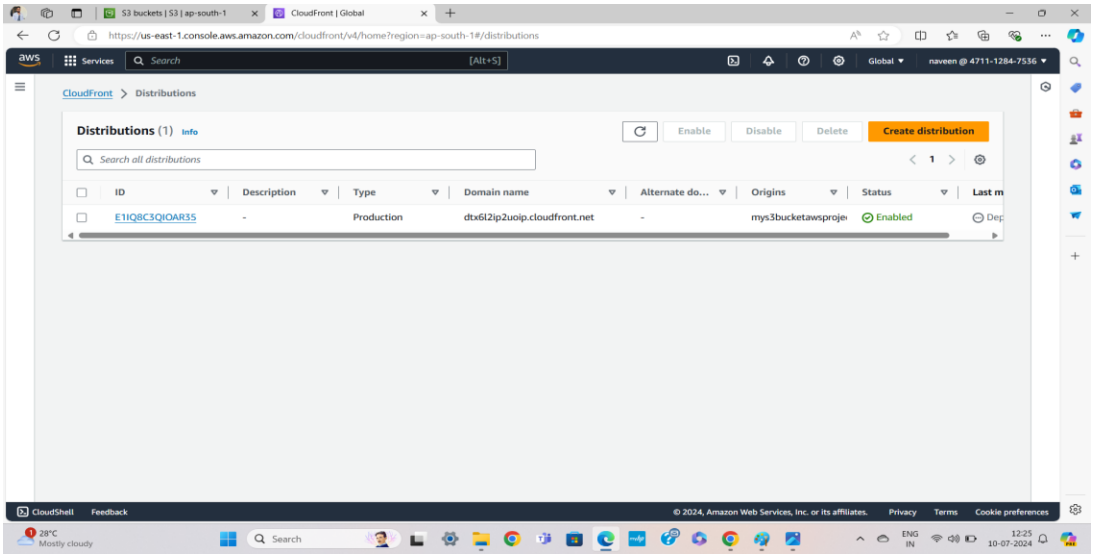
-mys3bucketawsproject

-mys3buketawsprojet2

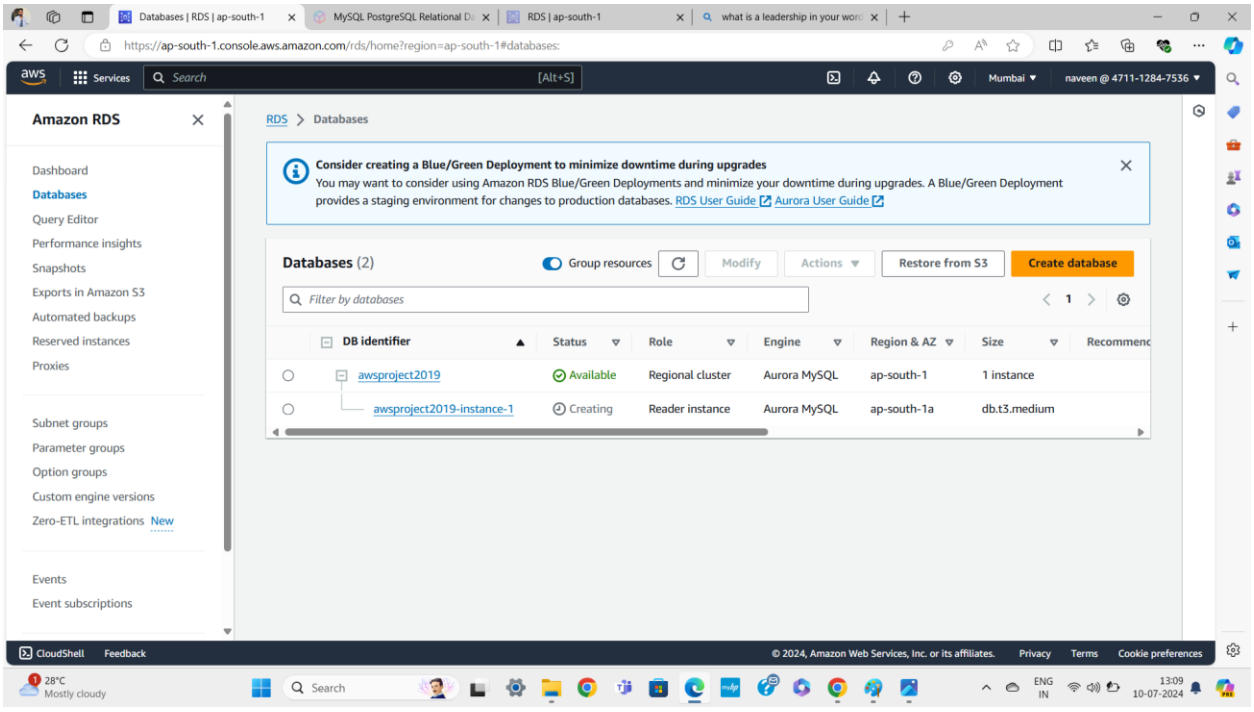


Step 5: Create cloud front Distribution

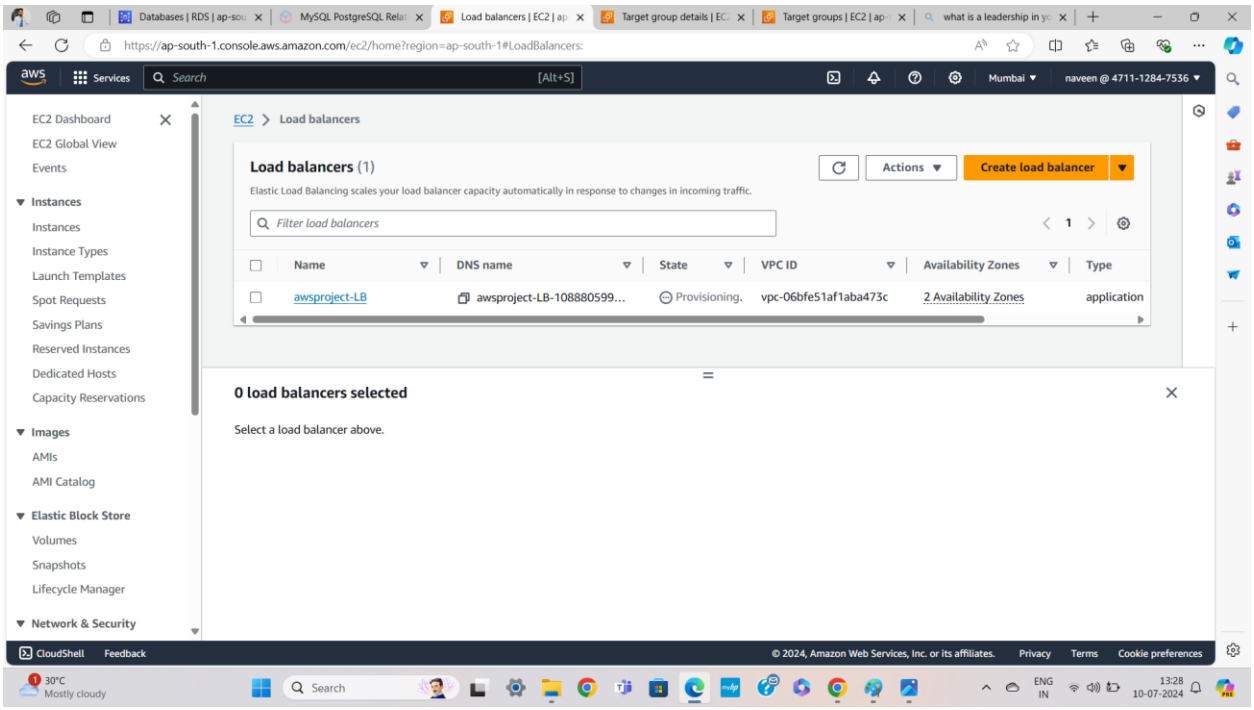
