––Project Assessment

Unit code, name and release number

ITNET313A

Qualification/Course code, name and release number

Cloud Computing (Bachelor of Cybersecurity)

Student details

Student number

801301053

Student name

Murilo de Grandi

Assessment Declaration

- This assessment is my original work and no part of it has been copied from any other source except where due acknowledgement is made.
- No part of this assessment has been written for me by any other person except where such collaboration has been authorised by the assessor concerned.
- I understand that plagiarism is the presentation of the work, idea or creation of another person as though it is your own. Plagiarism occurs when the origin of the material used is not appropriately cited. No part of this assessment is plagiarised.

Student signature and Date

[illegible]	13/04/2022

A4 - Capstone Project Design

ITNET313A – Cloud Computing

Murilo de Grandi - 801301053 **Date:** 01 May 2022

Table of Contents

EXECUTIVE SUMMARY

Example Social Research website on Amazon Web Services (AWS) provides global development statistics data and it is an essential source of information for social science researchers. This project design document initially analysed the current implementation of the website solution. Then, a new design was proposed which leverages additional AWS services such as Elastic Load Balancer (ELB) and Application Auto Scaling to create a more secure, robust, and highly available solution for the Example Social Research website. Finally, the project implementation plan is presented.

1. Introduction

AWS is currently the world-leading cloud provider and encourages web developers to leverage its cloud infrastructure services to deploy cost-effective, highly scalable and fault-tolerant multi-layered web applications that support all types of workloads (AWS 2022a).

A preliminary analysis identified that the current Example Social Research website and database are hosted on a single public-facing EC2 instance. This design model is not recommended for production environments due to limitations in security, performance, scalability and reliability aspects, making the Example Social Research website inefficient and vulnerable to availability issues and Denial of Service (DOS) attacks. Moreover, the current implementation is unnecessarily exposing the database server to the internet, thus creating major security risks for the stored data.

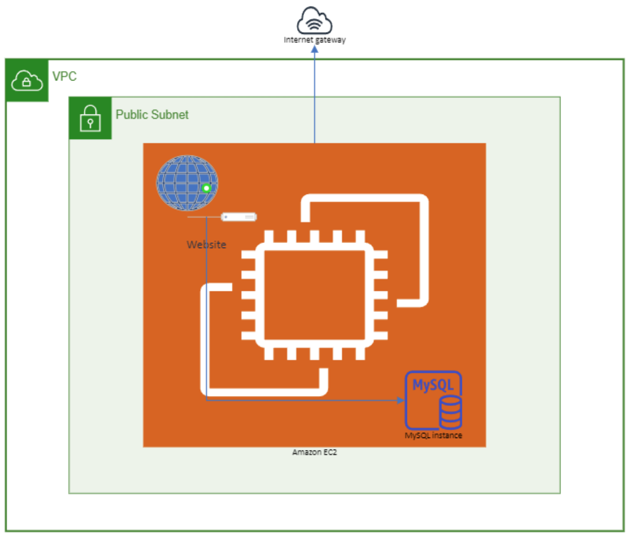
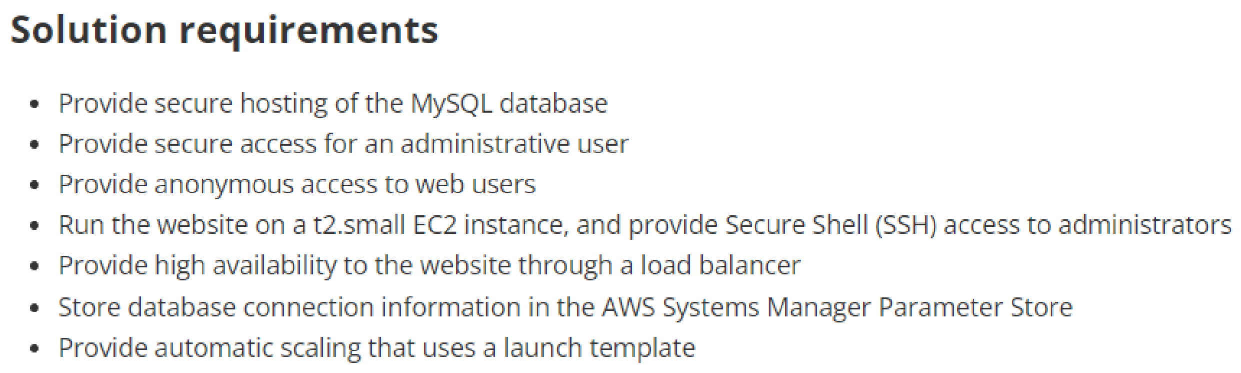


Figure 1: Current website Implementation

The purpose of this project design document is to analyse its current website and database implementations on Amazon Web Services (AWS), aiming at designing and implementing a more robust, efficient and secure website by leveraging the cloud computing infrastructure of AWS.

