

AWS 3-TIER HIGH AVAILABILITY ARCHITECTURE ON AWS

Author -Praveen P

Complete Infrastructure Deployment Guide

Architecture: Multi-Tier Web Application on AWS

Availability: Multi-AZ (Availability Zone 1a & 1b)

Web Tier Stack: Nginx + Node.js (React Frontend)

App Tier Stack: Nginx + MySQL (Backend / Database Proxy)

Database: Amazon RDS (Multi-AZ, Primary + Backup)

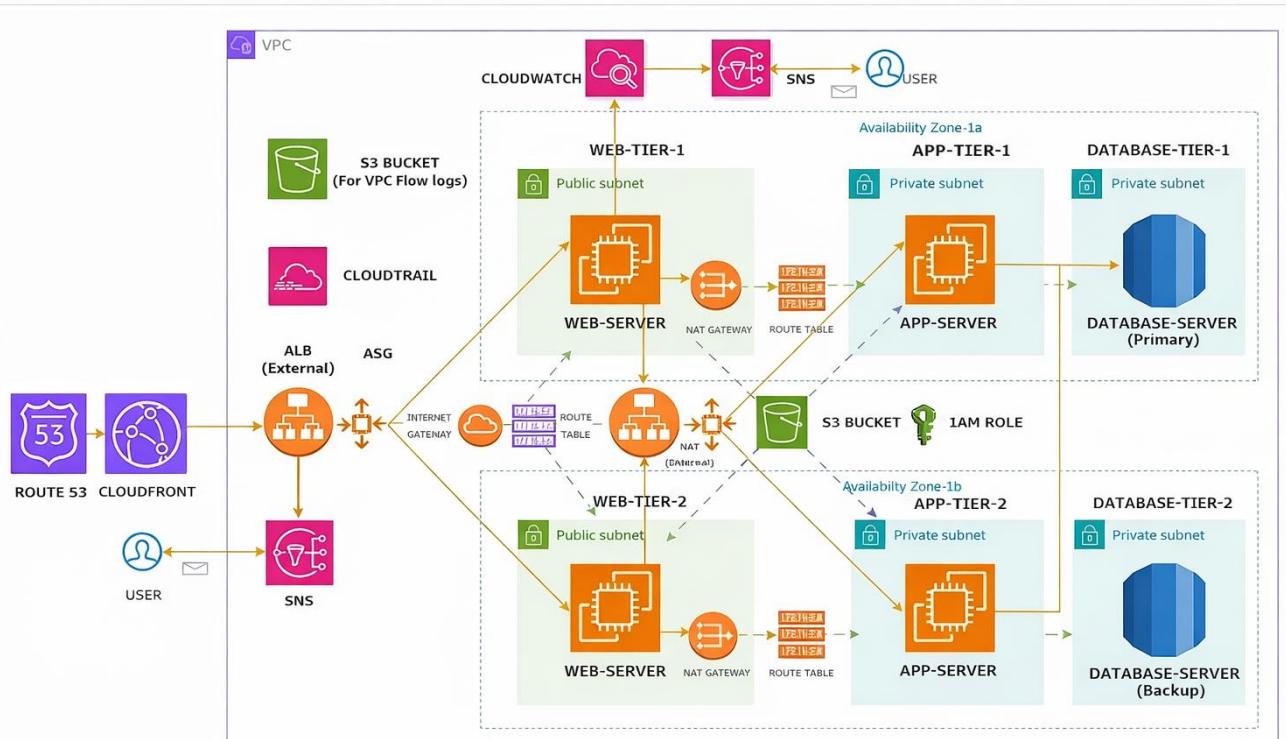


Table of Contents

1. Project Overview
2. Architecture Diagram & Component Breakdown
3. Prerequisites
4. Step 1 — Download Code to Local System
5. Step 2 — Create S3 Buckets & Upload Code
6. Step 3 — Create IAM Role
7. Step 4 — Create VPC, Subnets, IGW, NAT Gateway & Route Tables
8. Step 5 — Create Security Groups
9. Step 6 — Create RDS (Database Tier)
10. Step 7 — Create App Server (App Tier)
11. Step 8 — Create Web Server (Web Tier)
12. Step 9 — Create ALB DNS Record in Route 53
13. Step 10 — Create CloudWatch Alarms with SNS
14. Step 11 — Create CloudTrail
15. Verification & Testing
16. Troubleshooting Tips

1. Project Overview

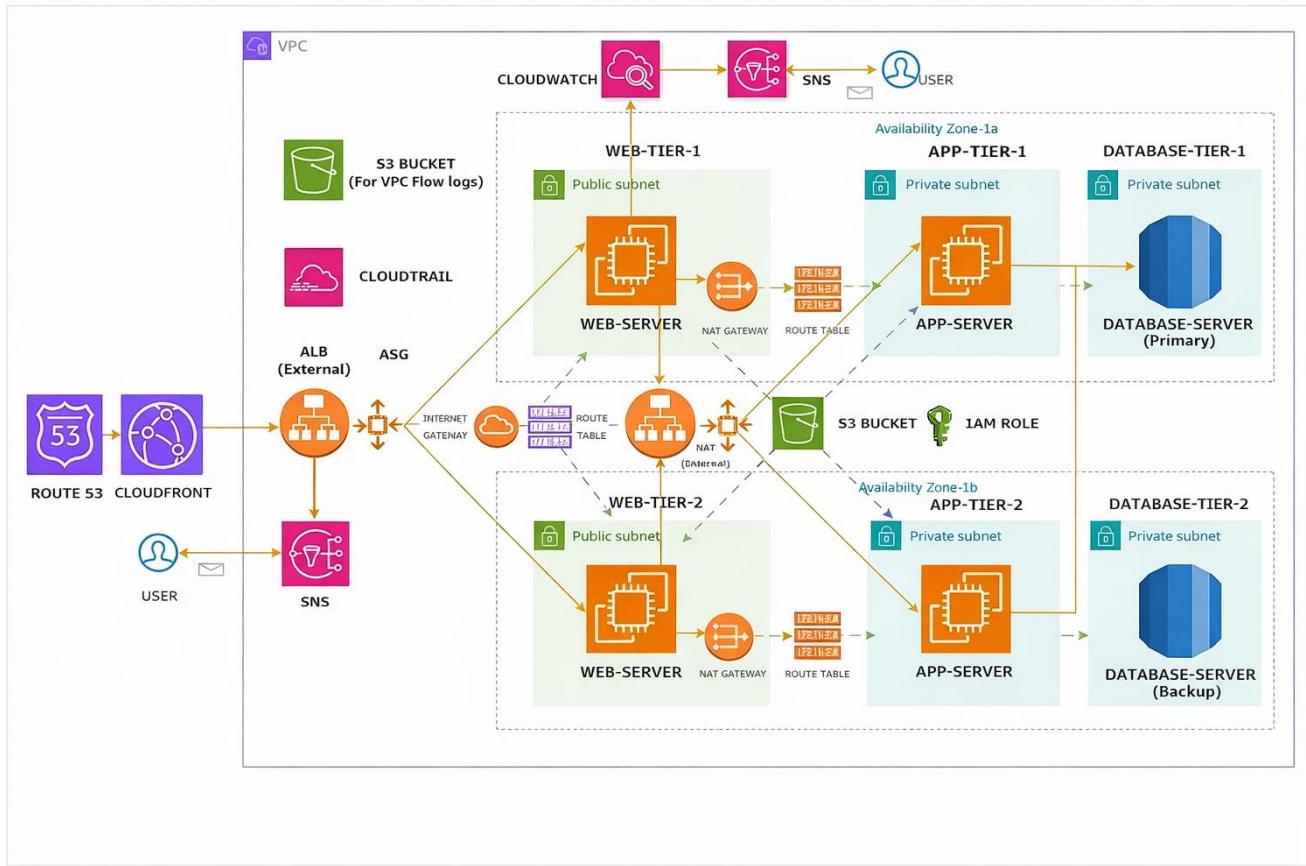
This document is a step-by-step guide to deploy a production-grade, highly available three-tier web application on Amazon Web Services (AWS). The infrastructure is designed with fault tolerance, auto-scaling, monitoring, and security best practices in mind.

