

Project-1- Deployment of 3 Tier Applications Using AWS.

Submitted By – Arya Chaurasia

Date of Submission – 17/12/2025

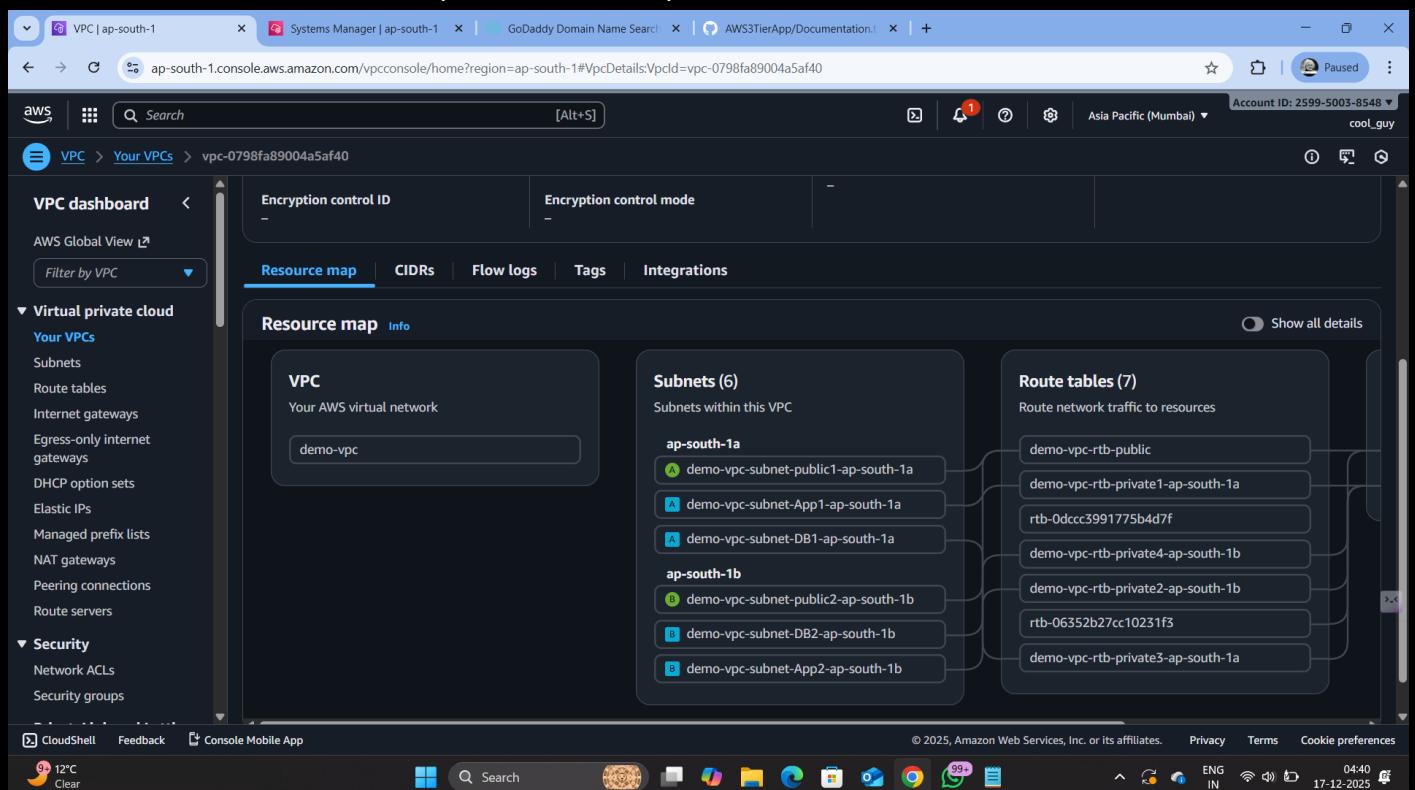
Project Overview

This project demonstrates a manually deployed 3-tier architecture on AWS for a web application where users can submit data, which is then stored in a backend database. The setup does not use DevOps tools (like Jenkins, Docker, or Kubernetes) and relies on AWS native services for hosting, networking, and storage.

GitHub Link- <https://github.com/Aryachaurasia23/AWS3TierApp.git>

Steps to Deploy the Website successfully

Step-1- First we created the VPC and we will select the option “VPC & more” which we will create the Subnets, Route tables, Network connections.



Step-2- Now we name our Subnets Public subnets – {Public-1, Public-2} , Private Subnets {App-1, App-2, DB-1, DB-2}

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
demo-vpc-subnet-App1-ap-south-1a	subnet-0669c42d07ed81cab	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.2.0/26
-	subnet-0f812df37e00916f2	Available	vpc-06b68ccce24cd3aa	Off	172.31.32.0/20
demo-vpc-subnet-public2-ap-south-1b	subnet-0fe9cde6f683d858b	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.0.64/26
demo-vpc-subnet-public1-ap-south-1a	subnet-0806f0794ab851da5	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.0.0/26
-	subnet-062c9c1026cff20c	Available	vpc-06b68ccce24cd3aa	Off	172.31.16.0/20
demo-vpc-subnet-DB2-ap-south-1b	subnet-0df7032c67af0ba6b	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.2.192/2
demo-vpc-subnet-App2-ap-south-1b	subnet-0a8e4d7aca90c1fec	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.2.64/26
demo-vpc-subnet-DB1-ap-south-1a	subnet-0ba31cb586985dcad	Available	vpc-0798fa89004a5af40 dem...	Off	192.168.2.128/2
-	subnet-0125367886cd5c062	Available	vpc-06b68ccce24cd3aa	Off	172.31.0.0/20

Step-3- RDS-SG has enabled IPv4 CIDR block.

IP version	Type	Protocol	Port range	Source	Description
IPv4	MySQL/Aurora	TCP	3306	192.168.0.0/22	-

Step-4- Only Web-ALB-SG has all traffic enabled rest all other security group has enabled IPv4 CIDR block.

sg-062f86b0027781d8d - Web-ALB-SG

IP version	Type	Protocol	Port range	Source	Description
IPv4	HTTP	TCP	80	0.0.0.0/0	-

Step-5- In Web-SG we have enabled IPv4 CIDR Block.

sg-06c9278c894062065 - Web-SG

IP version	Type	Protocol	Port range	Source	Description
IPv4	HTTP	TCP	80	192.168.0.0/22	-
6b	-	TCP	80	sg-062f86b0027781d8d...	-

Step-6- Now we made a S3 bucket to upload our application code here.

The screenshot shows the AWS S3 console with the '3tier-project' bucket selected. The 'Objects' tab is active, showing a single object named 'AWS3TierApp-main/'. The sidebar on the left provides navigation links for Amazon S3, Buckets, Access management and security, Storage management and insights, and AWS Marketplace for S3. The top right corner shows account information and a user icon.

Step-7- We created a role and attached AmazonEC2RoleforSSM.

The screenshot shows the AWS IAM console with the 'Demo-EC2-Role' role selected. The 'Permissions' tab is active, showing the 'AmazonEC2RoleforSSM' policy attached. The sidebar on the left provides navigation links for Identity and Access Management (IAM), Access management, and Access reports. The top right corner shows account information and a user icon.

Step-8- We created a RDS database named as database-1 with RDS-SG security group attached and in our private VPC (demo-vpc).

database-1 - Database Details | Systems Manager | ap-south-1 | GoDaddy Domain Name Search | AWS3TierApp/Documentation... | +

aws | Search [Alt+S] | Account ID: 2599-5003-8548 | Asia Pacific (Mumbai) | cool_guy

Aurora and RDS > Databases > database-1

database-1

Summary

DB identifier database-1	Status Available	Role Instance	Engine MySQL Community	Recommendations
CPU 3.28%	Class db.t4g.micro	Current activity 1 Connections	Region & AZ ap-south-1b	

Connectivity & security

Endpoint & port Endpoint: database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com Port: 3306	Networking Availability Zone: ap-south-1b VPC: demo-vpc (vpc-0798fa89004a5af40) Subnet group: db-snpg Subnets: subnet-0ba31cb586985dcad, subnet-0df7032c67af0ba6b	Security VPC security groups: RDS-SG (sg-0348b5d4700027145) (Active) Publicly accessible: No Certificate authority: rds-ca-rsa2048-g1
--	--	---

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 12°C Clear 0445 ENG IN 17-12-2025

Step-9- This is the configuration pf the database-1

database-1 - Database Details | Systems Manager | ap-south-1 | GoDaddy Domain Name Search | AWS3TierApp/Documentation... | +

aws | Search [Alt+S] | Account ID: 2599-5003-8548 | Asia Pacific (Mumbai) | cool_guy

Aurora and RDS > Databases > database-1

database-1

CPU 3.19%	Class db.t4g.micro	Current activity 1 Connections	Region & AZ ap-south-1b
--------------	-----------------------	-----------------------------------	----------------------------

Configuration

Endpoint & port Endpoint: database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com Port: 3306	Networking Availability Zone: ap-south-1b VPC: demo-vpc (vpc-0798fa89004a5af40) Subnet group: db-snpg Subnets: subnet-0ba31cb586985dcad, subnet-0df7032c67af0ba6b	Security VPC security groups: RDS-SG (sg-0348b5d4700027145) (Active) Publicly accessible: No Certificate authority: rds-ca-rsa2048-g1 Certificate authority date: May 20, 2061, 00:10 (UTC+05:30) DB instance certificate expiration date: December 17, 2026, 04:04 (UTC+05:30)
--	--	---

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 12°C Clear 0446 ENG IN 17-12-2025

Step-10- Now we created an EC2 Instance (AppTierInstance) and connected with Session manager

The screenshot shows the AWS EC2 Instances page. There are two instances listed:

- AppTierInstance**: Instance ID i-0cc4c2d1907cbdfa2, Running, t2.micro, 2/2 checks passed, View alarms +, ap-south-1a, Public IPv4 -.
- demo**: Instance ID i-00c4eb7f8f570f439, Terminated, t2.micro, -.

The details for the AppTierInstance are expanded:

- Instance ID**: i-0cc4c2d1907cbdfa2
- IPv6 address**: -
- Hostname type**: IP name: ip-192-168-2-19.ap-south-1.compute.internal
- Answer private resource DNS name**: -
- Auto-assigned IP address**: -
- Public IPv4 address**: -
- Instance state**: Running
- Private IP DNS name (IPv4 only)**: ip-192-168-2-19.ap-south-1.compute.internal
- Instance type**: t2.micro
- VPC ID**: vpc-0798fa89004a5af40 (demo-vpc)
- Private IPv4 addresses**: 192.168.2.19
- Public DNS**: -
- Elastic IP addresses**: -
- AWS Compute Optimizer finding**: Opt-in to AWS Compute Optimizer for recommendations.

