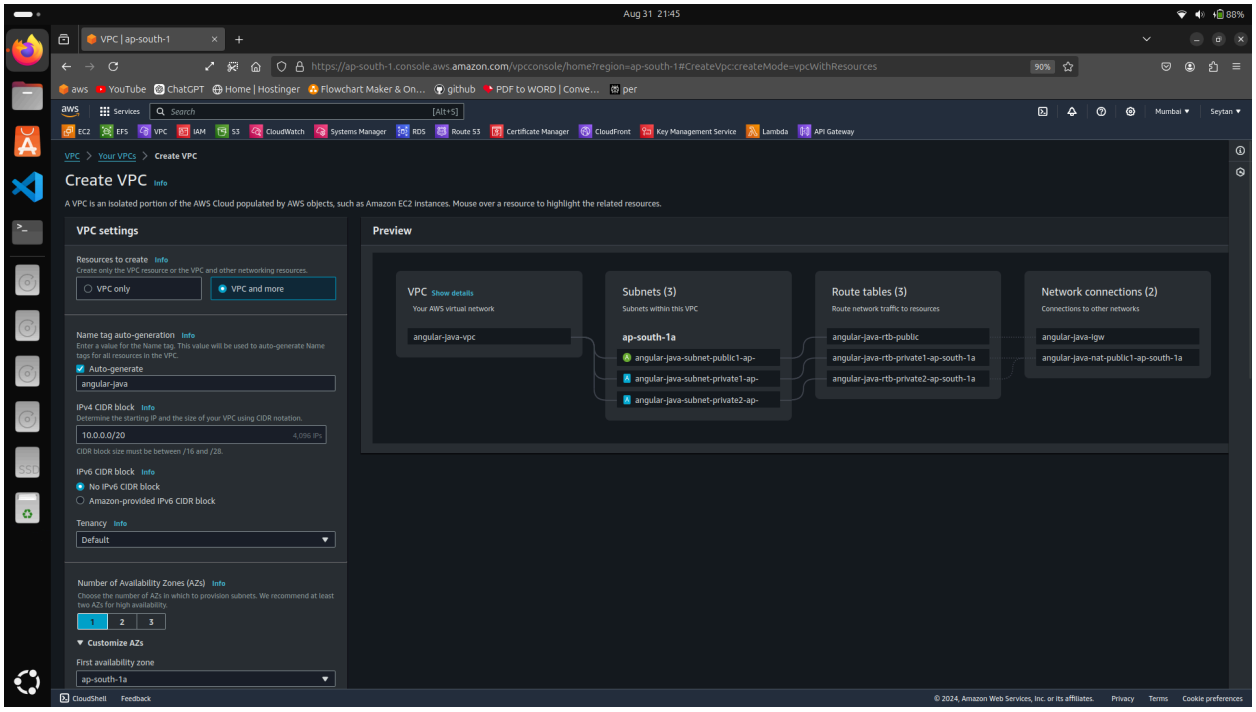
Task : **Secure 3-Tier AWS Deployment : Angular Frontend, Java Backend, RDS**

Diagrammatic Representation:
https://drive.google.com/file/d/1w9YR_SkHr1GC3JmxukGFPqMwSeWVHbR8/view?usp=sharing

Step 1: Create a VPC with 3 subnets (1 public and 2 private+ add nat to private subnets).

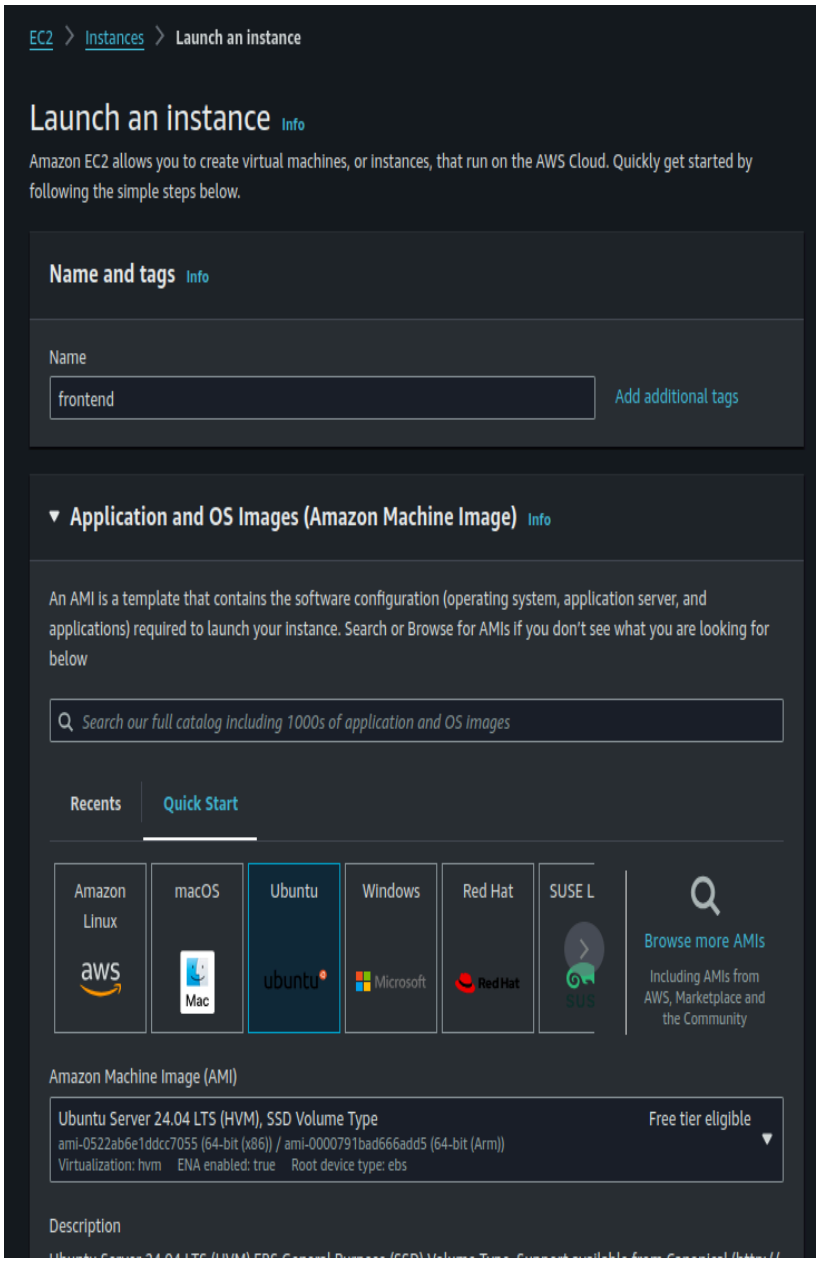
What is vpc ,subnet,nat gateway? →

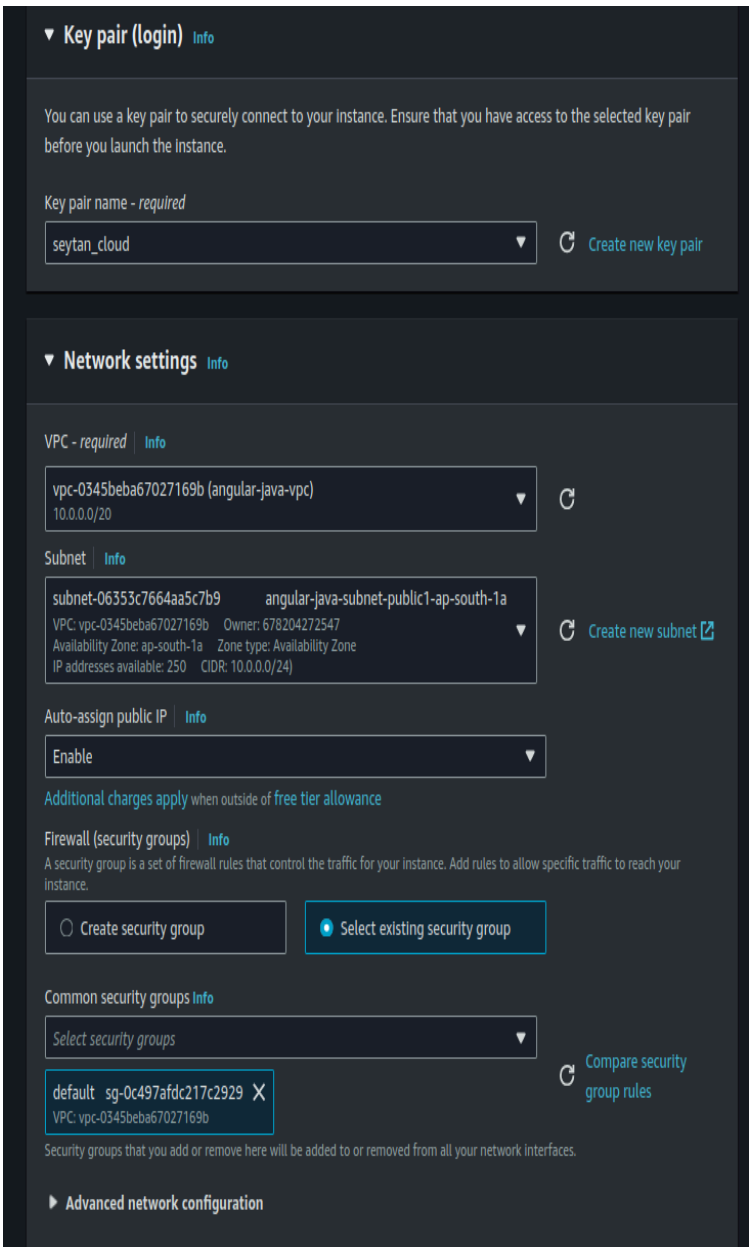
1. A VPC (Virtual Private Cloud) is an isolated network within AWS where you can launch resources.
2. subnet is a segment of a VPC's IP address range where resources are placed.
3. A NAT Gateway allows instances in a private subnet to access the internet while keeping them secure.



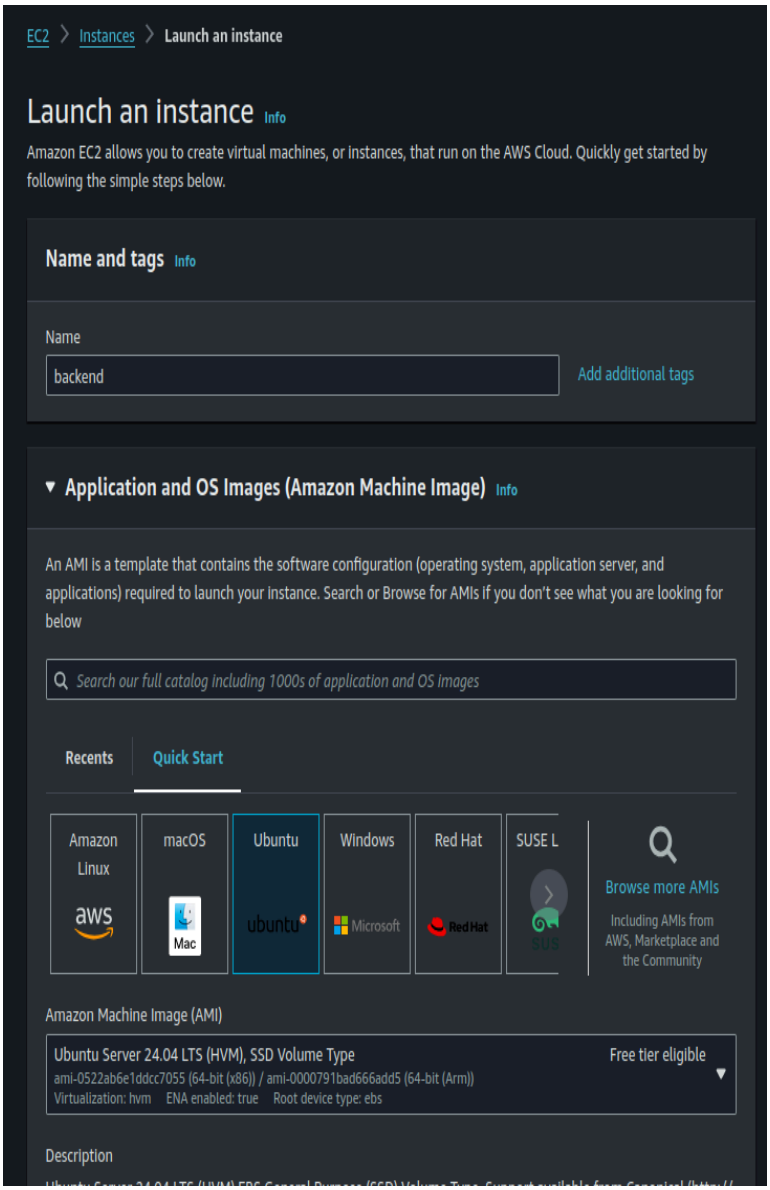
Step 2: Now Launch 3 instances (frontend server, backend server, database server.

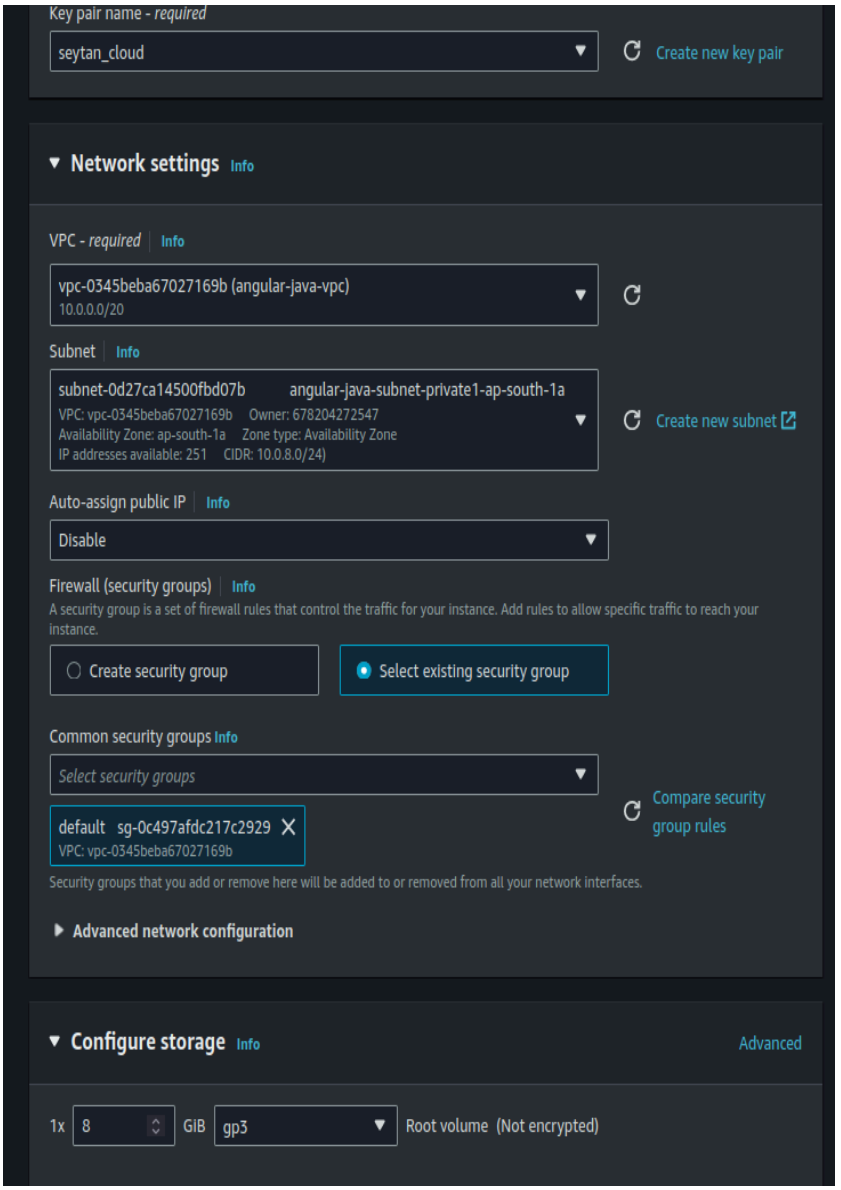
 # <u>For the frontend server</u> chose the Networking settings and edit it. Select the our newly created vpc and select the public subnet also make sure to enable the auto assing ip for the fronted server.



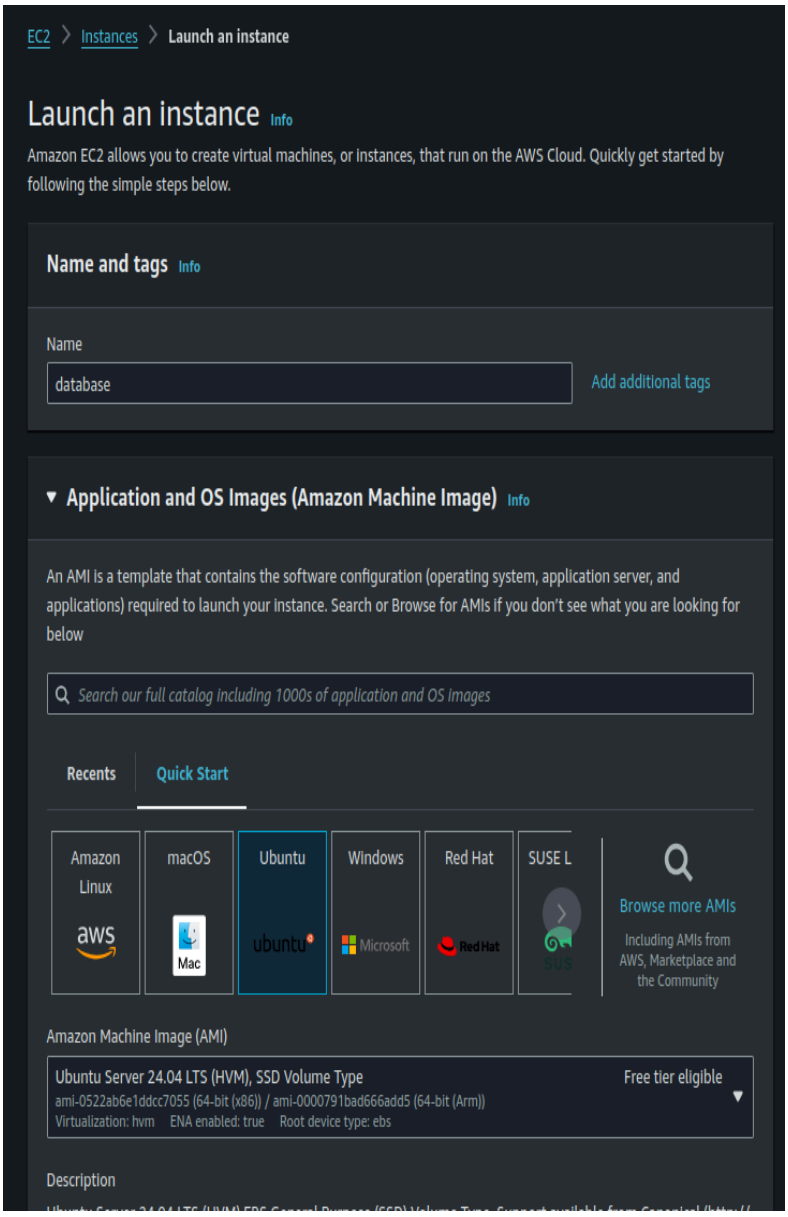


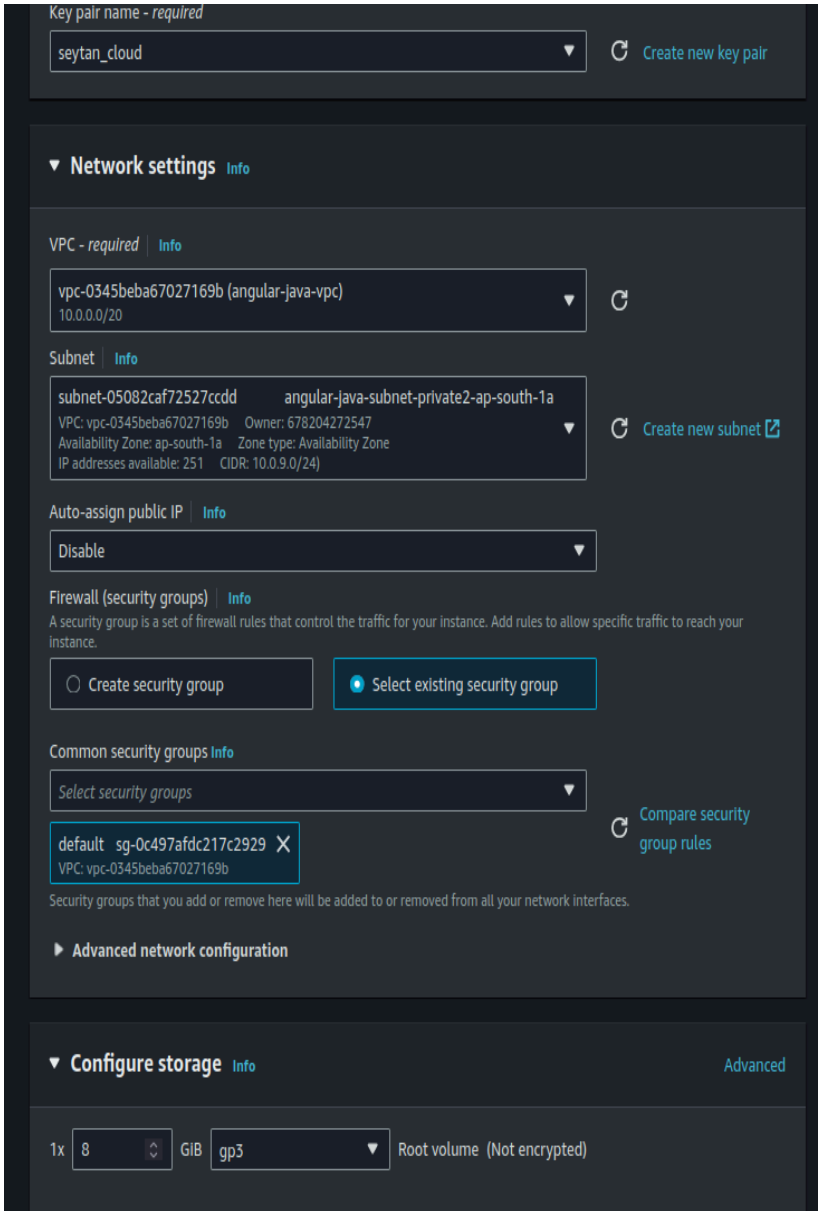
<u>Now create the Backend Server</u> and select the appropriate vpc and private subnet 1 and disable the auto assing publi ip from network settings.