2. Solution Requirements


Solution requirements

- Provide secure hosting of the MySQL database
- Provide secure access for an administrative user
- Provide anonymous access to web users
- Run the website on a t2.small EC2 instance, and provide Secure Shell (SSH) access to administrators
- Provide high availability to the website through a load balancer
- Store database connection information in the AWS Systems Manager Parameter Store
- Provide automatic scaling that uses a launch template

Figure 2: Solution requirements (Capstone Project 2022).

3. Architecture Design

Reference Design

In order to meet the solution requirements, a layered architecture will be used. The layered architecture approach provides a cost-effective, secure and efficient design for web applications. This client-server model is traditionally used to break down web application architectures into 3 layers (tiers): Presentation tier (user interface), Application tier (business logic), and Data tier (data storage or access), where each tier is developed and maintained as independent modules (Chang Su et al. 2010).

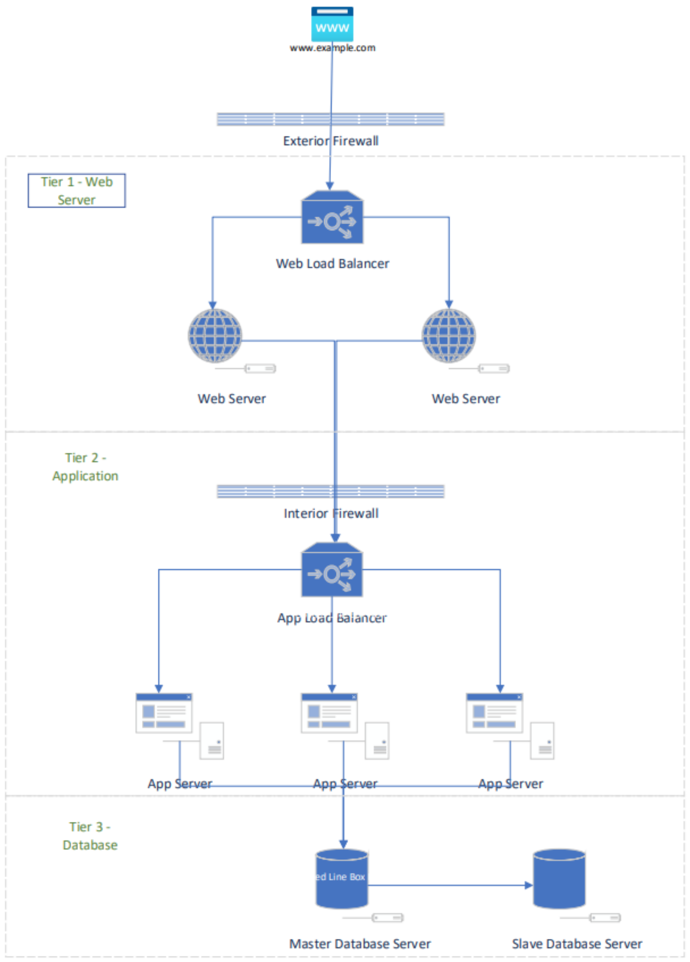


Figure 3: Traditional three-tier architecture. Adapted from (AWS 2022b).

New Architecture Design

Considering that the PHP application will be hosted within the web server, the website design will be based on a 2-tier architecture, allocating the application load balancer and the web server/application in one tier and the database on a separate tier.

The following infrastructure will be deployed to implement the new website:

- 1x VPC (US-East region)
- 1x Network ACL (NACL)
- 2x Availability Zones (AZ)
- 4x Subnets (2x Public – 2x Private)
- 5x Security groups (SG)
- 1x Bastion Host
- 1x Application Load Balancer (ELB)
- 1x Application Auto Scaler (EC2 instances)
- 1x RDS SQL Database (primary – replica)
- 1x Parameter Store

Core Components

The new website is designed to be secure, reliable, and highly available by leveraging a public-facing **Application Load Balancer (ELB)** deployed in two *Availability Zones (AZs)* within the same *VPC*. ELBs are key networking components used to prevent accessibility issues by distributing traffic across different targets (AWS 2022c). The ELB in this case will receive and redirect web requests to an **Application Auto Scaling Group** that is also deployed across the same two AZs.