Step-11- Here we clicked on the connect.

The screenshot shows the AWS EC2 Instances page. The interface is identical to the previous one, but the 'Connect' button is highlighted in red at the top of the instance details panel for the AppTierInstance.

The details for the AppTierInstance are expanded:

- Auto-assigned IP address**: -
- IAM Role**: Demo-EC2-Role
- IMDSv2**: Required
- Operator**: -
- VPC ID**: vpc-0798fa89004a5af40 (demo-vpc)
- Subnet ID**: subnet-0669c42d07ed81cab (demo-vpc-subnet-App1-ap-south-1a)
- Instance ARN**: arn:aws:ec2:ap-south-1:259950038548:instance/i-0cc4c2d1907cbdfa2
- AWS Compute Optimizer finding**: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
- Auto Scaling Group name**: -
- Managed**: false

Step-12- Now we will go to the Session Manager and click on Connect.

The screenshot shows the AWS Session Manager usage page for an EC2 instance. The 'Session Manager' tab is selected. Below it, a section titled 'Session Manager usage:' lists the following benefits:

- Connect to your instance without SSH keys, a bastion host, or opening any inbound ports.
- Sessions are secured using an AWS Key Management Service key.
- You can log session commands and details in an Amazon S3 bucket or CloudWatch Logs log group.
- Configure sessions on the Session Manager [Preferences](#) page.

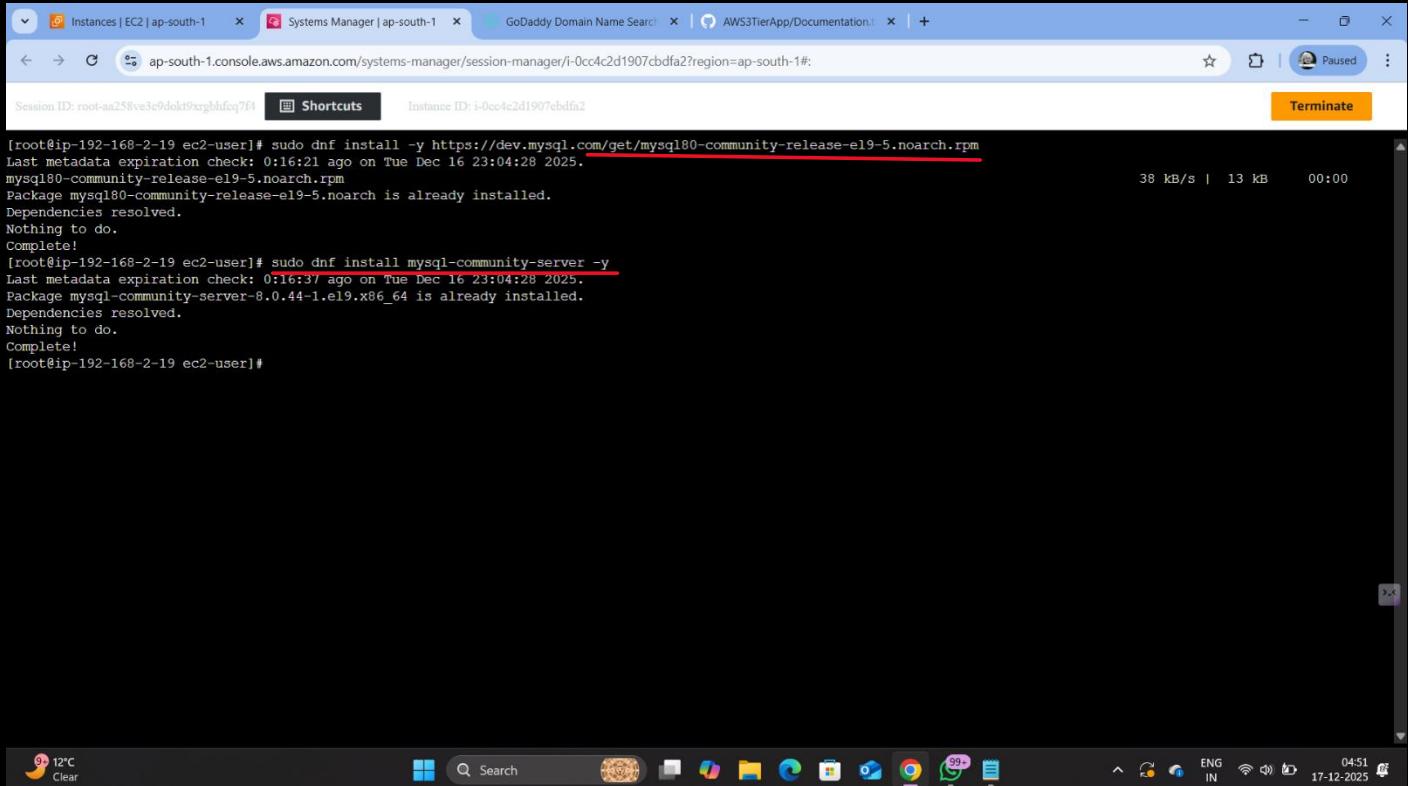
At the bottom right of the page, there are 'Cancel' and 'Connect' buttons. The 'Connect' button is highlighted with a yellow box.

Step-13- We will go to the home directory and check whether the internet is accessible or not.

The screenshot shows the AWS Session Manager terminal window. The terminal prompt is 'sh-5.2\$'. The user has run the command 'ping 8.8.8.8' and is viewing the output. The output shows several ICMP echo requests being sent to the Google DNS server at 8.8.8.8, with round-trip times ranging from 2.44 ms to 2.74 ms. The terminal also displays the user's path: '/home/ec2-user'.

```
sh-5.2$ sudo su
[root@ip-192-168-2-19 bin]# cd ..
[root@ip-192-168-2-19 usr]# cd /home/ec2-user/
[root@ip-192-168-2-19 ec2-user]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=2.74 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=2.48 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=2.44 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.436/2.552/2.742/0.135 ms
[root@ip-192-168-2-19 ec2-user]#
```

Step-14- Now we installed MySql in the server.



A screenshot of a terminal window titled "Session ID: root-aa258ve3c9dokt9xrgbhfcq744". The terminal shows the following command being run:

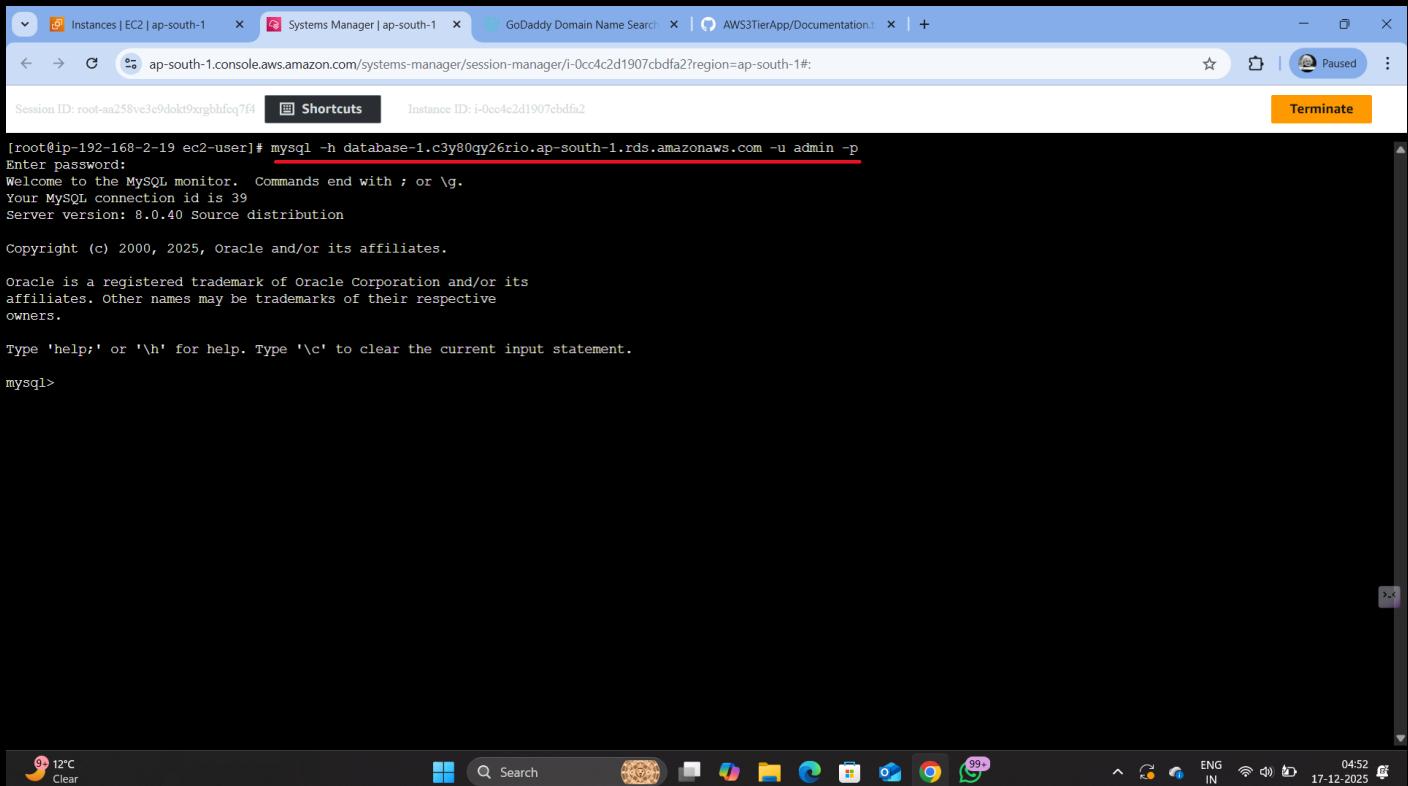
```
[root@ip-192-168-2-19 ec2-user]# sudo dnf install -y https://dev.mysql.com/get/mysql80-community-release-el9-5.noarch.rpm
```

The output indicates that the package "mysql80-community-release-el9-5.noarch" is already installed, so nothing is done. The session ends with:

```
[root@ip-192-168-2-19 ec2-user]#
```

The terminal window has a blue header bar with tabs for "Instances | EC2 | ap-south-1", "Systems Manager | ap-south-1", "GoDaddy Domain Name Search", and "AWS3TierApp/Documentation". A yellow "Terminate" button is visible in the top right corner. The bottom of the screen shows a Windows taskbar with various icons and system status indicators.

