Serverless Computing, A revolution of Server?

KTH DevOps Course

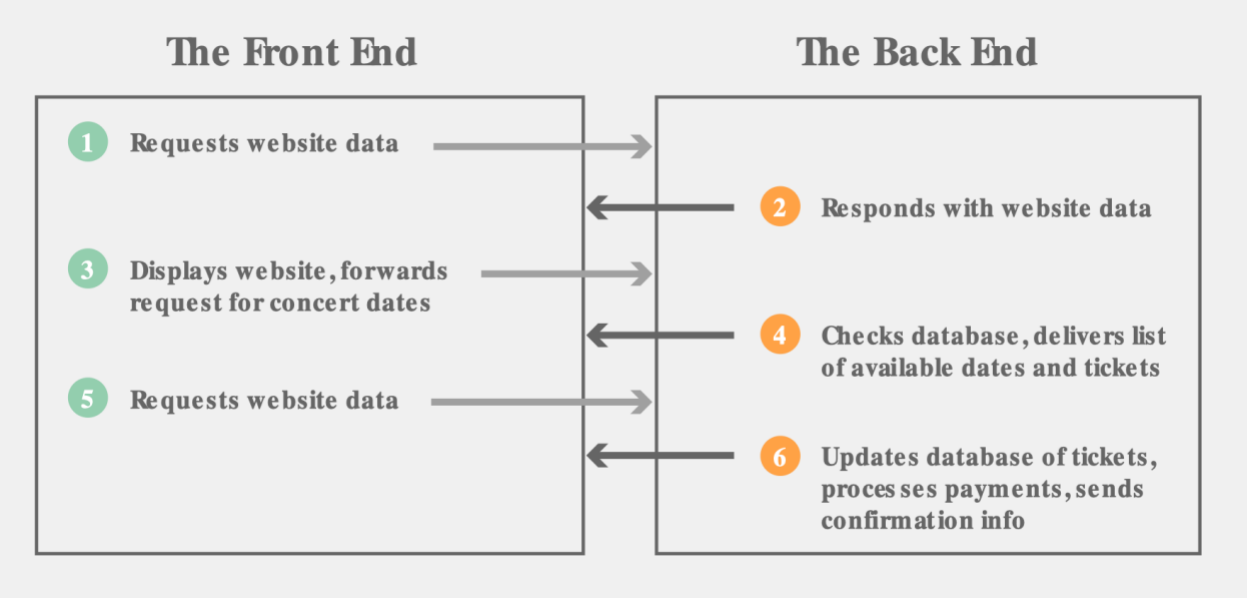
Dingli Mao dingli@kth.se

Introduction

Serverless computing has been a popular solution for running applications and services. It can be misleading to call it serverless. Serverless does not mean that the application is able to run without a server. By using serverless computing, the operation and maintenance of the server are managed by the cloud computing provider. The developers do not have the responsibility of any work which is related to the configuration and management of the server. Therefore, the developers of the application can focus more on programming. Serverless service accelerates the development and release of new applications. In this article, the advantages and disadvantages are presented to show its best use cases. Besides, the technical architecture and economic influence of serverless applications will be discussed. Finally, the usage of serverless computing in practical terms and the relationship with DevOps will be examined.

The definition of serverless

Serverless computing is the platform for the developers to host the back end service of their application. All the resources used by executing the back end of the application are dynamically allocated. Besides, the back end of the application is divided into many functions. Every serverless function can be triggered by an event, such as a request.



