Overview

The following tasks are provided to every candidate to properly evaluate and cover the position requirements.

This test contains 8 tasks, You don't need to do all the tasks, but of course you can do all :)

Conditions

- You need to do at least 4 tasks to be able to enter to the evaluation.
- The more tasks you do, the more you are in preferred candidates for this position.

Tasks

Task 1: Repository Initialization for Task Publication and CI/CD Implementation

Objective: Set up an empty repository for example on GitHub or GitLab which should be used for the publication of the completed tasks.

Expected Deliverable:

- A URL to the newly created repository on GitHub or GitLab.

Task 2: CSV Property Display in PowerShell

Objective: Utilize PowerShell to read a CSV file and display a specific property in the console.

Instructions:

1. Choose a sample CSV file as your input file (for example Top 10000 Books - Good Reads Dataset) goodreads_cleaned.csv
2. Read the content of the file that contains various properties
3. Extract and display at least two properties for each entry.
4. For each displayed entry, include its order/position in the CSV (e.g., "The Hunger Games is the 1st entry in the CSV.").

Expected Deliverable:

- A PowerShell script that performs the above operation.

Task 3: JSON Property Manipulation and CSV Output in Powershell

Objective: Read a JSON file, modify its properties, merge with another JSON, and save the result as a CSV.

Instructions:

1. Read the file “tv_shows_and_movies_sample.json" and remove the property “scraped_at”.tv_shows_and_movies_sample.json
2. Read a second file “movies.csv” and add at least the “imdb_votes” value for each movie that matches by “name”.movies.csv
3. Convert the combined data into a CSV file named "output.csv".

Expected Deliverable:

- A PowerShell script that reads, manipulates, and exports the data as described.

Task 4: Azure create Resource

Objective: Create an Azure Functions app tailored for running backup scripts.

Instructions (use Portal or CLI):

1. Provision a new Azure Functions app named "BackupFunctionsApp".
2. The app should be configured with the minimum required settings suitable for executing backup scripts.

Expected Deliverables:

- A documented step-by-step guide or script that creates the Azure Functions app.

Optional Deliverables:

- Screenshots demonstrating the successful setup.

Task 5: Change Azure Permissions

Objective: Temporarily grant developer George access to an Azure Functions App to analyze an issue and propose a method for temporary access management.

Instructions:

1. Choose an existing Resource, for example an Azure Functions App resource.
2. Grant George "Contributor" rights to this resource on a temporary basis.

Expected Deliverables:

- A PowerShell script or Azure CLI commands that grant the temporary permissions.
- Screenshots or command output confirming that the permissions have been successfully set.
- A possible plan outlining a method or process for providing temporary access to Azure resources. This should include consideration of access expiry, monitoring, and any necessary cleanup steps.

Task 6: Get Information from Azure, temporarely save them and upload to Blob Storage

Objective: List Azure resources, categorize them by type into a JSON file, and upload this file to Azure Blob Storage.

Instructions:

1. Retrieve the list of all Azure resources.
2. Group resources by their type and collate them into a JSON object.
3. Save this JSON data into a file named "resources_by_type.json".
4. Upload the file to a pre-existing blob container named "resource-backup".

Expected Deliverables:

- PowerShell code that generates the JSON file
- Upload the JSON file to Blob Storage.
- Screenshots or command output showing the execution and the file in Blob Storage.

Task 7: GitLab CI/CD Configuration File Analysis

Objective: Analyze a `.gitlab-ci.yml` file to understand its configurations.

Instructions:

1. Review the provided `example.gitlab-ci.yml` configuration file for a GitLab CI/CD pipeline.example.gitlab-ci.yml

2. Prepare a brief explanation covering:
    - Defined pipeline stages.
    - Purpose and sequence of jobs.
    - Key commands or scripts executed.

Expected Deliverable:

- A summary of the `.gitlab-ci.yml` file, detailing its functional elements.

Task 8: Set up a CI/CD Pipeline

Objective: Set up a Continuous Integration and Continuous Deployment (CI/CD) pipeline that triggers a script and uses an environment variable.

Instructions:

1. Create a CI/CD pipeline on your preferred platform (e.g., Azure DevOps, GitHub Actions).
2. The pipeline should be able to run a PowerShell script named "deploy_script.ps1".
3. It must include the `ENVIRONMENT` variable, defaulting to "dev", but changeable for different environments.

Expected Deliverables:

- A definition file (e.g., `azure-pipelines.yml`, `workflow.yml`) for the pipeline.
- Steps explaining how to execute the pipeline with different `ENVIRONMENT` variable values.

Optional Deliverables:

- Evidence of successful pipeline execution, such as screenshots or logs.