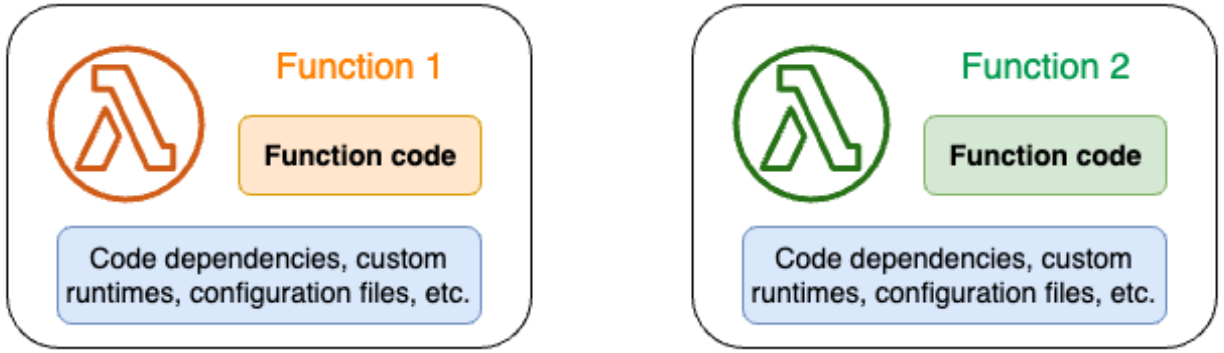
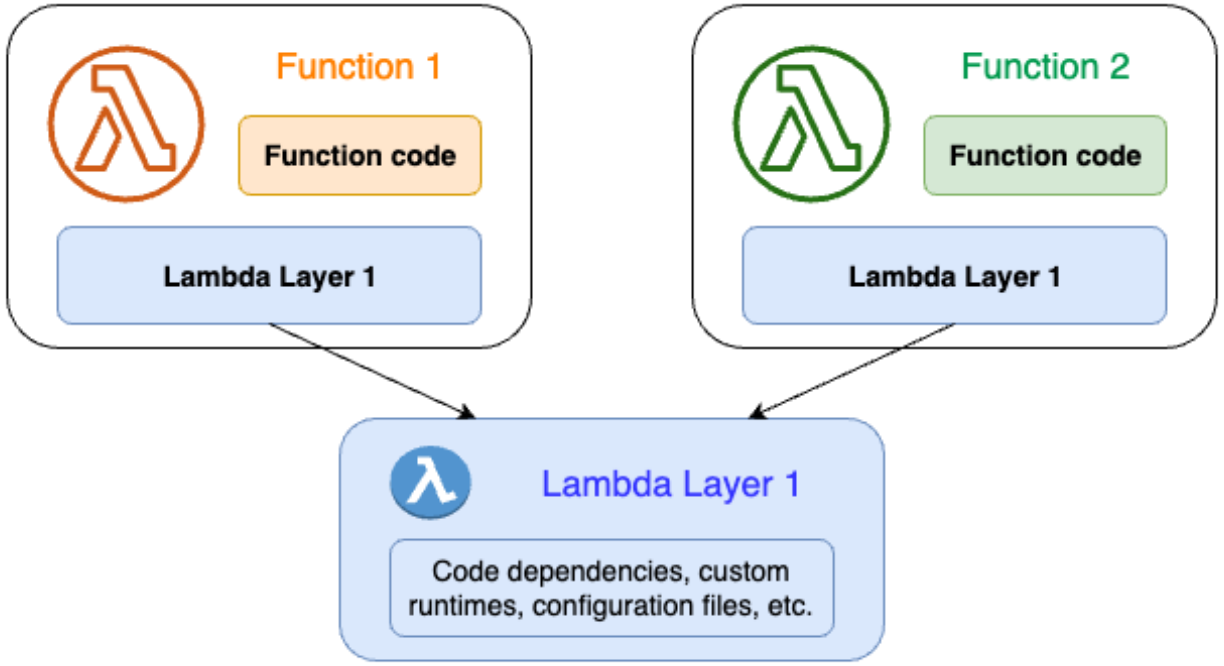
