Deploy a 3 Tier Architecture On AWS using RDS Aurora Read/Write replica

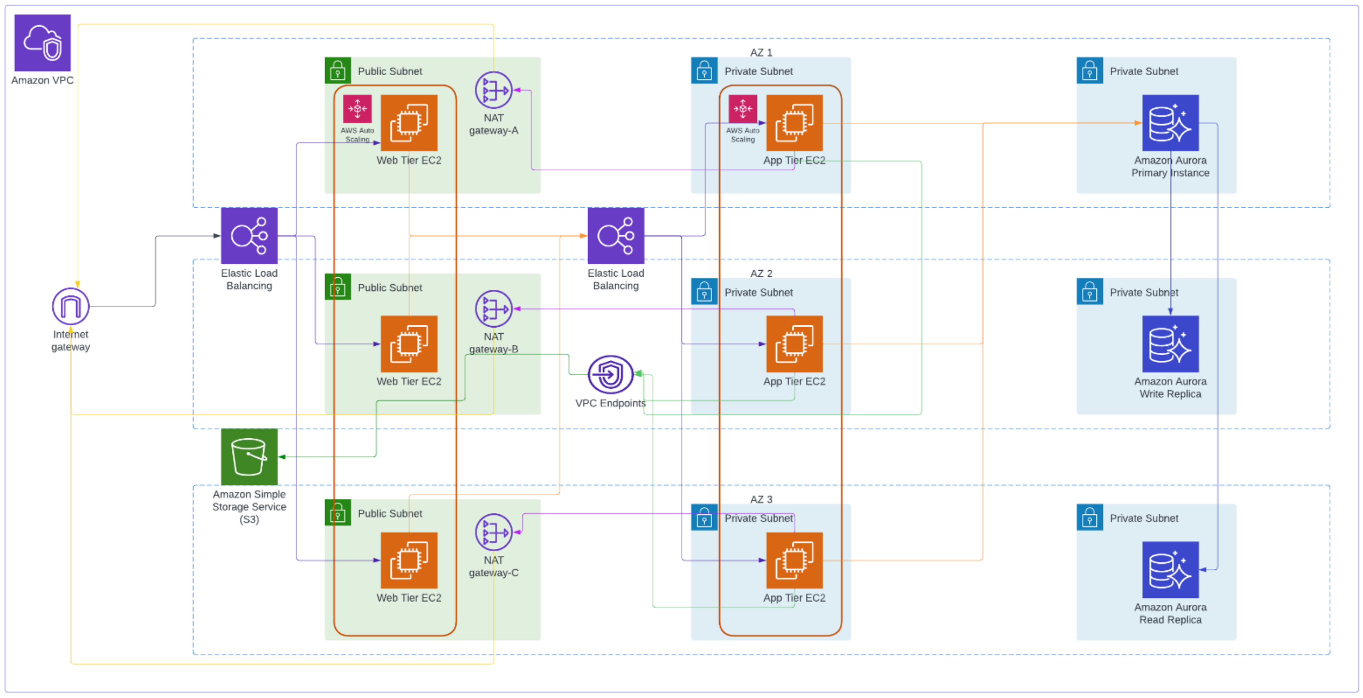
Objectives:

- Highly scalable, highly available, & fault-tolerant.
- Using custom VPC instead of default ones.
- Secure & apply best practices of IAM & security.

Summary:

In this project, we are going to be building a 3 Tier Web Application. In this project, the goal is to create a highly resilient website which can quickly auto-scale with respect to the incoming traffic & Use the best security practices for Access control to various resources. We will be using IAM roles for EC2 to use S3 & SSM. Then we will be creating various network resources in one click in the VPC instance & create 5 Security groups for External Facing Load balancer, Web tier EC2, Internal Facing Load balancer, App tier Private instances & DB. Aurora RDS DB will be created with Multi-AZ read/write replica module for high scalability & availability of the Database.

Architecture:



Services Used:

1. IAM Role
2. S3
3. 3 NAT Gateways
4. 3 Elastic IPs
5. S3 Endpoint
6. 9 Subnets
7. IGW
8. 1 Public 6 Private Route Tables
9. RDS

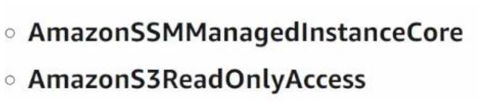
10. 5 SGs
11. EC2 (5-10)
12. ALB - 2
13. ASG - 2
14. TG - 2
15. Template -2
16. AMI - 2
17. Snapshots Auto Created

Steps:

Part 1: Setup

Step 1: Create a S3 bucket with all default settings. Use this S3 bucket to deploy code to the server.

