

AWS 3 TIER PROJECT

Link to folk project

<https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code>

Real time snap shots

3/10/25

The screenshot shows the 'Create VPC' configuration page in the AWS Management Console. The left sidebar lists 'VPC settings', 'Preview', and 'Route tables'. The 'Preview' section displays the following network components:

- VPC Show details:** Your AWS virtual network, labeled 'Yesu-vpc'.
- Subnets (6):** Subnets within this VPC, categorized by Availability Zone:
 - us-east-1a:** Yesu-vpc-subnet-public1-us-east-1a, Yesu-vpc-subnet-private1-us-east-1a, Yesu-vpc-subnet-private2-us-east-1a, Yesu-vpc-subnet-private3-us-east-1a.
 - us-east-1b:** Yesu-vpc-subnet-public2-us-east-1b, Yesu-vpc-subnet-private2-us-east-1b, Yesu-vpc-subnet-private4-us-east-1b.
- Route tables (5):** Route network traffic to resources, including route tables for each subnet.
- Network connections (2):** Connections to other networks, including 'Yesu-vpc-igw' and 'Yesu-vpc-nat-public1-us-east-1a'.

The right sidebar includes sections for 'Tenancy' (set to 'Default'), 'Number of Availability Zones (AZs)' (set to 3), 'Number of public subnets' (set to 2), 'Number of private subnets' (set to 2), 'Customize subnets CIDR blocks' (with a dropdown menu showing 'Yesu-vpc-subnet-private4-us-east-1'), 'NAT gateways (\$)' (set to 1 AZ, 1 per AZ), 'VPC endpoint policy' (set to 'None'), and 'DNS options' (checkboxes for 'Create DNS Hostnames' and 'Create DNS resolution').

The screenshot shows the 'Create VPC workflow' page in the AWS VPC console. At the top, there's a success message: 'Success' with a green checkmark. Below it, a detailed list of steps is shown, each with a green checkmark indicating completion:

- Create VPC: vpc-0583263a9613799b3
- Enable DNS resolution
- Verifying subnet creation: vpc-0583263a9613799b3
- Create subnet: subnets-071fcf93e323edc1
- Create subnet: subnets-0f22d286f605640d
- Create subnet: subnets-017bd3917ee391d5
- Create subnet: subnets-08ff8fb2f505d9e5
- Create subnet: subnets-0baf6cf3930101723
- Create internet gateway: igw-07c76a063363764c
- Create route table: rtb-0d02c52349b98ab1
- Create route
- Associate route table
- Associate route table
- Allocate elastic IP: eipalloc-0f7740b030307402
- Create NAT gateway: nat-0699fb8195693952
- Wait for NAT Gateways to activate
- Create route table: rtb-0477009a956e38140
- Create route
- Associate route table
- Create route table: rtb-0024fbaa4096fa3f4
- Create route
- Associate route table
- Create route table: rtb-08530eaa15aae2d1
- Create route
- Associate route table
- Create route table: rtb-0f05f8a67efaf29a0
- Create route
- Associate route table
- Verifying route table creation

At the bottom right, there is a yellow 'View VPC' button.

View VPC

The screenshot shows the 'VPC dashboard' for the VPC 'vpc-0583263a9613799b3 / Yesu-vpc-vpc'. The left sidebar has a 'Virtual private cloud' section with various subnets, route tables, and other network components listed under 'Your VPCs'.

VPC Details:

- VPC ID: vpc-0583263a9613799b3
- State: Available
- Tenancy: default
- Default VPC: No
- Block Public Access: Off
- DNS hostnames: Enabled
- Main route table: rtb-0349f9f5f41d9e80
- IPv4 CIDR: 192.168.0.0/22
- IPv6 pool: -
- Network Address Usage metrics: Disabled
- Route 53 Resolver DNS Firewall rule groups: -
- Owner ID: 67727608751

Resource Map:

- VPC:** Your AWS virtual network (Yesu-vpc-vpc)
- Subnets (6):** Subnets within this VPC
 - us-east-1a: Yesu-vpc-subnet-public1-us-east-1a, Yesu-vpc-subnet-private1-us-east-1a, Yesu-vpc-subnet-private3-us-east-1a
 - us-east-1b: Yesu-vpc-subnet-public2-us-east-1b, Yesu-vpc-subnet-private2-us-east-1b
- Route tables (6):** Route network traffic to resources
 - rtb-0349f9f5f41d9e80 (highlighted in blue)
 - rtb-014bf9fb41d9e80
 - rtb-014bf9fb41d9e80
 - rtb-014bf9fb41d9e80
 - rtb-014bf9fb41d9e80
 - rtb-014bf9fb41d9e80
- Network connections (2):** Connections to other networks
 - Yesu-vpc-igw
 - Yesu-vpc-nat-public1-us-east-1a

Customize Subnets

Here we are going to edit the names of the subnets to fit their function.

The public subnets will be used for Web layer because we need web connectivity for them.

The private subnets for App layer and Database layer

The screenshot shows the AWS VPC dashboard with the following details:

- Subnets (1/6) Info**: A table listing 6 subnets across 3 availability zones (us-east-1a, us-east-1b, us-east-1c). Each subnet has a specific ID, route table, network ACL, and other configuration details.
- subnet-02ead30b58d0c5d46 / Yesu-vpc-subnet-DB2-us-east-1b**: A detailed view of this specific subnet.
 - Details Tab**: Shows the subnet ID (subnet-02ead30b58d0c5d46), IPv4 CIDR (192.168.2.192/26), and availability zone (us-east-1b).
 - IPv4 CDR Reservations**: Shows 59 available IPv4 addresses.
 - Network ACL**: Associated with acl-000560fbfb3b98525.
 - State**: Available.
 - IPv6 CDR**: None.
 - Network border group**: us-east-1.
 - Default subnet**: No.
 - Customer-owned IPv4 pool**: None.
 - Block Public Access**: Off.
 - VPC**: vpc-0583263a9e13799b3 | Yesu-vpc-vpc.
 - Auto-assign public IPv4 address**: No.
 - Outpost ID**: None.
 - Hostname type**: None.

-Security groups

Next we are going to create security groups. Total five security groups.

-One for web servers, another for internet facing load balancer(external load balancer) , one for app servers and another one for database servers.

- 1) Create Security group name -- As Web-ALB-SG , description the same then attache the vpc we created i.e Yesu-vpc

Inbound traffic

http traffic – protocol TCP port range 80 , source type Anywhere –IPV4 (that means open it to anybody) .

Outbound rules remain the same. Create security group

2)

Next security group for the web tier

-Name it Web-SG ,description the same , attached to Yesu-vpc .

-Add inbound rule as HTTP , protocol TCP port 80 . However , source should be custom and attached to Web-ALB-SG security group we created earlier.

-In addition , we can create another rule with HTTP traffic, protocol TCP , port 80 , custom IP 192.168.0.0/22

If you recall we used the IP address 192.168.0.0/22 to create vpc CIDR block range.

Security group (sg-00efb062a73293f27 | Web-SG) was created successfully

sg-00efb062a73293f27 - Web-SG

Details

Security group name Web-SG	Security group ID sg-00efb062a73293f27	Description Web-SG	VPC ID vpc-0583263a9613799b
Owner 677276087515	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sg-0c55a0f9d15383c	IPv4	HTTP	TCP	80	192.168.0.0/22	-
-	sg-0defad94d75a5784	-	HTTP	TCP	80	sg-02329670a11a130ef...	-

3)Now we need to create security group for private Apps running in the app server. Create security group for the App layer or App tier.

* React- js based application run Custom traffic TCP , protocol TCP on port range 4000 in the inbound rule and attach to vpc CDIR block 192.168.0.0/22.

Security group (sg-07edebfd8070bef60 | App-SG) was created successfully

sg-07edebfd8070bef60 - App-SG

Details

Security group name App-SG	Security group ID sg-07edebfd8070bef60	Description App-SG	VPC ID vpc-0583263a9613799b
Owner 677276087515	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Sharing - new | VPC associations - new | Tags

Inbound rules (1)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sg-0d349669991d96fea	IPv4	Custom TCP	TCP	4000	192.168.0.0/22	-

4) Create security group for internal load balancer . Name it internal-ALB-SG. Next attach our created vpc Yesu-vpc.

Here we open HTTP traffic, TCP protocol on port 80 , destination custom to 192.168.0.0/22 (vpc CIDR block)

The screenshot shows the AWS VPC dashboard with the 'Security Groups' section selected. A success message at the top indicates that the security group 'sg-01ff86e6b1fe8f331 | Internal-ALB-SG' was created successfully. The 'Details' tab is active, showing the security group name, ID, owner, and description. The 'Inbound rules' tab is selected, displaying a single rule: 'sg-0722b5403fe074dab' (IP version IPv4, Type HTTP, Protocol TCP, Port range 80, Source 192.168.0.0/22). The sidebar on the left provides navigation for VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, and Peering connections. The 'Security' section includes Network ACLs, Security groups, PrivateLink and Lattice, DNS firewall, and Rule groups.