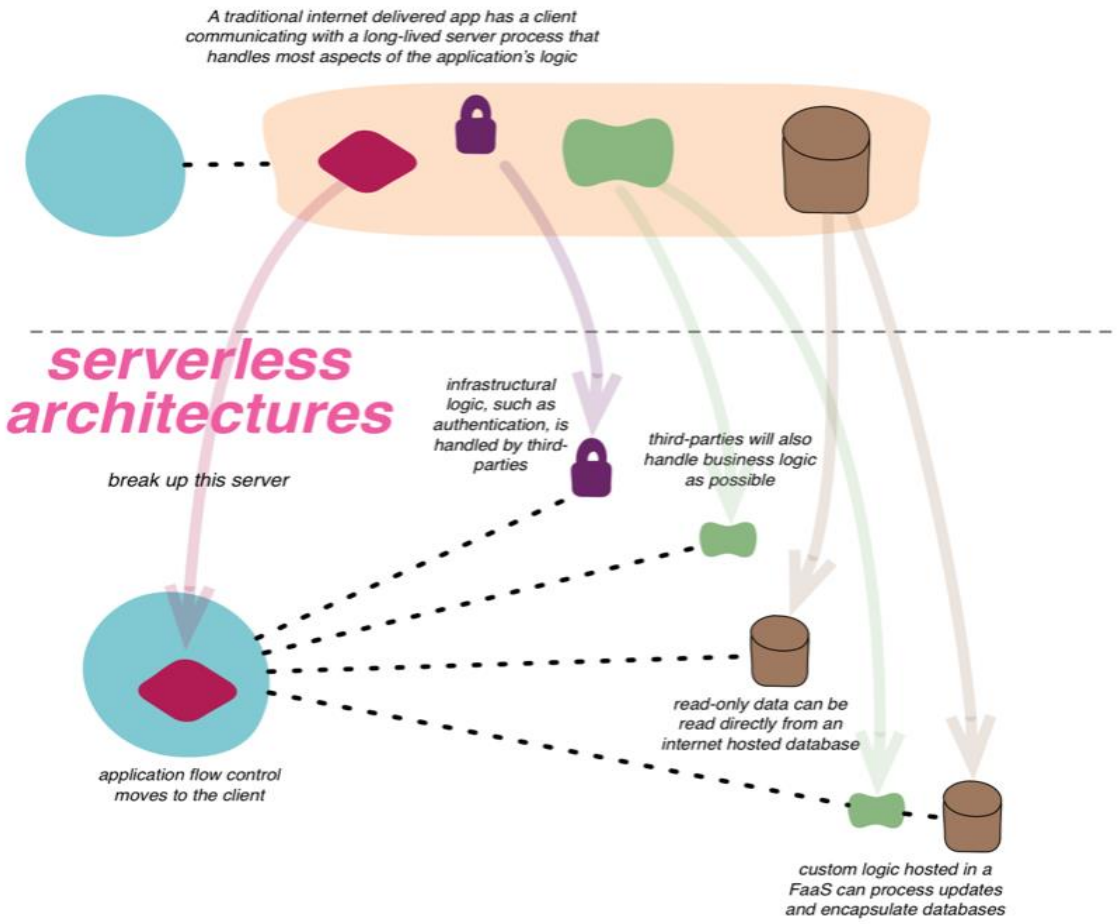
Pic 1 [1] Front-end and Back-end of a website

For example, a website can be divided into the front end and the back end.[Pic 1, 1] The front end is the user interface that the users can interact with. The back end stores the data which is related to the information of the website and users. While the user interacts with the website, the front end tends to send multiple requests to the back end in order to write or read the data. If the serverless computing is used, those requests trigger the functions of the back end. Only if those functions are triggered and executed, the service providers will charge the developers of the website. Once the executions of all functions are finished, the bill from serverless provides is stopped until the new event triggers the functions.

The comparison between Serverless and Traditional Architecture

The advantages of serverless over Traditional Architecture

Many industries have migrated their service from traditional architecture to serverless computing. Numerous advantages while using serverless computing motivate them to perform such migration. First, the developer can delegate the duty of managing the server to the provider of serverless. It means that the developers do not have to worry about the distinct running environment of the service while writing the program. The developers only require to maintain the connections among different third-party services such as authentication and database [Pic2, 2]. Those third party services are running individually. The developer is able to use them by accessing the API of the service only when it is needed. However, the workflow of developing the application in the traditional server is quite different. All the services are keeping running on one or several servers even they are not currently demanded. The developers have to deal with the connection and integration of services by themselves. The development of the application and the operations of the server are both the responsibilities of the developers.