----Origin as mys3bucketawsproject.s3.ap-south-1.amazonaws.com and update new permission



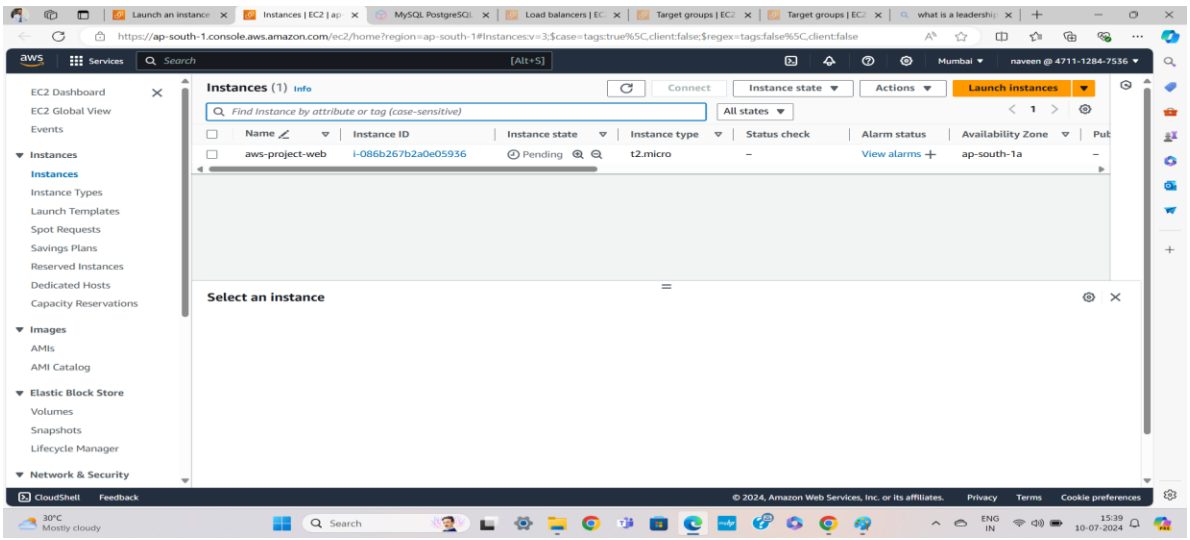
Step 6: Create RDS instance with MYSQL multi-AZ



