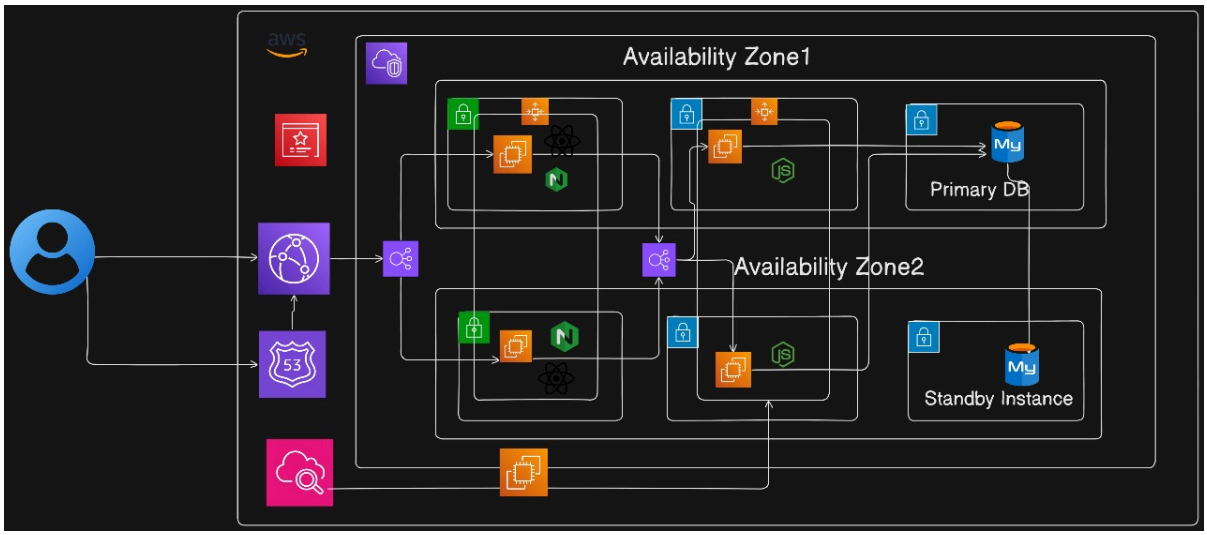
Project-1

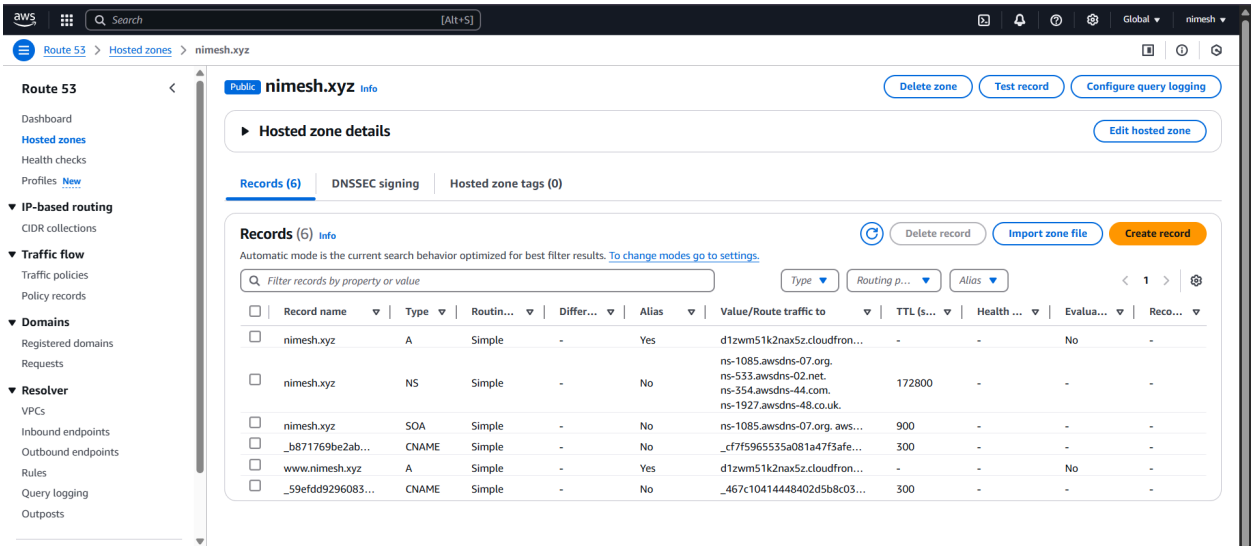
Deploying a Production Grade Highly Available & Scalable 3 Tier Architecture in AWS

Architecture of the application:



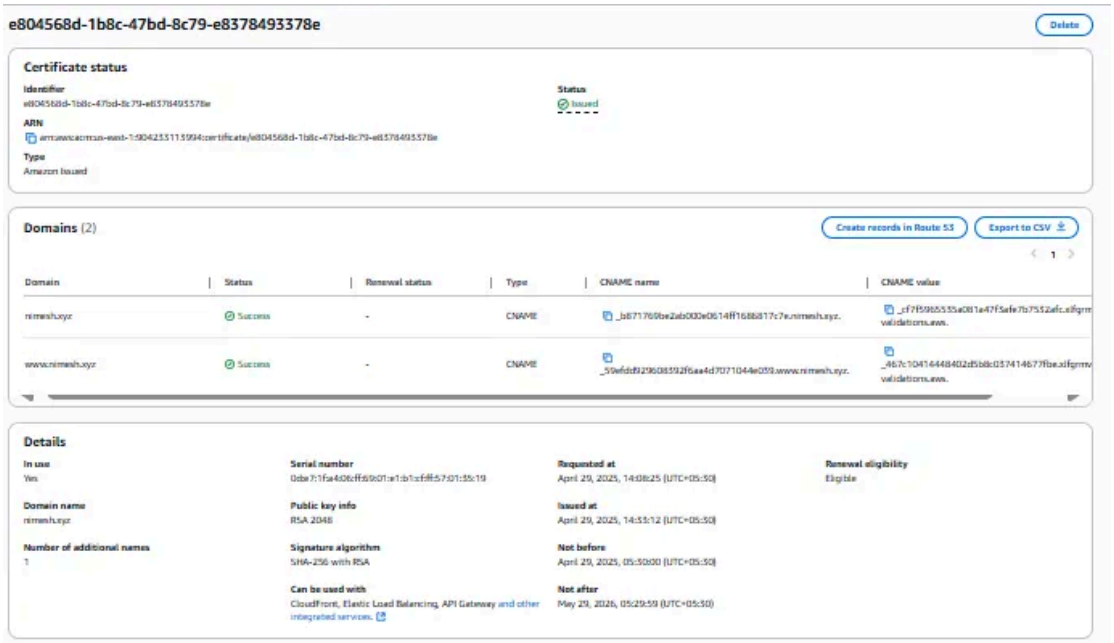
Step 1: Configuring Route53:

- Go to Route53 in AWS -
- Click on “Create Hosted Zone” .
- Give “Domain name” and Select Public Hosted Zone.
- Once Created, You will get NS records.
- Add those NS records in DNS of your Domain Provider (Our Case, GoDaddy).
- It will take more than an hour to propagate.



Step 2: Requesting a Public SSL Certificate using Amazon ACM:

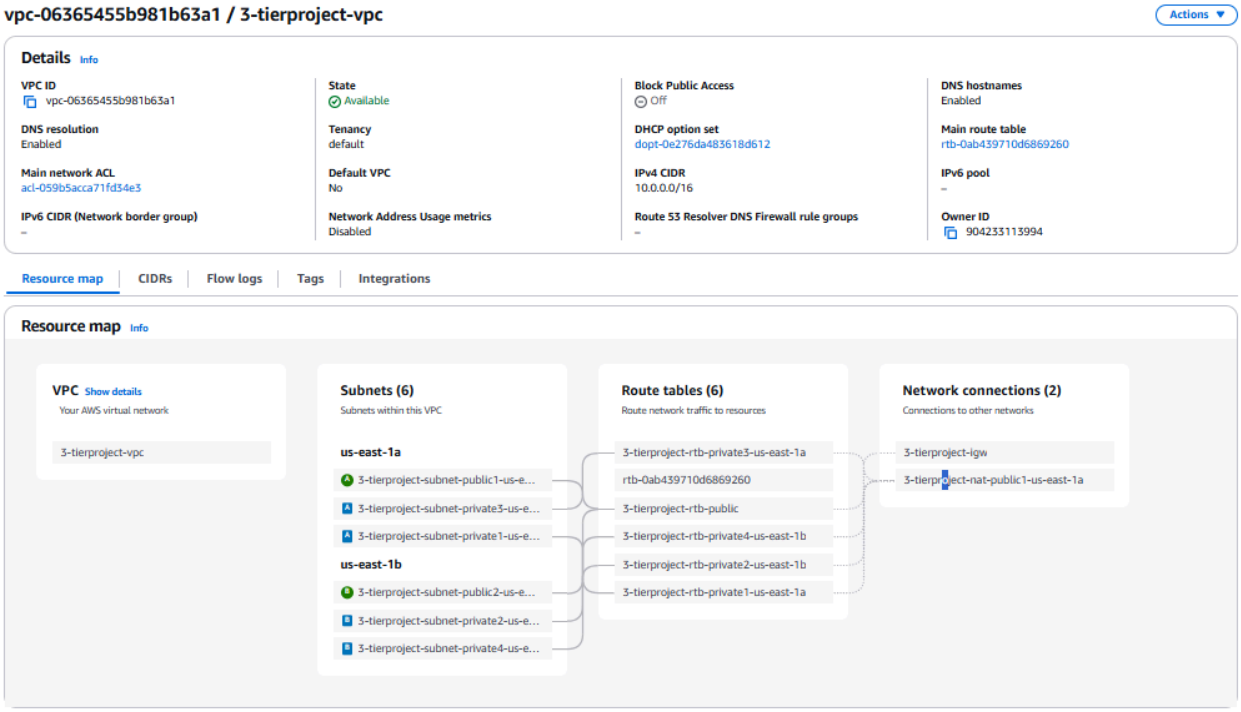
- Go to Certificate Manger in AWS -
- Click on “Request a Certificate”.
- Give your Domain Name, (Ex: `nimesh.xyz,www.nimesh.xyz`).
- Once Created, Click on “Create Records in Route53”.
- Certificate will be issued.



Step 3: Creating VPC & Subnets:

- Go to VPC Dashboard in AWS -
- Click on “Create VPC”.
- Click on “VPC and More”.
- Name: 3-tierproject.
- Availability Zones: 2
- Public Subnets: 2
- Private Subnets: 4
- Nat Gateway: In 1 AZ
- VPC End Points: None
- Click on Create.
- Go to Subnets, Select Public subnets.
- Click on Actions.
- Click on Edit Subnet Settings.

- Enable Public IP Automatically.



Step 4: Creating Security Groups:

4.1: Creating security group for Bastion Host

- Go to Security groups in EC2 Dashboard.
- Click on Create Security Group.
- Name: Bastion-Host.
- VPC: Select 3-tierproject.
- Inbound Rule: SSH from Anywhere from Ipv4.

4.2: Create a Security Group for Presentation Tier ALB

- Click on Create Security Group.
- Name: Presentation-Tier-ALB.
- VPC: Select 3-tierproject.
- Inbound Rule: HTTP from Anywhere from Ipv4.

