DevOps Final Report

By: Bhrigu Mahajan - Alvaro Bilbao - Samer Masaad

04/02/2019

# Table of content

[**Table of content**](#_udewbaskkfs) **1**

[**Introduction**](#_scargzgewzu0) **2**

[**How DevOps Works**](#_a7jypbt3ihqq) **3**

[**Why DevOps Matters**](#_njw0sw1cxlhz) **4**

[**DevOps responsibilities**](#_y723iy2p787e) **5**

[**DevOps Practices**](#_30xwuyls6eqi) **6**

[Continuous Integration](#_z5p7nlegkhfq) 6

[Continuous Delivery](#_q2pzw336adqr) 6

[Microservices](#_c1ud1ib2njt3) 6

[Infrastructure as Code](#_8zpzwicjb7xa) 6

[Configuration Management](#_77vj3mrobyle) 6

[Policy as Code](#_e9syfgbsej6c) 6

[Monitoring and Logging](#_qnkcq9i90bvx) 7

[Communication and Collaboration](#_xmidr2q44th3) 7

[**Benefits of DevOps**](#_mjexdw2r8d5h) **7**

[Speed](#_bx4v08ea2h24) 7

[Rapid Delivery](#_17fgvypsaipv) 7

[Reliability](#_d1o59pav8snq) 8

[Scale](#_atocrhvqnp9j) 8

[Improved Collaboration](#_91g6ywf5zd32) 8

[Security](#_gmpw49cxkare) 8

[**Docker**](#_qq1jv9fdkuxf) **9**

**Software Factory 9**

[**JVM and JBoss**](#_vcs5v5pq6i38) **11**

[**MySQL**](#_2scv7v7kdm7l) **11**

[**Conclusion - Interactions between the components**](#_e7q94rzb2dzl) **1**2

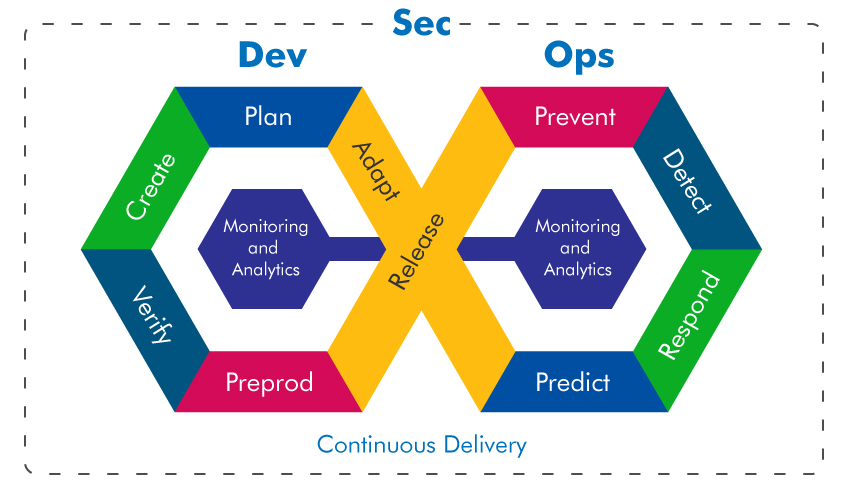
# Introduction

DevOps is a methodology to create a new mindset, when developers and IT operations combine their efforts to achieve a common goal.It is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

DevOps culture is aimed at maximum efficiency of delivering value to the customers. This means eliminating all kinds of waste, as well as ensuring optimal performance and security of operations. In terms of software delivery, automated unit tests ensure the thorough code testing to minimize the risk of code bugs and backdoor access to the apps.Automation of daily routine tasks.