Step 7: Create Elastic Load Balancer



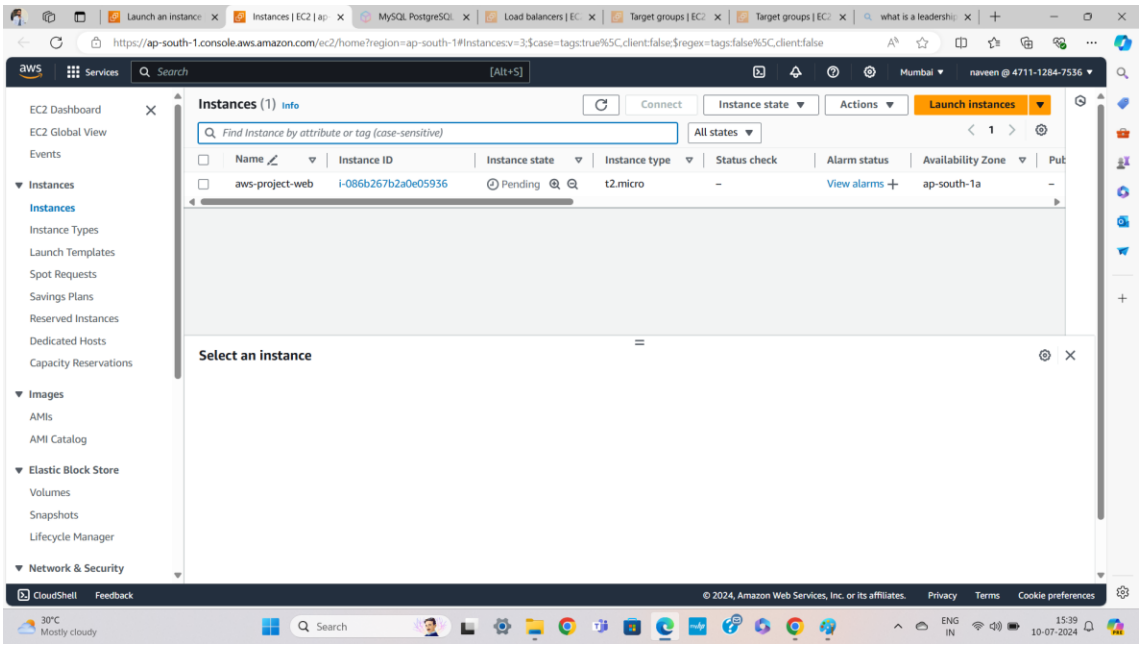
Step 8: Create a record set in the Route 53 and map the naked domain to ELB created in step 7 (my domain is dtx6l2ip2uoip.cloudfront.net)



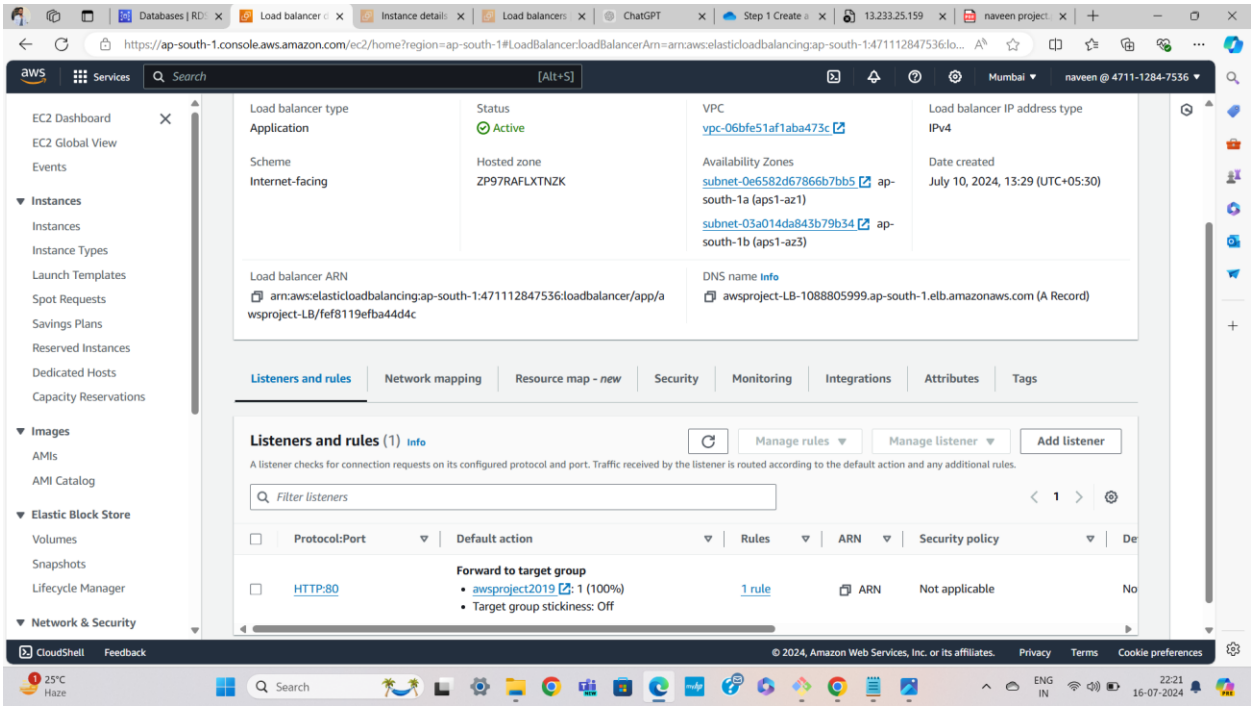
Step 9: Launch EC2 instance and just associate with the role created in the above step 2

