DD2482 AUTOMATED SOFTWARE TESTING AND DEVOPS

ESSAY

# AWS INFLUENCE ON DEVOPS PRACTICES

By

ALBERT ASRATYAN, SIGRÚN ARNA SIGURÐARDÓTTIR AND JUSTAS DAUTARAS

KTH ROYAL INSTITUTE OF TECHNOLOGY
APRIL 2020

# TABLE OF CONTENTS

**Page**

**CHAPTER**

# Chapter One

# Introduction

Amazon with its Amazon Web Services is the world's leading cloud service provider, which has revolutionized software deployment and management. It provides more than 175 fully integrated services that range all the way from basic webpage hosting to complex serverless database management.

Even though this essay puts its focus on the Amazon Web Services (AWS from now on), most of the arguments presented here could be applied to similar cloud providers, such as Microsoft Azure, Google Cloud, Alibaba Cloud, and many others.

# Chapter Two

# Background

## 2.1 What is a Cloud Service?

Cloud computing is a term referred to storing and accessing data over the internet. It doesn't store any data on the hard disk of your personal computer. In cloud computing, you can access data from a remote server. So, in turn cloud services are just pieces of software that are kept on the internet, but used as if it was running locally. And Amazon is one of the leading companies providing these services. This allows companies not only to forget about the maintenance of the servers and focus on the software itself, but also many more benefits which are discussed below.

## 2.2 Benefits of Cloud

While there are many benefits of using cloud solutions, we would like to focus on some key benefits (guru99, 2018): cost savings, speed, reliability, mobility and back-ups. We have chosen these qualities due to the fact that cloud computing allows a company to give up employees in charge of maintenance of these systems and it is taken care of by the providing company for a much less expensive amount. Next, it is a lot easier and faster to deploy, run and back-up a system, because much of the difficulty is hidden behind layers of software. And

lastly, but not any less importantly, systems kept on the cloud can be accessed independently by any user from any location or any device. These features allow developers to focus on the development of the systems themselves and in turn is the basis for rapid development.

## 2.3 DevOps Practices

Most companies want to increase their competitiveness in today's swiftly changing world, and so they cannot ignore digital transformation. DevOps and cloud computing have become two of the ways companies can achieve this needed transformation, though the relationship between the two is not easily reconciled - DevOps is about the process and process improvement, while cloud computing is about technology and services. It's important to understand how the cloud and DevOps work together to help businesses achieve their transformation goals. We will discuss some common practices to better understand this (Ryadel, 2019).

### 2.3.1 Continuous Integration

Continuous Integration is a software development practice in which the teams of developers deploy their code changes into a central repository and that repository is then used for running automated tests and builds (Rehkopf, 2020). The goal of Continuous Integration is to smooth out the process of releasing software updates, as it can be hinted from the name.

The reason that Continuous Integration has even surfaced as a practice is the fact that before the rise of DevOps and CI specifically the software was developed by teams of engineers mostly in isolation until the final product was ready, and merged with the master only then. This has made product delivery rather slow and sluggish.

### 2.3.2 Continuous delivery

Continuous Delivery is the extension of Continuous Integration that makes sure that you can release the changes to product/customer quickly and efficiently. This mainly refers to implementing an automated release process on top of the automated testing from CI. Theoretically, CD allows you to adjust for daily, weekly or whatever other types of releases. But the best practice is to not stick to a specific release cycle but rather deliver small batches as soon as the product is ready to ensure that the released versions are easy to debug in case of a problem.

### 2.3.3 Microservices

Microservices is the binding element between different standalone programs. Microservices architecture can be seen as a design approach for building applications as a set of smaller services. Each service runs on its own and communicates with others in a designed manner, usually utilizing HTTP (and REST API) based communication. Usually microservices are built for extremely specific purposes, so usually it is a good idea to have a collection of them that performs a greater function (Nemer, 2019).

### 2.3.4 Infrastructure as Code

Infrastructure as Code is a practice that allows software developers to interact with the system's infrastructure programmatically instead of manual configurations and setups. The benefits of Infrastructure as Code is the ability to scale up easily. Therefore, engineers are able to work with the product's infrastructure using code-based tools directly without needing external resources and are also able to treat it in a similar way to the application code.

