10/31/2024

CLOUD MANAGEMENT

MINI PROJECT PROPOSAL

JOVITA ANDREWS

1. Project Goals

The goal of this project is to develop a cloud-based text-to-speech converter application that can generate high-quality, natural-sounding audio from text input. By leveraging AWS services, the project aims to build a scalable, reliable, and serverless solution accessible to users needing audio versions of written text for accessibility, content creation, and educational purposes.

Objectives:

- Convert text to speech on demand with minimal latency.
- Support scalable architecture to handle varying loads.
- Enable easy access and interaction through a simple API endpoint.

2. Cloud Services/Tools & Additional Resources

Primary AWS Services:

- **AWS Lambda**: Serves as the serverless compute layer, triggering the text-to-speech process without provisioning or managing servers.
- **AWS Polly**: Processes the text input and generates speech output in a variety of natural-sounding voices and languages.

Optional AWS Services:

- **AWS S3**: For storing audio files generated by AWS Polly, enabling later access and reusability.
- **AWS API Gateway**: Provides a public-facing API endpoint to access the Lambda function for external users or applications.

Additional Resources:

- **IDE**: Visual Studio Code for development.
- **AWS SDK for Python (Boto3)**: For interfacing with AWS services programmatically within Lambda.

3. Description of Cloud Services for Solution

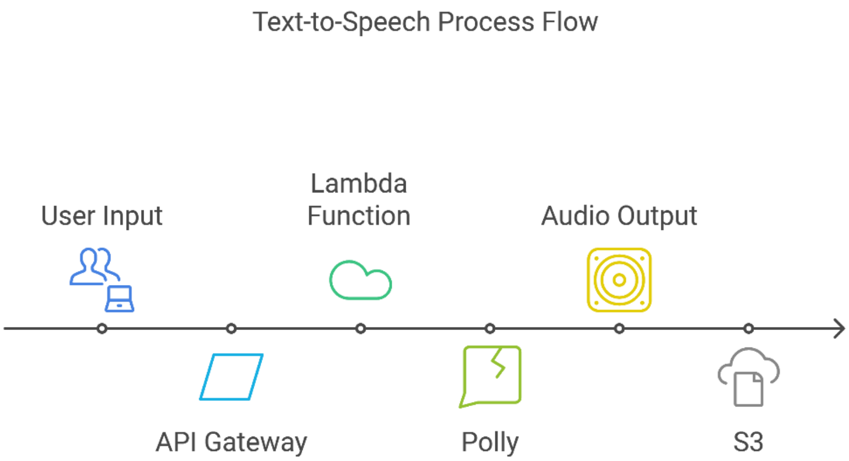
This section focuses on how each cloud service will specifically support our solution:

- **AWS Lambda**: Acts as the serverless execution environment, automatically scaling based on request load. When a text input is provided via an API Gateway endpoint or a direct trigger, Lambda will initiate the text processing flow. This choice of a serverless environment ensures that the application is cost-effective, as charges are based only on actual usage.
- **AWS Polly**: Converts the text input into an audio format. Polly's capability to use different languages and voices makes it ideal for providing a natural user experience. Polly also

supports customization, allowing us to choose from various voice tones and styles depending on user needs.

- **AWS S3** (if used): Stores audio files if users need to access them later. This storage solution enables persistent audio access and allows for batch processing of multiple requests if needed.
- **AWS API Gateway** (if used): Exposes the Lambda function as a RESTful API, allowing for a secure, managed interface for end users to submit text inputs.

4. Architecture Diagram



5. Detailed Architecture Design

1. **Request Flow**:
    - User inputs text through a web or mobile interface, sending it to the **API Gateway** endpoint.
    - **API Gateway** passes the request to the **Lambda** function, which initiates the text processing.
    - **Lambda** calls **AWS Polly** with the text input, receiving an audio stream in return.
    - The audio stream is either immediately returned to the user (through the API Gateway response) or saved in **S3** if longer-term access is needed.
2. **Scalability Considerations**:
    - **Lambda** handles request concurrency and scales automatically to meet demand.
    - **Polly** supports multiple requests and can be configured for different voices or languages as needed.

3. **Data Security**:
    - IAM roles and policies will secure access to AWS resources.
    - Enabling encryption for S3 storage (if used) will protect audio files

6. References / Sources

- **AWS Lambda Documentation:** AWS Lambda
- **AWS Polly Documentation:** AWS Polly
- **AWS S3 Documentation (if used):** AWS S3
- **AWS API Gateway Documentation (if used):** API Gateway