5)

Next create security group for the database . Name it RDS-SG , attached to the vpc I created named Yesu-vpc

Now click on the inbound rule. Here for the traffic is going to be Mysql/aurora . The protocol is TCP and it runs of default port 3306 , source custom is custom network at IP 192.168.0.0/22. Recall this is vpc CIDR block. The resources communicating with the database will be restricted to the private subnets in the App tier.

The screenshot shows the AWS VPC dashboard with the 'Security Groups' section selected. A green alert bar at the top right says 'Security group (sg-009bea374420d47fc | RDS-SG) was created successfully'. Below it, the 'Details' tab for 'sg-009bea374420d47fc - RDS-SG' is open. The 'Inbound rules' tab is selected, showing one rule:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
sgr-008432e0d36a8a1e4	IPv4	MySQL/Aurora	TCP	3306	192.168.0.0/22	-	-

Note that Web-ALB-SG is internet facing if you examine the inbound rules. We have given the source as anywhere meaning it is public facing network. Anybody from outside can access the application which is running in the web tier.

II

- 1) The second step is to create the s3 bucket. Reason we need to upload the our application code in the S3 bucket. Why ? To avoid the complexity of uploading the application code directly to our EC2 instances.

The source of our application code may be found in the Github repository using the link provided below.

<https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code>

In my case I cloned the application code on my local machine via git bash. As follows below step by step

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

```
$ git clone https://github.com/kibush/3TierAWSArchitectureApp.git
```

Cloning into '3TierAWSArchitectureApp'...

remote: Enumerating objects: 93, done.

remote: Counting objects: 100% (93/93), done.

remote: Compressing objects: 100% (83/83), done.

Receiving objects: 50% (47/93) used 0 (delta 0), pack-reused 0 (from 0)

Receiving objects: 100% (93/93), 369.10 KiB | 14.20 MiB/s, done.

Resolving deltas: 100% (19/19), done.

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

\$ ls

3TierAWSArchitectureApp/

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

2)

Here I am going upload the entire application folder into the s3 bucket on AWS. Type s3 in the search window to find the s3 service (Scalable storage in the Cloud). Here a bucket with the default settings.

Note it is important to remember the region of your service . In my case it is US East (N. Virginia) US-east-1. We will call out bucket name yesu-3tier-project.

Keep all settings in default mode . Now upload only application code folder. In the bucket you can drag and drop folder directly in the s3 bucket from the local machine.

The screenshot shows the AWS S3 'Upload' interface. At the top, there's a search bar and a breadcrumb navigation path: Amazon S3 > Buckets > yesu-3tier-project > Upload. Below this is a large 'Upload' section with a 'Drag and drop files and folders here or choose Add files or Add folder...' area. A note says 'Add the files and folders you want to upload to S3. To upload a file larger than 1600GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API.' Below this is a table titled 'Files and folders' showing 34 items (34 total, 192.9 KB). The table has columns for Name, Folder, Type, and Size. The 'Destination' section shows 'yesu-3tier-project' selected. It includes 'Destination details' (Bucket settings that impact new objects stored in the specified destination) and 'Properties' (Specify storage class, encryption settings, tags, and more).

Name	Folder	Type	Size
Burger.js	application-code/web-tier/src/components/Burger/	text/javascript	545.0 B
Burger.styled.js	application-code/web-tier/src/components/Burger/	text/javascript	940.0 B
index.js	application-code/web-tier/src/components/Burger/	text/javascript	35.0 B
DatabaseDemo.css	application-code/web-tier/src/components/Database...	text/css	655.0 B
DatabaseDemo.js	application-code/web-tier/src/components/Database...	text/javascript	4.0 KB
Home.js	application-code/web-tier/src/components/Home/	text/javascript	451.0 B
index.js	application-code/web-tier/src/components/Menus/	text/javascript	35.0 B
Menus.js	application-code/web-tier/src/components/Menus/	text/javascript	1.0 KB
Menus.styled.js	application-code/web-tier/src/components/Menus/	text/javascript	934.0 B
AWS Snap Shots	application-code/	-	2.0 B

Click the up load file once you drag and drop file.

The screenshot shows the AWS S3 console with the 'Upload: status' page. At the top, a green success message says 'Upload succeeded' with a link to 'View details'. Below it, a summary table shows the destination as 's3://your-tier-project' and the upload status as 'Succeeded' for 14 files (192.9 KB). A note says 'After you navigate away from this page, the following information is no longer available.' The main area displays a table of 'Files and folders' with columns for Name, Folder, Type, Size, Status, and Error. All files listed have a status of 'Succeeded'. The table includes files like 'Burger.js', 'Burger.styles.css', 'index.js', 'DatabaseDemo.css', 'DatabaseDemo.js', 'Home.js', 'index.css', 'Menu.js', 'Menu.styles.js', and 'AWS Snap Shots'.

Here you can view the application code folder uploaded to s3 bucket.

The screenshot shows the AWS S3 console with the 'Objects' page for the 'application-code/' folder. The top navigation bar shows 'Amazon S3 > Buckets > your-tier-project > application-code/'. The main table lists five objects: 'app-tier/' (Folder), 'AWS Snap Shots' (Folder), 'nginx-without-SSL.conf' (conf), 'nginx.conf' (conf), and 'web-tier/' (Folder). The table has columns for Name, Type, Last modified, Size, and Storage class. All objects are in the 'Standard' storage class.

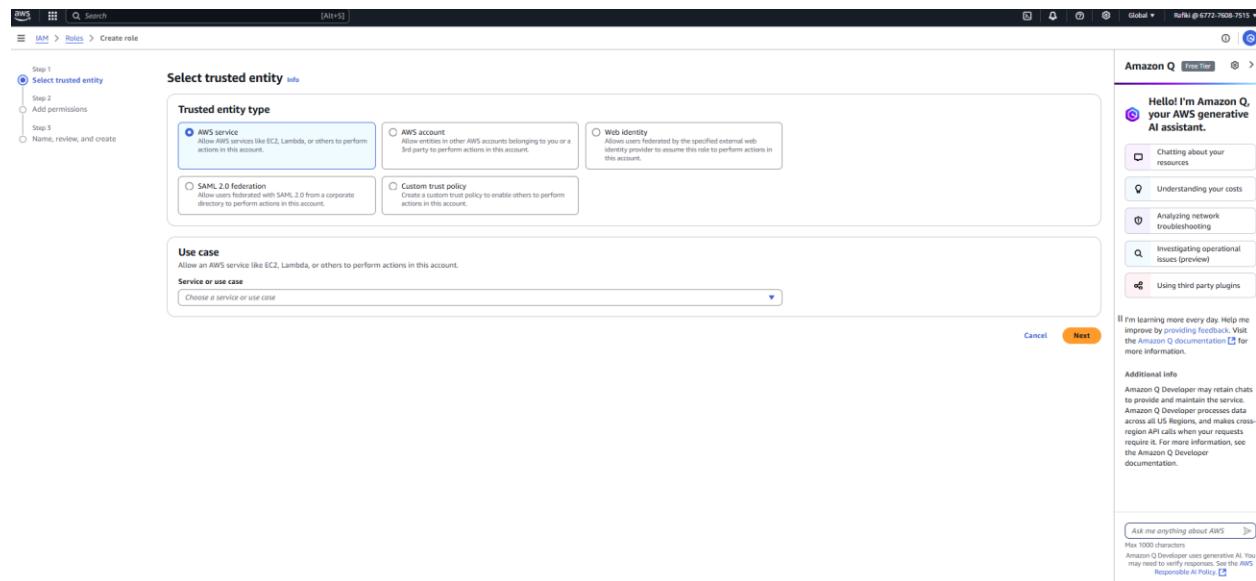
3) Next lets create an IAM role . What is the reason for creating IAM role. Remember we have two EC2 instances in the App tier. In addition they are in the private subnet . Also , the database is also unavailable in the private subnet. It is important to not give these 2 services access to outsiders. So , we are going to close any vulnerability. There may be 2 ways to connect to these two resources i.e. EC2 instances and the RDS. We can use the bastion host which the concept in vpc. That means we create a bastion host in the public subnet and by using the bastion host we will be able to connect to resources in the private subnet. The second way is instead of creating a bastion host an the complexities that come with it we can instead use the AWS service called the systems manager called SSM. That means if we attaché the appropriate role to EC2 instance which are there in the private subnet. Even though the

machines are there in the private subnet we access this virtual machines using the SSM agent. So, in our case we use the IAM role to be able to connect to these private layer or tier.

To, define our role we are going to create a policy and attach Amazon EC2 role for SSM.

4)

Go to IAM (Identity & Access Management) and click on roles to create one.



- Initially I select entity as AWS service . At the bottom option use case option I chose EC2 then next . Now which role or policy we need to access , in my case I choose **amazonec2roleforSSM**.