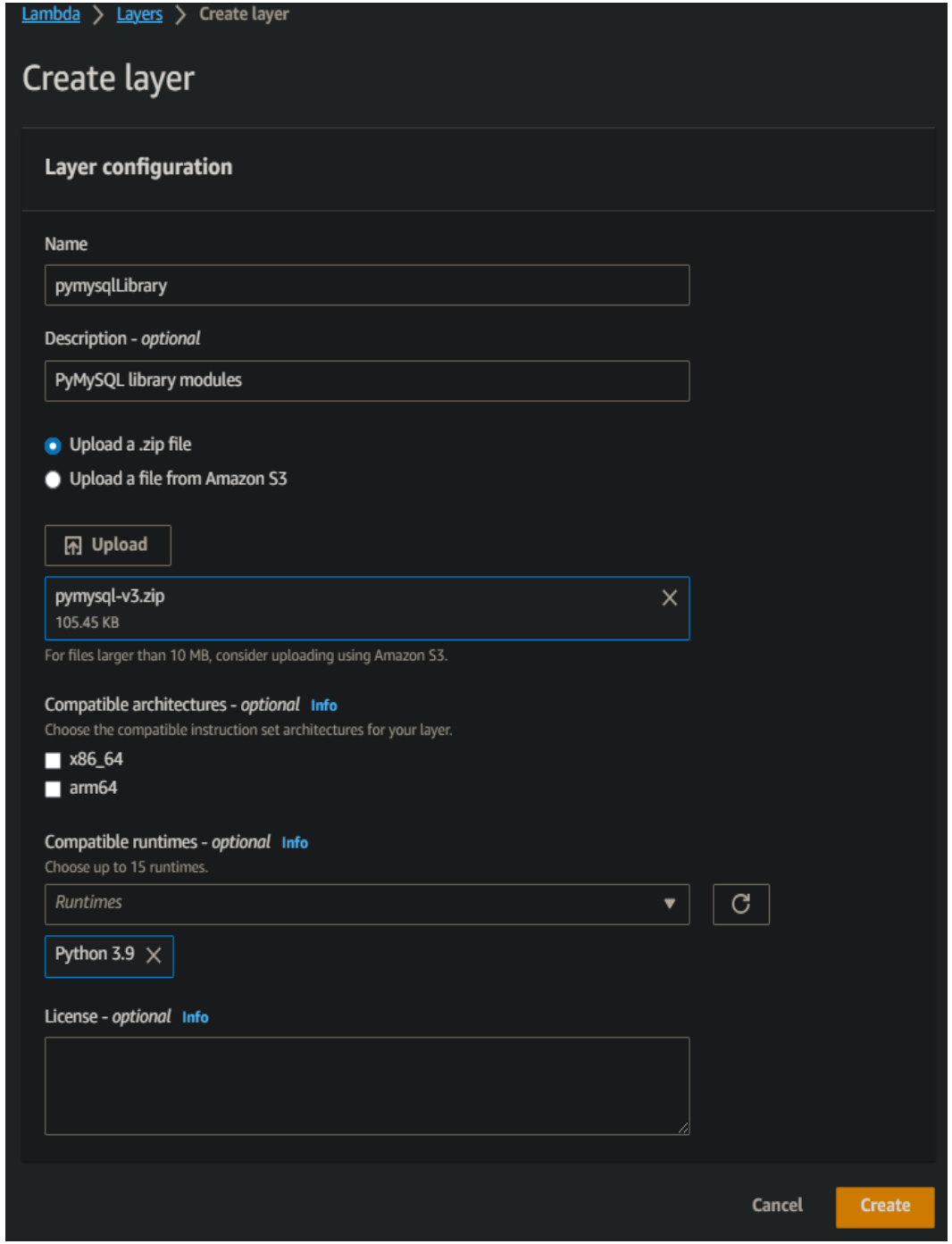
Lambda function components: Without layers