Step 10: SSH into instance and run the scripts (installing the required software and patches)

```
#!/bin/bash
        Yum update –y
         Yum install httpd php php-mysql stress –y
        cd /etc/httpd/conf
         cp   httpd.conf  httpdconfbackup.conf
         rm –rf httpd.conf
         wget  httpd://s3-eu-west-1.amazon.com/ <bucketname>/httpd.conf
          cd /var/www/html/
          echo “welcome world” > naveen.html
          wget htpps://wordpress.org/latest.tar.gz
          tar –xzf latest.tar.gz
        cp –r wordpress/* /var/www/html/
        rm –rf wordpress
        rm –rf latest/tar.gz
         chmod –R 755 wp-content
        chown _R apache.apache wp-content
         service httpd start
         chkconfig httpd on
```



Step 11: Add the EC2 instance to the load balancer & log into domain name and configure wordpress website



Step 12: Create wp-content.php and copy the content from WordPress website and paste that

Step13: Try to access S3 bucket from EC2 instance

Note : attach IAM role to ec2 access the s3 bucket

Step14: Copy all the contents from the html folder to the S3 bucket

-Enter the cmd aws s3 cp –recursive /var/www/html/ s3://mys3bucketawsproject/

Step 15: delete the html folder and generate a real time failover

Step 16: recreate the html directory & now copy back the content from s3 bucket

Step 17: update the permissions

chmod –R 755 wp-content

chown _R apache.apache wp-content

Step 18: check out the directory wp-content/uploads/ and see that the media files that we have uploads on wordpress website appear

Step 19: Automate the media files to move to s3 bucket ---> mys3buketawsprojet2

-So that cloud front will serve the media files

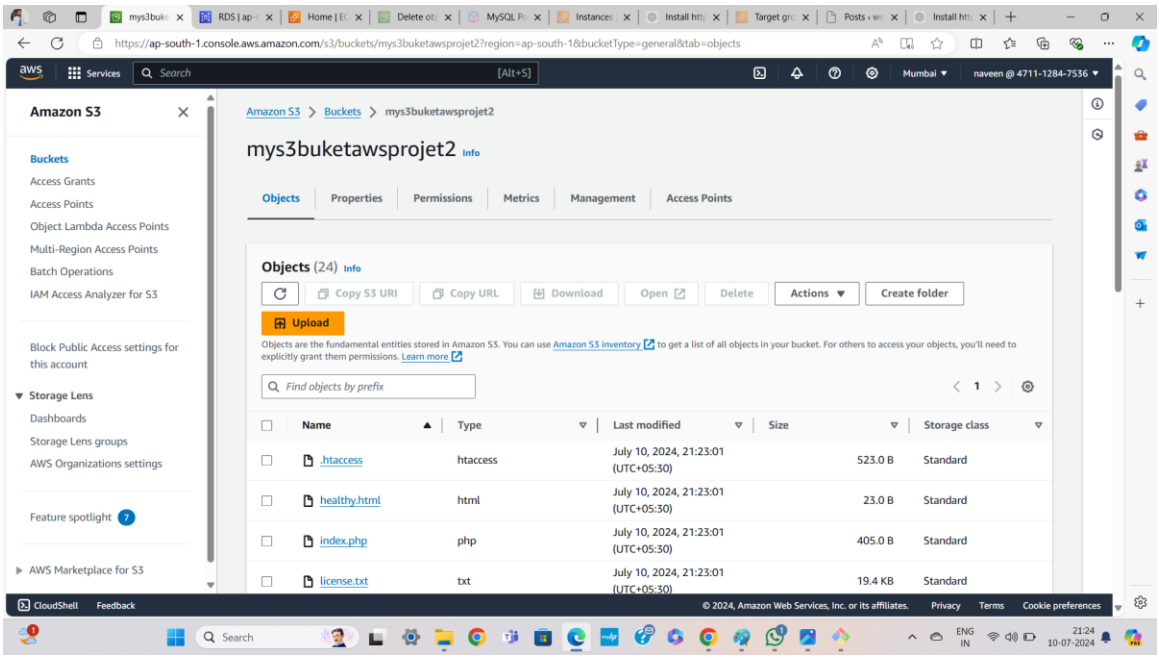
Step 20: Enter the cmd aws s3 cp --recursive /var/html/wp-content/uploads/ s3:// mys3buketawsprojet2/

