

FAILOVER ROUTING POLICY USING AWS

*Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Information Technology**

By

**E.SAI KOTESWARARAO (21UTCC0011) (VTU NO 19657)
G.POORNA CHANDU (21UTCC0002) (VTU NO 19504)**

*Under the guidance of
Dr.C.SURESH KUMAR, ME., Ph.D.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

FAILOVER ROUTING POLICY USING AWS

*Minor project-II report submitted
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology
in
Information Technology**

By

**E.SAI KOTESWARARAO (21UTCC0011) (VTU NO 19657)
G.POORNA CHANDU (21UTCC0002) (VTU NO 19504)**

*Under the guidance of
Dr.C.SURESH KUMAR, ME., Ph.D.,
ASSISTANT PROFESSOR*



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF
SCIENCE & TECHNOLOGY**

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

**Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA**

May, 2024

CERTIFICATE

It is certified that the work contained in the project report titled “FAILOVER ROUTING POLICY USING AWS” by E.SAI KOTESWARARAO (21UTCC0011), G.POORNA CHANDU (21UTCC0002) has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Information Technology

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

May, 2024

Signature of HOD

Information Technology

School of Computing

Vel Tech Rangarajan Dr. Sagunthala R&D

Institute of Science & Technology

May, 2024

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

E.SAI KOTESWARARAO

Date: / /

G.POORNA CHANDU

Date: / /

APPROVAL SHEET

This project report entitled “FAILOVER ROUTING POLICY USING AWS” by E.SAI KOTESWARA RAO (21UTCC0011), G.POORNA CHANDU (21UTCC0002), is approved for the degree of B.Tech in Information Technology.

Examiners

Supervisor

Dr. C. SURESH KUMAR, ME., Ph.D.,

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Information Technology, Dr. J. VISUMATHI, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Dr.C.SURESH KUMAR, ME., Ph.D.**, for his cordial support, valuable information and guidance, he/she helped us in completing this project through various stages.

A special thanks to our **Project Coordinator Dr. N.KATHIRVEL, M.E., Ph.D** for his valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

E.SAI KOTESWARARAO (21UTCC0011)

G.POORNA CHANDU (21UTCC0002)

ABSTRACT

Failover routing policy in Amazon Web Services (AWS) refers to a strategy used to ensure high availability and fault tolerance of applications and services deployed in the AWS cloud environment. This policy is particularly crucial for applications that require continuous availability and minimal downtime. The primary goal of a failover routing policy is to automatically redirect traffic from a failed or unhealthy resource to a healthy resource, thereby minimizing service disruption and ensuring that users can access the application without interruptions. This is achieved through intelligent routing decisions based on the health status of resources. Maintaining service availability during unexpected disruptions is crucial. AWS Route 53's failover routing policy comes to the rescue in such situations. It acts as a traffic director, ensuring your users can still access your resources even if your primary infrastructure encounters an outage. Imagine your website hosted in a particular AWS region. With a failover routing policy in place, Route 53 directs all incoming traffic to this primary resource as long as it's functioning properly. This is typically monitored through health checks configured by you. If a health check detects an issue with the primary resource, Route 53 seamlessly takes over. It automatically reroutes traffic to a pre-defined backup resource located in a different region, minimizing downtime and maintaining service continuity for your users. Implementing a failover routing policy is a straightforward process within Route 53. You'll define both your primary and backup resources, which can be anything from an Amazon S3 bucket to a complex web application. Health checks are then created to actively monitor the health of the primary resource. These checks can ping the resource periodically or follow custom health checks tailored to your specific needs. If a health check fails a predetermined number of times, Route 53 recognizes the primary as unhealthy and triggers the failover. Traffic seamlessly switches to the backup resource, and users likely won't even notice the brief interruption. Once the primary resource recovers and passes its health checks again, Route 53 can be configured to automatically switch traffic back, completing the failover-failback cycle. This automated approach ensures your service remains available and resilient even during unforeseen circumstances.

Keywords: Amazon Web Services, Route 53, Routing, Failover, Health Checks, Hosted Zone, Domain Name System

LIST OF FIGURES

4.1	General Architecture for Failover Routing Policy	11
4.2	Dataflow diagram for Failover Routing Policy	13
4.3	Usecase diagram for Failover Routing Policy	14
4.4	Class diagram for Failover Routing Policy	16
4.5	Sequence diagram for Failover Routing Policy	17
4.6	Activity diagram for Failover Routing Policy	19
5.1	Creating EC2 Instances	28
5.2	Successfully Created EC2 Instances	29
5.3	Create Route53 in AWS	29
5.4	Create Hosted Zone in AWS	30
5.5	Create Health Checks	31
5.6	Successfully Created Records	32
5.7	Health Checks Passed	34
5.8	Checking Health Checks	35
5.9	Test Image for Failover Rouitng Policy	36
6.1	Output 1 for Failover Routing Policy	41
6.2	Output 2 for Failover Routing Policy	41

LIST OF ACRONYMS AND ABBREVIATIONS

S.NO	ACRONYMS	ABBREVIATION
1	AWS	Amazon Web Service
2	CI	Continuous Integration
3	CD	Continuous Delivery
4	CDN	Content Delivery Network
7	DNS	Domain Name System
5	EC2	Elastic Compute Cloud
6	HC	Health Check
7	S3	Simple Storage Service

TABLE OF CONTENTS

	Page.No
ABSTRACT	v
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ACRONYMS AND ABBREVIATIONS	vii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Aim of the project	2
1.3 Project Domain	2
1.4 Scope of the Project	3
2 LITERATURE REVIEW	4
3 PROJECT DESCRIPTION	7
3.1 Existing System	7
3.2 Proposed System	7
3.3 Feasibility Study	8
3.3.1 Economic Feasibility	8
3.3.2 Technical Feasibility	8
3.3.3 Social Feasibility	8
3.4 System Specification	9
3.4.1 Hardware Specification	9
3.4.2 Software Specification	9
3.4.3 Standards and Policies	10
4 METHODOLOGY	11
4.1 General Architecture for Failover Routing Policy	11
4.2 Design Phase	13
4.2.1 Data Flow Diagram for Failover Routing Policy	13

4.2.2	Use Case Diagram for Failover Routing Policy	14
4.2.3	Class Diagram for Failover Routing Policy	16
4.2.4	Sequence Diagram for Failover Routing Policy	17
4.2.5	Activity Diagram for Failover Routing Policy	19
4.3	Algorithm & Pseudo Code	20
4.3.1	Route 53 Algorithm	20
4.3.2	Route 53 Pseudo Code	21
4.4	Module Description	22
4.4.1	Create a Failover Routing Policy	22
4.4.2	Test the failover Routing Policy	23
4.4.3	Create Routing Policy Management Module	24
4.5	Steps to execute/run/implement the project	25
4.5.1	Resource Setup	25
4.5.2	Policy Creation	26
4.5.3	Test and Monitor	26
5	IMPLEMENTATION AND TESTING	28
5.1	Input and Output	28
5.1.1	Input Design for Failover Routing	28
5.1.2	Output Design for Created Records	32
5.2	Testing	33
5.3	Types of Testing	33
5.3.1	Unit testing	33
5.3.2	Integration testing	34
5.3.3	System testing	35
5.3.4	Test Result	36
6	RESULTS AND DISCUSSIONS	37
6.1	Efficiency of the Proposed System	37
6.2	Comparison of Existing and Proposed System	37
6.3	Sample Code	39
7	CONCLUSION AND FUTURE ENHANCEMENTS	42
7.1	Conclusion	42
7.2	Future Enhancements	43

8	PLAGIARISM REPORT	44
9	SOURCE CODE & POSTER PRESENTATION	45
9.1	Source Code	45
9.2	Poster Presentation	47
	References	47

Chapter 1

INTRODUCTION

1.1 Introduction

In today's digital landscape, maintaining service availability is paramount. Users expect uninterrupted access to applications and websites, and any downtime can lead to lost revenue and frustrated customers. Fortunately, Amazon Web Services (AWS) offers robust tools to build resilient infrastructure. One such tool is Route 53's failover routing policy, a critical component for crafting a comprehensive disaster recovery strategy.

Route 53 acts as your domain name system (DNS) service in the cloud, directing user traffic to your resources. A failover routing policy within Route 53 adds an extra layer of protection. It establishes a primary resource your website, application, or any service you want users to access and a designated backup resource. The policy intelligently routes traffic to the healthy primary resource under normal conditions. However, if the primary resource encounters an outage or malfunctions, the failover policy automatically steps in. This automatic intervention is the heart of the failover routing policy's power. By continuously monitoring the health of your primary resource using customizable health checks, Route 53 can detect issues promptly. These health checks can range from simple ping checks to more elaborate tests that verify specific functionalities of your resource. If a pre-defined number of health check failures occur, Route 53 recognizes the primary resource as unhealthy and triggers a seamless failover. This ensures service continuity even during regional outages, minimizing downtime for your users.

The failover routing policy operates within an active-passive model. This means that under normal circumstances, only the primary resource receives user traffic. The backup resource remains on standby, ready to take over if needed. This approach optimizes resource allocation while guaranteeing service availability in the event of a disruption. By implementing a failover routing policy, you can significantly enhance the resiliency of your infrastructure and provide a superior user experience by minimizing downtime and ensuring consistent service access.

1.2 Aim of the project

The primary aim of implementing a failover routing policy in AWS Route 53 is to guarantee service continuity for your users even when disruptions or outages occur. It achieves this by intelligently managing traffic flow between a designated primary resource and a backup resource. During normal operation, the policy directs traffic to your primary resource, which can be anything from a website to a complex application. However, if the primary resource malfunctions or experiences an outage detected by health checks, the failover policy seamlessly switches traffic to the predefined backup resource, potentially located in a different region. This automatic failover ensures minimal downtime for your users and fosters a more resilient infrastructure that can withstand unforeseen circumstances. By employing a failover routing policy, you prioritize service availability and provide a reliable user experience.

1.3 Project Domain

The project domain for implementing a failover routing policy in AWS revolves around ensuring high availability and fault tolerance for applications and services deployed in the AWS cloud environment. This domain encompasses designing and implementing robust architectures that can automatically handle failures and redirect traffic from unhealthy resources to healthy ones. It involves leveraging AWS services such as Amazon Route 53, Elastic Load Balancing (ELB), AWS Global Accelerator, Auto Scaling, and Multi-AZ deployments to achieve seamless failover and minimize service disruptions.

Key areas within this project domain include architecture design, failover mechanisms, health monitoring, traffic routing policies, testing and validation, automation and orchestration, and documentation of best practices. Architectural design focuses on creating fault-tolerant setups that can withstand infrastructure failures and maintain service continuity. Failover mechanisms involve setting up health checks, defining routing policies, and automating failover processes to redirect traffic to healthy resources. Health monitoring ensures continuous assessment of resource health, while traffic routing policies determine how traffic should be routed during failover events.

1.4 Scope of the Project

The project scope for a failover routing policy in AWS Route 53 centers on building an automated traffic redirection system to guarantee service continuity during outages. This involves defining two key resources: the primary resource that users access normally (websites, applications, etc.) and the backup resource that takes over if the primary fails (located potentially in a different region for broader disaster recovery). The project encompasses configuring health checks to monitor the primary resource's health (simple ping checks or customized tests). If these checks fail a predetermined number of times, the policy triggers a failover, seamlessly rerouting traffic to the backup resource. The scope can be further expanded to include automatic failback, switching traffic back to the primary resource upon recovery, thus completing the failover-failback cycle. This ensures service availability even in unforeseen circumstances. Remember, the failover policy operates within an active-passive model, with only the primary resource receiving traffic until it becomes unavailable.

Chapter 2

LITERATURE REVIEW

[1]Ambika Gupta et al., (2020) proposed a literature review on cloud storage. In this digital era, the emergence of many smart applications makes the human lives in a way more smart but at the same time it also increases the amount of data to an unprecedented rate. This makes the traditional data storage framework, accessibility and backup to remain more under risk. Recently, to overcome this an on-demand storage service will be provided by the cloud. Despite the hype, the cloud also comes up with the data security concerns. The primary objective of this research work is to develop a framework that permits confirmed clients to get access to the sensitive information within the organization. In contrast to the traditional system, the proposed system helps in data recovery if there arise any circumstances, the storage can be expanded by checking the accessibility of users logged into the storage system to access the data.

[2] Anirban Mukhopadhyay et al.,(2006) conducted a review on Static sites, Static websites have become increasingly popular for their simplicity, cost-effectiveness, and ease of deployment. With the rise of cloud computing platforms like Amazon Web Services (AWS), hosting static sites has become even more accessible. This literature review aims to explore the performance aspects of static sites hosted on AWS, focusing on key factors such as scalability, reliability, speed, and cost efficiency.

[3] A.S. Rodge et al.,(2015) proposed a literature review on Multicast routing with load balancing using Amazon Web Service, investigates a method for optimizing data delivery within a network. Multicast routing allows a single source to send data to multiple destinations simultaneously, while load balancing distributes the workload across available resources to prevent any one resource from becoming overloaded. The authors propose leveraging the functionalities of Amazon Web Services (AWS), a cloud computing platform, to establish this combined approach.

[4]Brian Beach et al.,(2017) conducted a review on Achieving High Availability with Amazon Route 53 Health Checks and DNS Failover by Brian Beach, presented at the Second International Conference on Computing Methodologies and Communication (ICCMC 2018) in July 2017, likely explores the utilization of Amazon Route 53's health checks and DNS failover features to achieve high availability in distributed systems hosted on AWS. The study may discuss how these features enable proactive monitoring of service health and automatic failover to healthy endpoints, thereby enhancing the reliability and resilience of applications.

[5] David Clinton et al.,(2021) proposed a literature review on Domain Name System and Network Routing, Amazon Route 53 and Amazon CloudFront, focusing on user education, detection techniques, and preventive measures. It discusses the various types of keyloggers, including hardware-based, software-based, and remote keyloggers, and examines strategies for identifying and mitigating keylogger threats.

[6]Darus et al.,(2020) proposed a comprehensive literature review on soil testing provides a nuanced understanding of the diverse methodologies, technologies, and applications employed in assessing soil health and quality. Research by darus. elucidates the significance of soil testing as a fundamental tool for precision agriculture, emphasizing the role of advanced sensing technologies and data analytics in modern soil analysis. Their work underscores the integration of remote sensing, GIS, and machine learning techniques to enhance the spatial and temporal resolution of soil assessments, contributing to more informed agricultural practices.

[7]Hrishikesh Dewan et al.,(2011) Conducted a Survey on Cloud Storage Facilities , presented at the IEEE World Congress on Services in 2011, likely provides a comprehensive overview of cloud storage solutions and their functionalities. The authors may have surveyed various aspects such as storage architectures, data models, access mechanisms, security features, scalability, and performance characteristics of different cloud storage providers. The paper might discuss key challenges and trends in cloud storage technology, including topics like data consistency, reliability, cost-effectiveness, and compliance with regulatory requirements.