### 2.3.5 Monitoring and Logging

Monitoring and logging the course of development/operation using different sorts of metrics can have a big impact on what the next approach is going to be in order to improve user experience. Active monitoring is an extremely important aspect since nowadays many services can and have to be available without any downtime.

### 2.3.6 Communication and Collaboration

The key feature of the whole DevOps culture is increased communication and collaboration. All the DevOps software automation and other tools bring closer development and operations workflows in order to make the product release smoothly.

## 2.4 Benefits of DevOps

From the aforementioned cloud benefits and devops practices, we can see that DevOps greatly changes the way software is developed and deployed. Some of the key benefits are discussed below.

### 2.4.1 Speed

Speed of product development is one of the top factors that can define a good team of engineers. In the context of DevOps, it can boost the speed of development by being able to adapt faster to changes in the market or product requirements. Being able to adapt to certain changes in time can be a business critical ability, because this is what can make the difference between the business going down or being able to adjust just in time.

### 2.4.2 Rapid Delivery

Speedy development of a product is of course a great benefit to have. However, without good means of delivering the said product, its value may be totally zeroed. This is where the rapid delivery aspect of DevOps comes in. Bundled together with increased speed of development, faster delivery may help with meeting the client's needs faster. This is also where the Continuous Integration and Continuous Deployment (CI/CD) practices come in.

CI/CD is a DevOps concept that includes Continuous Integration (such as running automated code quality scans, generating reports on the changes, running automated tests (nirespire, 2018)) and Continuous Deployment (such as automating new version deployments to whatever cloud/local servers).

### 2.4.3 Reliability

Automating certain parts of the development cycle may lead to increased reliability of the final product. This is due to the fact that the element of human error is minimized, if not eliminated at all. CI/CD can be considered to be a part of this category as well due to the automation aspect.

### 2.4.4 Scale

Operating using DevOps and its practices can help immensely with scaling up the production system. If we assume that automated testing or automated deployment techniques are used, the scaling of the final product becomes a rather trivial problem, which is mostly dependent on proper deployment automation methods rather than on coming up with new solutions every time to just make sure that the scalability can be preserved. Consistency of the product ensured by the previous point also plays a huge role for providing scalability because project micromanagement becomes borderline impossible at scale (Oteyowo, 2018).

### 2.4.5 Improved Collaboration

DevOps has an emphasis on product/feature ownership and its accountability. Teams of developers have to collaborate much closer with teams of operations and share the responsibility because their workflows are somewhat combined in order to align with DevOps practices. An advantage of improved collaboration techniques could be reduced inefficiencies by saving time due to, for example, reduced handover periods between developers and operations or writing code that is specifically tailored for the environment in which it is run.

### 2.4.6 Security

Automated compliance policies can be used for DevOps adoption and security implementation. Ensuring that the CI/CD workflow is configured properly can also add a layer of security without slowing down the long term project development.

# Chapter Three

# AWS DevOps Oriented Services

AWS is the most popular solution, as well as being regarded as the best one to use due to the community and documentation it provides. It has over 170 different service offerings, but a select few of them have extremely obvious DevOps flavours. Due to it being so popular and widely used, we are going to discuss some of these services in this section (Avery, 2019).

## 3.1 Lambda

AWS Lambda is a service that provides cloud based serverless code execution that is also charged only for the compute time used rather than the whole running time. However, we are going to focus on what it brings to the DevOps table rather than its business side.

The biggest benefit of Lambda is the fact that it requires almost zero administration. The very few properties that can be set up outside of the code include the amount of memory that the function can consume, the longest execution time before timing out as well as the rights that the function has in order to be able to access other AWS services. Once the function is written and all of the required components are imported (such as other AWS services like a serverless database DynamoDB or AWS API Gateway for connecting the resources outside of AWS). The functions can be tested in real time by simulating invocations with specific parameters. Its biggest advantage is also incredible scalability potential (Mendlawy, 2017).

