

Terraform Lab4— Lambda Python + IAM Role + DynamoDB

(DVA-C02 Foundation)

(Architecture IAM + Lambda + DynamoDB)



Objectif :

- Démontrer la séparation Rôle / Policy
- Déployer une table DynamoDB On-Demand
- Préparer l'exécution Lambda avec logs

Prérequis rapides :

- Windows OS
- Compte AWS
- AWS CLI installé
- Terraform installé et dans le PATH

Contents

1 Présentation :	2
2 Prérequis techniques.....	3
3 Vérifications initiales	3
3.1 Vérifications de base	3
3.2 vérifications détaillées.....	3
4 Création du projet :	3
4.1 Création du dossier	3
4.2 Création du fichier lambda_function.py :	4
4.3 Création du fichier main.tf	4
4.4 ajout de la table dynamDB.....	5

4.5 IAM + policy.....	5
4.6 zip du code lambda	7
4.7 Lambda function – Terraform.....	7
5 Déploiement.....	8
5.1 Initialiser Terraform :.....	8
5.2 Voir le plan Terraform:	8
5.3 Création de la ressource :.....	8
6 Verifications :.....	8
6.1 vérifier les ressources créées (terraform stat list).....	8
6.2 vérifications via CLI.....	8
6.2.1 fonction lambda	8
6.2.2. Vérifier dynamoDB.....	9
6.2.3 Vérifier le rôle IAM	11
6.2.4 Vérifier les policies attachées au rôle.....	11
6.3 Vérification via AWS management console.....	11
6.3.1 Présence de la table DynamoDB	11
6.4 Vérifications cloudwatch	12
6.4.1 log group	12
6.4.2 log stream	13
7 Suppression des ressources	14
7.1 Avec Terraform :	14
7.2 Vérifications dans AWS/CLI	14

1 Présentation :

Dans ce lab, je me concentre sur la **fondation IAM**, souvent négligée mais critique en production.

Objectif : démontrer la maîtrise de **Terraform**, du **principe du moindre privilège**, et de la **séparation rôle / policy** pour une intégration Lambda–DynamoDB.

The lab intentionally focuses on IAM and infrastructure design, not on application execution.

All resources are deployed using Infrastructure as Code (Terraform), without using the AWS Console.

- ✓ Création d'une table **DynamoDB On-Demand**
- ✓ Définition d'un **IAM Role dédié à Lambda**
- ✓ Création de **policies minimales** :
 - accès DynamoDB ciblé (`GetItem` / `PutItem` / `UpdateItem`)
 - accès CloudWatch Logs (**observabilité**)
 - ✓ Attachement des policies via Terraform
 - ✓ Infrastructure prête pour Lambda, sans clé d'accès

2 Prérequis techniques

- OS : Windows
- Un compte AWS actif
- AWS CLI installé et configuré
- Terraform installé

installation Terraform (Windows)

- Télécharger `terraform.exe` depuis le site officiel **HashiCorp**
👉 <https://developer.hashicorp.com/terraform>
- Copier `terraform.exe` dans le répertoire : `c:\terraform`

3 Vérifications initiales

3.1 Vérifications de base

```
#terraform -version
```

```
C:\Windows\System32>terraform -version
Terraform v1.14.3
on windows_amd64
```

```
#aws --version
```

```
C:\Windows\System32>aws --version
aws-cli/1.42.17 Python/3.12.6 Windows/11 botocore/1.40.17
```

3.2 vérifications détaillées

```
#aws sts get-caller-identity
```

4 Création du projet :

4.1 Création du dossier

```
mkdir terraform-lab-lambda-dynamodb
cd terraform-lab-lambda-dynamodb
mkdir lambda

notepad lambda\lambda_function.py
```

```
Directory of C:\terraform\terraform-lab-lambda-dynamodb

12/22/2025  01:12 PM    <DIR>          .
12/22/2025  12:56 PM    <DIR>          ..
12/22/2025  01:24 PM    <DIR>          lambda
12/22/2025  01:25 PM           2,144 main.tf
                           1 File(s)      2,144 bytes
                           3 Dir(s)   694,060,797,952 bytes free
```

```
C:\terraform\terraform-lab-lambda-dynamodb>
```