The implementation will enable the traffic coming from the internet to be routed to a target group of **Web servers** (EC2 instances) located in private subnets within two separated *Availability Zones (AZs)*. This approach also enables organisations to optimize resource utilization while creating a highly available web solution (Sharvani and Resma 2021).

The **RDS SQL database** to serve the PHP application (web server) will be also deployed as a multi-AZ implementation between the two private subnets. It will use a MySQL engine and consists of a primary and a replica node for redundancy with storage auto-scaling feature enabled to provide elastic storage.

Security features

Network Access Control Lists (NACLs) could be configured to restrict access to private subnets at the network edge. However, in this implementation security will be enforced at the host level by using *Security Groups (SGs)*, thus preventing internet traffic from reaching the back end. The configuration and management of the PHP application and SQL database will be performed via the Bastion host, which is an EC2 instance accessible from the internet.

Thalagatti et al. (2015) argue that a Bastion host acts as a proxy to the private instances and should be used for administrative and maintenance purposes of web applications. This implementation allows keeping the web and database servers in private subnets, thus considerably increasing security. An inbound rule will be added to the existing Inventory-App security group (SG) to open port 22 (SSH) exclusively for traffic coming from the Bastion host. Similarly to an inbound network firewall, SGs can limit and control the traffic allowed to reach an EC2 host through protocol, port and source IP filtering (AWS 2022d)

Finally, the database connection information will be stored in the parameter store from the AWS Systems Manager, being only accessible by the web server. According to Beach et al. (2019), this practice improves the security posture of cloud applications by allowing secure sensitive data such as access credentials to be centrally stored rather than exposed in the code.

Additional Considerations

Once the complete solution is deployed successfully in the development environment, the same infrastructure will be replicated as code and the template generated will be used to deploy the final solution to production, thus providing more consistency and minimising chances of human errors. Furthermore, the template will be kept as a backup to enable the deployment of the same solution effortlessly if needed.

Infrastructure as code can be achieved by using the AWS CloudFormation service. CloudFormation templates can be written in JSON or YAML format, enabling easier deployment, update and redeployment of infrastructure resources (AWS 2022e).