****

# How DevOps Works

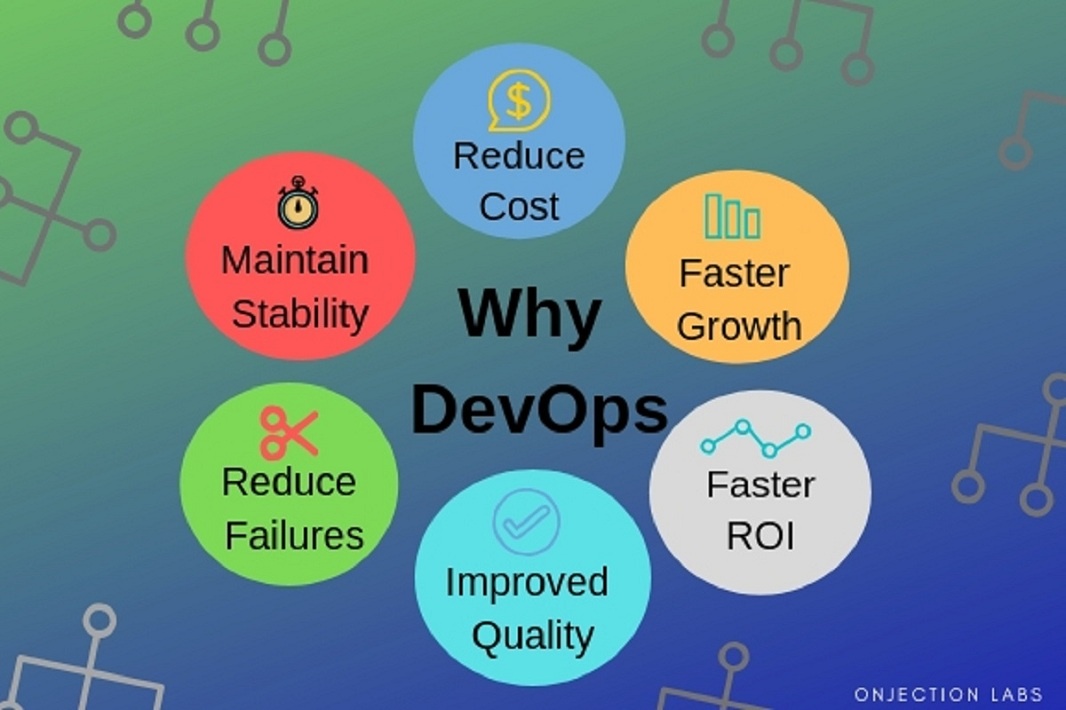
****

Under a DevOps model, development and operations teams are no longer “siloed.” Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

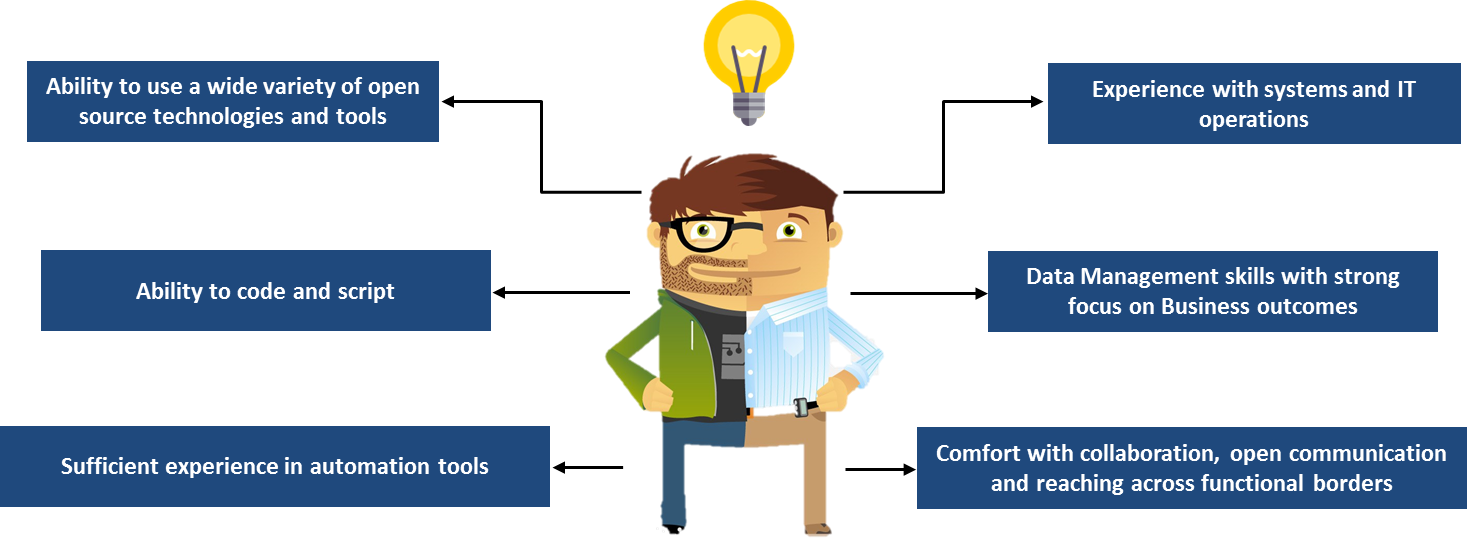
These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team’s velocity.