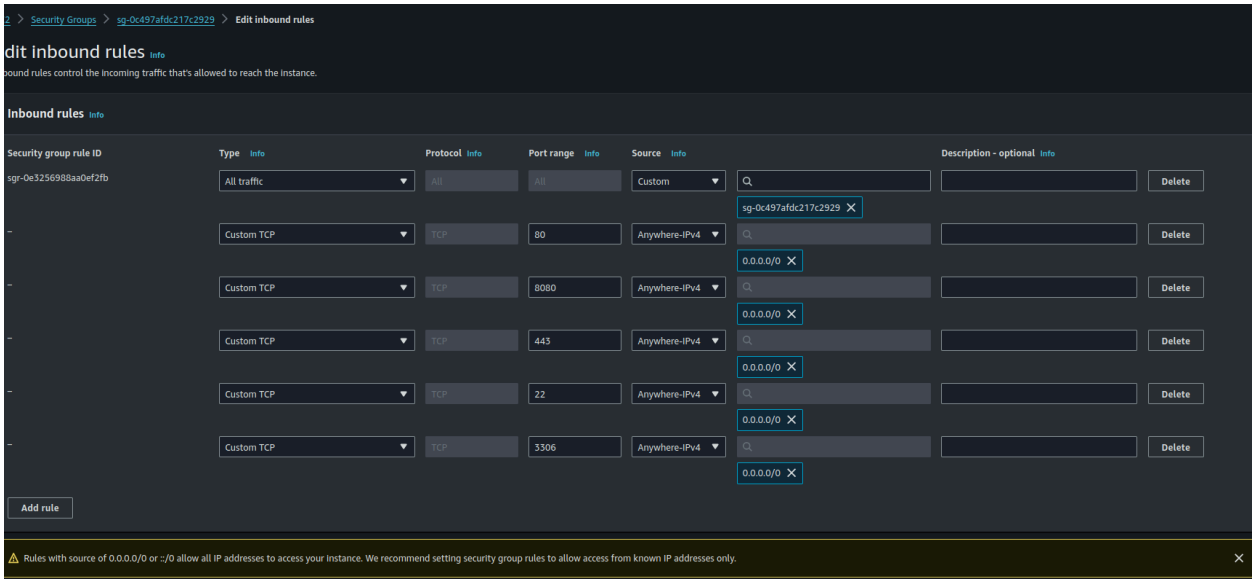


<u>Now create the final database server</u> , and perform the same instructions on this which were applied on backend while creating.



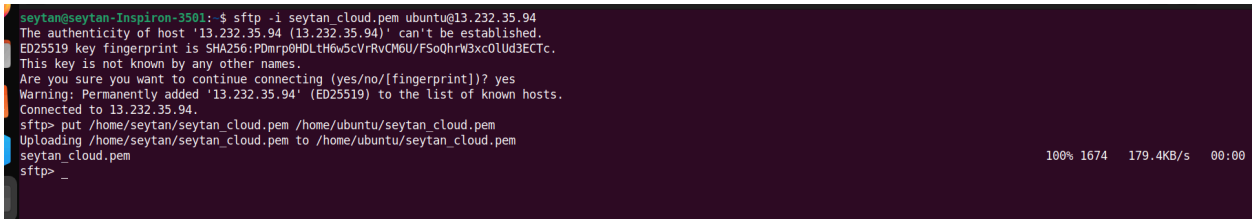


Step 3: Whitelist the following ports in the security group of servers. (22,443,80,8080,3306)



Step 4: Now as we are gonna connect to the private servers through bastion server i.e. from public servers ,we need the private key pair so we will transfer this key from our local machine to the frontend server using sftp .

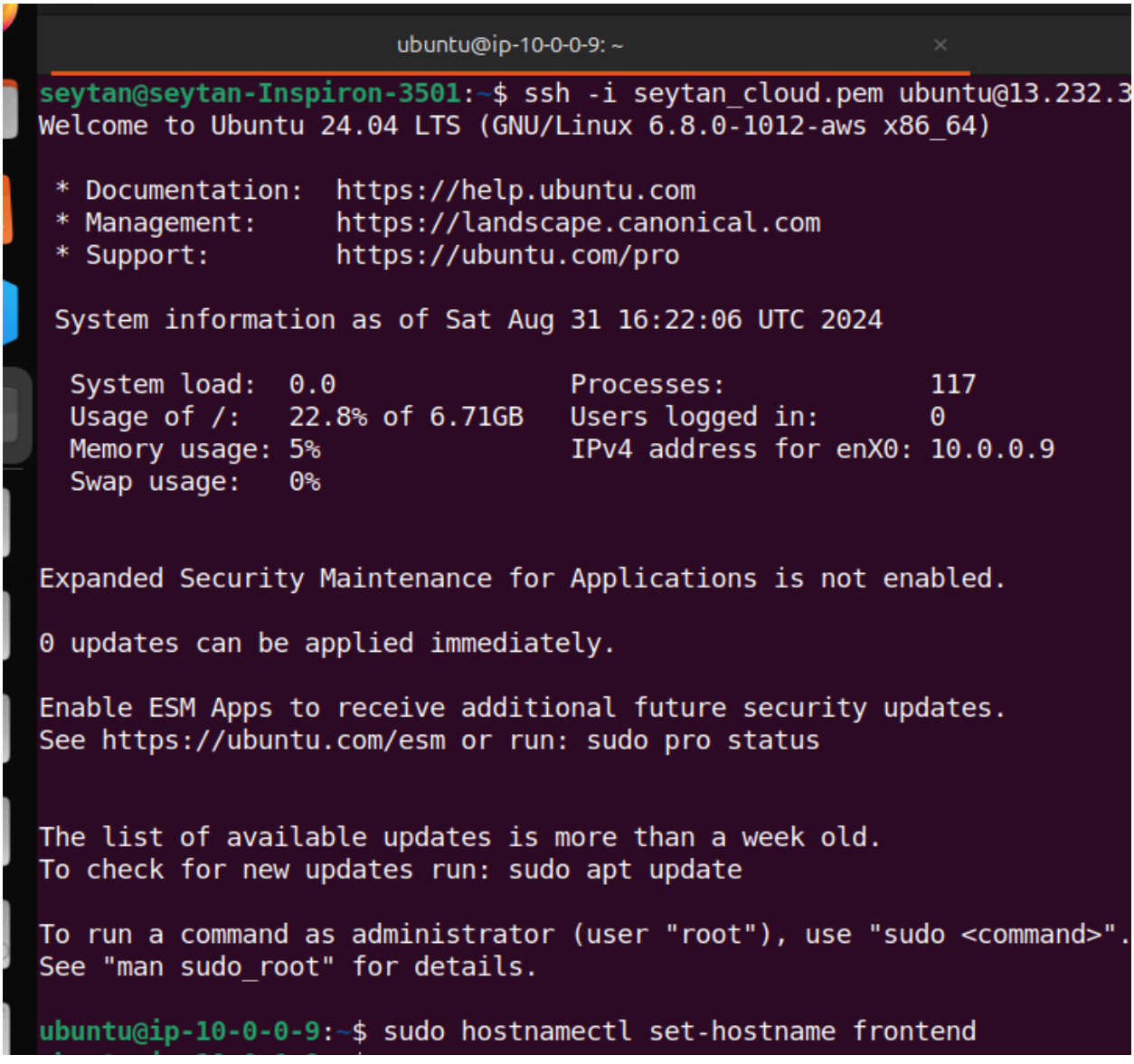
What is sftp ? →
SFTP (Secure File Transfer Protocol) is a secure version of FTP (File Transfer Protocol) that uses SSH (Secure Shell) to encrypt data during transfer, ensuring secure file exchange over a network.

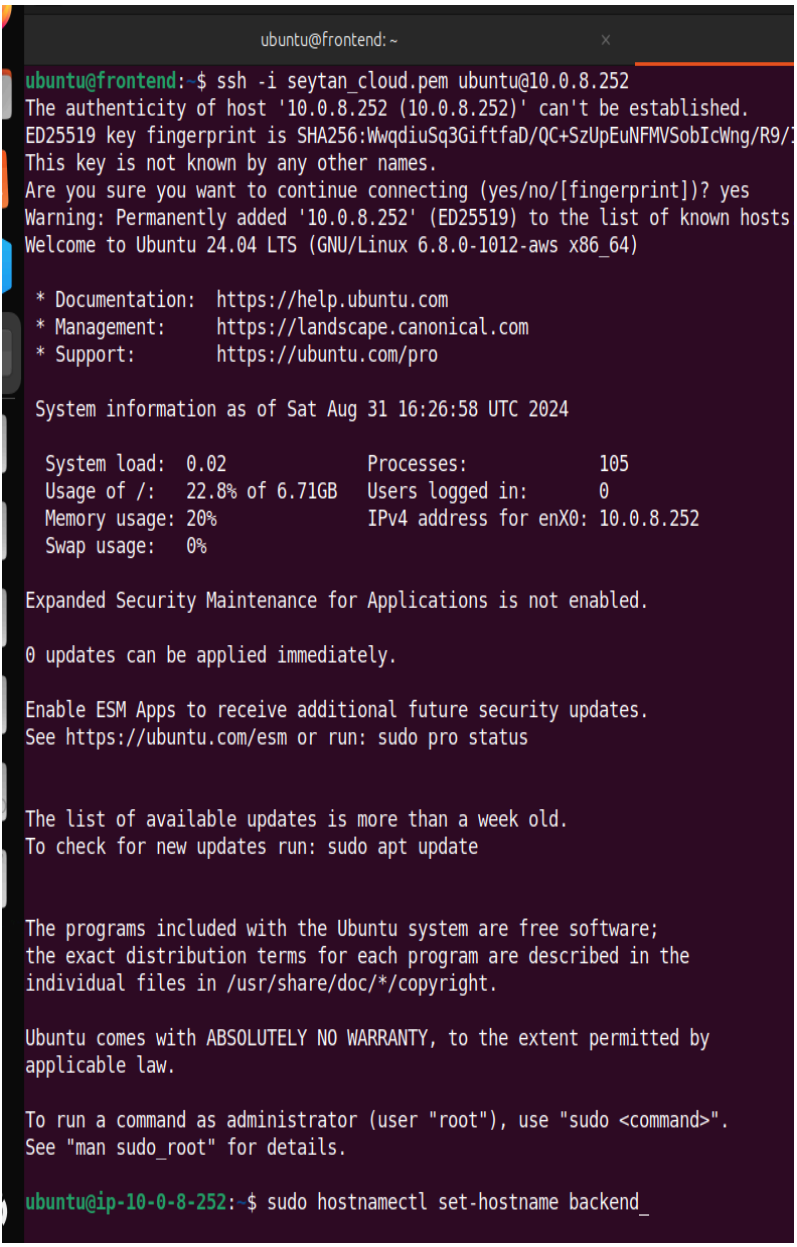


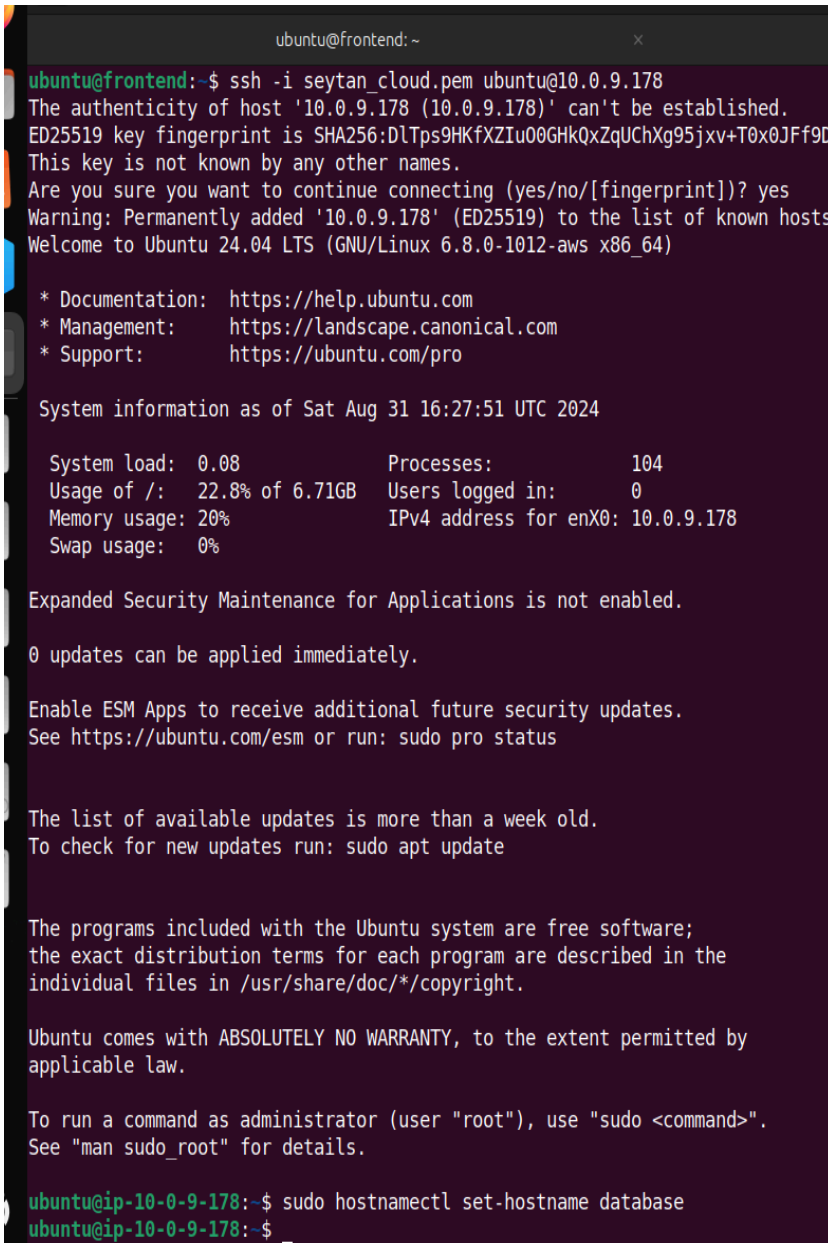
Step 5: Now Connect to bastion server or jump server (frontend server) and connect to the backend and database server from it.

What is Bastion or jump servers ? →
Public servers that provide secure access to a private network are called bastion servers or jump servers.

Use the sudo hostnamectl set-hostname command to change the server names accordingly.

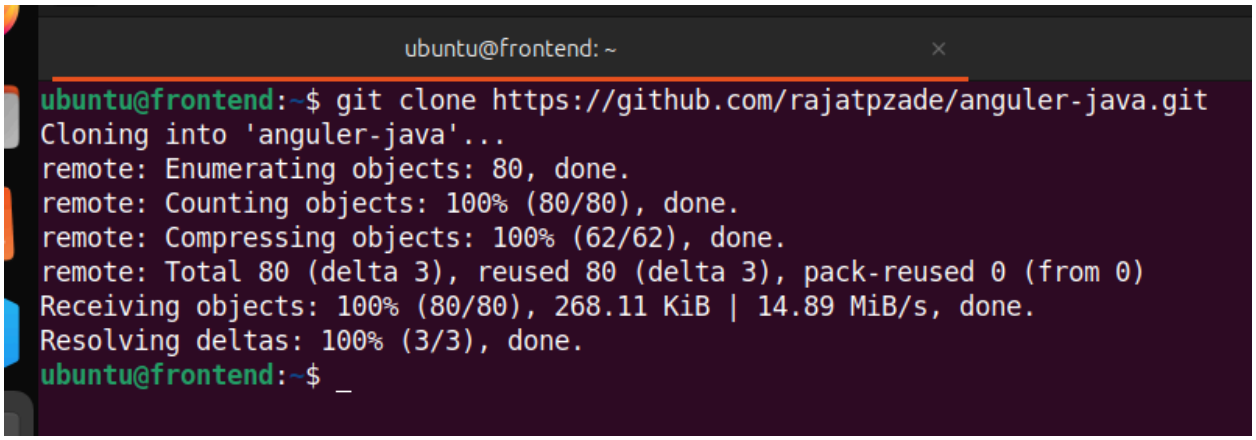


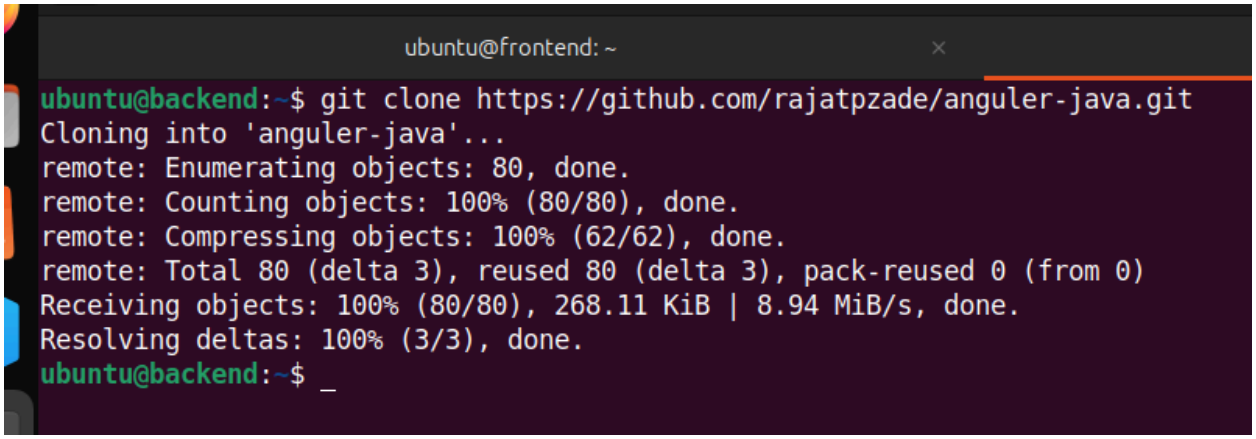




Backend and Database

Step 6: Now clone the github repository in all 3 servers. Using the command
git clone https://github.com/rajatpzade/angular-java.git





