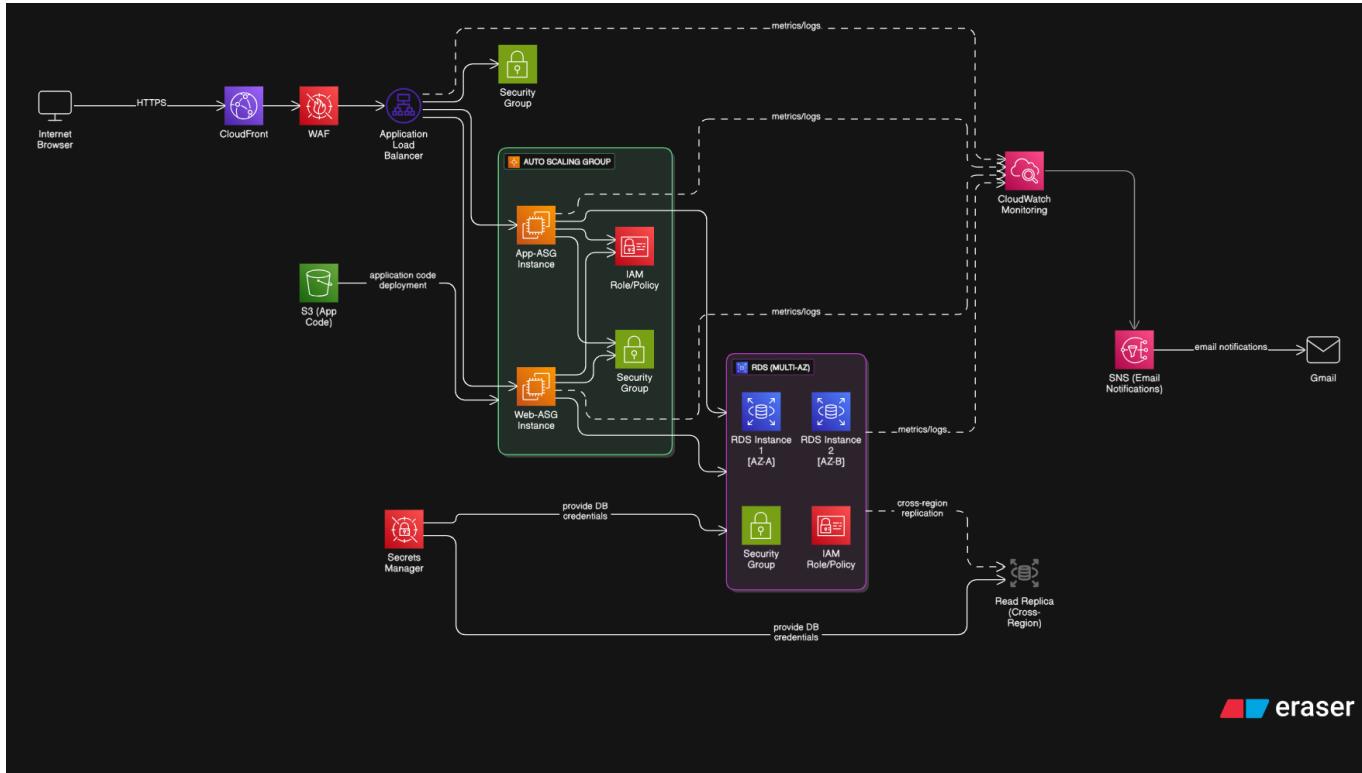


# Three Tier 15 Microservices Application

In this project I will be showing you step-by-step how I built and deployed a three tier application which had 15 microservices on AWS EC2 with Multi-AZ RDS instances.



This project is by **Harish Shetty** and it has a lot of AWS services used as you can see in the project diagram. There are EC2 instances in an Auto-Scaling Group which has the **Application Code** and the **RDS Database** also has two instances for **High-Availability** with its credentials stored in a **Secrets Manager**.

## Step-by-Step Project Guide:

- Deploying the infrastructure
- Configuring the Jump Servers
- Createing Jump Servers AMIs
- Creating a CloudTrail
- Cloning the Application Repository
- Creating the RDS
- Creating Launch Templates
- Creating Target Groups
- Creating Load Balancers
- Updating Application Files

- Creating Auto Scaling Groups
- Accessing the Application
- Creating a CloudFront Distribution
- Customizing WAF and CloudFront
- Testing Auto Scaling

## 1. Deploying the infrastructure

First of all, its infrastructure is mainly deployed and provisioned with **Terraform** by deploying its main services like **VPC**, **Security Groups**, **EC2 Instances**, **S3 Buckets**, **RDS Secrets**, **IAM Roles**, and **SNS Notifications**.

- Download these terraform files from [here](#).
- Run `ssh-keygen` and name it `3-tier-app` to make an **EC2 Key-Pair**.
- In this first step, don't include the `secrets.tf` file as we don't have the **RDS Endpoint** yet.
- Update the `variables.tf` file according to you.
- Execute these commands to deploy the infrastructure:

```

1  terraform init
2  terraform validate
3  terraform plan
4  terraform apply -auto-approve

```

*This will take some time to deploy the infrastructure.*

## 2. Configuring the Jump Servers

Now you have to configure the **Jump Servers** which will then be used for deploying your application automatically in an **Auto-Scaling Group**.

- SSH into `jump-server-web` .
- Run these commands to install **nginx** and **git**:

```

1  sudo yum install nginx -y
2  sudo systemctl start nginx
3  sudo systemctl enable nginx
4  sudo service nginx restart
5  sudo chkconfig nginx on
6  sudo yum install git -y

```

- Run these commands to install **AWS CLI**:

```
1 curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
2 unzip awscliv2.zip
3 sudo ./aws/install
```

- Now run `aws configure` to configure your **Access Keys** and **Secret Access Keys**.
- Run these commands to install **Node JS**:

```
1 # Download and install nvm:
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
3
4 # in lieu of restarting the shell
5 \. "$HOME/.nvm/nvm.sh"
6
7 # Download and install Node.js:
8 nvm install 22
9
10 # Verify the Node.js version:
11 node -v # Should print "v22.19.0".
12 nvm current # Should print "v22.19.0".
13
14 # Verify npm version:
15 npm -v # Should print "10.9.3".
```

- Now SSH into `jump-server-app`.
- Run these commands to install **MySQL Client**:

```
1 sudo wget https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm
2 sudo dnf install mysql80-community-release-el9-1.noarch.rpm -y
3 sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2023
4 sudo dnf install mysql-community-client -y
5 mysql --version
```

- Run these commands to install **AWS CLI**:

```
1 curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
2 unzip awscliv2.zip
3 sudo ./aws/install
```

- Now run `aws configure` to configure your **Access Keys** and **Secret Access Keys**.
- Run these commands to install **Node JS**:

```
1 # Download and install nvm:
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
```

```

3
4  # in lieu of restarting the shell
5  \. "$HOME/.nvm/nvm.sh"
6
7  # Download and install Node.js:
8  nvm install 22
9  npm install -g pm2
10
11 # Verify the Node.js version:
12 node -v # Should print "v22.19.0".
13 nvm current # Should print "v22.19.0".
14
15 # Verify npm version:
16 npm -v # Should print "10.9.3".

```

### 3. Creating Jump Servers AMIs

Now you'll have to create **Amazon Machine Images (AMIs)** of the configured **Jump Servers** so that it'll be later used to create **Launch Templates**.

- Go to **EC2 --> Instances**.
- Select `jump-server-web` .
- Go to **Actions --> Image and templates --> Create image**.

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (selected), AMIs, and AMI Catalog. The main area displays two instances: 'jump-server-app' (running, t2.micro) and 'jump-server-web' (running, t2.micro). The 'jump-server-web' instance is currently selected. A context menu is open over this instance, with the 'Image and templates' option highlighted. Other options in the menu include 'Create image', 'Create template from instance', and 'Launch more like this'. At the bottom of the page, there's an 'i-029ba8a7d7d6123f3 (jump-server-web)' details section showing Public IPv4 address (3.237.192.161) and Private IPv4 addresses (10.75.1.216).

- Enter **Image name** = `Web-AMI` .
- Enter **Image description** = An AMI for Web Tier .

The screenshot shows the 'Create image' page for an EC2 instance. The instance ID is i-029ba8a7d7d6123f3 (jump-server-web). The 'Image name' field is set to 'Web-AMI'. The 'Image description - optional' field contains 'An AMI for Web Tier'. The 'Reboot instance' checkbox is checked. The 'Instance volumes' section shows a table with columns: Storage type, Device, Snapshot, Size, Volume type, IOPS, Throughput, Delete on termination, and Encrypted. The table has one row corresponding to the instance. At the bottom, there are links for CloudShell and Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

- Click **Create image**.
- Now select `jump-server-app`.
- Go to **Actions --> Image and templates --> Create image**.
- Enter **Image name** = `App-AMI`.
- Enter **Image description** = `An AMI for App Tier`.

The screenshot shows the 'Create image' page for an EC2 instance. The instance ID is i-0d20a9938b4fb299 (jump-server-app). The 'Image name' field is set to 'App-AMI'. The 'Image description - optional' field contains 'An AMI for App Tier'. The 'Reboot instance' checkbox is checked. The 'Instance volumes' section shows a table with columns: Storage type, Device, Snapshot, Size, Volume type, IOPS, Throughput, Delete on termination, and Encrypted. The table has one row corresponding to the instance. At the bottom, there are links for CloudShell and Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

**Now check both of the AMIs.**

Amazon Machine Images (AMIs) (2)						
Owned by me		Find AMI by attribute or tag				
Name	AMI name	AMI ID	Source	Owner	Visibility	Status
App-AMI	App-AMI	ami-0da96a91fbfa84a9	311141542374/App-AMI	311141542374	Private	Pending
Web-AMI	Web-AMI	ami-08a6ea835cf6357df	311141542374/Web-AMI	311141542374	Private	Pending

## 4. Creating a CloudTrail

Now you'll have to create a **CloudTrail** to track your **AWS Account Activity** and dumping it to an **S3 Bucket**.