Lambda function components: With layers



- You can configure your Lambda function to use additional code and content in the form of layers. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. With layers, you can use libraries in your function without needing to include them in your deployment package.
- If your Lambda function includes layers, Lambda extracts the layer contents into the /opt directory in the function execution environment. Lambda extracts the layers in the order that you added them to the function. If the same file appears in multiple layers, the function uses the version in the last extracted layer.
- You can use the **Merge earlier** and **Merge later** buttons to adjust the merge order of the layers.
- You can update the layer version by entering a new version in **Layer version**.
- You can update the layer version that your functions use by selecting the functions you want to update from **Functions using this version**. Layer version updates will apply to the functions listed on this page.



A Lambda layer is a .zip file archive that contains supplementary code or data. Layers usually contain library dependencies, a custom runtime, or configuration files.

This section explains how to create and delete layers in Lambda. For more conceptual information about layers and why you might consider using them, see Working with Lambda layers.

After you've packaged your layer content, the next step is to create the layer in Lambda. This section demonstrates how to create and delete layers using the Lambda console or the Lambda API only. To create a layer using AWS CloudFormation, see Using AWS CloudFormation with layers. To create a layer using the AWS Serverless Application Model (AWS SAM), see Using AWS SAM with layers.

- Lambda supports multiple languages through the use of runtimes. A runtime provides a language-specific environment that relays invocation events, context information, and responses between Lambda and the function. You can use runtimes that Lambda provides, or build your own.

- Each major programming language release has a separate runtime, with a unique runtime identifier, such as python3.10 or nodejs18.x. To configure a function to use a new major language version, you need to change the runtime identifier. Since AWS Lambda cannot guarantee backward compatibility between major versions, this is a customer-driven operation.

Lambda > Layers > Add layer

Add layer

Function runtime settings

Runtime	Architecture
Python 3.9	x86_64

Choose a layer

Layer source Info

Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) version. You can also create a new layer.

- AWS layers — Choose a layer from a list of layers provided by AWS.
- Custom layers — Choose a layer from a list of layers created by your AWS account or organization.
- Specify an ARN — Specify a layer by pr ARN.

Custom layers

Layers created by your AWS account or organization that are compatible with your function's runtime.

pymysqlLibrary

Version

7

Canc

Code | Test | Monitor | Configuration | Aliases | Versions

Code source Info

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

salesAnalysisReport

lambda_function.py

lambda_function | Environment Var

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

Code properties Info

Package size	SHA256 hash	Last modified
299.0 byte	HAPq9EReJVEC5gLavtc/gyd5vZtd9elUGF932t0jBxY=	September 6, 20

Runtime settings Info

Runtime	Handler Info	Architecture Info
Python 3.9	lambda_function.lambda_handler	x86_64

Runtime management configuration

Layers Info

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Vers
1	pymysqlLibrary	7	python3.9	-	arn:a

Runtime settings Info

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9

New runtime available

A new runtime is available for your function's language: Python 3.11

Handler Info

salesAnalysisReportDataExtractor.lambda_handler

Architecture Info
Choose the instruction set architecture you want for your function code.

- x86_64
- arm64

You can change either the function's runtime or the instruction set architecture in one update. To update both, you must repeat the update process.

Connected layers

You can configure layers to include runtimes and the instruction set architectures they support. If you are updating the runtime or instruction set architecture of your function, afterward test your function to verify your invocation is successful.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures
1	pymysqlLibrary	7	python3.9	-

Runtime settings Info

Runtime
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9

New runtime available

A new runtime is available for your function's language: Python 3.11

Handler Info

salesAnalysisReportDataExtractor.opener_db

Architecture Info
Choose the instruction set architecture you want for your function code.

- x86_64
- arm64

Connected layers

You can configure layers to include runtimes and the instruction set architectures they support. If you are updating the runtime or instruction set architecture of your function, afterward test your function to verify your invocation is successful.

Merge order	Name	Layer version	Compatible runtimes	Compatible architectures
1	pymysqlLibrary	7	python3.9	-

Cancel Save

Code source Info

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

salesAnalysisReport

salesAnalysisReportDataE

Environment Vari × salesAnalysisRe × Execution results ×

```python
import boto3
import pymysql
import sys

def opener_db(event, context):

    # Retrieve the database connection information from the event input parameter.

    dbUrl = event['dbUrl']
    dbName = event['dbName']
    dbUser = event['dbUser']
    dbPassword = event['dbPassword']

    # Establish a connection to the Cafe database, and set the cursor to return results as a Python dictionary.

    try:
        conn = pymysql.connect(host=dbUrl, user=dbUser, passwd=dbPassword, db=dbName, cursorclass=pymysql.cursors.DictCursor)

    except pymysql.Error as e:
        print('ERROR: Failed to connect to the Cafe database.')
        print('Error Details: %d %s' % (e.args[0], e.args[1]))
        sys.exit()

    # Execute the query to generate the daily sales analysis result set.

    with conn.cursor() as cur:
        cur.execute("SELECT  c.product_group_number, c.product_group_name, a.product_id, b.product_name, CAST(sum(a.quantity) AS int) as qu
        result = cur.fetchall()

    # Close the connection.

    conn.close()

    # Return the result set.

    return {'statusCode': 200, 'body': result}
```