AmazonEC2RoleforSSM Info

This policy will soon be deprecated. Please use AmazonSSHManagedInstanceCore policy to enable AWS Systems Manager service core functionality on EC2 instances. For more information see <https://docs.aws.amazon.com/systems-manager/latest/userguide/setup-instance-profile.html>

Policy details

Type: AWS managed	Creation time: May 29, 2015, 13:48 (UTC-04:00)	Edited time: January 24, 2019, 14:20 (UTC-05:00)	ARN: arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM
-------------------	--	--	---

Permissions **Entities attached** **Policy versions** **Last Accessed**

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

```

1 <[{"version": "2012-10-17", "statement": [{"effect": "Allow", "action": "ssm:DescribeAssociation", "resource": "*"}, {"effect": "Allow", "action": "ssm:GetDeployableSnapshotForInstance", "resource": "*"}, {"effect": "Allow", "action": "ssm:DescribeDocument", "resource": "*"}, {"effect": "Allow", "action": "ssm:GetManifest", "resource": "*"}, {"effect": "Allow", "action": "ssm:ListAssociations", "resource": "*"}, {"effect": "Allow", "action": "ssm:ListDocumentAssociations", "resource": "*"}, {"effect": "Allow", "action": "ssm:PutInventory", "resource": "*"}, {"effect": "Allow", "action": "ssm:PutCompliance", "resource": "*"}, {"effect": "Allow", "action": "ssm:PutComplianceResult", "resource": "*"}, {"effect": "Allow", "action": "ssm:UpdateAssociationStatus", "resource": "*"}, {"effect": "Allow", "action": "ssm:UpdateInstantAssociationStatus", "resource": "*"}], "Resource": "*"}, {"effect": "Allow", "action": "ssm:MessagesCreateControlChannel", "resource": "*"}, {"effect": "Allow", "action": "ssm:MessagesCreateDatatChannel", "resource": "*"}, {"effect": "Allow", "action": "ssm:MessagesDeleteControlChannel", "resource": "*"}, {"effect": "Allow", "action": "ssm:MessagesOpenDatatChannel", "resource": "*"}, {"Resource": "*"}, {"effect": "Allow", "action": "ec2:MessagesAcknowledgeMessage", "resource": "*"}, {"effect": "Allow", "action": "ec2:MessagesDeliverMessage", "resource": "*"}, {"effect": "Allow", "action": "ec2:MessagesFailMessage", "resource": "*"}], "Resource": "*"}]>
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```

Ask me anything about AWS >

Max 1000 characters.
Amazon Q is a generative AI. You may need to verify responses. See the AWS Responsible AI Policy.

-Most importantly we access to s3 bucket.

AmazonEC2RoleforSSM Info

This policy will soon be deprecated. Please use AmazonSSHManagedInstanceCore policy to enable AWS Systems Manager service core functionality on EC2 instances. For more information see <https://docs.aws.amazon.com/systems-manager/latest/userguide/setup-instance-profile.html>

Policy details

Type: AWS managed	Creation time: May 29, 2015, 13:48 (UTC-04:00)	Edited time: January 24, 2019, 14:20 (UTC-05:00)	ARN: arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM
-------------------	--	--	---

Permissions **Entities attached** **Policy versions** **Last Accessed**

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Allow (8 of 440 services)

Service	Access level	Resource	Request condition
CloudWatch	Limited: Write	All resources	None
CloudWatch Logs	Limited: List, Write	All resources	None
Directory Service	Limited: List, Write	All resources	None
EC2	Limited: List	All resources	None
EC2 Messages	Full access	All resources	None
S3	Limited: List, Read, Write	All resources	None
SSH Messages	Full access	All resources	None
Systems Manager	Limited: List, Read, Write	All resources	None

Show remaining 432 services

Ask me anything about AWS >

Max 1000 characters.
Amazon Q is a generative AI. You may need to verify responses. See the AWS Responsible AI Policy.

-Also attach or add permission by clicking the box on the left to make sure it is attached.

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. The sidebar on the right features an AI assistant named 'Amazon Q' with sections like 'Chatting about your resources', 'Understanding your costs', 'Analyzing network troubleshooting', 'Investigating operational issues (preview)', and 'Using third party plugins'. It also includes a feedback section and developer documentation links.

-Click next , give the role name as Yesu-EC2-Role. Scroll down and create role.

The screenshot shows the 'Roles' page in the IAM console. The newly created role 'Yesu-ec2-Role' is highlighted. Other roles listed include AWS service roles like autoscaling, cost-optimization-hub, elasticloadbalancing, organizations, rds, and support, along with a few AWS Lambda roles. The sidebar on the left provides navigation links for IAM management, reports, and activity.

View role as shown below

5) Next we have to now create a database. Here we can use a service called RDS in the AWS cloud to create database instances. In the search bar on the AWS cloud type RDS.

-Note in RDS billing is chargeable unless you are covered under free tier.

Now it important to create RDS with the specific subnets. Also, within the RDS we do not have permission to select specific subnets. So the other option we have is to use the subnet groups. Here , we will add our custom vpc and DB1 and DB2 subnets we have created then attaché DB-subnet- group to RDS instance.

Create subnet group . Name it DB-SNGP then chose custom vpc. Yesu-vpc we created before.

-Select availability zone available. I will chose us-east-1a and us-east -1b. Next you see option to select subnets .In our care we choose Yesu-vpc-DB1- us-east-1a and Yesu-vpc-DB2- us-east-1b private subnets previously created at the beginning of the project. So , click create .

The screenshot shows the 'Create DB subnet group' wizard. Step 1: 'Subnet group details' shows a success message: 'Successfully deleted db-subnet-group'. Step 2: 'Create DB subnet group' shows fields for Name (DB-SNGP), Description (DB-SNGP), and VPC (Yesu-vpc). Step 3: 'Add subnets' shows availability zones us-east-1a and us-east-1b selected. Step 4: 'Subnets selected' shows two subnets: Yesu-vpc-subnet-DB1-us-east-1a and Yesu-vpc-subnet-DB2-us-east-1b. A note says 'For Multi-AZ DB clusters, you must select 3 subnets in 3 different Availability Zones.' Step 5: 'Subnets selected (2)' shows the same two subnets. On the right, the Amazon Q AI assistant is visible.

-As shown below we have created subnet gropus.

The screenshot shows the 'Subnet groups' page with one entry: db-sngp. It has a status of 'Complete' and is associated with VPC 'vpc-0583265a9e1379963'. The left sidebar shows the RDS navigation menu.

-Hence, we should attached subnet group to the database. How ? Click database on the left select tab option. Create databases click standard create and select MySQL or Aurora (MySQL compatible) . The commands are the same. Edition is MySQL community choose an earlier version. Templates chose free tier option. Self managed password or AmazonRDS can create one for you and view on the credential details after database creation.

Next make sure Instance configuration chooses burstable classes as db.t4g.micro. Storage as GP2 allocation 20 GiB as the default allocation.

Connectivity to remain as default.

-Note it is important to chose the vpc we created Yesu-vpc at the beginning of the project and the DB subnet group called db-sngp we created. No public access too because have a BD as a private subnet.

-Also, note we have a own security group we created before. So, need to uncheck default and chose RDS- SG security group we created earlier. Availability zone is not necessary for our project.

-Certificate of authority we go with the default one and monitoring remain default.

-For additional configuration ensure that option fall under the free tier estimated monthly cost.

View set as follows ;

Screenshot of the AWS RDS Create database page for MySQL. The 'Standard create' option is selected. The 'Engine options' section shows MySQL as the chosen engine type. To the right, a detailed description of MySQL is provided, including its popularity as the most popular open-source database and its compatibility with MySQL Community Edition. It also lists supported features like up to 64 TiB database size, General Purpose, Memory Optimized, and Burstable Performance instance classes, automated backup, and point-in-time recovery, up to 15 Read Replicas per instance, and cross-region replication.

Screenshot of the AWS RDS Create database page for MySQL. The 'MySQL 8.0.40' engine version is selected. The 'Enable RDS Extended Support' checkbox is unchecked. The 'Templates' section shows 'Production' and 'Dev/Test' as options, with 'Free Tier' selected. The 'Availability and durability' section details deployment options: 'Multi-AZ DB cluster deployment (3 instances)' (selected), 'Multi-AZ DB instance deployment (2 instances)', and 'Single-AZ DB instance deployment (1 instance)'. Each option is described with its benefits. To the right, a detailed description of MySQL is provided, including its popularity as the most popular open-source database and its compatibility with MySQL Community Edition. It also lists supported features like up to 64 TiB database size, General Purpose, Memory Optimized, and Burstable Performance instance classes, automated backup, and point-in-time recovery, up to 15 Read Replicas per instance, and cross-region replication.

RDS > Create database

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "myDatabase"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Credentials Settings

Master username: info

Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management

Amazon RDS can manage or manage your master user credentials.

Managed in AWS Secrets Manager - most secure
AWS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

Self managed
Create your own password or have RDS create a password that you manage.

Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.

You can view your credentials after you create your database. Click the "View credential details" in the database creation banner to view the password.

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class: info

Hide filters

Show instance classes that support Amazon RDS Optimized Writes info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Include previous generation classes

Standard classes (includes m classes)

