Activity4 — Exercise: Image Processing Pipeline (Student Assignment)

Objective 🎯

In this activity, you will **independently build a complete serverless image processing pipeline** using Serverless Framework v4 on AWS.

By the end of this assignment you should be able to:

- Write a full `serverless.yml` that wires together AWS Lambda, S3, DynamoDB, and SNS.
- Configure IAM roles and policies for least privilege.
- Deploy and verify a multi-step workflow where:
  1. A user uploads an image to S3.
  2. A Lambda resizes it, stores metadata in DynamoDB, and publishes an SNS notification.
  3. Another Lambda consumes SNS and logs structured analytics data into an S3 bucket.

This exercise is **evaluated** — you must implement everything end-to-end and submit your work.

Prerequisites

Before starting this activity, make sure you have:

1. Completed **Activity1** to **Activity3**.
2. Configured AWS CLI with the profile `serverless-lab`:

```Shell
aws configure --profile serverless-lab
```

3. Logged in to Serverless Framework Dashboard:

```Shell
serverless login
```

⚠️ **Note:** AWS CLI and Serverless Dashboard login are **one-time setup per activity**. You must do it at the start of each new activity.

Plugins Required

This activity requires the following Serverless Framework plugin:

- `serverless-python-requirements` → to bundle Python dependencies (like Pillow for image resizing).

Installation Steps

Run this in your `activity4-image-pipeline/` project folder:

```Shell
npm init -y   # if package.json does not exist
npm install --save-dev serverless-python-requirements
```

Enable it in your `serverless.yml`:

```None
plugins:
  - serverless-python-requirements

custom:
  pythonRequirements:
    dockerizePip: true
    zip: true
    slim: true
```

This ensures dependencies listed in `requirements.txt` (like `boto3`, `pillow`) are packaged inside your Lambda zip correctly.

Problem Statement 📝

We want to build a **Serverless Image Processing Pipeline** that automates:

1. **Uploads** → User uploads image to a private S3 bucket.
2. **Processing** → A Lambda function resizes the image, stores metadata in DynamoDB, and publishes result to SNS.
3. **Analytics** → Another Lambda subscribes to SNS and stores structured logs into a separate Analytics S3 bucket.

The flow:

Start → Upload → Process → Store Metadata → Publish → Log Analytics → End.

Folder Structure 📂

Your workspace should look like this:

```
None
/home/labDirectory/activity4-image-pipeline
├── data.json
├── handlers
|   ├── analytics_logger.py
|   └── image_processor.py
├── instructor_policy.json
├── problem_statement.txt
├── requirements.txt
├── LEARN.jpg
└── serverless.yml
```

Lambda Functions (Provided) 🐍

You are given two Lambda functions inside `handlers/`.

🔹 `image_processor.py` — Image Processor Lambda

- Triggered automatically when a new object is uploaded to the Uploads bucket.
- Steps:
  1. Downloads image from Uploads bucket.
  2. Creates a 128x128 thumbnail using Pillow (`PIL`).
  3. Uploads thumbnail to the Thumbnails bucket (public-read).
  4. Stores metadata (jobId, keys, status, timestamp) in DynamoDB table.
  5. Publishes success/failure notification to SNS Topic.
- Uses **logging** for every step.

🔹 `analytics_logger.py` — Analytics Logger Lambda

- Subscribed to the SNS Topic.
- Steps:
  1. Receives notification from SNS.
  2. Formats into structured JSON log.
  3. Saves log to Analytics S3 bucket.
- This bucket acts like a **data lake**.

🔹 requirements.txt

```
None
boto3>=1.26.0
Pillow>=9.0.0
```

Your Task (Student Assignment) 🛠️

You must write the **`serverless.yml`** file that ties everything together:

1. **Buckets**
   - `UploadsBucket` (private)
   - `ThumbnailsBucket` (public-read)
   - `AnalyticsBucket` (private)
2. **DynamoDB Table**
   - `JobsTable` with primary key `jobId` (string).
3. **SNS Topic**
   - `ImageEventsTopic`.
4. **Functions**
   - `imageProcessor` triggered by S3 uploads bucket.
   - `analyticsLogger` subscribed to SNS topic.
   - Inject environment variables (bucket names, table, topic ARN).
5. **IAM Role**
   - CloudWatch Logs permissions.
   - S3 read/write (scoped to relevant buckets).
   - DynamoDB write for `JobsTable`.

- SNS publish (for processor).
- SNS subscribe/invoke (for logger).

6. **Outputs**

    - All resource names (buckets, table, topic ARN).

