

Project Thesis

Multi-Zone High Availability Cloud Solution

Author: Abdelrahman Mohamed Ezzat Said Ghazy

Submission Date: 14th of July, 2025

Project Thesis

Multi-Zone High Availability Cloud Solution

Author: Abdelrahman Mohamed Ezzat Said Ghazy

Submission Date: 14th of July, 2025

This is to certify that:

- (i) the thesis comprises only my original work

Abdelrahman Mohamed Ghazy
14th of July, 2025

Acknowledgment

I would like to express my sincere gratitude to Manara for their invaluable support and guidance throughout my academic journey. Participating in their program was a transformative experience that significantly enhanced my technical, professional, and personal development.

Manara's mentorship, training sessions, and career support provided me with the skills and confidence needed to pursue ambitious goals in the cloud and software engineering fields. Their commitment to empowering talent across the MENA region is truly inspiring, and I am deeply thankful for being part of such a purpose-driven community.

Abstract

This thesis explores the development and implementation of a multi-zone, high-availability cloud system using Amazon Web Services (AWS). The process starts with a detailed analysis of the issue statement, followed by an overview of the solution's implementation. Include the essential building components and testing cases. The article finishes with practical consequences for the developed solution.

The system uses several AWS services, including VPC, ELB, and Route 53. The goal is to design a system that is highly available, fault-tolerant, and can withstand active-active scenarios and active-passive arrangements to help prevent unforeseen failures and tragedies. Active-active deployments allow uninterrupted operation even during unexpected occurrences, whereas active-passive arrangements provide redundancy by using one EC2 instance as the primary and another as a standby to assume control in case of failure.

The system's resilience to disasters is demonstrated through testing scenarios, such as deliberately terminating EC2 instances in one availability zone to imitate data center shutdowns. Furthermore, active/passive setups are evaluated. By abruptly halting the primary EC2 instance and switching to the backup instance.

The testing results show that the system can resist instance failures and provide flawless operation for users. This makes it a dependable and highly available option for enterprises that need uninterrupted access to computing resources. This thesis offers a complete approach on establishing a highly available cloud solution on AWS for enterprises and organizations in many industries.

Contents

Acknowledgments	V
1 Introduction	1
1.1 Introduction	1
1.2 Motivation	1
1.3 Aim of the Project	1
1.4 Thesis Organization	2
2 Background	3
2.1 Introduction	3
2.2 Definition of Virtualization	3
2.3 Hypervisors	5
2.4 Types of Virtualization	5
2.4.1 Full Virtualization	5
2.4.2 Para-virtualization	6
2.4.3 Operating System Virtualization	6
2.4.4 Hardware-assisted Virtualization	6
2.5 Benefits of Virtualization	6
2.6 Cloud Computing	7
2.6.1 Introduction to Cloud Computing	7
2.6.2 Definition and Characteristics	7
2.6.3 Cloud Service Models	8
2.6.4 Cloud Deployment Models	9
2.6.5 Benefits of Cloud Computing	9
2.6.6 AWS Overview	10
2.7 High Availability	10
2.7.1 Definition	10
2.7.2 Benefits	11
2.8 High Availability Architecture	11
2.8.1 High Availability Components	12
2.8.2 Workflow for Ensuring High Availability	13
2.8.3 Benefits of High Availability Architecture	13
2.9 Challenges in High Availability	14
2.9.1 Complexity	14

2.9.2	Cost Management	14
2.9.3	Application Dependencies	14
2.9.4	Data Inconsistency	15
2.9.5	Fault Tolerance	15
2.9.6	Lack of Visibility	15
2.9.7	Configuration Management	15
3	Methodology	17
3.1	Project Overview	17
3.1.1	Project Ontology	17
3.1.2	Relations	20
3.2	Project Challenges	21
3.2.1	Security	21
3.2.2	Application Compatibility	22
3.2.3	System Complexity	22
3.3	Additional Enhancements: Automation, Performance, and Monitoring . .	23
3.3.1	Infrastructure as Code (IaC) with AWS CloudFormation	23
3.3.2	Full Annotated CloudFormation YAML Template	23
3.3.3	Performance Optimization with Amazon CloudFront	30
3.3.4	Operational Monitoring with Amazon CloudWatch	30
3.3.5	Summary	31
4	Testing	33
4.1	Testing	33
4.2	Active-Active	33
4.2.1	Test Case 1: Terminate Instance in Availability Zone	33
4.2.2	Setup	33
4.2.3	Testing Steps	34
4.2.4	Results	34
4.2.5	Conclude the test	35
4.3	Active-Passive	35
4.3.1	Test Case 2 : Terminate The Primary Instance	36
4.3.2	Setup	36
4.3.3	Testing Steps	36
4.3.4	Results	37
4.3.5	Conclude the test	37
5	Cost Analysis	39
5.1	Active-Active Architecture	39
5.2	Active-Passive Architecture	40
5.3	Cost-Availability Trade-off	40
5.4	Conclusion	41
6	Conclusion	43

Appendix	44
A Lists	45

Chapter 1

Introduction

1.1 Introduction

To be competitive in today's fast-paced digital world, businesses cannot afford downtime. Success depends on uninterrupted access to vital services. Multi-zone, high-availability cloud Solutions step in, acting as a bulwark against disturbances. Our solutions, available in active-active and active-passive configurations, are precisely developed to enable flawless operation across various zones. This provides organizations with the confidence to supply uninterrupted service even under unanticipated difficulties..

1.2 Motivation

This project aims to create a reliable cloud infrastructure for enterprises and organizations of all sizes, ensuring high availability of computing resources. In today's digital environment, downtime poses a major Risks to operations, customer happiness, and overall business performance. Modern corporate settings prioritize scalability, dependability, and resilience, making traditional on-premises infrastructure insufficient. This project aims to provide a high-availability cloud system that can survive disturbances and provide uninterrupted service delivery. For example, in March 2019, Facebook crashed. The catastrophe caused severe losses and interruptions, emphasizing the need for dependable and robust infrastructure. By addressing the issues raised by such instances, this initiative hopes to reduce such dangers for businesses and organizations in the future.

1.3 Aim of the Project

This project aims to develop and construct a multi-zone, high-availability cloud system utilizing Amazon Web Services (AWS). This system will be highly scalable and

fault-tolerant, with the capability to provide continuous service delivery across various zones. The project will provide redundancy and failover solutions across AWS availability zones to reduce downtime caused by hardware failures, network difficulties, or interruptions.

1.4 Thesis Organization

- **Chapter 1: Introduction:** This chapter presents the introduction and motivation for the research topic, as well as the background and objectives of the study.
- **Chapter 2: Background:** This chapter provides a comprehensive overview of related work and technologies relevant to the research, including cloud computing, Amazon Web Services, and high availability solutions.
- **Chapter 3: Methodology:** This chapter describes the architecture and implementation of the infrastructure used in the research, as well as the experimental setup and the use of AWS services to ensure high availability includes the results of the testing scenario,
- **Chapter 4: Testing:** This chapter includes the results of the testing scenario, such as the performance of the infrastructure under failures and disasters for both types (active-active),(active-passive)
- **Chapter 5: Conclusion:** This chapter presents the conclusions drawn from the research
- **Chapter 6: Future Work:** This chapter outlines potential future work that could build upon the research presented in this thesis

Chapter 2

Background

2.1 Introduction

Cloud computing has transformed current company operations, making it a widely used word in IT. Its disruptive impact goes beyond just utilizing IT infrastructure and significantly alters The paradigm for resource management. Cloud computing allows organizations to access a dynamic reservoir of computing assets, including servers, storage, and applications, leading to increased agility and efficiency. Cloud computing has become increasingly important, with many businesses using it to improve operations, decrease costs, and respond to market changes.

The remainder of this chapter is arranged as follows. Section 2.2-2.4 defines and explores virtualization, including hardware-assisted virtualization. Hypervisors are crucial components. This article demonstrates how virtualization improves resource optimization and operational efficiency. In Section 2.5, we explore the fundamentals of cloud computing, including its service models, deployment methodologies, and advantages. In Section 2.6, we discuss the necessity and benefits of high availability in providing ongoing access to key services. Section 2.7 explores high availability architecture, including components and concepts for constructing resilient IT infrastructures. Our organized talks attempt to offer a thorough comprehension of the fundamental ideas and technology required for the analysis presented in this thesis.

2.2 Definition of Virtualization

Virtualization [1] creates virtual versions of servers, storage, networks, and other real equipment. Virtual software mimics actual hardware, allowing for simultaneous operation. Figure 2.1 depicts many virtual machines running on a single physical computer. This strategy maximizes hardware consumption while simultaneously improving flexibility and scalability in IT infrastructures. Virtualization is the foundation of cloud computing, allowing enterprises to manage infrastructure effectively and optimize ROI.