4.3: Creating Security Group for Presentation Tier EC2

- Click on Create Security Group
- Name: Presentation-Tier-EC2
- VPC: Select 3-tierproject.
- Inbound Rules:
  - SSH from Bastion Host
  - HTTP from Presentation-Tier-ALB

4.4: Creating Security Group for Application Tier ALB

- Click on Create Security Group
- Name: Application-Tier-ALB
- VPC: Select 3-tierproject.
- Inbound Rule: HTTP from Presentation-Tier-EC2

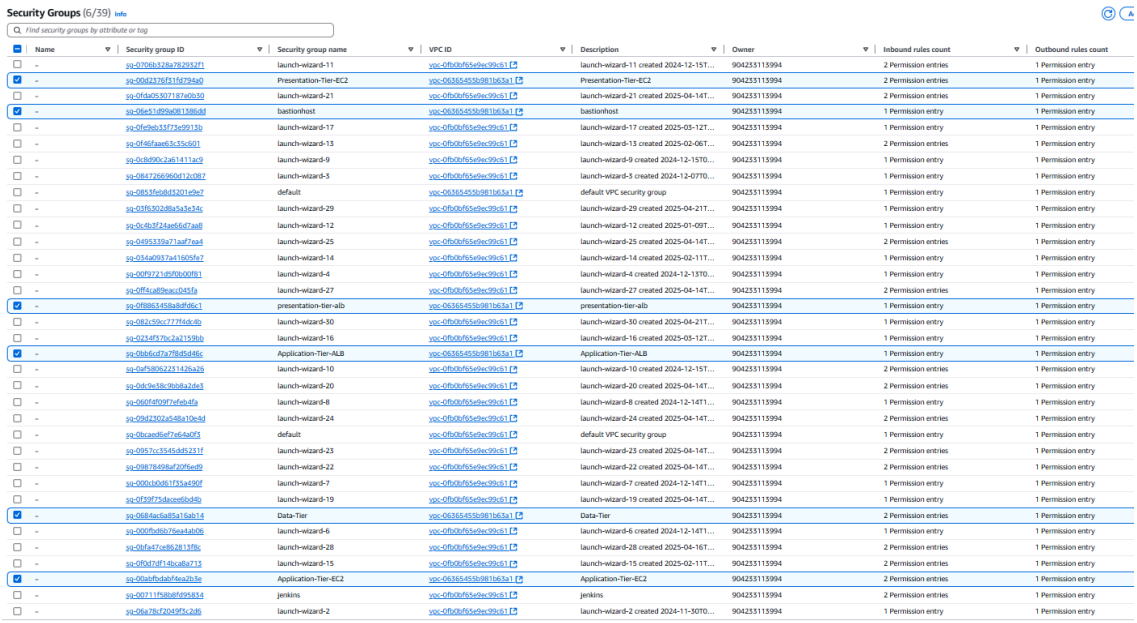
4.5: Creating Security Group for Application Tier EC2

- Click on Create Security Group
- Name: Application-Tier-EC2

- VPC: Select 3-tierproject.
- Inbound Rules:
  - SSH from Bastion Host
  - Custom TCP (Port 3200) from App-Tier-ALB

4.6: Creating Security Group for Data Tier

- Click on Create Security Group
- Name: Data-Tier
- VPC: Select 3-tierproject.
- Inbound Rules:
  - MySQL/Aurora from Bastion Host
  - MySQL/Aurora from Application-Tier-EC2.



Step 5: Launching Bastion Host:

Go to EC2 Dashboard and click on "Launch Instance." Configure the following settings:

• Name: Bastion-Host

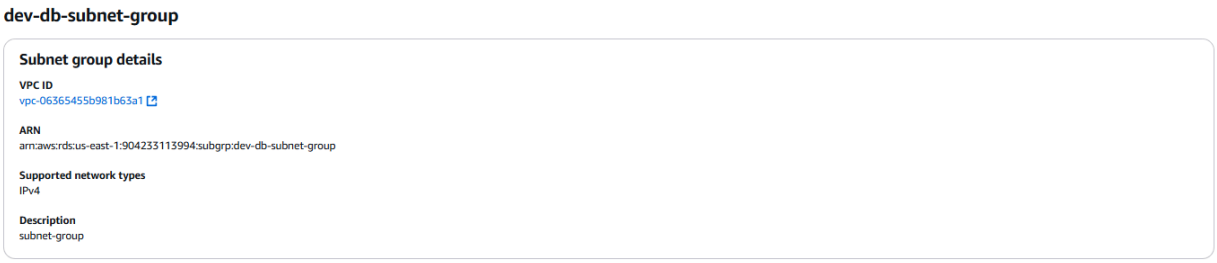
- AMI: Amazon Linux 2023 AMI
- Instance Type: t2.micro
- Key Pair: Select existing or create new
- VPC: 3-tierproject.
- Subnet: Public Subnet
- Security Group: Bastion-Host

Finally, click "Launch Instance."

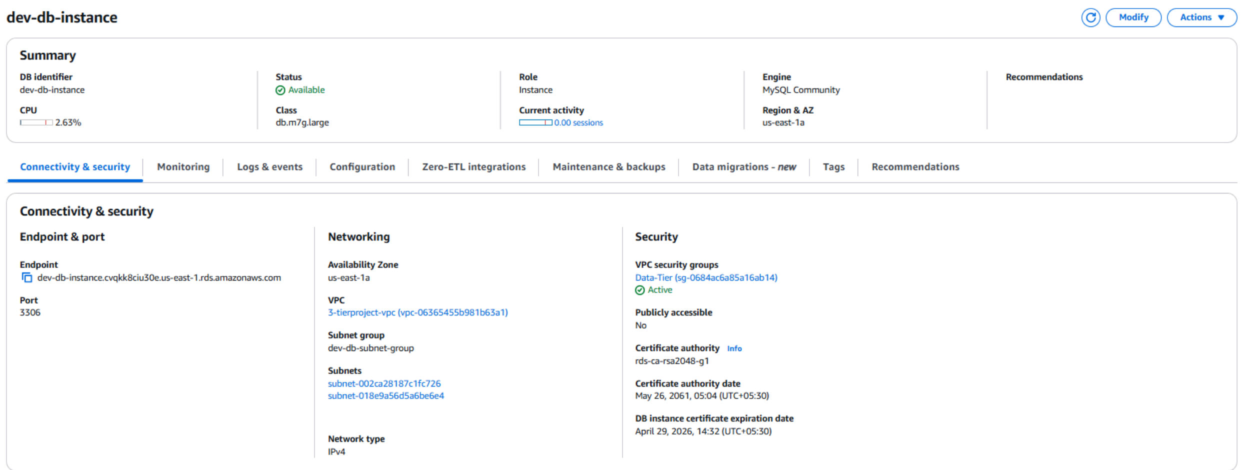
Step 6: Setting Up Data Tier with RDS:

Go to Amazon RDS in AWS. Click "Create DB Subnet Group" and configure:
- Name: dev-db-subnet-group
- VPC: 3-tierproject
- AZs: Select both
- Subnets: Select 2 private subnets

Click "Create Subnet Group."



Then click "Create Database" and configure:
- Choose Standard creation
- Select SQL. Template: dev/test
- Enable Multi-AZ DB Instance
- Database identifier: dev-db-instance
- Set password: Create your own (use "admin123")
- VPC: 3-tierproject
- DB subnet group: Choose the one created earlier
- Security group: Data-Tier



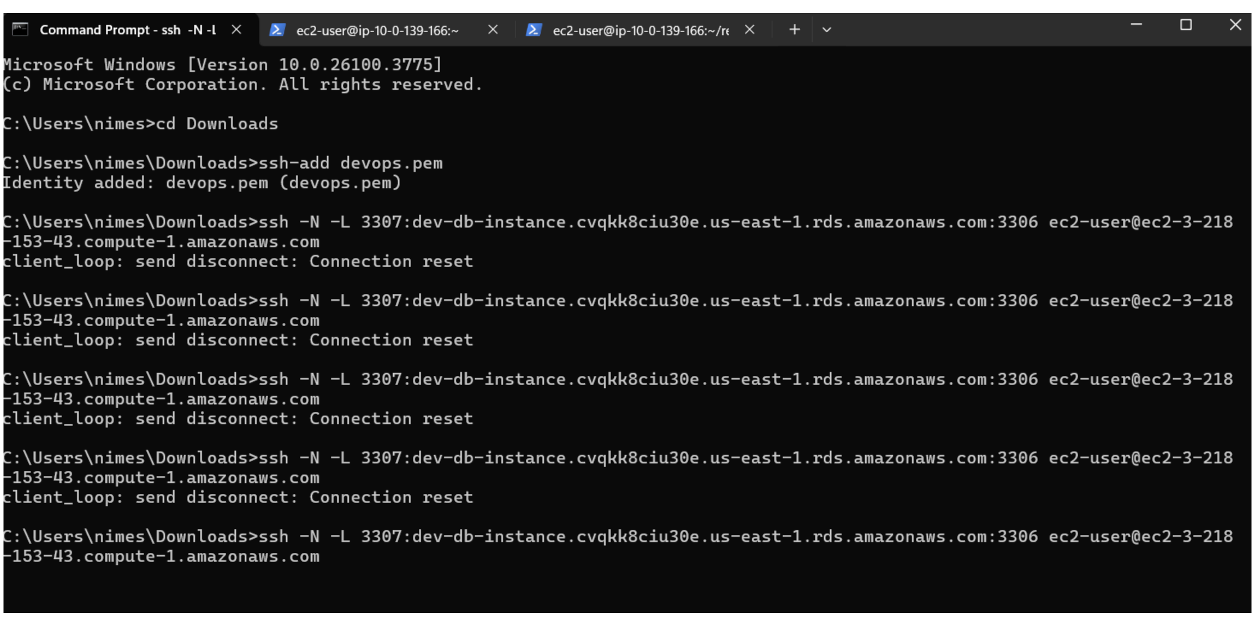
Click "Create Database." Note that database creation typically takes 15-20 minutes. After your database is running, open Command Prompt and locate your key pair.

- Add key to SSH Agent.

```
ssh-add your_key.pem
```

- Connect to DataBase.

```
ssh -N -L 3307:dev-db-instance.cvqkk8ci30e.us-east-1.rds.amazonaws.com:3306 ec2-user@ec2-2-3-218-153-43.compute-1.amazonaws.com
```



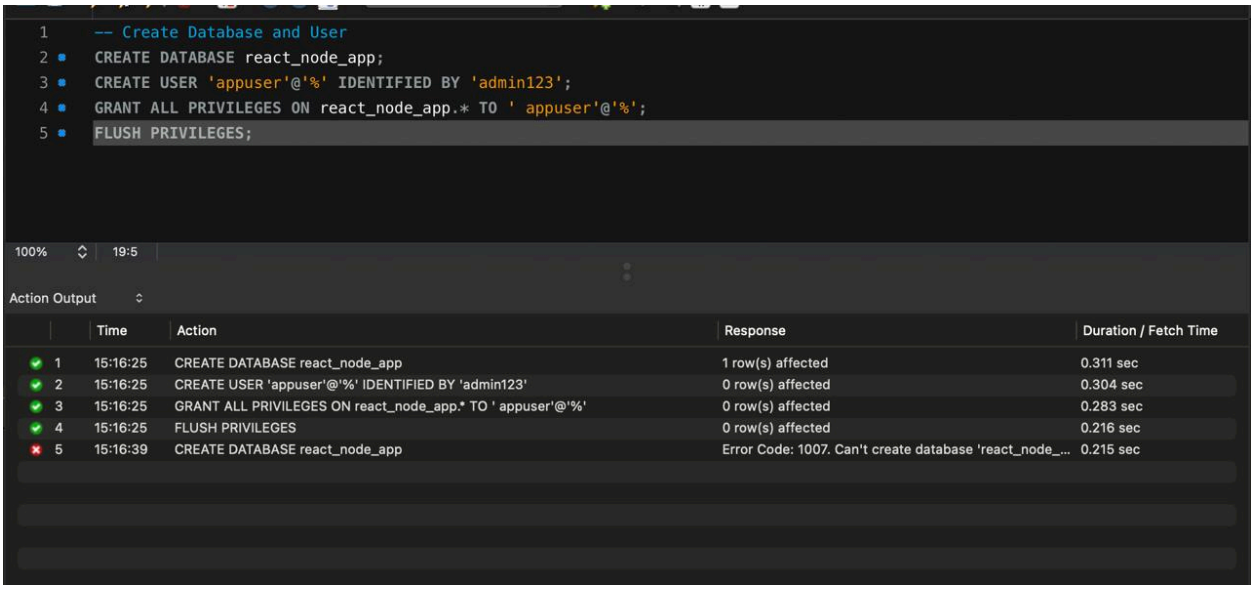
Open MySQL Workbench.

