AWS Lambda Runtime Updater – Assignment

1. Overview

This Python program automates the process of updating **AWS Lambda functions** to a specific target runtime version based on given **tags**.

It iterates through all Lambda functions in an AWS account, checks if each function has a specific tag key and value, and updates the runtime (for example, from Python 3.11 to Python 3.13) for those that match.

2. Features

- Filters Lambda functions using a tag key and tag value.
- Updates the runtime only for matching functions.
- Supports **dry-run mode** that prints what would be updated without making real changes.
- Returns a clear summary of results including successful updates, skipped functions, and errors.

3. Function Description

Function name: `update_python_runtime(target_runtime, tag_key, tag_value, dry_run=False)`

Purpose:
Updates all AWS Lambda functions with a specific tag (`tag_key = tag_value`) to the given target runtime.

Parameters:

Parameter	Type	Description
`target_runtime`	String	The desired new runtime (for example, “python3.13”).
`tag_key`	String	The Lambda tag key to use for filtering functions.
`tag_value`	String	The value of the tag that must match.

`dry_run`	Boolean	If set to True, only displays the updates without actually changing anything.

Return Value:
A list of tuples, where each tuple represents one Lambda function and contains:

1. Function name
2. Old runtime
3. New runtime or reason (e.g., “already_python3.13”, “dry-run”, or “error: …”)

Example Output:

```
[
  ("my-func-1", "python3.11", "python3.13"),
  ("my-func-2", "python3.13", "already_python3.13"),
  ("my-func-3", "python3.9", "dry-run")
]
```

4. Example Usage

```
from my_lambda_updater import update_python_runtime

# Example 1 — Dry run (no actual updates)
update_python_runtime(
    target_runtime="python3.13",
    tag_key="Environment",
    tag_value="Production",
    dry_run=True
)

# Example 2 — Actual update
update_python_runtime(
    target_runtime="python3.13",
    tag_key="owner",
    tag_value="cloud",
    dry_run=False
)
```

5. Prerequisites

To run this program successfully, you need:

- **Python 3.9 or later**
- **AWS CLI** configured with valid credentials
- **boto3** library installed

Install dependencies using:

```
pip install boto3
```

Required AWS Permissions:

- `lambda:ListFunctions`
- `lambda:ListTags`
- `lambda:UpdateFunctionConfiguration`

6. AWS Environment Setup

Before executing the script, configure AWS credentials in one of the following ways:

Option 1: Using AWS CLI

```
aws configure
```

Option 2: Using Environment Variables

```
export AWS_ACCESS_KEY_ID=YOUR_ACCESS_KEY
export AWS_SECRET_ACCESS_KEY=YOUR_SECRET_KEY
export AWS_DEFAULT_REGION=us-east-1
```

7. Program Logic and Flow

1. The script lists all Lambda functions using the AWS `list_functions()` API.
2. For each function:
   - It retrieves tags using `list_tags()`.
   - It checks whether the tag key and tag value match the specified criteria.
   - If the runtime already matches the target runtime, it skips updating that function.
   - If in dry-run mode, it only prints what would be updated.
   - Otherwise, it performs an actual runtime update using `update_function_configuration()`.
3. It repeats this process until all pages of Lambda functions are processed (using Marker and NextMarker).
4. Finally, it returns a list summarizing all actions (updated, skipped, or error).

