Technical and organizational challenges when adopting DevOps principles for the integration of Artificial Intelligence into classical Software Engineering

Bachelor Thesis

for the degree of

Bachelor of Science

from the Course of Studies Applied Computer Science

at the Cooperative State University Baden-Württemberg Mannheim

by

Pascal Schroeder

Time of Project:	01/07/2019 - 23/09/2019
Student ID, Course:	5501463, TINF16AI-BC
Company:	IBM Deutschland GmbH, Ehningen
Supervisor in the Company:	Steffen Krause
Reviewer in Corporate State University:	Prof. Dr. Holger Hofmann

Declaration of Sincerity

Hereby I solemnly declare that my project report, titled *Technical and organizational challenges when adopting DevOps principles for the integration of Artificial Intelligence into classical Software Engineering* is independently authored and no sources other than those specified have been used.
I also declare that the submitted electronic version matches the printed version.

Mannheim, 23rd September, 2019

Pascal Schroeder

Contents

List of Figures

List of Listings

Abkürzungsverzeichnis

AI Artificial Intelligence

IT Information Technology

DevOps Development and Operations

SaaS Software as a Service

ML Machine Learning

VM Virtual Machine

CPU Central Processing Unit

GPU Graphics Processing Unit

IDE Integrated Development Environment

1 Introduction

The upcoming trends of Artificial Intelligence (AI) and Cloud Computing are changing the world of Information Technology (IT) as a whole. One very important aspect is the way applications are developed and operated. These processes can be summarized under the term Development and Operations (DevOps). This work will deal with how AI and Cloud force DevOps to transform and will discuss different approaches to adopt and realize those principles for AI applications. The following chapter will introduce the terms of AI and Cloud computing and will describe how these innovations force DevOps to transform.

1.1 Changes in Software Development caused by AI

First discussed in the 1940s, Artificial Intelligence has made great progress in recent years thanks to advances in computing capacity as well as Deep Neural networks and the accessibility of huge amounts of data. [1] This leads to companies investing in AI about 32€ Billion in 2019 according to a forecast by IDC, which makes it one of the most important fields of businesses. [2]

These progresses do not only impact the products itself, but also their development. In contrast to a traditional software development process in which the business logic is explicitly encoded in rules, solutions that rely on Machine Learning (ML) are fed with huge amounts of data that contains the business logic only implicitly.

This leads to a completely different development cycle. While classical software development is focused on the design and the development of the code, which is followed by testing and deployment, the machine learning lifecycle consists out of data collection, preparation, the training of the model with those data as well as the deployment of this model. [3] The differences between both is described in more detail in chapter 2.5.

This new type of software development is getting more and more important. Some people like Andrej Karpathy, Director of AI at Tesla, even predict, that these new type of software will replace traditional software development as a whole. [4] This has some advantages, e.g. that it is easier to learn and to manage, more homogeneous and very well portable. In return it is harder to understand for humans so that those systems appear to us as “black boxes”. Also debugging can be very difficult as well because usually there is no real error message. [3]

In this work, all the progresses that have led to this transformation will be explained in more detail. Additionally, it will be described which difficulties and challenges come along with it, focused on necessary changes for operations of the development cycle, called DevOps.

1.2 Cloud Computing

In 2009 the university of Berkeley published a paper, in which the potential of Cloud Computing has been discussed. In doing so Cloud Computing has been defined as both - the applications delivered as services over the Internet, referred to as Software as a Service (SaaS), as well as the hardware and the systems software in the datacenters that provide those services. [5]

Thereby a distinction is made between a Public Cloud and a Private Cloud. A Cloud is called public when it is made available in a pay-as-you-go manner to the public. This means, that you pay for a service in advance and then you can only use what you have paid for. Private Cloud on the other hand refers to internal datacenters of a business or other organizations unavailable to the public. [5]

SaaS in general has advantages for both end-users as well as service providers. While the service provider can install, maintain and control their services more easily, the user can access the service anytime, anywhere, collaborate more easily and keep their data inside the infrastructure. [5]

Cloud Computing enables this possibility to more application providers and additionally also the possibility to scale their applications on demand. Based on this possibility there are some more advantages identified for Cloud Computing in particular:

- The illusion of infinite computing resources on demand
- The elimination of an up-front commitment by Cloud users
- The ability to pay for use of computing resources [5]

The first point eliminates the need to plan far ahead how many resources are needed but enables the user to buy as many resources as necessary as soon as it is needed.