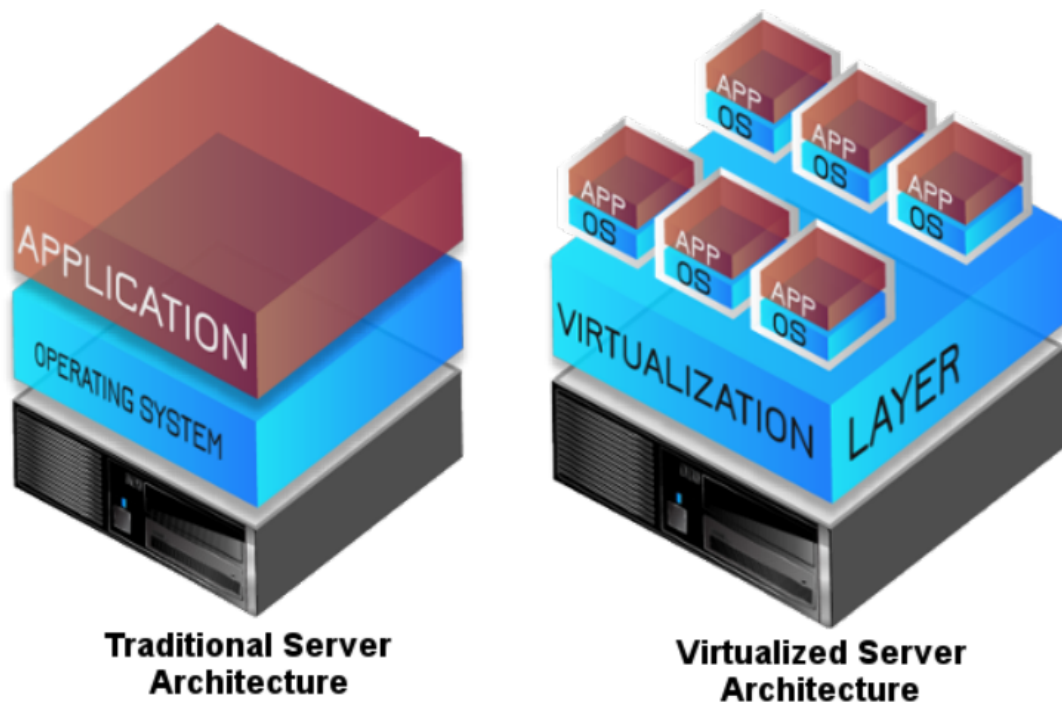


Figure 2.1: Comparison between Traditional and Visualized Servers

Virtualization allows for more flexible access to hardware resources, without the limits of traditional servers like power usage, storage capacity, and maintenance. Virtualization overcomes these constraints. The functionality of physical hardware is being abstracted into software. This enables physical infrastructure to be managed, maintained, and utilized similarly to web-based applications. There are three forms of virtualization: complete virtualization, para-virtualization, and hardware-assisted virtualization. Each kind has benefits and drawbacks, and the choice is based on individual requirements and use cases.

Virtualization allows for the independence of programs and infrastructure, enabling servers to be used by several applications running remotely.

2.3 Hypervisors

Hypervisors construct and manage virtual computers on a real host machine. Virtual machines (VMs) may run several operating systems [2] on the same hardware due to their isolation from the host machine and other VMs.

There are two kinds of hypervisors:

- **A type 1 hypervisor**, or bare-metal hypervisor, maintains guest virtual machines directly on the host machine's hardware. Examples of type 1 hypervisors are VMware ESXi, Microsoft Hyper-V, and Citrix Hypervisor.
- **Type 2 hypervisors**, or hosted hypervisors, use a host operating system to generate and operate guest virtual machines (VMs). Examples of type 2 hypervisors are Oracle VirtualBox, VMware Workstation, and Parallels Desktop.

2.4 Types of Virtualization

Modern computing systems often incorporate various forms of virtualization [3]. This includes:

2.4.1 Full Virtualization

Virtualization enables many guest operating systems to operate on a single physical machine without requiring any changes. Each guest operating system is oblivious of its virtualization and interacts with the hypervisor's virtual hardware functions similarly to actual hardware.

2.4.2 Para-virtualization

Allows numerous guest operating systems to operate simultaneously on a single physical computer. Para-virtualization, unlike full virtualization, needs guest operating systems to be aware of the virtualization layer. This enables better. Enhanced performance by enabling direct communication between guest operating systems and the hypervisor.

2.4.3 Operating System Virtualization

Operating system virtualization, or containerization, creates segregated user-space instances within a single kernel. Each container shares the host operating system's kernel and functions as an independent environment, enabling a lightweight and efficient type of virtualization.

2.4.4 Hardware-assisted Virtualization

This sort of virtualization uses hardware extensions like Intel VT-x or AMD-V to enhance its efficiency and performance. Hardware-assisted virtualization enables direct processor assistance for virtualization activities, decreasing overhead. connected with software-based virtualization methods.

2.5 Benefits of Virtualization

Some of the advantages are:

1. **Improve Resilience and Reduce Downtime:** Virtual machines may be easily relocated to another server in case of hardware failure or other difficulties, facilitating disaster recovery.
2. **Increase productivity, efficiency, and agility** by reducing the number of servers your team has to manage. This saves time on technical support. You can utilize this time to adapt to shifting settings and focus on important activities.
3. **Improved System Security:** Virtualization separates applications and operating systems, lowering the risk of security breaches and malware infestations.
4. **Sustainability:** Virtualization reduces the need for physical servers and lowers power consumption. This decrease drastically lowers the carbon footprint of the data center.
5. **Resource optimization:** Virtualization enables enterprises to dynamically distribute resources to virtual computers, improving resource consumption and performance.

2.6 Cloud Computing

Cloud computing has gained popularity in recent years. Users can use computer resources on-demand and pay based on usage. Cloud computing has various benefits, including scalability, flexibility, Reliability and cost savings.

2.6.1 Introduction to Cloud Computing

This overview of cloud computing highlights its transformational impact in current technological environments. Cloud computing fundamentally alters how computer resources are accessed, managed, and consumed. By harnessing the power, Cloud computing provides exceptional scalability, flexibility, and cost-efficiency for both enterprises and people.

2.6.2 Definition and Characteristics

Cloud computing provides on-demand computing resources, including servers, storage, databases, networking, software, and analytics, via the internet. A virtual pool of resources may be quickly supplied and removed with minimum administrative effort. Cloud computing allows enterprises to save capital costs by offering a pay-per-use approach for IT services. Cloud computing enables enterprises to adjust resources based on demand, minimizing the requirement for underutilized capacity. NIST defines cloud computing as "a model for enabling ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services)." This technology allows for quick provisioning and release with minimum management or service provider engagement.

Cloud computing has five key characteristics:

- **On-demand self-service:** enables autonomous provisioning and release of resources without human interaction.
- **Broad network access:** Cloud resources may be accessed from any internet-connected device.
- **Resource pooling:** is the capacity to share computing resources across several users.
- **Rapid elasticity:** Automatically scale resources to meet demand.
- **Measured service:** Monitoring and charging for cloud resources depending on consumption.

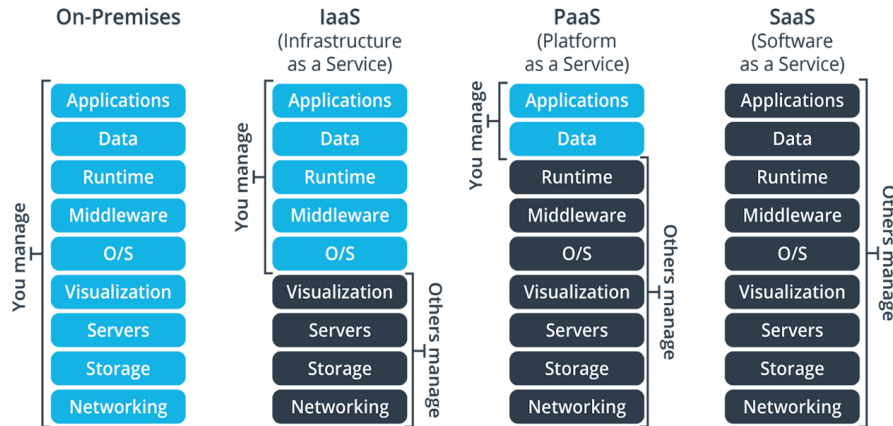


Figure 2.2: Cloud Service Models

2.6.3 Cloud Service Models

Cloud computing offers three service models: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Each model has unique features and functionalities, as seen in the image above :

Infrastructure as a Service (IaaS)

is a cloud computing concept that delivers virtualized computer resources via the internet. IaaS allows customers to rent virtual servers, storage, networking, and other computer infrastructure components on a pay-as-you-go model.