Pic 2[2] The Application in traditional architecture and Serverless Computing

Besides, serverless computing offers high flexibility. The required resources used by the application are dynamically allocated. If the application tends to consume more resources, such as disk space, memory space, the serverless computing will allocate more resources to match the commands of the application. The billing model of serverless computing demonstrates its flexibility as well. The user is merely charged for the used resources while the application is executed. Unlike using a server, the users have to pay even if the server is idle. It is worth mentioning that many serverless providers offer a limited usage of their service for free. Such a trial might be enough to satisfy the needs of a small website. Nevertheless, the resources and billing are fixed if using a traditional server. The services which are related to an application always run on the traditional architecture. Even the service is not used, it will still occupy the resources of the serves. The developers will be charged the same amount of money regardless of the running time of the services. Thus, the cost of using traditional architecture will be higher than that of using serverless computing.

The disadvantages of serverless over Traditional Architecture

The serverless also brings several disadvantages to the developers. The developers are recommended to carefully consider the restriction of serverless computing before deciding to migrate to it. The application is most likely to be bound to a serverless provider. Because the

architecture and API used by the serverless providers are different. The developers have to invest a lot of time in revising the program before migrating to another serverless provider.

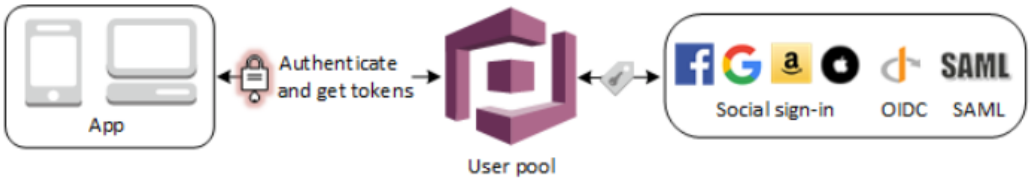
In addition, serverless providers restrict the size of the functions and time of execution. The main reason is that the providers try to balance the usage between different customers to avoid the situation that many applications are negatively influenced because a specific application occupies too many resources. Therefore, serverless computing may not be suitable for the application which demands high resources because of the limitation of space, time, and other resources and the difficulties of monitoring.

Monitoring and testing are other challenges while using the serverless platform. Since it is not the developers' duty of maintaining the server, they do not have access to the server anymore. The developers heavily depend on the monitoring tool offered by serverless providers to speculate about the potential problem in their application. Unlike programming and deploying the application in the traditional servers, most monitor and debug tools are unavailable to the developers if they choose serverless computing. [3] The basic monitoring tool is hard to satisfy programmers' demands for locating the problem. Moreover, the running environment of the serverless platform is hard to be replicated if the developers choose to debug locally. It is highly encouraged for the serverless vendors to offer open APIs so that other companies can develop more efficient tools for testing and debugging.

In contrast, the developers have the freedom of integrating different third party service with their applications. They will not be bounded with a server as because it takes much less time to migrate their application to another server provider. Once they migrate their application to another server, it is most likely that they do not have to rebuild their application with third party services. Moreover, since the developer and operations engineers have the complete authorization of utilizing the traditional server, the testing, monitoring and debugging are much more convenient.

The state-of-the-art of serverless

As serverless computing is a new choice for the developers to deploy and maintain their applications, many companies such as Amazon, Microsoft have published their own serverless service. Furthermore, they have implemented new features on their serverless architecture. For example, Amazon has connected Lambda, serverless service of Amazon, to AWS Cloud9 integrated development environment [4]. Therefore, AWS Cloud9 enables developers to import the functions of serverless computing. Such a connection between serverless computing and IDE relieves or even eliminates the difficulties of running and debugging a function which was written for serverless platform in user's developing platform.



Pic 3 [5] AWS UserPool for social sign-in and authentication

In addition, the providers of serverless computing release several features to enhance the security of users' applications. AWS Cognito UserPools [Pic3, 5] and Azure B2C are the identity solution by Amazon and Microsoft. By utilizing the service, the developers enable their application to have the function of allowing the users to sign in directly or using a third-party account. It is much easier for the developers to securely manage user's profiles and authenticate the attempt to log in or create an account.

The comparison between BaaS and FaaS

There are two architectures of serverless computing. They are backend as a service (BaaS) and function as a service (FaaS).