Step 21: add some media files via wordpress website and find new uploads file in the wp-content/uploads/ directory

-We need to now sync the s3 bucket to with the wp-content directory

-Enter the cmd aws s3 sync –delete /var/www/html/wp-content/uploads/ s3:// mys3buketawsprojet2/

Note: we use –delete to ensure that all our directory wp-content is completely in sync with inserts/deletes that happen



Step 22: now how do we ensure to deliver the media content from cloud front ==> htaccess files

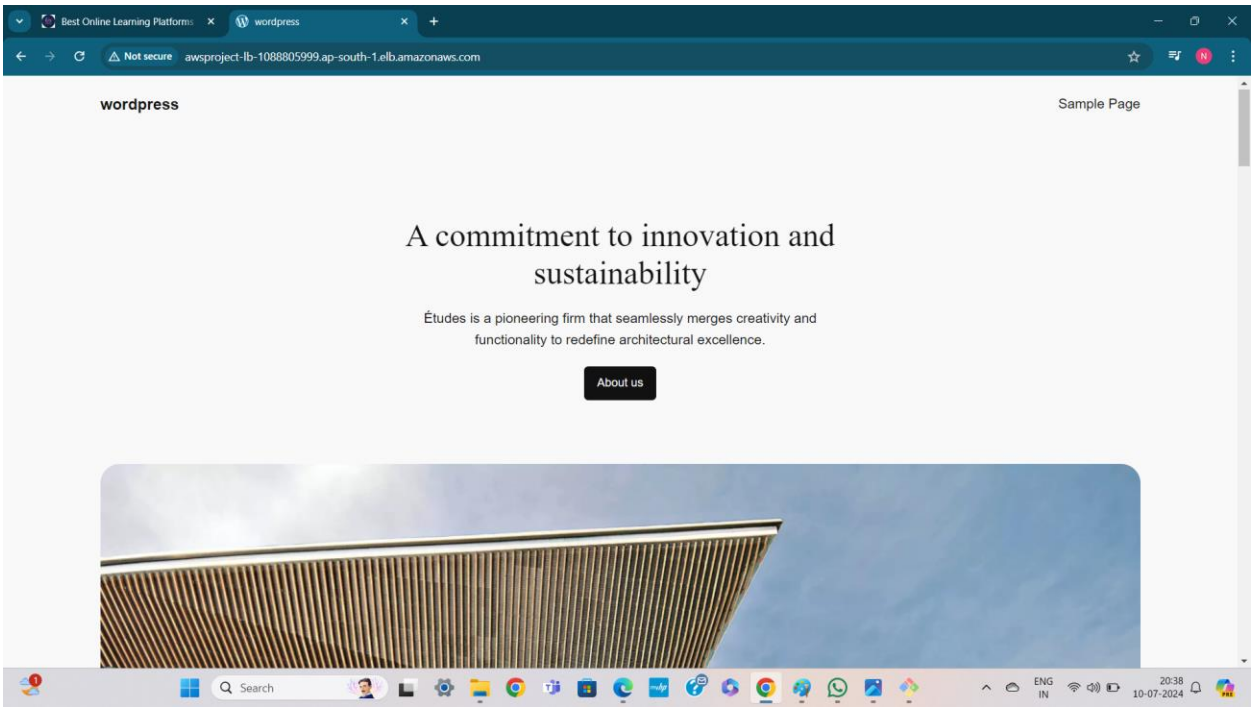
--Download the htaccess file to /var/www/html/ folder and rename it as .htaccess

-Now update the cloudfront url in the .htaccess file

-Nano .htaccess (update the cloud front URL)

Step 23: restart your httpd sever (service httpd restart)

Step 24: now try to go to your wordpress website and check out if the images are from the cloudfront or not



Step 25: setting up the crontab job

===> */2 * * * * root aws s3 sync –delete /var/www/html/ s3:// mys3buketawsprojet2/