Platform as a Service (PaaS)

is a cloud computing platform that enables users to design, deploy, and maintain applications without relying on underlying infrastructure. It provides tools and frameworks that Streamline the development process so developers may concentrate entirely on code and deployment. Examples include Microsoft Azure App Service and Google App Engine.

Software as a Service (SaaS)

refers to cloud-hosted software programs that consumers may access via the internet. The supplier oversees the infrastructure, operating system, and applications, while consumers control their own data. SaaS eliminates the need for local installation and maintenance, providing both convenience and cost savings.

2.6.4 Cloud Deployment Models

Cloud deployment models [6] describe how cloud services are delivered and accessed. There are four major deployment models:

Public Cloud

like a massive internet-based marketplace for computer resources. This platform allows for on-demand access to third-party storage, apps, and services. This solution is cost-effective and adaptable, making it ideal for firms who want to simplify their IT without maintaining their infrastructure.

Private Cloud

is a bespoke cloud solution for a single enterprise. This solution provides safe and customized computing resources behind a firewall, ideal for enterprises seeking control and privacy over their data. Private clouds can be implemented on-premises or in a third-party data center, and maintained either internally or by a provider.

Hybrid Cloud

offers the advantages of both cloud computing models. This hybrid approach mixes public and private clouds, enabling enterprises to utilize on-premises infrastructure, private cloud services, and public cloud resources. It provides flexibility, scalability, and control, allowing enterprises to customize their IT infrastructure for individual demands and workloads.

2.6.5 Benefits of Cloud Computing

- Cloud computing provides various advantages over traditional computer paradigms. Some of the most prominent benefits include:
- **Availability:** Cloud services allow for remote work and communication among teams in multiple places, accessible from any internet-connected location.
- **Scalability:** Cloud services can simply scale up or down to meet changing demand, allowing organizations to optimize resource utilization and minimize over-provision.
- **Cost reduction:** Cloud computing eliminates the need for significant upfront expenditures in hardware and infrastructure. Instead, customers pay for resources as needed, lowering total IT expenses.

- **Security:** Cloud providers spend extensively on security measures to safeguard data and infrastructure from cyber attacks. They use modern security technology and industry best practices to maintain data confidentiality and integrity.

2.6.6 AWS Overview

Amazon Web Services (AWS) is a popular cloud computing platform developed by Amazon. It offers a wide range of services for organizations of all sizes, including startups and major corporations.

AWS's platform offers flexibility, scalability, and cost-effectiveness, among other benefits (see image below).

Businesses aiming to design, implement, and manage cloud-based apps and IT infrastructure. AWS's pay-as-you-go pricing model and worldwide data center network provide dependable and high-performance cloud services, allowing businesses to innovate and thrive in the digital economy.

2.7 High Availability

Modern distributed systems require high availability to ensure continuous operation despite hardware or software faults. Implementing redundancy in architecture reduces the danger of a single failure.

Failure point. Redundancy can take several forms, including data replication over multiple nodes, deploying multiple copies of the same application across different availability zones or regions, and using load balancers to distribute traffic evenly among healthy instances. A high availability design distributes workload over numerous components, ensuring almost constant uptime and eliminating costly interruptions that impact revenue and customer satisfaction.

2.7.1 Definition

High Availability (HA) refers to a system or service's capacity to operate continuously for an extended period of time without interruption. It comprises developing strategies to reduce downtime and guarantee continuity.

Ensure key resources are available despite hardware, software, or other disturbances. HA prioritizes service uptime and dependability, ensuring consumers can access important services without disruptions or outages.

2.7.2 Benefits

High Availability promotes cloud applications by instilling confidence and trust among users, clients, and stakeholders through a Service Level Agreement (SLA) with a pledge of "100 percent". It ensures service providers adhere to agreed-upon performance metrics and criteria. Clearly declaring the SLA with the 100 percent commitment promotes accountability and openness in service delivery. Aligning SLAs with performance goals reduces financial penalties for violations and protects both parties' interests.

- **Enhanced performance:** High availability systems use redundant components and load balancing to manage workload variations and improve performance and responsiveness.
- **Enhanced Disaster Recovery:** High availability designs often include sophisticated disaster recovery measures, such as data replication and failover systems, to maintain business continuity during unanticipated catastrophes. By reproducing data over time and using automatic failover operations in geographically scattered sites, enterprises may limit catastrophe damage and swiftly recover from interruptions, decreasing downtime and data loss.
- **Enhanced Reliability:** High availability systems minimize single points of failure and offer redundancy, keeping the system operational even if a component fails. This improved dependability reduces the danger of service interruptions and downtime, ensuring enterprises have ongoing access to key resources.

High availability is an important property of cloud computing and is employed by various clouds. Services that offer consistent, simple, and cost-effective applications. Implementing High Availability may be complex and requires proper design and configuration.

2.8 High Availability Architecture

High availability architecture guarantees that IT systems and services are available to users for a considerable amount of the time, including planned and unscheduled disruptions. High availability is often assessed as the proportion of

The system aims for 99.999 percent availability (commonly known as "Five Nines" dependability). High availability is achieved through two architectures: Active-Active (AA) and Active-Passive (AP), as shown in Figure 2.6. An Active-Active architecture consists of many redundant systems that serve user requests concurrently. This method is suited for high-traffic cases like streaming services. It distributes the burden across multiple components, ensuring uninterrupted operation even if one fails.

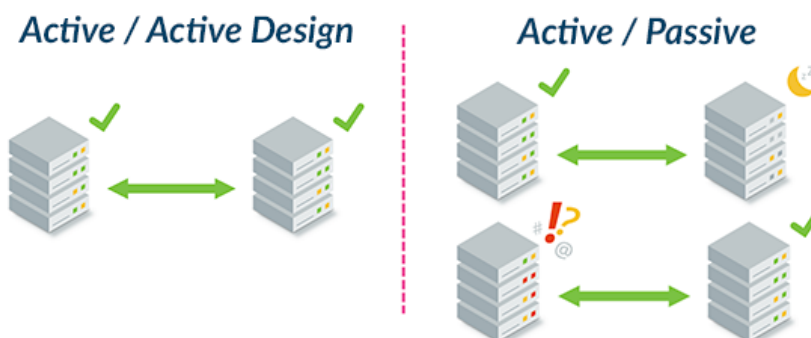


Figure 2.3: High Availability Clusters

An Active-Passive design maintains redundant systems in a standby mode, with one actively serving user requests and the other remaining on standby. This architecture is widely used for scenarios that include database replication.

System upgrades include using one system while having another ready to take over in case of breakdown or repair. Each design offers benefits and use cases, based on application needs, resource restrictions, and desired redundancy. Understanding these designs helps firms create durable and resilient systems suited to their unique needs.

2.8.1 High Availability Components

High-availability architecture often consists of the following components:

- Numerous Availability Zones: AWS supports numerous availability zones within a region, which are physically independent data centers that allow enterprises to deploy resources across different zones to assure high availability .
- Load balancers divide incoming traffic over many resources to optimize usage and minimize overloading. This improves fault tolerance and dependability.
- Amazon Route 53 is a scalable domain name system (DNS) web service offered by AWS. The system directs users to relevant resources based on routing policies, health checks, and latency metrics. Route 53 offers DNS failover, enabling enterprises to automatically reroute traffic to healthy endpoints in case of failure.
- High availability designs use redundant networking components, such as switches, routers, and routes, to reduce network failures and provide continuous communication.
- Automated Fail-over Systems: These systems monitor the health and performance of architectural components and shift traffic to healthy ones, reducing downtime and service interruptions.

- **Monitoring and alerting systems** regularly monitor the health and performance of architectural components, generating alerts or notifications for abnormalities or failures. This allows for proactive management and prompt reaction to concerns.