One possible downside of Lambda is the fact that it is usually used for simple lightweight computing due to its modular nature. If heavier computing is required, a more classical service such as EC2 for deploying virtual machine instances can be used with proper server deployment.

If to compare AWS Lambda against the benefits that DevOps provides described in the previous sections, quite a few conclusions can be made. First and foremost, is the scalability. Since the functions don't have a concrete server that they are being run on (from the developer's perspective of course), they can be invoked as many times as you would like to at the same time. The next thing is the fact that they are essentially a piece of small isolated code that is usually not longer than a couple of hundred lines. This provides increased reliability since they are simple and easy to debug, as well as increased speed of development due to the fact that many Lambda functions can be chained together to keep the infrastructure clear but still be able to have complex behaviour. This argument can also be used to argue that Lambda represents Microservices.

Each and every Lambda function can also be logged extensively in AWS CloudWatch, but this will be discussed in the following section.

## 3.2 CloudWatch

AWS CloudWatch is a service for extensive monitoring of other AWS services. It allows users to gather logs or performance reports. Instead of monitoring every single service one by one, CloudWatch allows to monitor the whole stack of resources. For example, if your solution uses a mixture of Lambda (for data processing), DynamoDB (for fresh data storage) and S3 (an online storage for storing old and rarely used data), CloudWatch can tell exactly how many errors each of them had and, for example, the times when Lambda has failed to write to DynamoDB specifically.

As a service, CloudWatch requires very little configuration but provides an incredible

amount of insights into how the whole system/product is behaving, which allows it to pinpoint the weaknesses and put the focus on them instead.

CloudWatch strengthens the CI/CD aspects and improves collaboration between development and operations since it can directly trace the behaviour of the application code.

## 3.3 Cloud9

AWS Cloud9 is a service that provides a cloud based IDE. It allows you to write, run, and debug code in a browser. It supports a wide range of languages but it has two distinct advantages over any normal IDE.

The first of the main benefits is the fact that it allows you to code together in real time. This improves the collaboration aspect of working on the project. If many people are working on the same part of the project, it improves teamwork and efficiency of people collaborating, as they learn how to handle each other better. Being able to code together also improves reliability of software since the same chunk of code can be validated by multiple developers.

The second one is its ability to access AWS through an integrated terminal. This integration allows for increased development and delivery speed since the overhead of running multiple features/services is taken away.

## 3.4 DynamoDB

DynamoDB is a serverless NoSQL database that is a perfect choice for being combined with AWS Lambda for designing serverless solutions. Being serverless, its two major benefits are performance at scale and lack of servers to manage.

The database has two modes: provisioned read/write capacity, or auto-scaling. Automatic scaling falls in line with the DevOps practices and is usually the one that is used by most of the users to ensure that no throttling can occur.

Lack of servers to manage can also fall under the scaling part of DevOps. Removing the complexity of managing servers, scaling becomes almost a non-existent problem. Also, since there are no servers to manage, another point of failure is taken away, thus making the solution more reliable (Ranganathan, 2018).

## 3.5 Concrete Examples

To better illustrate how the described AWS services practice DevOps, let's consider a concrete example. Let's consider a very basic serverless architecture shown on the image below. The system consists of three main components, AWS API Gateway (for providing an external link to the AWS system), AWS Lambda (for processing HTTP requests from API Gateway), and AWS DynamoDB (for storing or retrieving the data from the requests).

API Gateway can be used for configuring a RESTful API that can, for example, use AWS Lambda function on invocation. Depending on a request type (GET or POST), a different Lambda function can be invoked.