Step-15- Now we will connect to the database and perform basic configurations.



A screenshot of a terminal window titled "Session ID: root-aa258ve3c9dokt9xrgbhfcq744". The terminal shows the following command being run:

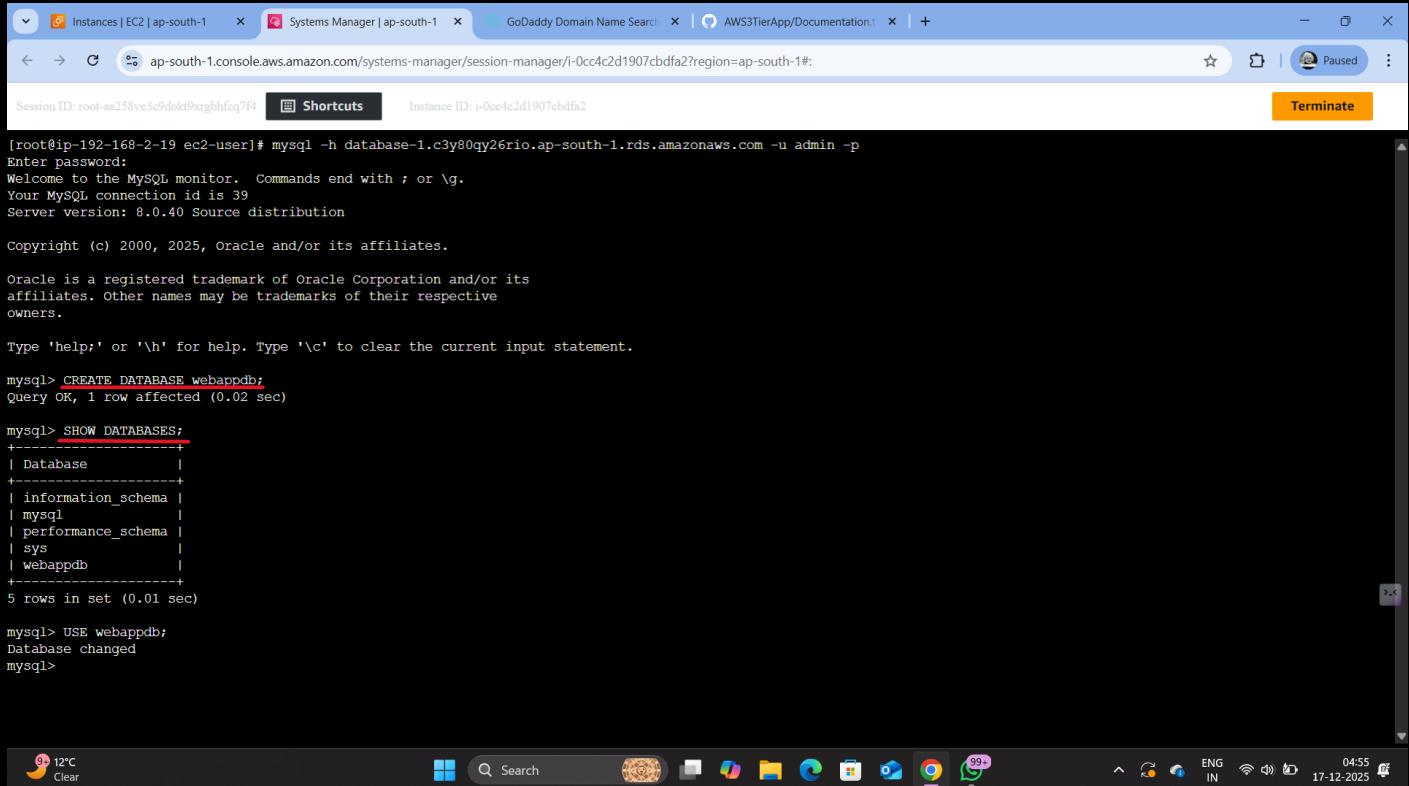
```
[root@ip-192-168-2-19 ec2-user]# mysql -h database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com -u admin -p
```

The output shows the MySQL monitor prompt, indicating a successful connection to the RDS instance. The session ends with:

```
mysql>
```

The terminal window has a blue header bar with tabs for "Instances | EC2 | ap-south-1", "Systems Manager | ap-south-1", "GoDaddy Domain Name Search", and "AWS3TierApp/Documentation". A yellow "Terminate" button is visible in the top right corner. The bottom of the screen shows a Windows taskbar with various icons and system status indicators.

Step-16- Now we will create a database named webappdb.



```
[root@ip-192-168-2-19 ec2-user]# mysql -h database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

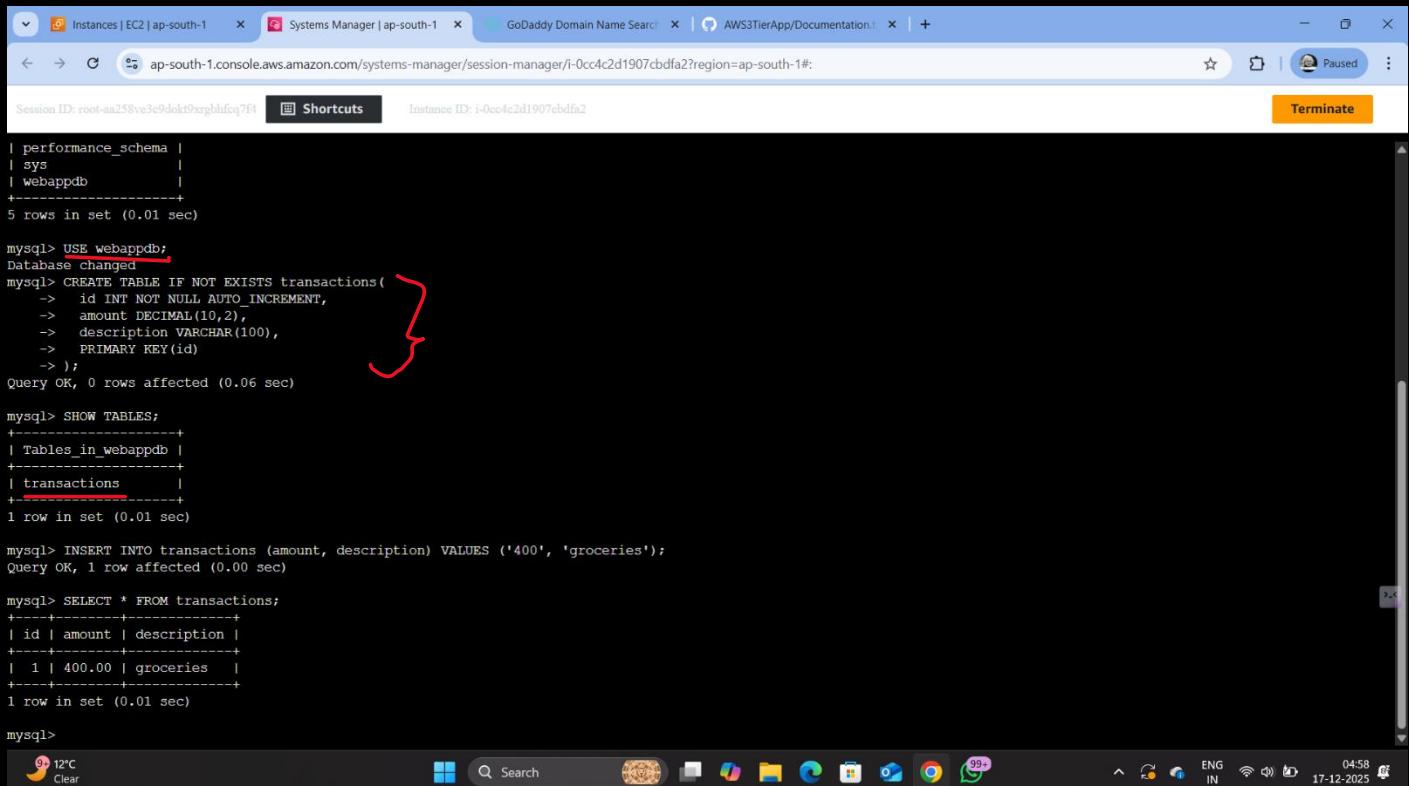
mysql> CREATE DATABASE webappdb;
Query OK, 1 row affected (0.02 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| webappdb |
+-----+
5 rows in set (0.01 sec)

mysql> USE webappdb;
Database changed
mysql>
```

12°C Clear 0455 17-12-2025

Step-17- Here we are creating a table named as transaction.



```
| performance_schema |
| sys |
| webappdb |
+-----+
5 rows in set (0.01 sec)

mysql> USE webappdb;
Database changed
mysql> CREATE TABLE IF NOT EXISTS transactions (
    >     id INT NOT NULL AUTO_INCREMENT,
    >     amount DECIMAL(10,2),
    >     description VARCHAR(100),
    >     PRIMARY KEY(id)
    > );
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW TABLES;
+-----+
| Tables_in_webappdb |
+-----+
| transactions |
+-----+
1 row in set (0.01 sec)

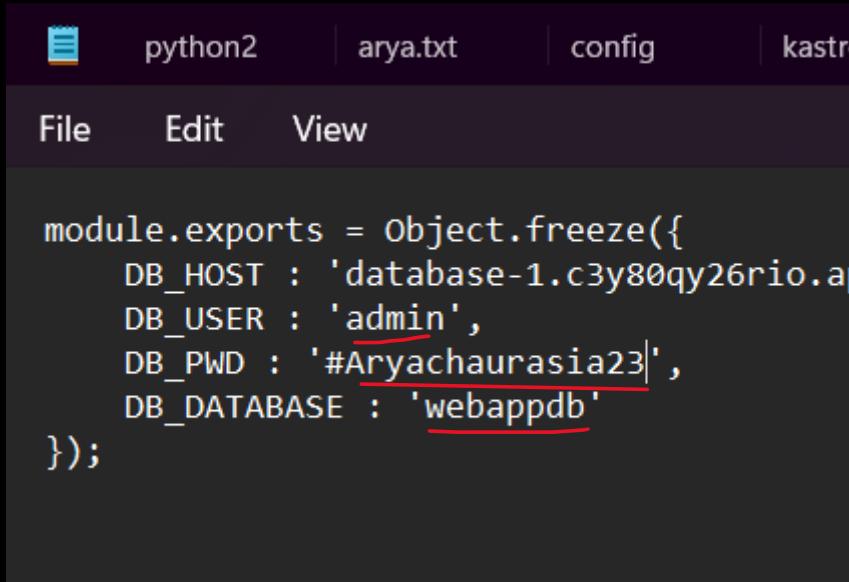
mysql> INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM transactions;
+----+----+----+
| id | amount | description |
+----+----+----+
| 1  | 400.00 | groceries   |
+----+----+----+
1 row in set (0.01 sec)

mysql>
```