2.8.2 Workflow for Ensuring High Availability

The procedure for achieving high availability generally consists of the following steps:

1. **Assess needs:** Determine the system or application's high availability needs, such as uptime, performance, and disaster recovery objectives.
2. **Architecture Design:** Create a high-availability architecture with redundancy, fault tolerance, and automatic failover procedures to reduce downtime and maintain continuous operation. This design may use active-active or active-passive topologies, various AWS availability zones, and services like Amazon Route 53 for DNS routing and health checks.
3. **Implement the high availability design** by installing redundant components, establishing load balancers, automating failovers, and enabling data replication as needed. Configure Amazon Route 53 to direct traffic to certain endpoints depending on health checks and failover policies.
4. **Testing** the high availability architecture to ensure it meets uptime and performance requirements. This may involve simulated failure scenarios, load testing, and disaster recovery drills.
5. **Monitoring and Maintenance:** Use robust monitoring and alerting systems to continually check component health and performance throughout the architecture.
6. Regularly execute maintenance chores, such as software updates and hardware repairs, for maximum performance.
7. **Continuously optimization** and iterate on high availability architecture using monitoring data, performance indicators, and input from testing and maintenance. Improve dependability and resilience by identifying and implementing improvement opportunities.

2.8.3 Benefits of High Availability Architecture

Some of the rewards include the following:

- **Improved performance:** Highly available designs use load balancing to distribute traffic across numerous servers, resulting in quicker response times and optimal resource use.

- **Enhanced fault tolerance** Highly available architectures with redundant systems reduce the risk of data loss or service disruption by immediately assuming control during failures.
- **Resource Utilization:** Auto Scaling adjusts resources based on demand, reducing waste during low traffic periods.
- **Improved uptime:** Highly available designs reduce downtime due to hardware or software problems, leading to higher user satisfaction.
- **Reduced risk:** Highly available architectures reduce the risk of catastrophic data loss or system failure by employing backup techniques and failover systems to maintain data integrity and stability.

2.9 Challenges in High Availability

Although High Availability improves cloud infrastructure, it also presents difficulties that must be handled for successful and efficient operation. Some of the main issues in high availability are described below:

2.9.1 Complexity

Continuous availability in high availability systems adds complexity to cloud infrastructure. Sophisticated algorithms are necessary to handle workload patterns, application requirements, and cost optimization. Implementing Managing algorithms may be complex and requires particular skills and knowledge.

2.9.2 Cost Management

One of the problems of high availability design is successfully managing expenses. Although redundancy and failover measures are necessary for continuing operation, they might contribute to higher infrastructure expenditures. Balancing redundancy and cost demands careful planning and optimization. Also, dynamically Managing resources to fulfill demand and save costs is a difficult process that requires advanced cost management methods and technologies.

2.9.3 Application Dependencies

To minimize unforeseen consequences, consider application dependencies while implementing high availability. Increasing database instances may not enhance availability if other variables, like as network bandwidth or CPU capacity, hinder performance. Understanding these relationships is crucial for efficient resource allocation and system resilience.

2.9.4 Data Inconsistency

Maintaining data consistency in a highly accessible system is tough owing to scattered resources. Synchronization difficulties between replicas or cached data can cause inconsistencies and compromise system integrity. Implementing Effective data synchronization and error management are crucial for reducing data inconsistency in high availability designs.

2.9.5 Fault Tolerance

Fault tolerance is crucial in high availability designs to prevent downtime due to hardware or software problems. This is accomplished by allocating resources across several availability zones and ensuring that availability choices are consistent. Has a detrimental influence on application availability. Failing to consider fault tolerance might result in protracted downtime or data loss.

2.9.6 Lack of Visibility

To ensure high availability, it's important to continuously monitor and analyze resource use to detect possible issues before they impact application availability. However, little insight into resource utilization might make it difficult to notice and respond to potential issues ,difficulties, which may result in downtime or other availability concerns.

2.9.7 Configuration Management

Proper configuration management is critical to guaranteeing high availability. Maintaining consistency between instances and limiting configuration drift is crucial to avoid performance concerns or security vulnerabilities. Effective configuration management is critical for ensuring the stability and dependability of cloud infrastructure.

Chapter 3

Methodology

3.1 Project Overview

The project intends to provide a highly available cloud solution on Amazon Web Services (AWS) to meet scalable and dependable infrastructure requirements. The initiative aims to address the growing need for continuous computer resources.

Using AWS services to build a strong architecture that can tolerate failures and maintain continuous operation. This project involves developing and deploying a multi-zone high availability architecture in the AWS environment. The project scope includes creating VPC setups, routing tables, deploying EC2 machines across different availability zones, integrating load balancers for traffic distribution, and implementing DNS routing using Route 53. The project aims to improve system dependability, enhance scalability, and optimize resource consumption. The project will use AWS's flexible architecture and services to provide high availability and fault tolerance.

Ensure smooth operation even amid unforeseen situations like instance failures or network outages. This project uses a systematic way to build, implement, and test cloud infrastructure. Effective planning during peak hours and scaling down during low demand optimizes costs and resources. The project aims to provide a complete guide for creating a highly available cloud solution on AWS, meeting the unique demands of enterprises and organizations in various industries.

3.1.1 Project Ontology

The project ontology represents essential ideas and components in the active-active and active-passive architectures being studied. These principles are crucial for understanding the operation of both systems:

- **Amazon VPC:** enables customers to install AWS resources within a self-configured virtual network. This virtual network offers customers more control and protection over AWS resources hosted within it.

- **An Internet Gateway:** connects resources within a Virtual Private Cloud (VPC) to the internet. It connects outbound traffic to the internet and receives inbound traffic. Resources in the VPC. Internet Gateways enable connection for public-facing services and applications hosted in the AWS cloud environment.
- **Routing Table:** A set of rules that route network traffic within a VPC. It specifies traffic routes depending on destination. Routing tables are coupled with subnets and establish local routes. This includes traffic within the VPC and routes to the internet and other networks. Configuring routing tables allows users to regulate traffic flow between subnets, internet gateways, virtual private gateways, and other network devices.
- **Public Subnets:** These are sectors of a VPC's IP address range that connect to the Internet Gateway, allowing resources inside the subnet to communicate directly with the internet. Public subnets are commonly used to host public-facing Services and apps that require internet access. Security groups and network access control lists (ACLs) provide public resource access while ensuring security.
- **Amazon Elastic Compute Cloud (EC2):** provides scalable computing resources in the AWS cloud. The auto scaling group manages EC2 instances, which serve as the cloud application's hosting infrastructure.
- **Load balancers:** ensure application availability and responsiveness by spreading network traffic evenly across numerous servers. In this project, an Application Load Balancer is used to spread traffic among EC2 instances.
- **Route 53 :** Amazon AWS provides Route 53, a scalable DNS web service. Users may control domain names and route traffic to AWS services or external endpoints using various routing policies. Route 53 supports DNS failover, latency-based routing, and health checks are important aspects for guaranteeing dependable and efficient access to cloud services.

Specific Components for Active-Active Architecture:

In addition to the main components stated above, the active-active architecture includes the following: Multiple availability zones: Deploying resources across several availability zones provides redundancy and fault tolerance, improving system reliability.

Load balancers may uniformly distribute traffic across EC2 instances in multiple availability zones, enhancing fault tolerance and responsiveness. Specific Components of Active-Passive Architecture

For the active-passive architecture, specific considerations include:

