Master of Science in Software Engineering
June 2019

An Introduction to the DevOps Tool Related Challenges

Sujeet Bheri
SaiKeerthana Vummenthala

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

Contact Information:
Author(s):
Sujeet Bheri
E-mail: subh17@student.bth.se

SaiKeerthana Vummenthala
E-mail: savu16@student.bth.se

Jefferson Seide Molleri
SERL - Software Engineering Research Lab
Faculty of Computing
Blekinge Institute of Technology
SE-37179, Karlskrona, Sweden
jefferson.molleri@bth.se

ABSTRACT

Introduction: DevOps bridges the gap between the development and operations by improving the collaboration while automating the as many as steps from developing the software to releasing the product to the customers. To automate the software development activities, DevOps relies on the tools. There are many challenges associated with the tool implementation such as choosing the suitable tools and integrating tools with existed tools and practices. There must be a clear understanding on what kind of tools are used by the DevOps practitioners and what challenges does each tool create for them.
Objectives: The main aim of our study is to investigate the challenges faced by the DevOps practitioners related to the tools and compare the findings with the related literature. Our contributions are (i) a comprehensive set of tools used by Developers and Operators in the software industries; (ii) challenges related to tools faced by the practitioners; and (iii) suggested recommendations and its effectiveness to mitigate the above challenges.
Methods: we adopted case study for our study to achieve our research objectives. We have chosen literature review and semi-structured interviews as our data collection methods.
Results: In our study we identified seven tools used by developers and operators which were not reported in the literature such as Intellij, Neo4j, and Postman. We identified tool related challenges from the practitioners such as difficulty in choosing the suitable tools, lack of maturity in tools such as Git, and learning new tools. We also identified recommendations for addressing tool related challenges such as Tech-Talks and seminars using complementary tools to overcome the limitations of other tools. We also identified benefits related to the adoption of such recommendations.
Conclusion: We expect the DevOps tool landscape to change as old tools either become more sophisticated or outdated and new tools are being developed to better support DevOps and more easily integrate with deployment pipeline. With regard to tool related challenges literature review as well as interviews show that there is a lack of knowledge on how to select appropriate tools and the time it takes to learn the DevOps practices are common challenges. Regarding suggested recommendations, the most feasible one appears to be seminars and knowledge sharing events which educate practitioners how to use better tools and how to possible identify suitable tools.

Keywords: DevOps, tools, Automation, Challenges

ACKNOWLEDGMENT

The time during our master's degree at Blekinge Tekniska Högskola, Sweden (BTH) has a path that made us to grow and taught us a lot of lessons we will cherish and treasure always during our life, both personally and career wise. The first part of our journey was during our bachelor's at Jawaharlal Nehru Technological University, Kakinada (JNTUK), which led us to BTH, Karlskrona that gave us an opportunity to develop not only as a student but also reinforced our skills and helped to explore our inner sense and individuality. we want to take this acknowledgement as a chance to show our appreciation and gratitude mainly to our parents for having faith in us and guiding us through any and every hurdle we faced during our journey. Subsequently, we want to thank our family and friends for supporting us and having confidence in us that we would not fail despite of our lack of self-confidence. We will be thankful to our supervisor ***Jefferson Molleri*** with heartful gratitude towards his support and guidance throughout the thesis. We also thank the Department of Software Engineering, BTH for trusting us and encouragement throughout the studies – ***Sujeet Bheri, SaiKeerthana Vummenthala***

CONTENTS

1 INTRODUCTION

Software has become a vital part of the modern society. Not only it is expected to be of high quality, but also to meet the customer's changing requirements within the allocated budget and schedules [1]. Several software development processes have emerged through the years to ensure high quality software products [26]. On the one hand, Plan-driven approaches are better suited to projects where the requirements are not expected to change dramatically during its lifecycle, projects that are large in size, or mission-critical projects, because of the emphasis that's put on up-front requirement specification, rigorous documentation and compliance to standards [1, 2]. On the other hand, many business applications are better developed using agile methods, because they allow more flexibility to changing requirements [18, 26] which is often the case in the competitive environment they operate [1, 2]. Agile methods have their own limitations, however. They promise to deliver software faster by developing incrementally, and to ensure it meets the needs of the users by relying on user and/or customer feedback more frequently than in plan-driven approaches. However, rapid software release does not only involve software development activities, but also operational activities, such as installation of the release, configuration of the production environments, and monitoring the production environment to ensure qualities such as high availability and performance [3, 24].

DevOps has emerged in an attempt to address these issues, by putting the primary focus on the collaboration between development and operations [4], while automating as many steps as possible before a change in the software can become visible to the users [5]. Main benefits typically faster and more frequent releases through automation, increased knowledge sharing through more effective collaboration and communication [27], as well as better products through improved ability to adapt to changing requirements [6].

Despite the advantages that DevOps has reported to bring, there are still challenges in practicing DevOps, such as lack of training on how to practice DevOps, lack of clear definition for DevOps, use of multiple production environments which creates impediments for the implementing continuous practices [6, 22].

Several challenges in practicing DevOps, however, are associated with the tools being used to automate the development and operation activities. The large availability of tools makes it hard to know which ones are better suited for each project to begin with [7, 8]. Integrating a set of tools together into a single deployment pipeline is even more challenging [9]. Since automation is a key practice in DevOps [11, 12], challenges related to tools in DevOps practice is a topic worth investigating. It seems this topic has received little to no attention so far from the research, we decided to investigate it further with our own research.

2 BACKGROUND

2.1 The road to DevOps: plan-driven software development

The modern world requires software to run. Software is used extensively in various domains of our society, such as industrial production, product distribution, transportation and entertainment to name a few. Good software is expected to meet the requirements of its users, be easy to use and dependable. Moreover, its production is expected to be within budget and on schedule [1].
Meeting these requirements is not a trivial endeavor, since software systems are characterized by high degrees of complexity [1], which has been increasing for decades and is estimated to continue to increase [2]. This led to software being developed by teams, rather than individuals [1]. Additionally, a systematic way of creating software was needed. This is known as a software development process [1].
A software development process is a set of activities that aims at the production of a software product. Such activities typically include [1]:

- software specification, which defines what a software product should do and the constraints of its operation, such as constraints with respect to performance, ease of use or security
- software design and implementation, which transforms the specification into a working product
- software validation, which aims at making sure the software product meets the needs of its intended users; for instance, by performing software testing, defects in the software can be found and fixed before the final product reaches the users
- software evolution, which ensures the product remains useful after it has been delivered to the users, with fixing bugs and adding new features

Over time, various development processes have evolved [26] to meet the demands of different types of software. For example, a plan-driven process that relies on the full specification of the software requirements before software design, implementation and testing can begin, is more suited for safety-critical control systems or very large systems [1, 2].
A well-known model [29] for a plan-drive software process, and the first to be published, is the Waterfall model [1]. The term "waterfall", however, was coined by Bell, Thayer in [13], most likely referring to the fact that, in principle, once a phase was considered complete, it would not be repeated again Sommerville in [1] just like the water does not travel up [14]. In this model, every phase of the development process must first be planned and scheduled, before work can begin. One or more documents are used to mark the end of a phase as well as input to the next one [1, 15]. This made it easier for management to track the progress of a software development project, which helped in the popularity of the model.
Plan-driven approaches to software development, have their challenges. For example, the Waterfall model was interpreted as a purely sequential model, where each phase could start only if the previous phase was completed [2]. This meant that testing the software for potential problems would happen late in the development process [1,15], and the later problems were found, the more time and effort would be required to fix them.
These problems are not always caused by the people who develop the software. The requirements of a software product are inherently problematic. In particular, they are sometimes "incorrect, ambiguous, inconsistent or simply missing" [13]. The risk of incomplete or erroneous requirements is further exacerbated by the fact that software is developed in a rapidly changing environment. For example, software is vital for businesses to respond to opportunities or pressures in the market [1], and is, thus, an important strategic advantage [2]. Therefore, software development needs to be emergent, as the requirements of software products evolve.
Since conventional plan-driven approaches to software development lack the adaptability needed to successfully deal with a changing environment and evolving requirements, a new approach to software development has emerge, the agile approach [1, 2, 16].

2.2 The road to DevOps: agile software development

In February 2001, 17 prominent software engineers attended a summit to promote a more effective software development approach [18, 26, 28]. The outcome of this summit, known as the "Manifesto for Agile Software Development" can be found at [17]. Its four main points are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Well-known examples of agile methods include, but are not limited to, Extreme Programming (XP), Scrum and Lean Software Development (LD) [18, 26, 28].

The term "agile" refers to the main characteristic of such software development methods, to allow for flexibility to changing software requirements [18, 26]. To accomplish that, they rely on incremental software development. This means that software is developed not in one long iteration, but in several shorter ones. During each iteration, all the software activities are performed in a subset of the software requirements, usually starting with the most basic ones in the first iterations. This approach to software development leads to the following main benefits when compared to plan-driven methods [1, 19, 20]:

- improved ability to adapt to changing requirements
- easier to get feedback from the users/customers about the software
- faster delivery of useful software to the users/customers

However, agile software development is not without its own challenges. One in particular stands out, because it disrupts the primary goal of rapid software delivery; namely the lack of cooperation between development activities and operation activities.

Hutterman in [21] explains that even after a software product has been developed, additional steps need to be taken in order for it to be used by the end users. Additionally, after the initial version of the software product, requirements for new features or requests for bug fixes emerge, so changes have to be made to the software. Every such change has to pass through several steps before it becomes available to the end users. These steps typically include:

1. changes have to be made to the source code of the software by the developers
2. then the changes have to be integrated with the rest of the software [22].
3. this results in a new version of the software (also known as release), which must now be turned into an executable program, or set of programs; this process is known as build
4. once the changes in the source code have been built into a new release, this has to be tested by the testers, to lower the risk of defects in the software
5. this new release needs to be shipped to the end users and installed (also known as deployed [21] in their particular environment (also known as production environment) by the operators.

The series of steps that a change in the code takes until it becomes visible to the end users is called deployment pipeline [5]. Only when a change in the software becomes available to the end users, it adds some value to the software, so all of the steps in the pipeline are important [21, 23].

When it comes to operation activities, they include [3, 24]:

- user support, e.g. installing the software to the production environment
- monitoring of software to detect potential problems
- meeting requirements such as performance, security and system availability

While developers need to be skilled at designing, writing and to a certain extent testing the source code of software, the operators usually have different skills, such as system, database and network administration [3, 21]. The development and operation activities are usually performed by the same team in small software companies, but most medium and large companies assign these activities to independent groups [5].

Huttermann in [21] further explains that in software companies where plan-driven approaches are used for software development, it is not uncommon for different development tasks to be performed by different roles. For example, writing source code is done by developers where software testing is done by testers. In contrast, in agile development environments, the boundaries between these roles are removed, so both development and testing are done by the same team and it is a shared responsibility. Such boundaries, be it in the formal of organizational structures, i.e. different teams with different specializations, or in the mindsets of the people involved, are known as organizational silos [25].

Removing these silos has led to the creation of cross-functional teams, where different team members may have different skill sets, but as a team they have all the skills required for the development of software [21]. This removal of the boundaries between different roles leads to better collaboration and faster development [11]. This difference can be seen in the following diagram:

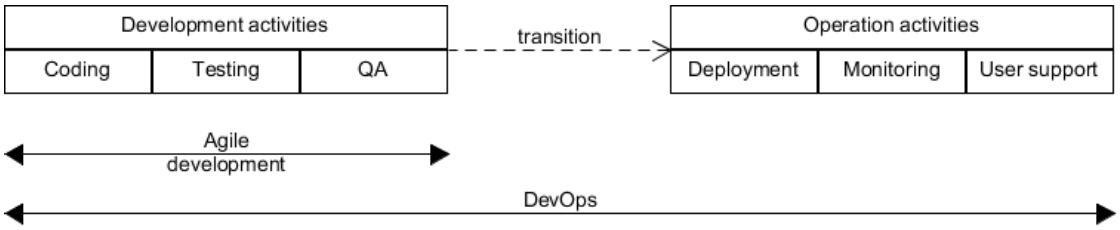


Figure 1. The process of Agile software development from inception to transition and DevOps extends from elaboration to Operations Hutterman, (2012).

As a side node, QA, which is seen in the diagram above, refers to Quality Assurance. QA aims at achieving higher quality by means of proper execution of the software development process. On the other hand, Quality Control aims at improving the quality of the software through certain activities, such as testing activities. Thus, QA supports testing activities by ensuring required resource allocation for them and their proper execution [84].
As can be seen in the figure above, agile methods do not take the concept of collaboration far enough. The focus of agile methods on improved communication and collaboration regards the development activities, so it stops when the software is given to the operations team to handle its delivery and deployment to the production environment [21]. That is where the challenge arises. Since operation activities and development activities are performed by different people with different roles and priorities, certain problems occur [21].
On the one hand, the developers are pressured to create new releases as quickly as possible [19, 20] in order to satisfy user and customer requirements for new features or bug fixes for the current release. On the other hand, operation teams do not welcome change, since new releases usually come with problems that need to be fixed, and with more user complaints when a new release has stability issues [21].
Usually, the problems become visible when a new release from the developers has been tested for potential defects and passed the tests, but it does not actually work in the production environment (the end users' system). This could be due to the fact the developers use different environments than the operators. Alternatively, it could be due to incorrect expectations about the production environment [21]. In any case, such problems can lead to tensions between the development team and the operations team [21] which lead in turn to delays; namely the opposite of what the users and the customers need.
To address these kinds of issues, a new way of developing software emerged; DevOps.

2.3 What is DevOps?

The word "DevOps" is a portmanteau for Development and Operations [11] meaning that it puts a strong emphasis on the communication and collaboration between the development and operations to achieve frequent software releases [4]. The term first was first used in a conference by Patrick Debois, in 2009 [30]. But what exactly is DevOps?
Research has reported repeatedly that DevOps lacks a common definition [21, 22, 31, 32]. However, DevOps advocates a set of core principles to effectively address the problems mentioned in the previous section [31] and to increase the rate at which new software is released successfully and efficiently [4]. Since 2009 when the term first appeared, there have been many interpretations about what actually DevOps is [22]. In particular Huttermann in [21] mentioned that there is no proper definition for

DevOps which covers all the aspects of this new phenomenon. He claims that this is due to DevOps being a multifaceted concept.
Most of the definitions published about DevOps in organizations are based on the personal needs of team members [32]. Additionally, most of the online blogs and surveys mentioned DevOps as a job description which requires development and operation skills [22] while Degrandis (2011) describes DevOps as a software revolution, which requires effective leadership to be adopted successfully and provide the promised benefits of rapid deployment cycles.
DevOps was also mentioned as an extension of Agile approaches that includes operation activities, not just development [1,34]. Jabbari, et al (2016) proposed different definitions of DevOps with relation to other software development methods, like Waterfall and Agile. For example, Jabbari claims that Agile is a suitable DevOps enabler, because DevOps extends Agile by supporting collaboration between the development and operations, and automates build, test and deployment activities.
By 2016, the phenomenon of DevOps has already been referred in the literature as methodology, philosophy, tool, set of strategies, culture, set of continuous practices or methods, set of values and principles, process and role in organizations [78]. Banica et al in [36] also shares the view that DevOps is a methodology, albeit an "early-stage" one, i.e. a methodology that has yet to mature. This methodology includes the software development activities from development to operations Banica et al [36] in agreement to what Huttermann in [21] and Airaj in [34] have claimed before. Hussain, Clear, MacDonell in [34] however, describes DevOps as movement, which has gained popularity due to the benefits it brings with continuous practices, such as continuous integration and continuous deployment. Further support to the claim that DevOps extends Agile is given by [20], who conclude that the two approaches share values and goals but have different scope, i.e. DevOps is broader since it encompasses operations as well as development activities.
The most recent work in this area is the work of [32], who attempted to solve the problem of lack of clarity around the DevOps phenomenon, and investigate the possibility of a commonly acceptable definition of DevOps, based on both a literature review and empirical findings. He found that none of the previously formulated definitions to reflect accurately enough how DevOps is practiced in the software development industry. Therefore, he advocates that, when the term DevOps is used, the collaboration between development and operations should be the only core concept implied, while any additional concepts or practices related to DevOps depend on how DevOps is implemented in a certain context.
However, the definition of DevOps that reflects our current understanding better is the one given in the work of Smeds, Nybom and Porres (2015). They described DevOps as a set of capabilities geared towards frequent releases and deployment, such as continuous integration and testing, and continuous release and deployment. They also include a set of cultural enablers aimed at breaking the organizational silos between the development and operations, such as shared goals and incentives, shared responsibilities, respect, and trust. Finally, they include a set of technological enablers that are necessary to achieve the aforementioned capabilities, such as tools for automating build, test and deployment activities. We summarize these capabilities and enablers in the table below Smeds, Nybom and Porres (2015) in [22]:

Capabilities	Continuous planning Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behavior monitoring and feedback Service failure recovery without delay

Cultural Enablers	Shared goals, definition of success, incentives Shared ways of working, responsibility, collective ownership Shared values, respect and trust Constant, effortless communication Continuous experimentation and learning
Technological Enablers	Build automation Test automation Deployment automation Monitoring automation Recovery automation Infrastructure automation Configuration management for code and infrastructure

Table 1: DevOps capabilities and enablers Smeds, Nybom and Porres (2015)

As Smeds, Nybom and Porres (2015) explain in their work, the word "continuous" in the above table means "in small increments and without delay". The word "automation" means appropriate tool support. Smeds, Nybom and Porres (2015) state, establishing the technological enablers is "a matter of tool choice, tool configuration, and tool design."