[8]H. Huang, et al.,(2011) performed a literature review that Magazine, Proactive Failure Recovery for NFV, including climate change, air quality monitoring,

water resource management, and ecosystem conservation. It examines recent research findings, methodologies, and technologies for understanding and mitigating environmental issues.

[9]K.Veena et al.,(2016) conducted comprehensive review on "Temporal and Spatial Trend Analysis of Cloud Spot Instance Pricing in Amazon EC2" by K. Veena, C. Anand, and G. C. Prakash, published in 2016, likely presents a detailed examination of spot instance pricing trends within Amazon EC2. The authors likely analyze historical data to identify temporal and spatial patterns in spot instance pricing, aiming to provide insights into the factors influencing pricing fluctuations over time and across different regions.

[10] Patterson et al.,(2020) conducted a literature review on "High Availability System Design", presented at the 11th International Conference on Signal Processing Systems, likely discusses various strategies and principles for designing systems that prioritize high availability. This could include concepts such as redundancy, fault tolerance, load balancing, disaster recovery, and failover mechanisms. The author may have explored architectures, technologies, and best practices aimed at ensuring continuous operation and minimizing downtime in critical systems.

[11] Urvesh Domadiya et al.,(2021) conducted a literature review on Disaster Recovery with Route 53's Failover Routing Policy likely explores the use of AWS Route 53's failover routing policy as a component of disaster recovery strategies. The study may delve into how organizations can leverage Route 53's failover capabilities to ensure high availability and resilience of their applications and services hosted on AWS. It might discuss implementation details, best practices, case studies, and the benefits of using Route 53's failover routing policy in disaster recovery scenarios. Readers can expect to gain insights into using cloud-based routing policies for enhancing disaster recovery strategies and maintaining business continuity in AWS environments.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

Primary Path: In most network setups, traffic follows a primary path defined by the routing protocol. This path is usually the most efficient route based on metrics like cost, hop count, or bandwidth. The routing table determines this path.

Single Point of Failure: If there's no redundancy built into the network, a single point of failure (like a router outage) can cause traffic to simply stall or drop completely. There's no mechanism to automatically reroute traffic in this scenario.

- Increased Complexity
- Potential for Routing Loops
- Delayed Failover

3.2 Proposed System

The proposed system builds upon traditional failover routing policies by addressing their limitations. Additionally, exploring cost-effective solutions like open-source tools or cloud services makes this system accessible for various budgets. By implementing continuous monitoring and failover simulations, this proposal ensures the system's effectiveness and preparedness for real-world scenarios. This comprehensive approach offers significant advantages over basic failover routing, resulting in a more resilient, efficient, and cost-conscious network infrastructure.

- Reduced Complexity and Errors
- Enhanced Reliability
- Improved Cost-Effectiveness

3.3 Feasibility Study

3.3.1 Economic Feasibility

Assessing the economic feasibility of implementing a failover routing policy in AWS involves a detailed analysis of costs and benefits. On the cost side, organizations need to consider expenses related to redundancy, high availability services, auto-scaling resources, monitoring tools, data replication, and operational management. These costs can vary based on the scale of infrastructure, usage patterns, and the complexity of failover mechanisms.

However, the investment in a robust failover routing policy can lead to significant benefits. Enhanced high availability can minimize downtime, which is crucial for mission-critical applications and services. Reduced downtime translates to better customer experience, higher productivity, and avoidance of potential revenue loss due to service interruptions. Additionally, improved resilience can strengthen the organization's overall disaster recovery capabilities and mitigate risks associated with infrastructure failures or unexpected events.

3.3.2 Technical Feasibility

Assessing the technical feasibility of implementing a failover routing policy in AWS involves evaluating various aspects to ensure seamless and reliable failover mechanisms. Firstly, organizations need to analyze the compatibility and availability of essential AWS services such as Amazon Route 53 for DNS failover, Elastic Load Balancing (ELB) for load distribution, Auto Scaling for resource optimization, and Multi-AZ deployments for database redundancy. This assessment ensures that the selected AWS services align with the organization's infrastructure requirements and can effectively handle failover events without compromising performance or reliability.

3.3.3 Social Feasibility

Assessing the social feasibility of implementing a failover routing policy in AWS involves considering the impact on various stakeholders and the organization's social environment. This aspect delves into how the failover policy affects people, processes, and relationships within and outside the organization.

One crucial consideration is the impact on end-users and customers. A failover routing policy aims to minimize service disruptions and downtime, which can significantly improve customer satisfaction and trust in the organization's services. By ensuring reliable and continuous access to applications and resources, the failover policy can enhance user experience and loyalty, leading to positive social outcomes.

3.4 System Specification

3.4.1 Hardware Specification

- CPU: Intel Core i5 or i7 processor (specific model and generation may vary)
- GPU: Integrated Intel UHD Graphics or discrete graphics card (model-specific)
- RAM: 8GB or 16GB DDR4 RAM (can vary based on configuration)
- Storage: 256GB, 512GB, or 1TB Solid State Drive (SSD) storage options
- Operating System: Windows 11 Pro or Windows 11 Home edition

3.4.2 Software Specification

- Operating System: Windows 11 Pro or Windows 11 Home
- Programming Languages: No specific language comes pre-installed, but commonly used languages like Python, Java, C++, etc., can be installed as needed.
- Development Environment: Software development environments like Visual Studio, Eclipse, IntelliJ IDEA, or JetBrains PyCharm can be installed based on user preferences.
- Additional Libraries: Libraries and frameworks such as TensorFlow, Pandas, NumPy for Python, .NET Framework for C, and others can be installed for specific programming needs.
- Version Control: Git or other version control systems can be installed for code versioning and collaboration.
- Cloud Computing: HP EliteBook can access various cloud platforms like AWS, Microsoft Azure, or Google Cloud Platform through web browsers or cloud-specific software for development or deployment purposes.

- **Data Handling Tools:** Microsoft Excel, SQL Server Management Studio, MySQL Workbench, or specialized tools like Tableau for data visualization can be installed to handle data tasks.

3.4.3 Standards and Policies

Amazon Web Services

Amazon Web Services (AWS) stands as the pioneer and leading force in the realm of cloud computing, providing a vast and sophisticated suite of on-demand computing resources to individuals, businesses, and governments. Launched in 2006, AWS has transformed the IT industry by offering a scalable and flexible infrastructure that allows users to avoid the complexities and costs of owning and maintaining physical servers. AWS encompasses a comprehensive range of services, spanning computing power with services like Amazon EC2, storage with Amazon S3, and databases with offerings like Amazon RDS. Its infrastructure spans the globe, with data centers strategically located in various regions, allowing for low-latency and high-availability solutions.

Standard Used: ISO/IEC 27001

Route53

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) service offered by Amazon Web Services (AWS). Think of DNS as a giant phonebook for the internet. When you enter a web address into your browser, your computer contacts a DNS server to look up the IP address associated with that domain name. Route 53 acts as that DNS server, translating domain names into IP addresses and directing traffic to your website, web application, or other resources.

Standard Used: ISO/IEC 27001

Chapter 4

METHODOLOGY

4.1 General Architecture for Failover Routing Policy

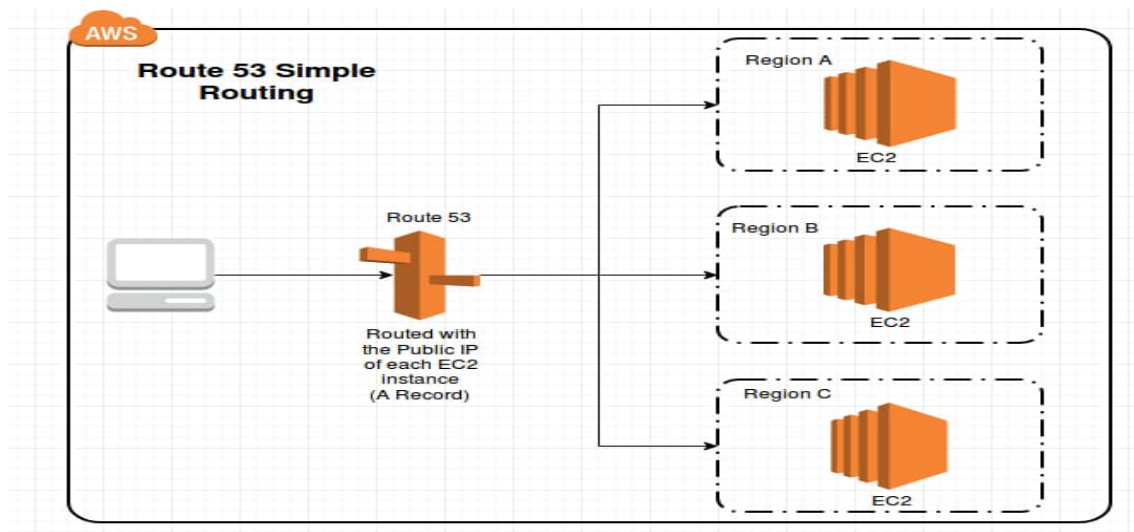


Figure 4.1: General Architecture for Failover Routing Policy

The above Figure 4.1 describes to implement a failover routing policy in AWS using Amazon Route 53 for three EC2 instances, you establish a setup where user traffic is directed based on failover conditions. Route 53 acts as the DNS service, managing domain names and routing internet traffic. Within this configuration, Designate one EC2 instance as the primary endpoint (e.g., Instance A) to handle user requests under normal operating conditions. Additionally, you set up two other EC2 instances, denoted as Instance B and Instance C, as secondary endpoints to act as backup resources. Route 53 monitors the health of the primary instance using health checks and, if it detects issues, automatically routes traffic to one of the healthy secondary instances. This failover mechanism ensures high availability, minimizes downtime, and provides a seamless user experience by dynamically rerouting traffic based on real-time health assessments.

Imagine users accessing the application. Their requests travel through the internet and reach Route 53, your AWS DNS service. Route 53 acts as a central hub,

directing user traffic efficiently. It maintains a record of your application's domain name and the IP addresses of your EC2 instances spread across different Availability Zones.

Here's where Route 53's intelligence comes into play. It continuously performs health checks on your EC2 instances. These checks verify if the instances are up and running and can respond to requests effectively. Based on the results of these health checks and your configured failover routing policy (which prioritizes healthy instances), Route 53 intelligently directs user traffic. It might use a round-robin approach to distribute user requests evenly across all healthy EC2 instances if everything functions normally.

The true power of this architecture lies in its ability to handle outages. If a health check fails, indicating an unhealthy EC2 instance or an entire Availability Zone experiencing issues, Route 53 reacts swiftly. It automatically removes the unhealthy element from the pool of available resources, ensuring user traffic isn't directed to a non-functional instance. This eliminates potential delays or errors for users.

More importantly, Route 53 seamlessly reroutes the user's request to the nearest healthy EC2 instance in another Availability Zone. This rerouting happens automatically, minimizing downtime for your users. They might experience a slight delay due to the rerouting process, but overall, their experience remains uninterrupted. This architecture with Route 53 health checks and failover routing policy offers a highly resilient and reliable solution. It guarantees that your application remains accessible to users even during isolated outages, keeping your business operational and users satisfied.

This failover routing architecture with Route 53 health checks offers several advantages beyond just handling outages. It improves the overall availability and scalability of your application. By distributing traffic across healthy instances in different Availability Zones, you avoid overloading any single resource and ensure a smoother user experience. Additionally, this architecture allows you to easily scale your application by adding more EC2 instances across Availability Zones. Route 53 automatically detects and integrates the new instances, further enhancing redundancy and fault tolerance. This robust system ensures your application can handle increased traffic demands while maintaining high availability and minimizing downtime.

4.2 Design Phase

4.2.1 Data Flow Diagram for Failover Routing Policy

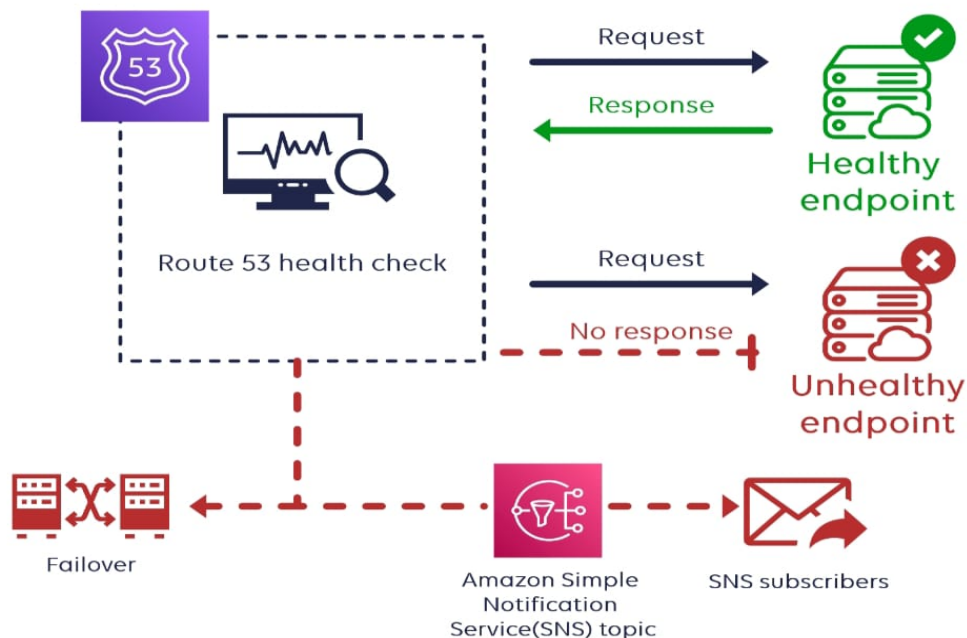


Figure 4.2: Dataflow diagram for Failover Routing Policy

The above Figure 4.2 describes Dataflow diagram for failover routing policy. Imagine data flowing from a user's request on the internet. It reaches Route 53, which acts like a traffic director. Route 53 constantly sends health checks to your EC2 instances across Availability Zones. If all instances are healthy, Route 53 directs traffic according to your policy (e.g., round robin). If a health check fails, Route 53 automatically removes that EC2 instance from the pool and directs traffic only to the remaining healthy ones. This ensures continuous flow of data even during outages, maximizing uptime for your application.

The data flow diagram for a failover routing policy with Route 53 health checks in AWS illustrates a user's request journey to your application. Imagine the request travelling from the user over the internet to Route 53, which acts like a smart traffic director. Route 53 continuously performs health checks on your EC2 instances across Availability Zones. Based on these checks and your failover policy, it directs traffic. If a health check fails, Route 53 swiftly removes the unhealthy option and reroutes the user's request to the nearest healthy instance in another zone, ensuring

minimal disruption and maintaining application accessibility.

Imagine user requests flowing from the internet like a river. Route 53 acts as a dam, constantly checking the "health" (water flow) of your EC2 instances across Availability Zones. Based on these checks and your failover policy (prioritizing good flow), Route 53 directs traffic (like opening specific gates). If a health check fails (indicating a blockage), Route 53 automatically reroutes the user request (bypassing the blockage) to the nearest healthy instance in another zone, ensuring a continuous flow of data and minimal disruption for users.