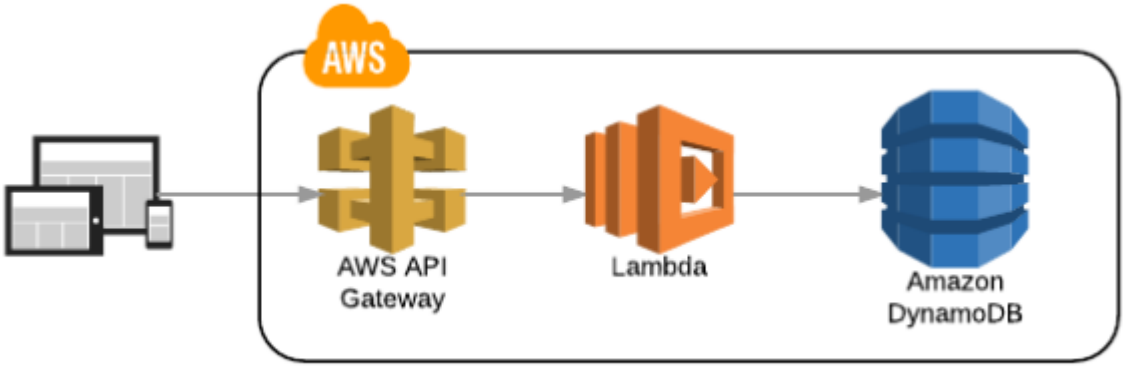


**Figure 3.1** Example

In turn, the Lambda functions can invoke the DynamoDB instance for either retrieving or posting data. An example of such a system could be a simple data collection service, where some of the client devices are sensors that post data, whereas others are client phones that

retrieve data for further processing, such as graphing or any other type of data analytics.

Now, how does this relate to DevOps? First, even though the system is extremely simple, the system design is still modular, which opens doors for CI/CD practices. If we want to upgrade a component, no other one has to be touched apart from just changing a reference object in the code, which is a trivial task. The Lambda functions are the binding component of the system, thus providing the infrastructure as code. Its microservice nature also allows for more flexible development, in case if there is a need for a more complex functionality, a new Lambda function may be added on top of the old one without changing the existing infrastructure. The scalability of the presented system is also only limited by the budget since both Lambda and DynamoDB are autoscalable and serverless. Being a serverless solution, not much resources will be spent on maintaining the setup, meaning that the focus can be put on the functionality, rather than operability.

# Chapter Four

# Conclusion

To sum up, it can be seen that all of the AWS services covered above have one thing in common - speed of development. Getting rid of development overhead such as excessive testing or long build times or manual deployment can help engineers to focus on the one thing they do best - engineering solutions. The convenience that AWS offers is obviously great, but the fact that it pushes you as a developer towards using DevOps practices without even realising it is even greater. If you get involved in working with one of the services, the chances are that you will have to look into others to make them work together, and this is where it will overlap with someone else's work. This cycle will repeat itself until a full ecosystem is formed, in which the speed of development and delivery is boosted because AWS has shaped its products in a way to encourage the most efficient use.

# REFERENCES


Avery, Dirk (2019). *Amazon Web Services (AWS) 101*. URL: https://medium.com/faun/amazon-web-services-aws-101-4f5545567937.

guru99 (2018). *Advantages and Disadvantages Of Cloud Computing*. URL: https://www.guru99.com/advantages-disadvantages-cloud-computing.html.

Mendlawy, Michael (2017). *From 0 to 100 K in seconds: Instant Scale with AWS Lambda*. URL: https://aws.amazon.com/blogs/startups/from-0-to-100-k-in-seconds-instant-scale-with-aws-lambda/.

Nemer, Joe (2019). *Advantages and Disadvantages of Microservices Architecture*. URL: https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/.

nirespire (2018). *What is CICD — Concepts in Continuous Integration and Deployment*. URL: https://medium.com/@nirespire/what-is-cicd-concepts-in-continuous-integration-and-deployment-4fe3f6625007.

Oteyowo, Temitope (2018). *DevOps in a Scaling Environment*. URL: https://medium.com/tech-tajawal/devops-in-a-scaling-environment-9d5416ecb928.

Ranganathan, Karthik (2018). *11 Things You Wish You Knew Before Starting with DynamoDB*. URL: https://blog.yugabyte.com/11-things-you-wish-you-knew-before-starting-with-dynamodb/.

Rehkopf, Max (2020). *What is Continuous Integration*. URL: https://www.atlassian.com/continuous-delivery/continuous-integration.

Ryadel (2019). *DevOps Methodology, Lifecycle and Best Practices explained*. URL: https://medium.com/ryadel/devops-methodology-lifecycle-and-best-practices-explained-761d526048cfl.