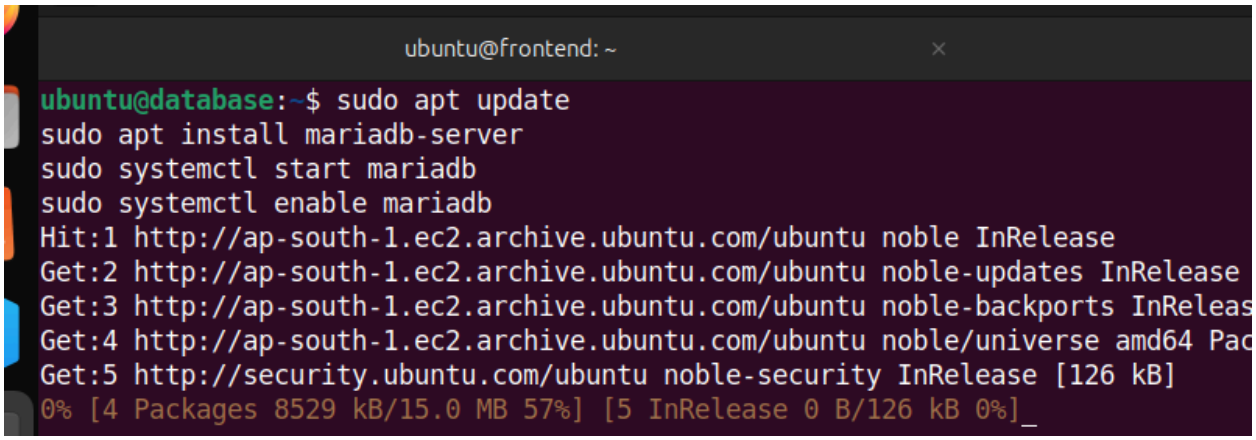
Step 7: Now go to the database server and run following commands →

sudo apt update

sudo apt install mariadb-server

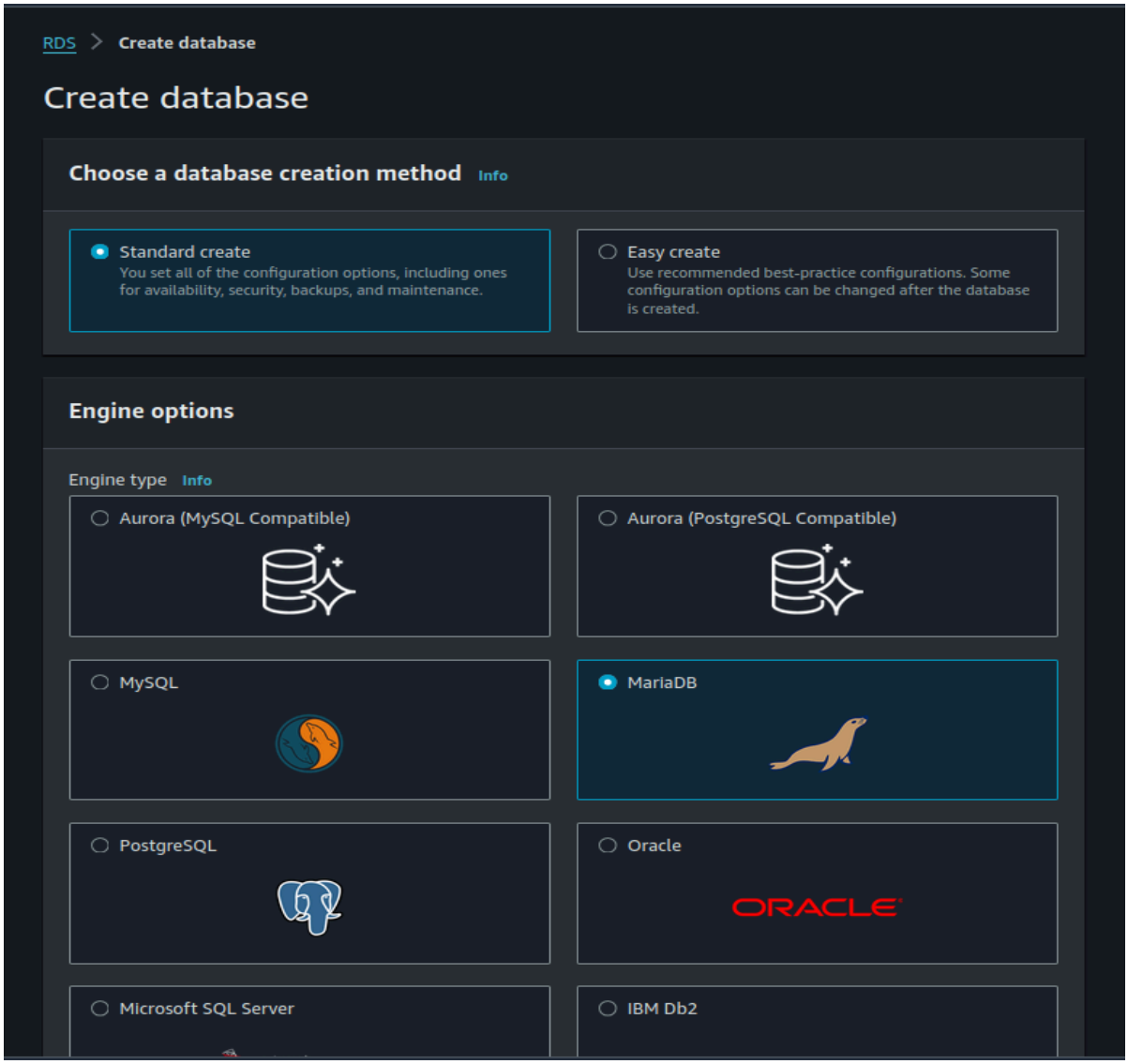
sudo systemctl start mariadb-server

sudo systemctl enable mariadb-server

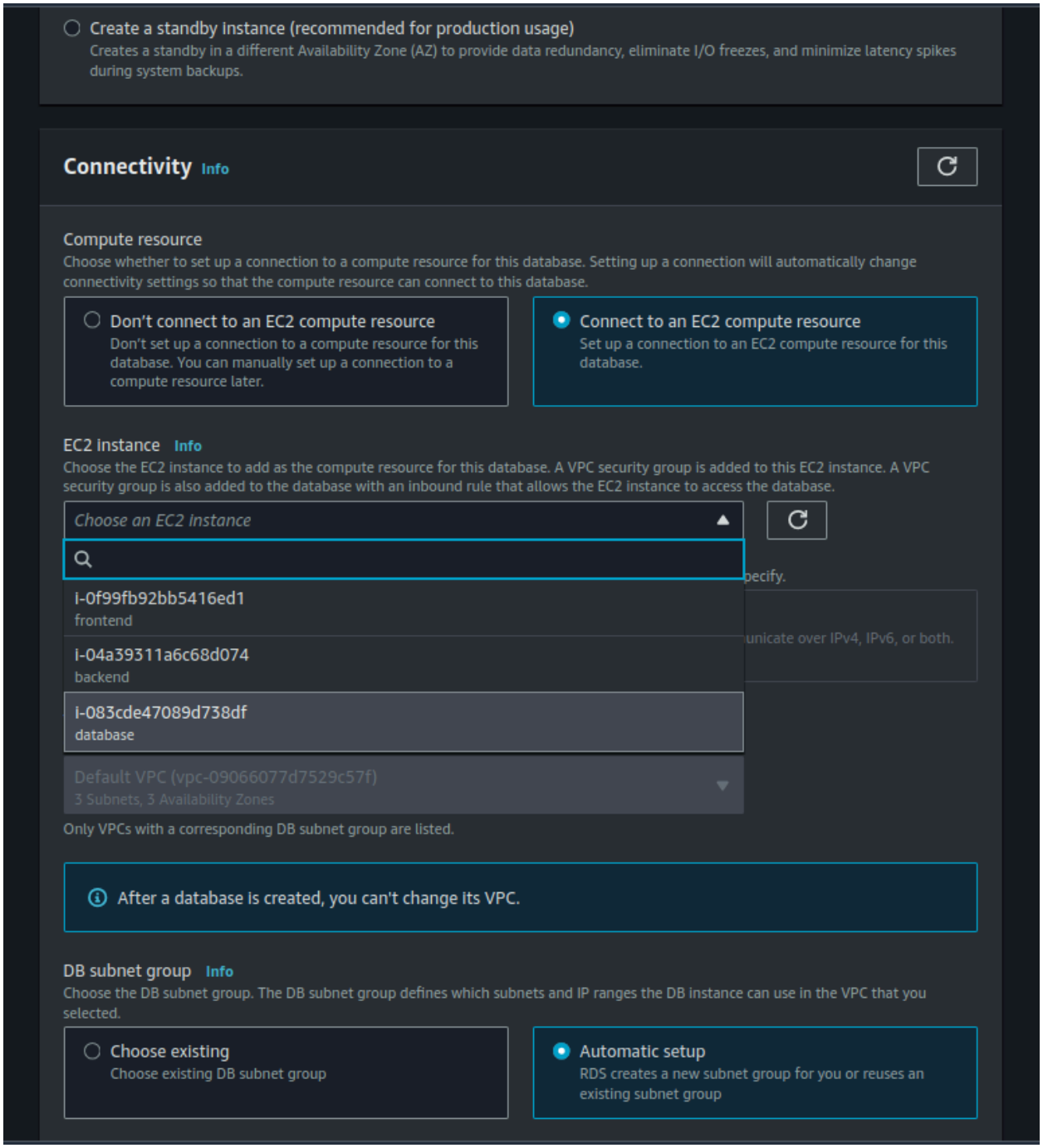


Step 8: Now Create a RDS Prodcution Database named angular-java and choose mariadb as engine.

What is RDS ? →
RDS (Relational Database Service) is a managed database service provided by AWS that simplifies the setup, operation, and scaling of a relational database in the cloud. It supports multiple database engines, including MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB.



Now in connectivity configuration select the connect to and EC2 compute resource option and select the database instance in it.



Step 9: Now connect to the RDS database from database server using

sudo mysql -h rds-endpoint -u admin -p

Once connected execute the following commands:

CREATE DATABASE springbackend;

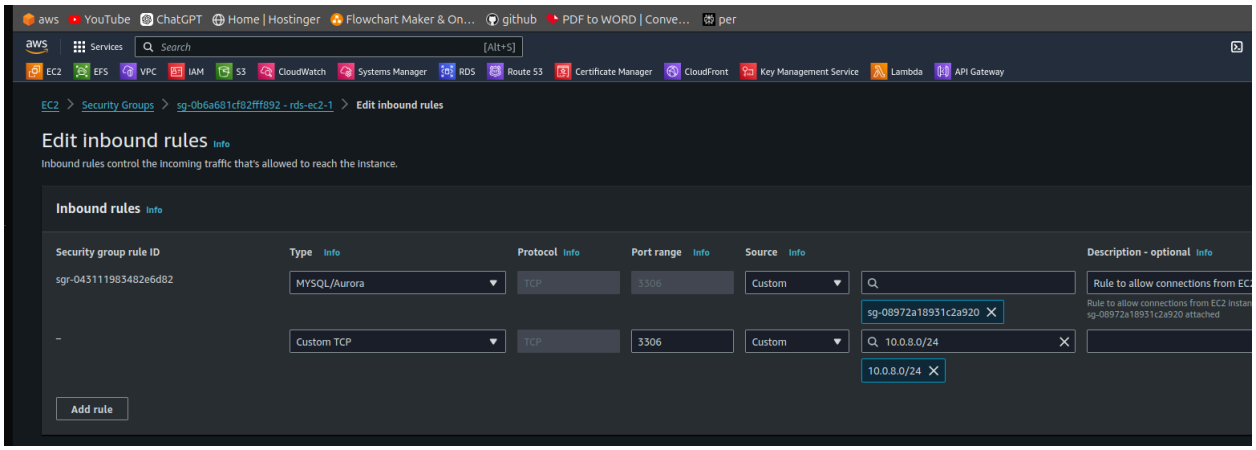
GRANT ALL PRIVILEGES ON springbackend.* TO ‘admin’@’backend-subnet_cidr’ IDETIFIED BY ‘database_pasword’

Once it is done do exit and run the next command from database server shell :

cd angular-java/

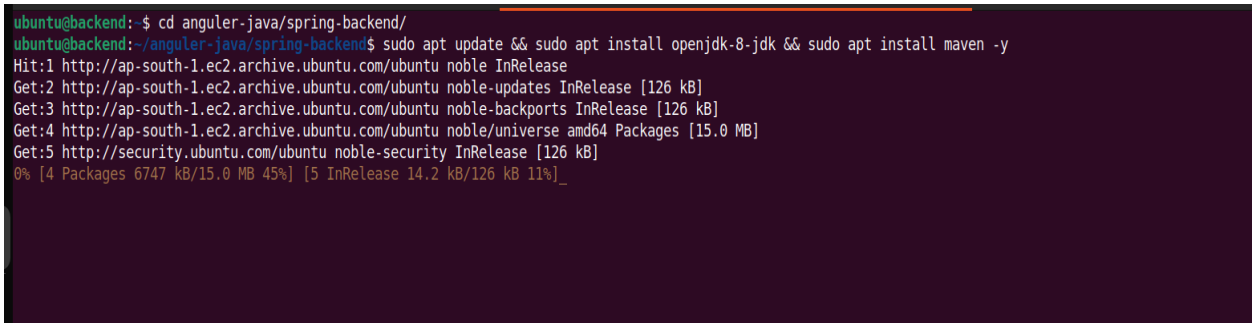
sudo mysql -h rds-endpoint -u admin -p springbackend < springbackend.sql

Edit the inbound rules for the rds security group and add the backend cidr for port 3306 allowing backend to make connection with the rds database .



Step 10: Now go to the backend server and run the following commands →

sudo apt update && sudo apt install openjdk-8-jdk && sudo apt install maven -y

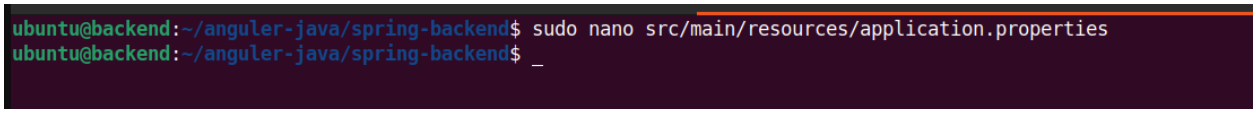


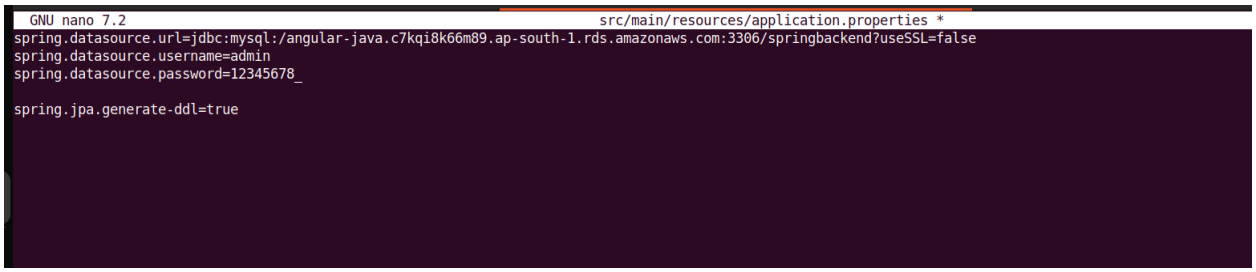
Step 11 : Now go and edit the file application.properties present at the location

→

angular-java/springabackend/src/main/resources/application.properties

Add the rds endpoint in place localhost and username as rds username and password as rds password.





Now build the backend using following command

→

mvn clean package -Dmaven.test.skip=true

Step 12 : Now go to the frontend server and navigate to the angular-java/angular-frontend directory and run the following commands.

sudo apt update

sudo apt install nodejs npm

```
ubuntu@frontend:~$ cd anguler-java/angular-frontend/
ubuntu@frontend:~/anguler-java/angular-frontend$ sudo apt update
sudo apt install nodejs npm
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [469 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [117 kB]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [7768 B]
```

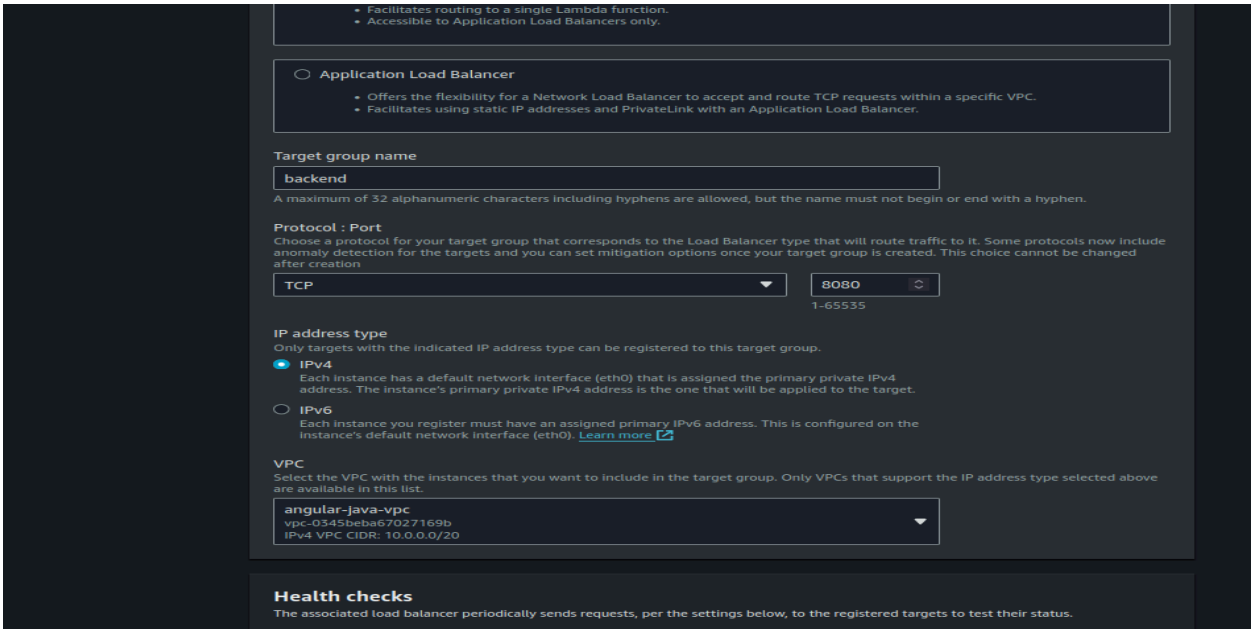
Now run the next commands :
sudo npm install -g @angular/cli@14.2.1

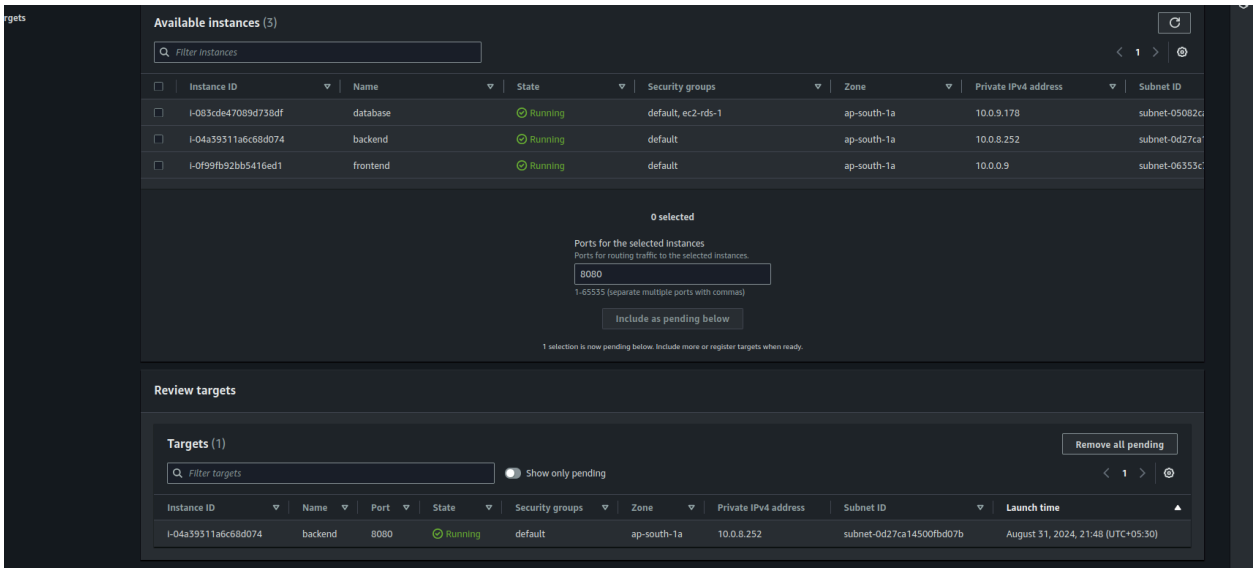
npm install

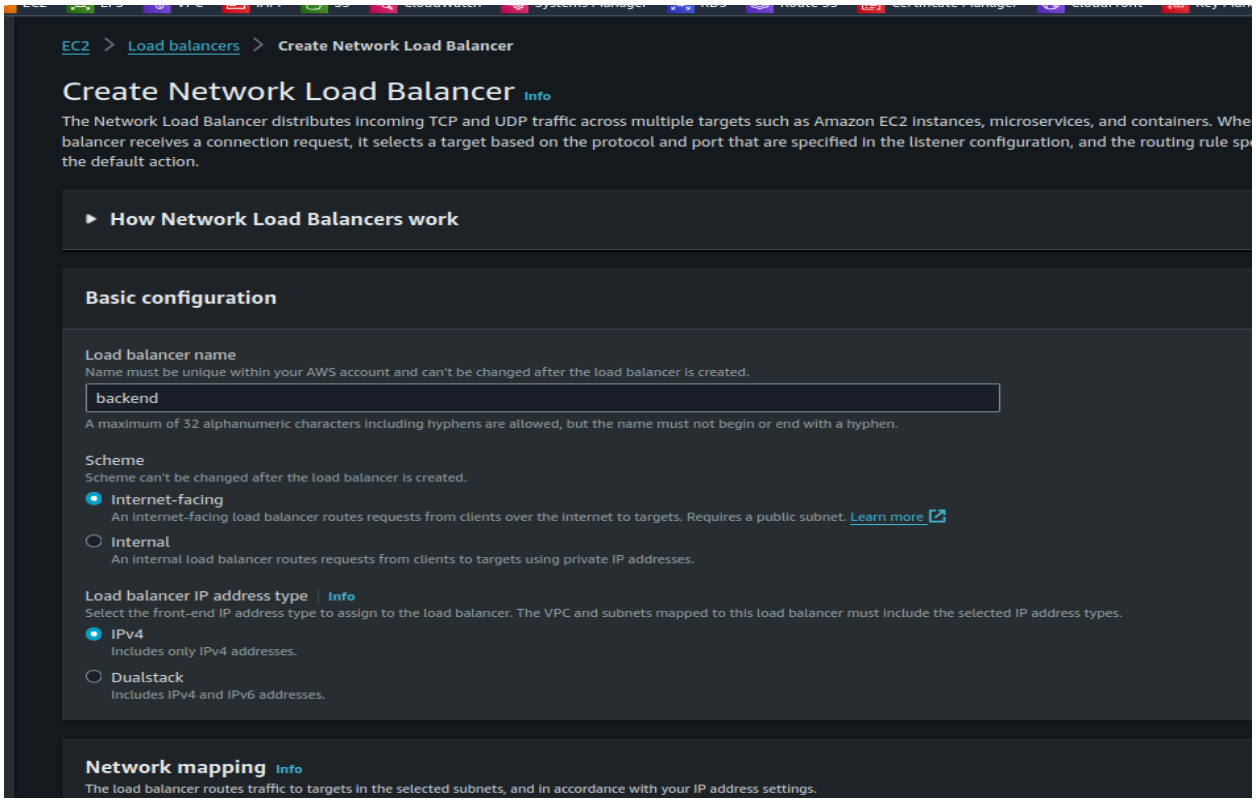
```
ubuntu@frontend:~/anguler-java/angular-frontend$ sudo npm install -g @angular/cli@14.2.1
npm WARN deprecated npmlog@6.0.2: This package is no longer supported.
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use
a key value, which is much more comprehensive and powerful.
npm WARN deprecated @npmcli/move-file@2.0.1: This functionality has been moved to @npmcli/fs
npm WARN deprecated read-package-json@5.0.2: This package is no longer supported. Please use @
npm WARN deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated are-we-there-yet@3.0.1: This package is no longer supported.
npm WARN deprecated glob@8.1.0: Glob versions prior to v9 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated gauge@4.0.4: This package is no longer supported.

added 212 packages in 11s

27 packages are looking for funding
  run `npm fund` for details
ubuntu@frontend:~/anguler-java/angular-frontend$ npm install

added 918 packages, and audited 919 packages in 29s

124 packages are looking for funding
  run `npm fund` for details

23 vulnerabilities (6 moderate, 13 high, 4 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
ubuntu@frontend:~/anguler-java/angular-frontend$ _
```

Step 13: Now as we need to provide the ip of the backend server in the worker.service.ts file . But as the backend is running on private server/subnet it is not accessible over internet so we have to create a Network load balancer for the backend as well as frontend to access the frontend over the browser using nginx server.