Smeds, Nybom and Porres (2015) state that, without these cultural and technological enablers, the DevOps will not work efficiently.

While many of the terms used in this table are explained in [22] we felt that additional information was needed to better understand some of the continuous practices included in the table, so we list our findings below:

- Continuous Integration (CI) [37]: It involves steps taken to manage changes made to the source code of the software under development. These changes are typically merged with the rest of the software's code. They are often validated by code inspection tools for potential defects and/or measures by quality metrics. They are also tested with unit tests. They need to be built and they also need to be tested with acceptance tests. These steps are usually automated with the help of appropriate tools, but continuous integration puts emphasis on the fact that they take place regularly, so that the developers can get quick feedback in case of faulty code changes [30]
- Continuous Delivery (CD) [37]: It is the natural next step after CI. It involves steps related to the building a new release of the software under development and automatically installing it in some environment, but not necessarily the production environment [30], e.g. for testing purposes [38]. According to Humble and Farley (2010), continuous delivery requires that new software is released easily and frequently, even several times a day.
- Continuous Deployment (CDE) [37]: Similar to CD, it is related to be able to rapidly install new releases, but this time, at the production environment of the actual end users. It therefore implies CD [30]. A software company without CD cannot achieve CDE.

Similarly, we list some core DevOps characteristics we found during our literature review, so as to provide more information related to the DevOps phenomenon, as it has been sketched by the research community.

2.4 DevOps key characteristics

Despite Erich's claim in [32], that DevOps should imply only the collaboration between development and operations, we would consider it an omission not to mention other reported core characteristics of DevOps. In particular, we list the following main DevOps characteristics, as reported by [78]:

- Culture: DevOps advocates respect and shared responsibility between the people who carry out operations and development activities, as well as emphasis on the effective communication between them.

- Automation: Using appropriate tools to automate tasks such as source code integration, testing, software delivery and deployment, as well as to keep track of changes to artifacts related to the software development (configuration management).
- Measurement: Performance measurement of both operations and developers are based on the same metrics, driven by business value (only changes to the software that have been integrated, tested, built and deployed successfully add value for the users of the software [21, 23]. Thus, meaningful metrics have to relate to user needs and they should be the same for both the operations and the development, so that the respective roles will be motivated to align their goals and collaborate more effectively
- Collaboration: As Smeds, Nybom and Porres (2015) have stated, improved collaboration is key to enable effective practice of DevOps, so the developers and operators should conduct several software development activities together, such as writing scripts for deployment and for running test, as well as solve emerging problems together.
- Monitoring: Everyone involved in the software project should be involved in actively monitoring the production environments as well, in order to prevent problems in future releases more easily.

From these DevOps characteristics, our research focuses on automation, because it is an important practice for DevOps [11, 12]. We, thus, dedicate the following section to it.

Automation in DevOps

Automation allows the developers, operators, testers and other stakeholders in DevOps to automate the tasks performed in the creation and deployment of software [5, 12]. Automating activities such as integration, testing, build and software delivery, reduces delays and risks because such activities are time consuming and error prone if done manually [21].

Regarding delivery of a new release in particular, it can be a complicated work for many applications, as explained by Humble and Farley in [5]. For example, it could mean setting up and configuring web servers in the case of web applications. It can also involve fixing any unpredictable errors, so that the new release can run properly. Such activities are hard to do manually [5]. Delivering and deploying a new release manually is error-prone and can lead to delays and additional expenses [5]. These problems affect primarily the operations team, further increasing the risk of tension between them and the development team [21]. For all the aforementioned reasons, DevOps relies heavily on automation [39].

One of the core concepts in DevOps automation is the deployment pipeline Humble and Farley in [5]. Based on Gill, Loumish and Riat in [8] and Huttermann in [21], we understood that the term delivery pipeline is often used instead. A deployment pipeline, according to Humble and Farley in [5] is the process that changes in the source code have to go through to become visible to the end users. In other words, it is the process of getting the software under development from version control to the production environment.

DevOps optimizes the deployment pipeline, by automating every step of the pipeline, in order to avoid delays during the development process [8]. In doing so, DevOps achieves continuous practices, like CI, CD, CDE and continuous monitoring [22]. We illustrate our understanding of how DevOps interacts with the practices of CI, CD and CDE with the following figure:

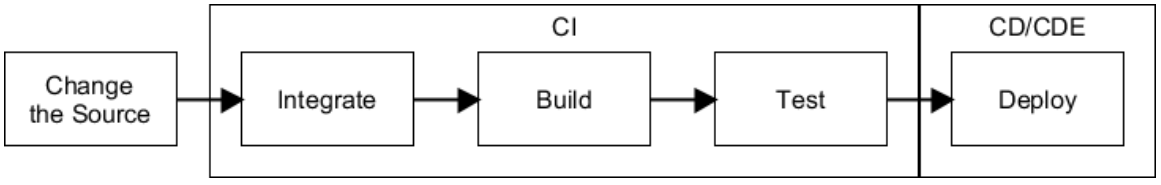


Figure 2: DevOps automated deployment pipeline with CI, CD, CDE.

Due to the DevOps automating the deployment pipeline, frequent feedback from the pipeline is enabled, so that potential errors can be detected and fixed quickly. Additionally, each release is continuously monitored by the operators for potential performance, stability, security or other issues [11, 21, 31]. This can help the team respond to such issues faster as well as avoid them in the future.

Tools in DevOps

A wide variety of tools can be used to automate various steps of the development process [7] such as source code management [40], build, test and deploy [65]. DevOps readily makes use of many of them, not just for reducing manual labor and potential errors that comes with it, but also as a means to get continuous feedback during the development process so that problems can be quickly detected and fixed, as well to support better communication and collaboration between the teams [10, 32].
From the literature review Huttermann [21], we understood that tools for the automation of development activities already existed before the advent of DevOps, but many such tools renamed themselves as DevOps tools after DevOps came.
Selecting the appropriate tools is important for any DevOps organization [39]. Different tools are used to support different aspects of the development process. Tool selection depends mainly upon the product features and customer/user needs.

We identified various DevOps tools categories from the literature which are listed below:

1. **Source code management (SCM) Tools**: Source code management is a set of practices to track changes in the source code of the software. SCM is used for versioning and enabling teams to work together from different locations [40]. Basically, version control manages the changes in the source code, and controls the collaboration among developers during coding [40]. Source code management can be and should be used to track changes in any artifact that can evolve during the software development process, such as scripts for configuring systems or networks [42]. Through SCM tools developers can share their code and can work from multiple locations [40]. The most used and popular tool in version control is GitHub because it supports distributed systems and is a freemium and an open source tool. Other popular source code tools include Subversion, Mercurial, and Bitbucket [65].
2. **Build Tools**: Build is the process of preparing an executable program or set of programs out of a set of source code files for a particular software product [5]. This can involve the compilation of the source code into machine instructions for a specific computer architecture, but it can involve other steps such as handling dependencies. For example, in case the source code uses an external library. There are many tools available for the building the software application, such as Ant, Maven, Gradle, MS Build, NANT, [21, 79]. Gradle is the most widely used tool as it combines the features from other tools, such as of Ant, Maven, Gant and MSbuild [65].
3. **Continuous Integration (CI) Tools**: Developers integrate and merge code in an automated way. During this process, the code is submitted to the common source code of the software under development for building and testing. In case something went wrong, CI tools typically give related feedback immediately to alert the developers [40]. Travis CI, Jenkins, TeamCity, and Codeship fall into this category. Jenkins is one of the most popular tools among them[12].
4. **Configuration Management:** It is defined as the process of establishing and maintaining the consistency of software products throughout their life cycle [43], by tracking changes to any artifacts used and managing the different versions of each artifact [1]. For example, a user request for a new feature has to be tracked by the developers throughout its journey from a requirement to an implemented feature in the final software product. This may involve tracking what any related changes to the source code as well as the test that were written for this feature [1]. There are many tools available for configuration, such as Chef, Puppet, Ansible [65].
5. **Cloud Tools:** Cloud tools integrate deployment and collaboration to support DevOps practices, so they are often used by DevOps practitioners [34]. Popular cloud tools include but are not limited to Microsoft Azure and IBM services [34]. Amazon web services provide a variety of services for DevOps. For example, Amazon Beanstalk supports Continuous Deployment [12].
6. **Automated Testing:** Testing is an important process in automated DevOps pipeline. DevOps combines with cloud software and testing which is called Testing as a Service. It improves the collaboration and quality of the software product. DevOps always makes testing in a continuous manner in combination with automation [44]. Tools such as Cucumber, Selenium, JMeter etc. are some of the examples of testing tools [65].

7. **Containers:** Containers are generally used for developing the platforms and deploying the applications in the infrastructure. Containers reduce the time between the developing code and [1]production. For DevOps people containers are easily for deploying and maintaining. Docker Containers and Kubernetes for orchestrating the containers are the important ones. Containers are really helpful for DevOps developers as they lower overheads [80]. In microservices, each service is divided into smaller parts called micro. Developers and Operators work together on each smaller part so that coordination among team members will be improved. Containers helps microservices to deploy independently as they are operated in isolated environments [80].
8. **Deployment tools:** The purpose of these tools is to handle the deployment of new releases automatically. After every change in the source code, tests are performed, and new updated version will be released without manual work. Many of the big software companies Facebook, Netflix, GitHub uses continuous deployment to improve the efficiency of their work [40]. Tools like Capistrano, Jenkins, Ansible are used for deployment [79].
9. **DevOps Database Tools** Airaj, (2016)**:** Database management tools are responsible for handling the data, the metadata, procedures and the database schemas. In DevOps, databases can be used as code (DbaC), which means they are treated the same as the source code of the software under development and go through the same process, i.e. continuous delivery and continuous deployment. Tools that are often used for database management in DevOps include DBMaestro, MongoDB.
10. **Monitoring tools:** These tools monitor CPU load, RAM, memory space, and try to solve the infrastructure problems which might affect the business solutions [12]. Nagios is most used monitoring tools. Other than Nagios, NewRelic, Cacti is also used for monitoring [79]. Monitoring tools are more beneficial to the cloud applications [12]. Example for monitoring tools are New Relic, Graphite [65].
11. **Collaboration Tools:** DevOps is mainly established based on trust, open communications and good collaboration. DevOps encourages teams to share responsibility, ideas, and goals. Tools like Jira, Slack are mainly used for collaboration [78].

[1] A comprehensive list of identified tools, their categorization and characteristics is provided in: https://1drv.ms/x/s!AhtSO8VIshVwgVixidGVyPQUlUvr

3 RELATED WORK

Shahin, Ali Babar, and Zhu in [45], pointed out that there are many limitations in tools that support continuous practices, such as CI and CD, and continuous monitoring [46]. For example, CI tools do not properly support the activity of code review. Build and deployment tools have been reported to suffer from security and reliability issues by [45]. Moreover, they provide inadequate feedback during testing. Shahin, Ali Babar, and Zhu [45] also stated that there is lacking support in the automation of the development pipeline. In addition, using cloud tools for rapid release cycles has been reported to lead to reliability issues that lead to further delays in the development process [45].

Furthermore, limited support from tools has been reported for configuration [10], and monitoring [6]. Several authors in [6, 9, 48, 79] have reported that it can be challenging to integrate different tools in the same deployment pipeline.

Jones, Noppen, and Lettice [47] and Shahin, Ali Babar, and Zhu [45], have reported that the main barrier in implementing continuous practices is learning new tools. This is clearly an impediment to DevOps that relies on continuous practices to accomplish rapid release cycles [22]. Jabbari et al in [6] has also stated that deployment tools can be hard to use due to increase in demand for Continuous Integration and Continuous Delivery in software organizations.

Choosing the right tools is not a trivial matter, due to the wide variety of available options [7, 8]. At the same time, choosing the right tools has a great impact on the successful implementation of DevOps [8, 9, 45, 79].

Meanwhile, there seems to be a misconception that a single tool is sufficient to automate the entire deployment pipeline [8]. Typically, a variety of tools need to be used and integrated for continuous practices to be achieved. This impairs the challenge of selecting the appropriate tools for practicing DevOps.

The lack of knowledge in choosing the appropriate tools leads to another problem, according to [9] which relates to the developers and operators using different sets of tools, because it makes it hard to integrate them smoothly in the development process [9]. This problem has also been reported by [8].

Worse yet, it is not always possible to use the same tools even among the operations team. In particular, if a software product is intended to run in different production environments, these may have different requirements and may require different deployment tools, which leads to additional complexity [22].

Moreover, using different version of the same tools also leads to problems. For example, several tools have different versions under different licenses, e.g. free, enterprise, and premium, accompanied by different sets of features. This makes integrating tools harder and can even lead to problems with access rights [22].

Tools play a critical role in automating the DevOps deployment pipeline [12]. All the aforementioned problems make the practice of DevOps challenging. It is therefore clear that more research in the area of tools and related challenges in DevOps is needed. To the best of our knowledge, no research has focused on this topic. We therefore undertook this research in order to shed more light in the area of challenges faced by DevOps practitioners with respect to tools they use.

3.1 AIMS AND OBJECTIVES

Aims:
The main of this thesis is to investigate what are the challenges faced by DevOps practitioners with respect to tools they use as well as to compare those findings with related search.

Objectives:
To identify the set of tools that Developers and Operators use, as reported both by research (state-of-art) and by the industry (state-of-the-practice).
To identify the set of challenges that Developers and Operators face with respect to the tools they use, as reported both by research and by the industry.
To identify or formulate possible suggestions to the challenges of Developers and Operators with respect to the tools they use, as reported both by research and by the industry.

4 METHOD

4.1 Research Method:

We followed Wohlin and Aurum [49] guidelines for our research design structure. Three important phases compose the research design process: Strategy phase, Tactical phase and Operational phase. Below we describe the necessary steps we took for each phase.

Strategy phase: This phase enables the researchers understanding towards the selected topic and it involves the selection of four important steps. We described each step in the below sections.

Research Questions:

1. What tools do Developers and Operators use to carry out their tasks?
2. What tool-related challenges do Developers and Operators face when they carry out their tasks?
3. What recommendations were taken to overcome the tool-related challenges faced by Developers and Operators?
4. How effective were the recommendations taken to overcome those challenges?

Research Outcome:

There are two types of research outcomes namely, basic and applied research. Basic research refers to understanding of the problem based on the knowledge gained from the research but not necessarily providing solution to the problem. In our thesis, RQ1 & RQ2 belongs to basic research category. We formulated these research questions to understand what was happening in the software industries based on the knowledge we obtained from the literature.

RQ3 & RQ4 refers to applied research. Applied research is providing solution to the problem identified from the basic research. In RQ3 we tried to identify the recommendations to overcome the tool challenges faced by industry practitioners. And in RQ4 we identified whether these recommendations taken were effective.

Research Logic:

Research logic determines the direction and logical reasoning of the research. Our research follows the Inductive approach, also known as bottom-up research. Inductive research refers to understanding the theoretical concepts from the research and developing conclusions from the theory. In our thesis, we begin with understanding the DevOps concept, its practices, principles, tools and challenges and identified that literature has given little attention to the tool related challenges and aimed to draw conclusions or theories from our observations.

Research purpose:

The purpose of our study was primarily exploratory, as our primary objectives is to identify which tools are used by DevOps practitioners and what challenges are related to them. Our objectives also included identifying reasons for adopting certain practices, for example why would the development and operators not use the same tools. Hence, the purpose of our research was also explanatory to some extent.

Research Approach:

Research approaches are based on identifying the relation between the concepts and its categories in the specific domain, distinguishing between the beliefs and opinions by providing justification through methods. Our research approach is interpretivist. This approach allows the researchers to observe the human behavior aiming to provide a better understanding of the participant's perspective through different qualitative methods such as interviews and ethnographies.

Tactical Phase: In this phase we identify the solutions to investigate our research questions.

Research Process:

There are two types of research process namely, qualitative and quantitative. For our research method, we chose a qualitative approach. Questions like "what", "how" and "why" are more suitable for qualitative research methods, as stated in [53, 81].

