Agile Development and Operation of Complex Systems in Multi-technology and Multi-company Environments: Following a DevOps Approach

Gorka Benguria(✉), Juncal Alonso(✉), Iñaki Etxaniz(✉), Leire Orue-Echevarria(✉), and Marisa Escalante(✉)

TECNALIA, Parque Tecnológico de Bizkaia Ed: 700, 48160 Zamudio, Spain
{Gorka.Benguria, Juncal.Alonso, Inaki.Etxaniz, Leire.Orue-Echevarria, Marisa.Escalante}@tecnalia.com

Abstract. Big innovation and research projects usually require merging contributions form organizations with expertise in different domains. Managing and participating in multi-company projects that use multiple state-of-the-art technologies constitute a challenging activity due to many factors such as integration inexperience, evolving components, tentative requirements, independent teams or independent management centers. In the late nineties and in the early years of 2000, several methodologies arose with the focus on fast releases of working software, commonly known as agile, that aimed to address many of the challenges that this kind of projects face. However, in most cases, these methodologies were not fully adoptable as the automation investment was too high and it was not recoverable during the duration of the project. The global servitization trend and the appearance of approaches, such as DevOps, to support the continuous and fast adjustment of those services to stay in business has also impacted innovation and research projects. On one hand, matured technologies that reduce the automation investment have arisen. On the other hand, whenever it makes sense, services which benefit from the application of DevOps approaches are required to be implemented. This paper explains the implementation of DevOps approaches to support the agile development in the context of innovation and research projects. It also describes two practical implementation cases where such approaches were implemented and how they evolved in the course of the time.

Keywords: DevOps · Continuous integration · Continuous delivery
Agile

1 Introduction

Globalization and the increase of competition is pushing organisations to deliver new products and features at increasing rate to stay attractive for their customers base. In the software domain, this implies to be able to faster and more frequent release features to the final users. To increase the releasing capability, software organisations have been forced to adopt DevOps [1, 2] approaches to reduce the time required to develop, integrate, test, and deploy. The DevOps philosophy attempts to use automated systems to bridge the information gap between project team entities (Development and Operations teams) with the main objective of fostering their collaboration in the development and operation stages of a software application [3]. The DevOps philosophy incorporates some principles, methods and practices related with the agile methodologies [4] such as continuous delivery, continuous integration and collaboration.

Agile methodologies [5, 6] have afforded many companies a much-needed tool in the digital era: a collaborative and flexible way to develop software. Considering Agile manifesto [7] as compendium of principles, it can be concluded that they represent an attitude towards change that is common for both Agile and DevOps. In essence, DevOps is Agile applied beyond the software team. A DevOps approach takes agile a step further and applies it beyond the plan, design, build, and test stages of software development to the rest of the software lifecycle: deployment, release, operation, and monitoring [8].

Innovation and research projects usually require involving different organizations with expertise in different areas. The number of organizations is variable depending on the scope and domain of the project (for example, fi-ware project [9] was participated by close to 90 companies), but usually the number varies between five and ten. These organizations work together during years towards a common objective. During that time, a subset of these organizations can be required to work together to produce and validate a software solution that solves a set of needs. This kind of collaboration has some challenges to consider:

- Non-collocated Information Technology (IT) teams from different organizations with different technologies, cultures, languages, working calendars, priorities, quality standards, etc.
- Disperse development effort to be leveraged with research, reporting, dissemination, and other parallel production-oriented activities.
- Tentative requirements, integration inexperience and evolving components.
- Solutions or demonstrations should be repeatable long after the project's end based on commercial or audit requests.

With the global servitization trend, innovation and research projects are required to implement services with similar challenges that match most of the situations given also in projects with distributed teams. This increase the development and operation complexity as it introduces additional requirements to the applications such as servers, domains, networks, platforms, authorization, authentication, certificates, and so on. Fortunately, more and more features of those services are being packaged as easily reusable open source assets, which enable an easier and faster management of that complexity.

Nowadays, provided that the required resources are available, it is feasible and affordable to deploy a service together with its corresponding platform in few minutes. However, a few years ago, this was simply cumbersome.

The current paper presents a DevOps approach implemented in Tecnalia to enable the agile development of applications, especially service oriented ones, in multi-technology and multi-company environments.

2 A DevOps Framework for Agile Development

Agile methodologies include characteristics that make them interesting for innovation and research projects. For example, one of the principles of the agile methodologies is to “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale” [7]. This may help in the clarification of tentative requirements and integration approach. Besides, it may also help with the evolution of the components.

