



Desenvolvimento e manutenção de pipelines de integração contínua para aplicativos móveis

MAHA ASFOUR

Julho de 2020

Development and Maintenance of Continuous Integration pipelines for Mobile Applications

MAHA ASFOUR

**Dissertation to obtain the master's degree in
Computer Engineering, Specialization Area in
COMPUTER SYSTEMS**

Advisor: Prof. Alexandre Bragança

Jury:

President:

[President's Name, Category, School]

Vowels:

[Name of vowel1, Category, School]

[Name of vowel2, Category, School] (up to 4 vowels)

Porto, [July] [2020]

Thanks

I would like to pay tribute to the support and great love of my parents and siblings, it would not have been possible to achieve my ambitions without their trust and encouragement.

The contribution of 'The Global Platform for Syrian students' is truly appreciated, especially, the former President of Portugal DR. Jorge Sampaio, DR. Helena Barroco that facilitated my first two years in Portugal, and all the sponsors. Without their support and funding, I could not have a chance to continue my studies in Portugal.

I would also like to express my deep appreciation to my academic supervisor, Professor Alexandre Bragança for his support and the cognitive value that he presented, and to extend my deepest gratitude to COCUS's DevOps team, especially my technical supervisor, the engineer Pedro Coutinho, who was generous, patient, flexible, and supportive within nine months of the internship, and to the scrum master MR. Nuno Alves.

Finally, I would like to say, I always felt at home thanks to Portugal, the beautiful and hospitable country, Instituto superior de engenharia do Porto, and COCUS IT company.

Resumo

Recentemente, o mercado de TI tornou-se cada vez competitivo, à medida que os produtos de software são desenvolvidos frequentemente a um ritmo notável. Isto deve-se ao facto as empresas automatizarem os seus processos de release dos seus produtos de software tais como compilação, testes e lançamento, utilizando ferramentas de integração e entrega contínuas.

Atualmente, existem ferramentas de DevOps capazes de ajudar a acelerar o lançamento dos produtos de software, como aplicativos móveis, através pipelines de CI/CD, tornando o processo de release estável e eficiente.

O objetivo desta tese é propor uma ferramenta para CI / CD e fornecer uma solução para implementar e manter as pipelines de CI / CD para os projetos Android e IOS da TUI na COCUS.

Keywords: DevOps, CI/CD pipeline, Metrics system, Mobile Applications

Abstract

Recently, the IT market has become more competitive, as software products are developed and updated daily at a remarkable pace. This forces companies to automate the process of their release life cycle, such as building, testing, and deployment using continuous integration and continuous delivery tools to provide better quality and speed for their products

Currently, many DevOps tools are making efforts to accelerate the deployment of software products, like mobile applications, using CI / CD pipelines and make this process constant and reliable.

The purpose of this thesis is to propose a CI / CD tool and provide a solution to develop and maintain the CI / CD pipeline for TUI's Android and IOS projects at COCUS.

Keywords: DevOps, CI/CD pipeline, Metrics system, Mobile Applications

Index

| | |
|--|-----------|
| 1 Introduction..... | 1 |
| 1.1 Background/Context | 2 |
| 1.2 Problem Description | 3 |
| 1.3 Goals..... | 3 |
| 1.4 Work Organization | 3 |
| 1.5 Report Contents..... | 4 |
| 2 State of The Art | 7 |
| 2.1 Related Work | 7 |
| 2.2 Existing Technologies | 9 |
| 2.2.1 Jenkins | 9 |
| 2.2.2 Buddy..... | 11 |
| 2.2.3 Travis CI | 14 |
| 2.2.4 Teamcity..... | 15 |
| 2.2.5 CircleCI | 18 |
| 2.2.6 BuddyBuild | 20 |
| 2.2.7 Bitrise | 21 |
| 2.2.8 Drone..... | 24 |
| 2.2.9 Bamboo | 26 |
| 2.2.10 GitLab | 29 |
| 2.3 Comparing the Technologies | 31 |
| 3 Business Value Analysis | 35 |
| 3.1 New Concept Deployment | 35 |
| 3.1.1 Opportunity Identification | 36 |
| 3.1.2 Opportunity Analysis | 36 |
| 3.1.3 Idea Genesis & Enrichment | 39 |
| 3.1.4 Idea Selection | 39 |
| 3.1.5 Concept and Technology Definition..... | 39 |
| 3.2 Value Creation | 40 |
| 3.2.1 Value | 40 |
| 3.2.2 Customer Value | 40 |
| 3.2.3 Longitude Perspective of Value | 41 |
| 3.3 Value Proposition | 42 |

| | |
|---|-----------|
| 3.4 Quality Function Deployment (QFD) | 43 |
| 4 Design | 47 |
| 4.1 Non-Functional & Functional Requirements | 47 |
| 4.2 Use-Case Diagram..... | 48 |
| 4.3 Activity Diagram | 49 |
| 4.4 CI/CD Pipelines diagrams for Android & IOS Apps..... | 50 |
| 4.5 Component Diagram..... | 51 |
| 4.6 Alternatives and Selected Solution..... | 52 |
| 4.6.1 Shared Runners & Docker image (Android) | 53 |
| 4.6.2 AWS EC2 instance & registering Shell Runner (Android) | 53 |
| 4.6.3 Kubernetes cluster & Docker image (Android) | 54 |
| 4.6.4 Mac Servers (IOS) | 56 |
| 5 Implementation..... | 59 |
| 5.1 Implementation for Android..... | 59 |
| 5.1.1 Create Cluster in EKS AWS service | 59 |
| 5.1.2 Configure the cluster as runner in Gitlab | 63 |
| 5.1.3 Build Docker image..... | 66 |
| 5.1.4 Push Docker image to ECR AWS registry | 67 |
| 5.1.5 Use S3 as Cache server | 68 |
| 5.1.6 Add Gitlab Pipeline & bash script to Repository | 68 |
| 5.2 Implementation for IOS | 72 |
| 5.2.1 Configure the Mac servers as runners in Gitlab | 72 |
| 5.2.2 Use S3 as Cache server | 73 |
| 5.2.3 Add Gitlab Pipeline & bash script to Repository | 74 |
| 6 Experimentation & Evaluation..... | 79 |
| 6.1 Research Hypothesis Specification | 79 |
| 6.2 Indicators & Source of Information..... | 79 |
| 6.3 Evaluation Methodology | 80 |
| 6.4 Evaluation Results | 80 |
| 7 Conclusion..... | 87 |
| 7.1 Summary | 87 |
| 7.2 Future Work | 88 |

List of Figures

| | |
|---|----|
| Figure 1 - Create a new Freestyle job in Jenkins | 10 |
| Figure 2 - Add new pipeline by UI Buddy (pipeline-examples, 2020) | 12 |
| Figure 3 - configure an action in UI Buddy pipeline (buddy-action, 2020) | 13 |
| Figure 4 - build stages work on Travis ci (Tarvis-build, 2020) | 15 |
| Figure 5 - Teamcity Architecture (Despa, 2020) | 16 |
| Figure 6 - Build configurations using UI Teamcity (Teamcity-Build Configurations, 2020)..... | 17 |
| Figure 7 - REST API requests examples for Teamcity (REST API-Teamcity, 2020)..... | 17 |
| Figure 8 - CircleCI Architecture (Overview – CircleCI, 2020)..... | 19 |
| Figure 9 - choosing language to generate a YML pipeline-CircleCI (CircleCI-start, 2020)..... | 19 |
| Figure 10 - BuddyBuild-Release process (BuddyBuild-Release 2020)..... | 21 |
| Figure 11 - Bitrise Architecture – Android apps (Startupcraft.io, 2020) | 22 |
| Figure 12 - Bitrise Architecture – IOS apps (Startupcraft.io-IOS, 2020) | 22 |
| Figure 13 - Bitrise – verification workflow Android example (Droids Roids – Poland, 2020) ... | 23 |
| Figure 14 - Bitrise – Debug workflow IOS example (Savvy Apps., 2020) | 23 |
| Figure 15 - Drone Architecture (Slideshare.net., 2020) | 25 |
| Figure 16 - Bamboo’s Plan Architecture (Atlassian-Configuring plans 2020) | 27 |
| Figure 17 - Bamboo-Plan configuration (Atlassian-Configuring plans 2020)..... | 27 |
| Figure 18 - Bamboo-Plan configuration-as-code (Bamboo 2020)..... | 28 |
| Figure 19 - GitLab pipeline graph (Gitlab-pipeline 2020)..... | 30 |
| Figure 20 - NCD Model’s components (Koen et la., 2001, p.47)..... | 35 |
| Figure 21 - life cycle of the mobile application with focusing on CI/CD steps..... | 40 |
| Figure 22 - Desired & Perceived Value of our customer TUI in this project | 41 |
| Figure 23 - A Longitudinal Perspective on VC (Woodall, 2003) | 41 |
| Figure 24 - House Of Quality for QFD (Bouchereau and Rowlands, 2020) | 43 |
| Figure 25 - the house of quality of TUI’s CI/CD pipeline mobile applications project | 44 |
| Figure 26 - Use case diagram for CI/CD system | 48 |
| Figure 27 - Activity Diagram for release life cycle..... | 49 |
| Figure 28 - CI/CD pipeline for TUI’s Android application | 50 |
| Figure 29 - CD/CD pipeline for TUI’s IOS application | 50 |
| Figure 30 - Component Diagram for CI/CD tool..... | 52 |
| Figure 31 - Architecture of shared Runners & Docker Image solution | 53 |
| Figure 32 - Architecture of AWS EC2 instance & registering shell Runner | 54 |
| Figure 33 - Architecture of Kubernetes cluster & Docker image | 55 |
| Figure 34 - prices of EC2 & EKS based on the project..... | 55 |
| Figure 35 - Architecture of Mac servers & S3 bucket | 56 |
| Figure 36 - navigate IAM service & create role..... | 59 |
| Figure 37 - select AWS services-granting permissions for AWS service | 60 |
| Figure 38 - Choose EKS-allowing EKS to manage the resources | 60 |
| Figure 39 - Name the role and create it..... | 60 |

| | |
|--|----|
| Figure 40 - Show the new EKS role in IAM Roles page | 61 |
| Figure 41 - Create cluster in EKS service | 61 |
| Figure 42 - Name the cluster and choose the IAM EKS Role..... | 61 |
| Figure 43 - adding node group in cluster | 62 |
| Figure 44 - Details of the node | 62 |
| Figure 45 - Runners page in Gitlab | 63 |
| Figure 46 - Adding existing cluster in Gitlab..... | 63 |
| Figure 47 - Cluster details in AWS | 64 |
| Figure 48 - the details of service account (Gitlab-Kubernetes, 2020)..... | 65 |
| Figure 49 - Running Cluster with tags in Gitlab | 65 |
| Figure 50 - Create new Repository in ECR..... | 67 |
| Figure 51 - View the push commands in the repository | 67 |
| Figure 52 - insert environment variable to build a specific app..... | 71 |
| Figure 53 - Running Pipeline for Android project..... | 72 |
| Figure 54 - passed pipelines for the Android apps..... | 72 |
| Figure 55 - Gitlab Runner for IOS project..... | 73 |
| Figure 56 - Running Pipeline for IOS project | 76 |
| Figure 57 - passed pipelines for IOS apps..... | 76 |
| Figure 58 - passed pipelines for IOS apps..... | 77 |
| Figure 59 - Jenkins and Gitlab Android pipelines results | 81 |
| Figure 60 - Analytics CI/CD times' charts in Gitlab for Android project..... | 81 |
| Figure 61 - Jenkins and Gitlab IOS pipelines results..... | 82 |
| Figure 62 - Analytics CI/CD times' charts in Gitlab for IOS project | 82 |
| Figure 63 - Analytics CI/CD success rate' charts in Gitlab for Android project | 83 |
| Figure 64 - Analytics CI/CD success rate' charts in Gitlab for IOS project..... | 83 |
| Figure 65 - Gitlab survey results..... | 84 |
| Figure 66 - Comparison between Gitlab, Jenkins, and BuddyBuild..... | 85 |

List of Tables

| | |
|--|----|
| Table 1 - part of the comparison table for the filtered CI/CD tools..... | 31 |
| Table 2 - 1 st comparison table of CI/CD tools based on the company's requirements..... | 37 |
| Table 3 - 2nd comparison table of CI/CD tools based on the company's requirements | 38 |
| Table 4 - Benefits & sacrifices of the value analysis | 42 |

Acronyms and Symbols

List of Acronyms

API Application program interface

APK JAR Android Package

AWS Amazon Web services

CD Continuous Delivery

CI Continuous Integration

CLI Command Line Interface

DEV Development/Developers

ECR Elastic Container Registry

EC2 Elastic Compute Cloud service

EKS Elastic Kubernetes Service

GUI Graphical user interface

IAM Identity and Access Management

IPA iPhone Application Archive

JDK Java Development Kit

NCD New Concept Deployment

OPS *operations team*

PaaS Platform as service

QA Quality Assurance

QFD Quality Function Deployment

REST Representational state transfer

SDK Software Development Kit

S3 Simple Storage Service

VCS Version control system

1 Introduction

The humanity ages were divided according to the most important inventions/achievements that occurred during several periods.

The information age began in the 1970s and continues to this day. The common feature of this age is the speed, and the most important environmentally friendly products are software products, especially mobile applications.

Before 2008, the software's life cycle suffered from slow progress, isolated teams (Development, Quality Assurance, and Operations) and disorganization in the workflow, causing the responsibility of problems among the teams, which led to slow and unreliable releases and many products issues.

Then, the DevOps (development and operation) concept began as a discussion between Patrick Debois and Andrew Clay Shaffer regarding the agile infrastructure concept, to solve these problems, by improving collaboration between all stakeholders from planning to delivery and automating the delivery process.

As a result, according to the 2015 DevOps Status Report, "High-performance IT organizations deploy maximum 30 times with short lead times of up to 200 times; they have 60x less failure and 168x faster recovery " (Forsgren, 2015).

Also, some related practices showed, like continuous integration (CI) and continuous delivery (CD).

- CI: is the practice of automating the integration of code changes from multiple contributors into a single software project.
- CD: is an extension of continuous integration to make sure that the developer can release new changes to the customers quickly in a sustainable way. This means that on top of having automated tests, it's possible to have automated releases processes and deploy the application at any point in time by clicking on a button.

In theory, with continuous delivery, teams can decide to release daily, weekly, fortnightly, or whatever suits the business requirements.

In the first quarter of 2018, new Android app releases reached to 6,140 applications per day (Statista, 2019), which means new applications are released and old ones are updated daily, trying to cover the needs of people in most of the fields, and this makes the competition and the challenges very intense among application producers.

1.1 Background/Context

This thesis had been prepared, during the internship in COCUS company. (COCUS AG, 2019)



COCUS is a consulting medium-size firm, founded in 2000 with locations in Germany, Portugal, and Switzerland, it offers IT solutions in several areas: Internet of Things, Blockchain, Data Analytics, and Information Security.

COCUS has direct engagement with a tourism company called TUI.

TUI (Touristik Union International) is an Anglo-German multinational, largest leisure, travel and tourism company in the world.

One of the services that COCUS offers to TUI, is developing their mobile applications, to make it easy for TUI's clients, to enjoy with their offers and services.

With 11 TUI's Applications are developed by COCUS and spread around the world, COCUS strives to improve them, and this demand extra efforts from the DevOps team, which unfortunately has just two DevOps engineers right now, to develop and maintain the pipelines.

1.2 Problem Description

The pipeline of TUI's mobile applications project suffers from some problems using Jenkins and BuddyBuild CI/CD tools

- The first problem is the long time that pipeline needs in every build job (around 31 minutes per build in Jenkins) which means lateness in the product's life cycle.
- The second one is the need for using a metrics system, where the DevOps team can monitor the pipeline, and measure the performance of it.

Because of these problems, the company decides to move TUI's mobile applications project from Bitbucket to Gitlab and use just one CI/CD tool instead of Jenkins and BuddyBuild.

To implement this decision, this internship will focus on the latest DevOps technology solutions to build test and release new updates of TUI's applications, in the fastest possible successful way, then compare the solutions' results, by the metrics systems.

1.3 Goals

The goals of this thesis are to use one CI/CD tool for Android and IOS applications project instead both of previous tools to improve the speed of the CI/CD pipeline builds, satisfy the need of the developers to use a new simpler and clearer tool where it will be easy to detect and understand any issue and implement metrics system that can monitor the performance of the pipeline automatically, then discover easily the other problems that may the pipeline has and make the differences between the pipelines' performances clear.

1.4 Work Organization

The agile approach is adopted at COCUS, where it helps teams manage their projects, by organizing the work and delivering small but consumable values to the customers, in stages (Agile, 2020).

To implement this methodology, the Jira tool has been used and it is provided by Atlassian (Agile-Jira, 2020).

DevOps team has its board on Jira, where it contains the members' tasks that are ready to do, in progress, in review and the done ones.

The team organizes the work through 3 kinds of meetings:

- Daily meetings to update the state of the tasks.
- Weekly meetings to add and score tasks (evaluate the task's difficulty).

- Half-month meetings to review the sprint, which is amount of the tasks that must be completed to deliver value to the customer.

1.5 Report Contents

As this thesis will show, the first chapter, introduction, is where the main problem is defined along with the objectives and the expected results.

The second chapter is state of the art and will contain the related work and ten of the existing CI/CD tools.

In the third chapter, a study about the value of this project is included, and the NCD model, with the QFD representation.

The fourth chapter is the design, where it includes the requirements, the diagrams of the system, and analyzing the alternatives and selected solutions.

In the fifth chapter, the implementation, the adopted solution will be implemented based on the system's architecture.

The sixth chapter, experimentation & evaluation. It is presented the factors that will be used to evaluate this solution to decide if it reached the goals of this thesis or not

Finally, the future work with the conclusion will be found in the seventh chapter.

2 State of The Art

In this chapter, the latest CI/CD technologies will be discussed, to decide the suitable tool for this project, where it's important to match the collected data with the project's requirements, besides the deep understanding for the limitations of the project, and the priorities of the company, these include technical limitations, available resources, company flexibility, maintainability, costs, and security.

In light of the above, and as a result of comparing the available solutions to this problem, it will be easy for the engineer to take the optimal solution, to implement it.

This chapter also refers to all the work environment requirements, and the necessary tools, technologies, and systems, that are used in this project.

2.1 Related Work

After the rapid development of DevOps concepts and CI/CD systems and increased the need to implement them in IT projects, a lot of related projects can be observed today, but every one of them has its problems on the level of CI/CD pipelines, where it is not possible to have fixed solutions that can be adapted by any project, but they can help to have a wide vision about the possible and the available approaches.

One of these projects is Glintt web-application project (Ramalho, 2018) that suffers from the delayed releases in the life cycle of the app, one of its problems is related directly to CI/CD process, where the CI build triggers once per day during the night in TFS(team foundation server) this causes unclear issues because the build process is started after a lot of changes made by developers during the day, as well as the project lacks to the automated tests which means a lot of issues observed by QA team in manual test statement. The problems solved by adding all the needed tests(like unit tests, integration tests) and creating a pipeline using Jenkins to run these tests automatically, on the other side they kept using TFS after changing the configuration of builds times, whenever the developers edit the code and merge their edits in Dev branch with CI branch, the build process will start in TFS, then the tests workflow will run in Jenkins.

Jenkins was chosen for tests because it's an open-source tool with plenty of plugins, but it wasn't chosen for the build phase because of the integrity constraints with the TFS version where Jenkins can't detect the changes in TFS and automatically start a build.

The followed solution for Glintt application is useful for a specific kind of projects, regardless the limitations of TFS if its features are compared with the CI/CD tools' ones, besides that the solution is limited, because it increases the number of times of CI/CD process, but the developers still have to merge their changes in CI branch to test them, this causes unclear

issues again in case of more than one developer merge their changes in the same time or one developer merges many changes, which means the problem of the delayed-release life cycle of the app still exists, and this what must be avoided in this thesis, where the solution will focus on decreasing the consuming time of the release life cycle, by choosing the modern CI/CD systems that guarantee pipelines with better performance and implementing the solution in the more efficient way to get faster releases versions that satisfy the final customer.

Another project about the Shopping List application (Ezugbaya, 2019). The main purpose of it was to create a CD pipeline for ReactNative application, where the ReactNative is a framework for building IOS and Android Apps (React Native, 2020). This means the pipeline must be created using a CI/CD system that supports both Android and IOS projects.

To solve the problem Circle CI was nominated to implement the solution and to trigger the IOS version, they bought a macOS plan that provided on Circle cloud (mac VMs) (CircleCI-macos, 2020), on the other hand, they used Docker image as executor for the Android version.

The goal of the thesis has been achieved perfectly, although Circle CI has a limitation by imposing hiring its mac VMs to build IOS projects, where it does not provide the self-hosted option for the user, on the opposite of what this thesis is looking for.

From the student's perspective, the pipeline can be more enhanced too, by including the environmental requirements in the Docker image instead of installing them in the pipeline, and this is an important part of this project.

One more project covers strengthening the development environment, here for Piceasoft mobile applications (Salminen, 2020), through the design and implementation of continuous delivery, continuous integration, and continuous deployment, as the project lacks automated builds, tests, and releases.

To reduce the required manual effort, Gitlab CI was chosen to implement the CI / CD concepts, and as a result, the project became fully automated and the time required to integrate and release the application was reduced, leading to the goal being achieved.

Piceasoft's project focuses on automating the application life-cycle and includes the continuous deployment, while this thesis aims to enhance the performance of the application's CI/CD pipeline and finding the optimal architecture to reduce the pipeline's time without implementing the continuous deployment.

2.2 Existing Technologies

In the last few years, the market of DevOps tools became competitive and diversified, where a suitable tool for the DevOps department can be found, based on the project's requirements.

However, because of the need for better, faster, cheaper, and more secure service is unlimited in IT fields, and the important role of DevOps by being the bridge between the developers and the operation teams, the progress of DevOps tools market is still increased, and DevOps teams still follow the latest technologies to improve their works.

In this thesis, the latest available CI / CD tools will be introduced and investigated, which can help to solve the project's problems, then the advantages and disadvantages of each one will be identified, to compare them, and adopt the optimal solution.

2.2.1 Jenkins



According to Katalon, Jenkins is one of the most popular CI/CD tools (Katalon Solution, 2020)

“It's open-source free automation server, provides hundreds of plugins to support building, deploying, and automating any project” (Jenkins, 2020).

Jenkins is compatible with Windows, Mac, and Linux systems, and integrated with important tools like Docker, which means it offers the ability to hire Docker containers as agents, where they can run the pipeline or pieces of the pipeline to implements the distributed pipeline concept.

This feature is useful to split the tasks of the pipeline between many agents, then having better performance and faster builds.

It is easy to download and run, but before installing it, be sure that the machine has JDK 8 or 11 after that just type the following command line in terminal to install it on Mac:

```
Brew install jenkins-lts
```

Go to the directory of Jenkins in the terminal and type the following line to run the service:

```
Java -jar jenkins.war --httpPort=8080
```

The main feature about Jenkins is the ability to implement the automation build, test, release, and the other works in 3 different ways, which means, it's flexible and easy to learn:

- GUI (Graphical User Interface): or it's called Freestyle, this is the first and the easiest way to create a job in Jenkins, but it's not practical because, it's necessary to repeat

all the configurations and the steps in the case of using another physical machine or another Jenkins account.

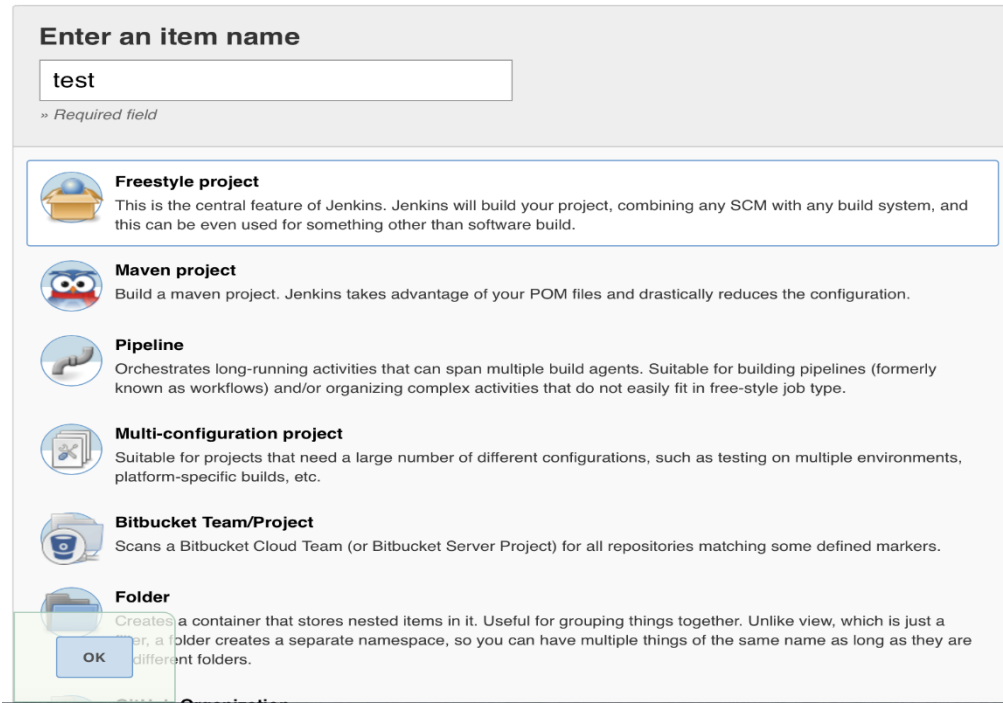


Figure 1 - Create a new Freestyle job in Jenkins

- Declarative Pipeline: to build pipeline-as-code by Groovy programming language for the same work that was implemented by GUI, it is a little bit complicated, but as well as it is more useful, because it allows adding the work to the project's repository, in Jenkins files, which means there is no need to repeat the work.

“In contrast to freestyle jobs, pipelines enable you to define the whole application lifecycle. Pipeline functionality helps Jenkins to support continuous delivery (CD)” (Pipeline Syntax, 2020).

