National College of Ireland

Architecting a High-Availability and Secure Three-Tier Web Infrastructure Using AWS Services

MSc Research Project
Cloud computing

GuruChandra Arivukkarasu Mahalakshmi
Student ID:23307323

School of Computing
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: GuruChandra Arivukkarasu Mahalakshmi

Student ID: 23307323

Programme: Msc cloud computing **Year:** 2024-2025

Module: Research project

Supervisor: Shaguna Gupta

Submission Due Date: 15-09-25

Project Title: Architecting a High-Availability and Secure Three-Tier Web Infrastructure Using AWS Services

Word Count:1428 5 Page

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Guru chandra Arivukkarasu Mahalakshmi

Date: 15-9-25

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Architecting a High-Availability and Secure Three-Tier Web Infrastructure Using AWS Services

GuruChandra Arivukkarasu mahalakshmi
23307323

Configuration Manual

1 Introduction

The configuration guide established elaborate step-by-step process of initiating a safe, scalable, and fault-tolerant three-layer architecture of web application within Amazon Web Services (AWS). The architecture is geared towards satisfying the needs of the modern web applications that need high availability, swift scalability, and good security. The architecture will include a Virtual Private Cloud (VPC) which consists of public subnets and private subnets, a Route 53 DNS service, an Application Load Balancer (ALB) used to distribute traffic intelligently, Amazon EC2 instances at application layer, Amazon RDS owned by database layer, Amazon S3 with CloudFront to deliver static information, and integrated caching (Memcached) and messaging (RabbitMQ) services. It is of special concern to apply a unique security group mapping approach internally, in order to provide a strict control of access between each tier without losing auto-scaled features. The manual is a resource to be used by cloud engineers, students and anyone who would like to have a step-by-step guidance to implementing a enterprise grade AWS system.

2. pre-requisites

- Multiple permission (VPC, EC2, RDS, Route 53, IAM, S3, CloudFront, ACM) on the account involved with AWS.
- A registered domain name can be purchased (using route 53 or external registrar).
- To log in to the EC2, SSH key pair is used.
- Aws the region (e.g. us-east-1).
- The background of basic Linux command knowledge in setting EC2.

3. setting up configuration

- **Vpc creation and subnets**
  Start by creating a VPC that is going to be dedicated and a CIDR block of 10.0.0.0/16 using IPv4. In this VPC, the two public subnets are a load balancer (Application Load Balancer (ALB)) and two private subnets are the application, caching, messaging, and database resources. Connect VPC to Internet Gateway and set up route table accordingly the public route table needs to have route to 0.0.0.0/0 via the Internet Gateway, and the private route table needs no direct internet access. This segregation is made in such a way that valuable assets are kept at arm length.

- **Configuration of security groups(mapping)**
  Make the use of strict security group policies that are specific to a component in order to control communication. Inbound requests on port 80 to HTTP and port 443 to HTTPS should be open to the ALB security group that should forward the request to application server security group over port 8080. The security group of the application server must authorize incoming connections with ALB on the 8080 port and outgoing connections with the database on port 3306 (MySQL), Memcached on port 11211 and RabbitMQ on port 5672. Server security group should be accepting only incoming traffic on port 3306 on the application server only, through security group. Each Memcached and RabbitMQ should be produced with their own security groups that accept traffic inbound only on their respective parts i.e. Memcached and RabbitMQ ports. Such mapping makes it so that only those tiers can talk where it is required, and the attack surface is substantially minimized.

- **Route R3**
  In Route 53, it is a hosted zone to host DNS configuration of the domain of the application. An A record is also added on this hosted zone to directly associate the domain name to a DNS name of the Application Load balancer (ALB). Such a setup would ensure that once visiting the domain address on the browser as a user, Route 53 would resolve the host to the ALB that redirects this request to the applications servers. Consequently, the users will be able to access the app using an easy and pleasing Web address rather than the raw IP address or the AWS-created endpoint.

- **Setting up load balancer**
  Launch the Application Load Balancer into the public subnets, the Application Load Balancer associate with ALB security group. Set up HTTPS on port 443, elliptic curve SSL Certificate CMAC machine, and HTTPS on port 80 to redirect HTTPS in order to make communications secure. Assuming that we have application servers, specify a

target group that will listen on HTTP via port 8080 and then enable it to establish a health check where traffic will be sent to only healthy instances.

- **Ec2 with Apache Tomcat**
  Instances EC2 can be launched to the private subnets and configured with the application security group. Install Java and Apache Tomcat on these instances and after that copy the application WAR file into Tomcat webapps folder. Set the application to talk to Amazon RDS database as persistent storage, Memcached as caching and RabbitMQ as asynchronous messager.

- **Caching layer**
  Launch EC2 instance on the private subnet then assign that instance to the Memcached security group. Install and start using Memcached service and configure to listen to port number 11211. This caching layer will save most frequently accessed data on the memory and lessen the burden on the database and speed up the response time.

- **Messaging layer (Rabbit MQ)**
  Start RabbitMQ security group in ec2 in-house subnet. Install Erlang as a precedent and after that RabbitMQ and in order to achieve simpler administration its management plugin should get activated. RabbitMQ needs to listen on port 5672 to transform RabbitMQ into asynchronous message channel between services.

- **Database layer** select to provision Amazon RDS instance using MySQL engine or you can provision instances in Multi-AZ cluster to achieve highly available cluster and automatic failover. Set the security group that includes the database as well as design the automatic backups. Make sure that the database only accepts inbound connection in the port 3306 to the application server security group.

- **Static content delivery**
  S3 bucket has to be established to host the static application resources such as images, the CSS and JavaScript files. Put them up and grant the proper level of permission to view them. Next, put in place a CloudFront distribution using the S3 bucket as an origin which would allow distributing global content with less latency.

- **Auto scale and monitor**
  Design a launch template of application EC2 instances and apply it to create Auto Scaling Group. Configure the scaling of the policies such that when the cpu utilization efficiency exceeds 70 percent add instances and when the CPU utilization efficiency is below 30 percent remove instances. Tune AWS CloudWatch alerts to monitor the AWS statistics of CPU, memory and network and enable AWS CloudTrail so that every API call is recorded and can be used as an audit and security compliance control.

4. verification and testing methods

Following configuration, it is then recommended that to apply Apache JMeter performance testing to simulate a high traffic environment to ensure that Auto Scaling aligns to proper response. Terminate an EC2 instance to simulate faults tolerance using the ALB rerouting and simulate an RDS fail over. Conduct security tests that should ensure direct database access via the internet is disabled. Last, have the serving of static content done on CloudFront with increased latency.