Architecture Summary

The deployment spans two Availability Zones (AZ-1a and AZ-1b) within a single AWS VPC. Traffic flows from end users through Route 53 and CloudFront to an External Application Load Balancer, then through the Web Tier, App Tier, and finally to a Multi-AZ RDS database.

Tier	Technology	Subnet Type	Availability
Web Tier	Nginx + Node.js (React)	Public Subnet	AZ-1a & AZ-1b
App Tier	Nginx + MySQL	Private Subnet	AZ-1a & AZ-1b
Database Tier	Amazon RDS MySQL (Multi-AZ)	Private Subnet	Primary + Backup

2. Architecture Diagram & Component Breakdown



Traffic Flow

The following describes how a user request flows through the infrastructure:

1. User sends a DNS request. Route 53 resolves the domain and directs traffic to CloudFront.
2. CloudFront forwards the request to the External ALB (Application Load Balancer).
3. The External ALB distributes traffic to the Web Tier (Web-Server in AZ-1a or AZ-1b).
4. The Web Server (Nginx + Node.js) handles the frontend and forwards API calls to the Internal ALB.
5. The Internal ALB routes API traffic to the App Tier (App-Server in AZ-1a or AZ-1b).
6. The App Server connects to RDS (Primary) in the Database Tier to read/write data.
7. If the Primary RDS fails, RDS automatically fails over to the Backup instance in the other AZ.

Supporting Services

- S3 Bucket (x2): One for application code, one for VPC Flow Logs
- CloudTrail: Audit logging for all AWS API calls
- CloudWatch + SNS: Real-time monitoring and alerting to end user via email/SMS
- NAT Gateway: Allows private subnet instances to initiate outbound internet connections
- IAM Role: Grants EC2 instances permission to access S3 (S3ReadOnly + SSM Managed Instance Core)

3. Prerequisites

Ensure the following are in place before beginning the deployment:

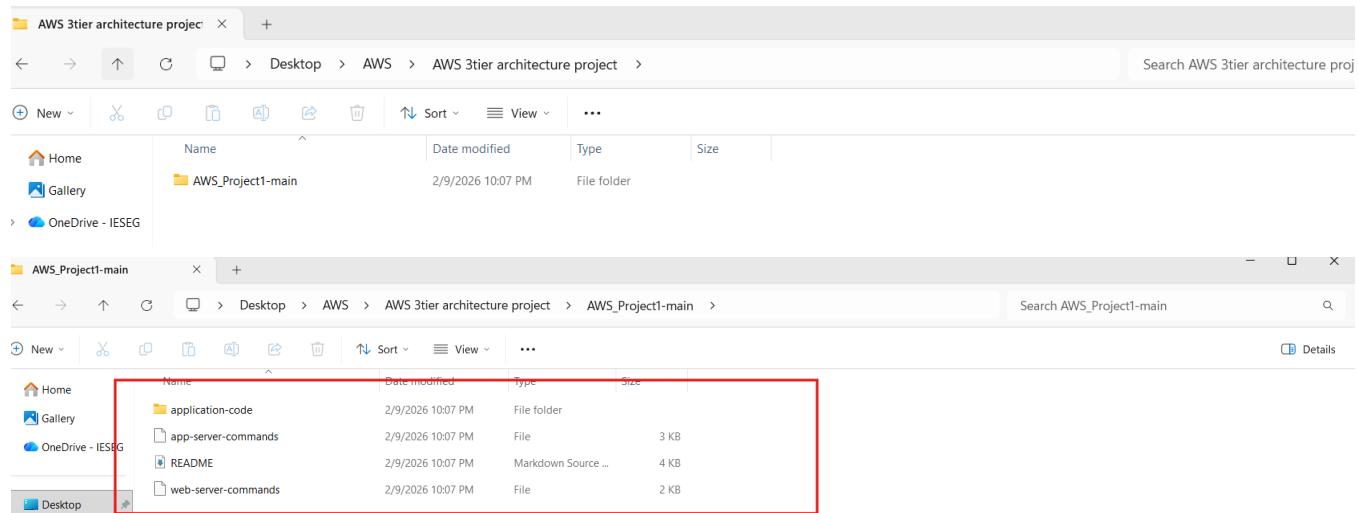
- An active AWS account with sufficient IAM permissions (AdministratorAccess recommended for initial setup)
- AWS CLI installed and configured on your local machine
- Git installed on your local machine
- Basic understanding of AWS services: EC2, VPC, RDS, ALB, Route 53
- A registered domain name (for Route 53 DNS configuration)
- Application source code ready in your local repository

Region Note:

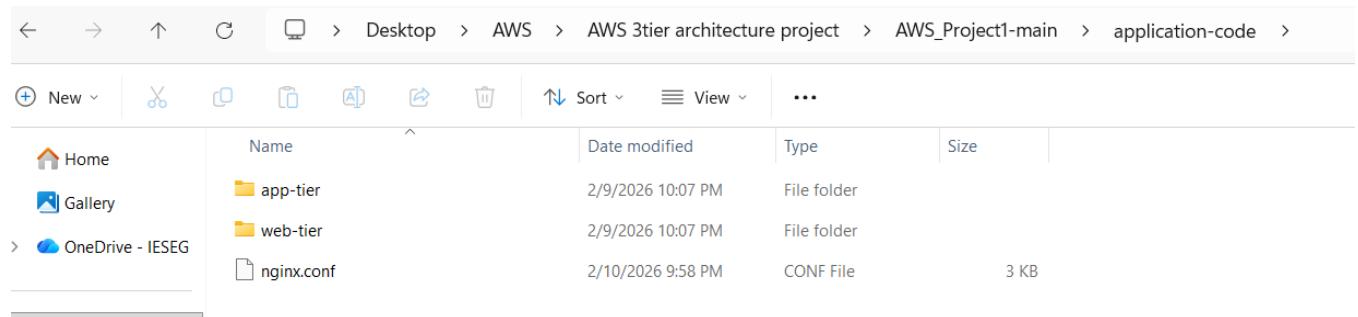
All resources in this guide are created in a single AWS region. Choose a region that supports all required services (e.g., us-east-1, us-west-2, ap-south-1). The VPC CIDR used in this guide is 172.16.0.0/16.

4. Step 1 — Download Folder from github to Local System

Begin by cloning or downloading the application source code to your local machine. This code will later be uploaded to S3 and deployed to the EC2 instances.



Note: Make sure both the web-tier code (React/Node.js) and app-tier code are present in separate folders before proceeding.



5. Step 2 — Create S3 Buckets & Upload Code

Two S3 buckets are required: one for storing the application code, and one that will be used for VPC Flow Logs.

5.1 Create the Application Code Bucket

8. Go to AWS Console → S3 → Create Bucket
9. Enter a unique bucket name (e.g., my-app-code-bucket-2024)
10. Select your target AWS Region
11. Keep all other settings as default → Click Create Bucket, and upload the downloaded folder to the s3 bucket

The screenshot shows the AWS S3 console under the 'General purpose buckets' tab. It lists two buckets: 'myvpc-flowgogs2026' and 'praveens3bucket-2026'. Both buckets were created in the 'Europe (Paris) eu-west-3' region on February 10, 2026, at 19:38:58 (UTC+01:00) and 19:28:13 (UTC+01:00) respectively. The 'Create bucket' button is visible at the top right of the list.

