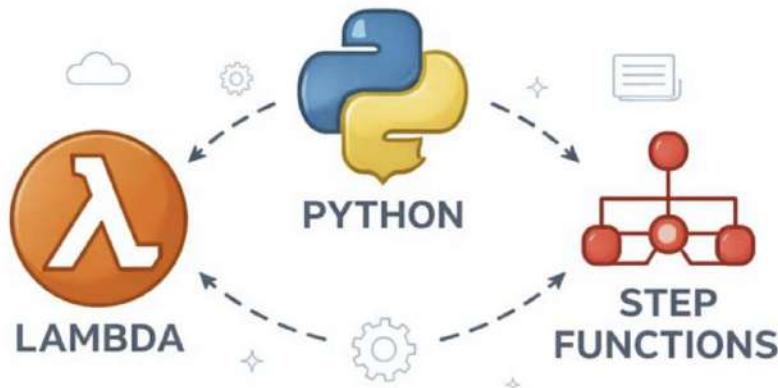


AWS Hands-on Lab #2 | Lambda (Python) & Step Functions



Réalisation d'un lab très simple pour comprendre le fonctionnement des **AWS Step Functions** et leur intégration avec **AWS Lambda**.

Objectifs :

- Créer deux fonctions Lambda minimales en code Python
- Orchestrer leur exécution avec une **State Machine (Standard)**
- Comprendre le workflow visuel, le code JSON généré automatiquement et les rôles IAM associés
- Tester l'exécution et analyser le résultat

Les opérations se font via la console AWS

Sommaire

1: Create Lambda Functions	2
2: Create Step Functions State Machine	4
3: Test the State Machine via Console	5
Step 4: Tests via CLI	7
4.1 Lister les Step Functions.....	7
4.2 Démarrer une exécution (CLI)	8
4.3 Vérifier le statut de l'exécution	8

4.3.1 Lister toutes les executions	8
4.4 Voir l'historique détaillé	9
5 Tester les Lambdas seules (CLI)	9
Syntaxe pour l'OS Windows (prompt cmd) :	10
CheckResult.....	10

1: Create Lambda Functions

Lambda 1: Add Numbers

Aller dans la console : AWS - Lambda

Runtime: Python 3.x

Function name: AddNumbers

```
def lambda_handler(event, context):
    result = event['a'] + event['b']
    return {
        "result": result
    }
```

Récupérer (copier) l'Arn qui a été crée pour cette fonction lambda :
L'ARN est affiché sur la fenêtre de la fonction

arn:aws:lambda:ca-central-1:ACCOUNT_ID:function:AddNumbers

ACCOUNT_ID correspond à l'ID de votre compte AWS

The screenshot shows the AWS Lambda Functions interface. The navigation bar at the top indicates the path: Lambda > Functions > AddNumbers. A success message in a green box states: "Successfully created the function **AddNumbers**. You can now change its code and configuration. To invoke your function with a test event, choose "Test". The main area has tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The Code tab is selected. Below it, the "Code source" section shows the file structure under EXPLORER: ADDNUMBERS/lambda_function.py. The code editor displays the contents of lambda_function.py:

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     result = event['a'] + event['b']
5     return {
6         "result": result
7     }
8
9
```

Ensuite cliquer sur “Deploy”

Lambda 2: Check Result

Function name: CheckResult

```
def lambda_handler(event, context):
    if event['result'] > 10:
        return {"status": "SUCCESS"}
    else:
        return {"status": "FAILURE"}
```

The screenshot shows the AWS Lambda Function Editor interface. At the top, a breadcrumb navigation bar indicates the path: Lambda > Functions > CheckResult. A success message states: "Successfully created the function CheckResult. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'". Below this, a "Code source" tab is selected, with an "Info" link. The main area is titled "lambda_function.py" and contains the following Python code:

```
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     if event['result'] > 10:
5         return {"status": "SUCCESS"}
6     else:
7         return {"status": "FAILURE"}
```

The "EXPLORER" sidebar shows a folder named "CHECKRESULT" containing "lambda_function.py". In the "DEPLOY" section, there are two buttons: "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+I)". The "Deploy" button is highlighted with a red box.

Ensuite , cliquer sur le bouton : ‘Deploy’

Récupérer l’ARN qui a été créé pour cette fonction :

arn:aws:lambda:ca-central-1:ACCOUNT_ID:function:CheckResult

ACCOUNT_ID correspond à l’ID de votre compte AWS

2: Create Step Functions State Machine

1. Go to AWS Step Functions
2. Click Create state machine
3. Choose Standard
4. Use Amazon States Language (ASL)
5. Insérer le code ASL ci-dessous dans la fenêtre de codage
6. Ensuite cliquer sur “create”

State Machine Definition :

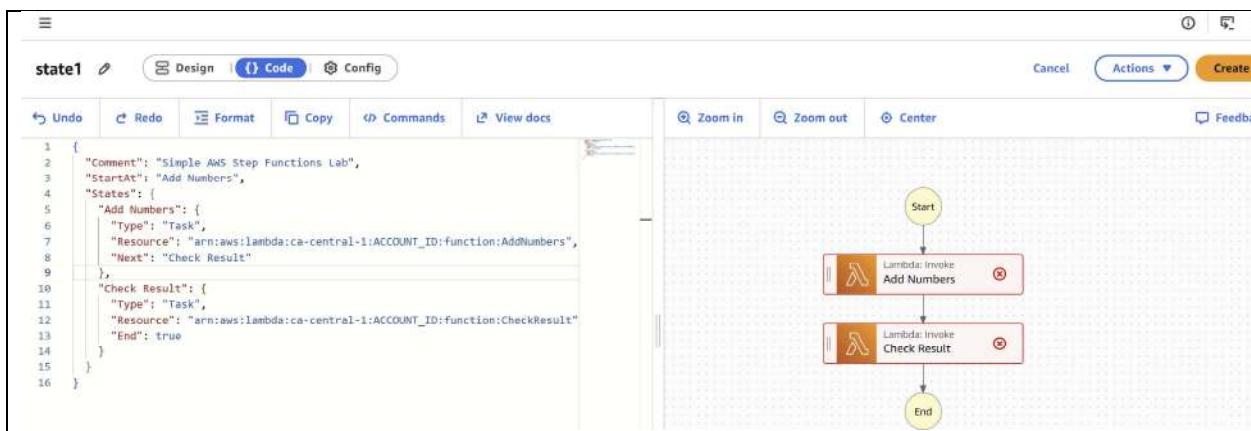
```
{  
    "Comment": "Simple AWS Step Functions Lab",  
    "StartAt": "Add Numbers",  
    "States": {
```

```

    "Add Numbers": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:AddNumbers",
        "Next": "Check Result"
    },
    "Check Result": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:CheckResult",
        "End": true
    }
}
}

```

Replace REGION and ACCOUNT_ID with your actual values.



3: Test the State Machine via Console

Tester via la console AWS

Cliquer sur le bouton “execute” .

Dans La fenêtre suivante qui s’Affiche, saisir les paramètres en input de la fonction. La partie ‘input’, correspond au niveau du code Python au terme : ‘event’ que l’on trouve , ici au niveau du handler Python de notre code : def lambda_handler(event, context):

On teste , par exemple, avec les valeurs ci-dessous, que l’on saisit (copier -coller)

Input

```
{
    "a": 7,
    "b": 6
}
```

Start execution

Name
33294146-0ec1-4306-8e01-71bee19e6f0f
Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

Input - optional
Enter input values for this execution in JSON format

Format JSON **Export** **Import**