4.2 Création du fichier lambda_function.py :

Dans lambda\lambda_function.py :

Ajouter le code suivant dans le fichier lambda_function.py

```
import json
import boto3
import uuid

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Lab1-Events')

def lambda_handler(event, context):
    event_id = str(uuid.uuid4())

    table.put_item(
        Item={
            'event_id': event_id,
            'message': 'Hello from Lambda'
        }
    )

    return {
        'statusCode': 200,
        'body': json.dumps({
            'event_id': event_id,
            'status': 'inserted'
        })
    }
```

4.3 Création du fichier main.tf

Dans le répertoire : terraform-lab-lambda-dynamodb

notepad [main.tf](#)

Copier le code suivant dans le fichier : main.tf

```
terraform {
  required_version = ">= 1.5"
```

```

required_providers {
  aws = {
    source = "hashicorp/aws"
    version = "~> 5.0"
  }
}

provider "aws" {
  region = "ca-central-1"
}

```

4.4 ajout de la table dynamDB

Ajouter dans le fichier main.tf , le code ci-dessous

Table dynamoDB

Table de type on demand

```

resource "aws_dynamodb_table" "lab_table" {
  name          = "Lab1-Events"
  billing_mode = "PAY_PER_REQUEST"

  hash_key = "event_id"

  attribute {
    name = "event_id"
    type = "S"
  }
}

```

4.5 IAM + policy

Rajouter la section suivante dans main.tf

Création d'un rôle pour la fonction lambda

```

resource "aws_iam_role" "lambda_exec_role" {
  name = "lab-lambda-dynamodb-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = {
          Service = "lambda.amazonaws.com"
        }
        Action = "sts:AssumeRole"
      }
    ]
  })
}

```

Rajouter la section suivante à main.tf :

IAM policy – accès à DynamoDB

Ce que le rôle doit pouvoir faire

```
resource "aws_iam_policy" "dynamodb_rw_policy" {
  name = "lab-dynamodb-rw-policy"

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Action = [
          "dynamodb:GetItem",
          "dynamodb:PutItem",
          "dynamodb:UpdateItem"
        ]
        Resource = aws_dynamodb_table.lab_table.arn
      }
    ]
  })
}
```

Rajouter la section suivante :

Ce que le rôle doit pouvoir faire

Policy log cloudwatch

```
resource "aws_iam_policy" "lambda_logs_policy" {
  name = "lab-lambda-logs-policy"

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Action = [
          "logs>CreateLogGroup",
          "logs>CreateLogStream",
          "logs:PutLogEvents"
        ]
        Resource = "*"
      }
    ]
  })
}
```

Rajouter la section suivante :

Attacher les polices au rôle

```
resource "aws_iam_role_policy_attachment" "attach_dynamodb" {
  role      = aws_iam_role.lambda_exec_role.name
  policy_arn = aws_iam_policy.dynamodb_rw_policy.arn
}

resource "aws_iam_role_policy_attachment" "attach_logs" {
  role      = aws_iam_role.lambda_exec_role.name
  policy_arn = aws_iam_policy.lambda_logs_policy.arn
}
```

