OULU UNIVERSITY OF APPLIED SCIENCES

Roman Zakharenkov

DEVOPS IN E-COMMERCE SOFTWARE DEVELOPMENT: DEMAND FOR CONTAINERIZATION.

Roman Zakharenkov

DEVOPS IN E-COMMERCE SOFTWARE DEVELOPMENT: DEMAND FOR CONTAINERIZATION.

Roman Zakharenkov
Bachelor's Thesis
Spring 2019
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author: Roman Zakharenkov
Title of Bachelor´s thesis: DevOps in E-commerce software development: Demand for Containerization.
Supervisor: Kari Jyrkkä
Term and year of completion: Spring 2019 Number of pages: 67 pages + 4 appendices

This thesis aimed at researching trends in software delivery as well as the underlying changes made by companies in order to enhance their technological capabilities and be a stronger competitor on the market. This research was implemented and documented under the strict supervision of the Vaimo Group.

The research was done in a company that decided to apply a set of capabilities to a newly developed CI/CD pipeline based on the containerization technology. Description of the implemented and newly developed pipelines was provided to highlight capabilities of the new containerized software delivery pipeline.

As part of this research, in order to find an attitude of professionals in software delivery towards different technological capabilities, survey was carried out. Moreover, a survey was designed to benchmark current CI/CD pipeline using metrics such as stress, efficiency and visualization. Along with assessing implemented and potential solutions, survey questioned respondents on the DevOps culture, continuous integration and delivery practices.

The research revealed a set of capabilities and cultural patterns that developers see as most beneficial for the future of software delivery for an e-commerce solutions provider. These capabilities include automatic deployments, the usage of Git VCS, deployment to cloud environment using containers. In addition to that, professionals revealed positive attitude to taking operational responsibility and provided feedback on deficiencies of currently implemented CI/CD pipeline. The conclusions following from the survey were as well confirmed by a correlation between a given capability/solution and competence as measured via self-evaluation.

Having statistically valid data and conclusive decision in favor of or against a given technological capability allows e-commerce providers to build a software delivery pipeline that adresses the needs of the organization and its clients.

Keywords: DevOps, Docker, Kubernetes, Magento, E-commerce, Continuous Integration, Continuous Delivery

CONTENTS

VOCABULARY

WORD	DEFINITION
CD	Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time (1).
CI	Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly (2).
DevOps	DevOps is a methodology that is focused on encouraging greater collaboration between everyone involved in software delivery in order to release valuable software faster and more reliably (3).
GDPR	Regulation (EU) 2016/679[1], the European Union's ('EU') new General Data Protection Regulation ('GDPR'), regulates the processing by an individual, a company or an organisation of personal data relating to individuals in the EU (4).
QA	The maintenance of a desired level of quality in a service or product, especially by means of attention to every stage of the process of delivery or production (5).

1 INTRODUCTION

In recent years DevOps has entered the stage of software development as a culture that promotes fast, frequent and reliable software delivery. The culture has appeared as processes of continuous integration and delivery have occupied a major role in software development procedures along with agile methodologies for managing software development.

CI/CD has become part of any working process along with the Agile management and DevOps culture. The development that strongly suggests that companies competing on the market are looking at how improved processes and culture, change in development paradigm could give an advantage in competition with other businesses. The energy that used to be concentrated on the development of software functionality and its optimization now flows into enhancing software delivery procedures. The described change can give businesses an “edge”, in the world where even short downtime means big financial loss and new features are expected on increasingly tighter schedules.

One of the biggest participants of DevOps movement is e-commerce sector. It is expected that the share of global e-commerce retail sales will raise to stunning 17.5% of all retail sales in 2021, which is an approximate of 30% increase from the 2019 level (6).

Vaimo is one of the world’s leading e-commerce solution providers which has undertaken a process of changing CI/CD procedures through the introduction of DevOps culture and development of a new software delivery platform based on containers with their subsequent delivery to the public cloud.

This research aims at discovering software delivery processes as well as strategic business, procedural and technological changes that the company, aiming at being a strong competitor on the market, brings forward. Two methods of finding suitable software delivery solutions applicable to e-

commerce software that are applied in the thesis: researching literature on software delivery and surveying a large number of professionals in the field of e-commerce. In the following chapters of the thesis, a brief introduction to CI/CD pipelines used in Vaimo will set a common ground for communication. Survey data will be used so as to predict capabilities that would likely bring an outmost value for the software delivery pipeline being developed.

The author of this Bachelor's Thesis is part of the team involved in the development of the CI/CD pipeline for Vaimo Finland Oy. The author is responsible for templating project repository structure (so that it complies with the CI/CD pipeline architecture), testing the CI/CD pipeline, educating of employees, communicating between affected teams and documentation.

2 SOFTWARE DELIVERY

2.1 Perspectives of looking at software delivery

Software delivery is a multifaceted process. There have appeared different names to highlight different perspectives to software delivery.

From the managerial perspective, software delivery is fast-paced with ever-changing customer requests and high cost of architecting a complete solution. Therefore, agility is the key to producing a product that would meet customer's specifications. In 2001 leading experts in the software industry created the Agile Manifesto which states principles through which agility in software development can be achieved. Since then the popularity of agile frameworks for managing working processes has grown and they have become a de-facto standard driven mostly by managerial drive to accelerate time-to-market (with 76% of developers reporting the use of an Agile framework in 2013) (7).

On the other hand, bringing software to the customer is a technological process. It requires a delivery pipeline for bringing a solution from developer's workstation to the client in iterative cycles. This process is based on what is normally referred to as the CI/CD pipeline. It is a set of tools and workflows, designed to automate software bundling, which provide automated quality assurance as well as deployment to the production environment.

Bringing these endeavors further, in later years (as trends discussed in the chapter 2.2 show) the DevOps movement has gained momentum, bringing the cultural change in software delivery to the forefront of discussion in software circles. The transformation that can be generally described as a change in paradigm within which development and operations work is carried out. Traditionally software development work was separated from operations, which created a gap between requirements and the latest data existing in these departments, that in turn led to the situation, where CI/CD solutions developed

did not fit the needs of the product. Moreover, frequency of releases decreased due to low information bandwidth between the departments. As can be seen from the figure 1, which describes the cost of delay in release of software features in one of the Maersk software projects, some features are vital and even a small delay in their delivery means substantial losses for the software operator (8). Within the DevOps paradigm, development is tightly connected with operations and the quality assurance, which brings agility to development that could be characterized by a shared responsibility and constant information exchange between product developers and operations professionals.

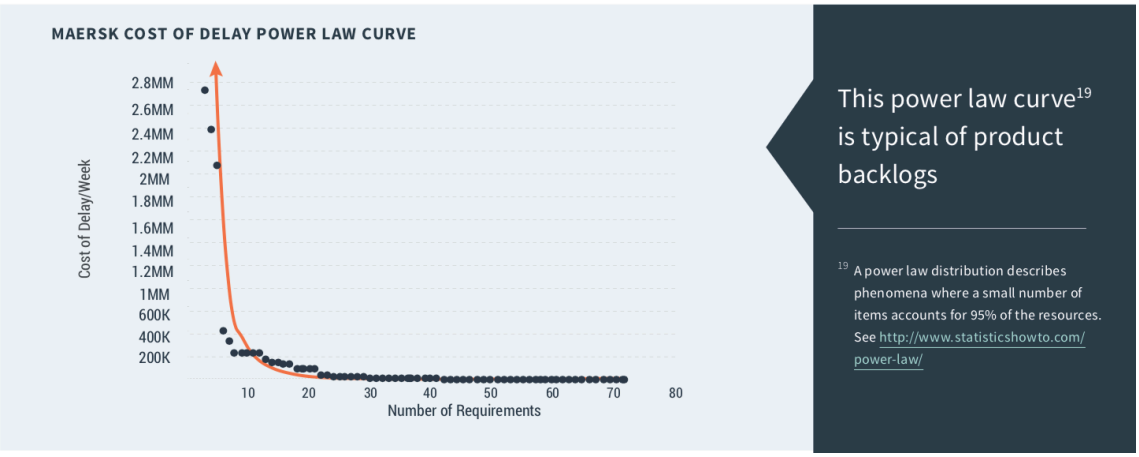


FIGURE 1. Cost of delay for software requirements.

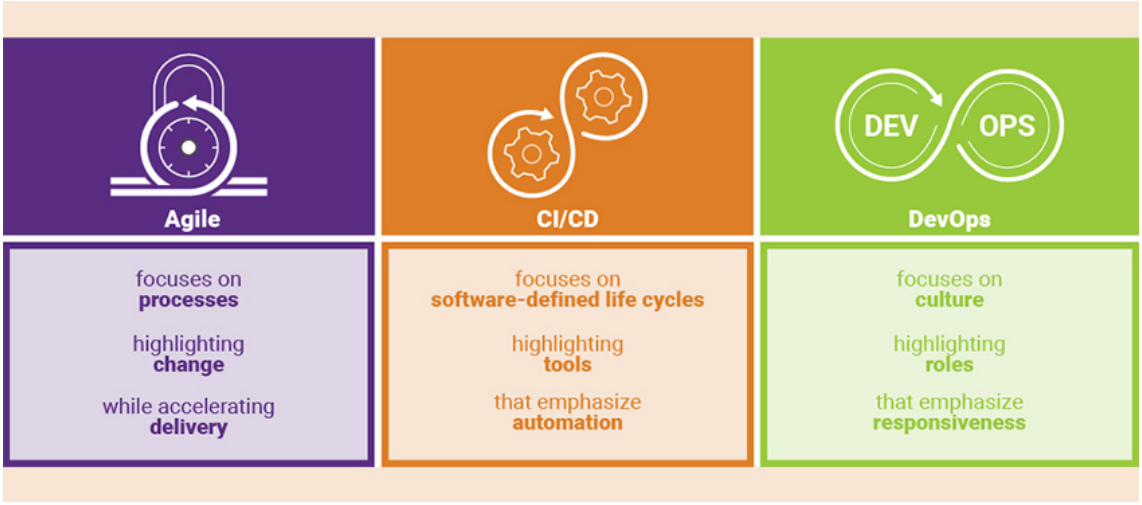


FIGURE 2. Perspectives of looking at software delivery (9).

Each of the perspectives towards software delivery described above highlights important aspects of work a business should undertake to improve the software

delivery performance for the goal of creating a larger value for the customer at a lesser cost.

This explains the cause of such a close correlation in online search queries as calculated by Google Trends between terms “Agile”, “Continuous Integration”, “Continuous Delivery” and “DevOps" (see ch. 2.2).

2.2 Software delivery as a trending area among software creators

The software delivery culture, in more specific terms named DevOps, has been rapidly rising in its adoptability by enterprises that produce software solutions. Since word combinations “Continuous Integration” and “Continuous Delivery” can act as proxies for “DevOps”, the interest in effective software delivery practices has significantly intensified since the year 2010 as can be seen through data provided by Google Trends service (see figures 3-5). The trend is likely to be fueled by multiple factors which have played a role in software development during the past decade: a constant increase in bandwidth of information transferred, a maturity of scalable software delivery solutions along with adoption of the cloud infrastructure. Both of them have allowed the possibility to deliver software within minutes or hours. They have also allowed a significant degree of assurance that software delivered works according to specifications.

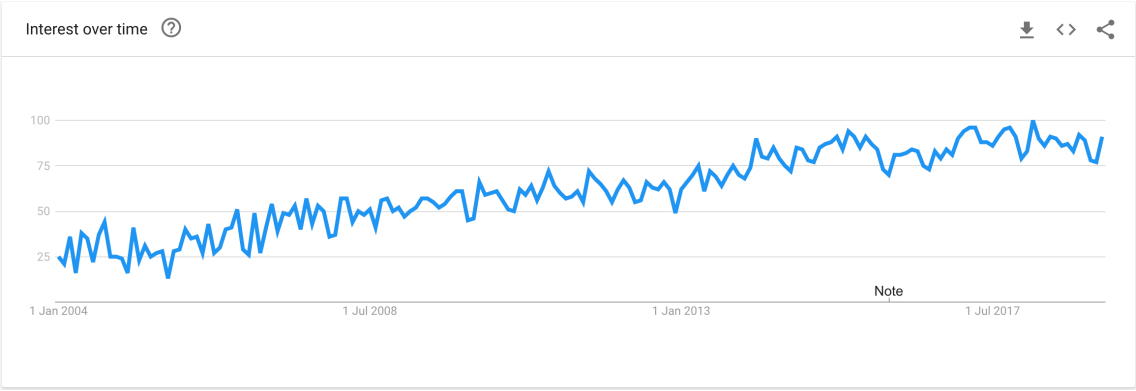


FIGURE 3. Google Trends. Trend in search for the word combination “Continuous Integration”.

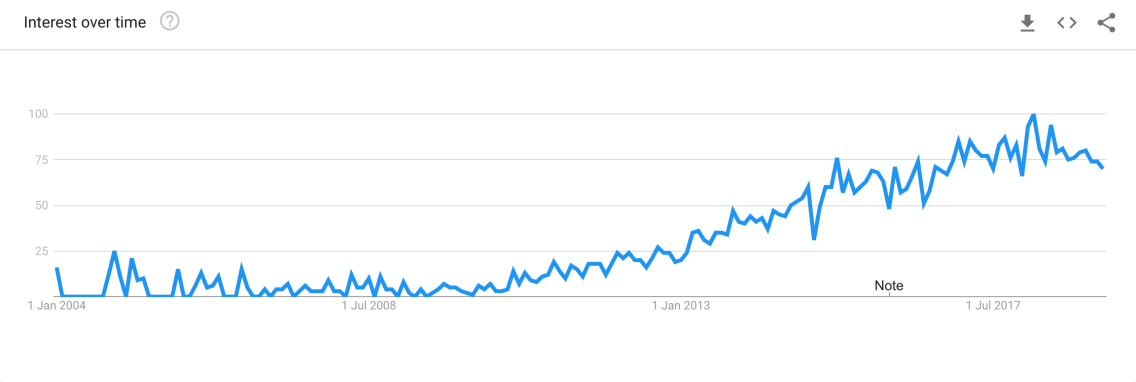


FIGURE 4. Google Trends. Trend in search for the word combination "Continuous Delivery".

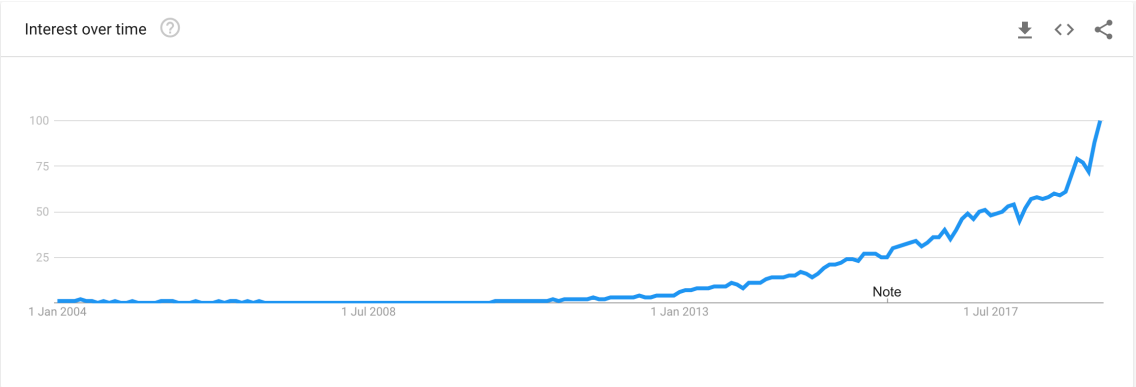


FIGURE 5. Google Trends. Trend in search for the word "DevOps".