```
1 + {
2   "a": 7,
3   "b": 6
4 }
```

Start execution with latest revision

Open in a new browser tab

Start execution

Ensuite , cliquer sur le bouton : ‘start execution’
La fenêtre ci-dessous s’Affiche alors, qui donne le résultat de l’exécution.

L’intérêt est que cet outil possède des caractéristiques visuelles, graphiques et montre clairement les résultats des fonctions lambda et aussi le statuts du déroulement des diverses phase du workflow .

Step Functions

Execution started successfully

Execution: 33294146-0ec1-4306-8e01-71bee19e6f0f

Details **Execution input and output** **Definition**

Execution status: Running

Execution type: Standard

Execution ARN: arn:aws:states:ca-central-1:...:execution:state1:33294146-0ec1-4306-8e01-71bee19e6f0f

IAM role ARN: arn:aws:iam::760606734887:role/service-role/StepFunctions-state1-role-r08cjp5ae

State transitions Learn more ↗ 4

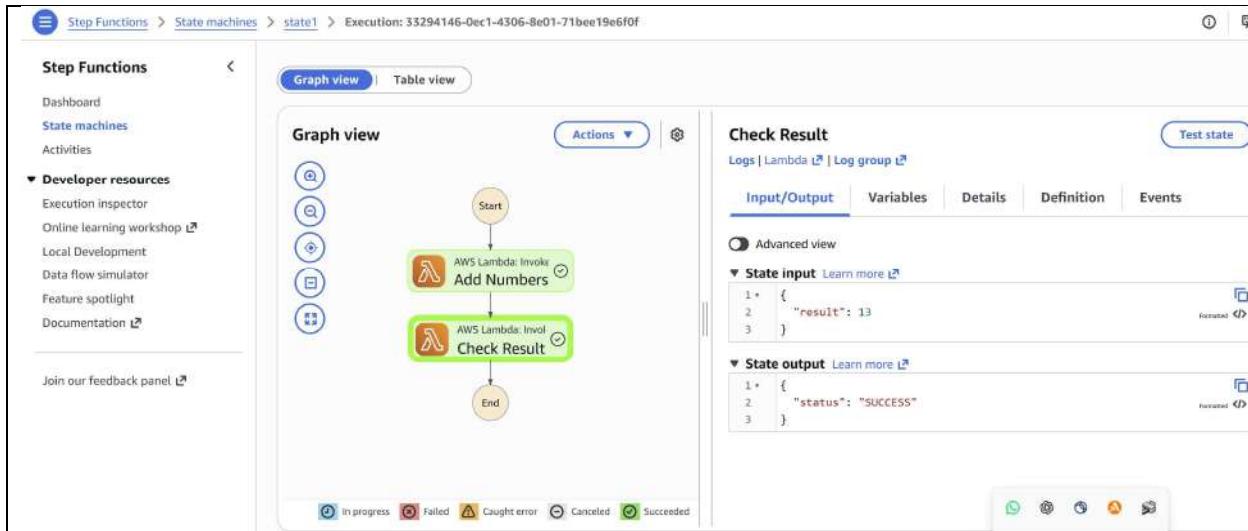
Start time: Jan 17, 2026, 20:49:07.347 (UTC-05:00)

End time: Jan 17, 2026, 20:49:08.204 (UTC-05:00)

Duration: 00:00:00.857

Alias: -

Version: -



Expected Output

```
{
  "status": "SUCCESS"
}
```

Step 4: Tests via CLI

4.1 Lister les Step Functions

aws stepfunctions list-state-machines

Repérer : stateMachineArn

```
C:\temp>
C:\temp>aws stepfunctions list-state-machines
{
  "stateMachines": [
    {
      "stateMachineArn": "arn:aws:states:ca-central-1:████████:stateMachine:state1",
      "name": "state1",
      "type": "STANDARD",
      "creationDate": 1768700339.677
    }
  ]
}
C:\temp>
```

4.2 Démarrer une exécution (CLI)

Remplace ACCOUNT_ID par l'ID de votre compte AWS :

```
aws stepfunctions start-execution \
--state-machine-arn arn:aws:states:ca-central-1:ACCOUNT_ID:stateMachine:SimpleLab \
--name test-exec-001 \
--input '{"a":7,"b":6}'
```

Au format OS Windows, la syntaxe de la commande , est :

Fonctionne avec la version suivante de AWS CLI : 2.33.2

```
aws stepfunctions start-execution ^
--state-machine-arn arn:aws:states:ca-central-1:ACCOUNT_ID:stateMachine:state1 ^
--cli-binary-format raw-in-base64-out ^
--input "{\"a\":7,\"b\":6}"
```

Résultat attendu :

