

TP 3 - Lambda

Mise en place


Allez sur la plateforme AWS academy et accédez au cours AWS Academy Learner Lab [36853]. Puis cliquez sur `Modules > Learner Lab`. Lancez votre environnement en cliquant sur `Start Lab`. Une fois le cercle passé au vert, cliquez sur `AWS Details` et `AWS CLI`. Les clefs que vous voyez vont permettre un accès programmatique à votre compte.

Ouvrez un terminal et exécutez la commande `aws configure`. Un prompt va vous demander votre AWS Access Key ID, collez la valeur de `aws_access_key_id`. Faites de même pour la Secret Access Key. Pour la région par défaut entrez "us-east-1". Validez le dernier prompt.

Ma première lambda

Définition de la Lambda

Une fois sur la console AWS, cherchez le service `Lambda` dans la barre de recherches. Sur le dashboard Lambda cliquez sur `Créer une fonction`. Laissez l'option `Créer à partir de zéro` cochée, donnez un nom à votre fonction lambda, et pour le langage d'exécution sélectionnez `python3.9`. Conservez l'architecture `x86_64`, et dépliez `Modifier le rôle d'exécution par défaut`, sélectionnez `Utiliser un rôle existant` et sélectionnez le rôle `LabRole`. Créez votre fonction

 À la différence des instances EC2, une fonction Lambda a besoin d'un rôle pour fonctionner. Sans rentrer dans les détails un rôle va déterminer les droits de la fonction. Comme votre compte n'a pas le droit de création de rôle, vous ne pouvez pas créer un rôle à la volée, et il faut sélectionner le rôle `LabRole` déjà créé.

Une fois sur la page de votre fonction, un code de base est proposé par AWS. Ce code retourne simplement un code 200 et le texte `Hello from Lambda!`. Vous allez lancer cette fonction via le bouton `Test`. Créez un nouvel événement de test, que l'on va appeler `test_basique`, et laissez le json de base. Votre événement de test sera un simple json de la forme


```
1 {  
2     "key1": "value1",  
3     "key2": "value2",  
4     "key3": "value3"  
5 }
```

Enregistrez-le et cliquez de nouveau sur `Test`. Normalement tout devrait bien se passer.

Maintenant vous allez légèrement modifier la fonction. Au lieu de juste retourner une chaîne de caractères fixe, elle va retourner l'heure actuelle. Importez la classe `datetime` du module éponyme et utilisez la méthode `now()` sur la classe `datetime`. Comme votre fonction a été modifiée, il faut la redéployer avec le bouton `Deploy`. Une fois fait testez-la de nouveau pour voir si tout fonctionne.

Ajout de l'invocation toutes les minutes

Sur la page de votre lambda vous allez cliquer sur `+ Ajoutez un déclencheur`. La source va être `EventBridge`. Créez une nouvelle règle avec le nom `minuteur`. Le type de règle sera `Expression de planification` et l'expression `rate(1 minute)`.

 EventBridge permet de gérer les événements comme des alarmes quand un seuil est dépassé, mais aussi les événements planifiés.

Votre fonction sera désormais appelée toutes les minutes. Malheureusement, comme il n'y a pas de destination pour votre fonction, les résultats disparaissent dans le néant du cloud. Il est bien possible de voir qu'elle est invoquée en allant sur l'onglet `Surveiller` puis `Journaux`. Vous allez voir une ligne par minute mais comme notre fonction de log rien, vous n'allez voir aucun résultat.

Poussez les résultats dans une file SQS

Maintenant vous allez faire en sorte que votre fonction envoie ses résultats dans une file SQS. Cherchez le service SQS et créez une file. Elle sera du type Standard et donnez lui le nom que vous souhaitez. Gardez toutes les valeurs par défaut et créez votre file.

Une fois votre file créée, retournez sur la page de votre Lambda et cliquez sur `+ Ajouter une destination`. Gardez l'option `Appel asynchrone`, pour la condition sélectionnez `En cas de réussite`, pour le type de destination `File d'attente SQS` et enfin sélectionnez votre file. Validez et attendez quelques minutes (allez vous aérer l'esprit ou prendre un café).

