AWS Lambda Deployment Instructions

This document provides step-by-step instructions for deploying the two Lambda functions for your AI trend forecasting solution:

1. **Web Scraper Lambda** - Scrapes images, posts, likes, hashtags, and comments from social media platforms
2. **Trend Forecasting Lambda** - Analyzes images and metadata to identify fashion trends

Prerequisites

- AWS Account with appropriate permissions
- AWS CLI installed and configured (optional, for command-line deployment)
- Basic familiarity with AWS services (Lambda, S3, IAM, DynamoDB)

Step 1: Create S3 Buckets

First, create the S3 buckets that will store your images, metadata, and analysis results:

1. Sign in to the AWS Management Console
2. Navigate to the S3 service
3. Create three buckets with the following names (or your preferred names):
4. `fashion-trend-images` - For storing scraped images
5. `fashion-trend-metadata` - For storing post metadata
6. `fashion-trend-results` - For storing analysis results and reports

For each bucket: 1. Click "Create bucket" 2. Enter the bucket name 3. Select the region (ap-northeast-1 for Tokyo) 4. Keep default settings for most options 5. Ensure "Block all public access" is enabled for security 6. Click "Create bucket"

Step 2: Create DynamoDB Table

Create a DynamoDB table to store trend data:

1. Navigate to the DynamoDB service in the AWS Management Console
2. Click "Create table"
3. Enter table name: `fashion-trends`
4. Primary key: `trend_id` (String)

5. Sort key: `timestamp` (String)
6. Leave default settings for the rest
7. Click "Create"

Step 3: Create IAM Role for Lambda Functions

Create an IAM role with the necessary permissions:

1. Navigate to the IAM service
2. Click "Roles" in the left navigation
3. Click "Create role"
4. Select "AWS service" as the trusted entity
5. Select "Lambda" as the use case
6. Click "Next: Permissions"
7. Attach the following policies:
8. `AmazonS3FullAccess` (or create a custom policy with more restricted permissions)
9. `AmazonDynamoDBFullAccess` (or create a custom policy with more restricted permissions)
10. `CloudWatchLogsFullAccess` (for logging)
11. Click "Next: Tags" (add optional tags)
12. Click "Next: Review"
13. Name the role: `TrendForecastingLambdaRole`
14. Click "Create role"

Step 4: Deploy Web Scraper Lambda

1. Navigate to the Lambda service in the AWS Management Console
2. Click "Create function"
3. Select "Author from scratch"
4. Enter function name: `WebScraperLambda`
5. Runtime: Python 3.11
6. Architecture: x86_64
7. Execution role: Use the existing role `TrendForecastingLambdaRole`
8. Click "Create function"

After the function is created:

1. In the "Code" tab, delete the default code
2. Copy and paste the entire code from `webscraper_lambda.py`
3. Click "Deploy"

Configure the function settings:

1. Go to the "Configuration" tab
2. Click "General configuration" and edit:
3. Memory: 512 MB (increase if needed)
4. Timeout: 3 minutes (increase if needed for larger profiles)
5. Click "Save"

Add required layers for dependencies:

1. In the "Layers" section, click "Add a layer"
2. Select "Create layer" (or use an existing layer if you have one)
3. Create a layer with the following Python packages:
4. requests
5. beautifulsoup4
6. Pillow
7. (You can create a deployment package following AWS documentation)
8. Alternatively, use a public layer ARN that includes these packages

Step 5: Deploy Trend Forecasting Lambda

1. Navigate to the Lambda service
2. Click "Create function"
3. Select "Author from scratch"
4. Enter function name: `TrendForecastingLambda`
5. Runtime: Python 3.11
6. Architecture: x86_64
7. Execution role: Use the existing role `TrendForecastingLambdaRole`
8. Click "Create function"

After the function is created:

1. In the "Code" tab, delete the default code
2. Copy and paste the entire code from `trending_forecast_lambda.py`
3. Click "Deploy"

Configure the function settings:

1. Go to the "Configuration" tab
2. Click "General configuration" and edit:
3. Memory: 1024 MB (trend analysis requires more memory)
4. Timeout: 5 minutes
5. Click "Save"

Add required layers for dependencies (same as for the Web Scraper Lambda).