Also, Cloud users are allowed to change their need for resources any time, so they can scale up their application and don't have to commit to their need beforehand as described in point two.

The last point is about saving money when the resources are no longer needed. As it is possible to scale up, it is also possible to scale down, which eliminates the need to pay for the resources. This relieves the Cloud users from taking too much risk when paying for resources they may not need for a long time.

For realizing those advantages different approaches were competing with each other: One looking similar to physical hardware with the ability to control the entire software stack similar to a low-level Virtual Machine (VM). The alternative was a clean separation between a stateless computation tier and a stateful storage tier. In this competition, the first option has become established, because in the beginning most users wanted to recreate their local environment instead of writing new programs solely for the cloud. [6]

But this solution still forced the User to manage their environment themselves. Especially for simpler applications, it is desirable to have an easier path to deploy their applications to the Cloud.

Amazon provisioned a new solution for those needs in 2015 called AWS Lambda. Lambda offered the possibility to simply write the code to be executed and leaves all server provisioning and administration tasks to the cloud provider. [7]

This is known as serverless computing, because - even though there are still servers being used for the computation tasks - the user does not have to care about any tasks on serverside and can focus on writing the code. Serverless services must scale and bill automatically based on usage without any need for explicit provisioning. [6]

Those new possibilities with the Cloud technology changed the way of how developers can deploy and manage their products. Additionally, also AI development benefit from this very strongly because of several reasons.

The first one is automated scalability, which eliminates the need to worry about raising workloads. Additionally, when AI models are being trained, there are many resources needed, but besides this task, there are not as many resources necessary. Through serverless computing, this idle time will not be billed. [8]

Serverless computing also reduces the interdependence, because the functions can be updated, deleted or invoked any time without having any side effect on the rest of the system. [8]

All this forced developers to change their way of developing and operating their products. This change of the DevOps lifecycle will be introduced in chapter 2.5

1.3 Development Operations in time of AI

The traditional software development has continuously been improving for years, defining standards and principles to increase the efficiency, portability, and quality of the development cycle. Those principles and practices are called DevOps. In the new type of software development described in chapter 1.1, not all of those are applicable any more and new ones need to be defined. Additionally, the upcoming Cloud technology changed the way how products can be deployed and managed as described in chapter 1.2. In this chapter, the consequences of those changes for DevOps will be examined.

Starting with development itself and ending with maintaining and improving the product, there are DevOps principles for every single stage of the development cycle. This is described in more detail in chapter 2.1. In the new world of AI and Cloud, those stages are different, some new stages have been added, others have become unnecessary.

A short, simplified comparison between the required steps of traditional operations and Machine Learning operations can be seen in figure 1.1.

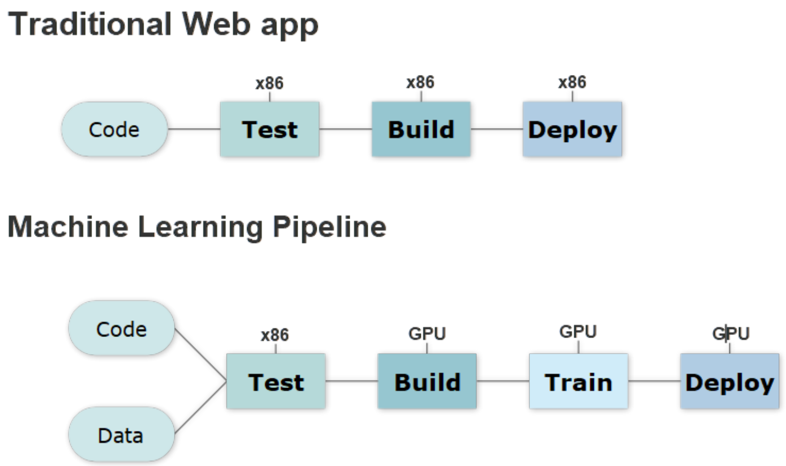

Figure 1.1: Comparison development lifecycle traditional app and ML pipeline[9]

In traditional development, there is one input - the code. This code has to be tested, built and then deployed.

The Machine Learning pipeline looks different. First, there is an additional input - data, which is very important for Machine Learning applications. The code and the data still need to be tested afterward.