In this case, Jenkins will be able to grab the build. Groovy file from Jenkins files and run it, in every time you click build now.

```
pipeline {
  agent none
  stages {
    stage('Example Build') {
      agent { docker 'maven:3-alpine' }
      steps {
        echo 'Hello, Maven'
        sh 'mvn --version'
      }
    }
    stage('Example Test') {
      agent { docker 'openjdk:8-jre' }
      steps {
        echo 'Hello, JDK'
      }
    }
  }
}
```

```

        sh 'java -version'
    }
}
}

```

Code 1 - Declarative pipeline example (Pipeline Syntax, 2020)

- Scripted Pipeline: another way to achieve the concept pipeline-as-code, “Unlike the Declarative pipeline, the scripted pipeline strictly uses groovy based syntax. Since this, the scripted pipeline provides huge control over the script and can manipulate the flow of script extensively” (Edureka Community, 2020), the difference between the Declarative pipeline and Scripted pipeline can be noticed by the syntax.

```

node {
    stage('Example') {
        try {
            sh 'exit 1'
        }
        catch (exc) {
            echo 'Something failed'
            throw
        }
    }
}

```

Code 2 - Scripted pipeline example (Pipeline Syntax, 2020)

Cons

- Jenkins is a useful tool in the case of a simple pipeline, but it will be more complicated to work with it in case of advanced pipelines (Tools and Reviews, 2020).
- Hard-understandable issues in complex pipelines, which means the need for a support team to solve the builds issues in Jenkins.

2.2.2 Buddy



It is a CI/CD tool, with 87% faster CI/CD adoption time by teams, and a very fast deployment process about 12 seconds on average, it supports Docker containers uses and concurrent pipelines. (Buddy: The DevOps Automation Platform, 2020)

Buddy is integrated with many version control repository hosting services, like GitLab, Bitbucket, and GitHub, where the project’s source code can be stored easily, while it has over 100 ready-to-use actions that can make it very easy to build, test, and deploy any project immediately.

There is no need to set up Buddy on the machine, just create an account and start the first work, after importing the repository.

by Buddy, the CI/CD pipeline can be achieved in 2 ways:

- GUI: using an extremely friendly user interface, the user can add a new pipeline, type any related name and choose a trigger mood (for example choose On push to run the pipeline every time the changes are pushed to the repository, or manually, or scheduled), then specify the branch of your repository.

Add a new pipeline [?]

NAME

TRIGGER MODE

Manual On push Recurrently

BRANCH OR TAG THAT TRIGGERS THE PIPELINE

MORE OPTIONS

+ Site URL, Currently deployed revision, Clone depth & Visibility

Add a new pipeline

Figure 2 - Add new pipeline by UI Buddy (pipeline-examples, 2020)

In Actions section the user can configure or add a new step in the pipeline by selecting the appropriate choices for the project then test the action, in this case Buddy could help busy teams to accelerate the continuous integration and continuous delivery.

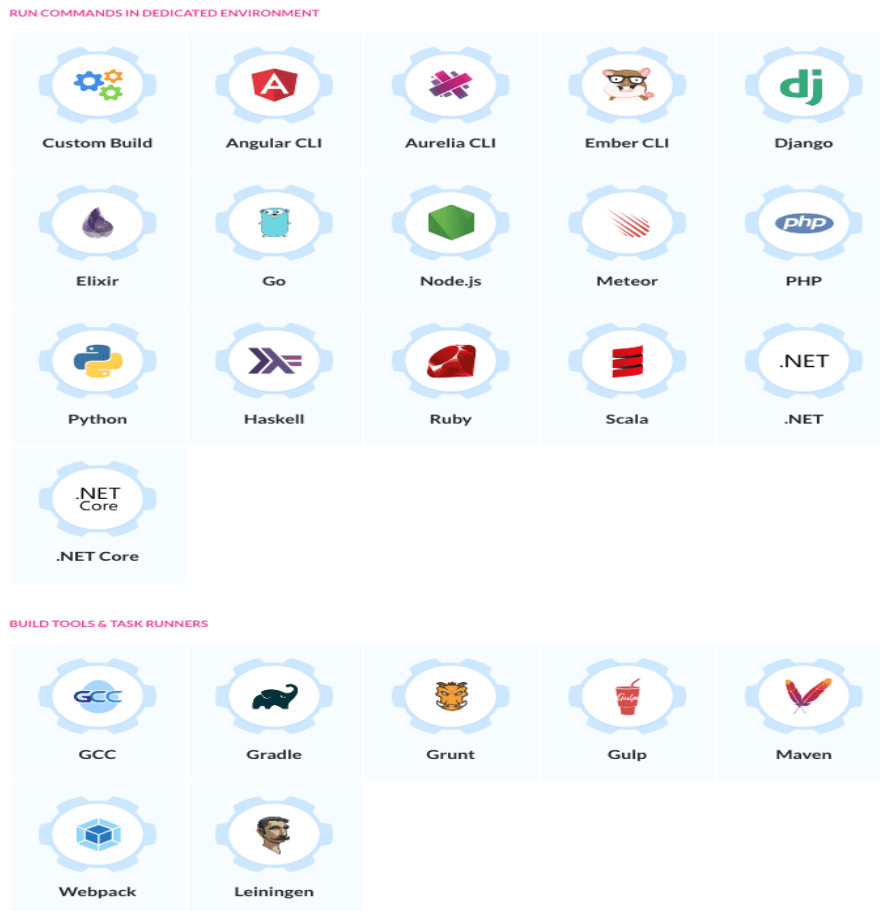


Figure 3 - configure an action in UI Buddy pipeline (buddy-action, 2020)

- **YAML:** where the configuration is performed as pipeline-as-code via `buddy.yml` located in the repository.
Buddy gives the possibility for saving the work in the repository, and much more, where it can convert the GUI pipeline to code and export it as a YML file.

After saving `Buddy.yml` in the repository, it's necessary to flip the switch to YAML, in Buddy pipelines page, to be able to run the pipeline.

Cons

According to many reviewers, Buddy is a very expensive tool if the project requires a lot of parallel jobs running comparing with other CI/CD tools (buddy-review 2020).

2.2.3 Travis CI



One of the strongest competitors for Jenkins, and one of the most favorite tools for developers (StackShare, 2020).

According to the Travis CI website, it defines as " The continuous integration and continuous delivery platform your team knows and loves, on your infrastructure" (Travis CI Enterprise, 2020).

Travis ci is integrated with GitHub, has high simplicity for setting up, supports building on Docker, and reduces the potential of leaking the credentials.

To start working with Travis ci, the user needs to sign in the GitHub account, then give Travis permission to reach to the repositories.

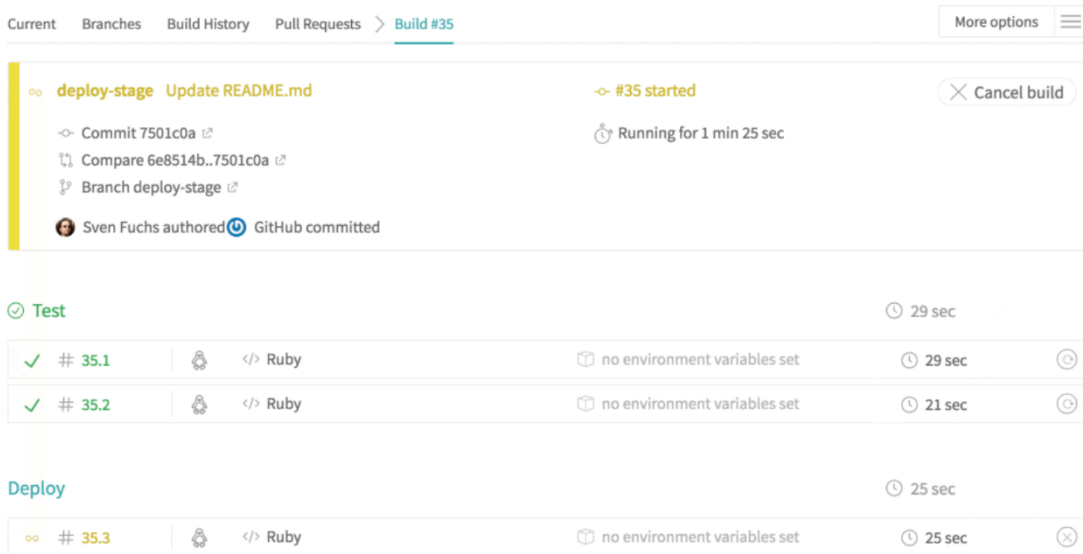
The main difference between this tool and the previous ones, that it just supports the pipeline-as-code concept, which means it's not possible to start the CI/CD job using GUI, but Travis is not complicated to work with, if the user was familiar with YAML language, and followed its Documentation.

To build, test, and deploy the project using Travis ci, add a .travis.yml file to the repository, after including all the stages and the dependencies.

```
jobs:
  include:
    - stage: "Tests"
      stage
      name: "Unit Tests"
      script: ./unit-tests
    - script: ./integration-tests
      name: "Integration Tests"
    - stage: deploy
      name: "Deploy to GCP"
      script: ./deploy
```

Code 3 -.travis.yml example (Travis-example, 2020)

After pushing the changes to GitHub, the build will start on Travis ci, as Figure 4



The screenshot shows the Travis CI interface for build #35. The top navigation bar includes 'Current', 'Branches', 'Build History', 'Pull Requests', and 'Build #35'. The main content area displays the build progress:

- deploy-stage Update README.md**: Status is '#35 started' and 'Running for 1 min 25 sec'. It includes a 'Cancel build' button. Below this, it lists commit details: 'Commit 7501c0a', 'Compare 6e8514b..7501c0a', and 'Branch deploy-stage'. It also shows 'Sven Fuchs authored' and 'GitHub committed'.
- Test**: Status is '29 sec'. It contains two jobs:
 - Job #35.1: Status is '✓', using 'Ruby', with 'no environment variables set' and a duration of '29 sec'.
 - Job #35.2: Status is '✓', using 'Ruby', with 'no environment variables set' and a duration of '21 sec'.
- Deploy**: Status is '25 sec'. It contains one job:
 - Job #35.3: Status is '○', using 'Ruby', with 'no environment variables set' and a duration of '25 sec'.

Figure 4 - build stages work on Travis ci (Travis-build, 2020)

Cons

- It isn't the best choice for high-security projects (Guru99.com, 2020).
- It requires building the environment, and match it with Travis requirements (Tobies, 2020).

2.2.4 Teamcity



This tool defined as CI/CD server, developed by JetBrains, and written in Java language.

In addition to it's a rich tool with many important features and has interesting documentation, one of its features is Technology awareness.

According to the TeamCity website, the Technology awareness concept is explained as " When we say we support a tool, we mean it in every way possible. For example, support for Visual Studio projects provides automatic detection of tool versions, testing frameworks support, code coverage, static code analysis, and more. And the best thing is, you get all this support without installing any plugins" (JetBrains, 2020)

After making sure of installing Java version 8 on the machine, it's possible to download Teamcity, then go to Teamcity/bin directory using the terminal, and run the service by the following command line

```
./runAll.sh start
```

To stop it

```
./runAll.sh stop
```

Using the Browser, Teamcity will run on localhost:8111 URL

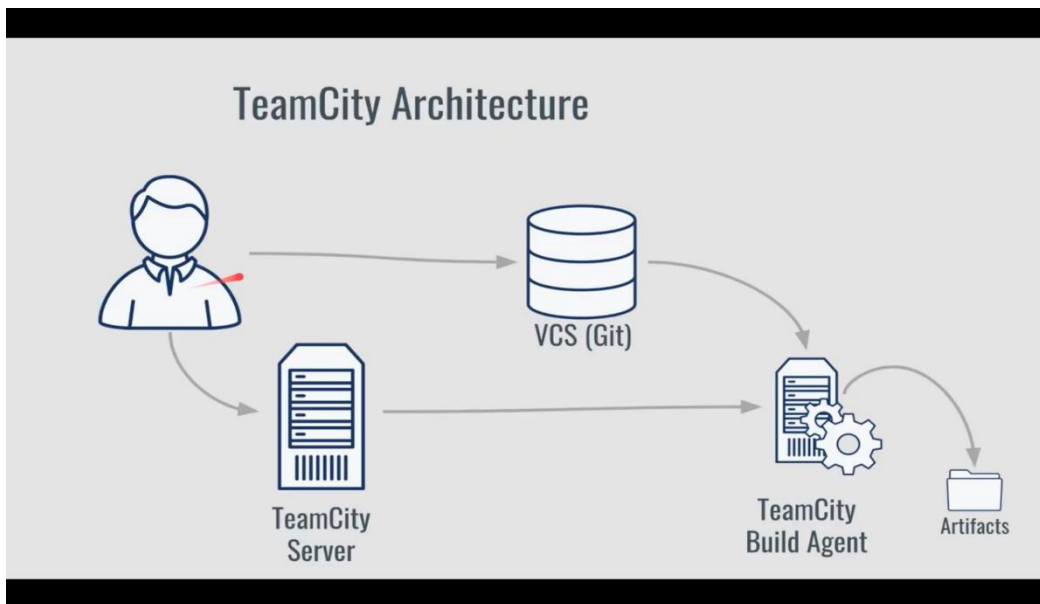


Figure 5 - Teamcity Architecture (Despa, 2020)

Teamcity has three ways to configure the build of the project.

- GUI: Teamcity has various features, and it offers different approaches to achieve the work, even in the UI the job can be configured, manually, using URL where Teamcity can detect the build steps automatically, by creating a template, or by many other choices.

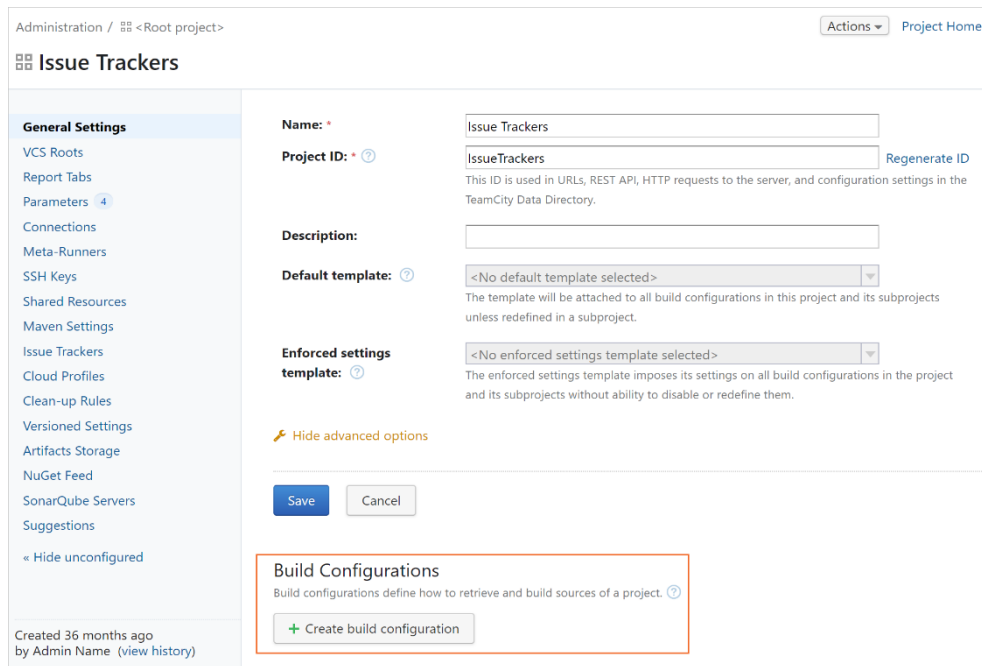


Figure 6 - Build configurations using UI Teamcity (Teamcity-Build Configurations, 2020)

- REST API: shortcut of Representational State Transfer, this approach uses HTTP protocol, and indicates sending requests to the server and waiting for its response, with the needed information (Mode, 2020).

Teamcity can configure the build via REST API, after authenticating REST, by giving it access token, used instead of the password.

Get/set paused build configuration state: **GET/PUT**

<http://teamcity:8111/app/rest/buildTypes/<buildTypeLocator>/paused> (put "true" or "false" text as text/plain).

Build configuration settings: **GET/DELETE/PUT**

http://teamcity:8111/app/rest/buildTypes/<buildTypeLocator>/settings/<setting_name>

Build configuration parameters: **GET/DELETE/PUT**

http://teamcity:8111/app/rest/buildTypes/<buildTypeLocator>/parameters/<parameter_name>

Figure 7 - REST API requests examples for Teamcity (REST API-Teamcity, 2020)

- Pipeline-as-code: based on the Kotlin language, the UI build configurations can be converted to code automatically, from versioned settings in the project's settings, then push it to the repository directory.

```

jobs:
Object Build: BuildType({
    Name = "Build"    artifactRules = "target/*.jar"

    Vcs {
        Root(PetclinicVcs)
    }
    Steps {
        Maven {
            Goals = "clean package"
        }
    }
    Triggers {
        Vcs { }
    }
})

```

Code 4 - Teamcity Kotlin code (Blog.jetbrains.com, 2020)

Cons

- For the beginners, it is hard to cover and understand the work logic of this tool even with the well-documentation (Teamcity reviews, 2020).
- So many settings, for each project, make it difficult to track down the errors (Teamcity reviews, 2020).

2.2.5 CircleCI



Defined as the continuous integration and delivery platform for macOS, Linux, Windows operating systems, and Android (CircleCI, 2020).

The most important characterizes in CircleCI are the speed of its jobs and the ability to configure very complex pipelines with high efficiencies, as well as supports the parallelism in the pipeline's workflow, and the caching concept to reuse any data from your previous jobs (Overview – CircleCI, 2020).

According to Forrester, CircleCI is defined as one of the top 10 significant cloud-native continuous integration tools (Forrester.com, 2020).

CircleCI supports many programming languages, with any framework or version, besides the flexibility to configure the team’s needs from memory and CPU (product – CircleCI, 2020).

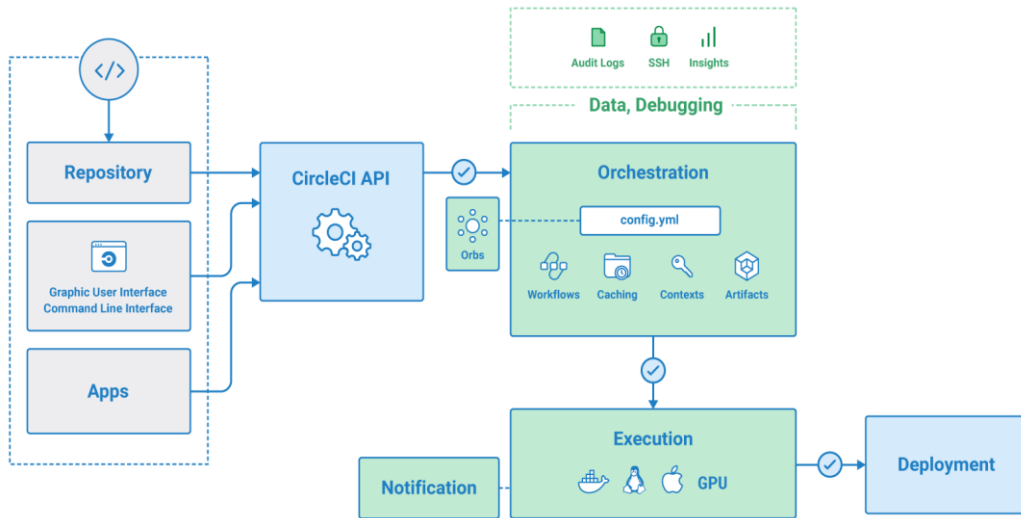


Figure 8 - CircleCI Architecture (Overview – CircleCI, 2020)

To start working with CircleCI, sign up with the GitHub or Bitbucket account.

After that add project to the dashboard directly, and using the CircleCI’s friendly UI, it will be easy to configure the first pipeline-as-code in YAML language by determining the language that was used in the code, then add the automatically generated code as a config.yml file to the repository, to start ci/cd pipeline.

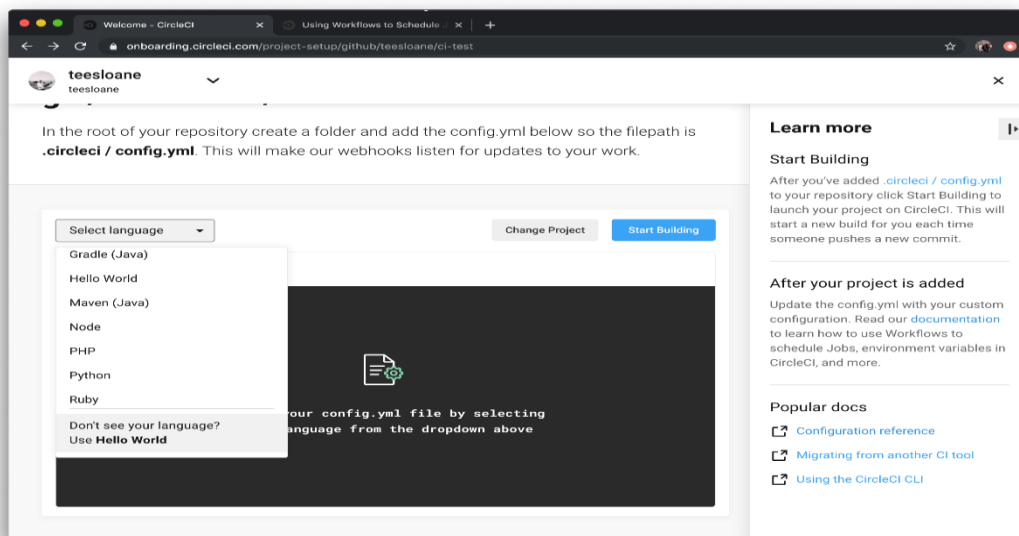


Figure 9 - choosing language to generate a YAML pipeline-CircleCI (CircleCI-start, 2020)


```

version: 2
jobs:
  build:
    docker:
      - image: circleci/<language>:<version TAG>
    steps:
      - checkout
      - run: <command>
  test:
    docker:
      - image: circleci/<language>:<version TAG>
    steps:
      - checkout
      - run: <command>
workflows:
  version: 2
  build_and_test:
    jobs:
      - build
      - test

```

Code 5 - config.yml-CircleCI (CircleCI-config.yml 2020)

Cons

- Migrating repository from Gitlab to CircleCI directly is not possible, the code must be imported in Bitbucket or GitHub first, where they are supported from CircleCI (Migrating from GitLab – CircleCI, 2020).
- No caching for docker images (Slant and Pipelines, 2020).

2.2.6 BuddyBuild



According to BuddyBuild's website, it's " a continuous integration, continuous deployment, and user feedback platform" (Buddybuild.com, 2020).

This tool is specialized to serve IOS applications projects after joining the XCode group, where the developer can build, deploy the app and know the users' feedbacks about it.

On the other hand, it doesn't support Android applications anymore since March 2018 and does not accept new customers (buddy build-apple, 2020).

It is integrated with the TestFlight tool, to make it easy to achieve Beta testing by inviting the users to test the IOS application (Apple Developer. 2020), and it Provides crash reporting, pointing to the line that caused the crash (Buddybuild.com, 2020).

BuddyBuild is integrated with any Git repository, whenever the changes are pushed to a version control repository hosting service, like Bitbucket, the ci tool will start its job, like preparing a secure build environment, starting building, unit test, UI tests, then preparing the application for Beta testing (Buddybuild.com, 2020).

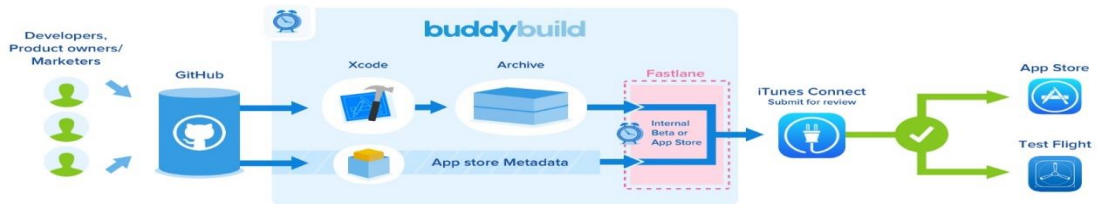


Figure 10 - BuddyBuild-Release process (BuddyBuild-Release 2020)

Cons

- According to Alternative.me, The build time is long (Tobies-BuddyBuild 2020).
- BuddyBuild doesn't accept new members, so currently there is no way to sign up.

2.2.7 Bitrise



CI/CD platform as service (PaaS), which means it provides the aide to the customer to build their pipelines without the complexity of preparing agents, virtual machines, or any kind of infrastructure that are usually needed to build the pipeline, this platform focuses on mobile application projects and can prepare the configuration of the project for any mobile platform (Bitrise - Mobile Continuous Integration and Delivery. 2020).

Reversed BuddyBuild, it supports both Android and IOS applications.

Its goal is automating the build, test, code signing, and deploying the mobile applications (Bitrise - Mobile Continuous Integration and Delivery. 2020).

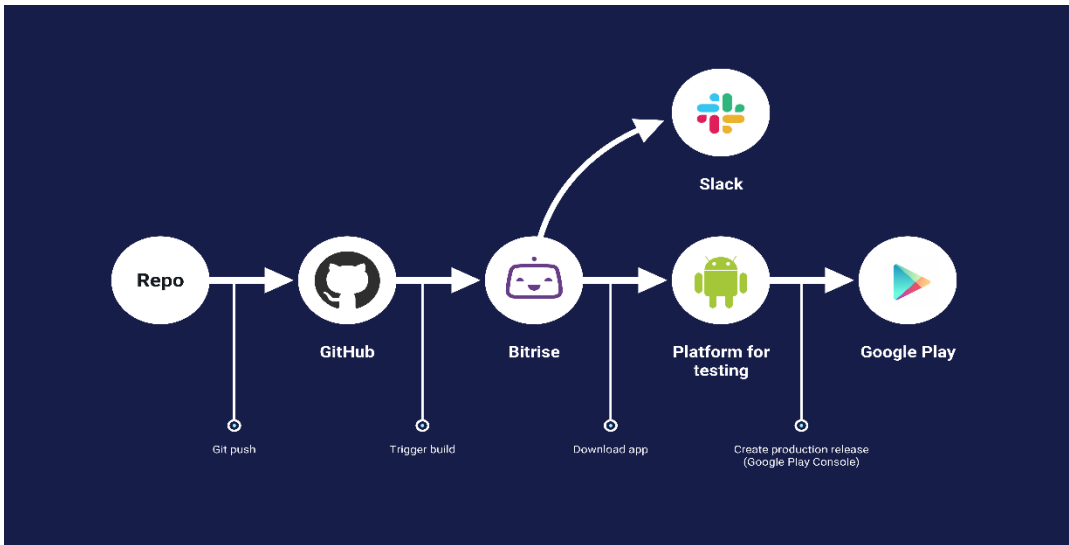


Figure 11 - Bitrise Architecture – Android apps (Startupcraft.io, 2020)



Figure 12 - Bitrise Architecture – IOS apps (Startupcraft.io-IOS, 2020)

To start, sign up in Bitrise by Gitlab, Bitbucket, GitHub, or using an Email, after that add a project, where Bitrise gives some choices to configure it, like privacy, branch, build configuration, so on.

Then it will automatically generate bitrise.YML code and start the first job, note that there are more configurations, in the workflow tab for each project.

The following Figures are examples about how the workflow for Android and IOS projects look like in Bitrise workflow editor.