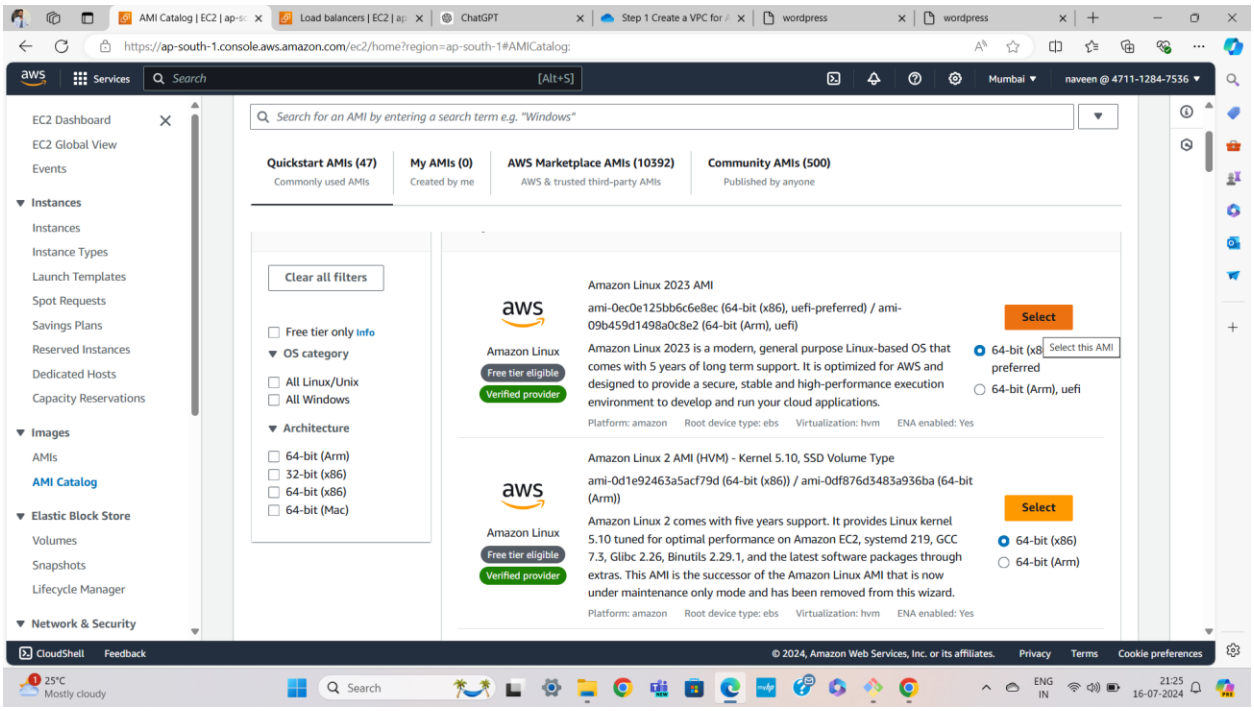
===> */2 * * * * root aws s3 sync –delete /var/www/html/wp-content/uploads/ s3:// mys3bucketawsproject/

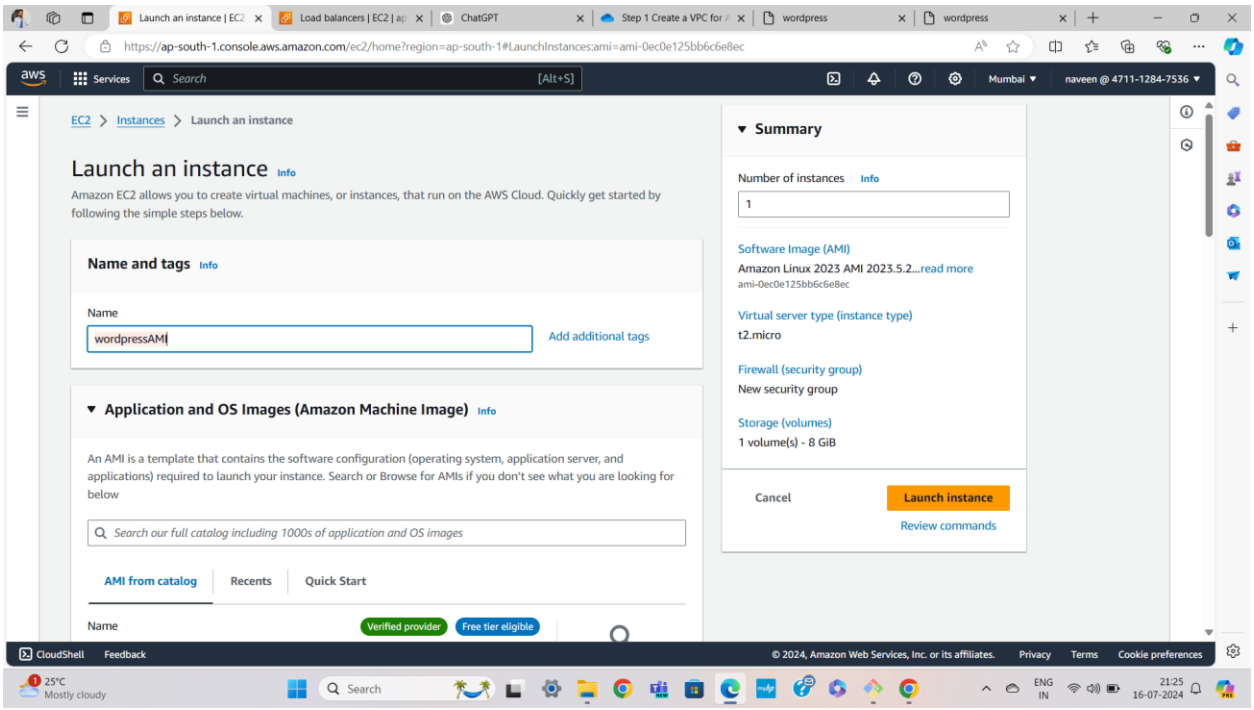
Step 26: try to upload a new file the media content of the WordPress server and now see that thumbnails will take little while the image as crontab sync will happen there by will deliver content