- Go to **CloudTrail**.
- Click on **Create a trail**.
- Enter **Trail name = AWS-Account-Activity-3-Tier**.
- Click **Create trail**.

The screenshot shows the 'Quick trail create' wizard. It starts with a 'Trail details' section where users can log management events. A note says logs are sent to an S3 bucket we create on their behalf. Below this, a 'Trail name' field is filled with 'AWS-Accout-Activity-3-Tier'. A note below the field specifies a 3-128 character limit. The next section is 'Trail log bucket and folder', which contains a pre-filled S3 bucket name 'aws-cloudtrail-logs-311141542374-09cf7c5e'. A note states logs will be stored in this bucket. A warning box notes that while there's no cost to log events, users incur charges for the S3 bucket used for storage. At the bottom right are 'Cancel' and 'Create trail' buttons.

Now your **CloudTrail** and an **S3 Bucket** for it will be created.

Trails									
Name	Home region	Multi-region trail	ARN	Insights	Organization trail	S3 bucket	Log file prefix	CloudWatch Logs log group	Status
AWS-Accout-Activity-3-Tier	US East (N. Virginia)	Yes	arn:aws:cloudtraitus-east-1:311141542374:4trail/AWS-Accout-Activity-3-Tier	Disabled	No	aws-cloudtrail-logs-311141542374-09cf7c5e	-	-	Logging

## 5. Cloning the Application Repository

Now you'll have to clone the **Application Repository** on your local computer.

- Run this command to clone it:

```
1 git clone https://github.com/harishnshetty/3-tier-aws-15-services.git
```

## 6. Creating the RDS Database

In this step, you will create an **RDS Database** with **Multi-AZ** deployment for **High-Availability** and **Fault Tolerance**.

- Go to **Aurora and RDS**.
- Go to **Subnet groups**.
- Click on **Create DB subnet group**.
- Enter the following:
  - Name** = three-tier-rds-subnetgroup
  - Description** = A Subnet Group for RDS Database
  - VPC** = 3-tier-vpc

The screenshot shows the 'Create DB subnet group' wizard. In the 'Subnet group details' section, the 'Name' field is set to 'Three-Tier-RDS-SubnetGroup', the 'Description' field is 'A Subnet Group for RDS Database', and the 'VPC' dropdown is set to '3-tier-vpc (vpc-0ee4e81534db19fe4)'. The 'Subnets' dropdown below shows '12 Subnets, 3 Availability Zones'.

- Now select:
  - Availability Zones** = us-east-1a , us-east-1b , us-east-1c
  - Subnets** = DB-Private-Subnet-1a , DB-Private-Subnet-1b , DB-Private-Subnet-1c

The screenshot shows the 'Add subnets' wizard. Under 'Availability Zones', 'us-east-1a', 'us-east-1b', and 'us-east-1c' are selected. Under 'Subnets', 'DB-Private-Subnet-1a', 'DB-Private-Subnet-1b', and 'DB-Private-Subnet-1c' are selected. A note at the bottom states: 'For Multi-AZ DB clusters, you must select 3 subnets in 3 different Availability Zones.'

- Click **Create**.
- Now go to **Databases**.
- Click on **Create database**.
- Choose the following:
  - **Choose a database creation method** = Standard create
  - **Engine type** = MySQL

The screenshot shows the AWS RDS 'Create database' wizard. The 'Create database' step is selected. In the 'Choose a database creation method' section, 'Standard create' is selected. Below it, 'Easy create' is described as using recommended best-practice configurations. In the 'Engine options' section, 'MySQL' is selected, indicated by a blue outline around its radio button and icon. Other engine options shown are Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), PostgreSQL, MariaDB, and Oracle.

- Now choose:
  - **Templates** = Dev/Test
  - **Deployment options** = Multi-AZ DB instance deployment (2 instances)

**Templates**  
Choose a sample template to meet your use case.

- Production  
Use defaults for high availability and fast, consistent performance.
- Dev/Test  
This instance is intended for development use outside of a production environment.
- Free tier  
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

**Availability and durability**  
Deployment options [Info](#)  
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

- Multi-AZ DB cluster deployment (3 instances)  
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones
  - Increased read capacity
  - Reduced write latency
- Multi-AZ DB instance deployment (2 instances)  
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:
  - 99.95% uptime
  - Redundancy across Availability Zones
- Single-AZ DB instance deployment (1 instance)  
Creates a single DB instance without standby instances. This setup provides:
  - 99.5% uptime
  - No data redundancy

- Now Enter:
  - DB instance identifier** = db-3-tier
  - Master username** = admin
  - Credentials Management** = Self managed
  - Master password** = [Your RDS DB Password]

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). 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.

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

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

- Managed in AWS Secrets Manager - most secure  
RDS 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.

**Master password** [Info](#)

- In Instance configuration choose:
  - Burstable classes (includes t classes)
  - db.t3.small

**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 x classes)

Burstable classes (includes t classes)

**db.t3.small**  
2 vCPUs 2 GiB RAM Network: Up to 2,085 Mbps

- In Storage choose:
  - **Storage type** = General Purpose SSD (gp2)
  - **Allocated storage** = 20

**Storage**

**Storage type** [Info](#)  
Provisioned IOPS SSD (io2) storage volumes are now available.

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

**Allocated storage** [Info](#)  
20 GiB  
Allocated storage value must be 20 GiB to 16,384 GiB

ⓘ For high-throughput workloads, we recommend provisioning at least 100 GiB of General Purpose (SSD) storage. Lower values might result in higher latencies when the initial I/O credit balance is used up. [Learn more](#)

► Additional storage configuration

- In Connectivity choose:
  - **Virtual Private Cloud (VPC)** = 3-tier-vpc
  - **DB subnet group** = three-tier-rds-subnetgroup
  - **Public access** = No

Console Home | Console Home | EC2 | us-east-1 | Trails | CloudTrail | us-east-1 | Create database | Aurora and RDS | +

us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:

Aurora and RDS > Create database

**Network type** [Info](#)  
To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

IPv4  
Your resources can communicate only over the IPv4 addressing protocol.

Dual-stack mode  
Your resources can communicate over IPv4, IPv6, or both.

**Virtual private cloud (VPC)** [Info](#)  
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

3-tier-vpc (vpc-0ee4e81534db19fe4)  
12 Subnets, 3 Availability Zones

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.

three-tier-rds-subnetgroup  
3 Subnets, 3 Availability Zones

**Public access** [Info](#)  
 Yes  
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.  
 No  
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.

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

- Now choose **VPC security group (firewall)** = db-srv-sg

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

db-srv-sg X

**RDS Proxy**  
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)  
Expiry: May 26, 2061

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

**Additional configuration**

- In **Monitoring** uncheck the **Enable Enhanced monitoring**.

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

Database Insights - Advanced

- Retains 15 months of performance history
- Fleet-level monitoring
- Integration with CloudWatch Application Signals

Database Insights - Standard

**Additional monitoring settings**  
Enhanced Monitoring, CloudWatch Logs and DevOps Guru

**Enhanced Monitoring**  
 Enable 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

logs-db-auth-access-log

- Click **Create database**.
- Now update the `variables.tf` file with your RDS Endpoint which you can get from here:

- Now Download/Copy the secrets.tf file from [here](#).
- Apply it by executing:

```

1  terraform validate
2  terraform plan
3  terraform apply -auto-approve

```

- Now SSH to jump-server-app .
- Connect to your database by executing:

```

1  mysql -h CHANGE-TO-YOUR-RDS-ENDPOINT -u admin -p

```

- Paste this SQL query to create a **Transactions** table in **webappdb** database:

```

1  CREATE DATABASE webappdb;
2  SHOW DATABASES;
3  USE webappdb;
4  CREATE TABLE IF NOT EXISTS transactions(id INT NOT NULL AUTO_INCREMENT, amount
DECIMAL(10,2), description VARCHAR(100), PRIMARY KEY(id));
5  SHOW TABLES;
6  INSERT INTO transactions (amount,description) VALUES ('400','groceries');
7  SELECT * FROM transactions;

```

## 7. Creating Launch Templates

Now you'll have to create the **Launch Templates** from the **AMIs** you have created before.

- Go to **EC2 --> Launch Templates**.
- Click on **Create launch template**.
- Enter the following:
  - **Launch template name** = Web-Tier-LT
  - **Template version description** = A Launch Template for Web Tier

**Create launch template**

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

**Launch template name and description**

Launch template name - required

Web-Tier-LT

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description

A Launch Template for Web Tier

Max 255 chars

**Auto Scaling guidance** Info

Select this if you intend to use this template with EC2 Auto Scaling

Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

► Template tags

► Source template

**Launch template contents**

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

CloudShell Feedback

Cancel Create launch template

- In the **AMI** section, choose **Web-AMI** .

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents My AMIs Quick Start

Don't include in launch template  Owned by me  Shared with me

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

**Amazon Machine Image (AMI)**

**Web-AMI**

ami-08a6ea835cf6357df

2025-05-11T05:19:22.000Z Virtualization: hvm ENA enabled: true Root device type: ebs Boot mode: uefi-preferred

**Description**

An AMI for Web Tier

**Architecture**

x86\_64

**AMI ID**

ami-08a6ea835cf6357df

Cancel Create launch template

- In **Security groups** choose **web-srv-sg** .

The screenshot shows the 'Create launch template' page in the AWS CloudFormation console. The 'Network settings' section is expanded, showing options for Subnet, Availability Zone, Firewall (security groups), and Security groups. A tooltip for the Free tier is visible on the right side of the screen.

- In the IAM instance profile choose 3-tier-web-profile .

