Serverless Text Processing Pipeline

CDK project that deploys a serverless API for text processing with AWS Lambda, API Gateway, CloudWatch, and DynamoDB.

Project Structure

```
text-processing-pipeline/
├── bin/
│   └── text-processing-pipeline.ts # CDK app entry
├── lib/
│   └── text-processing-pipeline-stack.ts # Infrastructure code
├── lambda/
│   └── text-processor/ # Lambda code
│   ├── index.ts
│   ├── package.json
├── README.md
└── package.json # CDK dependencies
```

🚀 Deployment

Prerequisites
- AWS account & CLI configured (`aws configure`)
- Node.js v16+ and npm
- AWS CDK installed (`npm install -g aws-cdk`)

1. Install dependencies

```
npm install
```

2. Build Lambda function

```
cd lambda/text-processor
npm install
npm run build  # Compiles TypeScript
cd ../..
```

3. Deploy infrastructure

```
npm run build
cdk deploy
```

Testing the API

Use the endpoint from CDK outputs:

```
curl -X POST \
  -H "Content-Type: text/plain" \
  -d "Your text here" \

https://[api-id].execute-api.[region].amazonaws.com/prod/process-text
```

Or

```
curl -X POST \
  -H "Content-Type: text/plain" \
 --data-binary "@yourTestfile.txt" \

https://[api-id].execute-api.[region].amazonaws.com/prod/process-text
```

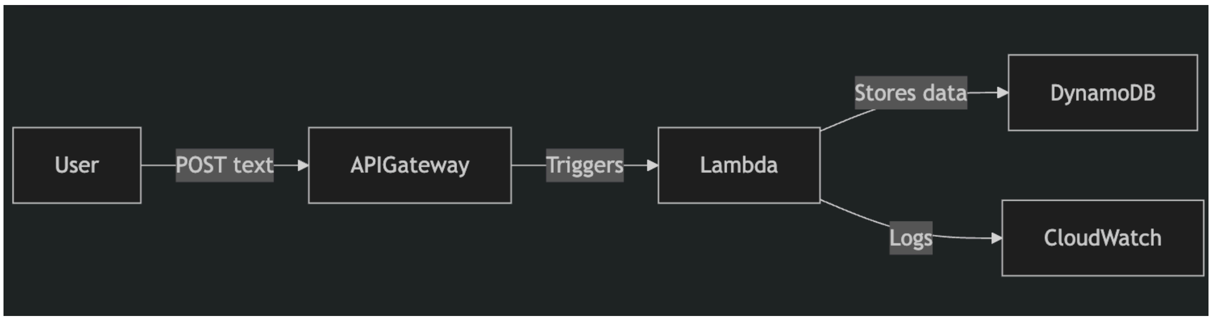
Example response:

```
{
  "message": "Text processed successfully",
  "processingId": "123456789",
  "wordCount": 3,
  "lineCount": 1
}
```

Useful Commands...

Command	Description
`cdk synth`	Generate CloudFormation template
`cdk deploy`	Deploy stack to AWS
`cdk destroy`	Remove all resources
`aws dynamodb scan --table-name TextDataTable`	View processed texts

Architecture



Security

- Least-privilege IAM roles auto-generated
- API Gateway uses HTTPS
- Set authorizationType: IAM in production

Components:

API Gateway (REST API endpoint)

Lambda (Text processing logic)

DynamoDB (Stores processed data)

CloudWatch (Monitoring/logging)

Cleanup

```
cdk destroy
```

Code...

https://github.com/Gmanojgupta/text-processing-pipeline

Server URL

https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text

Test local curl -X POST \

```
  -H "Content-Type: text/plain" \
  --d "Manoj" \
 https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text
```

Text Processing API - Test Cases

API Endpoint:
`https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text`

Overview

This document outlines various test scenarios for the **Text Processing API**, which accepts a `.txt` file via POST requests, processes the text, and stores some analytics data. The API enforces validation rules including file type, file size, and content presence.

Validation Rules

- **File type:** Must be `text/plain` (.txt files only)
- **File size:** Must be greater than 0 bytes and no more than 1MB
- **File content:** Must not be empty or contain only whitespace characters
- **Encoding:** Supports plain text or base64 encoded payloads (if configured)

Test Cases Using `curl`

1. No File Uploaded (Empty Body)

Description: Test the API with an empty request body.
Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  -d ""
```

Expected Response:

- HTTP Status: 400 Bad Request
- Body: `{ "message": "No file uploaded." }`

---

2. Invalid Content-Type

Description: Test the API with an incorrect Content-Type header.
Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: application/json" \
  -d "{\"some\":\"json\"}"
```

Expected Response:

- HTTP Status: 400 Bad Request
- Body: `{ "message": "Invalid file type. Only text/plain (.txt) files are allowed." }`

---

3. Empty File (0 Bytes)

Description: Test uploading an empty file.
Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  -d ""
```

Expected Response:

- HTTP Status: 400 Bad Request
- Body: `{ "message": "Uploaded file is empty." }`

---

4. File with Whitespace Only

Description: Test uploading a file that contains only whitespace characters.
Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  -d "    \n   \t  "
```

Expected Response:

- HTTP Status: 400 Bad Request
- Body: `{ "message": "Uploaded file contains no meaningful text (only whitespace)." }`

---

5. File Larger than 1MB

Description: Test uploading a file exceeding the 1MB limit.
Create a large test file (locally):

```
head -c 1048577 </dev/zero | tr '\0' 'a' > largefile.txt
```

Upload Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  --data-binary @largefile.txt
```

Expected Response:

- HTTP Status: 400 Bad Request
- Body: `{ "message": "File size exceeds the 1MB limit. Your file size: 1048577 bytes." }`

---

6. Valid Small Text File

Description: Test uploading a valid small text file.
Create a test file:

```
echo "Hello world! This is a test." > testfile.txt
```

Upload Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  --data-binary @testfile.txt
```

Expected Response:

- HTTP Status: 200 OK
- Body Example:

```
{
  "message": "Text processed successfully",
  "processingId": "1681234567890",
  "wordCount": 6,
  "lineCount": 1
}
```

---

7. Base64 Encoded File Upload (If Supported)

Description: Test uploading a base64-encoded text file (requires API Gateway to forward base64 flag).
Encode file to base64:

```
base64 testfile.txt > testfile.b64
```

Upload Command:

```
curl -X POST \
  https://hpp7ddb3ql.execute-api.ap-south-1.amazonaws.com/prod/process-text \
  -H "Content-Type: text/plain" \
  --data-binary @testfile.b64
```

Note: Ensure your API Gateway is configured to mark `isBase64Encoded = true` on the event.

Expected Response: Same as valid small text file.