BaaS

BaaS is a model of serverless computing that depends on third party software and applications[6]. By using backend as a service, the developers are merely required to maintain the frontend of their applications. Instead of writing the program of the backend, the provider of BaaS offers services such as authentication, encryption, and database management which are mandatory functions to the backend of an application. By accessing the API of serverless providers, it is convenient and efficient for the developers to connect those services to their application. BaaS is commonly used by the programmers when they build a mobile application. The most popular BaaS platforms are Firebase and Parse.

FaaS

Function as a service means that the application is divided into several functions. These functions are developed, executed, and managed while using FaaS. The developer has a higher level of control over their application since they still have to write the backend of their application. For instance, the programmers are required to write the functions for managing the database and sending messages. Nevertheless, the functions of the backend are running in a container. They are triggered by events. AWS Lambda, Google Cloud Functions, and OpenFaas are the FaaS platforms that are widely used.

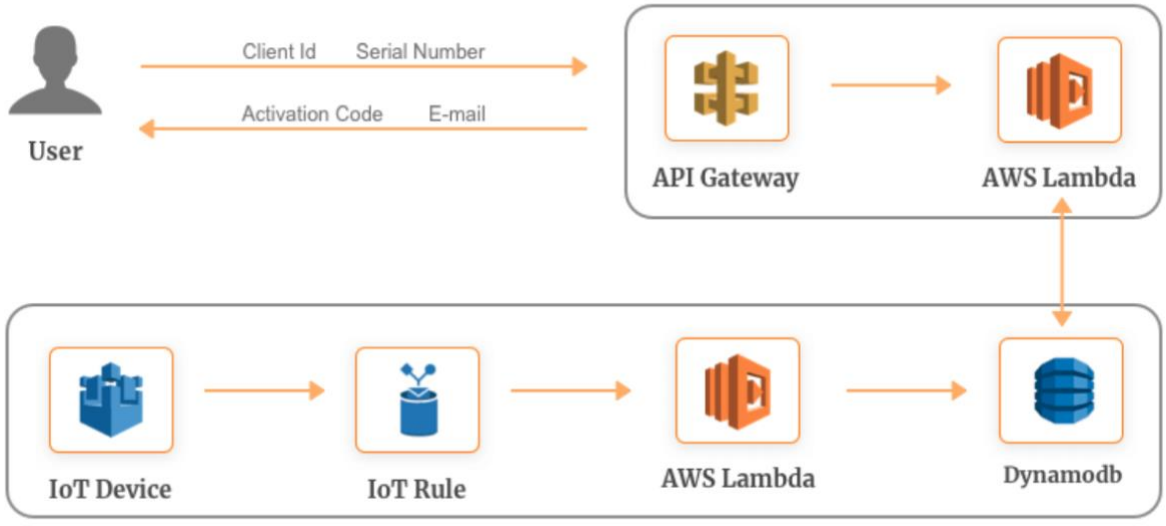
The connection between BaaS and FaaS

BaaS and FaaS are two architectures of serverless. They have different use cases. Meanwhile, it is possible for the developers to use both BaaS and FaaS to maintain their application. Amazon

API Gateway, Cogito, Dynamo DB [7]are the typical Backend as A service. They offer the purposes of creating, maintaining APIs, sign-in, sign-up account, and database service. BaaS tends to offers users the basic features of an application. The programmers can integrate one or more BaaS with their application. Thus, they do not have to write the program to implement features by themselves. On the other hand, Azure Functions and Amazon lambda are the representative of Function as a Service. While using FaaS, the developers have to write the codes for implementing the features. The codes will be triggered by distinct events. By using serverless computing, the developers can build their application by using both BaaS and FaaS. If a feature is provided by BaaS, the developers can use it instead of writing a new program. For the interaction between different features or components of an application, the developers can maintain the connection between them by utilizing FaaS.

Use cases of serverless

In the above section, the advantages and disadvantages of serverless computing have been analyzed. The difference and connection between BaaS and FaaS have been discussed. Therefore, it is easier to conclude that not all applications are suitable to run on serverless service. Thanks to the characteristics of serverless computing, it fits the requirements of many applications. Lambda is one of the most popular serverless computing provided by Amazon. Besides, Amazon also offers various services that can interact with serverless computing. Therefore, the following use cases are investigated based on Lambada.



