Cloud-Based Multi-Tier Application Deployment (vProfile Project)

1. Setup AWS Infrastructure

1.1. Create EC2 Instances

1. Log in to your **AWS Console**.
2. Navigate to **EC2 > Instances > Launch Instance**.
3. Create the following EC2 instances (Choose Amazon Linux 2 or Ubuntu):
    - **db01** (MySQL)
    - **mc01** (Memcached)
    - **rmq01** (RabbitMQ)
    - **app01** (Tomcat + Application)
    - **web01** (Nginx)
4. Choose an appropriate instance type (t2.micro for learning, t2.medium for better performance).
5. Configure Security Groups:
    - **db01:** Open port **3306** (MySQL) for app01.
    - **mc01:** Open port **11211** (Memcached) for app01.
    - **rmq01:** Open port **5672** (RabbitMQ) for app01.
    - **app01:** Open port **8080** (Tomcat) for web01.
    - **web01:** Open port **80** (HTTP) for the internet.
6. Assign a key pair for SSH access.
7. Click **Launch**.

1.2 Create RDS MySQL Instance (Optional)

1. Navigate to AWS RDS > Create Database.
2. Choose MySQL, select Free Tier.
3. Set a username & password.
4. Enable Public Access (if needed for testing).
5. Click Create Database.

2. Provisioning the Servers

2.1. Install and Configure MySQL

SSH into db01:

```
ssh -i your-key.pem ec2-user@db01-public-ip
```

Install MySQL:

```
sudo yum update -y
sudo yum install -y mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

Secure MySQL:

```
sudo mysql_secure_installation
```

Create Database and User

```
mysql -u root -p
CREATE DATABASE accounts;
CREATE USER 'admin'@'%' IDENTIFIED BY 'admin123';
GRANT ALL PRIVILEGES ON accounts.* TO 'admin'@'%';
FLUSH PRIVILEGES;
EXIT;
```

2.1. Install Memcached

SSH into mc01:

```
ssh -i your-key.pem ec2-user@mc01-public-ip
```

Install and configure Memcached:

```
sudo yum install -y memcached
sudo systemctl start memcached
sudo systemctl enable memcached
```

2.2. Install RabbitMQ

SSH into rmq01:

```
ssh -i your-key.pem ec2-user@rmq01-public-ip
```

Install RabbitMQ:

```
sudo yum install -y epel-release
sudo yum install -y rabbitmq-server
sudo systemctl start rabbitmq-server
sudo systemctl enable rabbitmq-server
```

2.3. Install and Deploy Tomcat Application

SSH into rmq01:

```
ssh -i your-key.pem ec2-user@app01-public-ip
```

Install Java and Tomcat:

```
sudo yum install -y java-17-openjdk
wget https://downloads.apache.org/tomcat/tomcat-10/v10.1.26/bin/apache-tomcat-10.1.26.tar.gz
tar xzvf apache-tomcat-10.1.26.tar.gz
sudo mv apache-tomcat-10.1.26 /usr/local/tomcat
```

Deploy vProfile application:

```
wget https://github.com/hkhcoder/vprofile-project/releases/latest/download/vprofile-v2.war
sudo mv vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war
sudo systemctl start tomcat
```

2.4. Install and Configure Nginx

SSH into web01:

```
ssh -i your-key.pem ec2-user@web01-public-ip
```

Install Nginx:

```
sudo yum install -y nginx
```

Configure Nginx:

```
sudo nano /etc/nginx/nginx.conf
```

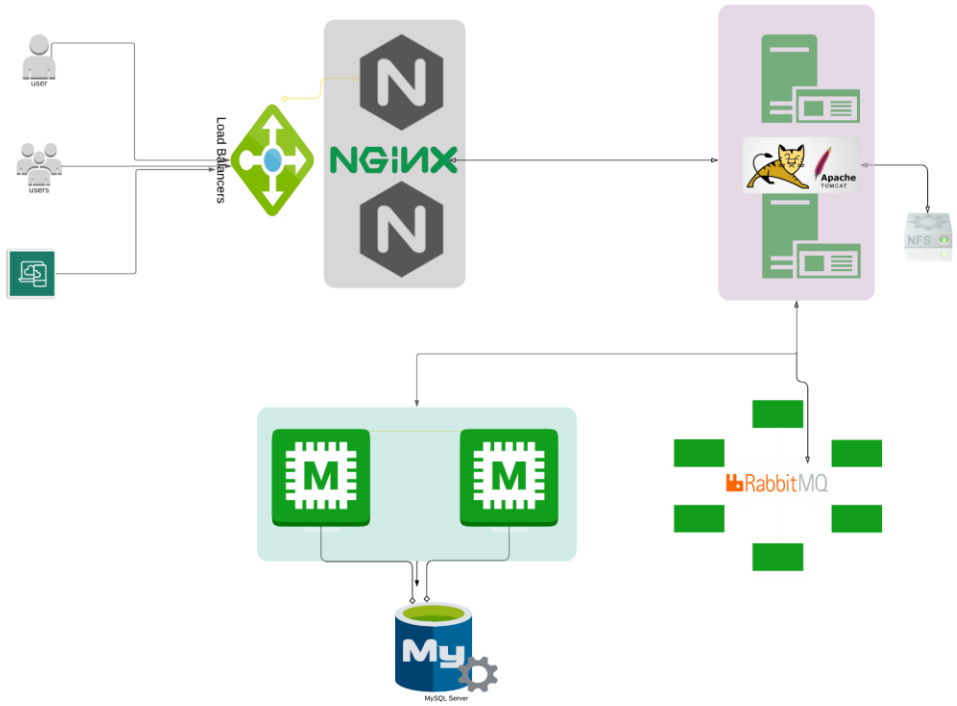
Add the following inside the server {} block:

```
location / {
    proxy_pass http://app01:8080;
}
```

Restart Nginx:

```
sudo systemctl restart nginx
```

Architecture



3. Automating with Terraform & Ansible

3.1. Automating AWS Setup with Terraform

Install Terraform on your local system.

Create a main.tf file:

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "app" {
  ami           = "ami-12345678" # Choose correct AMI
  instance_type = "t2.micro"
  key_name      = "your-key"
  tags = {
    Name = "app01"
  }
}
```

Deploy with:

```
terraform init
terraform apply -auto-approve
```

3.2. Automating Server Setup with Ansible

Install Ansible on your local machine.

Create an ansible-playbook.yml:

```
- hosts: all
  become: yes
  tasks:
    - name: Install MySQL
      yum:
        name: mariadb-server
        state: present
```

Run the playbook:

```
ansible-playbook -i inventory ansible-playbook.yml
```

4. Testing the Application

4.1. Get the public IP of web01.

4.2.Open a browser and visit:

```
http://web01-public-ip/
```

4.2. The vProfile application should load successfully!