New Design Diagram

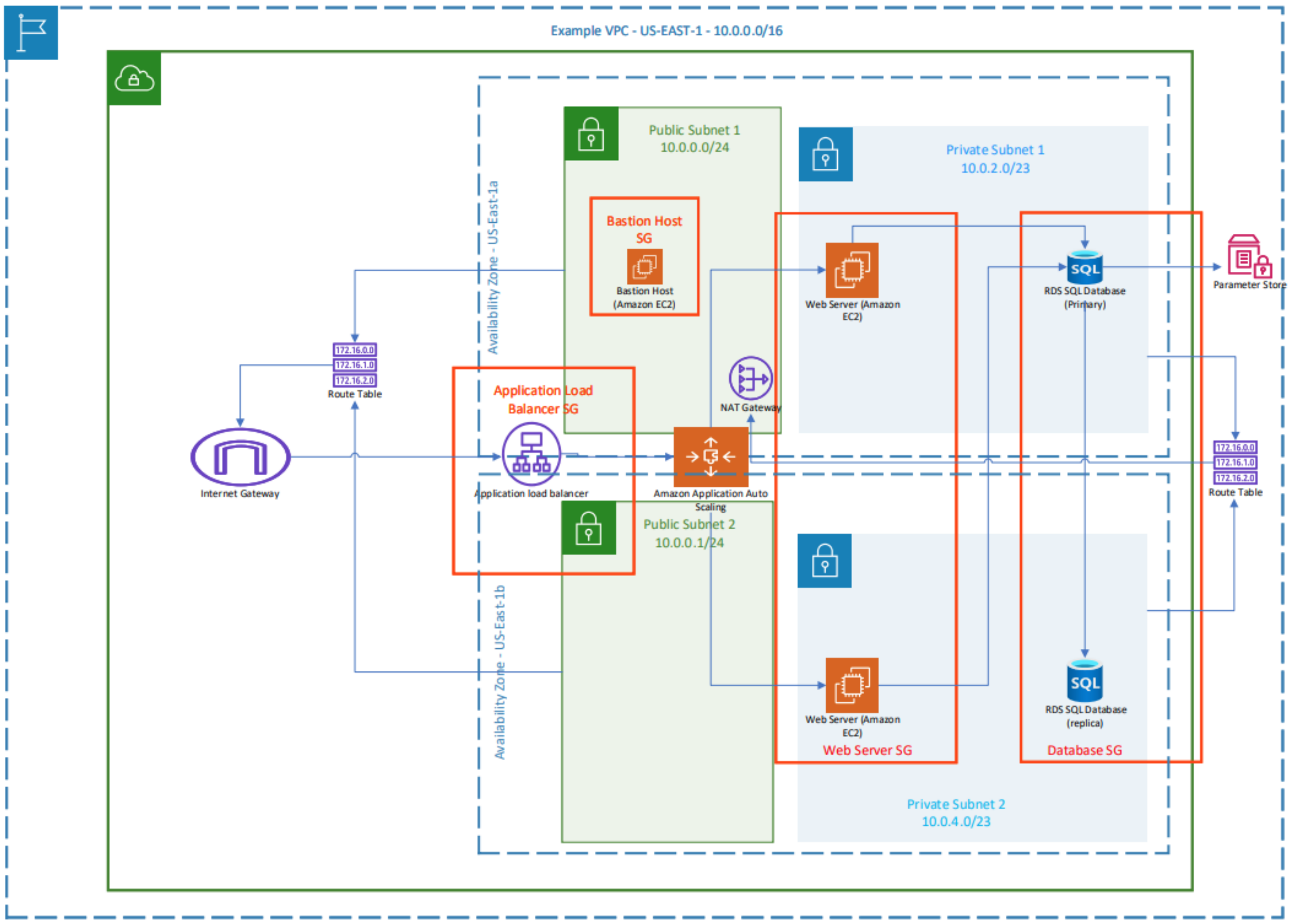


Figure 4: New website architecture design.

4. Implementation Plan

Deliverables

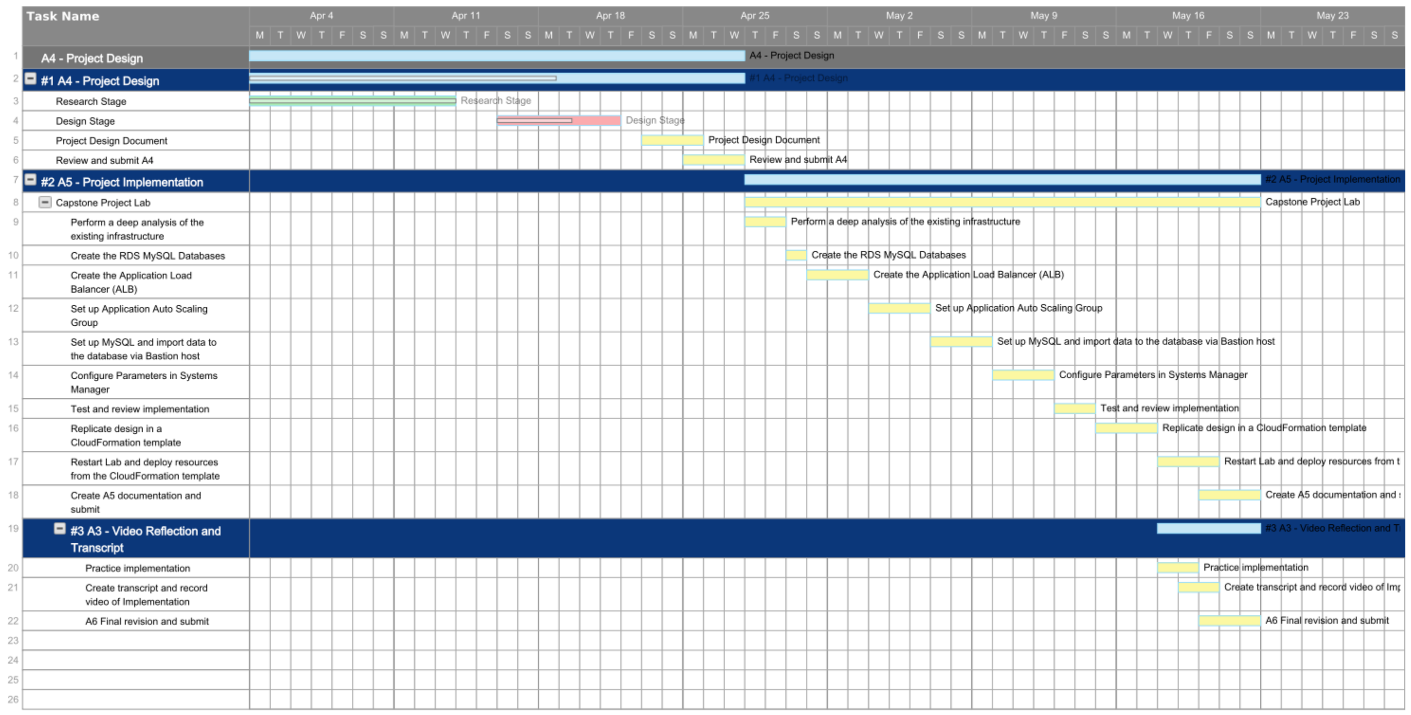
- To create an **Application Load Balancer (ALB)** spanning public subnets 1 and 2.
- To launch the web server from the **Application Auto Scaling group** using the Example-LT template provided.
- To create a Multi-AZ **RDS MySQL database** configured with the SQL data provided.
- To configure the **Systems Manager** with parameters for the RDS database.
- Develop an **AWS CloudFormation template**.
- Deploy the infrastructure as code.