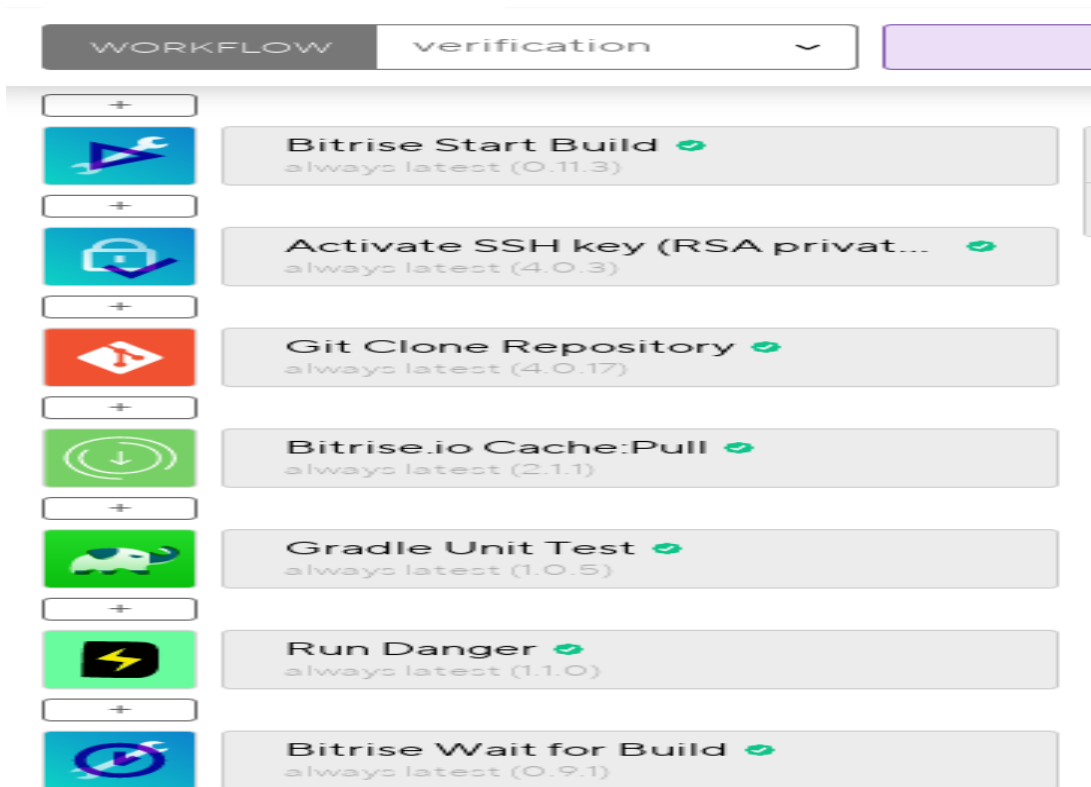


Figure 13 - Bitrise – verification workflow Android example (Droids Roids – Poland, 2020)

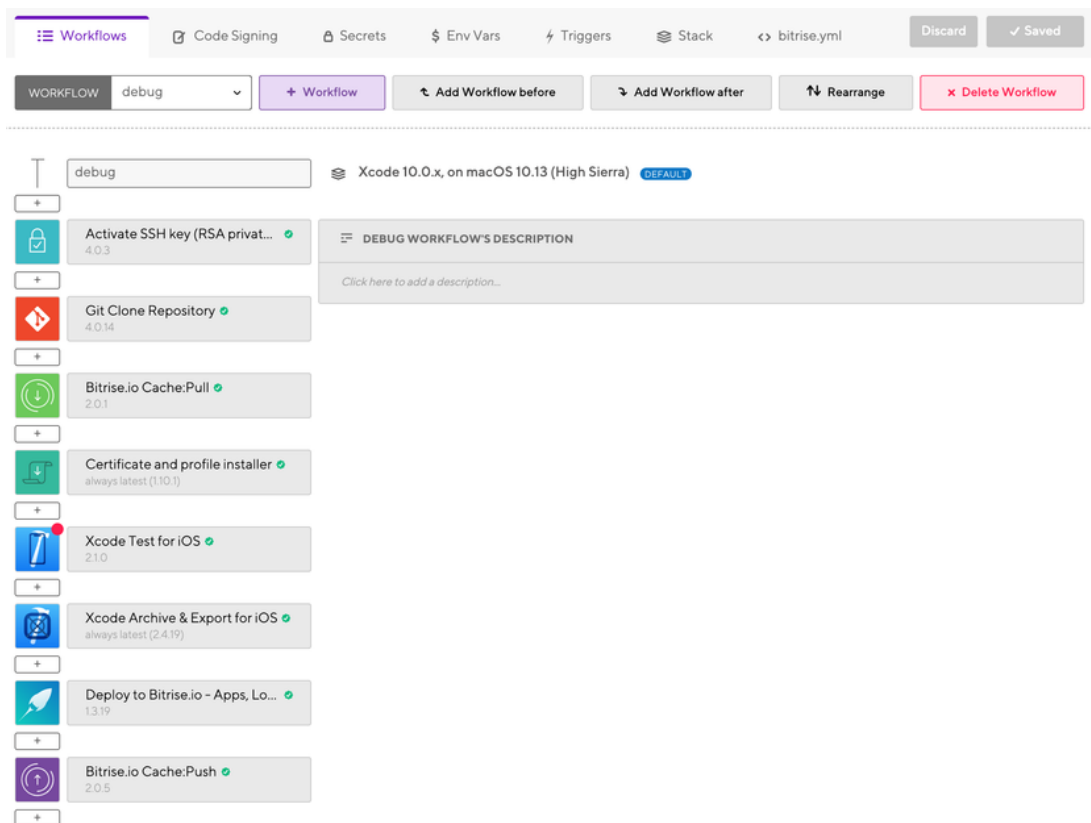


Figure 14 - Bitrise – Debug workflow IOS example (Savvy Apps., 2020)

The methodology of Bitrise is based on the virtual machines, whenever it starts a new build, a new VM will be generated to run the build then will be discarded after finishing its task (Bitrise-infrastructure, 2020), these VMs must be Mac OS or Linux (Bitrise-virtual-machines, 2020).

Cons

- Every time Apple changes anything, the user must deal with some troubles in IOS projects on Bitrise (Bitrise-Capterra, 2020).
- According to many Capterra’s reviewers, Bitrise Needs to speed up the builds, especially when it deals with IOS projects (Bitrise-Capterra, 2020).
- Settings start to be complicated when Bitrise deals with many projects (Bitrise-Capterra, 2020).

2.2.8 Drone



According to the Drone’s website, “Drone is a self-service continuous delivery platform for busy development teams” (Drone.io, 2020).

According to the readme.md file of Drone repository in GitHub “Drone is a Continuous Delivery system built on container technology” (GitHub-drone, 2020).

The uniqueness of this tool comes from its architecture, where Drone can be triggered as a container on the machine.

In addition, the plugins of this tool are Docker containers too (Slideshare.net., 2020).

This tool gives the possibility to install and configure its server and many Runners on the machine. The ability to run the builds in isolated containers made the work on Drone faster than other tools (Jane, 2020).

By following the formal tutorial, the user can find out in the server installation section that it is integrated with the different version controls management providers, like Gitlab, GitHub, and Bitbucket.

On the other side, in the Runner installation section, Drone provides many choices to pick the suitable Runners, like Docker, SSH, Digital ocean and Exec, where the user can install many of them (Drone-installation, 2020).

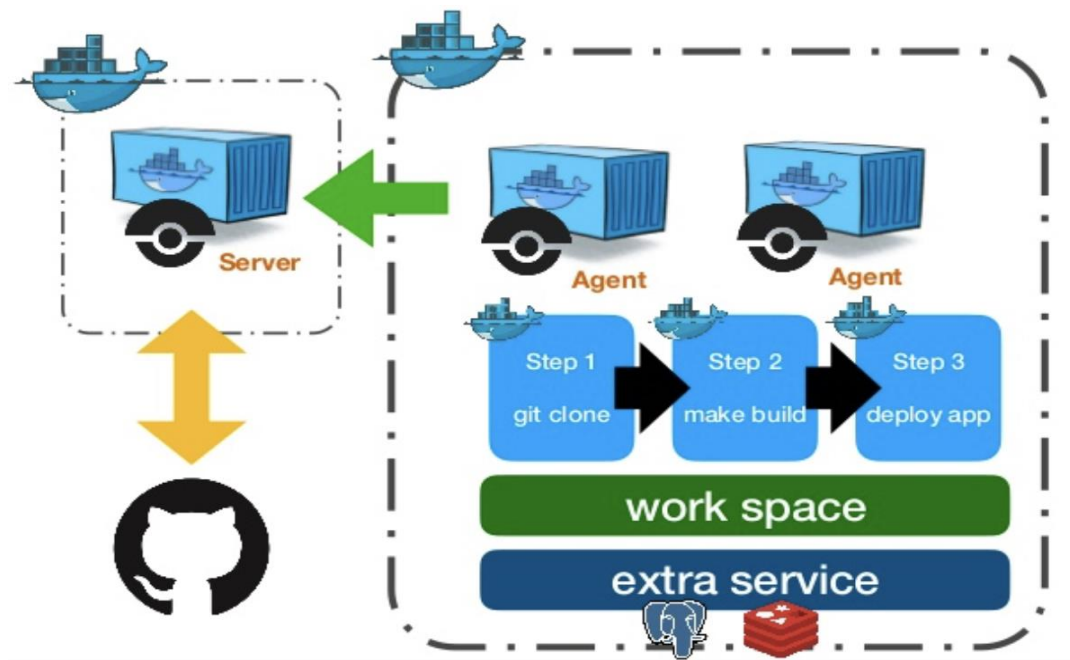


Figure 15 - Drone Architecture (Slideshare.net., 2020)

After finishing the setup process and connecting the Drone server with one of the VCS providers, `.drone.yml` file can be pushed to the root repository, which is a pipeline-as-code, then the role of Runners comes to execute this pipeline in isolated containers.

```
kind: pipeline
steps:
- name: test
  image: node
  commands:
  - npm install
  - npm test
services:
- name: database
  image: mysql
  ports:
  - 3306
```

Code 6 -.drone.yml simple example (Drone.io, 2020)

Cons

- There are no separation branches (drone-reviews, 2020).
- It's difficult to set up (Reddit-drone, 2020).
- According to many reviewers, Drone has very simple UI and needs improving (drone-reviews, 2020).

2.2.9 Bamboo



One of the Atlassian products, it is continuous integration, delivery, and deployment server (Atlassian., 2020).

Bamboo is integrated with the most important Atlassian products, like Jira which is an agile project management tool helps the teams to plan and manage the projects, as well as track issues and bugs (Jira, 2020), SourceTree which is Git GUI that helps to interact with the Git repositories using a graphical interface instead of the git commands lines (SourceTree, 2020), Bitbucket, and hundreds of other tools, and it's connected with different control version systems (Atlassian., 2020), also, Bamboo can support any language, and Amazon web services (Bamboo – Features, 2020).

One of its features that helps to speed up the job is the dedicated agents, where the DevOps team can dedicate the agents to run specific tasks, in this case, the agents will not be able to run other activities, then no more waiting for a free agent.

It supports builds based on Docker images, where the containers run as agents and gives the possibility to create a Docker image and push it to the registry (Bamboo – Features, 2020).

To get started with Bamboo, before installing it, make sure that the machine has JDK, then create a new directory called bamboo-home to save its data in a separate location from the installation directory, to avoid losing information when the Bamboo version is upgraded, then open <Bamboo installation directory>/atlassian-bamboo/WEB-INF/classes/bamboo-init.properties, and uncomment bamboo.home line, with providing the absolute path that was chosen for the bamboo-home directory (Atlassian-installing Bamboo, 2020).

Bamboo can be found on localhost:8085, then it's the time to add a repository, configure the first plan that is included one or many stages, and start the CI jobs.

The following Figure shows the architecture of the Bamboo's plan, where it's noticed the parallelism concept is implemented inside the sequential workflow.

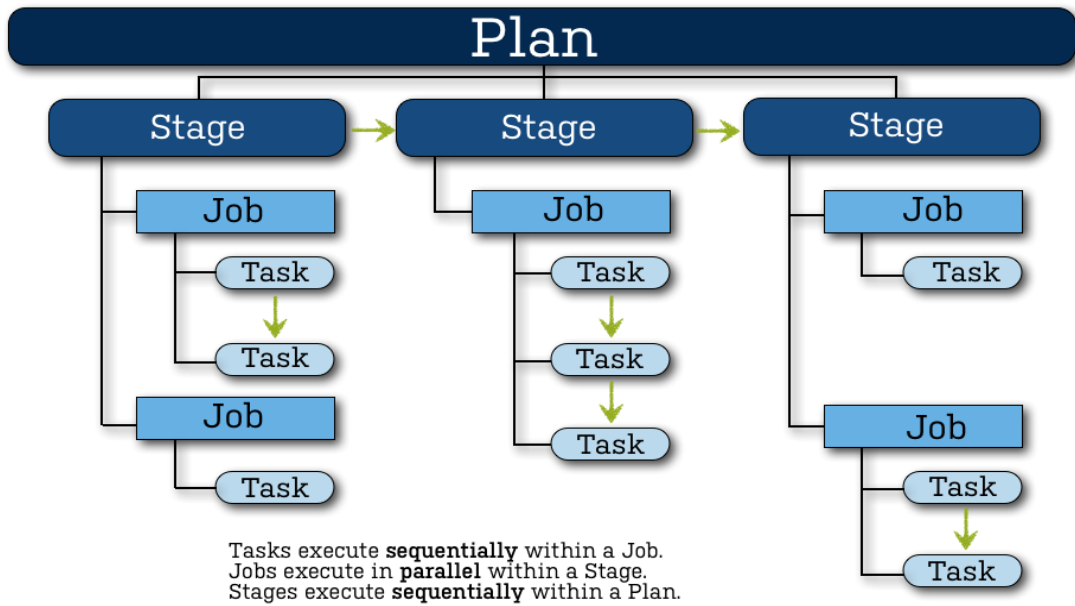


Figure 16 - Bamboo's Plan Architecture (Atlassian-Configuring plans 2020)

To configure the plan, there are two ways:

- UI: using a user interface, bamboo provides all the necessary configurations to generate one or more plans, to build, test, and deploy the project, send notifications, and much more.

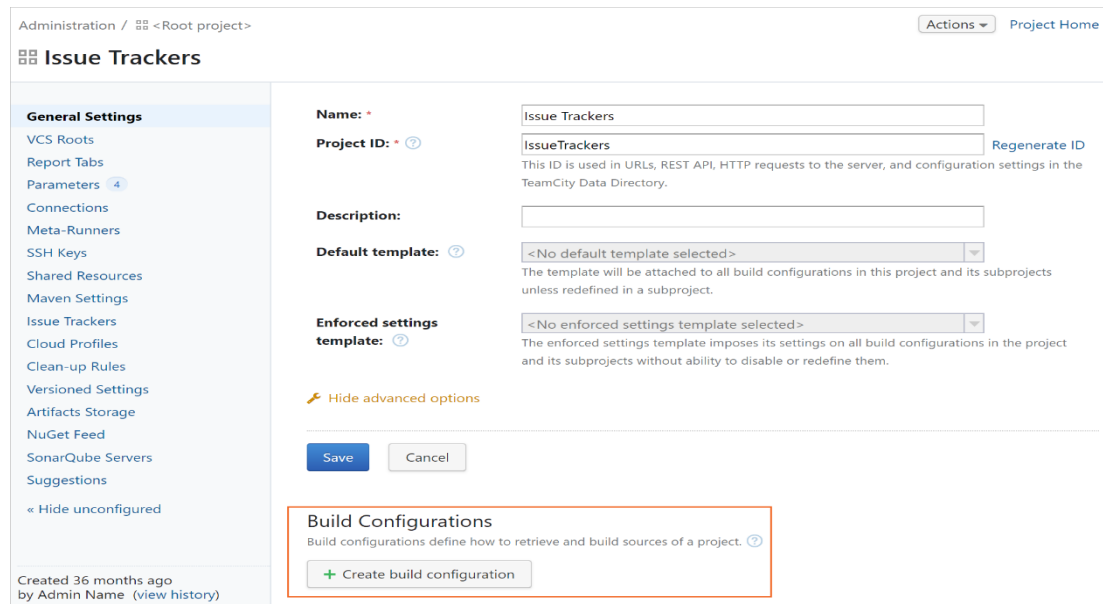


Figure 17 - Bamboo-Plan configuration (Atlassian-Configuring plans 2020)

- Configuration-as-code: or Specs, it is similar to the previous tools, which means the Plan can be configured as code using Java or YAML language, note that Specs in Java are more integrated (Bamboo-Specs 2020).

The following Figure is an example of how the configurations of a project look like a code.



Figure 18 - Bamboo-Plan configuration-as-code (Bamboo 2020)

Cons

- Java knowledge is necessary in the case of configuring an advanced plan as a code.
- According to many reviewers, Bamboo is not a rich system of plugins, if it was compared with Jenkins (TrustRadius-Bamboo, 2020).
- It is an expensive tool comparing with the other ones (TrustRadius-Bamboo, 2020).
- It is hard to understand the differences between stage, job, and task in the architecture of the plan and to determine the best way to use them in the projects (DeployPlace., 2020).

2.2.10 GitLab



According to the formal website “ it is a complete DevOps platform, delivered as a single application that provides everything you need to Manage, Plan, Create, Verify, Package, Release, Configure, Monitor, and Secure your applications” (GitLab., 2020).

In addition to the fact that Gitlab is an integrated version control provider, and code review, it is a CI/CD system that provides the ability to use just one tool to keep, display, build, test, and deploy the project.

It supports AWS EC2 service which is Amazon Elastic Compute Cloud service that provides virtual machines to build the pipelines, as well as the On-premise hosting or self-hosted which is the ability to trigger the pipelines in our own servers.

Gitlab helps to monitor the performance of the pipeline and shows the consumed time for every step.

The great feature about Gitlab is the clear and accurate charts and graphs that are generated automatically as results for the monitoring, as well as the friendly user interface.

In the Forrester Q3 2017 Report, Gitlab Rated as a leader in CI (Gitlab-Forrester, 2020).

It can work with all the operating systems, and support IOS, and Android projects.

To start Gitlab, register on the website, then add the repository, and to implement CI/CD tasks for the project, add a pipeline-as-code using YAML language to the repository.

Gitlab provides two ways to run the pipeline

- Runner: it’s a Docker container, a cluster of containers, a Virtual machine, or a Virtual private server, that can be installed on the machine then be configured to run the pipeline that was defined in the .gitlab-ci.yml file (Gitlab-runner, 2020).
- Docker Image: there is no need to any extra configuration to use this way, just need to put the suitable Docker image in the pipeline’s script then push the changes to the repository, where the pipeline will run inside the container of this image.

```
image: gradle:alpine
variables:
  GRADLE_OPTS: "-Dorg.gradle.daemon=false"
before_script:
  - export GRADLE_USER_HOME=`pwd`/.gradle
build:
  stage: build
  script: gradle --build-cache assemble
cache:
```

```

key: "$CI_COMMIT_REF_NAME"
policy: push
paths:
  - build
  - .gradle
test:
  stage: test
  script: gradle check
  cache:
    key: "$CI_COMMIT_REF_NAME"
    policy: pull
    paths:
      - build
      - .gradle

```

Code 7 - GitLab pipeline example (Gitlab-ci.yml, 2020)

Another important feature in Gitlab is the running on parallel for the steps that belong to one stage.

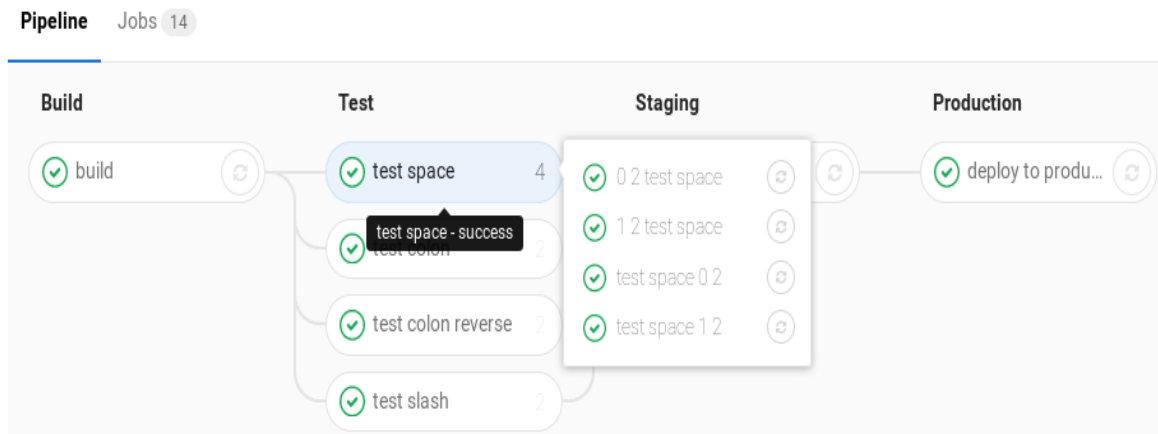


Figure 19 - GitLab pipeline graph (Gitlab-pipeline 2020)

Cons

- It's not a lightweight tool because it requires huge memory to do the tasks.
- Extremely slow builds using Docker images.

2.3 Comparing the Technologies

After referring to the latest CI/CD tools in the market and mention their most important related features to this project, in this section, the most appropriate tools will be compared, based on the project’s perspective, that were filtered by the comparison tables in Chapter 3 (section of Opportunity Analysis), to choose the tool that will be implemented during this thesis.

| | Teamcity | Bitrise | Gitlab |
|-------------------------------|----------|--------------------------------------|--------|
| Support Android projects | ✓ | ✓ | ✓ |
| Support IOS projects | ✓ | ✓ | ✓ |
| Support hosting On-premise | ✓ | ✗ | ✓ |
| Git VCS | ✓ | ✓ | ✓ |
| AWS EC2 or EKS Infrastructure | ✓ | ✗ | ✓ |
| Metrics system | ✓ | ✗ | ✓ |
| Jobs Parallelism | ✓ | ✓ | ✓ |
| Support Docker | ✓ | ✓ Needed 20-25 GB free disc space | ✓ |
| Friendly GUI | ✓ | ✓ | ✓ |

Table 1 - part of the comparison table for the filtered CI/CD tools

Although that Bitrise doesn’t support hosting on-premise, or AWS EC2 infrastructure, doesn't mean it’s out of the competition, because it provides all the support to build the pipeline concurrently using its ready agents, in other words, it offers a complete automated infrastructure for building, which means there is no need to make extra efforts to configure any infrastructure and it will generate automatically the pipeline-as-code in YAML language, in addition to the fact that is specialized CI/CD tool for the mobile application projects, as a result, Bitrise saves the teams’ time, but they will not have any control on the system, besides that it will be a very expensive tool in this case.

On other hand, Teamcity successfully covered all the needed requirements of this project, also it provides the teams the ability to control the system, but it is a very complicated and hard-

understandable tool, and need to configure a lot of settings for each project, which means it's consuming the time of the teams.

The last candidate is Gitlab that covers all the requirements too and offers the ability to control the system, besides the fact that it is a friendly, easy tool to learn and understand, and the cheapest one comparing with the last two tools.

In light of the above, Gitlab was selected to implement the CI / CD pipeline of TUI's projects.

3 Business Value Analysis

Value analysis is defined as “a process of a systematic review that is applied to existing product designs in order to compare the function of the product required by a customer to meet their requirements at the lowest cost consistent with the specified performance and reliability needed” (Rich and Holweg 2000).

It’s an approach that collects and examines the effects of various factors on the costs, and utility of the product/service that the company seeks to produce/improve in the business. In light of this search, the goal of value analysis is achieved by taking the optimized solution that reduces the production costs, increases the performance and reliability standards of product/service, and match the requirements of the customer as much possible.

3.1 New Concept Deployment

The NCD Model is the methodology that provides a new service/product to the market. The need for engaging this process comes from the changeable priorities and requirements of the customers, or because of the new available technologies that provide extra features and competitive prices to enhance the efficiency of the product/service and minimize the business’s costs (Business.qld.gov.au, 2020).

In this NCD Model, the latest technologies in CI/CD field will be highlighted, to develop the service and match the requirements of the customer.

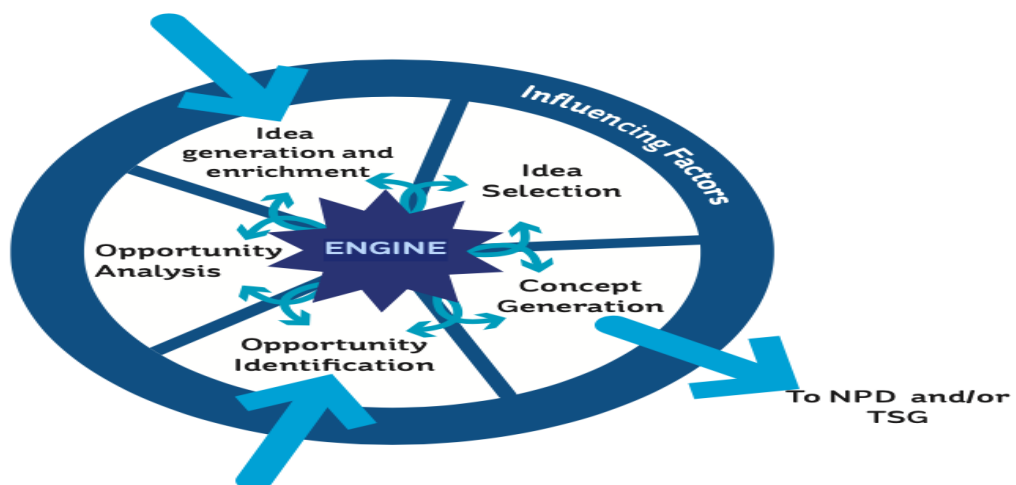


Figure 20 - NCD Model’s components (Koen et la., 2001, p.47)

It includes the five following elements

3.1.1 Opportunity Identification

The first step in the innovation process is defining the opportunities, which means it is necessary to have an extended and clear understanding of the new opportunities in the market, where it is not enough to know what should do, it is necessary to know how to do it and how to develop it, by having a clear vision about the latest technologies and the available opportunities in the market that can create new expectations for the future of the product/service (Dewulf, 2013, p.146).

after determining these opportunities, the needs to develop the product/service, and the priorities and preferences of the customer, the process can be started by adding new demanded features or reducing the costs using the best solutions for the business case.

