praxis.md

# Praxis

## Simple Lambda Examples:

1. Logge dich in der AWS Console (Techstarter Sandbox) ein
2. Navigiere zum Lambda Service
3. Erstelle eine neue Function und nenne sie `MyLambdaFunction`:
   - Runtime: NodeJs 18.x
   - Architecture: x86_64
   - Permissions: Create a new role with basic Lambda Permissions
4. Füge folgenden Code hinzu und drücke auf `Deploy`:

```
export const handler = async (event) => {
  console.log(event);
};
```

5. Erstelle ein neues Test Scenario:
   - Klicke auf `Test`
   - Füge einen Namen hinzu: `MyLambdaTest`
   - Klicke auf `Save`
6. Klicke auf `Test` und schaue dir die Logs in Cloudwatch an
   - Klicke auf den `Monitor` Reiter und dann auf `View in Cloudwatch Logs`
   - Wähle den Log Stream aus
   - Stelle sicher, dass das Test Event richtig gelogged wurde

```
2023-10-08T17:25:52.446Z 40f74b6e-794d-4e5c-b3e5-18fa23852295 INFO { key1: 'value1', key2: 'value2', key3: 'v
```

7. Finde die Lambda Execution Role heraus und speichere sie

## Basic S3 Setup

1. Navigiere zum S3 Service
2. Erstelle einen neune Bucket (hier werden die Bilder hochgeladen und unsere Lambda function getriggered):
   - Wähle einen Namen aus (zum Beispiel: `my-lambda-bucket-8245109834`)
   - Region: gleiche Region wie die Lambda Function (eu-central-1)
   - Object Ownership: ACLs disabled
   - Block all public access
   - Versioning: disabled
   - Encryption: SSE-S3
3. Erstelle eine neue Policy
   - Erstelle eine neue Policy mit dem Namen `LambdaS3SourceAccessPolicy`

- Entweder durch Visual Editor oder JSON
- Füge die ARN deines S3 Buckets hinzu
- WICHTIG: nicht das `/*` am Ende der Bucket ARN vergessen!

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "DEINE_BUCKET_ARN/*"
        }
    ]
}
```

4. Füge die Policy der Lambda Execution Role hinzu
    - Navigiere zur Rolle und klicke auf Attach Role

## S3 Notifications

1. Navigiere zu deinem Bucket und klicke auf `Properties`
2. Unter dem Punkt `Event Notifications`, klicke auf `Create Event Notification`:
    - Name: `ThumbnailCreationNotification`
    - Prefix: Frei lassen
    - Suffix: .jpg
    - Event Types: `s3:ObjectCreated:Put`, s3:ObjectCreated:Post`
    - Destination: Wähle deine Lambda Function aus
3. Lade eine jpg Datei in dem Bucket hoch und stelle sicher, dass die Lambda Funktion ausgeführt wurde

## Lambda Logik

1. Ändere den Lambda Code ab um zu testen, dass die S3 Daten richtig übergeben werden:

```
export const handler = async (event) => {
    const records = event.Records;
    console.log(records);
    for (let i = 0; i < records.length; i++) {
        const record = records[i];

        console.log(record)

        const s3 = record.s3;
        console.log(s3);

        const bucketArn = s3.bucket.arn;

        console.log(bucketArn);

        const objectKey = s3.object.key;

        console.log(objectKey);
    }
};
```

2. Teste den Upload einer JPG Datei erneut und schaue in die CloudWatch Logs:
3. Ändere die Lambda Funktion erneut ab um zu testen ob Lambda auf den S3 Bucket zugreifen kann:

```
import { S3Client, GetObjectCommand, PutObjectCommand } from '@aws-sdk/client-s3';

import { Readable } from 'stream';

// create S3 client
const s3 = new S3Client({region: 'eu-central-1'});

export const handler = async (event) => {
    const records = event.Records;
    console.log(records);
    for (let i = 0; i < records.length; i++) {
        const record = records[i];

        console.log(record)

        const s3Obj = record.s3;
        console.log(s3Obj);

        const bucketName = s3Obj.bucket.name;

        console.log(bucketName);

        const objectKey = s3Obj.object.key;

        console.log(objectKey);