Code source Info

Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

salesAnalysisReport

salesAnalysisReportDataE

Environment Vari × salesAnalysisRe × Execution results ×

```python
import boto3
import pymysql
import sys

def opener_db(event, context):

    # Retrieve the database connection information from the event input parameter.

    dbUrl = event['dbUrl']
    dbName = event['dbName']
    dbUser = event['dbUser']
    dbPassword = event['dbPassword']

    # Establish a connection to the Ca

    try:
        conn = pymysql.connect(host=db

    except pymysql.Error as e:
        print('ERROR: Failed to connec
        print('Error Details: %d %s' %
        sys.exit()

    # Execute the query to generate th

    with conn.cursor() as cur:
        cur.execute("SELECT  c.product
        result = cur.fetchall()

    # Close the connection.

    conn.close()

    # Return the result set.

    return {'statusCode': 200, 'body':
```

Executing function: succeeded (logs)

▼ Details

The area below shows the last 4 KB of the execution log.

```
{
  "statusCode": 200,
  "body": []
}
```

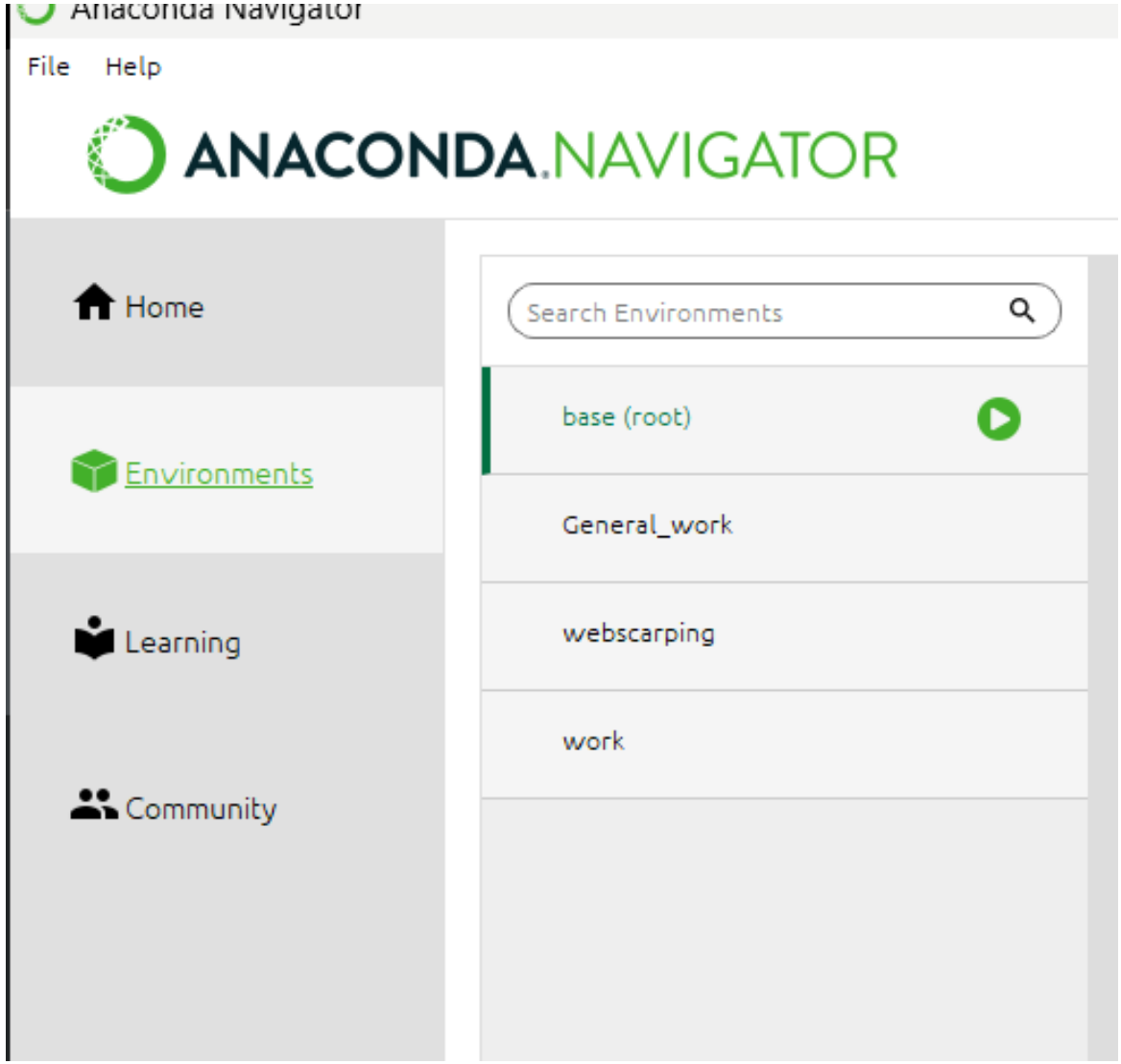
Summary

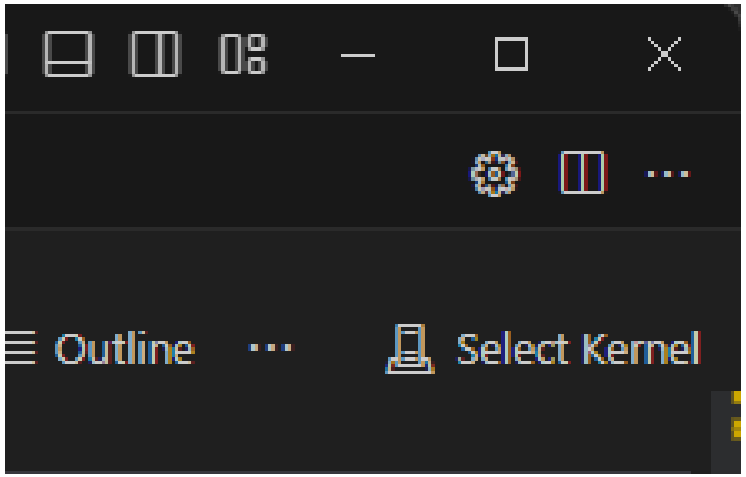
Code SHA-256 LuGDcNdAyUKhu2+HS/zxe1Z6e9Z4ry+QOz45L05wu9Q=	Execution time 2 minutes ago (September 6, 2023 at 01:22 PM GMT+3)
Request ID e81d8601-7029-475a-ad4d-b1941b20bb94	Function version $LATEST
Init duration 314.97 ms	Duration 49.77 ms
Billed duration 50 ms	Resources configured 128 MB
Max memory used 57 MB	

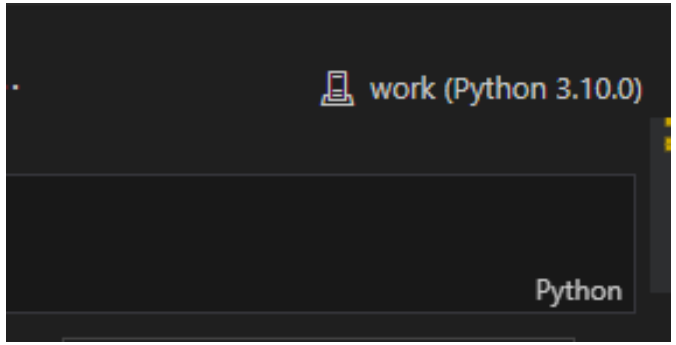
Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
