Cloud & Serverless: Game Changer?

Taqui Syed Shah

March 2021

1 Introduction

DevOps defines principles and practices that automate the process of software development to build, test, and release software more reliably [9]. One aspect of software development is the underlying infrastructure it rests upon. That is the servers, the databases, and file storage, etc. Cloud is a technology used to simplify the creation, utilization, and scaling of an IT infrastructure. Cloud enables on-demand IT-related resources such as computing or storage which is maintained by the cloud provider.[10]. This enabled the infrastructure as a service(IaaS) model within software development. Recent developments in the cloud have led to a new model called Serverless. Serverless is an abstraction level above the cloud, this created the backend as a service (BaaS) and functions as a service (FaaS) models. "Serverless" is storage or computing where the developer avoids managing the underlying infrastructure.

This essay will explore how Serverless can be leveraged to DevOpsify infrastructure.

2 Serverless

The question now is "how can serverless be leveraged to DevOpsify an IT infrastructure?". Understanding the services that exist and how to replace traditional cloud resources will aid with the understanding of the benefits. In this case, AWS will be used as a base for discussing the benefits of a Serverless platform. This section will introduce different services and technologies within serverless and how they can replace traditional IT resources.

Grasping Serverless requires an understanding of the key metrics of the cloud. Therefore, the next subsection will define the key metrics for cloud infrastructure.

2.1 Metrics of Cloud

The main metrics of cloud infrastructure are availability, scalability, and elasticity. Availability refers to the up-time of software. Scalability refers to how well the software allocates new resources depending on the load of the application. Elasticity refers to the application's ability to increase and decrease the resources depending on the application load.

There are two types of scalability within Cloud, vertical scalability vs horizontal scalability. Vertical scalability utilizing s adding more processing power for example a CPU with 16 cores instead of 8 cores. Horizontal scalability is utilizing additional compute units for example leveraging 2 CPUs with 8 cores instead of a single CPU with 8 cores. [14]

To ensure high availability cloud providers have created regions and availability zones. Regions are physical locations where cloud providers cluster multiple data centers. Each cluster of datacenters is called an availability zone (AZ). This structure enables high availability within the cloud by enabling developers to host software on multiple AZs. [7]

2.2 Serverless Introduction

Serverless is used to describe services where the developer avoids managing the underlying infrastructure and where the service deals with scalability, availability, and elasticity. Serverless leverages servers however it is abstracted away from the developer. A misconception is that serverless is purely a FaaS(functions as a service) platform. Computing, storage, and databases are resources within the serverless sphere. [15]

The traits of Serverless services are:

- No server management
- Automatic and Elastic Horizontal Scaling
- High Availability
- Pay as you go model

2.3 Serverless Computing

FaaS or Serverless Computing is a service that enables server-side logic by using stateless compute containers that are managed by the cloud provider. The server-side logic is run in functions and is called serverless functions. The ease of scaling for serverless functions comes from the replication of the compute container. Serverless functions have a short life cycle, approximately 900s or 15 minutes[8]. FaaS is considered to be part of event-driven computing.[17] Serverless functions are only called based on an event whether that is based on time, an email, or an HTTP request. The stateless nature of Serverless functions forces uses other services for state. [15]

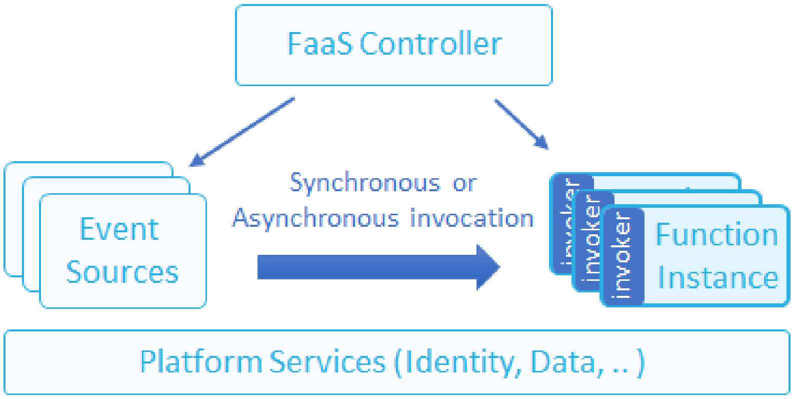
FaaS Controller
Event Sources
Synchronous or Asynchronous invocation
invoker
invoker
invoker
Function Instance
Platform Services (Identity, Data, ..)