Memory optimized classes (Includes r and a classes)

Unstable classes (includes t classes)

db.t4g.micro
2 vCPUs · 1 GB RAM · Network: Up to 2,085 Mbps

Storage

Storage type: info

Amazon RDS (Provisioned IOPS SSD) storage volumes are now available.

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage: info

20 GB

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL engine in combination with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes
- Supports automated backup and point-in-time recovery
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region

RDS > Create database

Connectivity info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) info

Choose the VPC. The VPC is your own custom networking environment for this DB instance.

Region: us-east-1 Subnet: 1 Availability Zone: 1

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group: info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

db-sng-1
2 Subnets, 2 Availability Zones

Public access: info

No
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

Yes
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (Firewall) info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing
Choose existing VPC security groups

Create new
Create new VPC security group

Existing VPC security groups:
Choose one or more options

RDS Proxy: info

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

Create an RDS Proxy info
RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see [Amazon RDS Proxy pricing](#).

Certificate authority - optional: info

Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rds-ca-rsa2048-g1 (Default)
Default: Yes, D.2021

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL engine in combination with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB
- Supports General Purpose, Memory Optimized, and Burstable Performance Instance classes
- Supports automated backup and point-in-time recovery
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region

RDS > Create database [48+13] United States (N. Virginia) United States (N. Virginia) United States (N. Virginia) United States (N. Virginia)

RDS-CA-HA24H-1 (Default)
Buyer May 12, 2018

If you don't select a certificate authority, RDS chooses one for you.

Additional configuration

Tags - optional
A tag consists of a case-sensitive key-value pair.
No tags associated with the resource.
[Add new tag](#)
You can add up to 50 more tags.

Database authentication
Database authentication options [Info](#)
 MySQL authentication
Authenticates using database passwords.
 Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.
 Password and Kerberos authentication
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Monitoring [Info](#)
Choose monitoring tools for this database. Database Insights provides a combined view of Performance Insights and Enhanced Monitoring for your fleet of databases.
 Database Insights - Advanced

- Retains 15 months of performance history
- Performance Metrics
- Integration with CloudWatch Application Signals

 Database Insights - Standard

- Retains 7 days of performance history, with the option to pay for the retention of up to 24 months of performance history

Database Insights pricing is separate from RDS monthly estimates. See [Amazon CloudWatch pricing](#).

Additional monitoring settings
Enhanced Monitoring, CloudWatch Logs and DevOps Guru
 Enhanced monitoring
Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.
Log exports
Select the log types to publish to Amazon CloudWatch Logs.
 Audit log
 Error log
 General log

MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the full feature set of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB
- Supports General Purpose, Memory Optimized, and Burstable Performance Instances
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replica cross-region.

RDS > Create database [48+13] United States (N. Virginia) United States (N. Virginia) United States (N. Virginia) United States (N. Virginia)

RDS-CA-HA24H-1 (Default)
Buyer May 12, 2018

If you do not specify a database name, Amazon RDS does not create a database.

Additional configuration
Database options, encryption turned on, backup turned off, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options
Initial database name [Info](#)

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)

default.mysql8.0

Option group [Info](#)

default.mysql8.0

Backup
 Enable automated backups
Creates a point-in-time snapshot of your database

Encryption
 Enable encryption
Chooses to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console.

AWS KMS key [Info](#)

(default) arn:aws:kms:us-east-1:677276087515:alias/MyRDSKey

Account
677276087515

KMS key ID
956-101f-9cc1-4a25-466b-630f03ed5d42

Maintenance
Auto minor version upgrade info
 Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the full feature set of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB
- Supports General Purpose, Memory Optimized, and Burstable Performance Instances
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replica cross-region.

Enable automated backups
Creates a point-in-time snapshot of your database

Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. [Info](#)

AWS KMS key [Info](#)

Accounts
67726087515

KMS key ID
956745f6-0cc-4a25-b660-630f03ed54d2

Maintenance

Automatic minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

Maintenance window [Info](#)
Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.
 Choose a window
 No preference

Deletion protection
 Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:
 • 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
 • 20 GB of General Purpose Storage (SSD)
 • 20 GB for AWS CloudWatch Metrics and any user-initiated DB Snapshots.

Learn more about using Free Tier [\[?\]](#)
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#).

You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

[Cancel](#) [Create database](#)

-Enable database creation. The creation is not instantaneous so give the process of creation sometime.

Creating database database-1											View credential details																										
Your database might take a few minutes to launch. The only way to view your master password is to choose View credential details during database creation. You can modify your DB instance to create a new password at any time. You can use settings from database-1 to simplify configuration of suggested database add-on while we finish creating your DB for you.																																					
Group resources Modify Actions Restore from S3 Create database																																					
Databases (1)																																					
<table border="1"> <thead> <tr> <th>DB identifier</th> <th>Status</th> <th>Role</th> <th>Engine</th> <th>Region</th> <th>Size</th> <th>Recommendations</th> <th>CPU</th> <th>Current activity</th> <th>Maint...</th> <th>VPC</th> <th>Multi-AZ</th> </tr> </thead> <tbody> <tr> <td>database-1</td> <td>Creating</td> <td>Instance</td> <td>MySQL Co...</td> <td>us-east-1b</td> <td>db.t4g.micro</td> <td>-</td> <td>-</td> <td>-</td> <td>none</td> <td>vpc-d583265a9613799b5</td> <td>No</td> </tr> </tbody> </table>														DB identifier	Status	Role	Engine	Region	Size	Recommendations	CPU	Current activity	Maint...	VPC	Multi-AZ	database-1	Creating	Instance	MySQL Co...	us-east-1b	db.t4g.micro	-	-	-	none	vpc-d583265a9613799b5	No
DB identifier	Status	Role	Engine	Region	Size	Recommendations	CPU	Current activity	Maint...	VPC	Multi-AZ																										
database-1	Creating	Instance	MySQL Co...	us-east-1b	db.t4g.micro	-	-	-	none	vpc-d583265a9613799b5	No																										

6)

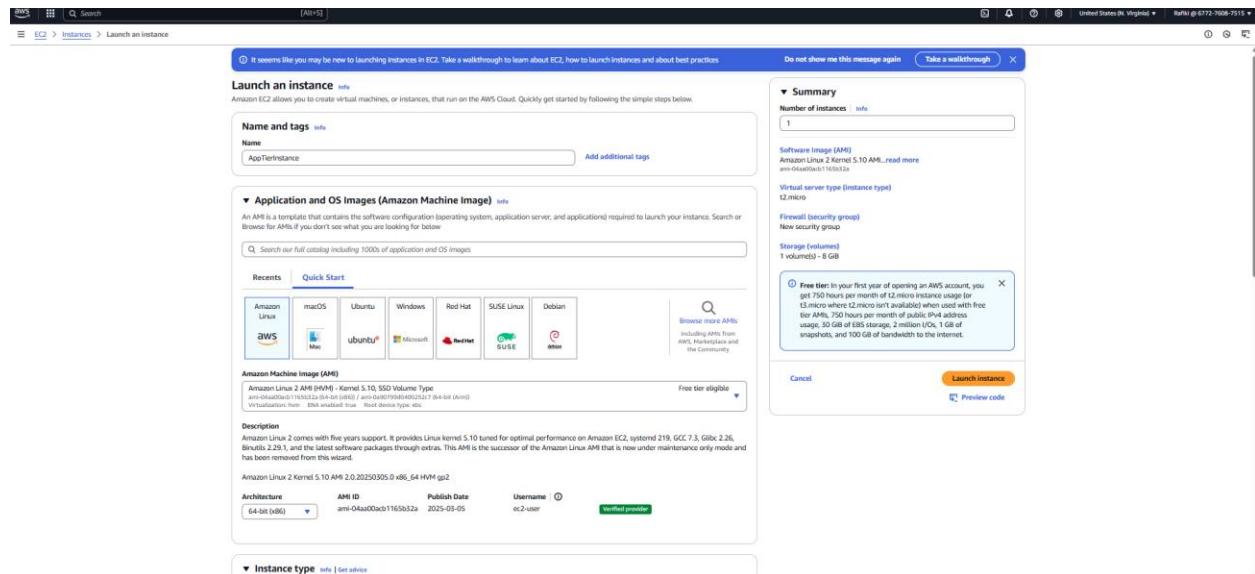
Next is lets create app tier resources.

-Deploy application-tier resources , include the configuration of an internal load balancer for traffic distribution within the tier.

-Here we are going to create ec2 instances to connect to (DB) database. When ever end user connect to database thru the App tier instances the goal is let the data get stored in the Database tier. Further, the required configuration will be implemented to connect instance to database.

-First, ec2 instance will be launched to connect to database. The configuration will produced from the first of 2 instances. The second instance will not be created manually , but will use an auto scaling group to launch the second instance. Then connect the auto scaling group to the load balancer. Load balancer will support and maintain AZ1 and AZ2 instances.

Below start by launching an instance.



Here we will not connect with key pair.

In network settings, select the vpc we created. Yesu-vpc and should be connected to Yesu-vpc-App1-us-east-1a subnet only. Recall this is a private subnet only and should be connected to public subnet.

Firewall (security group) . Here recall we created the app security group called App-SG so attached to it.

Storage can remain default for the free tier consumption.

-In the advanced details attaché the IAM role we created. **Yesu-ec2-Role**. Then go ahead and launch instance.

-In-order to connect instance to app tier or DB tier. Select ec2 and click connect session manager and connect tab. Here we will be connected ec2 instance session manager via a terminal.

-Now if we do not get to the part above , where you are not able to connect to session manager thru ec2 instance it means the security group has not been setup correctly. This also means Port 4000 is not open.

Session ID: Rafftsi-1vttdh04ukpeanw7orfajnlq Instance ID: i-0d851b282110783c

- On this instance terminal we have to setup configuration to connect the database.

-First lets switch to root user by typing the following command ; sudo su -

-To certain you are in the root user type ;**whoami** and the output should state **root**

-At the prompt type to reset to default directory ; prompt :\$ `cd ..`

-Then type type pwd

-The goal is to be in the user directory or /usr

-From here we need to move to ec2user. So type ; **cd /home/ec2-user/**

-Type `pwd` to know where you are or current working directory

-Next test for connectivity by pinging google ; # 8.8.8.8 ; Therefore you should see a response

-So how is the internet connecting to this instance ? Via internet gateway.

-Next install mysql client