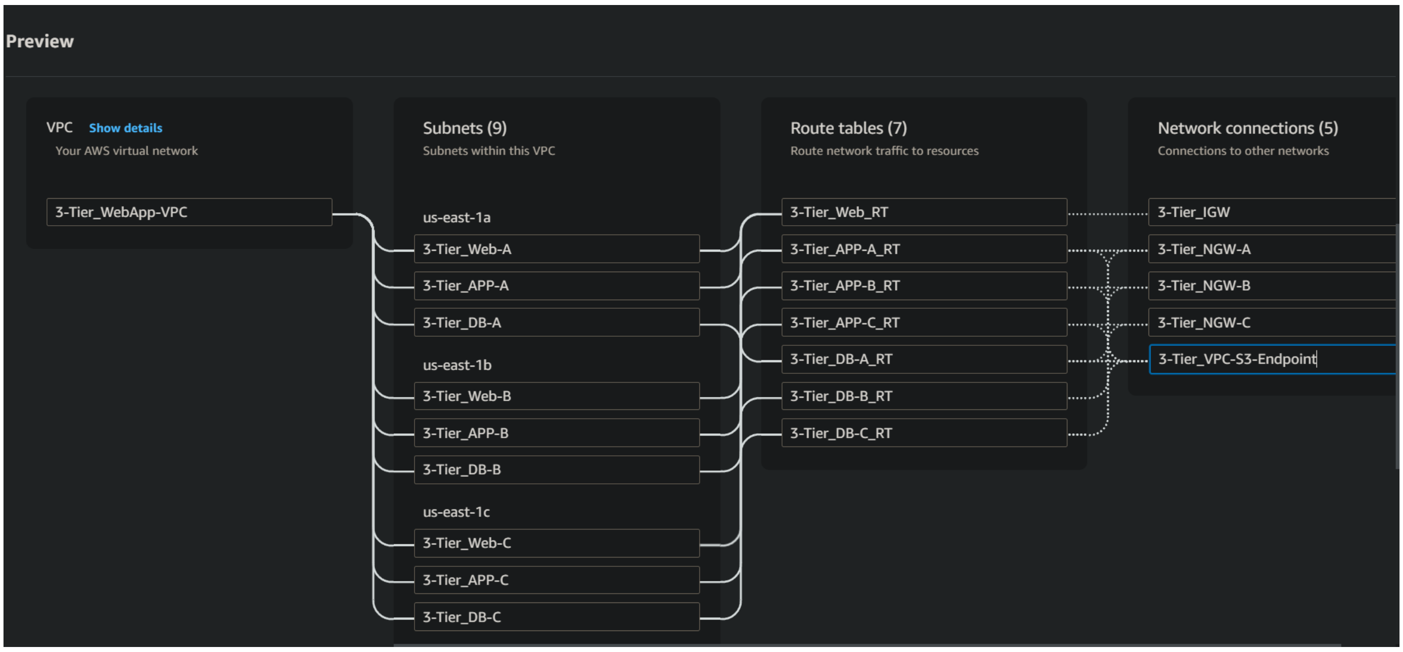
Step 2: Create IAM Role for EC2 instance. Use following permissions



