Re-Architecting AWS Cloud Project: Comprehensive Implementation Guide

1. Introduction

1.1 Project Overview

This document outlines the re-architecting of an existing application to leverage modern AWS managed services, moving away from self-managed instances where appropriate. The project aims to host and run a production application on the AWS Cloud, focusing on enhancing scalability, reliability, and operational efficiency by utilizing Platform as a Service (PaaS) and Software as a Service (SaaS) offerings.

1.2 AWS Services Utilized

This re-architecture project will extensively utilize the following AWS services:

- **AWS Elastic Beanstalk:** A Platform as a Service (PaaS) offering that simplifies the deployment and scaling of web applications and services. It will manage the underlying EC2 instances for the Tomcat application server, replacing the need for manual VM management and providing built-in load balancing (ALB) and auto-scaling capabilities.
- **Amazon S3/EFS:** For object storage (S3) and potentially shared file systems (EFS) for application artifacts and persistent data.
- **Amazon RDS (Relational Database Service):** A managed relational database service that makes it easy to set up, operate, and scale a relational database in the cloud. This will replace self-managed MySQL instances.
- **Amazon ElastiCache:** A managed caching service that supports Memcached and Redis. This will replace self-managed Memcached instances.
- **Amazon MQ:** A managed message broker service for Apache ActiveMQ and RabbitMQ. This will replace self-managed RabbitMQ instances.
- **Amazon Route 53:** A highly available and scalable cloud Domain Name System (DNS) web service, used for both public and private DNS resolution.
- **Amazon CloudFront:** A fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency and high transfer speeds. This will be used to cache content and serve global audiences.
- **AWS Identity and Access Management (IAM):** For securely managing access to AWS services and resources.

- **AWS Certificate Manager (ACM):** For provisioning, managing, and deploying SSL/TLS certificates for use with AWS services.
- **Amazon CloudWatch:** For monitoring and observability, including setting up alarms for auto-scaling events.

1.3 Architectural Design

The re-architected solution aims for a highly available, scalable, and secure application environment by transitioning to managed AWS services. The architectural flow is as follows:

- **User Access via CloudFront:** Users will access the website using a URL. This URL will point to an Amazon CloudFront distribution, which acts as a Content Delivery Network (CDN). CloudFront will cache static and dynamic content, serving it from edge locations globally to reduce latency and improve user experience. This entry will be configured in a public DNS service (e.g., GoDaddy).
- **Elastic Beanstalk as Application Layer:** CloudFront will forward requests to the Elastic Beanstalk environment. Elastic Beanstalk automatically provisions and manages the underlying Application Load Balancer (ALB) and EC2 instances running the Tomcat application. The ALB, part of Elastic Beanstalk, will handle HTTPS traffic, with SSL/TLS certificates managed by AWS Certificate Manager (ACM). The ALB will reside in a security group allowing only HTTPS traffic. CloudWatch alarms will monitor the auto-scaling group managed by Elastic Beanstalk, ensuring dynamic scaling of Tomcat instances based on load.
- **Managed Backend Services:** Instead of self-managed instances, the application will connect to AWS managed services for its backend:
  - **Amazon RDS:** For the relational database (MySQL), providing automated backups, patching, and scaling.
  - **Amazon ElastiCache:** For the in-memory data store (Memcached), enhancing application performance.
  - **Amazon MQ:** For the message broker (RabbitMQ), facilitating asynchronous communication between application components.
- **S3 for Artifacts:** An S3 bucket will continue to be used for storing application artifacts, providing a durable and scalable storage solution.

1.4 Flow of Execution

The project implementation will follow a structured flow to set up the re-architected AWS infrastructure and deploy the application:

1. Login to AWS Account.
2. Create key pair for Elastic Beanstalk instance login.
3. Create security groups for ElastiCache, RDS, and Amazon MQ.

4. Create RDS database, Amazon ElastiCache cluster, and Amazon MQ broker.
5. Create Elastic Beanstalk Environment.
6. Update the security groups of backend services (RDS, ElastiCache, Amazon MQ) to allow traffic from Elastic Beanstalk instances.
7. Update the security groups of backend services to allow internal traffic (if necessary for inter-service communication).
8. Launch a temporary EC2 instance for database initialization.
9. Log in to the temporary instance and initialize the RDS database.
10. Change Elastic Beanstalk health check path to `/login` (or appropriate application endpoint).
11. Configure 443 HTTPS Listener with the Elastic Beanstalk's Application Load Balancer.
12. Build application artifact with updated backend service information.
13. Deploy artifact to Elastic Beanstalk.
14. Create CloudFront distribution with SSL certificate.
15. Update DNS entry in GoDaddy (or your domain registrar) to point to CloudFront.
16. Test the application URL.