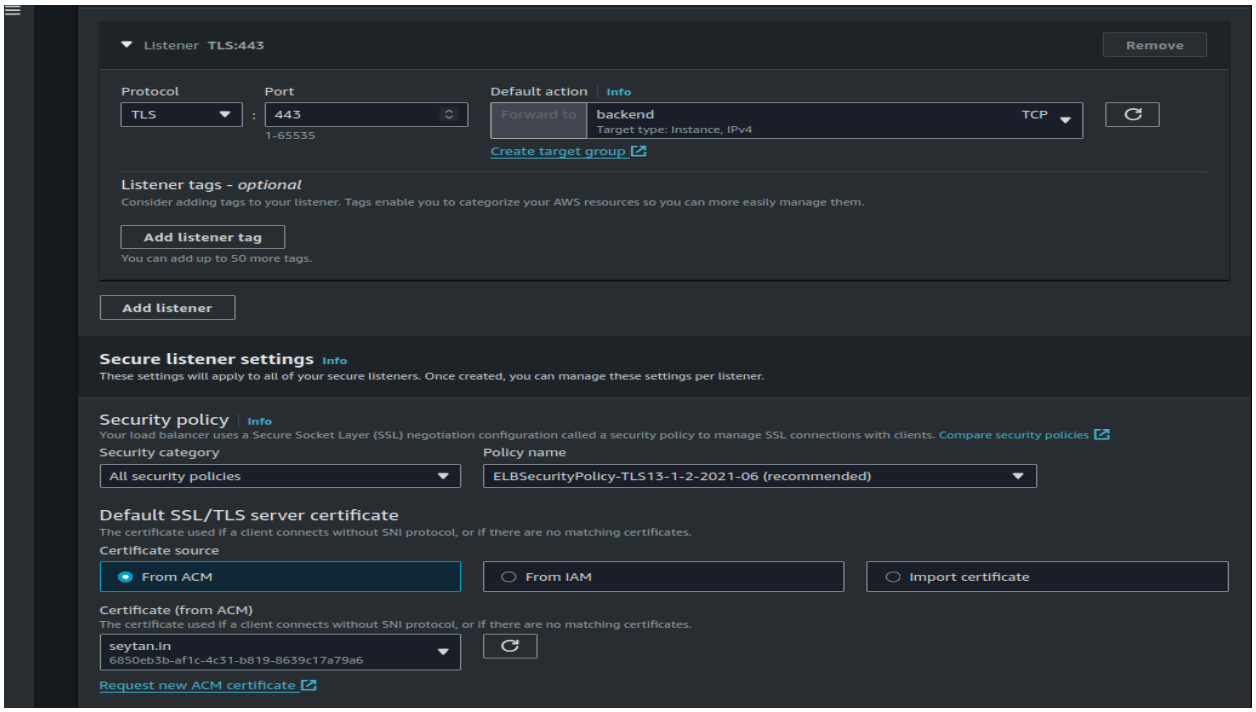
First Create a target group for the backend and use tcp 8080 .





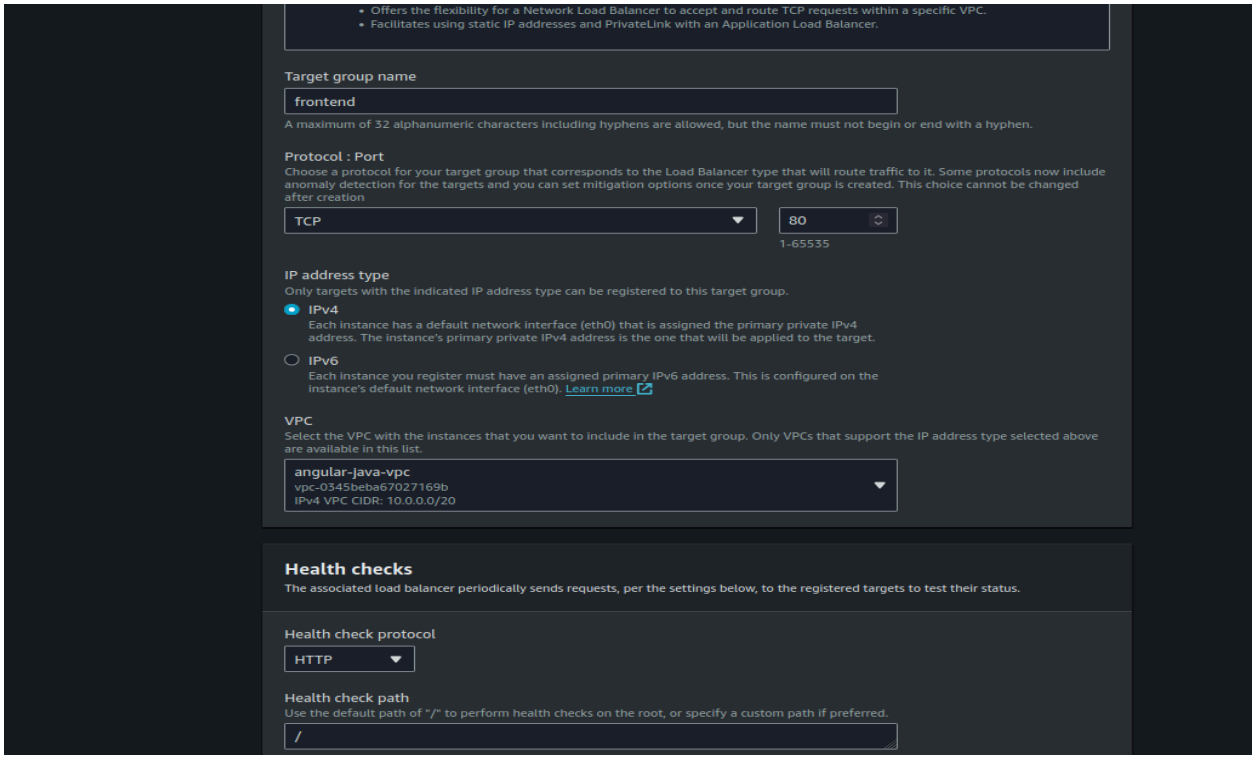


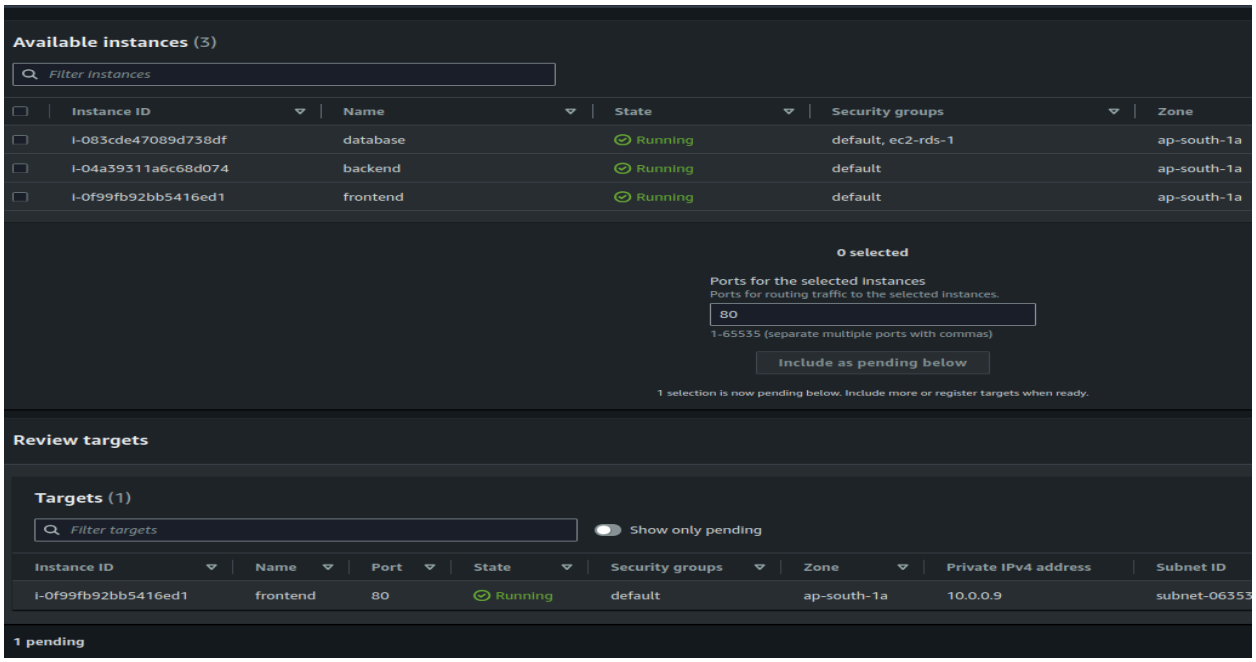
Add the protocol tls and port 443 ,you have to add the certificate also .



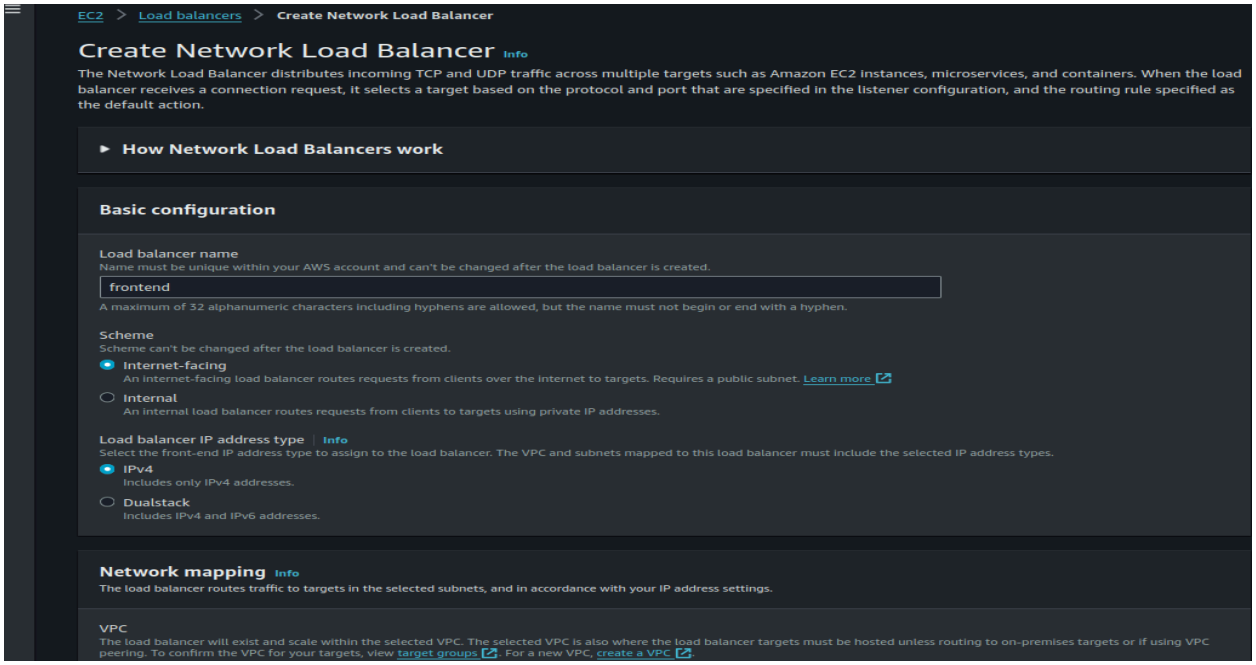
Now create a network load balancer for the frontend also.

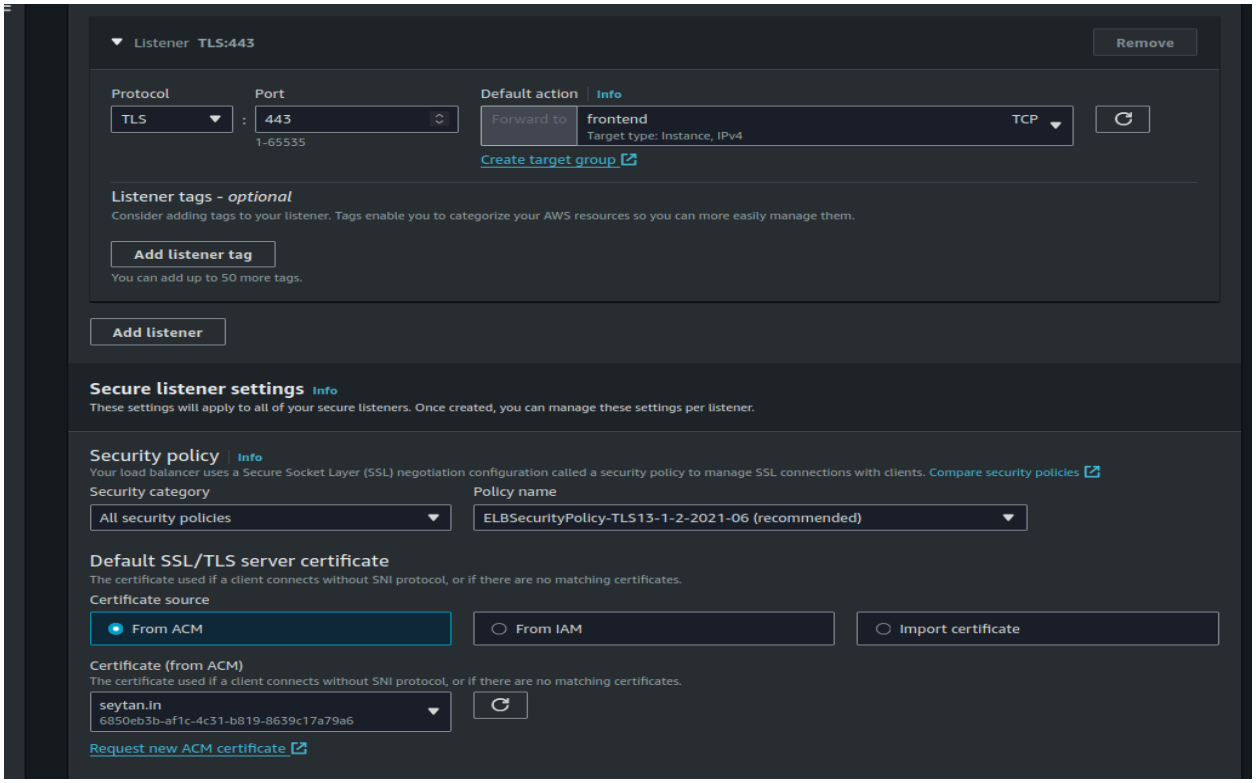
Create a target group for the frontend add tcp and port 80.



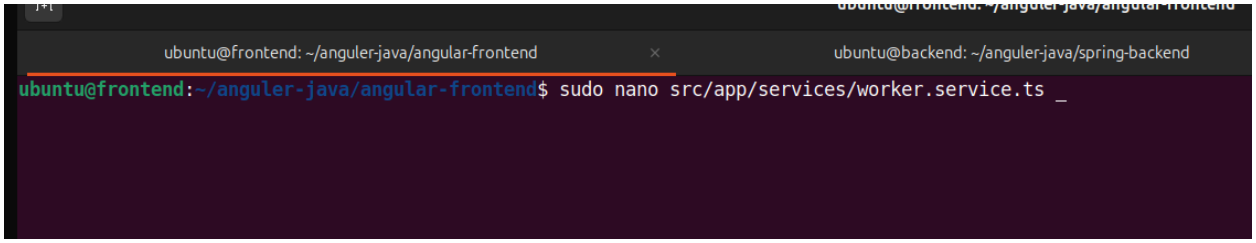


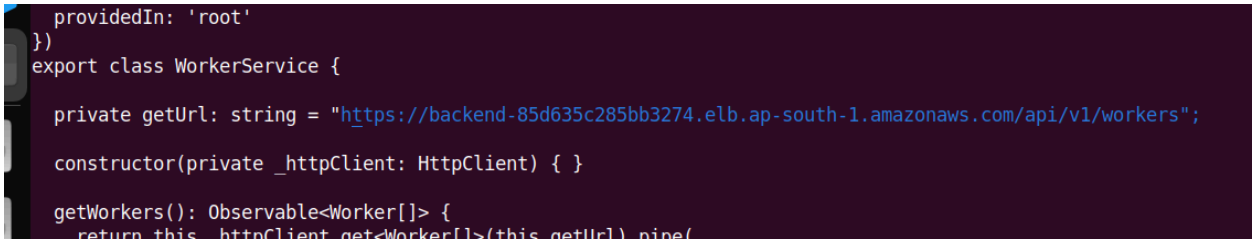
Now create load balancer for the frontend and add tls 443 and attach a ssl certificate from ACM.





Step 14: Now go to the worker.service.ts file and edit it and add the backend nlb dns in the place of localhost.

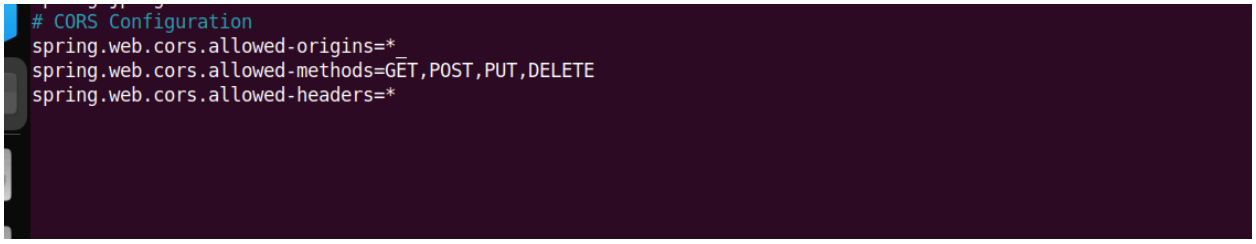




Step 15: Now as we are using 2 different domain this will cause the cors issue .

What is CORS ? →CORS (Cross-Origin Resource Sharing) is a security feature implemented in web browsers that allows or restricts web pages from making requests to a domain different from the one that served the web page. It is used to enable secure cross-origin requests and data sharing between different domains

To solve this issue we have add the following in the backend application.properties file.



Once it is done make sure add the dns names in the @CrossOrigin = dns-names in the controller file.

Note : Each time when we made any changes in the backend files make sure to run the following command .

mvn clean package -Dmaven.test.skip=true

Step 16: Run the following command to build the frontend .

ng build –configuration production

```
ubuntu@frontend:~/anguler-java/angular-frontend$ ng build --configuration production
✔ Browser application bundle generation complete.
✔ Copying assets complete.
✔ Index html generation complete.

Initial Chunk Files           | Names         |  Raw Size | Estimated Transfer Size
main.03002de15067b11b.js      | main          | 237.78 kB |                61.81 kB
polyfills.b525ededa71d3b7f.js | polyfills     |  33.08 kB |                10.63 kB
runtime.e411e20b75d2e1de.js   | runtime       |   1.06 kB |               607 bytes
styles.ef46db3751d8e999.css   | styles        |   0 bytes |                       -

                              | Initial Total | 271.93 kB |                73.03 kB

Build at: 2024-08-31T17:23:53.631Z - Hash: 0eae4a10deab2325 - Time: 9797ms
ubuntu@frontend:~/anguler-java/angular-frontend$ _
```

This command will create dist directory inside of it there will be a angular-frontend directory which will contain the static files of our frontend.

ng build --configuration production → is an Angular CLI command that compiles the application in production mode, optimizing the build by enabling features like Ahead-of-Time (AOT) compilation, minification, and tree-shaking for better performance and smaller bundle sizes.