Research Methodology:
According to Wohlin, Host and Henningsson in [54], the four major research strategies are experiment, case study, survey and post-mortem. From these, the experiment is a purely quantitative method, while the other three can be used both as quantitative and as qualitative methods.
We decided that a survey did not match our research goals and objectives with. Surveys aim to generalize the results found from the sample to the population, and are conducted in retrospective, also doesn't provide in-depth and descriptive information which we knew was not applicable in this context. Also, our aim is not to provide statistical analysis because from the literature we understood that DevOps is context dependent. Action research is also a kind of qualitative research, but it involves affecting in some way the phenomenon under study [53], which was never our intent. Our aim was to discover what was happening with DevOps in a real-life context, but not influence it in any way. Since we did not get any data from concluded project we did not choose post-mortem.
We reached this stage by rejecting the other research methods due to their limitations. Therefore, we decided to use elements from a case study. Host et al [53] states that a case study is feasible when the goal is to observe "a contemporary phenomenon within its real-life context", which matches well with our research objectives. Additionally, Runeson [53] claims that a case study is suitable for exploratory research questions, and three of our research questions are exploratory in nature.
Cases and Unit of Analysis:
Since we were expecting multiple participants from different companies, we planned to follow an embedded multi-case study [53]. We ended up taking interviews from seven different participants coming from seven different contexts. Each context is a case study because the participants varies in many factors such as application domain, company size and experiences which we clearly explained in (section 5.2 Table 11) and the units of analysis for each context were:

- the tools used by DevOps
- the challenges associated to these tool
- the recommendations that were taken to solve the challenges, if any

Triangulation:
Triangulation is an important step in qualitative case study. It means, collecting the data from different perspectives to reduce the validity threats of the research [53]. In our research, two researchers conducted a literature review and multiple interviews. Hence, we followed data triangulation and observer triangulation.
In particular, we carried out data collection from multiple participants and compared them to the literature sources (see section 5.1) rather than relying on data from a single source. we also carried out data collection from literature and compared them with the participants (see section 5.2) This is data triangulation.
Observer triangulation was achieved by having two observers participating in various steps of the research, instead of one, to reduce personal bias. These steps include conducting the interviews for data collection, transcribing the interviews for analysis, choosing the transcript that was more accurate of the two for each participant, performing thematic analysis on the transcripts independently and coming to an agreement on the results.
Replication:
Replication raises the validity of a research by making it possible for other researchers to replicate it. For that, the research needs to be transparent, so all its steps need to be clearly described. For case studies to be replicated, same theory must be supported by two or more participants. In our case study, our participants predicted the similar results for example, Tool challenges such learning new tools. We can say that our case study can be replicated.
We have attempted as best as we could to describe the steps we took and the motivation for taking these steps in the following parts of this section.
Operational phase: This phase describes the actions we took to conduct our research study.
Data collection Methods: We have two data collection methods namely, literature review and Semi structured Interviews. We used two methods because we followed data triangulation to improve the validity of our research.
Literature Review
We initially performed an ad-hoc literature review in the area of DevOps. Despite we did not use the

systematic literature review method, we employed systematic practices for primary studies selection and database searches to ensure the validity of the literature review. The main objective of our literature review was to investigate the state-of-art and state-of-research with respect to the tools and related challenges in the context of DevOps. This knowledge was required to explore the background and related work for our research. We collected the required sources from a wide variety of databases,

by searching through a common interface, the BTH Summon Library [1]. The interested reader can take a look at which databases are included at BTH Summon Library [2]

Study Selection Process

In this section, we describe the steps we followed to derive the studies from the available research. We used practices from Kitchenham in [50] to fill in the information of this section.

Study selection criteria

This section contains the criteria based on which we selected the studies that we used in this research, namely the inclusion and exclusion criteria. In particular:

Inclusion Criteria:

- Studies published in the last ten years, in an attempt to focus on more recent findings.
- Studies should be in the area of Computer Science and (Software) Engineering.
- Studies that are relevant to our research focus, which is tools and challenges in DevOps (more information on relevancy is provided below)

Exclusion Criteria:

- Studies that are not available in English.
- Studies that are not available in full text.
- Books or e-book types of publications were not considered, due to limited time scope.
- Duplicated studies

Below, you can see the searches we performed and the search terms that we used:

Search String	**Number of Retrieved Articles**	**Included studies**	**After Duplicates**
DevOps AND Adoption AND Challenges	280	26	24
DevOps AND Tools	1505	87	87
DevOps AND Tools AND Challenges	697	41	41
DevOps AND Challenges	916	62	60
DevOps AND Tool AND Automation	664	41	41
		257	112

Table 2: Search strings and number of articles

The above table 2 shows the relevant search strings and the number of the articles we found in the databases. We employed the following filters on the database search to match our inclusion/exclusion criteria:

- Full Text Online
- Content Type: excluded “Book / eBook”
- Discipline: included “engineering”, “computer science”
- Publication Date: 01/01/2008 – 31/12/2018
- Subject term: DevOps
- Language: English

The total articles we found during our initial search is 257 articles and after removing the duplicates we ended up with 112 articles. Out of these 112 research papers, we selected 51 relevant papers, i.e. studies that provide tools and challenges for DevOps.
To identify relevant studies in the context of this research, three levels of relevancy were considered based on title, based on abstract (when available), and based on keywords (when available).

Reliability of inclusion decisions:

There were cases when we disagreed about some of the studies on whether they should be included or not. In such cases, we had a discussion among us on a case by case basis, until we would reach an agreement. For instance, the study from Bang et al [51] seemed relevant, but we had a hard time understanding some of the content and how the conclusions were derived. We noticed that it had a high citation count in Google Scholar, so we originally thought it was an important paper, but later we realized the high citation count was not always for the paper's merits. In particular, it was mentioned in [4] that there was no validation in Bang et al [51] and the process followed was unclear. So, we agreed that including this paper would not raise the validity of our research.

Complementary Search:

After data extraction, we carried out a complementary search based on forward and backward snowballing search. The complementary search is intended, to identify relevant studies, a that were published as this research was being carried out, so as to reduce the risk of missing important contemporary studies from our research. This complementary search was based on the same searches as the original one. As a result, 35 studies such as [24, 32] were identified. However, this time, we included also some books, because we found from the initial and the first complementary search that some books were being referenced often, when important claims were made. As a result, we included books such as Huttermann [21]and Humble and Farley [5] as well.

Interviews:

According to Wohlin, Host and Henningsson in [54], the most common data collection methods are interviews and questionnaires. We chose interviews because it gathers useful and detailed responses from the interviewees regarding the investigated topic. Besides that, if an interviewee has difficulty in answering interview questions, the interviewer can provide clarifications. The interview also decreases the risk of not getting an answer in some interview question. Interviews take more time to carry out, but in our case where the number of responses we got was small, this was not a problem. We only conducted single source of study i.e. interviews for one single case study because we didn't have access to other artifacts in the industries such as projects and test cases. We also don't want much information about one single project from multiple participants instead we want sufficient information about each context and also want to increase the scope of various contexts from multiple participants. Also, data collection costs time but data analysis results more in terms of time.
We used semi-structured interviews [53]. We intended to ask the interview questions in a specific order, but we also wanted to allow the possibility for open questions and, possibly, additional questions that would not be pre-planned, in order to get additional information about the context of the participant. Finally, the interviews were conducted using Skype video calls. With the permission of the interviewees, the calls were recorded for data analysis at a later time.

Sampling

Interview requests were sent to practitioners through LinkedIn. LinkedIn is a large social network of professionals, with more than 610 million users of over 200 countries (https://about.linkedin.com/?trk=homepage-basic_directory). Through LinkedIn, several people with DevOps experience were identified. Sampling was limited only to DevOps practitioners because they might provide us the useful information, and to better understand the phenomenon. People from our own contacts as well as people outside our contacts were invited. Unfortunately, no responses were received from the latter ones. Interview requests were attempted to send all the participants through personal contacts. The selected participants were happened to be in Sweden. A total of 7 respondents agreed to take the interviews. If there is an increase in sample size, we might end up with more tools, challenges and recommendations which might increases the validity of our findings.

Interview Guide

First, based on our objectives and research questions, we formulated the questions of the interview. The table 3 below groups the questions in terms of aims/objectives and the motivation behind them.

	How big is your company? How many people are involved in software development? How many projects are you currently working on? What kind of applications do you develop? How is software being developed in your company? What processes do you use?	These questions were included in order to get a better idea of the context of each participant.
RQ2	Do you use DevOps in all your projects? Why not? What kind of projects in your experience are not suitable for using DevOps?	This set of questions was aimed at identifying possible challenges that made a company not want to use DevOps in some of its projects.
	How long have you been using DevOps? When did you start using it?	These questions are aimed at getting information about the participant's, as well about the company where the participant is employed, experience on DevOps.
RQ2	Did you face any challenges adopting DevOps? What challenges did you face? What suggestions would you give to other software development companies that want to start using, to avoid these challenges?	These questions were aimed at eliciting information about the adoption challenges that the company of the participant faced. The intent was to get information on challenges related to tools, but we left the question open, so as not to limit potentially informative and useful answers. The goal was to also get information about possible recommendations to deal with reported challenges.
RQ2	What benefits does DevOps bring to your company? Will you continue using DevOps? Why will you not continue using DevOps?	These questions were focused on getting insight on whether the benefits derived from practice of DevOps outweighs potential challenges, as well as insight on how practicing DevOps can be improved. The intent was to get information on how to mitigate any reported challenges, particularly with respect to tools; again, we left the questions open to invite more information from the participants.
RQ3	Have you found opportunities for improvements with DevOps? What opportunities to improve DevOps in your company have you found?	

RQ1	What kind of tools do your Developers use? What kind of tools do your Operators use? What is their respective goal with each tool? What is the purpose of each tool? What challenges is it meant to address?	This set of questions was primarily related to our first objective, namely to identify what tools are used in DevOps in practice.
RQ2	Do the Developers and Operators use the same tools? Why do they not use the same tools? Do the Developers and Operators use the same version of tools? Why do they not use the same version of tools? Do the Developers or Operators face any challenges related to the tools they use? What are these challenges?	These questions are aimed towards finding potential challenges that are related to the tools used in DevOps, and particularly, challenges when the developers and operators use different tools or different versions of tools, which leads to problems, as explained by [8, 9].
RQ3	Have you tried to solve these challenges? Why have you not tried to deal with these challenges? What recommendations have you taken to deal with these challenges?	These questions are all related to identifying possible recommendations to deal with challenges in DevOps related to tools, as well as their effectiveness and the context in which they are effective.
RQ4	Were these recommendations effective? Did these recommendations work for all your projects? Were they effective for some but not for other projects? Which projects were they ineffective and why?	
RQ4	Will you investigate the effectiveness of (further) possible solutions?	These questions, like the previous ones, were aimed at identifying possible recommendations to solve challenges related to DevOps tools. However, they refer to future recommendations that a company would be interesting to try out, so their effectiveness would be unclear. Asking

RQ3	Will you ignore the challenges (completely) and why? Will you stop using DevOps in these projects?	these questions, however, was considered important, in order to determine the severity of the challenges faced and whether they were worth investing resources to overcome, as well as ideas on how this could be done.

Table 3: Interview Questions and its motivation

RQ1	The light blue marks questions related to identifying which tools are used by DevOps practitioners.
RQ2	The light red marks questions related to challenges with DevOps practice, especially challenges related to DevOps tools.
RQ3	The light green marks questions related to possible recommendations that may be taken in order deal with reported challenges.
RQ4	The light-yellow marks questions related to the effectiveness of reported recommendations.

Table 4: Themes for each interview Questions

The interviews were planned for about 35 to 40 minutes. There would be a single round for each interview and, as explained previously, the interviews would be semi-structured. We attached the questionnaire in the Appendix C.

In each interview, there were three participants, the DevOps participant as the interviewee and the two researchers of this thesis as the interviewers. One of the researchers was recording the call and other researcher was involved in asking questions, in order to reduce the risk of bias, especially during open questions. Two researchers were involved since two researchers were involved in conducting data analysis. Finally, the interviews were entirely transcribed manually.

4.2 Data Analysis:

The main goal of data analysis is to draw conclusions from the data collected [53], namely the interviews we conducted. To reduce personal bias, both researchers carried out the data analysis independently.

Process for data analysis:

Thematic analysis is the process of identifying codes and themes on the data to support data analysis [53]. We used structural coding [55] first, in order to group the data into classes with similar characteristics, based on our research questions.

The chosen level of formalism used in our data analysis was the template approach [53] because we carried out the data analysis using themes and codes based on our research questions. In particular, we used one code for each different tool, challenge, and recommendations that was reported by the participants.

Furthermore, the tools were grouped into subcategories, according to their purpose/goal. For example, tools were grouped as Build, Configuration and Deployment. Challenges related to the tool were only identified. The reason for this was that the focus of our research was on tool-related challenges.

Intercoder Validity:

In order to achieve an acceptable level of reliability [56] and, hence, improve the validity of our research, we adopted certain practices. In particular, we strived for intercoder reliability [57] thus we coded the same data independently. Additionally, we strived for intercoder agreement [57], i.e. we discussed our two versions of structural coding of our transcripts, to identify any inconsistencies and come to an agreement for each separate case.

Process:

- We transcribed each interview independently.
- We discussed each transcript and kept one for each participant; the one that we thought was more accurate.
- We performed structural coding on each transcript independently.
- Later, we compared our resulting codes, and identified disagreements, that were either caused by (a) unitization Campbell et.al [57], i.e. we phrased our codes in different ways or (b) subjective interpretation, due to the lack of objective knowledge in the domain ofDevOps.
- In order to reduce the effect of unitization as much as possible, we repeated the coding for the parts of the transcripts we had disagreements on, until we reached an agreement.

We followed the guidelines of Campbell et.al [57] for intercoder validity.
We used simple method for calculating the intercoder validity i.e. Percent agreement. It is not usually recommended to go with this method. The most common method is Krippendorff's alpha coefficient. We did not choose this method because alpha mainly determines that all codes are used with equal probability. This doesn't account for our situation because DevOps is practiced differently in various organizations. Alpha also assumes that coders have equal capabilities and qualifications. In our case, one researcher might be more qualified than the other. But we have chosen simple percent agreement method because (a) since we had many codes to reduce the chance of agreement by coders (b) we also had multiple codes for one unit of text which is not suitable for complex methods such as Krippendorff (c) our main aim is to not to provide statistical analysis instead we want to provide systematic results which is suitable for qualitative analysis. It was also reported that for exploratory research percent agreement is acceptable [57].

$$P(Ao)=\text{Totals A'S}/N$$

Where
A= Each column represents the coding agreements of a particular coder for a particular variable.
N=Total No. of Variables.
By using the above formulae, we have made our intercoder validity.

As explained above, until we reach an agreement we repeated the coding process.

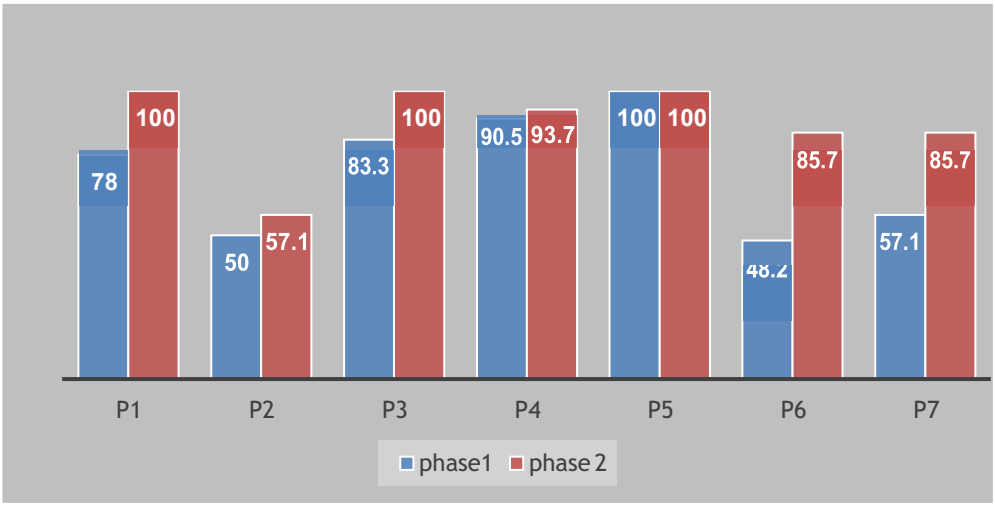
Participant	Tools	Challenges	Recommendations	Agreement
P1	100%	66.6%	66.6%	78.0%
P2	57.1%	66.6%	25%	50.0%
P3	100%	66.6%	-	83.3%
P4	81.1%	100%	-	90.5%
P5	100%	100%	-	100%
P6	71.4%	25.0%	-	48.2%
P7	57.14%	57.4%	-	57.14%
Average	81%	69.0%	46%	72.44%

Table 5: Intercoder validity percentage of each participant in Phase1

Participant	Tools Percentage	Challenges	Recommendations	Agreement
P1	100%	100%	-	100.0%
P2	57.1%	-	-	57.1%
P3	100%	100%	-	100.0%
P4	81.1%	100%	100%	93.7%
P5	100%	100%	100%	100%
P6	71.4%	100%	-	85.7%
P7	57.1%	100%	100%	85.7%
Average	81.0%	100%	100%	89.0%

Table6: Intercoder validity Agreement for Phase 2

The steps we took for reaching an agreement of our independent coding can be seen in below graph. Each row represents the agreement achieved for a particular interview, e.g. with participant P1.



We provided a small example for our intercoder reliability in the below table.

Interview Question	Provided answers	Codes	Themes
Q40 What are the challenges Developers and Operators face related to the tools they use?	if people don't use same kind of tools they might face **configuration issues** like if developer have something different infrastructure part it may not work in the operator's part or productions servers. So, if they don't use similar environmental structure operators cannot deploy their code.	Configuration issues multiple environments	Tool challenges

Table 7: Example for Intercoder Reliability

We did not consider this challenge because P6 mentioned that they are not facing challenges with configuration issues, but they might face if they don't use same kind of tools. This is more like a suggestion but not like a challenge.

5 RESULTS

In this section, we present our findings from our data collection methods. We first group our findings together based on the data collection method, namely the literature review and the interviews. We then group the findings further, based on our research questions.

5.1 Results from the Literature Review

RQ1: What tools do Developers and Operators use to carry out their tasks?

There are many tools identified during the literature review. Listing them all here would occupy too much space. However, we provide a link to a MS Excel spreadsheet instead (appendix B), where the tools are listed. For each tool, we include the tool's code, name, categories, license types, main purpose (high level description), as well as which software engineering role the tool aims to support, the literature it was derived from, and the participants that mentioned it.

