CLOUD COMPUTING, CM3

PREMIERS PAS DANS LE SERVERLESS

Pépin Rémi, Ensai, 2025

remi.pepin@insee.fr

LE SERVERLESS

UN SERVICE AVEC SERVEUR

Cas d'un service classique type web service

- Je choisis la configuration de mon instance EC2
- J'installe mon web service
- J'ai un accès total à ma machine

Je suis facturé au temps d'utilisation de mon service. Tant qu'il est allumée je le paie. Qu'il traite 2 ou 100 000 requêtes le prix sera le même.

UN SERVICE SANS SERVEUR ?

Cas d'une fonction as a Service

- Je choisis la puissance et le déclencheur de ma fonction
- J'upload le code de ma fonction
- Quand un événement déclenche ma fonction, elle est exécutée

Je suis facturé au temps d'exécution de ma fonction. Si elle est souvent déclenchée je paye cher, si elle ne l'est pas, je ne paye rien.

LE SERVERLESS

- Terme mal choisi : des serveurs sont bien utilisés
- Mais ils sont 100% gérés par le CP et pas par vous !

Un service dit "serverless" est un service qui masque totalement la partie serveur au client, qui devient à la charge du CP. PaaS ?

DIFFÉRENTS SERVICES SERVERLESS

- **Function as a Service** : AWS Lambda, GCP Cloud Functions, Azure Functions
- **Base de données** : AWS Aurora serverless, AWS DynamoDB, GCP Datastore
- **Stockage objet** : Amazon S3, GCP Cloud Storage, OVH Object Storage
- Et bien d'autres

AVANTAGES / INCONVÉNIENTS

✅ Avantages

- Coût 💴 : si utilisation imprévisible et plutôt faible
- Échelle / Élasticité 📈: théoriquement pas de limite car gérées par CP
- Productivité 👷 : facile à déployer car moins de configuration/maintenance

❌ Désavantages possible

- Latence 🕐: temps de lancement à prendre en compte (il existe des solutions)
- Puissance limitée 🛴: conçu pour des traitements courts et légers (surtout fonctions)
- Supervision et débogage 🚑: si plusieurs services communiquent entre eux difficile de comprendre ce qui se passe.
- Coût 💸: ils peuvent exploser si on ne fait pas attention
- Enfermement 😈: techno propriétaire

Permet de se concentrer sur le métier. Attention par moment il faut soulever le capot.

AWS LAMBDA

- Compatible avec de nombreux langages (dont python)
- Configuration de la puissance, timeout, concurrence, variable d'environnement
- Pas de serveurs à gérer
- Facturation au temps d'exécution (ms)
- Environnement éphémère géré par AWS
- Temps d'exécution max : 15 min

Bout de code qui sera exécuté en fonction de déclencheurs (périodicité, requête HTTP, alerte etc)

AWS LAMBDA : EXEMPLE

```
import json
import os
print('Loading function')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))
    print("value1 = " + event['key1'])
    print(os.getenv("foo"))
    return event['key1']  # Echo back the first key value
```

- **lambda_handler** : la fonction qui sera appelée
- **event** : un dictionnaire avec les paramètres. Dépend du déclencheur
- **context** : des métadonnées (nom de la fonction, mémoire max, etc.)

CODE CDKTF

```
class LambdaStack(TerraformStack):
    def __init__(self, scope: Construct, id: str):
        super().__init__(scope, id)
        AwsProvider(self, "AWS", region="us-east-1")

        # Packaging du code
        code = TerraformAsset(
            self, "code",
            path="./lambda",
            type=AssetType.ARCHIVE
        )

        # Create Lambda function
        lambda_function = LambdaFunction(self,
                "lambda",
                function_name="first_lambda",
                runtime="python3.8",
                memory_size=128,
                timeout=60,
                role="arn:aws:iam::147552475298:role/LabRole",
                filename= code.path,
                handler="lambda_function.lambda_handler",
                environment={"variables":{"foo":"bar"}}
            )
app = App()
LambdaStack(app, "cdktf_lambda")
app.synth()
```

AWS SQS : SIMPLE QUEUE SERVICE

- Service pour avoir une queue d'événements (~FIFO)
- Peut déclencher une lambda ou servir d'output
- Pas de serveurs à gérer
- Facturation au nombre d'événements

Permet de faire le lien entre différents services. On parle de découplage

AWS SQS : SIMPLE QUEUE SERVICE

```
{
    "Records": [
        {
        "messageId": "19dd0b57-b21e-4ac1-bd88-01bbb068cb78",
        "receiptHandle": "MessageReceiptHandle",
        "body": "Hello from SQS!", // <<==== event ici, souvent un objet json (clé:valeur)
        "attributes": {
            "ApproximateReceiveCount": "1",
            "SentTimestamp": "1523232000000",
            "SenderId": "123456789012",
            "ApproximateFirstReceiveTimestamp": "1523232000001"
        },
        "messageAttributes": {},
        "md5OfBody": "{{{md5_of_body}}}",
        "eventSource": "aws:sqs",
        "eventSourceARN": "arn:aws:sqs:us-east-1:123456789012:MyQueue",
        "awsRegion": "us-east-1"
        }
    ]
}
```

AWS SQS + LAMBDA

```
class LambdaStack(TerraformStack):
    def __init__(self, scope: Construct, id: str):
        super().__init__(scope, id)
        AwsProvider(self, "AWS", region="us-east-1")
        # Packaging du code
        code = TerraformAsset(...)
        # Create Lambda function
        lambda_function = LambdaFunction(...)
        # Create SQS queue
        queue = SqsQueue(
            self,
            "queue",
            name="input_queue",
            visibility_timeout_seconds=60
        )
        # Link SQS as Lambda's source
        LambdaEventSourceMapping(
            self, "event_source_mapping",
            event_source_arn=queue.arn,
            function_name=lambda_function.arn
        )
app = App()
LambdaStack(app, "cdktf_lambda")
app.synth()
```

LIEN ENTRE SQS ET LAMBDA

```python
import json
import boto3
from datetime import datetime
sqs = boto3.client('sqs')  #client is required to interact with sqs

def lambda_handler(event, context):
    # event provenant d'une file SQS
    data = int(json.loads(event["Records"][0]["body"])["data"])

    sqs.send_message(
        QueueUrl="https://sqs.us-east-1.amazonaws.com/675696485075/lab-output-queue",
        MessageBody=json.dumps({"body" : data})
    )
    return {
        'statusCode': 200,
        'body': data
    }
```

TP AWS LAMBDA

Objectif

- Créer une fonction Lambda simple qui se déclenche périodiquement via la console
- Créer une fonction Lambda qui somme 2 nombres provenant d'une queue SQS, puis écrit le résultat dans une autre queue SQS