Figure 1: Serverless Processing Model

This picture describes in detail the process of serverless computing. The event sources trigger events to create the function call. Function instances are the actual serverless functions. FaaS controllers deploy, control, and monitor function instances and their sources. Lastly, platform services describe the services leveraged by the serverless function. [6]

Serverless Computing can be used to replace virtual machines such as EC2 instances in AWS. [11]

In AWS autoscaling groups are leveraged to horizontally scale a group of EC2 instances. Serverless has automatic horizontal scaling for each serverless function, which enables ease of scaling. [5]

The pay-as-you-go model enables enormous cost-saving since the developer only pay per request rather than for each EC2 instance that is running[5]. This means that when the system is idle the entire infrastructure is free. Serverless functions are also free to a certain amount of requests, for a start-up the entire backend infrastructure can be free until it scales to the larger size.[4] [11]

2.4 Serverless Storage/Persistence

Storage refers to two different forms of storage, object-based and block-based.[18] Simply stated, object-based storage refers to files such as pictures, music, or large text documents. Block-based storage is used by the operating system or hosting databases. In the serverless sphere, storage is split in two ways Serverless Object Storage and Serverless Relational Databases. In this case, serverless object storage replaces classic object storage with EC2 instances and Serverless Databases replace block storage. [12]

2.5 Serverless Object Storage

Serverless Object Storage aims to deal with the use of object-based storage is used EC2 based applications. AWS offers a service called S3 which is serverless object storage. Serverless storage offers buckets which is the main way to store object-based storage. Security is an aspect of serverless storage that is important to mention since it can offer powerful access management through granular policy creation using JSON. This access management is called bucket policies, which manages access through AWS identity and access management service. [16]

In an application, Serverless Object Storage replaces certain block storage that is used in an application. EBS(Elastic Block Storage) is one of the main ways to generate block-based storage for an EC2 instance. Previously, applications needed to leverage EBS or EFS to store object-based storage. This also applies to before the cloud was used as people also used FTP servers for file storage. S3 solves issues like this by dealing with the possible complexities of building object storage infrastructure. Security is an aspect that enables S3 to be more efficient compared to EBS. For EBS an application layer is necessary for determining which have access to which resources, however with S3 using bucket policies enables access control management.

2.6 Serverless Databases

Cloud providers often provide both serverless relational databases and non-relational databases. [16]

One serverless relational database service is Aurora, and a common non-relational database service is DynamoDB.[1][2]

In AWS the most common way to have relational databases is using Amazon RDS which is used to create a SQL database of any flavor. However, developers have to manage backups, scaling using read replicas, and dealing with possible infrastructure failures and failover to a standby database.[3] If AWS Aurora is leveraged that would not be a problem since the service is dealt with availability by replicating to different AZs similar to S3.

For non-relational databases, DynamoBD would be appropriate, which deals with availability similar to Aurora.

3 Limitations Of Serverless

Serverless have huge benefits but there exist limitations.

3.1 State

In serverless functions, there is no state saved between each function invocation. The inherent nature of statelessness enables immense scalability. This means that serverless functions need to leverage some form of state using serverless services, for example, DynamoDB. This increases the functions' complexity and increases the latency for certain applications that could leverage local state. [13]

3.2 Latency and Cold Starts

There exist an inherent latency issue within Serverless. Traditional IT resources can be adjusted with great granularity and can reduce latency by using specialized network protocols, communication channels, or data formats. With Serverless that level of granularity disappears since the cloud provider becomes responsible for most of that interaction. Most communication in a Serverless context occurs over HTTP which can be slower than other transport protocols. This is also a problem for BaaS application since BaaS follow similar patterns for creating the underlying infrastructure. [13]

Cold Start is a common issue within the Serverless sphere. Cold start refers to an initial latency for calling the function for the first time, configuration changes, scaling of lambda functions, or when the lambda function has not been called for a duration. This latency occurs because the container running the lambda function has to be initialized and the code needs to initialize (for example downloading packages). Serverless functions that have to be initialized are called "cold" and therefore the problem is referred to as cold start. If a container has been initialized it handle lambda calls without undergoing the same initialization process. These functions can handle invocation much quicker and therefore these functions are called "warm". [13] This delay can be anywhere from 30 seconds to 1minutes depending on the application. Previously, lambda functions that were used in VPC the delay would be closer to 15 minutes for initializing a function. However, AWS has immensely towards reducing the cold start currently the average seems to be closer to 0.7s for a cold start. There also exists Provisioned Concurrency which deals with this by having a small serverless function constantly running. The problem is that this will incur additional costs for holding the function warm. This defeat negates some of the benefits of having a fully serverless infrastructure.