- AmazonSSMManagedInstanceCore
- AmazonS3ReadOnlyAccess

Part 2: Networking & Security

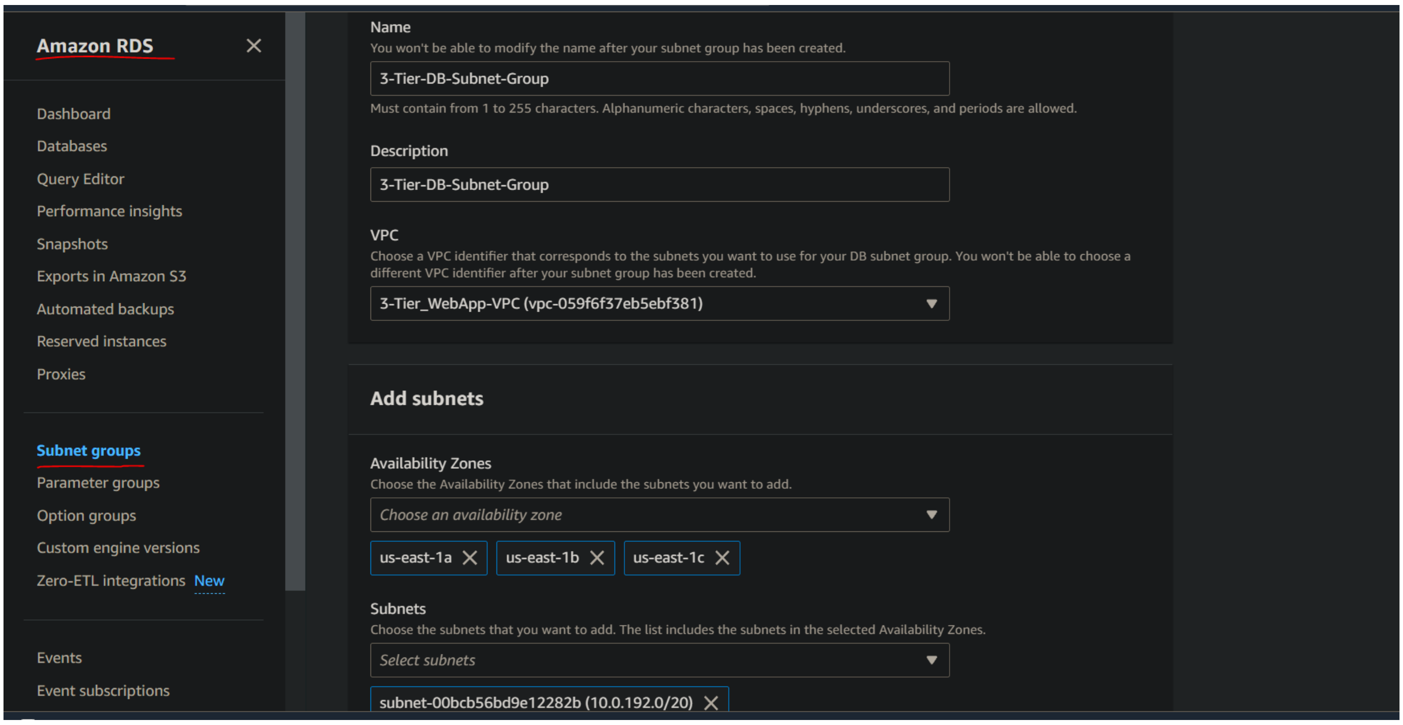
Step 1: Create following resources in VPC & more.



Step 2: Create 5 Security groups for External Facing Load balancer, For Web tier EC2, for Internal Facing Load balancer, For Private instances (use port 4000 for private instance SG) & for DB. Use SG of previously created as source for security group 2, 3, 4 & 5.

Part 3: Database Deployment

Step 1: Create DB subnet group



Name
You won't be able to modify the name after your subnet group has been created.
3-Tier-DB-Subnet-Group
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

Description
3-Tier-DB-Subnet-Group

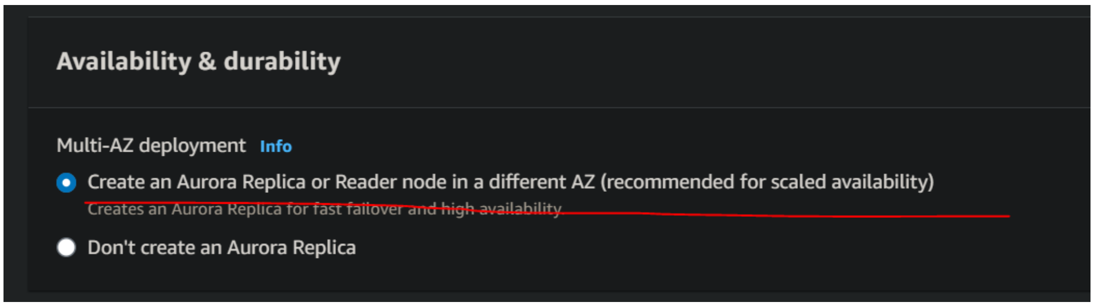
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
3-Tier_WebApp-VPC (vpc-059f6f37eb5ebf381)

Add subnets

Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a, us-east-1b, us-east-1c

Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-00bcb56bd9e12282b (10.0.192.0/20)

Step 2: Create DB (AuroraSQL). Use this option



Availability & durability

Multi-AZ deployment Info
[✓ Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)]
Creates an Aurora Replica for fast failover and high availability
Don't create an Aurora Replica

Part 4: App Tier Instance Deployment

Step 1: Create App tier EC2 instance. Use appropriate VPCs, subnets & SGs. Use IAM role created.