Step 17 : Install the nginx server on frontend ,

```
ubuntu@frontend:~/anguler-java/angular-frontend$ sudo apt install nginx -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  nginx-common
Suggested packages:
  fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
```

and remove the default files in the home directory of nginx and copy the static files in the dist/angular-frontend to the nginx home directory.

```
Build at: 2024-08-31T17:23:53.631Z - Hash: 0eae4a10deab2325 - Time: 9797ms
ubuntu@frontend:~/anguler-java/angular-frontend$ sudo rm /var/www/html/*
ubuntu@frontend:~/anguler-java/angular-frontend$ sudo cp dist/angular-frontend/* /var/www/html/
ubuntu@frontend:~/anguler-java/angular-frontend$ _
```

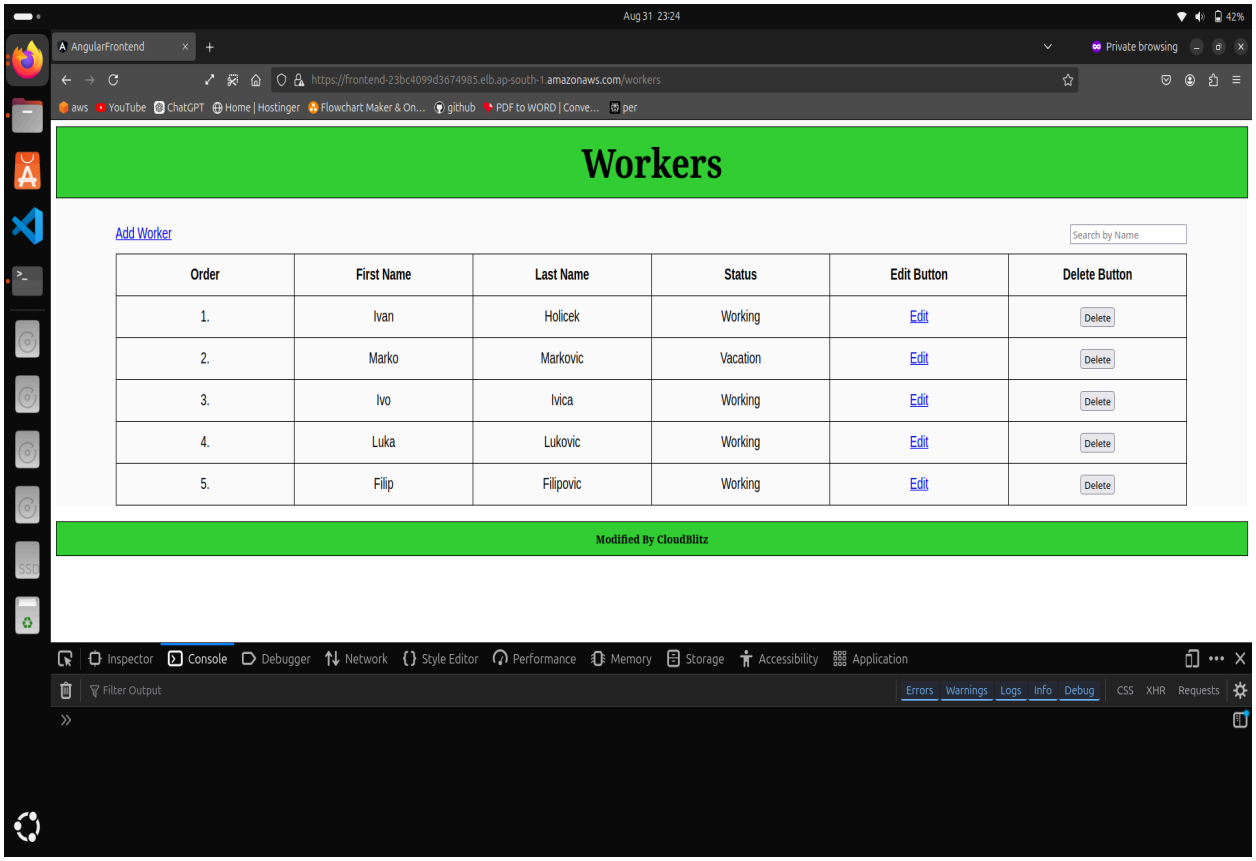
Step 18 : Now run the next command in backend server

java -jar target/springbackend-v1.jar&

```
ubuntu@backend:~/anguler-java/spring-backend$ java -jar target/spring-backend-v1.jar&
[1] 4821
ubuntu@backend:~/anguler-java/spring-backend$
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.4)

2024-08-31 17:24:37.677  INFO 4821 --- [           main] c.e.s.SpringBackendApplication           : Starting SpringBackendApplication vv1 using Java 1.8.0_
 (/home/ubuntu/anguler-java/spring-backend/target/spring-backend-v1.jar started by ubuntu in /home/ubuntu/anguler-java/spring-backend)
2024-08-31 17:24:37.685  INFO 4821 --- [           main] c.e.s.SpringBackendApplication           : No active profile set, falling back to 1 default profi
2024-08-31 17:24:39.583  INFO 4821 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT m
2024-08-31 17:24:39.688  INFO 4821 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 86 ms. Fou
s.
2024-08-31 17:24:41.315  INFO 4821 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2024-08-31 17:24:41.356  INFO 4821 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-08-31 17:24:41.357  INFO 4821 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.65]
2024-08-31 17:24:41.583  INFO 4821 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-08-31 17:24:41.585  INFO 4821 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
2024-08-31 17:24:42.687  INFO 4821 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-31 17:24:43.200  INFO 4821 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-31 17:24:43.364  INFO 4821 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: defaul
2024-08-31 17:24:43.544  INFO 4821 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.11.Final
2024-08-31 17:24:43.962  INFO 4821 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final
2024-08-31 17:24:44.275  INFO 4821 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.MySQL5
2024-08-31 17:24:45.415  INFO 4821 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibe
platform.internal.NoJtaPlatform]
2024-08-31 17:24:45.435  INFO 4821 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence un
2024-08-31 17:24:46.369  WARN 4821 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
erformed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-08-31 17:24:47.448  INFO 4821 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context pat
2024-08-31 17:24:47.480  INFO 4821 --- [           main] c.e.s.SpringBackendApplication           : Started SpringBackendApplication in 10.972 seconds (JVM
2024-08-31 17:24:47.943  INFO 4821 --- [nio-8080-exec-4] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServle
2024-08-31 17:24:47.945  INFO 4821 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2024-08-31 17:24:47.948  INFO 4821 --- [nio-8080-exec-4] o.s.web.servlet.DispatcherServlet        : Completed initialization in 3 ms
```

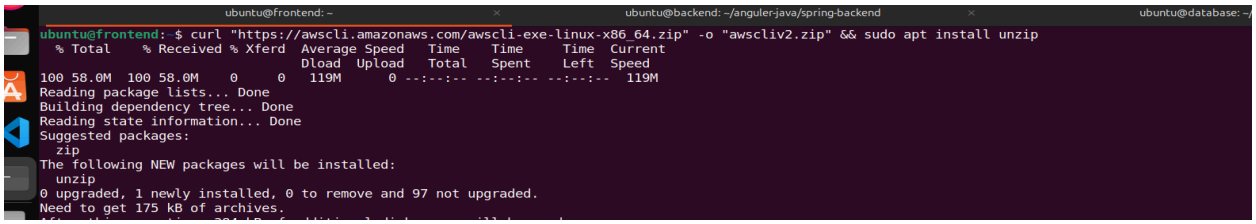
After this enter the frontend NLB dns in the browser and see the output

Output :

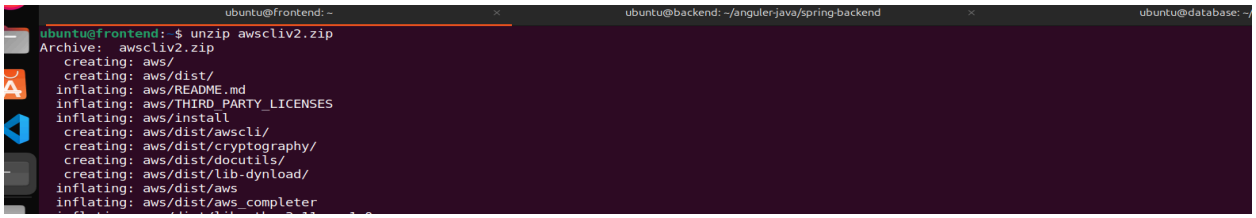


Step 19 : Now install the aws cli on frontend and attach s3 full permission role to the frontend server.

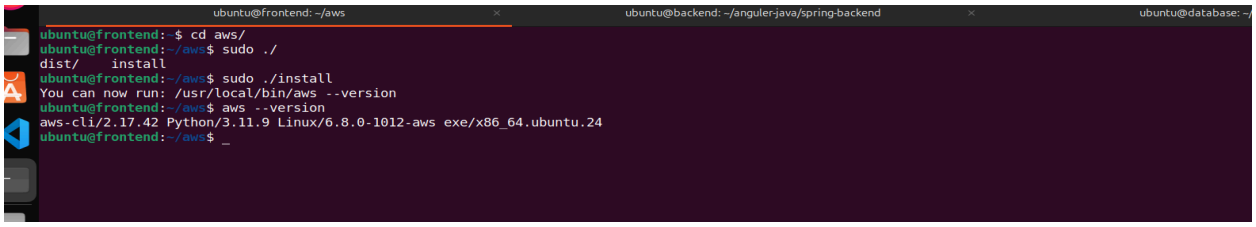
Command :
sudo apt install unzip && curl “https://awscli/amazonaws.com/awscli-exe-linux-x86_64.zip” -o “awscliv2.zip



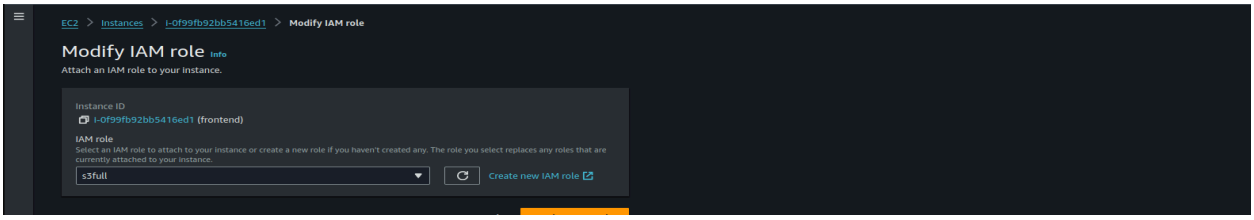
Unzip the file



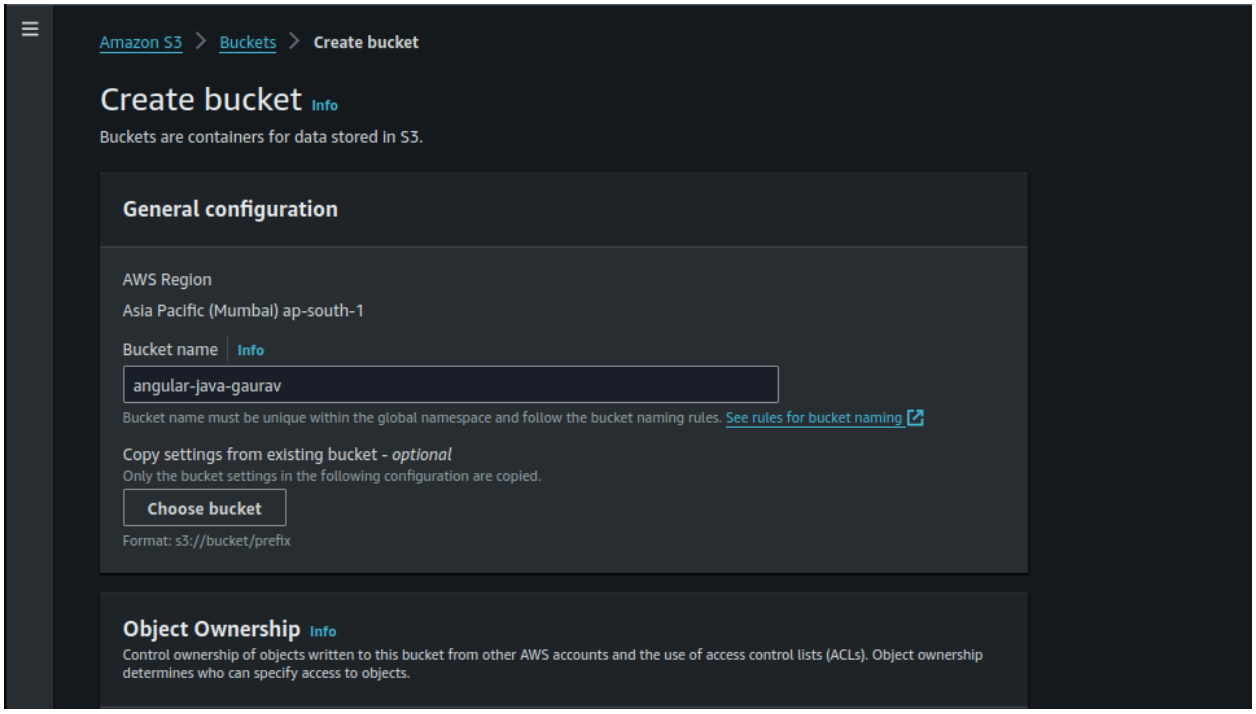
Execute the install file and verify aws installation



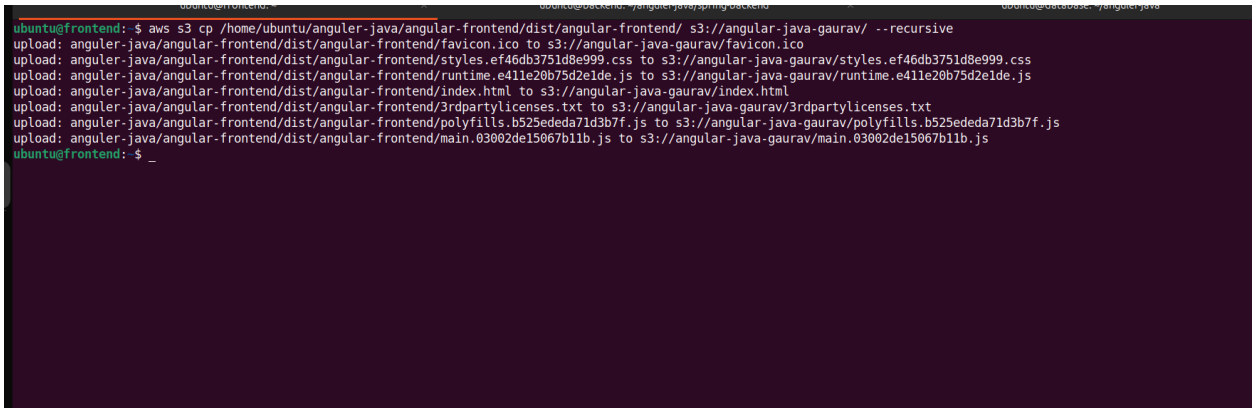
Attach the S3 IAM role to the frontend server.



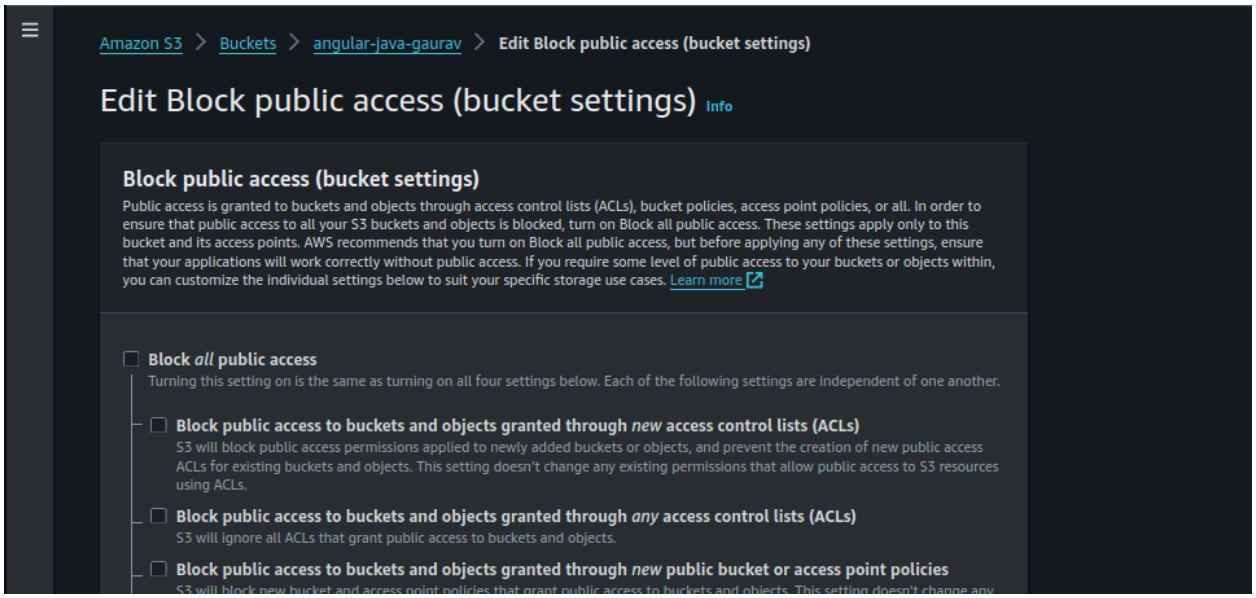
Step 20 : Create a s3 bucket .

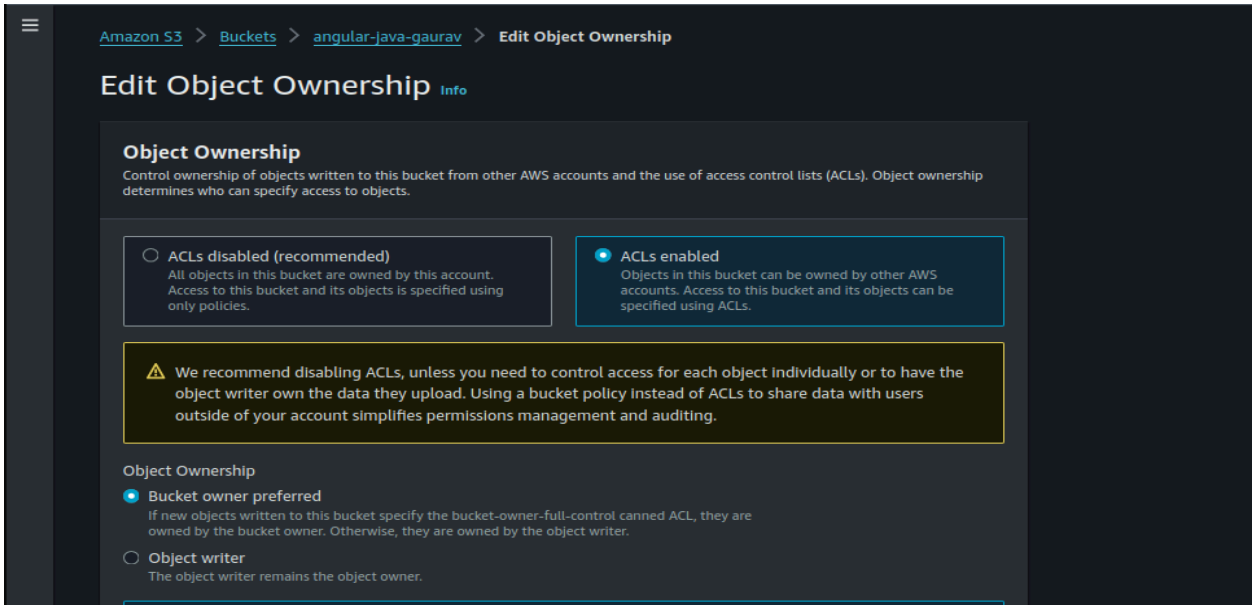


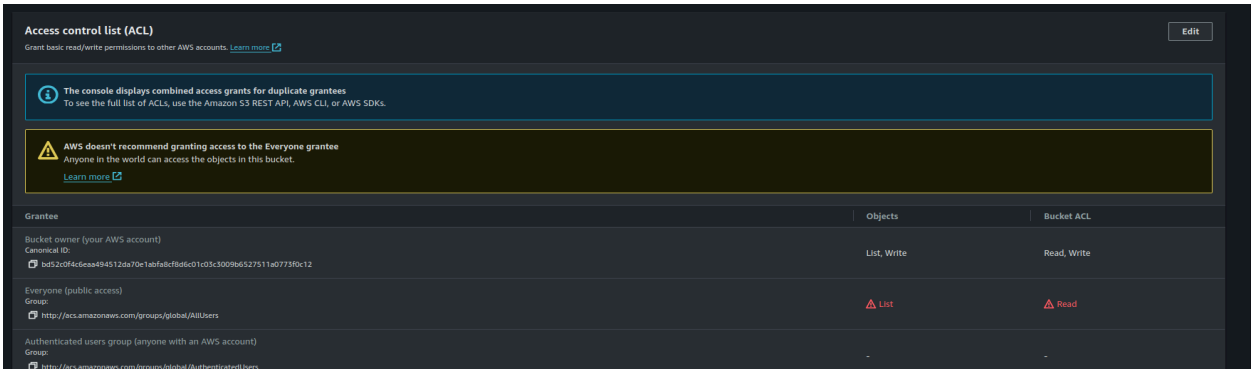
Once bucket is created upload the necessary object files in it.



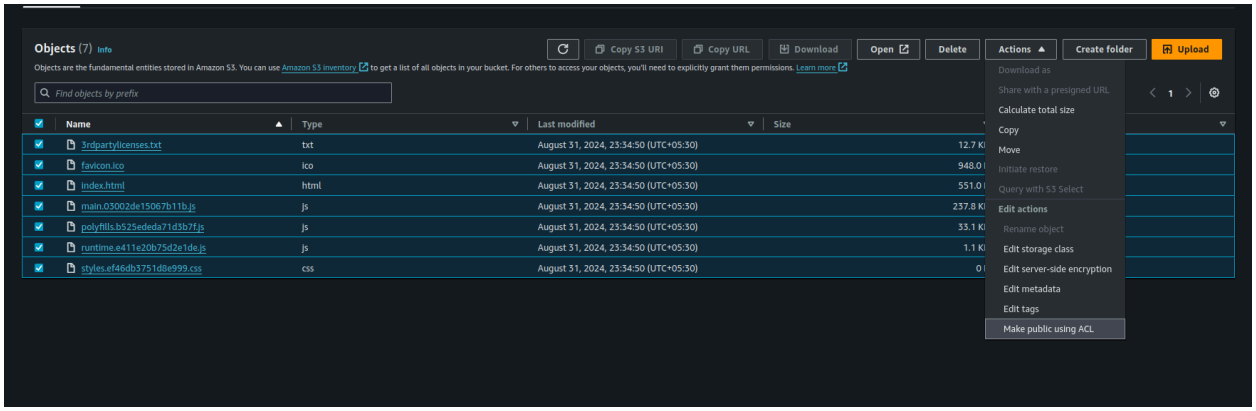
After adding the files go to the permission tab of bucket and make necessary changes for allowing anyone(publically access the objects in the bucket).