Step 6: Configure S3 Trigger for Trend Forecasting Lambda

Set up an S3 trigger to automatically invoke the Trend Forecasting Lambda when new files are uploaded:

1. In the Trend Forecasting Lambda function, go to the "Configuration" tab
2. Click on "Triggers" in the left navigation
3. Click "Add trigger"
4. Select "S3" as the trigger source
5. Select the bucket: `fashion-trend-images`
6. Event type: "All object create events"
7. Prefix: (leave empty to process all files, or specify a prefix if needed)
8. Suffix: .jpg (to only trigger on image files)
9. Click "Add"

Repeat the process to add another trigger for the metadata bucket:

1. Click "Add trigger" again
2. Select "S3" as the trigger source
3. Select the bucket: `fashion-trend-metadata`
4. Event type: "All object create events"
5. Prefix: (leave empty)
6. Suffix: _metadata.json
7. Click "Add"

Step 7: Test the Web Scraper Lambda

1. In the Web Scraper Lambda function, click the "Test" tab
2. Create a new test event with the following JSON:

```
{
 "queryStringParameters": {
  "url": "https://www.instagram.com/yusaku2020/"
 }
}
```

1. Name the test event: "TestInstagramScrape"
2. Click "Save"

3. Click "Test" to run the function
4. Check the execution results and logs
5. Verify that images and metadata are being stored in the S3 buckets

Step 8: Test the Trend Forecasting Lambda

The Trend Forecasting Lambda should be automatically triggered when new files are uploaded to the S3 buckets. To test it manually:

1. In the Trend Forecasting Lambda function, click the "Test" tab
2. Create a new test event with the following JSON:

```
{
 "Records": [
  {
   "eventSource": "aws:s3",
   "s3": {
    "bucket": {
     "name": "fashion-trend-images"
    },
    "object": {
     "key": "path/to/your/test/image.jpg"
    }
   }
  }
 ]
}
```

1. Replace "path/to/your/test/image.jpg" with the actual path of an image in your S3 bucket
2. Name the test event: "TestTrendAnalysis"
3. Click "Save"
4. Click "Test" to run the function
5. Check the execution results and logs
6. Verify that trend analysis results are being stored in the results bucket

Step 9: Set Up Scheduled Trend Reports (Optional)

To generate periodic trend reports:

1. Navigate to Amazon EventBridge (CloudWatch Events)
2. Click "Create rule"
3. Enter a name and description

4. For "Define pattern", select "Schedule"
5. Set up a schedule expression (e.g., "cron(0 0 * * ? *)" for daily at midnight)
6. Click "Add target"
7. Select "Lambda function" as the target
8. Select your "TrendForecastingLambda" function
9. Configure input:

```
{
 "detail-type": "Scheduled Event",
 "source": "aws.events"
}
```

1. Click "Create"

Step 10: Monitoring and Maintenance

Set up monitoring for your Lambda functions:

1. Navigate to CloudWatch
2. Set up alarms for Lambda errors and duration
3. Create a dashboard to monitor:
4. Lambda invocations and errors
5. S3 bucket metrics
6. DynamoDB table metrics

Regularly check the CloudWatch logs for both Lambda functions to ensure they're running correctly.

Customization Notes

1. **S3 Bucket Names**: If you use different bucket names, update the variables at the top of both Lambda function codes: `python S3_BUCKET_IMAGES = 'your-images-bucket-name' S3_BUCKET_METADATA = 'your-metadata-bucket-name' S3_BUCKET_RESULTS = 'your-results-bucket-name'`

2. **AWS Region**: If you're not using the Tokyo region, update the region variable: `python AWS_REGION = 'your-preferred-region'`

3. **DynamoDB Table Name**: If you use a different table name, update the variable: `python DYNAMODB_TABLE = 'your-table-name'`

4. **Web Scraping Enhancements**: The web scraper uses a simplified approach. For production use, consider:

5. Using official APIs where available
6. Implementing more robust error handling
7. Adding rate limiting to avoid being blocked

8. Enhancing the parsing logic for different profile layouts

9. **Trend Analysis Enhancements**: The trend analysis uses simplified algorithms. For production use, consider:

10. Integrating with Amazon Rekognition for better image analysis
11. Using Amazon Comprehend for more sophisticated text analysis
12. Implementing more advanced forecasting algorithms
13. Training custom ML models for specific fashion attributes

Troubleshooting

1. **Lambda Timeout**: If your functions time out, increase the timeout setting in the Lambda configuration.

2. **Memory Issues**: If you encounter memory errors, increase the allocated memory.

3. **Permission Errors**: Ensure your IAM role has all necessary permissions for S3, DynamoDB, and CloudWatch.

4. **Dependency Issues**: Make sure all required Python packages are included in your Lambda layers.

5. **S3 Trigger Not Working**: Verify the bucket notification configuration and ensure the Lambda function has permission to be invoked by S3.