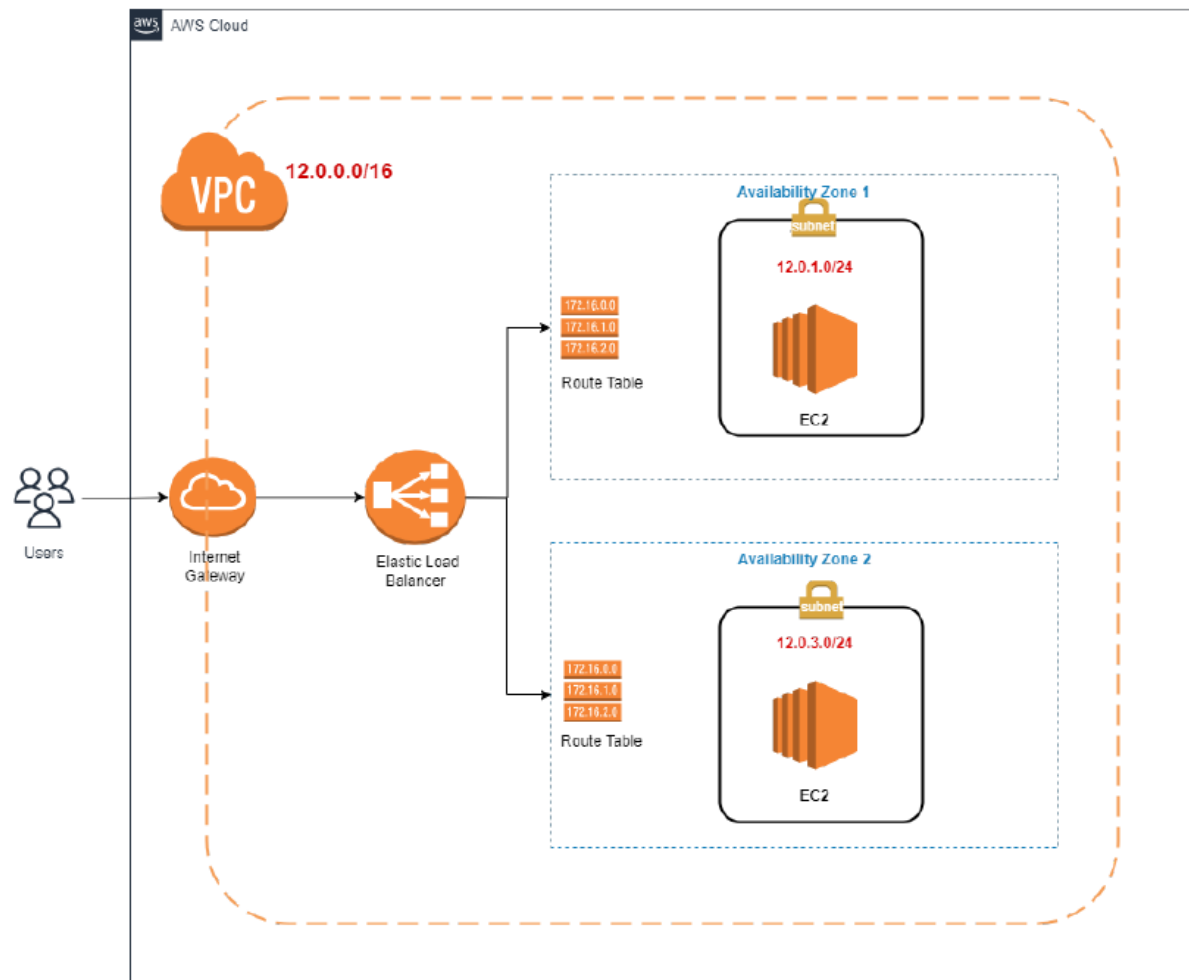


Figure 3.1: Project Overview Active-Active Diagram

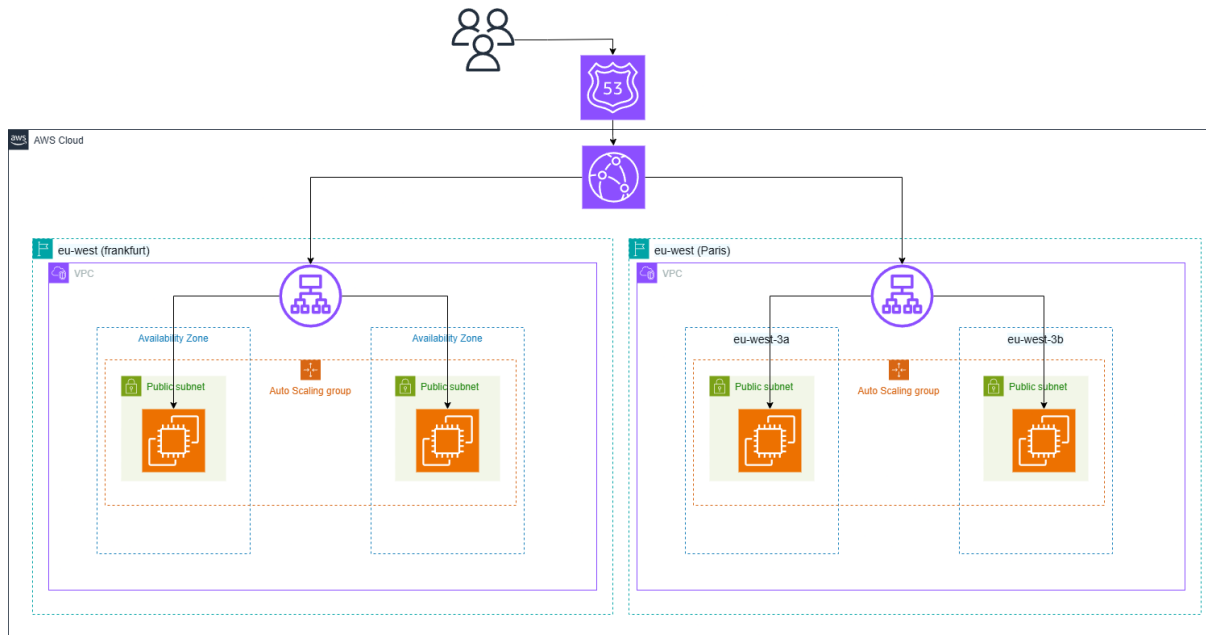


Figure 3.2: Project Overview

Create two independent VPC instances in different availability zones, each with unique resources and specifications. Route 53 is used for DNS routing, connecting active and passive instances and simplifying traffic flow.

failover and ensure ongoing availability. The project attempts to deliver high availability, fault tolerance, and scalability for cloud-based applications by integrating these components into their respective designs.

3.1.2 Relations

Relations in Active-Active Architecture:

1. Internet Gateway Connectivity: The Amazon VPC's active instances connect to the internet through the Internet Gateway, allowing bidirectional traffic flow. Several instances exist at the same time.
2. Routing Table Configuration: These tables guide traffic between subnets and availability zones, balancing load across active instances and ensuring high availability.
3. Load Balancer Distribution: The Load Balancer uniformly distributes incoming requests among active instances, improving application availability.

Relationships for Active-Passive Architecture:

1. Internet Gateway Connectivity: The active instance in the Amazon VPC connects to the internet through the Internet Gateway, while the passive instance is disconnected. Until failover happens, only the active instance handles traffic under normal circumstances.
2. Routing Table Configuration: Routing tables guide traffic inside the subnet and to the active instance, with failover routes set for the passive instance in the event of failure.
3. Route 53 DNS Routing: Route 53 directs traffic to the active instance in the Amazon VPC, guaranteeing efficient access via DNS resolution. During failover, Route 53 automatically transfers traffic to the passive instance, ensuring uninterrupted service continuity.

3.2 Project Challenges

Designing a high-availability solution for cloud applications is challenging, but offers significant benefits in cloud computing. These challenges include:

To secure our high-availability cloud computing system, we used numerous methods.

3.2.1 Security

1. Secure Instance Configuration: Our high availability system deployed instances with optimal security settings. This included employing strong passwords. Disable unneeded services and update security fixes on a regular basis. We created these instances in a secure Virtual Private Cloud (VPC) with proper security groups and ACLs.
2. Access Control Policies: We implemented access control measures to prevent unwanted access to our high availability system. We used AWS Identity and Access Management (IAM) to manage and limit user access to our high availability resources.

Implementing these procedures addressed security issues and assured our high-availability system met industry standards and best practices.

3.2.2 Application Compatibility

Application compatibility is a critical difficulty in high-availability environments. Some programs may not be optimized for high-availability situations. Consequently, certain applications may display malfunctions, or cease to operate entirely. Before implementing an application, ensure that it is designed to run smoothly in a high availability environment. To determine compatibility with a high availability system, the following procedures were followed.

1. **Application Analysis:** We evaluated the application's preparedness for high availability. This entailed analyzing if the program uses particular resources or keeps state on individual instances. If the app Due to incompatibility with high availability, adjustments were recommended to align with the system.
2. **reworking:** For applications not built for high availability, reworking was necessary. The technique involved breaking down the application into smaller, more manageable components. Load balancers and distributed databases were used to provide high availability.
3. **Testing:** After restructuring, the application was thoroughly tested to ensure it can work in a high availability environment. This involved creating a test environment that closely mirrored the production arrangement. The program was tested using simulated loads to evaluate its handling performance. Increased traffic and failover scenarios.

Following these steps ensures the application interacts easily with the high availability infrastructure. It also improves the system's capacity to handle high traffic and failover occurrences efficiently.

3.2.3 System Complexity

High availability systems can be complicated and difficult to set up and configure. Proficiency with Amazon services such as EC2 and Elastic Load Balancing is necessary. Maintenance and troubleshooting of the system may become difficult owing to its intricacy. Addressing these barriers is critical for successfully implementing high-availability solutions for cloud applications. Setting up and configuring high availability systems may be complex and challenging. They should be conversant with various cloud services, including load balancers, database replication, and monitoring tools. The system's complexity may make maintenance and debugging tough. To solve this issue, you can do the following actions:

- **Training and documentation** can help manage the complexity of high-availability systems. The system management staff should receive training on cloud services and best practices. Practices for handling these services. Document the system's deployment and administration methods. The documentation should include the system design, deployment, setup, and troubleshooting methods.

