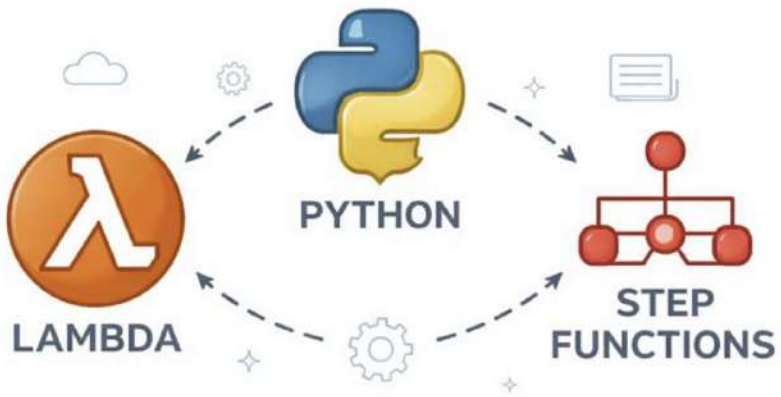
AWS Hands-on Lab #2 | Lambda (Python) & Step Functions



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

Lambda 1: Add Numbers

Aller dans la console : AWS - Lambda

Runtime: Python 3.x

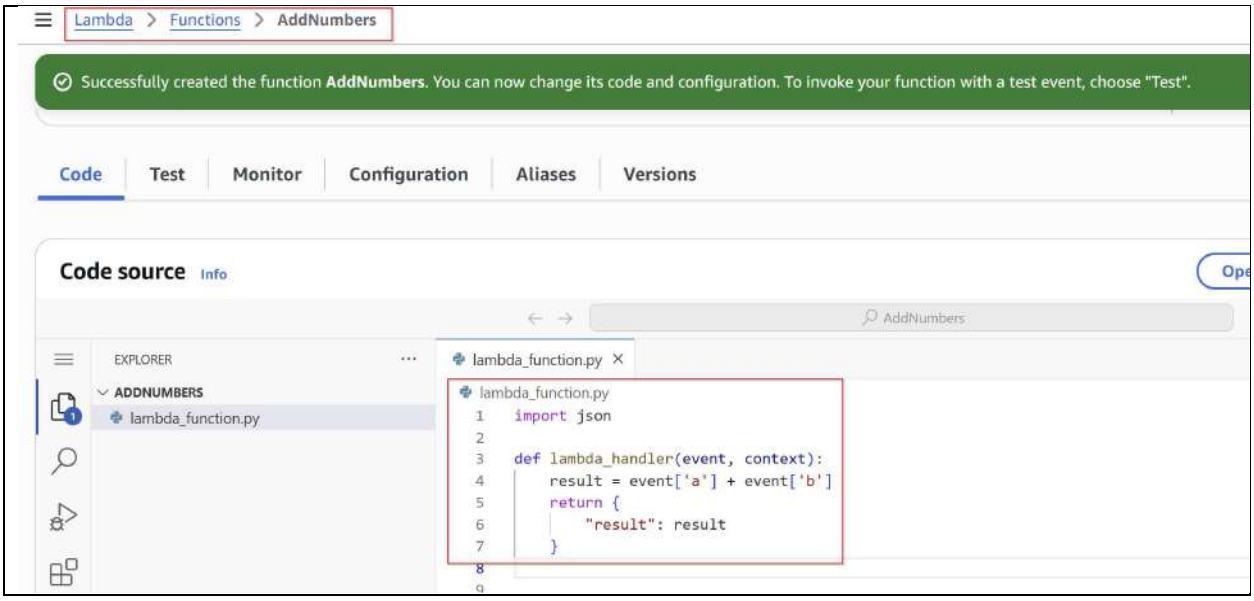
Function name: AddNumbers

```
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

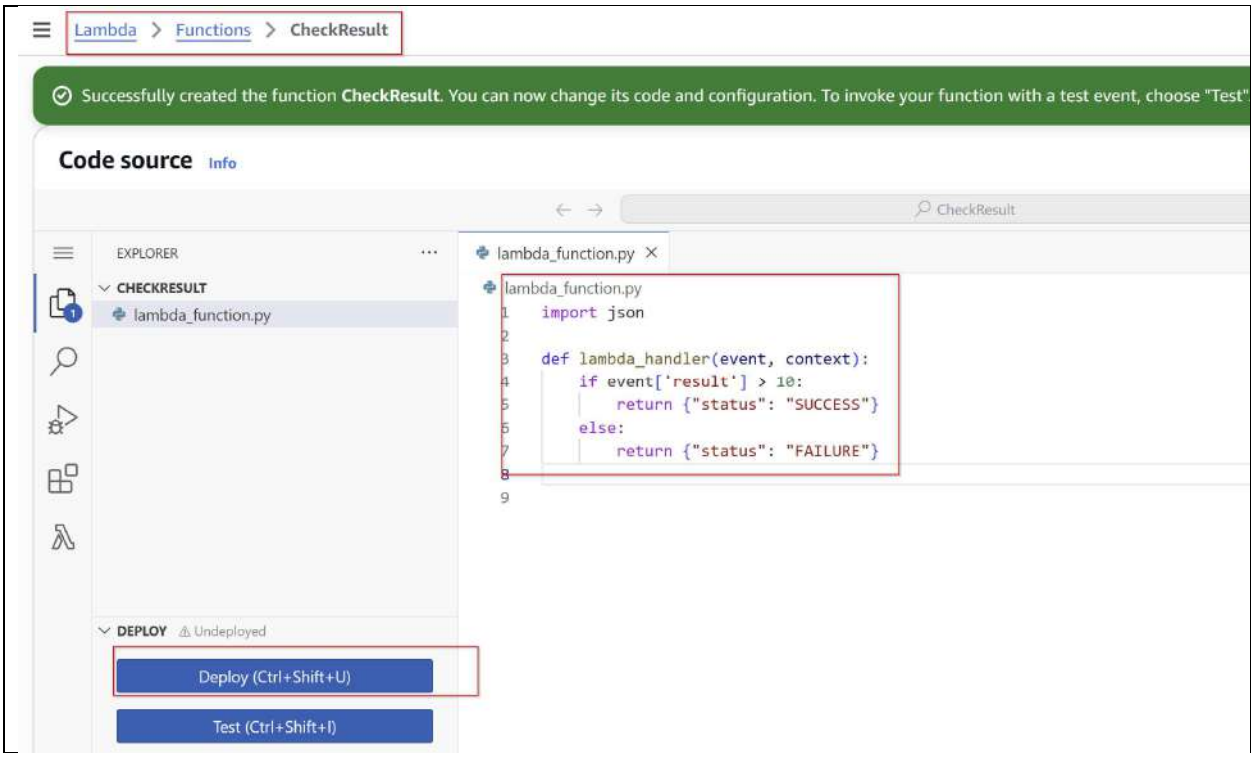


Ensuite cliquer sur ‘’Deploy’’