Click the "+" button and configure the connection:

- Name: rds-dev-db-admin
- Port: 3307
- Username: admin
- Store in Key: admin123

Test the connection. If successful, click OK. The connection will open in the SQL Editor.

run the below queries

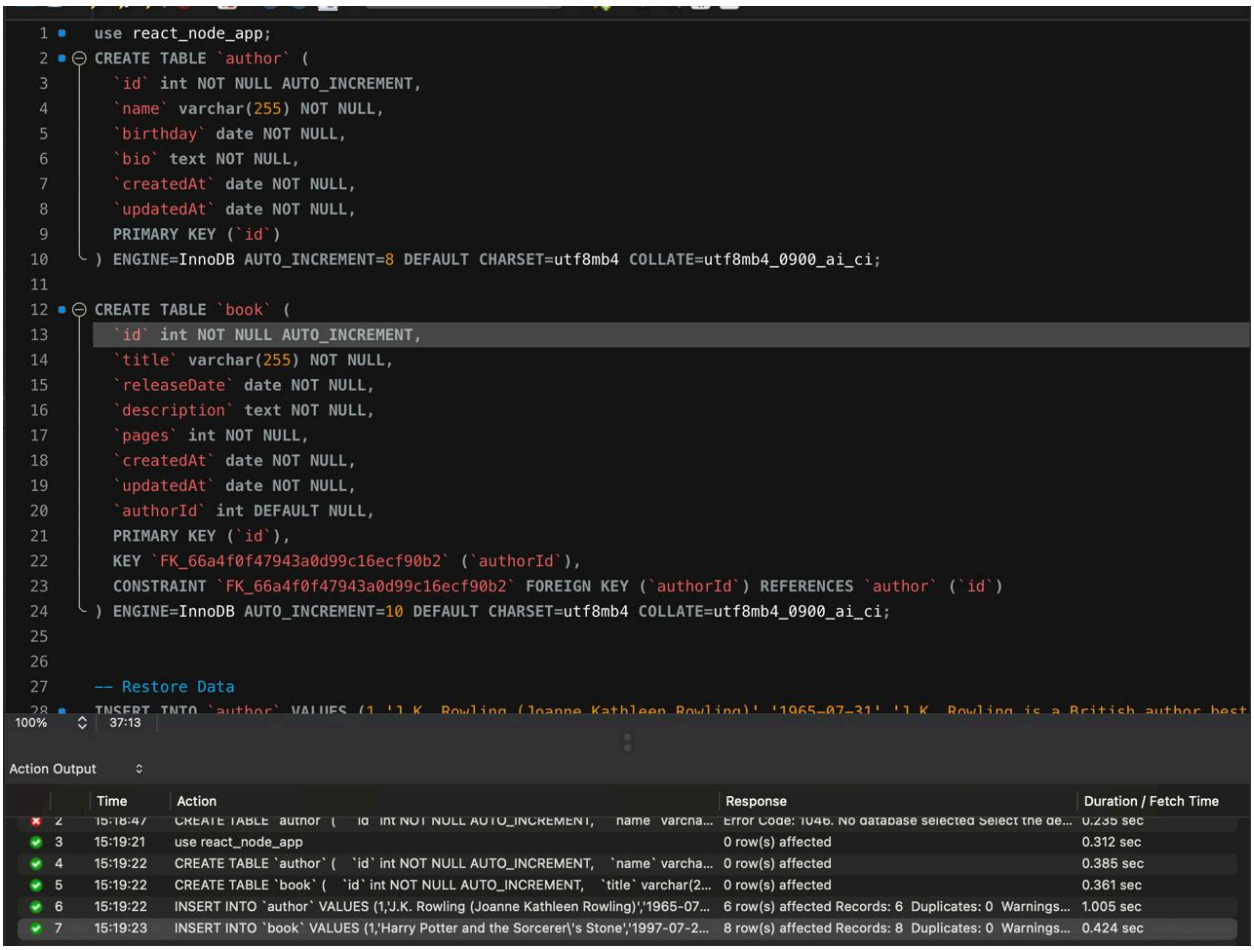


Open MySQL Workbench again and click the "+" button. Configure the connection with these settings:

- Name: rds-dev-db-appuser
- Port: 3307
- Username: appuser
- Store in Key: admin123

Test the connection. If successful, click OK.

open SQL Editor and run the following
Queries:



```sql
use react_node_app;
CREATE TABLE `author` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `birthday` date NOT NULL,
  `bio` text NOT NULL,
  `createdAt` date NOT NULL,
  `updatedAt` date NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `book` (
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL,
  `releaseDate` date NOT NULL,
  `description` text NOT NULL,
  `pages` int NOT NULL,
  `createdAt` date NOT NULL,
  `updatedAt` date NOT NULL,
  `authorId` int DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_66a4f0f47943a0d99c16ecf90b2` (`authorId`),
  CONSTRAINT `FK_66a4f0f47943a0d99c16ecf90b2` FOREIGN KEY (`authorId`) REFERENCES `author` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


-- Restore Data
INSERT INTO `author` VALUES (1,'J.K. Rowling (Joanne Kathleen Rowling)','1965-07-31','J.K. Rowling is a British author best
```

100% 37:13

Action Output

	Time	Action	Response	Duration / Fetch Time
2	15:18:47	CREATE TABLE `author` (`id` int NOT NULL AUTO_INCREMENT, `name` varcha...	Error Code: 1046. No database selected Select the de...	0.235 sec
3	15:19:21	use react_node_app	0 row(s) affected	0.312 sec
4	15:19:22	CREATE TABLE `author` (`id` int NOT NULL AUTO_INCREMENT, `name` varcha...	0 row(s) affected	0.385 sec
5	15:19:22	CREATE TABLE `book` (`id` int NOT NULL AUTO_INCREMENT, `title` varchar(2...	0 row(s) affected	0.361 sec
6	15:19:22	INSERT INTO `author` VALUES (1,'J.K. Rowling (Joanne Kathleen Rowling)','1965-07...	6 row(s) affected Records: 6 Duplicates: 0 Warnings...	1.005 sec
7	15:19:23	INSERT INTO `book` VALUES (1,'Harry Potter and the Sorcerer\'s Stone','1997-07-2...	8 row(s) affected Records: 8 Duplicates: 0 Warnings...	0.424 sec

You'll find this code in repository
backend folder/db.sql

https://github.com/n-nimesh/react-node-mysql-app

Step 7: Setting Up Presentation Tier:

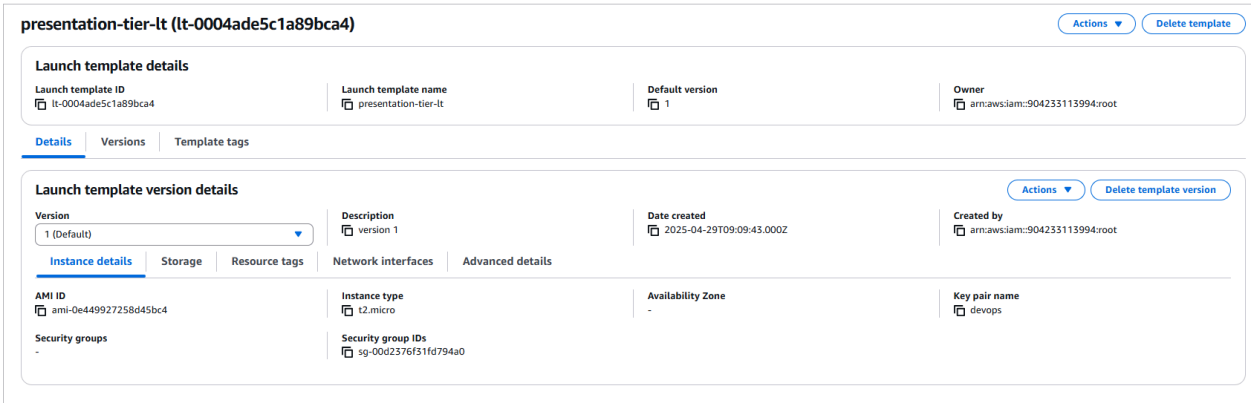
7.1 Creating Launch Template

Go to Launch Templates in EC2 dashboard:

- Name: presentation-tier-lt.
- Version: 01
- Enable Auto Scaling Guidance
- AMI: Amazon Linux (Quick Start)
- Instance Type: t2.micro
- Key Pair: Select the same one used for bastion host
- Security Group: Presentation-Tier-EC2
- Advanced Settings → User Data: Paste the following script

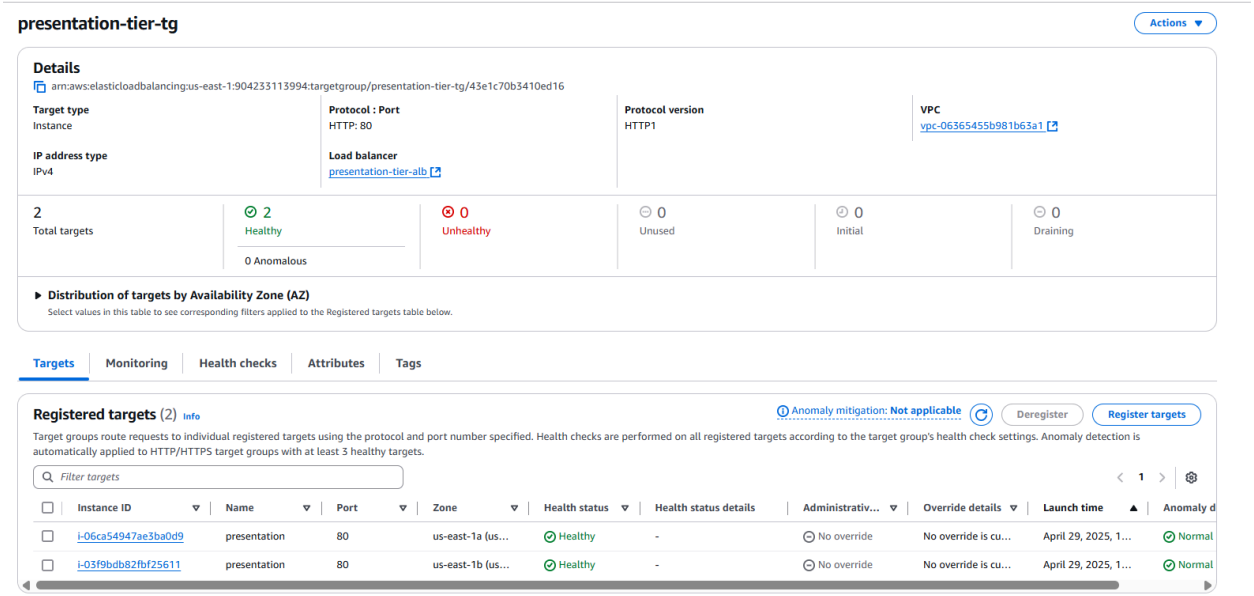
For Auto Scaling Group setup.

```
#!/bin/bash
# Update the package list and install NGINX
sudo yum update -y
sudo yum install nginx -y

# Start and enable NGINX
sudo systemctl start nginx
sudo systemctl enable nginx

# Fetch metadata token
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

# Fetch instance details using IMDSv2
INSTANCE_ID=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" "http://169.254.169.254/latest/meta-data/instance-id")
AVAILABILITY_ZONE=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" "http://169.254.169.254/latest/meta-data/placement/availability-
PUBLIC_IP=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" "http://169.254.169.254/latest/meta-data/public-ipv4")

# Create a simple HTML page displaying instance details
sudo bash -c "cat > /usr/share/nginx/html/index.html <<EOF
<h1>Instance Details</h1>
<p><b>Instance ID:</b> $INSTANCE_ID</p>
<p><b>Availability Zone:</b> $AVAILABILITY_ZONE</p>
<p><b>Public IP:</b> $PUBLIC_IP</p>
EOF"

# Restart NGINX to ensure changes are applied
sudo systemctl restart nginx
```