3.3 Additional Enhancements: Automation, Performance, and Monitoring

In the continuation of this project, additional AWS services were integrated to further improve the solution’s automation, performance, and operational visibility. These services include **Infrastructure as Code (IaC)** using **AWS CloudFormation**, content delivery optimization through **Amazon CloudFront**, and comprehensive observability via **Amazon CloudWatch**. The following subsections elaborate on the rationale, design, and impact of each addition.

3.3.1 Infrastructure as Code (IaC) with AWS CloudFormation

To ensure consistent, repeatable, and scalable deployments of the infrastructure, the project leveraged AWS CloudFormation as the Infrastructure as Code (IaC) tool. By defining infrastructure resources in declarative **YAML** templates, all core components—such as VPCs, subnets, EC2 instances, security groups, and route configurations—were codified and automated.

Motivation. Manual provisioning is error-prone and difficult to maintain at scale. IaC enables developers and operations teams to provision infrastructure in a predictable and controlled manner, which significantly reduces configuration drift and human error.

Implementation. A modular CloudFormation template was authored using **YAML** syntax. Parameters and mappings were utilized to generalize the template across environments. The stack defined all necessary AWS resources and relationships, including dependencies and lifecycle policies.

3.3.2 Full Annotated CloudFormation YAML Template

The following is the complete annotated CloudFormation template used to deploy the high availability infrastructure for the project. It is formatted using **YAML** and structured into logical sections to facilitate modularity, automation, and reuse across environments.

Listing 3.1: Full CloudFormation YAML Template with Annotations

```
# =====  
# Parameters Section  
# =====  
Parameters:  
  EnvironmentName:  
    Description: High-Availability Cloud Solution.  
    Type: String
```

```

Vpc1CIDR:
  Description: VPC CIDR block
  Type: String
  Default: 10.0.0.0/16

PublicSubnet1CIDR:
  Description: CIDR block for AZ1 subnet
  Type: String
  Default: 10.0.1.0/24

PublicSubnet2CIDR:
  Description: CIDR block for AZ2 subnet
  Type: String
  Default: 10.0.2.0/24

# =====
# VPC and Networking Resources
# =====
Resources:
  VPC1:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref Vpc1CIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: VPC1

  InternetGateway1:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: IGW1

  InternetGatewayAttachment1:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway1
      VpcId: !Ref VPC1

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC1

```

3.3. ADDITIONAL ENHANCEMENTS: AUTOMATION, PERFORMANCE, AND MONITORING25

```
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet1CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (1a) (AZ1)

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC1
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    CidrBlock: !Ref PublicSubnet2CIDR
    MapPublicIpOnLaunch: true
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Subnet (1b) (AZ2)

PublicRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC1
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Public Routes

DefaultPublicRoute1:
  Type: AWS::EC2::Route
  DependsOn: InternetGatewayAttachment1
  Properties:
    RouteTableId: !Ref PublicRouteTable1
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway1

PublicSubnet1RouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable1
    SubnetId: !Ref PublicSubnet1

PublicSubnet2RouteTableAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId: !Ref PublicRouteTable1
    SubnetId: !Ref PublicSubnet2

# =====
```

```

# Security Group for Web Server
# =====
WebServerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName: "webSG"
    GroupDescription: "Allow SSH, HTTP and HTTPS"
    VpcId: !Ref VPC1
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 0.0.0.0/0
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0
    Tags:
      - Key: Name
        Value: !Sub ${EnvironmentName} Allow SSH HTTP HTTPS

# =====
# Launch Template with UserData
# =====
LaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: Week7AutoScaling
    LaunchTemplateData:
      ImageId: ami-0b5673b5f6e8f7fa7
      InstanceType: t2.micro
      NetworkInterfaces:
        - DeviceIndex: 0
          AssociatePublicIpAddress: true
          Groups:
            - !Ref WebServerSecurityGroup
    UserData:
      Fn::Base64: !Sub |
        #!/bin/bash
        sudo yum update -y
        sudo amazon-linux-extras install epel -y
        sudo yum install stress -y
        sudo yum install -y httpd

```

3.3. ADDITIONAL ENHANCEMENTS: AUTOMATION, PERFORMANCE, AND MONITORING27

```
sudo systemctl start httpd
sudo systemctl enable httpd
echo "<h1>Ain-Shams University</h1>
      <h2>Project: Multi-Zone HA Cloud Solution</h2>
      <p>Supervisor: Dr. Wagdy Anis</p>
      <p>Author: Abdelrahman Ghazy</p>
      <p>Hostname: $(hostname -f)</p>" > /var/www/
      html/index.html

# =====
# Auto Scaling Group
# =====
AutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  Properties:
    MinSize: 2
    MaxSize: 4
    DesiredCapacity: 2
    LaunchTemplate:
      LaunchTemplateId: !Ref LaunchTemplate
      Version: !GetAtt LaunchTemplate.LatestVersionNumber
    HealthCheckType: ELB
    VPCZoneIdentifier:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    TargetGroupARNs:
      - !Ref ALBTargetGroups

# =====
# Auto Scaling Policy + Alarm
# =====
cpuUsage:
  Type: AWS::AutoScaling::ScalingPolicy
  Properties:
    AutoScalingGroupName: !Ref AutoScalingGroup
    PolicyType: TargetTrackingScaling
    TargetTrackingConfiguration:
      PredefinedMetricSpecification:
        PredefinedMetricType: ASGAverageCPUUtilization
      TargetValue: 50

ScalingPolicy:
  Type: 'AWS::AutoScaling::ScalingPolicy'
  Properties:
    AdjustmentType: ChangeInCapacity
    AutoScalingGroupName: !Ref AutoScalingGroup
```

```

        ScalingAdjustment: '1'

CloudWatchAlarm:
  Type: 'AWS::CloudWatch::Alarm'
  Properties:
    EvaluationPeriods: '1'
    Statistic: Average
    Threshold: '50'
    AlarmDescription: Alarm if CPU > 50%
    Period: '60'
    AlarmActions:
      - !Ref ScalingPolicy
    Namespace: AWS/EC2
    Dimensions:
      - Name: AutoScalingGroupName
        Value: !Ref AutoScalingGroup
    ComparisonOperator: GreaterThanThreshold
    MetricName: CPUUtilization

# =====
# Application Load Balancer Setup
# =====
ALBTargetGroups:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    VpcId: !Ref VPC1
    TargetType: instance
    HealthCheckPath: /index.html
    Port: 80
    Protocol: HTTP
    Tags:
      - Key: Name
        Value: Project2TG

Week7ALB:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Type: application
    Scheme: internet-facing
    SecurityGroups:
      - !Ref WebServerSecurityGroup
    Subnets:
      - !Ref PublicSubnet1
      - !Ref PublicSubnet2
    Tags:
      - Key: Name
        Value: Project2ALB

```

3.3. ADDITIONAL ENHANCEMENTS: AUTOMATION, PERFORMANCE, AND MONITORING29

```
Week7ALBListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    Protocol: HTTP
    Port: 80
    DefaultActions:
      - Type: forward
        TargetGroupArn: !Ref ALBTargetGroups
    LoadBalancerArn: !Ref ALB

# =====
# Outputs
# =====
Outputs:
  VPC1:
    Description: Created VPC
    Value: !Ref VPC1

  PublicSubnets:
    Description: List of public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2
      ] ]

  PublicSubnet1:
    Description: Public subnet AZ1
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: Public subnet AZ2
    Value: !Ref PublicSubnet2

  WebServerSecurityGroup:
    Description: SG with HTTP/HTTPS/SSH
    Value: !Ref WebServerSecurityGroup
```

Advantages.

- **Consistency:** Environments can be reliably replicated across development, staging, and production.
- **Scalability:** Resources can be provisioned and scaled without manual intervention.
- **Version Control:** Infrastructure templates were stored in a Git repository, allowing change tracking and rollbacks.

3.3.3 Performance Optimization with Amazon CloudFront

To enhance user experience and reduce latency, especially for geographically distributed users, **Amazon CloudFront** was integrated as a global Content Delivery Network (CDN) service. CloudFront accelerates the delivery of web content by caching it at AWS edge locations, thereby reducing the load on the backend EC2 instances.

Use Case. The application’s static assets—such as HTML, JavaScript, CSS, and image files—were configured to be delivered through CloudFront. The origin for the CloudFront distribution was set to the public endpoint of the EC2 instances hosting the application.

Benefits.

- **Reduced Latency:** End users access content from the nearest edge location, minimizing delays.
- **Increased Availability:** By offloading traffic from EC2 instances, the backend remains more responsive under high load.
- **Improved Scalability:** CloudFront seamlessly scales to handle surges in traffic without requiring additional provisioning.