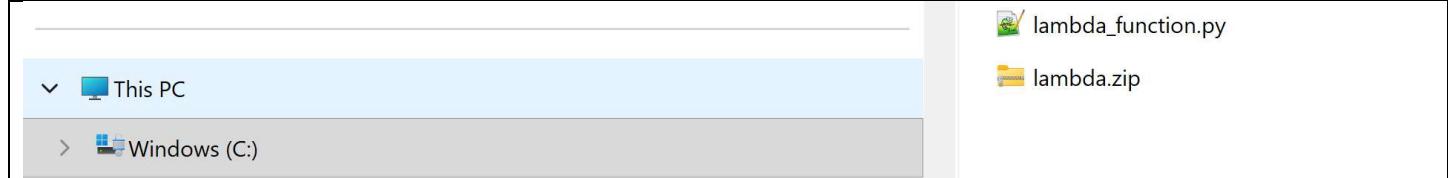
4.6 zip du code lambda

Dans le dossier projet :

```
cd lambda
tar -a -c -f lambda.zip lambda_function.py
cd ..
```

Résultat :

lambda\lambda.zip



```
Directory of C:\terraform\terraform-lab-lambda-dynamodb\lambda
```

```
12/22/2025  01:24 PM    <DIR>      .
12/22/2025  01:12 PM    <DIR>      ..
12/22/2025  01:24 PM          455 lambda.zip
12/22/2025  01:08 PM          491 lambda_function.py
                           2 File(s)   946 bytes
                           2 Dir(s)  694,058,655,744 bytes free
```

```
C:\terraform\terraform-lab-lambda-dynamodb\lambda>
```

4.7 Lambda function – Terraform

Ajouter le code suivant , dans main.tf

```
resource "aws_lambda_function" "lab_lambda" {
  function_name = "lab-lambda-dynamodb"
  role          = aws_iam_role.lambda_exec_role.arn
  handler       = "lambda_function.lambda_handler"
  runtime        = "python3.12"

  filename        = "lambda/lambda.zip"
  source_code_hash = filebase64sha256("lambda/lambda.zip")

  timeout = 10
}
```

5 Déploiement

5.1 Initialiser Terraform :

```
terraform init
```

5.2 Voir le plan Terraform:

```
terraform plan
```

5.3 Création de la ressource :

```
Terraform apply
```

6 Verifications :

6.1 vérifier les ressources créées (terraform stat list)

```
terraform state list
```

Résultats :

```
C:\terraform\terraform-lab-lambda-dynamodb>
C:\terraform\terraform-lab-lambda-dynamodb>terraform state list
aws_dynamodb_table.lab_table
aws_iam_policy.dynamodb_rw_policy
aws_iam_policy.lambda_logs_policy
aws_iam_role.lambda_exec_role
aws_iam_role_policy_attachment.attach_dynamodb
aws_iam_role_policy_attachment.attach_logs
aws_lambda_function.lab_lambda

C:\terraform\terraform-lab-lambda-dynamodb>
```

6.2 vérifications via CLI

6.2.1 fonction lambda

```
aws lambda invoke --function-name lab-lambda-dynamodb response.json
```

type response.json

On doit voir :

```
{"statusCode":200,"body":"{\"status\":\"inserted\"}"}  
{"statusCode":200,"body":"{}"}
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws lambda invoke --function-name lab-lambda-dynamodb response.json
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}

C:\terraform\terraform-lab-lambda-dynamodb>
```

6.2.2. Vérifier dynamoDB

Existence de la table , on vérifie que la table a été créée :

```
aws dynamodb describe-table --table-name Lab1-Events
```

On vérifie que l'item est inséré :

```
aws dynamodb scan --table-name Lab1-Events
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws dynamodb scan --table-name Lab1-Events
{
    "Items": [
        {
            "event_id": {
                "S": "16e3864e-2047-4b8d-9588-17b85e4a2b56"
            },
            "message": {
                "S": "Hello from Lambda"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

```
C:\terraform\terraform-lab-lambda-dynamodb>
```

Vérifier le ou les items enregistrés dans la table dynamDB :

On saisit la requête suivante ;