4.2.2 Use Case Diagram for Failover Routing Policy

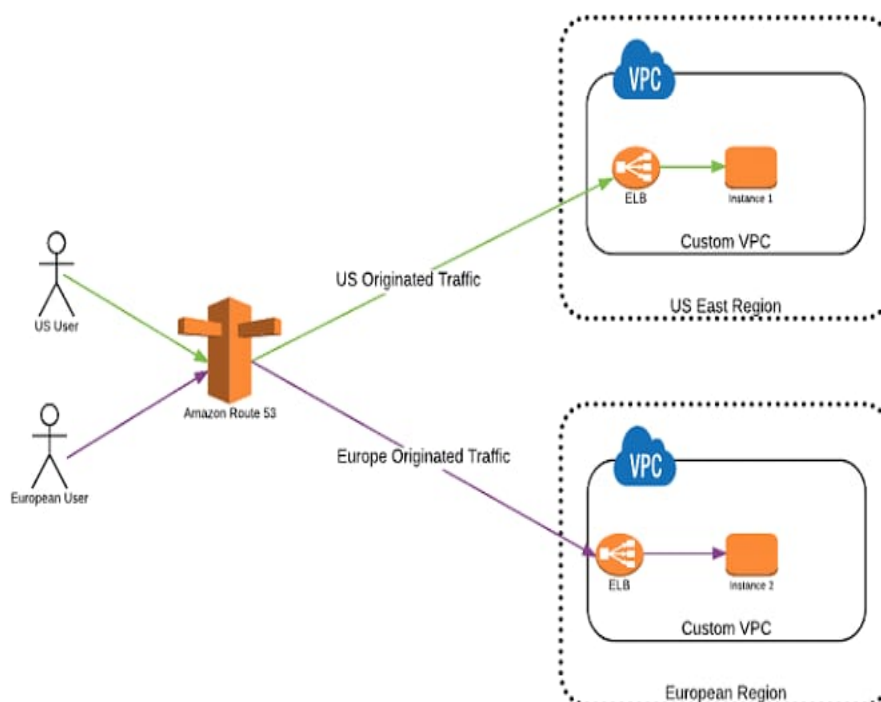


Figure 4.3: Usecase diagram for Failover Routing Policy

The above Figure 4.3 describes Usecase diagram for failover routing policy. The user's journey starts with a request traveling over the internet. This request reaches Route 53, your AWS managed DNS service. Route 53 acts like a smart switchboard. It maintains a record of your application's domain name and the IP addresses of your EC2 instances spread across different Availability Zones within a Virtual Private Cloud (VPC).

However, Route 53 doesn't simply forward the request. It constantly performs health checks on your EC2 instances (though not directly shown in the diagram).

These checks ensure the instances are up and running and can respond to requests effectively. Based on these checks and your configured failover policy (which prioritizes healthy instances), Route 53 makes an intelligent decision about where to direct the user's traffic.

Using the health check results and the failover policy, Route 53 selects the most suitable Availability Zone. This selection prioritizes Availability Zones with healthy EC2 instances. Route 53 then directs the user's request towards the Elastic Load Balancer (ELB) associated with that particular Availability Zone within the VPC. The ELB acts as a traffic director within your VPC. It efficiently distributes the incoming traffic across the healthy EC2 instances residing within the chosen Availability Zone. This ensures optimal load balancing and prevents any single instance from being overloaded.

The real strength of this system lies in its ability to handle outages. If a health check fails, indicating an unhealthy EC2 instance or an issue within an Availability Zone, Route 53 reacts swiftly. It automatically excludes the unhealthy zone from the pool of available options, ensuring user traffic isn't directed to non-functional resources that could cause delays or errors. More importantly, Route 53 seamlessly reroutes the user's request to the ELB in another healthy Availability Zone. This rerouting happens automatically, minimizing downtime for users.

They might experience a slight delay, but overall, their experience remains uninterrupted. This use case diagram emphasizes how this system, with its interplay between Route 53's failover policy, health checks, the ELB, and your EC2 instances, guarantees continuous application access for users even during outages by intelligently directing traffic based on real-time health information and efficient load balancing.

The use case diagram depicts a user's journey to your application. Their request reaches Route 53, the DNS service, which acts like a smart switchboard. Based on health checks and your failover policy, Route 53 directs traffic to the appropriate Availability Zone within your VPC. There, an Elastic Load Balancer distributes the request across healthy EC2 instances, ensuring optimal performance and seamless failover to a healthy zone if needed, minimizing downtime for users.

4.2.3 Class Diagram for Failover Routing Policy

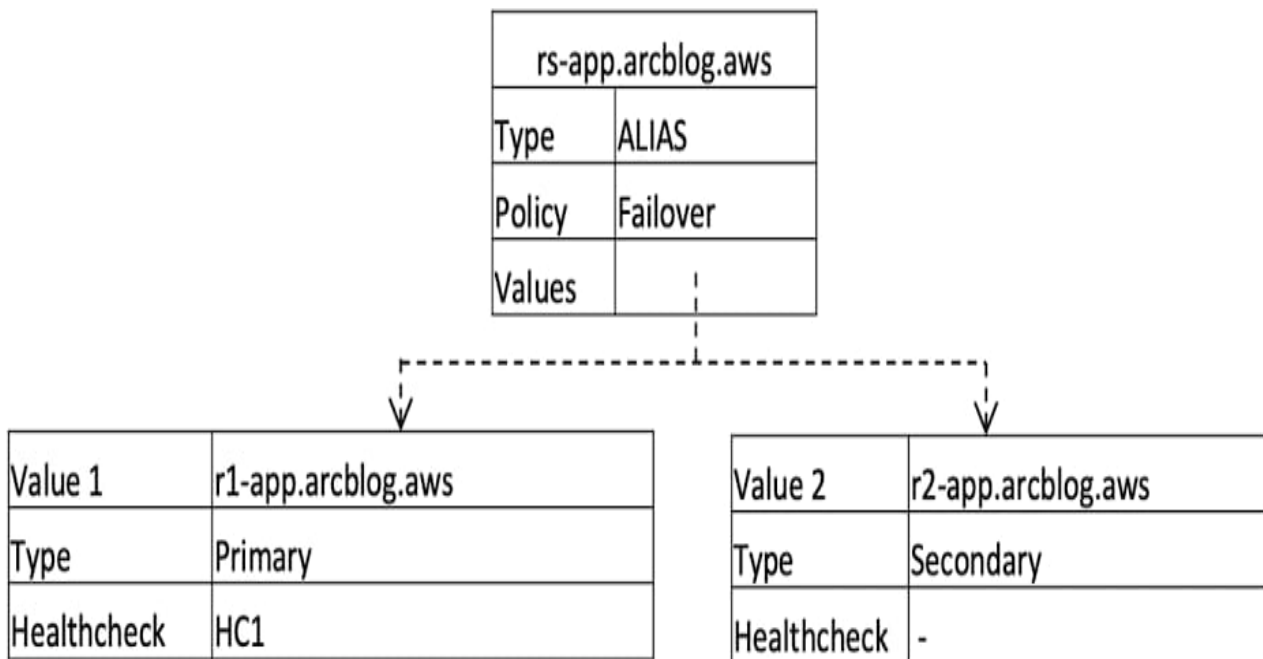


Figure 4.4: Class diagram for Failover Routing Policy

The above Figure 4.4 describes, Class diagram for a failover routing policy in AWS it is like a blueprint for managing traffic flow and application uptime. Here's a simplified, Imagine different actors like classes working together. Route53 acts as the main control center. It keeps track of HealthChecks for your EC2Instances across Availability Zones. These health checks ensure the instances are healthy and can handle requests. Based on these checks and a defined FailoverPolicy (which prioritizes healthy resources), Route53 intelligently directs user traffic. The failover policy might consider factors like Availability Zone or response times. This diagram highlights the relationships between these classes: Route53 uses health checks to monitor EC2 instances, relies on the failover policy for decisions, and ultimately directs traffic to healthy instances within your AWS setup.

Imagine building blocks (classes) working together for resilient application access. Route 53 acts as the conductor, managing a list of HealthChecks for your EC2Instances scattered across Availability Zones. These health checks are like constant check-ups, ensuring instances are healthy and responsive. Based on these checks and a pre-defined FailoverPolicy (prioritizing healthy resources), Route 53 intelligently directs user traffic. The failover policy might consider factors like location or performance. This diagram depicts the connections: Route 53 uses health checks

to monitor EC2 instances, leverages the failover policy for routing decisions, and ultimately directs user traffic towards healthy instances within your AWS infrastructure. This collaborative approach ensures continuous application access even during potential outages.

4.2.4 Sequence Diagram for Failover Routing Policy

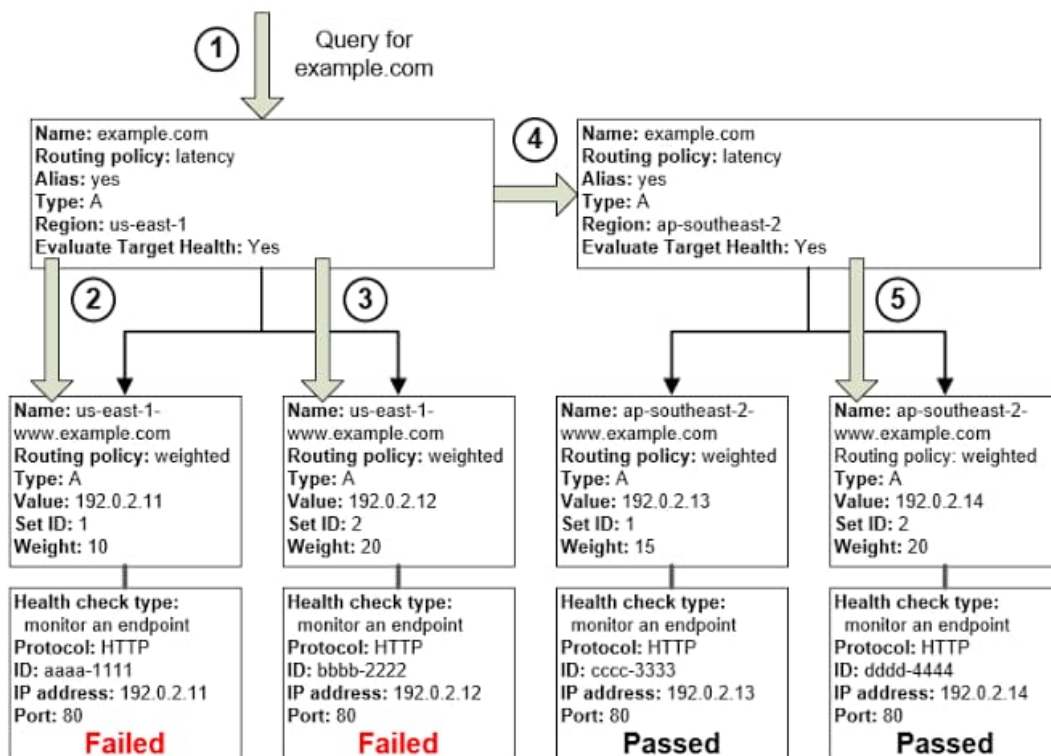


Figure 4.5: Sequence diagram for Failover Routing Policy

The above Figure 4.5 describes Sequence diagram for failover routing policy. Imagine a sequence of events like a conversation between different parts of the system. A user sends a request to your application. This request first reaches Route 53, your AWS DNS service. Route 53 might quickly check on the health of your EC2 instances in different Availability Zones (like asking if they're healthy). It then consults your failover policy (think of it as a rulebook for choosing healthy resources). Based on this information, Route 53 picks a healthy EC2 instance and directs the user's request there. The chosen EC2 instance then processes the request and sends a response back to the user.

However, the sequence diagram also considers situations where things might go wrong. If Route 53 finds all instances unhealthy, or if the chosen instance has an issue while processing the request, a failover happens. Route 53 selects a different

healthy instance based on the failover policy and redirects the user's request there. This ensures minimal disruption for the user, even during outages. This sequence diagram captures these interactions between user requests, Route 53's health checks and failover logic, and your EC2 instances, showcasing how the system works together to maintain application accessibility.

The sequence diagram hinges on the health checks performed by Route 53. These checks, like a doctor's visit for your EC2 instances, happen concurrently (at the same time) to ensure a quick assessment. Based on the results (healthy or unhealthy) and your failover policy's preferences (like prioritizing specific Availability Zones), Route 53 selects the most suitable instance. If a healthy instance is found, the user's request continues smoothly. However, if all health checks fail, indicating widespread issues, or if the chosen instance malfunctions during processing, Route 53 triggers its failover mechanism. This highlights the critical role of health checks in identifying potential problems and ensuring the entire system remains responsive by directing traffic to healthy resources.

The sequence diagram depicts a user's request navigating the intricate dance between health checks and failover policies. Imagine the user's request as a guest at a party, and Route 53 acts as the welcoming host. Route 53 first uses health checks, like quick security checks, to verify if any EC2 instances (the party rooms) are healthy and available. Based on the results and your pre-defined failover policy (think of it as the party seating chart, prioritizing specific locations or features), Route 53 selects the most suitable instance and directs the user's request there. If a healthy instance is found, the user seamlessly enters the party. However, if all health checks fail (indicating a power outage at the venue), or the chosen instance encounters an issue while processing the request (maybe the room is over capacity), Route 53 triggers a failover. It selects a different healthy instance based on the failover policy, essentially finding a new party room for the user. This sequence diagram emphasizes the dynamic interplay between health checks and failover policies, ensuring the user gets the best possible experience even during unexpected situations.