Step 27 : so we are ready to create AMI (go to console issue a request to new AMI)

Step 28: launch new EC2 instance with AMI and put this small bootstrap script and enable the role on instance

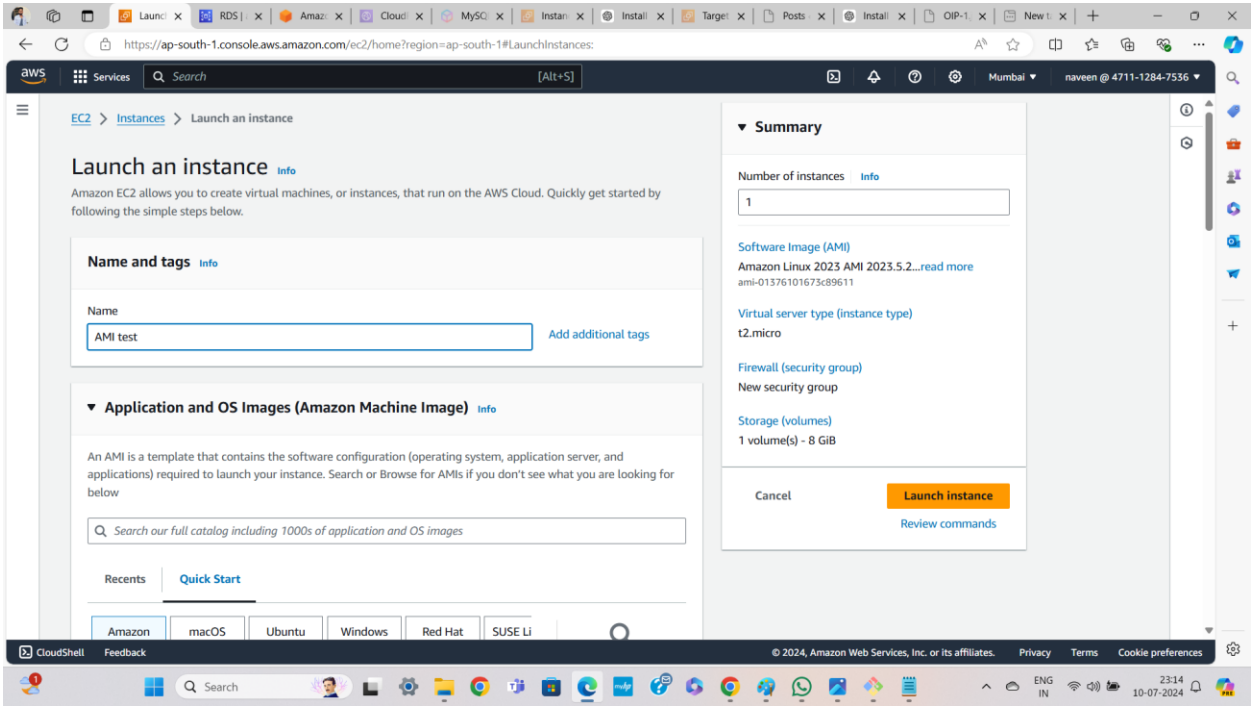
```
#!/bin/bash
yum update –y
aws s3 sync –delete s3://mys3bucketawsproject/ /var/www/html/
```