Security Integration. CloudFront was also configured with HTTPS support and integrated with AWS Web Application Firewall (WAF) to enhance the security posture of the delivery pipeline.

3.3.4 Operational Monitoring with Amazon CloudWatch

To enable real-time observability and proactive management of system health, **Amazon CloudWatch** was adopted. CloudWatch aggregates operational data in the form of logs, metrics, and events, offering a unified view of infrastructure performance.

Monitoring Setup. Key metrics such as CPU utilization, network throughput, and disk I/O were collected from EC2 instances. Custom CloudWatch Alarms were configured to monitor thresholds and trigger alerts when anomalies were detected.

Notification and Automation. CloudWatch Alarms were integrated with Amazon Simple Notification Service (SNS) to send alerts via email when predefined thresholds were exceeded. This ensured rapid awareness of potential issues and enabled timely intervention.

Advantages.

- **Proactive Detection:** Potential performance issues are identified before they impact users.
- **Improved Uptime:** Rapid notifications support quicker response times, reducing mean time to resolution (MTTR).
- **Data-Driven Insights:** Historical trends and patterns aid in capacity planning and optimization.

3.3.5 Summary

The integration of AWS CloudFormation, Amazon CloudFront, and Amazon CloudWatch significantly enhanced the robustness, scalability, and maintainability of the high-availability cloud solution. These services collectively contributed to a more automated, performant, and observable system architecture—aligning with industry best practices for modern cloud-native deployments.

This phase helps enterprises manage the complexity of high availability systems and sustain cloud application availability and uptime.

Chapter 4

Testing

4.1 Testing

This chapter focuses mostly on the project's testing phase. This phase aims to ensure that the high availability solution and instance maintenance satisfy the requirements established in previous chapters. In This section outlines the testing techniques used to ensure the application's preparedness for high availability deployment. These tests assessed the application's performance, scalability, and resilience in managing high traffic loads and fail-over situations.

4.2 Active-Active

This section focuses on evaluating the 'Active-Active' configuration of our high availability system. This arrangement distributes incoming traffic evenly among numerous instances, allowing them to share the strain and provide uninterrupted service.

Even in the case of an instance failure. We will evaluate the system's capacity to manage increasing traffic and failover scenarios while maintaining optimal performance and availability.

4.2.1 Test Case 1: Terminate Instance in Availability Zone

The active-active architecture will be tested using the following steps:

4.2.2 Setup

To evaluate a high availability solution's capacity to maintain the appropriate number of instances during an instance failure, we established the following:

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	Webserver_1a	i-0946f654e14ce5bfc	Stopped	t2.micro	-	View alarms +	eu-central-1a	-	-	-
<input type="checkbox"/>	Webserver...	i-08f78b6705ef52e9e	Running	t2.micro	2/2 checks passed	View alarms +	eu-central-1b	ec2-3-122-231-15.eu-c...	3.122.231.15	-

Figure 4.1: Termination of instances in AZ eu-central-1a

- Created an EC2 Launch Configuration with the necessary specifications for launching new instances.
- Implemented an Application Load Balancer to balance traffic among Auto Scaling Group instances.
- Application: Deployed a basic web application across selected availability zones.

4.2.3 Testing Steps

We carried out the following testing procedures for this test case:

1. **Verify Instance Health:** Before starting a test, ensure that all instances are operating and healthy in both Availability Zones to establish a baseline condition.
2. **Terminate Instance:** To simulate an instance failure, purposefully terminate the operating instance in one of the Availability Zones (eu-north-1b).
3. **Observing Traffic Redistribution:** Monitors system activity after instance termination to ensure traffic is immediately diverted to the remaining healthy instance in the other Availability Zone (eu-north-1a). This step ensures the load balancer can appropriately route traffic.
4. **Verify Application Accessibility:** Ensures the application remains accessible and functioning after an instance failure, allowing users to engage with the system uninterrupted.

4.2.4 Results

1. We terminated the operating instance in availability zone eu-central-1a using the AWS interface.
2. Verify the health check status of the instances in both Availability.

Zones after instance termination to confirm the load

3. By requesting the load balancer's DNS name and examining the instance AZ next to the red arrow, we confirmed that the application was still accessible and traffic was being routed to the other AZ.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	Webserver_1a	i-0946f654e14ce5bfc	Stopping	t2.micro	-	View alarms +	eu-central-1a	ec2-52-29-217-196.eu-...	52.29.217.196	-
<input type="checkbox"/>	Webserver...	i-08f78b6705ef52e9e	Running	t2.micro	2/2 checks passed	View alarms +	eu-central-1b	ec2-3-122-231-15.eu-c...	3.122.231.15	-

Figure 4.2: The Health Check Status of instances

Ain-Shams University

Project Name: Multi-Zone High Availability Cloud Solution

This Project is supervised by: Dr. Wagdy Anis

Presented By: Abdelrahman Ghazy

from: ip-10-0-2-196.eu-central-1.compute.internal

Figure 4.3: Load balancer’s DNS name and viewing the instance

4.2.5 Conclude the test

The test results show that the Active-Active configuration operated well in the event of an instance failure. Our observations indicated the following. Traffic Rerouting: The system successfully diverted traffic to healthy instances in the alternative. Availability Zone once the instance is terminated. During testing, the application was completely functional and accessible, providing a smooth user experience with no service disruptions. The load balancer effectively directed traffic to the remaining instances, ensuring optimal performance and availability.

The Active-Active architecture provides robust high availability solutions that can resist instance failures and maintain smooth service operations. These findings highlight its efficacy.

4.3 Active-Passive

This chapter focuses on evaluating the 'Active-Passive' configuration of our high availability system. Unlike the 'Active-Active' setup, where all instances actively handle incoming traffic, the 'Active-Passive' configuration assigns a single group of instances.

as active, while the others stay in standby or inactive mode. We will analyze how the system changes between active and inactive modes in response to instance failures, guaranteeing ongoing service uptime and minimal downtime. We will evaluate the failover procedure and its influence on application performance and user experience.

4.3.1 Test Case 2 : Terminate The Primary Instance

We will evaluate the active-passive design using the following steps:

4.3.2 Setup

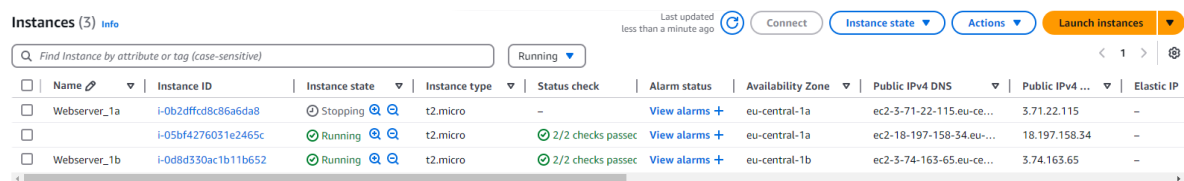
- Configure two sets of instances in the Auto Scaling Group: primary (active) and secondary (passive). The primary instances actively manage incoming traffic, whilst the secondary Instances remain on standby mode.
- Create an EC2 Launch Configuration with customized specs for primary and secondary instances. Make that the launch configuration contains options for automatically starting additional instances in case of failure.
- Configure Amazon Route 53 health checks to monitor the status of primary and secondary instances. Define failover rules based on health check results to automatically reroute traffic to backup instances in case the primary ones go offline.
- Deploy the web application across both instances in the defined availability zones. Configure the program to automatically swap between primary and secondary instances during failover occurrences.

via Amazon Route 53. Using Amazon Route 53 for DNS-based failover provides high availability and failover capabilities, comparable to a load balancer architecture. This ensures ongoing service availability during instance failures.