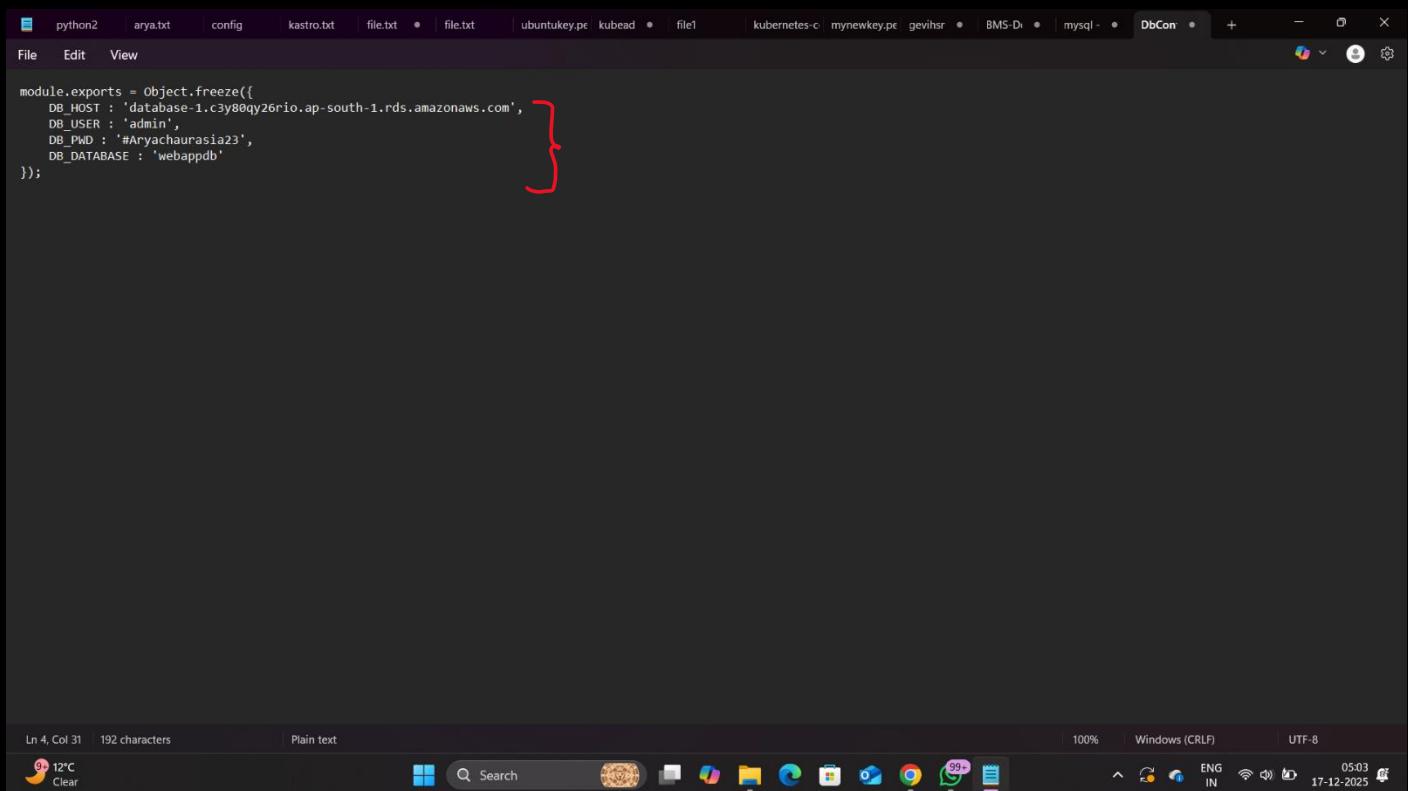
12°C Clear 0458 17-12-2025

Step-18- Update Application Configuration to with DB information



```
module.exports = Object.freeze({
  DB_HOST : 'database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com',
  DB_USER : 'admin',
  DB_PWD : '#Aryachaurasia23',
  DB_DATABASE : 'webappdb'
});
```

Step-19- Update DbConfig.js file with DB information and we will save it



```
module.exports = Object.freeze({
  DB_HOST : 'database-1.c3y80qy26rio.ap-south-1.rds.amazonaws.com',
  DB_USER : 'admin',
  DB_PWD : '#Aryachaurasia23',
  DB_DATABASE : 'webappdb'
});
```

Step-20- Now we will upload the updated file in S3 bucket and the older one will be replaced by the updated one.

The screenshot shows the AWS S3 'Upload objects' interface. In the 'Files and folders' section, a single file 'DbConfig.js' is listed with a size of 198.0 B and a type of text/javascript. Below this, the 'Destination' section shows the target path as 's3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/app-tier/'. Under 'Properties', it indicates 'Grant public access and access to other AWS accounts.' At the bottom right, there are 'Cancel' and 'Upload' buttons, with 'Upload' being highlighted with a red oval.

Step-21- Install and Configure Node.js

The screenshot shows a terminal session on an EC2 instance. The user runs a command to download and install Node.js using curl:

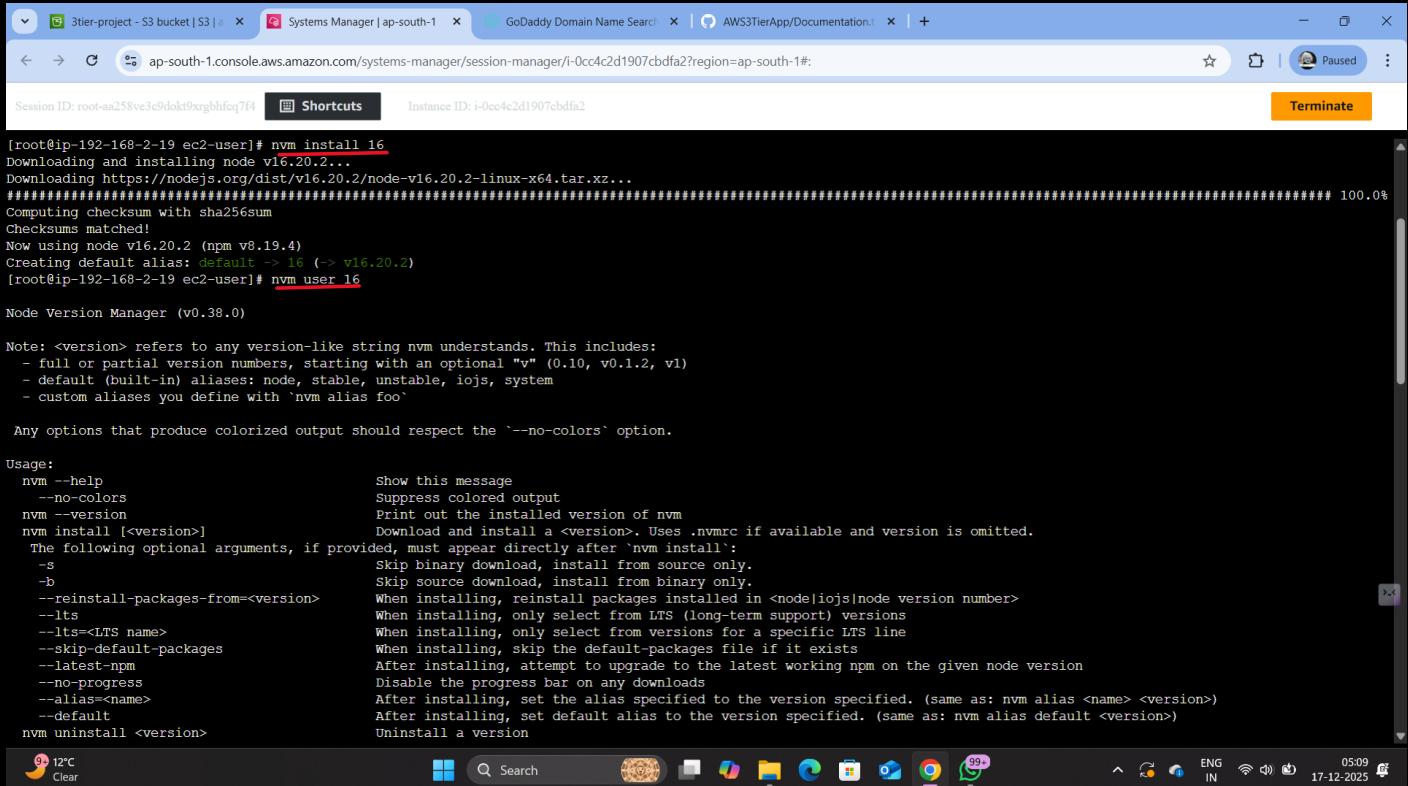
```
[root@ip-192-168-2-19 ec2-user]# curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload Total Spent   Left Speed
100 14926  100 14926    0      0  54191  0 --:--:-- --:--:-- 54276
-> Downloading nvm as script to '/root/.nvm'.
```

After the download, the user adds the nvm source to their .bashrc and reloads it:

```
=> Appending nvm source string to /root/.bashrc
=> Appending bash_completion source string to /root/.bashrc
=> Close and reopēn your terminal to start using nvm or run the following to use it now:
```

```
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \."$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \."$NVM_DIR/bash_completion" # This loads nvm bash_completion
[root@ip-192-168-2-19 ec2-user]# source ~/.bashrc
[root@ip-192-168-2-19 ec2-user]#
```