4.2.5 Activity Diagram for Failover Routing Policy

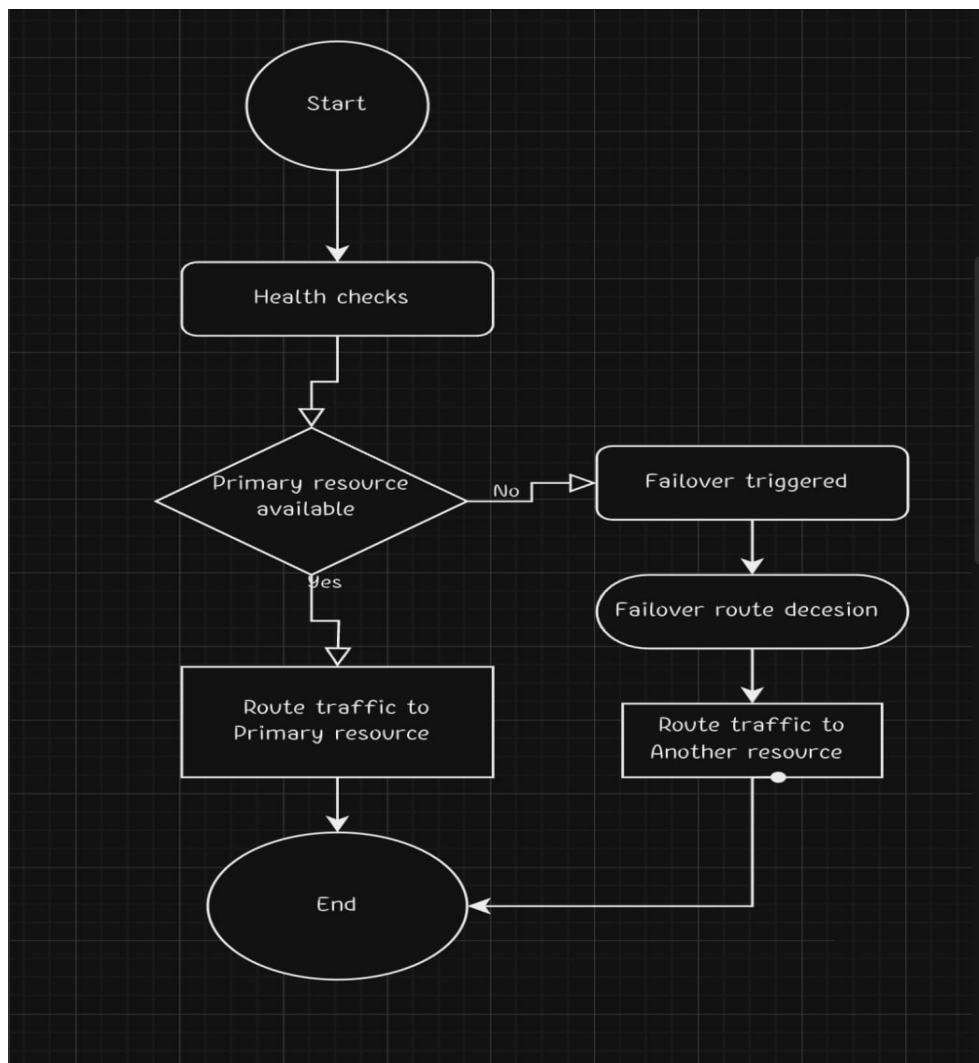


Figure 4.6: Activity diagram for Failover Routing Policy

The above Figure 4.6 describes Activity diagram for failover routing policy. Imagine a user requesting access to your application. This request travels through the internet and reaches Route 53, your reliable DNS service. Route 53, like a vigilant gatekeeper, first performs health checks on your EC2 instances across Availability Zones (think of them as individual buildings). These checks ensure the instances are healthy and ready to handle requests.

Once the health check results arrive, Route 53 determines if all instances are operational. If everything is healthy, Route 53 consults your failover policy (think of it as a traffic management plan). This policy might prioritize specific locations or distribute traffic evenly. Based on this plan, Route 53 selects a healthy EC2 instance as the primary source for handling the user's request.

Finally, Route 53 directs the user's request towards the designated primary source. The chosen EC2 instance then receives the request and starts processing it. The activity diagram can then branch out depending on the processing outcome: a successful response sent back to the user or a loop back to Route 53 for potential retries if an error occurs within the primary source. This simplified explanation portrays the initial flow of a failover routing policy, emphasizing how Route 53, health checks, and your failover policy work together to seamlessly route user traffic to healthy resources.

An activity diagram for a failover routing policy using AWS would depict the sequential steps and decision points involved in handling failover scenarios. It would typically start with the system monitoring the health status of primary servers. This monitoring includes checks for responsiveness, resource utilization, and overall server health. If the primary server experiences a failure or becomes unresponsive, the diagram would show a decision point where the failover policy is triggered.

4.3 Algorithm & Pseudo Code

4.3.1 Route 53 Algorithm

- Step 1: Create two EC2 instances in different availability zones
- Step 2: Configure the EC2 instances
- Step 3: Create Elastic IPs for both instances
- Step 4: Create an Amazon Route 53 hosted zone
- Step 5: Create two resource record sets in Route 53
- Step 6: Create a health check for each instance
- Step 7: Create a failover routing policy
- Step 8: Associate the health checks with the instances
- Step 9: Test the failover routing policy
- Step 10: Monitor the health of your instances
- Step 11: Stop

4.3.2 Route 53 Pseudo Code

```
1      # Function to perform a health check on an EC2 instance
2  def is_healthy(instance):
3
4      return True
5
6  def handle_success(response):
7
8      pass
9
10 def handle_failure(error):
11
12     pass
13
14 def main():
15
16     user_request = get_user_request()
17
18     instances = get_ec2_instances()
19     health_checks = [is_healthy(instance) for instance in instances]
20
21     all_healthy = all(health_checks)
22
23     if all_healthy:
24
25         primary_source = choose_healthy_instance(instances, failover_policy)
26
27         route_request(primary_source, user_request)
28
29         response = process_request(primary_source, user_request)
30         handle_success(response)
31     else:
32
33         healthy_instances = [instance for instance, health in zip(instances, health_checks) if health]
34
35         if healthy_instances:
36
37             primary_source = choose_healthy_instance(healthy_instances, failover_policy)
38             route_request(primary_source, user_request)
39             response = process_request(primary_source, user_request)
40             handle_success(response)
41         else:
42
43             trigger_failover()
44     if __name__ == "__main__":
45         main()
```

4.4 Module Description

4.4.1 Create a Failover Routing Policy

Failover routing within Route 53 empowers you to design highly available AWS projects. It works by directing traffic to a primary resource when it's healthy and automatically rerouting to a designated backup (secondary resource) during outages. This is achieved through Route 53 health checks that monitor the primary resource's health and Route 53 records configured with the failover routing policy. These records specify the primary and secondary resources. The improved disaster recovery by automatically rerouting traffic, minimized downtime for your users, and increased scalability by incorporating multiple backups. However, it's vital to define appropriate health checks to ensure accurate failover and consider your Recovery Time Objective (RTO) when designing the system. Finally, regular testing is crucial to guarantee the effectiveness of your failover routing during an actual outage. Failover routing within Route 53 isn't directly applicable to processing data within your AWS project. It specifically deals with managing traffic flow at the DNS level. However, failover routing can indirectly improve data processing by ensuring high availability for your resources. Here's how it connects: Imagine your data processing involves multiple resources working together. These resources could be EC2 instances, Lambda functions, or containers running on ECS.

By configuring failover routing for the DNS records pointing to these processing resources, you guarantee that even if a primary resource encounters issues, traffic automatically reroutes to a healthy backup. This uninterrupted traffic flow ensures that data keeps getting processed by the available resources, minimizing downtime and potential data loss. In essence, failover routing acts as a safety net for your data processing infrastructure, promoting its resilience and ability to handle disruptions.

The Failover Routing Policy module is designed to provide high availability and fault tolerance to applications and services deployed on AWS infrastructure. It leverages Amazon Route 53's DNS failover capabilities to automatically reroute traffic from unhealthy resources to healthy ones, ensuring minimal downtime and uninterrupted service delivery.

- **Health Checks:** The module performs regular health checks on target resources, such as EC2 instances or load balancers, to assess their availability and responsiveness.

- **DNS Failover:** In case of detected failures or unavailability, the module dynamically updates DNS records to direct traffic away from unhealthy resources to healthy ones.
- **Availability Zones Awareness:** It supports failover across different AWS Availability Zones, allowing for redundancy and resilience against zone-specific outages or disruptions.
- **Failover Policies:** Administrators can define customized failover policies based on health check results, specifying conditions for failover triggers and routing priorities.
- **Monitoring and Alerts:** The module provides monitoring and alerting functionalities to track the health status of resources and notify administrators of any detected failures or failover events.
- **Seamless Recovery:** Once the failed resource is restored to a healthy state, the module automatically switches back traffic to the primary resource, ensuring seamless recovery and minimal impact on end-users.

4.4.2 Test the failover Routing Policy

Testing a failover routing policy in AWS Route 53 requires a cautious approach to avoid disrupting live traffic. Here are two methods you can employ:

Method 1: Using a Testing Environment

1. Create a Staging Environment: Ideally, you should have a staging environment that mirrors your production setup, including resources and configurations. This allows testing without impacting live users.

2. Configure Failover Routing in Staging: Within your staging environment's Route 53 hosted zone, set up a record with the failover routing policy. Define the primary and secondary resources you want to test.

3. Simulate Primary Resource Failure: Here, you have options:

Manual Failover: If your primary resource allows it, you can temporarily take it offline to simulate a failure.

Mocking Tools: Consider using tools that mimic resource failure by returning pre-defined error responses during health checks.

4. Validate Traffic Routing: With the primary resource down (simulated or actual), use tools like 'dig' or 'nslookup' to query the DNS record. You should see the

response resolving to the secondary resource's IP address. This confirms failover functionality.

5. Restore Primary Resource: Once testing is complete, bring your primary resource back online (if manually disabled) and ensure it starts functioning normally.

Method 2: Using Route 53 Traffic Flow Visualizer

1. Enable Route 53 Traffic Flow Visualizer: This feature provides insights into traffic routing behavior. Activate it for the hosted zone containing your failover routing record.

2. Weighting for Testing: Route 53 allows you to temporarily assign a small weight to the secondary resource within the failover record. This routes a minimal portion of live traffic to the secondary resource for testing purposes.

3. Monitor Traffic Flow: Use the Traffic Flow Visualizer to observe how traffic gets routed. You should see a small percentage of queries resolving to the secondary resource.

4. Disable Weighting: After confirming functionality, remove the weight assigned to the secondary resource, ensuring all traffic flows to the primary again.

4.4.3 Create Routing Policy Management Module

Routing Policy Management Module

This module can be designed to manage failover routing policies within AWS Route 53. Here's a breakdown of its potential functionalities:

1. Policy Configuration:

- **Create Policy:** This function allows users to define a new failover routing policy.
- **Hosted Zone:** Selection of the relevant Route 53 hosted zone where the policy will be applied.
- **DNS Record:** Specification of the DNS record type (e.g., A record for hostname) and name for which failover needs to be configured.
- **Health Checks:** Configuration of health checks to monitor the primary resource's health. This includes specifying the health check type endpoint URL, and frequency.
- **Resource Records:** Defining both the primary and secondary resources involved in the failover. This could involve specifying their DNS names or IP addresses.

2. Policy Management:

- **List Policies:** This function retrieves and displays a list of all configured failover

routing policies within the system.

- **View Policy Details:** Users can select a specific policy to view its detailed configuration, including the associated hosted zone, DNS record, health check settings, and primary/secondary resources.

- **Edit Policy:** This function allows modifying an existing policy's details. Users can update health check configurations, resource records, or other relevant parameters.

- **Delete Policy:** This function enables users to remove an unwanted failover routing policy.

3. Testing and Monitoring:

- **Simulate Failover :** This advanced feature could allow simulating a primary resource failure for testing purposes. This might involve functionalities like:

1. Temporarily disabling health checks for the primary resource.
2. Triggering pre-defined error responses during health checks.

- **Traffic Flow Monitoring :** This feature would integrate with Route 53's Traffic Flow Visualizer to display how traffic gets routed for the chosen policy. This can be helpful for verifying failover functionality.

4.5 Steps to execute/run/implement the project

4.5.1 Resource Setup

- **Primary Secondary Resources:** Ensure you have your primary and secondary resources configured. These could be Application Load Balancers (ALBs) set up in different regions for better redundancy.
- **Health Check Creation:** Create a health check within Route53. This check will continuously monitor the health (availability and functionality) of both your primary and secondary resources.
- **Resource Health Verification:** Double-check that both your primary and secondary resources are functioning correctly before proceeding.
- **Route53 Resource Access:** Navigate to the Route53 console and locate the hosted zone you want to configure for failover routing.
- **Resource Readiness:** Ensure you have all the necessary information about your resources readily available, such as domain names and IP addresses. This will be used during the policy creation phase.

4.5.2 Policy Creation

- **New Record Target:** Access the Route53 console and create a new record within your chosen hosted zone. This record will initially point to your primary resource.
- **Failover Routing Selection:** During record configuration, choose "Failover routing" as the routing policy. This activates failover functionality for this specific record.
- **Primary Resource Designation:** Within the failover policy settings, designate your primary resource. By default, Route53 assumes the primary resource is healthy.
- **Secondary Resource Health Checks:** Define your secondary resource as the alternate that will take over if the primary fails. Crucially, enable health checks for the secondary resource to ensure its readiness.
- **Policy Review Save:** Carefully review your failover policy configuration, ensuring the primary and secondary resources are defined correctly, and health checks are enabled. Once satisfied, save the record to activate the policy.
- **Advanced Health Checks:** While basic health checks are included, Route53 offers more sophisticated options. You can configure checks based on specific HTTP status codes, path pings, or even full content checks to ensure your resources deliver the expected functionality.
- **Route53 Traffic Flow:** Utilize Route53 Traffic Flow to visualize traffic routing within your hosted zone. This visual aid helps understand how traffic flows during normal operation and during failover scenarios.

4.5.3 Test and Monitor

- **Simulate Failover:** After saving the record with the failover policy, simulate a failover scenario. This can be done by stopping your primary resource
- **Traffic Routing Verification:** Use tools like 'dig' or your browser to verify if traffic starts routing to the secondary resource. You can perform a DNS lookup for your domain name and see if it resolves to the IP address of your secondary resource.

