# Lab: Deploying Flask Microservices with AWS Lambda (Serverless) on Windows

1. **Prerequisites**

- An AWS account.
- Python 3.9+ installed (e.g., via Anaconda, python.org).
- Docker Desktop installed (needed for local SAM builds).
- Windows PowerShell or Command Prompt.

2. **Install AWS CLI**

1. Download and run the **AWS CLI Windows Installer**: https://awscli.amazonaws.com/AWSCLIV2.msi
2. In **Command Prompt**, verify the installation:

```
aws --version
```

Output should be like:

```
aws-cli/2.x.x Python/3.x Windows/...
```

3. Configure your AWS credentials:

```
aws configure
```

Enter:

- *AWS Access Key ID*
- *AWS Secret Access Key*
- *Region (e.g., us-east-1)*
- *Output format (json)*

3. **Install AWS SAM CLI**

1. Download the **AWS SAM CLI Windows Installer**: https://github.com/aws/aws-sam-cli/releases/latest
   Download the .msi file (AWS_SAM_CLI_64_PY3.msi).

2. Run the installer and complete the installation.
3. Verify the SAM CLI is working:

```
sam --version
```

Output should be like:

```
SAM CLI, version 1.x.x
```

**4. Create Project Structure**

In **PowerShell** or **Command Prompt**:

```
mkdir C:\Users\YourUser\microservices-lab
```

```
cd C:\Users\YourUser\microservices-lab
```

Create these folders:

```
mkdir user-service order-service product-service
```

**5. Create Flask Microservices**

For each microservice, create an app.py file and a requirements.txt file.

**user-service/app.py**

```
from flask import Flask, jsonify
import awsgi
app = Flask(__name__)
@app.route('/users')
def get_users():
    return jsonify({"users": ["Alice", "Bob", "Charlie"]})
def lambda_handler(event, context):
    return awsgi.response(app, event, context)
```

**user-service/requirements.txt**

```
Flask==2.2.5
awsgi==0.0.6
```

**order-service/app.py**

```
from flask import Flask, jsonify
import awsgi
app = Flask(__name__)
@app.route('/orders')
def get_orders():
```

```
    return jsonify({"orders": ["Order1", "Order2", "Order3"]})
def lambda_handler(event, context):
    return awsgi.response(app, event, context)
```

**order-service/requirements.txt**

```
Flask==2.2.5
awsgi==0.0.6
```

**product-service/app.py**

```
from flask import Flask, jsonify
import awsgi
app = Flask(__name__)
@app.route('/products')
def get_products():
    return jsonify({"products": ["Product1", "Product2",
"Product3"]})
def lambda_handler(event, context):
    return awsgi.response(app, event, context)
```

**product-service/requirements.txt**

```
Flask==2.2.5
awsgi==0.0.6
```

### 6. Create template.yaml (SAM Template)

Create C:\Users\YourUser\microservices-lab\template.yaml:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Deploy 3 Flask microservices to Lambda

Globals:
  Function:
    Timeout: 10
    Runtime: python3.9
    MemorySize: 128
Resources:
  UserServiceFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: user-service/
      Handler: app.lambda_handler
```

```
      Events:
        UserApi:
          Type: Api
          Properties:
            Path: /users
            Method: GET
  OrderServiceFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: order-service/
      Handler: app.lambda_handler
      Events:
        OrderApi:
          Type: Api
          Properties:
            Path: /orders
            Method: GET
  ProductServiceFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: product-service/
      Handler: app.lambda_handler
      Events:
        ProductApi:
          Type: Api
          Properties:
            Path: /products
            Method: GET
```

7. **Build the Project**

In **Command Prompt** (from the microservices-lab directory):

```
sam build
```

SAM will:

- Create .aws-sam/ folder
- Package your Flask apps and dependencies

8. **Deploy to AWS**

Run:

```
sam deploy --guided
```

Follow the prompts:

- Stack Name: microservices-stack

- AWS Region: us-east-1
- Accept the default settings
- Confirm

SAM will:

- Create Lambda functions for each microservice
- Create an API Gateway
- Output the API endpoint URL (like https://xyz123.execute-api.us-east-1.amazonaws.com/Prod)

**9. Test the Endpoints**

Copy the **API endpoint** URL from the deploy output.

Test in the browser or with curl:

*curl https://xyz123.execute-api.us-east-1.amazonaws.com/Prod/users*

*curl https://xyz123.execute-api.us-east-1.amazonaws.com/Prod/orders*

*curl https://xyz123.execute-api.us-east-1.amazonaws.com/Prod/products*

**You should get JSON responses like:**

{"users": ["Alice", "Bob", "Charlie"]}

**Local Testing (Optional)**

If you want to test locally before deploying:
Start local API gateway:

sam local start-api

Visit:

- http://127.0.0.1:3000/users
- http://127.0.0.1:3000/orders
- http://127.0.0.1:3000/products

sam local uses Docker to simulate Lambda.

**Wrap-up and Summary**

You now have:

- 3 Flask microservices
- Deployed as **serverless Lambda functions**
- Managed via a single template.yaml
- Fronted by a single API Gateway

**Cleanup (Optional)**

When done, to avoid charges:

```
aws cloudformation delete-stack --stack-name microservices-stack
```