This structured approach ensures that each component is configured correctly and integrated seamlessly into the overall re-architected solution.

2. Security Groups & Key Pairs Configuration

Security groups are crucial for controlling traffic to and from your AWS resources, acting as virtual firewalls. Key pairs are used for secure SSH access to EC2 instances. This section details the configuration of these fundamental security components for the re-architected environment.

2.1 Security Groups

For this project, the primary security group to configure is for the backend services (RDS, ElastiCache, Amazon MQ). Elastic Beanstalk will automatically create its own security groups for the load balancer and EC2 instances.

2.1.1 Backend Services Security Group

This security group will be applied to your Amazon RDS, ElastiCache, and Amazon MQ instances. It needs to allow inbound traffic from the Elastic Beanstalk instances and also permit inter-service communication among the backend components themselves.

- **Purpose:** To allow specific traffic from Elastic Beanstalk instances to backend services and enable inter-backend communication.
- **Inbound Rules (Initial Configuration - to be updated later):**

- Initially, you will create this security group without specific inbound rules, as the source for traffic (Elastic Beanstalk's security group) will only be known after Elastic Beanstalk environment creation.
- **After Elastic Beanstalk Environment Creation (Update Rule):**
  - **Type:** MySQL/Aurora (Port 3306) - for RDS
    - **Source:** The security group ID of the Elastic Beanstalk instances (e.g., `sg-elasticbeanstalk-instances`). This ensures only your application instances can connect to the database.
  - **Type:** Custom TCP (Port 11211) - for ElastiCache (Memcached)
    - **Source:** The security group ID of the Elastic Beanstalk instances.
  - **Type:** Custom TCP (Port 5672) - for Amazon MQ (RabbitMQ)
    - **Source:** The security group ID of the Elastic Beanstalk instances.
  - **Type:** All Traffic
    - **Source:** The security group ID of the **Backend Services Security Group itself** (e.g., `sg-backend-services`). This crucial rule allows MySQL, ElastiCache, and Amazon MQ instances within this security group to communicate with each other, which is often necessary for clustering or internal operations.

2.2 Key Pairs

While Elastic Beanstalk manages the underlying EC2 instances, you can still specify a key pair for SSH access if you need to directly inspect or troubleshoot the instances. This key pair will be associated with the EC2 instances launched by Elastic Beanstalk.

- **Action:** Create a new key pair in the AWS EC2 console if you don't have one already.
- **Recommendation:** Name the key pair descriptively (e.g., `beanstalk-key`). Ensure you download the `.pem` file and store it securely, as it is required for SSH access to your Elastic Beanstalk instances. This key pair will also be used for the temporary EC2 instance for database initialization.

3. Amazon RDS Configuration

Amazon Relational Database Service (RDS) is a managed service that simplifies the setup, operation, and scaling of relational databases. In this project, RDS will host the MySQL database, replacing a self-managed EC2 instance. This provides benefits such as automated backups, patching, and high availability.

3.1 Creating a DB Parameter Group

A DB parameter group acts as a container for engine configuration values that are applied to one or more DB instances. You can use it to manage your database configuration.

1. **Navigate to RDS:** Log in to your AWS Management Console and navigate to the RDS service.

2. **Create Parameter Group:**

   - In the navigation pane, under `RDS`, choose `Parameter groups`.
   - Click `Create parameter group`.
   - **Parameter group family:** Select `mysql8.0`.
   - **Type:** Select `DB Parameter Group`.
   - **Group name:** Provide a descriptive name (e.g., `vprofile-mysql-param-group`).
   - **Description:** Add a brief description.
   - Click `Create`.

3.2 Creating a DB Subnet Group

A DB subnet group is a collection of subnets (typically private) that you designate for your DB instances in a VPC. This ensures that your DB instances are deployed in a highly available and fault-tolerant manner across multiple Availability Zones.