- **Health Check Monitoring:** Continuously monitor the health checks associated with both primary and secondary resources in the Route53 console. This ensures they are functioning correctly and can detect any issues promptly.
- **Performance Observation:** Observe the performance of your application during the simulated failover. Look for any latency spikes or service disruptions to identify areas for improvement.
- **Failback Testing :** If desired, you can test failback functionality by restarting your primary resource and verifying traffic automatically routes back to it. This step is optional but helps validate the complete failover and failback process.
- **Latency-Based Routing:** If you have resources in geographically distributed regions, consider latency-based routing. Route53 can direct traffic to the resource with the lowest latency, improving user experience.
- **Security Groups:** Ensure your security groups are configured correctly to allow traffic flow between your resources and Route53 health checks. Improper security group rules can prevent successful failover.
- **Alerts and Notifications:** Set up alerts and notifications in Route53 or CloudWatch to receive timely updates regarding resource health or failover events. This allows for proactive troubleshooting and minimizes downtime.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design for Failover Routing

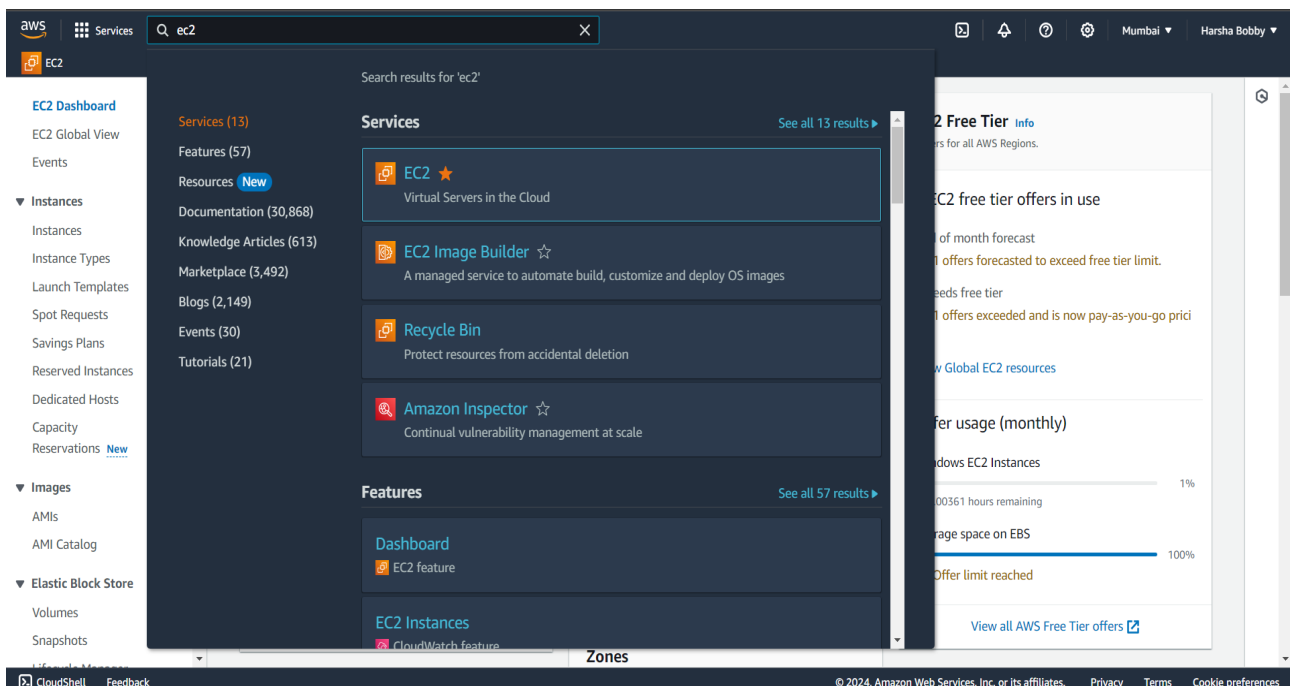


Figure 5.1: Creating EC2 Instances

- Create two EC2 instances in different availability zones The first step is to create two EC2 instances in different availability zones to ensure that if one zone goes down, the other will continue to operate. You can create EC2 instances by going to the EC2 dashboard and clicking on the "Launch Instance" button.
- Configure the EC2 instances After creating the EC2 instances, you need to configure them with the necessary applications and settings required for your application or website. You can use Amazon Machine Images (AMIs) or create your own custom AMIs to configure the instances.

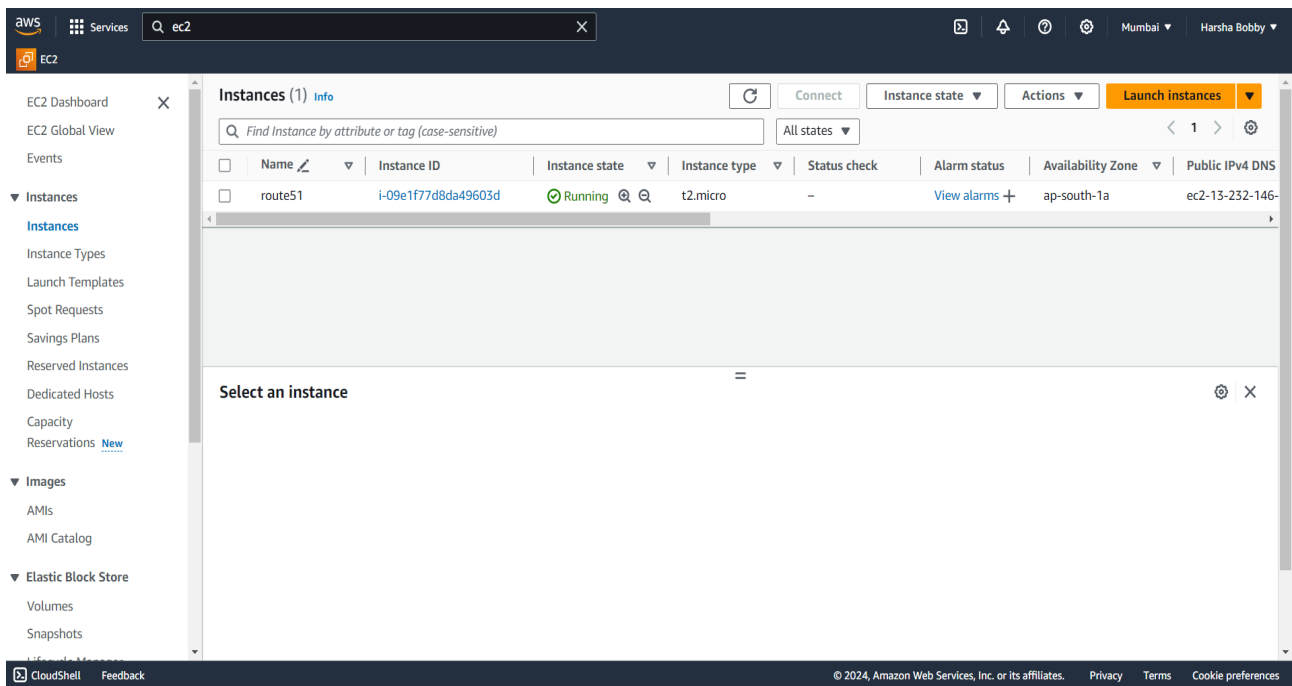


Figure 5.2: Successfully Created EC2 Instances

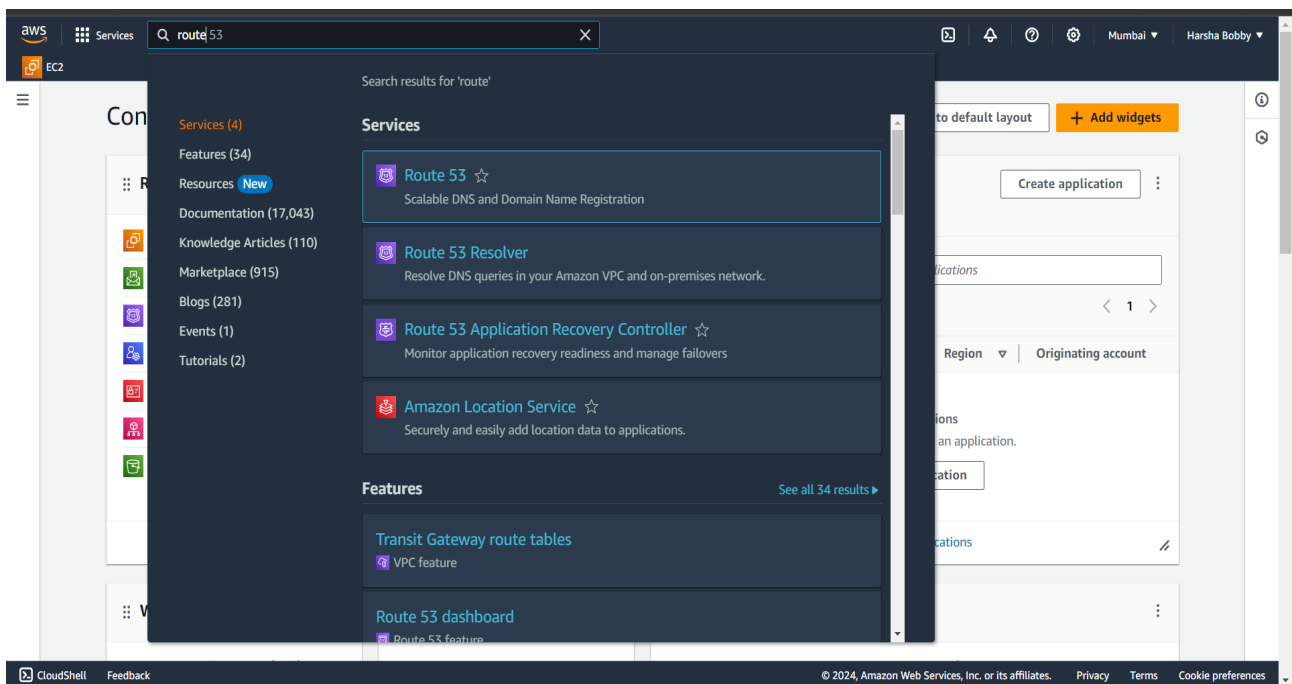


Figure 5.3: Create Route53 in AWS

- Click on Route 53 ,Amazon Route53 is an aws service than offers a DNS (Domain Name System) web service which is scalable and high available.It is essential for conversion of user friendly domain names into IP addresses so that internet communication can proceed without difficulties.

- Route 53 is a highly available, scalable Domain Name System (DNS) service. Released in 2010, its name refers to both the classic highway US Route 66 and the destination for DNS server requests: TCP or UDP port 53.

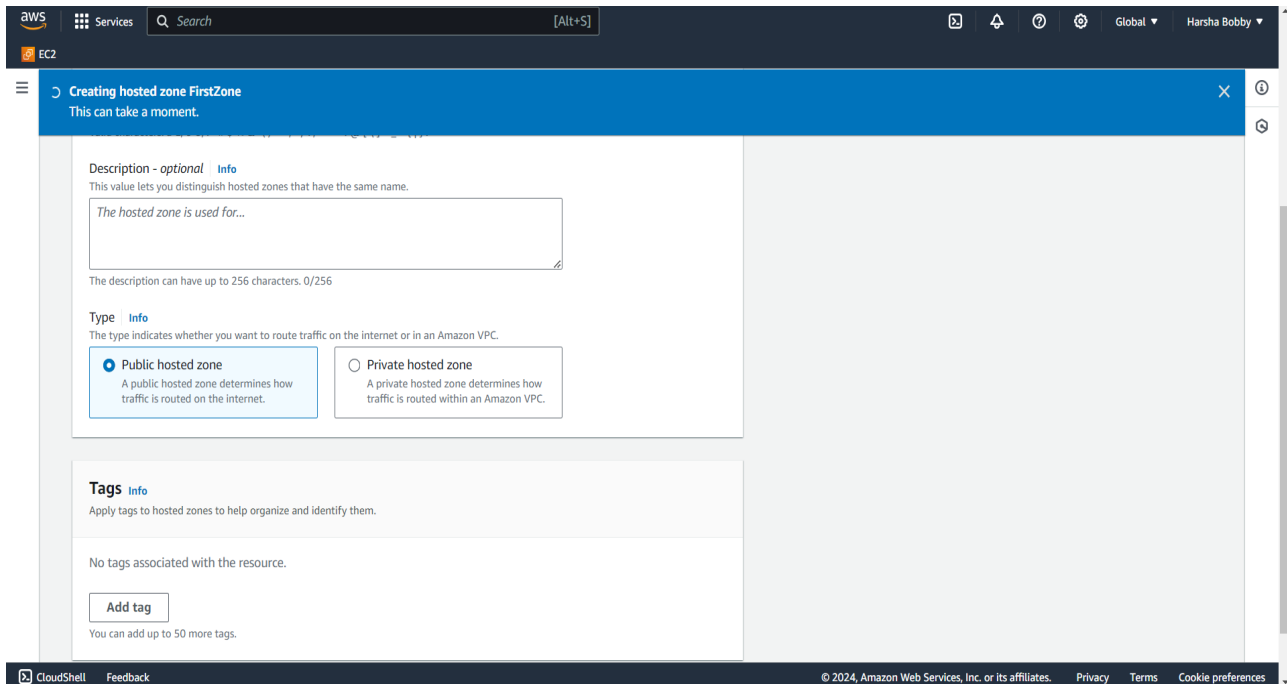


Figure 5.4: Create Hosted Zone in AWS

- Create an Amazon Route 53 hosted zone Next, you need to create an Amazon Route 53 hosted zone. A hosted zone is a container for all the DNS records for your domain. You can create a hosted zone by going to the Route 53 dashboard and clicking on the "Create Hosted Zone" button.
- Creating a hosted zone in AWS Route 53 involves navigating to the Route 53 service within the AWS Management Console after logging in. Once in the Route 53 dashboard, you proceed to the "Hosted zones" section and click on the "Create hosted zone" button. Here, you enter the domain name for which you want to create the hosted zone, such as example.com.
- After entering the domain name, you configure various settings for the hosted zone, including the routing policy (e.g., simple routing, failover routing), adding resource record sets (like A, CNAME, MX records), and defining optional settings like DNSSEC. It's essential to review these settings to ensure they align with your requirements.

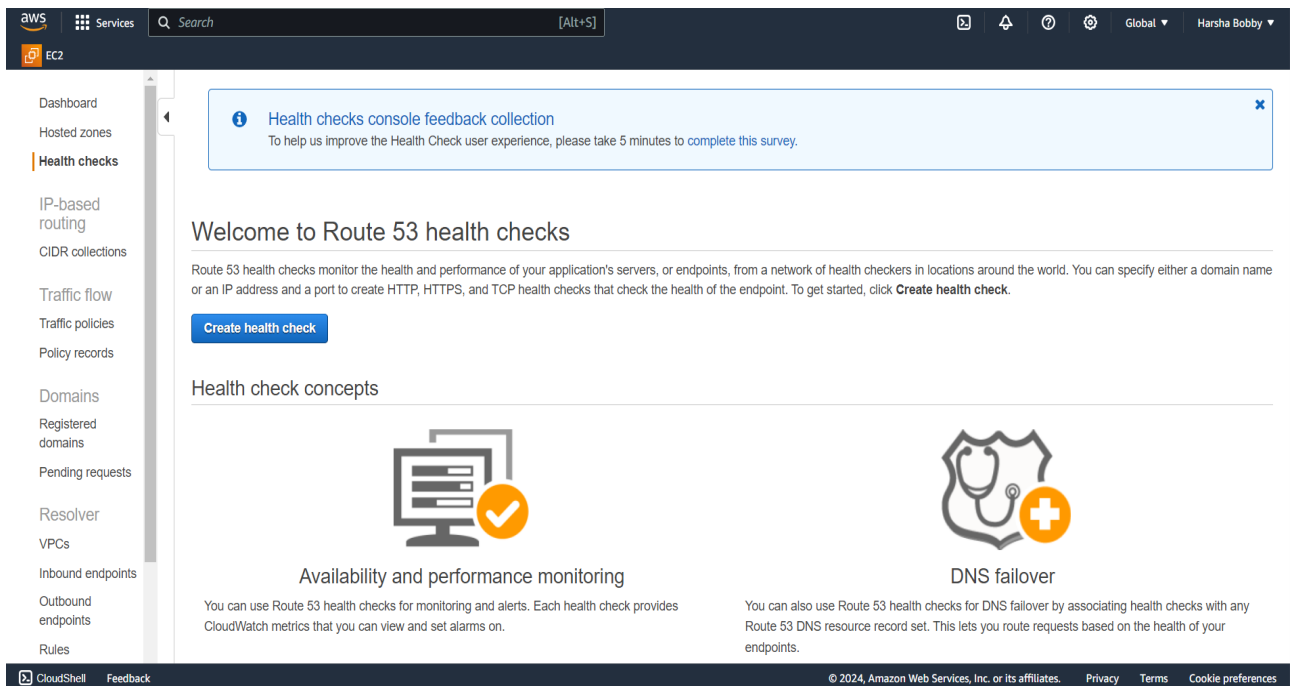


Figure 5.5: Create Health Checks

- Create a health check for each instance To ensure that Route 53 only routes traffic to healthy instances, you need to create a health check for each instance. You can create health checks by going to the Route 53 dashboard and selecting Health Checks from the left-hand menu.
- Creating health checks in AWS Route 53 is an essential step in maintaining the reliability and availability of your applications and services. To begin, log in to the AWS Management Console and navigate to the Route 53 service. In the Route 53 console, select the "Health checks" option to start creating a new health check.
- When creating a health check, you specify the endpoint or domain name to monitor, choose the protocol (such as HTTP, HTTPS, or TCP), define the port number, and set up additional parameters like the request interval and failure threshold. It's also important to select the AWS regions from which the health checks should originate to ensure comprehensive monitoring.
- Once you've configured the health check settings according to your monitoring requirements, review the configuration to ensure accuracy and then proceed to create the health check. AWS Route 53 will then start performing health checks based on the specified criteria, periodically checking the endpoint's health and providing status updates in the Route 53 dashboard. These health checks play a

critical role in identifying and addressing potential issues proactively, helping to maintain a robust and stable infrastructure for your applications and services.