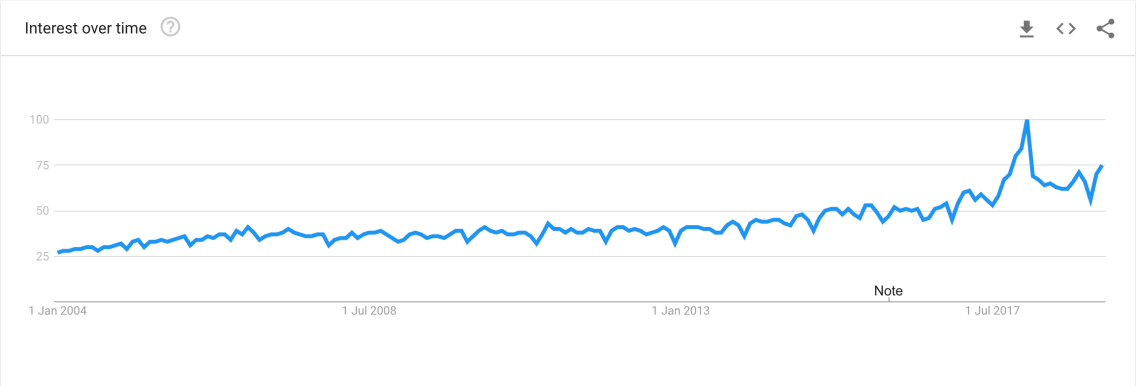


FIGURE 6. Google Trends. Trend in search for the word "Agile".

The search queries analyzed with the Google Trends service propose a strong correlation between "Continuous Integration", "Continuous Delivery" and "DevOps" activities (figures 7-8), which leads to a conclusion that these methodologies have been growing in popularity simultaneously, supporting

each other.

Agile methodologies for managing software development are also significantly correlated with continuous integration as can be perceived from the figure 9. However, the correlation of 0.752 is considerably lower than the correlation between Continuous Integration and Delivery practices (0.922) or between Continuous Integration practices and the DevOps (0.951) culture. This reflects the fact that implementation of Agile methodologies is a good proxy for a high-performance organization on a managerial level, which may or may not lead to the adoption of a strong DevOps culture with processes it entails.

Attributes	Continuous integration: (Worldwide)	Continuous delivery: (Worldwide)
Continuous integration: (Worldwide)	1	0.922
Continuous delivery: (Worldwide)	0.922	1

FIGURE 7. Google Trends. The correlation between search results for the word combinations "Continuous Integration" and "Continuous Delivery".

Attributes	Continuous integration: (Worldwide)	DevOps: (Worldwide)
Continuous integration: (Worldwide)	1	0.951
DevOps: (Worldwide)	0.951	1

FIGURE 8. Google Trends. The correlation between search results for the word combination "Continuous Integration" and word "DevOps".

Attributes	Continuous integration: (Worldwide)	Agile software development: (Worldwide)
Continuous integration: (Worldwide)	1	0.752
Agile software development: (Worldwide)	0.752	1

FIGURE 9. Google Trends. The correlation between search results for the word combination "Continuous Integration" and word "Agile".

Enterprises and software solution providers faced a challenge of adhering to the new standards of software delivery or facing an inevitable decline due to the competition with companies that adhere.

2.3 Demand for containerization

One of the technologies to drastically impact on processes of continuous integration and delivery has been containerization, the use of which allows for the following capabilities:

- Replicating a production environment on the developer's machine, so as to decrease the delta between the two, ensuring that the product being further developed runs on the production environment as expected from being tested on local machine.
- Reproduction of the application in a given state irrespective of the host operating system, through the deployment of container to the target environment. (10)

Both of the capabilities are reflected in the definition of container as being "a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another." (11)

The following capabilities are provided by two types of technology: containerization and virtualization. They both are often referred to as "virtualization" since technologies are often addressing the same issue, with VMs appearing on the market significantly earlier than containers. Whereas the former allows to define a set of processes and their dependencies to run on top of a shared virtual machine, the later allows to create virtual machines with software and software dependencies installed on them (see figure 10).

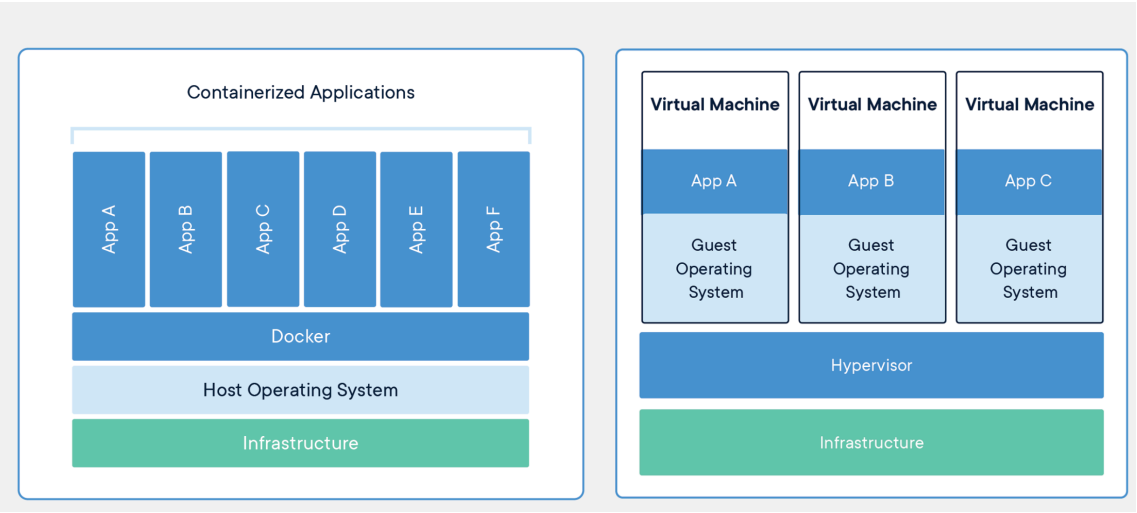


FIGURE 10. Docker Inc. Comparing Containers and Virtual Machines (10).

Two of the capabilities described above are provided differently by virtualization and containerization software. In e-commerce web store development, of which Vaimo Group has had a significant experience in, the impact of introducing containers in place of virtual machines is especially noticeable in two aspects: local development and operations. The detailed comparison of these technologies is out of scope of this research.

Vaimo Group has selected the Docker containerization technology and Kubernetes container-orchestration software for its environment. The former is designed to run developed software in a pre-defined environment, which is containerized and separated from the operating system, upon which the Docker is run, the latter was later on created to manage instances of the containers being run by Docker. There has been no other mature general-purpose containerization or container-orchestration technology to select from in the time this research was written, therefore Vaimo Group, having the software availability/reliability as one of the priorities, had to rely on these technologies.

2.3.1 Local development

It has been a practice for the software industry to rely on the development on replicas of production environment to develop software upon. Traditionally it has been done through installing a Vagrant tool on a development machine, which

would run a virtual machine with the same environment as used on the production server.

Setting up a development environment in the following manner is suitable for cases where the developer has to work on a single project for prolonged periods of time. This entails the use of a single virtual machine with the defined configuration (meaning it does not have to be spun off regularly) and a relatively small size of project contents, making synchronizing operation rather fast (one project involved).

However, in many software houses development tasks assigned to a single employee belong to different projects. This implies that the developer has to have multiple, often large-scale projects, on a local machine that entails either of two cases.

The first case is where the environments, which projects require differ. This means that the developer has to spin down and up virtual machines when switching between tasks. It makes development process longer due to the fact that spinning up and down a virtual machine is a heavy computational task. In addition to that, configuring multiple virtual machines to be ran on Vagrant is a much more demanding task when compared to configuring multiple containers, due to the fact that containers are designed to be lightweight and created fast.

The second case is where environments required by projects are the same. This allows to setup one virtual machine with Vagrant and VirtualBox to embed a large number of projects, which would entail the fact that the tool designed to synchronize the files on the local machine with files on the virtual machine, would need to keep track on n (n – number of projects) times more data, which makes synchronizing slower. Moreover, NFS is a de-facto standard tool for synchronizing files between a host and a virtual machine, which becomes a bottleneck for development in large projects or cases with multiple projects on a single virtual machine. This is due to the fact that NFS is significantly (3x-5x times) slower than, for example, unidirectional file-synchronizer rsync (12) or a similar, but bi-directional implementation Unison (13) as was demonstrated by

Jeff Geerling using Drupal content management system.

There is a small subset of developers that use project-based bi-directional synchronization while working with VMs (usually using Unison). Although developers working on Magento 2 framework in Vaimo report increased speed of local development when using Unison, most developers tend to avoid its use and rely on a default solution based on NFS, which does not work on a per-project basis. NFS remains a file-synchronization tool of choice due to the fact that setting up and using per-project file-synchronization (e.g. using Unison) remains less reliable and more complex than NFS.

One of the benefits of using Docker containers for local development is speed achieved by addressing the issues described above. Firstly, Docker containers are designed to embed only one project, which decreases the number of files to be watched and synchronized to files contained in only one project. Secondly, Docker is using the Unison file-synchronizer by default, which is faster for development work on at least a subset of solutions (12, 13), including the Magento 2 e-commerce platform. Thirdly, Docker containers encourage the usage of only a subset of programs and processes that an operational system has, making it slimmer and hence faster.

2.3.2 Operations

After having software (in this case an e-commerce web-store) developed in a container on the developer's workstation, the next step is to deliver the update to the customer. As described in the chapter 2.3.3, it is cost-efficient for the company to develop software using containers, since it speeds up and simplifies the development as observed and reported by professionals (12, 13) and within Vaimo Group. Hence, it reduces development costs. In addition to that, the value of the product delivered increases due to the added reliability, scalability and ease of maintenance which containerization provides.

One of the core features behind Docker containers is that they run ignorant of the environment where Docker is installed in and, therefore, they are designed to run on any environment that Docker supports. Even though it is possible with a larger overhead to achieve the same result using a virtual machine, it does not come out of the box and requires more complicated configuration, setup and maintenance efforts later in the exploitation. (14) Once containerized software is deployed, with Kubernetes, it is possible to define where containers are run (on which “cluster”) and on how many instances (called “pods”). It is also possible to define memory usage, enable auto-scaling, connect the container via a network to other containers and more.

Kubernetes is designed to work in the cloud. Together with Docker it produces the environment not only where an application is ignorant of operating system it is run on, but also of the underlying hardware, since it becomes abstracted through the cloud.

Docker and Kubernetes, therefore, allow to abstract the application layer from the underlying environment of the operating system, as well as from hardware. This provides a possibility of running an application on most hosting solutions with precisely-defined configuration for software and hardware that most public clouds are able to readily fulfill.

The containerization technology also allows for isolation of applications. Vaimo Group currently relies on the virtualization technology and the CI/CD pipeline, which are designed to deploy several applications (unrelated ecommerce web-stores) on the same server to achieve efficient resource utilization. These web-stores share the operating system and its environment along with resources. Therefore, having one web-store failing can cause other web-stores to be unavailable. The software delivery pipeline based on containers allows to isolate web-stores, whilst achieving efficient resource utilization via resource pooling from a public/private cloud. Moreover, not only does the containerization technology provide isolation from other web-stores, it also allows for isolation between services (e.g. MySQL, Redis) that web-store relies upon. (15)

Concluding the benefits of the containerization technology in operations, it is important to point to the ability of containers to run being abstracted from the software environment and hardware, with operations employees having an opportunity to define resources and configuration for the container. This ability allows maintenance efforts to concentrate on keeping the application available and it having enough resources to sustain itself at all times. Along with that, abstracting from the software environment and hardware entails the ease of reproduction of the application in any number of targeted environments, used for e.g. quality assurance and production.

2.4 DevOps as change in culture

One of the most significant changes to be introduced with a new software delivery pipeline is an improvement in DevOps culture with a shared responsibility lying in the core of it. This change has to be brought along with the introduction of the container-based pipeline in order to have the most efficient usage of resources and a shared responsibility for the project.

Currently, in Vaimo Group the teams responsible for development and operations have the responsibility only for their part of the job. The communication bandwidth between two departments is limited due to differences in the physical location and limited size of the hosting/operations team. In the DevOps environment, companies value notion of a shared responsibility.

One of the ways to achieve that is to push part of operational responsibility to developers, having a hosting/operations team organizing the cloud environment and providing capabilities for monitoring, visualization and automation of CI/CD processes. These changes would allow for a closer connection of development, operations and quality assurance teams. Developers would have more control of the environments their software is operated in. Operations would be able to use the freed time for creating solutions that would suite the requirements and workflows of development and quality assurance procedures implemented in

the company so that the CI/CD pipeline and solutions coming with it correspond to the structure of processes in the company.

From QA professionals' perspective, the DevOps culture allows to enhance communication with developers so that the QA solutions implemented would be effective when finding inefficient or failing code or feature passed for quality assurance tests. It would also allow for a closer cooperation with operations professionals responsible for including QA automation in the software delivery pipeline.

In order to succeed in pursuits, such as the creation of a QA automation suite, promotion of operational responsibility or complying with GDPR requirements (see figure 11), it is necessary to share the responsibility from the point A (a development team receives the task to develop a feature or a fix for the existing problem) to the point B (a feature or a fix is successfully deployed) through the DevOps culture, see figure 12.

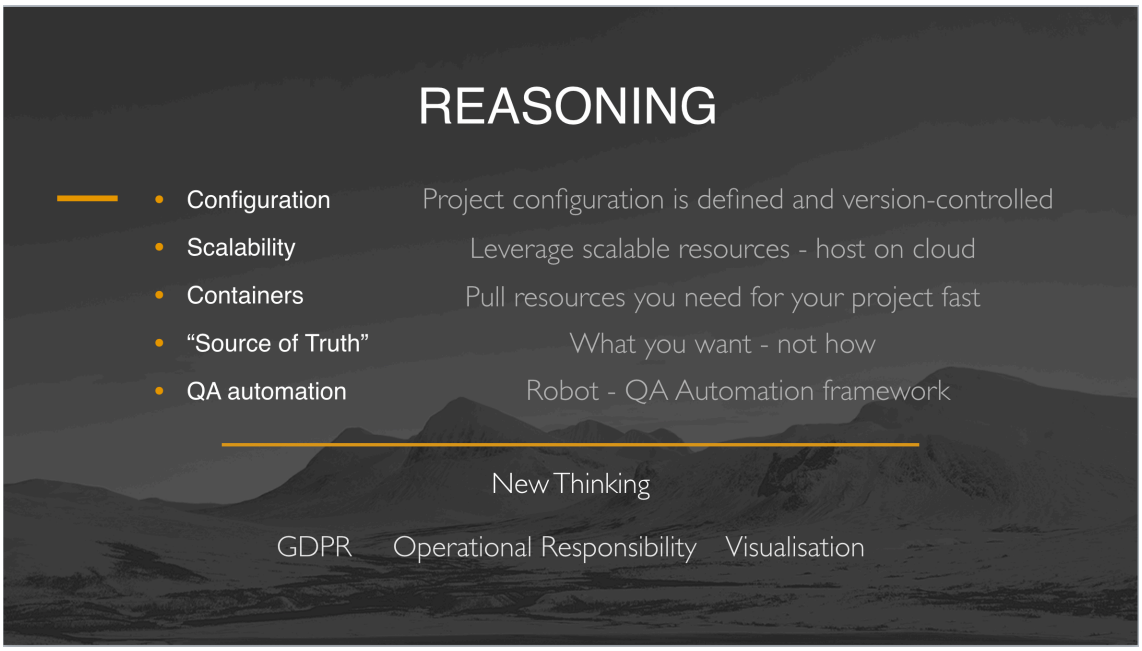


FIGURE 11. Vaimo Finland. Reasoning for usage of a container-based software delivery solution.

