**Resume Website Deployment**

**University of Canada West**

**Shilpi Kumari**

# Table of Contents

[Table of Contents 2](#_Toc194006231)

[1. Introduction 3](#_Toc194006232)

[2. Architecture Components 3](#_Toc194006233)

[2.1 AWS S3 3](#_Toc194006234)

[2.2 AWS Dynamodb 3](#_Toc194006235)

[2.3 AWS Lambda Function 3](#_Toc194006236)

[2.4 API Gateway 4](#_Toc194006237)

[2.5 USER 4](#_Toc194006238)

[2.6 Architecture Diagram of Serverless Website 4](#_Toc194006239)

[3.Serverless versus Traditional Architecture 5](#_Toc194006240)

[4. Deployment of serverless website 5](#_Toc194006241)

[5. Estimated Billing Cost 17](#_Toc194006242)

[6. Deployment URL 17](#_Toc194006243)

[References 18](#_Toc194006244)

# Introduction

This report contains a detailed analysis, deployment, and implementation of a serverless architecture used to host a resume webpage with a dynamic visitor counter. This report also includes the cost analysis of the components used in deployment. This AWS solution comprises many AWS services like Amazon S3, AWS Lambda, DynamoDB and API Gateway to ensure a top-notch architecture ensuring scalability, reliability, and cost-efficiency.

# 2. Architecture Components

## 2.1 AWS S3

**The S3** component of AWS is used for hosting static websites, where multiple objects can be used. It provides high availability and durability. This can be configured for public access. This static webpage loads and contains the script for fetching visitor counts.

## 2.2 AWS DynamoDB

**DynamoDB,** a No-SQL database, is used for storing visitor data counters. It is fast, scalable and cost-effective. A single table with attributes: VisitorID (Primary key) and Count. The updated count is stored and returned to the webpage for display.

## 2.3 AWS Lambda Function

**The** Lambda function is a serverless function created for updating and retrieving the visitor count. The code is written in Java and connected to DynamoDB. It also eliminates the need for traditional server management.

## 2.4 API Gateway

**The** API gateway acts as an interface between the webpage and Lambda, providing secure and scalable end API points. This also handles the HTTP request to retrieve and update the visitor counter. The script calls an API gateway endpoint that triggers the AWS lambda function.

## 2.5 USER

Any user or administrator visiting the resume site hosted on S3.

## 2.6 Architecture Diagram of Serverless Website

**Figure 1 –**

**A diagram of a software flow

Description automatically generated with medium confidence**

# 3.Serverless versus Traditional Architecture

**Table 1 – Features of Serverless and Traditional**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Serverless (AWS Lambda)** | **Traditional (EC2, RDS)** |
| **Scalability** | Auto-scaling based on usage. | manual scaling required. |
| **Cost** | Pay-per-use, lower cost for low traffic | Fixed cost for running instances |
| **Maintenance** | No server management | Requires updates and monitoring |
| **Availability** | High availability by design | Requires load balancer |
| **Performance** | Optimized for event-driven workloads | Better for long-running processes |