# Why DevOps Matters

****

Software no longer merely supports a business; rather it becomes an integral component of every part of a business. Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today’s world must transform how they build and deliver software.

# DevOps responsibilities

****

* DevOps automate processes so that developers can more easily deploy code. Once the DevOps Engineer is done building the process, it’s not uncommon for deployment to take only one click.
* DevOps can create a new environment for your project from scratch. This includes handling server architecture and storage services, as well as setting up server hardware and software.
* A DevOps Engineer can also help you maintain and optimize your existing environment(s).
* DevOps can help you decide on the best server tech stack to handle your code, taking into account scalability, security, and maintainability.
* DevOps Engineer will also keep an eye on app and infrastructure statistics to identify insights and potential problems ahead of time.

# DevOps Practices

## Continuous Integration

Developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

## Continuous Delivery

Code changes are automatically built, tested, and prepared for a release to production.When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

## Microservices

The microservices architecture is a design approach to build a single application as a set of small services.Different frameworks or programming languages to write microservices and deploy them independently, as a single service, or as a group of services.

## Infrastructure as Code

Infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration.The cloud’s API-driven model enables developers and system administrators to interact with infrastructure programmatically.

### Configuration Management

Developers and system administrators use code to automate operating system and host configuration, operational tasks, and more. It frees developers and systems administrators from manually configuring operating systems, system applications, or server software.

### Policy as Code

With infrastructure and its configuration codified with the cloud, organizations can monitor and enforce compliance dynamically and at scale.Infrastructure that is described by code can thus be tracked, validated, and reconfigured in an automated way. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner

## Monitoring and Logging

Monitor metrics and logs to see how application and infrastructure performance impacts the experience of their product’s end user.

## Communication and Collaboration

The use of DevOps tooling and automation of the software delivery process helps speed up communication across developers, operations, and even other teams like marketing or sales, allowing all parts of the organization to align more closely on goals and projects.

# Benefits of DevOps

## Speed

Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results

E.g: Microservices and Continuous Delivery let teams take ownership of services and then release updates to them quicker.

## Rapid Delivery

Increase the frequency and pace of releases so innovate and improve product faster.Users receive new features and updates frequently. As a result, company becomes more competitive.

Eg: **Continuous Integration** and **Continuous Delivery** are practices that automate the software release process, from build to deploy.

## Reliability

Ensure the quality of application updates and infrastructure changes.Minimize the chances of outrages. Even when the outrages do happen, everything can be fixed in a matter of minutes so that users won’t even notice that something is wrong.

E.g: **Continuous Integration** and **Continuous Delivery** to test that each change is functional and safe. Monitoring and Logging practices help you stay informed of performance in real-time.

## Scale

Automation and consistency help you manage complex or changing systems efficiently and with reduced risk.

E.g: **Infrastructure as code** helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

## Improved Collaboration

Build more effective teams under a DevOps cultural model.By removing the barriers between development and operations, two traditionally separate or siloed teams, a DevOps model creates an environment where the teams work together, optimising both the productivity of developers and the reliability of operations.