Also, it needs to be built. The difference in this stage is, that the resources used for building an ML application are usually different. While traditional applications are usually built on regular CPUs (Central Processing Unit), Machine Learning pretty much rely on GPUs (Graphics Processing Unit), which leads to a more heterogeneous and complex hardware landscape.

A third difference is the training step. ML applications have to be fed with training data. This step can take very long especially when insufficient hardware is being used.

The last differentiator is, that ML pipelines are continuously updated with new data. This process is called “refitting” the model.

Besides the rise of AI, Cloud Computing also changed the way of how products are deployed and managed. The Cloud can serve as a centralized platform for testing, deployment, and production. Also, it can automatically scale applications and allocate needed resources. Serverless computing even eliminates the need to set up a complete system and offers a simple way to deploy Machine Learning or other functions. [10]

AI tools on the other hand are enabling new possibilities for improving the IT operations functions like monitoring, analysis, service management, and automation.

All those progresses led to two big trends in the DevOps area. The first one is using Artificial Intelligence for DevOps. In doing so Machine Learning functionalities are used to enhance all IT operations. Gartner calls this “AIOps” and predicts, that the use of AIOps will rise from 5% in 2018 to 30% in 2023. [11]

The second trend is using DevOps for the development of AI. This means adopting existing principles and practices for the traditional development cycle to the development cycle of AI tools. The objective of this is to upheld existing standards and maximize the efficiency of the development processes. [9]

In this work this second trend will be analyzed, the influences leading to it will be described and possible solutions will be shown, compared and discussed.

2 Theory

In this chapter the theoretical basics, that are needed for creating DevOps principles for AI, will be given. For that it will be started with an explanation of what DevOps is in general. Then it will be shown, which new possibilites came with Cloud and 12-factor-apps in chapter 2.2. After that the basics of Machine Learning and AI will be given in chapter 2.3, specializing on the lifecycle of AI development in chapter 2.4. Last in chapter 2.5 all these knowledges will be combined to adopt the DevOps principles explained in chapter 2.1 to the new world of AI with the help of new Cloud technologies.

2.1 Development Operations principles

Since people started manufacturing products on a mass scale, the goal is to increase the efficiency of this manufacturing process and reduce waste of time and material.

One early set of best pracitces for manufacturing was the concept of “Lean manufacturing”, which tries to reduce the waste of ressources and time of a production cycle as much as possible. With the upcoming use of Software as commercial product in the 1970s [**???**] a desire came on to create best practices for developing and operating products the same way as it was already usual in common manufacturing. [**???**]

In 2009 two Flickr employees - John Allspaw and Paul Hammond, presented their way of combining Development and Operations. Inspired by this presentation a belgian consultant named Patrick Debois formed a new conference - the “Devopsday” in Ghent. This is how the term “DevOps” has been created and prevailed. [**???**]

Since then DevOps has been established or at least planned in 91% of all companies as an important way to increase their efficiency of Software development. [**???**] For almost every stage of development there are principles and practices defined and continiuously improved. But before those practices will be explained, a further insight in the business need will be given.

Every process or product need a business value that cover the costs caused by it. For that there must be either an outcome for the customer or reduced producing costs.

For DevOps it is usually even both - on the one hand an enhanced customer experience can be guaranteed and on the other hand the efficiency of the production cycle can be increased.

One example for an enhanced customer experience are practices to get fast feedback from all stakeholders. This feedback can then be used to improve the designed product. One of such practices is the so called “A-B testing”. There two different sets of features are enrolled to two groups of randomly chosen users. Both can give their feedback to the producer and the set with the better feedback will then later be enrolled for every user.

The efficiency can be increased through reduced waste and rework because of practices to write reusable components. Another example are tools for planning a product or fast ways to deliver a product without the need to redeploy everything step by step. In this chapter the advantages of DevOps will be delighted in more detail and some of the practices will be described and explained.

The DevOps movement is generally based on four principles. The first one is to develop and test against production like systemt to move operations concerns earlier in the life cycle. The purpose of this is to see how th esystem behaves and performs before it gets deployed. This is also advantageous from an operations perspective, because it can be seen, how the system behaves when it supports the application.

The second principle is to deploy with repeatable and reliable processes. The objective is to create a delivery pipeline, that enables continuous, automated deployment and testing of the product.