```
{
  "executionArn": "...",
  "startDate": "..."
}
```

```
C:\Windows\System32>
C:\Windows\System32>aws stepfunctions start-execution ^
More? --state-machine-arn arn:aws:states:ca-central-1:[REDACTED] stateMachine:state1 ^
More? --cli-binary-format raw-in-base64-out ^
More? --input "{\"a\":7,\"b\":6}"
{
  "executionArn": "arn:aws:states:ca-central-1:[REDACTED]:execution:state1:b109381d-71b7-4fb3-a3ab-3d7dd2bbb0a1",
  "startDate": "2026-01-17T21:30:17.918000-05:00"
}

C:\Windows\System32>
```

Le nom d'exécution doit être unique.

4.3 Vérifier le statut de l'exécution

4.3.1 Lister toutes les executions

```
aws stepfunctions list-executions --state-machine-arn arn:aws:states:ca-central-
1:ACCOUNT_ID:stateMachine:state1
```

recuperer dans le resultat de la commande ci-dessus, **executionArn** qui est ce que l'on dois mettre dans **describe-execution**, pour la commande ci-dessous .

Executionarn vaut, dans cet exemple :

arn:aws:states:ca-central-1:ACCOUNT_ID:execution:state1:33294146-0ec1-4306-8e01-71bee19e6f0f

```
aws stepfunctions describe-execution \
--execution-arn EXECUTION_ARN
```

```
aws stepfunctions describe-execution ^
--execution-arn EXECUTION_ARN
```

```
C:\Windows\System32>aws stepfunctions describe-execution ^
More? --execution-arn arn:aws:states:ca-central-1:██████████:execution:state1:33294146-0ec1-4306-8e01-71bee19e6f0f
{
    "executionArn": "arn:aws:states:ca-central-1:██████████:execution:state1:33294146-0ec1-4306-8e01-71bee19e6f0f",
    "stateMachineArn": "arn:aws:states:ca-central-1:██████████:stateMachine:state1",
    "name": "33294146-0ec1-4306-8e01-71bee19e6f0f",
    "status": "SUCCEEDED",
    "startDate": "2026-01-17T20:49:07.347000-05:00",
    "stopDate": "2026-01-17T20:49:08.204000-05:00",
    "input": "{\"a\":7,\"b\":6}",
    "inputDetails": {
        "included": true
    },
    "output": "{\"status\": \"SUCCESS\"}",
    "outputDetails": {
        "included": true
    },
    "redriveCount": 0,
    "redriveStatus": "NOT_REDRIABLE",
    "redriveStatusReason": "Execution is SUCCEEDED and cannot be redriven"
}

C:\Windows\System32>
```

Champs importants :

status → SUCCEEDED

```
output :
{
    "status": "SUCCESS"
}
```

4.4 Voir l'historique détaillé

```
aws stepfunctions get-execution-history \
--execution-arn EXECUTION_ARN \
--max-items 20
```

Permet de voir :

TaskStateEntered

LambdaFunctionSucceeded

Données échangées entre états

5 Tester les Lambdas seules (CLI)

AddNumbers

Syntaxe pour l'OS Windows (prompt cmd) :

```
aws lambda invoke ^
--function-name AddNumbers ^
--payload "{\"a\":5,\"b\":8}" ^
response.json
```

```
C:\temp>
C:\temp>aws lambda invoke ^
More?   --function-name AddNumbers ^
More?   --payload "{\"a\":5,\"b\":8}" ^
More?   response.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
C:\temp>
```

```
cat response.json
```

```
C:\temp>type response.json
{"result": 13}
C:\temp>
C:\temp>
```

CheckResult

```
aws lambda invoke \
--function-name CheckResult \
--payload '{"result":12}' \
response.json
```

ou pour OS Windows, prompt cmd.exe

La bonne syntaxe est :

```
aws lambda invoke ^
--function-name CheckResult ^
--cli-binary-format raw-in-base64-out ^
--payload "{\"result\":12}" ^
response.json
```

```
type response.json
```