Name	AWS Region	Creation date
myvpc-flowgogs2026	Europe (Paris) eu-west-3	February 10, 2026, 19:38:58 (UTC+01:00)
praveens3bucket-2026	Europe (Paris) eu-west-3	February 10, 2026, 19:28:13 (UTC+01:00)

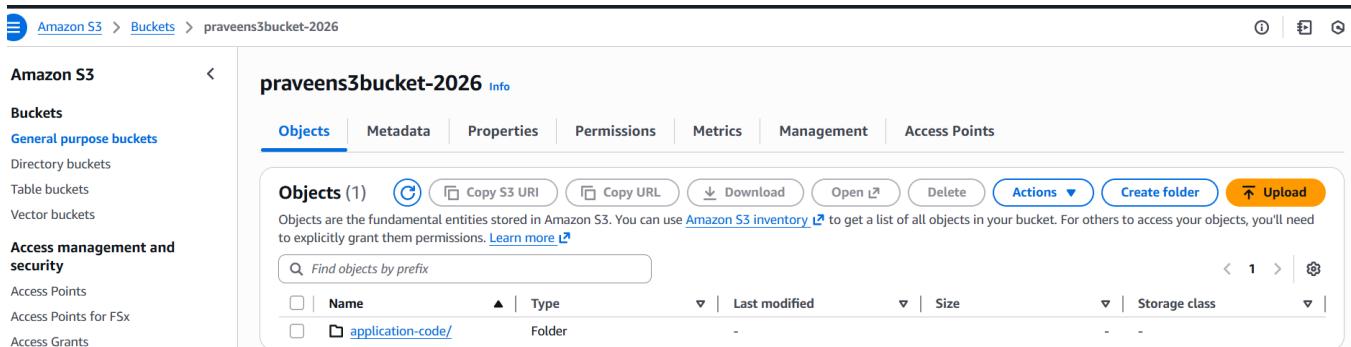
5.2 Create the VPC Flow Logs Bucket

12. Go to AWS Console → S3 → Create Another Bucket to store VPC flow logs
13. Enter a unique bucket name (e.g., my-vpc-flowlogs-bucket-2024)
14. Select the same AWS Region
15. Click Create Bucket

The screenshot shows the AWS S3 console under the 'General purpose buckets' tab. It lists the same two buckets: 'myvpc-flowgogs2026' and 'praveens3bucket-2026'. The interface includes a 'Create bucket' button, a search bar, and sorting options for Name, AWS Region, and Creation date. On the right side, there are three informational boxes: 'Account snapshot' (updated daily), 'External access summary' (updated daily), and 'Storage Lens' which provides visibility into storage usage and activity trends.

5.3 Upload Application Code to S3

16. Open the application code bucket
17. Click Upload → Add Files/Folder
18. Upload both the web-tier and app-tier code folders
19. Click Upload



The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'Amazon S3', 'Buckets', 'General purpose buckets', 'Table buckets', 'Vector buckets', 'Access management and security', 'Access Points', 'Access Points for FSx', and 'Access Grants'. The main area is titled 'praveens3bucket-2026' with a 'Info' link. Below the title, there are tabs for 'Objects', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is selected. It displays one object: a folder named 'application-code/'. There are buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload'. A search bar says 'Find objects by prefix' and a pagination indicator shows '1'. The table below lists the object with columns for Name, Type, Last modified, Size, and Storage class.

Name	Type	Last modified	Size	Storage class
application-code/	Folder	-	-	-

6. Step 3 — Create IAM Role

An IAM Role must be created and attached to EC2 instances so they can securely access S3 (to pull code) and be managed via AWS Systems Manager (SSM) without requiring direct SSH key access.

20. Navigate to AWS Console → IAM → Roles → Create Role

21. Select Trusted entity type: AWS service

22. Select Use case: EC2 → Click Next

23. Attach the following policies:

- AmazonS3ReadOnlyAccess
- AmazonSSMManagedInstanceCore

24. Click Next → Enter Role Name (e.g., ec2-3tier-role)

25. Click Create Role

The screenshot shows the 'Roles' list page in the AWS IAM console. There are two roles listed: 'abcd-ec2-s3-ssm-role-922026' and 'AWSServiceRoleForAutoScaling'. Both roles are associated with the 'ec2' AWS Service and were last active 9 minutes ago. The 'Create role' button is highlighted with a red arrow pointing to it.

Role name	Trusted entities	Last activity
abcd-ec2-s3-ssm-role-922026	AWS Service: ec2	9 minutes ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Li...)	9 minutes ago

The screenshot shows the 'Permissions policies' list page in the AWS IAM console. There are two policies attached: 'AmazonS3ReadOnlyAccess' and 'AmazonSSMManagedInstanceCore'. Both are AWS managed policies and are attached to one entity. The 'Add permissions' button is highlighted with a red arrow pointing to it.

Policy name	Type	Attached entities
AmazonS3ReadOnlyAccess	AWS managed	1
AmazonSSMManagedInstanceCore	AWS managed	1

Why SSM Managed Instance Core?

This policy enables AWS Systems Manager Session Manager, which allows you to connect to EC2 instances securely through the AWS Console without needing SSH keys or open port 22. This is a security best practice.

7. Step 4 — Create VPC, Subnets, IGW, NAT Gateway & Route Tables

7.1 Create VPC

26. Go to AWS Console → VPC → Your VPCs → Create VPC
27. Select VPC Only (not VPC and more)
28. Enter a name (e.g., 3tier-vpc)
29. IPv4 CIDR: 172.16.0.0/16
30. Leave IPv6 CIDR as No IPv6 CIDR block
31. Tenancy: Default → Click Create VPC

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

- Your VPCs (1/2) Info:** Shows 1 VPC listed.
- Actions:** Includes a blue 'Edit' button, a green 'Actions' dropdown, and an orange 'Create VPC' button.
- Table Headers:** Name, VPC ID, State, Encryption c..., Encryption control..., Block Public..., IPv...
- Table Data:** A single row for 'my-vpc' with VPC ID 'vpc-054821ecd4a7c1b36', State 'Available', and Block Public Access 'Off'.
- VPC Details:**
 - Details Tab:** Shows VPC ID 'vpc-054821ecd4a7c1b36 / my-vpc'.
 - Resource map, CIDs, Flow logs, Tags, Integrations:** These tabs are visible but not expanded.
 - Details Section:** Contains the following information:

VPC ID: vpc-054821ecd4a7c1b36	State: Available	Block Public Access: Off	DNS hostnames: Disabled
DNS resolution: Enabled	Tenancy: default	DHCP option set: dopt-05a6e189010d9cd77	Main route table: rtb-03fa30068d8543bf6
Main network ACL: acl-0aa969f1c87cb7cbc	Default VPC: No	IPv4 CIDR: 10.0.0.0/16	IPv6 pool: -

7.2 Create Subnets

Choose a cidr and configure .Create the following 6 subnets across two Availability Zones: Example :

Subnet Name	Availability Zone	Type	CIDR Block
web-tier-public-1a	AZ-1a	Public	172.16.0.0/24
web-tier-public-1b	AZ-1b	Public	172.16.1.0/24
app-tier-private-1a	AZ-1a	Private	172.16.2.0/24
app-tier-private-1b	AZ-1b	Private	172.16.3.0/24
db-tier-private-1a	AZ-1a	Private	172.16.4.0/24
db-tier-private-1b	AZ-1b	Private	172.16.5.0/24

Subnets (6) Info						Last updated 2 minutes ago	Actions	Create subnet
<input type="checkbox"/>	Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR		
<input type="checkbox"/>	WEB-TIER-2	subnet-035382dd84301efa6	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.4.0/24		
<input type="checkbox"/>	WEB-TIER-1	subnet-07ba5d5b758a9f009	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.1.0/24		
<input type="checkbox"/>	DB-TIER-1	subnet-0e60e7c5439267e68	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.3.0/24		
<input type="checkbox"/>	DB-TIER-2	subnet-041e29e9477ce1275	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.6.0/24		
<input type="checkbox"/>	APP-TIER-1	subnet-0ff6be5d1869188c5	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.2.0/24		
<input type="checkbox"/>	APP-TIER-2	subnet-0bcc19fe3e7c8d8b3	Available	vpc-054821ecd4a7c1b36 my-vpc	<input type="radio"/> Off	10.0.5.0/24		

7.3 Create Internet Gateway (IGW)

32. Go to VPC → Internet Gateways → Create Internet Gateway
33. Name: 3tier-igw → Create
34. Select the new IGW → Actions → Attach to VPC → Select 3tier-vpc → Attach

The screenshot shows the AWS VPC Internet Gateways page. A new Internet Gateway has been created and is displayed with the following details:

- Internet gateway ID: igw-0800d2f9449626c11
- State: Attached
- VPC ID: vpc-054821ecd4a7c1b36 | my-vpc
- Owner: 864946423844

The Tags section shows one tag: Name = my-igw.