5.1.2 Output Design for Created Records

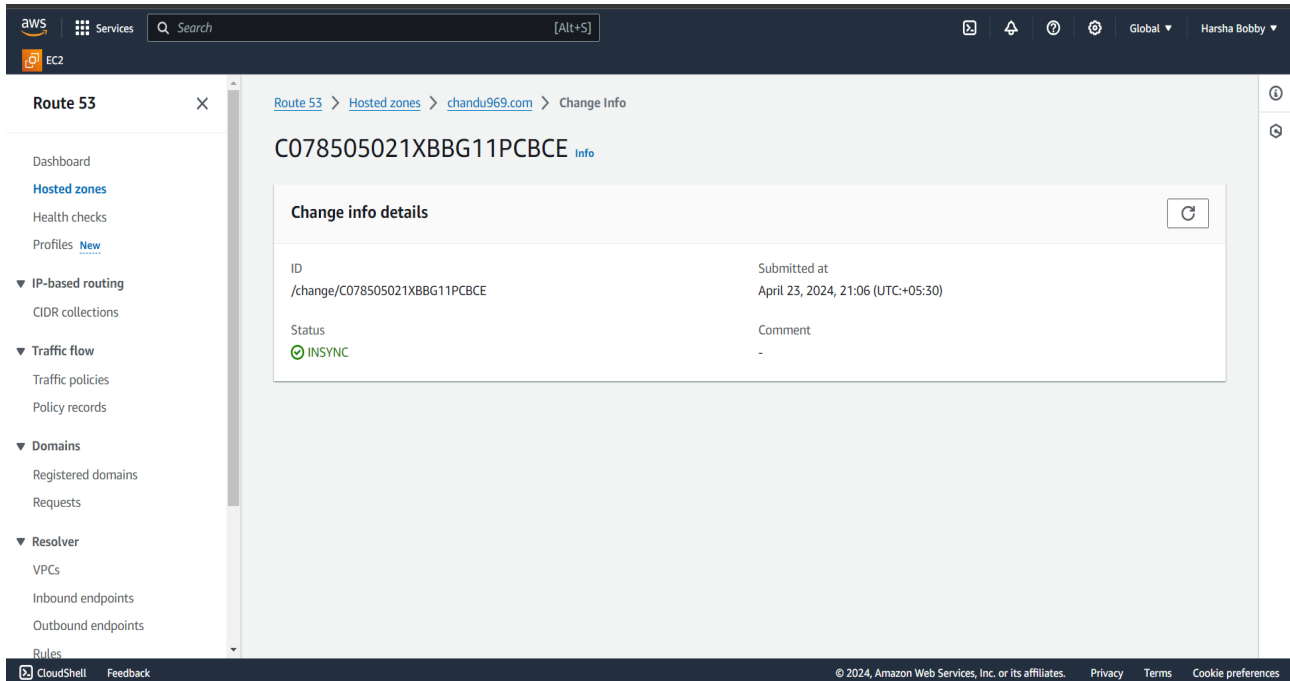


Figure 5.6: Successfully Created Records

- Certainly! Failover routing in Amazon Route 53 is a critical mechanism that ensures high availability and resilience for your resources. It involves setting up a primary record associated with the healthy resource and a secondary record linked to the backup resource. When the primary resource is healthy, Route 53 directs traffic to it.
- However, if the primary resource becomes unhealthy, Route 53 seamlessly switches to the secondary resource, ensuring continuity. Failover routing is essential for disaster recovery scenarios and can be used within private hosted zones in your Virtual Private Cloud (VPC). Regardless of the resource type, failover routing plays a crucial role in maintaining infrastructure availability. For more detailed information, refer to the official Amazon Route 53 documentation .

5.2 Testing

5.3 Types of Testing

5.3.1 Unit testing

Input

Unit testing plays a supporting role in ensuring the reliability of your failover routing policies in AWS Route 53. While it doesn't directly test the policies within Route 53 itself, it becomes valuable if you develop a custom module to manage these policies. Unit tests can verify functions that handle resource record creation and updates. This might involve ensuring compatibility with failover routing and proper formatting of resource record data (like IP addresses or DNS names). Error handling scenarios can also be tested by simulating potential issues during interactions with Route 53, verifying that your module responds appropriately and provides informative messages to the user. Results are verified to ensure that each component behaves as expected, and any deviations are debugged and documented. Iterative testing helps improve test coverage and accuracy.

```
1 class MockHealthCheck:
2     def __init__(self, healthy):
3         self.healthy = healthy
4
5     def is_healthy(self):
6         return self.healthy
7
8 # Function to parse failover policy configuration
9 def parse_failover_policy(config):
10     # Replace with your actual parsing logic
11     source = config.get("source")
12     destination = config.get("destination")
13     primary_path = config.get("primary_path")
14     secondary_path = config.get("secondary_path")
15     health_check = None
16     if config.get("health_check"):
17         health_check_config = config.get("health_check")
18         # Replace with actual health check parsing logic (e.g., interval, threshold)
19         health_check = MockHealthCheck(health_check_config.get("healthy", True))
20     return source, destination, primary_path, secondary_path, health_check
```

Test result :Passed

5.3.2 Integration testing

Input

Integration testing for failover routing policy using AWS is a critical step to validate the robustness and effectiveness of the failover mechanisms in your system. This testing phase involves verifying the seamless coordination and interaction among various AWS components to ensure smooth failover and recovery processes during adverse scenarios. During integration testing, you first set up a test environment that mirrors your production setup, including AWS resources like EC2 instances, Route 53 health checks, and DNS configurations. You then define a range of test scenarios that simulate different failure conditions, such as instance unavailability, region outages, or network disruptions. These scenarios help evaluate how well the failover routing policies respond to real-world incidents. During failover events, you verify that DNS resolution behaves as expected, routing traffic to healthy instances according to failover routing configurations. You also validate the recovery process to ensure that once the primary resources are restored, traffic is correctly redirected back to them without service disruptions.

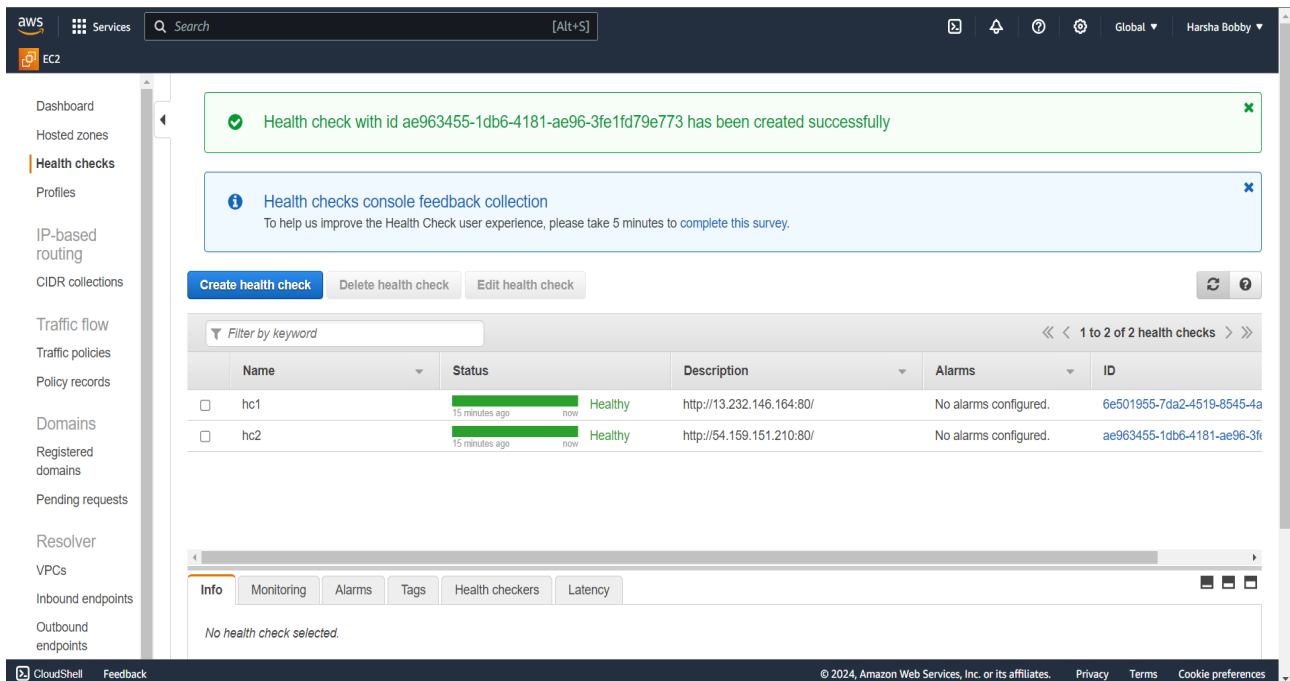


Figure 5.7: Health Checks Passed

Test result : Passed

5.3.3 System testing

Input

- Navigate to the Route 53 console.
- Click on “Health Checks” and choose “Create Health Check”.
- Specify the endpoint’s IP address or domain name, the protocol (HTTP, HTTPS, or TCP), and other relevant parameters.
- Optionally, configure notification settings to receive alerts when the endpoint is unhealthy.
- AWS allows creating custom health checks using scripts written in languages like Python or Shell. This flexibility enables you to design checks tailored to your specific testing needs. You can integrate these checks with tools like Amazon CloudWatch Logs to collect and analyze the results.

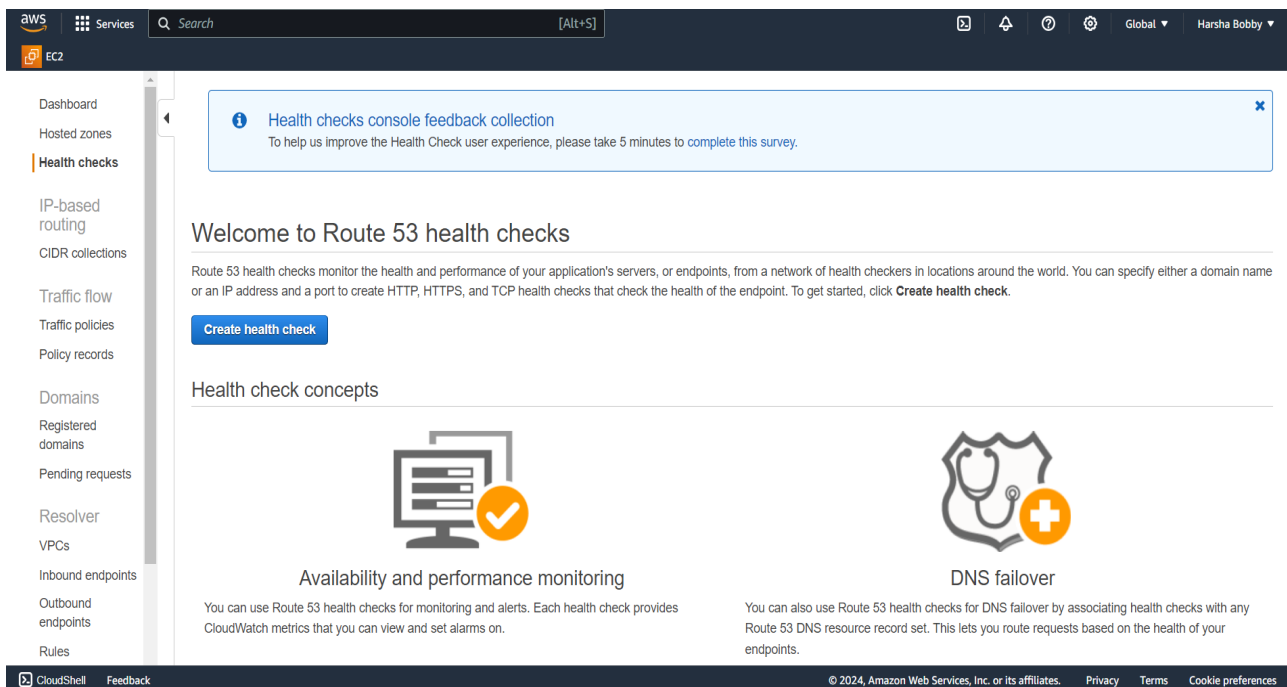


Figure 5.8: Checking Health Checks

Test Result : Passed

5.3.4 Test Result

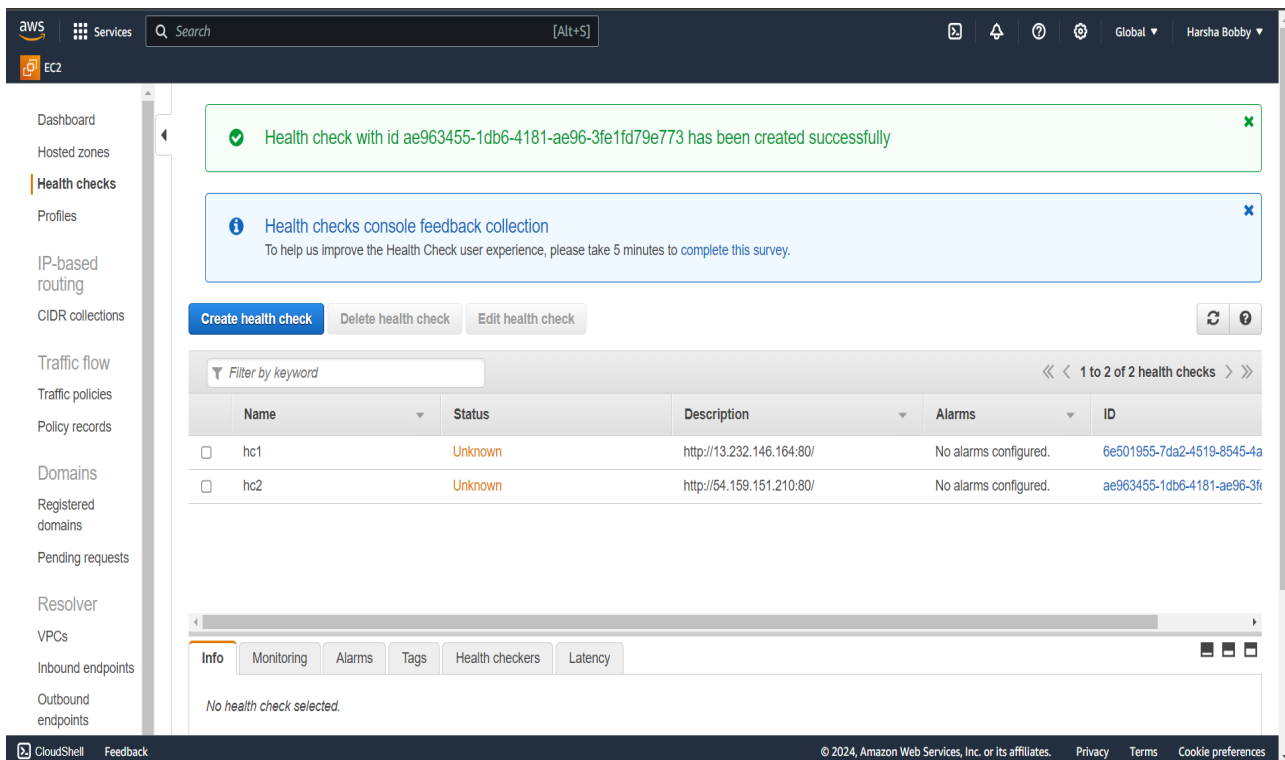


Figure 5.9: Test Image for Failover Routing Policy

A failover routing policy is a crucial aspect of modern network architecture, especially in cloud environments, where maintaining high availability and minimizing downtime are paramount. This policy ensures that traffic is automatically rerouted from a primary resource to a secondary or backup resource when the primary resource becomes unavailable due to failures, maintenance, or other issues.

