Qu’est-ce que « Serverless »?

In fact, *serverless computing* simply means that you, the developer, do not have to *deal with* the server. A serverless computing platform like AWS Lambda allows you to build your code and deploy it without ever needing to configure or manage underlying servers. Your unit of deployment is your code; not the container that hosts the code, or the server that runs the code, but simply the code itself. From a productivity standpoint, there are obvious benefits to offloading the details of where code is stored and how the execution environment is managed. Serverless computing is also priced based on execution metrics, so there is a financial advantage, as well.[1]

Qu’est-ce que FaaS?

Sometimes serverless computing is called Function as a Service (FaaS), because the granularity of the code that you build is a *function*. The platform executes your function on its own server and orchestrates the process between function requests and function responses.

Here's the serverless execution model in a nutshell:

1. A client makes a request to the serverless computing platform to execute a specific function.
2. The serverless computing platform first checks to see if the function is running on any of its servers. If the function isn't already running, then the platform loads the function from a data store.
3. The platform then deploys the function to one of its servers, which are preconfigured with an execution environment that can run the function.
4. It executes the function and captures the result.
5. It returns the result back to the client.

Theses functions are *stateless*. That means they are taking input and generate output without cache or memorystack. Each instance of a FaaS is destoyed after usage.

Architechture

Nanoservices and Microservices

Like (Haines, 2018) said, « The term "nanoservices" is not an industry recognized term, but the idea is simple: each nanoservice should implement a single action or responsibility.”

[1] https://www.javaworld.com/article/3210726/serverless-computing-with-aws-lambda.html

As you can see in the next pictures, the idea of “nanoservices” represents an smaller entity then microservice.

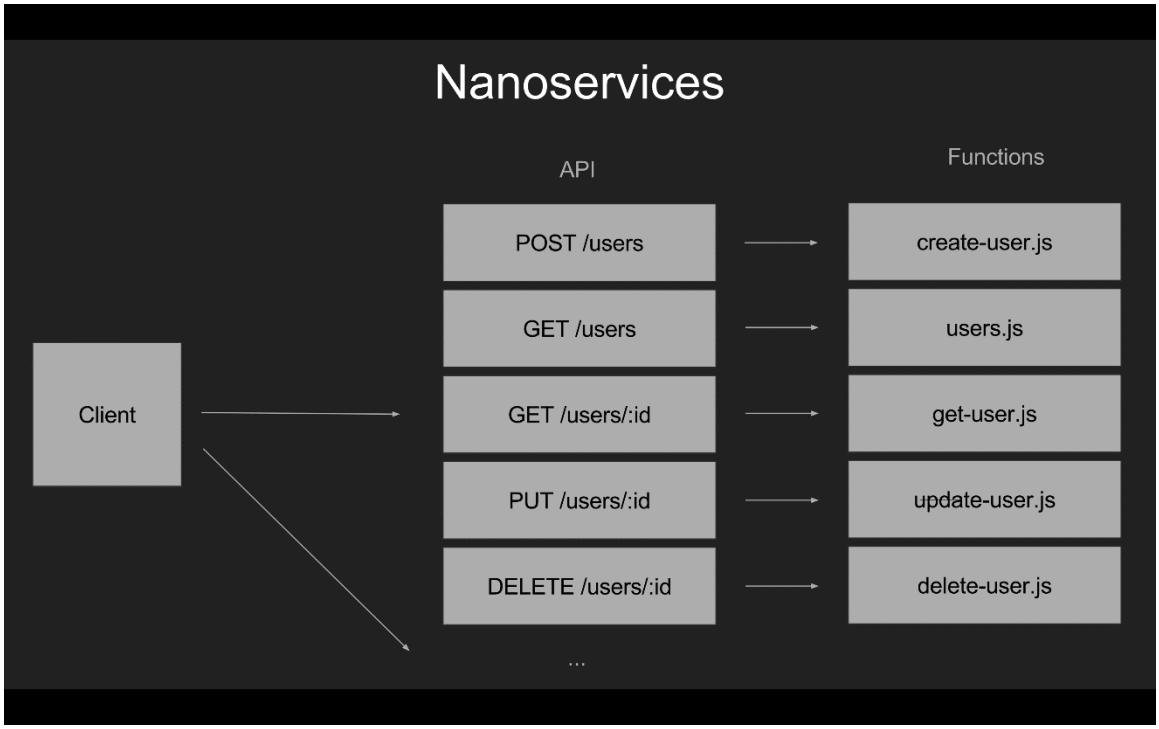

Figure 1 (Muens, 2016) Représentation d'un modèle Serverless avec des FaaS pour chaque opération CRUD..