Step-22- nvm use 16 (You will see 'Now using node v16.20.2)



```
[root@ip-192-168-2-19 ec2-user]# nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (<-> v16.20.2)
[root@ip-192-168-2-19 ec2-user]# nvm user 16

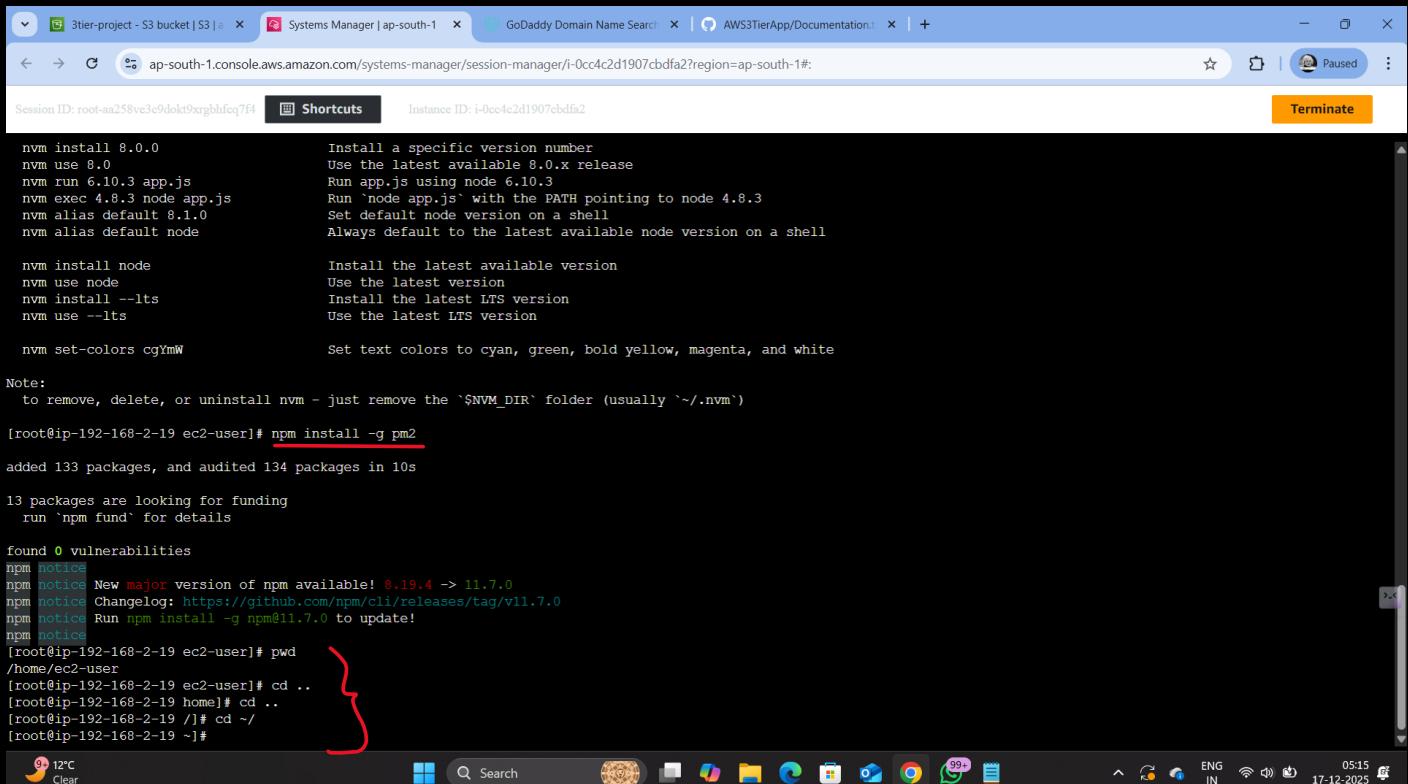
Node Version Manager (v0.38.0)

Note: <version> refers to any version-like string nvm understands. This includes:
- full or partial version numbers, starting with an optional "v" (0.10, v0.1.2, v1)
- default (built-in) aliases: node, stable, unstable, iojs, system
- custom aliases you define with 'nvm alias foo'

Any options that produce colorized output should respect the '--no-colors' option.

Usage:
  nvm --help                      Show this message
  --no-colors                     Suppress colored output
  nvm --version                   Print out the installed version of nvm
  nvm install [<version>]         Download and install a <version>. Uses .nvmrc if available and version is omitted.
    The following optional arguments, if provided, must appear directly after 'nvm install':
    -s                           Skip binary download, install from source only.
    -b                           Skip source download, install from binary only.
    --reinstall-packages-from=<version> When installing, reinstall packages installed in <node|iojs|node version number>
    --lts                         When installing, only select from LTS (long-term support) versions
    --lts=<LTS name>             When installing, only select from versions for a specific LTS line
    --skip-default-packages       When installing, skip the default-packages file if it exists
    --latest-npm                  After installing, attempt to upgrade to the latest working npm on the given node version
    --no-progress                 Disable the progress bar on any downloads
    --alias=<name>                After installing, set the alias specified to the version specified. (same as: nvm alias <name> <version>)
    --default                      After installing, set default alias to the version specified. (same as: nvm alias default <version>)
  nvm uninstall <version>        Uninstall a version
```

Step-23- To run node as a service, we will install pm2



```
nvm install 8.0.0
nvm use 8.0
Install a specific version number
Use the latest available 8.0.x release
nvm run 6.10.3 app.js
Run app.js using node 6.10.3
nvm exec 4.8.3 node app.js
Run `node app.js` with the PATH pointing to node 4.8.3
nvm alias default 8.1.0
Set default node version on a shell
nvm alias default node
Always default to the latest available node version on a shell

nvm install node
Install the latest available version
nvm use node
Use the latest version
nvm install --lts
Install the latest LTS version
nvm use --lts
Use the latest LTS version

nvm set-colors cgYmW
Set text colors to cyan, green, bold yellow, magenta, and white

Note:
  to remove, delete, or uninstall nvm - just remove the '$NVM_DIR' folder (usually '~/.nvm')

[root@ip-192-168-2-19 ec2-user]# npm install -g pm2
added 133 packages, and audited 134 packages in 10s

13 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
npm notice New major version of npm available! 8.19.4 -> 11.7.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.7.0
npm notice Run npm install -g npm@11.7.0 to update!
npm notices
[root@ip-192-168-2-19 ec2-user]# pwd
/home/ec2-user
{ [root@ip-192-168-2-19 ec2-user]# cd ..
[root@ip-192-168-2-19 home]# cd ..
[root@ip-192-168-2-19 /]# cd ~
[root@ip-192-168-2-19 ~]# }
```

Step-24- Download application code from S3 and start the application

Step-25- Here we can see that the application is downloading.

Step-26- pm2 start index.js (You will see the status as 'online')

```
Session ID: root-aa258ve5c9dokt9xrgbhfcq7f4 | Shortcuts | Instance ID: i-0cc4c2d1907cbdfa2?region=ap-south-1# | Terminate
```

Runtime Edition

PM2 is a Production Process Manager for Node.js applications with a built-in Load Balancer.

Start and Daemonize any application:
\$ pm2 start app.js

Load Balance 4 instances of api.js:
\$ pm2 start api.js -i 4

Monitor in production:
\$ pm2 monitor

Make pm2 auto-boot at server restart:
\$ pm2 startup

To go further checkout:
http://pm2.io/

```
[PM2] Spawning PM2 daemon with pm2_home=/root/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /root/app-tier/index.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	namespace	version	mode	pid	uptime	status	cpu	mem	user	watching	
0	index	default	1.0.0	fork	28577	0s	0	online	0%	25.8mb	root	disabled

```
[root@ip-192-168-2-19 app-tier]#
```

12°C Clear Search ENG IN 0518 17-12-2025

Step-27- Now we will create the target group.

Step 1 Create target group | EC2 | Systems Manager | ap-south-1 | GoDaddy Domain Name Search | AWS3TierApp/Documentation | +

EC2 > Target groups > Create target group

Target group name
Name must be unique per Region per AWS account.
App-Internal-TG
Accepts: a-z, A-Z, 0-9, and hyphen (-). Can't begin or end with hyphen. 1-32 total characters; Count: 15/32

Protocol
Protocol for communication between the load balancer and targets.
HTTP
Port
Port number where targets receive traffic. Can be overridden for individual targets during registration.
4000
1-65535

IP address type
Only targets with the indicated IP address type can be registered to this target group.
 IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
 IPv6
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.
vpc-06b68cec2e4cd3aa
(default)
Create VPC

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

12°C Clear Search ENG IN 0522 17-12-2025

Step-28- We will click include as pending below and click create.

The screenshot shows the 'Register targets - recommended' step of the EC2 target group creation wizard. A single instance, 'i-0cc4c2d1907cbdfa2' named 'AppTierInstance', is listed as 'Running' in the 'Security groups' column. The 'Ports for the selected instances' field contains '4000'. The 'Include as pending below' button is circled in red.

Step-29- Here we will create the Load-Balancer (App-internal-LB)

The screenshot shows the 'Basic configuration' step of the Application Load Balancer creation wizard. The 'Load balancer name' is set to 'App-internal-LB'. The 'Scheme' dropdown is set to 'Internal', which is highlighted with a blue border. The 'Load balancer IP address type' dropdown is set to 'IPv4', which is also highlighted with a blue border. The 'Network mapping' section at the bottom is partially visible.

Step-30- Here we will select 2 subnets (APP1 & APP2) and Security group (Internal_ALB_SG).

The screenshot shows the 'Create Application Load Balancer' wizard in the AWS CloudFormation console. In the 'Availability Zones and subnets' section, two subnets are selected: 'subnet-0669c42d07ed81cab' under 'ap-south-1a (aps1-az1)' and 'subnet-0a8e4d7aca90cf1fec' under 'ap-south-1b (aps1-az3)'. In the 'Security groups' section, the security group 'Internal_ALB_SG' (sg-061d67de40b92386a) is selected. A red arrow points to the 'Internal_ALB_SG' entry in the dropdown menu.

Step-31- Open nginx.conf file and in the end of the file you will see something like below; #proxy for internal lb , REPLACE-IT-WITH-INTERNAL-LB-DNS

```

File Edit View
arya.txt config kastro.txt file.txt ubuntukey.p kubeair file1 kubernetes-i mynewkey.p gevhs BMS-C mysql DbCor nginx
sendfile on;
tcp_nopush on;
tcp_nodelay on;
keepalive_timeout 65;
types_hash_max_size 4096;

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

include /etc/nginx/conf.d/*.conf;

server {
    listen 80;
    listen [::]:80;
    server_name _;

    # HTTP to HTTPS redirection
    if ($http_x_forwarded_proto = 'http') {
        return 301 https://$host$request_uri;
    }

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

    #react app and front end files
    location / {
        root /home/ec2-user/web-tier/build;
        index index.html index.htm;
        try_files $uri /index.html;
    }