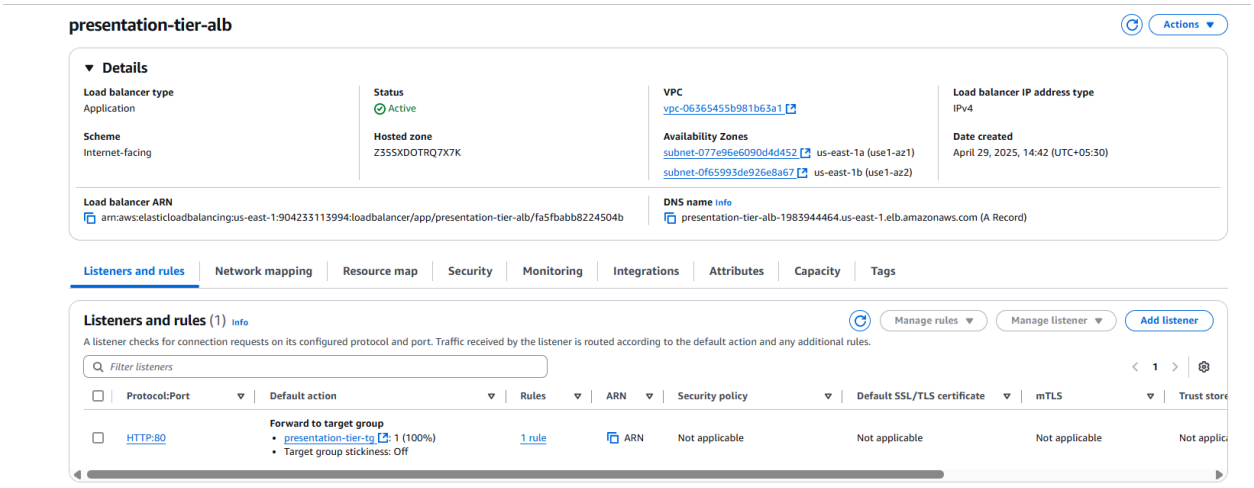
7.2 Creating Presentation Tier Target Group

- Navigate to Target Groups in EC2 Dashboard
- Select Instances
- Name: Presentation-Tier-TG
- VPC: Select 3-tierproject
- Health Check Path: /health
- Click Next, then Create Target Group



7.3 Creating Presentation Tier Load Balancer

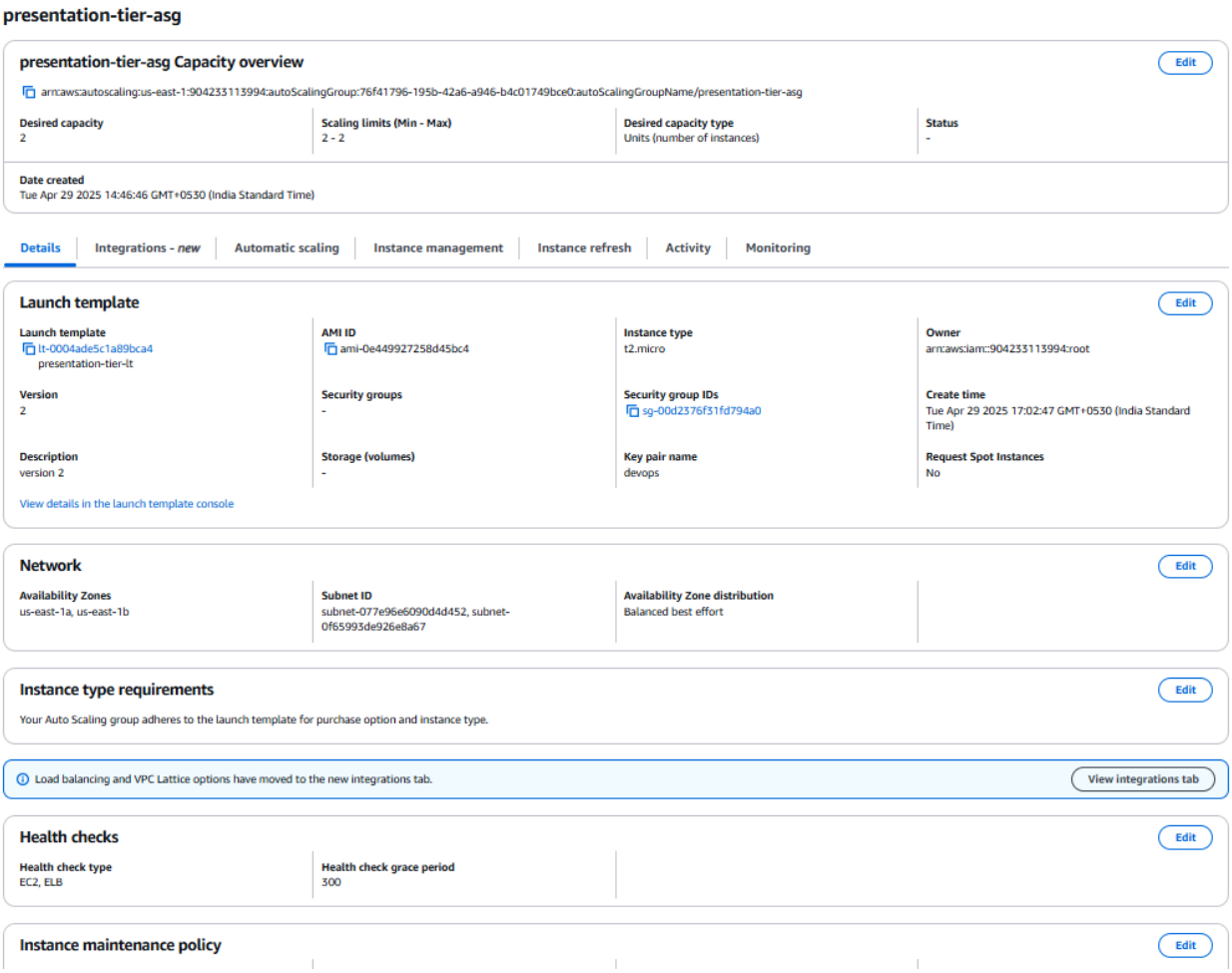
- Go to Load Balancer in EC2 Dashboard
- Click Create Load Balancer
- Name: Presentation-Tier-ALB
- Type: Internet Facing
- VPC: Select 3-tierproject
- Subnets: Select Public Subnets
- Security Group: Presentation-Tier-ALB
- Select the previously created Target Group
- Click Create Load Balancer.



7.4 Creating Presentation Tier Auto Scaling Group

- Navigate to Auto Scaling Group in EC2 Dashboard
- Click Create ASG
- Name: Presentation-Tier-ASG
- Select Launch Template from step 7.1
- VPC: 3-tierproject
- Subnets: Select public subnets
- Click Next
- Attach to existing Load Balancer and select "Presentation-Tier-TG"

- Enable Load Balancer health checks
- Enable CloudWatch monitoring
- Click Next
- Set capacity:
  - Desired: 3
  - Minimum: 2
  - Maximum: 4
    - Select Target Tracking Scaling Policy
    - Set Average CPU Utilization to 90%
    - Create ASG.



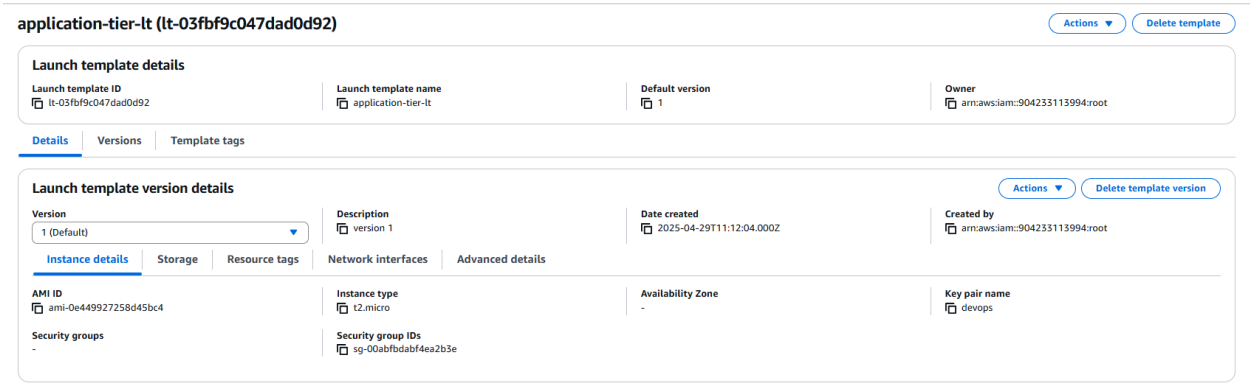
Once configured correctly, three instances will appear in the EC2 Dashboard. Access the Load Balancer DNS name to verify the Instance Metadata Details.