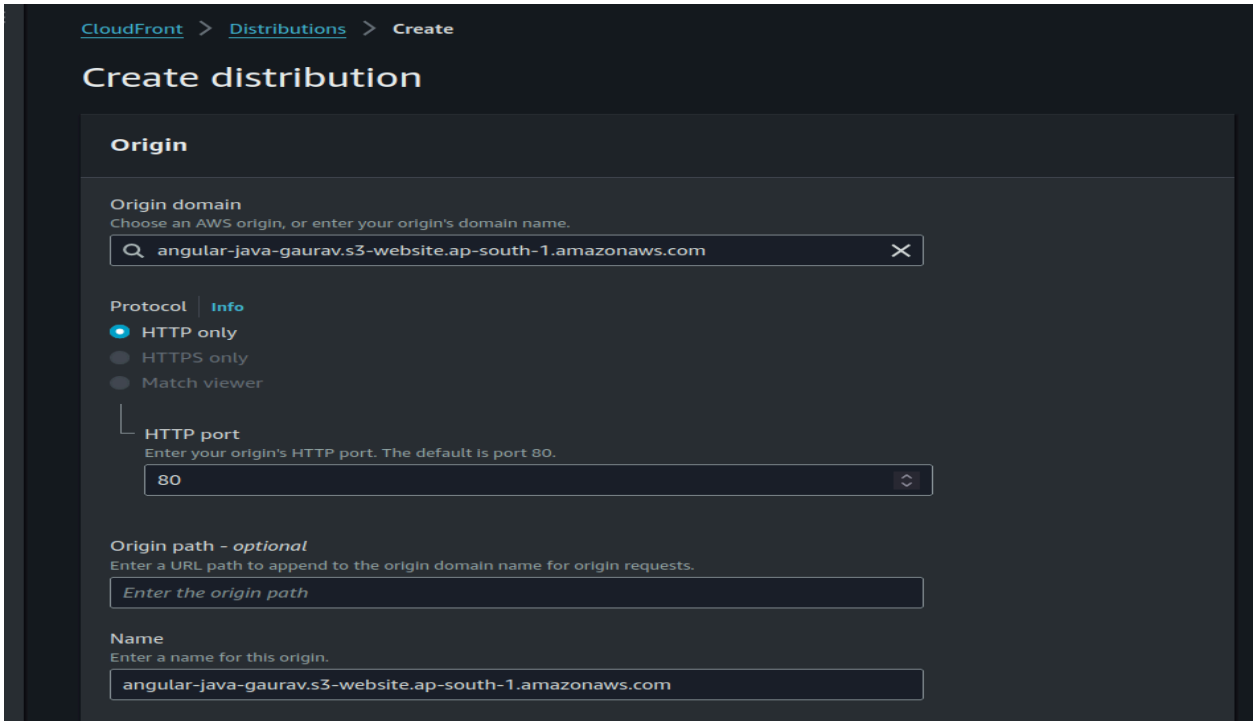


After this make the objects public using acl from actions tab .

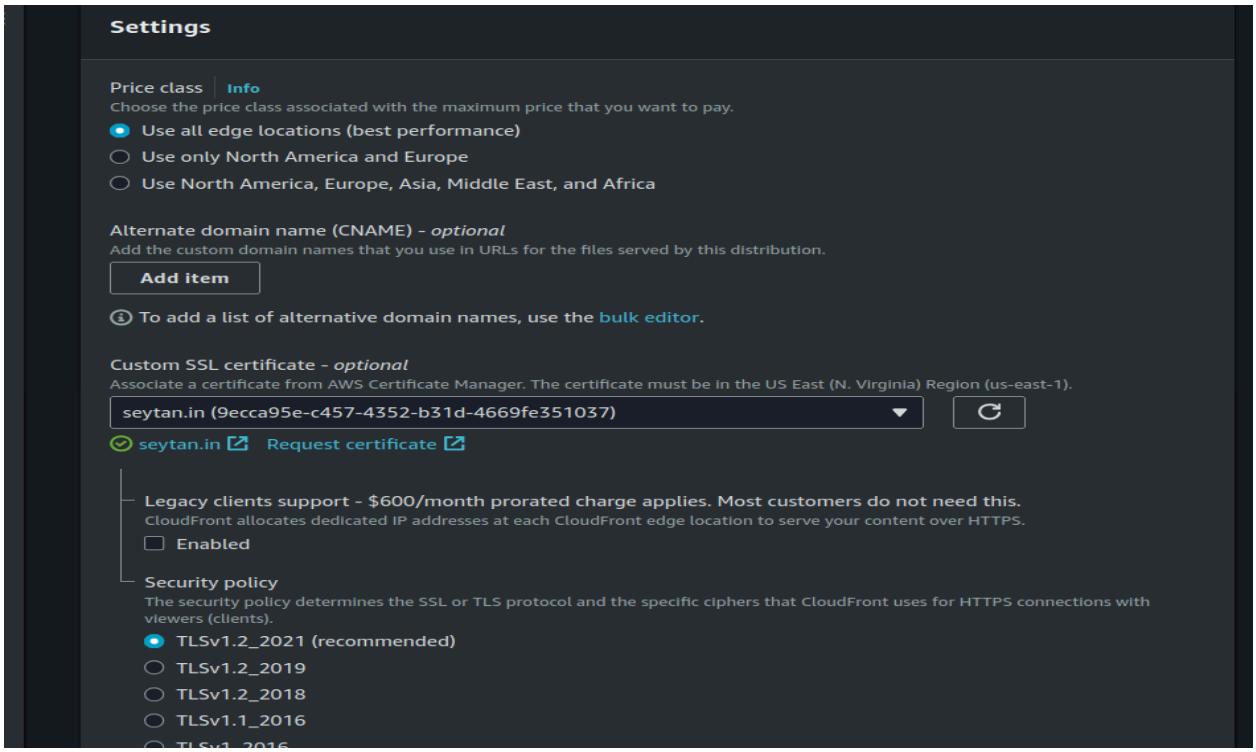


Now go to the properties tab of bucket and in the last enable the static web hosting and gave the index.html as serving page.

Step 21 : Now create a Distribution in the cloudfront with origin domain as the s3 bucket .

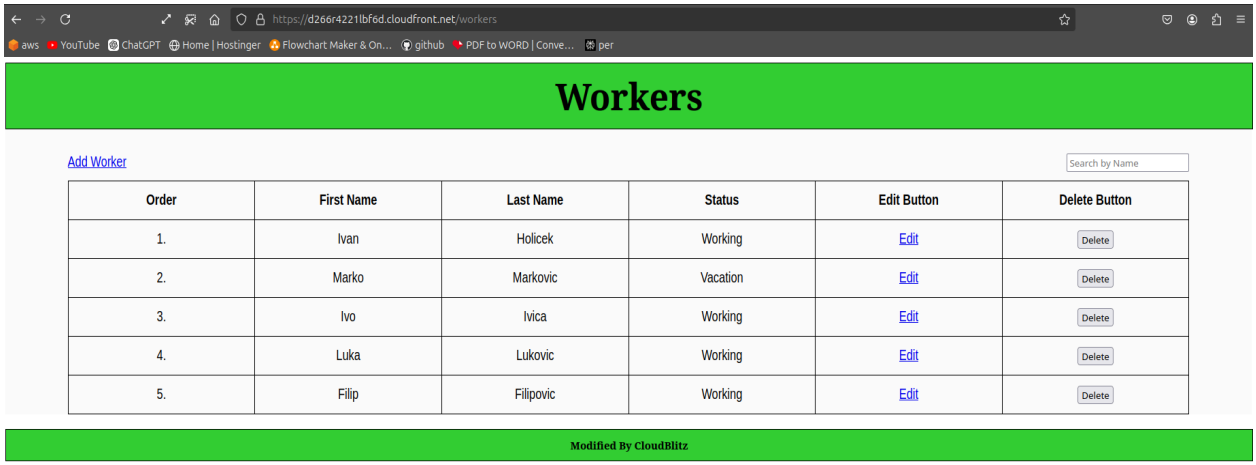


Also add ssl certificate to make our CDN DNS secured.

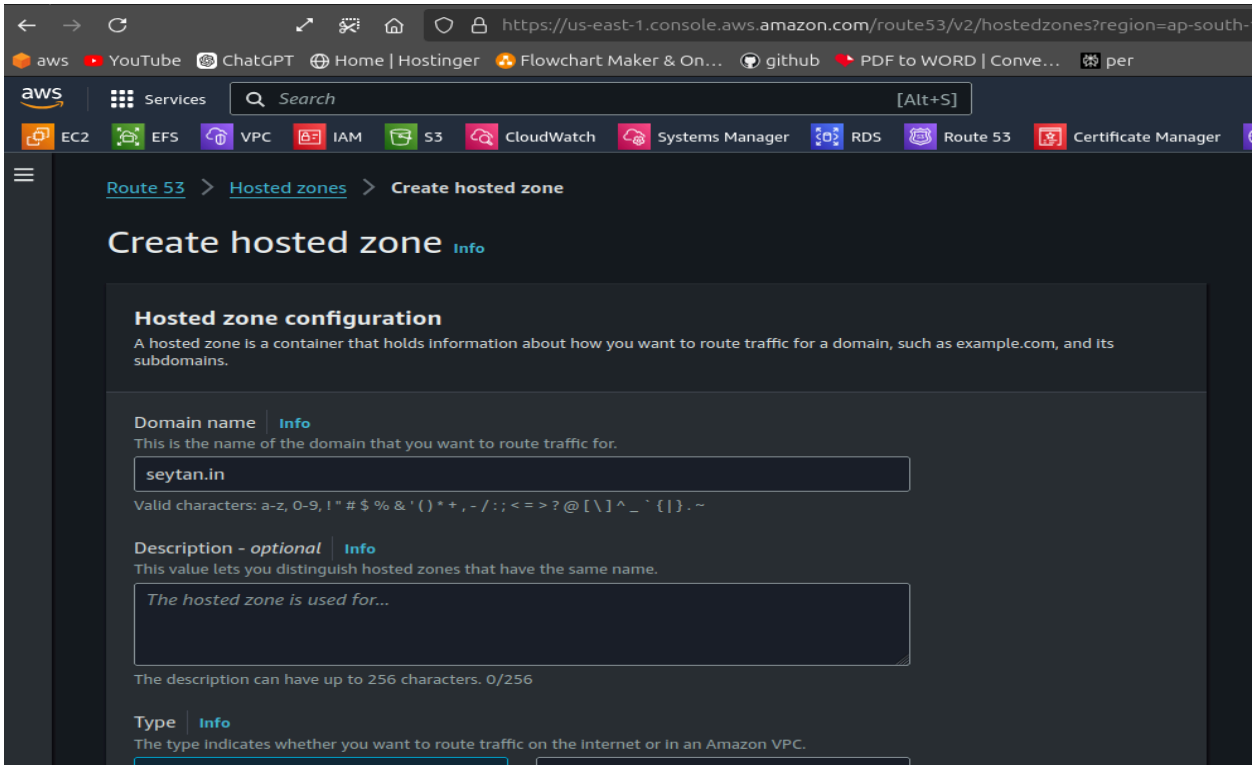


Now once it is deployed paste the CDN DNS in the browser and see the Output

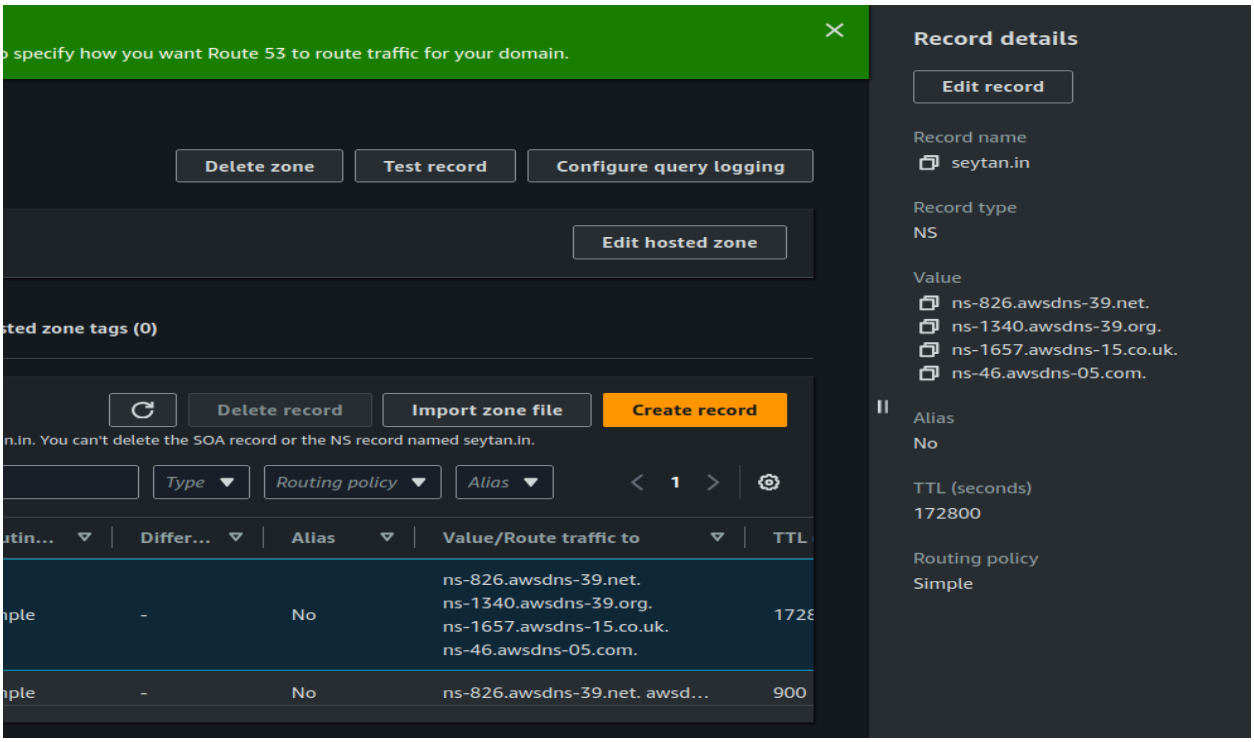
Output :

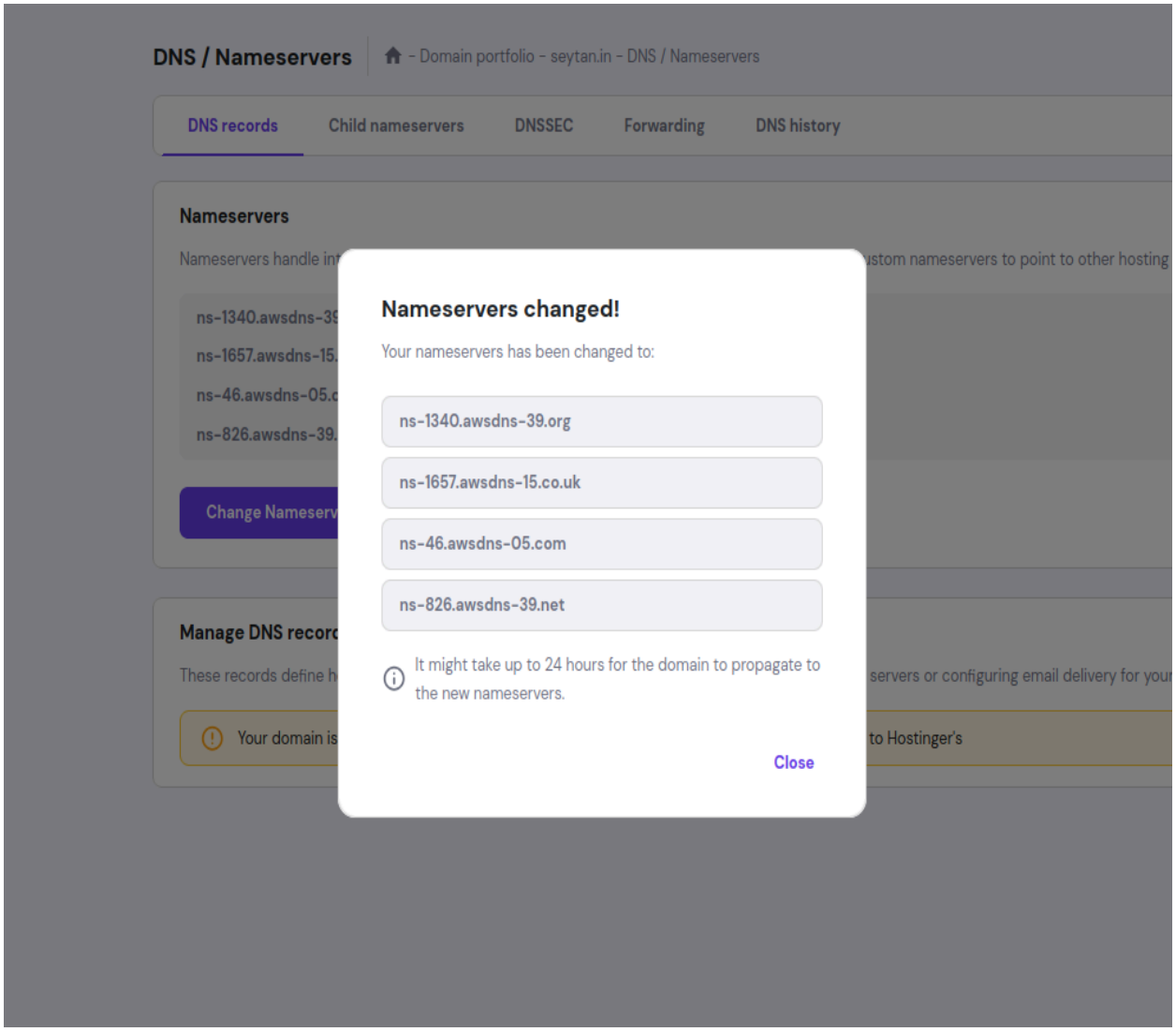


Step 22: Now Create a Hosted Zone in the route53 with your domain.



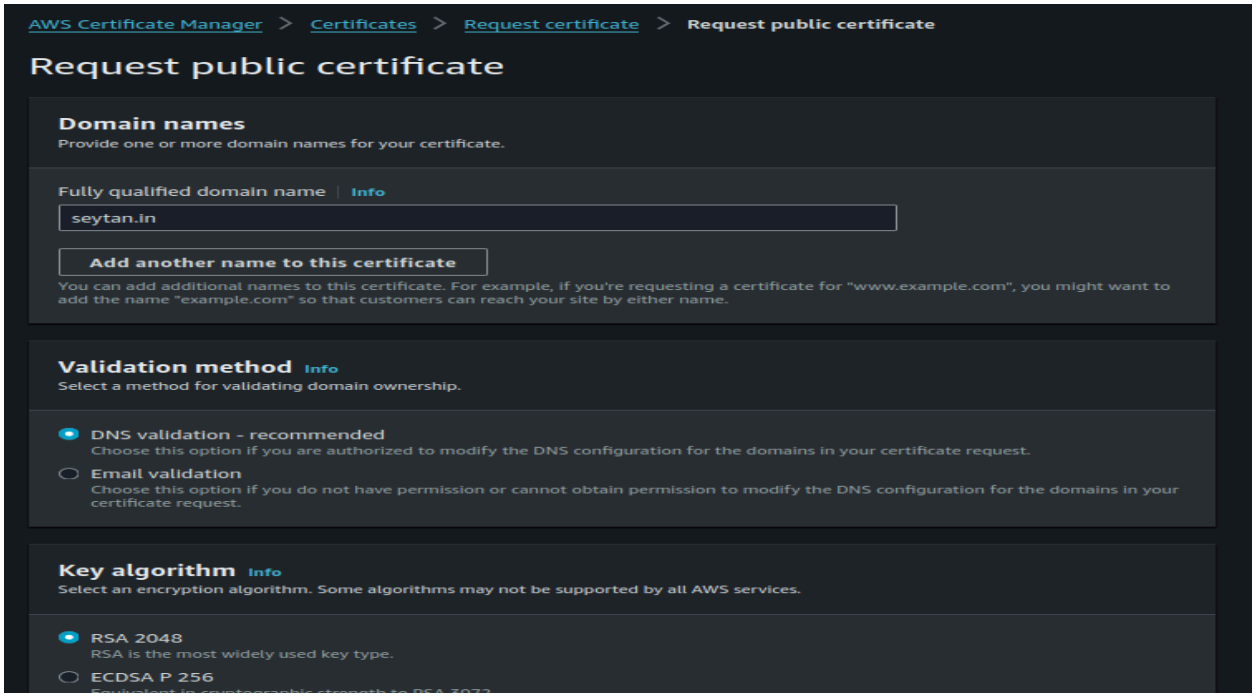
Now copy the ns-record and paste it in the 3rd party domain provider ns-records.