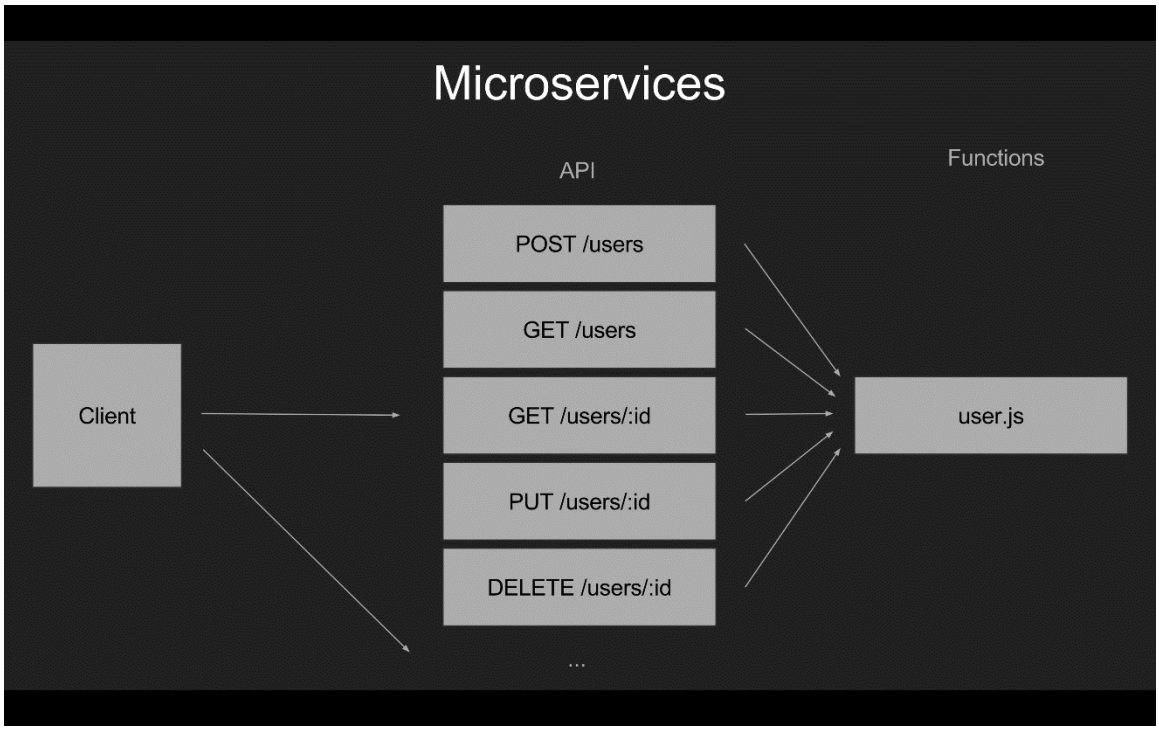

Figure 2 (Muens,2016) Représentation d'un modèle Serverless avec Faas unique.

Quoi que les deux modèles soient modélisable en « Serverless », le découpage de chaque opération par API diminue le couplage et représente mieux la mentalité FaaS où la plus petite unité de code est une fonction.

Advantages

- Flexible and automatic scalability

  La structure des fournisseurs de service FaaS en environnement Serverless permettent l'exécution de plus de 3000 (aws.amazon.com) instances simultanées d'une fonction, à la demande. Une instance de fonction n'est pas détruite à la fin de l'opération. Celle-ci reste active temporairement afin de répondre à un nouvel appel. Elle est par la suite détruite.