Step 8: Setup Application Tier:

8.1 Creating Launch Template:
• Name: Application-Tier-lt
• Version: 1
• Enable Auto Scaling Guidance
• AMI: Amazon Linux
• Instance Type: T2.micro
• Key Pair: Select existing key pair
• Security Group: Application-Tier-EC2
• Advanced Settings → User Data: Copy and paste the script below with these modifications:

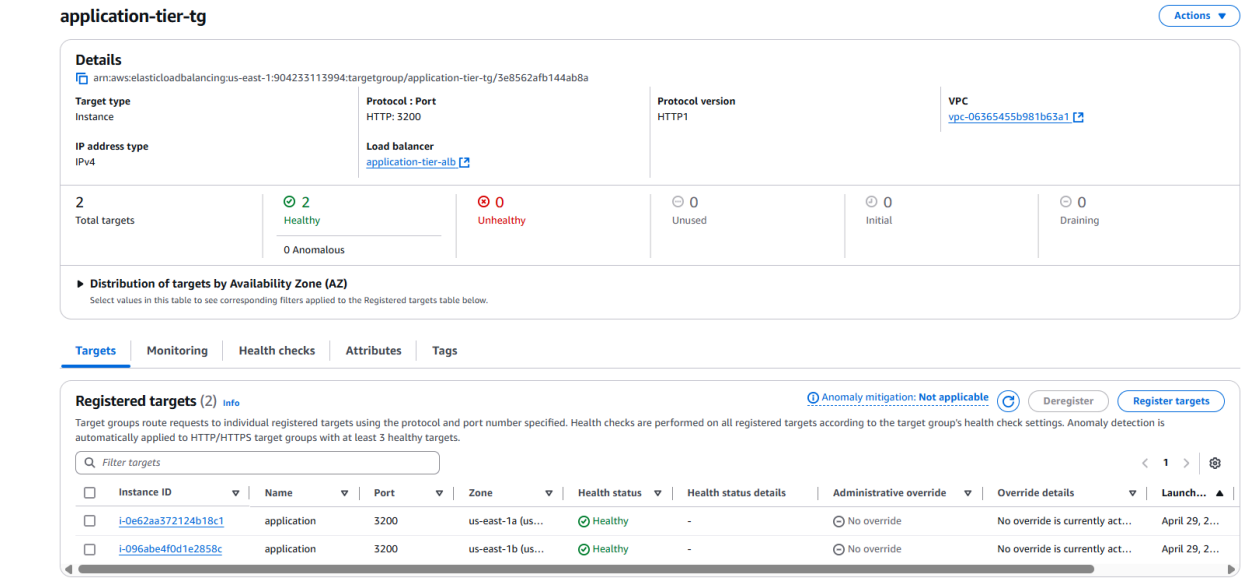
- RDS Endpoint
- Database User: "appuser"
- Database Password: "admin123"
- Database Name: "react_node_app"

```
#!/bin/bash
# Update package list and install required packages
sudo yum update -y
sudo yum install -y git

# Install Node.js (use NodeSource for the latest version)
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install -y nodejs

# Install PM2 globally
sudo npm install -g pm2

# Define variables
REPO_URL="https://github.com/suneelprojects/react-node-mysql-app.git"
BRANCH_NAME="feature/add-logging"
REPO_DIR="/home/ec2-user/react-node-mysql-app/backend"
ENV_FILE="$REPO_DIR/.env"

# Clone the repository
cd /home/ec2-user
sudo -u ec2-user git clone $REPO_URL
cd react-node-mysql-app

# Checkout to the specific branch
sudo -u ec2-user git checkout $BRANCH_NAME
cd backend

# Define the log directory and ensure it exists
LOG_DIR="/home/ec2-user/react-node-mysql-app/backend/logs"
mkdir -p $LOG_DIR
sudo chown -R ec2-user:ec2-user $LOG_DIR

# Append environment variables to the .env file
echo "LOG_DIR=$LOG_DIR" >> "$ENV_FILE"
echo "DB_HOST=\"<rds-instance.end.point.region.rds.amazonaws.com>\"" >> "$ENV_FILE"
echo "DB_PORT=\"3306\"" >> "$ENV_FILE"
echo "DB_USER=\"<db-user>\"" >> "$ENV_FILE"
echo "DB_PASSWORD=\"<db-user-password>\"" >> "$ENV_FILE"  # Replace with actual password
echo "DB_NAME=\"<db-name>\"" >> "$ENV_FILE"

# Install Node.js dependencies as ec2-user
sudo -u ec2-user npm install

# Start the application using PM2 as ec2-user
sudo -u ec2-user npm run serve

# Ensure PM2 restarts on reboot as ec2-user
sudo -u ec2-user pm2 startup systemd
sudo -u ec2-user pm2 save
```



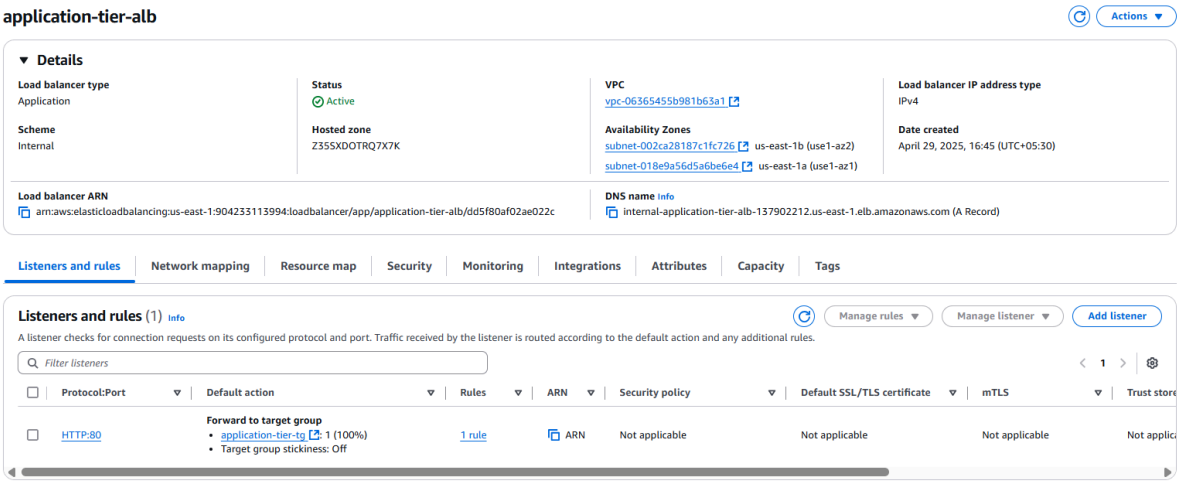
8.2 Creating Target Group

- Click on Create Target Group → Instances
- Name: Application-Tier-TG
- Port: 3200
- VPC: 3-Tier
- Health Check Path: /health
- Click Create Target Group



8.3 Creating Application Tier Load Balancer

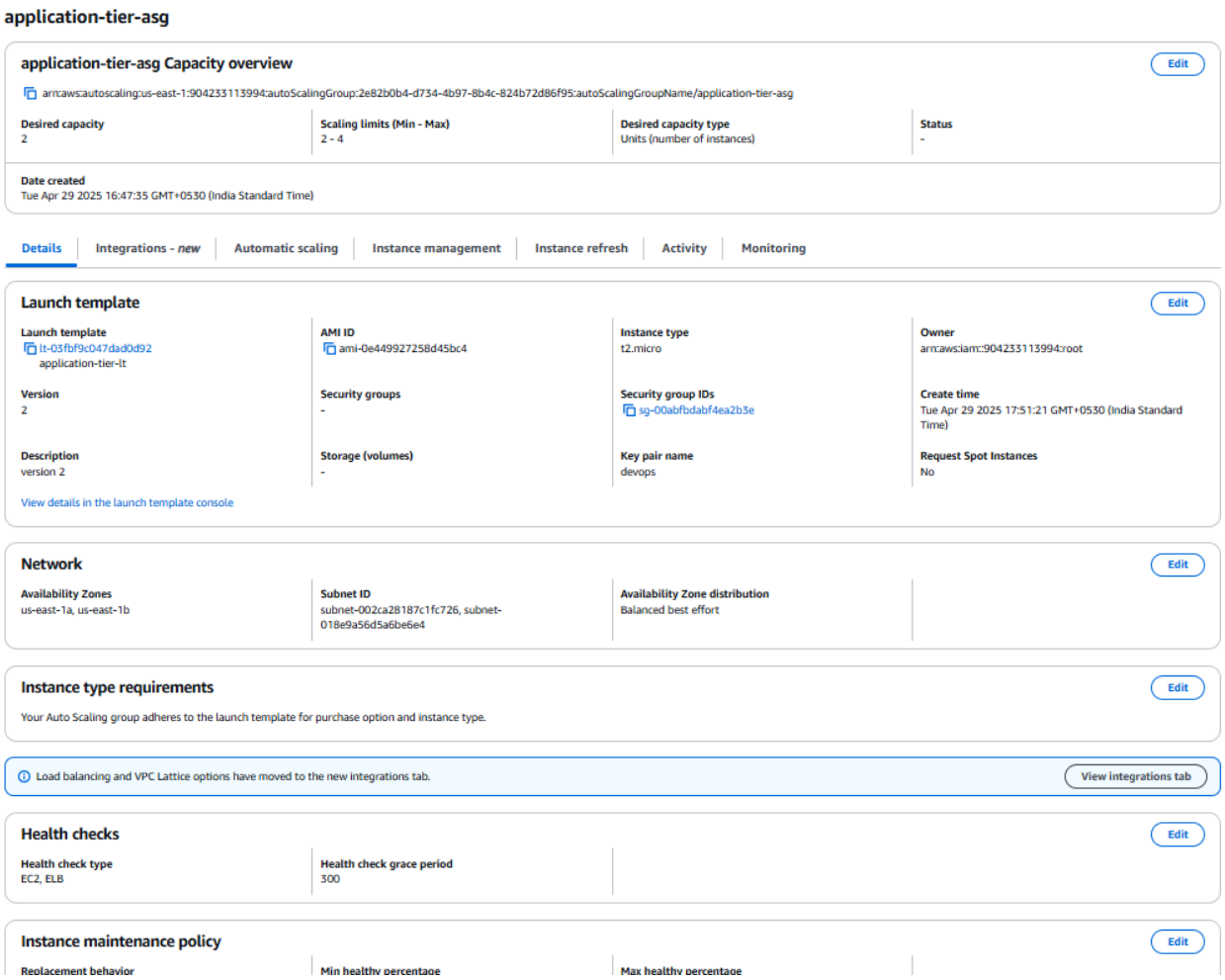
- Click on Create Load Balancer
- Name: Application-Tier-ALB
- Scheme/Type: Internal
- VPC: 3-Tier-Architecture
- Subnets: Select private subnets
- Security Group: Application-Tier-ALB
- Target Group: Select Application-Tier-TG
- Click Create Load Balancer



8.4 Creating Auto Scaling Group for Application Tier

- Click on Create Auto Scaling Group
- Select Launch Template: Application-Tier-lt
- VPC: 3-Tier-Architecture
- Subnets: Private
- Load Balancer: Select existing "Application-Tier-TG"
- Enable Load Balancer health checks
- Enable CloudWatch Monitoring
- Configure capacity:
  - Desired: 3
  - Minimum: 4

- Maximum: 4
  - Set Target Tracking Scaling Policy with 90% average CPU utilization
  - Click Create ASG



Once configured correctly, three new instances will appear in EC2. To verify the backend is working, connect to the Bastion Host from Terminal:

`ssh -A ec2-user@` BastionHost IP

Login to Private Instance of Application Tier

`ssh ec2-user@private-ip`

Once Connected, Run the following command

`pm2 logs`

```
Last login: Tue Apr 29 10:35:01 2025 from 124.123.164.161
[ec2-user@ip-10-0-9-243 ~]$ ssh ec2-user@10.0.139.166
The authenticity of host '10.0.139.166 (10.0.139.166)' can't be established.
ED25519 key fingerprint is SHA256:JPmnroCpne/NLadiuSpvAF39ETGrbyqRBtjdBWGJFZE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.139.166' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-139-166 ~]$ pm2 logs
[TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)
/home/ec2-user/.pm2/pm2.log last 15 lines:
PM2        | 2025-04-29T11:19:18: PM2 log: Node.js version      : 18.20.8
PM2        | 2025-04-29T11:19:18: PM2 log: Current arch         : x64
PM2        | 2025-04-29T11:19:18: PM2 log: PM2 home             : /home/ec2-user/.pm2
PM2        | 2025-04-29T11:19:18: PM2 log: PM2 PID file         : /home/ec2-user/.pm2/pm2.pid
PM2        | 2025-04-29T11:19:18: PM2 log: RPC socket file      : /home/ec2-user/.pm2/rpc.sock
PM2        | 2025-04-29T11:19:18: PM2 log: BUS socket file      : /home/ec2-user/.pm2/pub.sock
PM2        | 2025-04-29T11:19:18: PM2 log: Application log path : /home/ec2-user/.pm2/logs
PM2        | 2025-04-29T11:19:18: PM2 log: Worker Interval      : 30000
PM2        | 2025-04-29T11:19:18: PM2 log: Process dump file    : /home/ec2-user/.pm2/dump.pm2
PM2        | 2025-04-29T11:19:18: PM2 log: Concurrent actions   : 2
PM2        | 2025-04-29T11:19:18: PM2 log: SIGTERM timeout      : 1600
PM2        | 2025-04-29T11:19:18: PM2 log: Runtime Binary       : /usr/bin/node
PM2        | 2025-04-29T11:19:18: PM2 log: ============================================================
PM2        | 2025-04-29T11:19:18: PM2 log: App [server:0] starting in -fork mode-
PM2        | 2025-04-29T11:19:18: PM2 log: App [server:0] online

/home/ec2-user/.pm2/logs/server-error.log last 15 lines:
/home/ec2-user/.pm2/logs/server-out.log last 15 lines:
0|server   | Server is running on port http://localhost:3200
0|server   | 2025-04-29 11:19:18 [INFO]: Connected to MySQL Database
```

Step 9: Modifying Presentation Tier Launch Template:

Now, we will deploy our ReactJS Frontend.

9.1 Modifying Launch Template

- Go to Launch Templates
- Select Presentation-Tier-lt
- Actions → Modify →Version: 2

- Go to Advanced → User Data → Copy and paste the below script with necessary changes

  Changes to do: -Add Application Tier DNS Name -

  Add Domain names `nimesh.xyz,www.nimesh.xyz`

```
#!/bin/bash
# Update package list and install required packages
sudo yum update -y
sudo yum install -y git

# Install Node.js (use NodeSource for the latest version)
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install -y nodejs

# Install NGINX
sudo yum install -y nginx

# Start and enable NGINX
sudo systemctl start nginx
sudo systemctl enable nginx

# Define variables
REPO_URL="https://github.com/suneelprojects/react-node-mysql-app.git"
BRANCH_NAME="feature/add-logging"
REPO_DIR="/home/ec2-user/react-node-mysql-app/frontend"
ENV_FILE="$REPO_DIR/.env"
APP_TIER_ALB_URL="http://<internal-application-tier-alb-end-point.region.elb.amazonaws.com>"  # Replace with your actual alb en
API_URL="/api"

# Clone the repository as ec2-user
cd /home/ec2-user
sudo -u ec2-user git clone $REPO_URL
cd react-node-mysql-app

# Checkout to the specific branch
sudo -u ec2-user git checkout $BRANCH_NAME
cd frontend

# Ensure ec2-user owns the directory
sudo chown -R ec2-user:ec2-user /home/ec2-user/react-node-mysql-app

# Create .env file with the API_URL
echo "VITE_API_URL=\"$API_URL\"" >> "$ENV_FILE"

# Install Node.js dependencies as ec2-user
sudo -u ec2-user npm install

# Build the frontend application as ec2-user
sudo -u ec2-user npm run build

# Copy the build files to the NGINX directory
sudo cp -r dist /usr/share/nginx/html/
```

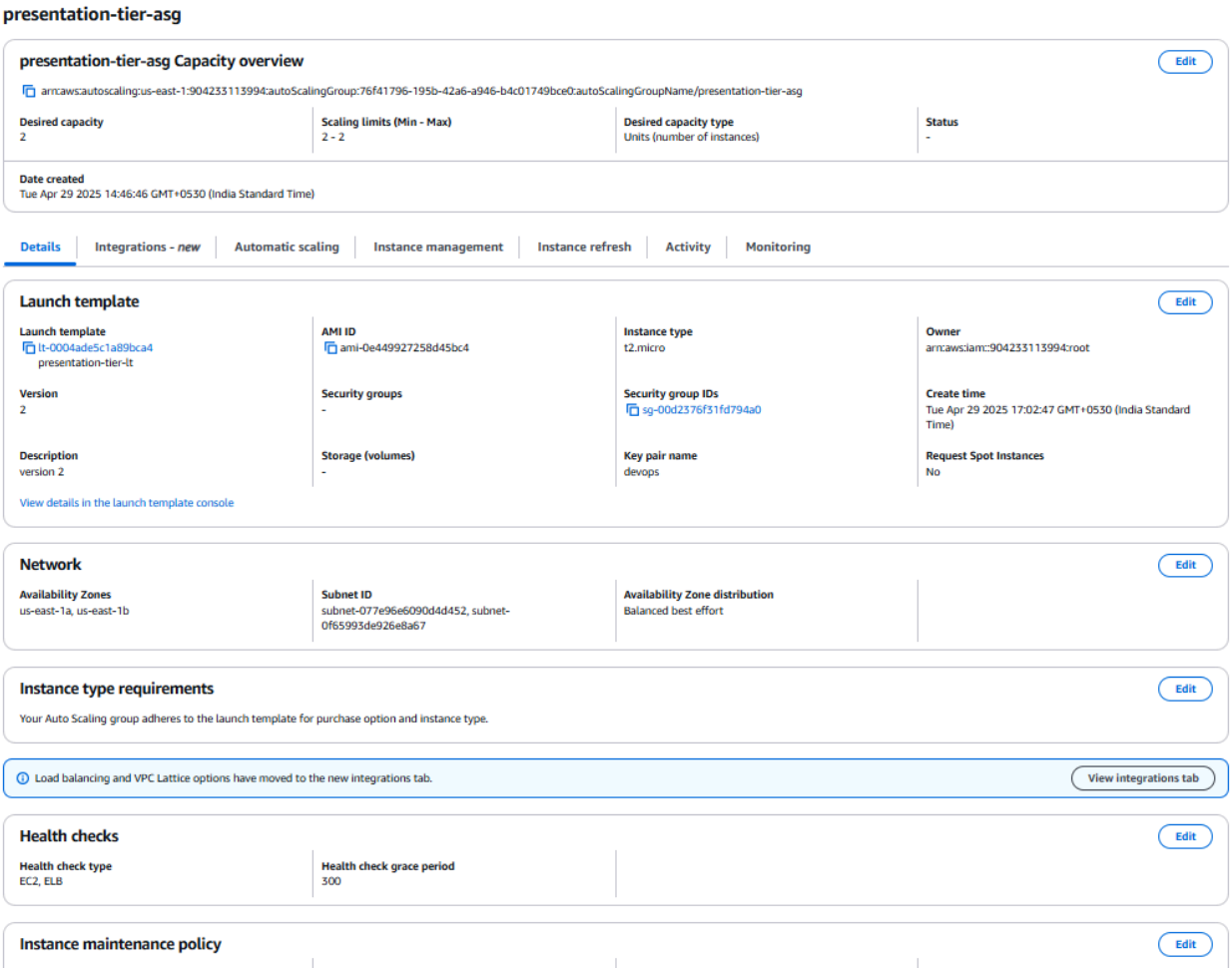
```
# Update NGINX configuration
NGINX_CONF="/etc/nginx/nginx.conf"
SERVER_NAME="<domain subdomain>"  # Replace with your actual domain name

# Backup existing NGINX configuration
sudo cp $NGINX_CONF ${NGINX_CONF}.bak

# Write new NGINX configuration
sudo tee $NGINX_CONF > /dev/null <<EOL
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log warn;
pid /run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format main '\$remote_addr - \$remote_user [\$time_local] "\$request" '
                    '\$status \$body_bytes_sent "\$http_referer" '
                    '"\$http_user_agent" "\$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;

    include /etc/nginx/conf.d/*.conf;
}
EOL
```

```
# Create a separate NGINX configuration file
sudo tee /etc/nginx/conf.d/presentation-tier.conf > /dev/null <<EOL
server {
    listen 80;
    server_name $SERVER_NAME;
    root /usr/share/nginx/html/dist;
    index index.html index.htm;

    #health check
    location /health {
        default_type text/html;
        return 200 "<!DOCTYPE html><p>Health check endpoint</p>\n";
    }

    location / {
        try_files \$uri /index.html;
    }

    location /api/ {
        proxy_pass $APP_TIER_ALB_URL;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}
EOL


# Restart NGINX to apply the new configuration
sudo systemctl restart nginx
```

- Click on Update

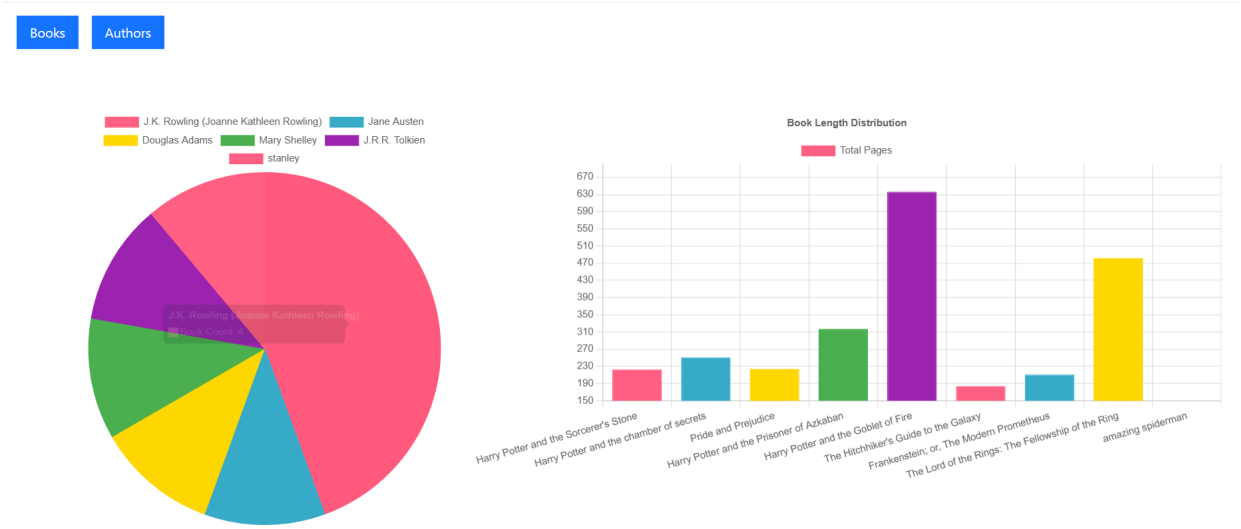
9.2 Modifying Auto Scaling Group of Presentation Tier

- Click on Edit
- Desired: 2
- Min: 2
- Max: 2
- Select Launch Template Version: 2
- click on update.



- Terminate Old Instances to get new instances with new configuration.

Now You can Access the Application using DNS Name of Presentation Tier Load Balancer.