But on the other hand, agile methodologies have certain principles that make them incompatible with this kind of projects. For example, the principles related to “Business people and developers must work together daily throughout the project” and “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation” are totally incompatible with the facts that the teams are not collocated and that the effort should be leveraged with other activities.

Therefore, more than establishing a DevOps Framework to enable an agile methodology at project level, our objective was to establish a DevOps Framework that enable the organizations participating in the project to be agile in their own terms.

2.1 DevOps Framework Approach

In the same way that there is no standard definition of DevOps, there is no agreement about what a DevOps framework should consist of. For us, a DevOps framework is a set of tools that facilitate the transition and interactions between the application development and its exploitation.

The DevOps framework that we are currently using in different projects such as OPERANDO [10], CITADEL [11] or DECIDE [12] was not designed as a holistic solution to be applied in all projects of Tecnalia. Indeed, this approach started in OPERANDO based on the lessons learned from a previous project, MOVEUS [13].

MOVEUS [13] was a project in the smart cities domain that provided a set of mobility services that encouraged sustainable transport. In that project, Tecnalia was the partner responsible for the infrastructure. During the development, more than thirty servers were used, some were virtual, some were physical. Servers were split between two environments, integration and piloting. Integration was supported with an on-premise Open Stack cluster for cost constraint requirements, while piloting was supported in AWS [2] for reliability and performance requirements. From that project there were several lessons learned:

- Infrastructure migration can happen, and it will happen several times
- Following the installation guidelines is time consuming, and finding problems is very common
- The domain strategy is important as the number of servers and environments grow
- An improper use of cloud services may generate many expenses
- Non-root users are configured by default with contracting capabilities by AWS
- The virtual machines created using the AWS images cannot be exported later on
- The storage of the application and the demonstrations is an issue in the long term
- Integration infrastructure also requires some reliability

So, when the project OPERANDO started in which Tecnalia was also in charge of the infrastructure, it was decided to change the approach to be able to deal with some of the abovementioned problems. Our principles were as presented next.

The first set of requirements for our framework were to use reliable platforms in all environments, to control the expenses on the selected cloud environments, to support the replication of the platform, and to facilitate the deployment of new features. From those requirements, the first version that included a subset of the components of OPERANDO was developed. From that version and through a very simple process (Fig. 1), the platform continued to evolve.

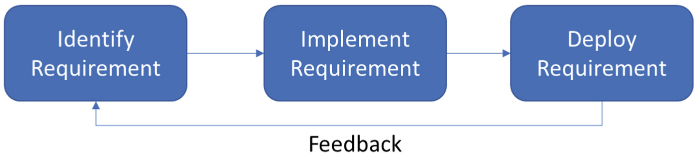


Fig. 1. DevOps framework approach

Most of the changes that had to be performed in a later stage consisted of minor adjustments. Nevertheless, certain changes involved the introduction of new requirements such as adding monitoring capabilities over the framework, supporting guidelines over the platform (with the knowledge necessary to adjust them), supporting pilots or making the infrastructure instantiable to other projects.

2.2 DevOps Framework Configuration Process

The DevOps Framework is adapted to each project following the generic process described in the Fig. 2.

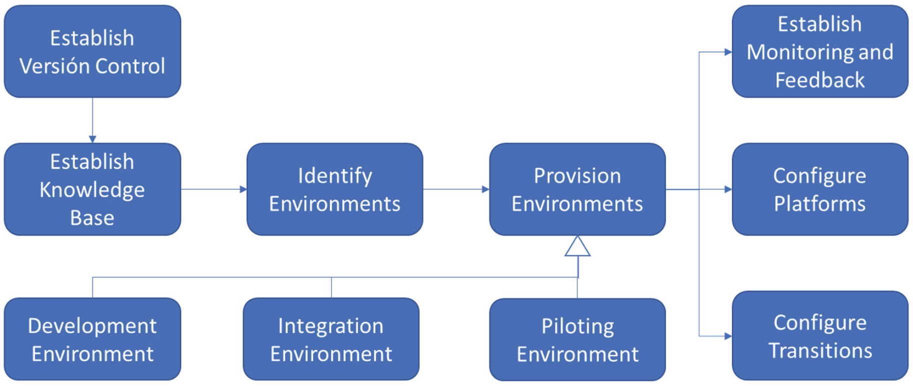


Fig. 2. DevOps framework configuration process

All these activities can be grouped into four categories: Knowledge Management, Environments establishment, Transitions configuration and Feedback Management.