Connect to the instance & run following commands for creating mysql.

```
sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm

sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y

sudo dnf install mysql-community-server -y

sudo systemctl start mysqld
```

Step 2: Copy DB writer instance endpoint & use it to run “mysql -h YourEndpoint -u YourUsername -p”. Mysql will be open connected to that instance. Now create a database, table, & some values.

Step 3: Copy dbconfig file from my github repo – code>apptier>dbconfig . Fill the details & upload it to S3 bucket. Also copy app tier folder there.

Run the commands:

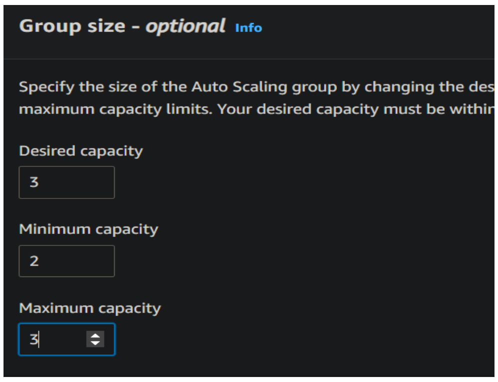
```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
source ~/.bashrc

nvm install 16
nvm use 16

npm install -g pm2

cd ~/
aws s3 cp s3://BUCKET_NAME/app-tier/ app-tier --recursive
```

Step 4: Internal Load Balancing and Auto Scaling

Step 1: Create image of App Server 1.

Step 2: Create TG. Then create ALB (both for App-Tier). Then create launch template using the image.

Step 3: Create ASG for App-Tier. Use these values.



Group size - *optional* Info

Specify the size of the Auto Scaling group by changing the des
maximum capacity limits. Your desired capacity must be withir

Desired capacity: 3

Minimum capacity: 2

Maximum capacity: 3

Step 5: Web Tier Instance Deployment

Step 1: Edit internal load balancer dns into nginx config file (code folder) & then deploy nginx file & web-tier folder into s3 bucket.

Step 2: Create Web server instance.

Connect to it & run following commands:

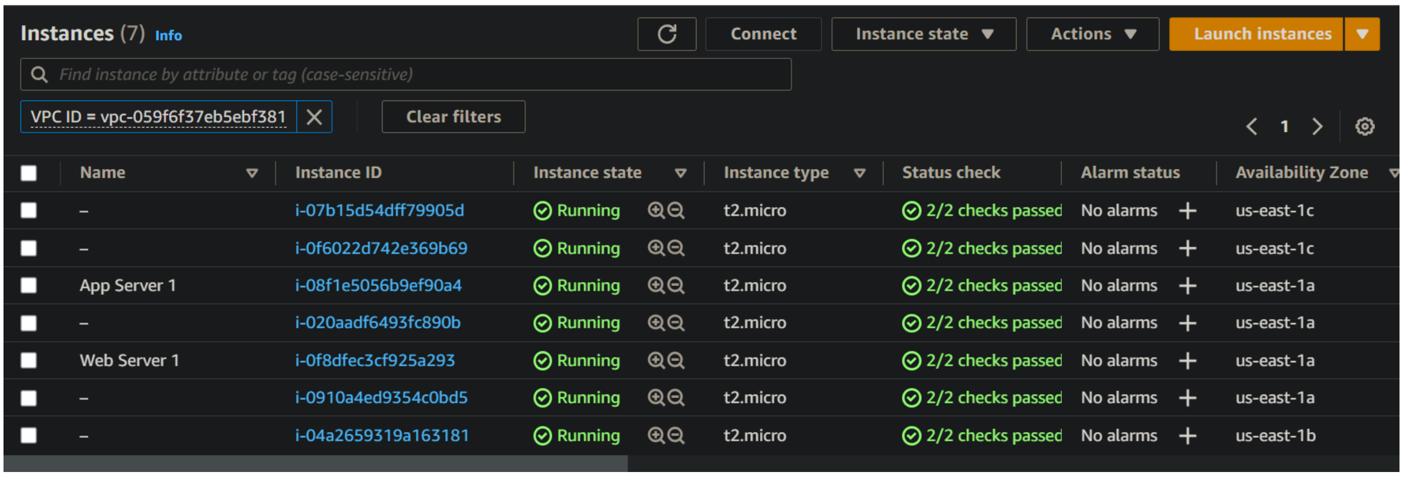
```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
source ~/.bashrc
```

```
nvm install 16

nvm use 16


cd ~/
aws s3 cp s3://3tier-codedeploy-bucket/web-tier/ web-tier --recursive


cd ~/web-tier

npm install

npm run build


sudo amazon-linux-extras install nginx1 -y


cd ~/
aws s3 cp s3://BUCKET_NAME/web-tier/ web-tier --recursive


sudo rm nginx.conf

sudo aws s3 cp s3://3tier-codedeploy-bucket/nginx.conf .


sudo service nginx restart


chmod -R 755 /home/ec2-user


sudo chkconfig nginx on
```