The idea of this project comes from the desire of the customer, TUI company, to move all the repositories of the mobile applications from Bitbucket to GitLab, and replace Jenkins and BuddyBuild with one CI/CD tool, this step is important because there are several competitors for Jenkins and BuddyBuild in the IT market, that combine their features, offer advanced solutions, and make it easier to achieve the goals to speed up the continuous integration pipeline of the mobile applications, and provide an efficient metrics system that can monitor the performance of our pipeline.

3.1.2 Opportunity Analysis

The second step demands intensive efforts to make matches between the customer's requirements, and the existing technologies, by being aware of the expectations, limitations, and requirements of the customer, and collecting wide information about the potential tools, technologies, or methodologies that are related to our idea.

Then analyzing all this gathered information to have a clear vision about the most attractive, and most suitable solution for the business strategies, that can achieve the goals, you defined previously (Dewulf, 2013, p.146).

In the next two tables, the collected data of the existing technologies were converted to meaningful information for the company by making a comparison among all the CI/CD tools that were mentioned previously based on the requirements of the TUI project. Where the first column of the tables expresses the requirements.

Completed: ✓ Limited: ⚠ Poor: ✗

| | Jenkins | Buddy | TravisCI | Teamcity | CircleCI |
|-------------------------------|-----------------------------|-------|---------------------------------------|----------|---------------------------------|
| Support Android projects | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support IOS projects | ⚠ Limited infrastructure | ✗ | ✓ | ✓ | ✓ |
| Support hosting On-premise | ✓ | ✓ | ✓ | ✓ | ⚠ Limited with IOS projects |
| Git VCS | ✓ | ✓ | ⚠ just GitHub | ✓ | ⚠ Not integrated with Gitlab |
| AWS EC2 or EKS Infrastructure | ✓ | ✓ | ✓ | ✓ | ✓ |
| Metrics system | ✗ | ✓ | ✓ | ✓ | ✓ |
| Jobs Parallelism | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support Docker | ✓ | ✓ | ⚠ Not supported Docker with Mac OS | ✓ | ✓ |
| Friendly GUI | ✗ | ✓ | ✓ | ✓ | ✓ |

Table 2 - 1st comparison table of CI/CD tools based on the company's requirements

| | BuddyBuild | Bitrise | Drone | Bamboo | Gitlab |
|-------------------------------|------------|---|---------------|------------------|--------|
| Support Android projects | ✗ | ✓ | ✓ | ✓ | ✓ |
| Support IOS projects | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support hosting On-premise | ✓ | ✗ | ✓ | ✓ | ✓ |
| Git VCS | ✓ | ✓ | ✓ | ✓ | ✓ |
| AWS EC2 or EKS Infrastructure | ✗ | ✗ | ⚠ Just ECS | ✓ | ✓ |
| Metrics system | ✓ | ✗ | ✓ | ✓ | ✓ |
| Jobs Parallelism | ✓ | ✓ | ✓ | ✓ | ✓ |
| Support Docker | ✗ | ✓ Needed 20-25 GB free disc space | ✓ | ✓ | ✓ |
| Friendly GUI | ✓ | ✓ | ✗ | ⚠ Complicated | ✓ |

Table 3 - 2nd comparison table of CI/CD tools based on the company's requirements

Highlight on some of the previous requirements:

- Support On-premise hosting or self-hosted this means; a tool must provide the ability to build the pipelines in the company's servers according to the project's needs (for example Mac servers).
- Integrated with Git VCS means the tool gives the choice to reach to the repositories in any git version control provider. (for example, Bitbucket, Gitlab, and GitHub).
- Support AWS EC2 infrastructure or EKS means the tool must be integrated with Amazon Elastic Compute Cloud service or Elastic Kubernetes service, that provides virtual machines to build the pipelines.
- Metrics System means the demanded system to measure the speed and the performance of the pipeline.
- Jobs Parallelism is one of the most important features that help to speed up the pipeline, where the steps trigger concurrently and provide distributed builds.
- Support Docker, to use the docker containers as agents to trigger parts of the pipelines where it provides a clean and isolated environment for CI/CD builds.

As a result of analyzing these tables and filtering the choices in Chapter 2 (section of Comparing the Technologies), the most suitable tool was Gitlab, where it can achieve the goals of this thesis.

3.1.3 Idea Genesis & Enrichment

This process involves the birth, developing, changing, and editing of the idea frequently, before reaching the maturity stage and becoming a tangible idea.

The birth of the idea may occur due to internal or external factors, like launching a new technology in the market that has better performance, cheaper, or can minimize the consumption time of doing our business (Dewulf, 2013, p.147).

The origin of the project's idea starts because of two main factors, external one represents the wide variety of CI/CD tools, that were launched, and developed in IT market in the last few years, as a result to the incremental importance of the DevOps field, and internal one represents the need of the company to enhance the CI pipeline of TUI's mobile applications, and the desire of developers to deal with more flexible, faster, and easier CI/CD tool.

Since the DevOps team, managers, and developers discussed this idea many times, and after many changes in the requirements, the idea was grown.

3.1.4 Idea Selection

At this stage, it is important to determine the final idea and the opportunity of the whole process, because covering all ideas and available opportunities is hard, and consuming time, efforts, and budget.

This activity is very sensitive and critical because it must achieve the most business and consumer value (Dewulf, 2013, p.147).

In this thesis, the main idea and the opportunity were defined after examining the current state, and the performance of the pipeline, then studying the possible solutions that were offered by the DevOps market and comparing them.

the project's idea represents speeding up the CI pipeline of Android and IOS applications and monitoring its performance using metrics system, after replacing Jenkins & BuddyBuild tools with just one CI/CD tool.

3.1.5 Concept and Technology Definition

The final element in NCD Model represents studying the business case that is affected by many factors, like "market potential, customer needs, investment requirements, competition analysis, and project uncertainty" (Dewulf, 2013, p.147).

After deciding the idea, the optimal tool, and the suitable way to implement the solution, the business case of the TUI project will be defined through the discussing with managers and taking care of the previous factors.

3.2 Value Creation

After choosing the idea, and the tools that can help to complete the task must be aware of the solution’s value that must be created at the end of the process, and make sure this new value makes the state of the project better than the previous one and its benefits are bigger than its costs and risks, and match the requirements of the customer very well.

3.2.1 Value

This concept can be defined, according to the perspective of the engineers and economists as following, it’s any derived benefit of product, service, or activity that satisfies the needs of the customer, regardless if this benefit is tangible or not (Haksever, Chaganti, and Cook, 2004, p.292).

The expected value at the end of this project must be reducing the consumption time on the following release lifecycle’s four steps that are shown in the next Figure. Where the Build, Test, Release, and Deploy steps can be covered using the CI/CD pipeline and providing a monitoring system for the performance of this pipeline.

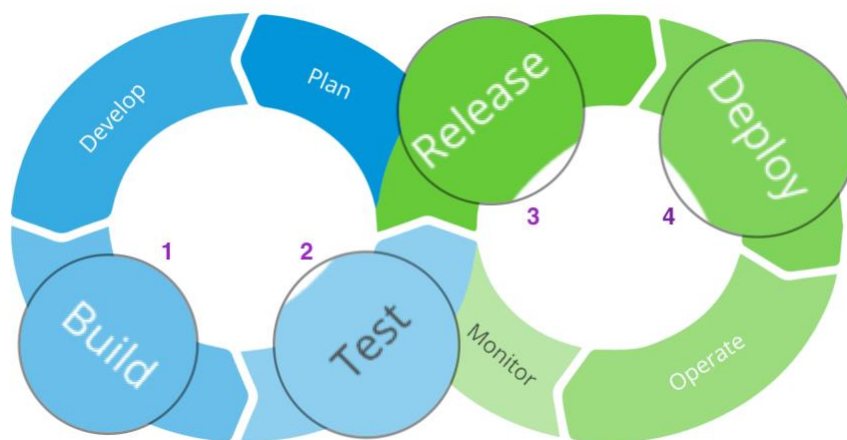


Figure 21 - life cycle of the mobile application with focusing on CI/CD steps

3.2.2 Customer Value

“there are two aspects to customer value: desired value and perceived value. Desired value refers to what customers desire in a product or service. Perceived value is the benefit that a customer believes he or she received from a product after it was purchased” (Shanker, 2012).

According to the last definition, the customer value can be divided into two stages, before and after delivering the product/service.

The Customer’s desired and the perceived value were clarified, in Figure 22.

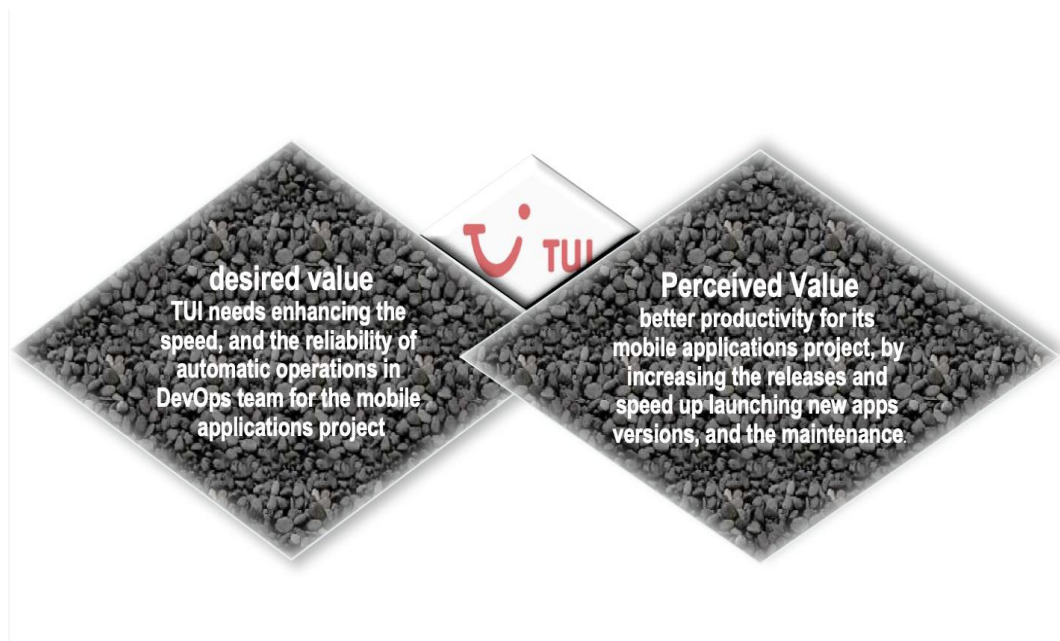


Figure 22 - Desired & Perceived Value of our customer TUI in this project

3.2.3 Longitude Perspective of Value

After describing the problem and the values that must be added to the end of the work, another concept that affects the customer must be analyzed, which is the set of benefits and sacrifices of the customer along the process.

Tony Woodall considered that can divide these benefits and sacrifices into four stages, starting from the pre-purchase stage to after use/experience one.

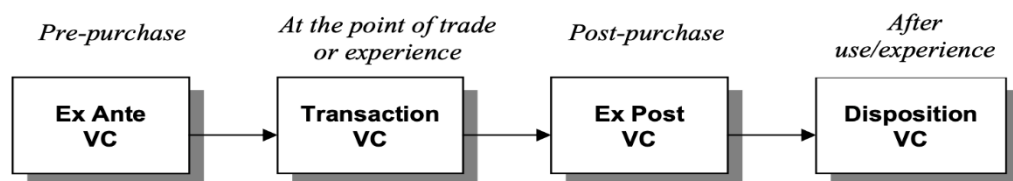


Figure 23 - A Longitudinal Perspective on VC (Woodall, 2003)

In Table 4, the potential benefits, and sacrifices of COCUS & TUI along this project were categorized.

| Stages | Benefits | Sacrifices |
|----------------------------|--|--|
| Pre-purchase | <ul style="list-style-type: none"> ☺ Knowledge. ☺ Strategic benefits. | <ul style="list-style-type: none"> ☹ Human energy. ☹ Search costs. ☹ Efforts. ☹ Time. |
| Point of experience | <ul style="list-style-type: none"> ☺ Functional benefits. ☺ Results for the customer. | <ul style="list-style-type: none"> ☹ Opportunity costs. ☹ Time. ☹ Human energy. |
| Post-purchase | <ul style="list-style-type: none"> ☺ Product benefits. ☺ Operational benefits. ☺ Logistical benefits. | <ul style="list-style-type: none"> ☹ Delivery and installation costs. ☹ Training and maintenance costs. ☹ Effort. |
| After experience | <ul style="list-style-type: none"> ☺ Convenience. ☺ Utility. ☺ Appreciation from users. | <ul style="list-style-type: none"> ☹ Market price ☹ Costs of repair |

Table 4 - Benefits & sacrifices of the value analysis

3.3 Value Proposition

This concept refers to the promise the company has made to the customer to provide a product/service including specific features and characteristics, then this means the expectation of the customer to receive this product/service with the same features that they determined previously.

The value proposition of this dissertation is a service offered for TUI’s mobile applications, this service represents finding the optimal CI/CD tool for the mobile app project, including developing CI/CD pipeline that speeds up the release process, in addition to, monitoring its performance, then gaining better quality and reliability for the TUI's mobile applications.

This thesis may be an optimal solution or helpful for similar projects, considering the limitations and the requirements for each project.

3.4 Quality Function Deployment (QFD)

This system was developed by Professor Yoji Akao. It is a management tool included a set of tools provides the ability to determine the Customer Requirements and the engineering characteristics to develop a product/service, this tool helps to build a strong relationship between the client and the company, to provide the demanded product/service after making the trade-offs between the customer requirements and the technical requirements (Isoconsultantpune.com, 2020).

Figure 24 represents the first phase of the system.

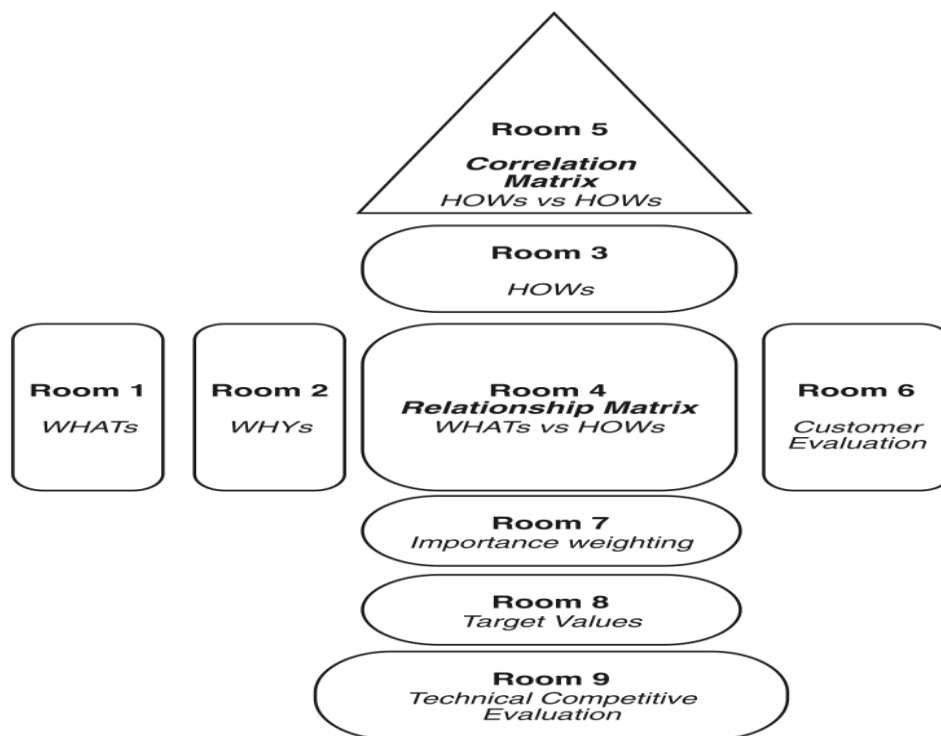


Figure 24 - House of Quality for QFD (Bouchereau and Rowlands, 2020)

To understand how this tool is working, each component/room was explained in this diagram:

- Room 1: or customer requirements that refers to (WHAT) the customer needs in the demanded product/service.
- Room 2: or importance to the customer that refers to the priority of each requirement.
- Rooms 3: or engineering characteristics refer to (HOW) the company can make this product/service.
- Room 4: or central matrix/relationship matrix that shows the strength of the relationship between the customer requirements and engineering characteristics.

- Room 5: or correlation matrix that shows the strength of the relationship between the engineering characteristics.
- Room 6: or customer evaluation is a customer survey that shows if the company could achieve each customer requirement.
- Room 7: or importance weighting is the sum of each column value that shows the most important engineering characteristic.
- Room 8: or target value is a design target set for the engineering characteristics.
- Room 9: or technical evaluation is an engineering survey and test results that evaluates the work of the company for each engineering characteristic.

To clarify the goal of this project and make it easy to produce the demanded service, it's important to collect the non-functional requirements (customer requirements) in other words what the features of the service that the customer wants at the end of this internship, then in the light of these needs, the functional requirements (technical requirements) were determined, this means how the customer requirements can be achieved.

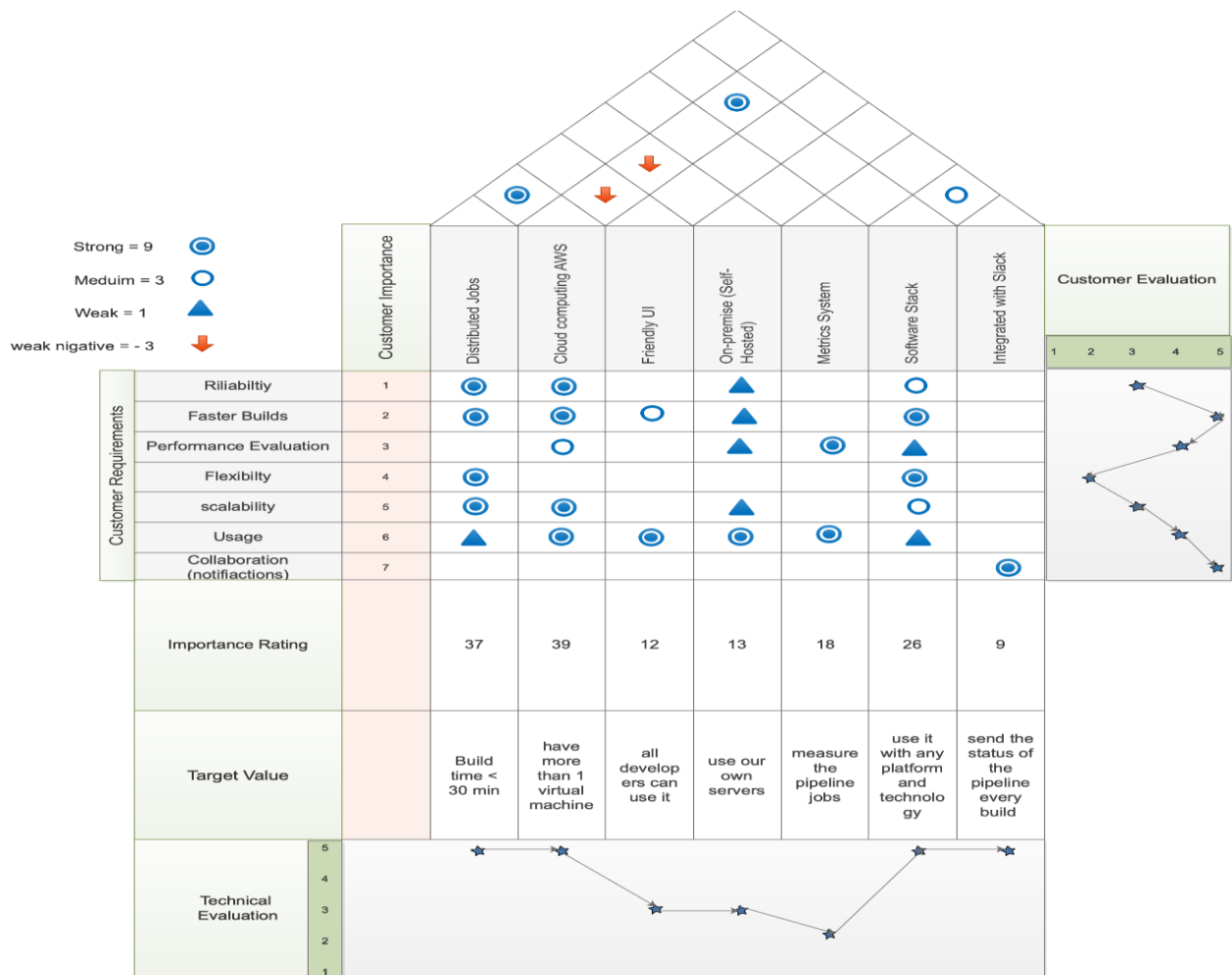


Figure 25 - the house of quality of TUI's CI/CD pipeline mobile applications project

As shown in Figure 25:

- In the customer importance room, the highest priority for the customer is choosing a CI/CD tool that guarantees the reliability of our pipeline, and the lowest priority is choosing one that sends notifications about the status of the pipeline.
- In the importance rating room, the most important characteristic is cloud computing AWS then distributed jobs, and the less one is the integration with Slack or Email.
- According to the technical evaluation, the company does relatively poorly on the characteristic of the Metrics System, on the other side it does well of Distributed Jobs, Cloud Computing AWS, Software Stack, and integration with Slack or Email.
- According to the customer evaluation, TUI thinks that the company does worse in term of flexibility, and does the best in terms of Faster Builds, and Collaboration.

4 Design

This chapter will display the Non-functional requirements that were collected from the client, functional requirements, and implementing the chosen solution after comparing it with the alternatives' methodologies.

4.1 Non-Functional & Functional Requirements

The Non-functional requirements were collected to understand TUI's expectations about the final desired service, where they expect a CI/CD tool implements CI/CD pipelines for their mobile applications project, where these pipelines included the following characteristics:

- Reliability (Availability): the chosen tool must work stably regardless of the conditions that may affect it.
- Performance (Speed): Faster builds which mean the tool must guarantee running the pipelines with less than 30 minutes for each build
- Functionality (Capability): Performance Evaluation means the CI/CD tool can monitor the performance of the pipeline.
- Supportability (Flexibility): it is important to use a flexible tool, where it supports different technologies and environments as much as possible and provides the ability to manage and extend the pipeline in an easy way for many projects.
- Performance(Scalability): the pipelines are not fixed pieces of code, which means we need the possibility to add extra tasks for the pipeline depending on the project, and in this case, the extra memory and more powerful CPU will be needed.
- Functionality: Collaboration offers a way to interact with teams automatically, to inform them about the results and failed cases in the system or the pipelines.
- Usability: the chosen CI/CD tool can be used by anyone of the teams easily without limitations and complication, and it has a simple, friendly, and clear user interface to make the interactions easier.
- Portability: the selected solution can be accessed using many machines and systems
- Security: the tool is protected against attacks and hackers.

The next functional requirements explain the tools' behaviors that the client expects:

- Distributed Jobs: the pipeline's tasks can run in several agents(machines), which means jobs of the pipeline will work concurrently, and this will reduce the consumed time, and support the reliability and the scalability of the pipeline.

- Cloud computing AWS: the tool must be integrated with Amazon Elastic Compute Cloud service or Elastic Kubernetes service that provides virtual machines to run the pipelines in, as well as it gives the demanded memory for implementing the builds.
- On-premise hosting or self-hosted: the tool must support self-hosted, where it provides the ability and flexibility to configure and build the pipelines in the company's servers according to the project's needs (for example Mac servers).
- The Metrics System is the demanded system to measure the speed and the performance of the pipeline in order to help to improve it.
- Software Stack: the tool must support multiple integrations (for example Git VCS, Docker tool) for IOS and Android projects.
- Integration with Slack or email: the tool can send notifications for teams using the Slack messaging platform or by email to inform them about the state of the pipeline, and this is an implementation for the collaboration requirement.

4.2 Use-Case Diagram

Figure 26 shows the use-case diagram prepared to have a clear vision about the whole process of the project and to highlight the teams' interactions with it, where the DevOps team interacts with the system by adding the CI/CD pipelines, and enhancing their performance, by developing them, meanwhile, the Servers run these pipelines.

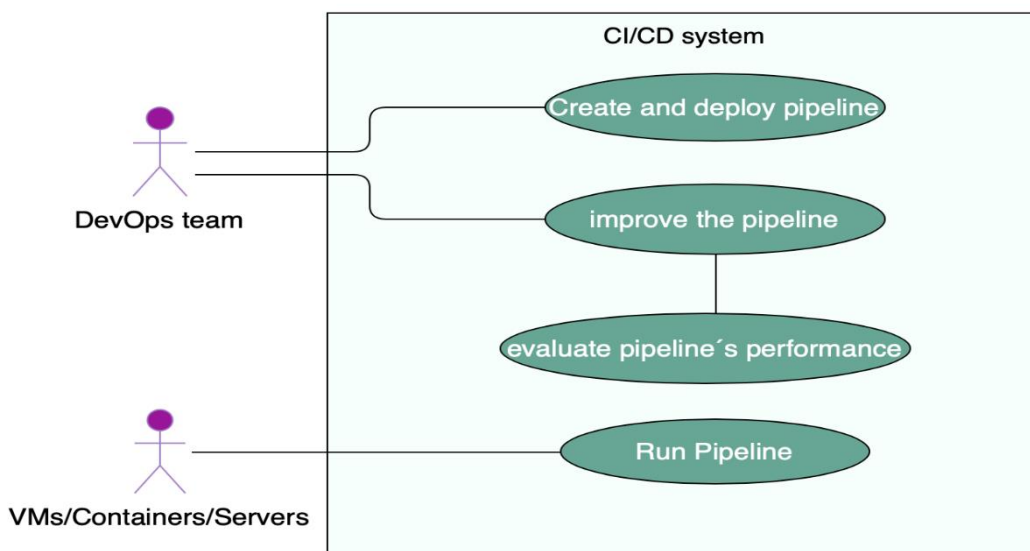


Figure 26 - Use case diagram for CI/CD system

4.3 Activity Diagram

Figure 27 displays the workflow of the release life cycle for Android/iOS apps, whenever Development or DevOps team commit changes:

- Gitlab CI triggers the pipeline to build/test the project, and in case of completing the process, a Beta version of the app would be available in Appcenter, where the QA team will test the application manually before publishing it to apps markets, otherwise the Developers/DevOps team would recommit changes to fix the issues.
- On the other side, Gitlab starts calculating the consumed time for each job, and at the end of the process, it displays graphs of success cases, the duration of the last 30 commits, and other statistics. Based on these graphs, the DevOps team can decide if the pipelines' performance is optimal or requires some improvements.