1. **Navigate to RDS:** In the navigation pane, under `RDS`, choose `Subnet groups`.

2. **Create DB Subnet Group:**

   - Click `Create DB subnet group`.
   - **Name:** Provide a descriptive name (e.g., `vprofile-mysql-subnet-group`).
   - **Description:** Add a brief description.
   - **VPC:** Select the VPC where your application instances are located.
   - **Add subnets:** Select all Availability Zones within your chosen region and then select the private subnets within those Availability Zones.
   - Click `Create`.

3.3 Creating the MySQL RDS Database Instance

Now, you will provision the actual MySQL database instance using the parameter and subnet groups created previously.

1. **Navigate to RDS:** In the navigation pane, under `RDS`, choose `Databases`.
2. **Create Database:**
   - Click `Create database`.
   - **Choose a database creation method:** Select `Standard create`.
   - **Engine options:** Select `MySQL`.
   - **Engine version:** Choose a suitable version (e.g., `MySQL 8.0.xx`).
   - **Templates:** Select `Free tier` for testing or `Production` for a more robust setup.
   - **DB instance identifier:** Provide a unique name (e.g., `vprofile-mysql-db`).
   - **Credentials:**
     - **Master username:** `admin` (or your preferred username).
     - **Master password:** Choose `Auto generate a password` or provide a strong password. **If auto-generated, make sure to copy and save it securely after creation.**
   - **Connectivity:**
     - **VPC:** Select your VPC.
     - **Subnet group:** Select `vprofile-mysql-subnet-group`.
     - **Public access:** Select `No` (recommended for security).
     - **VPC security groups:** Select the **Backend Services Security Group** created in Section 2.1.1.
   - **Additional configuration:**
     - **Initial database name:** Provide the name of the database your application will use (e.g., `accounts`).
   - Review all settings and click `Create database`.

After the database instance is created, you will be able to view its credentials (if auto-generated) and endpoint. Make sure to save the master password and the database endpoint, as these will be needed for application configuration and database initialization.

4. Amazon ElastiCache Configuration

Amazon ElastiCache is a fully managed in-memory data store and caching service. In this project, ElastiCache (Memcached) will replace a self-managed Memcache instance, providing a high-performance, scalable, and managed caching solution for the application.

4.1 Creating an ElastiCache Parameter Group

An ElastiCache parameter group is a container for engine configuration values that are applied to one or more cache clusters. You can use it to manage your cache engine configuration.

1. **Navigate to ElastiCache:** Log in to your AWS Management Console and navigate to the ElastiCache service.

2. **Create Parameter Group:**

   - In the navigation pane, under `Memcached`, choose `Parameter Groups`.
   - Click `Create Parameter Group`.
   - **Name:** Provide a descriptive name (e.g., `vprofile-memcached-param-group`).
   - **Family:** Select `memcached1.6`.
   - **Description:** Add a brief description.
   - Click `Create`.

4.2 Creating an ElastiCache Subnet Group

An ElastiCache subnet group is a collection of subnets (typically private) that you designate for your cache clusters in a VPC. This ensures that your cache clusters are deployed in a highly available and fault-tolerant manner across multiple Availability Zones.

1. **Navigate to ElastiCache:** In the navigation pane, under `Memcached`, choose `Subnet Groups`.

2. **Create Subnet Group:**

   - Click `Create Subnet Group`.
   - **Name:** Provide a descriptive name (e.g., `vprofile-memcached-subnet-group`).
   - **Description:** Add a brief description.
   - **VPC:** Select the VPC where your application instances are located.
   - **Add subnets:** Select all Availability Zones within your chosen region and then select the private subnets within those Availability Zones.
   - Click `Create`.

4.3 Creating the Memcached Cluster

Now, you will provision the actual Memcached cluster using the parameter and subnet groups created previously.

1. **Navigate to ElastiCache:** In the navigation pane, under `Memcached`, choose `Memcached`.

2. **Create Memcached Cluster:**

   – Click `Create`.
   – **Choose a cluster creation method:** Select `Design your own cache`.
   – **Cluster engine:** Select `Memcached`.
   – **Location:** Select `AWS Cloud`.
   – **Memcached settings:**
     - **Name:** Provide a unique name (e.g., `vprofile-memcached-cluster`).
     - **Engine version compatibility:** Select a compatible version (e.g., `1.6.x`).
     - **Port:** `11211` (default for Memcached).
     - **Parameter group:** Select `vprofile-memcached-param-group`.
   – **Network & security:**
     - **Subnet group:** Select `vprofile-memcached-subnet-group`.
     - **VPC security groups:** Select the **Backend Services Security Group** created in Section 2.1.1.
   – Review all settings and click `Create`.