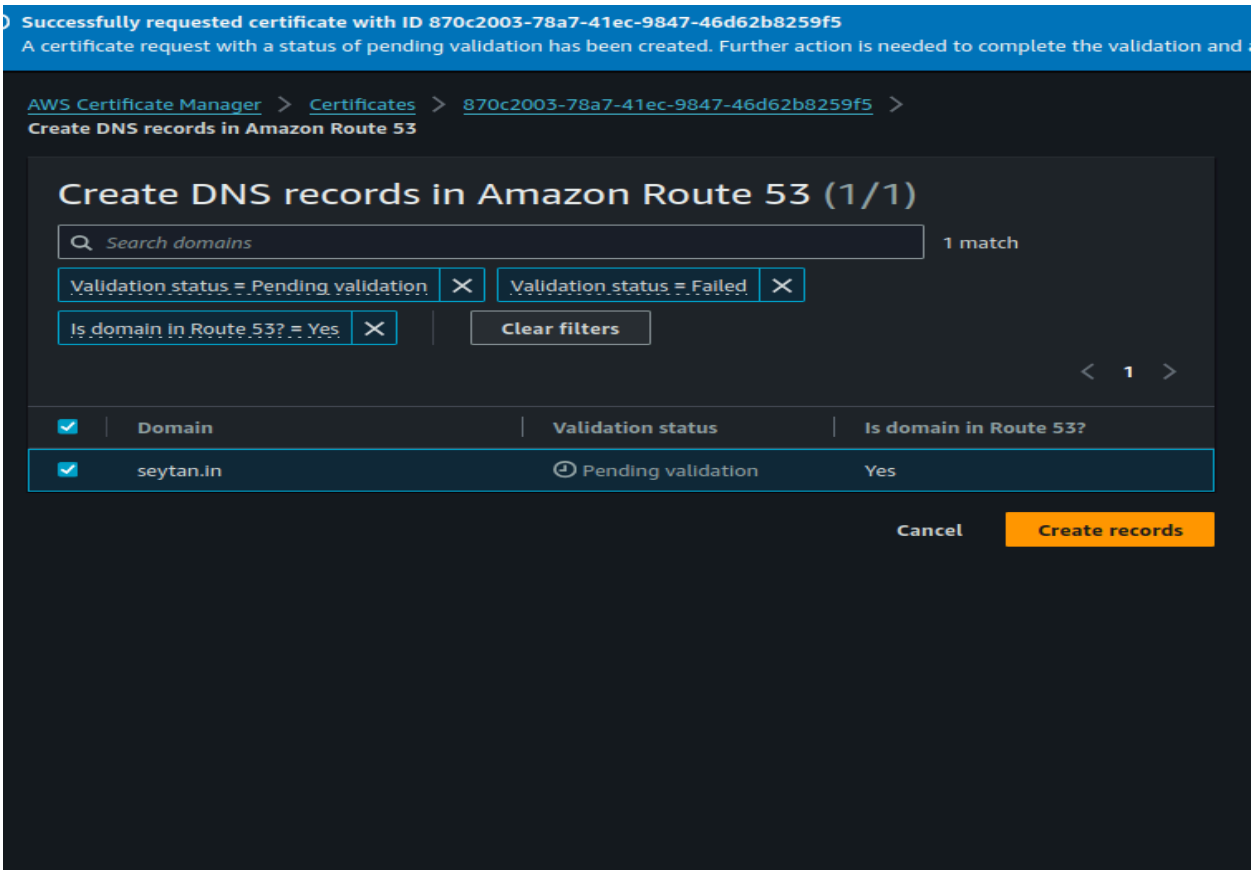




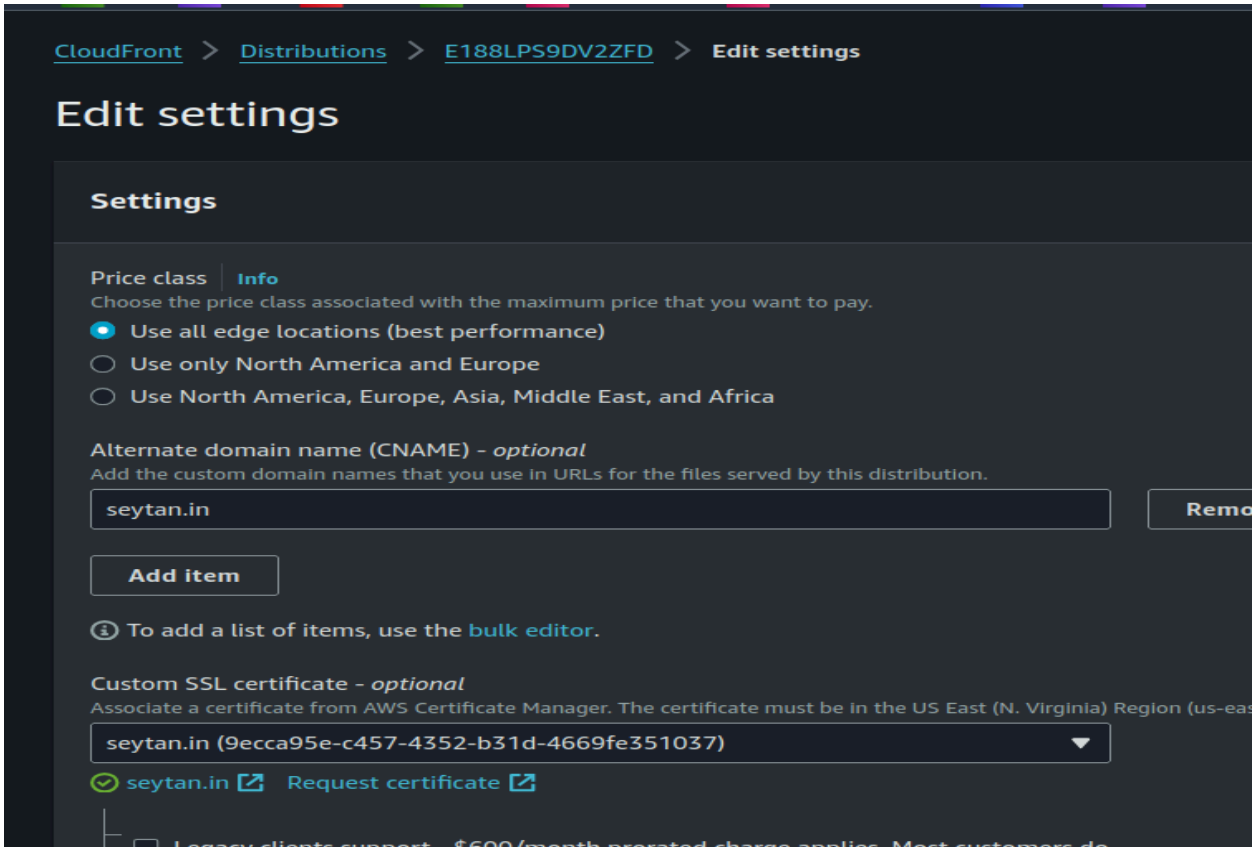
Ns-records changed in 3rd party domain provider .

Now Request a ssl certificate from ACM and add it to the domain.

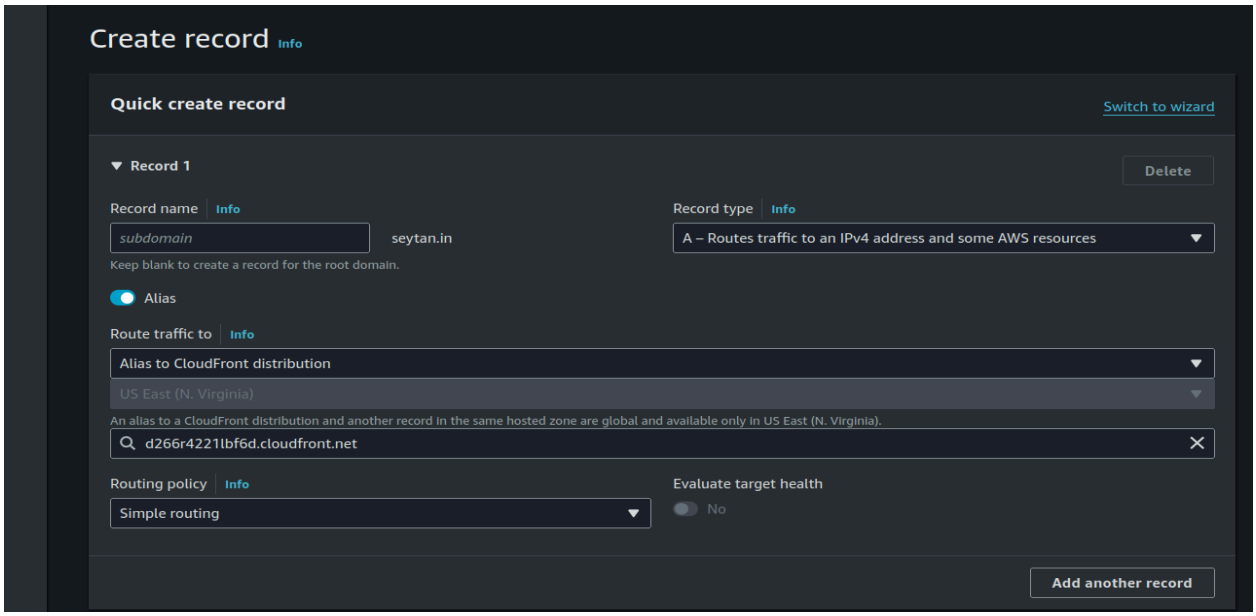




Step 23: Now add a CNAME to the CDN Distribution which we created by editing it .

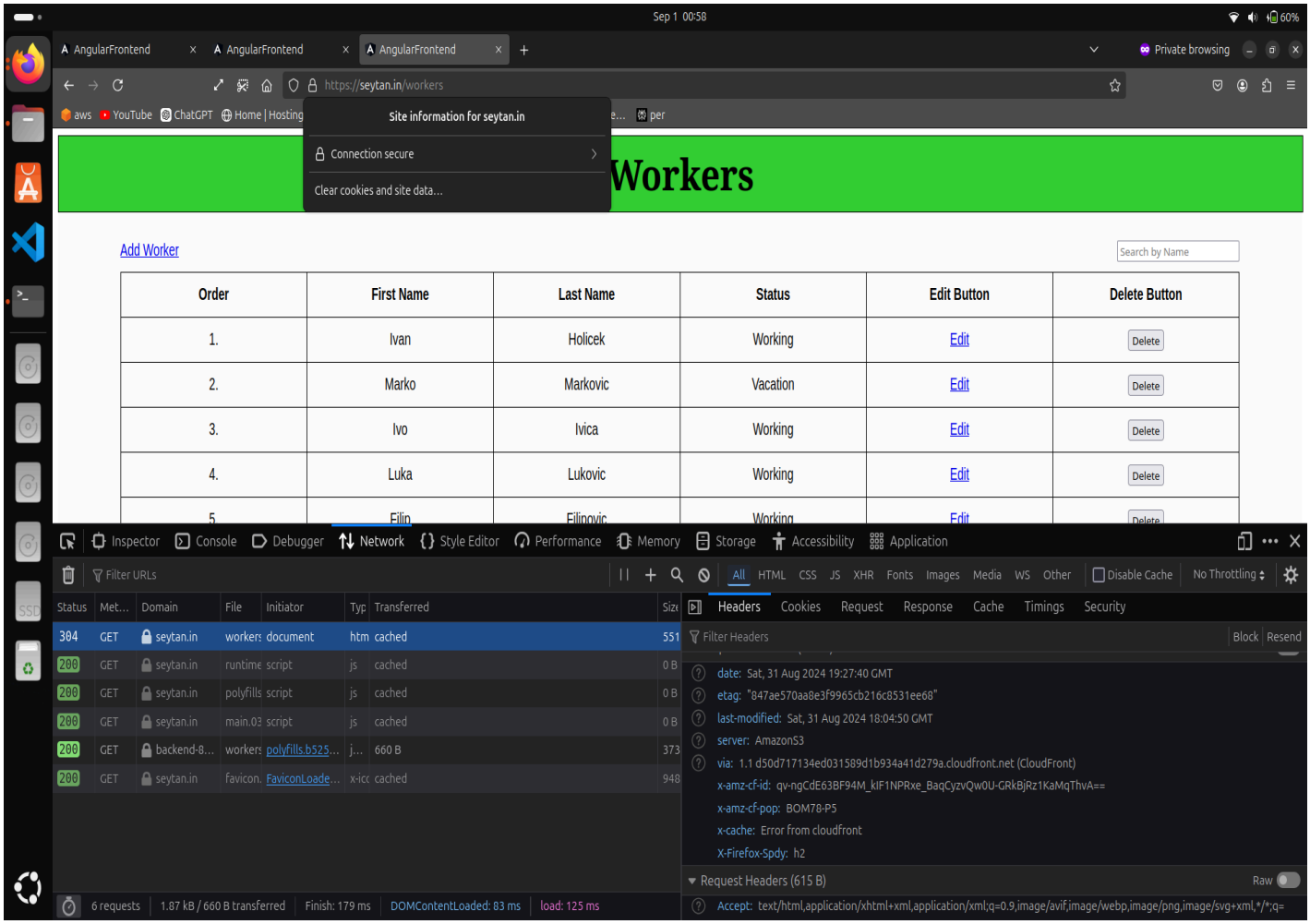


Once it is done wait for deployed state after that add an A-record with alias as the CDN DNS



Step 24: Now enter your domain seytan.in
And see it is secured and the content is coming via CDN DNS.

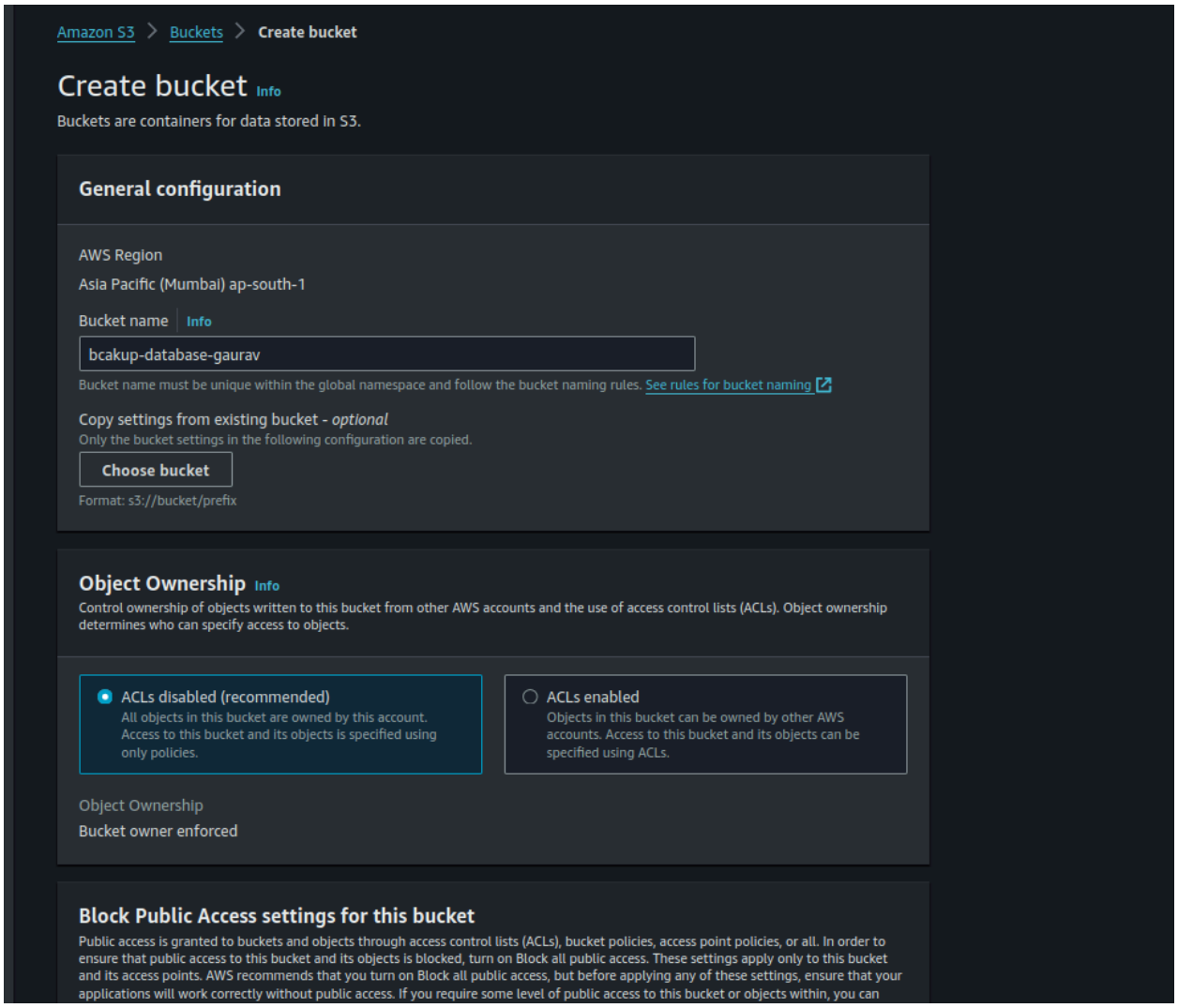
Output :



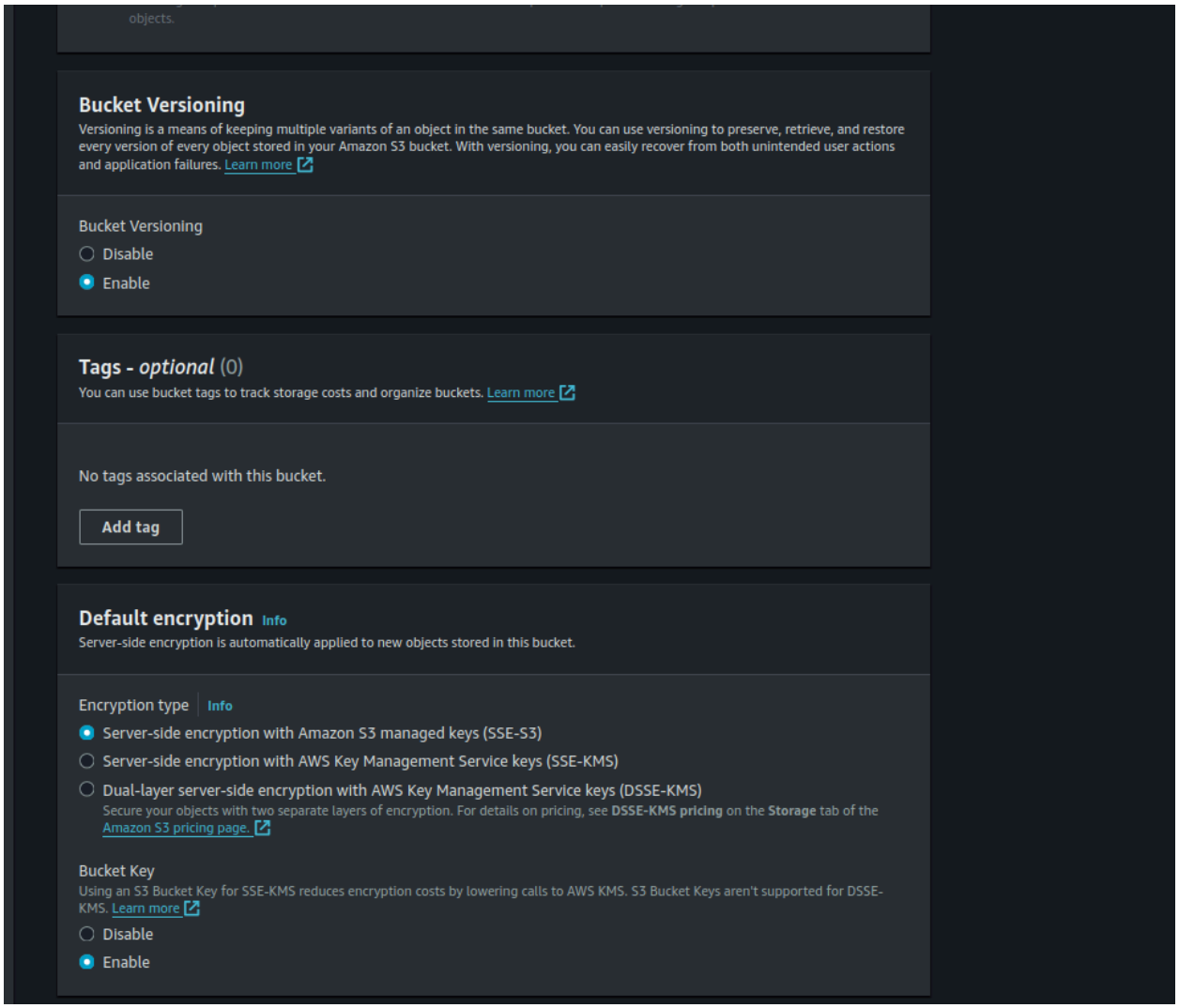
Step 25: Database backup to s3 bucket automatically .
Now go to the database server and install aws cli and attach the s3 Full IAM role to the database server .

[Refer this above step with step Number 19]

Step 26 : Create a s3 bucket for database backup.



Also enable the bucket versioning in this case .