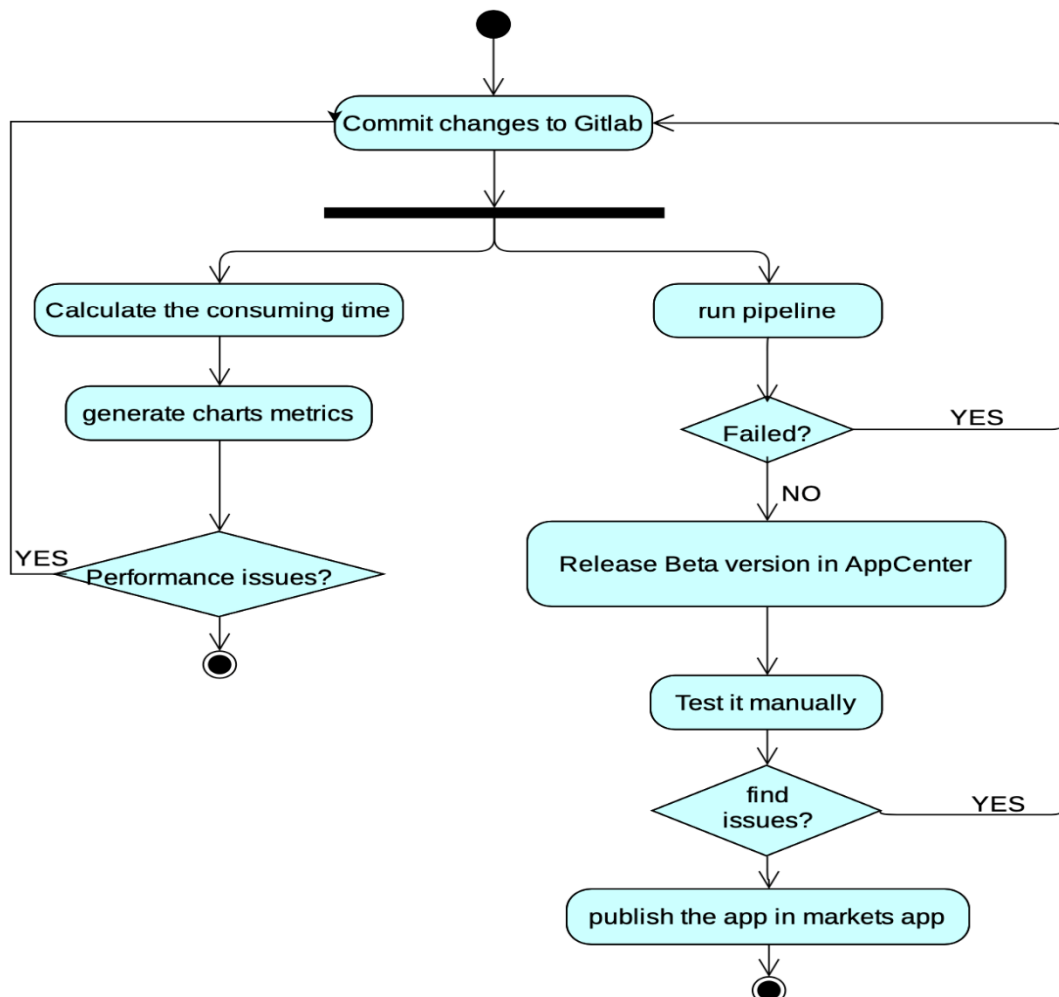


Figure 27 - Activity Diagram for release life cycle

4.4 CI/CD Pipelines diagrams for Android & IOS Apps

Figures 28, and 29 show the workflow of the CI/CD pipeline for Android and IOS apps.

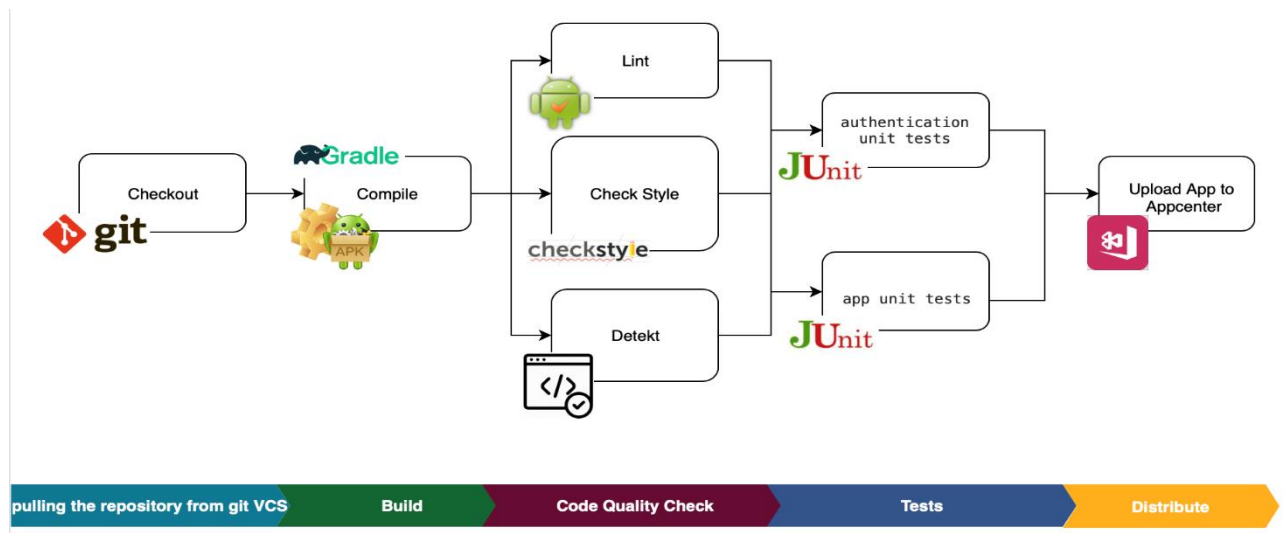


Figure 28 - CI/CD pipeline for TUI's Android application

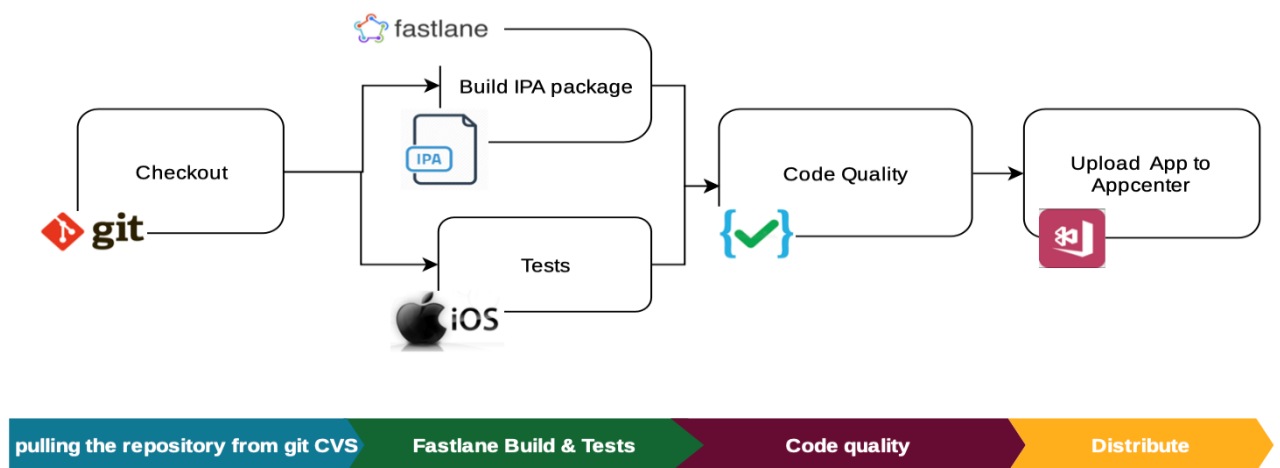


Figure 29 - CD/CD pipeline for TUI's IOS application

The pipelines' workflow starts with pulling the project from a Git version control system, which requires authentications, but in the case of Gitlab CI, this step was skipped because the repositories are stored in Gitlab and they are automatically checked out.

All the following tasks were generated by developers, where the Android project's tasks use a build automation tool called Gradle (Releases, 2020), and the IOS project's tasks use Fastlane tool (fastlane, 2020).

In Figure 28, Android's pipeline starts with compiling and generating the APK(JAR Android Package) package the project, then checking the code quality, this stage contains three jobs in parallel, Lint, which is scanning the source code for the potential bugs (Android-Lint, 2020), Check style, works to verify Java code standards without human intervention (checkstyle, 2020), and Detekt to analyze the Kotlin code (detekt, 2020), then unit tests stage to verify the logic of units in the project's source code (Android-unit-test, 2020), then upload the application (APK file) to Appcenter (Visual Studio, 2020) to allow QA team tests the application within the company's environment before publishing it to the final user.

In Figure 29, IOS's pipeline starts with two jobs in parallel, one of them to compile and test NO-SV project, which is one of the 11 apps in the source markets, and the other one to compile and generate IPA(iPhone Application Archive) package for any of the 11 applications, then Code quality stage(can be merged with tests job) to check the quality of the code like maintainability, reliability, and security, and generate reports allows the Developers to define if the low quality causes the failed cases (Disputesoft.com., 2020), the final stage of the pipeline's lifecycle is also uploading the application to Appcenter (can be merged with build job).

4.5 Component Diagram

Figure 30 includes the system components, and their interactions, where the responsibility of each component can be shorted as following, Gitlab is responsible on storing the source code, pipeline script, artifacts, and analytics charts, Docker and AWS components run the Android's pipeline, while Mac servers run the IOS's pipeline, and Appcenter stores the apps' Beta versions.

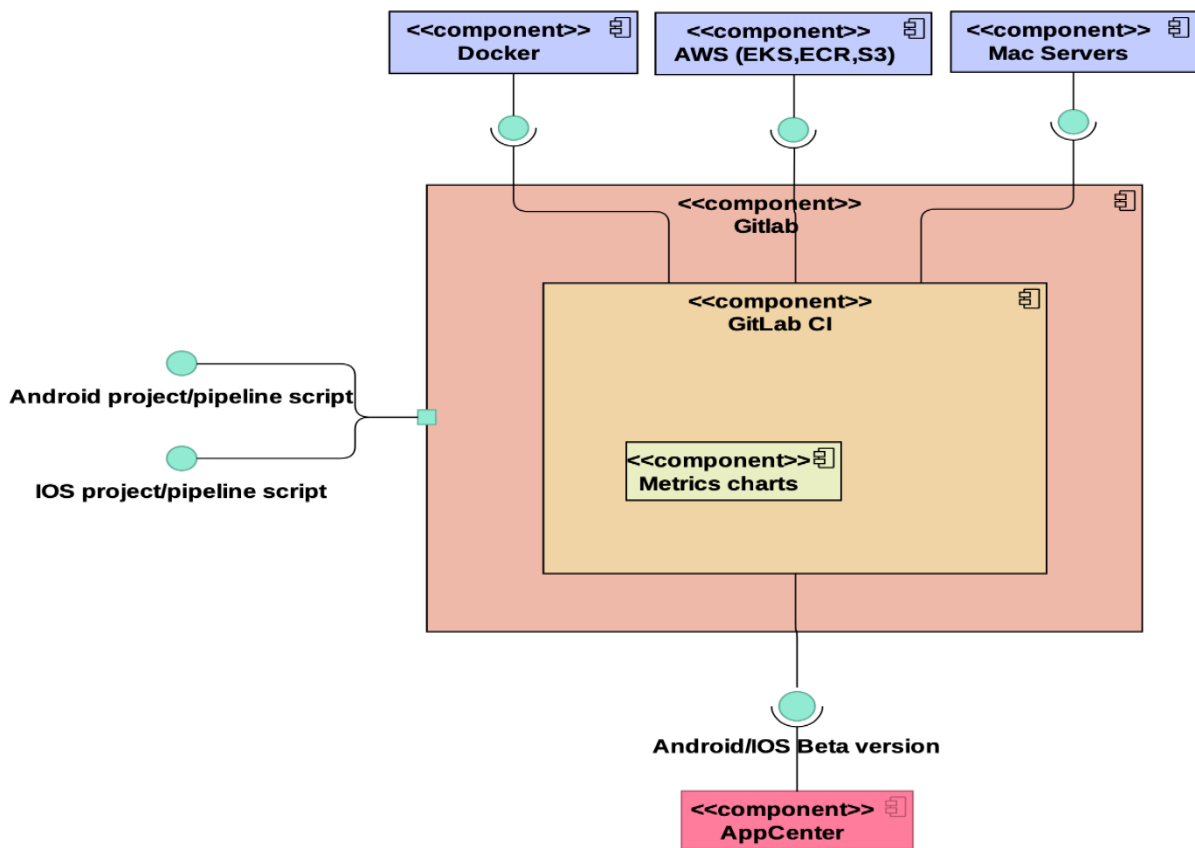


Figure 30 - Component Diagram for CI/CD tool

4.6 Alternatives and Selected Solution

Gitlab CI uses Runners to run the pipelines.

These Runners may be virtual machines, servers, Docker containers, clusters, or machines (GitLab-runner-doc, 2020). and since Gitlab provides different ways to configure the Runner, based on the requirements of the project and regarding the available resources in the company like AWS services, three suitable ways were nominated for the Android project and will be filtered to implement one of them as a final solution, and just one solution for the IOS project because of the availability of physical servers within the company.

These configurations will be achieved in the project page >> Settings >> CI/CD >> Runners.

4.6.1 Shared Runners & Docker image (Android)

This method based on enabling the shared runners, that are provided from Gitlab.com free and powered by Google Cloud Platform (Gitlab-Doc, 2020) but these runners don't support android projects, this problem may be solved by Docker image that includes the Android environment's requirements, like SDK, it is stored in Docker Hub registry, and pulled in the pipeline.

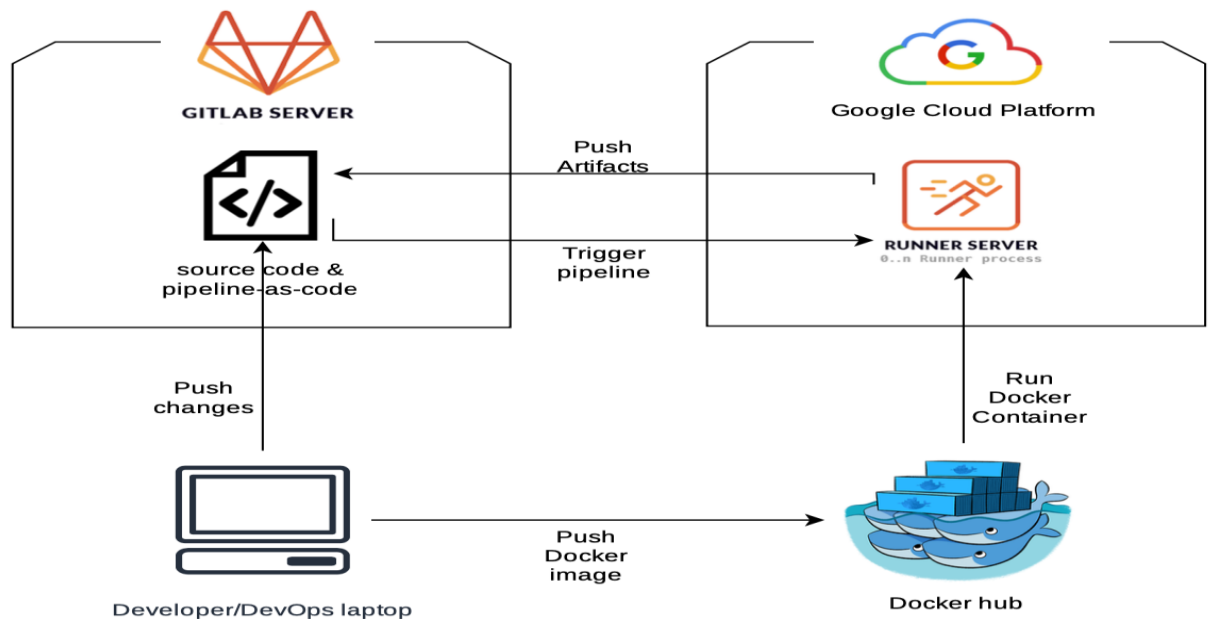


Figure 31 - Architecture of shared Runners & Docker Image solution

This solution is limited and will cause many failed cases in the pipeline, where the Runners are (n1-standard-1 instances) with 3.75GB of RAM (Gitlab-Doc, 2020) while the required memory to complete all jobs is around 16 GB.

4.6.2 AWS EC2 instance & registering Shell Runner (Android)

Instead of the shared Runners, it's possible to use EC2 service from AWS to create an instance (virtual machine) with required memory for triggering the pipeline, install Gitlab Runner and the Android environment's requirements on this machine and register Shell Runner for achieving the collaboration between Gitlab and the instance.

After hiring the instance as Runner on Gitlab, it can run the created pipeline of the Android project with tags' names, these tags used to associate the pipeline's job with the runner.

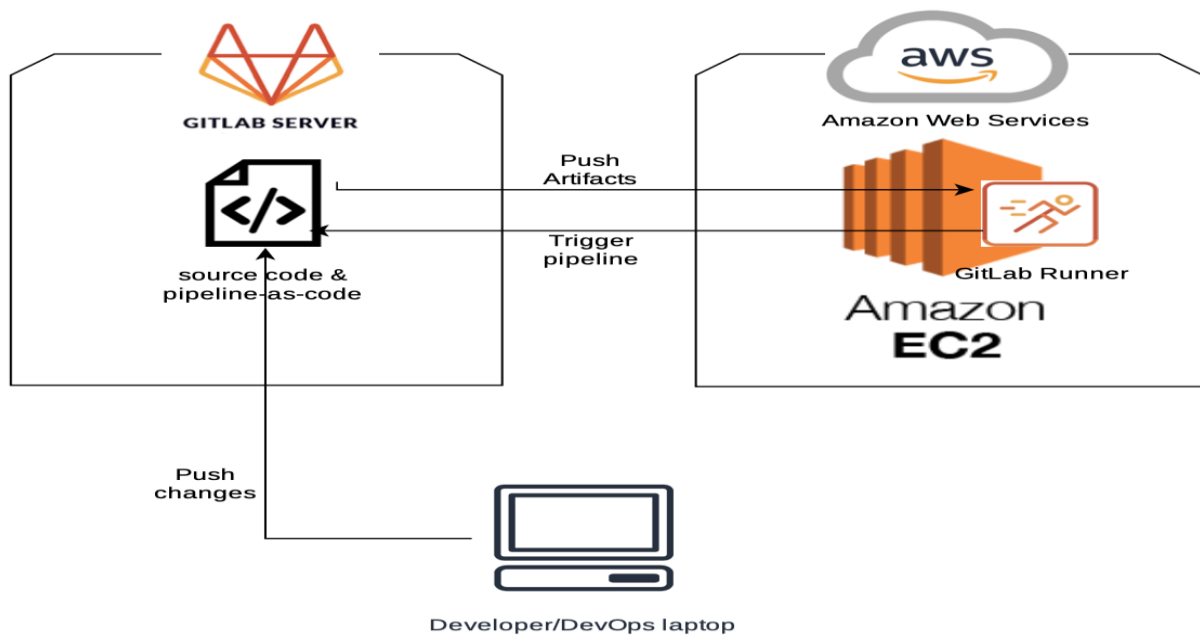


Figure 32 - Architecture of AWS EC2 instance & registering shell Runner

This solution is not the optimal one, due to the need to create many instances to build multiple projects, this project contains 11 Android apps, which means that creating multiple servers on AWS is financially harmful. On the other hand, it's not a reliable approach to use one instance to trigger the pipelines, because if this machine breaks down, all the work will be stopped.

4.6.3 Kubernetes cluster & Docker image (Android)

To avoid recent problems, Kubernetes was chosen to manage and deploy containerized applications (Docker container) on the machines (Kubernetes-aws, 2020).

In short, the user can create a cluster in Kubernetes, which is a group of nodes (virtual machines/workers/servers/Runners), then specifies the minimum and the maximum number of these nodes that Kubernetes can create or kill on behalf of the user in this cluster, based on the number of running pipelines, and the number of running jobs in parallel per pipeline.

This solution can be implemented using AWS EKS service (elastic Kubernetes service) to create the cluster and configure it, Docker image to prepare the Android environment's requirements, where Docker containers run in the nodes, AWS ECR service (elastic container registry) to store this image (ecr-aws, 2020), and AWS S3 service (Simple Storage Service) to store the builds' caches (AWS-s3, 2020).

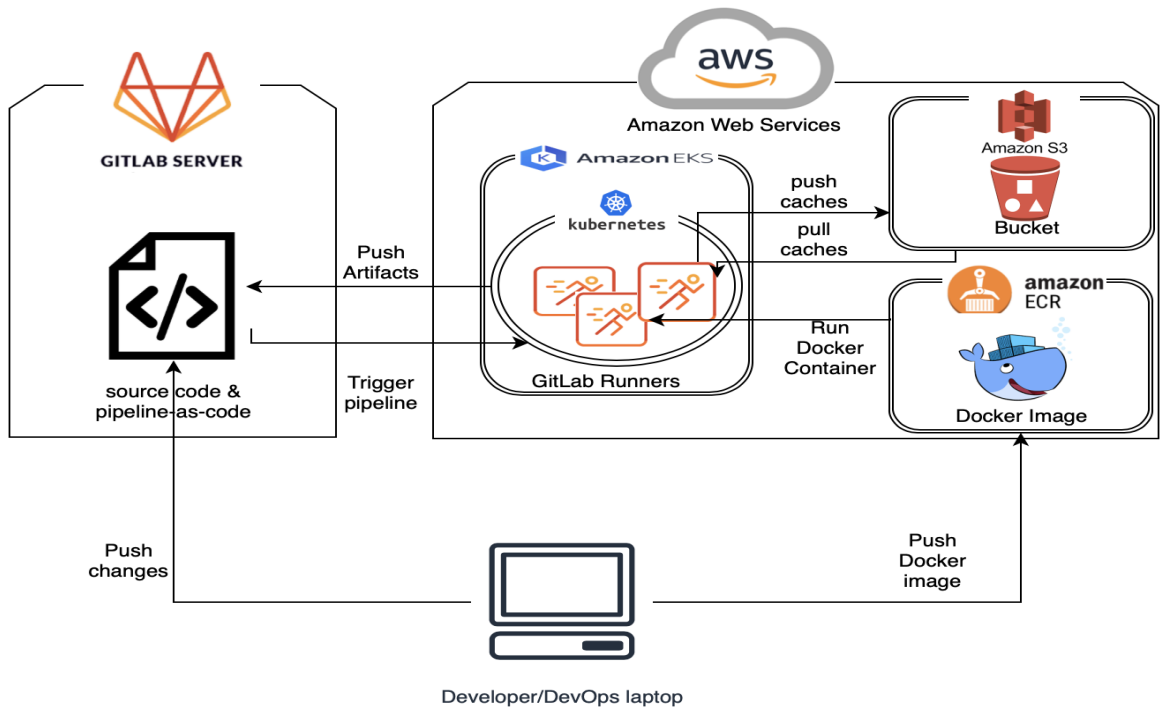


Figure 33 - Architecture of Kubernetes cluster & Docker image

This approach is adopted because it does not have memory limitations and can reduce the expenses of multi-projects-builds in many servers.

Considering 1000 builds per month, Figure 34 shows the prices of the instances, in the case of EC2 and EKS services, where it's clear that EKS's expenses are cheaper than the EC2's ones, even with more powerful c5.4xlarge instance (AWS-EC2 on demand, 2020).

| | EC2 | | EKS | | |
|---|--|-------------------------------|-----------------------------|-----------------------------|------------------------------------|
| | Pipeline | Build | Node | Node | Cluster |
| Instance types | t3a.medium | c5.2xlarge | c5.2xlarge | c5.4xlarge | - |
| Number of builds in instance/hour | Depends of selected markets | 1-2 | 2 | 4 | - |
| Expected number of instances for 1000 builds | Depends of selected markets | 700-1000 | up to 500 instances | up to 250 instances | - |
| Expected maximum cost (on-demand) for 1000 builds | Consider an half of number of builds, because most markets have 2 builds | $0,388 \times 1000 = 388 \$$ | $0,388 \times 550 = 194 \$$ | $0,776 \times 250 = 194 \$$ | $0,10 \times 24 \times 30 = 72 \$$ |
| | | $0,0432 \times 500 = 21,6 \$$ | | | |

Figure 34 - prices of EC2 & EKS based on the project

4.6.4 Mac Servers (IOS)

Due to the company's reliance on its own servers in terms of IOS project, this solution has been selected to run the IOS pipeline, where the servers contain the IOS environment's requirements, and AWS S3 bucket uses to store the caches of the dependencies to pull them later in many servers to complete the pipeline's jobs, as shown in Figure 35.

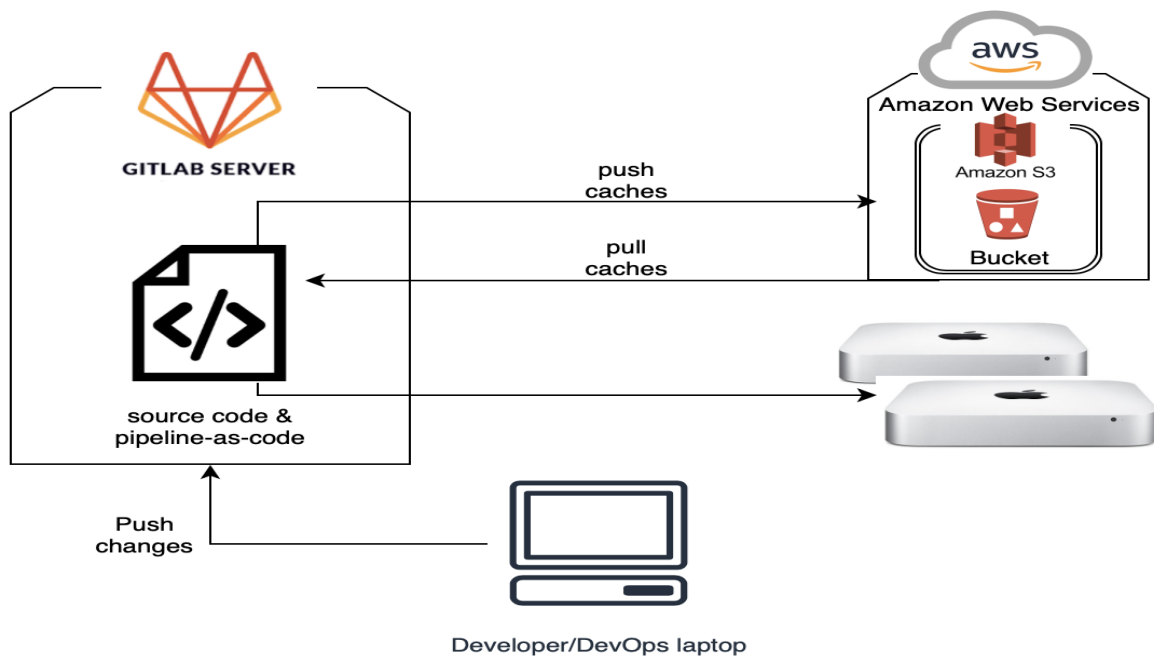


Figure 35 - Architecture of Mac servers & S3 bucket

5 Implementation

This Chapter provides the implementation of Kubernetes cluster and Docker image for Android's project, and physical runners for IOS's project.