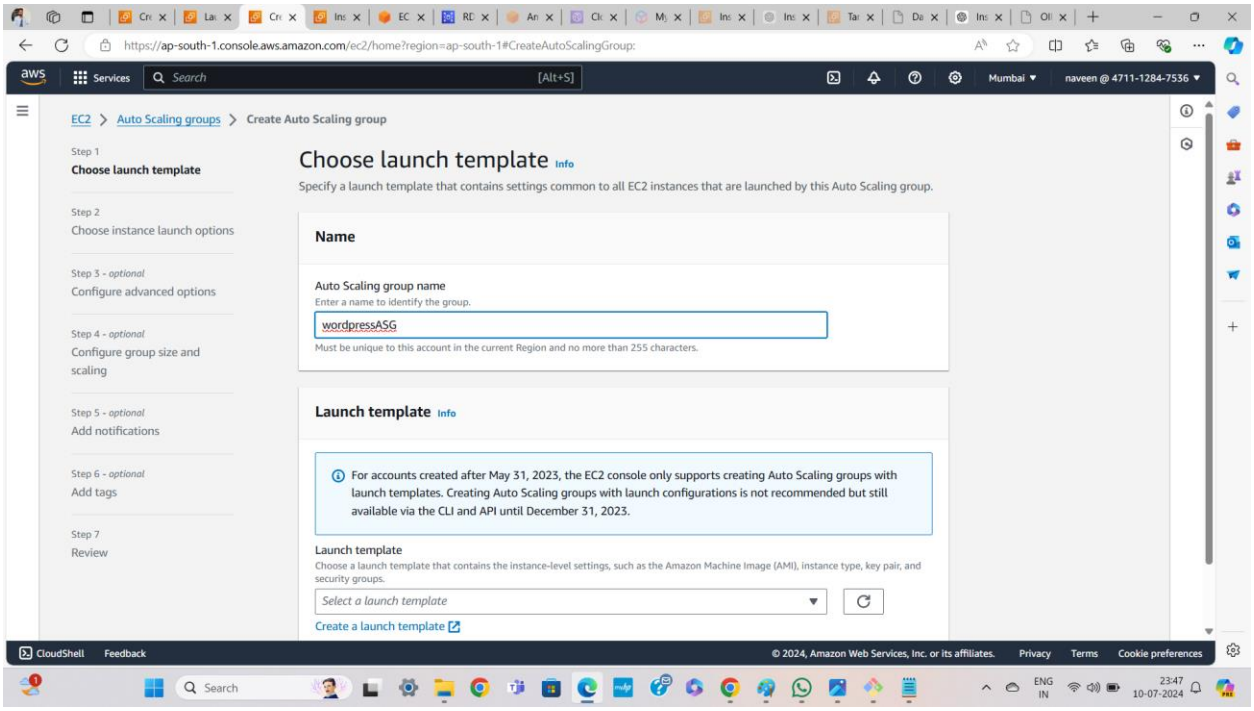




Step 29: Create Launch configuaration & then Auto scalling groups

Note: Before you perform above step, ensure that there are no Active EC2 instance





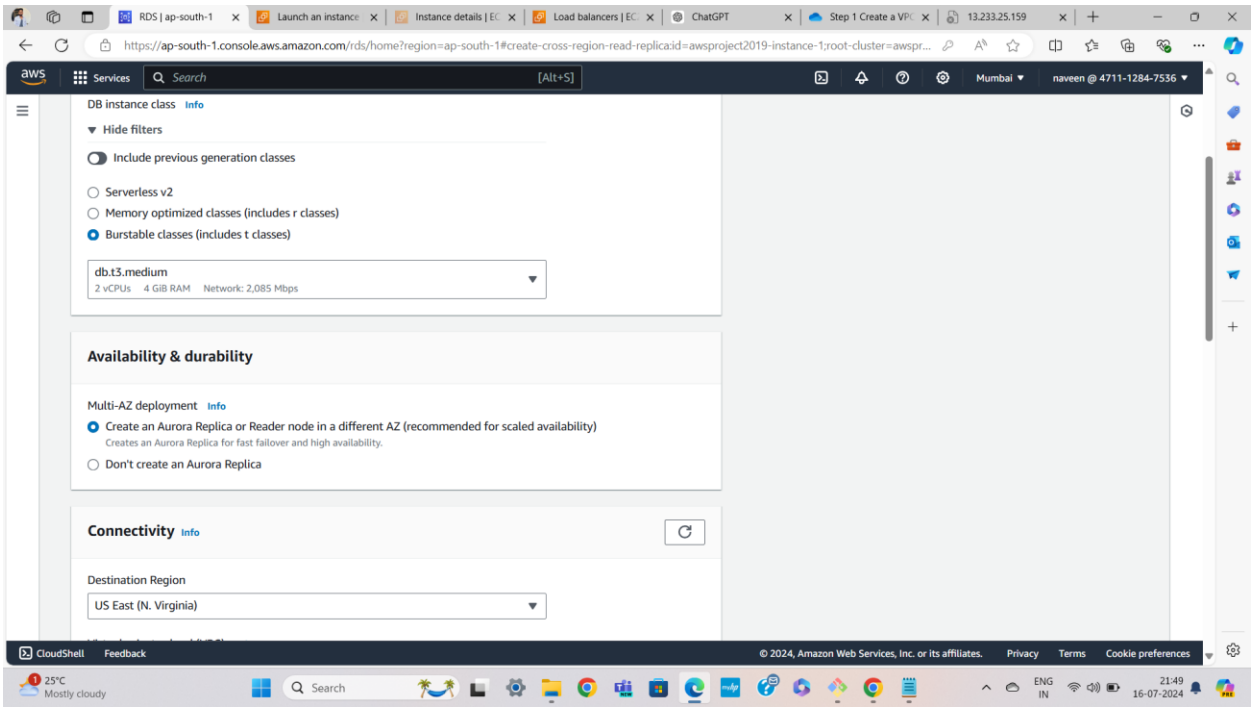
Step 30: verify the new instance EC2 instances, verify ELB & then check your web server

Step 31: simulate the scalling by stressing out the cpu (stress –cpu 150) to verify the scalling

Step 32: then we see that ASG scales up by introducing a new EC2 instance

Step 33: you may also want to generate the failover of RDS instance (incase you are doing Multi AZ)

Go ahead and click on reboot the RDS instance, that will automatically cause failover



Step 34: verify that your WordPress server is still up and running in all the failover above.