START RequestId: e81d8601-7029-475a-ad4d-b1941b20bb94 Version: $LATEST
END RequestId: e81d8601-7029-475a-ad4d-b1941b20bb94
REPORT RequestId: e81d8601-7029-475a-ad4d-b1941b20bb94 Duration: 49.77 ms	Billed Duration: 50 ms	Memory Size: 128 MB	Max Memory Used: 57 MB	Init Duration: 314.97 ms
```


Anaconda Navigator
File
Help
ANACONDA.NAVIGATOR
Home
Environments
Learning
Community
Search Environments
base (root)
General_work
webscarping
work


Outline
Select Kernel


work (Python 3.10.0)
Python

Beni dinlediğiniz için teşekkür ederim.

Hüseyin IŞIK

Kaynakça

- https://docs.aws.amazon.com/lambda/latest/dg/chapter-layers.html?icmpid=docs_lambda_help
- https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html
- https://docs.aws.amazon.com/lambda/latest/dg/python-handler.html?icmpid=docs_lambda_help
- https://ipython.org/ipython-doc/3/development/kernels.html#:~:text=A%20'kernel'%20is%20a%20program,up%20communications%20with%20the%20frontend.
- https://docs.jupyter.org/en/latest/projects/kernels.html
- https://code.visualstudio.com/docs/datascience/jupyter-kernel-management
- https://github.com/awsdocs/aws-lambda-developer-guide/tree/main/sample-apps/blank-python
- https://docs.python.org/3/library/venv.html
- https://www.freecodecamp.org/news/why-you-need-python-environments-and-how-to-manage-them-with-conda-85f155f4353c/
- https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#activating-an-environment
- https://docs.aws.amazon.com/lambda/latest/dg/configuration-function-zip.html
- https://docs.aws.amazon.com/lambda/latest/dg/python-context.html