5.1 Implementation for Android

To reach the final target in terms of the Android project, the Kubernetes cluster and Docker image solution was implemented using the following steps.

5.1.1 Create Cluster in EKS AWS service

To create cluster with nodes in AWS account:

- First, navigate to IAM (Identity and Access Management) service to create IAM Role for EKS service, this is important to grant the permissions for EKS to manage the resources on behalf of the user, as shown in steps 1 to 8 in Figures 36, 37, 38, and 39.

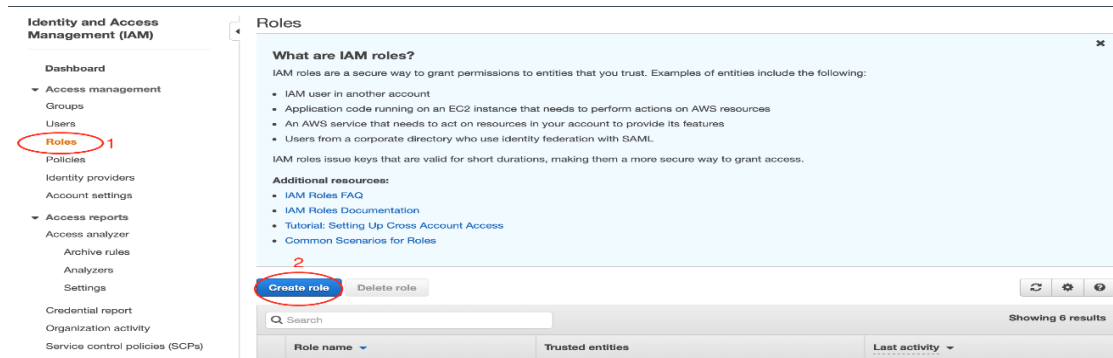


Figure 36 - navigate IAM service & create role

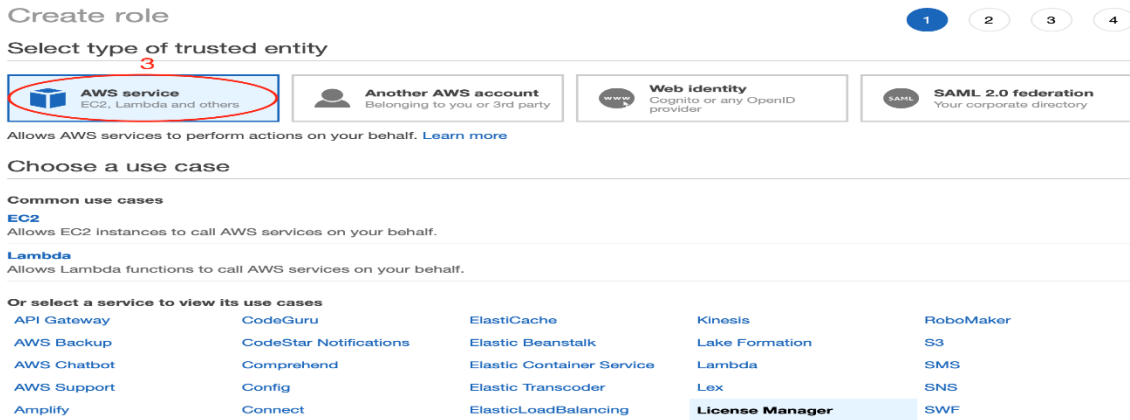


Figure 37 - select AWS services-granting permissions for AWS service

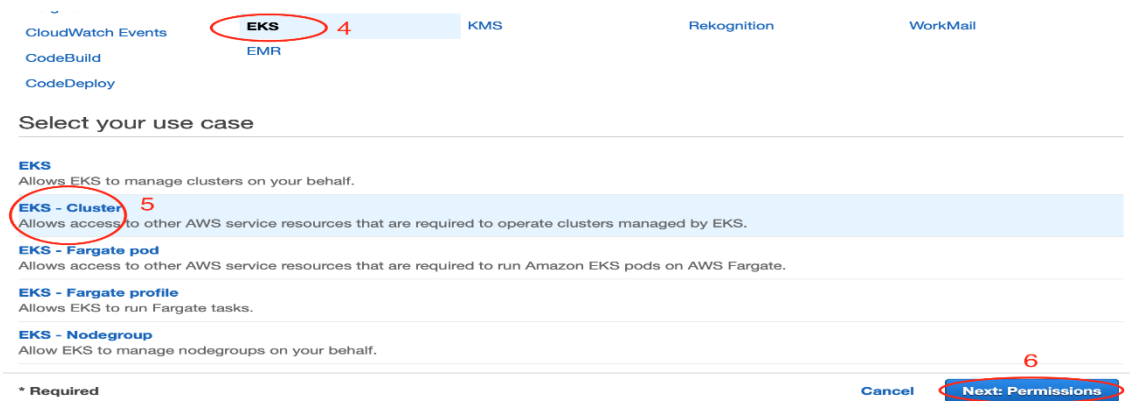


Figure 38 - Choose EKS-allowing EKS to manage the resources

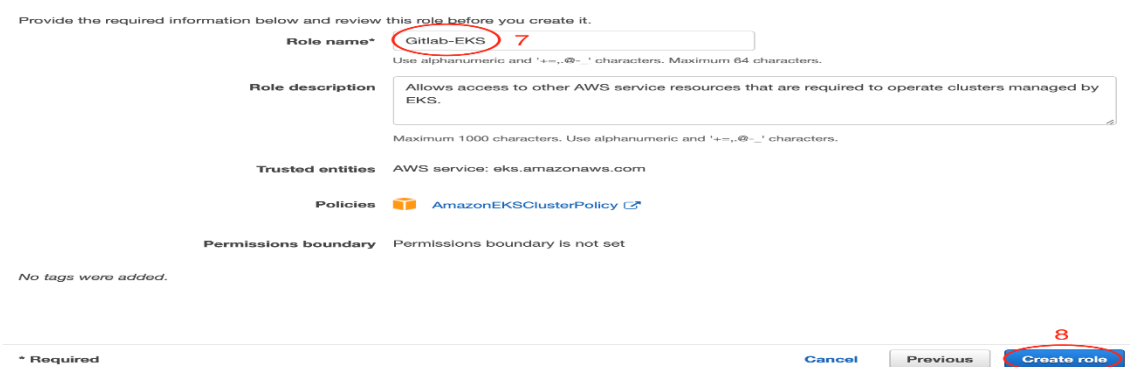


Figure 39 - Name the role and create it

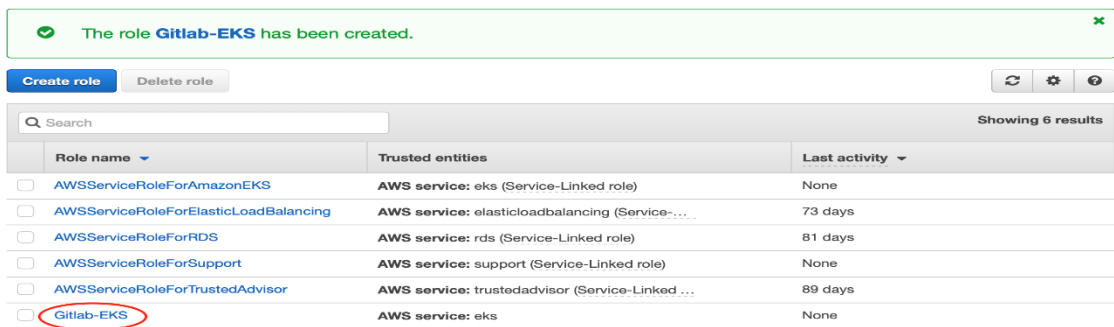


Figure 40 - Show the new EKS role in IAM Roles page

- Then, navigate to EKS service to create cluster, as shown in steps 1 to 4 in Figures 41, and 42

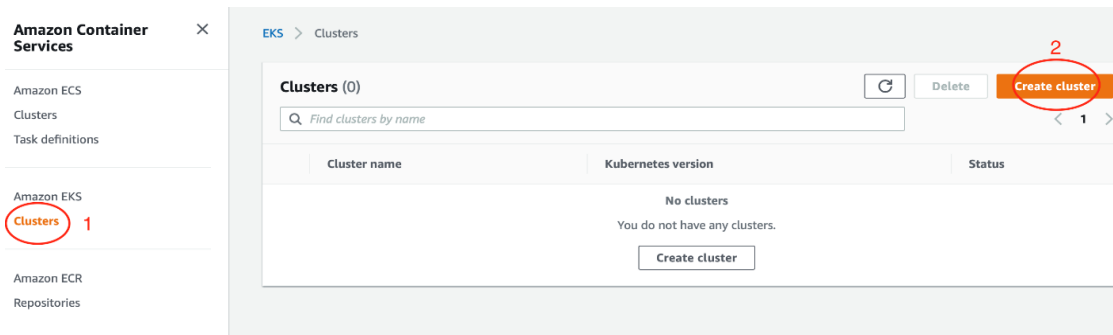


Figure 41 - Create cluster in EKS service



Figure 42 - Name the cluster and choose the IAM EKS Role

- To use the new cluster in Gitlab, it's necessary to add at least one node from (Compute) of the cluster as shown in Figure 43 and configure the minimum/maximum number of nodes that Kubernetes can add to the cluster.

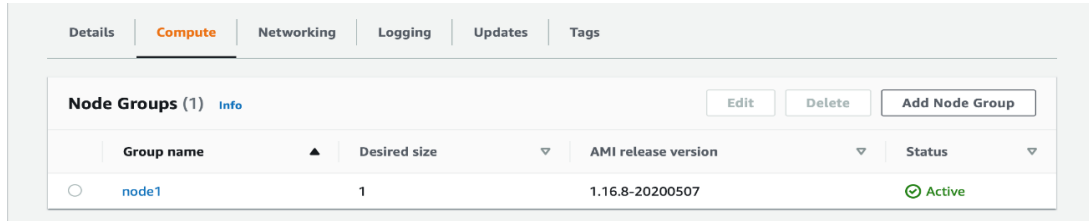


Figure 43 - adding node group in cluster

- To define the maximum number of the required nodes for this project, this equation was used:

$$\text{max num of nodes in the cluster} = \text{num of apps} * \text{max num of jobs on parallel per pipeline}$$

$$33 \text{ nodes} = 11 \text{ apps} * 3 \text{ job on parallel per pipeline}$$

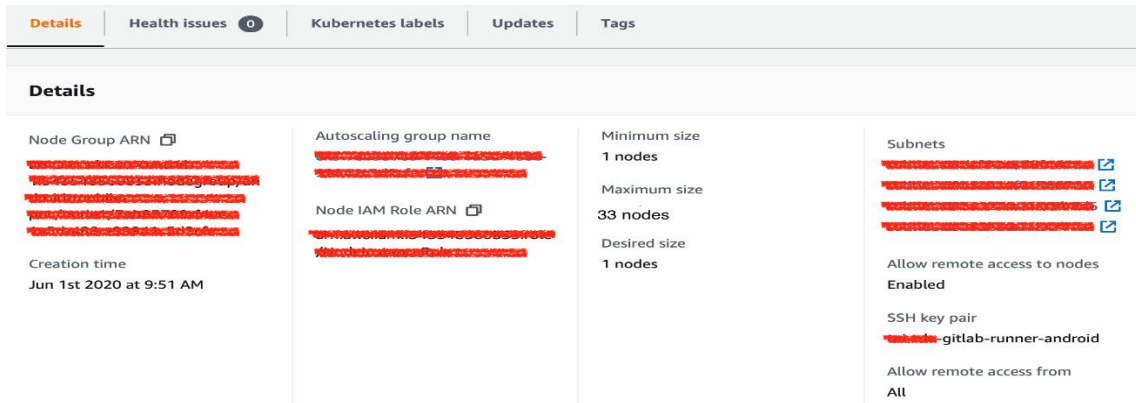


Figure 44 - Details of the node

5.1.2 Configure the cluster as runner in Gitlab

To hire the created cluster as a Runner in Gitlab.

Navigate the Android project >> settings >> CI/CD >> Runners.

The screenshot shows the GitLab Runners configuration page. On the left is a sidebar with navigation options: Merge Requests (0), CI / CD, Operations, Packages, Analytics, Snippets, and Settings. The Settings menu is expanded to show CI / CD, which is selected. The main content area is divided into three sections: 'Specific Runners', 'Shared Runners', and 'Group Runners'. The 'Specific Runners' section has two sub-sections: 'Set up a specific Runner automatically' and 'Set up a specific Runner manually'. The 'Set up a specific Runner automatically' section contains a list of steps and a button labeled 'Install Runner on Kubernetes' which is circled in red. The 'Set up a specific Runner manually' section contains a list of steps and a 'Reset runners registration token' button. The 'Shared Runners' section includes a description of shared runners, an 'Enable shared Runners' button, and a list of 'Available shared Runners: 1' with a redacted name and a 'gitlab-runner' tag. The 'Group Runners' section includes a description and a note that the project does not belong to a group.

Figure 45 - Runners page in Gitlab

The screenshot shows the 'Add existing cluster' form in GitLab. The form is titled 'Enter the details for your Kubernetes cluster' and includes a sub-header 'Please enter access information for your Kubernetes cluster. If you need help, you can read our documentation on Kubernetes'. The form has four main sections: 'Kubernetes cluster name' with a text input field; 'API URL' with a text input field and a note 'The URL used to access the Kubernetes API. More information'; 'CA Certificate' with a large text area for a 'Certificate Authority bundle (PEM format)' and a note 'The Kubernetes certificate used to authenticate to the cluster. More information'; and 'Service Token' with a text input field.

Figure 46 - Adding existing cluster in Gitlab

To fill the first three fields, check the details of the cluster created previously in AWS, and find the name, API URL, and CA Certificates.

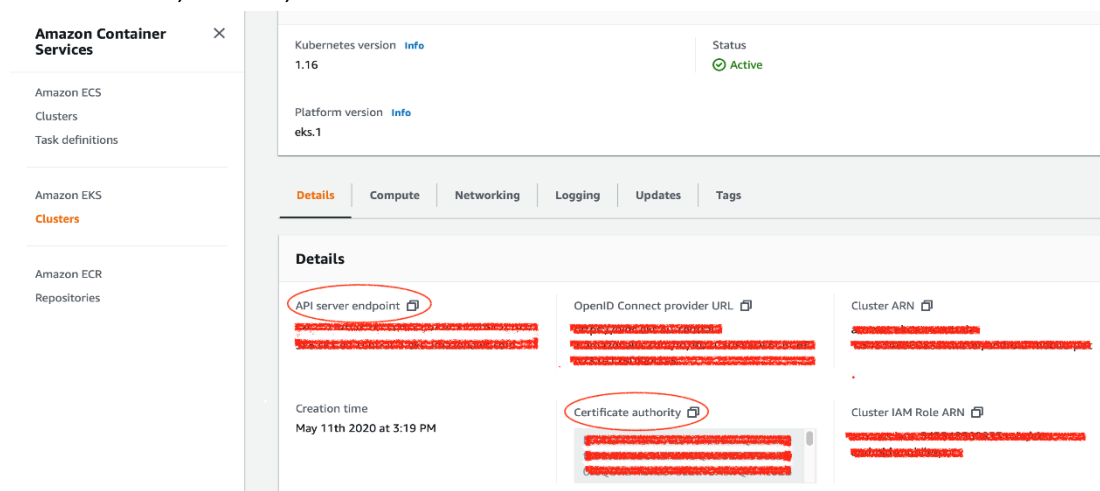


Figure 47 - Cluster details in AWS

In terms of the service token field, follow the Gitlab Documentation (Gitlab-Kubernetes, 2020).

- To connect to the EKS cluster locally, install AWS CLI (AWS Command Line Interface), from the terminal (AWS-CLI, 2020):
 - `curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"`
 - `unzip awscli-bundle.zip`
 - `sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws`
 - `aws --version`
- Install Kubectl (Kubernetes command-line utility)(AWS-eksctl, 2020):
 - `curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.16.8/2020-04-16/bin/darwin/amd64/kubectl`
 - `chmod +x ./kubectl`
 - `mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin`
 - `kubectl version --short -client`
- Then, add the kubeconfig file for the cluster (AWS-EKS-Connection, 2020):
 - `aws eks --region region update-kubeconfig --name cluster_name`
- It's necessary to have an IAM role with the required permissions to access and manage the cluster locally.
- Create YAML file, called (eks-admin-service-account.yaml):


```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-admin
  namespace: kube-system
```

Code 8 - create service account file (Gitlab-Kubernetes, 2020)
- Add this file to the cluster, using terminal:
 - `kubectl apply -f eks-admin-service-account.yaml`
- Create another YAML file, called (eks-admin-cluster-role-binding.yaml):

```

apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system

```

Code 9 - create cluster role binding file (Gitlab-Kubernetes, 2020)

- Add this file to the cluster, using terminal:
`kubectl apply -f eks-admin-cluster-role-binding.yaml`
- As shown in Figure 48, Get the token for the service account to fill the fourth field in Gitlab, by typing in terminal:
`kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')`

```

Name:          eks-admin-token-b5zv4
Namespace:    kube-system
Labels:       <none>
Annotations:  kubernetes.io/service-account.name=eks-admin
              kubernetes.io/service-account.uid=bcfe66ac-39be-11e8-97e8-026dce96b6e
Type:         kubernetes.io/service-account-token

Data
====
ca.crt:      1025 bytes
namespace:   11 bytes
token:       <authentication_token>

```

Figure 48 - the details of service account (Gitlab-Kubernetes, 2020)

Finally, the Kubernetes cluster as Runner is shown from Gitlab project >> settings >> CI/CD >> Runners

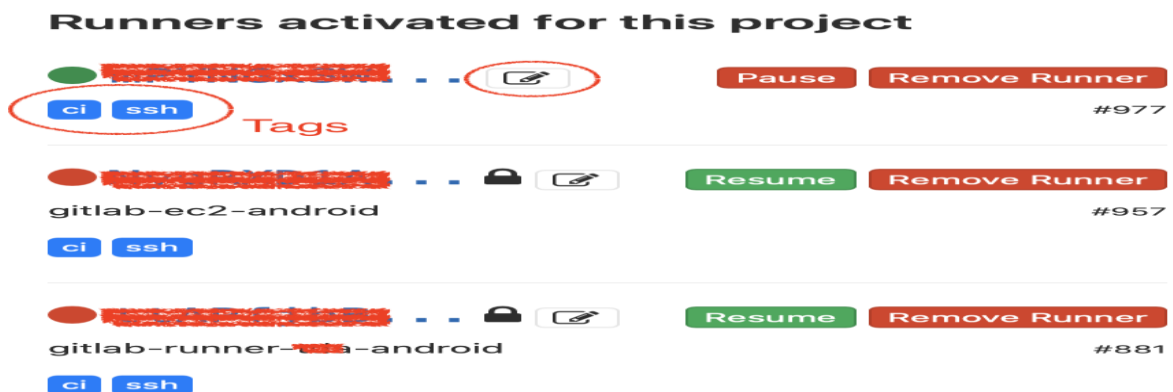


Figure 49 - Running Cluster with tags in Gitlab

5.1.3 Build Docker image

To build and test an Android project, configure the project's requirements on the machine's environment, and since using a cluster with multi machines, it was more effective to use configuration as code using Docker to reuse it in several machines.

After installing Docker from the formal website (Docker-installing, 2020), create Dockerfile included all the requirements.

```
FROM openjdk:8-jdk
ENV SDK_URL="https://dl.google.com/android/repository/sdk-tools-linux-3859397.zip"
ENV ANDROID_HOME="/opt/android-sdk"
ENV ANDROID_VERSION=28
ENV ANDROID_BUILD_TOOLS_VERSION=28.0.2
# Install Build Essentials
RUN apt-get --quiet update --yes\
&& apt-get upgrade --yes\
&& apt-get upgrade python --yes\
&& apt-get --quiet install --yes wget tar unzip lib32stdc++6 lib32z1
RUN mkdir "$ANDROID_HOME" .android \
  && cd "$ANDROID_HOME" \
  && curl -o sdk.zip $SDK_URL \
  && unzip sdk.zip \
  && rm sdk.zip \
  && mkdir "$ANDROID_HOME/licenses" || true \
  && echo "24333f8a63b6825ea9c5514f83c2829b004d1fee" >
"$ANDROID_HOME/licenses/android-sdk-license"
# Install Android Build Tool and Libraries
RUN $ANDROID_HOME/tools/bin/sdkmanager --update
RUN $ANDROID_HOME/tools/bin/sdkmanager "build-
tools;${ANDROID_BUILD_TOOLS_VERSION}" \
  "platforms;android-${ANDROID_VERSION}" \
  "platform-tools"
#Import signing keys & Install appcenter & Install jq
RUN mkdir /root/certificates/
#RUN chmod 777 /root/certificates
ADD keystores /root/certificates/
ENV NODE_VERSION=12.6.0
RUN apt install -y curl
RUN curl -o-
https://raw.githubusercontent.com/creationix/nvm/v0.34.0/install.sh | bash
ENV NVM_DIR=/root/.nvm
RUN . "$NVM_DIR/nvm.sh" && nvm install ${NODE_VERSION}
RUN . "$NVM_DIR/nvm.sh" && nvm use v${NODE_VERSION}
RUN . "$NVM_DIR/nvm.sh" && nvm alias default v${NODE_VERSION}
ENV PATH="/root/.nvm/versions/node/v${NODE_VERSION}/bin/:${PATH}"
RUN node --version
RUN npm --version
RUN npm install -g appcenter-cli \
  && appcenter --version
RUN curl -o /usr/local/bin/jq
http://stedolan.github.io/jq/download/linux64/jq && \
  chmod 755 /usr/local/bin/jq
```

Code 10 – Dockerfile for the android environment configuration

5.1.4 Push Docker image to ECR AWS registry

To use the image later in the cluster, it must be built and stored in AWS registry.

First, navigate ECR AWS service to create new Repository

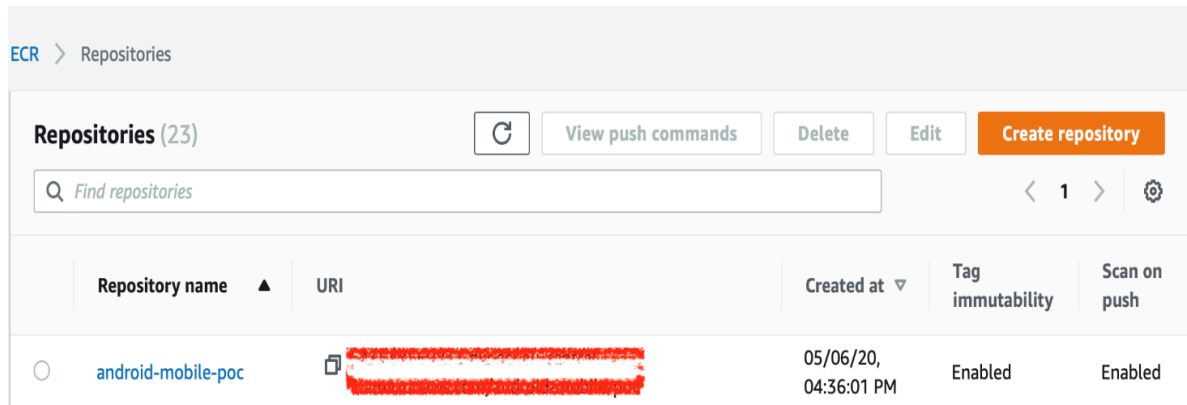


Figure 50 - Create new Repository in ECR

Navigate the repository to push the Docker image

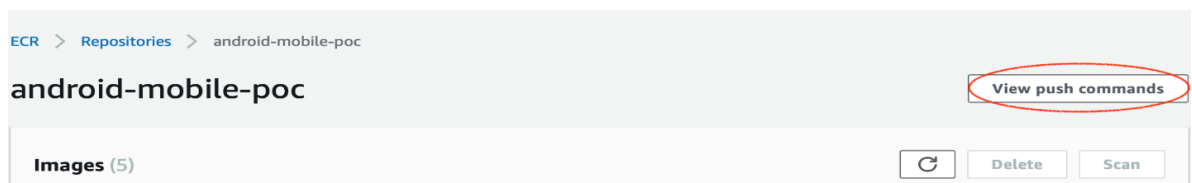


Figure 51 - View the push commands in the repository

Using terminal, access to AWS ECR, then build, tag, and push the image to the repository.

- `aws ecr get-login-password --region XX-XXXXXXX-X | docker login --username AWS --password-stdin XXXXXXXXXXXX.XXX.XXX.XXX-XXXXXXX-X.amazonaws.com`
- `docker build -t android-mobile-poc .`
- `docker tag android-mobile-poc:latest XXXXXXXXXXXX.XXX.XXX.XXX-XXXXXXX-X.amazonaws.com/android-mobile-poc:latest`
- `docker push XXXXXXXXXXXX.XXX.XXX.XXX-XXXXXXX-X.amazonaws.com/android-mobile-poc:latest`

5.1.5 Use S3 as Cache server

To reduce the consumed time of running, GitLab provides extra configurations in Runner to cache the builds and store them in the cache server, for instance, AWS S3.

In the case of Kubernetes Runner, it requires a third party called Helm Chart, to allow the user to add configurations to the cluster (Gitlab-Helm Chart, 2020).