```
aws dynamodb query ^
--table-name Lab1-Events ^
--key-condition-expression "event_id = :v1" ^
--expression-attribute-values "{\":v1\":{\"S\":\"16e3864e-2047-4b8d-9588-17b85e4a2b56\"}}" ^
--output table
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws dynamodb query ^
More? --table-name Lab1-Events ^
More? --key-condition-expression "event_id = :v1" ^
More? --expression-attribute-values "{\":v1\":{\"S\":\"16e3864e-2047-4b8d-9588-17b85e4a2b56\"}}" ^
More? --output table
+-----+-----+
|       Query      | +-----+
| ConsumedCapacity | | Count | ScannedCount |
+-----+-----+-----+
| None            | | 1    | 1           |
+-----+-----+
+-----+-----+
|       Items      | +-----+
|   event_id       | | S    | 16e3864e-2047-4b8d-9588-17b85e4a2b56 |
+-----+-----+
|       message     | +-----+
|   S             | | Hello from Lambda |
+-----+-----+
C:\terraform\terraform-lab-lambda-dynamodb>
```

Scan de la table :

```
aws dynamodb scan ^
--table-name Lab1-Events ^
--output table
C:\terraform\terraform-lab-lambda-dynamodb>aws dynamodb scan ^
More? --table-name Lab1-Events ^
More? --output table
+-----+-----+
|       Scan      | +-----+
| ConsumedCapacity | | Count | ScannedCount |
+-----+-----+-----+
| None            | | 1    | 1           |
+-----+-----+
+-----+-----+
|       Items      | +-----+
|   event_id       | | S    | 16e3864e-2047-4b8d-9588-17b85e4a2b56 |
+-----+-----+
|       message     | +-----+
|   S             | | Hello from Lambda |
+-----+-----+
C:\terraform\terraform-lab-lambda-dynamodb>
```

Tester en mode absolu (get-item) :

Utile lorsqu'on connaît la clé de l'item (*event_id - Partition key*)

```
aws dynamodb get-item ^
--table-name Lab1-Events ^
--key "{\"event_id\":{\"S\":\"16e3864e-2047-4b8d-9588-17b85e4a2b56\"}}" ^
--output table
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws dynamodb get-item ^
More? --table-name Lab1-Events ^
More? --key "{\"event_id\":{\"S\":\"16e3864e-2047-4b8d-9588-17b85e4a2b56\"}}" ^
More? --output table
```

GetItem	
Item	
event_id	
S	16e3864e-2047-4b8d-9588-17b85e4a2b56
message	
S	Hello from Lambda

```
C:\terraform\terraform-lab-lambda-dynamodb>
```

6.2.3 Vérifier le rôle IAM

On vérifie que le rôle a bien été créé

```
aws iam get-role --role-name lab-lambda-dynamodb-role
```

6.2.4 Vérifier les policies attachées au rôle

```
aws iam list-attached-role-policies --role-name lab-lambda-dynamodb-role
```

6.3 Vérification via AWS management console

6.3.1 Présence de la table DynamoDB

Tables (1) Info		Last updated December 19, 2025, 12:42 (UTC-5:00)		↻		
Name	Status	Partition key	Sort key	Indexes	Replication Regions	Delete
Lab1-Events	Active	event_id (S)	-	0	0	Delete

DynamoDB > Explore items > Lab1-Events

DynamoDB

- Dashboard
- Tables
- Explore items**
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations
- Reserved capacity
- Settings

Scan or query items

Any tag key ▾

Filter by tag value Any tag value ▾

Find tables ▾

1

Lab1-Events

Scan **Query**

Select a table or index Table - Lab1-Events ▾

Select attr All attr

Filters - optional

Run **Reset**

Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCU

Table: Lab1-Events - Items returned (1)

Scan started on December 22, 2025, 14:23:40

	event_id (String)	message
<input type="checkbox"/>	16e3864e-2047-4b8d...	Hello from Lambda

6.4 Vérifications cloudwatch

6.4.1 log group

Vérification via la console de gestion AWS du log group créé pour ce lab.

On vérifie la création d'un log group : aws/lambda/lab-lambda-dynamodb

Et aussi qu'un log Stream a été créé (à l'intérieur du log group)

EC2 AWS Resource Explorer Resource Groups & Tag E... CloudWatch Systems Manager Trusted Advisor S3 VPC CloudTrail R

CloudWatch > Log management

CloudWatch

- Favorites and recents ▶
- Dashboards
- ▶ Alarms ⚠ 0 ○ 0 ○ 0
- ▶ Application Signals (APM) New
- ▶ Infrastructure Monitoring
- ▼ Logs

Log groups Data sources - new Summary

Log groups (1)

By default, we only load up to 10,000 log groups.

Filter log groups or try pattern search

Actions

Log group	Log class	Anomaly d...
/aws/lambda/lab-lambda-dynamodb	Standard	Configure

Verifications via CLI :