3.3 HPC and Long-Running Applications

In applications such as High-Performance Computing, the requirement for long-running applications can make it impossible to replicate into a serverless context, for example, Machine Learning or Deep Learning. In these workloads, training models can take huge amounts of time which require a lambda function to be constantly running which just is not possible with how the current model exists. Machine Learning requires local state to be leveraged and specific hardware such as GPUs. These features just currently do not exist in a serverless context. However, there exist papers that indicate that there exists application in scientific computing where some of the workloads can be reduced to smaller lambda functions.

4 Conclusion

Serverless and Cloud have a clear spot in DevOps since these technologies enable extreme automation by automating the infrastructure management of availability, scalability, and elasticity. Serverless has enabled enormous strides in simplifying the creation of software by completely abstracting the infrastructure which enables a developer to purely focus on the code rather than the underlying infrastructure the code rest on top of. There are clear issues with serverless such as no state, cold start and not being applicable for HPC. Even with the limitations, the future for Serverless is bright as it enables developers to build software with a simple and cheap infrastructure, enables more companies to scale with extreme simplification of the infrastructure, and reduces the cost.

References

[1] Amazon aurora. "https://aws.amazon.com/rds/aurora/", 2020. Accessed: 2020-05-27.

[2] Amazon dynamodb. "https://aws.amazon.com/dynamodb/", 2020. Accessed: 2020-05-27.

[3] Amazon relational database service (rds). "https://aws.amazon.com/rds/", 2020. Accessed: 2020-04-07.

[4] Aws lambda pricing. "https://aws.amazon.com/lambda/pricing/", 2020. Accessed: 2020-04-07.

[5] Aws lambda vs ec2: Comparison of aws compute resources. "https://www.simform.com/aws-lambda-vs-ec2/", 2020. Accessed: 2020-04-07.

[6] Cncf serverless whitepaper v1.0. "https://github.com/cncf/wg-serverless/blob/master/whitepapers/serverless-overview/README.md", 2020. Accessed: 2020-05-27.

[7] Regions and availability zones. "https://aws.amazon.com/about-aws/global-infrastructure/regions_az", 2021. Accessed: 2020-04-02.

[8] Serverless architectures with aws lambda: Timeout. "https://docs.aws.amazon.com/whitepapers/latest/serverless-architectures-lambda/timeout.html", 2021. Accessed: 2020-04-07.

[9] Christophe Capel. Devops: Breaking the development-operations barrier. "https://www.atlassian.com/devops", 2021. Accessed: 2020-04-02.

[10] T. Dillon, C. Wu, and E. Chang. Cloud computing: Issues and challenges. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 27–33, 2010.

[11] an NTT Data Company Flux7. Serverless vs ec2 vs containers: A comparative study. "https://cdn2.hubspot.net/hubfs/411552/Whitepapers/Whitepapers\%20in\%20the\%20sales\%20library/WP_\%20\%20Serverless\%20vs\%20EC2\%20vs\%20Containers_\%20A\%20Comparative\%20Study.pdf". Accessed: 2020-04-07.

[12] Gali Kovacs. Aws ebs and s3: Object storage vs. block storage in the aws cloud. "https://cloud.netapp.com/blog/block-storage-vs-object-storage-cloud", 2017. Accessed: 2020-04-07.

[13] John Chapin Mike Roberts. Chapter 4. limitations of serverless. "https://www.oreilly.com/library/view/what-is-serverless/9781491984178/ch04.html", 2020. Accessed: 2020-04-28.

[14] Matt Rasband and Eugene Ciurana. Scalability and high availability. "https://dzone.com/refcardz/scalability?chapter=1", 2017. Accessed: 2020-04-02.

[15] Mike Roberts. Serverless architectures. "https://martinfowler.com/articles/serverless.html#WhatIsServerless". Accessed: 2020-04-06.

[16] Rajind Ruparathna. What makes a storage service truly serverless? "https://dzone.com/articles/what-makes-a-storage-service-truly-serverless", 2019. Accessed: 2020-04-07.

[17] PIOTR SROCZKOWSKI. Cloud: Iaas vs paas vs saas vs daas vs faas vs dbaas. "https://brainhub.eu/blog/cloud-architecture-saas-faas-xaas/", 2021-01-11. Accessed: 2020-04-06.

[18] Tony Piscopo Yadin Porter de León. Object storage versus block storage: Understanding the technology differences. "https://www.druva.com/blog/object-storage-versus-block-storage-understanding-technology-differences/", 2019. Accessed: 2020-04-07.