The screenshot shows the 'Create launch template' page in the AWS CloudFormation console. The 'Advanced details' section is expanded, showing options for IAM instance profile, Hostname type, DNS Hostname, Instance auto-recovery, Shutdown behavior, Stop - Hibernate behavior, and Termination protection. A tooltip for the Free tier is visible on the right side of the screen.

- In the User data write:

```

1 #!/bin/bash
2  # Log everything to /var/log/user-data.log
3  exec > >(tee /var/log/user-data.log|logger -t user-data -s 2>/dev/console) 2>&1
4
5  # Install AWS CLI v2 (if not already)
6  yum install -y awscli

```

```

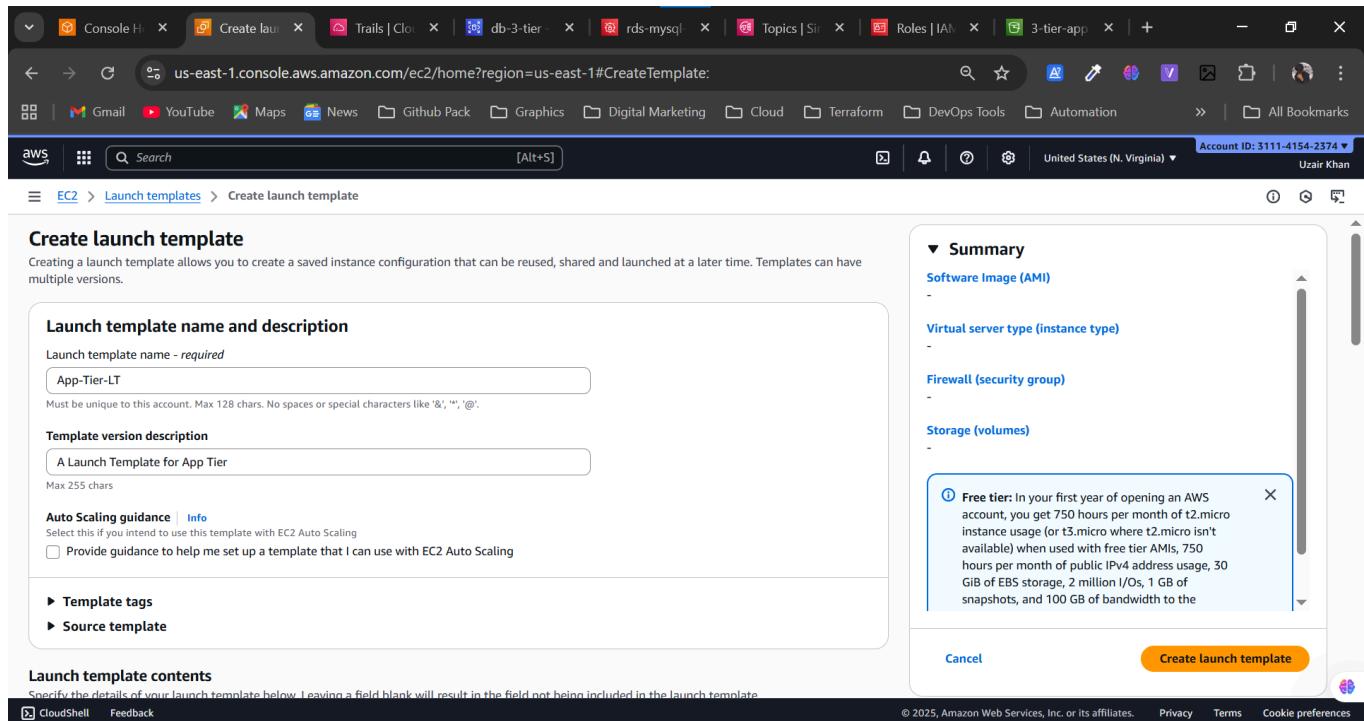
7
8 # Download application code from S3
9 aws s3 cp s3://<YOUR-S3-BUCKET-NAME>/application-code /home/ec2-user/application-
   code --recursive
10
11 # Go to app directory
12 cd /home/ec2-user/application-code
13
14 # Make script executable and run it
15 chmod +x web.sh
16 sudo ./web.sh

```

## 🔥 Important

Update the [YOUR-S3-BUCKET-NAME] with your actual S3 Bucket name for the Application Code.

- Click **Create launch Template**.
- Now create another **Launch Template** by clicking again on **Create launch template**.
- Enter the following:
  - **Launch template name** = App-Tier-LT
  - **Template version description** = A Launch Template for App Tier



The screenshot shows a browser window for the AWS CloudShell. The URL is [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTemplate](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTemplate). The page title is "Create launch template". The main content area is titled "Create launch template name and description". It contains fields for "Launch template name - required" (set to "App-Tier-LT") and "Template version description" (set to "A Launch Template for App Tier"). Below these are sections for "Auto Scaling guidance" (with a link to "Info" and a checkbox for "Provide guidance to help me set up a template that I can use with EC2 Auto Scaling"), "Template tags", and "Source template". At the bottom, there's a section titled "Launch template contents" with a note about specifying details. On the right side, there's a "Summary" panel with sections for "Software Image (AMI)", "Virtual server type (instance type)", "Firewall (security group)", and "Storage (volumes)". A tooltip for "Free tier" is open, explaining the benefits of the first year of an AWS account. At the bottom right is a "Create launch template" button.

- In the **AMI** section, choose **App-AMI**.

The screenshot shows the AWS EC2 'Create launch template' wizard. In the 'Amazon Machine Image (AMI)' section, a specific AMI is selected: 'App-AMI' (ami-0da964a91fbfa84a9). The description notes it's an AMI for App Tier. The architecture is listed as x86\_64. On the right, a summary panel includes a note about the Free tier, detailing 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of memory, and 1000 CPU credits. A 'Create launch template' button is at the bottom.

- In **Security groups** choose app-srv-sg .

The screenshot shows the 'Network settings' step of the wizard. Under 'Subnet', 'Don't include in launch template' is selected. Under 'Availability Zone', 'Don't include in launch template' is selected. Under 'Firewall (security group)', 'Select existing security group' is chosen, and 'app-srv-sg' is selected. The summary panel on the right contains the same information as the previous screenshot, including the Free tier details and a 'Create launch template' button.

- In the **IAM instance profile** choose 3-tier-app-profile .

The screenshot shows the AWS EC2 'Create launch template' interface. In the 'Advanced details' section, the IAM instance profile is set to '3-tier-app-profile'. The 'Hostname type' dropdown is set to 'Don't include in launch template'. Under 'DNS Hostname', both 'Enable resource-based IPv4 (A record)' and 'Enable resource-based IPv6 (AAAA record)' are unchecked. The 'Instance auto-recovery' and 'Shutdown behavior' dropdowns are also set to 'Don't include in launch template'. In the 'Summary' section, it shows an AMI for App Tier (ami-0da964a91fbfa84a9) and a virtual server type of 'app-srv-sg'. A note about the free tier is displayed, stating that in the first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of Amazon VPC Flow Log traffic, and 100 GB of Amazon CloudWatch Metrics.

- In the **User data** write:

```

1  #!/bin/bash
2  # Log everything to /var/log/user-data.log
3  exec > >(tee /var/log/user-data.log|logger -t user-data -s 2>/dev/console) 2>&1
4
5  # Install AWS CLI v2 (if not already)
6  yum install -y awscli
7
8  # Download application code from S3
9  aws s3 cp s3://<YOUR-S3-BUCKET-NAME>/application-code /home/ec2-user/application-
   code --recursive
10
11 # Go to app directory
12 cd /home/ec2-user/application-code
13
14 # Make script executable and run it
15 chmod +x app.sh
16 sudo ./app.sh

```

## 🔥 Important

Update the [YOUR-S3-BUCKET-NAME] with your actual S3 Bucket name for the Application Code.

- Click **Create launch template**.

## 8. Creating Target Groups

In this step, you'll have to create **Target Groups** for both **Web Tier** and **App Tier** so that it can be later used by the **Load Balancer** as a listener.

- Go to **EC2 --> Target groups**.
- Click on **Create target group**.
- Choose a **Target type** as **Instances**.

The screenshot shows the 'Specify group details' step of the 'Create target group' wizard. The 'Basic configuration' section is visible, showing that settings can't be changed after creation. Under 'Choose a target type', the 'Instances' option is selected, highlighted with a blue border. A list of benefits for using Instances is provided:

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

Other options shown but not selected are:

- IP addresses
  - Supports load balancing to VPC and on-premises 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 IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.
- Lambda function
  - Facilitates routing to a single Lambda function.
  - Accessible to Application Load Balancers only.

At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information and links for Privacy, Terms, and Cookie preferences.

- Enter the following:
  - **Target group name** = Web-Tier-TG
  - **Protocol** = HTTP
  - **Port** = 80
  - **VPC** = 3-tier-vpc

The screenshot shows the 'Create target group' page in the AWS Management Console. The URL is [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup). The page title is 'Create target group'. The 'Target group name' field contains 'Web-Tier-TG'. The 'Protocol' dropdown is set to 'HTTP' and the 'Port' input is '80'. The 'IP address type' section shows 'IPv4' selected. The 'VPC' section lists 'vpc-0ee4e81534db19fe4 (3-tier-vpc)'. The bottom navigation bar includes 'CloudShell', 'Feedback', and links to '© 2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- Click **Next**.
- Click **Create target group**.
- Now create the **App Tier Target Group**.
- Choose a **Target type** as **Instances**.