```
# sudo yum install mysql -y
```

```
Session ID: Rafa4-Fvtbhekkpeadw/0rfsyqtlq
Instance ID: I-0dd510282110785c
Termina

[monit@ip-192-168-2-63 ~]$ sudo yum install mysql -y
Last metadata expiration check: 0:00:00 ago on Mon Jul 10 10:45:00 2017.
Dependencies Resolved
Transaction Summary
Install 1 Package
Total download size: 8.8 M
Is this ok [y/n]: y
Downloaded packages:
mysql-5.5.48-1.x86_64.rpm | 3.6 kB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Installing : 1:mysql-5.5.48-1.x86_64
Verifying : 1:mysql-5.5.48-1.x86_64
Installed:
mysql.x86_64 1:5.5.48-1.mysql0.1

Complete!
[monit@ip-192-168-2-63 ~]$
```

- For us to connect to database and perform basic configuration. Use the following command below;

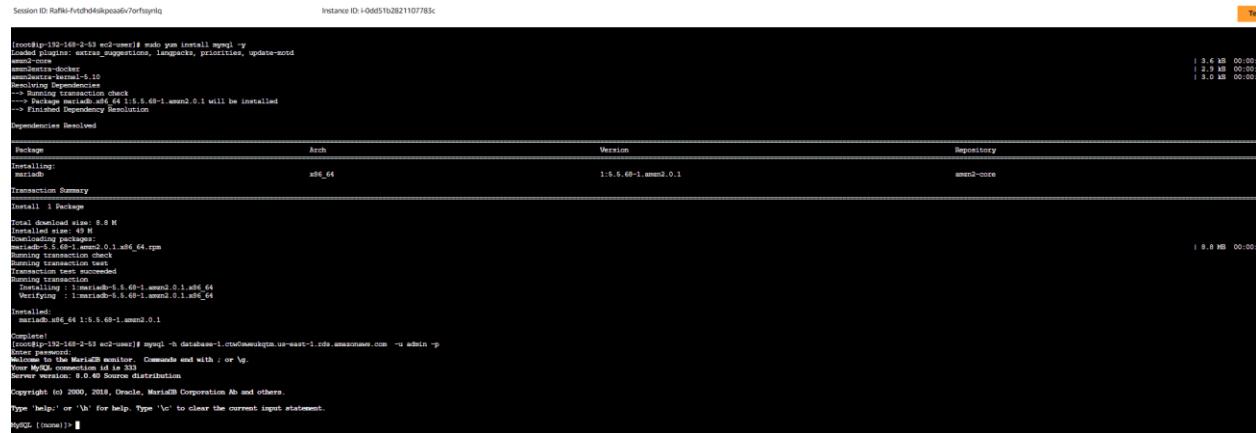
Db password : 3TqrzMz4T6IkKksUqY9u

Endpoints : database-1.ctw0sweukqtm.us-east-1.rds.amazonaws.com

Port : 3306

-Execute the following command to log into database ; mysql -h database-1.ctw0sweukqtm.us-east-1.rds.amazonaws.com -u admin -p

-Below we are logged into Mysql database



```
Session ID: Rnfki-4tldh4dkpcos6v7orf5mytq
Instance ID: i-0dd51b282110778c

[root@ip-192-168-2-53 ec2-user]# sudo yum install mysql -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-mod
Resolving Dependencies
--> Running transaction check
--> Checking on conflicts: mariadb-libs-5.5.60-1.mmm2.0.1 will be installed
--> Finished Dependency Resolution
Dependencies Resolved

Transaction Summary
Package           Arch      Version            Repository
Installing:
mariadb          x86_64   1:5.5.60-1.mmm2.0.1    mariadb-core
Transaction Summary
Install  1 Package
Total download size: 0.8 M
Installed size: 49 M
Transaction test
Running transaction test
Transacting packages succeeded
Running transaction
  Installing : mariadb-5.5.60-1.mmm2.0.1.x86_64
  Verifying  : mariadb-5.5.60-1.mmm2.0.1.x86_64

Installed:
  mariadb.x86_64 1:5.5.60-1.mmm2.0.1

Complete!
[root@ip-192-168-2-53 ec2-user]# mysql -h database-1.ctw0sweukqtm.us-east-1.rds.amazonaws.com -u admin -p
Welcome to the MariaDB monitor. Commands end with ; or \g.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql: [Warning] Using a password on the command line interface can be insecure.
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql: [ERROR] (conn) 1130: Host 'ip-192-168-2-53' is not allowed to connect to this MariaDB server.
```

-Once logged into database create a database named webappdb . Once the database has been created also create a table.

-Also, check whether we created database or not ? Type SHOW DATABASES (command) .

-After database creation has been verified we also need to use it. So at the prompt type : USE webappdb

-Next we are going to create a table called ‘transactions’ using the commands from git hub repository.

-Using the following command create table ;

CREATE TABLE IF NOT EXISTS transactions(

id INT NOT NULL AUTO_INCREMENT,

amount DECIMAL(10,2),

description VARCHAR(100),

PRIMARY KEY(id)

);

-Now verify the table was created type the following command > SHOW TABLES;

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 1081
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MySQL [(none)]> CREATE DATABASE webappdb;
Query OK, 1 row affected (0.03 sec)

MySQL [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| webappdb |
+-----+
5 rows in set (0.02 sec)

MySQL [(none)]> USE webappdb;
Database changed
MySQL [webappdb]> 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 (0.09 sec)

MySQL [webappdb]> SHOW TABLES ;
+-----+
| Tables_in_webappdb |
+-----+
| transactions |
+-----+
1 row in set (0.00 sec)

MySQL [webappdb]> INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');
Query OK, 1 row affected (0.00 sec)
```

-Now we need to insert some dummy information in the table to test its functionality. Now check the entries in the table by executing the following command ; `SELECT *FROM transactions;`

- Now since we tests the entering contents from the instance this may not be the reality in the real time senario.

-In conclusion – We /I was able to able to connect to the database server through the appTier and able to write information via the database server.

Now recall database Tier is not required to conenct to appTier or webTier. It exists in the back end.

-Next we need to exit the database so type **exit** command.

One more configuration is required once you exit database. We need to locate the application code on our host machine and find the file called **DBConfig.js**. Open this file with visual studio code and make some changes. Copy our the database endpoints and paste it to line number 2 my code. User remains default admin . Change the database password to current database. Database name should be the same as we created. Ensure file is saved sucessfully. Ctrl S

-Now lets go to the s3 service . Locate the application code we uploaded and update the DBconfig.js file as well. So we need to create a new one. So drag and drop the updated file into the s3 bucket to replace the old one. There will be no issues since we had not enable versioning in on our project folder.

8)

Next to need to install and configure nodejs. The instructions are in the 3 tier repository.

Using the following command; curl -o-

https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash.

-The we need to source it by going to the bash shell. Type the following command in the prompt ;
source ~/ .bashrc

-Next install nvm , also known as node version manager. nvm install 16

-Now tell our manager to use nvm 16 type ; **nvm user 16**

-Now run node as a service. For that install ; **npm install -g pm2**

-We need to switch our path from root to user. How ? Type `pwd` to know where you are in the file system. **Type `cd ..`** .