        // Get the image from the source bucket. GetObjectCommand returns a stream.
        try {
            const params = {
                Bucket: bucketName,
                Key: objectKey
            };
            var response = await s3.send(new GetObjectCommand(params));
            console.log(response);
            var stream = response.Body;

            // Convert stream to buffer to pass to sharp resize function.
            if (stream instanceof Readable) {
                var content_buffer = Buffer.concat(await stream.toArray());
            } else {
                throw new Error('Unknown object stream type');
            }

        } catch (error) {
            console.log(error);
            return;
        }
    }
};
```

## Sharp Library als Lambda Layer

1. Lese dir folgendes Repo durch Link
2. Lade die Layer Zip Datei herunter

```
wget https://github.com/Umkus/lambda-layer-sharp/releases/download/0.32.1/sharp-layer.zip
```

3. Uploade die Zip Datei als neuen Lambda Layer:
    - Stelle sicher, dass deine aws cli configuriert ist `aws s3 ls`
    - Füge den Layer hinzu

```
aws lambda publish-layer-version \
    --layer-name sharp \
    --description "Sharp layer" \
```

```
    --license-info "Apache License 2.0" \
    --zip-file fileb://sharp-layer.zip \
    --compatible-runtimes nodejs14.x nodejs16.x nodejs18.x \
    --compatible-architectures x86_64 arm64 \
    --profile techstarter # !!!Diese Zeile nur hinzufügen, wenn du dein aws profile auch als techstarter hinz
```

4. Im output des Commands wird dir die ARN des Layers angezeigt, kopiere diese!
5. Zurück im AWS UI, klicke auf `Layer` unter dem Namen der Lambda Function
    - `Add Layer`
    - `Specify an ARN`
    - Füge die ARN ein und am Ende `:1`

## Thumbnail Bucket

1. Erstelle einen weiteren S3 Bucket:
    - Wähle einen Namen aus (zum Beispiel: `my-lambda-thumbnail-bucket-8245109834`)
    - Region: gleiche Region wie die Lambda Function (eu-central-1)
    - Object Ownership: ACLs disabled
    - Block all public access
    - Versioning: disabled
    - Encryption: SSE-S3
2. Erstelle eine Policy und füge sie der Lambda Execution Role hinzu:
    - WICHTIG: nicht das `/*` am Ende der Bucket ARN vergessen!

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "DEINE_THUMBNAIL_BUCKET_ARN/*"
        }
    ]
}
```

## Finaler Code

```
import { S3Client, GetObjectCommand, PutObjectCommand } from '@aws-sdk/client-s3';

import { Readable } from 'stream';

import sharp from 'sharp';

const THUMBNAIL_BUCKET = "DEIN_THUMBNAIL_BUCKET_NAME";
const WIDTH = 200;

// create S3 client
const s3 = new S3Client({region: 'eu-central-1'});

export const handler = async (event) => {
    const records = event.Records;
    console.log(records);
    for (let i = 0; i < records.length; i++) {
        const record = records[i];

        console.log(record)
```

```
        const s3Obj = record.s3;
        console.log(s3Obj);

        const bucketName = s3Obj.bucket.name;

        console.log(bucketName);

        const objectKey = s3Obj.object.key;

        console.log(objectKey);

        // Get the image from the source bucket. GetObjectCommand returns a stream.
        try {
            const params = {
                Bucket: bucketName,
                Key: objectKey
            };
            var response = await s3.send(new GetObjectCommand(params));
            console.log(response);
            var stream = response.Body;

            // Convert stream to buffer to pass to sharp resize function.
            if (stream instanceof Readable) {
                var content_buffer = Buffer.concat(await stream.toArray());
            } else {
                throw new Error('Unknown object stream type');
            }

        } catch (error) {
            console.log(error);
            return;
        }

        // Use the sharp module to resize the image and save in a buffer.
        try {
            var output_buffer = await sharp(content_buffer).resize(WIDTH).toBuffer();

        } catch (error) {
            console.log(error);
            return;
        }

        // Upload the thumbnail image to the destination bucket
        try {
            const destparams = {
                Bucket: THUMBNAIL_BUCKET,
                Key: objectKey,
                Body: output_buffer,
                ContentType: "image"
            };

            const putResult = await s3.send(new PutObjectCommand(destparams));

        } catch (error) {
            console.log(error);
            return;
        }

        console.log('Successfully resized ', objectKey);
    }

};
```