The screenshot shows the 'Specify group details' step. The URL is [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup). The title is 'Specify group details'. It shows the 'Step 1: Specify group details' is selected. Under 'Basic configuration', it says 'Your load balancer routes requests to the targets in a target group and performs health checks on the targets.' The 'Choose a target type' section has 'Instances' selected, which is highlighted with a blue border. Other options include 'IP addresses' and 'Lambda function'. The bottom navigation bar includes 'CloudShell', 'Feedback', and links to '© 2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

- Enter the following:
  - **Target group name** = App-Tier-TG
  - **Protocol** = HTTP
  - **Port** = 4000

- **VPC = 3-tier-vpc**

Target group name  
App-Tier-TG

Protocol  
HTTP

Port  
4000

IP address type  
IPv4

VPC  
vpc-0ee4e81534db19fe4 (3-tier-vpc)

Protocol version  
1.65535

- In **Health checks** choose:

- **Health check path = /health**

**Health checks**  
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

**Health check protocol**  
HTTP

**Health check path**  
/health

**Advanced health check settings**

- In **Advanced health check settings** choose:

- **Timeout = 2**
- **Interval = 5**

The screenshot shows the AWS CloudWatch Metrics console with the URL [us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup](https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup). The page is titled "Create target group". It contains several configuration fields:

- Healthy threshold:** Set to 5 (range 2-10).
- Unhealthy threshold:** Set to 2 (range 2-10).
- Timeout:** Set to 2 seconds (range 2-120).
- Interval:** Set to 5 seconds (range 5-300).
- Success codes:** Set to 200.

At the bottom, there are links for "CloudShell", "Feedback", and copyright information: "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

- Click **Next**.
- Click **Create target group**.

## 9. Creating Load Balancers

In this step, You'll be creating **Load Balancers** which will be load-balancing the traffic between multiple EC2 instances in your **Auto-Scaling Group**.

- Go to **EC2 --> Load Balancers**.
- Click on **Create load balancer**.
- In **Load balancer types** select **Application Load Balancer**.

Screenshot of the AWS Cloud Console showing the 'Compare and select load balancer type' wizard. The page compares Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GWLB). ALB handles HTTP and HTTPS traffic, NLB handles TCP and UDP traffic, and GWLB handles various protocols like TLS.

- Now choose:
  - Load balancer name** = app-internal-alb
  - Scheme** = Internal

Screenshot of the AWS Cloud Console showing the 'Create Application Load Balancer' wizard. The 'Basic configuration' step is shown, where the load balancer name is set to 'app-internal-alb' and the scheme is chosen as 'Internal'.

- In **Network mapping** choose:
  - VPC** = 3-tier-vpc
  - Availability Zones and subnets** = App-Private-Subnet-1a, App-Private-Subnet-1b, App-Private-Subnet-1c

**Network mapping** [Info](#)

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

**VPC** [Info](#)

The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view [target groups](#).

vpc-0ee4e81534db19fe4 (3-tier-vpc) [Create VPC](#)

**IP pools** [Info](#)

You can optionally choose to configure an IPAM pool as the preferred source for your load balancers IP addresses. Create or view [Pools](#) in the [Amazon VPC IP Address Manager console](#).

Use IPAM pool for public IPv4 addresses

Compatible with Internet-facing scheme, IPv4 and Dualstack IP address types.

**Availability Zones and subnets** [Info](#)

Select at least two Availability Zones and a subnet for each zone. A load balancer node will be placed in each selected zone and will automatically scale in response to traffic. The load balancer routes traffic to targets in the selected Availability Zones only.

us-east-1a (use1-az1)

Subnet Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

subnet-07449b0f7531e1cb2  
IPv4 subnet CIDR: 10.75.7.0/24 [App-Private-Subnet-1a](#)

us-east-1b (use1-az2)

Subnet Only CIDR blocks corresponding to the load balancer IP address type are used. At least 8 available IP addresses are required for your load balancer to scale efficiently.

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- In **Security groups** choose app-internal-alb-sg .

**Security groups** [Info](#)

A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

**Security groups**

Select up to 5 security groups [Create target group](#)

app-internal-alb-sg  
sg-06ba61002b2583c57 VPC: vpc-0ee4e81534db19fe4

- In **Listeners and routing** choose:
  - **Protocol** = HTTP
  - **Port** = 80
  - **Default action** = App-Tier-TG

**Listeners and routing** [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener **HTTP:80** [Remove](#)

Protocol	Port
HTTP	80
1-65535	

**Default action** [Info](#)

Forward to	App-Tier-TG	HTTP
Target type: Instance, IPv4		

[Create target group](#)

**Listener tags - optional**  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)  
You can add up to 50 more tags.

[Add listener](#)  
You can add up to 49 more listeners.

- Click on **Create load balancer**.
- Now create **Web External LB**.
- In **Load balancer types** select Application Load Balancer .

Screenshot of the AWS Cloud Console showing the "Compare and select load balancer type" page. The page compares Application Load Balancer (ALB), Network Load Balancer (NLB), and Gateway Load Balancer (GWLB). ALB handles HTTP and HTTPS traffic, NLB handles TCP, UDP, and TLS traffic, and GWLB handles various protocols like HTTP, HTTPS, and TCP.

- Now choose:
  - Load balancer name** = external-web-alb
  - Scheme** = Internet-facing

Screenshot of the AWS Cloud Console showing the "Create Application Load Balancer" wizard. The "Basic configuration" step is shown, where the load balancer name is set to "external-web-alb" and the scheme is chosen as "Internet-facing".

- In **Network mapping** choose:
  - VPC** = 3-tier-vpc
  - Availability Zones and subnets** = Public-Subnet-1a, Public-Subnet-1b, Public-Subnet-1c

The screenshot shows the 'Network mapping' step of the 'Create Application Load Balancer' wizard. It includes sections for 'VPC' (selected VPC: 'vpc-0ee4e81534db19fe4 (3-tier-vpc)'), 'IP pools' (with an unchecked checkbox for 'Use IPAM pool for public IPv4 addresses'), and 'Availability Zones and subnets' (with checkboxes for 'us-east-1a (use1-az1)' and 'us-east-1b (use1-az2)', each with a corresponding subnet dropdown). At the bottom, there are links for CloudShell, Feedback, and a copyright notice.

- In **Security groups** choose `web-frontend-alb-sg`.

The screenshot shows the 'Security groups' step of the wizard. It displays a list of existing security groups ('Select up to 5 security groups') with one item selected: 'web-frontend-alb-sg'. A 'Create new security group' button is also visible.

- In **Listeners and routing** choose:
  - **Protocol** = `HTTP`
  - **Port** = `80`
  - **Default action** = `Web-Tier-TG`

The screenshot shows the 'Listeners and routing' step of the wizard. It displays a table for a listener named 'Listener HTTP:80' with the following details:
 

Protocol	Port	Default action
HTTP	80	Forward to <code>Web-Tier-TG</code> Target type: Instance, IPv4 <a href="#">Create target group</a>

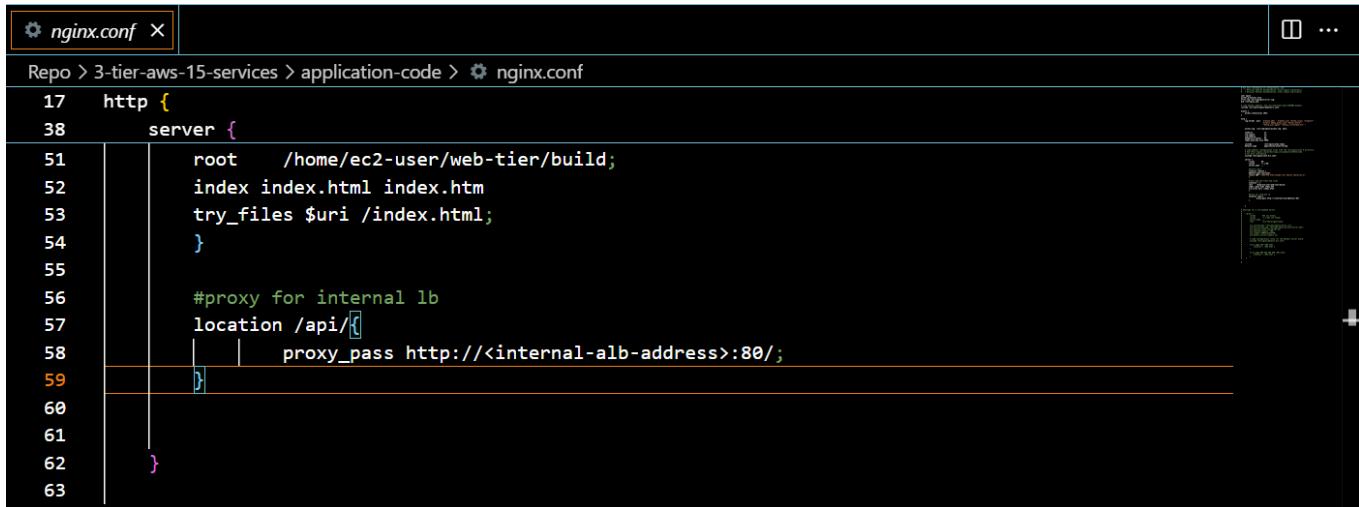
 Below the table, there are sections for 'Listener tags - optional' (with a 'Add listener tag' button) and 'Add listener' (with a note about the limit of 49 listeners).

- Click on **Create load balancer**.