The above command takes you home. Or type `cd ~`

-Copy the path for the s3 bucket to app-tier folder

```
Root ~] # sudo aws s3 cp s3://yesu-3tier-project/application-code/app-tier/ app-tier --recursive
```

```
Session ID: Rufe3-2ab5-2fb5ogZ-4tr-hqkhilpe Instance ID: 1-0d551b2821107785c [Terminate]

root@ip-192-168-2-53:~# ./yeus-3tier-project/application-code/app-tier/app-tier-requires
ReadConfig: s3://yeus-3tier-project/application-code/app-tier/transactionService.json to app-tier/transactionService.json
ReadConfig: s3://yeus-3tier-project/application-code/app-tier/package-lock.json to app-tier/package-lock.json
ReadConfig: s3://yeus-3tier-project/application-code/app-tier/package.json to app-tier/package.json
ReadConfig: s3://yeus-3tier-project/application-code/app-tier/decoupling.json to app-tier/decoupling.json
ReadConfig: s3://yeus-3tier-project/application-code/app-tier/decoupling.js to app-tier/decoupling.js
root@ip-192-168-2-53:~# ls
root@ip-192-168-2-53:~#
```

- Switch to app-tier directory type ; cd app-tier/

Install node port manager type ; npm install

- Type **ls** and you will able to view an **index.js** file.

-Start index.js file type ; pm2 start index.js

Now from the session manager terminal you should online status in the table described as the image above. Also type in the terminal ; **pm2 list**

-And **pm status**

- type ; pm2 logs

Type control C to return to terminal prompt.

```
Session ID: Rafiki-2d5a2b5igp2ir4tnfhqkh8pc
Instance ID: i-0dd51b2821107783c

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

Start and Deploy any application:
$ pm2 start app.js -i 4
Load Balance 4 instances of app.js:
$ pm2 start app.js -i 4
Monitor in production:
$ pm2 monitor
Make pm2 auto-boot at server restart:
$ pm2 startup
To go further check our:
http://pm2.io

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

[PM]
  id  name  namespace  version  mode  pid  uptime  ⚡  status  cpu  mem  user  watching
  0  index  default    1.0.0   [ ]  19624  0s  0  online  0%  24.9mb root  disabled
[PM]# ps aux | grep pm2
root@ip-192-168-0-53:~# pm2 list
[PM]
  id  name  namespace  version  mode  pid  uptime  ⚡  status  cpu  mem  user  watching
  0  index  default    1.0.0   [ ]  19624  2m  0  online  0%  51.6mb root  disabled
[PM]# pm2 status
[PM]
  id  name  namespace  version  mode  pid  uptime  ⚡  status  cpu  mem  user  watching
  0  index  default    1.0.0   [ ]  19624  3m  0  online  0%  51.5mb root  disabled
[PM]# pm2 logs
[PM]# pm2 log last 15 lines: (change the value with --lines option)
[PM]# pm2 log last 15 lines:
[PM] 2025-03-19T21:55:34: PM Log: Node.js version          v16.20.2
[PM] 2025-03-19T21:55:34: PM Log: PM2 version           v1.0.0
[PM] 2025-03-19T21:55:34: PM Log: PM2 home              : /root/.pm2
[PM] 2025-03-19T21:55:34: PM Log: PM2 pid file          : /root/.pm2/.pm2.pid
[PM] 2025-03-19T21:55:34: PM Log: PM2 socket file        : /root/.pm2/rpc.sock
[PM] 2025-03-19T21:55:34: PM Log: PM2 socket file path  : /root/.pm2/pid.sock
[PM] 2025-03-19T21:55:34: PM Log: PM2 pid file path     : /root/.pm2/pid.log
[PM] 2025-03-19T21:55:34: PM Log: Worker Interval       : 30000
[PM] 2025-03-19T21:55:34: PM Log: Worker Max CPU Usage  : 100
[PM] 2025-03-19T21:55:34: PM Log: Concurrent Actions     : 2
[PM] 2025-03-19T21:55:34: PM Log: SIGHUP timeout        : 1000
[PM] 2025-03-19T21:55:34: PM Log: Max CPU Usage          : 100
[PM] 2025-03-19T21:55:34: PM Log: App (index:0) starting in `fork mode`-----[PM]# pm2 log last 15 lines:
[PM]# pm2 log last 15 lines:
[PM] 2025-03-19T21:55:34: PM Log: App (index:0) starting in http://localhost:4000
[PM]# pm2 log last 15 lines:
[PM] 2025-03-19T21:55:34: PM Log: App (index:0) online
[PM]
```

- Now type; **pm2 startup**

-Ensure that we save current configuration by typing ; **app-tier]# pm2 save**

- Verify that the application is running by executing the curl command ;
 - Curl `http://localhost: 4000/health`
 - It should return this is a health check.

```
[PM2] Remove init script via:  
$ pm2 unstartup systemd  
[root@ip-192-168-2-53 app-tier]# pm2 save  
[PM2] Saving current process list...  
[PM2] Successfully saved in /root/.pm2/dump.pm2  
[root@ip-192-168-2-53 app-tier]# curl http://localhost:4000/health  
"This is the health check"[root@ip-192-168-2-53 app-tier]#
```

-Recall in the 3tier architecture rendering we have a internal load balancer that connects web-tier to app-tier. So, we need to create one. Prior to load balancer create target groups first . Click the target group tab chose target type as instances and give it a name. Say **App-Internal-TG**.

-Protocol as HTTP and recall our application runs on port 4000.

-IP address type ; IPv4

-Choose VPC is the same we created at beginning of project

Screenshot of the AWS EC2 Target Groups 'Create target group' wizard Step 1: Specify group details.

Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

Basic configuration

Settings in this action can't be changed after the target group is created.

Choose a target type

Instance

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP address

- Supports load balancing to VPC endpoint resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice-based architectures, simplifying inter-application communication.
- Supports IP targets, making end-to-end TCP communications and the-SQS-MQ.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

Target group name

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol & Port

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation.

HTTP	80
------	----

IP address type

Targets with the indicated IP address type can be registered to this target group.

EIP

Each instance has a default network interface (eni) 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 (eni). [Learn more](#)

-Health HTTP only . Health check path enter ; /health

Screenshot of the AWS EC2 Target Groups 'Create target group' wizard Step 2: Register targets.

Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (1/1)

Instance ID	Name	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
i-0dd51b282110778c	AppTierInstance	Running	App-SG	us-east-1a	192.168.2.53	subnet-071bd9517e0391cb	March 16, 2025, 16:11 (UTC-04:00)

Ports for the selected instances

Ports for routing traffic to the selected instances.

4000
80
443
8080
8443
1-65535 (separate multiple ports with commas)

Review targets

Targets (0)

Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet ID	Launch time
No instances added yet								

Specify instances above, or leave the group empty if you prefer to add targets later.

0 pending

Create target group

-Chose the target by clicking Register instance as **AppTierInstance** . Click include as pending below and activate create target group.

EC2 > Target groups > App-Internal-TG

App-Internal-TG

Details

arn:aws:elasticloadbalancing:us-east-1:677276087515:targetgroup/App-Internal-TG/91a7a1f8befc2ab3

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
1	0	0	1	0	0

Distribution of targets by Availability Zone (AZ)

Registered targets (1) info

Anomaly mitigation: Not applicable | Deregister | Register targets

Instance ID	Name	Port	Zone	Health status	Health status details	Administrative override
i-045102821107785c	AppTInstance	4000	us-east-1a (eu...)	Unused	Target group is not co...	-

-As the image shows we have created the target group.

-Next create load balancer. We are going to chose application Load Balancer.

EC2 > Load balancers > Compare and select load balancer type

Compare and select load balancer type

A complete feature-by-feature comparison along with detailed highlights is also available. Learn more [\[?\]](#)

Load balancer types

- Application Load Balancer** [Info](#)

Choose an Application Load Balancer when you need to provide access to your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and validation features targeted at application architectures, including microservices and containers.

[Create](#)
- Network Load Balancer** [Info](#)

Choose a Network Load Balancer when you need to support multiple protocols at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

[Create](#)
- Gateway Load Balancer** [Info](#)

Choose a Gateway Load Balancer when you need to proxy and transform traffic from third-party virtual appliances that support FENEV. These appliances enable you to improve security, compliance, and policy controls.

[Create](#)

Classic Load Balancer - previous generation

[Close](#)

-We choose application load balancer . Provider a load balancer name ; **App-Internal-LB**

-Network mapping – here we choose vpc we created. **Yesu-vpc**. Also, make sure select the correct private Subnets. In my case I chose Yesu-vpc-App1-us-east-1a and Yesu-vpc-App2-us-east-1b.

-Recall we created our own internal group at the beginning of the project so select **internal-ALB-SG**

-For the listeners and routing direct protocol as HTTP , port 80 , default action ; **App-internal-TG**

-Click create load balancer.

-Now it is also important to note the DNS of the load balancer.