The key components of a failover routing policy include continuous health monitoring of resources, predefined trigger conditions that initiate failover actions, an automated failover mechanism to redirect traffic to the backup resource, and the presence of redundant resources to handle the workload during failover events. Additionally, DNS management plays a vital role in updating DNS records dynamically to point to the backup resource's IP address during failover scenario. By implementing a failover routing policy, organizations can enhance their system's resilience, mitigate the impact of outages or disruptions, and ensure that critical services remain accessible to users without significant interruptions.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The efficiency of a failover routing policy using AWS can be evaluated through several aspects. Firstly, the automation and orchestration capabilities of AWS services such as Route 53 DNS Failover and AWS Global Accelerator can significantly enhance failover efficiency. These services can automatically detect failures in regions or endpoints and route traffic to healthy resources, reducing manual intervention and minimizing downtime. Additionally, leveraging AWS CloudWatch alarms and monitoring tools allows for real-time visibility into system health, enabling proactive failover actions based on predefined thresholds or metrics.

Secondly, the scalability and availability of AWS infrastructure play a crucial role in the efficiency of failover routing. AWS offers a global network infrastructure with multiple availability zones and regions, providing redundancy and high availability for failover scenarios. By designing failover routing policies that utilize AWS Regions and Availability Zones effectively, organizations can ensure that failover traffic is rerouted to geographically diverse and resilient resources. Implementing multi-region architectures, using services like AWS Auto Scaling and AWS Lambda for dynamic scaling, and regularly testing failover scenarios through AWS CloudFormation or AWS Disaster Recovery tools can further enhance the efficiency and reliability of failover routing policies on AWS.

6.2 Comparison of Existing and Proposed System

Existing system:(Manual Intervention)

In the existing failover routing system, manual intervention and limited automation are common. Failover processes often rely on human operators to de-

tect failures and initiate routing changes, which can lead to delays and increased downtime during critical events. The system may lack real-time monitoring and proactive failover capabilities, resulting in suboptimal performance and potential disruptions to service availability. In the existing failover routing system that relies heavily on manual intervention, there are inherent challenges and risks associated with human errors and delays. Manual failover processes often require trained personnel to identify issues, troubleshoot, and implement routing changes, which can be time-consuming and error-prone. Moreover, during high-stress situations such as widespread outages or network failures, the coordination and execution of manual failover procedures can be complex and prone to mistakes, leading to extended downtime and impact on user experience. The lack of automated failover mechanisms in such systems can hinder agility, scalability, and overall resilience, highlighting the need for a more automated and efficient failover solution like the proposed system leveraging AWS services.

Proposed system:(Failover Routing System)

On the other hand, the proposed failover routing system using AWS introduces advanced automation and orchestration capabilities. Leveraging AWS services such as Route 53 DNS Failover and AWS Global Accelerator enables automatic detection of failures and seamless rerouting of traffic to healthy resources across multiple regions and availability zones. This approach not only reduces the reliance on manual interventions but also improves scalability, resilience, and overall system efficiency. Additionally, AWS monitoring tools like CloudWatch provide real-time visibility into system health, allowing for proactive failover actions based on predefined thresholds or performance metrics, further enhancing the system's reliability and responsiveness during failover events.

Failover routing is a critical aspect of network design that ensures continuous availability and reliability of services by automatically redirecting traffic from failed or degraded resources to healthy ones. It involves defining alternate paths or backup resources to handle traffic in case of failures, minimizing downtime and service disruptions.

Failover routing strategies can vary based on the system architecture, such as using DNS-based failover, dynamic routing protocols, or application-level failover mechanisms, all aimed at maintaining seamless operations during unexpected failures or outages. Failover routing is a proactive approach to handling network failures, ensuring minimal disruption to services and maintaining high

availability. It involves monitoring the health and performance of network components and automatically redirecting traffic when issues are detected. Implementing robust failover routing mechanisms is essential for building resilient and fault-tolerant network infrastructures.

6.3 Sample Code

```
1 class Route:
2     def __init__(self, destination, primary_path, secondary_path=None):
3         self.destination = destination
4         self.primary_path = primary_path
5         self.secondary_path = secondary_path
6         self.healthy = True # Flag to track primary path health
7
8     def is_primary_healthy(self, health_check_function):
9         # Simulate a health check using a function (replace with actual implementation)
10        self.healthy = health_check_function(self.primary_path)
11        return self.healthy
12
13    def get_next_hop(self):
14        if self.healthy and self.primary_path:
15            return self.primary_path
16        elif self.secondary_path:
17            return self.secondary_path
18        else:
19            # Handle case where both paths are unavailable (raise exception or log error)
20            raise Exception("No healthy paths available")
21
22
23 def main():
24     # Sample routes
25     route1 = Route("10.0.0.0/16", "eth0")
26     route2 = Route("192.168.0.0/24", "eth1", "eth2")
27
28     # Sample health check function (replace with actual implementation)
29     def health_check(interface):
30         # Simulate a ping check
31         return True # Replace with actual health check logic
32
33     # Check route health and get next hop
34     for route in [route1, route2]:
35         if route.is_primary_healthy(health_check):
36             next_hop = route.get_next_hop()
37             print(f"Route: {route.destination}, Next Hop: {next_hop}")
38         else:
39             print(f"Route: {route.destination}, Primary Path Unhealthy")
40
41 if __name__ == "__main__":
```

Output

The failover routing policy in AWS ensures high availability and reliability by automatically redirecting traffic from one server to another in case of a failure. In this scenario, with one server running in Mumbai and another in North Virginia, the failover routing policy would continuously monitor the health of both servers. If the Mumbai server experiences downtime or becomes unreachable, the policy would seamlessly redirect incoming traffic to the North Virginia server, ensuring uninterrupted service for users. This setup enhances fault tolerance and minimizes service disruptions, crucial for maintaining a robust and dependable system architecture in distributed cloud environments. In AWS, you can configure a failover routing policy using Route 53 to ensure high availability for your web service. This policy directs traffic to a primary server in Mumbai. If that server fails a health check, Route 53 automatically switches traffic to the secondary server in North Virginia, minimizing downtime for your users. This ensures service continuity even if one server location experiences an outage.

For high availability in AWS, configure Route 53 with a failover routing policy for your domain. Traffic initially goes to your primary server in Mumbai. If Mumbai fails a health check, Route 53 automatically switches traffic to the secondary server in North Virginia, ensuring service continuity during outages. This minimizes downtime and keeps your users happy.

The failover routing policy continuously monitors the health and availability of server instances across all availability zones. It dynamically adjusts routing decisions based on real-time health check results, ensuring that traffic is always directed to healthy and operational instances. This approach enhances system reliability, fault tolerance, and ensures that critical services remain accessible even in the face of infrastructure failures or disruptions within specific availability zones.

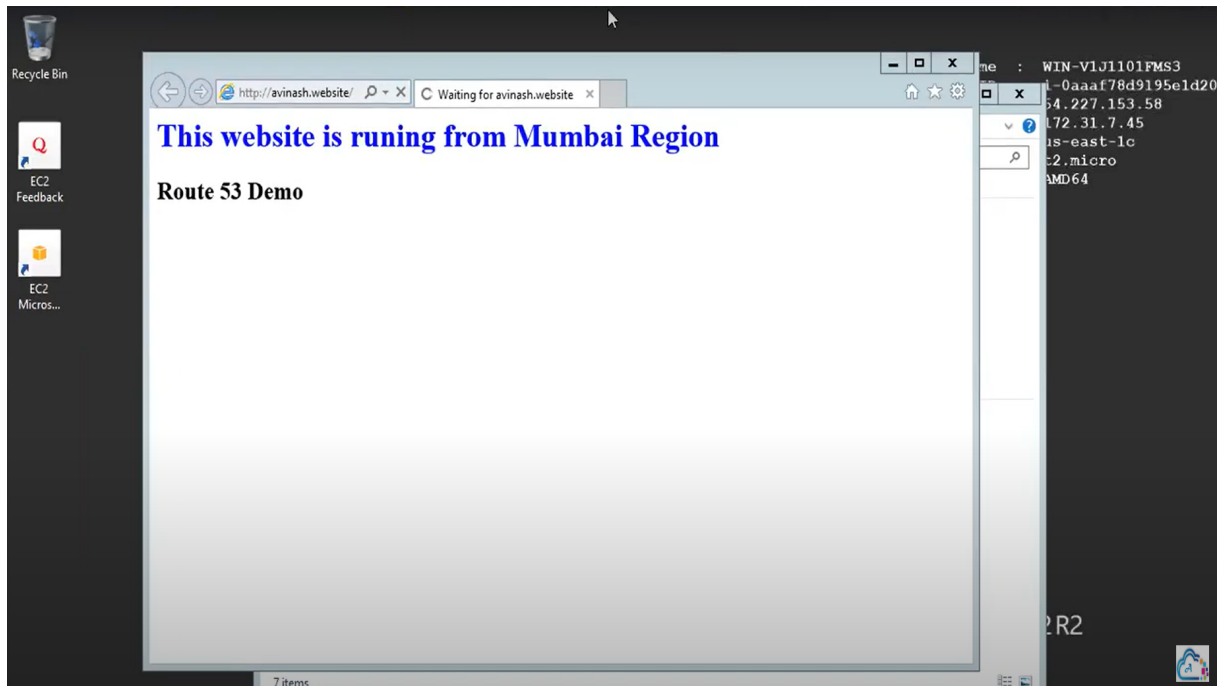


Figure 6.1: Output 1 for Failover Routing Policy

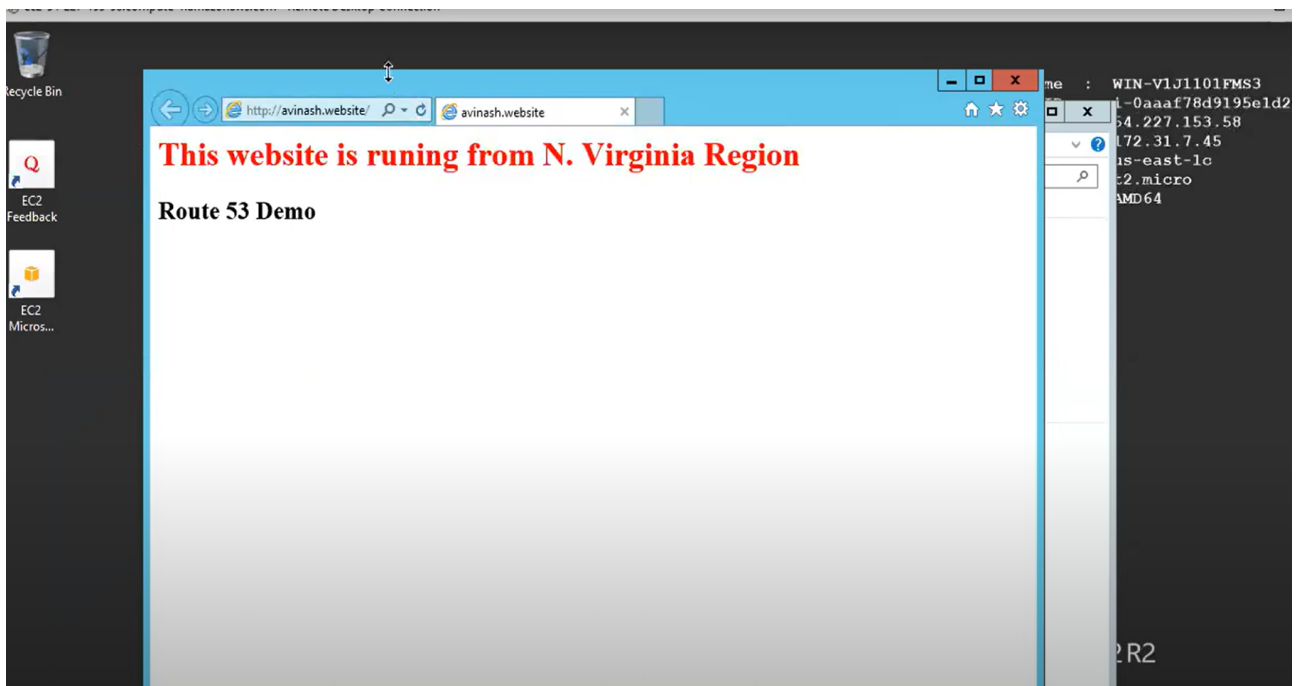


Figure 6.2: Output 2 for Failover Routing Policy

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

Implementing a failover routing policy in AWS using Route 53 and health checks provides a robust solution for ensuring high availability and minimizing downtime. By leveraging AWS services, organizations can automate failover processes, monitor endpoint health, and route traffic to healthy resources seamlessly during failures or degraded conditions. This approach improves system resilience, reduces manual intervention, and enhances overall reliability. Additionally, AWS's global infrastructure with multiple regions and availability zones offers geographical redundancy, further strengthening the failover capabilities. Overall, a well-designed failover routing policy in AWS contributes to a more resilient and fault-tolerant architecture, ensuring continuous operations and improved user experience even during unexpected events.