Step 3: Create Image of Web Server. Create TG, ALB, Launch template & ASG for Web Server.

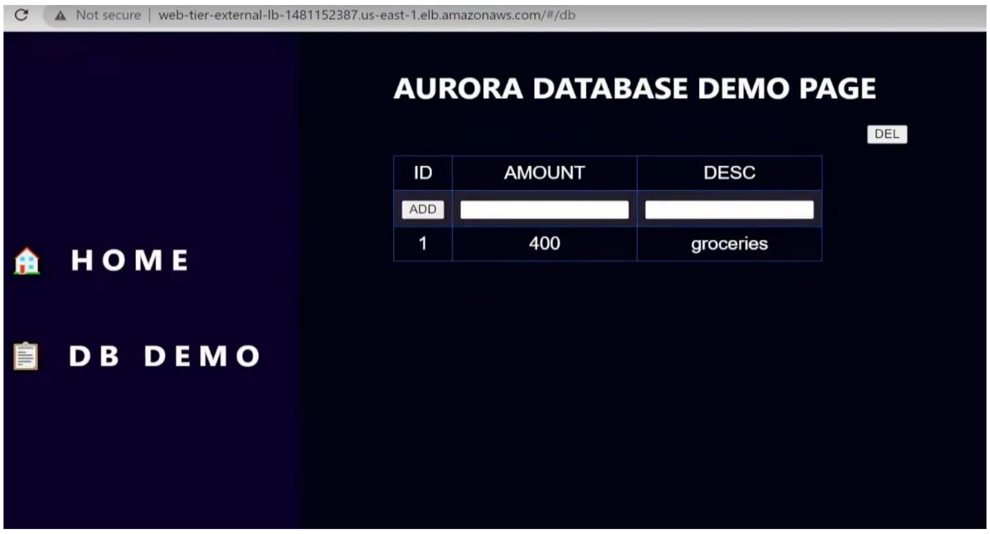
Step 4: Now use Public ALB DNS to use website.

Instances spun up as a result of all the ASGs at work


Instances (7) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
VPC ID = vpc-059f6f37eb5ebf381
Clear filters
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
–	i-07b15d54dff79905d	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
–	i-0f6022d742e369b69	Running	t2.micro	2/2 checks passed	No alarms	us-east-1c
App Server 1	i-08f1e5056b9ef90a4	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a
–	i-020aadf6493fc890b	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a
Web Server 1	i-0f8dfec3cf925a293	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a
–	i-0910a4ed9354c0bd5	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a
–	i-04a2659319a163181	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b

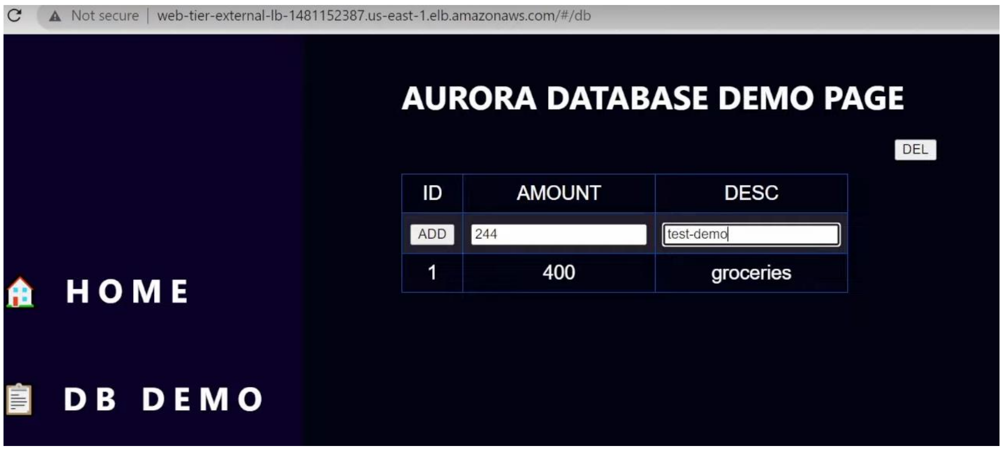
Outcome:


Not secure | web-tier-external-lb-1481152387.us-east-1.elb.amazonaws.com/#/db

AURORA DATABASE DEMO PAGE

ID	AMOUNT	DESC
ADD		
1	400	groceries

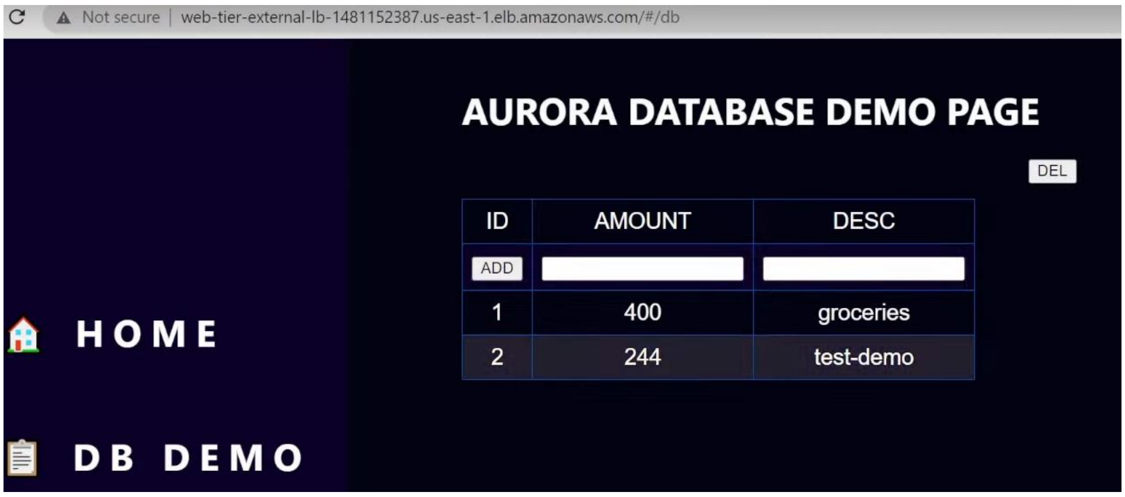
Now lets add some values


Not secure | web-tier-external-lb-1481152387.us-east-1.elb.amazonaws.com/#/db

AURORA DATABASE DEMO PAGE

ID	AMOUNT	DESC
ADD	244	test-demo
1	400	groceries

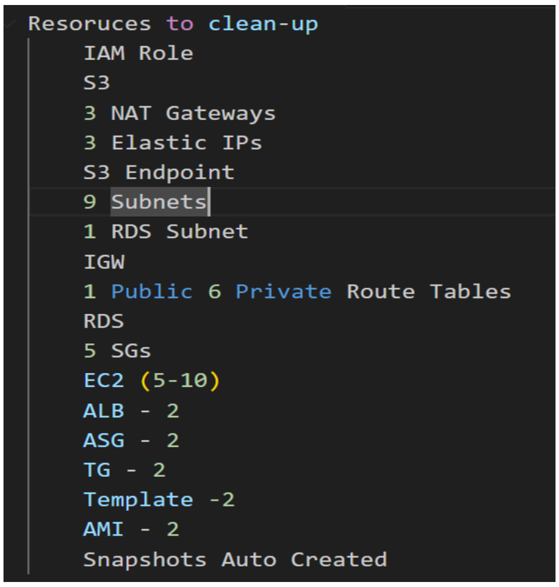
Values Added


Not secure | web-tier-external-lb-1481152387.us-east-1.elb.amazonaws.com/#/db

AURORA DATABASE DEMO PAGE

ID	AMOUNT	DESC
ADD		
1	400	groceries
2	244	test-demo

Resources to Cleanup:



```
Resoruces to clean-up
    IAM Role
    S3
    3 NAT Gateways
    3 Elastic IPs
    S3 Endpoint
    9 Subnets
    1 RDS Subnet
    IGW
    1 Public 6 Private Route Tables
    RDS
    5 SGs
    EC2 (5-10)
    ALB - 2
    ASG - 2
    TG - 2
    Template -2
    AMI - 2
    Snapshots Auto Created
```