7.4 Create NAT Gateway

35. Go to VPC → NAT Gateways → Create NAT Gateway
36. Subnet: Select web-tier-public-1a (public subnet)
37. Connectivity type: Public
38. Click Allocate Elastic IP → Create NAT Gateway
39. Wait for Status to become Available (takes ~1-2 minutes), repeat the process for another public . we need two nat gateway to this architecture

NAT gateways (2) Info						Last updated 2 minutes ago	Actions	Create NAT gateway
<input type="checkbox"/>	Name	NAT gateway ID	Connectivity...	State	State message	Availability...	Route table ID	F
<input type="checkbox"/>	NAT-GATEWAY-2	nat-0697970600c1d50b4	Public	Available	-	Zonal	-	1
<input type="checkbox"/>	NAT-1	nat-0dead67630033553f	Public	Available	-	Zonal	-	1

7.5 Create Route Tables

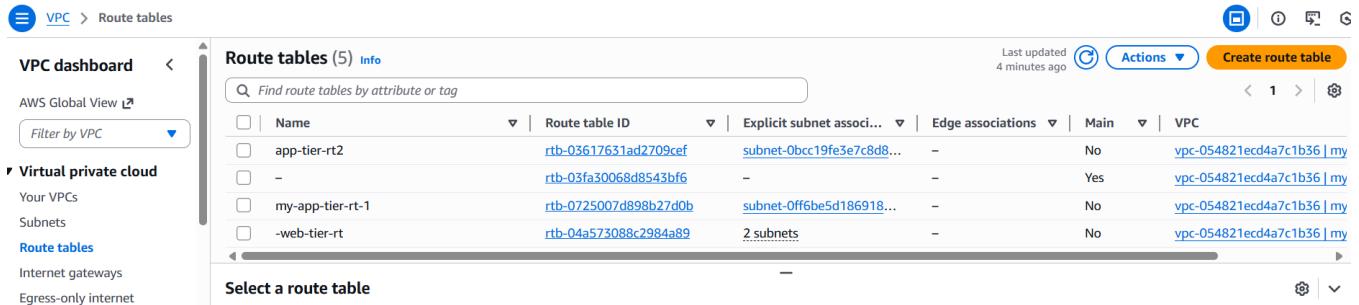
You need two route tables: a Public Route Table (for web tier) and a Private Route Table (for app and database tiers).

Public Route Table (Web Tier):

40. VPC → Route Tables → Create Route Table → Name: public-rt, VPC: 3tier-vpc
41. Select the route table → Routes → Edit Routes → Add Route: 0.0.0.0/0 → Target: Internet Gateway → Save
42. Subnet Associations → Edit Subnet Associations → Select both public subnets → Save

Private Route Table (App & DB Tiers):

43. Create Route Table → Name: private-rt, VPC: 3tier-vpc
44. Edit Routes → Add Route: 0.0.0.0/0 → Target: NAT Gateway → Save
45. Subnet Associations → Select all 4 private subnets → Save



The screenshot shows the AWS VPC Route Tables page. On the left, there's a navigation sidebar with 'VPC dashboard' selected. Under 'Virtual private cloud', 'Route tables' is also selected. The main area displays a table titled 'Route tables (5)'. The table has columns for Name, Route table ID, Explicit subnet associ..., Edge associations, Main, and VPC. The rows show the following data:

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
app-tier-rt2	rtb-03617631ad2709cef	subnet-0bcc19fe3e7c8d8...	-	No	vpc-054821ecd4a7c1b36 my
-	rtb-03fa30068d8543bf6	-	-	Yes	vpc-054821ecd4a7c1b36 my
my-app-tier-rt-1	rtb-0725007d898b27d0b	subnet-0ff6be5d186918...	-	No	vpc-054821ecd4a7c1b36 my
-web-tier-rt	rtb-04a573088c2984a89	2 subnets	-	No	vpc-054821ecd4a7c1b36 my

Below the table, a message says 'Select a route table'.

8. Step 5 — Create Security Groups

Security groups act as virtual firewalls. Create the following 5 security groups in your VPC, in the order listed below (as later groups reference earlier ones).

Security Group Name	Source	Port	Purpose
external-load-balancer-sg	0.0.0.0/0 (Internet)	HTTP 80	Allow all internet traffic to External ALB
web-tier-sg	external-load-balancer-sg	HTTP 80	Allow traffic from Ext ALB to Web Servers
internal-load-balancer-sg	web-tier-sg	HTTP 80	Allow traffic from Web Servers to Int ALB
app-tier-sg	internal-load-balancer-sg	Port 4000	Allow traffic from Int ALB to App Servers
db-tier-sg	app-tier-sg	MySQL 3306	Allow App Servers to connect to RDS

How to Create Each Security Group

46. Go to EC2 Console → Security Groups → Create Security Group
47. Enter the Security Group name and description
48. Select VPC: 3tier-vpc
49. Under Inbound Rules → Add Rule → Select Type (HTTP/Custom TCP/MySQL) → Set Source → Save

The screenshot shows the AWS EC2 Security Groups interface. On the left, there's a sidebar with 'Virtual private cloud' navigation. The main area displays a list of security groups. One group, 'external-alb', is selected and highlighted with a red box. Below the list, a detailed view of 'sg-00dd17e38e273b851 - external-alb' is shown. The 'Inbound rules' tab is active, showing a single rule: 'HTTP (TCP port 80)' from '0.0.0.0/0'. This row is also highlighted with a red box.

<input checked="" type="checkbox"/> web-sg	sg-052356bf8d1ee3dac	web-sg	vpc-054821ecd4a7c1b36	allow																		
<input type="checkbox"/> -	sg-096749605eccdd3a1	default	vpc-054821ecd4a7c1b36	default VPC secur																		
sg-052356bf8d1ee3dac - web-sg																						
Details Inbound rules Outbound rules Sharing VPC associations Related resources - new Tags																						
Inbound rules (2) <table border="1"> <thead> <tr> <th>IP version</th> <th>Type</th> <th>Protocol</th> <th>Port range</th> <th>Source</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>-</td> <td>HTTP</td> <td>TCP</td> <td>80</td> <td>sg-00dd17e38e273b85...</td> <td>-</td> </tr> <tr> <td>IPv4</td> <td>HTTP</td> <td>TCP</td> <td>80</td> <td>86.192.221.76/32</td> <td>-</td> </tr> </tbody> </table>					IP version	Type	Protocol	Port range	Source	Description	-	HTTP	TCP	80	sg-00dd17e38e273b85...	-	IPv4	HTTP	TCP	80	86.192.221.76/32	-
IP version	Type	Protocol	Port range	Source	Description																	
-	HTTP	TCP	80	sg-00dd17e38e273b85...	-																	
IPv4	HTTP	TCP	80	86.192.221.76/32	-																	

Create the security for all five by mentioned ports, Now you will have five security group

<input type="checkbox"/> app-tier-sg	sg-033d90182ede6507f	app-tier-sg	vpc-054821ecd4a7c1b36	all from internal
<input type="checkbox"/> INTERNAL-ALB	sg-02590c9a5c93809c1	INTERNAL-ALB	vpc-054821ecd4a7c1b36	allow all
<input type="checkbox"/> web-sg	sg-052356bf8d1ee3dac	web-sg	vpc-054821ecd4a7c1b36	allow
<input type="checkbox"/> -	sg-096749605eccdd3a1	default	vpc-054821ecd4a7c1b36	default VPC secur
<input type="checkbox"/> -	sg-00d0aa4798c3def3d	default	vpc-0b5a3d9d66bd264a6	default VPC secur
<input type="checkbox"/> DB-tier	sg-0bd7199505c306c48	DB-tier	vpc-054821ecd4a7c1b36	all from app-tier
<input type="checkbox"/> external-alb	sg-00dd17e38e273b851	external-alb	vpc-054821ecd4a7c1b36	allow-all