## 10. Updaing Application Files

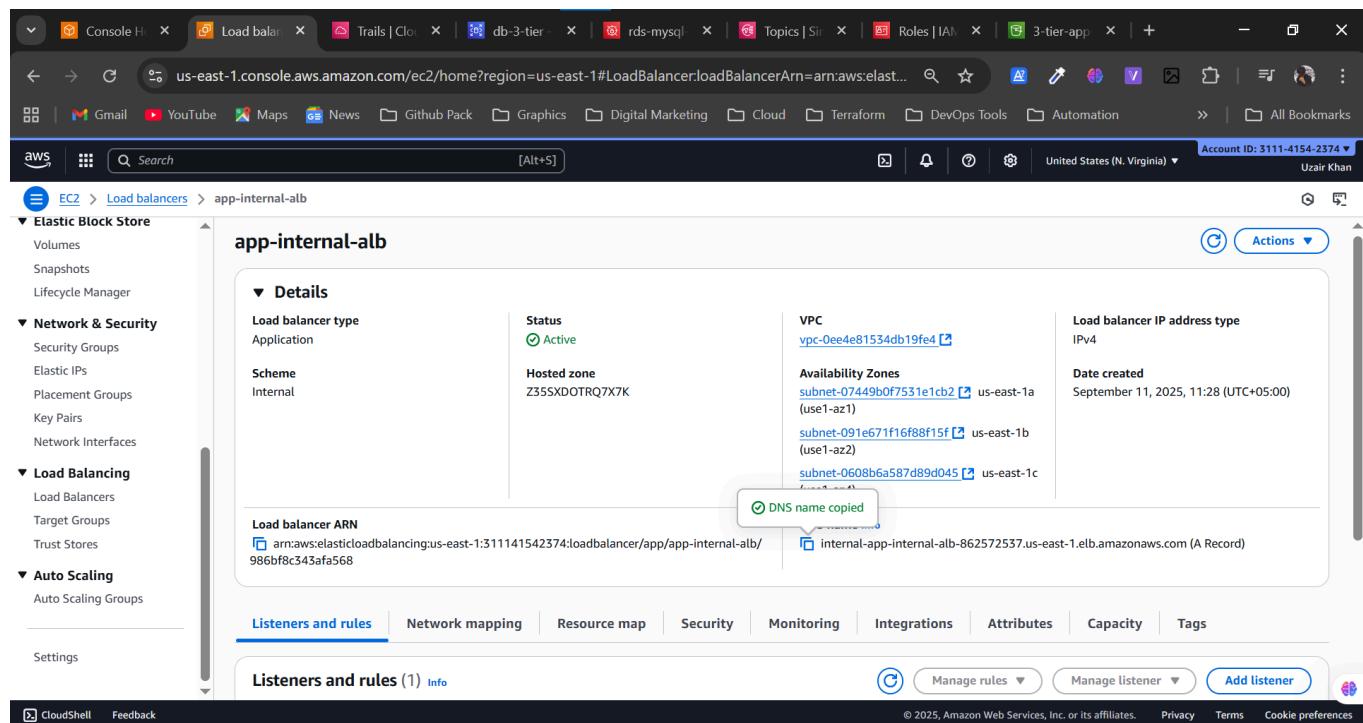
Now you'll have to update the **Application Files** in your cloned repo to get it to work correctly according to you. Then you'll upload it to your **Application Code S3 Bucket**.

- Open **application-code --> nginx.conf** in VS Code.



```
17 http {
38     server {
51         root    /home/ec2-user/web-tier/build;
52         index  index.html index.htm;
53         try_files $uri /index.html;
54     }
55
56     #proxy for internal lb
57     location /api/ {
58         proxy_pass http://<internal-alb-address>:80/;
59     }
60
61
62 }
63
```

- In **application-code/nginx.conf** update the `Internal-ALB-Address` to the one you just created.



- Open **application-code --> app.sh** in VS Code.

Repo > 3-tier-aws-15-services > application-code > \$ app.sh

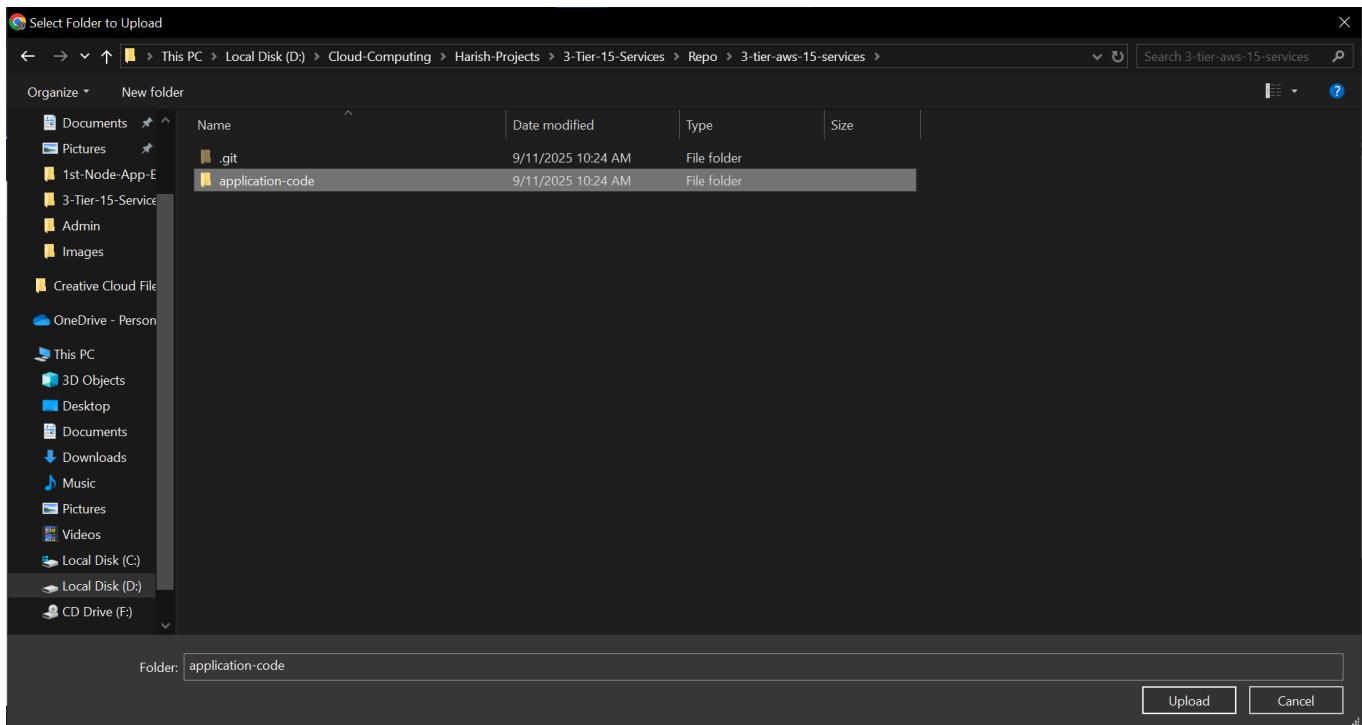
```
1 #!/bin/bash
2 set -e  # exit on error
3
4 # Download app-tier code
5 cd /home/ec2-user
6 aws s3 cp s3://<your-bucket-name>/application-code/app-tier app-tier --recursive
7
8 # Ensure correct ownership/permissions
9 sudo chown -R ec2-user:ec2-user /home/ec2-user
10 sudo chmod -R 755 /home/ec2-user/app-tier
11
12 # Run as ec2-user so nvm/npm/pm2 are available
13 su - ec2-user <<'EOF'
14 # Load nvm environment
15 export NVM_DIR="$HOME/.nvm"
```

- Update this **Bucket Name** with your Application Code Bucket name.
- Now open **application-code --> app-tier --> DbConfig.js** in VS Code.

Repo > 3-tier-aws-15-services > application-code > app-tier > JS DbConfig.js ●

```
1 const AWS = require('aws-sdk');
2 const secretsManager = new AWS.SecretsManager({
3   region: process.env.AWS_REGION || 'us-east-1'
4 });
5
6 async function getDatabaseSecrets() {
7   try {
8     const secretName = process.env.DB_SECRET_NAME || "rds-mysql-secret-3";
9     const data = await secretsManager.getSecretValue({ SecretId: secretName }).promise();
10
11     if ('SecretString' in data) {
12       return JSON.parse(data.SecretString);
13     } else {
14       throw new Error('Secret binary not supported');
15     }
16   }
```

- Update the following:
  - **AWS Region** = Your AWS Region
  - **Secret Name** = Your RDS Secrets Name
- Now go to your **Application Code S3 Bucket**.
- Click on **Upload folder**.
- Select the **application-code** folder and upload it.



*Wait for the upload to get complete.*

## 11. Creating Auto-Scaling Groups

Now you'll create the main component which is **Auto-Scaling Group** for scaling-up and scaling-down your EC2 Instances according to the traffic.

- Go to **EC2 --> Auto scaling groups.**
- Click on **Create Auto Scaling group.**
- Now select:

- **Name = Web-Tier-ASG**
- **Launch template = Web-Tier-LT**

**Choose launch template** Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

**Auto Scaling group name** Enter a name to identify the group.

Web-Tier-ASG

Must be unique to this account in the current Region and no more than 255 characters.

**Launch template** Info

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Web-Tier-LT

Create a launch template Info

Version

Default (1)

- Now in **Instance type requirements** select:
  - Manually add instance types
  - t2.small
  - Family and generation flexible

**Choose instance launch options** Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements** Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

**Specify instance attributes** Provide your compute requirements. We fulfill your desired capacity with matching instance types based on your allocation strategy selection.

**Manually add instance types** Add one or more instance types. Any of the instance types may be launched to fulfill your desired capacity based on your allocation strategy selection.

Primary instance type

1. t2.small 1vCPU 2 Gib Memory

Weight Info

Reset to launch template

Your launch template does not specify an instance type. As a result, "Reset to launch template" cannot be chosen. You can continue by adding an instance type above.

**Additional instance types**

Choose the types of instances you want recommended based on the primary instance type

Family and generation flexible  Size flexible

- Now select:

- **VPC** = 3-tier-vpc
- **Availability Zones and subnets** = Web-Private-Subnet-1a , Web-Private-Subnet-1b , Web-Private-Subnet-1c
- **Availability Zone distribution** = Balanced best effort

The screenshot shows the 'Create Auto Scaling group' wizard on the AWS EC2 console. The 'VPC' section is set to 'vpc-0ee4e81534db19fe4 (3-tier-vpc)'. The 'Availability Zones and subnets' section lists three subnets: 'use1-az1 (us-east-1a) | subnet-05fe7119314cef656 (Web-Private-Subnet-1a)', 'use1-az2 (us-east-1b) | subnet-002b47ef64e17edc3 (Web-Private-Subnet-1b)', and 'use1-az4 (us-east-1c) | subnet-0d1527ff5ac2e6449 (Web-Private-Subnet-1c)'. The 'Availability Zone distribution' section has 'Balanced best effort' selected. At the bottom, there are links for CloudShell, Feedback, and a footer with copyright information.