2.3 Knowledge Management

The first set of activities are oriented to ensure that the knowledge of the project is reliably stored and provided in the best way possible to all participants: developers, pilot owners, managers or external contributors.

The main requirement pursued by the knowledge management is to support the replication of the platform. This support should enable, even external people, to replicate the platform from scratch.

The Knowledge management in our approach is made differently depending on the project assets being managed. Application development projects make use of different assets which are subject of storage in the long term. Among others, the following can be cited:

- Code, including configuration, formal specifications, scripts, …
- Sensible information such as access passwords, private certificates, and licenses keys.
- Documents such as guidelines or reports, etc.
- Data for testing, such as anonymized data from pilots.
- External compiled libraries.
- Images of virtual machines and containers.
- Releases.

The Knowledge management in our DevOps framework include two mandatory components: the version control system and the knowledge storage.

The version control system is supported by Git [14]. It is of course possible to use other technologies such as subversion [15] or cvs [16]. However, git was selected because nowadays it is the one that is most widely used (about 87% as demonstrated in the last Stack Overflow Survey [17]). Depending on the project, different Git implementations such as Github [18], BitBucket [19] or Gitlab [20] can be used. Depending on the implementation between external SaaS (Software as a service) [21] or on-premise installation can also be chosen.

The first component in our DevOps framework, the version control system, is used as the communication channel from development to operation. Developers store the code along with the necessary resources for their components in the version control system and the operators retrieve them from there. In addition to that, the version control system is also used to store the code and scripts necessary to quickly build the different environments required to develop and deploy the application.

The second component in our DevOps framework, the knowledge storage, is focused on collecting the guidelines to help project participants during their development and deployment activities. The knowledge storage aims to contain information about how the technologies have been used and configured in order to support all aspects of the application. As a guidelines repository, we require a central content management system with strong search capabilities. With that thought in mind, a WordPress CMS [22] linked to a Solr [23] indexing system to enhance the WordPress searching mechanisms was implemented.

These two mechanisms cover most, but not all, of the assets of an application development project. The remain assets are stored locally. These include sensitive information, big binary data sets for testing, external compiled libraries, images (of virtual machines and containers). However, depending on their relevance to the project additional mechanisms such as keepass [24] for sensible information, or archiva [25] for external libraries, could be established.

2.4 Establishment of Environments

The establishment of environment is another set of activities that needs to be covered. This involves the identification of environments, the configuration of environments and the configuration of platforms.

Development and operation constitute two different environments, but depending on the characteristics of the application, it can make sense to introduce additional environments to ensure aspects such as the interoperability, performance or the behavior in a real environment. Figure 3 shows an example of environments between development and production.

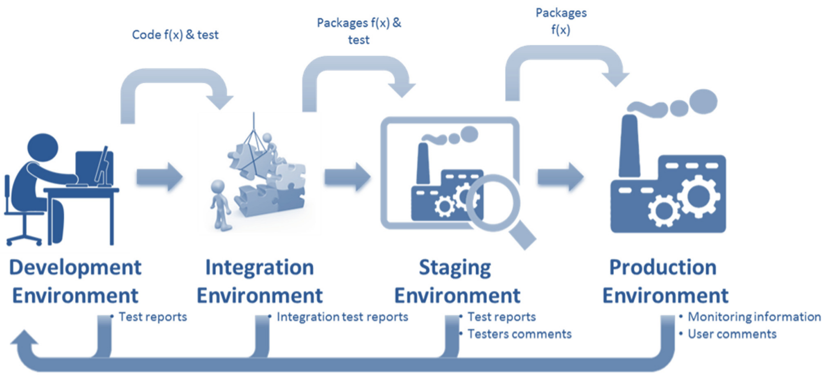


Fig. 3. Environments in an application development

In Innovation and research projects the environments shown above vary slightly. In this case, instead of having staging and production environments, these kinds of projects present a (or several) piloting environment. The current DevOps Framework deployed in Tecnalia provides support in the provisioning of the development, integration and piloting environments.