9. Step 6 — Create RDS (Database Tier)

9.1 Create DB Subnet Group

50. Go to RDS Console → Subnet Groups → Create DB Subnet Group
51. Name: 3tier-db-subnet-group
52. VPC: 3tier-vpc
53. Availability Zones: Select AZ-1a and AZ-1b
54. Subnets: Select db-tier-private-1a and db-tier-private-1b → Create

Aurora and RDS > Subnet groups

Subnet groups (1)

Name	Description	Status	VPC
my-rds-subnet	my-rds-subnet	Complete	vpc-054821ecd4a7c1b36

Create DB subnet group

ny-rds-subnet

Subnet group details

VPC ID: vpc-054821ecd4a7c1b36

ARN: arn:aws:rds:eu-west-3:864946423844:subgrp:my-rds-subnet

Supported network types: IPv4

Description: my-rds-subnet

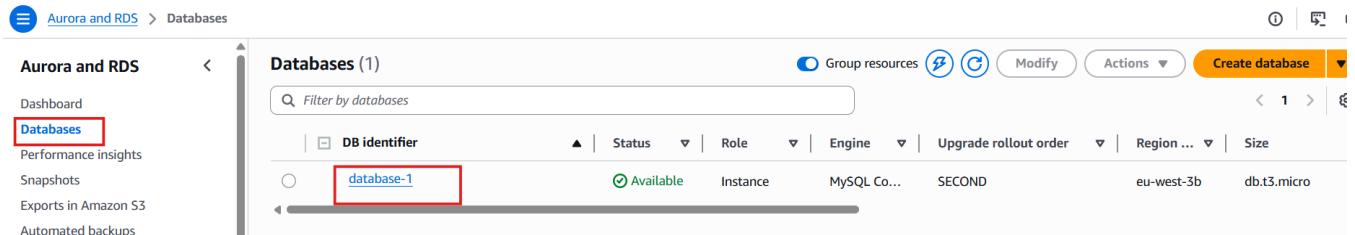
Subnets (2)

Availability zone	Subnet name	Subnet ID	CIDR block
eu-west-3a	DB-TIER-1	subnet-0e60e7c5439267e68	10.0.3.0/24
eu-west-3b	DB-TIER-2	subnet-041e29e9477ce1275	10.0.6.0/24

9.2 Create RDS Instance (Multi-AZ)

55. RDS Console → Databases → Create Database
56. Engine: MySQL, Version: (latest stable)
57. Template: Production (for Multi-AZ)
58. DB Instance Class: db.t3.micro (or appropriate tier)
59. Storage: 20 GiB (gp2), Enable storage autoscaling
60. Multi-AZ: Yes (creates Primary in AZ-1a, Backup/Standby in AZ-1b)
61. VPC: 3tier-vpc
62. Subnet Group: 3tier-db-subnet-group
63. Public Access: No

64. Security Group: db-tier-sg
65. Database Authentication: Password authentication
66. Set Master username and password (save these securely!)
67. Initial Database Name: appdb (or your preferred name)
68. Click Create Database and wait ~10-15 minutes for it to become available



The screenshot shows the 'Databases' section of the AWS Aurora and RDS console. On the left, there's a sidebar with links like 'Dashboard', 'Performance insights', 'Schemas', 'Exports in Amazon S3', and 'Automated backups'. The 'Databases' link is highlighted with a red box. The main area has a header 'Databases (1)' with a search bar and filter options. A table lists one database entry: 'database-1', which is 'Available', an 'Instance', using 'MySQL Co...', 'SECOND' upgrade rollout order, in 'eu-west-3b' region, and 'db.t3.micro' size. The 'DB identifier' column is highlighted with a red box.

Note: Note the RDS endpoint URL — you will need this when configuring the App Server's database connection.

10. Step 7 — Create App Server (App Tier)

The App Tier is responsible for handling business logic and database communication. It uses Nginx as a reverse proxy and MySQL client to connect to RDS.

10.1 Launch Test App Server

69. EC2 Console → Launch Instance
70. Name: test-app-server
71. AMI: Amazon Linux 2023 (or Amazon Linux 2)
72. Instance type: t2.micro
73. Key pair: (Select existing or create new)
74. VPC: 3tier-vpc, Subnet: app-tier-private-1a
75. Auto-assign public IP: Disable
76. Security Group: app-tier-sg
77. IAM Instance Profile: ec2-3tier-role
78. Launch Instance

EC2 > Instances > Launch an instance

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

[Take a walkthrough](#) [Do not show me this message again.](#)

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

Test-APP-server

Add additional tags

Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.10... [read more](#)

ami-030ebd4d4694126d2

10.2 Connect via SSM Session Manager

79. Select the instance in EC2 Console
80. Click Connect → Session Manager → Connect

EC2 > Instances > i-03755329805e934fc > Connect to instance

Connect [Info](#)

Connect to an instance using the browser-based client.

Session Manager

EC2 Instance Connect | **Session Manager** | SSH client | EC2 serial console

Session Manager usage:

- 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.

Cancel **Connect**

10.3 Install Node.js on App Server

```
# Switch to root
sudo su

# Update packages
yum update -y

# Install Node.js (using NVM - Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
source ~/.bashrc
nvm install 16
nvm use 16

# Verify Node.js installation
node -v
npm -v
```



Follow the comments in the file “web-server-commands”

10.4 Install MySQL Client on App Server

```
# Install MySQL client (to connect to RDS)
yum install mysql -y

# Test connection to RDS (replace with your RDS endpoint)
mysql -h <RDS-ENDPOINT> -u <DB-USERNAME> -p

# If successful, you'll see MySQL prompt
# Type 'exit' to quit
```

10.5 Download and Configure App Code from S3

```
# Create app directory
mkdir -p /home/ec2-user/app-tier
cd /home/ec2-user/app-tier

# Download code from S3
aws s3 cp s3://<YOUR-CODE-BUCKET>/app-tier/ . --recursive

# Install npm dependencies
npm install

# Configure database connection in your config file
# Edit the DB config to point to your RDS endpoint
```

10.6 Start the App Server

```
# Install PM2 process manager to keep app running
npm install -g pm2

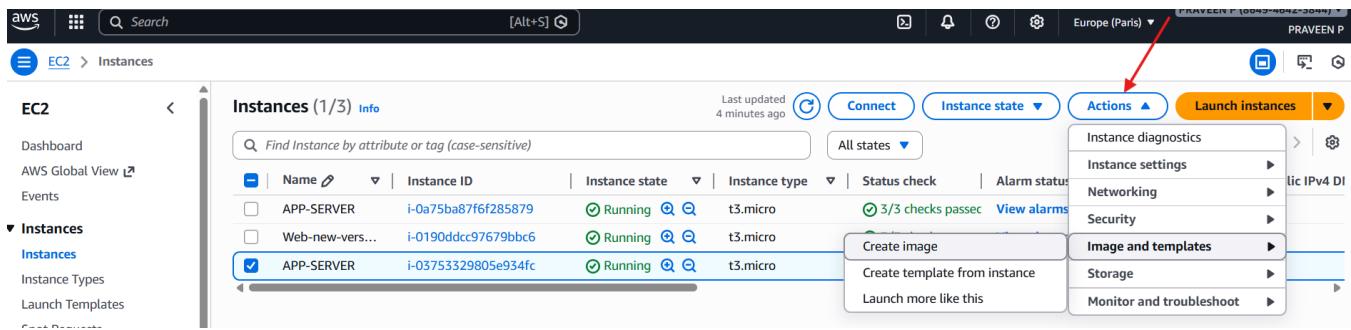
# Start app server on port 4000
pm2 start index.js

# Save PM2 config and set to restart on reboot
pm2 save
pm2 startup

# Verify app is running
pm2 list
curl http://localhost:4000/health
```