The first step is installing the Helm CLI (Installing Helm, 2020), using the terminal:

```
➤ brew install helm
```

Then, install the configuration file that is provided by the Gitlab community, and called values.yaml file (Gitlab-values.yaml, 2020), and edit the cache part as shown in code 11.

```
cache:
  ## General settings
  cacheType: s3
  cachePath: gitlab_runner
  cacheShared: true
  ## S3 settings
  s3ServerAddress: s3.amazonaws.com
  s3BucketName: xxxxxx
  s3BucketLocation: xx-xxxxxx-x
  s3CacheInsecure: false
  secretName: xxxx
```

Code 11 – cache configuration in Values.yaml

From terminal, access to EKS, to pass this file to the cluster:

```
➤ kubectl get svc
➤ helm install --namespace gitlab-managed-apps gitlab-runner -f values.yaml
  gitlab/gitlab-runner
```

5.1.6 Add Gitlab Pipeline & bash script to Repository

Create a new file in the project's directory and call it .gitlab-ci.yml, included the same stages that were explained in 4.4 Section:

```
image: XXXXXXXXXXXX.XXX.XXX.XX-XXXXXXX-X.amazonaws.com/android-mobile-
poc:x.x
variables:
# to disable the Gradle Daemon
  GRADLE_OPTS: "-Dorg.gradle.daemon=false"
# caching per branch, and determine the default policy of the caching,
pulling caches from S3 and push edits to S3
cache: &global_cache
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - .gradle
    - build
  policy: pull-push
before_script:
# run the bash script to export the GRADLE_APPNAME_TASKS variable
```

```

- source get-app-info.sh
- echo ${CI_COMMIT_TAG}
- echo ${GRADLE_APPNAME_TASKS}
- export GRADLE_USER_HOME=`pwd`/.gradle
stages:
- Build
- Code quality checks
- Tests
- Distribute
compile:
tags:
- ci
stage: "Build"
script:
- ./gradlew --build-cache assemble${GRADLE_APPNAME_TASKS}MinifiedDebug
- ls -la
artifacts:
paths:
- app/build/outputs/
lint:
tags:
- ci
stage: "Code quality checks"
dependencies:
- compile
script:
- ls -la
- ./gradlew --build-cache lint${GRADLE_APPNAME_TASKS}Debug
artifacts:
name: "reports_${CI_PROJECT_NAME}_${CI_BUILD_REF_NAME}"
when: always
paths:
- app/build/reports/
checkstyle:
tags:
- ci
stage: "Code quality checks"
script:
- ls -la
- ./gradlew --build-cache checkstyle
artifacts:
name: "reports_${CI_PROJECT_NAME}_${CI_BUILD_REF_NAME}"
when: always
paths:
- app/build/reports/
cache:
# inherit the cache to change the policy, pull the cache from S3 without
pushing the edits of this job
<<: *global_cache
policy: pull
detekt:
tags:
- ci
stage: "Code quality checks"
script:
- ls -la
- ./gradlew --build-cache detekt
artifacts:
name: "reports_${CI_PROJECT_NAME}_${CI_BUILD_REF_NAME}"
when: always

```

```

    paths:
      - app/build/reports/
  cache:
    <<: *global_cache
    policy: pull
  authentication unit tests:
    tags:
      - ci
    stage: "Tests"
    dependencies:
      - compile
    script:
      - ./gradlew --build-cache :tuiauthentication:testDebugUnitTest
  app unit tests:
    tags:
      - ci
    stage: "Tests"
  # specify extra memory and CPUs for the Unit test job
  variables:
    KUBERNETES_CPU_REQUEST: 4
    KUBERNETES_CPU_LIMIT: 8
    KUBERNETES_MEMORY_REQUEST: 4Gi
    KUBERNETES_MEMORY_LIMIT: 16Gi
  dependencies:
    - lint
  script:
    - ./gradlew --build-cache test${GRADLE_APPNAME_TASKS}DebugUnitTest
  artifacts:
    reports:
      junit: app/build/test-results/**/*.*.xml
  upload to App Center:
    tags:
      - ci
    stage: "Distribute"
    dependencies:
      - compile
    script:
      - appcenter login --token $APPCENTER_API_TOKEN
      - appcenter distribute release -f app/build/outputs/apk/**/*.*.apk -g
Collaborators --app xxx-Organization/Android-Mobile-POC
Code 12 – .gitlab-ci.yml-Android's pipeline

```

To use a single pipeline with multiple applications and give the user the ability to build a specific app, it was necessary to find a method like parameterized input in Jenkins (Jenkins-parameters, 2020).

Unfortunately, Gitlab doesn't support the parameters and for this reason, the git tags and the environment variable were adopted to implement the user input concept, where the developer can type the app name to be built during committing changes.

Using the terminal, and after adding and committing the changes, these command lines must be added:

- git tag -a (tag name) -m "commit description"
- git push origin (tag name)

knowing the tags must be the apps' names with (_number):

BE-JA, BE-TF, DE-MT, DE-TC, NL, NO-DK, NO-FI, NO-NO, NO-SV, UK-FC, UK-TH
Or using the Gitlab's GUI, where the APP_CODE variable must be inserted with one of the apps' names, as shown in Figure 52.

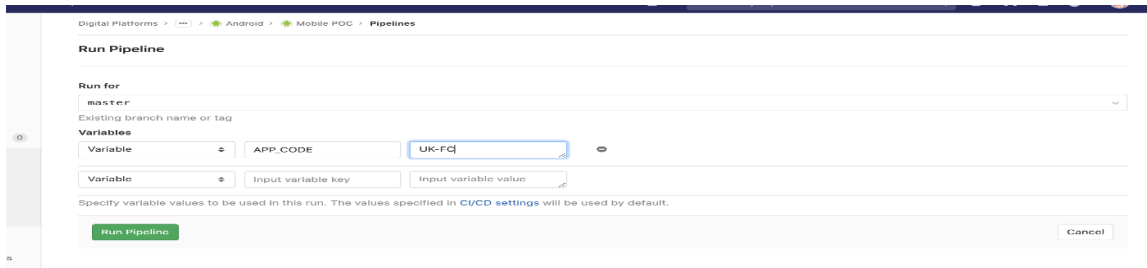


Figure 52 - insert environment variable to build a specific app

On the other side, bash script was added and called get-app-info.sh in the project's directory, to match the provided App name (CI_COMMIT_TAG) with its job name (GRADLE_APPNAME_TASKS) then export it.

```
# Get apps metadata
curl -s "https://s3.eu-central-1.amazonaws.com/tui.tda.jenkins.scripts/android/apps.json" --output
apps.json
# Detect APP_CODE env variable
if [ -n "$APP_CODE" ]
then
echo "Detected APP_CODE=$APP_CODE"
fi
# Check the tag in case if app code is empty
if [ -n "$CI_COMMIT_TAG" ] && [ -z "$APP_CODE" ]
then
echo "Detected app code in tag -> $CI_COMMIT_TAG"
APP_CODE=$(echo $CI_COMMIT_TAG | cut -f1 -d_)
echo "APP_CODE=$APP_CODE"
fi
# Load json info from app code
APP_JSON=$(cat apps.json | jq -r ".apps[\"$APP_CODE\"]")
# Set environment variables for gradle and appcenter based on app json
content
if [ "$APP_JSON" != "null" ]
then
    export GRADLE_APPNAME_TASKS=$(cat apps.json | jq -r
".apps[\"$APP_CODE\"][\\"graddleAppName\"]")
    export APPCENTER_APPNAME=$(cat apps.json | jq -r
".apps[\"$APP_CODE\"][\\"buildType\"][\\"dev\"][\\"appCenterAppName\"]")
else
    export GRADLE_APPNAME_TASKS="Netherlands"
    export APPCENTER_APPNAME="TDA-Organization/NL-Android-Develop"
fi
```

Code 13 – get-app-info.sh - Android's project

As shown in Figure 53, whenever the changes are pushed to GitLab, the pipeline will start work immediately.

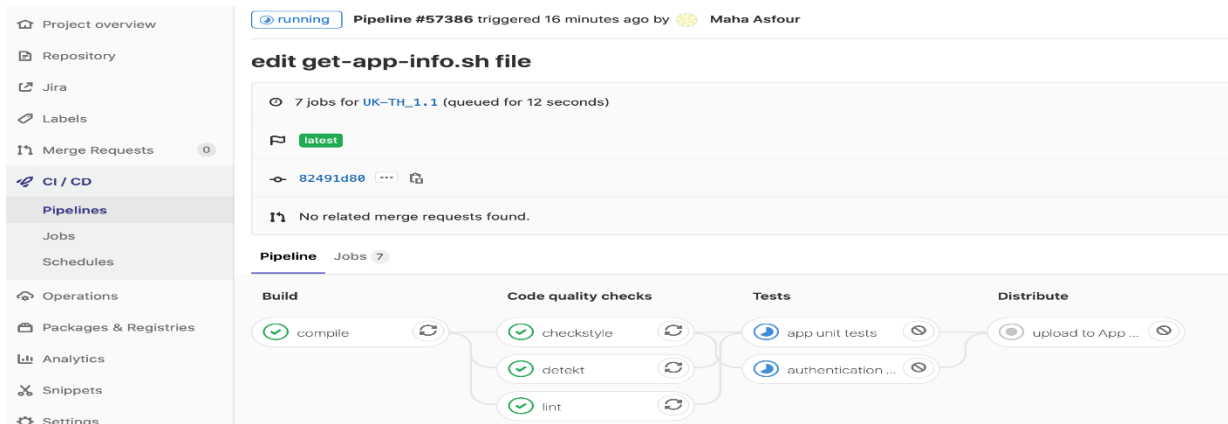


Figure 53 - Running Pipeline for Android project

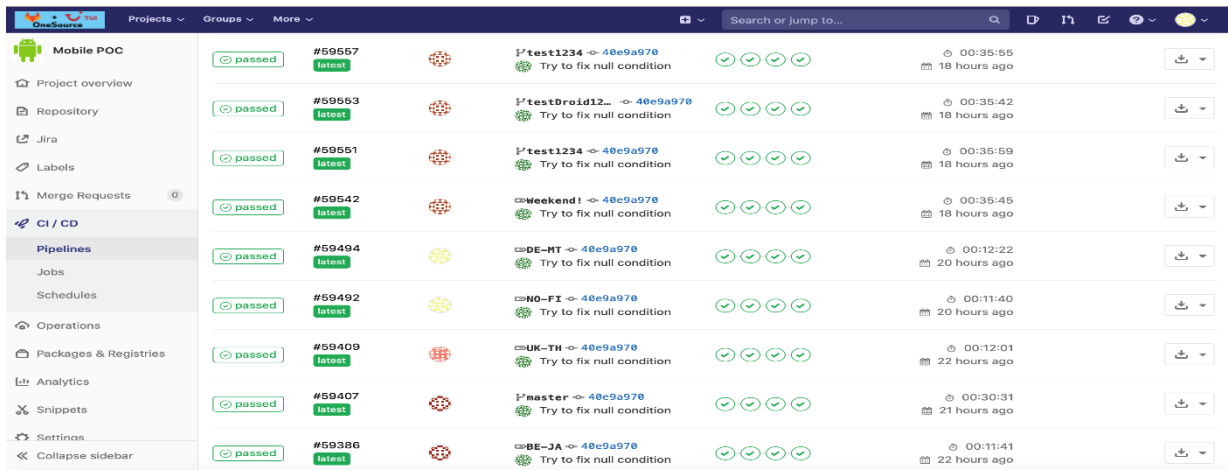


Figure 54 - passed pipelines for the Android apps

5.2 Implementation for IOS

This solution will be implemented based on physical mac servers provided by the company.

5.2.1 Configure the Mac servers as runners in Gitlab

To hire Mac servers/Mac machines as Runner in Gitlab, first install Gitlab Runner on the servers then register the runners.

- Install Gitlab Runner: Open the terminal and paste the following command lines to install and start the service.
 - `brew install gitlab-runner`
 - `brew services start gitlab-runner`

- Register Gitlab Runner: The following command lines can register the runner in Gitlab and run the pipeline.

- `gitlab-runner register`
- Please enter the gitlab-ci coordinator URL (e.g. <https://gitlab.com>)
`https://gitlab.com`
- Please enter the gitlab-ci token for this runner
`xxx`

To get the token, from Gitlab project >> settings >> CI/CD >> Runners

- Please enter the gitlab-ci description for this runner [hostname] optional description
- Please enter the gitlab-ci tags for this runner (comma separated):
`ios, mac`
- Please enter the executor: `ssh, docker+machine, docker-ssh+machine, kubernetes, docker, parallels, virtualbox, docker-ssh, shell`

Now the Runner is shown from Gitlab project >> settings >> CI/CD >> Runners

Runners activated for this project

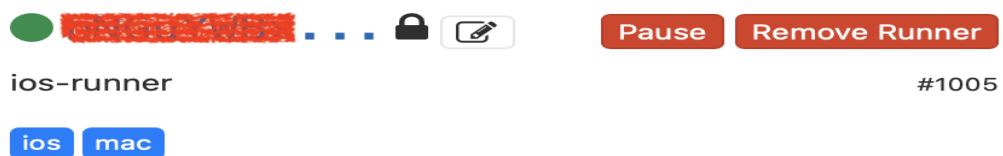


Figure 55 - Gitlab Runner for IOS project

Before building and testing the project in the runner, it's necessary to prepare the environment of the servers with the project's requirements like installing XCode(Xcode, 2020) and Fastlane to build the project, Appcenter CLI to push IPA version to Appcenter, sonar-scanner to detect bugs and check the quality of the code (SonarQube, 2020), and the development code signing certificates, where the certificates for Apple development are mandatory approach, to verify the security, and assure the end-users that the application wasn't attacked from third party (Code Signing - Support - Apple Developer 2020).

5.2.2 Use S3 as Cache server

The configuration of physical Runner is simpler than the Cluster, and doesn't need third party.

Navigate `.gitlab-runner` directory in the server:

- `cd ~/.gitlab-runner`
- `nano config.toml`

then edit the `config.toml` file, by adding the S3 cache server information (Gitlab-Runner-Cache 2020):

```
[runners.cache]
  Type = "s3"
  Path = "gitlab_runner"
  Shared = true
[runners.cache.s3]
  ServerAddress = "s3.amazonaws.com"
  AccessKey = "AWS_S3_ACCESS_KEY"
  SecretKey = "AWS_S3_SECRET_KEY"
  BucketName = "gitlab-runner-cache"
  BucketLocation = "eu-central-1"
  Insecure = false
```

Code 14 – caching configuration in config.toml Runner

5.2.3 Add Gitlab Pipeline & bash script to Repository

Create a new file in the project's directory and call it .gitlab-ci.yml, included the same stages that were explained in 4.4 Section:

```
variables:
  LC_ALL: "en_US.UTF-8"
  LANG: "en_US.UTF-8"
cache:
  key: ${CI_COMMIT_REF_SLUG}
  paths:
    - Carthage/
before_script:
  - source get-app-info.sh
  - echo ${APP}
stages:
  - Dependencies
  - Build and test
carthage dependencies:
  tags:
    - ios
  stage: Dependencies
  script:
    - bundle update
    - fastlane reset
    - fastlane env
    - fastlane dependencies
build and distribute:
  tags:
    - ios
  stage: Build and test
  script:
    - security unlock-keychain -p "$GITLAB_KEYCHAIN_PWD"
~/Library/Keychains/gitlab-ci.keychain-db
    - fastlane code_signing app:${APP}
    - FASTLANE_XCODEBUILD_SETTINGS_RETRIES=10 fastlane build app:${APP}
    - security lock-keychain ~/Library/Keychains/gitlab-ci.keychain-db
    - appcenter login --token $APPCENTER_API_TOKEN
    - appcenter distribute release -f ./fastlane/build/**/*.ipa -g
Collaborators --app ***-Organization/IOS-Mobile-POC
artifacts:
  paths:
    - ./fastlane/build/**/*.ipa
```

```

run tests & code quality:
  tags:
  - ios
  stage: Build and test
  script:
    - FASTLANE_XCODEBUILD_SETTINGS_RETRIES=10 fastlane tests
    - fastlane quality
    - echo $CI_COMMIT_BRANCH
    - sonar-scanner -Dsonar.branch.name="$CI_COMMIT_BRANCH"
  artifacts:
    reports:
      junit: ./fastlane/test_output/*.junit
    paths:
      - ./fastlane/coverage
      - ./fastlane/swiftlint.result.html
  expire_in: 15 days

```

Code 15 – .gitlab-ci.yml-IOS’s pipeline

To run a single pipeline with multiple applications and activate the concept of user input, the same Android project’s approach was implemented in IOS’s project, except the bash script that was edited as shown in code 16.

```

if [ -n "$APP_CODE" ]
then
  echo "Detected APP_CODE=$APP_CODE"
fi
if [ -n "$CI_COMMIT_TAG" ] && [ -z "$APP_CODE" ]
then
  echo "Detected app code in tag -> $CI_COMMIT_TAG"
  APP_CODE=$(echo $CI_COMMIT_TAG | cut -f1 -d_)
  echo "APP_CODE=$APP_CODE"
fi
if [ -n "$APP_CODE" ]
then
  # Export app name
  export APP_CODE
else
  # Export app name
  export APP_CODE="NL"
fi

```

Code 16 – get-app-info.sh - IOS’s project

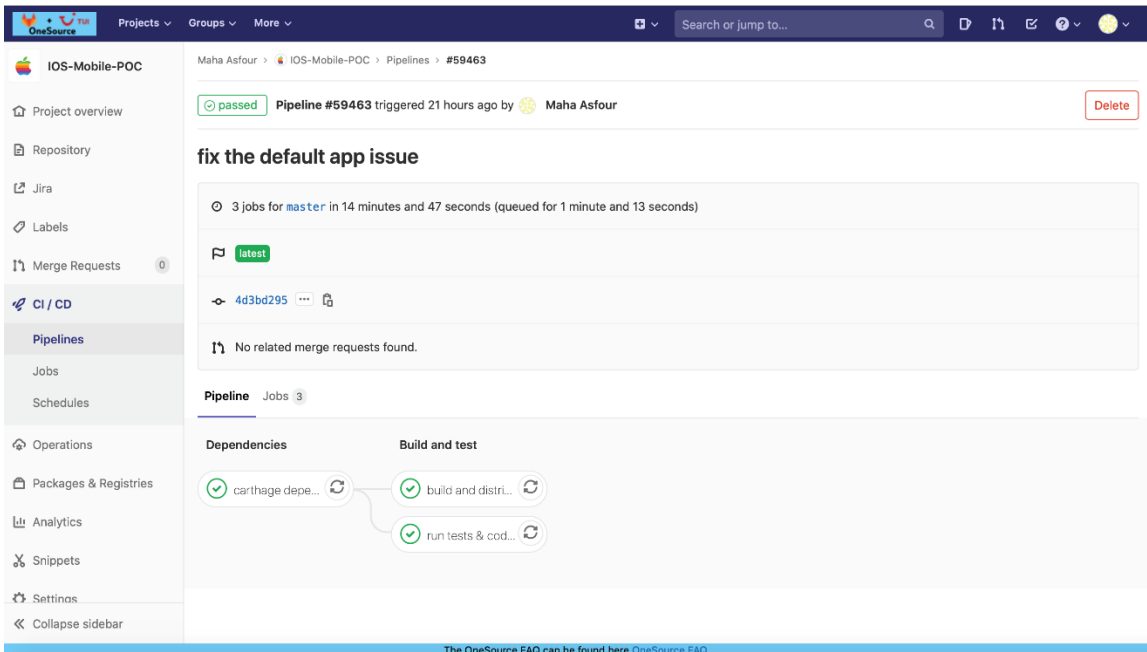


Figure 56 - Running Pipeline for IOS project

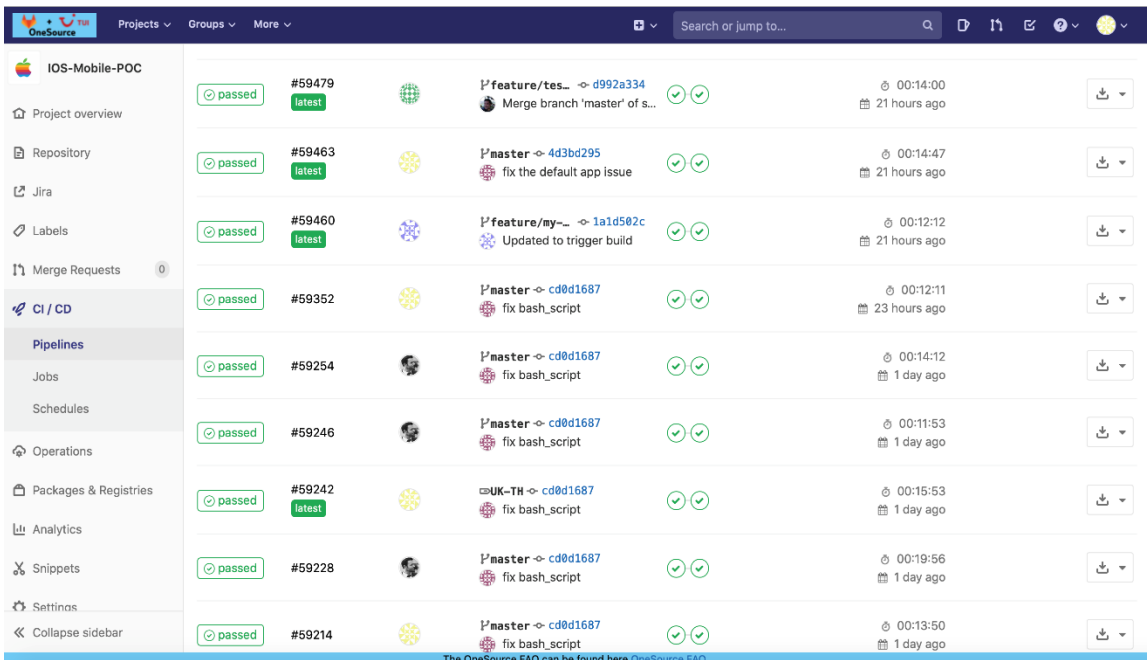
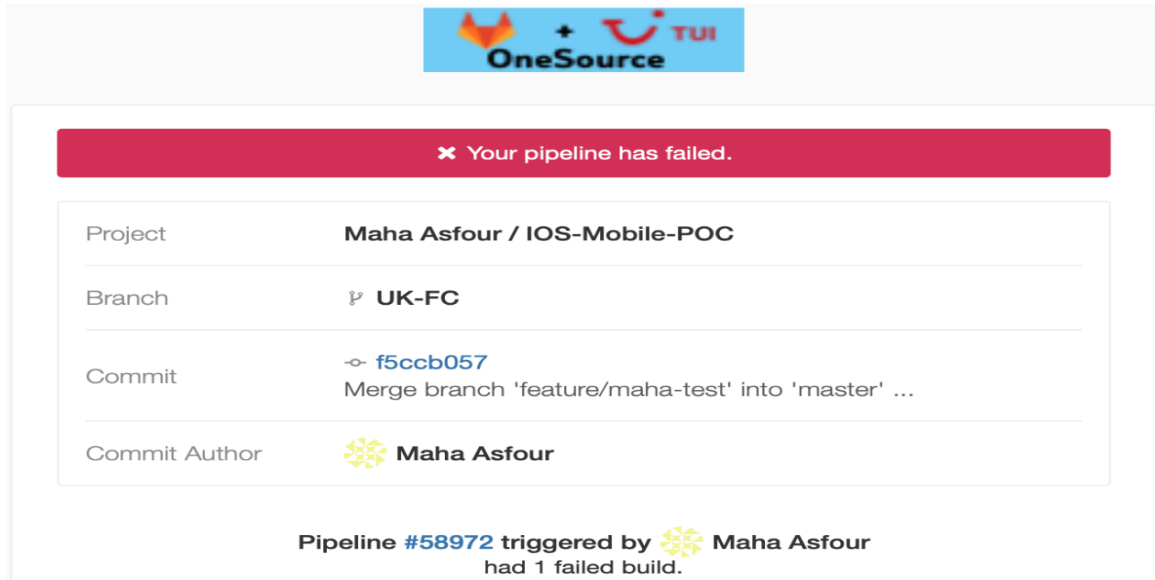




Figure 57 - passed pipelines for IOS apps

The collaboration is noted in Gitlab CI, where the tool sends emails automatically to the team to inform them in case of the failed builds, as shown in Figure 58.





✘ Your pipeline has failed.

| | |
|---------------|--|
| Project | Maha Asfour / IOS-Mobile-POC |
| Branch | UK-FC |
| Commit | f5ccb057 Merge branch 'feature/maha-test' into 'master' ... |
| Commit Author |  Maha Asfour |


Pipeline #58972 triggered by  **Maha Asfour**
had 1 failed build.

Figure 58 - passed pipelines for IOS apps

6 Experimentation & Evaluation

6.1 Research Hypothesis Specification

There are two hypotheses, the first one, Gitlab can improve the performance of the CI/CD pipeline, generating faster builds, meanwhile activates the metrics system to monitor the performance of the pipeline. The other hypothesis, Gitlab can give rise to improved user experience in comparison to current tools.

6.2 Indicators & Source of Information

After implementing the solution, it's necessary to check if the goals of the thesis have been achieved, by considering the following indicators for Gitlab:

- The performance of the pipeline: by scaling the consuming time of pipelines builds in the new solution and compare it with the old one. The time of the new pipeline builds must be smaller than the old status.
- Usability: The tangible measures are the required time for users to learn Gitlab (learning time) and complete their tasks (completion time) using it (Frokjaer, 2000). The learning and completion time must be smaller than the ones in the case of Jenkins and BuddyBuild.

- Extensibility: we can measure it by the number of the nodes/runners/machines/servers that we can add and configure in Gitlab to build the pipeline safely, and the number of the projects that we can add and build them in Gitlab in the same time without having critical issues.

It must support extending the solution up to 11 projects (the number of the total TUI's Apps) and add more nodes (machines) in the cluster up to 33 nodes and scale up the memory for the machine without having a crash in the system.

- Fault tolerance and Reliability: the system's ability to continue the pipeline builds in case of failure one or more machines (runners of Gitlab) and how often the machines fail to trigger the pipeline. The number of failed build cases per week must be rare and smaller than the old status.

The source of the information will be the low consumed time for building pipelines and the low number of failed build cases per week, they will be measured by the metrics system, as well as the low learning and completion time, and the high number of added machines and projects, they will be checked by the opinions of developers, and the DevOps team.

6.3 Evaluation Methodology

To evaluate the speed of the pipeline builds, Gitlab measures the demanded time for every stage of the pipeline during running the pipeline and shows the total consumed time when the process is done, then, the metrics system adds these results automatically to the Analytics CI/CD charts, where these bar charts include the running time of the last 30 commits, which facilitates the comparison between two commits.

The metrics system results are available on the page of the project >> Analytics >> CI/CD, the results were checked consistently to enhance the solution and were collected in the third week of June.

To measure reliability, other auto-generated CI / CD charts display the average of failed pipeline cases during the week (GitLab-analytics, 2020), which can be reached to these Area charts, from the page of the project >> Analytics >> CI/CD.

These results have been collected in the third week of June.

On the other side, the usability, and extensibility can be measured by creating a survey (questionnaire) using Google Forms (Google Forms: Free Online Surveys for Personal Use, 2020) in the third week of June. This Survey contains 5-point Likert Scale questions (Mcleod, 2020) to check the opinions of Developers, and DevOps team in the company, about the last indicators, and have feedback about the performance of the new tool and compare it with the old ones.

This survey was published to three Android developers, three IOS developers, and three of the DevOps team.