The support for the development environment is focused on two aspects: speed in setting it up and standardization. The first aspect, speed in setting it up aims to reduce the time required to configure a machine to start with the development. Depending on the complexity, a development environment may require multiple tools, each with different installation and activation procedures. Installing, activating, and configuring them, one by one could take days. Whenever possible, portable and activation independent technologies are prioritized. The objective is to create a folder that contains all the tools required to start working in the project properly configured. This has reduced the development environment setup to less than one hour. Standardization aims to share exactly the same development environment among all the developers with the goal of avoiding the "it works on my machine" syndrome. The portable approach, where we even include the java version, largely reduced this effect among the development environments. Technologies that have been packaged in the development environment for windows 64 bits include eclipse [26], mysql [27], git [18], squirrel [28], nodejs [29], Vagrant [30], Cygwin [31], Maven [32], gradle [33] or notepad ++ [34]. The only installation requirement is VirtualBox [35] but only in case Vagrant is needed.

The support for the integration environment is focused on the assurance that all components can be built and work seamlessly together from scratch. The integration environment is configured in a way that every 24 h, it is reset and all the components built again taking the last version stored in the master branch. After building the components, the environment is able to start all of them automatically and run some integration tests to ensure that they interoperate as expected. For the integration environment, we make use of the Docker [36] technology. Docker is a container technology that allows to simulate multiple independent operating systems over the same machine. This reduces heavily the infrastructure requirements in the integration environments. For instance, in the project MOVEUS previously referred, more than ten servers in AWS were used. This could have been reduced notably if Docker had been used. As a comparison, in OPERANDO only one server with 25 containers is used.

Finally, we have the piloting environment. The support in this environment is focused on the assistance to external companies in the instantiation of the components of the project in their own premises. In previous projects, the transference of the technical developments required getting the compiled libraries somehow and make them work in some preconfigured platform. This was highly error prone. As part of our DevOps Framework we provide two elements: a Docker registry from which the companies can download the last version of the components of the project packaged as Docker images, and a script that configures and runs those Docker images in a Docker server.

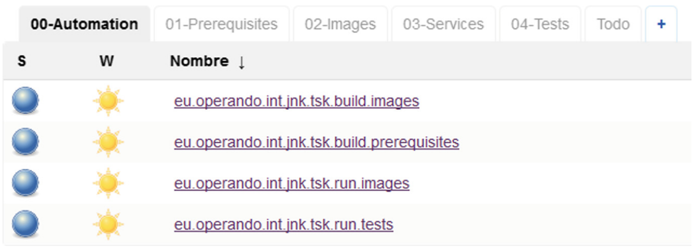
Another activity performed in relation to the establishment of the environments has been the configuration of platforms. In innovation and research projects involving multiple companies, components are often programmed with different technologies that require a diverse terrain of platforms. The DevOps framework of Tecnalia includes an increasing set of preconfigured elements to be integrated in different environments, and platforms such as tomcat, mono, nodejs, python [37], mysql, mongodb [38], Hadoop [39], cas [40], ldap [41], liferay [42], jhipster [43], Jenkins [44], …

2.5 Transitions Configuration

Transitions Configuration support the automation of the deployment of new features from the development environment to the operation one. In our DevOps Framework we provide support for the transition between development, integration and piloting.

To support these transitions, we use a continuous integration software and build automation tools. Specifically, as continuous integration software, we use Jenkins and as a build automation tool, we use Maven.

Nevertheless, the focus in our case is on the transition between the development and the integration environments. As shown in Fig. 4, Jenkins includes tasks for the provisioning of pre-requisites to compile the components, tasks to package the components (services and integration tests) as Docker images, tasks to run the services, and tasks to perform the integration tests over the running services.



S	W	Nombre ↓
●	☀	eu.operando.int.jnk.tsk.build.images
●	☀	eu.operando.int.jnk.tsk.build.prerequisites
●	☀	eu.operando.int.jnk.tsk.run.images
●	☀	eu.operando.int.jnk.tsk.run.tests

Fig. 4. Jenkins tasks categories

Each of these Jenkins tasks is specified in a *Jenkinsfile* stored in the version control system (in our case, git). The task is split in stages as shown in Fig. 5 for a task on service package. Among those stages, we can see stages in charge of gathering the source code provided by the developers, stages in charge of gathering the DevOps source code that describe how to build the necessary infrastructure for the component, stages to compile the code, stages to create the Docker image with the platform and with the component, and stages in charge of pushing the created Docker image into the Docker registry for their later use by the pilots.