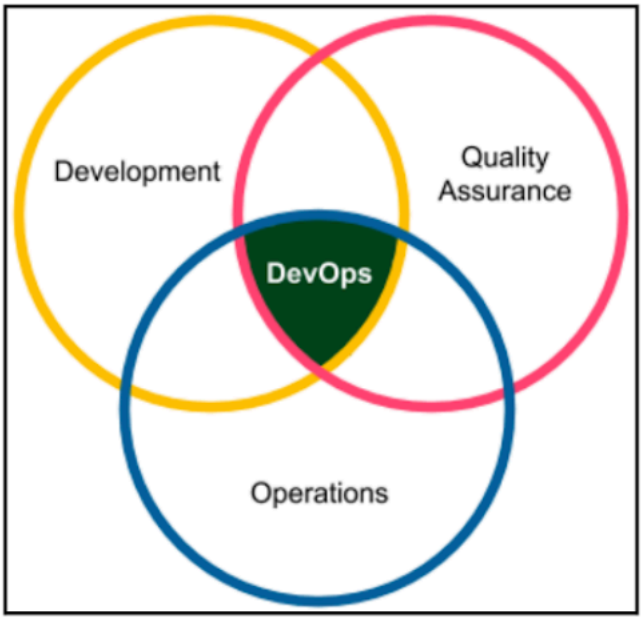


FIGURE 12. DevOps as culture uniting different aspects of software production. (15).

3 SOFTWARE DELIVERY METHODOLOGIES

This chapter is designed to describe and the contrast overall architecture of the implemented and proposed CI/CD pipeline. The description of CI/CD pipelines is provided to common ground for initiating a research into the DevOps and software delivery (see ch. 4), the goal of which is to select the most feasible capabilities/features (see ch. 4.3.2) for the CI/CD pipeline being developed.

3.1 Conservative approach to software delivery

Currently implemented, de-facto standard software delivery pipeline, is based on conservative software solutions (see figure 13).

The local development environment is based on a MAC host, upon which a virtual machine is installed using VirtualBox and Vagrant virtualization software along with their dependencies. Vagrant creates the environment for a guest OS to run using VirtualBox, which is later on reproduced in the production and staging phases. File synchronization between a guest and a host OS is implemented using the NFS technology. Aja software, which is a product of Vaimo Group, acts as an interface to the CI/CD infrastructure, via which most of the build, deploy and monitoring operations are carried out.

The CI/CD infrastructure is composed of a VCS that stores project code. It is upon build used by Jenkins continuous integration software in order to build artifacts for deployment to a targeted environment (dev/production/staging). The deploy request from Aja to Vaimo API is directed towards the traditional hosting environment (dedicated servers), where server software loads a pre-built artifact for the target environment.

It is important to mention that Aja provides import, monitoring and reference tools, with which it is possible to get logs, import database, media, receive information on modules as well as deployment and build statuses. Aja is not in

scope of the research and is mentioned to provide a brief description of its role in connecting the local environment to the CI/CD infrastructure.

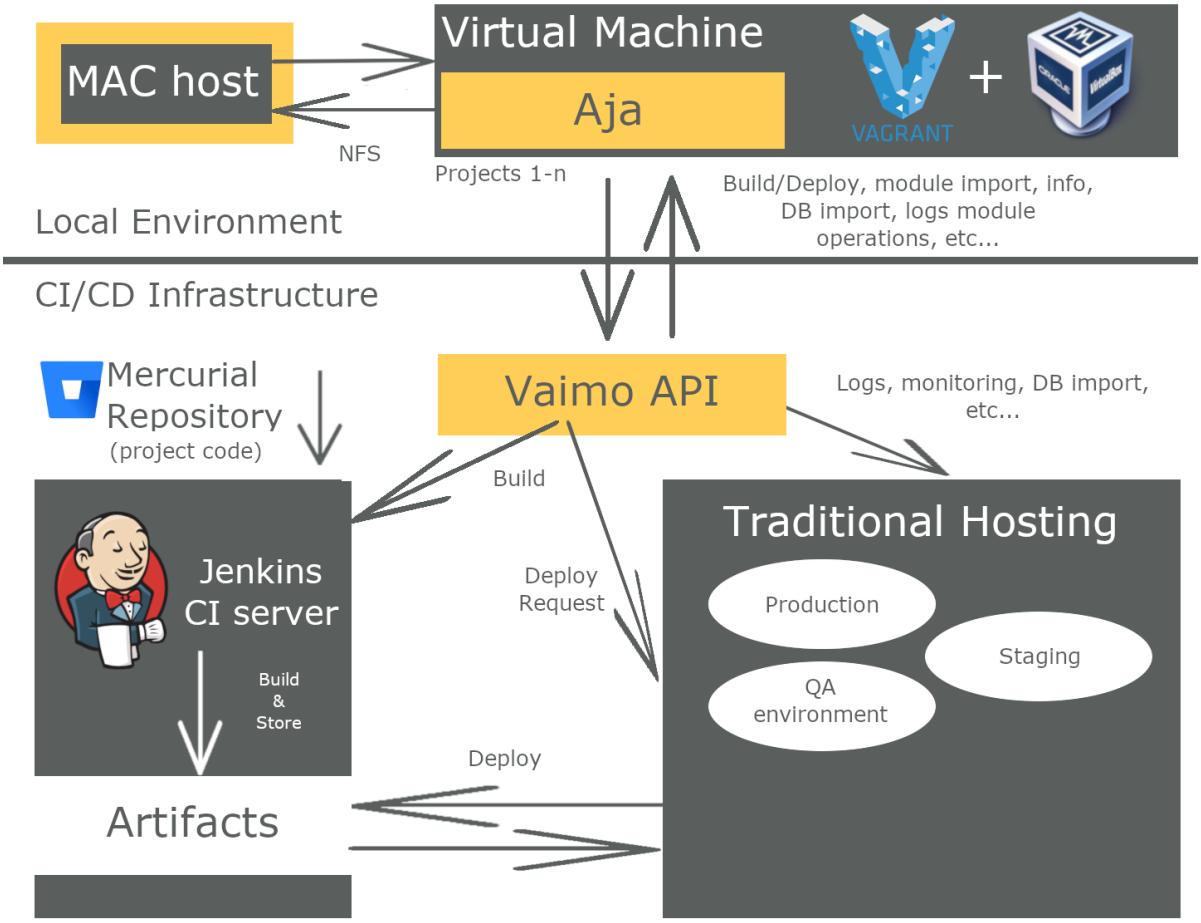


FIGURE 13. Vaimo Group. The implemented software delivery pipeline.

3.2 Continuous integration and delivery using containerization

The local development environment is based on a MAC host, upon which Docker containerization software is installed along with its dependencies. Docker runs the container, which is later on rebuilt and uploaded to production, staging and one-off environments, which are span for short amount of time for, e.g. testing, and then shut down. The synchronization of files between a container and a hosting OS is implemented using the Unison technology. Argo CLI and Kubernetes CLI act as an interface to the CI/CD infrastructure, via which most of the build, deploy and monitoring operations are carried out.

It is to be noticed that the containerized CI/CD pipeline entails the usage of separate containers for separate projects, providing a stronger encapsulation between project environments and a faster file synchronization.

The CI/CD infrastructure is based on Google Cloud facilities which are linked to the outer VCS repository. VCS repository is based on Git stores environment configurations and project code. When build operation is triggered, project code is used by Argo continuous integration software for building containers ready to be deployed to a targeted environment (e.g. dev/production/staging). These containers are then stored to Google Cloud's Container Registry.

Deploy is triggered via the ArgoCD web interface for a given build (that is version tagged software) stored in Container Registry. The cluster state repository is part of VCS and is used by ArgoCD to provide Kubernetes container orchestration software a set of environment configuration rules and variables. Kubernetes in turn makes sure that containers that are deployed follow these rules. Kubernetes is part of the Kubernetes Engine in the Google Cloud ecosystem. All containers that are run in the cloud are under direct supervision of Kubernetes Engine, which is under control of the developer via locally installed Kuberentes CLI and Google Cloud web and CLI interfaces as well as the cluster state repository.

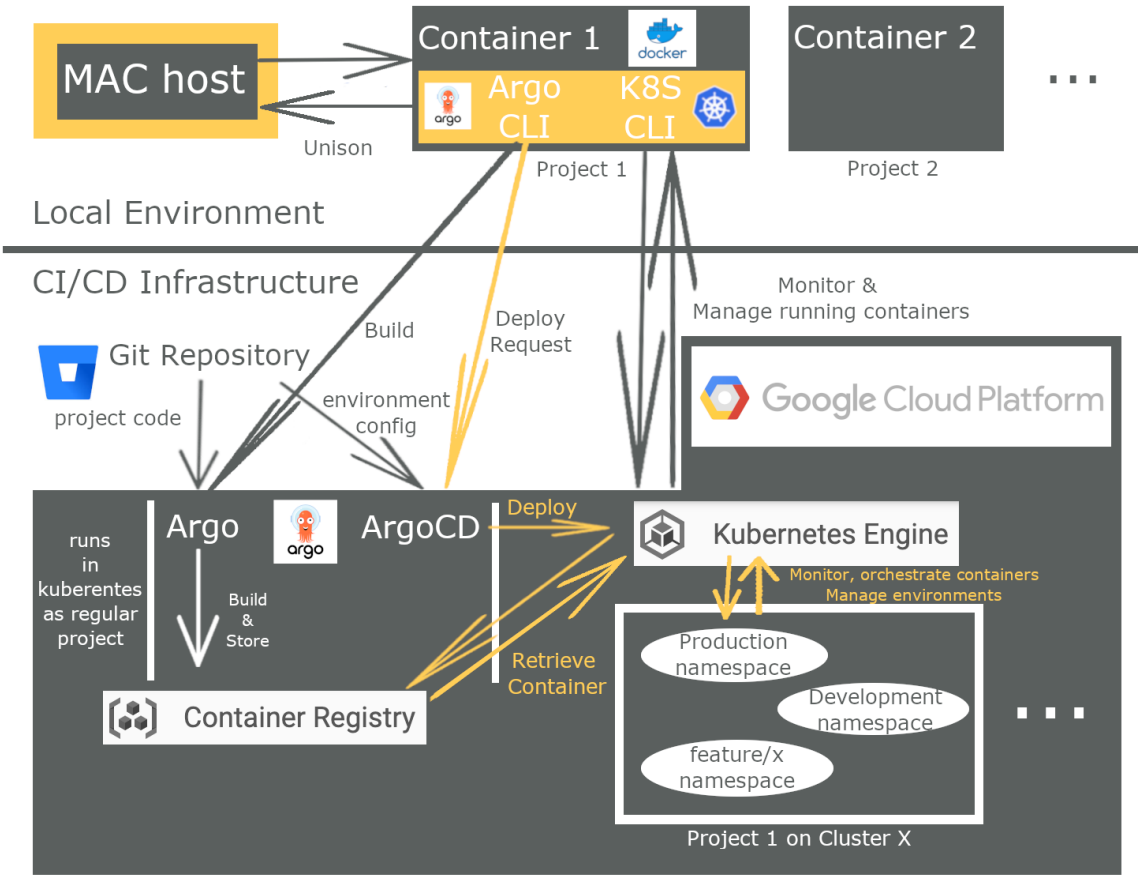


FIGURE 14. Vaimo Group. The proposed software delivery pipeline.

3.3 Capabilities driving change in software delivery methodologies

Whilst both of the software delivery pipelines described above serve the same purpose – the delivery of software to the client, there is a set of capabilities that characterize containerized software delivery pipeline that drive its development:

- Compliance to GDPR (use of test data instead of the production database import to keep sensitive data at one location and rely on sample data for development)
- Operational responsibility is shared with developers (DevOps)
- Configurations for environments are stored as files in VCS (the GitOps methodology)
- Cloud capabilities such as resource polling and rapid elasticity. (8)

- QA automation via the Robot/Codeception framework and utilization of one-off QA environments that are span off to test a particular feature and then shut the environment down.
- Efficiency characterized by a savvy resource consumption due to the utilization of resource-effective containers and cloud technology.

Capabilities described above affect both technological and cultural aspects of work. Their implementation is considered by the management of the company as a valuable investment that allows to be a highly competitive e-commerce software solutions provider from mid-term to long-term future.

4 ACCELERATING BUSINESS PERFORMANCE THROUGH DEVOPS

4.1 Management and Resources

The reports, which are represented in figure 15 and referenced at the end of research (8, 16), have provided ground for action after business and technical management of Vaimo Finland has acquainted themselves with the data and statistics presented in them.

FIGURE 15. Vaimo Finland. The research in DevOps used for software delivery pipeline development.

During the summer 2018 reasons (see chapter 2.3-2.4) for piloting the new CI/CD pipeline and changes in the DevOps culture were communicated to Vaimo Group. The management of the group has readily approved of the venture and agreed to provide resources for the continuous development of the system if it proves itself viable after being tested and applied to a customer's project.

Vaimo Finland has dedicated one technical manager to develop the Docker/Kubernetes pipeline. In December 2019 the pipeline was ready to be applied to a customer project, where Vaimo Finland provided a team of developers to test the new CI/CD pipeline for inefficiencies and vulnerabilities. The author was part of the assigned team responsible for communication, testing, documentation and research.

The reasoning for creation of the new pipeline and intermediate results were presented to the Vaimo Group management and employees in Tech Monthly in February 2019, where timeline for the project implementation was presented. During Tech Monthly, there has been made a preliminary agreement with units of other countries to create a joint effort in the implementation of the new pipeline.

4.2 Research

This research was carried out in pursuit of understanding and communicating the means of implementing a robust CI/CD pipeline and DevOps culture. Research and development work were carried out in parallel in a manner that allows to manifest advantages of agile processes.

The research was carried out in several stages described below, see the chapter 4.2.1.

The most value the company has placed on the following parts of the research: surveying employees' perception of deficiencies of the implemented software delivery pipeline (see APPENDIX 2) as well as proposals for improvement that address issues raised in other stages of the research. The surveying methodology was borrowed from references 8, 16.

This chapter after delineating the overall structure of research will present findings that revealed themselves after analyzing survey data. The survey was carried out in Vaimo Group between 4th and 14th of March 2019 using Google

Forms. 101 responses were gathered within the Vaimo Group. At the moment of writing this thesis, Vaimo Group employed approximately 450 professionals, being mostly developers, QA professionals, technical and business managers.