After the Memcached cluster is created, you will be able to view its endpoint. Make sure to save the endpoint, as it will be needed for application configuration.

5. Amazon MQ Configuration

Amazon MQ is a managed message broker service for Apache ActiveMQ and RabbitMQ that makes it easy to set up and operate message brokers in the cloud. In this project, Amazon MQ (RabbitMQ) will replace a self-managed RabbitMQ instance, providing a fully managed and highly available messaging solution.

5.1 Creating a RabbitMQ Broker

1. **Navigate to Amazon MQ:** Log in to your AWS Management Console and navigate to the Amazon MQ service.

2. **Create Broker:**

   – Click `Create broker`.
   – **Select broker engine:** Choose `RabbitMQ`.
   – **Deployment mode:** Select `Single-instance broker` for simplicity in this project. For production, consider `Active/standby broker` for high availability.

- **Broker instance type:** Select a suitable instance type (e.g., `mq.t2.micro`).
- **Broker name:** Provide a unique name (e.g., `vprofile-rabbitmq-broker`).
- **User management:**
  - **Username:** Provide a username (e.g., `test`).
  - **Password:** Provide a strong password (e.g., `test`). **Make sure to save this username and password securely.**
- **Network and security:**
  - **VPC:** Select the VPC where your application instances are located.
  - **Security groups:** Select the **Backend Services Security Group** created in Section 2.1.1.
  - **Public accessibility:** Keep `No` (recommended for security).
- Review all settings and click `Create broker`.

After the RabbitMQ broker is created, you will be able to view its endpoint (URL). Make sure to save the endpoint, username, and password, as these will be needed for application configuration.

6. Database Initialization

After setting up the RDS instance, you need to initialize its database with the application's schema and initial data. This is typically done by launching a temporary EC2 instance, connecting to the RDS database, and executing a SQL dump file. This ensures your application has the necessary database structure to function correctly.

6.1 Launching a Temporary EC2 Instance for DB Initialization

1. **Launch EC2 Instance:**
   - **AMI:** Ubuntu (e.g., Ubuntu Server 22.04 LTS) - or any Linux distribution you are comfortable with.
   - **Instance Type:** `t2.micro` (or a suitable small instance type).
   - **Key Pair:** Select the key pair you created in Section 2.2.
   - **Security Group:** Create a temporary security group that allows SSH (port 22) from your IP address. Alternatively, you can temporarily modify the **Backend Services Security Group** to allow SSH from your IP, but remember to revert this change for security.
   - Launch the instance.
2. **Update Backend Security Group:**
   - Go to the **Backend Services Security Group** (created in Section 2.1.1).

  - Add an inbound rule to allow `MySQL/Aurora (Port 3306)` traffic from the security group of this temporary EC2 instance. This allows the temporary instance to connect to your RDS database.

6.2 Initializing the RDS Database

1. **SSH into Temporary EC2 Instance:** Connect to the newly launched EC2 instance using SSH and your key pair.

2. **Switch to Root User (Optional but Recommended):** `bash     sudo su -`

3. **Install MySQL Client and Git:** `bash     sudo apt update     sudo apt install mysql-client git -y`

4. **Clone V-Profile Project Source Code:** `bash     git clone https://github.com/hkhcoder/vprofile-project.git`

5. **Navigate to Project Directory and Switch Branch:** `bash     cd vprofile-project     git checkout awsrefactor`

6. **Execute Database Dump:** Replace `<RDS_ENDPOINT>`, `<USERNAME>`, and `<PASSWORD>` with your actual RDS endpoint, master username, and master password. `bash     mysql -h <RDS_ENDPOINT> -u <USERNAME> -p<PASSWORD> accounts < src/main/resources/db_backup.sql`

   - You might be prompted for the password if you used `-p` without a space.