In particular, depending on the activities of software development a tool supports, the tool is classified under one or more categories. Information about each category has already been provided in the section "Background and Related work".

Regarding licenses, each tool is available in one or more versions, each of which is bound to a different license type. Each license type has its own benefits and constraints. We summarize information about the different license types of each tool in the following table:

License Type	Description	Tool Example
Free	The tool is available with its full set of features without any monetary cost.	Git[2] (T164)
Open Source	The source code of the tool is available, and anyone can contribute to its development, by adding new features or fixing bugs. https://opensource.org/osd (these links need to go to footnotes)	Ansible[3] (T16)
Freemium	The tool's features are divided into basic, which are available for free, and premium, which require a fee. Source: http://www.businessdictionary.com/definition/freemium.html	Selenium[4] (T24)
Paid	The full set of features of the tool is available at a price.	Jira[5] (T39)

[2] https://git-scm.com
[3] https://www.ansible.com
[4] https://www.seleniumhq.org
[5] https://www.atlassian.com/software/jira

Enterprise	The tool is intended for use by an entire company. Source: https://www.upcounsel.com/enterprise-license-agreement	Puppet[6] (T14)

Table 8: Description on license types of DevOps tools.

Last but not least, the role that each tool is able to support can be:

- Dev: suitable for development activities
- Ops: suitable for operation activities
- Both: suitable for both development and operation activities

RQ2: What tool-related challenges do Developers and Operators face when they carry out their tasks?

During the literature review, we identified many challenges related to the adoption or practice of DevOps. Since our focus was on challenges related to DevOps tools, we list only tool-related ones in the table below, and we provide more information about them right after it.

Code	Challenge	Literature
CH1	Difficulty in choosing appropriate tools.	[48, 67, 68, 73, 81]
CH2	Learning new tools.	[67, 73]
CH3	Integrating different tools in the deployment pipeline.	[48, 68, 77, 81]
CH4	Public CI systems suffer from security issues.	[70]
CH5	CD tools can suffer from reliability issues.	[71]
CH6	Lack of mature tools.	[67, 69, 73, 75, 77]
CH7	Lack of tools that support multiple environments.	[22, 68]
CH8	Lack of tools that support CD in the domain of embedded systems.	[72]

Table 9: Tool-related challenges identified during the literature review.

It has been reported that not all DevOps tools are mature enough to properly support CD [67, 68,69,73,75,76]. In particular, continuous integration tools may be missing important features, which can limit the ability to integrate code changes as often as needed [69]. In addition, continuous integration tools may not be able to scale to support continuous deployment, which requires much more frequent build jobs [75]. Shahin et al in [77], show that tools cannot always properly support or automate effectively various software development activities, such as testing, configuration,

[6] https://puppet.com

monitoring, database management, and deployment [67]. Some tools may even introduce problems, which may result in the need to create additional tests [73]. As [68] explains, DevOps tools can lack maturity, because many of them are relatively new.

At the same time, certain kinds of tools have been reported to suffer from specific issues. In particular, [71] have reported on the reliability problems that CD tools, like Amazon Web Services (with Amazon CodeDeploy[7], T126), Elastic Compute Cloud [8](T165) and OpsWorks[9] (T166). Reliability in this context is related to error during the upgrading of a service from its current version to a new one, as well as how much time it takes for the upgrading to take place. In addition, security vulnerabilities have been reported by [70], for public CI systems, such as Jenkins[10] (T7). A public CI system is a system where multiple projects are hosted, and each user of the system may have access to one or more projects. This is typically the case for FLOSS (Free, Libre and Open Source Software) projects, as explained by [70]. Securing the deployment pipeline has also been reported as a challenge by [75]. Finally, specifically regarding embedded systems, [72] has reported that there is a lack of tools in general, and in particular, tools for deployment.

There are many tools that can be used to automate the deployment pipeline, which makes it hard to know which tools are more suitable to a particular project than others [48, 68, 77, 81]. For example, [68] mentions some decisions about tool selection, such as deciding on paying for tools or relying on free ones, as well as choosing cloud-based tools over tools used with the company's own servers. Even if a company decides to use free open-source tools, there is still a wide variety to choose from [81]. At the same time, as explained in the previous paragraphs, tools are not without their own problems. Since this knowledge is not readily available, it makes it even more challenging for practitioners to find the tools that are appropriate for their project. For example, if the team is using a testing tool that has bugs, then other tools have to be investigated [67].

Because of the aforementioned reasons, practitioners often have to experiment with a lot of tools [73], but learning new tools takes significant amounts of time. For example, [67] found that it takes longer for seasoned practitioners to adapt to new kinds of tools, such as cloud-based tools. One of the tools that was hard to learn was Puppet (T14) [67]. In addition, [67] showed that some tools require customization or configuration in order to integrate them in a particular project, which also takes time to learn.

After acquiring new tools, learning them is only one side of the coin. The other is about integrating the tools to the deployment pipeline [67, 68, 77, 81]. There are various reasons why a team needs to integrate several tools. Since there is no single tool that can automate the entire deployment pipeline, several tools need to be used to support and automate various activities [8]. Further, companies already have access to certain tools. When these are not enough to practice DevOps or when certain project needs require it, new tools must be acquired and integrated with the existing tools [67]. In addition, there are times when a software must be deployed in different production environments, but there is a lack of tools that can be used for deployment in different environments [22]. Having to integrate tools together and into a deployment pipeline seems unavoidable, but at the same time it has also been reported as an important challenge [67, 68,77, 81], because of the lack of compatibility between them [77, 81] as well as their level of maturity [68, 69, 73].

The need to use different tools in the same pipeline may lead to the predicament where different team members or different teams use different tools. This has been reported as an "anti-pattern for the very foundation DevOps is built on [68]. The difference between development, testing and production environments has also been reported as a challenge [22].

RQ3: What recommendations were taken to overcome the tool-related challenges faced by Developers and Operators?

Some recommendations that can be taken to address the tool-related challenges or reduce their impact are listed below:

---

[7] https://aws.amazon.com/codedeploy/

[8] https://aws.amazon.com/ec2/

[9] https://aws.amazon.com/opsworks/

[10] https://jenkins.io

Recommendations	Type	Challenge targeted	Literature
(R1) Organize meetings, seminars.	proposed	Difficulty in choosing appropriate tools (CH1) Learning new tools takes time (CH2)	[74]
(R2) Develop your own tools.	applied	Lack of mature tools (CH6) Difficulty in integrating different tools in the deployment pipeline (CH3)	[76]
(R3) Use certain tools to overcome the limitations of other tools.	applied	Difficulty in integrating different tools in the deployment pipeline (CH3)	[67]

Table 10: Recommendations targeting tool-related challenges from literature review.

[74] suggested that having Tech-Talks and knowledge-sharing sessions on DevOps topics, such as tools, can improve the awareness of the staff around such topics. They do not mention in this paper which particular challenges can be addressed by this recommendation, but based on our understanding, we matched these recommendations with the challenges CH1 and CH2.

The second recommendations we identified was from [76], where it is clearly stated that two large organizations, namely OANDA and Facebook, could not meet their needs by relying exclusive on the tools available at that time. They claimed that existing tools were too limited (CH6) or could not fit into the organization's existing process (CH3). Instead, they developed their own deployment systems. Thirdly, [67] found that certain tools, such as CloudShell and Helion CloudSystem can be used to address the lack of compatibility between existing tools and cloud-based tools. Some of the interviewees in [67] stated that they used this recommendation.

Aside from these recommendations we have identified some more, that target specific issues regarding the deployment pipeline. For example, the works of [70] and [75] both target security problems with DevOps, but the recommendations proposed are highly technical and can be countered by malicious attackers, this is a testament to the difficulty and effort required to engineer proper secure systems for supporting CD. The work of [71] also has a technical discussion on possible recommendations that can improve the reliability of continuous deployment, and we encourage the interested reader to study their work. However, since we are by no means experts in the areas of security and reliability, it was not possible for us to provide comprehensive information on what the recommendations are and deduce how effective or ineffective they can be.

RQ4: How effective were the recommendations taken to overcome those challenges?

In the previous section, three recommendations were identified. Regarding R2 (Develop your own tools), [76] implied that this recommendation was both effective and necessary, since no other tools in existence could adequately meet the needs of the two organizations OANDA and Facebook. At the same time, however, [76] report that this is by no means an easy solution to the challenges of CH3 and CH6. They show that a considerable part of the engineers of these companies were dedicated to the development of the necessary tools. Therefore, they claim that this recommendation is likely to benefit only organizations of a size comparable to that of OANDA and Facebook.

Unfortunately, [67] has not provided any information about the effectiveness of R3, namely the use of tools like CloudShell[11] to help integrate cloud-based tools with the deployment pipeline.

[11] https://cloud.google.com/shell/

5.2 Results from the Interviews

Before we present the data gathered during our interviews, we provide some information about the participants themselves and the companies that employed them at the time of the interviews.

ID	Role	Interviewee experience in DevOps	Company experience in DevOps	Applications	Company size
P1	development and operation activities	3 months	1 year	Web services	20 people
P2	development activities	2.5 years	2.5+ years	unknown (Java applications using the Boot framework)	unknown (reported as medium to large)
P3	Backend development activities	2.5 years	2.5 years	An e-book streaming app	>200 people
P4	development and leadership activities (lead developer)	5 months	6 years	telecommunications and broadband services, web portal	>1000
P5	development and operation activities	1 year	1 year	Wide variety of applications, including computer games and web applications	>5000
P6	development and operation activities	1 month	3.5 years	Data processing (including image processing), with web interface	<15
P7	development and operation activities	4 years	5 years	Mobile and web applications	20

Table 11: Demographics of interview participants.

In the text that follows, we use names for the company of each participant, i.e. Comp1 for P1, Comp2 for P2, etc.

Based on the data we collected from each participant, it is clear to us that the context of each participant varies as do the companies they work for. The companies range from small start-ups, such

as Comp1 and Comp 6, to large multinational companies, like Comp5. This indicates the company size is not necessarily a limiting factor in practicing DevOps.
The data do not, however, show the same variation in application domain. It could be sheer coincidence, but all the companies of participants we interviewed were involved in web development to some extent. More precisely:

- Comp1 was developing a web service
- Comp2 was developing Java applications with Spring Boot framework, which is used for web applications (and supports DevOps by integrating with Docker[12])
- Comp3 was developing an e-book streaming application
- Comp4 was developing applications in the telecommunications domain, but also a web portal
- Comp5 was developing various applications, including web applications (even though P5 was involved in a project for an electronic game)
- Comp6 was developing a satellite data management application with a web interface
- Comp7 was developing mobile web applications

This is not a big surprise considering that web development is well-suited for practicing DevOps.
Likewise, the development process used by the teams of the participants seem to follow a common theme, that of agile. More precisely:

- The teams of P1 and P7 used agile principles, though no details were provided, while P2's team used 2-week sprints, including planning and retrospectives
- The teams of P3, P4, P5 and P6 are all using Scrum.

Naturally, we were not surprised by this, since we had already learnt from the research that DevOps is an extension of agile methods and shares values but has a greater scope [20].

With respect to DevOps, P1, P3, P6 and P7 actively practice it. P2 also mentioned that they practice DevOps actively, but not for legacy projects. This matches Researcher1's report that claims it is challenging to use DevOps legacy with systems [79]. P4 also practices DevOps on an experimental level. They use it primarily for automation, but culturally, they have not been able to fully overcome the silos between the development and operations, where there are still misunderstandings, such as who has the responsibility for manual testing. On the other hand, Comp5 does not want to commit to a full- fledged adoption without first trying out some of its principles and investigating situations that could turn into adoption challenges, such as potential communication problems. Hence, even though P5 claimed that Comp5 does not actively practice DevOps, one of the responsibilities of P5 and his team is to identify ways to make a potential adoption smoother.

Considering what P4 claimed about Comp4's failed attempt to transition fully to DevOps in the past, when planning suffered due to lacking experience in DevOps, we consider this to be a smarter approach and, perhaps, a lesson for other companies that wish to adopt it.

The experiences from participants vary from 1 month to 4 years, with an average of 1.5 years. In summary, P2, P3 and P7 are the most experienced DevOps practitioners with 2.5 years of experience or more, while Comp7 and Comp4 are the most experienced companies in DevOps, with 5 years of practice or more.

RQ1: What tools do Developers and Operators use to carry out their tasks?
Regarding the tools that our interview participants mentioned, all the related information is in the same spreadsheet as the one linked to in the corresponding section of the results from the literature review. All the reader has to do, is uncheck the checkbox "Blanks" in the column "Participant", which will only show the tools mentioned by at least one participant. Below we provided a table of popular tools mentioned by participants as well as literature.

[12] https://www.docker.com

Tool	Category	Literature	ID
Git	Version control, Source Code	[9, 34, 59, 60, 63, 79]	P1, P2, P3, P4, P5, P6, P7
Slack[13]	Collaboration	[9, 65, 79].	P1, P4
Amazon Web Services	Containerization, Configuration, Logging and Monitoring, Deploy, infrastructure as Code	[19, 62,59, 65]	P1, P4, P6
Ansible	Deploy, Configuration	[34, 61, 62, 65, 79].	P1, P7
GitHub	Source Code, Containerization, Version Control	[9, 19, 34, 62, 63, 64, 65, 79].	P2, P3
Jenkins	Build, CI server, Deploy	[9, 19, 34, 59, 61, 62, 63, 64, 65, 79].	P2, P4, P6
Selenium	Testing	[62, 65, 79].	P4, P6
Docker	Containerization	[19, 62, 65, 79].	P4, P6, P7
AWS ECS	Containerization	[65]	P6, P2

Table 12: List of tools mentioned by participants and literature.

Seven participants mentioned that they are using Git. P2 and P3 were using Git for version control and P1, P4, P5, P6 and P7 were using for Source Code. Git is used for integrating the developer's code. Next important tool mentioned by participants was Amazon Web Services. P1 was using Amazon Web Services for infrastructure as code, P4 was using it for deploying whereas, P6 was using for containerization. Amazon web services provides are cloud tools and provides many services for practitioners such as build, deploying the applications, providing frequent feedback, automating the entire deployment pipeline. P2 mentioned that they are using Jenkins for build whereas P4 and P6 were using for continuous integration. Another important tool was Docker which was used for containerization by respected participants. It was worth noticing that all the tools were reported in the literature as well.

RQ2: What tool-related challenges do Developers and Operators face when they carry out their tasks?

During the interviews we carried out, our participants reported a small number of tool-related challenges. These are listed in the table below:

Code	Challenge	ID
CH1	Difficulty in choosing appropriate tools.	P1, P6, P7
CH2	Learning new tools.	P1, P3, P4, P6, P7
CH6	Lack of mature tools.	P5

[13] https://slack.com/intl/en-se/

CH9	Limitations with tool support.	P7

Table 13: Tool-related challenges reported by the interviewees.

The most pronounced challenge among our participants was related to learning new tools (CH2). For instance, P1 explained that getting used to new tools, such as Ansible and Amazon Cloud Watch, took time. P4 is part of a Scrum team that is trying to move to DevOps. He explains: "*When we are using DevOps, we have to use different tools, adopting to those tools is little difficult. We could say we have to spend time on learning those tools. Tools is a big issue. We have to learn it, we have to install it and there would be a lot of time waste*." The participants P3, P6 and P7 also emphasized that it takes time to learn new tools, which affects the productivity of a development team. In particular, CH2 was mentioned by:

- P1 for the tools Ansible (T16) and Amazon cloud (T54)
- P3 for the tool Elastic Search[14] (T158)
- P4 for the tools Jenkins (T7), Puppet (T14) and Amazon Web Services (T57)
- P6 for the tool Amazon Web Services (T57)
- P7 for the tool Docker (T18)

Three out of the seven participants, namely P1, P6 and P7, mentioned that it was challenging for them to know how to choose tools appropriate for their projects (CH1). For example, P1 stated "*There are many tools available in the market. But we need to choose the appropriate one which suits for your project*", while P6 described the situation with more detail: "*We need to find the right and appropriate tools to save time and to develop the product in a right way. We are looking forward to work with many tools and we will try to figure out which tools do our job in a better way. Finding the right and appropriate tools is a challenge for those who want to adopt DevOps*."

Regarding the maturity of tools (CH6), P5 mentioned that some of his colleagues found Git to be somewhat complicated.

Last, but not least, P7 explained that there can be limitations with tool support (CH9). He described the situation where a bug may need fixing, which could take time. If the bug is a security vulnerability, this puts the provided service at risk and could hurt the users and the customers. He also explained that after a patch becomes available, a restart of the affected services is required. This means service downtime, which is undesirable.

Another point worth making is that Git (T164) was reported by all of our participants, but only P5 mentioned that it was found to be complicated by some of his team members. Therefore, we are uncertain how big impact this has on the productivity of engineers using Git in general.

A few of our questions were aimed at eliciting challenges, based on whether the operators and developers use the same tools and, if so, the same version of tools. As we have already stated before, we have found that when the two teams use different tools, their collaboration becomes harder [68]. In summary, P1, P3, P5 and P6 reported that the development and operations all use the same tools and the same version of tools, while P2, P4 and P6 reported the opposite. When asked why, they replied that the developers and operators use different tools, because they have different tasks to do. When asked about potential challenges because of this, no one reported any.

