UDC 004.725.4

FEATURES OF A SERVERLESS ARCHITECTURE FOR BUILDING DATA PIPELINES

Svirnovsky A.V.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Alexeev V.F. – PhD of technical sciences, Assistant professor

Annotation. If you have been in the industry for a long time, or are simply interested in modern architectural approaches, you have most likely heard about a new way of running applications in the cloud environment, called Serverless.

Keywords. Serverless, AWS, amazon, microservice, scalability, infrastructure.

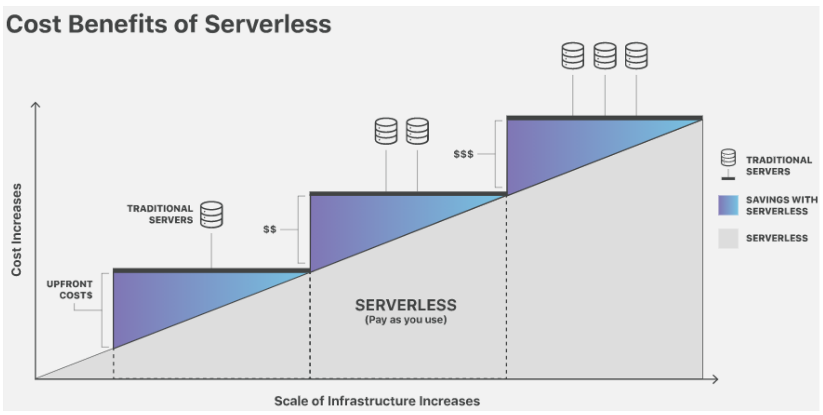
Introduction. Serverless computing is a method of providing backend services on an as-used basis. A serverless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company that gets backend services from a serverless vendor is charged based on their computation and do not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Note that despite the name serverless, physical servers are still used but developers do not need to be aware of them [1]. Serverless architecture is especially important for building data processing pipelines due to its ease of customization.

Main part. *Function as a service (FaaS*) is another thing that is linked with Serverless.

FaaS is a cloud computing model that enables cloud customers to develop applications and deploy functionalities and only be charged when the functionality executes. FaaS is often used to deploy microservices and may also be referred to as serverless computing [2].

These definitions summarize the three main features of what is called Serverless:

- abstraction. You do not control the server that runs your program. You do not know anything about it at all, all the nuances of the operating system, updates, network settings and other things are hidden from you. This is so that you can focus on developing useful functionality rather than administering the servers;
- elasticity. The Serverless service provider will automatically provide you with more or less computing resources, depending on how heavy the load is on your application;
- effective cost. If your application is idle, you do not pay anything. It does not use computing resources at this moment. Payment is made only for the time that your application works;
- limited lifecycle. Your application is launched in a container, and after a short time, from ten minutes to several hours, the service automatically stops it. Of course, if the application needs to be called again, the new container will be launched.



Picture 1 – Cost Benefits of Serverless

Areas of use. With some assumptions, the Serverless model can be used anywhere. However, there are several cases with which it is easier and safer to start using it.

These are cases of deferred, or background tasks. For example:

- creating additional copies of the image after uploading it to the site;
- creating a backup on a schedule;
- asynchronous notification to the user (push, email, sms);
- various exports and imports.

All these examples either run on schedule or do not imply an instant response to the user. This is since applications (functions) in the Serverless model do not work all the time, but are launched as needed and, if not used, are automatically disabled. This leads to the fact that it takes time for the function to start, sometimes up to several seconds.

One of the leaders in the FaaS market these days is AWS with its AWS Lambda service. They have support for a wide variety of programming languages (including Ruby, Python, Go, NodeJS, C # and Java) and a huge number of services that allow you to use not just serverless computing, but also database as a service, message queues, APIs gateway and others that simplify the work with this model.

In addition to AWS, it is worth noting Google Cloud and Microsoft Azure with their Google Functions and Azure Functions products.

For the Serverless approach, you can find many disadvantages and difficulties that you will have to face. Most of them follow from those for any distributed system.

You should always be careful to maintain backward compatibility because another service / function could potentially depend on your interface or business logic.

The interaction patterns in a classic monolithic application and in a distributed system are very different. You need to think about asynchronous communication, possible delays, monitoring of individual parts of the application.

Even though your functions are isolated, the wrong architecture can still lead to cascade failure - when a mistake in one of them makes many others inoperable.

The price of great scalability is that until your function is called, it has not run. And when you need to launch it, it can take up to several seconds, which can be critical for your business.

Conclusion. A serverless approach offers developers, teams, and organizations a level of abstraction that enables them to minimize the time and resources invested in infrastructure management. Every component of an application benefits from this approach, from computing and the database engine to messaging, analytics, and AI. Using an end-to-end serverless platform that provides a comprehensive set of serverless technologies is the best way to ensure that the organization gains the maximum benefit from going serverless.

Bibliography.

1. SearchITOperations [Electronic resource]. – Access mode: https://searchitoperations.techtarget.com/.

2. CLOUDFLARE [Electronic resource]. – Access mode: https://www.cloudflare.com/.