4.3.3 Testing Steps

We carried out the following testing procedures for this test case:

1. To trigger the failover process, simulate an instance failure or unavailability in the primary (active) instances in AZ eu-north-1a.
2. Configure Health Checks: Use Amazon Route 53 to continually check the health and availability of both active and inactive instances. These health checks ensure that instances are functioning and can handle incoming traffic.
3. Create failover rules in Route 53 to automatically transfer traffic from failing primary instances to backup instances in case of failure.
4. Monitor Failover Process: Track how the system automatically distributes traffic from failing main instances to secondary (passive) instances.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Webserver_1a	i-0b2dfcd8c86a6da8	Stopping	t2.micro	-	View alarms +	eu-central-1a	ec2-3-71-22-115.eu-ce...	3.71.22.115	-
Webserver_1b	i-05bf4276031e2465c	Running	t2.micro	2/2 checks passed	View alarms +	eu-central-1a	ec2-18-197-158-34.eu-...	18.197.158.34	-
Webserver_1c	i-0d8d330ac1b11b652	Running	t2.micro	2/2 checks passed	View alarms +	eu-central-1b	ec2-3-74-163-65.eu-ce...	3.74.163.65	-

Figure 4.4: Termination of the primary instances in AZ eu-north-1a



Name	Status	Description	Alarms	ID
ParisALB	Healthy	http://Paris-Week7A-S3AM006ahju2-120...	No alarms configured.	4d417440-2c8b-40ea-b836-8be7c31151d9
FrankfurtALB	Unhealthy	http://Frankf-Week7-yhN9qoYRLfTa-16...	1 of 1 in ALARM	70c25005-a0e5-49c6-a2ce-3b53c1e65e1c

Figure 4.5: The Health Check Status of instances

5. Verify Application Availability: Ensure the web application remains available and functional during the failover procedure to provide uninterrupted service to end customers.
6. Assess Recovery Time: Measure the time it takes for the failover procedure to finish and To ensure secondary instances are fully operating, examine the system's responsiveness and efficiency in recovering from faults.

4.3.4 Results

1. We terminated the main instances in availability zone eu-central-1a via the AWS Console.
2. After terminating the primary instance, monitor the health check status in Amazon Route 53 to confirm instance failure and initiate failover procedures.
3. We guaranteed that the online application was always accessible and fully working by Utilizing Amazon Route 53's failover method.

4.3.5 Conclude the test

In summary, our testing of the 'Active-Passive' configuration evaluated how the system manages instance failures and transitions between active and passive modes. After stopping the primary instance and observing the failover procedure, we discovered Route.

53 successfully redirects traffic to the backup instances. During the test, the web application remained completely functioning, confirming Route 53's dependable service. The findings demonstrate the system's capacity to maintain continuous availability and provide a flawless user experience, even during unexpected outages.

Ain-Shams University

Project Name: Multi-Zone High Availability Cloud Solution

This Project is supervised by: Dr. Wagdy Anis

Presented By: Abdelrahman Ghazy

from: ip-10-0-2-229.eu-west-3.compute.internal

Figure 4.6: Viewing web application by leveraging Amazon Route 53

Chapter 5

Cost Analysis

This chapter presents a cost evaluation of the proposed high availability and disaster recovery solutions implemented using Amazon Web Services (AWS). It compares two deployment strategies: an *Active-Active* multi-region configuration and a more cost-efficient *Active-Passive* alternative. The purpose of this analysis is to provide insight into the financial trade-offs involved when designing fault-tolerant cloud architectures.

5.1 Active-Active Architecture

The *Active-Active* architecture deploys full-capacity infrastructure in two separate AWS regions. Both regions remain online simultaneously and share the workload, providing real-time failover capability, low latency for global users, and the highest level of availability. However, this setup also leads to higher operational costs.

Table 5.1 shows the estimated monthly cost for running the Active-Active configuration, based on AWS pricing as of May 2025.

Table 5.1: Estimated Monthly Cost of Active-Active Setup

Resource	Quantity	Type	Estimated Cost
EC2 Instances	2 per Region (4 total)	t3.medium	$\$31 \times 4 = \124
Load Balancer	1 per Region (2 total)	ALB	$\$20 \times 2 = \40
RDS (Optional)	1 per Region (2 total)	db.t3.medium	$\$50 \times 2 = \100
CloudFront	1 Global Distribution	–	\$6 (1TB data transfer)
Route 53	Hosted Zone + Health Checks	–	\$1.50
Total Monthly Estimate			\$270

While more costly than simpler architectures such as warm standby or backup-restore, the Active-Active design is ideal for mission-critical applications where uninterrupted service is essential and any downtime would have severe consequences.

5.2 Active-Passive Architecture

To optimize cost while maintaining readiness for disaster scenarios, an *Active-Passive* configuration can be adopted. In this model, the passive region remains idle or minimally provisioned and becomes active only when a failure occurs in the primary region. This reduces monthly expenses while preserving a failover mechanism.

Table 5.2 presents the estimated cost of maintaining the passive region.

Table 5.2: Estimated Monthly Cost of Active-Passive Setup

Resource	Quantity	Type	Estimated Cost
EC2 Instances	1 (Passive Region)	t3.micro	\$8
Load Balancer	1 (Passive Region)	ALB	\$5
RDS (Optional)	Skipped or Minimized	db.t3.medium	\$20
Total Monthly Estimate			\$33

This model reduces costs by over \$100 per month compared to the Active-Active setup, while still allowing the system to recover quickly in the event of a disaster. It is particularly suitable for applications where a small delay in recovery is acceptable in exchange for cost efficiency.

5.3 Cost-Availability Trade-off

Choosing between Active-Active and Active-Passive architectures involves a trade-off between operational cost and system availability. Table 5.3 summarizes this relationship.

Table 5.3: Comparison of Active-Active vs. Active-Passive Architectures

Aspect	Active-Active	Active-Passive
Availability	Maximum (real-time failover)	Moderate (delayed failover)
Latency	Low (global distribution)	Higher (single region until failover)
Monthly Cost	\$270	\$33
Complexity	High (multi-region sync)	Moderate
Best For	Mission-critical applications	Cost-sensitive applications with acceptable RTO

5.4 Conclusion

The cost analysis highlights the importance of aligning architectural decisions with business requirements. While the Active-Active deployment offers the highest availability, its operational cost may not be justifiable for all use cases. The Active-Passive alternative provides a budget-friendly solution with acceptable performance trade-offs for many scenarios.

In production environments, organizations should also consider long-term cost optimization techniques, such as:

- Using Reserved or Spot Instances.
- Enabling auto-scaling to reduce over-provisioning.
- Leveraging S3 lifecycle policies and lower-cost storage tiers.
- Monitoring with AWS Cost Explorer and setting up budget alerts.

Ultimately, the selected architecture should reflect the criticality of the application, tolerance for downtime, and budget constraints, ensuring a resilient and cost-effective cloud infrastructure.

Chapter 6

Conclusion

This thesis explores high availability solutions for cloud computing settings. The purpose was to assess several configurations, with an emphasis on Active-Active and Active-Passive.

Setups are evaluated for their efficacy in ensuring continuous service availability and minimizing downtime. Extensive testing and analysis revealed that both arrangements have unique benefits and problems. The Active-Active system, which distributes workload and redundancy across numerous instances, proved resilient to instance failures and distributed traffic efficiently. The Active-Passive arrangement is effective for seamless failover and resource allocation, but may need a longer recovery time. The research highlights the importance of solutions like Amazon Route 53 in allowing failovers and monitoring instance health checks. These tools are essential for smooth traffic transitions and ensuring service continuity.

Availability in the face of interruption. This study's conclusions are noteworthy for both cloud practitioners and companies. Understanding high availability settings may help decision-makers create resilient cloud systems based on their individual demands and goals. This research can guide best practices for implementing and managing high availability systems, leading to improved cloud-based application dependability and performance. While this thesis has produced useful insights, it is important to recognize its limits. The study's scope was primarily concerned with testing and analyzing two particular configurations, allowing possibility for future examination of alternate settings and approaches. Furthermore, the dynamic nature of cloud technology requires continual research.

respond to changing trends and new difficulties. This research advances our understanding and optimization of high availability solutions for cloud computing. This study provides insights for practitioners to confidently manage the intricacies of cloud architecture, assuring continuous delivery of services and applications in a digital world.

Appendix

Appendix A

Lists

- AMI - Amazon Machine Image
- ASG - Auto Scaling Group
- AWS - Amazon Web Services
- AZ - Availability Zone
- CPU - Central Processing Unit
- LB - Load Balancer
- EC2 - Elastic Compute Cloud
- EIP - Elastic IP address
- ELB - Elastic Load Balancer
- IaaS - Infrastructure as a Service
- IaC - Infrastructure as Code
- IAM - Identity and Access Management
- k6 - A performance testing tool
- NAT - Network gateway
- PaaS - Platform as a Service
- SaaS - Software as a Service
- VM - Virtual Machine
- VPC - Virtual Private Cloud

List of Figures

2.1	Comparison between Traditional and Visualized Servers	4
2.2	Cloud Service Models	8
2.3	High Availability Clusters	12
3.1	Project Overview Active-Active Diagram	19
3.2	Project Overview	20
4.1	Termination of instances in AZ eu-central-1a	34
4.2	The Health Check Status of instances	35
4.3	Load balancer's DNS name and viewing the instance	35
4.4	Termination of the primary instances in AZ eu-north-1a	37
4.5	The Health Check Status of instances	37
4.6	Viewing web application by leveraging Amazon Route 53	38