E.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run.

## Security

Adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques.

Eg: **Infrastructure as code** and **policy as code**, you can define and then track compliance at scale.

# Virtual Servers

As we intend to implement different server instances, we are going to use “virtual servers”

A "virtual server" is a software implementation that executes programs like a real server. Multiple virtual servers can work simultaneously on one physical host server. Therefore, instead of operating many servers at low utilization, virtualization combines the processing power onto fewer servers that operate at higher total utilization.

Virtualization improves scalability, reduces downtime, and enables faster deployments. It speeds up disaster recovery efforts because virtual servers can restart applications much more rapidly than physical servers. With virtualization,moving entire systems from one physical server to another in just a few seconds to optimize workloads or to perform maintenance without causing downtime. Virtualization solutions also have built-in resiliency features, such as high availability, load balancing and failover capabilities.

# Docker

The idea is to create a software factory that will automate the deployment process to different maintenance servers, that will be used for different technical tasks like development, quality assurance, production, etc.

These servers all have to handle the same configuration, to ensure the performance of the system when it is under production, and also to be able to replicate issues and bugs in an accurate way.

In order to perform this, we will use Docker.

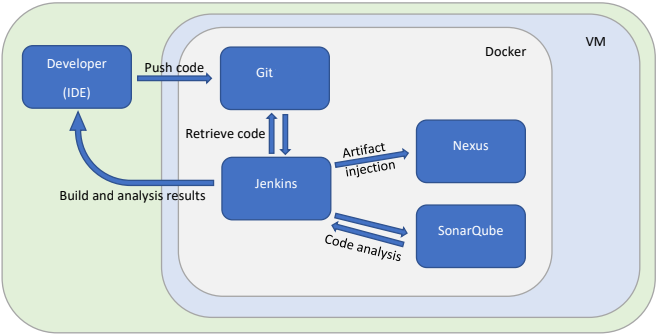
Docker is a containerization tool that enables users to virtualize an perform operating system level instructions to this virtualized containers.

In the present project Docker will enable us to create a container image with the closest configuration that the SIAM document describes. This way we will be able to achieve the standardization of the servers according to the DEVOPS approach.

One important feature about docker is its capability to create container images and easily deploy them, by using this capability, we also ensure the scalability of the solution, because if we make one configuration change in one of the images, we can generate its image and deploy it to the other servers as well.

# Software factory

In order to improve the development of the SIAM application, we are proposing the following software factory stack, that will enable the fast industrialization of any change within the source code.



In the diagram, it is depicted our proposal for technologies, that will be used by our team to deploy the application to the servers, and automate the development process.

All these technologies will be handled by docker and they will interact between themselves.

We are going to use Git as the version control software of the application, installing it on a virtual server run by docker, the main reason for this is that it will enable the developers to share their code and their changes in an efficient manner, it is also a free tool, thus it reduces the cost of buying a license for another version control tool.

As the SIAM application is built in Java, we have selected Jenkins as our continuous integration tool, given that it is built in java, and has lots of tools that will automate the deployment process, it also provide lots of plugins to communicate with the other applications of these software factory.

Nexus is the selection for our repository manager, the idea of this is to be able to have the different artifact versions of the application and its dependencies, this will enable us to work and test the newest versions in the QA and Development servers, while we also save stable versions that can run on the production site.

Finally SonarQube is our choice for continuous inspection of the code, which will enable our teams to identify potential issues within our code.

In terms of scalability, this architecture is capable of changing one of its components and through proper configurations keep working, this is in case that for instance someone decides to stop using the simple git instance to start using a more complex and secured application like gitlab.

# JVM and JBoss

JBoss is an application server that is responsible for building, deploying and hosting Java applications and services. JBoss is written in Java, so we will need to run it in a JVM.

Code is injected from the software factory into JBoss to be built, then it is automatically deployed and hosted to be accessible to users. In other words, developers can easily make updates to the application by automating the building and hosting process.