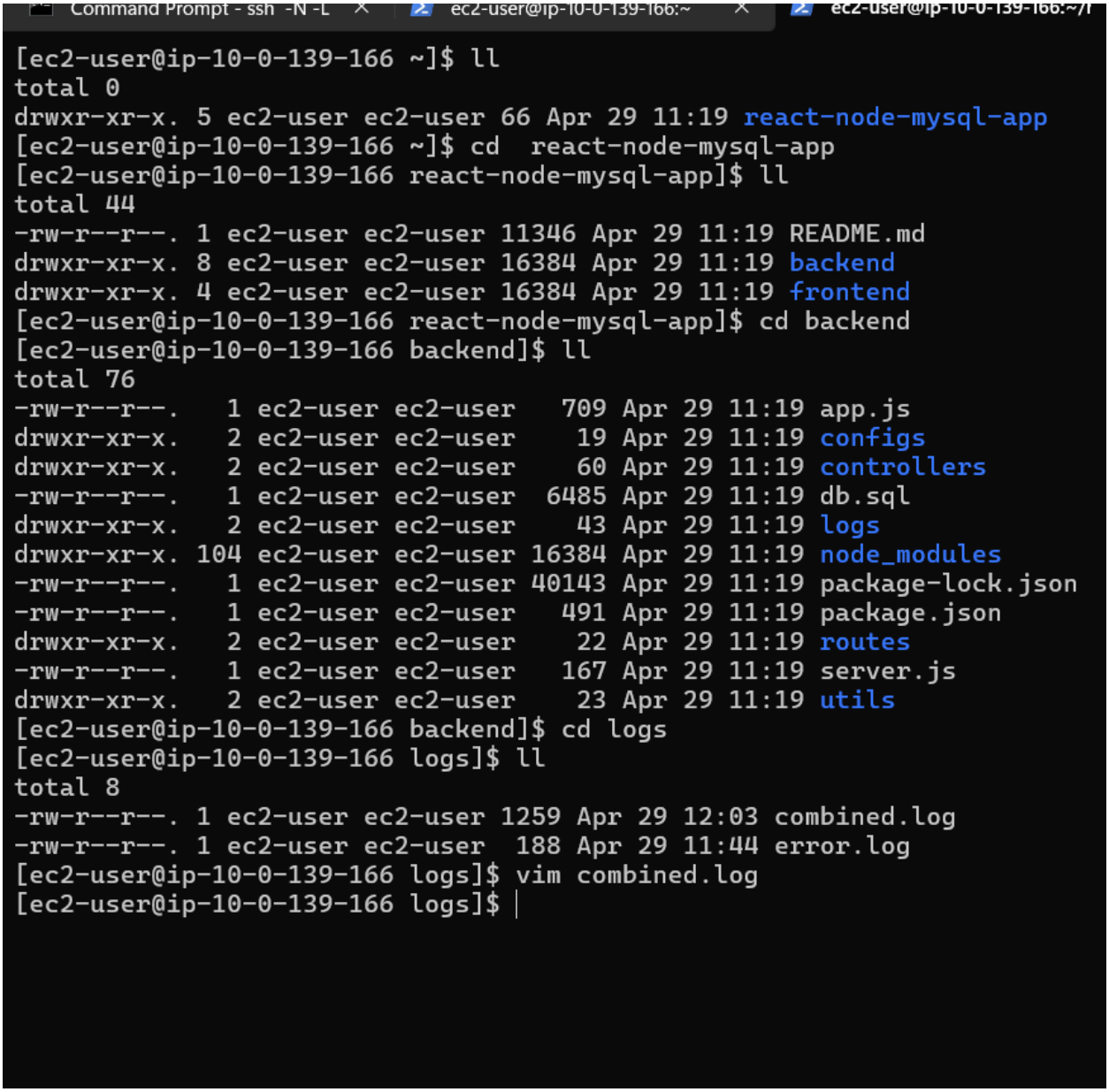
Step 10: Integrating Application Logs with CloudWatch

Connect one of our App Tier Ec2 instances

```
ls
```

Go to Repo Directory/bakcend/logs

```
vim combined.log
```

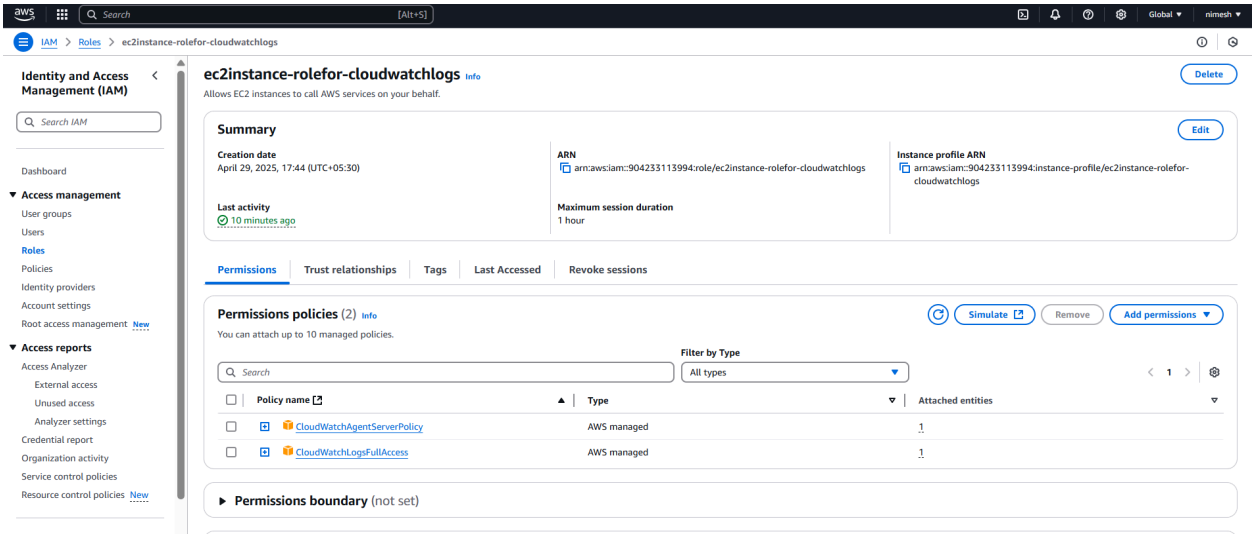


You can See logs

10.1 Creating IAM Role

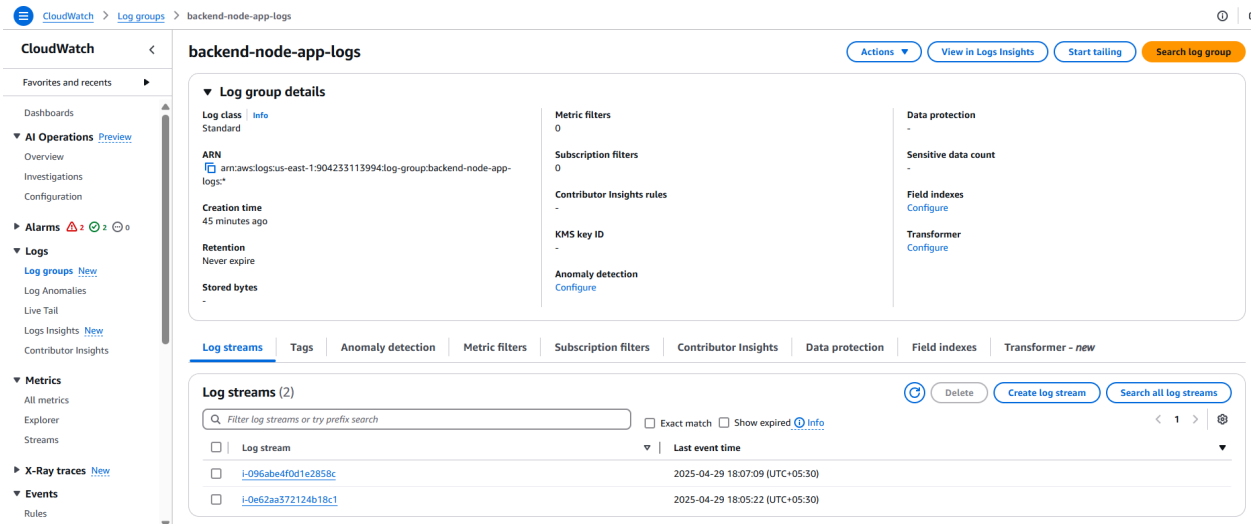
- Creating IAM Role for EC2 CloudWatch Logs
- Navigate to **IAM Dashboard** → **Roles**.
- Click **Create Role** → Select **EC2**.

- Click **Create**.



10.2 Creating CloudWatch Log Group

- Navigate to **CloudWatch** → **Log Groups**.
- Click **Create Log Group**.
- Enter **Name:** `backend-node-app-logs`.
- Click **Create**.



Step11:Modifying Application Tier Launch Template

- Select Application-Tier-lt
- Select Version 2
- Go to Advanced Details
- Select IAM Instance Profile
- Attach Role We created
- Enable Detailed CloudWatch Monitoring
- User Data → Add the Below Script

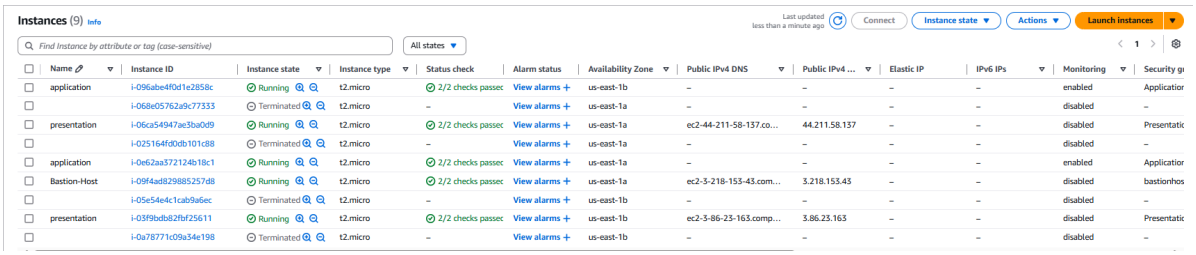
```
#!/bin/bash
# Update package list and install required packages
sudo yum update -y
sudo yum install -y git

# Install Node.js (use NodeSource for the latest version)
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install -y nodejs

# Install PM2 globally
sudo npm install -g pm2

# Define variables
REPO_URL="https://github.com/suneelprojects/react-node-mysql-app.git"
BRANCH_NAME="feature/add-logging"
REPO_DIR="/home/ec2-user/react-node-mysql-app/backend"
ENV_FILE="$REPO_DIR/.env"

# Clone the repository
cd /home/ec2-user
sudo -u ec2-user git clone $REPO_URL
cd react-node-mysql-app

# Checkout to the specific branch
sudo -u ec2-user git checkout $BRANCH_NAME
cd backend

# Define the log directory and ensure it exists
LOG_DIR="/home/ec2-user/react-node-mysql-app/backend/logs"
mkdir -p $LOG_DIR
sudo chown -R ec2-user:ec2-user $LOG_DIR

# Append environment variables to the .env file
echo "LOG_DIR=$LOG_DIR" >> "$ENV_FILE"
echo "DB_HOST=\"<rds-instance.end.point.region.rds.amazonaws.com>\"" >> "$ENV_FILE"
echo "DB_PORT=\"3306\"" >> "$ENV_FILE"
echo "DB_USER=\"<db-user>\"" >> "$ENV_FILE"
echo "DB_PASSWORD=\"<db-user-password>\"" >> "$ENV_FILE"  # Replace with actual password
echo "DB_NAME=\"<db-name>\"" >> "$ENV_FILE"

# Install Node.js dependencies as ec2-user
sudo -u ec2-user npm install

# Start the application using PM2 as ec2-user
sudo -u ec2-user npm run serve

# Ensure PM2 restarts on reboot as ec2-user
sudo -u ec2-user pm2 startup systemd
sudo -u ec2-user pm2 save
```

```
# Restart NGINX to apply the new configuration
sudo systemctl restart nginx

# Install CloudWatch agent
sudo yum install -y amazon-cloudwatch-agent

# Create CloudWatch agent configuration
sudo tee /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json >
/dev/null <<EOL
{
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/home/ec2-user/react-node-mysql-app/backend/logs/*.log",
            "log_group_name": "backend-node-app-logs",
            "log_stream_name": "{instance_id}",
            "timestamp_format": "%Y-%m-%d %H:%M:%S"
          }
        ]
      }
    }
  }
}
EOL
```