    #proxy for internal lb
    location /api/ {
        proxy_pass http://internal-App-internal-LB-1184260850.ap-south-1.elb.amazonaws.com:80/;
    }
}

```

Ln 56, Col 95 1,572 characters Plain text 100% Windows (CRLF) UTF-8

Step-32- Now upload the updated file to the S3 bucket.

The screenshot shows the AWS S3 console interface. In the top navigation bar, the URL is `ap-south-1.console.aws.amazon.com/s3/upload/3tier-project?region=ap-south-1&prefix=AWS3TierApp-main/AWS3TierApp-main/application-code/`. The main area displays a table titled "Files and folders (1 total, 1.6 KB)" containing one item: "nginx.conf". Below the table, there's a "Destination" section with a destination URL: `s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/`. At the bottom right of the page, there are "Upload" and "Cancel" buttons, with the "Upload" button highlighted by a red oval.

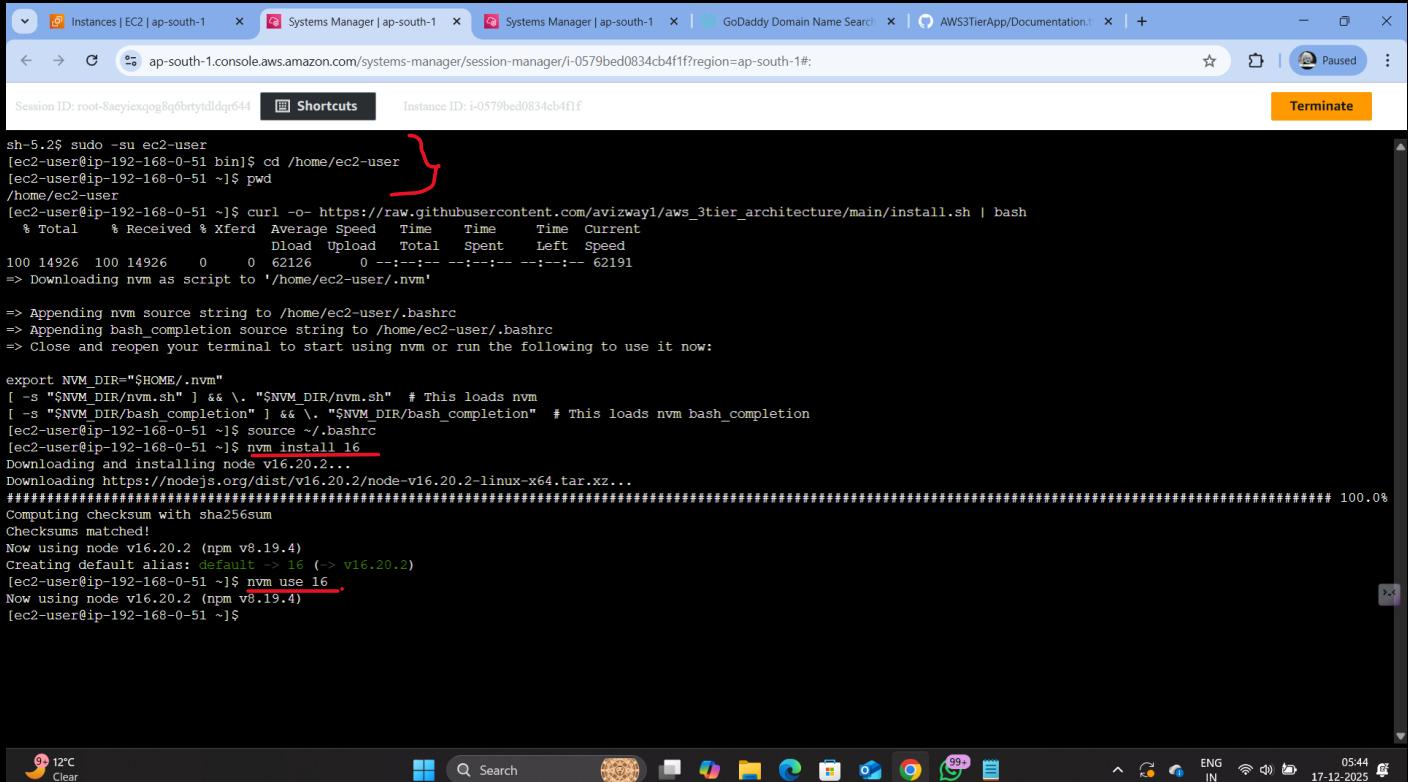
Step-33- Now we will create the another Instance which is Web-Tier-Instance

The screenshot shows the AWS EC2 Instances page. The left sidebar menu is expanded, showing sections like EC2, Instances, Images, and Elastic Block Store. The main content area displays a table titled "Instances (2) Info" with two entries:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
AppTierInstance	i-0cc4c2d1907cbdfa2	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	-
Web-Tier-Insta...	i-0579bed0834cb4f1f	Running	t2.micro	-	View alarms +	ap-south-1a	ec2-13-127-

A red arrow points to the "Web-Tier-Insta..." instance entry. The bottom of the screen shows the standard AWS navigation bar with CloudShell, Feedback, and Console Mobile App options.

Step-34- In this server we will install nvm 16

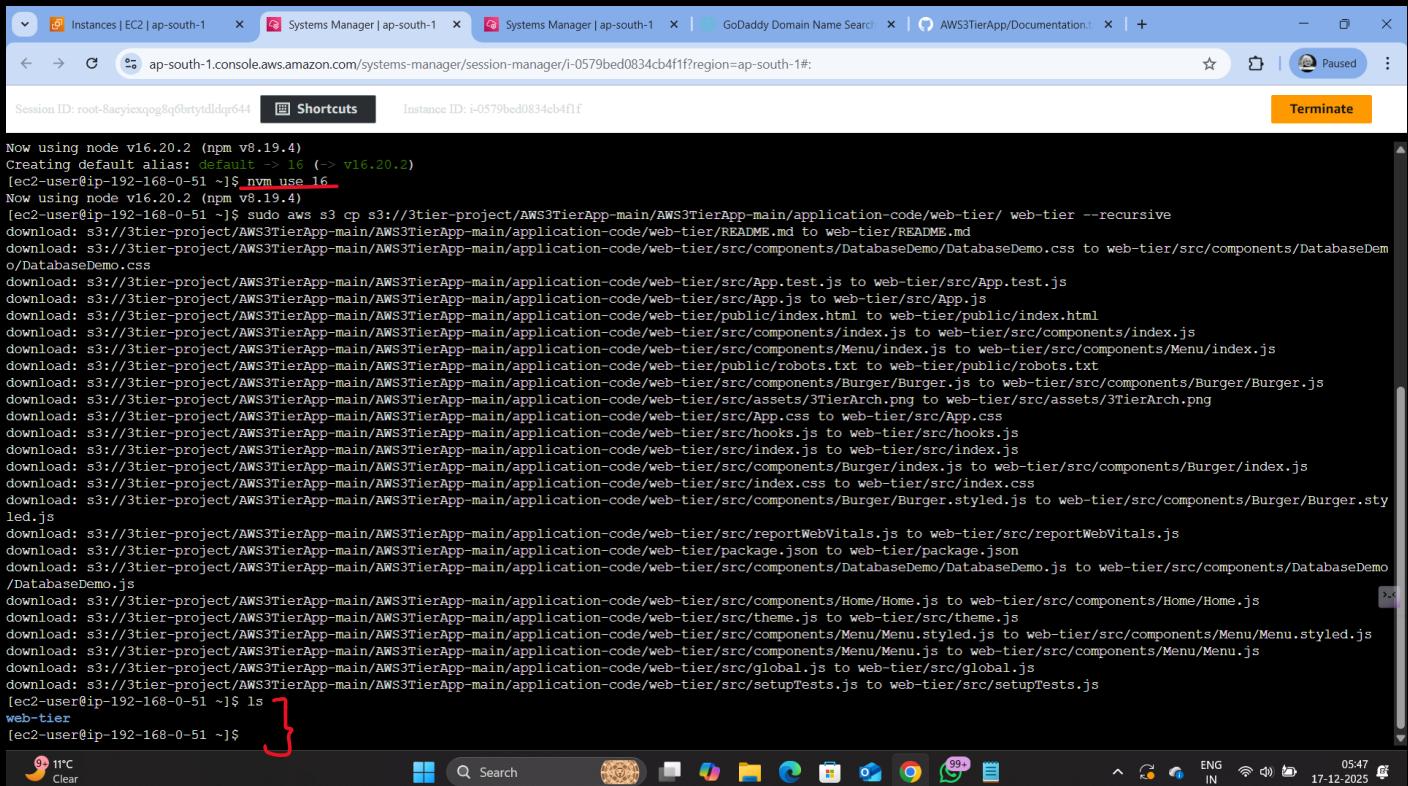


```
sh-5.2$ sudo -su ec2-user
[ec2-user@ip-192-168-0-51 bin]$ cd /home/ec2-user
[ec2-user@ip-192-168-0-51 ~]$ pwd
/home/ec2-user
[ec2-user@ip-192-168-0-51 ~]$ curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time     Time  Current
          Dload  Upload Total Spent   Left Speed
100 14926  100 14926    0      0  62126    0:--:-- --:--:-- 62191
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-192-168-0-51 ~]$ source ~/.bashrc
[ec2-user@ip-192-168-0-51 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloaded https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
creating default alias: default -> 16 (> v16.20.2)
[ec2-user@ip-192-168-0-51 ~]$ nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-192-168-0-51 ~]$
```

Step-35 - Now we will download the web-tier folder of our code.



```
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (> v16.20.2)
[ec2-user@ip-192-168-0-51 ~]$ nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-192-168-0-51 ~]$ sudo aws s3 cp s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/ web-tier --recursive
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/README.md to web-tier/README.md
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/DatabaseDemo/DatabaseDemo.css to web-tier/src/components/DatabaseDemo.css
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/App.test.js to web-tier/src/App.test.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/App.js to web-tier/src/App.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/public/index.html to web-tier/public/index.html
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/index.js to web-tier/src/components/index.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Menu/index.js to web-tier/src/components/Menu/index.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/public/robots.txt to web-tier/public/robots.txt
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Burger/Burger.js to web-tier/src/components/Burger/Burger.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/assets/3TierArch.png to web-tier/src/assets/3TierArch.png
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/App.css to web-tier/src/App.css
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/hooks.js to web-tier/src/hooks.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/index.js to web-tier/src/index.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Burger/index.js to web-tier/src/components/Burger/index.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/index.css to web-tier/src/index.css
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Burger/Burger.styled.js to web-tier/src/components/Burger/Burger.styled.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Home/Home.js to web-tier/src/components/Home/Home.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/theme.js to web-tier/src/theme.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Menu/Menu.stylesd.js to web-tier/src/components/Menu/Menu.stylesd.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/components/Menu/Menu.js to web-tier/src/components/Menu/Menu.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/global.js to web-tier/src/global.js
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/web-tier/src/setupTests.js to web-tier/src/setupTests.js
[ec2-user@ip-192-168-0-51 ~]$ ls
web-tier
[ec2-user@ip-192-168-0-51 ~]$
```

Step-36- Now we will go to the web-tier directory and install npm

```
[ec2-user@ip-192-168-0-51 ~]$ ls
[ec2-user@ip-192-168-0-51 ~]$ cd web-tier/
[ec2-user@ip-192-168-0-51 web-tier]$ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: '@testing-library/dom@10.4.1',
npm WARN   required: { node: '^18' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: 'expect@0.2.0',
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: 'pretty-format@30.2.0',
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: '@jest/expect-utils@30.2.0',
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: '@jest/get-type@30.1.0',
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: 'jest-matcher-utils@30.2.0',
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN   package: 'jest-message-util@30.2.0'.
npm WARN   required: { node: '^18.14.0 || ^20.0.0 || ^22.0.0 || >=24.0.0' },
npm WARN   current: { node: 'v16.20.2', npm: '8.19.4' }
```

Step-37- We will use command npm run build and the build folder is ready to be deployed.