In particular, P2 said that the developers and operators use some common tools, but when it comes to different tasks, they use different tools. P4 explained that they use different tools to integrate their work, since the development and operations use different tools. When asked about problems pertinent to integrating their work, none was reported. The only thing that was reported by P4 was the problematic situation where the operations think that the developers should do manual testing, while the latter think otherwise. We believe that this issue is not related to tools; rather it is related to organizational silos. To us, the developers and operators in Comp4 have not yet reached a point where their collaboration is based on trust and their goals are not yet aligned, as advocated by DevOps [21, 22]. We bring this point up in this section again to clarify our position further. Finally, P7 also said

[14] https://www.elastic.co

that it is not required for developers and operators to have the same tools when they need to perform different tasks.

RQ3: What suggested recommendations were taken to overcome the tool-related challenges faced by Developers and Operators?
Some of our participants mentioned suggested recommendations that were taken to address certain tool-related challenges. We list them in the table below:

Recommendations	Challenge targeted	ID
Organize meetings, seminars. (R1)	Learning new tools. (CH2) Lack of mature tools. (CH6)	P4, P5
Use certain tools to overcome the limitations of other tools. (R3)	Limitations with tool support. (CH9)	P7

Table 14: Recommendations targeting tool-related challenges from literature review.

With respect to the first recommendations that was taken, P4 mentioned that a DevOps expert from Finland was hired for a week to hold seminars that would educate the staff in DevOps practices as well as tools (CH2). In particular, P4 mentioned that the expert helped with the tools Amazon Web Services (T57), Jenkins (T7) and Puppet (T14). In the context of P5, workshops and monthly meetings were organized so as to educate the staff on better usage of tools, such as Git that was reported as complicated by some in P5's team.
Regarding P7, he stated that using containers can help avoid downtime during service upgrades. He explained: "*But with this technology growth when kubernetes came to industry and Docker come into picture and kubernetes integrated with Docker they started giving roll back updates. You can start your update without making any downtime. Your old code will be there, and new code will also be there. When the old code is ready to finish their connections, they will be killed from the containers level and they will bypass their connection from old containers to new containers.*"

RQ4: How effective were the suggested recommendations taken to overcome those challenges?
The recommendations of bringing in a DevOps expert to conduct seminars has been reported as very successful. More specifically, P4 claimed that this move made an observable difference and, due to the knowledge gained from the expert, it became easier to meet project deadlines. P5 also claimed that the workshops and monthly meetings were effective, he could not provide us with information that shows a considerable improvement.
In relation to P7's context, it is clear that containers can indeed be used effectively to avoid service downtime during service upgrading. That is an inherent capability with the tool itself. Hence, P7 reported this as an effective recommendation to deal with the downtime problem.

6 DISCUSSIONS

In this section, we compare the results we got from our two data collection methods, namely the literature review and the interviews, and we relate them to our research questions. Before that, however, we make a few points about the context of our findings.

Firstly, the number of participants was small, namely seven. This makes it particularly hard to draw a clear picture of this research area. Further, the participants' experience in DevOps was limited. Four of them, namely P6, P1, P4 and P5, had one year of experience or less. This could mean that their understanding of DevOps tools and related challenges may not be as mature as it would be otherwise. This, in turn, could mean that some of our findings do not accurately reflect the actual context of these participants.

Secondly, despite the fact that the respective companies of our interviewees are diverse in size, they are not diverse in application domain; nor in methodology. Most of the participants' projects were related to web-development. Regarding the development methodology, all 7 of our participants reported agile principles and practices, while 4 of them stated that they use Scrum practices. These observations led us to believe that our findings may be more relevant to this context. For instance, the challenges faced, or recommendations taken to address them may be more suitable for web development in agile settings, but perhaps not as applicable to embedded systems development.

Tools in DevOps (RQ1)

When we started our research, we set out to identify as many tools as possible that are used in the practice of DevOps. We found a large number of over 166 tools that are used to support or automate different activities of the software development process, from version control to deployment and monitoring. As has already been explained, this information is too much to fit in this paper, but we provide a link to a spreadsheet in the (appendix B) that lists these tools. In this list, the reader can find tools identified by the literature or reported by the participants or both.

It is worth pointing out that, from the total sum of 166 tools we identified, 30 were reported by the participants, and out of these 30 tools, 7 were not found during our literature review namely, the tools Intellij (T159), Neo4j (T160), Amazon S3 (T161), Amazon Cognito (T162), Amazon Machine Images (T163), Grafana (T60) and Postman (T61).

There might be various reasons for this. For example, the research sometimes refers to any Amazon tool simply as Amazon Web Services (T57), instead of referring to the particular tool provided by Amazon, such as Amazon CloudWatch (T54). Since different tools have different roles, we have decided to list them independently.

Another assumption why these 7 tools were not found during the literature review is that some of these tools may be known as standard development tools, rather than tools with capabilities for supporting continuous practices. For instance, Intellij, which was reported by P1, is a well-known IDE (Integrated Development Environment) for Java development[15]. Some of its features though make it easy to integrate to the development pipeline. For instance, it integrates with version control systems, such as Git (T164) and Perforce (T128), with build tools, such as Maven and Ant, and even allows management of Docker containers, via a plug-in. Therefore, even though it may be mostly known as an IDE, Intellij supports DevOps practices such as CI and CD.

In an attempt to provide more information about what purpose its tool has, we include the category or categories of each tool in the spreadsheet. For example, readers that are only interested in tools for "build" jobs can focus on this category. Second, we provide a short description of each tool in order to highlight its main purpose or features from a DevOps perspective. For example, for the tool CloudWatch (T54), we include the information that "it is a monitoring tool for developers, operators and for IT managers. It allows you to monitor the applications, infrastructure and optimizing the resources.

Zeng, Yan and Jin in [48] identified, among many others, the plethora of tools that can be used to support DevOps. They claimed that grouping the tools together can aid in finding tools for specific

[15] https://www.jetbrains.com/idea/

purposes, and they illustrate this with several tools and six categories. We hope to extend this work with the information we have provided in our spreadsheet
Finally, the fact that we found 7 tools that were not reported by the literature indicates the possibility of even more such tools that are in use by the industry. Even though the tools we found address our research question for now, it is clear to us that this is and will remain an open topic for the foreseeable future. Some tools become outdated, as [68] have found, while new tools are being developed and existing tools become more mature. We expect the landscape of DevOps tools to keep changing until it converges to a set of mature, reliable, highly customizable tools that can be easily integrated with the deployment pipeline.

Tool-related challenges in DevOps (RQ2)

We have identified challenges related to DevOps tools by either of our two data collection methods. These challenges are summarized below:

Code	Challenge	Literature	ID
CH1	Difficulty in choosing appropriate tools.	[48, 67, 68, 73, 81]	P1, P6, P7
CH2	Learning new tools.	[67, 73]	P1, P3, P4, P6, P7
CH3	Integrating different tools in the deployment pipeline.	[48, 68, 77, 81]	
CH4	Public CI systems suffer from security issues.	[70]	
CH5	CD tools can suffer from reliability issues.	[71]	
CH6	Lack of mature tools.	[67, 69, 73, 75, 77, 76]	P5
CH7	Lack of tools that support multiple environments.	[68, 22]	
CH8	Lack of tools that support CD in the domain of embedded systems.	[72]	
CH9	Limitations with tool support.		P7

Table 15: Tool-related challenges from both the literature review and the interviews.

Under no circumstance can we claim that the tool-related challenges that were commonly reported both the research and by the industry have more significance than the others. However, the former ones may be more common that the latter. Further, we can provide more contextual information about them. For instance, lacking mature tools (CH6) can be due to various reasons, such as the tools lack

important features [69], but it could also be because the tools may be complicated and, as such, hard to use (P5). This information is given in the sections "Results" of the respective data collection methods.

From the challenges mentioned by the participants, limitations with tool support (CH9) was the only one not found during our literature review. This may be because some researchers may have considered it as part of lacking mature tools (CH6). Alternatively, since empirical findings in the area of tool-related challenges are scarce, it may be that this challenge simply was not discovered until now. In the context of P7, it is related to the time it takes for bugs to be fixed, as well as the fact that every patch may require restarting services, hence lead to service downtime. P7 talked about open source tools, but we argue that the same situation can easily affect proprietary tools; they too can have bugs and need maintenance.

On a different note, our understanding from the challenges we identified is that they are actually related. This relation may be stronger in some contexts than in others, but there seems to be a pattern. For example, lack of mature tools (CH6) that can meet certain needs of a specific project makes it challenge to find appropriate tools (CH1). Similarly, tools suffering from security (CH4) or reliability issues (CH5) also make it hard to know which to use. This in turn forces DevOps practitioners to seek out more and more tools in a quest to find appropriate ones, which makes them have to learn new tools (CH2) as well as to try to integrate them in their deployment pipeline. This is also a challenging endeavor (CH3), due to the new tools not being mature enough (CH6) and/or not having appropriate support (CH9) from their creators. This leads the DevOps engineers to be on the lookout for more tools that may be more mature with less bugs or security issues and more features, which leads to the problem of not knowing which tools to choose and having to learn new tools.

To us, this cycle means that companies that practice or want to adopt DevOps should be mindful of these tool-related challenges and their inter-dependencies. The rationale is that even if one of these challenges seems to be low-risk in a certain context, it may be exacerbated by other challenges. For example, having mature tools with proper support that do not support multiple environments, creates the need for searching for additional tools, which means these tools will have to be learned and integrated, as well as maintained sufficiently.

Finally, the fact that our participants found a tool-related challenge, which was not found during the literature review, is an indication for us that there may be other challenges that have not been discovered yet. Considering certain factors, such as the small number of industry participants that got involved in our research, the rather homogeneous application domain pertaining their context, the limited experience of some of them in DevOps, it is quite likely that other tool-related challenges remain to be found. For instance, in none of our participants' context was a company developing and maintaining its own tools, so any challenges related to that were not found. Nor were any of our participants involved in embedded systems development. Even though we think our research is a good first step in discovering how the landscape of tool-related challenges in DevOps looks like, we cannot conclude that our research question has been fully answered.

Addressing tool-related challenges in DevOps (RQ3 and RQ4)

Without a doubt, there is a very small number of recommendations discovered with either during both of our data collection methods. The literature review gave us three recommendations, while the participants mentioned two of them. We summarize this information in the following table:

Recommendations	Challenge targeted	Literature	ID
(R1) Organize meetings, seminars.	(CH1) Difficulty in choosing appropriate tools. (CH2) Learning new tools. (CH6) Lack of mature tools.	[74]	P4, P5

(R2) Develop your own tools.	(CH3) Difficulty in integrating different tools in the deployment pipeline. (CH6) Lack of mature tools.	[76]	
(R3) Use certain tools to overcome the limitations of other tools.	(CH3) Difficulty in integrating different tools in the deployment pipeline. (CH9) Limitations with tool support.	[67]	P7

Table 16: Possible recommendations for tool-related challenges in DevOps.

In summary, organizing workshops, seminars and/or having knowledge sharing meetings has been reported by P4 and P5 as an effective way of dealing with the challenges of learning new tools (CH1). We think that increased education in DevOps tools can also help with the challenge of choosing appropriate tools (CH2) as well as using tools that lack in maturity more effectively (CH6).

In large software development companies that can afford to invest the resources necessary, DevOps tools can be developed and maintained in-house to suit the specific needs of these companies. Such tools can have the desirable level of maturity (CH6) and they can be tailored to integrate smoothly with the company's deployment pipeline (CH3). However, the company should evaluate first how many resources in manpower and software/hardware are required for such an endeavor, as empirical evidence shows it can be a significant investment.

Last but not least, the use of certain tools can overcome the limitation of others, for instance tools like CloudShell can be used to help integrate cloud-based tools with the existing deployment pipeline (CH3), while containerization can help with software upgrades with no downtime (CH9).

Regarding the recommendations effectiveness, [74] claimed that knowledge-sharing meetings and Tech-Talks (R1) would be useful in addressing knowledge gaps with tool-related challenges, and the participants P4 and P5 provided empirical support for this. [76] provided a real-world example where developing your own tools (R2) may be the better, if not the only, choice, albeit an expensive one. Finally, using specific tools to help deal with the problems of other tools has received empirical support from both the interviewees of [67] and one of our own, namely P7.

The very small number of potential recommendations that can be used to address the challenges we found gives us no confidence to say that our research question 3 has been answered successfully. Of course, these recommendations are relevant to our questions, but it is apparent to us that this is still an open issue. For example, we did not find any recommendations that address security or reliability issues. Such issues do not only cause delays [71] that can lead to customer dissatisfaction, but they can lead to assets being compromised with potentially catastrophic results [70], [75].

With respect to recommendations effectiveness, here too we have little to present, as only a few recommendations have been identified. All the recommendations we have found in our research have been reported as effective. For (R1), it has been proposed by [74], but our interviews have provided empirical support for it (P4 and P5). Likewise, (R2) and (R3) have also received empirical support from the literature, while (R3) in particular was also reported as effective by P7.

It is worth mentioning though that using additional tools to address the issues of existing ones (R3) is not a trivial matter. Firstly, identifying such tools is not easy (CH1). Secondly, throwing even more tools into the mix means even more reasons to worry about tool support. Thirdly, these tools have to be learned as well (CH2) before they can be used effectively. For instance, as [67] shows, tools often need to be configured or customized first, before they can be integrated in the deployment pipeline.

Regarding (R1), P4 was very clear about the observable difference it had on the development. P4 explained that it was easier to meet deadlines after the DevOps expert came in to help educate the staff. Regarding (R2), it seems that OANDA and Facebook had no other choice, so its effectiveness is self-evident. We reason that if these companies could have achieved the desired outcome, that is to find appropriate tools for their projects, without the need to invest the considerable resources required for in-house tools, they would have definitely done so. Similarly, when it comes to (R3), both

interviewees in [67] and our own P7 have applied it to good effect. In both cases, the recommendations seem to have achieved the desirable outcome.
However, we should not forget that there might be other reasons why a particular recommendation can be reported as effective. For example, P7 stated that relying on containers is not always feasible, which means that the effectiveness of containerization to avoid downtime during service upgrades is context dependent. He explained that in a different company that he worked for in the past, this was not possible, so an alternative had to be found. This included bypassing user traffic from servers running the old version to servers running the new version.
This may not be the only reason why recommendations are reported by interviewees as effective. There is also the potential for bias. An employee working in a company might not be willing to mention things that did not work very well for them, or he might be willing to exaggerate for things that worked to some extent. We got this suspicion from P5, who explained that workshops and monthly meetings were an effective recommendation, despite the fact that he could not provide any details about it. To the best of our knowledge, P5 may have heard but not necessarily observed any visible results.
Finally, considering P4's claim about (R1) being very effective, it is not entirely clear how effective it was. The DevOps expert may have helped the teams to meet deadlines, but the misunderstandings between development and operations about whose responsibility it is to perform manual testing is a sign that there is still room for improvement in how DevOps is practiced in Comp4. The collaboration between development and operations is the most fundamental aspect of DevOps [32], so if is still not well understood by the staff in Comp4, we are a bit reluctant to accept that the DevOps expert (R1) managed to provide adequate education in tool-related matters as well. Perhaps Comp4 needs to repeat (R1) and possibly enhance it by conducting more seminars and knowledge-sharing meetings. Meeting deadlines, as we understand it, is one of the things the software engineers care most about, so that may be the reason why (R1) was reported as very effective.
In any case, we believe that our fourth research question has only been partially answered, since the recommendations reported were few, as were the times they were applied to good effect. To us, this clearly indicates that more work needs to be done in this area.