Third, it is important to monitor and validate operational quality. This means, that applications and systems should not only be monitored in production, but already earlier. This forces automated testing to be done early and often to monitor the application. Metrics should alwways be captured and analyzed to provide an early warning system about potential issues and risks.

The last principle is to amplify feedback loops with the goal to enable a quick reaction to issues. For this organizations need to create a communication channel to its users, so that they can give feedback and the developers can react to it accordingly.

The DevOps practices, that have become commonplace, can be splitted in four different sets based on the different periods of a product lifecycle. To each set there are several pracices, standards and tools available, which help to achieve the best possible result. The four different sets and some example practices can be seen in figure **??**.

The first set is called “Steer”, which is about managing and planning the development and lifecycle of a product. This includes establishing and continuously adjusting business goals. The process resolving this issue is called “Continuous Business Planning”.

This includes three major points - the acceleration of software delivery, a good balance between speed, cost, quality and risk and the reduction of the time it needs to get customer feedback. [**???**]

This is mainly done via tools and practices to track the status, feedback and needs of a project in an efficient way. First, a vision of the projects overall objective should be created and every action should be guided by it. [**???**]

The strategy, which should lead to this vision, has to be monitored and agjusted continiuously based on new information and feedbacks. This is called continuous improvment. For that a good dialog between a Business and IT is necessary for defining good scopes and priorities. [**???**]

Then a good plan can be built,for example with a release roadmap, which determines which feature should be completed at what time. This is called release planning and helps to track the progress of the project and makes it easier to react on new trends, feedback or issues and adjust the single steps based on this. The status of each release as well as each single feature has to be continuously tracked, so that risks will be recognized as early as possible to increase the available time to react. [**???**]

The second set of practices is for the time of development and testing. Two eminent practices for this are collaborative development and continuous testing.

Collaborative development enables different practictioners - architects as well as analysts, developers, specialists etc. - to work together on one project. For that it provides a common set of practices and a common platform to create and deliver the software.

One core capability is a practice called continuous integration, in which developers continuously or frequently integrate their work with the other developers. For that a shared platform or repository is necessary, on which the developers can frequently commit their changes in the code. Usually this is done using a version control system like Git.

Also the application should be tested and verified continuously. For that the developer can run local unit tests to verify their changes before integrating. But this doesn't verify that the integrated code performs as designed. [**???**] A continous integration service linke Jenkins can relieve the developer of this task and automatically builds and runs unit tests on the newly commited and integrated code. [**???**] This process is called continuous testing.

Another important point is to shorten the delivery cycles through an end-to-end integration, so that it needs less time to enroll a new feature or similar. This leads to quickly given feedback and enables a faster reaction to this. [**???**] This is done in the Deployment stage of a product lifecycle and one of the root capabilities of DevOps. It deals with the automation of the deployment of the software to the different environments, which is called continuous delivery.

After the deployment follows the Operation. During this stage the performance of an application should be montoried and feedback should be collected. The results of this should be used to improve the product as well as other products, that will be developed in the future. For this there are two practices defined - continuous monitoring and continuous feedback and optimization.

Continuous monitoring provides data and metrics to the performance of an application as well as its running server, the development cycle, the production and other stakeholders.

Continuous feedback on the other hand provides data coming directly from the customer. This includes the behavior how they are using the system as well as feedback that the users provide.

Based on those retrieved data businesses may adjust their plans and priorities, improve the development cycle and features and enhance the environment in which the application is deployed in a more agile and responsive way. The objective of this is to improve the product and satisfy the users as well as use this knowledge for new products, that will be developed in the future.

IaaC

The most important technology for DevOps may be a Delivery pipeline. A delivery pipeline controls the product cycle of an application from development to production. Typically there are four or more stages - development, test, stage and production.

For every stage there is usually one specific environment. In the development environment all the code updates are being done. There are tools provided like IDEs (Integrated Development Environment) to write the code as well as tools that enable collaborative development like source control management or project planning.

Source control management is typically combined with version control. This enables the developer to also store previous versions of his application and reduces the risk of issues in new updates, because it can roll them back in case of an error.[**???**]

After the development the delivery pipeline must care for the application to be built.

Second there is the testing environment, in which single components can be tested. For that it has to manage test data, scenarios, scripts and associated results. Similar to the development environment it must not look like the production environment.