```
npm WARN deprecated source-map@0.8.0-beta.0: The work that was done in this beta branch won't be included in future versions
npm WARN deprecated @babel/plugin-proposal-private-property-in-object@7.21.11: This proposal has been merged to the ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-private-property-in-object instead.
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
npm WARN deprecated eslint@0.57.1: This version is no longer supported. Please see https://eslint.org/version-support for other options.

added 1559 packages, and audited 1560 packages in 1m

273 packages are looking for funding
  run 'npm fund' for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.

[ec2-user@ip-192-168-0-51 web-tier]$ npm run build

> aws-3tier-web-layer@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.

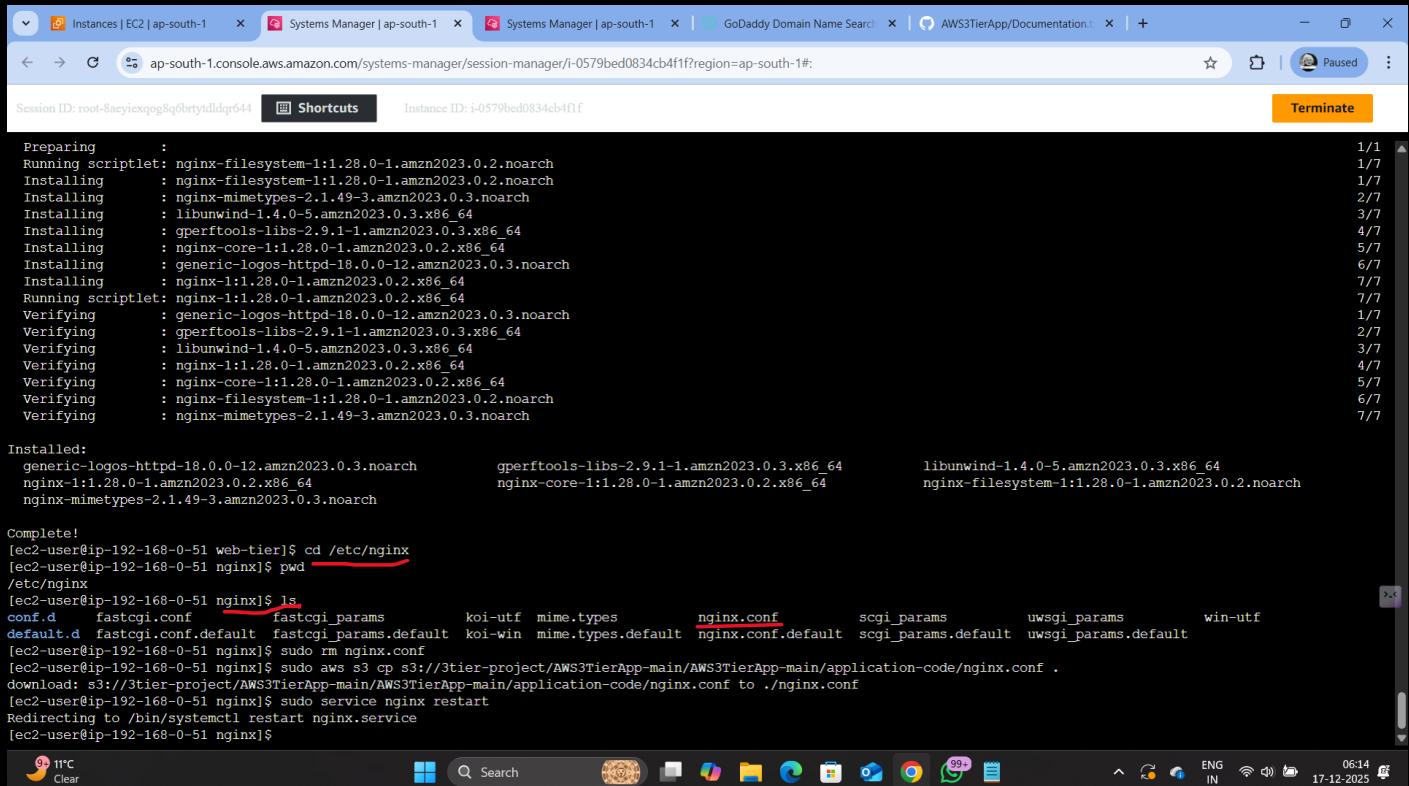
File sizes after gzip:

 75.11 kB  build/static/js/main.ab3be8dd.js
 1.77 kB   build/static/js/453.4cfe9ff7.chunk.js
 493 B     build/static/css/main.b20b6ac4.css

The project was built assuming it is hosted at ../.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.
```

Step-38- Update Nginx configuration and we will see nginx.conf file in the /etc/nginx directory.



```

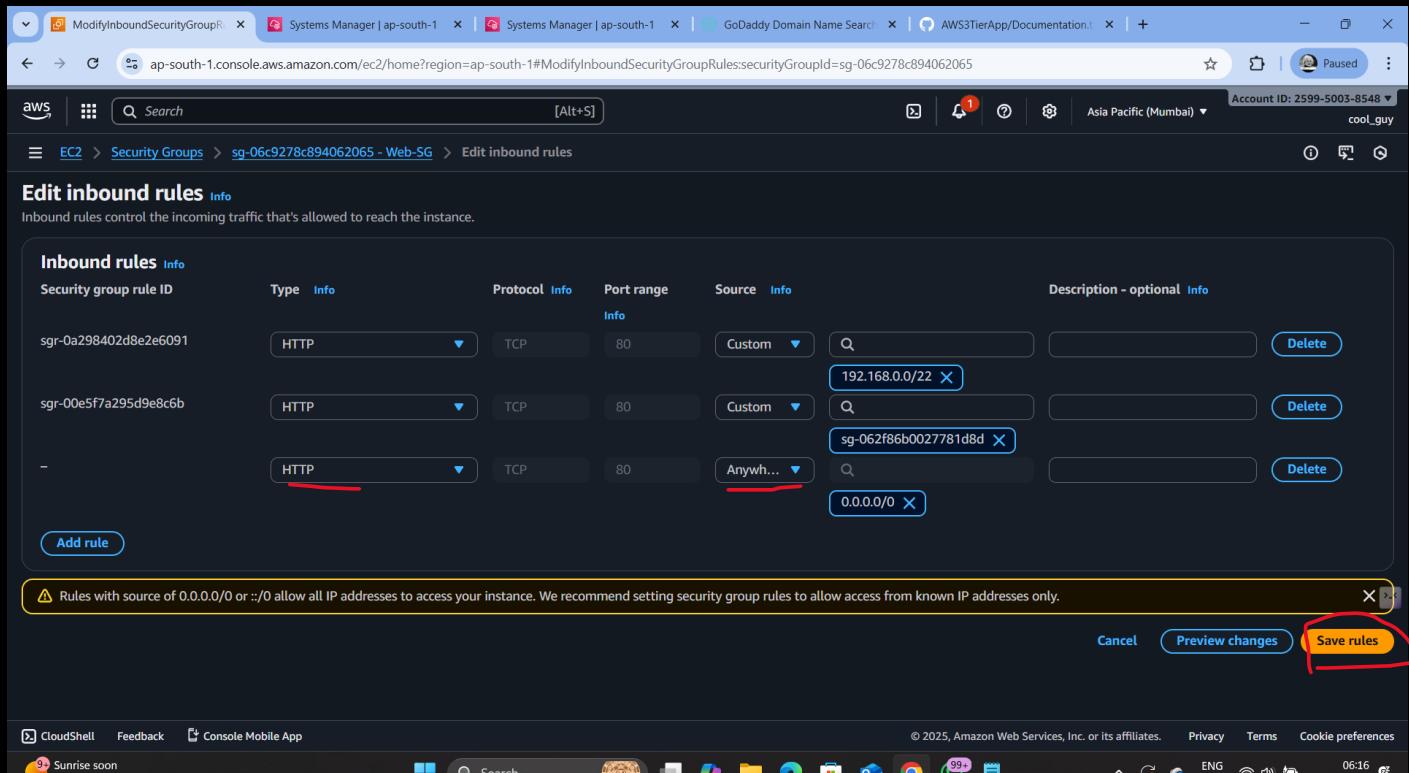
Preparing : 1/1
Running scriptlet: nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch 1/7
Installing : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch 1/7
Installing : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 2/7
Installing : libunwind-1.4.0-5.amzn2023.0.3.x86_64 3/7
Installing : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 4/7
Installing : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64 5/7
Installing : generic-logos-https-18.0.0-12.amzn2023.0.3.noarch 6/7
Installing : nginx-1:1.28.0-1.amzn2023.0.2.x86_64 7/7
Running scriptlet: nginx-1:1.28.0-1.amzn2023.0.2.x86_64 7/7
Verifying : generic-logos-https-18.0.0-12.amzn2023.0.3.noarch 1/7
Verifying : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 2/7
Verifying : libunwind-1.4.0-5.amzn2023.0.3.x86_64 3/7
Verifying : nginx-1:1.28.0-1.amzn2023.0.2.x86_64 4/7
Verifying : nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64 5/7
Verifying : nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch 6/7
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch 7/7

Installed:
generic-logos-https-18.0.0-12.amzn2023.0.3.noarch      gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
nginx-1:1.28.0-1.amzn2023.0.2.x86_64                 libunwind-1.4.0-5.amzn2023.0.3.x86_64
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch         nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch

Complete!
[ec2-user@ip-192-168-0-51 web-tier]$ cd /etc/nginx
[ec2-user@ip-192-168-0-51 nginx]$ pwd
/etc/nginx
[ec2-user@ip-192-168-0-51 nginx]$ ls
conf.d      fastcgi.conf      fastcgi_params      koi-utf      mime.types      nginx.conf      scgi_params      uwsgi_params      win-utf
default.d   fastcgi.conf.default  fastcgi_params.default  koi-win      mime.types.default  nginx.conf.default  scgi_params.default  uwsgi_params.default
[ec2-user@ip-192-168-0-51 nginx]$ sudo rm nginx.conf
[ec2-user@ip-192-168-0-51 nginx]$ sudo cp s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/nginx.conf .
download: s3://3tier-project/AWS3TierApp-main/AWS3TierApp-main/application-code/nginx.conf to ./nginx.conf
[ec2-user@ip-192-168-0-51 nginx]$ sudo service nginx restart
Redirecting to /bin/systemctl restart nginx.service
[ec2-user@ip-192-168-0-51 nginx]$

```

Step-39- We will edit the inbound rule for the security group Web-SG.



Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0a298402d8e2e6091	HTTP	TCP	80	Custom	192.168.0.0/22
sgr-00e5f7a295d9e8c6b	HTTP	TCP	80	Custom	sg-062f86b0027781d8d
-	HTTP	TCP	80	Anywhere	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Save rules (highlighted)

Step-40- Here we can see both the target group.

The screenshot shows the AWS EC2 Target groups page. On the left, there's a navigation sidebar with sections like AMIs, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main area is titled "Target groups (2)" and lists two entries:

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
External-Web-TG	arn:aws:elasticloadbalancing:ap-south-1:7000000000000000:targetgroup/External-Web-TG	80	HTTP	Instance	None associated	vpc-0798fa
App-Internal-TG	arn:aws:elasticloadbalancing:ap-south-1:7000000000000000:targetgroup/App-Internal-TG	4000	HTTP	Instance	App-internal-LB	vpc-0798fa

A red bracket highlights the first two columns of the table. Below the table, it says "0 target groups selected" and "Select a target group above."

Step-41- Now we had our domain which is co-om.in and we will create a record to use this domain as we can't give the IP address to the public. So our domain name will start from arya i.e arya.co-om.in

The screenshot shows the AWS Route 53 Quick create record page. The "Record name" field contains "arya" and the "Record type" dropdown is set to "CNAME". The "Route traffic to" section is expanded, showing "Alias to Application and Classic Load Balancer" and "Asia Pacific (Mumbai)". The "Evaluate target health" option is set to "Yes". At the bottom right, there are "Cancel" and "Create records" buttons, with "Create records" being highlighted by a red circle.

Step-42- Now we will register in the certificate manager make our domain Certified

The screenshot shows the 'Request public certificate' page in the AWS Certificate Manager. The URL is ap-south-1.console.aws.amazon.com/acm/home?region=ap-south-1#/certificates/request/public. The navigation bar includes 'AWS Certificate Manager > Certificates > Request certificate > Request public certificate'. The main section is titled 'Request public certificate' and contains a 'Domain names' field where 'arya.co-om.in' is entered. Below it is a 'Add another name to this certificate' button and a note about adding additional names. Under 'Allow export', there are two options: 'Disable export' (selected) and 'Enable export'. The 'Validation method' section shows 'DNS validation - recommended' selected. At the bottom, there are links for CloudShell, Feedback, and Console Mobile App, along with standard browser controls.

Step-43- We can see that our status of certificate is successful.

The screenshot shows the 'Certificate details' page for certificate ID **7519fc98-fdc4-43b8-80c9-d0bbf1c78d92**. The left sidebar shows 'AWS Certificate Manager (ACM)' with options like 'List certificates', 'Request certificate', 'Import certificate', and 'AWS Private CA'. The main area displays the 'Certificate status' with 'Identifier' as **7519fc98-fdc4-43b8-80c9-d0bbf1c78d92**, 'ARN' as **arn:aws:acm:ap-south-1:259950038548:certificate/7519fc98-fdc4-43b8-80c9-d0bbf1c78d92**, and 'Type' as 'Amazon Issued'. The 'Status' is shown as **Issued** with a green checkmark. Below this, the 'Domains (1)' section lists 'arya.co-om.in' with a status of **Success** and a green checkmark. There are buttons for 'Create records in Route 53' and 'Export to CSV'. At the bottom, there's a 'Details' section and standard browser controls.

Step-44- We will add a listener to our External-Web load balancer.

The screenshot shows the AWS CloudFront console with the path: EC2 > Load balancers > External-Web-ALB > Add listener. The 'Protocol' dropdown is set to 'HTTPS'. The 'Port' field contains '443'. Under 'Default action', 'No pre-routing action' is selected. In the 'Routing action' section, 'Forward to target groups' is chosen. A 'Target group' table lists 'External-Web-TG' with 'HTTP' as the protocol, a weight of '1', and a 'Percent' of '100%'. Below this, there's a link to '+ Add target group'. The 'Target group stickiness' section is collapsed. At the bottom, there are sections for 'Certificate source', 'Client certificate handling', and 'Listener tags - optional'.

Step-45- We will select our certificate from ACM and click on OK.

The screenshot shows the 'Certificate source' section where 'From ACM' is selected. A dropdown menu shows 'arya.co-om.in' with a red box around it. Below this, there's a link to 'Request new ACM certificate'. The 'Client certificate handling' section includes options for 'Mutual authentication (mTLS)'. A note at the bottom says 'Default attributes are applied when adding a listener. You can edit them after adding the listener.' There are also sections for 'Listener tags - optional' and 'Server-side tasks and status'.

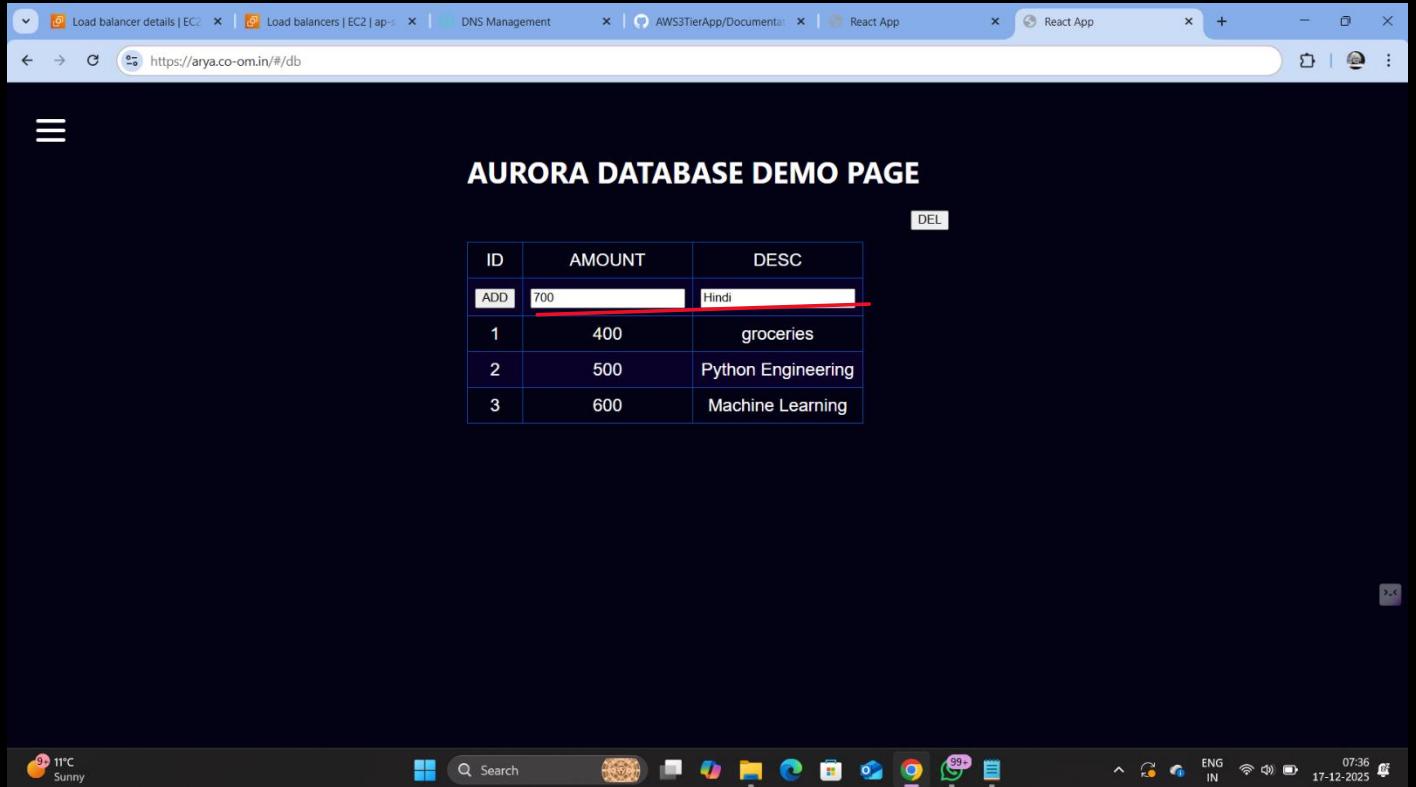
Step-46- In the security group we enabled the port 443 as it was not enabled earlier and now we can access our website.

The screenshot shows the AWS EC2 Load Balancers console. On the left, there's a navigation sidebar with sections like Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main area displays a table of load balancers. One row for 'External-Web-ALB' is selected and highlighted with a red box and arrow. Below the table, a detailed view for 'External-Web-ALB' is shown, specifically for the 'HTTPS:443' protocol. It lists two target groups: 'External-Web-TG' and 'External-Web-TG-1'. Both target groups have 100% health. The 'Forward to target group' rule for 'External-Web-TG-1' is highlighted with a red box and arrow. The browser status bar at the bottom shows the URL <https://arya.co-om.in/>.

Step-47- On browser we used our domain arya.co-om.in and now we can see that our website is running.

The screenshot shows a web browser window displaying a demo page for 'AWS THREE TIER ARCHITECTURE'. The page features a header with 'AWS PROJECT' and 'AWS THREE TIER ARCHITECTURE'. Below the header are icons for VPC (orange cloud), Amazon EC2 (server), and AWS RDS (blue cube). A banner at the bottom right reads 'KASTRO KIRAN V'. The browser status bar at the bottom shows the URL <https://arya.co-om.in/>. The browser interface includes standard controls like back, forward, search, and tabs.

Step-48- We can add some data also like an user for example I added Hindi and amount is 700, so it will be uploaded in the database also like an ordinary website.



AURORA DATABASE DEMO PAGE

ID	AMOUNT	DESC
ADD	700	Hindi
1	400	groceries
2	500	Python Engineering
3	600	Machine Learning