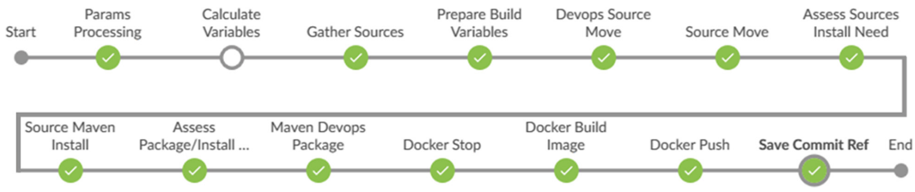


Fig. 5. Jenkins tasks stages

The usage of Maven, apart from building java components, is to fill some gaps or difficulties that we faced with Jenkins. In particular, we use Maven to enable the use of Docker in Jenkins. To save time in the deployment of the DevOps Framework, we decided to use Jenkins packaged in Docker, which is as easy to deploy as:

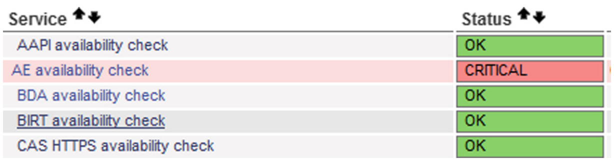
```
Docker run -p 8080:8080 Jenkins
```

Since we found the installation of a Docker inside so complex, we decided to use the Maven plugin for Docker from frabric4io [45]. Moreover, it facilitated the usage of common properties when instantiating the Docker components such as service end-points. We aggregated all of them in a common Maven project that is referenced in each of the service and integration test running tasks.

2.6 Management of Feedback

The final set of activities are those related with the management of feedback. This covers the collection of information from the different environments. Currently, the support for the management of feedback in our DevOps Framework is limited. Two mechanisms have been implemented: Automatic monitoring mechanisms and a ticket based system to be able collect feedback from testing activities.

Automatic monitoring mechanisms are based on state of practice of monitoring platforms, that collect information about the services running at the integration platform. Figure 6 shows how Nagios is used to monitor the status of the services of the OPERANDO project in the integration platform.



Service	Status
AAPI availability check	OK
AE availability check	CRITICAL
BDA availability check	OK
BIRT availability check	OK
CAS HTTPS availability check	OK

Fig. 6. Nagios monitoring of integration platform

For the ticket based system, different technologies depending on the project and the preferences of the participants are being used. Among others, Jira and GitLab issues.

3 Practical Implementation of DevOps Philosophy in Multi-technology Environments

The DevOps Framework introduced previously has been applied in different projects during the last two years. The DevOps Framework applied in each project present differences taking into consideration the continuous improvement of the platform.

3.1 OPERANDO Ecosystem

OPERANDO project aims to implement and validate an innovative privacy enforcement framework that will enable the Privacy as a Service (PaS) business paradigm and the market for online privacy services.

OPERANDO project is being carried out by nine companies from seven different countries. The architecture requires to leverage technologies such as Back end REST technologies (Jaxrs/Python Flask/Spring), single sign on mechanisms (cas), directory services for access control (ldap), relational databases (Mysql), non-relational databases (e.g. MongoDB), front end technologies (Html 5/css/Javascript/.Net), Reporting frameworks (Birt [46]) or Monitoring Frameworks (Nagios [47]).

The OPERANDO Ecosystem was the first project in which the DevOps Framework was applied. The project duration has been three years and is almost at its closure. We have been applying and adjusting the DevOps Framework for a timeframe of two and a half years. During this period, a lot of feedback and improvement suggestions about the DevOps Framework that has led to some fine-tuning has been acquired. This includes:

- The developer project should not be bothered with DevOps details as DevOps details should be contained in a separate project. i.e.: how the code is packaged in a Docker container.
- During the project, we were required to migrate the integration environment twice. The first time, the migration time needed was two weeks which was acceptable when comparing to MOVEUS, that took several months. In any case, lessons learned were gathered and the migration process improved. The second migration took just a couple of hours.
- Developers require feedback, therefore, we implemented a new profile of observer users to enable them to check the results of their deployment in the integration platform. However, that was not enough as they also required logs access which was later on integrated.
- Developers want to test things quickly. They require a certain degree of control over the integration process to enable them to trigger the compilation process at any time. They also require to be able to take a look inside the component. This is a more complex issue that is currently under analysis.