- Now choose:
  - **Load balancing** = Attach to an existing load balancer
  - **Attach to an existing load balancer** = Choose from your load balancer target groups
  - **Existing load balancer target groups** = Web-Tier-TG | HTTP

**Step 3 - optional**

**Integrate with other services**

- Step 4 - optional
- Configure group size and scaling
- Step 5 - optional
- Add notifications
- Step 6 - optional
- Add tags
- Step 7
- Review

**Load balancing** Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer  
Choose from your existing load balancers.

Attach to a new load balancer  
Quickly create a basic load balancer to attach to your Auto Scaling group.

**Attach to an existing load balancer**

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

**Existing load balancer target groups**

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

Web-Tier-TG | HTTP  
Application Load Balancer: external-web-alb

**VPC Lattice integration options** Info

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- In the **Scaling** section, choose:
  - Automatic Scaling** = Target tracking scaling policy
  - Target value** = 60

**Scaling limits**

Set limits on how much your desired capacity can be increased or decreased.

<b>Min desired capacity</b>	<b>Max desired capacity</b>
1	1
Equal or less than desired capacity	Equal or greater than desired capacity

**Automatic scaling - optional**

Choose whether to use a target tracking policy Info

You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**

Target Tracking Policy

**Metric type** Info

Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Average CPU utilization

**Target value**

60

**Instance warmup** Info

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

- In **SNS Topic** choose web-tier-sns

Step 1 Choose launch template

Step 2 Choose instance launch options

Step 3 - optional Integrate with other services

Step 4 - optional Configure group size and scaling

Step 5 - optional **Add notifications**

Step 6 - optional Add tags

Step 7 Review

**Add notifications - optional** Info

Send notifications to SNS topics whenever Amazon EC2 Auto Scaling launches or terminates the EC2 instances in your Auto Scaling group.

**Notification 1**

**SNS Topic**

Choose an SNS topic to use to send notifications

web-tier-sns (uzairikhhan2k2@gmail.com)

**Create a topic**

**Event types**

Notify subscribers whenever instances

Launch  
 Terminate  
 Fail to launch  
 Fail to terminate

**Add notification**

**Cancel** **Skip to review** **Previous** **Next**

- **Add a tag:**

- **Key = Name**
- **Value = Web-ASG**

Step 1 Choose launch template

Step 2 Choose instance launch options

Step 3 - optional Integrate with other services

Step 4 - optional Configure group size and scaling

Step 5 - optional **Add notifications**

Step 6 - optional **Add tags**

Step 7 Review

**Add tags - optional** Info

Add tags to help you search, filter, and track your Auto Scaling group across AWS. You can also choose to automatically add these tags to instances when they are launched.

You can optionally choose to add tags to instances (and their attached EBS volumes) by specifying tags in your launch template. We recommend caution, however, because the tag values for instances from your launch template will be overridden if there are any duplicate keys specified for the Auto Scaling group.

**Tags (1)**

Key	Value - optional	Tag new instances
Name	Web-ASG	<input checked="" type="checkbox"/>

**Add tag**

49 remaining

**Cancel** **Previous** **Next**

- Click **Next**
- Click **Create Auto Scaling group\***.
- Now create the **App Tier Auto-Scaling Group**.
- Now select:
  - **Name = App-Tier-ASG**

- **Launch template = App-Tier-LT**

**Choose launch template** Info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

**Name**

**Auto Scaling group name**  
Enter a name to identify the group.

**Launch template** Info

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

**App-Tier-LT** (selected)

[Create a launch template](#)

**Version**  
 (1)

- Now in **Instance type requirements** select:

- Manually add instance types
- **t2.small**
- Family and generation flexible

**Choose instance launch options**

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

**Instance type requirements** Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

**Specify instance attributes**  
Provide your compute requirements. We fulfill your desired capacity with matching instance types based on your allocation strategy selection.

**Manually add instance types**  
Add one or more instance types. Any of the instance types may be launched to fulfill your desired capacity based on your allocation strategy selection.

**Choose the instance types that best suit the needs of your application.**

**Primary instance type**

1.  1vCPU 2 Gib Memory

**Additional instance types**

Choose the types of instances you want recommended based on the primary instance type

Family and generation flexible  Size flexible

[Add instance type](#)

- Now select:

- **VPC = 3-tier-vpc**

- **Availability Zones and subnets** = App-Private-Subnet-1a , App-Private-Subnet-1b , App-Private-Subnet-1c
- **Availability Zone distribution** = Balanced best effort

**Network Info**

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0ee4e81534db19fe4 (3-tier-vpc)  
10.75.0.0/16

[Create a VPC](#)

**Availability Zones and subnets**

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

use1-az1 (us-east-1a) | subnet-07449b0f7531e1cb2 (App-Private-Subnet-1a)  
10.75.7.0/24

use1-az2 (us-east-1b) | subnet-091e671f16f88f15f (App-Private-Subnet-1b)  
10.75.8.0/24

use1-az4 (us-east-1c) | subnet-0608b6a587d89d045 (App-Private-Subnet-1c)  
10.75.9.0/24

[Create a subnet](#)

[Availability Zone distribution - new](#)

- Now choose:
  - **Load balancing** = Attach to an existing load balancer
  - **Attach to an existing load balancer** = Choose from your load balancer target groups
  - **Existing load balancer target groups** = App-Tier-TG | HTTP

Choose instance launch options

WITH zonal snr. You can also customize health check replacements and monitoring.

**Load balancing**

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer  
Choose from your existing load balancers.

Attach to a new load balancer  
Quickly create a basic load balancer to attach to your Auto Scaling group.

**Attach to an existing load balancer**

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

**Existing load balancer target groups**

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

App-Tier-TG | HTTP  
Application Load Balancer: app-internal-alb

**VPC Lattice integration options**

- In the **Scaling** section, choose:
  - **Automatic Scaling** = Target tracking scaling policy
  - **Target value** = 60

The screenshot shows the 'Create Auto Scaling group' page in the AWS Management Console. Under the 'Scaling' section, the 'Automatic scaling - optional' section is selected. It shows two options: 'No scaling policies' (disabled) and 'Target tracking scaling policy' (selected). The 'Target tracking scaling policy' section is highlighted with a blue border. Below it, the 'Scaling policy name' is set to 'Target Tracking Policy'. The 'Metric type' dropdown is set to 'Average CPU utilization'. The 'Target value' is set to '60'. The 'Instance warmup' is set to '300 seconds'. A checkbox for 'Disable scale in to create only a scale-out policy' is unchecked.

- In **SNS Topic** choose app-tier-sns

The screenshot shows the 'Create Auto Scaling group' page in the AWS Management Console, specifically the 'Add notifications - optional' step. On the left, a vertical navigation bar lists steps from 1 to 7, with 'Step 5 - optional Add notifications' currently selected. The main area shows a 'Notification 1' configuration. It includes a dropdown for 'SNS Topic' containing 'app-tier-sns (uzairikhank2@gmail.com, +1 more)', a 'Create a topic' button, and a 'Event types' section with checkboxes for 'Launch', 'Terminate', 'Fail to launch', and 'Fail to terminate', all of which are checked. At the bottom of the page, there are buttons for 'Cancel', 'Skip to review', 'Previous', and 'Next'.

- Add a **tag**:
  - **Key** = Name
  - **Value** = App-ASG

Step 1  
Choose launch template

Step 2  
Choose instance launch options

Step 3 - optional

Step 4 - optional

Step 5 - optional

Step 6 - optional  
**Add tags**

Step 7  
Review

**Add tags - optional** Info

Add tags to help you search, filter, and track your Auto Scaling group across AWS. You can also choose to automatically add these tags to instances when they are launched.

You can optionally choose to add tags to instances (and their attached EBS volumes) by specifying tags in your launch template. We recommend caution, however, because the tag values for instances from your launch template will be overridden if there are any duplicate keys specified for the Auto Scaling group.

**Tags (1)**

Key	Value - optional	Tag new instances
Name	App-ASG	<input checked="" type="checkbox"/>

**Add tag**

49 remaining

**Cancel** **Previous** **Next**

- Click **Next**

- Click **Create Auto Scaling group\***.