The spreadsheet containing responses to the survey was downloaded for the further data analysis, which was done using Rapidminer software. A manual classification and an analysis were carried out for survey questions demanding the textual feedback (questions 12, 16, 18, 32, 33, see APPENDIX 2).

4.2.1 Research stages

The research stages included the following:

1. Agreement on the research topic and structure reached.
2. Describing implemented and proposed CI/CD pipelines, as well as the advantage in capabilities of the latter.
3. Collecting data on the perceived value of implemented CI/CD solution based on the Aja/Jenkins pipeline as well as the existing DevOps culture. Measuring the perception of value/feasibility of capabilities that could be potentially implemented in the new software delivery pipeline.
    1) Collecting feedback on the research structure.
    2) Sharing the survey with the Vaimo community.
    3) Questionnaire is closed
    4) Data is analyzed.
    5) The final survey report is published and presented to the Vaimo community.

4.3 Survey Constructs

The following constructs will be used directly and indirectly in the survey analysis.

4.3.1 Competence

One of the most important constructs of the research is knowledge of employees, in other words competence. The most emphasis is given on questions 4-7 of the survey (see Appendix 2), which are designed to provide enough data that would allow to clusterize respondents into groups of “more experienced developers” or “less experienced developers”, hence grouping them based on their competence. The following questions were designed to allow respondents to self-evaluate their competence in software delivery:

- How knowledgeable are you in current day to day CI/CD operations using Aja and Jenkins?
- How knowledgeable are you in day to day CI/CD operations with a solution being deployed to a cloud using Docker/Kubernetes?
- How well do you know inner workings of the CI/CD pipeline currently implemented using Aja and Jenkins?
- How well do you know the inner workings of the container-based CI/CD pipeline based on Docker and Kubernetes?

The construct is important for decision-making, since architectural decisions made in CI/CD will have an impact on countless customer projects and it is important to find the opinion of experienced developers when consulting on these matters.

In the analysis below, words “competence”, “knowledge”, “experience” and their derivatives are referring to this construct.

4.3.2 Feasibility

The following construct is used for making a decision on whether a potential solution is feasible:

- Knowledge

- Capabilities of the solution
- Proposition among potential adaptors
- Opposition among potential adaptors

The construct above would describe the knowledge of the team in a given area of expertise, capabilities of a given solution as well as measures that would estimate the number of employees strongly supporting or opposing this solution. Having the data for the construct would give a good estimate on whether a solution has the capabilities looked for and whether team will be knowledgeable and driven enough to succeed in its implementation (17).

It is important to note that “knowledge” refers to the value of the solution as perceived by respondents, in other words, the ability of team members to utilize capabilities of the solution.

The feasibility construct should be seen as a model with each of its constituents being a proxy for feasibility. Due to the lack of data it is often the case not to have data available for all four constituents. Therefore, in subsequent analysis proxies for feasibility construct would be used to provide the evaluation for the feasibility of a given solution.

4.4 Survey Analysis

The data analysis for the survey aims at finding statistically relevant data, which would describe professionals’ perception of a correct pathway forward in the implementation of a container-based CI/CD pipeline as well as employees’ preferences and knowledge. The survey was also designed to reveal information about inadequacies of the implemented software delivery pipeline and DevOps culture.

4.4.1 Survey Data

One of the helpful techniques that will be employed in the data analysis is clusterization. Survey data contains four questions (see APPENDIX 2,

questions 4-7) which are designed in order to allow respondents to provide self-evaluation of knowledge in software delivery procedures and architecture. Due to the fact that there is a reasonably high number of respondents, it is possible to clusterize respondents into two sets: more experienced developers and less experienced developers. This technique would allow to correlate professional competence in software delivery with a preference for a given software delivery solution.

RapidMiner software was used as a data science software platform to analyze survey data.

The x-means clusterization method (18), which is part of RapidMiner software was used to clusterize respondents into two sets described above. Z-scores were used to specify how more experienced and less experienced developers scored with the relationship to the mean (19). Software found that more experienced developers have scored on average 2 times as high in self-evaluation in knowledge of Docker/Kubernetes technology and 1.6 times as high in self-evaluation of knowledge of CI/CD based on Aja/Jenkins compared to less experienced developers (see table 1 below):

TABLE 1. Clusterization of respondents into two groups based on competence.

	# of respondents
Less experienced developers	67
More experienced developers	34

In order to evaluate the performance of an already implemented solution, a proposed solution and performance of the company in a given sphere of expertise, respondents were asked to provide a score from “0” to “10”. The score of “0” out of “10”, depending on the context, stands for: “Very Ineffective”, “Low Influence”, “Not Preferable”, “No knowledge”. The score of “10” out of “10” stands in opposition to the meaning of the score “0” out of “10”: “Very Effective”, “High Influence”, “Highly Preferable”, “Expert”.

4.4.2 Version Control System

The new software delivery solution was developed with the vision of Git VCS being a version control system of choice. This led to the choice of GitOps – a software delivery approach where the configuration and state management for operations are done through the storage of configuration files in a Git repository that operations software has access to.

Another reason for moving away from the current VCS (Mercurial) has been trends data, which clearly shows a thinning Mercurial community in comparison to Git (see figure 16). This leads to three issues: decreasing support and the number of plugins to improve the VCS performance as well as difficulties in the adaptation of Mercurial by new employees.

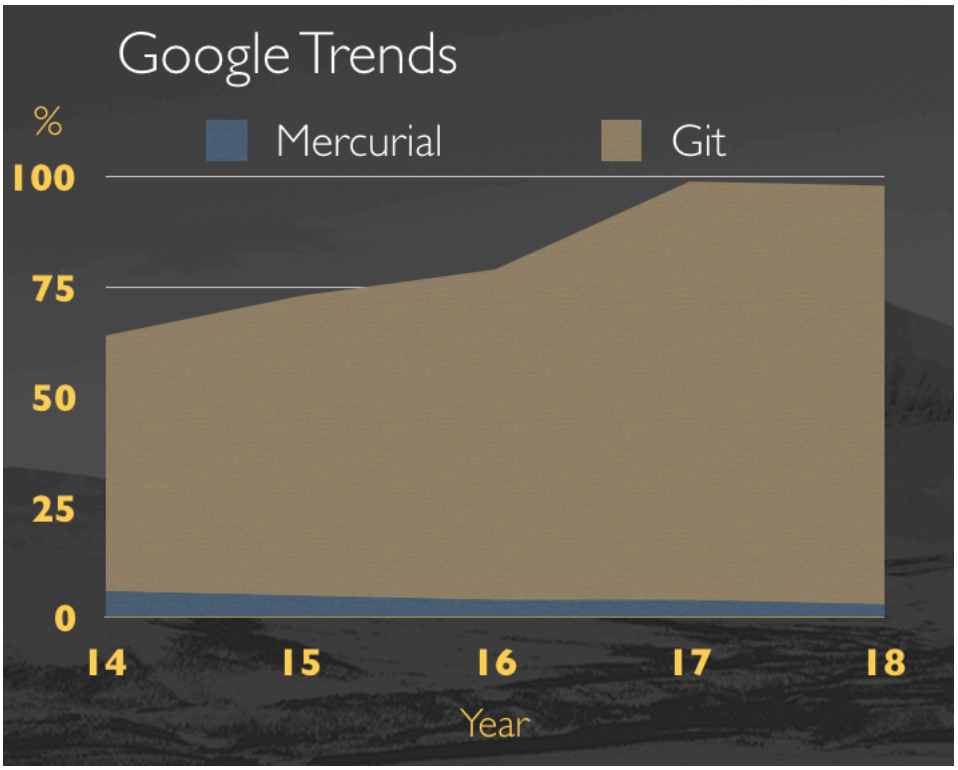


FIGURE 16. Google Trends. Trends in VCS.

The following questions were used to gather data about employees' knowledge, perception and opinions with regards to VCS (see APPENDIX 2):

- How experienced are you working with Mercurial VCS?

- How experienced are you working with Git VCS?
- How preferable is it to allow developers to pick VCS of their choice for a new project?
- How preferable it is to use Mercurial as a version control system?
- How preferable it is to use Git as a version control system?

Whilst for a quarter of respondents, the choice of VCS has not been of importance, on average Git scored significantly higher than Mercurial, mostly due to a large number of proponents for the usage of Git VCS (scoring 10) and a significant number of developers opposing usage of Mercurial VCS (scoring 0), see figure 17.

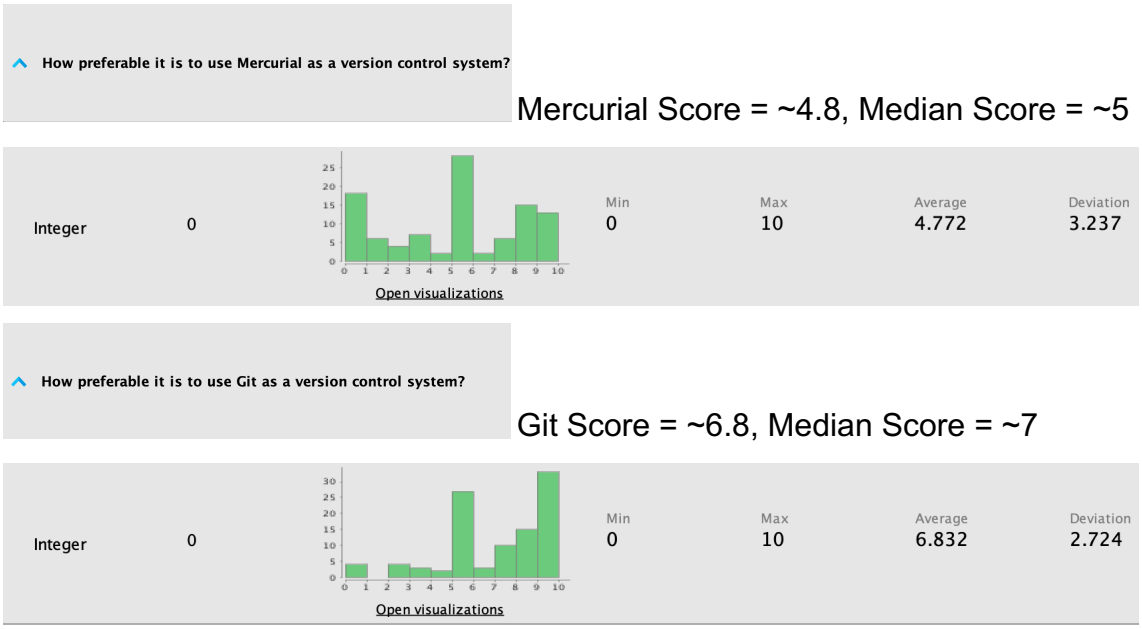


FIGURE 17. Vaimo Group. The preference in VCS.

Moreover, there is a significant correlation between the professional competence and choice of VCS, where more experienced developers prefer Git as a version control system. Less experienced developers, according to data, have a slight preference for Mercurial (see table 2 below).

TABLE 2. Z-scores for clusters of less experienced and more experienced developers with regards to the choice of VCS solution.

Z-score	Git	Mercurial
Less experienced developers	-0.157	0.029
More experienced developers	0.310	-0.057

Survey respondents were also asked to provide self-evaluation of their knowledge of Git and Mercurial (see figure 18). Data reveals similar knowledge level for both Mercurial VCS and Git VCS with a similar standard deviation.

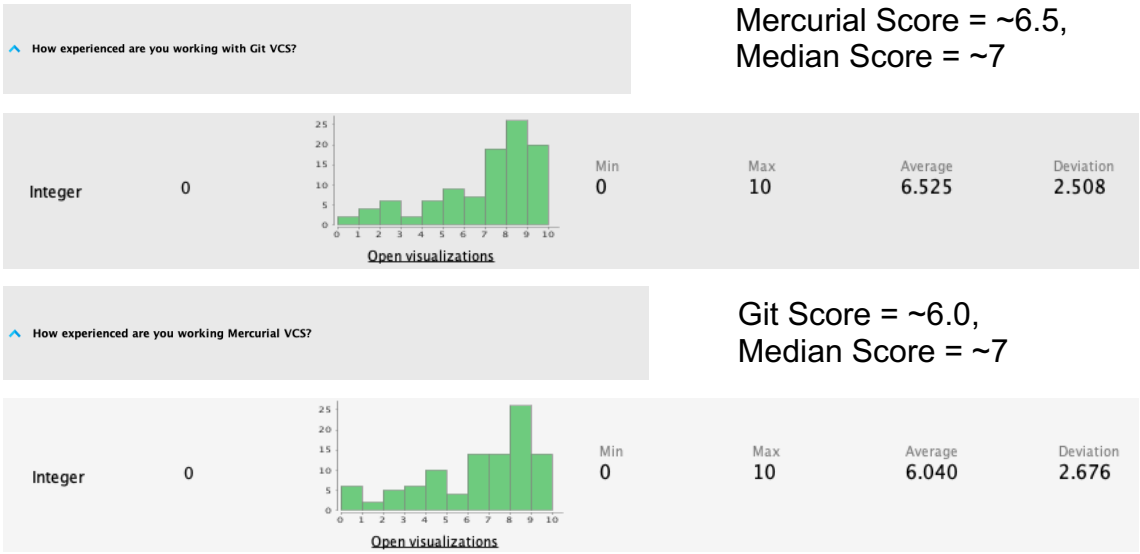


FIGURE 18. Vaimo Group. Knowledge of VCS.

One of the pathways forward is to allow developers to pick a version-control system of their choice for a given project. Even though it allows freedom in the way data is manipulated technically, which may bring advantages to a given project, it introduces a set of challenges:

- The software delivery framework would need to be heavily refactored, since it is designed to work with only one type of VCS.
- Communication between projects/teams would have barriers if projects/teams use different VCS solutions.

- Managing projects with different VCS and different plugins to enhance VCS would add more complexity for the version control processes.
- Developers would have to strive for excellence in several, mostly interchangeable technologies.

It can be seen from statistics below (figure 19) that such approach has both proponents and opponents with proponents supporting this approach due to the fact that it promotes the choice and responsibility, as well as the choice of tools that are best suited for a given team.

FIGURE 19. Vaimo Group. The preference in free choice of VCS for a new project.

As can be seen form table 3, more experienced developers are more likely to support the free choice of a version control system for a new project.

TABLE 3. Z-scores for clusters of less experienced and more experienced developers with regards to preference for free choice of VCS solution.

Z-score	Freedom of VCS Choice
Less experienced developers	-0.099
More experienced developers	0.196

Overall, it can be advised to use Git VCS for new projects due to the fact that it has a large number of proponents, especially among experienced developers. The free choice of VCS should be restricted due to the fact that there are technological and communicational difficulties that may arise out of such decision and, hence, the impossibility of making pipeline on schedule a feasible CI/CD that would support it. However, the free choice of VCS should be allowed for the cases where the CI/CD pipeline is not involved (e.g. internal projects) and the team reaches a consensus on using VCS other than Git.

4.4.3 Quality Assurance and Visualization

Survey data regarding the quality assurance, server and CI/CD status visualization, as currently implemented in the company, should be analyzed in the same chapter, due to the similarity of results.

