Azrieli College of Engineering Jerusalem עזריאלי מכללה אקדמית להנדסה ירושלים

## המחלקה להנדסת תוכנה

## פרויקט גמר – ה׳תשפ״ד

# גיבוי לבית מלון

# reservation backups for hotels

**מאת**

**Saja Abu Aisheh**

**ת.ז. 308444223**

**מנחה אקדמית: ד"ר Shimrit Tzur**

מערכות ניהול הפרויקט:

| # | מערכת | מיקום |
|---|---|---|
| 1 | מאגר קוד | https://github.com/sajabilal/reservation-backup-for-hotels |
| 2 | יומן | https://docs.google.com/spreadsheets/d/1DIBM57YSvE4yOHi6ZkDjtKxJR9NmcMMiaQtz846IhaQ/edit?gid=0#gid=0 |

מידע נוסף (מחקו את המיותר)

| | |
|---|---|
| סוג הפרויקט | יוזמה שלי |
| פרויקט ממשיך | זה פרויקט חדש |

## Introduction

In the context of big online platforms, fully solving downtime issues is a serious matter due to its cruciality. An online reservation system, for example, can endure losses in the millions if the users lose access to their data, even for a few hours. In this project, I am creating a Disaster Recovery as a Service (DRaaS) solution focused on lowering the impact of disaster. Which provides seamless access to reservation data for both customers and employees during outages.

DRaas solutions available in the market provide multiple mechanismswhich vary between those which work in the field of prevention from disasters and others which work on lowering the time to recover or the losses caused by a disaster. Still, none provide a solution during an outage which takes the whole system down.

DRaas solutions traditionally depend on traffic draining to a safe clone of the system where service can be resumed. On the contrary, this solution resides outside the system providing a 100% data reach possibility for customers and employees during a disaster which takes down the whole system.

The solution I am proposing uses AWS route53 health checks to detect issues in the main system, and reroute traffic seamlessly into another read-only secondary system -kept outside of the main network- that consists of an AWS S3 bucket for the front end along with AWS Lambda and a database as the backend as well as an AWS CloudFront for the CDN (Content Delivery Network). This will be kept in sync by performing regular backups which will take place between the primary and secondary databases.

# Problem Description

## Problem Requirements and Characterizations

The proposed solution provides the customer the ability to readily get access to his/her data during an outage by inserting booking ID or any other agreed-upon data.

When a customer requests the out-of-service page (the main website) he/she will be notified about the outage and the redirection by a ready webpage and then redirected to the secondary webpage where the customer is able to query the data needed, this also includes employees.

This solution is secure since the users will only be allowed to read their data if they have the required token.

## The problem in terms of software engineering

**Expected Challenges**

1. **Seamless Failover Implementation:**
   This is a technical challenge; the health checks and the failover need to be well-maintained in order to ensure the well-performing of the solution.

2. **Data Synchronization:**
   The frequency of main database and secondary database backup needs to be precise which will majorly affect the efficiency of the solution alongside the solution costs.

3. **Access Control Without Customer Credentials:**

   while not using login credentials the solution still needs to be accessed securely by using unique identifiers and multi-field data.

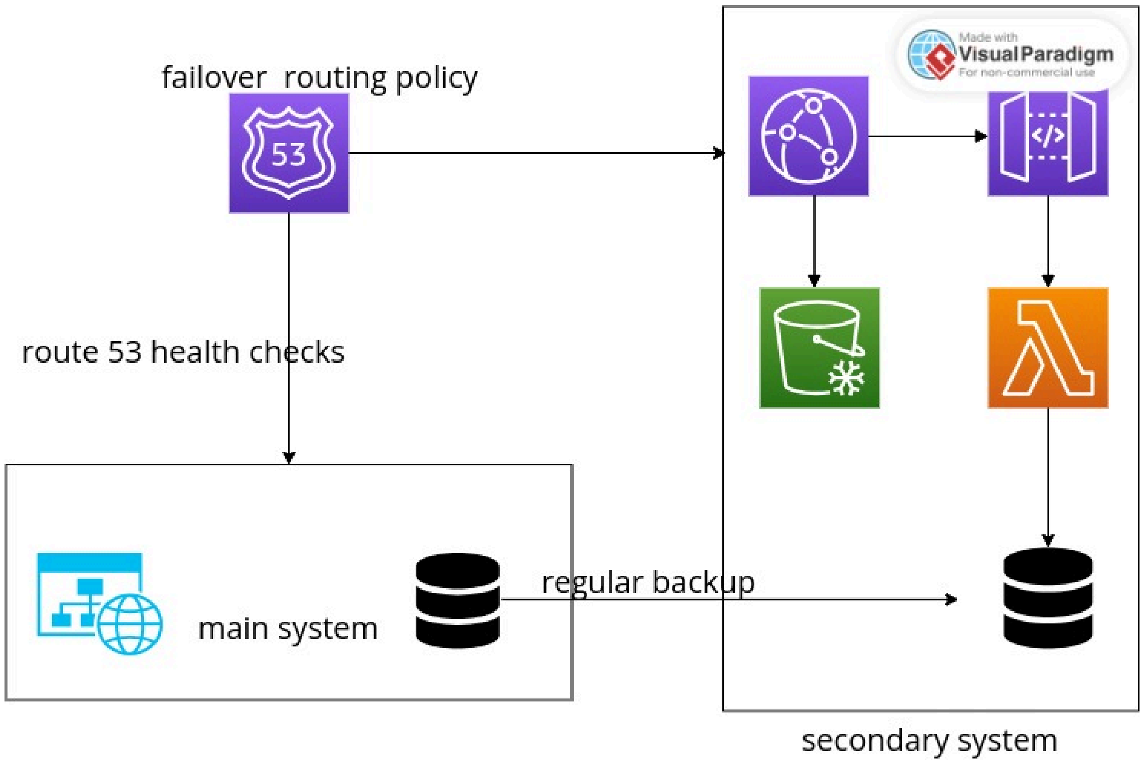
4. **System Security and Data Integrity:**