# 4. Deployment of serverless website

|  |
| --- |
| **Figure 2 S3 – Creating S3 Bucket** |
| **A screenshot of a computer  Description automatically generated** |
| **Figure 3 - Enabling static website hosting.**  **A screenshot of a computer  Description automatically generated** |
| **Figure 4 – Uploading objects (index.html)**  **A screenshot of a computer  Description automatically generated** |
| **Figure 5 – Updating Bucket Policy**  **A screenshot of a computer  Description automatically generated**  **Figure 6 – Encountered challenge in editing bucket policy as resource was not correctly updated above.**  **A screenshot of a computer  Description automatically generated**  **Figure 7 – Successfully edited bucket policy.**  **A screenshot of a computer  Description automatically generated** |
| **Figure 8 - DynamoDB- creating VisitorCounter Table**  **A screenshot of a computer  Description automatically generated**  **A screenshot of a computer  Description automatically generated** |
| **Figure 9 – Creating ITEM**  **A screenshot of a computer  Description automatically generated** |
| **Figure 10 - Creating AWS Lambda Function, Since IAM Role didn’t have permissions for creating an execution role for DynamoDB access, instead I chose an existing Lab Role for creation of AWS Lambda Function**  **A screenshot of a computer  Description automatically generated** |
| **Figure 11 – Adding the given source code for deployment of lambda function.**  **A screenshot of a computer  Description automatically generated** |
| **Figure 12 - Role permission didn’t allow to give full access for DynamoDB.**  **A screenshot of a computer  Description automatically generated** |
| **Figure 13 – Creating API Gateway**  **A screenshot of a computer  Description automatically generated** |
| **Figure 14 – Adding a New Method**  **A screenshot of a computer  Description automatically generated** |
| **Figure 15 – Enabling CORS for PROD**  **A screenshot of a computer  Description automatically generated** |
| **Testing the API -** <https://gb09pd3az2.execute-api.us-east-1.amazonaws.com/Prod> |
| **Updating the final JavaScript for modified information and updating the visitor count.**  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Shilpi Kumari - Resume</title>  <style>  body {  font-family: Arial, sans-serif;  margin: 40px;  padding: 20px;  background-color: #f4f4f4;  }  .container {  max-width: 800px;  background: white;  padding: 20px;  border-radius: 8px;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  }  h1, h2 {  color: #333;  }  .section {  margin-bottom: 20px;  }  .contact {  font-size: 14px;  margin-bottom: 10px;  }  ul {  padding-left: 20px;  }  </style>  </head>  <body>  <div class="container">  <h1>Shilpi Kumari</h1>  <p class="contact">  Email: <a href="mailto:kr.shilpi21@gmail.com">kr.shilpi21@gmail.com</a> |  Phone: +1 604-906-4935 |  <a href="https://www.linkedin.com/in/shilpi-kumari" target="\_blank">LinkedIn</a>  </p>  <div class="section">  <h2>Objective</h2>  <p>Hardworking individual with excellent teamwork, communication, and IT skills. Skilled in configuring, maintaining security standards, routing, and switching to maximize network efficiency.</p>  </div>  <div class="section">  <h2>Experience</h2>  <h3>Associate Consultant - Atos Global IT Solutions & Services Pvt. Ltd (May 2021 — Present)</h3>  <ul>  <li>Conducted reviews to improve operations for multi-vendor devices like Cisco, Juniper, FortiGate & F5.</li>  <li>Implemented firewall policies and security measures.</li>  <li>Mentored trainees and conducted technical enhancement sessions.</li>  </ul>  <h3>Process Specialist - Infosys (June 2019 — May 2021)</h3>  <ul>  <li>Optimized fiber network implementation processes.</li>  <li>Troubleshot WiFi networks, Cisco Meraki Solutions & Cisco devices.</li>  </ul>  <h3>Administrator - Wipro Technologies (July 2015 — June 2019)</h3>  <ul>  <li>Managed LAN/WAN administration and troubleshooting.</li>  <li>Configured and maintained network hardware and software.</li>  </ul>  </div>  <div class="section">  <h2>Education</h2>  <p><strong>MBA</strong> - University of Canada West (Jan 2024 — Ongoing) | GPA: 3.0</p>  <p><strong>BCA</strong> - Patna University (July 2012 — May 2015) | 71%</p>  </div>  <div class="section">  <h2>Certifications</h2>  <p>CCNA 200-125, Cisco Meraki Black Belt</p>  </div>  <div class="section">  <h2>Skills</h2>  <p><strong>Technologies:</strong> Cisco, Juniper, FortiGate, F5</p>  <p><strong>Routing:</strong> BGP, OSPF, EIGRP</p>  <p><strong>Tools:</strong> MS-Word, MS-Excel</p>  </div>  <div class="section">  <h2>Visitor Count</h2>  <p>Visitors: 683</p>  </div>  </div>  </body>  </html> |
| **Figure 16 : Updated the final index.html in S3 bucket.**  **A screenshot of a computer  Description automatically generated**  **Figure 17 – Final testing of Website**  **A screenshot of a white and green document  Description automatically generated**  **A screenshot of a computer  Description automatically generated**  **URL for website** - <http://ucw-resume-site.s3-website-us-east-1.amazonaws.com> |

4.1 **Key Design Decisions**

* Using AWS S3 instead of EC2 to host a static website, will largely reduce cost.
* Using DynamoDB because of its serverless nature and auto-scaling features.
* Implementing APIgateways to expose endpoints securely instead of direct lambda invocation.

4.2 **Challenges Encountered –**

* AWS Bucket policy updating was a challenge faced, and after correcting the correct resource name bucket policy was successfully implemented.
* **Unable** to give full access to DynamoDB due to restricted role access.
* **Configuring** CORS was quite a challenge, after enabling access control for GET and OPTIONS and defining the correct headers, it was successfully implemented.

# 5. Estimated Billing Cost

**Table 2 – Billing Cost**

|  |  |  |
| --- | --- | --- |
| Services | Estimated Usage | Cost Estimate (USD) |
| S3 Hosting | 10,000 Requests, 1GB storage | Free Tier / ~$0.10 |
| Lambda | 100,000 requests, 256MB memory | Free Tier / ~$0.50 |
| API Gateway | 100,000 API calls | ~$0.40 |
| DynamoDB | 25,000 reads/writes | Free Tier / ~$0.20 |
| Total Estimate | Under Free Tier | ~$1.20/month |

**Note**: This table shows the approximate cost, buts cost can vary on actual usage and regions.

# 6. Deployment URL

[**http://ucw-resume-site.s3-website-us-east-1.amazonaws.com**](http://ucw-resume-site.s3-website-us-east-1.amazonaws.com)

# References

* Amazon Web Services, Inc. (n.d.). AWS free tier. Amazon Web Services, Inc. [https://aws.amazon.com/free/?trk=31516735-77bb-4e6f-b031-2649837284cd&sc\_channel=ps&s\_kwcid=AL!4422!10!71880800488451!!!!71881348337014!!482210304!1150090143938060&ef\_id=234ff8407e301695bcf99f9f2d697b22:G:s&msclkid=234ff8407e301695bcf99f9f2d697b22&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=\*all&awsf.Free%20Tier%20Categories=\*all](https://aws.amazon.com/free/?trk=31516735-77bb-4e6f-b031-2649837284cd&sc_channel=ps&s_kwcid=AL!4422!10!71880800488451!!!!71881348337014!!482210304!1150090143938060&ef_id=234ff8407e301695bcf99f9f2d697b22:G:s&msclkid=234ff8407e301695bcf99f9f2d697b22&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)
* Amazon Web Services, Inc. (n.d.). AWS Free Tier. Retrieved from <https://aws.amazon.com/free/>
* Amazon Web Services, Inc. (n.d.). AWS S3 Storage and Optimization. Retrieved from <https://aws.amazon.com/s3/>
* AWS Well-Architected Framework (2022). Best Practices for Cost Optimization in AWS Cloud. Retrieved from <https://docs.aws.amazon.com/wellarchitected/latest/framework/cost-optimization.html>