Project Title: Rowby Pierre’s Resume (Azure Cloud edition)

Project Overview: Host my resume on a static website via a custom domain name and track the number of visitors through several Azure solutions (listed below).

Features: Static resume deployed to Azure Blob storage with java script displaying a visitor counter and triggering an Azure App function. The app functions interacts with Azure Cosmos Database services where the counter log is stored.

Getting Started: Required Tools:

- Azure Subscription

- .Net Core 7 SDK

- Visual Studio Code

- Azure Functions Core Tools Extension

- C# Extension

Usage: Resume can be accessed via <https://www.rowbypierre.me/>

Technologies Used: Visual Studio Code, Azure Cosmos Database, Azure App Functions, Azure Storage Account, Azure Content Delivery Network, Google Domain Name System Services, GitHub Actions

Credits: Special thanks to Gwyn Pena-Siguenza (https://www.youtube.com/@MadeByGPS), CeeVee Templates (https://www.styleshout.com/free-templates/ceevee/), Forrest Brazeal “The Cloud Resume Challenge Guidebook” (https://cloudresumechallenge.dev/)

Deployment: DIRECTIONS BELOW ARE FOR DEPLOYING AZURE CLOUD RESUME UNIQE TO USER. THESE ARE NOT THE STEPS THE CREATOR FOLLOWED TO CREATE THIS REPOSITORY.

Frontend:

1. Modify HTML template (resume information) frontend/index.html

(2) Replace frontend/images/me.png with photo of one self, keep naming structure

(9) Deploy frontend/ to static website via Azure Storage

- set index document to index.html

(10) Add provided primary web endpoint from dialog box to Azure > Function App > [Function App Name] > CORS > Allowed Origins

(11) Get domain from preferred domain name server provider, create corresponding CNAME record

(12) Create Azure > Storage Accounts > [Storage Account Name] > Blob service | Azure CDN > New endpoint

(13) Add New endpoint & custom domain to Azure > Function App > [Function App Name] > CORS > Allowed Origins

(14) Azure > Storage Accounts > [Storage Account Name] > [Endpoint Name ] Create custom domain with domain from step 11& Enable HTTPs for custom domain

CLOUD RESUME SHOULD BE ACCESSIBLE VIA CUSTOM DOMAIN AFTER PROPAGATING

Backend:

(3) Setup serverless Azure Cosmo DB account, database, container (named “counter” , partition key “/id”)

- add container item:

{

“id”: “1”

“count”: 0

{

- update AzureResumeConnectionString variable in backend/api/local.settings.json file with primary connection string (Azure > Cosmos DB account > Key > Primary Connection )

(4) Update Cosmos DB bindings in backend/api/GetResumeCounter.cs

- databaseId, containerId, cosmosDbConnectionString

(5) Test function from terminal using “func host start” , enter provide URL in preferred web browser. Count should increase at increment of 1 for every time the page is refreshed.

- use provided URL and update frontend/main.js constant “function Api”

(6) Deploy function to Azure from backend/api directory:

- advanced settings, .NET Core 7, Linus OS, Consumption hosting plan, use resource group with cosmos DB account, use region closest to you

(7) Azure > Function App > Configuation: add “New application setting” with AzureResumeConnectionString variable in backend/api/local.settings.json file

(8) Update constant “functionApiURL” from frontend/main.js with Azure > Function App > [Function App Name] > Overview > “Get Function URL”

CI/CD Pipeline:

(15) Create .github directory

- create workflows/frontend.main.yml and workflows/backend.main.yml

(16) Create Azure service principle named AzureResume with contributor role for subscription, generating authentication credentials

- “az ad sp create-for-rbac --name AzureResume --role contributor --scopes /subscriptions/subscription-id/resourceGroups/resource-group-name --sdk-auth”

- create repository secret with JSON return values in terminal

(17)Copy #6 template from [Use GitHub Actions to deploy a static site to Azure Storage | Microsoft Learn](https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-static-site-github-actions?tabs=userlevel) to .github/workflows/frontend

- set paths to frontend/\*\*, update account names, CDN profile name, corresponding resource group, endpoint name

(18)Test frontend workflow

- insert comment in index.html, add & commit changes to GitHub

- verify workflow updates in GitHub, visiti website and confirm update

(19) Setup Unit Test: execute following code in VS Studio Terminal from test directory

- dotnet new xunit [.NET command line tool, create new xUnit test project]

- dotnet add package Microsoft.AspNetCore.Mvc [add NuGet Microsoft.AspNetCore.Mvc package to project]

- dotnet add reference ../api/api.csproj [adding reference to api project]

(20)Modify backend.main.yml

- Paste template from https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-net

- set paths to backend/\*\*, change function name, set azure package path to backend, alter dot net version, ensure folder name, and proper credentials being referenced are correct. Verify target frameworks in both api and tests csproj files are the same

- push modification to GitHub

(21)Test backend workflows

- add comment in GetResumeCoutner.cs file, commit and push changes from backend directory within terminal

- verify workflow in GitHub, troubleshoot any errors