The following questions were used to gather data about employees' knowledge, perception and opinions with regards to existing QA procedures (see APPENDIX 2):

- How effective are current QA procedures?
- What QA procedures are currently most ineffective?

The following questions were used to gather data about employees' knowledge, perception and opinions with regards to existing visualization tools (see APPENDIX 2):

- How effective is the current CI/CD and server status visualization?
- What CI/CD or server visualization solution is lacking?

Vaimo engineers have estimated both status visualization and quality assurance to be carried out satisfactory, with a third of developers giving it ranking of 5 out of 10.

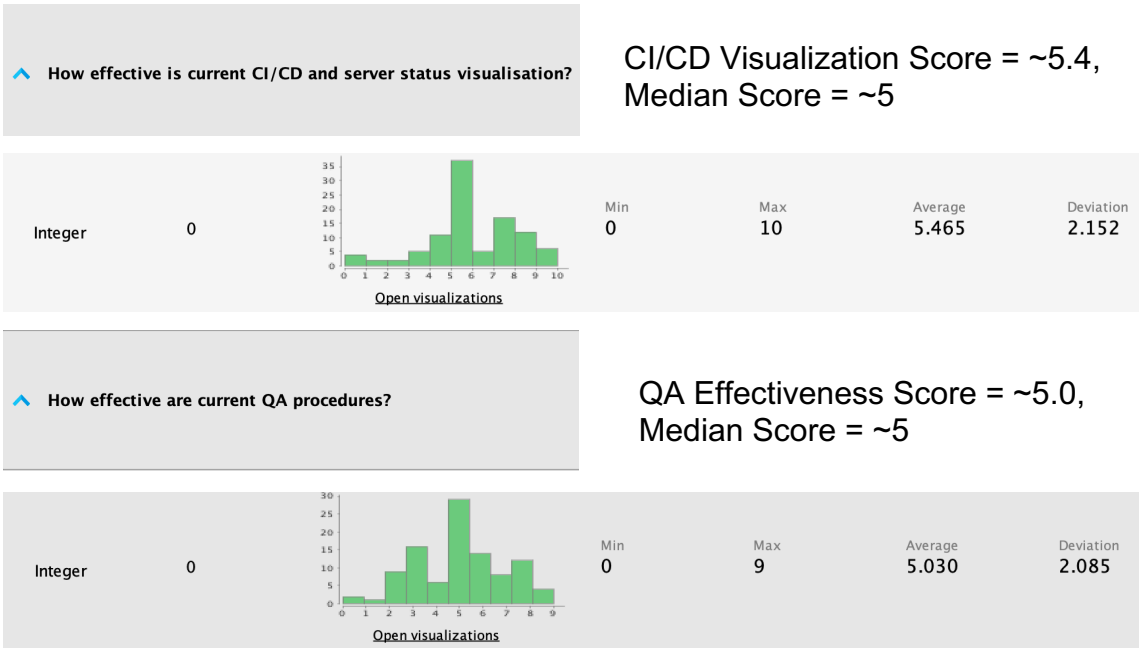


FIGURE 20. Vaimo Group. Effectiveness of QA and status visualization.

However, after having a close look at the answers distribution and text input gathered from respondents, the following observations can be made:

- Many of respondents, though giving scores higher or equal to five, have provided responses to questions “What QA procedures are currently most ineffective?” and “What CI/CD or server visualization solution is lacking?”, which signify that they are unaware of solutions that Vaimo offers for QA and status visualization. This suggests that the data is to a significant degree skewed toward positive values, due to the fact that for a subset of respondents, the real value of the solutions described is zero, since potential users are unaware of its existence.
- The distribution of values for status visualization and QA is different, with only 24% of respondents ranking status visualization below 5, whereas for QA 34% of respondents give this score. It is likely to be the case that status visualization does not require a high level of professional competence to be valuable, since in Vaimo it is largely GUI-based with Aja, New Relic, Jenkins and Grafana tools providing good user experience. At the same time, much of current QA automation is based on technologies that are project-specific and require a significant level of expertise in test automation to be utilized (e.g. unit tests, robot framework). This can explain the difference in data distribution. Statistics

show that “more experienced developers” (see table 4 below) are more likely to give higher rankings to visualization and especially, QA solutions implemented in Vaimo.

TABLE 4. Z-scores for clusters of less experienced and more experienced developers with regards to evaluation of QA suite and software delivery visualization effectiveness.

Z-score	QA Effectiveness	Visualization Effectiveness
Less experienced developers	-0.150	-0.029
More experienced developers	0.296	0.057

It is also of value to classify response types for the textual input.
Types of status visualization responses received from respondents are shown below in the table 5.

TABLE 5. Groups of most common response types with regards to inefficiencies in CI/CD and server visualization as currently implemented.

	Number of Answers
Confusion	6
”Big Picture Visualization”	5
Build/Deploy	7
Other	1

It is possible to see from table 6, that most respondents are either confused if there are any visualization solutions available or are unaware of big picture

visualization solutions provided by the company. Other group of answers belong to requests for the current CI/CD pipeline visualization improvement.

It is also of value to classify response types/requests for the text input. Types of QA visualization responses received from respondents are shown in the table 6 below.

TABLE 6. Groups of most common response types with regards to inefficiencies in QA as currently implemented.

	# of responses
Confusion	4
Management	4
Test Automation	26
Other	2

An overwhelming number of respondents, around 70%, claimed test automation to be the biggest bottleneck to effective quality assurance processes. Others described task management to be not efficient enough for effective testing, due to the fact that the scope of the task is often blurred, or acceptance criteria are not well-defined. Few respondents expressed their confusion with current QA processes implemented in Vaimo.

From data above it is possible to suggest the following changes:

- Implementing a QA test automation suite should be a top priority with over 70% of respondents who left written feedback naming it to be the core issue in company's quality assurance procedures. This is the most feasible solution that would lead to an increase in the QA effectiveness.
- The analysis of data related to visualization leads to propose that status visualization utilities/tools available in Vaimo are perceived, to a

significant degree, to be inexistent by the employees. In order to increase the perceived value of visualization tools, it is recommended to spread knowledge of tools such as Grafana, New Relic, Jenkins and Aja as well as of possibilities that they offer. In particular “big picture” visualization solutions, which allow to monitor the server load and provide the web analytics such as Grafana and New Relic, were unknown to at least 60% of respondents leaving written feedback.

- QA lectures for employees that would provide tools for an easy start in the development of unit tests are recommended so as to boost involvement of “less experienced developers” in QA procedures. Due to the fact that such improvement requires significant training of the employees and its capabilities are limited by its nature, feasibility (see ch 4.3.2) of pursuing this pathway is lower than of creation a QA automation suite.

4.4.4 CI/CD automation

Automation allows to reduce the amount of manual work so that the employees could concentrate on tasks that cannot be automated at a reasonable cost. One of the most essential technological capabilities to develop to achieve the efficient use of technology, from the DevOps perspective, is build and deployment automation among other capabilities, such as the effective use of version control.

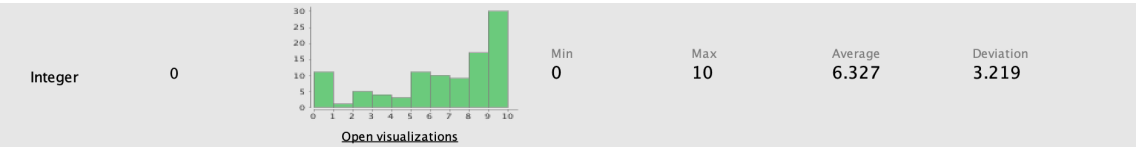
It is important to assess which of the automation capabilities could be used in the delivery of an e-commerce web-store platform. Currently, there are three software delivery capabilities, which are possible to implement within the new software delivery system: automatic builds, automatic deployment to the production environment, automatic deployment to a non-production environment.

The following questions were used to gather data about employees' opinion on build and deploy automation practices and their applicability in software delivery with regards to e-commerce web-stores (see APPENDIX 2).

- How preferable is it to have a build process initiated automatically after commit?
- How preferable is it to have deployment process initiated automatically after the build for production environment?
- How preferable is it to have deployment process initiated automatically after the build for non-production environment?

Since there are two stages, build and deployment, it is of relevance to separate these two processes to receive more accurate information on the opinion of professionals. In order to make the received data even more accurate, employees are requested to provide their opinion for a deployment process carried out for production and for non-production environments. It is necessary to gather these data entries separately to compare feasibility of a certain type of automation.

The response for build automation has been overwhelmingly positive with 60% of respondents providing score 8 out of 10 or higher. The median score for all employee groups was above 5 out of 10. Around 10% of employees expressed a negative attitude to the idea giving it score of 0 out of 10. (see figure 21)



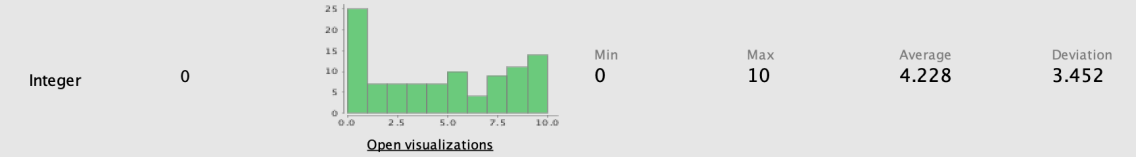
Build Automation Score = ~6.3,
Median Score = ~7

FIGURE 21. Vaimo Group. The build automation after commit has been pushed.

The only software delivery automation process that has received radically negative scores was deployment automation to the production environment with approximately 25% of employees giving it the score 0 out of 10 (see figure 22).

Such large number of opponents means that implementing this solution would require extensive communication with employees, who expressed such opinion to see whether the arguments on their part are valid and can be resolved. It is of knowledge among company's employees that many deployments are to be done on periodic bases, agreed upon with the client and done when the web-store is experiencing low traffic. These requirements make it desirable or required, as seen by many developers, to have a manual step in the deployment process to the environment, where availability is critical.

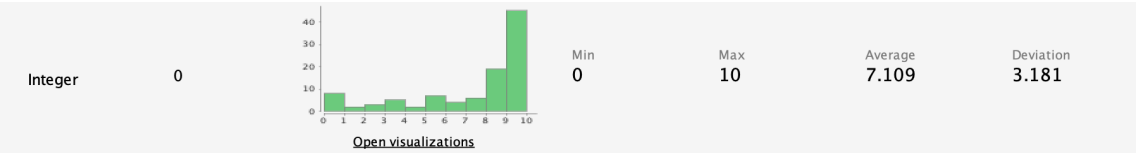
The further analysis shows that a negative attitude (the median score lower than 5 out of 10) to the solution was expressed by every group of employees except for QA, with the lowest median score given by technical managers and developers.



Deploy Automation (production) Score = ~4.2,
Median Score = ~4

FIGURE 22. Vaimo Group. The deployment automation for production environments.

Deployment automation to a non-production environment had an incredibly positive reception with 45% of employees giving it the score of 10 out of 10 and a very insignificant number of opponents of the idea.



Deploy Automation (non-production) Score = ~7.1,
Median Score = ~8

FIGURE 23. Vaimo Group. The deployment automation for non-production environments.

It holds true for all of the three processes described above that attitude to their use positively correlates with competence, making professional developers more likely to vote in favor of their use (see table 7 below).

TABLE 7. Z-scores for clusters of less experienced and more experienced developers with regards to preference for automatic build/deployment capabilities.

Z-score	Builds Automation	Non-production Deployments Automation	Production Deployments Automation
Less experienced developers	-0.092	-0.222	-0.183
More experienced developers	0.182	0.437	0.360

In conclusion, the practice of build and deployment automation received positive feedback, with a very significant positive correlation between competence and each of the practices. It is advised that the new CI/CD pipeline has capabilities for automating builds (all environments) and deployments (only to non-production environments) due to the fact that an average score given to these approaches is high. However, deployment automation to the production environment cannot be done due to the fact that it is largely opposed by employees for well-known business-related reasons. It is of benefit to go through the reasoning preventing the implementation of automated deployment for the production environment and see whether it is feasible to resolve the issues raised or, otherwise, not allow automated deployments to production.

4.4.5 Utilities

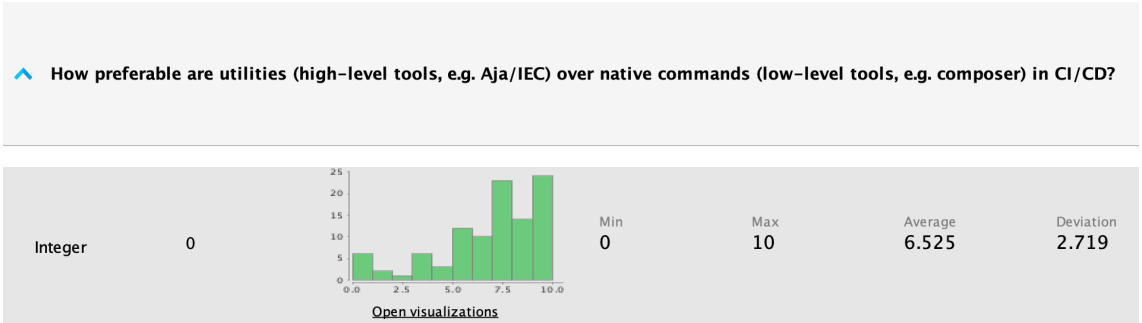
The following questions were used to gather data about employees' opinion on the degree to which utilities for working with software delivery pipeline should be provided for internal use. Employees' opinion on how preferable it is to have the free choice of CI/CD tools for new projects was also measured (see APPENDIX 2):

- How preferable are utilities (high-level tools, e.g. Aja/IEC) over native commands (low-level tools, e.g. composer) in CI/CD?
- How preferable is it to allow developers to pick CI/CD tools of their choice for a new project?

Implementing high level utilities allows to abstract from the underlying technology and provide automation for most commonly executed processes. This comes as an alternative to having multiple complex low-level tools that allow a fine-grained execution of commands, but at the same time, they make execution of most common commands/workflows less user-friendly and bring a room for accidental errors.

The survey requested employees to provide a score for how preferable it is to rely on high-level utilities (e.g. Aja or IEC, both of which are internal to the company).

The response was overwhelmingly positive (see figure 24), with more than half of respondents giving the score of 8 out of 10 or above.



Advanced Utilites Score = ~6.5,
Median Score = ~7

FIGURE 24. Vaimo Group. The request for advanced utilities for CI/CD processes.

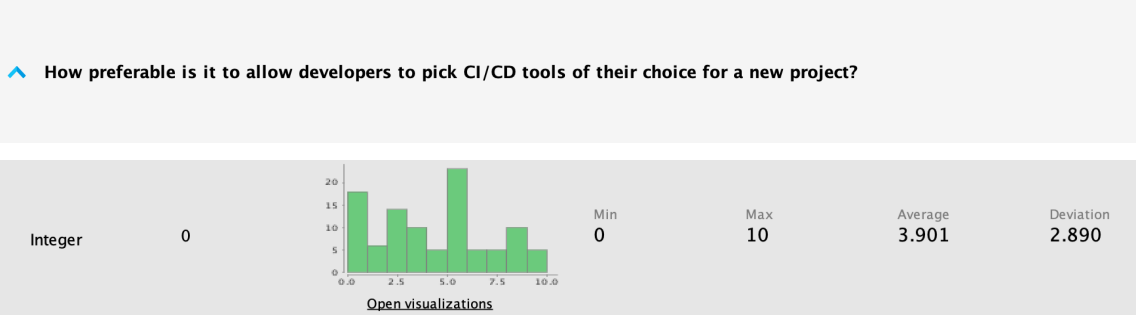
TABLE 8. Z-scores for clusters of less experienced and more experienced developers with regards to preference for full-fledged set of internal utilities as well as preference for free choice of the utilities.

