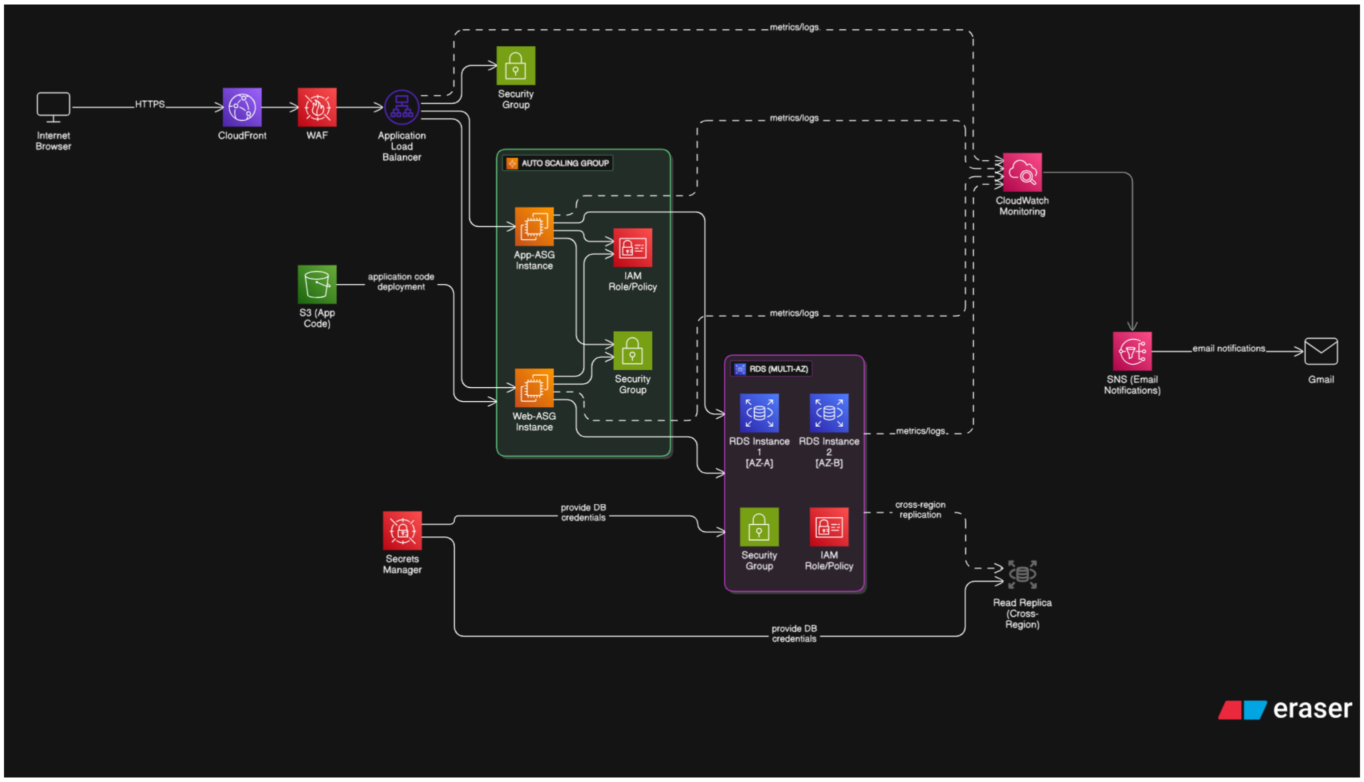
Three Tier 15 Microservices Application

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:

```
terraform init
terraform validate
terraform plan
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**:

```
sudo yum install nginx -y
sudo systemctl start nginx
sudo systemctl enable nginx
sudo service nginx restart
sudo chkconfig nginx on
sudo yum install git -y
```

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

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

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

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

# in lieu of restarting the shell
\. "$HOME/.nvm/nvm.sh"

# Download and install Node.js:
nvm install 22

# Verify the Node.js version:
node -v # Should print "v22.19.0".
nvm current # Should print "v22.19.0".

# Verify npm version:
npm -v # Should print "10.9.3".
```

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

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

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

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

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

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