Lambda 2: Check Result

Function name: CheckResult

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



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

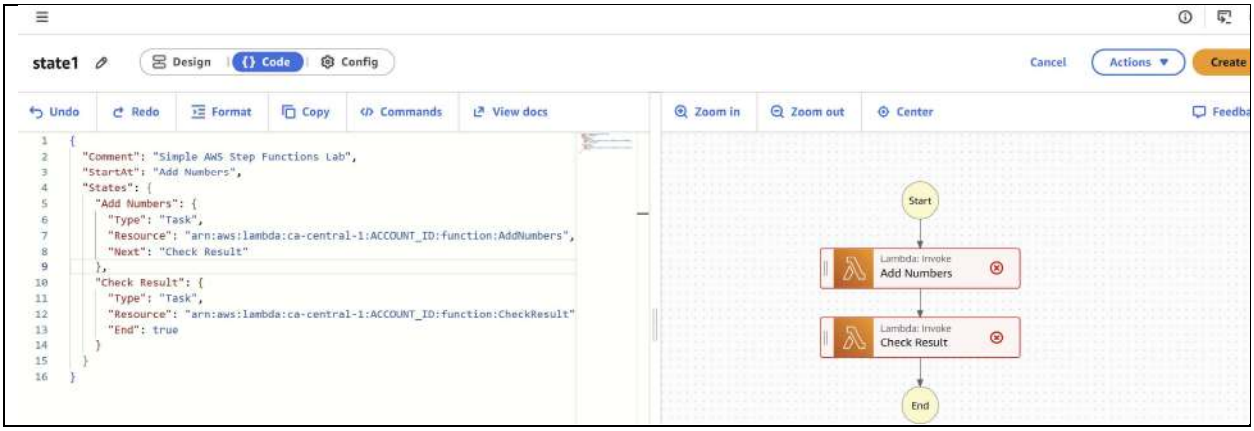
1. Go to AWS Step Functions
2. Click Create state machine
3. Choose Standard
4. Use Amazon States Language (ASL)
5. Insérer le code ASL ci-dessous dans la fenêtre de codage
6. Ensuite cliquer sur ‘’create’’

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.