- High availability
  As said by Amazon : « AWS Lambda is designed to use replication and redundancy to provide high availability. » Based on (Petrillo, 2019), an Highly availability means a percentage of 99.999%. Wich reprensents an outage of 26 seconds per month.
- Pay per use (PPU)
  Amazon explain that : « Billing is metered in increments of 100 milliseconds, making it cost-effective and easy to scale automatically from a few requests per day to thousands per second. »
  Un avantage intéresssant si on considère le coût des infrastructures matérielles souvent nécessaire à de petit projet en startup.

  De plus, lorsque ces projets grossissent ou subissent des augmentations de fréquentation, il n'est pas nécessaire d'acquiérir du matériel supplémenaire.

https://aws.amazon.com/fr/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc

Amazon AWS

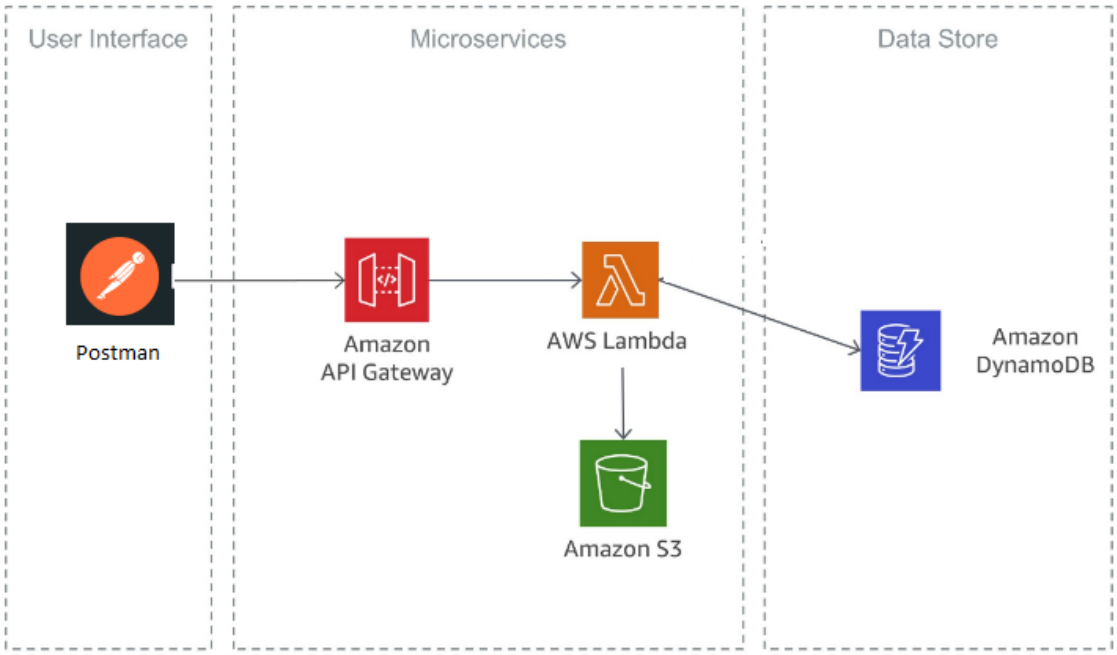

Figure 3: Serverless microservice using AWS Lambda

All definitions of figure 3 Amazons AWS services are available at aws.amazon.com serverless computing overview page that can be visit here.

In our tutorial, we will use :

- Amazon API Gateway
- AWS Lambda
- Amazon DynamoDB
- Amazon S3

Postman est un logiciel permettant d'effectuer des requêtes Post / Get afin de tester notre tutoriel

Policies and permissions

Afin de gérer la sécurité et les droits des divers composants de l'architecture Serverless AWS, il est important de comprendre grossièrement le fonctionnement de AWS Identity and Access Management (aka. IAM).

Les termes utilisés sont:

- Identities : users, groups of users, roles
- Ressources : AWS ressources
- Policy : Object in AWS that, when associated with an identity or ressource, defines their permissions.
- Permissions : Determine whether the request is alloweed or denied

La figure suivante exprime la relation de ces termes :