📌 **Note:** The folder structure is already provided in the **clab directory workspace**. You only need to **fill in** the following files:

- `serverless.yml` → define resources, functions, IAM, outputs.
- `data.json` → fill with required values after deployment.

Self-Testing Before Submission ✅

Before submission, run the following commands in order to fully test your stack.

Shell
```shell
# 1) Deploy stack with verbose output (shows detailed CFN events)
sls deploy --stage dev --region ap-south-1 --aws-profile serverless-lab
--verbose

# 2) Get deployed resource info (bucket names, table, topic ARN)
sls info --stage dev --region ap-south-1 --aws-profile serverless-lab --verbose

aws cloudformation describe-stack-resources --stack-name
activity4-image-pipeline-dev --region ap-south-1 --profile serverless-lab
aws cloudformation describe-stacks --stack-name activity4-image-pipeline-dev
--region ap-south-1 --profile serverless-lab

# 3) Upload a test image(kept in CLAB workspace) to the Uploads bucket (replace
<uploads-bucket-name>)
aws s3 cp LEARN.jpg s3://<uploads-bucket-name>/ --profile serverless-lab

# 4) Check DynamoDB table for job entry (replace <jobs-table-name>)
aws dynamodb scan --table-name <jobs-table-name> --profile serverless-lab
```

```
# 5) Tail logs of Image Processor Lambda (see resize, DynamoDB, SNS publish
actions)
sls logs -f imageProcessor --stage dev --tail
sls logs -f imageProcessor --stage dev --startTime 5h
sls logs -f imageProcessor --stage dev --startTime 2d

# 6) Tail logs of Analytics Logger Lambda (see SNS → S3 log writes)
sls logs -f analyticsLogger --stage dev --tail
sls logs -f analyticsLogger --stage dev --startTime 5h
sls logs -f analyticsLogger --stage dev --startTime 2d

# 7) (Optional) Receive a message directly from SQS queue (debug SNS→SQS if
added)
aws sqs receive-message --queue-url <queue-url> --profile serverless-lab

# 8) (Optional) Remove stack to avoid charges
sls remove --stage dev --region ap-south-1 --aws-profile serverless-lab
```

✅ If the test image appears in the **Thumbnails bucket**, a record shows up in **DynamoDB**, and a log JSON file is written in the **Analytics bucket**, then your stack is working correctly.

Evaluation 📤

This activity **will be tested using an automated script**.

You must:

1. Deploy your stack successfully (`sls deploy --stage dev`).
2. Create a temporary **Instructor IAM User** with the policy below.
3. Provide credentials and stack details in `data.json`.

🔹 **Instructor Policy (Save as `instructor_policy.json`)**

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "s3:*",
        "lambda:*",
        "iam:PassRole",
        "iam:CreateRole",
        "iam:GetRole",
        "iam:DeleteRole",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:DeleteRolePolicy",
        "iam:PutRolePolicy",
        "apigateway:*",
        "dynamodb:*",
        "sqs:*",
        "sns:*",
        "logs:*",
        "iam:ListRolePolicies",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListRoles",
        "iam:ListPolicies"
      ],
      "Resource": "*"
    }
  ]
}
```

⚠️ Note: This user must be **deleted after grading**.

🔹 data.json Format

```json
{
  "instructor_iam_username": "<IAM > Users (username you created for
instructor)>",
  "instructor_access_key_id": "<IAM > Security credentials (Access key)>",
  "instructor_secret_access_key": "<IAM > Security credentials (Secret key,
only shown at creation)>",
  "aws_region": "<Region where you deployed stack (e.g., ap-south-1)>",
  "stack_name": "<From 'sls info' → stack name (e.g., activity4-app-dev)>",
  "uploads_bucket": "<From 'sls info' → S3 Uploads bucket name>",
  "thumbnails_bucket": "<From 'sls info' → S3 Thumbnails bucket name>",
  "analytics_bucket": "<From 'sls info' → S3 Analytics bucket name>",
  "jobs_table": "<From 'sls info' → DynamoDB table name>",
  "sns_topic_arn": "<From 'sls info' or CloudFormation Outputs → SNS Topic
ARN>"
}
```

Submission Checklist ✅

- `serverless.yml` complete and correct.
- `instructor_policy.json` applied to Instructor IAM user.
- `data.json` filled.
- Stack deploys, processes image uploads, produces thumbnails, stores metadata, publishes SNS, and writes analytics logs.
- The stack is running live in AWS.

Final Note 📌

This assignment **is evaluated**.
Be precise with YAML, environment variables, and IAM.
After the evaluation is done, ensure resources are cleaned up (sls remove) unless instructed otherwise.

🏆 Congratulations: You’ve completed the Lab

═══ END OF DOCUMENT ═══