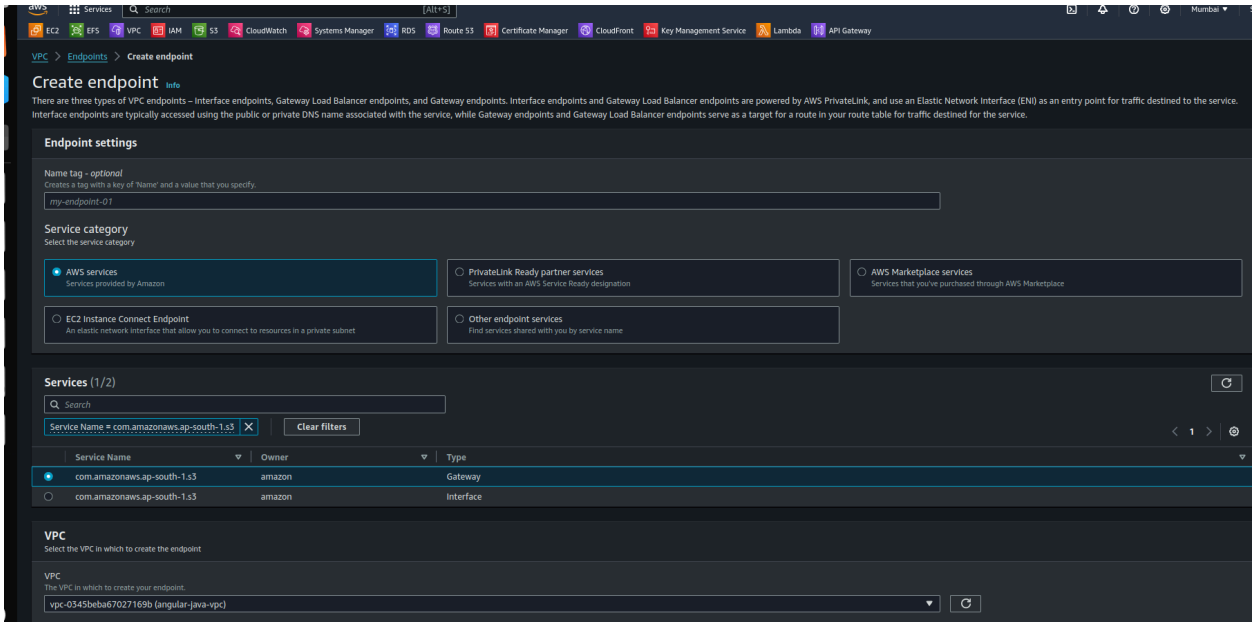


Step 27:

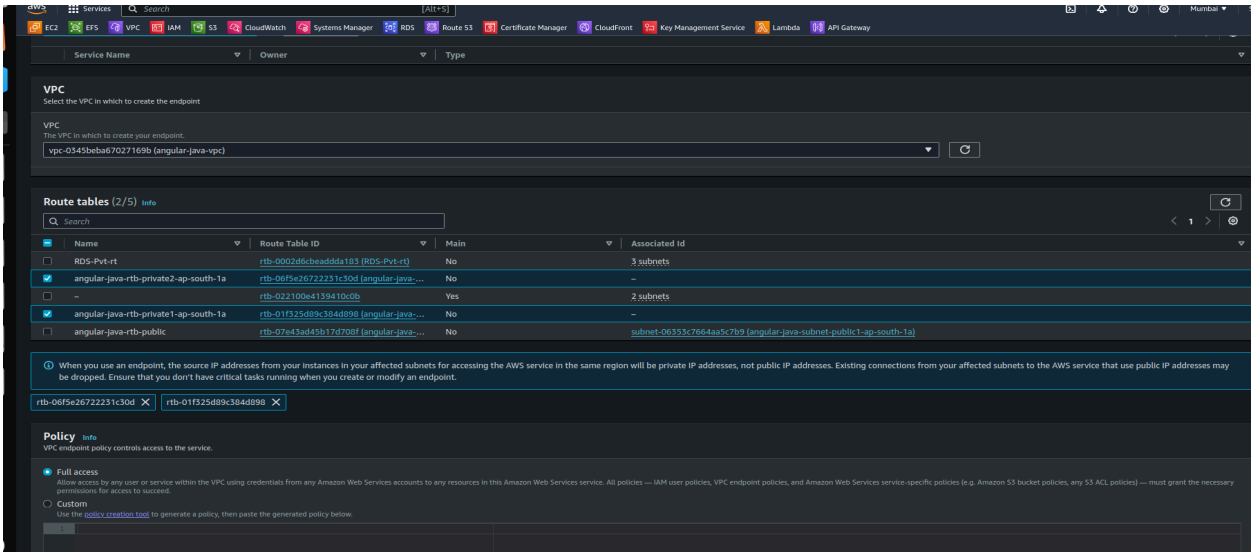
Once it is done now we have to close the nat gateway from this private subnets backend and database .
Go to the route tables of this 2 subnet and remove them from the subnet association explicitly section .

Step 28 : As Now we have removed the Nat gateway we have to add the s3 vpc endpoint in this vpc pointing to the private subnets so we can connect to the s3 without nat gateway ,internet.

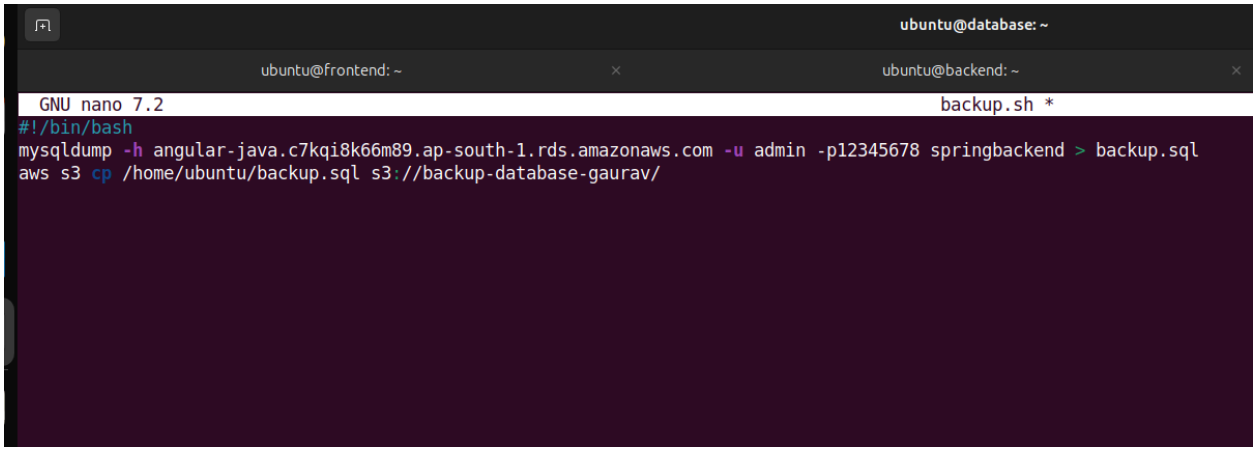
What is s3 vpc endpoint ?
→An S3 VPC endpoint allows you to connect your VPC directly to Amazon S3 without using an internet gateway, NAT device, VPN connection, or AWS Direct Connect. This enhances security by ensuring that S3 traffic stays within the AWS network.



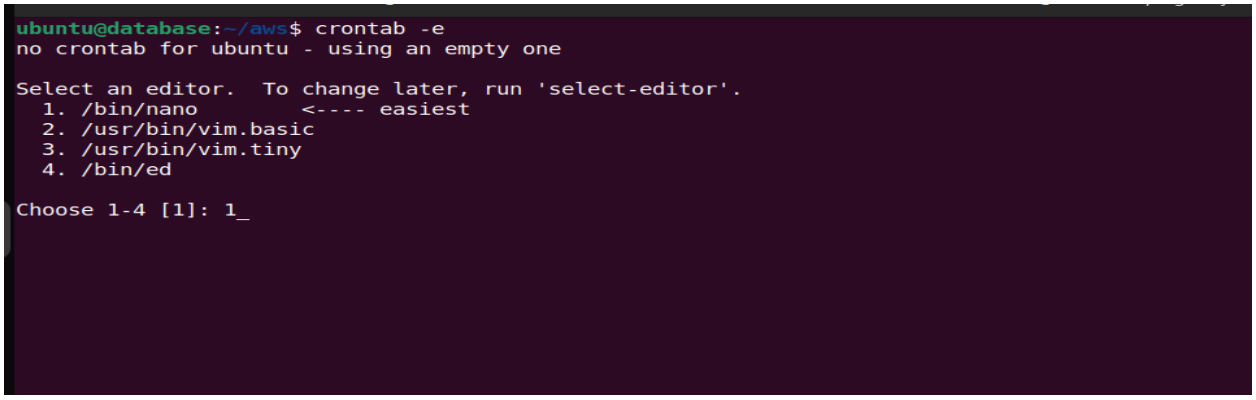
Add the private subnets in this .

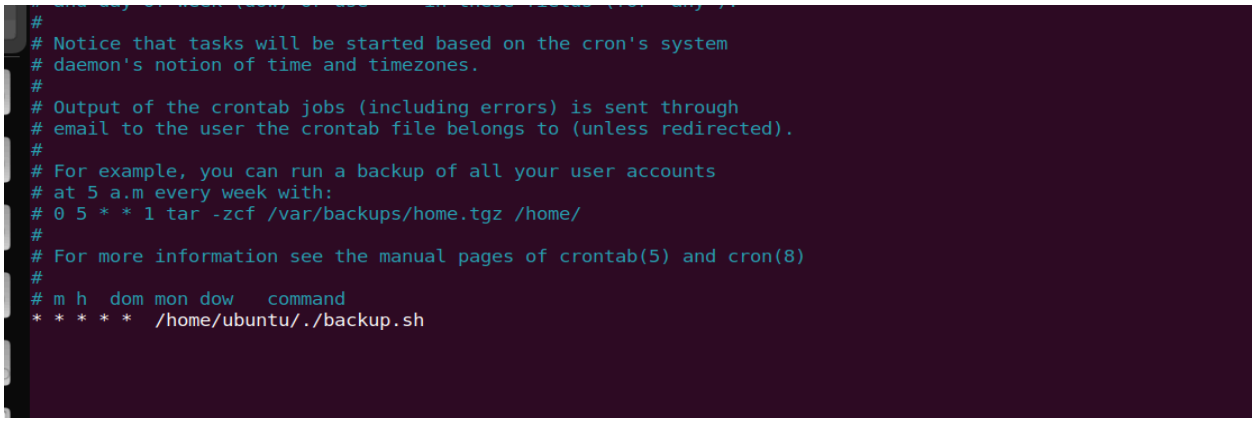


Step 29 : Now create a backup.sh file and add the followings lines to it .
It will create a backup of database and copy it to the backup-database-gaurav bucket.



As we want to make this process automatically we will use crontab for this.

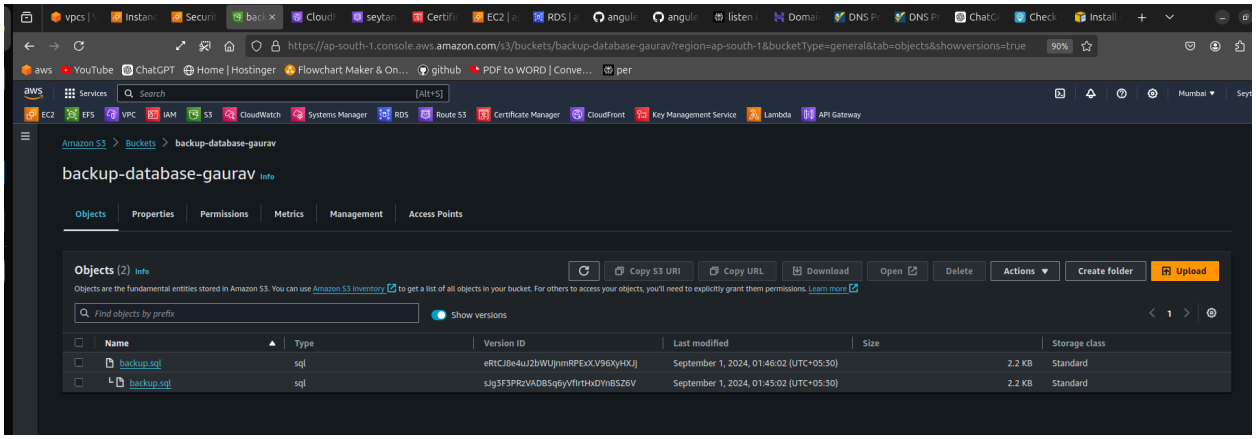




It will execute the task every 1 minute .

Output :

As we can see we had enabled the versioning so we can easily identify the first backup and latest backup of our database.

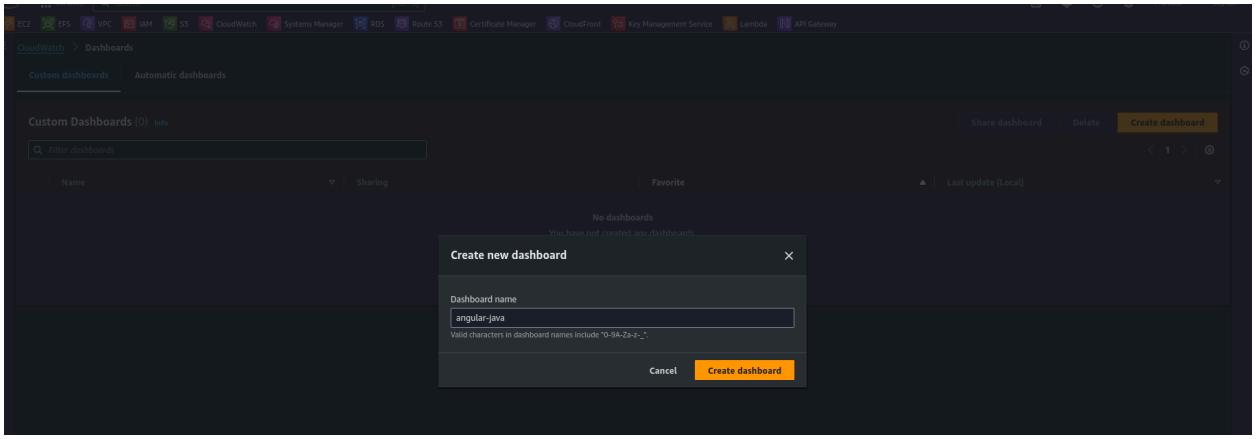


Step 30:

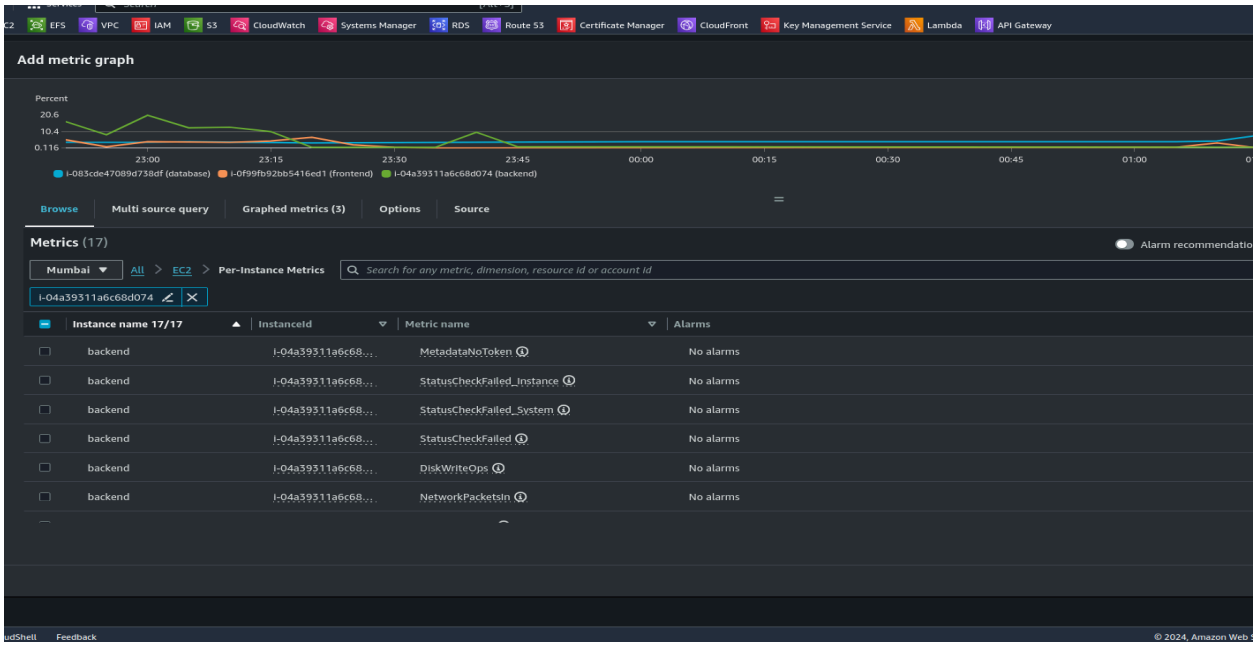
Now Create a Dashboard for monitoring our 3-tier application in Cloudwatch .

What is cloudwatch ?

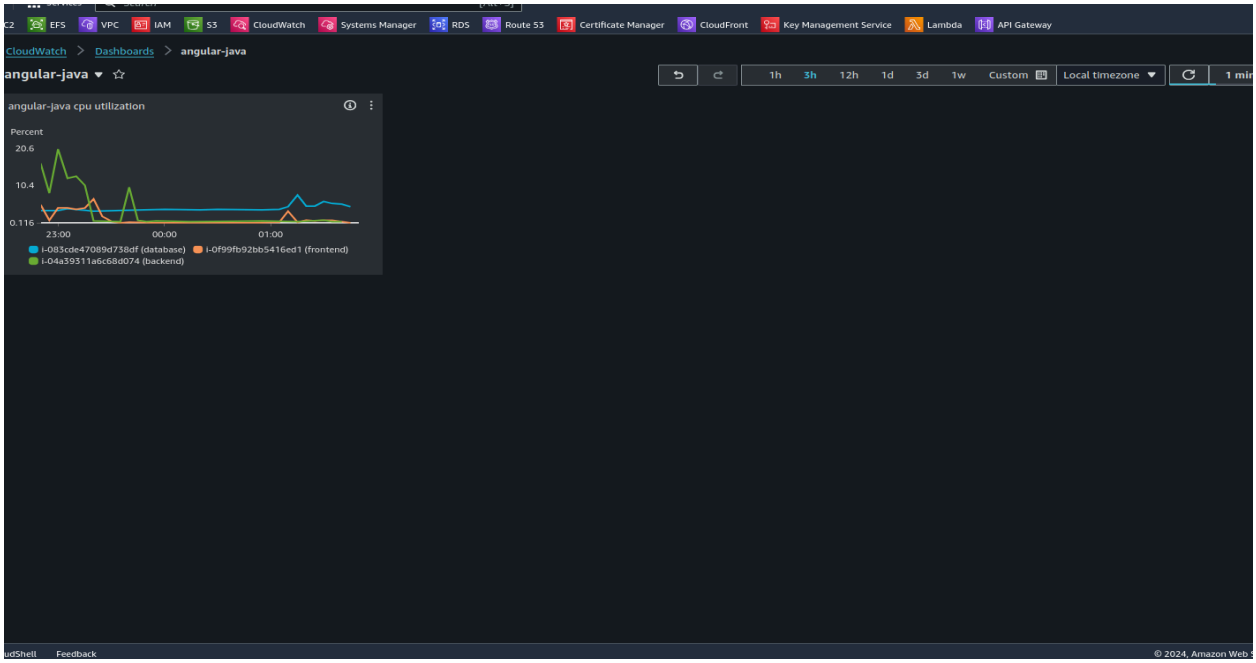
→CloudWatch is an AWS service that monitors and logs performance metrics and operational data from AWS resources and applications. It provides real-time insights, enabling you to collect, access, and analyze logs, set alarms, and automatically react to changes in your environment.

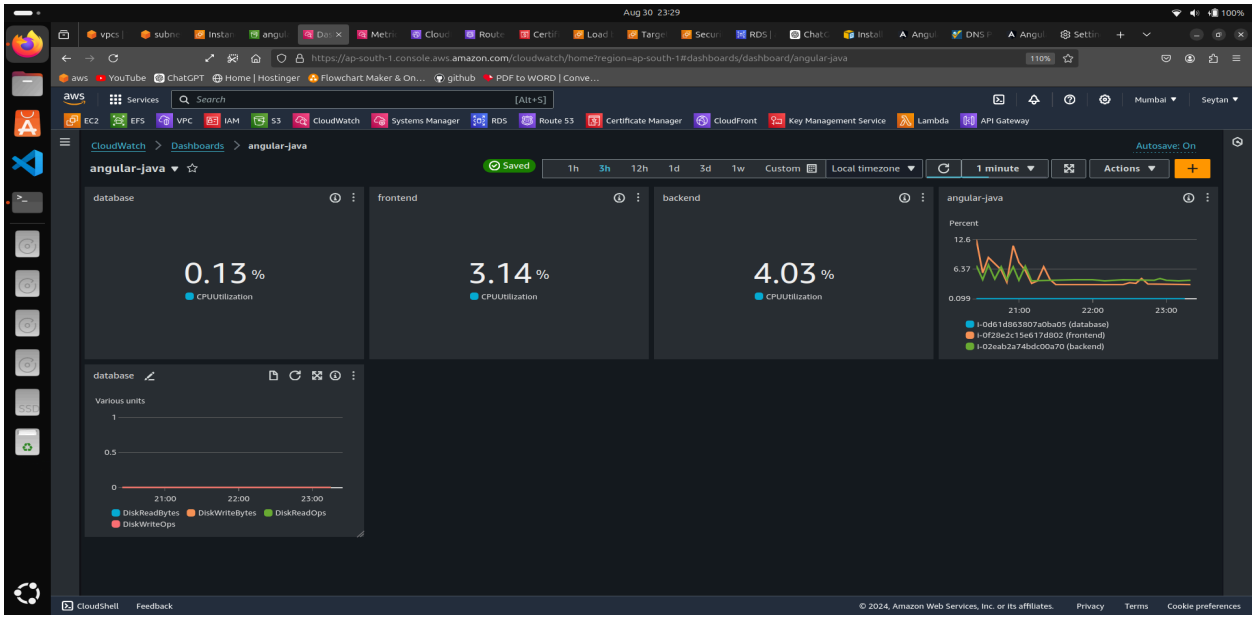


Add the necessary widgets.



Output:





Conclusion :

This deployment ensures a secure and scalable environment for your 3-tier application, leveraging AWS services effectively. By utilizing Route 53 and AWS CDN, you'll achieve optimal performance and reliability for both frontend and backend components.