Z-score	Utilites	Free Choice of CI/CD Tools
Less experienced developers	0.054	-0.162
More experienced developers	-0.106	0.319

Even though, employees seem to prefer having a utility/set that would simplify most of the processes related to software delivery, there exists a slight negative correlation between the professional competence and the usage of high-level utilities. This means that employees having enough experience would prefer to have direct access to low-level commands.

Respondents were also asked if they would want to have freedom of choice in selecting CI/CD tools when working with the existing software delivery pipeline or if it were more reasonable to offer a pre-defined set of tools for managing existing software delivery processes.

There is a significant number of respondents (around ~18%) strongly opposing the free choice in CI/CD tooling with half of respondents giving the score of 3 out of 10 or less. A quarter of respondents have not given preference to either of high-level or low-level utilities (see figure 24). There can also be found a significant correlation between the competence and preference for freedom of choice in CI/CD tooling (see table 8 below).



CI/CD Tools Choice Score = ~3.9,
Median Score = ~4

FIGURE 25. Vaimo Group. The free choice of CI/CD tools for a new project.

Due to the fact that high-level utilities have a lot of proponents, it can be advised that development of such tools is necessary, leaving the opportunity to work with low-level tools for more experienced developers in case there is a failure in CI/CD tooling or if they prefer to do so.

It is recommended to not allow the free choice in CI/CD tooling due to a significant number of negative scores received from respondents. However, due to the fact that there is a preference for the freedom of choice in CI/CD tooling among more experienced developers, it is advised to allow such freedom in projects/teams which are to a significant degree siloed from mainstream development (e.g. hosting team, platform team, Vaimo Group Services).

4.4.6 Hosting and virtualization

The following questions were used to gather data about employees' opinion on the type of infrastructure that should be used for deployment of an e-commerce

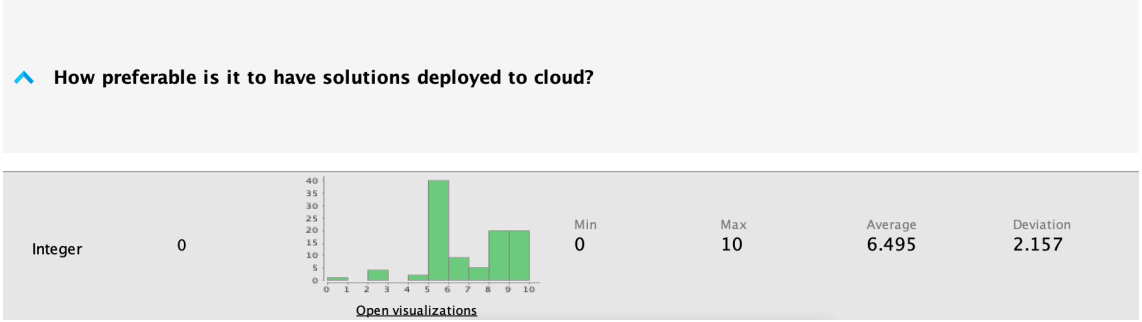
web-store as well as preference for the virtualization technology to be placed on it (see APPENDIX 2).

- How preferable is it to have solutions deployed to cloud?
- How preferable is it to have solutions deployed to traditional hosting?
- How preferable is it to have solutions deployed on containers running on a server/cloud?
- How preferable is it to have solutions deployed directly onto a VM running on a server/cloud?

The questions above are designed to request employees' preference for the type of infrastructure they see as the most beneficial for an e-commerce web-store (cloud or traditional hosting) as well as the technology used for creating an environment on such an infrastructure (a container or a virtual machine).

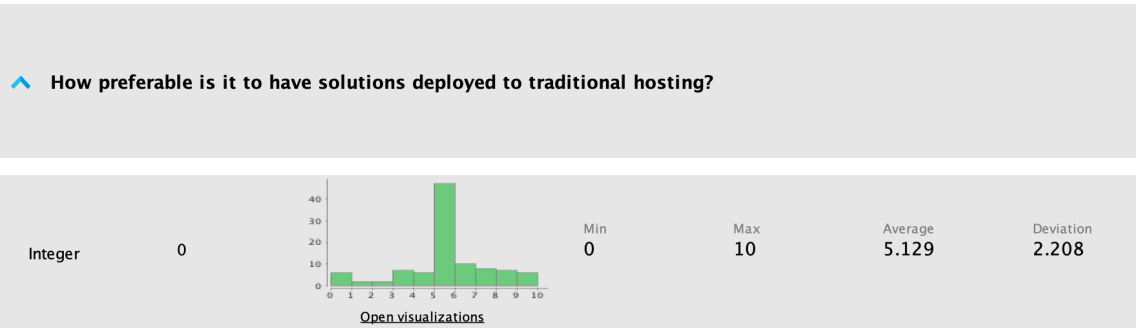
It is of benefit to analyze the results in terms of the infrastructure type and virtualization technology.

Respondents were asked to give scores to two types of infrastructure that could be used as a destination for the CI/CD pipeline being developed: cloud and traditional hosting. A significant number of respondents (over 40%) have given no relevance to the infrastructure type providing the score 5 out of 10. Even though almost half of respondents considered the choice in the infrastructure type as not relevant, the cloud infrastructure got many proponents with over 40% of respondents giving it the score of 9 out of 10 or higher (see figure 26). Traditional hosting solutions have received a rather flat score distribution on scale from 0 to 10. (see figure 27). In addition to that, there is a very significant positive correlation between the competence and infrastructure type preferred. A server-based solution is more likely to be chosen by less experienced developers. (see table 9 below)



Cloud Hosting Score = ~6.5,
Median Score = ~6

FIGURE 26. Vaimo Group. Deployment to cloud.



Traditional Hosting Score = ~5.1,
Median Score = ~5

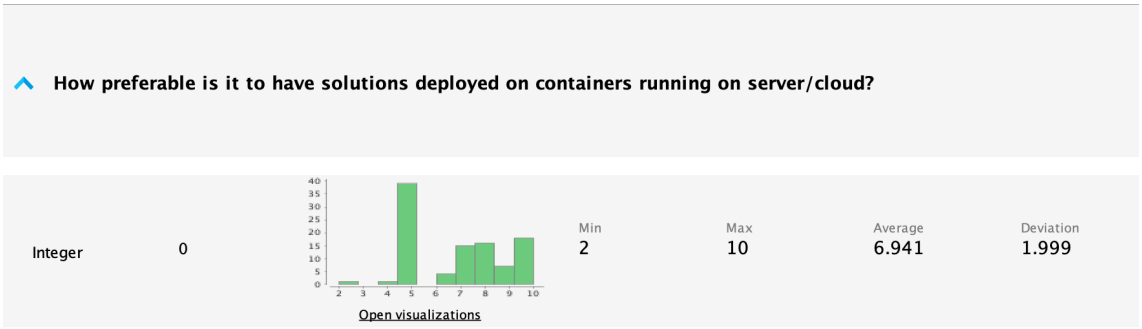
FIGURE 27. Vaimo Group. Deployment to traditional hosting (dedicated servers).

TABLE 9. Z-scores for clusters of less experienced and more experienced developers with regards to preference for hosting type.

Z-score	Server	Cloud
Less experienced developers	0.138	-0.458
More experienced developers	-0.271	0.902

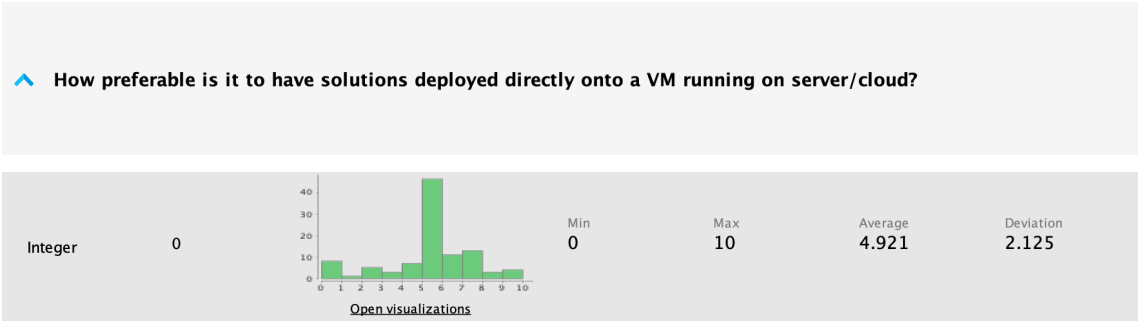
As with the infrastructure type and virtualization technology, there is over 40% of developers treating question as irrelevant giving the solutions the score of 5

out of 10 (see figure 26-29). For containers, however, virtually the rest of the respondents have provided scores higher than 5 out of 10, which signifies a positive attitude towards the container-based technology with the average score for it being 6.9 out of 10 compared to average score of 4.9 for traditional virtualization technologies. The correlation between the competence and choice in infrastructure is very strong with professional developers highly leaning towards a container-based pipeline (see table 10 below).



Containers Score = ~6.9,
Median Score = ~7

FIGURE 28. Vaimo Group. Deployment on containers.



Virtual Machines Score = ~4.9,
Median Score = ~5

FIGURE 29. Vaimo Group. Deployment on virtual machines.

TABLE 10. Z-scores for clusters of less experienced and more experienced developers with regards to preference for a virtualization technology.

Z-score	Containers	VM
Less experienced developers	-0.493	0.093
More experienced developers	0.971	-0.184

Since developing a CI/CD pipeline is a highly technical endeavor which would impact all groups of employees, bringing opinions of different groups of employees in Vaimo Group is of value. The attitude of different groups of employees towards the usage of cloud infrastructure and containers is best represented using boxplots (20) that are used in figure 30 and figure 31. From figures below, it can be stated that both technical and business (“Other”) management of the company support both containerization and cloud technologies giving them the score of 5 out of 10 or higher. Even though, developers and quality assurance professionals tend to give higher median scores to the technologies above, there is a minor number of employees at these positions who oppose building a new CI/CD pipeline using these technologies.

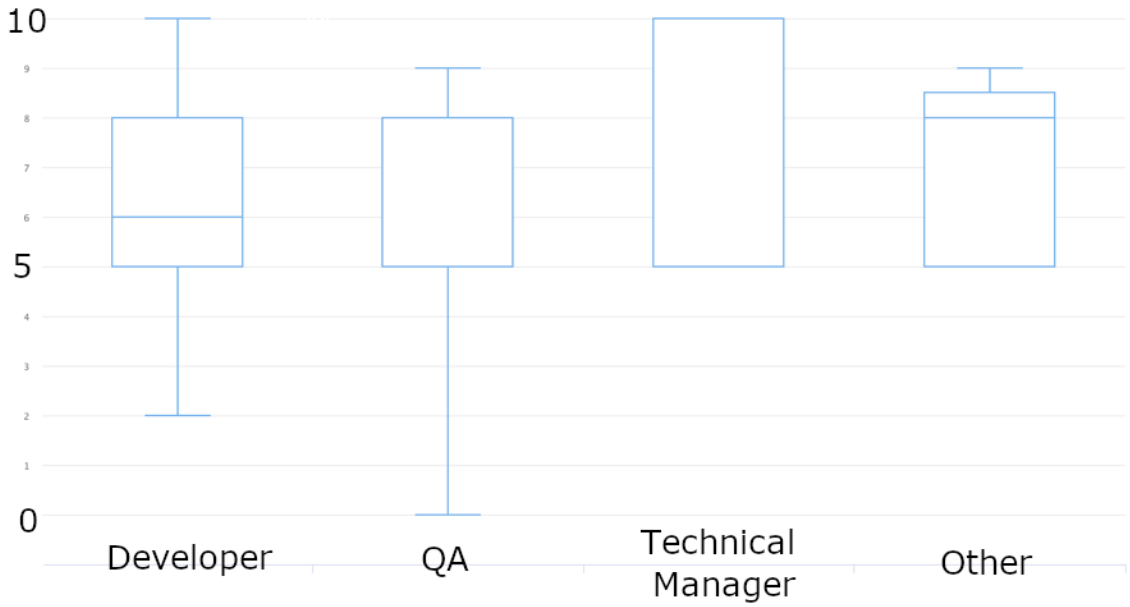


FIGURE 30. Vaimo Group. The preference for cloud infrastructure among different employee groups analyzed using boxplot.

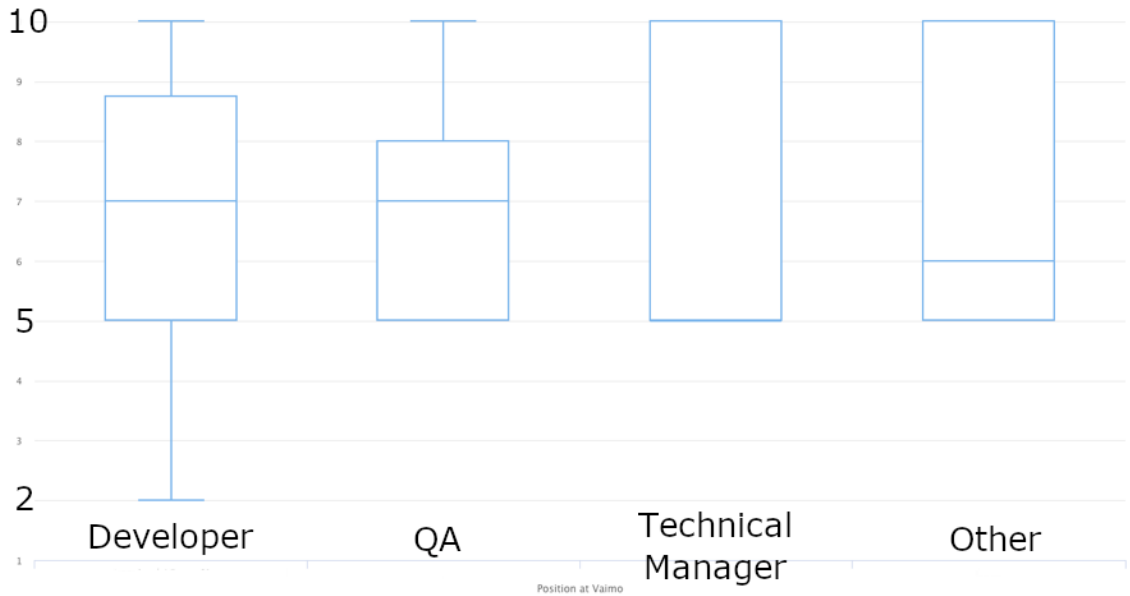


FIGURE 31. Vaimo Group. The preference for containerization among different employee groups analyzed using boxplot.