7. **Verify Database Initialization (Optional):** You can connect to the database and list tables to verify the dump was successful: `bash mysql -h <RDS_ENDPOINT> -u <USERNAME> -p<PASSWORD> accounts SHOW TABLES;     EXIT;`

8. **Terminate Temporary EC2 Instance:** Once the database is initialized, you can terminate this temporary EC2 instance to save costs and maintain security. Remember to also remove the temporary inbound rule from the **Backend Services Security Group** if you added one specifically for this instance.

7. AWS Elastic Beanstalk Environment Setup

AWS Elastic Beanstalk is a Platform as a Service (PaaS) that simplifies the deployment and scaling of web applications. It automatically handles the provisioning of infrastructure (EC2 instances, load balancers, auto-scaling groups) and application deployment. This section guides you through setting up your Elastic Beanstalk environment for the V-Profile application.

7.1 Preparing IAM Role for Elastic Beanstalk

Elastic Beanstalk requires specific IAM roles to operate and manage resources on your behalf. It's crucial to ensure these roles have the necessary permissions.

1. **Delete Existing `aws-elasticbeanstalk-service-role` (if present):** If you have previously used Elastic Beanstalk, an old service role might exist. It's good practice to delete it to ensure a clean setup with the correct permissions.
   - Navigate to the IAM service in your AWS Management Console.
   - Go to `Roles` and search for `aws-elasticbeanstalk-service-role`.
   - If found, select it and click `Delete role`.
2. **Create New IAM Role for Elastic Beanstalk:**
   - Click `Create role`.
   - **Trusted entity:** Select `AWS service`.
   - **Use case:** Select `Elastic Beanstalk`.
   - **Permissions:** Attach the following policies:
     - `AWSElasticBeanstalkFullAccess` (or more granular policies for production)
     - `AWSElasticBeanstalkWebTier`
     - `AWSElasticBeanstalkWorkerTier`
     - `AWSElasticBeanstalkMulticontainerDocker` (if using Docker)
     - `AmazonS3FullAccess` (if your application needs to access S3)
     - `AmazonEC2FullAccess` (if Elastic Beanstalk needs to manage EC2 instances)
     - `AmazonRDSFullAccess` (if Elastic Beanstalk needs to manage RDS)
     - `AmazonElastiCacheFullAccess` (if Elastic Beanstalk needs to manage ElastiCache)
     - `AmazonMQFullAccess` (if Elastic Beanstalk needs to manage Amazon MQ)
   - Give the role a descriptive name (e.g., `vprofile-beanstalk-service-role`) and create it.

7.2 Creating the Elastic Beanstalk Environment

1. **Navigate to Elastic Beanstalk:** Log in to your AWS Management Console and navigate to the Elastic Beanstalk service.

2. **Create a new environment:**

   - Click `Create a new environment`.
   - **Environment tier:** Select `Web server environment`.
   - **Application name:** Provide a name for your application (e.g., `vprofile-app`).

- **Environment name:** Provide a name for your environment (e.g., `vprofile-prod-env`).
- **Domain:** You can leave this blank for now, or provide a custom domain if you have one configured.
- **Platform:**
  - **Platform:** Select `Tomcat`.
  - **Platform branch:** Select `Tomcat 10 with Corretto 17` (or the appropriate version).
- **Application code:** Select `Sample application` for initial setup. You will deploy your actual artifact later.

3. **Configure more options (Custom configuration):**

   - **Software:**
     - **Environment properties:** You might need to add environment variables here later for database connection strings, etc.
   - **Instances:**
     - **EC2 key pair:** Select the key pair you created in Section 2.2 for SSH access.
     - **IAM instance profile:** Select the IAM role you created in Section 7.1 (e.g., `vprofile-beanstalk-service-role`).
   - **Network:**
     - **VPC:** Select your VPC.
     - **Visibility:** `Public`.
     - **Load balancer:** Select `Application Load Balancer`.
     - **Subnets:** Select all public subnets in your chosen Availability Zones.
     - **Database:** Ensure `Create an RDS database` is **unchecked**, as you are using an existing RDS instance.
   - **Capacity:**
     - **Environment type:** Select `Load balanced`.
     - **Instances:**
       - **Minimum instances:** `2` (for high availability).
       - **Maximum instances:** `4` (or based on your scaling needs).
     - **Instance type:** `t2.micro` (or suitable for your application).
   - **Load balancer:**
     - **Listeners:** You will configure HTTPS listeners later.
     - **Processes:**
       - Select the default process.
       - Go to `Actions` -> `Edit`.