```
aws logs describe-log-groups --query "logGroups[?starts_with(logGroupName,'/aws/lambda')].logGroupName" --output table
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws logs describe-log-groups --query "logGroups[?starts_with(logGroupName,'/aws/lambda')].logGroupName" --output table
-----
|   DescribeLogGroups   |
+-----+
| /aws/lambda/lab-lambda-dynamodb |
+-----+
C:\terraform\terraform-lab-lambda-dynamodb>
```

6.4.2 log stream

Vérification via la console de gestion AWS des événements au sein du log Stream :

The screenshot shows the AWS CloudWatch Log Events interface. At the top, there's a filter bar with a search input, a 'Clear' button, and time range buttons for 1m, 30m, and 1h. Below the filter bar is a 'Display' dropdown set to 'Table'. The main area has two columns: 'Timestamp' and 'Message'. There are four log entries:

Timestamp	Message
	No older events at this moment. Retry
2025-12-22T19:19:17.855Z	INIT_START Runtime Version: python:3.12.v101 Runtime Version ARN: arn:aws:lambda:ca-cer
2025-12-22T19:19:18.317Z	START RequestId: 6e93e240-9fd4-4472-ac81-b678f9b8abec Version: \$LATEST
2025-12-22T19:19:18.609Z	END RequestId: 6e93e240-9fd4-4472-ac81-b678f9b8abec
2025-12-22T19:19:18.609Z	REPORT RequestId: 6e93e240-9fd4-4472-ac81-b678f9b8abec Duration: 291.12 ms Billed Durat
	No newer events at this moment. Auto retry paused. Resume

Vérifications via CLI :

```
aws logs describe-log-streams ^
--log-group-name "/aws/lambda/lab-lambda-dynamodb" ^
--order-by "LastEventTime" ^
--descending ^
--limit 5 ^
--output table
```

```
C:\terraform\terraform-lab-lambda-dynamodb>aws logs describe-log-streams ^  
More? --log-group-name "/aws/lambda/lab-lambda-dynamodb" ^  
More? --order-by "LastEventTime" ^  
More? --descending ^  
More? --limit 5 ^  
More? --output table  
-----  
| DescribelogStreams  
+-----  
| | logStreams  
+-----  
|| arn:aws:logs:ca-central-1:760606734887:log-group:/aws/lambda/lab-lambda-dynamodb:log-stream:2025/12/22[$LATEST]08449bc804554e53aaaf26005c6a35350  
|| creationTime 1766431161591  
|| firstEventTimestamp 1766431157855  
|| lastEventTimestamp 1766431158609  
|| lastIngestionTime 1766431161601  
|| logStreamName 2025/12/22[$LATEST]08449bc804554e53aaaf26005c6a35350  
|| storedBytes 0  
|| uploadSequenceToken 49039859655698196094385195242241228851082188927372364734
```

7 Suppression des ressources

7.1 Avec Terraform :

```
Terraform destroy
```

7.2 Vérifications dans AWS/CLI

Soit via la console, soit via le CLI
on vérifie la suppression de la table
en mode CLI :

```
aws dynamodb describe-table --table-name Lab1-Events
```