6.1 Threats to Validity:

Internal validity:
When casual relationships are investigated, this aspect of validity is concerned. When a researcher is investigating the one specific factor which affects the other factor, there is a possibility the investigated factor might affect another factor.
Initially we performed literature review using search strings such as DevOps AND Tools, DevOps AND Challenges related to our domain area based on the inclusion criteria. There is a risk of missing important articles because of this reason. To mitigate this risk, we carried a complementary search by including another search string DevOps AND Adoption AND Challenges to educate more on the research domain. We also carried a secondary complementary search called backward and forward snowballing to identify more relevant articles. By carrying out this search we included books Huttermann [21] and Humble and Farley [5] to increase our relevance criteria. There is still a chance we might have missed high quality and relevant papers, so to mitigate this we have chosen books and journals to reduce the risk to certain extent.
There is a chance of validity threats to interview data due to the selection of interviewees, framing the questionnaire, misinterpretation of data collection and lack of experience in conducting interviews. To mitigate this, we selected the participants from the software companies practicing or practiced DevOps, we selected the participants through the LinkedIn and some personal contacts. Questionnaire was constantly updated with open ended and closed questions. Our questionnaire is designed with yes or no questions, even though few other questions were explanatory in nature, there is a chance we might have missed detailed answers from our data collection. Misinterpretation of data is reduced by recording the interview calls with participant's permission and both the researchers has transcribed

independently. We also sent follow up questions at a later date after the interviews. Unfortunately, not all of them responded.
Another validity threat could be there is a chance of low participant engagement. We couldn't obtain necessary data for our research objectives. Even though we sent follow up questions not all of them responded.
One of our study limitations is we didn't perform the thematic analysis for literature review. This could be our future work.
Another validity threat could be due to low percentage of intercoder validity. To mitigate this, we repeated the coding process to reach an accepted level of percentage and also compared our findings with the literature to support our study. But we still we have discrepancies in our agreement. This is one of our study limitations.
External Validity: This validity refers to what extent the results can be generalized. We conducted interviews for only seven participants due to limited amount of time, the results can hard to generalize. To mitigate this as we mentioned in the research methodology section, our aim is not to generalize the results. Also, from the literature Smeds, Nybom and Porres [22] Lwakatare et al [31] Humble and Farley [5] we understood that DevOps depends on the context and varies in different organizations and we took necessary steps to replicate our research study.
The other validity threat in our work could be identifying few recommendations and effectiveness for tool related challenges. To mitigate this, we sent follow up questions to our participants but not all of them responded. This is could be one of our study limitations.
Construct validity: This validity refers to the extent to which operational measures being studied really represents what the researcher has in mind. There is a chance of misinterpreting the purpose of the questionnaire by the interviewees. To mitigate this risk, we have sent an invitation letter for interviews describing our objectives, aims and purpose of conducting the interviews. Questions were designed with clear motivation which relates to our research objectives. We also gave a brief introduction of ourselves, as well as interview purpose and procedure in case they missed reading questionnaire.
Regarding the challenges identified in the literature, there is a high risk of misinterpreting the challenges due to the lack of experience in research domain. For example, Lack of training in DevOps process. We first identified this challenge in Jabbari et al [6] but we couldn't find any information about the mentioned challenge. Through backward snowballing Jabbari et al. [6], referenced Farroha et al [86] but this paper also lacks the adequate information to support the mentioned challenge. Hence, we did not consider this challenge. To mitigate this risk for other challenges as well, we used backward snowballing to identify the original papers which mentioned the potential challenges. Through this we are able to understand the context of the challenges and excluded few challenges which doesn't meet our research objectives. We selected challenges only which described the context clearly and generic challenges were excluded. This is also one of our study limitations.
Regarding the tools we identified from the literature, there is a high chance of eliciting wrong information or misunderstanding one tool as other. Due to inexperience of the researchers initially we misinterpreted GitHub and Git as single tool. We identified this inconsistency during our data analysis. Few of our participants mentioned Git, and others as GitHub. In an attempt to mitigate this threat, the researchers seek information about several tools online in their respective websites, so that they would be more capable to understand and evaluate related information that would come from the participants during the interviews.
Regarding the purpose of each tool, tool websites provided multiple purposes for each tool. Since it is highly technical it was difficult for the us to understand each purpose. Hence, we only provided the information based on our understanding. This is one of our study limitations.

Regarding validity threats, language is a big issue. Not only use of common language varies from person to person, but also technical terms are used in different ways by different people. There is a chance that we may have misinterpreted or misunderstood a situation as big a challenge when it is not or vice versa. This could be one of our study limitations.

Reliability of the study: Reliability if study could be affected by many factors such as if the interview questions are unclear or due to improper coding in the transcripts. To mitigate this, as mentioned above interview questions were designed with clear motivation, structural coding was used to code the different research questions, and also, we followed triangulation to collect the data from the multiple sources. Both researchers were involved in data collection process to reduce the personal bias.

7 CONCLUSIONS

We used two data collection methods, based on a case study in order to identify the tools that DevOps practitioners use, the challenges related to these tools, possible recommendations to help deal with these challenges, as well as contextual information about their effectiveness.
Our results have yielded over 166 tools, while 7 of them have only been found by our interviews. This could be for a number of reasons, such that standard tools gained at a later time features that support DevOps practices, such as CI or CD. In any case, we expect the landscape of DevOps tools to change, as old tools either become more sophisticated or become outdated, and as new tools are developed to support DevOps better and integrate with the deployment pipeline more easily. We tried to provide enough information that can help the interested reader to get a better idea of what tools are available, what is their main purpose and how accessible they are in terms of license.
Regarding the tool-related challenges we found, both the literature review and our interviews show that lack of knowledge on how to choose appropriate tools for DevOps practice, as well as the time it takes to learn them, are common challenges. Other challenges include limitations with tools, such as tools suffering from security or reliability issues. These challenges seem to remain mostly unaddressed. Our findings indicate that commonly accepted recommendations to these challenges have yet to be found. We conclude that they are dealt with on a case-by-case basis, if at all, though further research is needed to confirm this. Few tool categories such as AIOps, Analytics, Databases we couldn't find the information about these categories in the literature. Our future work also includes the providing more information about the tool categories.
We identified very few recommendations to deal with some of the tool-related challenges in DevOps. The most feasible one seems to be conducting seminars or other knowledge-sharing events to educate practitioners on how to better use DevOps tools and, potentially, how to identify suitable tools for their projects. Another recommendation, aimed at dealing with the limitations of existing tools in the deployment pipeline, is to use other appropriate tools depending on the context, though this recommendation may have its own caveats as well. Lastly, a company may choose to develop its own tools to suit the needs of its projects, but that is unlikely to work for small or medium companies, and it may not even be justified for large companies, since it requires a considerable investment.
In summary, our research questions have only been partially answered. From RQ1 to RQ4, we are less and less satisfied with our findings. We believe that the volume of knowledge on the topic of tool- related challenges in DevOps has not received adequate attention from the research community yet, and we hope that our research will help to generate new initiative for further research in this area.

7.1 FUTURE WORK

We have several ideas about future research in the area of tool-related challenges in DevOps. As explained in the discussion, our research questions have not been fully answered and we believe there is still much to learn.
Firstly, as explained previously, we expect the landscape of DevOps tools to keep evolving, as new tools arrive at the scene and old ones disappear from it. Therefore, we believe it is worth repeating and improving this research in the future, in order to keep track of the evolution of DevOps tools.
Secondly, there is a lot of work to be done in the area of tool-related challenges. We estimate that there are more such challenges out there. Their identification will be like adding an important piece of the puzzle. As explained in the discussion section, our limited number of participants and their demographics show that the sample is small and not representative of the population, so it is highly likely there are several other challenges left to be discovered yet. Perhaps quantitative studies would help draw a clearer picture and allow for some generalization from the results.
Further, we think that a deeper analysis is needed when it comes to these challenges. Shahin et al, (2017) has done good work in showing that various challenges in DevOps are related to one another, for example one challenge related to practicing DevOps in distributed settings exacerbates the risk of ineffective collaboration and transparency between development and operations. We have already reasoned about that some tool-related challenges are related in certain ways. For example, lack of mature DevOps tool that are available makes it harder to find appropriate tools for a particular project.

Therefore, we believe that a deeper analysis will provide more clarity on how these challenges are related to each other and if recommendations need to be taken to address several challenges as a group.
Thirdly, finding appropriate recommendations to tool-related challenges and how effective they are, is still an open issue. Our findings yielded only 3 potential recommendations, 1 of which is rather infeasible to many software development companies, i.e. (R2). Clearly, more research is needed in this area as well.
In addition, certain areas such as security and reliability issues with DevOps tools need further research. These areas are more focused and more technical. The recommendations described may have more context-based applicability. Still, until commonly acceptable solutions are found to address these challenges, we think this is an open topic that requires further research.
Not only further research is needed to provide more answers to our research questions, but to also improve upon our work. Our research has no lack of limitations, which are described in the section "Validity threats", and they can be used to drive better research in the area of DevOps tools and related challenges. We consider our research to be a good first step in this rather unexplored area, and we encourage researchers to investigate it further.

8 REFERENCES

[1] Sommerville, Ian. "Software engineering 9th Edition." ISBN-10 137035152 (2011).
[2] B. Boehm, “A view of 20th and 21st century software engineering,” in Proceedings of the 28th international conference on software engineering, pp. 12–29, 2006.
[3] B. Tessem and J. Iden, “Cooperation between developers and operations in software engineering projects,” in Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, pp. 105–108, 2008.
[4] F. M. A. Erich, C. Amrit, and M. Daneva, “A qualitative study of DevOps usage in practice,” Journal of Software: Evolution and Process, vol. 29, no. 6, p. e1885, 2017.
[5] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader). Pearson Education, 2010.
[6] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, “Towards a benefits dependency network for DevOps based on a systematic literature review,” Journal of Software: Evolution and Process, vol. 30, no. 11, p. e1957, 2018.
[7] Wettinger, Johannes, Vasilios Andrikopoulos, and Frank Leymann. "Automated Capturing and Systematic Usage of DevOps Knowledge." Proceedings of the IEEE International Conference on. IEEE Computer Society, 2015.
[8] A. Qumer Gill, A. Loumish, I. Riyat, and S. Han, “DevOps for information management systems,” VINE Journal of Information and Knowledge Management Systems, vol. 48, no. 1, pp. 122–139, 2018.
[9] G. Bou Ghantous and A. Gill, “DevOps: Concepts, Practices, Tools, Benefits and Challenges,” PACIS2017, 2017.
[10] K. Nybom, J. Smeds, and I. Porres, “On the impact of mixing responsibilities between devs and ops,” in International Conference on Agile Software Development, pp. 131–143, 2006.
[11] P. A. Nielsen, T. J. Winkler, and J. Nørbjerg, “Closing the IT Development-Operations Gap: The DevOps Knowledge Sharing Framework.,” in BIR Workshops, 2017.
[12] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps,” *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016.
[13] T. E. Bell and T. A. Thayer, “Software requirements: Are they really a problem,” in Proceedings of the 2nd international conference on Software engineering, pp. 61–68, 1976.
[14] P. A. Laplante and C. J. Neill, “The demise of the waterfall model is imminent and other urban myths,” ACM Queue, vol. 1, no. 10, pp. 10–15, 2004.
[15] W. W. Royce, “Managing the development of large software systems. proceedings of IEEE WESCON,” Los Angeles, pp. 328–388, 1970.
[16] W. Royce, Software project management. Pearson Education India, 1998.
[17] A. Manifesto, “Manifesto for agile software development,” 2001.
[18] T. Chow and D.-B. Cao, “A survey study of critical success factors in agile software projects,” *The Journal of Systems & Software*, vol. 81, no. 6, pp. 961–971, 2008.
[19] B. B. N. de França, H. Jeronimo Junior, and G. H. Travassos, “Characterizing DevOps by hearing multiple voices,” in Proceedings of the 30th Brazilian Symposium on Software Engineering, pp. 53– 62, 2016.
[20] A. D. Nagarajan and S. J. Overbeek, “A DevOps implementation framework for large agile-based financial organizations,” in OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”, pp. 172–188, 2018.
[21] M. Huttermann, DevOps for developers. Apress, 2012.
[22] J. Smeds, K. Nybom, and I. Porres, “DevOps: a definition and perceived adoption impediments,” in International Conference on Agile Software Development, pp. 166–177, 2015.
[23] P. Debois, “DevOps: A software revolution in the making,” Journal of Information Technology Management, vol. 24, no. 8, pp. 3–39, 2011.

[24] C. Jones, “A Proposal for Integrating DevOps into Software Engineering Curricula,” in Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, pp. 33–47, 2018.
[25] Cilliers, Frans, and Henk Greyvenstein. "The impact of silo mentality on team identity: An organisational case study." SA Journal of Industrial Psychology 38.2, pp.75-84, 2012.
[26] J. Erickson, K. Lyytinen, and K. Siau, “Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research,” JDM, vol. 16, no. 4, pp. 88–100, Oct. 2005.
[27] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, and T. Männistö, “DevOps Adoption Benefits and Challenges in Practice: A Case Study,” in Product-Focused Software Process Improvement, pp. 590–597, 2016.
[28] Dybå, Tore, and Torgeir Dingsøyr. "Empirical studies of agile software development: A systematic review." Information and software technology 50.9-10, pp. 833-859, 2018.
[29] M. A. Cusumano, A. MacCormack, C. F. Kemerer, and W. Crandall, “Critical Decisions in Software Development: Updating the State of the Practice,” IEEE Software, vol. 26, no. 5, pp. 84–87, Sep. 2009.
[30] Fitzgerald, Brian, and Klaas-Jan Stol. "Continuous software engineering: A roadmap and agenda." *Journal of Systems and Software* 123, 176-189, (2017)
[31] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Relationship of DevOps to agile, lean and continuous deployment: A multivocal literature review study,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10027, pp. 399–415, 2016.
[32] Erich, Floris. "DevOps is Simply Interaction Between Development and Operations." International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. Springer, Cham, 2018.
[33] DeGrandis, Dominica. "DevOps: A Software Revolution in the Making?", 2011.
[34] Airaj, Mohammed. "Enable cloud DevOps approach for industry and higher education." Concurrency and Computation: Practice and Experience 29.5, e3937, 2017
[35] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, “What is DevOps? A Systematic Mapping Study on Definitions and Practices”, in Proceedings of the Scientific Workshop Proceedings of XP2016, New York, NY, USA, pp. 12:1–12:11, 2016.
[36] L. BANICA, M. RADULESCU, D. ROSCA, and A. HAGIU, “Is DevOps another Project Management Methodology?” Informatică economică, vol. 21, no. 3, pp. 39–51, Jan. 2017.
[37] V. Ivanov and K. Smolander, “Implementation of a DevOps pipeline for serverless applications,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11271, pp. 48–64, 2018.
[38] S. Neely and S. Stolt, “Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy),” in 2013 Agile Conference, pp. 121–128, 2013.
[39] W. Hussain, T. Clear, and S. MacDonell, “Emerging Trends for Global DevOps: A New Zealand Perspective,” in 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE), pp. 21–30, 2017.
[40] S. Uzunbayir and K. Kurtel, “A Review of Source Code Management Tools for Continuous Software Development,” in 2018 3rd International Conference on Computer Science and Engineering (UBMK), pp. 414–419, 2018.
[41] Swartout, Paul. Continuous Delivery and DevOps–A Quickstart Guide: Start your journey to successful adoption of CD and DevOps. Packt Publishing Ltd, 2018.
[42] P. Swartout, Continuous Delivery and DevOps: A Quickstart Guide. Packt Publishing Ltd, 2014.
[43] U. Ali and C. Kidd, “Configuration management process capabilities,” Procedia CIRP, vol. 11, pp. 169–172, Jan. 2013.
[44] Angara, J., S. Prasad, and G. Sridevi. "The factors driving testing in DevOps setting-a systematic literature survey." Indian Journal of Science and Technology 9.48: 1-8, 2017.
[45] M. Shahin, M. Ali Babar, and L. Zhu, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices,” IEEE Access, vol. 5, pp. 3909–3943, 2017.

[46] S. A. I. B. S. Arachchi and I. Perera, “Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management,” in 2018 Moratuwa Engineering Research Conference (MERCon), pp. 156–161, 2018.
[47] S. Jones, J. Noppen, and F. Lettice, “Management challenges for DevOps adoption within UK SMEs,” in Proceedings of the 2nd International Workshop on quality-aware devops, 2016, pp. 7–11.
[48] J. Zheng, L. Yan, and L. Jin. "Exploring DevOps for Data Analytical System with Essential Demands Elicitation." SEKE, 2016.
[49] C. Wohlin and A. Aurum, “Towards a decision-making structure for selecting a research design in empirical software engineering,” Journal of Empirical Software Engineering, vol. 20, no. 6, p. 1427, 2015.
[50] Keele, Staffs. Guidelines for performing systematic literature reviews in software engineering. Vol. 5. Technical report, Ver. 2.3 EBSE Technical Report. EBSE, 2007.
[51] S. Bang, S. Chung, Y. Choh, and M. Dupuis, “A grounded theory analysis of modern web applications,” in Proceedings of the 2nd annual conference on research in information technology, pp. 61–62, 2013.
[52] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in Proceedings of the 18th International Conference on evaluation and assessment in software engineering, pp. 1–10, 2014.
[53] M. Höst, A. Rainer, B. Regnell, and P. Runeson, Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, 2012.
[54] C. Wohlin, M. Host, and K. Henningsson, “Empirical research methods in software engineering,” EMPIRICAL METHODS AND STUDIES IN SOFTWARE ENGINEERING: EXPERIENCE FROM ESERNET, vol. 2765, pp. 7–23, 2003.
[55] M. J. Belotto, “Data Analysis Methods for Qualitative Research: Managing the Challenges of Coding, Interrater Reliability, and Thematic Analysis,” The Qualitative Report, vol. 23, no. 11, pp. 2622–2633, Nov. 2018.
[56] K. Krippendorff, “Agreement and Information in the Reliability of Coding,” Communication Methods and Recommendationss, vol. 5, no. 2, pp. 93–112, Apr. 2011.
[57] J. L. Campbell, C. Quincy, J. Osserman, and O. K. Pedersen, “Coding In-depth Semistructured Interviews,” Sociological Methods & Research, vol. 42, no. 3, pp. 294–320, Aug. 2013.
[58] R. Vaasanthi, S. Philip and V. Prasanna, " Comparative Study of DevOps Build Automation Tools, " International Journal of Computer Applications, vol. 170, (7), pp.5-8, 2017.
[59] M. Soni, “End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery,” in 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 85–89, 2015.
[60] B. Vasilescu, S. van Schuylenburg, J. Wulms, and A. Serebrenik, “Continuous integration in a social-coding world: Empirical evidence from GitHub. Updated version with corrections,” Dec. 2015.
[61] Akshaya, H. L., J. Vidya, and K. Veena. "A Basic Introduction to DevOps Tools." *International Journal of Computer Science & Information Technologies* 6.3 (2015): 05-06.
[62] Vaasanthi, R., V. Prasanna Kumari, and S. Philip Kingston. "Analysis of Devops Tools using the Traditional Data Mining Techniques." International Journal of Computer Applications 161.11, 2017.
[63] M. Ohtsuki, K. Ohta, and T. Kakeshita, “Software engineer education support system ALECSS utilizing DevOps tools,” in Proceedings of the 18th International Conference on information integration and web-based applications and services, pp. 209–213, 2016.
[64] F. Ahmadighohandizi, and S. Kari. "ICDO: Integrated cloud-based development tool for DevOps." SPLST, 2015.
[65] “XEBIALABS EXTENDS JENKINS FOR DEVOPS TEAMS & TOOLS,” Computer Workstations, vol. 31, no. 10, Oct. 2018.
[66] J. Sandobalin, E. Insfran, and S. Abrahao, “ARGON: A Tool for Modeling Cloud Resources,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10797, pp. 393–397, 2018.
[67] Amaradri, Anand Srivatsav, and Swetha Bindu Nutalapati. "Continuous Integration, Deployment and Testing in DevOps Environment." (2016).
[68]. Hamunen, Joonas. "Challenges in adopting a Devops approach to software development and operations." (2016).