        - **Enable stickiness:** Enable session stickiness if your application requires it.
    - **Monitoring:** Configure CloudWatch alarms as needed.
4. **Review and create:** Review all settings and click `Create environment`.

Elastic Beanstalk will now provision the necessary resources, including EC2 instances, an Application Load Balancer, and an Auto Scaling Group. This process may take several minutes. Once complete, your environment will be ready for application deployment.

8. Update Security Group & ELB Configuration

After the Elastic Beanstalk environment is created, it automatically provisions an Application Load Balancer (ALB) and associated security groups for both the ALB and the EC2 instances managed by Beanstalk. To ensure proper communication, you need to update the backend services security group to allow traffic from the Elastic Beanstalk instance security group.

8.1 Update Backend Services Security Group

1. **Identify Elastic Beanstalk Instance Security Group:**
    - Navigate to the EC2 service in your AWS Management Console.
    - Under `Security Groups`, you will find a security group created by Elastic Beanstalk for its instances (it will typically have a name like `awseb-e-xxxxxxxxxx-stack-AWSEBAutoScalingGroup-xxxxxxxx-AWSEBSecurityGroup`). Copy the ID of this security group.
2. **Modify Backend Services Security Group:**
    - Go to the **Backend Services Security Group** (created in Section 2.1.1).
    - Edit its inbound rules.
    - Add new rules for each backend service (MySQL, Memcache, RabbitMQ) with the following configuration:
        - **Type:** MySQL/Aurora (Port 3306)
            - **Source:** Paste the ID of the Elastic Beanstalk instance security group.
        - **Type:** Custom TCP (Port 11211) - for ElastiCache (Memcached)
            - **Source:** Paste the ID of the Elastic Beanstalk instance security group.
        - **Type:** Custom TCP (Port 5672) - for Amazon MQ (RabbitMQ)
            - **Source:** Paste the ID of the Elastic Beanstalk instance security group.
    - Save the updated rules.

This step is critical to allow your Elastic Beanstalk application instances to connect to your managed RDS, ElastiCache, and Amazon MQ services.

9. Build & Deploy Application Artifact

This section details how to build your application artifact with the updated backend service endpoints and then deploy it to your Elastic Beanstalk environment. It also covers the final steps of configuring HTTPS on your Elastic Beanstalk Load Balancer and updating your public DNS to point to the Elastic Beanstalk environment.

9.1 Gathering Backend Service Endpoints

Before building your application, you need to gather the connection details for your newly provisioned managed backend services:

- **RDS Endpoint:** From the RDS console, select your MySQL DB instance and copy its endpoint (e.g., `vprofile-mysql-db.xxxxxxxxxxxx.eu-north-1.rds.amazonaws.com`). Also, retrieve the master username and password you set during creation.
- **Amazon MQ Endpoint:** From the Amazon MQ console, select your RabbitMQ broker and copy its URL (e.g., `amqps://b-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.mq.eu-north-1.amazonaws.com:5671`). Note the username and password you created for the broker.
- **ElastiCache Endpoint:** From the ElastiCache console, select your Memcached cluster and copy its configuration endpoint (e.g., `vprofile-memcached-cluster.xxxxxxxx.0001.eu-north-1.cache.amazonaws.com:11211`).

9.2 Updating Application Configuration

Your application's source code needs to be updated to use these new managed service endpoints. This typically involves modifying a configuration file within your project.

1. **Open Project in VS Code:** Open the source code of your V-Profile project (e.g., `https://github.com/hkhcoder/vprofile-project.git`) in Visual Studio Code.

2. **Switch to `awsrefactor` Branch:** Ensure you are on the correct branch that supports the re-architected AWS services: `bash      git checkout awsrefactor`

3. **Edit `application.properties`:** Navigate to `/src/main/resources/` and open the `application.properties` file. Update the following entries with the endpoints and credentials you gathered:

   - **Database (RDS):**
     - Change `db01` to your RDS endpoint.

        - Update the database username and password.
    - **Memcache (ElastiCache):**
        - Change `mc01` to your ElastiCache endpoint.
    - **RabbitMQ (Amazon MQ):**
        - Change `rmq01` to your Amazon MQ endpoint (e.g., `b-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx.mq.eu-north-1.amazonaws.com`).
        - Ensure the port is `5671`.
        - Update the RabbitMQ username and password.