The next one is the staging environment, in which the system can be tested as a whole. The staging environment should be as much production like as possible, so that as many as possible required services, databases and configurations can be connected and applied without touching the production environment. The stage environment is for testing the application before rolling out a major update.

The last one is the production environment, in which the application will be running live and accessible for the users. [**???**][**???**]

The delivery pipeline consists out of all those stages and manages an automated transition from one stage to the next one.

For this deployment automation tools are necessary, which perform the deployments and track which version is deployed on which node. It also manages changes that need to be performed for the middleware components and configurations as well as database components or the environment.

Last there should also be a tool for release management, which provides a single collabiration portal for all stakeholders participating in a project to plan and track the releases of an application and its components across all stages.

With such technologies most of the defined practices can be performed with the help of accordingly educated people and well thought-out processes. But DevOps is no static set of practices and tools, but it changes with the changes in the world IT, such as Cloud or AI. In the next chapters these technologies will be reflected and the impact those have on DevOps will be analyzed.

2.2 Microservices and 12 factor apps

2.3 Machine Learning

2.4 Artificial Intelligence lifecycle

2.5 DevOps for Artificial Intelligence

3 Method

3.1 Catalogue of criteria

3.2 Used frameworks and libraries

3.2.1 Kubeflow

3.2.2 Tensorflow

3.2.3 R-CNN

3.3 Creating the necessary environment

3.3.1 Minikube

3.3.2 Kubeflow

4 Result

4.1 Kubeflow pipeline

4.2 Azure pipeline

4.3 Other framework?

5 Discussion

5.1 Pipeline comparison

5.2 Outlook

6 Appendix

placeholder

Literature

1. JANAKIRAM MSV. Three Factors That Accelerate The Rise Of Artificial Intelligence. *2018-05-27* [online]. Available from: www.forbes.com/sites/janakirammsv/2018/05/27/here-are-three-factors-that-accelerate-the-rise-of-artificial-intelligence/%7B/#%7D4349a30badd9

2. MICHAEL SHIRER and MARIANNE D'AQUILA. Worldwide Spending on Artificial Intelligence Systems Will Grow to Nearly $35.8 Billion in 2019. *2019-03-11* [online]. Available from: www.idc.com/getdoc.jsp?containerId=prUS44911419

3. MARIYA YAO. 6 Ways AI Transforms How We Develop Software. *2018-04-18* [online]. Available from: www.forbes.com/sites/mariyayao/2018/04/18/6-ways-ai-transforms-how-we-develop-software/%7B/#%7D1ee109f026cf

4. ANDREJ KARPATHY. Software 2.0. *2017-11-11* [online]. Available from: medium.com/@karpathy/software-2-0-a64152b37c35

5. ARMBRUST, A. FOX, AND R. GRIFFITH, M. Above the clouds: A Berkeley view of cloud computing. *University of California, Berkeley, Tech. Rep. UCB* [online]. 2009. DOI 10.1145/1721654.1721672. Available from: http://arxiv.org/abs/0521865719%209780521865715

6. JONAS, Eric, SCHLEIER-SMITH, Johann, SREEKANTI, Vikram, TSAI, Chia-Che, KHANDELWAL, Anurag, PU, Qifan, SHANKAR, Vaishaal, CARREIRA, Joao, KRAUTH, Karl, YADWADKAR, Neeraja, GONZALEZ, Joseph, ADA POPA, Raluca, STOICA, Ion and A. PATTERSON, David. *Cloud Programming Simplified: A Berkeley View on Serverless Computing*. 2019.

7. AMAZON. *AWS Lambda – Serverless Compute - Amazon Web Services* [online]. [Accessed 25 July 2019]. Available from: aws.amazon.com/lambda/

8. BHAGYASHREE R. How Serverless computing is making AI development easier. *2018-09-12* [online]. Available from: hub.packtpub.com/how-serverless-computing-is-making-ai-development-easier/

9. DILLON. CI/CD for Machine Learning & AI. *2018-10-22* [online]. Available from: blog.paperspace.com/ci-cd-for-machine-learning-ai/

10. DAVID LINTHICUM. DevOps is dictating a new approach to cloud development. [online]. Available from: techbeacon.com/app-dev-testing/devops-dictates-new-approach-cloud-development

11. SUSAN MOORE. *How to Get Started With AIOps* [online]. Gartner, 2019. Available from: www.gartner.com/smarterwithgartner/how-to-get-started-with-aiops/