DNS name : internal-App-Internal-LB-680638924.us-east-1.elb.amazonaws.com

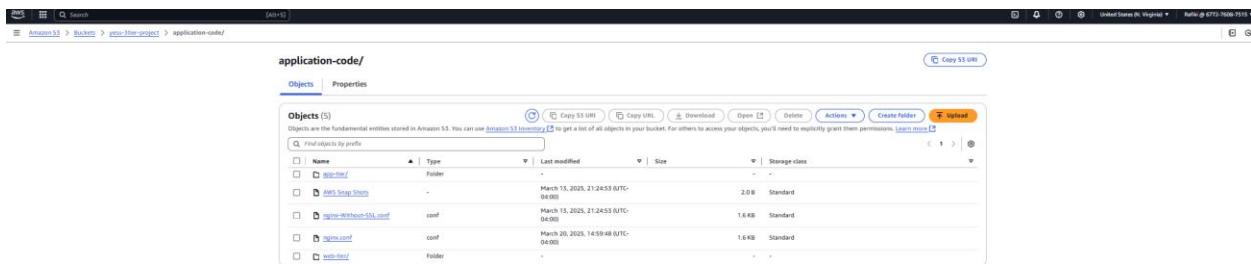
-So, with the DNS name we are going to configure the nginx.conf file. Open nginx.conf with code editor in my case is visual studio code. At the bottom of this file change the DNS name to name below and save file ;

internal-App-Internal-LB-680638924.us-east-1.elb.amazonaws.com.

The change should appear like this ;

```
#proxy for internal lb
location /api/ {
    proxy_pass http://internal-App-Internal-LB-680638924.us-east-
1.elb.amazonaws.com:80/;
}
```

-Also ,remember to upload the updated nginx.conf file to s3 bucket. Drag and drop file and the file the file will automatically update.



** IF you still the app-tier ec2 via SSM (session manager) . Do a health check on the host app-tier.

-Verify that the application is running by executing

```
App-tier ] # curl http://localhost:4000/health
```

```
[root@ip-192-168-2-53 app-tier]# curl http://localhost:4000/health
"This is the health check"[root@ip-192-168-2-53 app-tier]#
```

_With the message it means we have completed the app-tier configuration.

-Eventually we need to place this file in the webserver or Web-tier. Thus we have successfully created the app-tier resources.

8)

Next, we going to implement web-tier resource creation.

-So lets create ec2 instance for the web-tier. First , are going to intall node.js and nginx.

-

Here we will create **web-tier instance**

-select Amazon as your goto OS

-Then Amazon linux 2 AMI –SSD Volume type (free tier eligible)

-instance type - t2.micro

-No key pair. Proceed without key pair

Network settings

* In network setting make sure you edit and choose existing vpc we created at the beginning of the project **not default vpc**. In we recall the vpc we created is named Yesu-vpc.

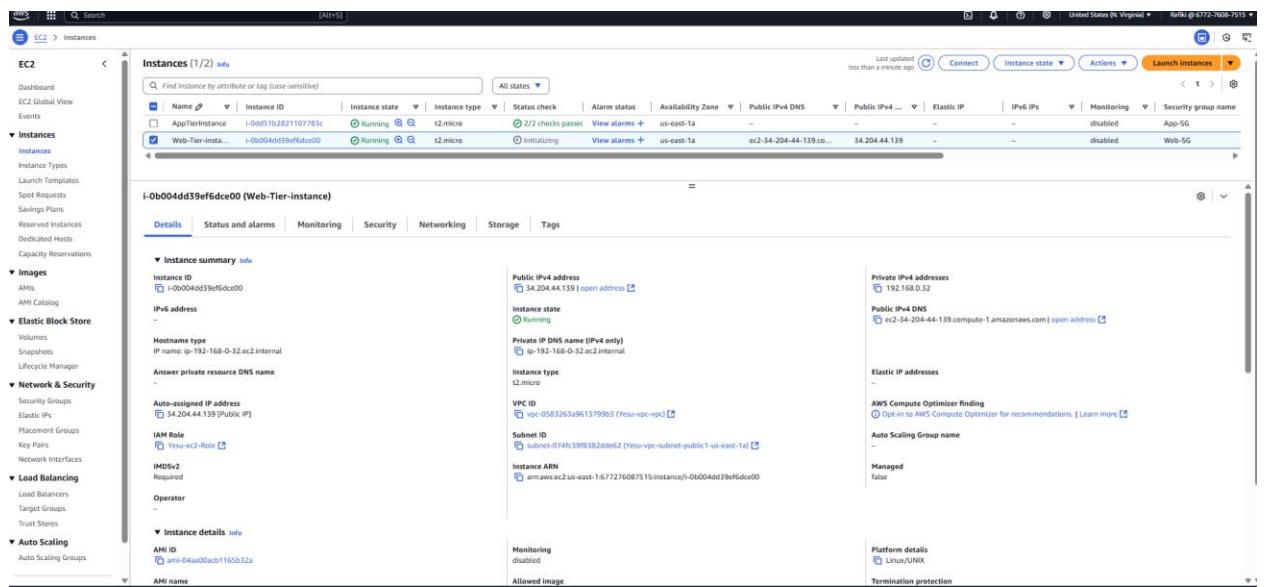
-Also, recall we also created 2 public subnets. Yesu-vpc-subnet-public1-us-east-1a and Yesu-vpc-subnet-public2-us-east-1a . We can choose either one.

-In addition enable assign public IP option too.

-Next select existing security group we created at the beginning of the project. In this case the security we created we named Web-SG (Web security group)

-In advanced configuration make we attach IAM role called **Yesu-ec2-Role**

See image below of the Web-Tier instance.



-Now connect ec2 instance via session manager

-At the terminal switch to root user.] \$ sudo -su ec2-user

```
sh-4.2$ sudo -su ec2-user
[ec2-user@ip-192-168-0-32 bin]$ 
[ec2-user@ip-192-168-0-32 bin]$ 
[ec2-user@ip-192-168-0-32 bin]$
```

-Switch to ec2-user HOME/ → type below command in terminal

] \$ cd /home/ec2-user

```
[ec2-user@ip-192-168-0-32 bin]$ cd /home/ec2-user
[ec2-user@ip-192-168-0-32 ~]$ pwd
/home/ec2-user
[ec2-user@ip-192-168-0-32 ~]$ █
```

-type pwd to ensure you are in the right location

-Now use curl command to install NMV source environment in the bash script. Then install NVM 16 and use it.

```
[ec2-user@ip-192-168-0-32 ~]$ curl -o- https://raw.githubusercontent.com/avizwayl/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  71117  0:--:-- --:--:--:--:-- 71076
=> 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" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-192-168-0-32 ~]$ source ~/.bashrc
[ec2-user@ip-192-168-0-32 ~]$ 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)
[ec2-user@ip-192-168-0-32 ~]$ use nvm 16
bash: use: command not found
[ec2-user@ip-192-168-0-32 ~]$ nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-192-168-0-32 ~]$ █
```

-Next we need to copy the contents from the s3 bucket suing the following script below ;

Buket name : yesu-3tier-project

aws s3 cp s3://yesu-3tier-project/application-code/web-tier/ web-tier –recursive

```
[ec2-user@ip-192-168-0-32 ~]$ aws s3 cp s3://yesu-3tier-project/application-code/web-tier/ web-tier --recursive
download: s3://yesu-3tier-project/application-code/web-tier/README.md to web-tier/README.md
download: s3://yesu-3tier-project/application-code/web-tier/src/components/DatabaseDemo/DatabaseDemo.js to web-tier/src/components/DatabaseDemo/DatabaseDemo.js
download: s3://yesu-3tier-project/application-code/web-tier/public/robots.txt to web-tier/public/robots.txt
download: s3://yesu-3tier-project/application-code/web-tier/public/index.html to web-tier/public/index.html
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/index.js to web-tier/src/components/Burger/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/index.js to web-tier/src/components/Menu/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Index.js to web-tier/src/components/Index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/Burger.js to web-tier/src/components/Burger/Burger.js
download: s3://yesu-3tier-project/application-code/web-tier/package.json to web-tier/package.json
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/Burger.styled.js to web-tier/src/components/Burger/Burger.styled.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/DatabaseDemo/DatabaseDemo.css to web-tier/src/components/DatabaseDemo/DatabaseDemo.css
download: s3://yesu-3tier-project/application-code/web-tier/src/global.js to web-tier/src/global.js
download: s3://yesu-3tier-project/application-code/web-tier/src/setupTests.js to web-tier/src/setupTests.js
download: s3://yesu-3tier-project/application-code/web-tier/src/theme.js to web-tier/src/theme.js
download: s3://yesu-3tier-project/application-code/web-tier/src/reportWebVitals.js to web-tier/src/reportWebVitals.js
download: s3://yesu-3tier-project/application-code/web-tier/src/index.js to web-tier/src/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/Menu.js to web-tier/src/components/Menu/Menu.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/Menu.styled.js to web-tier/src/components/Menu/Menu.styled.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Home/Home.js to web-tier/src/components/Home/Home.js
download: s3://yesu-3tier-project/application-code/web-tier/src/app.test.js to web-tier/src/App.test.js
download: s3://yesu-3tier-project/application-code/web-tier/src/hooks.js to web-tier/src/hooks.js
download: s3://yesu-3tier-project/application-code/web-tier/src/index.css to web-tier/src/index.css
download: s3://yesu-3tier-project/application-code/web-tier/src/App.css to web-tier/src/App.css
download: s3://yesu-3tier-project/application-code/web-tier/src/App.js to web-tier/src/App.js
download: s3://yesu-3tier-project/application-code/web-tier/src/assets/3TierArch.png to web-tier/src/assets/3TierArch.png
[ec2-user@ip-192-168-0-32 ~]$ █
```

