Cloud Computing Project 2: AWS Lambda File Sorting Application

---

Assignment: Build a Serverless File Sorting Application with AWS Lambda and S3

Objective:

The purpose of this assignment is to validate your understanding of AWS Lambda, S3 integration, and serverless architecture. You will create a web-based application that uploads an unsorted `.txt` file to an S3 bucket, triggers an AWS Lambda function to process and sort the file, and then retrieves the sorted file back to the web page for display.

Assignment Description:

You are tasked to develop a system that performs the following steps:

1. **Frontend (Web Page)**:
   - Create a simple web page that allows users to:
     - Upload a `.txt` file containing unsorted text (one value per line).
     - Display the sorted content of the file after processing.
2. **AWS S3 Buckets**:
   - Set up two S3 buckets:
     - **Input Bucket**: To store the uploaded `.txt` files.
     - **Output Bucket**: To store the processed and sorted `.srt` files.
3. **AWS Lambda Function**:
   - Write a Lambda function in **Python** or **JavaScript (Node.js)** that:
     - Is triggered when a file is uploaded to the **Input Bucket called SortIn**.
     - Reads the content of the uploaded `.txt` file.
     - Sorts the content of the file (line by line).
     - Writes the sorted content to a new file with the same name but with the extension `.srt`.
     - Saves the new file in the **Output Bucket called SortOut**.
4. **Frontend Integration**:
   - Modify the web page to:

    - Fetch the sorted `.srt` file from the **Output Bucket**.
    - Display the sorted content back on the web page.

Technical Requirements:

5. **Frontend**:
    - Use HTML, CSS, and JavaScript to create the web page.
    - Include an upload button to allow users to upload `.txt` files.
    - Use AWS SDK for JavaScript to interact with S3 and fetch the processed file.
6. **AWS Configuration**:
    - Create and configure two S3 buckets:
        - **Input Bucket (SortIn)**: For unsorted `.txt` files.
        - **Output Bucket (SortOut)**: For sorted `.srt` files.
    - Set appropriate IAM roles and policies to allow the Lambda function to read from the **Input Bucket** and write to the **Output Bucket**.
7. **AWS Lambda Function**:
    - Write the Lambda function in **Python** or **Node.js**.
    - Use the AWS SDK to interact with S3.
    - The Lambda function should:
        - Trigger upon file upload to the **Input Bucket**.
        - Read the file, sort its content, and save it with the `.srt` extension in the **Output Bucket**.
8. **Testing**:
    - Test the entire workflow to ensure:
        - The web page uploads the file to the **Input Bucket**.
        - The Lambda function processes the file and saves the sorted version in the **Output Bucket**.
        - The web page retrieves and displays the sorted content correctly.

Deliverables:

9. **Code Files**:
   - HTML, CSS, and JavaScript files for the web page.
   - AWS Lambda function code (Python or JavaScript).
   - Any additional configuration files (e.g., policy JSON files, AWS SDK setup).
10. **Documentation**:
    - A short report explaining:
      - How the system works.
      - AWS configuration screenshots
      - API documentation
      - Security implementation details
      - Steps to set up the AWS components (S3 buckets, Lambda function, IAM roles).
      - Any challenges faced and how they were resolved.
    - Screenshots of all required screens
11. **Demonstration**:
    - A video or screen shots demonstration showing:
      - Uploading an unsorted `.txt` file via the web page.
      - The Lambda function processing the file.
      - Displaying the sorted content back on the web page.

Evaluation Criteria:

Criteria	**Points**
Web Page Functionality	10
AWS S3 Bucket Configuration	20
AWS Lambda Function Implementation	30
Integration and Workflow	20
Documentation and Presentation	20
Total	**100**

Bonus (Optional):

- Add error handling for invalid file types or empty files.
- Display a progress bar or status indicator on the web page during file upload and processing.
- Use AWS CloudFormation or AWS CDK to automate the setup of the S3 buckets and Lambda function.

Instructions:

1. Submit all code files and documentation in a compressed `.zip` folder to Dropbox with the name of the file being your CNumber_P2.
2. Provide access to your AWS setup for evaluation purposes (e.g., IAM role credentials or temporary access keys in your documentation.
3. Ensure your Lambda function and S3 buckets are active during the evaluation period.
4. Leverage the work that you have done on Project 1 to ease your implementation.

Important Notes Checklist:

- All AWS credentials must be properly secured using environment variables
- Implement proper error handling throughout the application
- Follow security best practices
- Include (as many as possible) comments in your code
- Test thoroughly before submission

References:

https://aws.amazon.com/lambda/getting-started/

https://docs.aws.amazon.com/lambda/

https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html

https://chariotsolutions.com/blog/post/two-buckets-and-a-lambda-a-pattern-for-file-processing/

Good luck, and happy coding!