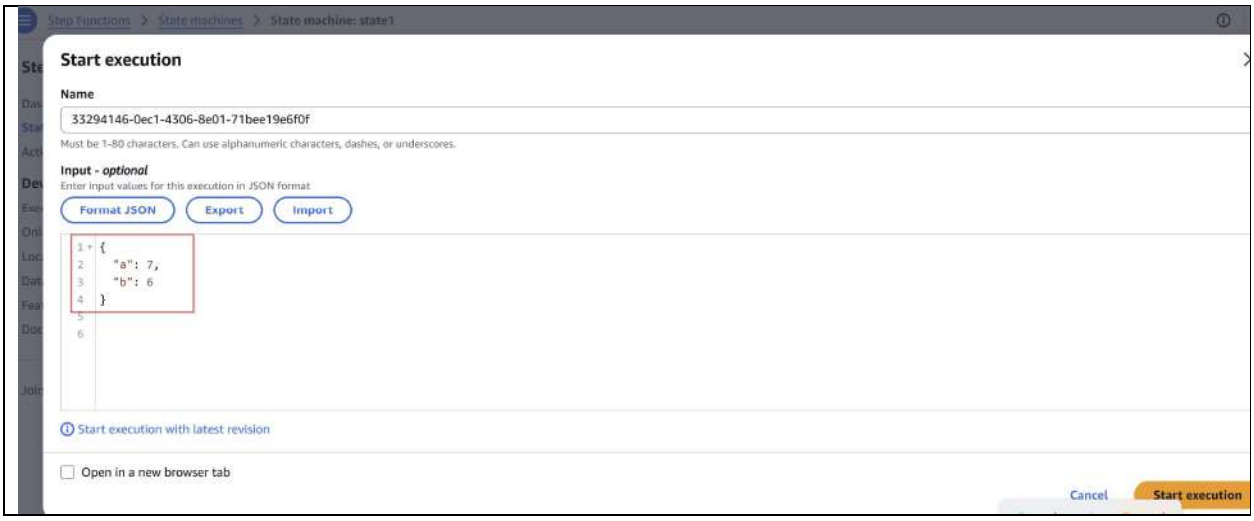
3: Test the State Machine via Console

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)

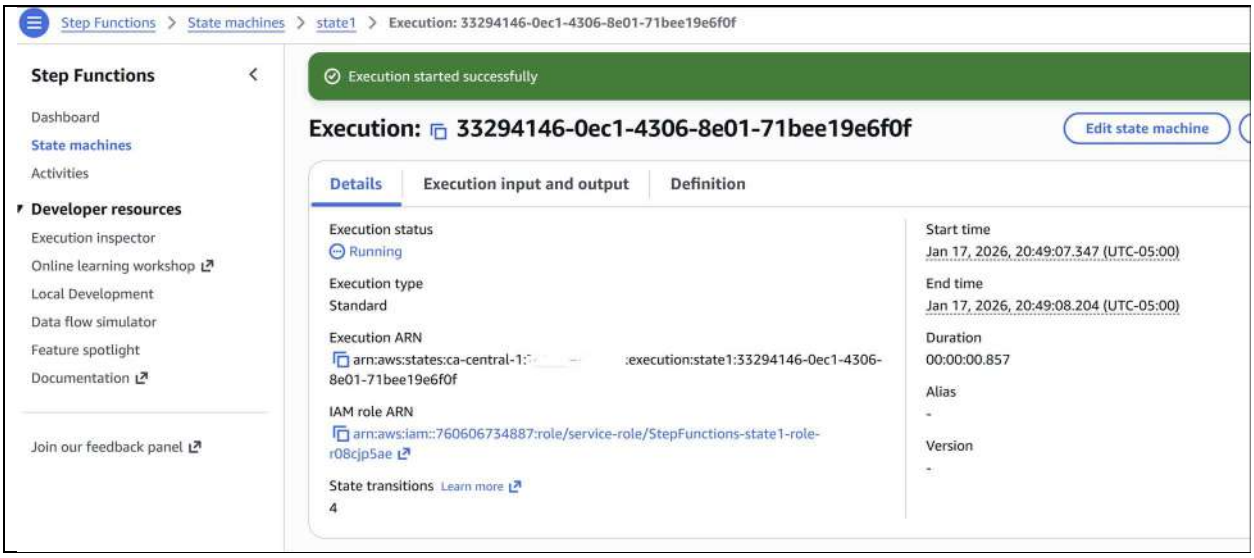
Input

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



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 .



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:[redacted]: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": "..."
}
```