[69] A. Debbiche, M. Dienér, and R. B. Svensson, “Challenges when adopting continuous integration: A case study,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8892, no. Journal Article, pp. 17–32, 2014.
[70] V. Gruhn, C. Hannebauer, and C. John, “Security of public continuous integration services,” in Proceedings of the 9th International Symposium on open collaboration, pp. 1–10, 2013.
[71] Liming Zhu et al., “Achieving Reliable High-Frequency Releases in Cloud Environments,” IEEE Software, vol. 32, no. 2, pp. 73–80, 2015.
[72] L. E. katare et al., “Towards DevOps in the Embedded Systems Domain: Why is It So Hard?,” in 2016 49th Hawaii International Conference on System Sciences (HICSS), pp. 5437–5446, 2016, vol. 2016.
[73] H. H. Olsson, H. Alahyari, and J. Bosch, “Climbing the ‘Stairway to Heaven’ -- A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software,” in 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, pp. 392–399, 2012.
[74] P. Perera, M. Bandara, and I. Perera, “Evaluating the impact of DevOps practice in Sri Lankan software development organizations,” in 2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 281–287, 2016.
[75] P. Rimba, Liming Zhu, L. Bass, I. Kuz, and S. Reeves, “Composing Patterns to Construct Secure Systems,” in 2015 11th European Dependable Computing Conference (EDCC), pp. 213–224, 2015.
[76] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, and M. Stumm, “Continuous deployment at Facebook and OANDA,” in Proceedings of the 38th International Conference on software engineering companion, 2016, pp. 21–30.
[77] M. Shahin, M. A. Babar, M. Zahedi, and Liming Zhu, “Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges,” in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Recommendationsment (ESEM)*, 2017, pp. 111– 120.
[78] Lwakatare, Lucy Ellen, Pasi Kuvaja, and Markku Oivo. "An exploratory study of devops extending the dimensions of devops with practices." *ICSEA 2016* 104 (2016).
[79] Bucena, Ineta, and Marite Kirikova. "Simplifying the DevOps Adoption Process." *BIR Workshops*. 2017.
[80] S. Maheshwari *et al*, "Comparative Study of Virtual Machines and Containers for DevOps Developers," 2018.
[81] M. Borrego, E. P. Douglas and C. T. Amelink, "Quantitative, qualitative, and mixed research methods in engineering education," *Journal of Engineering Education,* vol. 98, *(1),* pp. 53-66, 2009.
[81] J. Wettinger, U. Breitenbücher and F. Leymann, "DevOpSlang - bridging the gap between development and operations," in 2014.
[82] Jiménez, Miguel A., et al. "Deployment Specification challenges in the context of large scale systems." CASCON. 2017.
[83] Zhu, Liming, Len Bass, and George Champlin-Scharff. "Devops and its practices." IEEE Software 33.3 (2016): 32-34.
[84] Board, Qualifications. "Certified Tester Foundation Level Syllabus." (2010).
[85] Miglierina, Marco. "Application deployment and management in the cloud." 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. IEEE, 2014.
[86] Farroha, B. S., and D. L. Farroha. "A framework for managing mission needs, compliance, and trust in the DevOps environment." 2014 IEEE Military Communications Conference. IEEE, 2014.

APPENDIX-A

A LIST OF CHALLENGES WITH TOOL ID, CATEGORY, LICENCE TYPE, LITERATURE AND PARTICIPANTS

ID	Tool	Category	License Type	Literature	Participants
T1	Apache Subversion	Source Code	Open Source	[79], [59], [9]	
T2	GitHub	Source Code, Containerization, Version Control	Freemium	[79], [60], [9], [62], [63], [64], [65], [34], [19]	P2, P3
T3	StarTeam	Source Code	Freemium	[59]	
T4	Ant	Build	Open Source	[79], [58], [59], [6], [9], [61], [62], [63], [34], [19]	
T5	Maven	Build	Open Source	[79], [58], [59], [6], [9], [61], [62], [64], [34], [19]	
T6	Travis CI	Build, CI server, Deploy	Open Source, Paid, Enterprise	[79], [9], [61], [65], [34]	P3
T7	Jenkins	Build, CI server, Deploy	Open Source, Free	[79], [59], [9], [61], [62], [62], [63], [64], [65], [34], [19]	P2, P4, P6
T8	MsBuild	Build	Open Source	[58], [9], [62]	
T9	Gradle	Build	Freemium, Enterprise	[79], [58], [61], [62], [34], [19]	
T10	Phing	Build	Open Source	[58]	
T11	Atlassian Bamboo	Continuous Integration, Orchestration, Deploy, Testing	Paid	[79]	
T12	Subversion	Source Code	Open Source	[65], [34]	
T13	CodeShip	Continuous Integration	Freemium	[79], [9], [65]	
T14	Puppet	Deploy	Enterprise	[79], [61], [62], [64], [65], [34], [19]	P4
T15	Gitlab	Source Code	Open Source, Freemium	[79]	P6
T16	Ansible	Deploy, Configuration	Open Source Paid	[79], [61], [62], [65], [34]	P1, P7
T17	Chef	Configuration Deploy	Enterprise	[79], [59], [6], [61], [62], [65], [34], [19]	P7

T18	Docker	Containerization	Open Source, Freemium, Enterprise	[79], [62], [65], [19]	P4, P6, P7
T19	Vagrant	Containerization, Configuration	Open Source	[79]	
T20	Saltstack	Configuration	Open Source	[79], [62], [65], [34]	
T21	Argon	Infrastructure as code	Open Source	[66]	
T27	Nagios	Monitoring	Open Source	[79], [9], [62], [65], [34]	
T28	NewRelic	Monitoring	Freemium	[79], [9], [62], [65], [19]	
T29	Graphite	Monitoring	Open Source	[61], [34]	
T30	Ganglia	Monitoring	Open Source	[34]	
T31	Rake	Build	Open Source	[34]	
T32	Papertrial	Log Monitoring	Open Source	[9], [61]	
T33	Splunk	log Monitoring	Free, Enterprise	[79], [61], [62], [65]	
T34	Sumo Logic	log Monitoring	Freemium	[61], [65]	
T35	Logstash	log Monitoring	Open Source	[61], [62], [65], [34]	
T36	Kibana	log Monitoring	Open Source	[61], [65], [34]	
T37	Slack	Collaboration	Freemium	[79], [9], [65]	P1, P4
T38	Hip Chat	Collaboration	Freemium	[79], [9]	
T39	Jira	Collaboration	Paid	[79], [3], [62], [65], [19]	P4
T40	MangoDB	Databases	Free, Open Source, Enterprise	[9],	P4
T41	LiquiBase	Databases	Open Source	[79], [34]	
T42	RedGate	Databases	Enterprise	[79], [65]	
T43	DBMaestro	Databases	Enterprise	[79], [65]	
T44	TeamCity	Continuous integration, Orchestration	Freemium	[79], [6], [62], [65]	
T45	Circle CI	Deploy, Continuous Integration	Freemium	[79], [65]	
T46	JMeter	Testing	Freemium	[58], [62]	
T47	Cacti	Monitoring	Open Source	[62]	
T48	OpenStack	Cloud	Open Source	[65]	
T49	Zabbix	Monitoring	Open Source	[79], [65], [19]	
T50	Capistrano	Deploy	Open Source	[79]	
T51	PagerDuty	Communication	Paid	[79], [65], [19]	P2
T52	TestComplete	Testing	Paid	[79]	
T53	Mercurial	Source Code	Free	[79], 17]	
T54	CloudWatch	Deploy	Freemium	[62], [19]	P1
T55	Azure Watch	Cloud	Paid	[59], [65], [34]	
T56	AppDynamics	Monitoring, Analytics	Freemium	[79], [9], [61], [62], [65]	
T57	Amazon Web Services	Containerization, Configuration, Logging, Monitoring	Freemium	[59], [62], [65], [19]	P1, P4, P6
T58	VMware vSphere	Configuration	Paid	[59]	
T59	VMware	Configuration	Paid	[59], [34]	

	vCloud Director				
T60	Grafana	Monitoring	Open Source, Enterprise	Nil	P2
T61	Postman	Testing	Free, Enterprise	Nil	P6
T62	kubernetes	Deploy	Open Source, Free	[82], [65]	P3
T63	Rancid	Configuration	open Source	[61]	
T64	CFEngine	Configuration	Open Source	[61], [62], [65], [34]	
T65	Bitbucket	Source code, version control	Freemium	[79]	P4
T66	Graylog	Logging	Open Source, Enterprise	[62]	
T67	Rundeck	Monitoring Deploy	Open Source, Enterprise	[62], [64]	
T68	NUnit	Testing	Open Source	[6], [62]	
T69	TestNG	Testing	Freemium	[62]	
T70	JBehave	Testing	Open Source	[62]	
T71	RALLY	Testing, Project Management	Free, Enterprise	[62]	
T72	Redmine	Project Management	Free, Open Source	[63]	
T73	Gocd	Build, Deploy	Open Source, Free, Enterprise	[83], [65]	P1
T74	ISPW	Source code	Enterprise	[65]	
T75	Artifactory	Source code	Open Source	[62], [65]	
T76	Nexus	Source code	Open Source	[62], [65]	
T77	Delphix	Databases	Enterprise	[65]	
T78	Flyway	Databases	Open Source	[65], [34]	
T79	VSTS	Continuous Integration,	Freemium	[65], [34]	
T80	FitNesse	Testing	Open Source	[65]	
T81	Gating	Testing	Open Source	[65]	
T82	Mocha	Testing	Open Source	[65]	
T83	Karma	Testing	Freemium	[65]	
T84	Jasmine	Testing	open Source	[65]	
T85	Tricentis Tosca	Testing	Freemium	[65]	
T86	Locust io	Testing	Open Source	[65]	
T87	Perfecto	Testing	Paid	[65]	
T88	Sauce Labs	Testing	Paid	[65]	
T89	SoapUI	Testing	open source	[65]	
T90	Micro Focus UFT	Testing	Enterprise	[65]	
T91	Packer	Configuration	Open Source	[65]	
T92	Rudder	Configuration	Open Source	[65]	
T93	Terraform	Monitoring	Freemium	[65]	P7
T94	Octopus Deploy	Deploy	Enterprise	[65]	
T95	UrbanCode Deploy	Deploy	Enterprise	[65]	
T96	Electric cloud	Deploy	Enterprise	[65]	
T97	ElasticBox	Deploy	Enterprise	[65]	
T98	CA	Deploy	Enterprise	[65]	

	Automatic				
T99	XL Deploy	Deploy	Enterprise	[65]	
T100	Google Cloud	Deploy	Enterprise	[65], [34]	P3
T101	OpenShift	Cloud	Enterprise	[65]	
T102	Alibaba Cloud	Cloud	Paid	[65]	
T103	IBM Cloud	Cloud	Freemium	[65], [34], [19]	
T104	Cloud Foundry	Cloud	Open Source	[65]	
T105	Iron.io	Cloud	Paid	[65]	
T106	Apache Open Whisk	Cloud	Open Source	[65]	
T107	Lambda	Cloud	Paid	[65]	
T108	Zenoss	Monitoring	Enterprise	[65]	
T109	Checkmarx SAST	Security	Enterprise	[65]	
T110	Snort	Security	Open Source	[65]	
T111	Signal Sciences	Security	Enterprise	[65]	
T112	Tripwire	Security	Open Source	[65]	
T113	Black Duck	Security	Enterprise	[65]	
T114	CyberArk Conjur	Security	Enterprise	[65]	
T115	SonarQube	Security	open source	[62], [65]	
T116	Veracode	Security	Enterprise	[65]	
T117	HashiCorp vault	Security	Open Source	[65]	
T118	Fortify SCA	Security	Enterprise	[62], [65]	
T119	ServiceNow	Collaboration	Enterprise	[65]	
T120	Trello	Collaboration	Freemium	[65], [19]	
T121	Stride	Collaboration	Freemium	[65]	
T122	Remedy	Collaboration	Enterprise	[65]	
T123	Agile central	Collaboration	Enterprise	[65]	
T124	OpsGenie	Collaboration	Paid	[65]	
T125	CollabNet VersionOne	Collaboration	Enterprise	[65]	
T126	AWS CodeDeploy	Deploy	Freemium	[65]	
T127	Coverity	Code Analyzer	Paid	[62]	
T128	Perforce	Version Control	Freemium	[62]	
T129	JSHint	Code Analyzer	Open Source	[62]	
T130	CheckStyle	Code Analyzer	Open Source	[62]	
T131	Blueoptima	Code Analyzer	Freemium, Enterprise	[62]	
T132	Clover	Code Analyzer	Paid	[62]	
T133	Semmle	Code Analyzer	Paid	[62]	
T134	JaCoCo	Code Analyzer	Free	[62]	
T135	Gerrit	Code Analyzer	Free	[62]	
T136	PMD	Code Analyzer	Open Source	[62]	
T137	SharePoint	Collaboration	Paid	[62]	
T138	Archiva	Artifact Repository	Free	[62]	
T139	IBM Rational Clear Case	Configuration	Enterprise	[62]	

T140	Hudson	Build, CI Server	Free	[19]	
T141	PingDom	Monitoring	Freemium, Enterprise	[19]	
T142	CloudWave	Cloud for Orchestration	Paid	[19]	
T143	Glu	Build, Deploy, Monitoring	Free, Open Source	[19]	
T144	Mesos	Containerization	Open Source	[65]	
T145	Rancher	Containerization	Open Source	[65]	
T146	Docker Enterprise	Containerization	Enterprise	[65]	
T147	GKE	Containerization	Paid	[65]	
T148	AKS	Containerization	Paid	[65]	
T149	Rkt	Containerization	Open Source	[65]	
T150	Codefresh	Containerization	Freemium	[65]	
T151	Helm	Containerization	Open Source	[65]	
T152	Fluentd	AIOps	Open Source	[65]	
T153	Prometheus	AIOps	Open Source	[65]	
T154	ITRS	AIOps	Enterprise	[65]	
T155	Moogsoft	AIOps	Paid	[65]	
T156	Dynatrace	Analytics	Enterprise	[65]	
T157	Datadog	Analytics	Enterprise	[65]	
T158	Elasticsearch	Analytics	Open Source, Freemium	[65]	P3
T159	Intellij	Java Integrated Development Environment	Free, Open Source	n/a	P1
T160	Neo4j	Databases	Enterprise	n/a	P1
T161	Amazon S3	Databases	Freemium	n/a	P1
T162	Amazon Cognito	Authentication	Freemium	n/a	P1
T163	Amazon Machine Images	Logging	Freemium	n/a	P1
T164	Git	Version Control, Source Code	Free, Open Source	[79], [59], [60], [9], [63], [34],	P1, P2, P3, P4, P5, P6, P7
T165	AWS Elastic Compute Cloud	Cloud	Freemium	[71]	
T166	OpsWorks	Configuration	Freemium, Enterprise	[71]	

APPENDIX - B

A LIST OF TOOLS WITH ID, NAME AND ITS PURPOSE

ID	Tool	
T1	Apache Subversion	It is a version control tool which allows users to make edits on working files in repositories and make commits to the SVN server.
T2	GitHub	GitHub is generally used for developers to work together globally. it is greatly used for integration, team and project management, code reviews, and documenting etc.
T3	StarTeam	It contains centralized locations to update source code, provides monitoring, automates software delivery and brings developers and operators together through common change management.
T4	Ant	It is a java-based deployment tool written in XML used for compiling in JAR files
T5	Maven	It can solve problems which Ant cannot. it allows to download dependencies
T6	Travis CI	It is used to deploy, and to test the software throughout the world. it supports pull request and branch build flow. It integrates with GitHub.
T7	Jenkins	Jenkins is built in java-based program and provides plugins for build, deploy, test and automation. it serves as CI server and can be deployed anywhere in the world. it integrates with Docker and GitHub.
T8	MsBuild	It allows to build software which orchestrates scripts. it contains a several functionalities which can be accessed through command line.
T9	Gradle	It performs many operations such as build, automated executing and deploys faster.
T10	Phing	It is a build tool works on windows, Mac and extends in PHP classes.
T11	Atlassian Bamboo	It provides hundreds of plugins for integrations, build, deploys and automated test. it integrates with Jira, Bitbucket.
T12	Subversion	It manages and stores the files, directories in a centralized server and can be used by people over the world
T13	CodeShip	It has a simple interface which allows to build and deploy anywhere. it integrates with Docker, slack, and Gitlab.
T14	Puppet	Puppet allows you to discover resources, configure and manage the data, supports deployment pipeline and to code your own infrastructure which helps in the foundation of DevOps. it integrates with many CI/CD tools, communication tools like Jenkins, Jira, slack etc.
T15	Gitlab	In DevOps, it supports planning and monitoring, and it makes deployment faster.
T16	Ansible	It delivers faster feedbacks, detect bugs for developers and deploys faster for it operations. it integrates with containers, cloud services and many DevOps tools like Jenkins, Travis CI, TeamCity.
T17	Chef	It is used for build, deploy and to maintain continuous automation
T18	Docker	Docker helps the developers by building and provides same environment and for operators it helps to deploy your applications faster. it automates the deployment pipeline and integrates with choice of your DevOps tools.
T19	Vagrant	It manages the virtual machines and reduces the dependencies for developers, and for operators provides consistent environment for testing infrastructure scripts. it provides a same environment for Devs and Ops to configure and test.
T20	Saltstack	It ensures packages are installed and running, deploys and it control and optimizes the infrastructure.
T21	Argon	It reduces the complexity of infrastructure through Domain Specific language. it generates scripts for configuration tools.