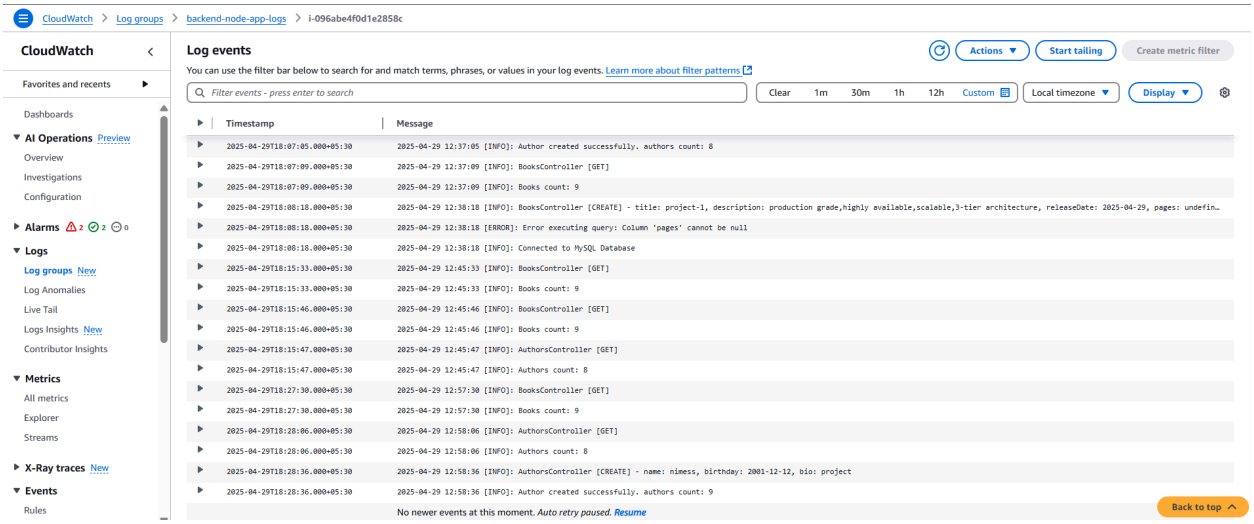
```
# Start CloudWatch agent
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a
fetch-config -m ec2 -c
file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json -s
```

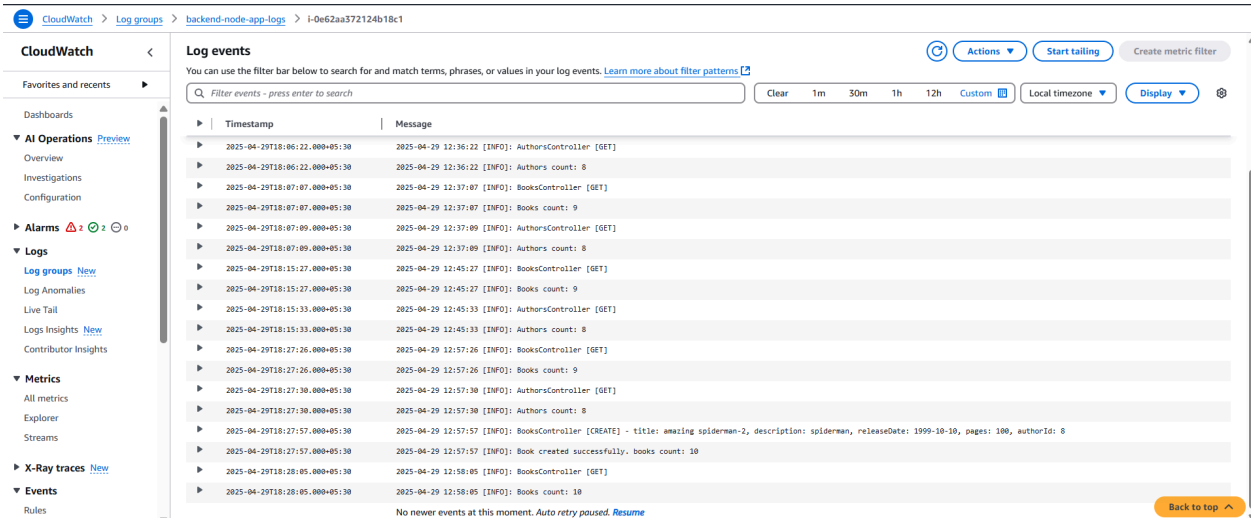
11.1 Modifying ASG

- Click on Application-Tier-ASG
- Edit - Version 2 - Update
  To see changes, Terminate the Old Instances, New instances will get created



Access the Application → Add Book → Go to Cloud Watch Log Group → Check Logs

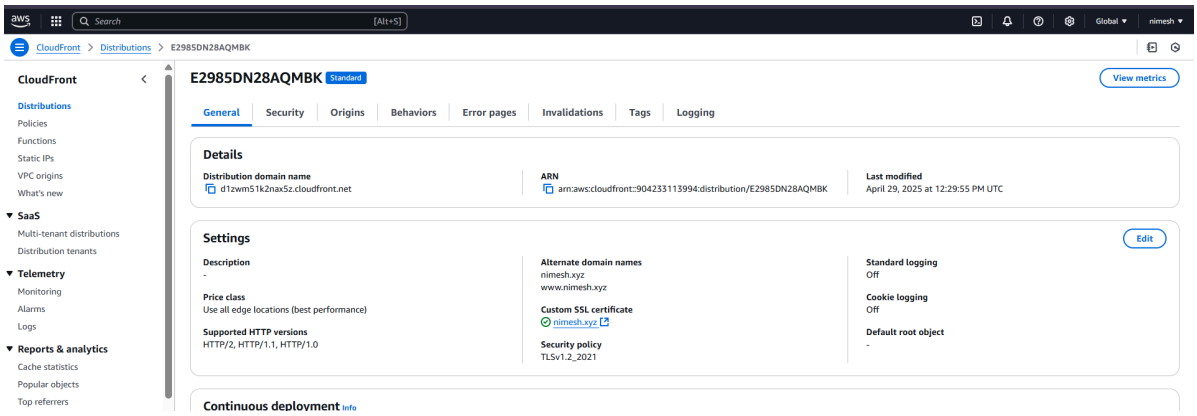




Step 12: Creating CloudFront Distribution

Go to CloudFront

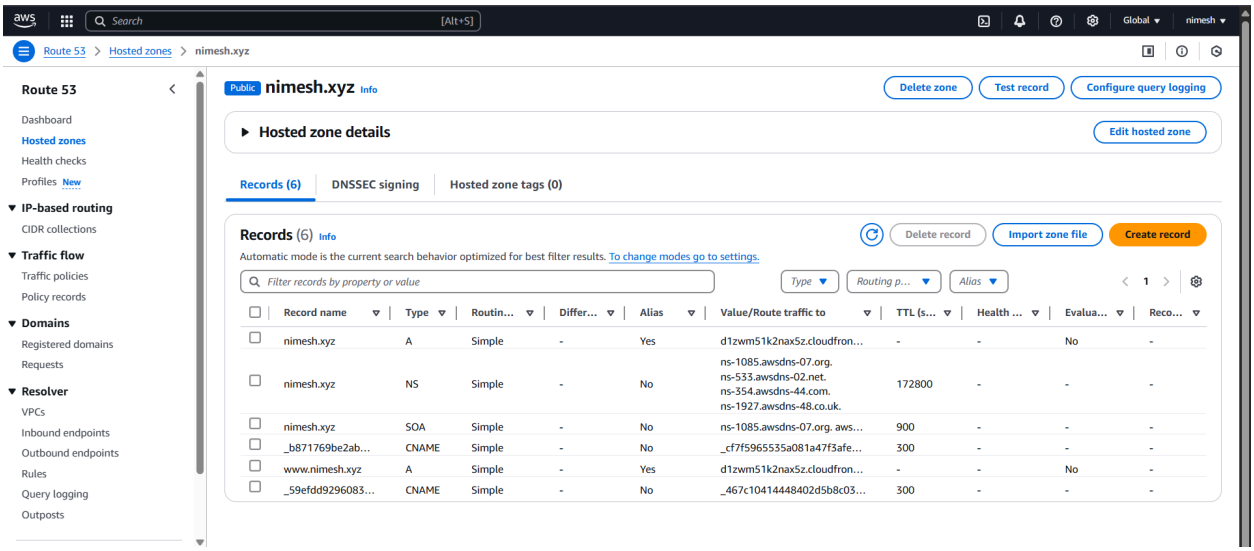
- Create → Origin Domain Presentation-Tier-ALB → HTTP Redirect HTTP to HTTPS
- Do not Enable WAF → Select needed regions Add Alternate Domain name → “nimesh.xyz” “www.nimesh.xyz”
- Attach SSL Certificate
  Create Distribution



Step 13: Create DNS Records from CloudFront in Route53

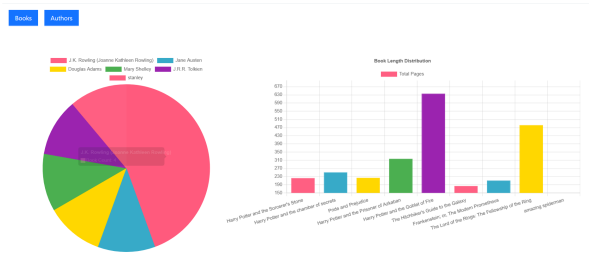
Go to Route53

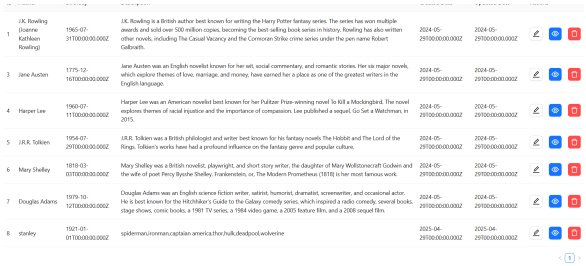
- Open Our Hosted Zone → Click on Add Record → Alias → To Cloud Front Distribution
  Select your distribution. Do the Same for www.devopsdost.com also.



Step 14: Test the Application

Access the Domain, You can see the Application





Book Details

ID
10

TITLE
amazing spiderman

RELEASE DATE
1952-01-12T00:00:00.000Z

BOOK DESCRIPTION
marvel

TOTAL PAGES
115

CREATED DATE
2025-04-29T00:00:00.000Z

UPDATED DATE
2025-04-29T00:00:00.000Z

8

stanley

BIRTHDAY
1921-01-01T00:00:00.000Z

AUTHOR DESCRIPTION
spiderman,ironman,captaian america,thor,hulk,deadpool,wolverine

AUTHOR
stanley