   The data residing in the secondary database needs to be safe and secure and that requires implementing encryption, transport protocols and access control policies.

5. **Scalability and Cost Optimization:**
   In AWS, querying databases increases the cost of the usage, meaning when building this solution the perfect balance between scaling the traffic directed to the solution and the cost needs to be found. therefore I will be utilizing some of the serverless services.

6. **User-Friendly Design:**
   A clear interface and an easy to use one needs to be implemented in this solution to ensure a seamless experience for the customer during an outage.

## Solution Description



The system implementing the proposed solution consists of two main components: the **primary system** and the **secondary system**, working together to ensure continuous access to reservation data during system downtime.

1. **Primary System:**
   The primary system operates as the main service, handling user requests and serving data during normal operation (system uptime). It is the primary point of interaction for users under regular conditions.

2. **Secondary System:**
   The secondary system functions as the failover solution, designed to take over during downtime. It ensures that users can still query reservation data when the primary system is unavailable. The secondary system will be a read-only, reverse index, NoSQL data storage to guarantee fast and simple database querying.

**AWS Route 53** is a critical component of this solution. It continuously performs health checks on the primary system to monitor its availability. In the event of a failure or an unhealthy status, Route 53 automatically reroutes the traffic to the secondary system using its failover routing policy.

The secondary system leverages several AWS services:

- **Amazon S3:** Hosts the static content, including the HTML and CSS files for the alternative user interface.
- **CloudFront:** Acts as a content delivery network (CDN) to efficiently serve the static website with low latency from edge cache.
- **API Gateway and Lambda:** Serve as the backend of the secondary system, processing user requests and interacting with the database.
- **Secondary Database:** A read-only database that stores synchronized reservation data, enabling user queries during downtime.

Together, these components create a seamless and automated failover system that ensures uninterrupted access to critical reservation data, enhancing user experience and system reliability.

This solution is an add-on to all Disaster Recovery as a Service (DRaaS) services used in a well-maintained infrastructure since it doesn't replace any of the approved and ready-to-use solutions; it rather solves another issue that available DRaas services don’t cover.

**An example use case for this solution:**

In 2021, Facebook performed a routine service pack which caused the DNS along with the routers to stop functioning causing the users and employees to not be able to access the Facebook accounts, Instagram, Messenger and Watsapp.

## Similar Existing Solutions And Comparison

| Platform/Service | Description | Estimated Cost (as of Nov 2024) | Services Provided during a catastrophic outage |
|---|---|---|---|
| **Pantheon, WP Engine, Vercel, Netlify** | Hosting platforms offering caching, failover, CDN, and serverless functions for static and dynamic sites. | Pantheon: $41/month+; WP Engine: $30/month+; Vercel: $20/month+; Netlify: $19/month+ | Primarily serve cached static content; dynamic functionalities are unavailable if DNS or API endpoints fail. |
| **Cloudflare Always Online** | Caches static versions of websites to serve during origin server outages, integrated with the Internet Archive. | Free plan available; higher-tier plans offer additional features | Serves cached static pages; dynamic functionalities (e.g., forms, shopping carts) are typically non-functional during outages. If DNS is down, services are inaccessible. |
| **Your Proposed Solution** | Custom AWS-based failover system for read-only access to reservation data during downtime. | Estimated $100–$500/month, depending on scale | Users get rerouted into a secondary minimal read-only system, with the ability to query their data using a reservation ID or forms of identity. |