Retournez sur la page de votre file, et cliquez sur `Envoyez et recevez des messages` puis sur `Recherchez des messages`. Vous devriez voir des messages apparaître. Cliquez sur un et vous devriez voir un texte similaire (le mien est formaté pour être lisible)

```
1  {
2    "version": "1.0",
3    "timestamp": "2023-03-09T15:07:17.795Z",
4    "requestContext": {
5      "requestId": "005132c9-348a-4ca1-85ab-3b838a991749",
6      "functionArn": "arn:aws:lambda:us-east-1:147552475298:function:myfunction:$LATEST",
7      "condition": "Success",
8      "approximateInvokeCount": 1
9    },
10   "requestPayload": {
11     "version": "0",
12     "id": "a5f70280-4b36-d07e-681f-b253f0af2e9e",
13     "detail-type": "ScheduledEvent",
14     "source": "aws.events",
15     "account": "147552475298",
16     "time": "2023-03-09T15:07:00Z",
17     "region": "us-east-1",
18     "resources": ["arn:aws:events:us-east-1:147552475298:rule/rule_minute"],
19     "detail": {}
20   },
21   "responseContext": {
```

```

22     "statusCode":200,
23     "executedVersion":"$LATEST"
24 },
25     "responsePayload":{
26         "statusCode":200,
27         "body":"\"2023-03-09 15:07:17.686040\""
28     }
29 }

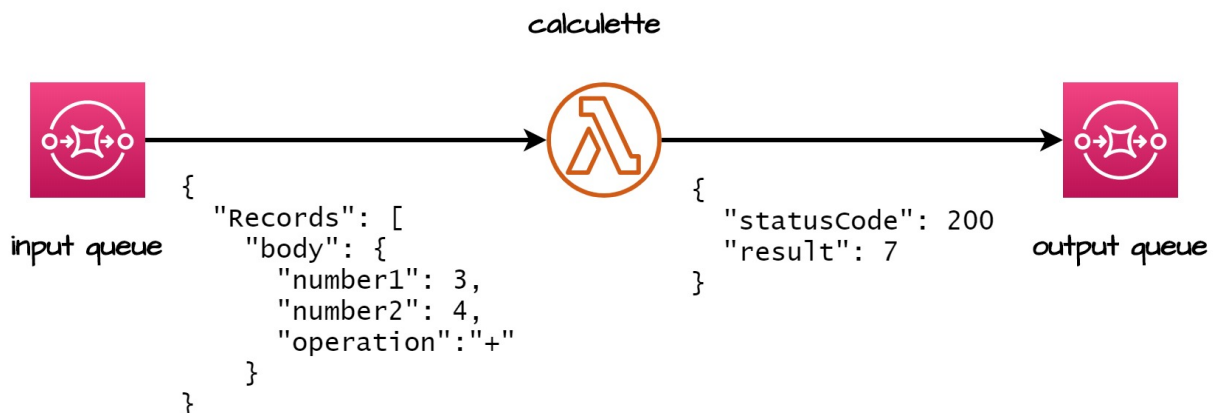
```

Notre lambda va mettre son résultat dans `responsePayload`, et on voit bien que le résultat.

🎉 Félicitation vous venez de mettre en place une architecture 100% serverless qui va réaliser un traitement toutes les minutes et pousser le résultat dans une file pour être utilisé par un autre service par la suite. Même si le code python du traitement est assez ridicule, l'architecture elle ne l'est pas.

🧮 Une calculatrice

Maintenant vous allez réaliser une calculatrice en utilisant une fonction lambda. Voici le schéma d'architecture globale.



Vous allez devoir créer :

- Deux files SQS, une pour l'input et une pour l'output
- Une fonction Lambda qui va aller chercher les clefs `number1`, `number2` et `operation` et faire le calcul demandé.

🎉 Félicitation vous venez de mettre en place une architecture 100% serverless avec trois services qui communiquent les uns les autres. Mettre des files entre des services permet de découpler les services et d'avoir un système plus modulable. Par exemple dans notre cas, notre lambda ne sait pas d'où proviennent les données, elle sais juste les prendre depuis une file. Ainsi la source des données peut changer, du moment que la nouvelle source alimentera la file SQS il n'y aura pas de raison de changer la lambda. De la même façon, notre lambda ne se préoccupe pas du service qui va utiliser les données qu'elle produit. Elle les pose simplement dans une file pour qu'un consommateur puisse les chercher. Les files SQS agissent comme des zones tampon entre les services.

S'il vous reste du temps pendant le TP commencez le TP noté.