It can be concluded, that the new CI/CD pipeline should be based on the container technology with the subsequent delivery of containers to a public/private cloud. The analysis of the data can be characterized by choice in hosting and virtualization technologies being irrelevant to a significant number of respondents. As of the remaining, especially more experienced workers, the

choice has been done overwhelmingly in favor of cloud infrastructure and containers.

4.4.7 Development time spent on CI/CD

It was determined to include questions that would allow to create a picture of how much work time is spent on CI/CD operations and failures that they cause (see APPENDIX 2):

- How much time do you spend on CI/CD operations?
- How much development time do you spend on failures caused by CI/CD processes?
- Which CI/CD operations are currently time-inefficient? Why?

Time spent on CI/CD operations is represented in the figure 32. It can be seen that 75% of developers spend less than 10% of their working time on CI/CD operations.

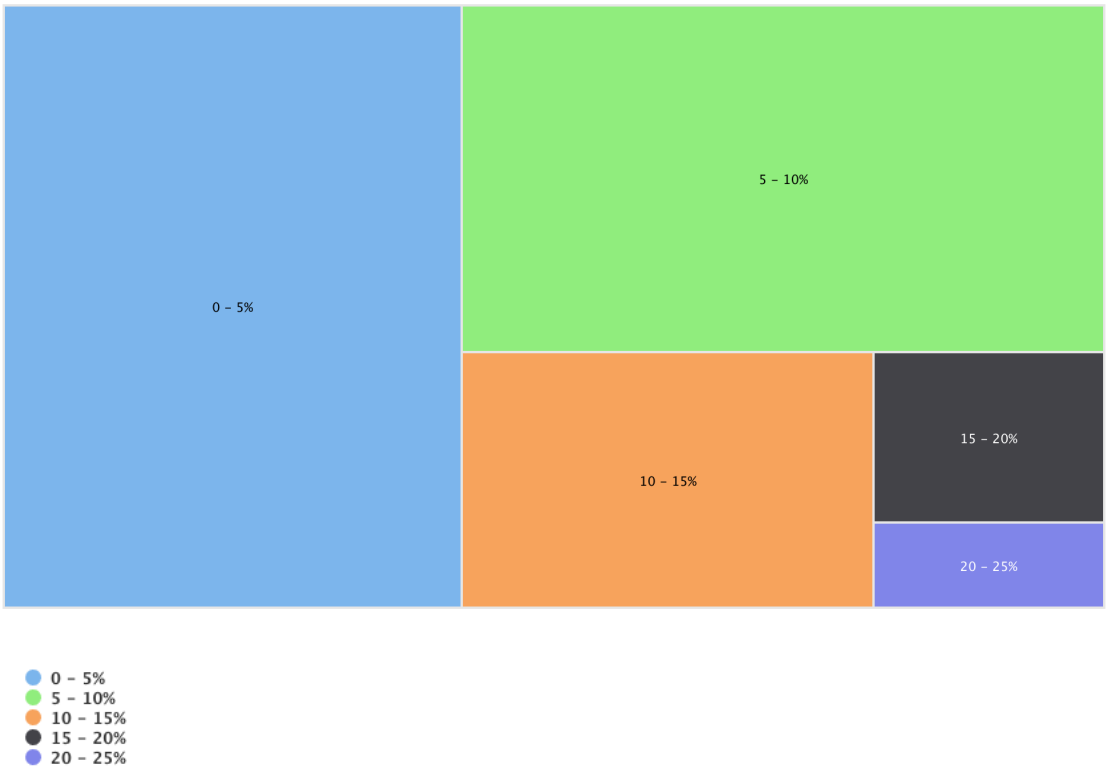


FIGURE 32. Vaimo Group. Work time (in per cent) spent on CI/CD processes, represented using treemap.

Time spent on resolving CI/CD failures is represented in the figure 33. The treemap shows similar results as with overall time spent on CI/CD operations, with 75% of developers spending less than 10% of working time on the CI/CD failures. However, in case of failures, it can be seen that almost 65% of employees spend less than 5% of time on resolving CI/CD issues, whereas for CI/CD operations overall this number is 45%.

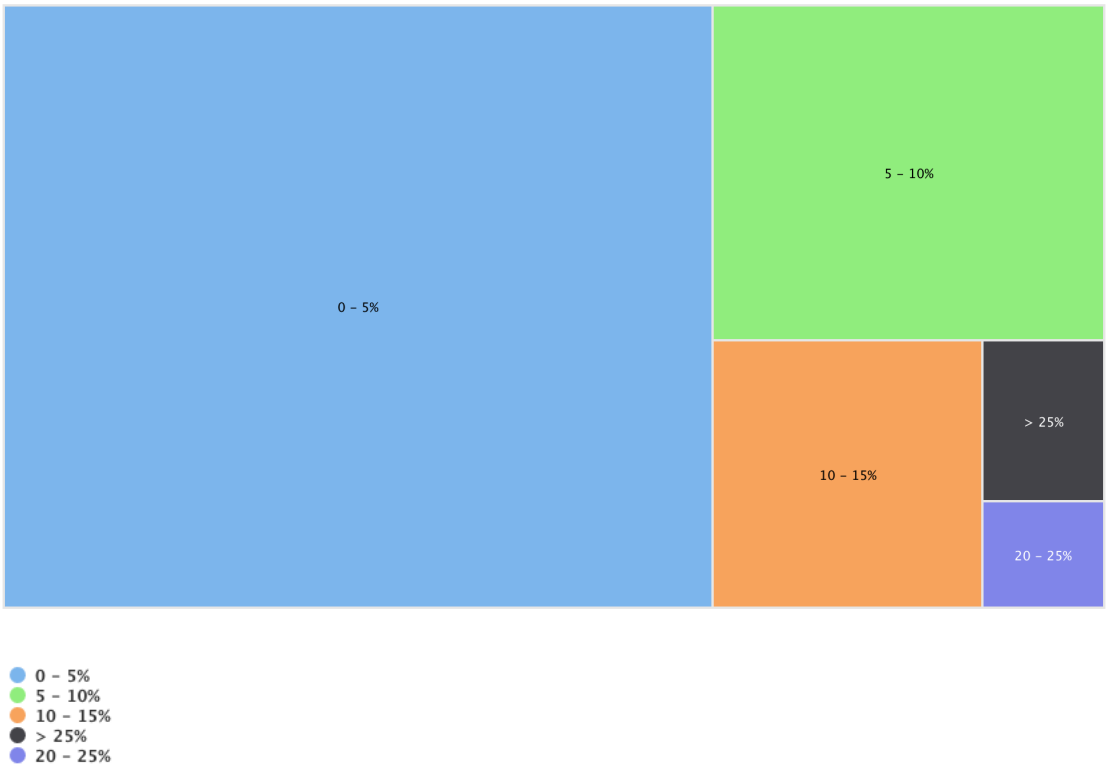


FIGURE 33. Vaimo Group. Work time (in per cent) spent on failures caused by CI/CD processes, represented using treemap.

From the analysis of textual input, it is possible to define several types of processes that are referred to as most time consuming by employees. Around 60% of respondents leaving textual feedback have claimed that the build process takes a significant amount of time, especially in case of a larger project. Around 10% of respondents were concerned with the inefficient Aja module/project dependency update mechanism, claiming it to take more time than would be desirable. Approximately 20% of respondents would define most time consuming to be CI configuration as well as debugging failures that appeared during build processes. (see table 11 below)

TABLE 11. Groups of most time-consuming CI/CD tasks.

	# of responses
Build	31
Environment Configuration	5
Aja module/project updates	5
Working with CI failures	6
Other	2

The current CI/CD pipeline is time-efficient. The CI/CD solution being developed should be characterized by 75% of developers spending less than 10% of their working time on CI/CD operations. One of the most time-consuming operations is build process as identified by most respondents. The issue that should lead to the inquiry and investigation in potential resolution strategies and a possibility of working with the Magento community on resolving this particular issue. Delays in Aja module/project updates as well as the complexity of environment configuration should also be further investigated.

4.4.8 Responsibility and Burnout

The following questions were used to gather data about employees' attitude towards taking on partial responsibility for the operations as well the perception of employees with regards to stress levels caused by software delivery practices (see APPENDIX 2).

- How preferable is it for development team to take the operational responsibility?
- How much influence do current CI/CD practices have on the stress level at work?

One of the key aspects of the DevOps culture is the shared responsibility. At the moment of writing, there exists a significant gap in Vaimo Group between development and operations tasks. This does not impact the quality of work done by operations and development directly, however, it lowers efficiency and limits the speed at which improvements could be introduced, due to the low bandwidth and creation of silos. The technical management of the company along with several other active proponents of the idea have been promoting to pass part of the operational responsibility to development. The survey provides data on employees' attitude to passing the operational responsibility to software developers.

In survey data there was found a strong positive correlation between competence and preference for passing the partial operational responsibility to developers (see table 12 below). Most respondents agreed to push the partial operational responsibility from operations to development (see figure 34) with, however, ten percent of respondents evaluating this change with the score of 0 out of 10 as "Not Preferable".

It can be seen from figure 35 that the technical management of the company supports passing the partial operational responsibility to developers with the median score given being 8 out of 10.

Operational Responsibility for Developers Score = ~5.5,
Median Score = ~6

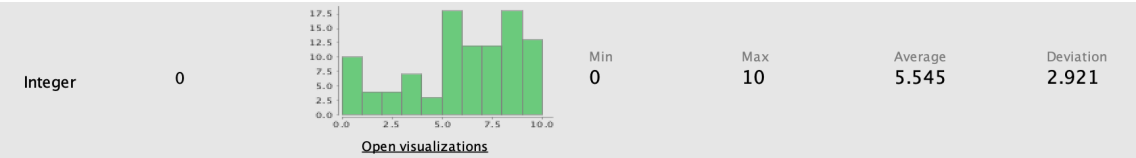


FIGURE 34. Vaimo Group. Developers having partial operational responsibility.

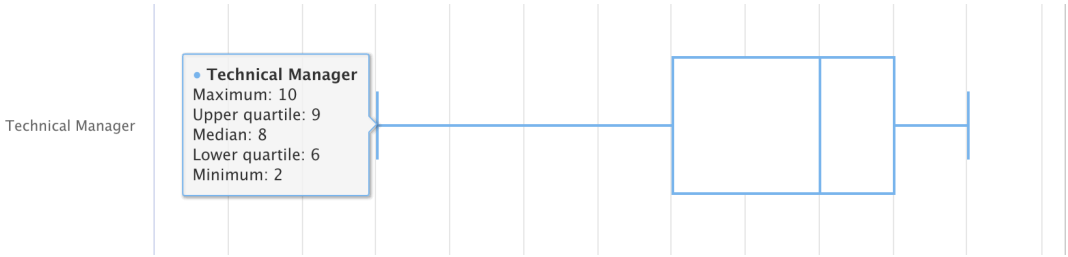


FIGURE 35. Vaimo Group. Developers having partial operational responsibility. Evaluation by technical management represented using boxplot.

It is important to note that changes regarding the operational responsibility, if implemented, will require a change in roles and responsibilities for employees involved in operations and development. Operations, for example, will need to concentrate on facilitating developers' access to hosting facilities (cloud) as well as concentrate on the improvement of existing tools and development of new ones.

TABLE 12. Z-scores for clusters of less experienced and more experienced developers with regards to preference for taking operational responsibility as well as evaluation of stress due to software delivery processes currently implemented.

Z-score	Operational Responsibility	Stress
Less experienced developers	-0.243	0.019
More experienced developers	0.478	-0.038

According to the information received from the survey, employees' stress levels caused by CI/CD processes are acceptable with only 25% of employees giving a score of 6 out of 10 or higher (see figure 36).

Stress Level Score = ~4.0,
Median Score = ~3

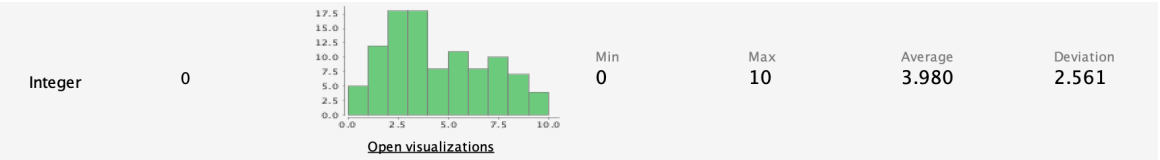


FIGURE 36. Vaimo Group. Stress as perceived by employees due to CI/CD operations.

It can also be seen from table 12 that there is a minor negative correlation between the competence and perceived stress level due to software delivery procedures. Another data analysis technique shows that technical managers have a slightly higher median value for the experienced stress level. Both of these scores can be explained by the fact that this group is often responsible for solving more complex issues, often failures caused by CI/CD processes. These results also suggest that more experienced developers have a slightly higher stress tolerance which comes with the professional competence that gives the ability to resolve complications arising in software delivery processes.

It can be concluded that the operational responsibility is recommended to be partially moved to developers, after restructuring roles assigned to employees in both operations and development areas of work as well as requesting detailed feedback and critique of such a change from respondents that were skeptical of the change (around 10-20% of respondents). It can be claimed that stress levels experienced by employees in the company due to software delivery processes are acceptable and are not reliably predictable by competence, which can be due to the shared responsibility and good communication.

5 SOFTWARE DELIVERY TRANSFORMATION

In this thesis, after describing the implemented and proposed software delivery pipelines (see ch. 3), the research in software delivery was carried out (see ch. 4). An in-detail description of the proposed software delivery pipeline architecture or the working processes is out of scope of this research. However, it is of relevance to delineate the work that was done in parallel with the research at hand and that has allowed to select a proper approach to surveying professionals in software delivery (see ch. 5.1). Moreover, it is of relevance to describe teams responsible for software delivery transformation (see ch. 5.2).

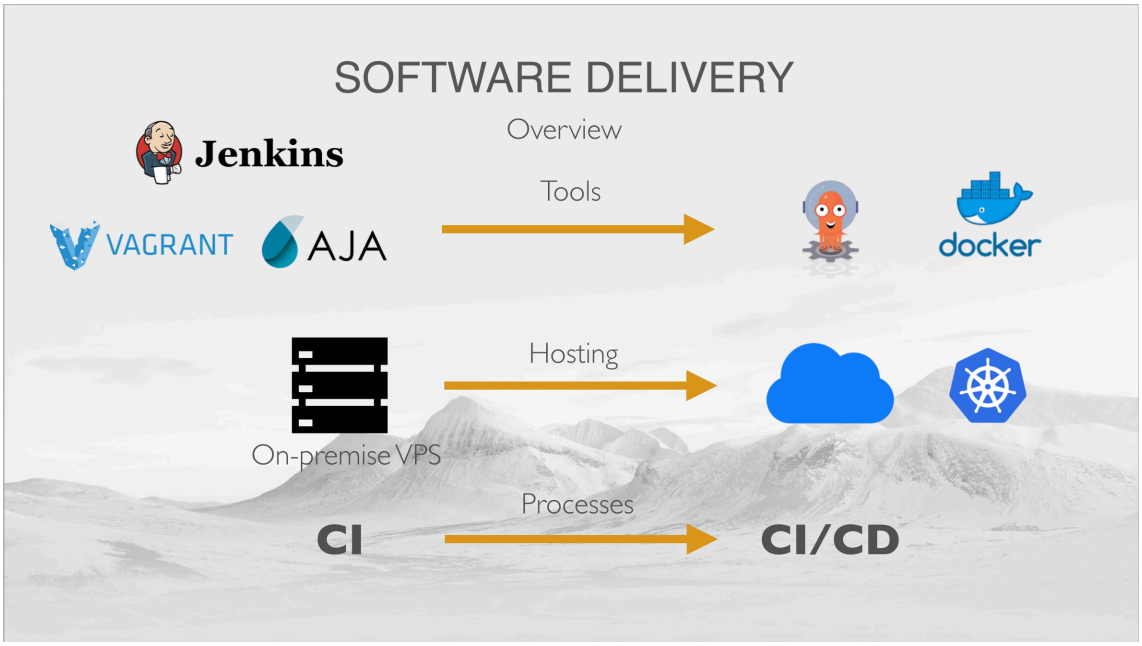


FIGURE 37. Vaimo Finland. The migration to a container-based software delivery pipeline.