Pic 4 [8] IoT device with Serverless computing

The developers may choose to use serverless computing if the application is frequently used in a specific period and rarely used in other periods. The back end of IoT devices [Pic4,8] can be deployed on serverless computing. Once the user creates the new IoT rules, it triggers the execution of the Lambda function and stores the data in the Dynamodb database. It is worth

mentioning that the products of a smart home device manufacturer are mainly installed in the house of the users. Thus, the users operate their IoT device using API Gateway more frequently in the morning and the evening when they stay at home than at any other time. Since serverless computing has high scalability, more resources can be allocated to the back end of IoT devices when they are heavily used by the customers. In the rest time of the day, the resources allocated to IoT devices are reduced to match the minimal demands from the customers. The high scalability and pay-as-you-go are the key characteristics to reduce the costs of the developers and meanwhile ensure IoT devices to utilize the resources it demands.

Pic 5 [9] The website deployed on serverless architecture

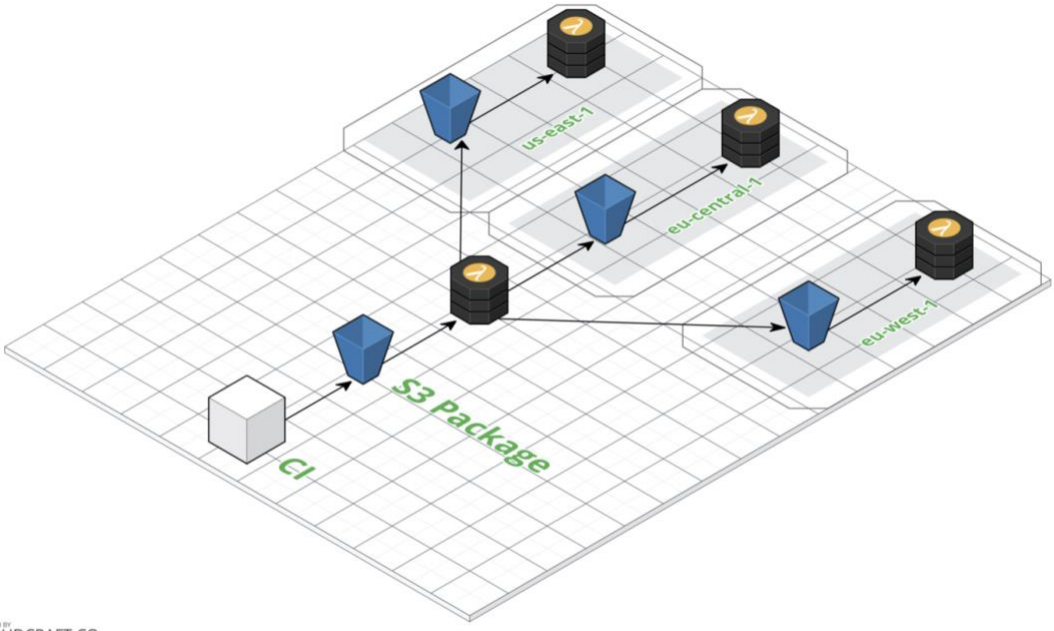
Many websites[Pic 5,9], especially social media, allow the user to access pictures, videos, and other multimedia. The short waiting time of viewing the multimedia greatly improves user experience. However, the pictures and videos have to be processed such as resize and transcode before they match the user's device. If the application is integrated with AWS lambda, a user request which is a lambda will first trigger Amazon CloudFront to check if the requested file exists in the cache. If it does not exist, it will trigger another lambda request to access the file in Amazon S3 to acquire the files. Such an approach of processing user requests definitely reduces the user's time before they are able to view the multimedia. It is easy for the developers to implement this approach because one of the key features of serverless computing is that the event triggers the execution of the functions. In contrast, it is quite difficult to efficiently process multimedia on conventional architecture as the developers have to spend much time dealing with the interaction among different servers.