To check that we created the web-tier folder with contents type the following command ;

```
~] $ ls
```

```
[ec2-user@ip-192-168-0-32 ~]$ ls  
web-tier
```

-Then switch to web-tier directory

```
[ec2-user@ip-192-168-0-32 ~]$ ls  
web-tier  
[ec2-user@ip-192-168-0-32 ~]$ cd web-tier/  
[ec2-user@ip-192-168-0-32 web-tier]$
```

```
Webtier]$ npm install
```

```
[ec2-user@ip-192-168-0-32 web-tier]$ npm install  
npm WARN deprecated es6-promise@4.2.5: Please use @jridgewell/sourcemap-coder instead  
npm WARN deprecated rollup-plugin-treror@0.2: This package has been deprecated and is no longer maintained. Please use #rollup/plugin-treror  
npm WARN deprecated sourcemap-coded@1.4.3: Please use @jridgewell/sourcemap-coder instead  
npm WARN deprecated rollup-plugin-treror@0.2: This package has been deprecated and is no longer maintained. Please use #rollup/plugin-treror  
npm WARN deprecated workbox-google-analytics@4.0.0: It is not compatible with newer versions of GA starting with v4, as long as you are using gav1 it should be ok, but the package is not longer being maintained  
npm WARN deprecated stabled@0.1.8: Modern JS already guarantees ArraySort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#Browser_compatibility  
npm WARN deprecated pfil@5.5.1: You or someone you depend on is using Q, the JavaScript Promise library that gave JavaScript developers strong feelings about promises. They can almost certainly migrate to the native JavaScript promise now. Thank you literally everyone for joining me in this her against the odds. Be excellent to each other.  
npm WARN deprecated  
npm WARN deprecated  
npm WARN deprecated (For a CAPIP with native promises, see #node/evenual-wend and #node/captpr)  
npm WARN deprecated Wic-hr-time@0.0.2: Use your platform's native performance.now() and performance.timeOrigin.  
npm WARN deprecated domexception@2.0.11: Use your platform's native DOMException instead  
npm WARN deprecated  
npm WARN deprecated  
npm WARN deprecated infinites@0.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerful.  
npm WARN deprecated glob@7.2.3: Glob versions prior to v5 are no longer supported  
npm WARN deprecated  
npm WARN deprecated browserify@0.12.1: Browserify versions prior to v0.13.0 are no longer supported  
npm WARN deprecated browserify-shim@3.5.0: Use browserify-shim@4.0.0 or browserify@0.13.0 instead  
npm WARN deprecated HumanizeCode@config-array@0.13.0: Use eslint/config-array instead  
npm WARN deprecated  
added 1505 packages, and audited 1506 packages in 53s  
774 packages are looking for funding  
  run 'npm fund' for details  
  vulnerabilities (2 moderate, 4 high)  
To address all issues (including breaking changes), run:  
  npm audit fix --force  
Run 'npm audit' for details.  
npm WARN New major version of tpm available! 0.19.4 -> 11.3.0  
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0  
npm notice Run npm install -g npm@11.3.0 to update!  
npm notice
```

```
Webtier]$ npm run build --- > thype this command to run it.
```

```
[ec2-user@ip-192-168-0-32 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:

 74.93 kB  build/static/js/main.4dff5ef7.js
  1.79 kB  build/static/js/453.a4ec9c9e.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.

Find out more about deployment here:

  https://cra.link/deployment
```

-Next we have to install nginx server using the following command ;
 Web-tier] \$ sudo amazon-linux-extras install nginx1 -y

-Once install update the nginx configuration. First switch to nginx path
 Web-tier]\$ cd /etc/nginx --> Use the ls command and find file called nginx.conf.
 -In is important that we remove this file and update with the new one from the s3 bucket.
 -Use the following command to remove it

Nginx] \$ sudo rm nginx.conf.

Then use the below command to update with the new file.

] \$ sudo aws s3 cp s3://yesu-3tier-project/application-code/nginx.conf . <.....Make sure in the end of the path there is a period or dot.

```
ec2-user@ip-192-168-0-32 web-tier$ cd /etc/nginx
ec2-user@ip-192-168-0-32 nginx$ ls
fastcgi_params  fastcgi.conf.default  fastcgi_params.default  koi-utf  koi-win  mime.types  mime.types.default  nginx.conf  nginx.conf.default  cgi_params  cgi_params.default  uwsgi_params  uwsgi_params.default  win-utf
ec2-user@ip-192-168-0-32 nginx$ sudo rm nginx.conf
ec2-user@ip-192-168-0-32 nginx$ sudo aws s3 cp s3://yesu-3tier-project/application-code/nginx.conf .
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:
  aws help
  aws <command> help
  aws <command> <subcommand> help
aws: error: too many arguments
ec2-user@ip-192-168-0-32 nginx$ sudo aws s3 cp s3://yesu-3tier-project/application-code/nginx.conf .
Download: s3://yesu-3tier-project/application-code/nginx.conf to ./nginx.conf
```

- Now lets restart nginx
- Nginx] \$ sudo service nginx restart

Give permission to read , write an execute

Nginx] \$ chmod -R 755 /home/ec2-user

-Now use the below following command to turn on nginx.

Nginx] \$ sudo chkconfig nginx on

- To check the output of the App, we can check using the Web-Tier-Instance public IP. But before checking lets open port no 80 with http, Anywhere IPv4, 0.0.0.0/0 --> Save rules ----> Now paste the pubic ip of Web-Tier-Instance in new tab of browser ----> You will see the app ----> Enter the data in the app

-Now go to web-tier instance select and click security tab. Click on security group link.

sg-00efb062a73293f27

-Edit inbound rules. Click add rule → protocol HTTP , anywhere IP , IP 0.0.0.0/0

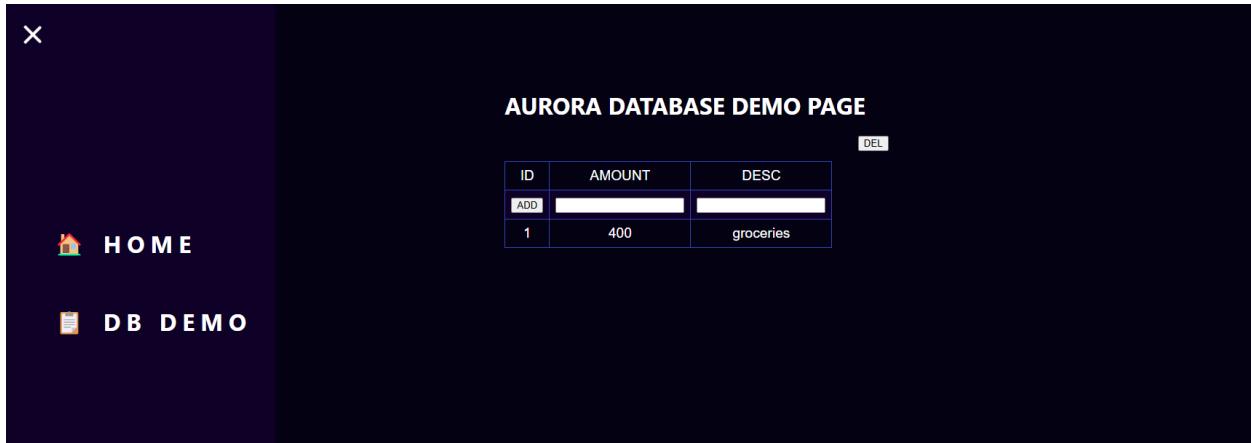
Inbound security group rules successfully modified on security group (sg-00efb062a73293f27 | Web-SG) > Details

sg-00efb062a73293f27 - Web-SG

Actions ▾

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0c55d98df15383c	IPv4	HTTP	TCP	80	192.168.0.0/22	-
-	sgr-07c22f7ed70d4965	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-0de6af94d075a5784	-	HTTP	TCP	80	54.229.67.64 (ec2-54-229-67-64.eu-central-1.compute.amazonaws.com)	-

-Next select instance copy the IP address paste it on the internet.



- Here our high availability fault tolerance 3-tier project is successfully complete as demonstrated.
- In the real work or production environment we do not share IP address to the end user. Therefore, for security we have to create our own domain. This we can do by using route 53 in aws.
- So the first thing is create an external load balancer. Here also, it is not recommended to give the DNS address to the end user. So, map the dns name of the load balancer to route 53 custom domain.