In the context of this project, the idea is to create different instances with the same JBoss configuration, so emulate the same behavior in each one of them, versions of the application will be deployed to each one of these instances.

To handle these server instances, a docker image will be configured and exported only once, but deployed in each server instance, these way we ensure the same conditions for each environment.

The servers will run the following configuration.

* Operating system CentOS 7.2.1511
* Java virtual machine (JVM) jdk1.8.0\_77
* JBoss 7.1.1.Final

For these specifications, a dockerFile will be created, and this will contain the configuration for a single server.

This DockerFile will be used to build a docker image with the command

$ docker build --tag=myuser/myimage:1.0 /path/to/dockerfile

Then you can run different instances of this image with

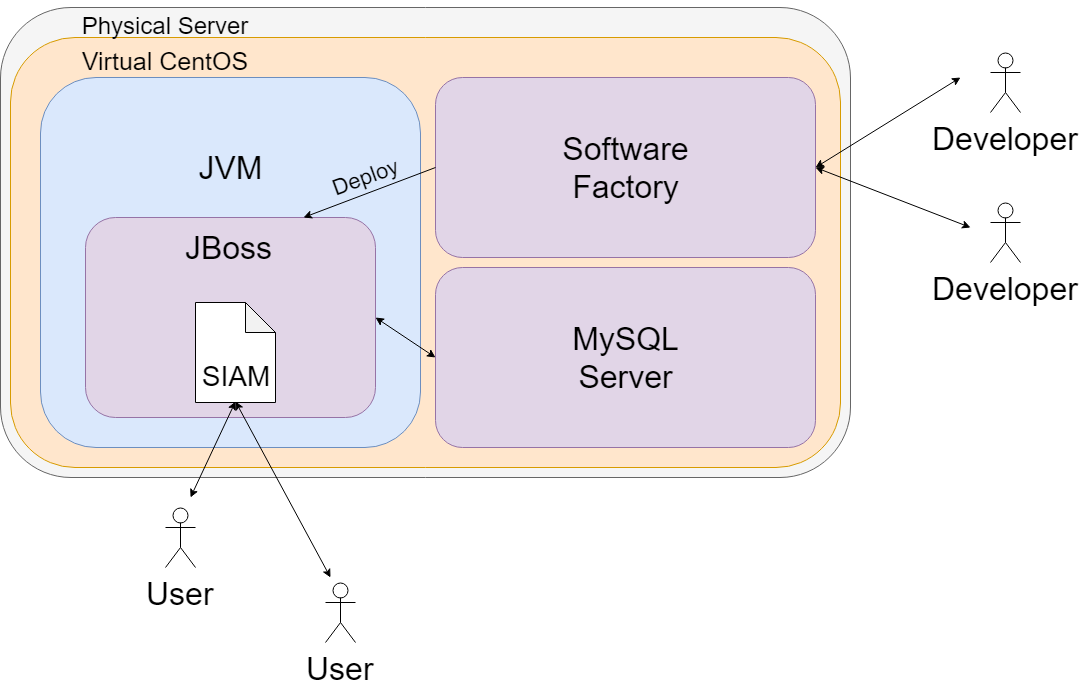
$ docker run -d -p 8080:8080 -p 9990:9990 myuser/myimage:1.0

Just change the port numbers.

# MySQL

Mainly, any application needs a database to store data. In our case, we are using MySQL server 5.5.47

# Conclusion - Interactions between the components



We can conclude by summarizing all the interactions between the above components.

* There is a physical server used for development purposes.
* A virtual CentOS in running on the physical server to minimise the effect of development/configuration problems.
* The coding environment that the developers will use is installed in Docker. Docker facilitate the duplication of environments.
* The software factory is a part of the coding environment, it is used to produce code.
* Code is then injected into JBoss that is running in a JVM.
* The code there is built and deployed, which makes the application running and accessible by the users.
* The application needs a database to run, so when needed, it communicates with the MySQL server to interact with the database.