5.1 Progress

Since the development of the core of the proposed CI/CD pipeline was carried out before the start of the research at hand, only piloting the proposed software delivery pipeline as well as the research in software delivery capabilities and the current performance are herein described as the progress.

For the duration of the research (3 months), 3 web stores on a single Magento 2 instance were delivered to the customer and maintained using the proposed CI/CD pipeline. Currently, there are scheduled CI/CD pipeline migrations for four independent projects. It is projected that within one month, a team of three professionals working on the task part-time, could deliver a migration strategy for existing projects. (see figure 38)

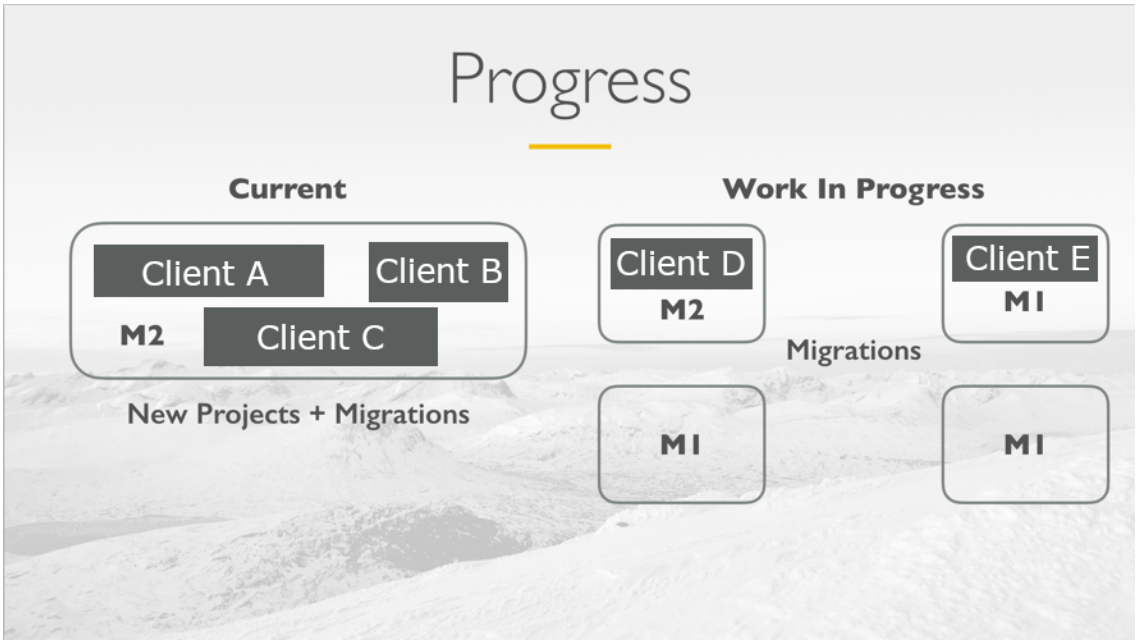


FIGURE 38. Vaimo Finland. Migrating customer projects to a container-based software delivery pipeline.

During the research, in order to provide user-friendly documentation, there was created a book which documents operations that the new software delivery pipeline is capable of handling (see APPENDIX 3). The author of this research was in charge of documenting the operations described above. As required by the management of the company, documentation was written using easy-to-follow instructions and the table of contents. In order to allow migration to the new software delivery pipeline as fast as possible with the least amount of further guidance for developers.

The results of the piloting software delivery pipeline and proposed software delivery pipeline architecture were presented on 13th of February 2019 to Vaimo

community on a corporate “Tech Monthly” event by the author of this research. (see ch. 3)

The results of the research were presented on 10th of April 2019 to Vaimo community on a corporate “Tech Monthly” event by the author of this research. (see ch. 4)

5.2 Teams

Developing software delivery pipeline requires solving a significant number of business and technical problems.

Due to the fact that the initiative for development of the new software delivery pipeline came from only from one of the country units within Vaimo Group, most of the development was carried out on basis of Vaimo Finland, the country unit that initiated the process.

In order to create the new software delivery pipeline Vaimo Finland has gathered a team of professionals with the following roles: technical manager, software architect, QA engineer, data analyst and project manager.
As the core team was formed and the piloting phase of the project was initiated, a group of developers was introduced to the new software delivery pipeline to participate in its adaptation and work with it to test if it complies with acceptance criteria.

After publishing results of the research, the company aims at utilizing received knowledge in the further development of software delivery pipeline. One of the further steps to be taken is cooperation with service teams such as hosting/operations and platform teams.

In order to enhance the DevOps culture and bring automation to the software delivery pipeline, it is of outmost importance to create persistent communication channels between members of hosting, platform and core teams on the

following issues: setup of hosting environment, creation and further advancement of utilities, CI/CD automation, QA automation, migration of existing projects, documentation and automation of development tasks.

6 CONCLUSION

The current research work done in parallel with development as well as piloting of the software delivery pipeline can be described as successful.

Before all else, software delivery was described from the managerial, technical and cultural perspectives. Demands of the market for the implementation of a containerized solution as well as the DevOps culture were specified.

A proper landscape for research in software delivery using the survey methodology had to be set. In order to achieve that, implemented and proposed software delivery pipelines were briefly compared, the capabilities of the proposed CI/CD pipeline and the characteristics of the DevOps culture were highlighted.

This thesis aimed at evaluating the implemented software delivery pipeline and capabilities that could be part of a new software delivery pipeline based on containers. Professionals were surveyed in order to provide a statistically valid evaluation based on a large data set. The results of the survey revealed correlations between competence and implementation of a set of capabilities as well as overall feasibility of potential solutions. Such findings include an overwhelming preference for the use of a container-based software delivery pipeline with automatic builds and automatic deployments to non-production environments. The data analysis has also revealed professionals' preferences in the choice of version control systems, utilities and professional evaluation of capabilities of the currently implemented software delivery pipeline.

Moreover, teams involved in DevOps processes and the progress achieved in the implementation of the CI/CD solution within the span of the research were described. Agility is an important managerial approach for any software enterprise, therefore, continuity in software development was underlined by describing current progress and scheduled works.

Finally, the research was presented to Vaimo Group and can be considered done, being one of the stages of development of a software delivery pipeline based on containers and the DevOps culture. The results of the research allow to make statements on a set of technological capabilities that are most beneficial for a technical e-commerce solutions provider and to set the most suitable cultural landscape for the further development processes.

REFERENCES

1. Fowler, M. 2013. Continuous Delivery. Cited 25.01.2019, https://martinfowler.com/bliki/ContinuousDelivery.html
2. Fowler, M. 2006. Continuous Integration. Cited 25.01.2019, https://www.martinfowler.com/articles/continuousIntegration.html
3. Humble, J. & Farley, D. 2011. Continuous delivery: reliable software releases through build, test, and deployment automation. Boston: Addison-Wesley, 28.
4. European Commission. Cited 02.05.2019, https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-does-general-data-protection-regulation-gdpr-govern_en
5. Oxford Dictionary. Cited 10.02.2019.
6. Statista. Cited 21.02.2019, https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/
7. Grigoryeva, S. 2016. Agile Methodologies in Large-scale Software Projects, 76. Lapland University of Applied Sciences. Degree Programme in Information Technology. Bachelor's thesis.
8. Forsgren, N., Humble, J. & Kim, G. 2018. Accelerate:State of DevOps 2018: Strategies for an New Economy, 46.
9. Steven, J. 2018. What's the difference between Agile, CI/CD, and DevOps? Cited 11.02.2019, https://www.synopsys.com/blogs/software-security/agile-cicd-devops-glossary
10. Alonso, R. 2017. Software Containerization with Docker, 3-4. Turku University of Applied Sciences. Degree Programme in Information Technology. Bachelor's thesis.
11. Docker. Cited 17.03.2019, https://www.docker.com/resources/what-container

12. Geerling, J. 2014. NFS, rsync, and shared folder performance in Vagrant VMs. Cited 25.02.2019, https://www.jeffgeerling.com/blogs/jeff-geerling/nfs-rsync-and-shared-folder
13. Shiala, G., Majhib, S. K. & Phatakc, D. B. 2015. A Comparison Study for File Synchronization. Procedia Computer Science 48, 133 – 141.
14. What's a Linux Container. Cited 18.03.2019, https://www.redhat.com/en/topics/containers/whats-a-linux-container
15. Leszko, R. 2017. Continuous Delivery with Docker and Jenkins. Birmingham: Packt Publishing.
16. Forsgren, N., Humble, J. & Kim, G. 2018. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. Portland: IT Revolution Press.
17. Rossinsky, E. 2017. Почему ivi перешел со Sphinx на Elasticsearch / Евгений Россинский (ivi). Date of Retrieval 20.03.2019. https://www.youtube.com/watch?v=y5OJSIC5yE8
18. Pelleg, D. 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. San Francisco: Morgan Kaufmann Publishers Inc.
19. Herve, A. 2007. Z-scores. Cited 07.05.2019, https://www.utdallas.edu/~herve/Abdi-Zscore2007-pretty.pdf
20. Galarnyk, M. 2018. Understanding Boxplots. Cited 07.05.2019, https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd5

SURVEY PROPOSAL

APPENDIX 1

Survey Proposal: "**CI/CD Vaimo Developers Survey**"

Current State:
1. Vaimo Finland has made a **strategic decision** to dedicate time and resources in order to improve software delivery performance, according to the practices presented in research publication "The State of DevOps Report" and book "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations".

2. Vaimo Finland has made a **technical decision** that is designed to pilot an improved way software quality assurance is provided to the customer, how software is delivered and served, if the VCS of preference can be Git, due to its technical capacities as well as spread of use.

3. There are a number of business and technical decisions that cannot be effectively made due to the **lack of data**. These are as follows:

- Should Vaimo provide more freedom to developers in a choice of a version control system on per-case basis?
- Should Vaimo provide better utilities to reduce stress and unpredictability caused by CI/CD procedures?
- Should Vaimo provide better QA automation solutions, so as to improve quality of a product and reduce cycle time?
- Should Vaimo provide better CI/CD visualization and failure prevention capabilities, so as to make product less crash-prone and reduce amount of time spent on CI/CD operations?
- Should Vaimo provide more education to employees in CI/CD practical knowledge?
- Should Vaimo provide more education to employees in CI/CD theoretical knowledge?
- How much time/money/investment (in abstract units) Vaimo has to provide in order to successfully migrate to the software delivery capabilities described in the papers above and pursued as of now?

Currently the task is to graphically represent the data gathered, so as to visually detect weaknesses in quality of software delivered, software delivery performance as well as employee satisfaction/self-realization.

Note: number of questions addressed by the survey is not limited to questions above, however these provide a decent representation of questions that can be answered by the survey designed.

4. Bachelor's Thesis "**DevOps in E-commerce software development: Demand for Containerization**" is written by a developer in Vaimo Finland and survey described is part of the thesis being written. Vaimo Finland is set to survey Vaimo employees in order to investigate current state of QA procedures, value of utilities used for server as well as CI/CD status visualization, theoretical and practical knowledge of employees in CI/CD, stress level caused by CI/CD, time spent of correcting failures caused by CI/CD operations as perceived by developers and determine tool preferences for the internal use, so as to make business and technical decisions that bring most value to the business.

SURVEY STRUCTURE APPENDIX 2

1. Position at Vaimo
2. Office
3. Country Unit
4. How knowledgeable are you in current day to day CI/CD operations using Aja and Jenkins?
5. How knowledgeable are you in day to day CI/CD operations with solution being deployed to a cloud using Docker/Kubernetes?
6. How well do you know inner workings of CI/CD pipeline currently implemented using Aja and Jenkins?
7. How well do you know inner workings of container-based CI/CD pipeline based on Docker and Kubernetes?
8. How experienced are you working with Git VCS?
9. How experienced are you working with Mercurial VCS?
10. How much time do you spend on CI/CD operations?
11. How much development time do you spend on failures caused by CI/CD processes?
12. Which CI/CD operations are currently time-inefficient? Why?
13. How much influence do current CI/CD practices have on stress level at work?
14. What causes most stress/unease in current CI/CD operations?
15. How effective is current CI/CD and server status visualisation?
16. Which CI/CD or server visualisation solution is lacking?
17. How effective are current QA procedures?
18. What QA procedures are currently most ineffective?
19. How preferable is it to have build process initiated automatically after commit?
20. How preferable is it to have deployment process initiated automatically after the build for production environment?
21. How preferable is it to have deployment process initiated automatically after the build for non-production environment?
22. How preferable are utilities (high-level tools, e.g. Aja/IEC) over native commands (low-level tools, e.g. composer) in CI/CD?

23. How preferable is it to allow developers to pick VCS of their choice for a new project?
24. How preferable is it to allow developers to pick CI/CD tools of their choice for a new project?
25. How preferable is it for development team to take operational responsibility?
26. How preferable it is to use Mercurial as a version control system?
27. How preferable it is to use Git as a version control system?
28. How preferable is it to have solutions deployed to cloud?
29. How preferable is it to have solutions deployed to traditional hosting?
30. How preferable is it to have solutions deployed on containers running on server/cloud?
31. How preferable is it to have solutions deployed directly onto a VM running on server/cloud?
32. Proposals for CI/CD pipeline and DevOps practices
33. Survey Feedback

Note:

- Answer to underlined questions was not required
- Questions in green were expected to receive a long answer
- Questions in yellow were expected to receive a short answer
- Questions without highlighting were expected to receive answer from 0 to 10
- Questions in light-blue highlighting were expected to be answered by selecting an option from dropdown.

DevOps on Docker and Kubernetes

Created by Roman Zakharenkov, last modified on Mar 07, 2019

Description

These set of pages aims at documenting new approach to software delivery, which is based on changing development environment and CI/CD framework in order to improve delivery speed, have direct access to servers (cloud), increase stability and reliability.

In a Nutshell:

Local Environment:

~~Devbox (Vagrant)~~ → **Container (Docker)**
P.S. Forget about Devbox and Vagrant, Aja tools are not any more available, iec tools still can be used.

Production/Development:

~~Mercurial + Jenkins + VPS's~~ → **Git + Kubernetes/Argo + Containers**
P.S. Mercurial and Jenkins still can be used for working with your projects dependencies.

Contents

Support:
@Paavo Pokkinen @Saana Tauriainen @Roman Zakharenkov
Documentation:
@Roman Zakharenkov