    Save the changes to the `application.properties` file.

9.3 Building the Application Artifact

After updating the configuration, rebuild your application to incorporate the new settings.

1. **Build with Maven:**
    - Open the integrated terminal in VS Code.
    - Navigate to the root of your project.
    - Run the Maven build command: `bash      mvn install`
    - After successful execution, you will find the compiled artifact (e.g., `vprofile-app.war`) in the `target/` directory of your project.

9.4 Deploying Artifact to Elastic Beanstalk

1. **Navigate to Elastic Beanstalk:** Go to your Elastic Beanstalk environment in the AWS Management Console.
2. **Upload and Deploy:**
    - Click on `Upload and Deploy`.
    - Choose the `.war` artifact you just generated from your local system.
    - Click `Deploy`.

    Elastic Beanstalk will now deploy the new version of your application to the environment. This process may take some time as it updates the instances.

9.5 Configuring HTTPS Listener on Elastic Beanstalk Load Balancer

To ensure secure communication, configure the Application Load Balancer managed by Elastic Beanstalk to listen for HTTPS traffic.

1. **Navigate to Elastic Beanstalk Environment:** Select your Elastic Beanstalk environment.
2. **Go to Configuration:**

- In the left-hand navigation pane, choose `Configuration`.
- Under the `Load balancer` category, click `Edit`.

3. **Add HTTPS Listener:**
   - Scroll down to the `Listeners` section.
   - Click `Add listener`.
   - **Port:** `443`
   - **Protocol:** `HTTPS`
   - **SSL certificate:** Select the SSL/TLS certificate you provisioned in AWS Certificate Manager (ACM) for your domain.
   - **Default process:** Select the process that routes to your application instances.
   - Click `Add`.
   - Click `Apply` to save the changes.

9.6 Updating Public DNS with CloudFront

Finally, update your public DNS record to point your domain name to the CloudFront distribution, which in turn points to your Elastic Beanstalk environment.

1. **Copy Elastic Beanstalk Domain Name:** From your Elastic Beanstalk environment dashboard, copy the `Domain` name (e.g., `vprofile-prod-env.xxxxxxxx.eu-north-1.elasticbeanstalk.com`).

2. **Create CloudFront Distribution:**
   - Navigate to the CloudFront service in your AWS Management Console.
   - Click `Create Distribution`.
   - **Origin Domain Name:** Paste the Elastic Beanstalk Domain Name.
   - **Viewer Protocol Policy:** `Redirect HTTP to HTTPS`.
   - **SSL Certificate:** Select `Custom SSL Certificate` and choose the certificate from ACM that matches your domain.
   - Configure other settings as needed (e.g., caching behavior, WAF).
   - Click `Create Distribution`.
3. **Copy CloudFront Distribution Domain Name:** Once the CloudFront distribution is deployed (this can take some time), copy its `Domain Name` (e.g., `dxxxxxxxxxxxx.cloudfront.net`).

4. **Update GoDaddy DNS (or your Domain Registrar):**
   - Log in to your domain management portal (e.g., GoDaddy).
   - Navigate to your domain’s DNS management settings.
   - Add a new DNS record or modify an existing one:

    - **Type:** `CNAME`
    - **Name/Host:** Enter the subdomain you want to use (e.g., `www` or `vprofileapp`).
    - **Value/Points to:** Paste the CloudFront Distribution Domain Name.
  - Save the record.

9.7 Final Verification

After DNS propagation, open a web browser and navigate to your configured domain name (e.g., `https://www.yourdomain.com` or `https://vprofileapp.yourdomain.com`). You should now see your V-Profile application accessible via HTTPS through CloudFront, demonstrating a successful re-architecture and deployment.

10. Conclusion

This project successfully re-architected the V-Profile application to leverage AWS managed services, demonstrating a modern approach to cloud deployment. By migrating from self-managed instances to services like Elastic Beanstalk, RDS, ElastiCache, and Amazon MQ, the application benefits from enhanced scalability, reliability, and reduced operational overhead. The integration of CloudFront further improves performance and global accessibility.

This completes the comprehensive implementation guide for the Re-Architecting AWS Cloud Project.