Milestones

Description	Forecast Date	Approval
#1 A4 - Project Design	27/04/2022	Teacher: Adnan Syed
#2 A5 - Project Implementation	22/05/2022	Teacher: Adnan Syed
#3 A6 – Video Reflection and Transcript	22/05/2022	Teacher: Adnan Syed

Table 1: Milestones

Gantt Chart



Capstone Project

	Task Name
1	A4 - Project Design
2	#1 A4 - Project Design
3	Research Stage
4	Design Stage
5	Project Design Document
6	Review and submit A4
7	#2 A5 - Project Implementation
8	Capstone Project Lab
9	Perform a deep analysis of the existing infrastructure
10	Create the RDS MySQL Databases
11	Create the Application Load Balancer (ALB)
12	Set up Application Auto Scaling Group
13	Set up MySQL and import data to the database via Bastion host
14	Configure Parameters in Systems Manager
15	Test and review implementation
16	Replicate design in a CloudFormation template
17	Restart Lab and deploy resources from the CloudFormation template
18	Create A5 documentation and submit
19	#3 A3 - Video Reflection and Transcript
20	Practice implementation
21	Create transcript and record video of Implementation
22	A6 Final revision and submit
23	
24	
25	
26	

Figure 5: Gantt Chart

References

AWS 2022a, *Key components of an AWS web hosting architecture – AWS Whitepaper*, Web Application Hosting in the AWS Cloud, viewed 20 April 2022, <https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/key-components-of-an-aws-web-hosting-architecture.html>.

AWS 2022b, *An overview of traditional web hosting*, Key components of an AWS web hosting architecture – AWS Whitepaper, viewed 21 April 2022, <https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/an-overview-of-traditional-web-hosting.html>.

AWS 2022c, *Elastic Load Balancing*, Networking and Content Delivery, viewed 21 April 2022, <https://aws.amazon.com/elasticloadbalancing/>.

AWS 2022d, *Amazon Virtual Private Cloud (Amazon VPC)*, Network and Content Delivery, viewed 27 April 2021, <https://aws.amazon.com/vpc/>.

AWS 2022e, *AWS CloudFormation Documentation*, AWS Docs, viewed 27 April 2021, <https://docs.aws.amazon.com/cloudformation/index.html>.

Beach, B, Armentrout, S, Bozo, R & Tsouris, E 2019, *Pro PowerShell for Amazon Web Services*, eBook, viewed 25 April 2022, <https://link.springer.com/content/pdf/10.1007/978-1-4842-4850-8.pdf>.

Chang Su, K, Hooi Yin, L & Yong Ju, L 2010, *Design and Implementation of Secure 3-Tier Web Application with Open Source Software*, eBook, viewed 20 April 2022, <https://www.koreascience.or.kr/article/JAKO201007762975106.pdf>.

Nadon, J 2017, *Website Hosting and Migration with Amazon Web Services*, eBook, viewed 26 April 2022, <https://link.springer.com/content/pdf/10.1007/978-1-4842-2589-9.pdf>.

Sharvani, GS & Resma, KS 2021, 'An Auto-Scaling Approach to Load Balance Dynamic Workloads for Cloud Systems', *Turkish Journal of Computer and Mathematics Education*, vol. Vol.12, no. 11, viewed 24 April 2022, <https://www.turcomat.org/index.php/turkbilmat/article/view/5915/4928>.

Thalagatti, MP, Asrar Naveed, M & Chaitra, B Acharya Institute of Technology 2015, 'Reliable Data Tier Architecture for Job Portal using AWS', *(IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 6, viewed 26 April 2022, <http://13.232.72.61:8080/jspui/bitstream/123456789/353/1/Reliable%20Data%20Tier%20Architecture%20for%20Job%20Portal%20using%20AWS.pdf>.