T22	JUNIT	It is a unit testing framework used in the test-driven development. it helps developers to test on java virtual machines.
T23	Cucumber	It allows the developer to test and run accepted tests in behavior driven development. it is written on ruby but also provides platforms other than ruby.
T24	Selenium	It allows you to automate the browsers and to test them. it supports various browsers such as opera, Firefox, safari and chrome.
T25	AWS ECS	It is used to run; scale containers and it supports orchestration services for Docker and eliminates the use of installing and operating of own container orchestration.
T26	Cruise Control	It continuously integrates the software and java-based framework. We can view the previous build and current build through web interface.
T27	Nagios	It monitors the application, sends notification alerts and sends second alert when the problem has solved.
T28	NewRelic	It monitors infrastructure to application health and provides agility for DevOps organizations.
T29	Graphite	It monitors the performance of computer systems. it collects and stores the data in real time. it integrates with Logstash, Sensu, and Nagios Core.
T30	Ganglia	It monitors the performance of computer systems such as clusters. Advanced features allow the users to view live CPU load averages.
T31	Rake	It is an automated build tool written in Ruby language. it manages dependencies for specific tasks as well as group tasks.
T32	Papertrial	It a centralized logging tool which tracks the customers' requests and configuration changes.
T33	Splunk	It makes development and testing simple by quickly identifying bugs. For Operators it monitors and troubleshoot logs.
T34	Sumo Logic	It provides a unified platform for both Devs and Ops. it monitors issues faster, reduces downtime and enhances customer experience.
T35	Logstash	It allows you to collect logs, and store them. You can view logs in ElasticSearch and can view in Kibana.
T36	Kibana	It is a data visualization plugin for ElasticSearch, it allows you to create bars, plots and pie charts.
T37	Slack	It allows to communicate with teammates, customers and clients. it has many features like files sharing, face to face communications, screen sharing etc. it integrates with many tools like Jenkins, GitHub, PagerDuty, Jira etc.
T38	Hip Chat	It is an instant messaging tool which allows to have group chats, image viewing and video calls
T39	Jira	It is project management tool mainly used by agile teams which allows you to plan your tasks, tracking work progress, releasing and reporting the performances. it integrates with many tools like Bamboo, Bitbucket.
T40	MangoDB	It is a document database tool and makes the development easy. it allows you to store data and provides indexing. it integrates with Slack, GitHub etc.
T41	LiquiBase	It provides tracking, managing, code branching and schema for database changes.
T42	RedGate	It automates the databases, monitors its performance and protects the data.
T43	DBMaestro	It automates the databases, reduces time in writing scripts and integrates with Jenkins, Bitbucket, etc.
T44	TeamCity	It is a CI server tool from JetBrains. it used for continuous integration and deployment
T45	Circle CI	It automates the pipeline form build to deploy. it runs automating testing and build in containers. it integrates with AWS, GitHub, Bitbucket.
T46	JMeter	It is used as a load testing tool for analyzing and measuring the performances of web applications.
T47	Cacti	It is a web-based log monitoring tool used for front end applications.
T48	OpenStack	It provides infrastructure as a service to DevOps people.

T49	Zabbix	It is a monitoring tool that tracks the status of network servers
T50	Capistrano	It allows you to run scripts on different servers and deploys web applications.
T51	PagerDuty	It is used for sending notifications, SMS, email for DevOps people
T52	TestComplete	It allows the developer to automate tests. They can be recorded, scripted and can be manually created. it is also used for error logging. it integrates with Bamboo, Jira, Git, Jenkins and Mercurial.
T53	Mercurial	It is written in python, used as version control for developers and is supported in Windows, Mac and Linux.
T54	CloudWatch	It is a monitoring tool for developers, operators and for it managers. it allows you to monitor the applications, infrastructure and optimizing the resources. For DevOps we can build, deploy, and automate test. it integrates with Jenkins, Slack, and many other tools.
T55	Azure Watch	It provides immediate alerts with data issues, it automatically recovers from production issues, and integrates with PagerDuty, Slack, Nagios, Zabbix, etc.
T56	AppDynamics	It releases and changes application quickly, improves collaboration between Devs and Ops, and supports continuous delivery.
T57	Amazon Web Services	AWS services help the companies to build, deploy the software rapidly and supports DevOps in automating tasks and to solve complex environments. They compile code, run tests, automates the deployment
T58	VMware vSphere	for DevOps people, it allows to manage container images, to organize image repositories, and gives access control to projects which allows particular users for access in repositories.
T59	VMware vCloud Director	It increases agility for DevOps people, provides containers as a service, and optimizes the cost of cloud infrastructure.
T60	Grafana	This allows us to visualize the data from different data sources. it integrates with Kubernetes, slack, MySQL
T61	Postman	It can be used for debug, monitoring, documentation, design and automated testing
T62	kubernetes	It allows you to deploy, configure and managing the containerization applications. it integrates with Docker, GitHub etc.
T63	Rancid	it runs various commands, increments data and commit these changes into version control.
T64	CFEngine	It can schedule tasks, configure, deploys application, integrates files and provides service and password management.
T65	Bitbucket	Bitbucket is the solution provided by the GitHub for professional teams.
T66	Graylog	It logs the applications, allows to visualize and explore data thoroughly, flexibility in passing User ID's and is a self-service tool which allows to deploy, and configure without it operations.
T67	Rundeck	It automates the development and production environments, schedules task, provides orchestration for workflow and logs the data.
T68	NUnit	It is a unit testing framework, it runs tests in Microsoft.Net language
T69	TestNG	It performs testing and checks whether the code is multithread safe.
T70	JBehave	It is a behavior driven development framework, and performs testing
T71	RALLY	It allows companies to keep track of their objectives. We can track daily works, manage deliveries and can recommendations performances. it allows to build, deploy, test and release.
T72	Redmine	it is written in ruby, can manage multiple projects, issue tracking, and sends notifications,
T73	Gocd	This tool allows you to compare builds, deploy anywhere, configure, and executes the tests in many programming languages.
T74	ISPW	It allows to build, deploy, and integrates with Jira, Splunk, and Jenkins
T75	Artifactory	It supports all packaging formats, automates deployment pipeline, and integrates with Jenkins, Bamboo, Travis CI, Kubernetes etc.
T76	Nexus	for developers it improves code quality, automates build and deployment. For

		operators reduce the cost of infrastructure.
T77	Delphix	It protects the data, and manages the data from anywhere on the cloud
T78	Flyway	It provides version control for database with plain SQL, migrates databases, and makes continuous delivery simple.
T79	VSTS	It can plan, track the work across teams, works with all languages in deployment pipeline integrates with Git and share packages across teams.
T80	FitNesse	It is a collaborative and test tool. it automates the acceptance tests, and provides feedbacks,
T81	Gating	It is a load testing tool which allows to analyze and measures the performances of web applications.
T82	Mocha	It is JavaScript framework, it runs tests, and provides accurate reports
T83	Karma	It provides easy debugging, controls the workflow and integrates with Jenkins, Travis CI.
T84	Jasmine	It is behavior driven testing tool written in JavaScript which allows to write tests easily.
T85	Tricentis Tosca	this tool integrates with any DevOps tool of your choice and helps to collaborate with other testers.
T86	Locust io	It allows to define user behaviors in python and supports in running load tests on multiple machines.
T87	Perfecto	It improves the efficiency of DevOps pipeline with continuous and automating testing, and integrates with TeamCity, Jenkins, Bamboo, IntelliJ, Slack, Jira etc.
T88	Saucelabs	It allows DevOps teams to develop, test, supports many browsers operating systems, and integrates with Jenkins, Bamboo, TeamCity, and VisualStudio.
T89	SoapUI	It is functional testing tool which provides test coverages, automates regression and load tests in a single environment.
T90	Micro Focus UFT	It accelerates the functional testing efforts for software application, it automates the tests, and integrates with Jenkins, Bamboo, Git.
T91	Packer	It builds the automated machine images for different platforms and runs on every operating system. it builds images for AWS, Docker, VMware, and Microsoft.
T92	Rudder	It manages configurations in middleware applications and works in many operating systems by continuously providing security.
T93	Terraform	Through terraform you can build, change configuration, execute plans and can share and automate infrastructure. it integrates with the monitoring services and enables special monitoring abilities.
T94	Octopus Deploy	It deploys the applications and makes release orchestrations simple.
T95	UrbanCode Deploy	It automates and deploys applications, provides rapid feedbacks, security in different environments, and integrates with Kubernetes, Jenkins, and Docker.
T96	Electric cloud	It provides self-service pipelines, it tracks and monitors the applications and provides security. it integrates with Logstash, and Kibana.
T97	ElasticBox	It brings Devs and Ops to manage and deploy applications on cloud platforms.
T98	CA Automatic	It helps in planning, releasing, builds, testing, and provides security management.
T99	XL Deploy	It speeds up the deployment while reducing errors, provides visual status of applications, and self-service deployment. it integrates with Jenkins, Puppet, Docker and Kubernetes.
T100	Google Cloud	Through this tool, you can protect your code, infrastructure, and can store your services. Google cloud provides configuration tools, developer choice of tools for CI, and testing tools. it supports DevOps to create and destroy environments
T101	OpenShift	It integrates with Jenkins and implements Continuous Integration, automates the build, deployment, provides container orchestration with Kubernetes
T102	Alibaba Cloud	It provides end to end monitoring, executes tests with automation, and deploys

		continuously. it deploys and manages microservices with Kubernetes.
T103	IBM Cloud	It is a full stack platform which provides services for containers, security, analytics, databases, and easily optimizes the environments.
T104	Cloud Foundry	It makes the application to build and deploy faster. Provides frameworks for developers and services for containers.
T105	Iron.io	It handles the container services, provides image and video processing, sends push notifications, and runs containers on shared infrastructure
T106	Apache Open Whisk	It provides serverless platforms, manages infrastructure, deploys and servers using Docker, deploys anywhere and integrates with many tools lke kubernetes, GitHub, slack, and Jira.
T107	Lambda	It lets to run your code without provisioning servers, runs code virtually on different applications, executes code in response to triggers and can be directly triggered by amazon web services.
T108	Zenoss	In DevOps, it improves agility by resolving potential issues, meet goals of DevOps process by deploying faster, and provides end to end services for infrastructure.
T109	Checkmarx SAST	It provides security for development and testing tools, provides early detection and remediation and reduces the analysis time.
T110	Snort	It performs real time traffic analysis, protocol analysis, content searching, and can be used to detect attacks
T111	Signal Sciences	It provides security instrumentation for developers which makes them to manage their data easily and brings Devs and Ops together by securing applications.
T112	Tripwire	It provides security for entire DevOps pipeline and protects data from unwanted changes.
T113	Black Duck	It provides security for DevOps people by managing the risks with open source tools and mitigates compliance risks.
T114	CyberArk Conjur	It is designed to protect account passwords which have an access for an entire organization.
T115	SonarQube	It inspects the whole application, manages the code qualities, and also provides analysis of pull requests. it integrates with Jenkins, Bamboo, Travis CI, TeamCity.
T116	Veracode	It provides security for entire software application in a single platform. it provides security for operation teams during deployment and integrates with Bugzilla, Jenkins.
T117	HashiCorp vault	Protects the data with centralized key management in dynamic infrastructure and integrates with AWS, MongoDB, MySQL.
T118	Fortify SCA	It protects the data from vulnerabilities, provides different practices to developers to code more securely. it detects the vulnerabilities, analyzes them and take actions.
T119	ServiceNow	It provides software as a service to build mobile applications, manages technology services tasks, it provides security, development, and supports real time communications.
T120	Trello	Trello helps to organize work, can view team's progress, it consists of boards which in turn consists of lists, and allows to collaborate with teams anywhere.
T121	Stride	It is an instant messaging app, allows to have a video calling, texting, image viewing, and runs on Windows, Mac, or Linux.
T122	Remedy	It is a service management tool which reduces the gap between Devs and Ops by making applications to deploy faster and automates the deployment pipeline.
T123	Agile central	It allows the agile teams to collaborate, plan their tasks and other business activities
T124	OpsGenie	It allows to deploy more frequently, it sends the notifications to detect errors more quickly, can manage all schedules in one platform, improves

		communication. it integrates with NewRelic, Jira, slack and AWS.
T125	CollabNet VersionOne	It integrates entire software cycle from development to deployment, provides common infrastructure for all the teams working in DevOps and integrates with Jira, Git, ServiceNow.
T126	AWS CodeDeploy	It automates the deployment process, reduces the downtime and eliminates the manual operations.
T127	Coverity	It is a static and dynamic analysis tool uses to fix defects in programming languages like Java, Python, C/C++.
T128	Perforce	It helps in continuous delivery and allows to plan, build, deploy and release. it integrates with Jira, Jenkins, and Microsoft.
T129	JSHint	It detects the errors in JavaScript and companies like Facebook, Mozilla, Medium uses this tool.
T130	CheckStyle	It helps developers to code and detect errors by automation.
T131	Blueoptima	It calculates actual coding effort, it performs automatic compilation and involves in planning of software development projects
T132	Clover	It runs several transactions and allows you to pay anywhere with barcode scanners print receipts and performs other mobile operations.
T133	Semmle	This provides knowledge base for source code, issue tickets, reduces development costs etc.
T134	JaCoCo	It provides access to JaCoCo runtime which executes coverage data and creates reports from the executed data.
T135	Gerrit	It provides framework for entire team to review code but gives access to limited number of users to make changes in the code.
T136	PMD	It finds programming flaws and support many programming languages like Java, JavaScript, XML, XSL
T137	SharePoint	It is a collaboration tool used for document management and storing system.
T138	Archiva	It is an artifact repository and provides other services like security access management, usage reporting and integrates with Maven, ANT.
T139	IBM Rational Clear Case	It provides access to software cycle activities and it performs control versioning and controls the development activities like workspaces.
T140	Hudson	It runs on servlet containers and supports many tools like Git, subversion, CVS.
T141	PingDom	It performs real time monitoring, tracks and analyzes websites load time, goes through the entire website and resolves issues
T142	CloudWave	It provides cloud security by maintaining privacy
T143	Glu	It is an automated deployment and monitoring tool. it deploys efficiently and detects bugs more quickly.
T144	Mesos	This tool provides containers for project tasks and are responsible for resource management. it integrates with Docker
T145	Rancher	It provides platform to deploy in multiple places and manages Kubernetes clusters.
T146	Docker Enterprise	It reduces infrastructure and maintenance costs and manages all the application needs such as security, cloud and solves it operational needs.
T147	GKE	It deploys containerized applications, performs health checks to find defects and crashed applications, manages many other applications and supports hardware accelerators.
T148	AKS	It manages by hosting kubernetes environment and deploys containerized applications and automates entire deployment pipeline.
T149	Rkt	It is an alternative to Docker developed for cloud applications and makes easy to integrate with other systems.
T150	Codefresh	It builds and deploys the applications in kubernetes environment and automates testing. it integrates with AWS, Docker, Mesos, Google Cloud and OpenShift.
T151	Helm	It manages kubernetes complex applications and provides charts to share on public and private servers.

T152	Fluentd	It is a unified data collector which runs to collect, analyze, transform and store data.
T153	Prometheus	It monitors multiple microservices. its architecture is modular and comes with many services called exporters which helps to capture metrics from the applications.
T154	ITRS	It monitors infrastructure, software applications and creates dashboards
T155	Moogsoft	It creates alerts by reducing signal to noise ratio and performs continuous delivery and root cause analysis.
T156	Dynatrace	It monitors the software applications.
T157	Datadog	It is a log monitoring tool integrates with many automated tools like slack, AWS, Docker, Hipchat.
T158	Elasticsearch	It performs monitoring, send notification and provides security.
T159	Intellij	In DevOps, Intellij helps to code in multiple languages, build and deploy the code. it makes to code your infrastructure and integrates with mercurial, Git, and SVN.
T160	Neo4j	It represents the data in graph databases. it integrates with Docker, MongoDB, ElasticSearch.
T161	Amazon S3	This service allows user to store and protect data. We can also organize the required data and configure business requirements.
T162	Amazon Cognito	It provides authentication services such as user sign ups, sign in, and access to many social platforms.
T163	Amazon Machine Images	Through AMI, we can create multiple instances with similar configuration. We can make them available for public or can keep it private.
T164	Git	It is used to track changes in the files and integrates developer's code.
T165	AWS Elastic Compute Cloud	It provides resizable computing capacity within minutes and allows the developers to develop and deploy easily
T166	OpsWorks	It integrates with Chef and Puppet and automates the configuration of servers, deployment and security.