Serverless and DevOps

DevOps is a concept which integrates a series of technology and tools in order to obtain high automation of developing, testing and deploying. It eliminates the boundary between operation and development. In other words, DevOps aims to turn all the works from development to deployment to be automatic. Therefore, it can increase the efficiency of developing the application and shorten the time of deploying it. Serverless computing perfectly matches the idea of DevOps. If a company chooses to use serverless computing instead of the traditional server, the developers is able to invest more time in a program that was used to do the operation of a

traditional server. It does not mean that serverless computing does not require operations. In contrast, it is more difficult to cut off the connection between development and operation. While deploying the applications on the serverless platform, the developers are required to maintain the basic knowledge of operations as well. On the other hand, the operations engineers are supposed to learn programming as well to deal with the operations of the serverless platform. For solving the problem or improving the application, the developers and operations engineers have to actively cooperate.

Continuous integration and continuous delivery are two significant stages in DevOps. Continuous integration means the developers test their code once they implement a feature or fix a bug. After that, their branch will be merged with the master branch if all tests passed. This process is totally automatic. Continuous deployment represents an automatic approach of deploying the program to different production environments. It is possible to implement CI and CD tools by only using serverless computing. For example, it is possible to integrate the functions of CI and CD with AWS lambda[Pic6, 10]. To be more specific, AWS Lambda provides “sls package” and “sls deploy ” command to package your program and then deploy it [11]. After the software passed all the test, different packages which are suitable to the distinct running environment are generated and stored in Amazon S3 cloud storage server. Since serverless computing is event triggered, S3 cloud storage service is configured to trigger the function of Lambda once a new package is ready to be launched. To efficiently deployed to the client in various regions, the lambda function will be used to distribute the packages to different storage servers in various regions. Those storage services will trigger a lambda function to deploy the package immediately once it received. Consequently, the perfect collaboration between serverless computing and CI/CD brings great convenience to the developers while they developing, testing, and deploying their application.



Pic 6 [11] CI and CD with AWS lambda

Conclusion

Serverless computing can be considered as a revolution of traditional servers. Because of its high flexibility of billing and scalability of resources, the economic cost can be saved if the industry chooses to replace it with the conventional servers and embrace DevOps concept. A serverless framework improves efficiency while developing the program as well. Instead of forcing the developers to worry about the operation of the server and compatibility of running environment, implementing CI/CD tools on the serverless platform helps them to accelerate the speed of development and achieve higher stability of the software.

Reference:

1. Cloudflare, “What is serverless computing?”, available at https://www.cloudflare.com/learning/serverless/what-is-serverless/
2. Badri Janakiraman, “Serverless” (2016). Available at https://martinfowler.com/bliki/Serverless.html
3. Baldini, Ioana & Castro, Paul & Chang, Kerry & Cheng, Perry & Fink, Stephen & Ishakian, Vatche & Mitchell, Nick & Muthusamy, Vinod & Rabbah, Rodric & Slominski, Aleksander & Suter, Philippe. (2017). Serverless Computing: Current Trends and Open Problems. 10.1007/978-981-10-5026-8_1.
4. Amazon, “Working with AWS Lambda Functions in the AWS Cloud9 Integrated Development Environment (IDE)”. Available at https://docs.aws.amazon.com/cloud9/latest/user-guide/lambda-functions.html
5. Amazon, https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html”. Available at “https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html
6. Cloudflare, “What is BaaS? | Backend-as-a-Service vs. Serverless”, available at https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/
7. Joab Jackson and Lawrence E Hecht,“TNS Guide to Serverless Technologies: The Best of FaaS and BaaS” Available athttps://thenewstack.io/guide-serverless-technologies-functions-backends-service/ (2016)
8. Simform, “10 AWS Lambda Use Cases to Start Your Serverless Journey” available at https://www.simform.com/serverless-examples-aws-lambda-use-cases/
9. Simform, “10 AWS Lambda Use Cases to Start Your Serverless Journey” available at https://www.simform.com/serverless-examples-aws-lambda-use-cases/
10. . Rupak Ganguly, “Automating CI/CD workflow for serverless apps with CircleCI”, available at https://www.serverless.com/blog/ci-cd-workflow-serverless-apps-with-circleci/
11. Rupak Ganguly, “Automating CI/CD workflow for serverless apps with CircleCI”, available at https://www.serverless.com/blog/ci-cd-workflow-serverless-apps-with-circleci/