3.2 CITADEL Ecosystem

CITADEL project is focused on transforming Public Administrations (PAs) to deliver more efficient, inclusive and citizen-centric public services. For this, the project captures data to understand how satisfied citizens are with digital public services, identifies new or unsatisfied needs and allows citizens to take active part in the creation of new public services to increase their uptake.

CITADEL project is composed of eleven organizations from four countries. The architecture requires to leverage technologies such as user registries for citizens, single sign on mechanisms (OpenId Connect), directory services for access control (ldap), front-end technologies (Html 5/css/Javascript), Back-end REST technologies [48] (Jaxrs), databases (Mysql), and Intranet Portals (Liferay) in an integrated set of services, named ecosystem in the jargon of the project.

The duration of the CITADEL project is also three years. We have been applying the DevOps Framework during almost one year. During that period, we have learned some additional lessons that will drive future improvements. These include:

- “it works on my machine” Syndrome also happens between environments; therefore, it could be interesting to bring the real (or almost real) infrastructure to the development environment. We are exploring the usage of Vagrant to enable the instantiation of the integration containers in the developers’ machines. This may require powerful development equipment.

- The continuous creation and destruction of Docker images and container leave behind garbage that should be automatically cleared out. Thus, we need garbage collection mechanisms for Docker.
- Development with http to avoid the creation of certificates is bringing more problems than benefits. Therefore, we need to consider certificates when configuring platforms and at the same time, we need to support its creation.
- As we increase the number of independent DevOps Frameworks, we are more in need of a homogeneous and integrable ways to monitor them. For this we are exploring monitoring alternatives such as Grafana [49] that allow to easily define dashboards for each of the frameworks, environments and services.

4 Conclusions and Future Work

The current paper documents practical implementations of the DevOps approach in diverse software-based environments, where multi-disciplinary distributed teams are co-developing and co-deploying software assets (often using immature technologies) to achieve a medium TRL [50] solution (usually TRL 5-6). The peculiarities of these collaborative developments imply several challenges and shortcomings that can be addressed setting up a proper DevOps approach with the corresponding enabling technologies and tools.

The solution proposed by the authors covers the "traditional" DevOps cycle, focusing mainly on the continuous integration, and continuous deployment phases of the Software Development Life Cycle (SDLC). Nevertheless, the advent of cloud-computing offerings, which is witnessing a growing trend by the increasing diversity of cloud service offerings leading to hybrid and multi-cloud infrastructures, are available for complex software applications which can profit from these variaty of offers and features. This new situation opens up a world of possibilities for increasing the efficiency of software applications by selecting a combination of the most appropriate resources on-demand and re-adapting the application on-the-fly to meet the working conditions of the software. The presented DevOps approach, can be therefore, adapted to support this upcoming scenario, where "multi-cloud" based applications can be (re-) adapted, considering their Non-Functional Requirements. This can be achieved trough the extension of the *"traditional"* DevOps cycle in what the authors call *"Extended DevOps"*. The authors will extend the concepts and approach presented in this paper in the context of the DECIDE [12] project which proposes an extension of the "traditional" DevOps approach on the two axis: Dev and Ops.

For the Dev axis, the authors propose to extend the development phase with previous and subsequent activities that cover: (1) the architectural design for multi-cloud applications (at generic level and specific to non-functional requirements), (2) the election of the best deployment configuration based on theoretical simulations. For the Ops axis, the extension will cover: (3) The orchestration of heterogenous cloud resources (4) The continuous re-deployment of software components to maintain the working conditions.

Acknowledgements. The projects leading to this paper have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 731533, 726755 and 653704.