*Now it will create two new EC2 instances.*

Instances (4) <small>Info</small>								
<input type="text"/> Find Instance by attribute or tag (case-sensitive)								
Connect <small>Instance state</small> Actions <small>Launch instances</small>								
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pul
<input type="checkbox"/>	jump-server-app	i-0d20a9938b4bfb299	<span>Running</span> <small>Q Q</small>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	ec2
<input type="checkbox"/>	jump-server-web	i-029ba8a7d7d6123f3	<span>Running</span> <small>Q Q</small>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	ec2
<input type="checkbox"/>	App-ASG	i-00ad03ebbe795c844	<span>Pending</span> <small>Q Q</small>	t2.small	-	<a href="#">View alarms +</a>	us-east-1a	-
<input type="checkbox"/>	Web-ASG	i-0a50d198903b3783d	<span>Running</span> <small>Q Q</small>	t2.small	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	-

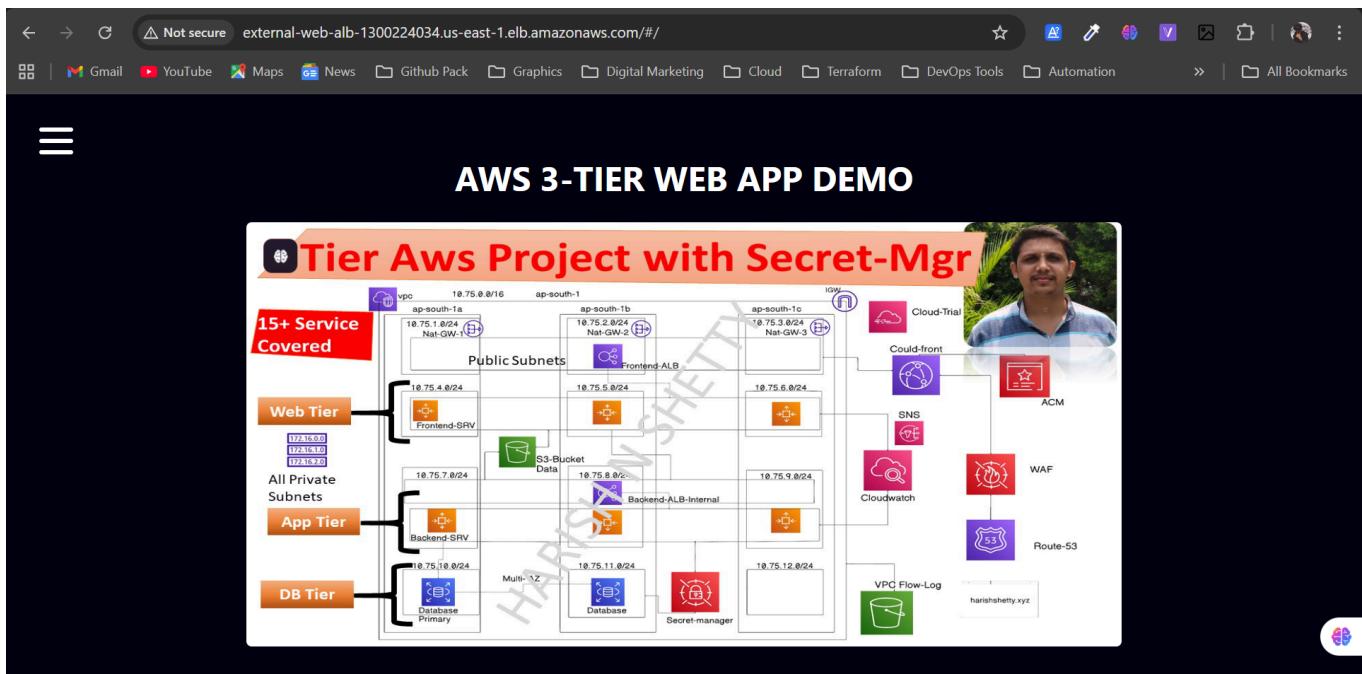
## 12. Accessing the Application

Now you can access your **Application** with your **External Load Balancer DNS**.

- Go to **external-web-alb**.
- Copy this **DNS**.

Screenshot of the AWS CloudWatch Metrics console showing metrics for an AWS Lambda function named "HelloWorld" over time. The chart displays CPU Usage,吞吐量 (Throughput), and Error Rate. A tooltip provides detailed information about the error rate metric.

- Paste it in your **web browser**.



- Now add some data in the database.

AURORA DATABASE DEMO PAGE

ID	AMOUNT	DESC
6	8000	Nike Air Jordan
7	5000	Denim

- You'll see it also in your Database internally.

```

Copyright (c) 2000, 2025, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> USE webappdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_webappdb |
+-----+
| transactions |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM transactions;
+----+----+-----+
| id | amount | description |
+----+----+-----+
| 6 | 8000.00 | Nike Air Jordan |
| 7 | 5000.00 | Denim |
| 8 | 100.00 | Coffee |
+----+----+-----+
3 rows in set (0.00 sec)

mysql> 
```

## 13. Creating a CloudFront Distribution

Now you'll have to create a **CloudFront Distribution** for a better latency and access to your application.

- Go to **CloudFront**.
- Click on **Create distribution**.
- Enter the following:
  - Distribution name** = three-tier-aws
  - Description** = A CloudFront Distribution for Three Tier App .
  - Distribution type** = Single website or app
  - Custom domain** = Add if you have one otherwise leave it blank
  - Tag**
    - Name = three-tier-aws

- Click **Next**.
- In **Specify origin** select:
  - **Origin type = Elastic Load Balancer**
  - **Elastic Load Balancing origin = external-web-alb**

The screenshot shows the 'Specify origin' step of creating a CloudFront distribution. On the left, a vertical navigation bar lists steps: Step 1 (Get started), Step 2 (Specify origin, which is selected and highlighted in blue), Step 3 (Enable security), and Step 4 (Review and create). The main content area is titled 'Specify origin' and contains a section for 'Origin type'. It lists five options: 'Amazon S3', 'Elastic Load Balancer' (which is selected and highlighted in blue), 'Elemental MediaPackage', 'VPC origin', and 'Other'. Below this is another section for 'Origin' with a sub-section for 'Elastic Load Balancing origin'. A text input field contains the value 'external-web-alb-1300224034.us-east-1.elb.amazonaws.com', and a 'Browse load balancers' button is visible.

- In **Customize origin settings** choose:
  - **Enable Origin Shield = No**
  - **Protocol = HTTP only**
  - **HTTP port = 80**

The screenshot shows the 'Customize origin settings' step of creating a CloudFront distribution. The main content area contains several configuration sections. At the top are two radio buttons: 'Use recommended origin settings' (unchecked) and 'Customize origin settings' (checked and highlighted in blue). Below this is a section for 'Add custom header - optional' with a 'Add header' button. Under 'Enable Origin Shield', the 'No' option is selected. The 'Protocol' section shows 'HTTP only' selected. The 'HTTP port' section has a text input field containing '80'. At the bottom, there's a 'Connection attempts' section with a note about the number of times CloudFront attempts to connect to the origin.

- In Web Application Firewall (WAF) select Enable security protections .

We've streamlined the process of creating a CloudFront distribution. Continue here and let us know what you think. Or go to the previous Create Distribution page.

**Step 1**  
 Get started  
 Specify origin  
 **Enable security**  
 Review and create

### Enable security

#### Web Application Firewall (WAF)

**Enable security protections**  
 Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

**Do not enable security protections**  
 Select this option if your application does not need security protections from AWS WAF.

**Use monitor mode**  
 Count how many of your requests would be blocked by this WAF configuration. When ready, you can disable monitor mode to begin blocking requests.

**Included security protections**

- Protect against the most common vulnerabilities found in web applications.
- Protect against malicious actors discovering application vulnerabilities.
- Block IP addresses from potential threats based on Amazon internal threat intelligence

**Additional protections for dynamic applications and APIs** Recommended

**SQL protections**  
 Block malicious request patterns that attempt to exploit SQL databases, like SQL injection. Recommended for applications that connect to a SQL database.

- Click **Next**.
- Click **Create distribution**.
- Now copy this **DNS**.

**three-tier-aws** Standard

**View metrics**

**General** **Security** **Origins** **Behaviors** **Error pages** **Invalidations** **Tags** **Logging**

**Details**

**Distribution domain name copied**

Name: three-tier-aws

ARN: arn:aws:cloudfront:311141542374:distribution/EDDYED8P46PUE

Last modified: Deploying

**Settings**

Description: A CloudFront Distribution for Three Tier App

Price class: Use all edge locations (best performance)

Supported HTTP versions: HTTP/2, HTTP/1.1, HTTP/1.0

Alternate domain names: [Add domain](#)

Standard logging: [Off](#)

Cookie logging: [Off](#)

Default root object: -

**Continuous deployment** Info

- Paste it into your web browser.

*Well done! You are now accessing your application with CloudFront + AWS WAF for Website Security*

## 14. Customizing WAF and CloudFront

Now this is the epic and fun part where you'll be customizing **Web ACL rules** for any incoming traffic to your application.

- Go to your **three-tier-aws** distribution.
- Go to **Security** tab.
- Click **Manage protections**.

- Check-Mark **Rate limiting**.

The screenshot shows the AWS CloudFront 'Edit security' page for a distribution named 'EDDYED8P46PUE'. On the left, a sidebar lists various services like Policies, Functions, and Telemetry. The main area is titled 'Edit security' and contains a 'Web Application Firewall (WAF)' section. It shows 'AWS WAF protection enabled' is selected, with a note that AWS WAF is enabled on this distribution. There are also sections for 'Additional protections for dynamic applications and APIs' (with 'Rate limiting' checked) and 'Price estimate'. A note at the bottom states that enabling SQL protections will add \$1/mo to the AWS WAF bill.

- Click **Save changes**.
- In **CloudFront geographic restrictions** add [Your Country] to **Block list**.

The screenshot shows the same 'Edit security' page for the 'EDDYED8P46PUE' distribution. The 'CloudFront geographic restrictions' section is highlighted. Under 'Restriction type', 'Block list' is selected. Under 'Countries', 'Pakistan' is listed in the dropdown. At the bottom right, there are 'Cancel' and 'Save changes' buttons, with 'Save changes' being highlighted.

- Click **Save changes**.
- Now you'll get a **403 error** from **CloudFront**.

The screenshot shows a web browser window with the URL `d2bn4yarq2jcm1.cloudfront.net/#/`. The page displays a 403 error message: "The request could not be satisfied." Below the message, it states: "The Amazon CloudFront distribution is configured to block access from your country. We can't connect to the server for this app or website at this time. There might be too much traffic or a configuration error. Try again later, or contact the app or website owner. If you provide content to customers through CloudFront, you can find steps to troubleshoot and help prevent this error by reviewing the CloudFront documentation." At the bottom of the page, there is a footer with the text "Generated by cloudfront (CloudFront)" and a Request ID: `a8v1qyXdwNadBwoqQJBS-p0gVOousdRRdXiimCp5fJzIV53-GAvV8A==`.

## 403 ERROR

### The request could not be satisfied.

The Amazon CloudFront distribution is configured to block access from your country. We can't connect to the server for this app or website at this time. There might be too much traffic or a configuration error. Try again later, or contact the app or website owner.