10.7 Create AMI from App Server

81. EC2 Console → Select test-app-server → Actions → Image and Templates → Create Image
82. Image Name: app-tier-ami
83. No reboot: Check (to avoid downtime) → Create Image
84. Wait for AMI status to become Available (EC2 → AMIs)



Amazon Machine Images (AMIs) (1/2) [Info](#)

Owned by me [▼](#)

[Actions ▾](#) [Launch instance from AMI](#)

Name	AMI name	AMI ID	Source	Owner	Visibility
app-serve-image	app-serve-image	ami-0ba0493e2a6d5acc1	864946423844/app-serve-image	864946423844	Private
<input checked="" type="checkbox"/> Web-server-IM...	WEB-SERVER-AMI	ami-0fd9b8b3c207ef38d	864946423844/WEB-SERVER-AMI	864946423844	Private

AMI ID: ami-0fd9b8b3c207ef38d (Web-server-IMAGE)

[Details](#) [Permissions](#) [Storage](#) [My AMI usage](#) [AMI ancestry - new](#) [Tags](#)

AMI ID ami-0fd9b8b3c207ef38d	Image type machine	Platform details Linux/UNIX	Root device type EBS
AMI name WEB-SERVER-AMI	Owner account ID 864946423844	Architecture x86_64	Usage operation RunInstances
Root device name /dev/xvda	Status Available	Source 864946423844/WEB-SERVER-AMI	Virtualization type hvm
Boot mode uefi-preferred	State reason -	Creation date 2026-02-10T21:49:20.000Z	Kernel ID -

10.8 Create App Tier Target Group

85. EC2 Console → Target Groups → Create Target Group
86. Target type: Instances
87. Target group name: app-tier-tg
88. Protocol: HTTP, Port: 4000
89. VPC: 3tier-vpc
90. Health check path: /health → Create Target Group

EC2 > Target groups

Target groups (1/2) [Info](#) | [What's new?](#)

[Actions ▾](#) [Create target group](#)

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
<input checked="" type="checkbox"/> app-targets	arn:aws:elasticloadbalancing:us-east-1:864946423844:targetgroup/app-targets/53f0a0a04a3a4a3a	4000	HTTP	Instance	Internalloadbalancer	vpc-05482
<input type="checkbox"/> web-server-tg	arn:aws:elasticloadbalancing:us-east-1:864946423844:targetgroup/web-server-tg/53f0a0a04a3a4a3a	80	HTTP	Instance	EXTERNAL-LAOD-BALA...	vpc-05482

10.9 Create Internal Load Balancer (ALB)

91. EC2 Console → Load Balancers → Create Load Balancer → Application Load Balancer
92. Name: internal-alb
93. Scheme: Internal
94. VPC: 3tier-vpc
95. Subnets: app-tier-private-1a and app-tier-private-1b
96. Security Group: internal-load-balancer-sg
97. Listener: HTTP:80 → Forward to: app-tier-tg → Create

The screenshot shows the AWS Lambda console with the function 'HelloWorld' selected. The 'Handler' dropdown is set to 'lambda.lambda_handler'. The 'Runtime' dropdown is set to 'Python 3.8'. The 'Memory size' input field is set to '128'. The 'Timeout' input field is set to '3'. The 'Environment' section shows 'Environment variables' with 'AWS_LAMBDA_FUNCTION_NAME' set to 'HelloWorld'. The 'Tracing' dropdown is set to 'None'. The 'Logs' tab is selected, showing log entries for the function's execution.

10.10 Create App Tier Launch Template

98. EC2 Console → Launch Templates → Create Launch Template
99. Name: app-tier-lt
100. AMI: Select app-tier-ami (created above)
101. Instance type: t2.micro
102. Security Group: app-tier-sg
103. IAM Instance Profile: ec2-3tier-role
104. Advanced Details → User Data: Add startup script to launch app on boot

```
#!/bin/bash
# User Data script for App Server
cd /home/ec2-user/app-tier
pm2 start index.js
pm2 save
```

The screenshot shows the AWS Lambda console with the function 'HelloWorld' selected. The 'Handler' dropdown is set to 'lambda.lambda_handler'. The 'Runtime' dropdown is set to 'Python 3.8'. The 'Memory size' input field is set to '128'. The 'Timeout' input field is set to '3'. The 'Environment' section shows 'Environment variables' with 'AWS_LAMBDA_FUNCTION_NAME' set to 'HelloWorld'. The 'Tracing' dropdown is set to 'None'. The 'Logs' tab is selected, showing log entries for the function's execution.

10.11 Create App Tier Auto Scaling Group (ASG)

105. EC2 Console → Auto Scaling Groups → Create Auto Scaling Group
106. Name: app-tier-asg
107. Launch Template: app-tier-lt
108. VPC and Subnets: app-tier-private-1a, app-tier-private-1b
109. Load Balancing: Attach to existing load balancer → Choose app-tier-tg

110. Desired: 2, Minimum: 2, Maximum: 4 (adjust based on your needs)
111. Scaling Policies: Target tracking — CPU Utilization 70% → Create

The screenshot shows the AWS Auto Scaling Groups page. On the left, there's a sidebar with navigation links like EC2 Manager, Auto Scaling, and others. The main area displays a table of Auto Scaling groups:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
web-server-newversion	web-server-launch-template Version Default	1	-	1	1	4
APP-SEVER	app-server Version Default	2	-	2	1	4

A red box highlights the "APP-SEVER" row. Below the table, a modal window titled "Auto Scaling group: APP-SEVER" is open, showing the "Details" tab. It contains the following information:

Desired capacity	Scaling limits	Desired capacity type	Status
2	1 - 4	Units (number of instances)	-

Date created: Tue Feb 10 2026 22:18:05 GMT+0100 (Central European Standard Time)

11. Step 8 — Create Web Server (Web Tier)

The Web Tier serves the React frontend and proxies API calls to the App Tier through the Internal ALB. It uses Nginx as a reverse proxy and Node.js to serve the React application.

11.1 Launch Test Web Server

112. EC2 Console → Launch Instance
113. Name: test-web-server
114. AMI: Amazon Linux 2023
115. Instance type: t2.micro
116. VPC: 3tier-vpc, Subnet: web-tier-public-1a
117. Auto-assign public IP: Enable
118. Security Group: web-tier-sg
119. IAM Instance Profile: ec2-3tier-role → Launch

11.2 Install Nginx on Web Server

```
# Connect via SSM Session Manager
sudo su
yum update -y

# Install Nginx
yum install nginx -y

# Start and enable Nginx
systemctl start nginx
systemctl enable nginx

# Verify Nginx is running
systemctl status nginx
```

11.3 Install Node.js on Web Server

```
# Install NVM and Node.js
curl -o https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
source ~/.bashrc
nvm install 16
nvm use 16

# Verify installation
node -v
npm -v
```

11.4 Download Web Code from S3 and Build React

```
# Create web directory
mkdir -p /home/ec2-user/web-tier
cd /home/ec2-user/web-tier

# Download from S3
aws s3 cp s3://<YOUR-CODE-BUCKET>/web-tier/ . --recursive

# Install dependencies
npm install

# Build the React app for production
npm run build
```

11.5 Configure Nginx

Edit the Nginx configuration to serve the React app and proxy API requests to the Internal ALB:

```
# Edit Nginx config
nano /etc/nginx/nginx.conf

# Or create a new server block:
nano /etc/nginx/conf.d/app.conf
```

Nginx configuration to serve React and proxy API calls to Internal ALB:

```
server {
    listen 80;
    server_name _;

    # Serve the React build folder
    root /home/ec2-user/web-tier/build;
    index index.html;

    # For React Router support
    location / {
        try_files $uri $uri/ /index.html;
    }

    # Proxy API requests to Internal ALB
    location /api/ {
        proxy_pass http://<INTERNAL-ALB-DNS>;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

# Test Nginx configuration
nginx -t

# Reload Nginx to apply changes
systemctl reload nginx
```

11.6 Create AMI from Web Server

120. EC2 → Select test-web-server → Actions → Image and Templates → Create Image
121. Image Name: web-tier-ami → Create Image → Wait for Available status

The screenshot shows the AWS EC2 console with the 'AMIs' tab selected. On the left, there's a sidebar with 'EC2' and 'Instances' sections. The main area is titled 'Amazon Machine Images (AMIs) (1/2)' and shows a table with two rows. The first row is for 'app-serve-image' with AMI ID 'ami-0ba0493e2a6d5acc1'. The second row is for 'Web-server-AMI' with AMI ID 'ami-0fd9b8b3c207ef38d'. Both rows have columns for 'Name', 'AMI name', 'AMI ID', 'Source', 'Owner', and 'Visibility'. The 'Web-server-AMI' row has its 'Name' and 'AMI name' fields highlighted.