References

1. DevOps, Wikipedia, 18 April 2018
2. What is DevOps? - Amazon Web Services (AWS), Amazon Web Services, Inc. https://aws.amazon.com/devops/what-is-devops/. Accessed 23 Apr 2018
3. "DevOps" an Extension of Agile Methodology – How It will Impact QA? AFourTech : Software Development Company | Software Testing Services, 25 Apr 2014
4. Navigating DevOps - What it is and why it matters to you and your business (2017)
5. Sutherland, J., Schwaber, K.: The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework, p. 224, April 2012
6. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional (2000)
7. Beck, K., et al.: Agile Manifesto (2001)
8. DevOps Takes Agile Further - What is DevOps? https://www.bcg.com. https://www.bcg.com/agile/devops/default.aspx. Accessed 19 Apr 2018
9. FI-WARE: Future Internet Core Platform: Project and results. https://cordis.europa.eu/project/rcn/99929_en.html
10. OPERANDO CONSORTIUM. https://www.operando.eu/servizi/notizie/notizie_homepage.aspx. Accessed 19 Apr 2018
11. CITADEL H2020 (2016)
12. Home | DECIDE: Multicloud Applications Towards the Digital Single Market. https://www.decide-h2020.eu/. Accessed 16 Apr 2018
13. MoveUS | ICT cloud-based platform and mobility services available, universal and safe for all users | MoveUS. http://www.moveus-project.eu/. Accessed 19 Apr 2018
14. Git. https://git-scm.com/. Accessed 20 Apr 2018
15. Apache Subversion. https://subversion.apache.org/. Accessed 20 Apr 2018
16. CVS - Open Source Version Control. http://www.nongnu.org/cvs/. Accessed 20 Apr 2018
17. Stack Overflow Developer Survey 2018, Stack Overflow. https://stackoverflow.com/insights/survey/2018/?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2018&utm_content=social-share. Accessed 20 Apr 2018
18. Build software better, together, GitHub. https://github.com. Accessed 20 Apr 2018
19. Bitbucket | The Git solution for professional teams. https://bitbucket.org/. Accessed 20 Apr 2018
20. Gitlab. https://about.gitlab.com/
21. Software as a service, Wikipedia, 10 Apr 2018
22. WordPress. https://en.wordpress.com
23. Apache Solr. http://lucene.apache.org/solr/. Accessed 20 Apr 2018
24. Reichl, D.: KeePass Password Safe. https://keepass.info/. Accessed 20 Apr 2018
25. Archiva – The Build Artifact Repository Manager. https://archiva.apache.org/index.cgi. Accessed 20 Apr 2018
26. Open Innovation Community - Eclipse IDE | The Eclipse Foundation. http://www.eclipse.org/. Accessed 23 Apr 2018
27. MySQL. https://www.mysql.com/. Accessed 23 Apr 2018
28. SQuirreL SQL Client Home Page. http://squirrel-sql.sourceforge.net/. Accessed 23 Apr 2018

29. Node.js Foundation: Node.js, Node.js. https://nodejs.org/en/. Accessed 23 Apr 2018
30. Vagrant by HashiCorp, Vagrant by HashiCorp. https://www.Vagrantup.com/index.html. Accessed 13 Apr 2018
31. Cygwin. http://www.cygwin.com/. Accessed 23 Apr 2018
32. Maven – Welcome to Apache Maven. https://Maven.apache.org/. Accessed 13 Apr 2018
33. Gradle Build Tool. https://gradle.org/. Accessed 23 Apr 2018
34. Notepad++ Home. https://notepad-plus-plus.org/. Accessed 23 Apr 2018
35. Oracle VM VirtualBox. https://www.virtualbox.org/. Accessed 23 Apr 2018
36. Docker, Docker. https://www.Docker.com/. Accessed 11 Apr 2018
37. Welcome to Python.org, Python.org. https://www.python.org/. Accessed 23 Apr 2018
38. MongoDB for GIANT Ideas | MongoDB. https://www.mongodb.com/. Accessed 23 Apr 2018
39. Welcome to ApacheTM Hadoop®! http://hadoop.apache.org/. Accessed 23 Apr 2018
40. CAS | Apereo. https://www.apereo.org/projects/cas. Accessed 11 Apr 2018
41. Lightweight Directory Access Protocol, Wikipedia, 02 April 2018
42. Liferay: Software de experiencia digital adaptado a tus necesidades. https://www.liferay.com/. Accessed 23 Apr 2018
43. JHipster - Generate your Spring Boot + Angular/React applications! https://www.jhipster.tech/. Accessed 23 Apr 2018
44. Jenkins, Jenkins. https://Jenkins.io/index.html
45. Docker-Maven-plugin: Maven plugin for running and creating Docker images. fabric8 (2018)
46. BIRT Home. http://www.eclipse.org/birt/. Accessed 23 Apr 2018
47. Nagios - The Industry Standard In IT Infrastructure Monitoring. https://www.nagios.org/. Accessed 13 Apr 2018
48. Representational state transfer, Wikipedia, 13 April 2018
49. Grafana - The open platform for analytics and monitoring. https://grafana.com/. Accessed 13 Apr 2018
50. Technology readiness level, Wikipedia, 04 April 2018