8. Sample Output

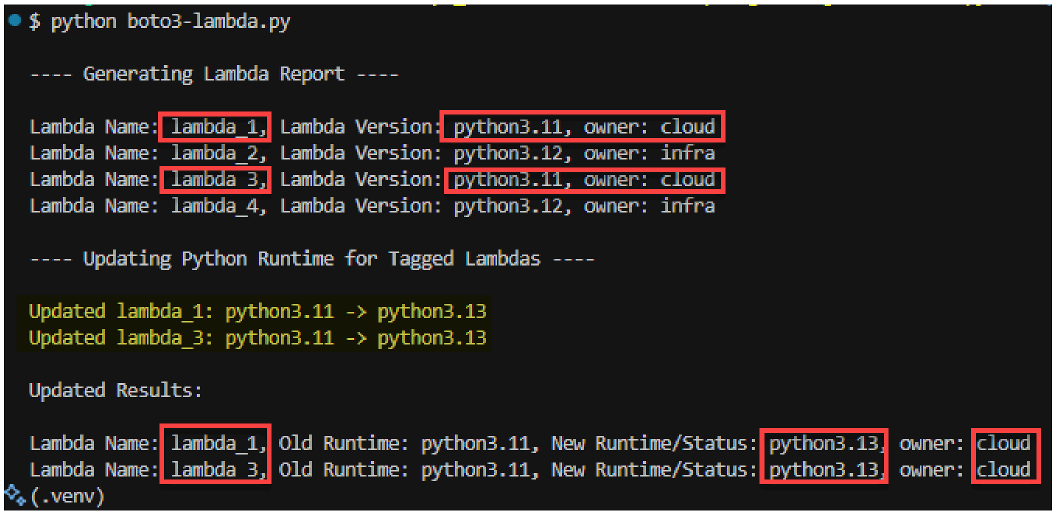
When running in dry-run=True mode:

```
[dry-run] Would update my-func-1: python3.11 -> python3.13
[dry-run] Would update my-func-2: python3.11 -> python3.13
```

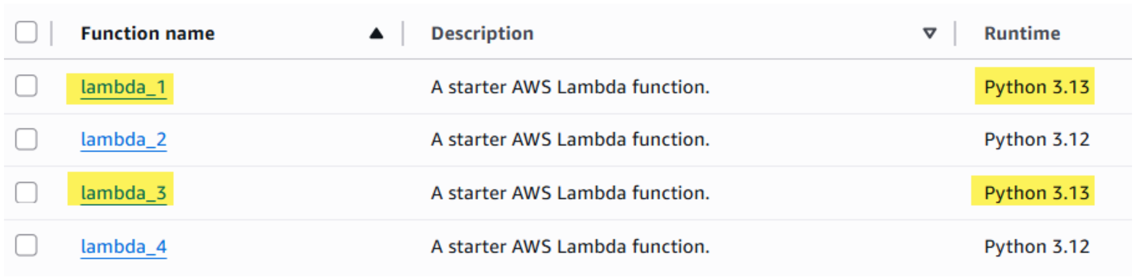
Returned Result:

```
[
  ("my-func-1", "python3.11", "dry-run"),
  ("my-func-2", "python3.11", "dry-run")
]
```

When running in dry-run=False mode:



```
$ python boto3-lambda.py

 ---- Generating Lambda Report ----

 Lambda Name: lambda_1, Lambda Version: python3.11, owner: cloud
 Lambda Name: lambda_2, Lambda Version: python3.12, owner: infra
 Lambda Name: lambda_3, Lambda Version: python3.11, owner: cloud
 Lambda Name: lambda_4, Lambda Version: python3.12, owner: infra

 ---- Updating Python Runtime for Tagged Lambdas ----

 Updated lambda_1: python3.11 -> python3.13
 Updated lambda_3: python3.11 -> python3.13

 Updated Results:

 Lambda Name: lambda_1, Old Runtime: python3.11, New Runtime/Status: python3.13, owner: cloud
 Lambda Name: lambda_3, Old Runtime: python3.11, New Runtime/Status: python3.13, owner: cloud
(.venv)
```



Function name	Description	Runtime
lambda_1	A starter AWS Lambda function.	Python 3.13
lambda_2	A starter AWS Lambda function.	Python 3.12
lambda_3	A starter AWS Lambda function.	Python 3.13
lambda_4	A starter AWS Lambda function.	Python 3.12

9. Notes and Best Practices

- Always test the script in **dry-run mode** before making real updates.
- Use a **non-production AWS account** first to verify correctness.
- If there are permission or configuration issues, they will be caught and recorded in the results list.

10. Conclusion

This project demonstrates a practical use case of automating AWS Lambda configuration management using **Python and boto3**.

It highlights good practices such as safe exception handling, tag-based filtering, and dry-run capability for controlled execution.

The script is easy to understand, extensible for real-world automation, and provides a clear learning example for students interested in **AWS cloud automation with Python**.

11. References

- AWS Lambda Developer Guide – https://docs.aws.amazon.com/lambda
- Boto3 Documentation – https://boto3.amazonaws.com/v1/documentation/api/latest/index.html