Le nom d'exécution doit être unique.

4.3 Vérifier le statut de l'exécution

4.3.1 Lister toutes les executions

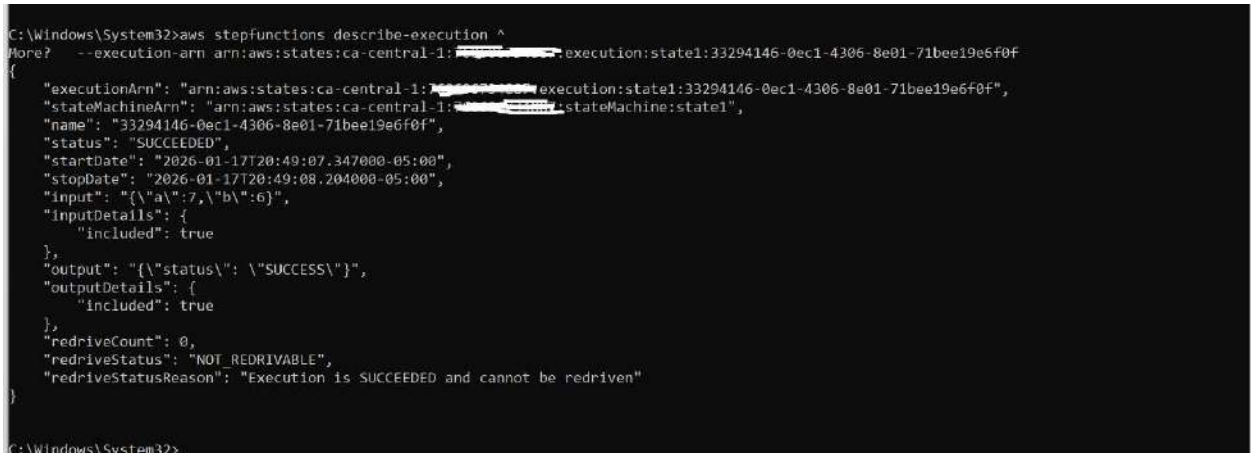
aws stepfunctions list-executions --state-machine-arn arn:aws:states:ca-central-1:ACCOUNT_ID:stateMachine:state1

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



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 (promt 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