11.7 Create Web Tier Target Group

122. EC2 Console → Target Groups → Create Target Group
123. Target type: Instances, Name: web-tier-tg
124. Protocol: HTTP, Port: 80, VPC: 3tier-vpc
125. Health check path: / → Create

The screenshot shows the AWS EC2 Target Groups page. At the top, it says 'Target groups (1/2)'. Below is a table with two rows. The first row is for 'app-targets' and the second is for 'web-server-tg'. The 'web-server-tg' row is selected, indicated by a blue border. The columns are 'Name', 'ARN', 'Port', 'Protocol', 'Target type', 'Load balancer', and 'VPC ID'. The 'web-server-tg' row has its 'Name' and 'ARN' fields highlighted.

The screenshot shows the details for the 'web-server-tg' target group. At the top, it says 'Target group: web-server-tg'. Below is a navigation bar with tabs: Details, Targets, Monitoring, Health checks, Attributes, and Tags. The 'Details' tab is selected. The main area shows a table with several fields. The 'Target type' field is set to 'Instance'. The 'Protocol : Port' field shows 'HTTP: 80'. The 'Protocol version' field is 'HTTP1'. The 'VPC' field shows 'vpc-054821ecd4a7c1b36'. Other fields include 'IP address type' (IPv4), 'Load balancer' (EXTERNAL-LAOD-BALAN...), and 'ARN' (arn:aws:elasticloadbalancing:eu-west-3:864946423844:targetgroup/web-server-tg/00ce304fd892c257).

11.8 Create External Load Balancer

126. EC2 Console → Load Balancers → Create → Application Load Balancer
127. Name: external-alb, Scheme: Internet-facing
128. VPC: 3tier-vpc, Subnets: web-tier-public-1a and web-tier-public-1b
129. Security Group: external-load-balancer-sg
130. Listener: HTTP:80 → Forward to: web-tier-tg → Create

Load balancers (1/2) What's new?

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	State	Type	Scheme	IP address type	VPC ID	Availability Zones
INTERNALLOADBALANCER	Active	application	Internal	IPv4	vpc-054821ecd4a7c1b36	2 Availability Zones
EXTERNAL-LAOD-BALANCER	Active	application	Internet-facing	IPv4	vpc-054821ecd4a7c1b36	2 Availability Zones

Load balancer: EXTERNAL-LAOD-BALANCER

Details

Load balancer type Application	Status Active	VPC vpc-054821ecd4a7c1b36	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z3Q77PNBQS71R4	Availability Zones subnet-035382dd84301efa6 eu-west-3b (euw3-az2) subnet-07ba5d5b758a9f009 eu-west-3a (euw3-az1)	Date created February 10, 2026, 23:07 (UTC+01:00)

11.9 Create Web Tier Launch Template

131. Launch Templates → Create → Name: web-tier-lt
132. AMI: web-tier-ami, Instance type: t2.micro
133. Security Group: web-tier-sg, IAM Profile: ec2-3tier-role

11.10 Create Web Tier Auto Scaling Group

134. Auto Scaling Groups → Create → Name: web-tier-asg
135. Launch Template: web-tier-lt
136. Subnets: web-tier-public-1a, web-tier-public-1b
137. Load Balancing: Attach to web-tier-tg
138. Desired: 2, Min: 2, Max: 4 → Create

Auto Scaling groups (1/2) Info

Last updated 3 minutes ago

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max
web-server-newversion	web-server-launch-template Version Default	1	-	1	1	4
APP-SEVER	app-server Version Default	2	-	2	1	4

Auto Scaling group: web-server-newversion

Capacity overview

Desired capacity 1	Scaling limits 1 - 4	Desired capacity type Units (number of instances)	Status -
-----------------------	-------------------------	--	-------------

12. Step 9 — Create ALB DNS Record in Route 53

Point your domain to the External ALB by creating an Alias A record in Route 53.

139. Go to Route 53 Console → Hosted Zones → Select your domain
140. Click Create Record
141. Record type: A (IPv4)
142. Toggle Alias to ON
143. Route traffic to: Alias to Application and Classic Load Balancer
144. Region: Select your deployment region
145. Choose the External ALB from the dropdown
146. Routing Policy: Simple → Create Record

The screenshot shows the AWS Route 53 console. On the left, there's a navigation sidebar with 'Route 53' selected. Below it are links for 'Dashboard', 'Hosted zones' (which is also selected), 'Health checks', and 'Profiles'. Under 'Global Resolver', there's a link for 'Global resolvers'. The main content area has a header 'Hosted zones (1)'. It includes a search bar with placeholder 'Filter records by property or value' and several buttons: 'View details', 'Edit', 'Delete', and a yellow 'Create hosted zone' button. Below the header is a table with columns: Hosted zone name, Type, Created by, Record count, Description, and Hosted zone ID. The table contains one row for 'praveen7477.com', which is Public, created by 'Route 53', has a record count of 3, and a long Hosted zone ID.

Hosted zone name	Type	Created by	Record count	Description	Hosted zone ID
praveen7477.com	Public	Route 53	3	-	Z04984932BLKP51...

DNS Propagation Note:

DNS changes can take up to 60 seconds to propagate. Test by navigating to your domain in a browser after a short wait.

13. Step 10 — Create CloudWatch Alarms with SNS

Set up monitoring and alerting to be notified when instances experience high CPU usage or health check failures.

13.1 Create SNS Topic

147. Go to SNS Console → Topics → Create Topic
148. Type: Standard, Name: 3tier-alerts
149. Create Topic
150. Click the topic → Create Subscription
151. Protocol: Email, Endpoint: your-email@example.com → Create Subscription
152. Check your email and confirm the subscription

The screenshot shows the Amazon SNS console with the 'Topics' tab selected. A blue banner at the top says 'New Feature: Amazon SNS now supports High Throughput FIFO topics. Learn more.' Below it, a table lists three topics:

Name	Type	ARN
APP-SERVER-NOTIFICATION	Standard	arn:aws:sns:eu-west-3:864946423844:APP-SERVER-...
Cloud-watch-notification	Standard	arn:aws:sns:eu-west-3:864946423844:Cloud-watch-...
WEB-SERVER-NOTIFICATION	Standard	arn:aws:sns:eu-west-3:864946423844:WEB-SERVER-...

13.2 Create CloudWatch Alarms

153. Go to CloudWatch Console → Alarms → Create Alarm
154. Select metric: EC2 → By Auto Scaling Group → CPUUtilization
155. Select the web-tier-asg metric → Next
156. Conditions: Greater than 80% for 2 consecutive periods
157. Notification: Send to 3tier-alerts SNS topic
158. Alarm Name: web-tier-high-cpu → Create Alarm
159. Repeat for app-tier-asg CPU alarm

13.3 Configure VPC Flow Logs

160. Go to VPC Console → Select 3tier-vpc → Flow Logs tab → Create Flow Log
161. Filter: All
162. Destination: Send to Amazon S3 bucket
163. S3 Bucket ARN: arn:aws:s3:::<YOUR-FLOWLOGS-BUCKET>
164. Click Create Flow Log

14. Step 11 — Create CloudTrail

CloudTrail records all AWS API activity in your account, providing a complete audit trail for security and compliance.