If you provide content to customers through CloudFront, you can find steps to troubleshoot and help prevent this error by reviewing the CloudFront documentation.

Generated by cloudfront (CloudFront)  
Request ID: a8v1qyXdwNadBwoqQJBS-p0gVOousdRRdXiimCp5fJzIV53-GAvV8A==

- Now go to **WAF & Shield**.
- Click **Create web ACL**.
- Select **Global resources**.
- Enter **Name = Three-Tier-WAF**

The screenshot shows a web browser window with the URL `us-east-1.console.aws.amazon.com/wafv2/homev2/web-acls/new?region=global#`. The page is titled "Describe web ACL and associate it to AWS resources". On the left, there is a sidebar with steps: Step 1 (Describe web ACL and associate it to AWS resources), Step 2 (Add rules and rule groups), Step 3 (Set rule priority), Step 4 (Configure metrics), and Step 5 (Review and create web ACL). The main content area is titled "Web ACL details". It includes fields for "Resource type" (set to "Global resources (CloudFront Distributions, CloudFront Distribution Tenants and AWS Amplify Applications)"), "Name" (set to "Three-Tier-WAF"), "Description - optional" (empty), and "CloudWatch metric name" (set to "Three-Tier-WAF"). The bottom of the page includes links for "CloudShell", "Feedback", "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", and "Cookie preferences".

- Click on **Add AWS resources** and add your **CloudFront distribution**.
- Click **Next**.
- Select **Add my own rules and rule groups**.
- In **Rule type** select **Rule builder**.

- Enter Name = Block[MyCountry]WithCustomResponse .

The screenshot shows the AWS WAF Web ACL creation interface. On the left, a sidebar lists steps: Step 1 (Describe web ACL and associate it to AWS resources), Step 2 (Add my own rules and rule groups), Step 3 (Set rule priority), Step 4 (Configure metrics), and Step 5 (Review and create web ACL). The current step is Step 2. The main area is titled "Add my own rules and rule groups". It shows a "Rule type" section with three options: "IP set" (radio button not selected), "Rule builder" (selected, highlighted in blue), and "Rule group" (radio button not selected). Below this is a "Rule builder" section with tabs for "Rule visual editor" (selected) and "Rule JSON editor". A note states: "You can use the JSON editor for complex statement nesting, for example to nest two OR statements inside an AND statement. The visual editor handles one level of nesting. For web ACLs and rule groups with complex nesting, the visual editor is disabled." A "Rule" section contains a "Name" input field with the value "BlockPakistan". At the bottom right of the main area is a "Validate" button.

- Now select:

- If a request = matches the statement
- Inspect = Originates from a country in
- Country codes = [Your Country Code]

The screenshot shows the AWS WAF Rule builder configuration interface. The "Statement" section is expanded, showing the "Inspect" dropdown set to "Originates from a country in" and the "Country codes" dropdown set to "Choose country codes". A list box shows "Pakistan - PK" selected. Below this is a note about IP address determination and a radio button for "Source IP address" (selected) or "IP address in header". The "Then" section is collapsed. The "Action" section is also collapsed at the bottom.

- In Action choose Block .
- In Custom response select:

- **Response code =** 404
- **Choose how you would like to specify the response body =** Create a custom response body
- **Response body object name =** Block[MyCountry]WithCustomResponse
- **Content type =** HTML
- Paste this in **Response body**:

```

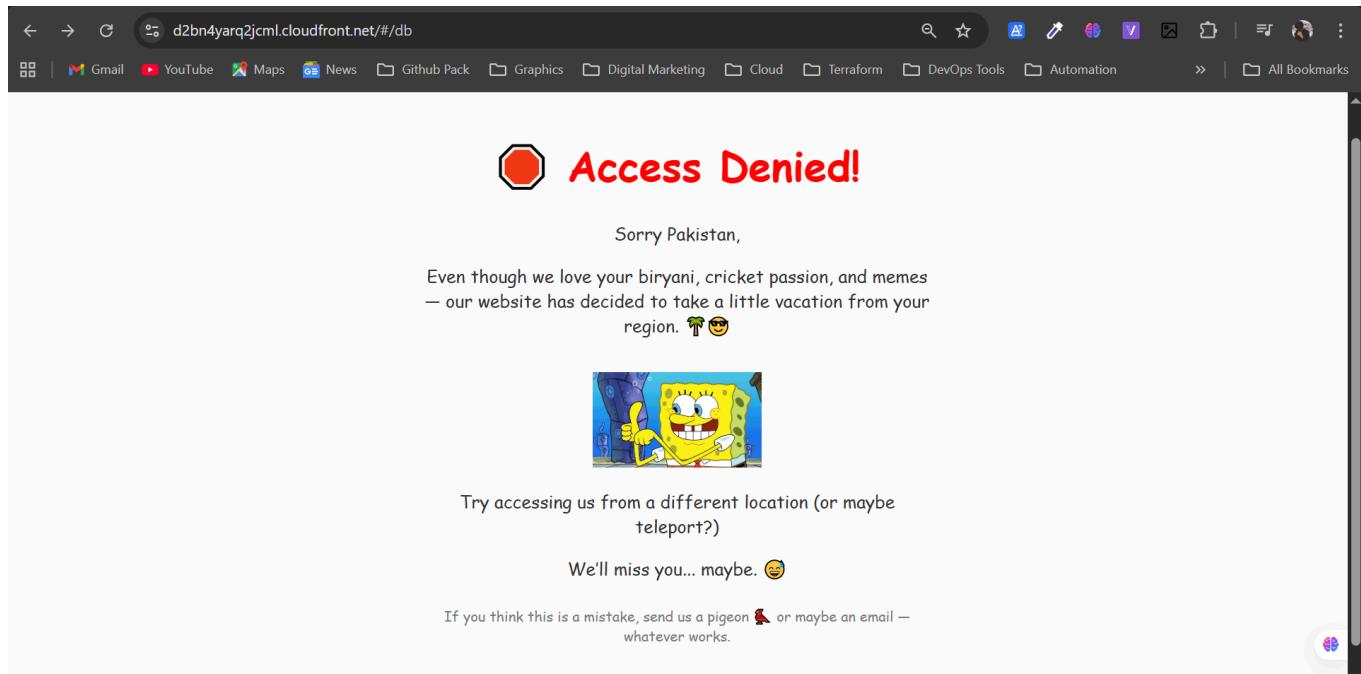
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>Oops! PK Not Today, Pakistan</title>
6    <style>
7      body {
8        font-family: 'Comic Sans MS', cursive, sans-serif;
9        background-color: #fefefe;
10       color: #333;
11       text-align: center;
12       padding: 50px;
13     }
14     img {
15       max-width: 200px;
16       margin-top: 20px;
17     }
18     .container {
19       max-width: 600px;
20       margin: auto;
21     }
22     h1 {
23       font-size: 3em;
24       color: #ff0000;
25     }
26     p {
27       font-size: 1.3em;
28       margin-top: 20px;
29     }
30     .tiny {
31       font-size: 0.8em;
32       color: gray;
33       margin-top: 30px;
34     }
35   </style>
36 </head>
37 <body>
```

```

38   <div class="container">
39     <h1>🚫 Access Denied!</h1>
40     <p>Sorry Pakistan,</p>
41     <p>Even though we love your biryani, cricket passion, and memes – our website
42       has decided to take a little vacation from your region. 🌴😎</p>
43     
45     <p>Try accessing us from a different location (or maybe teleport?)</p>
46     <p>We'll miss you... maybe. 😢</p>
47     <div class="tiny">
48       <p>If you think this is a mistake, send us a pigeon 🦜 or maybe an email –
49         whatever works.</p>
50   </div>
51 </div>
52 </body>
53 </html>

```

- Click **Add rule**.
- Select `Block[MyCountry]WithCustomResponse` .
- Click **Create web ACL**.
- Now access it again with your **CloudFront DNS**.



## 15. Testing Auto Scaling

Now finally, you have to test the **Auto-Scaling** of your application to handle the traffic load.

- Go to both of your **ASGs**.
- Update the **capacity** as:



## Group size

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

### Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

### Desired capacity

Specify your group size.

3

### Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

#### Min desired capacity

3

Equal or less than desired capacity

#### Max desired capacity

6

Equal or greater than desired capacity

Cancel

Update

Now your ASG will create 2 new EC2 instances

Instances (1/6) <a href="#">Info</a>										
EC2		Instances (1/6) <a href="#">Info</a>								
Dashboard		Last updated 1 minute ago <a href="#">Edit</a> <a href="#">Connect</a> <a href="#">Instance state</a> <a href="#">Actions</a> <a href="#">Launch instances</a>								
EC2 Global View		All states <a href="#">Edit</a>								
Events		Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pul	
<a href="#">Instances</a>		<input checked="" type="checkbox"/> Web-ASG	i-0e796da8fdd25a2b0	<span>Running</span>	t2.small	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	-	
<a href="#">Instances Types</a>		<input type="checkbox"/> Web-ASG	i-0bb0e0e11c1f7051	<span>Running</span>	t2.small	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1b	-	
<a href="#">Launch Templates</a>		<input type="checkbox"/> Web-ASG	i-0e317790562f89fe3	<span>Running</span>	t2.small	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1c	-	
<a href="#">Spot Requests</a>		<input type="checkbox"/> jump-server-app	i-0d20a9938b4fb299	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	ec2	
<a href="#">Savings Plans</a>		<input type="checkbox"/> jump-server-web	i-029ba8a7d7d6123f3	<span>Running</span>	t2.micro	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	ec2	
<a href="#">Reserved Instances</a>		<input type="checkbox"/> App-ASG	i-00ad03eb0e795c844	<span>Running</span>	t2.small	<span>2/2 checks passed</span>	<a href="#">View alarms +</a>	us-east-1a	-	

Congratulations to you on successfully building this entire project from end-to-end and achieving scalability, fault tolerance, high-availability and security.