Implementing a failover routing policy in AWS not only enhances system reliability but also contributes to cost-effective and scalable infrastructure management. By automating failover processes and leveraging AWS's pay-as-you-go pricing model, organizations can optimize resource utilization and reduce operational costs associated with manual interventions or prolonged downtime. Additionally, AWS's scalability allows failover routing policies to adapt to changing traffic patterns and scale resources dynamically, ensuring efficient resource allocation during normal operation and failover scenarios. This combination of reliability, cost-effectiveness, and scalability makes AWS a compelling choice for implementing robust failover routing policies that meet the demands of modern, resilient IT infrastructures.

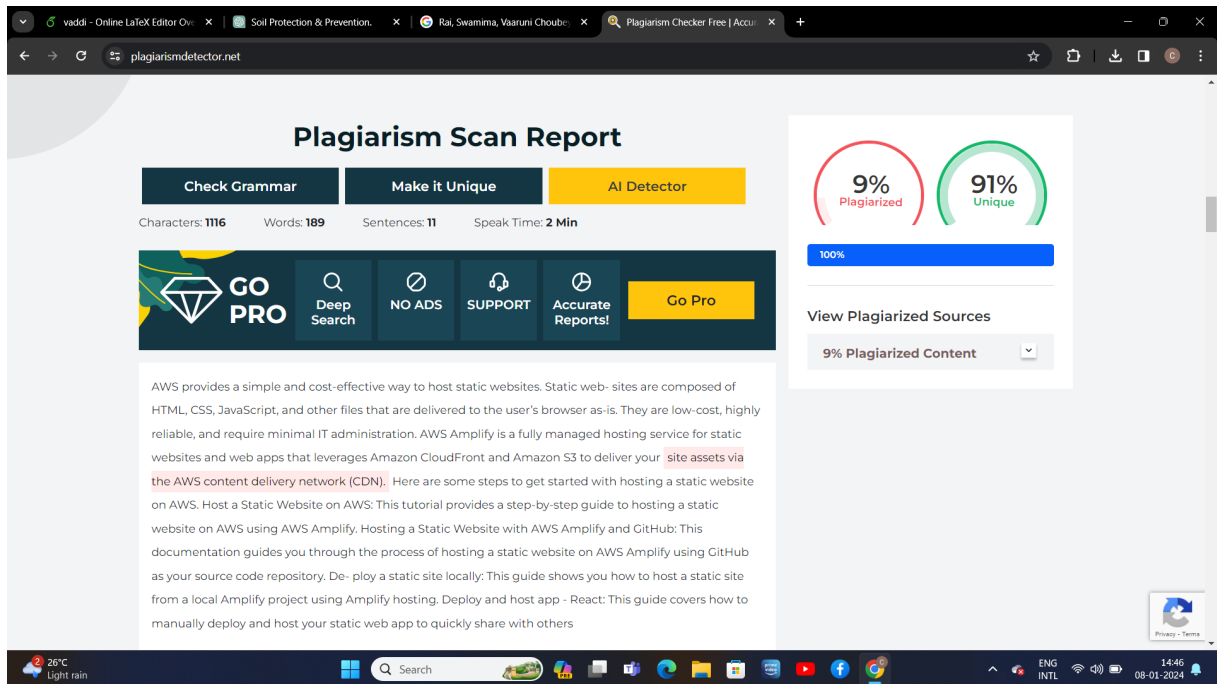
7.2 Future Enhancements

Building upon a strong foundation of failover routing, several strategies can further enhance your system's resilience and performance. Multi-region failover with geographically redundant resources in AWS regions provides a safety net during regional outages. Active-active failover utilizes both primary and secondary resources for load balancing and reduced latency, requiring additional configuration for synchronization. Additionally, integrating application-level failover mechanisms like database replication ensures data consistency during disruptions.

To streamline failover processes and improve overall system health, automation plays a key role. Auto Scaling and AWS Lambda can dynamically provision resources based on traffic demands, while advanced monitoring with CloudWatch and SNS allows for automated responses to pre-defined events or anomalies. Regularly testing your failover plan through disaster recovery drills and simulations helps identify and address any potential issues before real-world incidents occur. These strategies, combined with a robust failover routing policy, contribute to a highly available and reliable infrastructure.

Chapter 8

PLAGIARISM REPORT



Chapter 9

SOURCE CODE & POSTER PRESENTATION

9.1 Source Code




```
1 import boto3
2
3 # Initialize Route 53 client
4 route53_client = boto3.client('route53')
5 # Define the hosted zone ID and record set name
6 hosted_zone_id = 'YOUR_HOSTED_ZONE_ID'
7 record_set_name = 'example.com'
8 failover_record_set_name = 'failover.example.com'
9 # Define the primary and secondary endpoint information
10 primary_endpoint = 'PRIMARY_ENDPOINT_IP'
11 secondary_endpoint = 'SECONDARY_ENDPOINT_IP'
12 # Create the primary and secondary record sets
13 primary_record_set = {
14     'Name': record_set_name,
15     'Type': 'A',
16     'SetIdentifier': 'Primary',
17     'Failover': 'PRIMARY',
18     'TTL': 60,
19     'ResourceRecords': [{ 'Value': primary_endpoint }]
20 }
21 secondary_record_set = {
22     'Name': record_set_name,
23     'Type': 'A',
24     'SetIdentifier': 'Secondary',
25     'Failover': 'SECONDARY',
26     'TTL': 60,
27     'ResourceRecords': [{ 'Value': secondary_endpoint }]
28 }
29 # Create the failover record set
30 failover_record_set = {
31     'Name': failover_record_set_name,
32     'Type': 'A',
33     'SetIdentifier': 'Failover',
34     'Failover': 'FAILOVER',
```

```

35     'TTL': 60,
36     'ResourceRecords': [{ 'Value': primary_endpoint }],
37     'HealthCheckId': 'YOUR_HEALTH_CHECK_ID',
38     'TrafficPolicyInstanceId': 'YOUR_TRAFFIC_POLICY_ID'
39 }
40 {
41     "Action": "UPSERT",
42     "ResourceRecordSet": {
43         "Name": "example.com",
44         "Type": "A",
45         "SetIdentifier": "Secondary",
46         "Failover": "SECONDARY",
47         "TTL": 60,
48         "ResourceRecords": [{ "Value": "SECONDARY_ENDPOINT_IP" }],
49         "HealthCheckId": "YOUR_HEALTH_CHECK_ID"
50     }
51 # Create or update the primary and secondary record sets
52 route53_client.change_resource_record_sets(
53     HostedZoneId=hosted_zone_id,
54     ChangeBatch={
55         'Changes': [
56             {
57                 'Action': 'UPSERT',
58                 'ResourceRecordSet': primary_record_set
59             },
60             {
61                 'Action': 'UPSERT',
62                 'ResourceRecordSet': secondary_record_set
63             }
64         ]
65     }
66 )
67
68 # Create or update the failover record set
69 route53_client.change_resource_record_sets(
70     HostedZoneId=hosted_zone_id,
71     ChangeBatch={
72         'Changes': [
73             {
74                 'Action': 'UPSERT',
75                 'ResourceRecordSet': failover_record_set
76             }
77         ]
78     }
79 )

```

9.2 Poster Presentation



FAILOVER ROUTING POLICY USING AWS

Department of Information Technology
School of Computing
10214IT602- MINOR PROJECT-II
WINTER SEMESTER 2023-2024

ABSTRACT

The Failover Routing Policy in AWS provides a resilient solution for directing traffic between primary and secondary resources, ensuring high availability and fault tolerance. Leveraging Route 53's health checks, this policy dynamically routes traffic to alternative endpoints when the primary resource becomes unavailable. By configuring health checks and setting up failover rules, AWS users can seamlessly manage failover scenarios, maintaining service continuity and minimizing downtime for applications and services hosted on the AWS infrastructure. In implementing the Failover Routing Policy, AWS users can customize routing behavior based on various criteria such as endpoint health, geographic location, or latency. This flexibility allows for tailored failover strategies to meet specific application requirements and optimize performance. Moreover, by integrating with other AWS services like Elastic Load Balancing and Amazon CloudWatch, users gain deeper insights into resource health and performance.

INTRODUCTION

1)Health Checks: AWS Route 53 continuously monitors the health of endpoints by performing health checks. These checks assess the availability and performance of resources, such as EC2 instances, ELB load balancers, or S3 buckets.

2)DNS Failover: With DNS failover, Route 53 automatically routes traffic away from unhealthy or degraded endpoints to healthy ones based on the results of health checks. This dynamic rerouting occurs at the DNS level, ensuring swift and seamless failover without manual intervention.

3)Active-Active and Active-Passive Configurations: Organizations can configure failover routing in AWS in either an active-active or active-passive configuration. In an active-active setup, both primary and secondary endpoints actively serve traffic, providing scalability and load distribution. In contrast, an active-passive configuration maintains a standby secondary endpoint that becomes active only when the primary endpoint fails, conserving resources and cost.

4) In conclusion, implementing a robust failover routing policy using AWS empowers organizations to build resilient, fault-tolerant architectures that can withstand disruptions and deliver uninterrupted services to end-users. By leveraging AWS Route 53 and its suite of features, businesses can enhance their operational resilience and maintain a competitive edge in today's digital landscape.

RESULTS

The results of implementing a Failover Routing Policy using AWS often demonstrate significant improvements in system reliability, uptime, and user experience. By leveraging AWS's robust infrastructure and features like Route 53's health checks and failover rules, organizations can achieve seamless failover between primary and secondary resources. This ensures that services remain available even in the event of outages or disruptions, minimizing downtime and preserving business continuity. Additionally, the customizable nature of the Failover Routing Policy allows for tailored strategies that align with specific application requirements, optimizing performance and resource utilization. Overall, the adoption of this policy enables businesses to build resilient architectures on AWS, leading to enhanced reliability, scalability, and customer satisfaction.

STANDARDS AND POLICIES

1)AWS Well-Architected Framework: Follow the principles outlined in the AWS Well-Architected Framework, which includes reliability, performance efficiency, and operational excellence pillars. This framework provides guidance on designing and operating reliable, scalable, and cost-effective systems on AWS.

2)Service Level Agreements (SLAs): Define SLAs for failover scenarios, including target recovery times, availability goals, and performance metrics. Ensure that these SLAs align with business requirements and customer expectations.

3)AWS Route 53 Health Checks: Configure health checks in AWS Route 53 to monitor the health of primary and secondary resources. Define thresholds and criteria for determining resource availability and triggering failover actions.

4)Failover Routing Policies: Establish clear failover routing policies based on predefined criteria such as endpoint health, geographic location, or latency. Document these policies and ensure that they are regularly reviewed and updated as needed.

METHODOLOGIES

1)Risk Assessment: Conduct a thorough risk assessment to identify potential failure scenarios and their impact on business operations. This assessment helps prioritize resources for failover routing implementation and determine the level of redundancy required.

2)Architecture Design: Design a resilient architecture that incorporates failover routing policies using AWS services such as Route 53, Elastic Load Balancing (ELB), Amazon CloudWatch, and AWS Lambda. Consider factors like geographic redundancy, fault tolerance, and scalability to build a robust failover solution.

3)Health Monitoring: Implement robust health monitoring mechanisms to continuously monitor the health and availability of primary and secondary resources. Utilize Route 53 health checks, CloudWatch alarms, and other monitoring tools to detect failures and trigger failover actions automatically.

CONCLUSIONS

In conclusion, the implementation of a Failover Routing Policy using AWS is a critical component of building resilient and highly available architectures in the cloud. By leveraging AWS services like Route 53 and adhering to best practices and standards, organizations can ensure seamless failover between primary and secondary resources, minimizing downtime and preserving business continuity. The establishment of clear failover routing policies, automated processes, and comprehensive testing procedures contributes to the effectiveness and reliability of the failover mechanism.

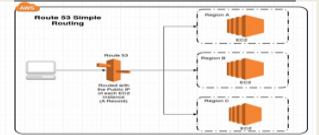
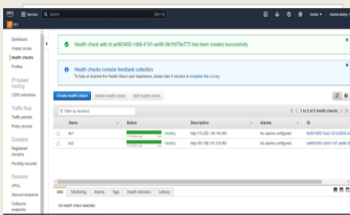
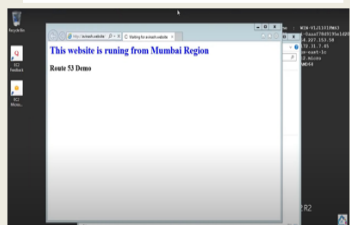
ACKNOWLEDGEMENT

1. Dr. C. Suresh Kumar ME, Ph.D.,
2. 9500684095
3. dsureshkumar@veltech.edu.in

TEAM MEMBER DETAILS

1) E. Sai Koteswara Rao(vtu19657)
(9391453758)
vtu19657@veltech.edu.in

1) G. Poorna Chandu(vtu19504)
(9346906729)
vtu19504@veltech.edu.in



47

References

- [1] Ambika Gupta, Anjani Mehta, Lakshya Daver, Priya Banga, “Implementation of Storage in Virtual Private Cloud using Simple Storage Service on AWS,” 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020.
- [2] Anirban Mukhopadhyay, Sourav Ghosh, “On the Performance of Deep Harnessing AWS for static sites,” The increased automation and integration with emerging technologies, IEEE, 2006.
- [3] A. S. Rodge, C. Pramanik, J. Bose, S. K. Soni “Multicast Routing with Load Balancing using Amazon Web Service ”Annual IEEE India Conference (INDICON), 2014.
- [4] Brian Beach, “Achieving High Availability with Amazon Route 53 Health Checks and DNS Failover”. Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018), 22-24 July 2017.
- [5] David Clinton, Ben Piper, AWS Certified Solutions Architect Study Associate SAA-C02 Exam “The Domain Name System and Network Routing: Amazon Route 53 and Amazon CloudFront.” 2021.
- [6] Darus, Mohamad Yusof, and Muhammad Azizi Mohd Ariffin, “Enhancement Keylogger Application for Static Host Journal of AWS Websites,” 6.3, 8482-8492, 2020.
- [7] Hrishikesh Dewan, R.C. Hansdah, “A Survey of Cloud Storage Facilities,” IEEE World Congress on Services, 2011.

- [8] H. Huang, S. Guo ,IEEE Communications “Magazine, Proactive Failure Recovery for NFV in Distributed Edge Computing”, 2019.
- [9] K. Veena, C. Anand, G. C. Prakash “Temporal and Spatial Trend Analysis of Cloud Spot Instance Pricing in Amazon EC2”, 2016 .
- [10] Patterson,“High Availability System Design”, 11th International Conference on Signal Processing System(ISSPS) 2020
- [11] Urvesh Domadiya,“Disaster Recovery with Route 53’s Failover Routing Policy”.AWS Certified Solutions Architect Study Associate (SAA-C02) June 2021.

General Instructions

width=!,height=!,page=-