165. Go to CloudTrail Console → Trails → Create Trail
166. Trail name: 3tier-cloudtrail
167. Storage location: Create new S3 bucket (or use existing logs bucket)
168. Log file SSE-KMS encryption: Optional (recommended for production)
169. Enable CloudWatch Logs: Yes → Create new log group: 3tier-cloudtrail-logs
170. Management events: Read and Write
171. Data events: Optional (S3 and Lambda)
172. Insights: Optional
173. Click Create Trail

Important:

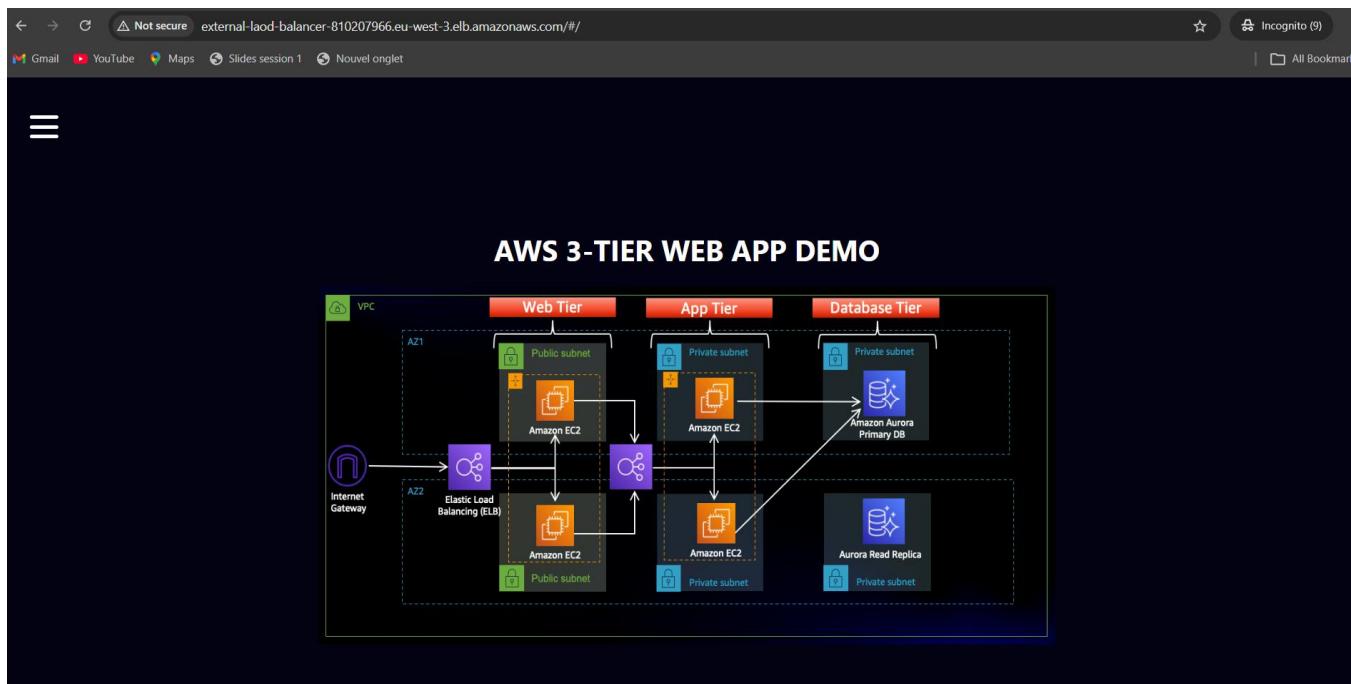
CloudTrail records every API call (Console, CLI, SDK). This is essential for auditing who did what and when. Store logs in a secure S3 bucket with restricted access.

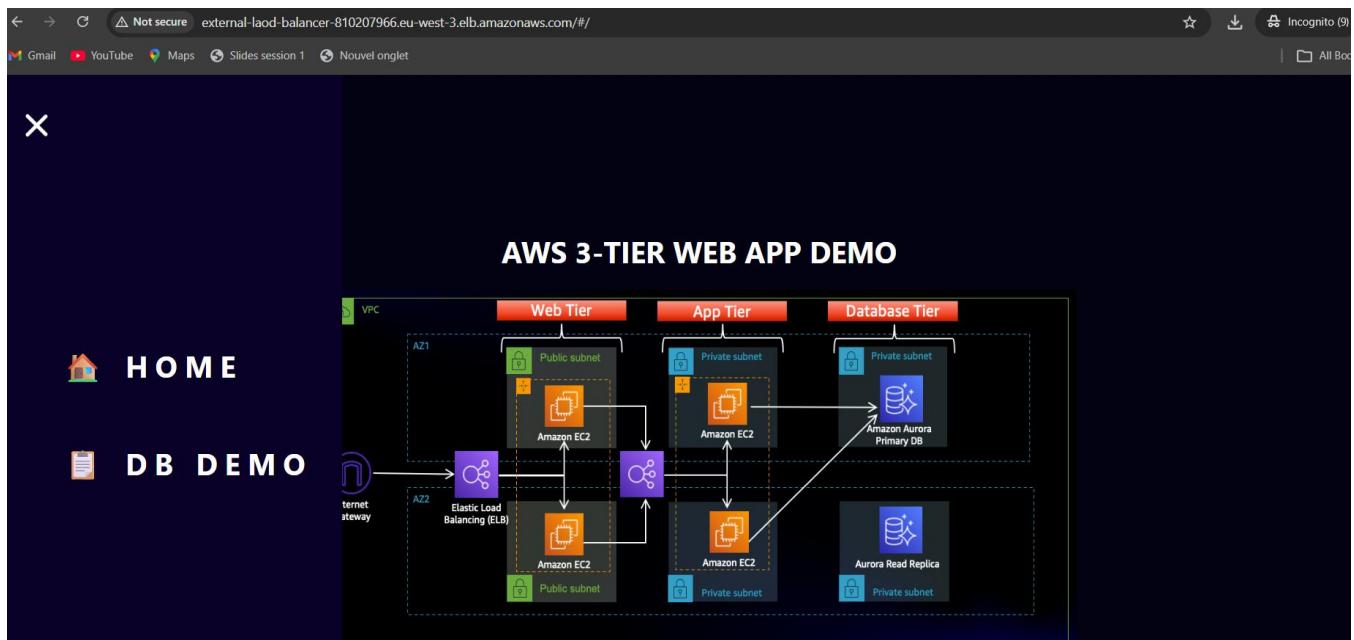
15. Verification & Testing

After completing all steps, perform the following checks to validate the deployment:

End-to-End Test Checklist

- Open your domain in a browser — the React application should load
- Verify HTTPS (if SSL was configured via ACM and ALB listener)
- Test API functionality — form submissions, data retrieval should work
- Verify data is being written to and read from RDS correctly
- Check EC2 Console — both ASGs should show healthy instances in each AZ
- Check Target Group health — all registered instances should be Healthy
- Simulate high load — verify ASG scales out correctly
- Test Multi-AZ failover — RDS failover should complete within 60 seconds
- Verify CloudWatch alarms trigger and SNS email is received
- Check CloudTrail logs are appearing in S3





16. Troubleshooting Tips

Issue	Solution
Target Group targets showing Unhealthy	Check security group rules. Ensure the ALB SG is allowed as a source in the instance SG on the correct port.
Cannot connect to RDS from App Server	Verify db-tier-sg allows MySQL port 3306 from app-tier-sg. Check RDS subnet group includes the correct private subnets.
NAT Gateway not routing traffic	Ensure private route table has a route 0.0.0.0/0 pointing to the NAT Gateway. Check NAT GW is in a public subnet.
SSM Session Manager not connecting	Confirm ec2-3tier-role is attached to the instance. Verify AmazonSSMManagedInstanceCore policy is included.
React app showing blank page	Check Nginx configuration root path. Ensure 'npm run build' completed successfully and build folder exists.
CloudWatch alarm not triggering	Verify the metric namespace and dimension are correctly set to the ASG name. Confirm SNS subscription is confirmed.

— *End of Document* —

AWS 3-Tier High Availability Architecture — Infrastructure Deployment Guide