Seven questions were about the usability of Gitlab, like the learning time, completion time, and their impression of Gitlab's GUI, two questions about the ability of Gitlab to handle many projects and runners, one question, to measure the acceptance of Gitlab as a new CI/CD tool, and a bunch of questions to compare Gitlab, Jenkins, and BuddyBuild, in terms of the usability, the performance, and the customizing.

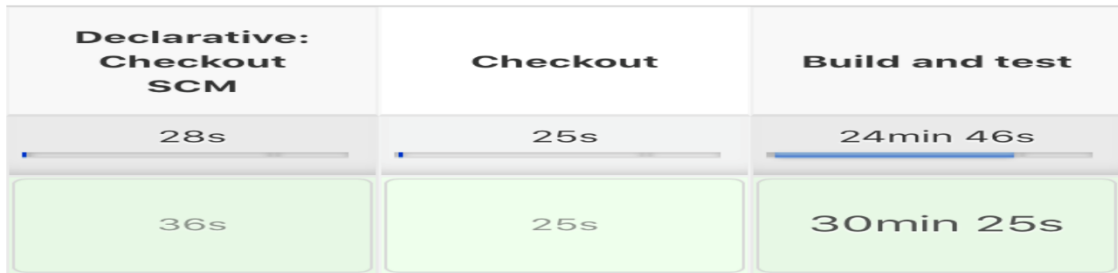
6.4 Evaluation Results

As a result of implementing the evaluation methodology to measure the pipeline's speed.

Figure 59 displays the Android pipeline's time in Jenkins (31 min and 26 every build) and Gitlab (36 min and 13 sec in the first build, and 12 min and 20 sec when it builds with caches).

In the case of 10 builds every day with one commit, Jenkins required around $(31 * 10 = 310)$ min per day, while Gitlab required around $(36 + 12 * 9 = 144)$ min per day, which leads that the Gitlab's pipeline is two times faster than the previous one.

Jenkins: Android pipeline



Gitlab: Android pipeline

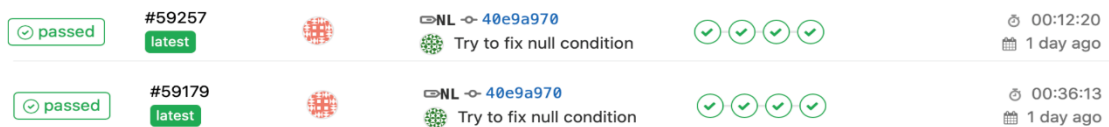


Figure 59 - Jenkins and Gitlab Android pipelines results

As shown in Figure 60, navigate the Analytics >> CI/CD to check the metrics of the project in Gitlab CI.

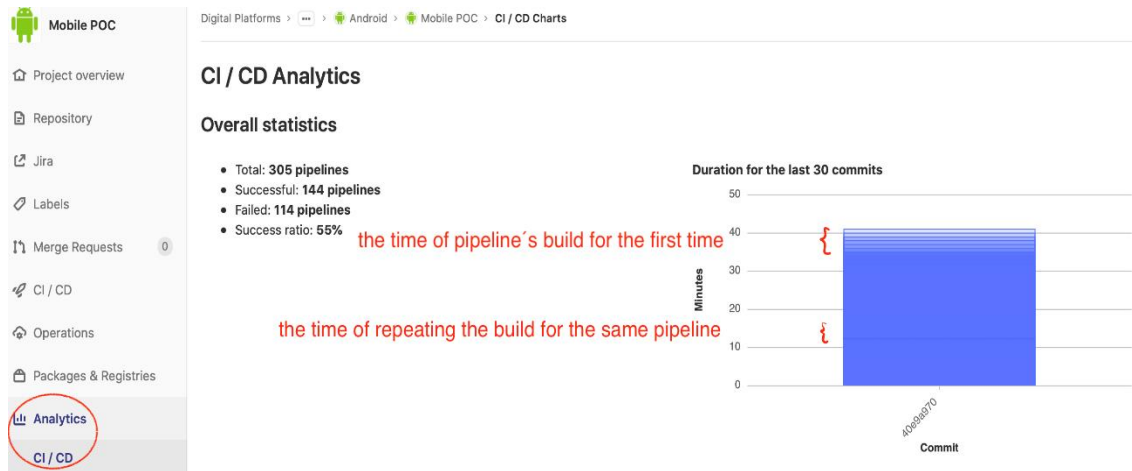


Figure 60 - Analytics CI/CD times' charts in Gitlab for Android project

Figure 61 shows the IOS pipelines' time, in Jenkins around 18 min for every test, in BuddyBuild (31 min and 29 sec) for every build, while in Gitlab around 13 min for every build and test.

Considering these results, the Gitlab pipeline achieved better speed compared to Jenkins and BuddyBuild ones.

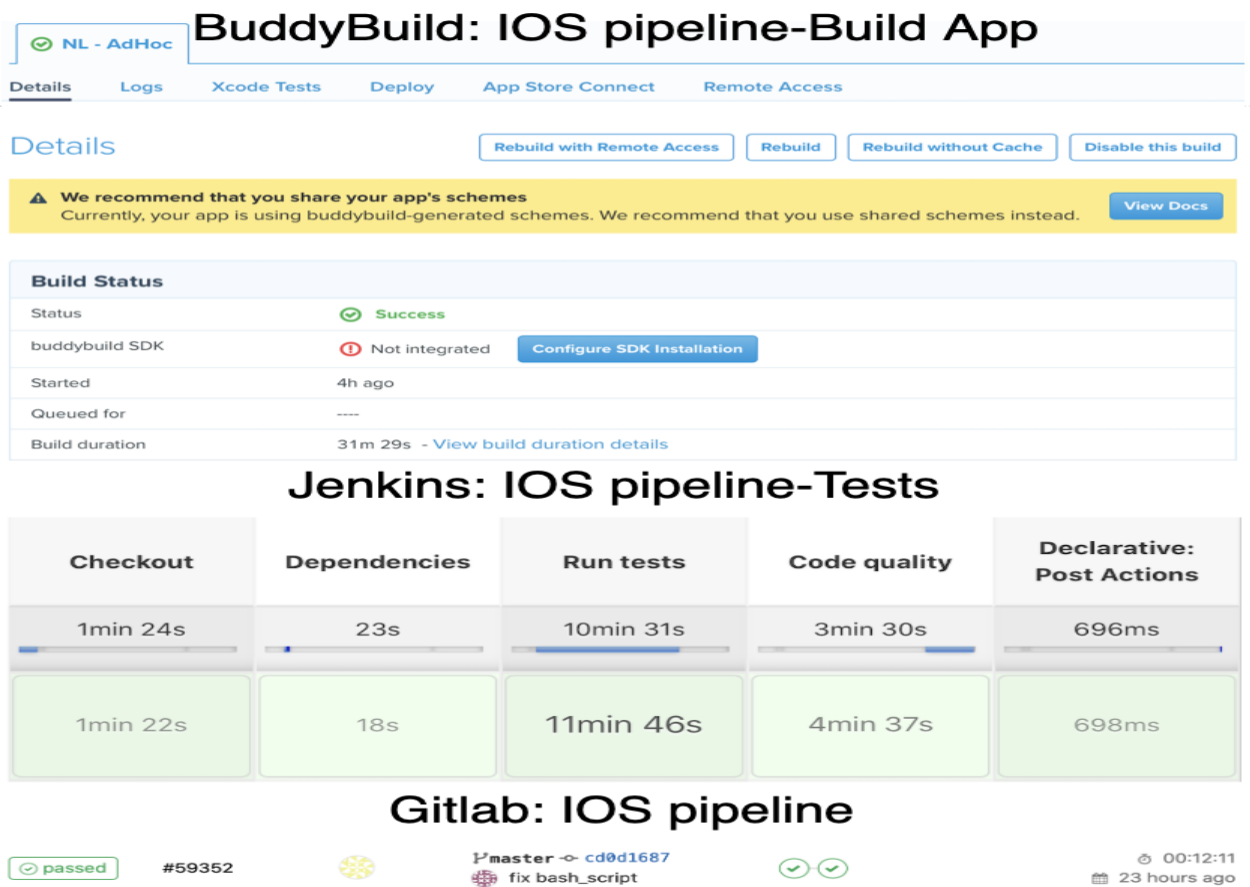


Figure 61 - Jenkins and Gitlab IOS pipelines results

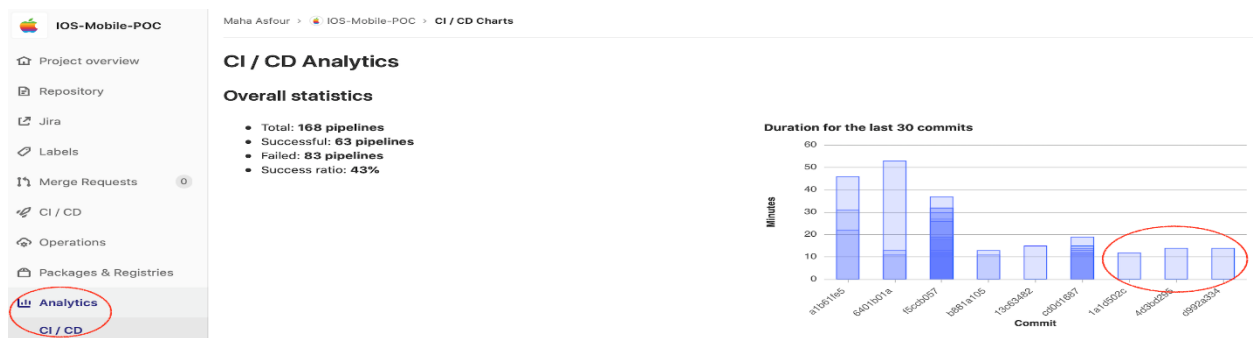


Figure 62 - Analytics CI/CD times' charts in Gitlab for IOS project

While the results of measuring the reliability are shown in Figures 63 and 64, where the charts display the number of successful pipelines per day compared to the total number of the pipelines, this metric needs more time to be monitored to decide if the solution achieves the reliability or not.

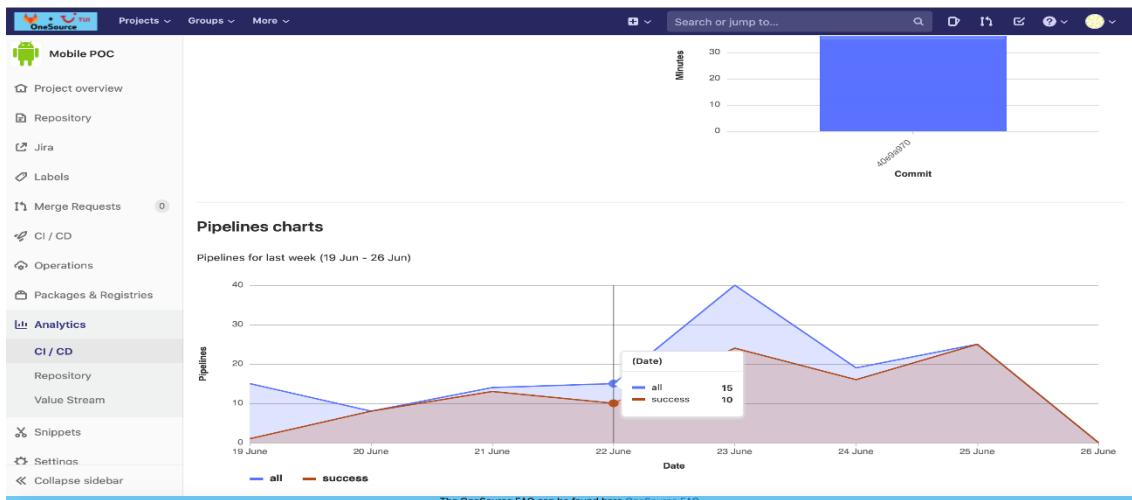


Figure 63 - Analytics CI/CD success rate' charts in Gitlab for Android project

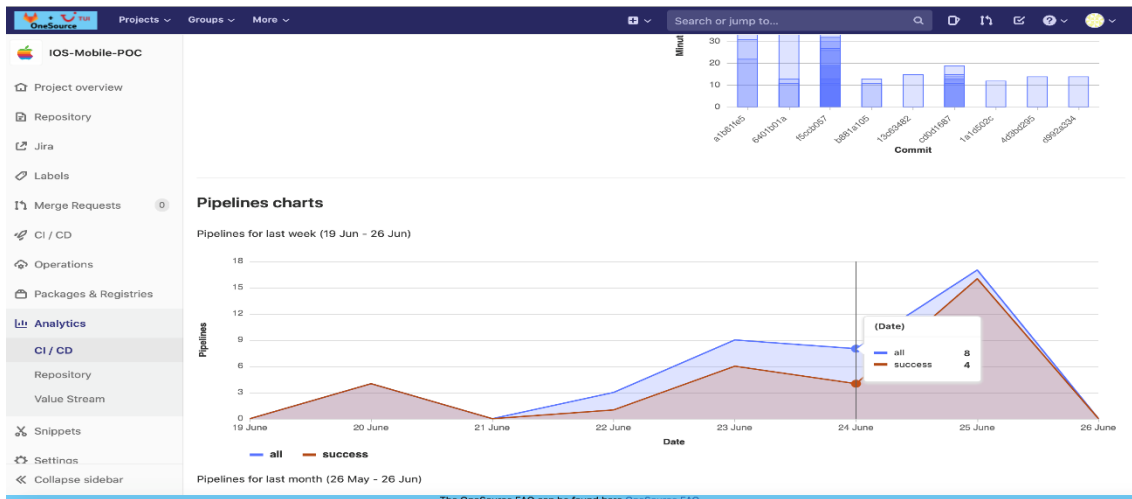


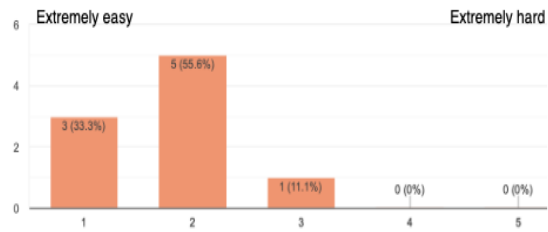
Figure 64 - Analytics CI/CD success rate' charts in Gitlab for IOS project

The final methodology's results for measuring usability and extensibility.

According to the results that are shown in Figure 65, all the members of teams think that Gitlab achieved the usability and has a friendly GUI, while most of them think that it achieved the extensibility, and no one is against adopting this solution instead of the old tools.

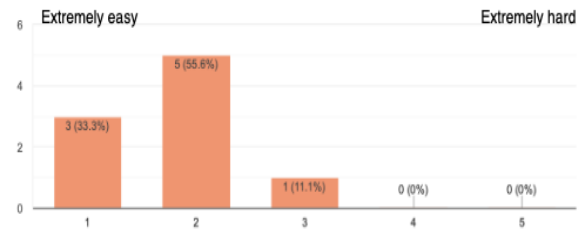
2. At first impression, do you think understanding the logic of Gitlab was:

9 responses



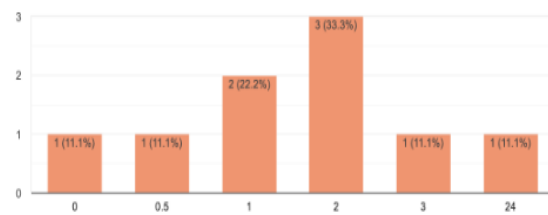
3. What about learning the initial steps to start working with Gitlab? are they:

9 responses



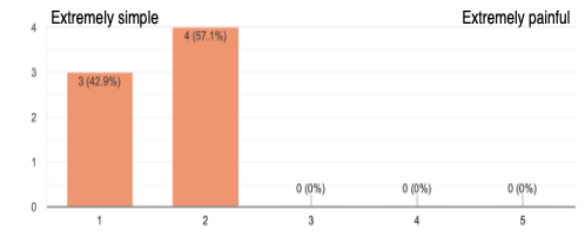
4. How many hours did you need to learn how to work with Gitlab, and Gitlab CI? (learning time)

9 responses



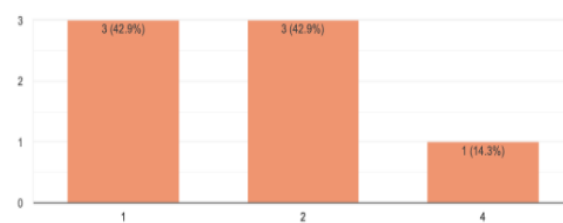
5. Completing your tasks using Gitlab was:

7 responses



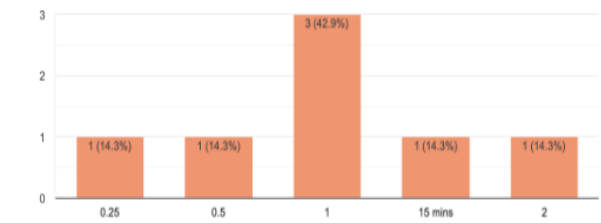
6. How many times did you face barriers during completing one task on average?

7 responses



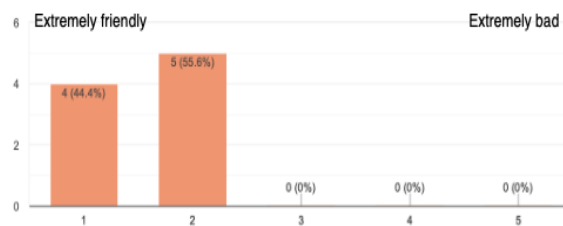
7. How many hours did you need to complete one task with Gitlab on average? (completion time)

7 responses



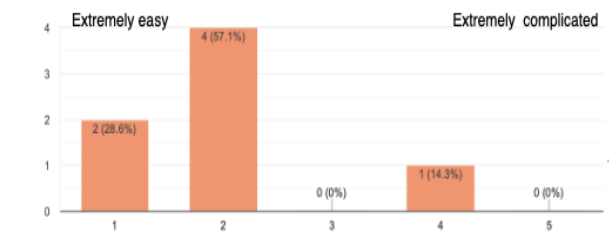
8. In terms of the GUI (the interfaces) of Gitlab, do you think the Gui is:

9 responses



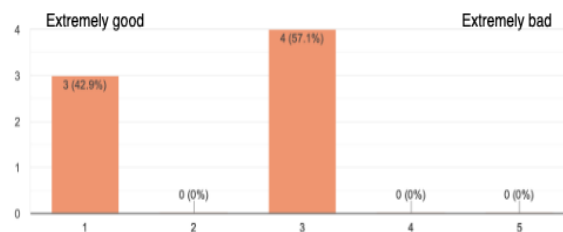
9. If you want to add many projects to your Gitlab, is the extensibility:

7 responses



11. what do you think about the performance of Gitlab in case it had many projects and many runners?

7 responses



13. If the company wants to replace Jenkins and Buddybuild with just one CI/CD tool (Gitlab), are you:

9 responses

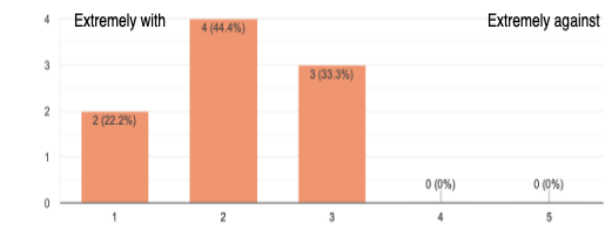


Figure 65 - Gitlab survey results

The comparison questions' results.

In terms of the usability, Gitlab outperformed its peers, while Jenkins and Gitlab had the same results in the performance questions, and Jenkins considered as the richer tool with the plugins.

12. Comparing between Jenkins, buddybuild, and Gitlab, which tool of them:

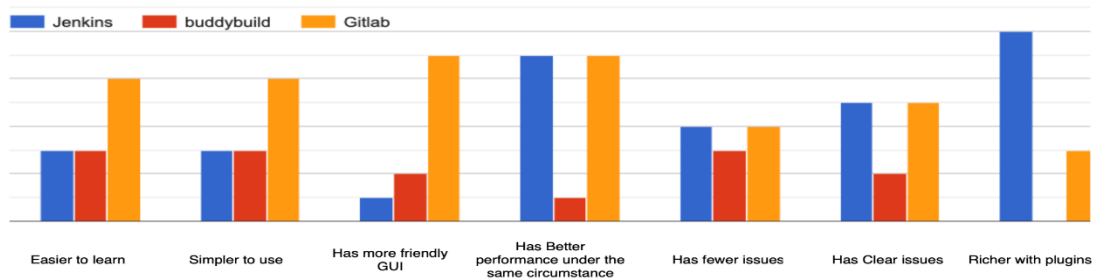


Figure 66 - Comparison between Gitlab, Jenkins, and BuddyBuild

The thesis's goals have been achieved after considering the evaluation results, as the CI/CD pipelines' of Android and IOS projects were speeded up using a new architecture with Gitlab CI, instead of Jenkins and BuddyBuild, besides that the Gitlab provides the metrics system, and its GUI gained the satisfaction of the developers and the DevOps team, this leads that the company will adopt this solution soon, after adding more features to the solution's architecture, that will be mentioned later in future work.

7 Conclusion

This chapter will contain a summary of what has been done in this thesis and the future work of the project.

7.1 Summary

This project covered the CI/CD pipelines' problems for TUI's mobile applications, as the pipelines suffered from slow builds, needed metrics system to monitor their performance, and the company wanted to replace the current CI/CD tools with just one tool, that can handle all the automated builds, tests, and releases steps in both of Android and IOS projects.

To achieve these goals, the most CI/CD tools in the IT labor market has been studied and investigated in terms of the customer priorities and requirements, and after making the required comparison among these tools, Gitlab CI tool has been selected to implement the solution and reach the targets of the thesis.

Considering the resources of the company and the expenses of the servers, like AWS virtual machines, many solutions architectures were analyzed, and the Kubernetes clusters & Docker image solution has been chosen for the Android project, while the Mac servers solution was adopted for the IOS one.

The Android solution has been implemented using the EKS service provided by AWS, where a group of virtual machines in the Kubernetes cluster were used to run the Gitlab's pipeline, besides using Docker image to organize the machines' environment, and AWS S3 service to store the builds' caches.

On the other side, IOS solution used the physical Mac servers to run the Gitlab's pipeline, and S3 to store the dependencies of the project.

To evaluate the results of the new pipelines, the metrics system that is provided by Gitlab, calculated and showed the pipelines times and the success rate graphs, where it displayed that the Android's pipeline is two times faster than the old one, and the IOS's pipeline is so much faster than the old pipelines in Jenkins and BuddyBuild, while the success rate graphs couldn't provide real results yet, because it requires a long time to decide the stability and the reliability of the pipelines.

Another way was provided to evaluate more factors in Gitlab tool, Survey was published for the Development and DevOps team at the end of the internship, where the developers thought the new solution had a friendly GUI and was easy to learn and work with (usability), while the DevOps team thought that Gitlab CI had the ability to add more projects and runners (extensibility).

As a result, Gitlab CI reached the purpose of the thesis.

7.2 Future Work

Although the project's goals were achieved, it's not the optimal case, where the project requires more enhancements can be added in the future, in terms of the following levels:

- Automate uploading the new versions of the Apps to the App markets, by adding new steps to the pipeline.
- Automate renewing the code signing certificates annually, where they are expired every 12 months.
- Making the pipelines faster and fully automated.
- Improving a build list of apps using different approaches.
- Finding a way to build multi apps using one input.
- implementing the autoscaling concept in Gitlab and EKS cluster to create the nodes when needed and destroy them later, this reduces the expenses of the machines.
- Automate the UI tests (GUI tests) before releasing the App.
- Developing Lambda function in AWS (AWS-Lambda, 2020) to spin up the cluster in the morning and break it down in the night, to reduce the expenses too.
- Using Terraform to have the infrastructure as code (Terraform by HashiCorp, 2020), this automates all the configurations and processes in the AWS side, instead of creating and configuring the machines manually.

References

(Agile 2020) Agile, 2020. *What Is Agile? | Atlassian*. [online] Atlassian. Available at: <<https://www.atlassian.com/agile>> [Accessed 28 June 2020].

(Agile-Jira 2020) Agile-Jira, 2020. *Agile Tools For Software Teams - Jira Software | Atlassian*. [online] Atlassian. Available at: <<https://www.atlassian.com/software/jira/agile>> [Accessed 28 June 2020].

(Apple Developer. 2020) Apple Developer. (2020). *TestFlight - Apple Developer*. [online] Available at: <https://developer.apple.com/testflight/> [Accessed 20 Jan. 2020].

(Android-Lint 2020) Android-Lint, 2020. *Android Lint - Android Studio Project Site*. [online] Tools.android.com. Available at: <<http://tools.android.com/tips/lint>> [Accessed 20 June 2020].

(Android-unit-test 2020) Android-unit-test, 2020. [online] Available at: <<https://developer.android.com/training/testing/unit-testing>> [Accessed 20 June 2020].

(Atlassian. 2020) Atlassian. (2020). *Bamboo Continuous Integration and Deployment Build Server*. [online] Available at: <https://www.atlassian.com/software/bamboo> [Accessed 23 Jan. 2020].

(Atlassian-Configuring plans 2020) Atlassian-Configuring plans (2020). *Configuring plans - Atlassian Documentation*. [online] Confluence.atlassian.com. Available at: <https://confluence.atlassian.com/bamboo/configuring-plans-289276853.html> [Accessed 23 Jan. 2020].

(Atlassian-installing Bamboo 2020) Atlassian-installing Bamboo (2020). *Installing Bamboo on Mac OS X - Atlassian Documentation*. [online] Confluence.atlassian.com. Available at: <https://confluence.atlassian.com/bamboo/installing-bamboo-on-mac-os-x-289276789.html> [Accessed 23 Jan. 2020].

(AWS-CLI 2020) AWS-CLI, 2020. *Install The AWS CLI Version 1 On MacOS - AWS Command Line Interface*. [online] Docs.aws.amazon.com. Available at: <<https://docs.aws.amazon.com/cli/latest/userguide/install-macos.html>> [Accessed 1 June 2020].

(AWS-EC2 on demand 2020) AWS-EC2 on demand, 2020. *Amazon EC2 On Demand - Preço Sob Demanda - AWS*. [online] Amazon Web Services, Inc. Available at: <<https://aws.amazon.com/pt/ec2/pricing/on-demand/>> [Accessed 30 June 2020].

(AWS-EKS-Connection 2020) AWS-EKS-Connection, 2020. *Connect To Amazon EKS Cluster*. [online] Amazon Web Services, Inc. Available at: <<https://aws.amazon.com/premiumsupport/knowledge-center/eks-cluster-connection/>> [Accessed 1 June 2020].

(AWS-eksctl 2020) AWS-eksctl, 2020. *Getting Started With Eksctl - Amazon EKS*. [online] Docs.aws.amazon.com. Available at: <<https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html>> [Accessed 1 June 2020].

(AWS-Lambda 2020) AWS-Lambda, 2020. *AWS Lambda – Product Features*. [online] Amazon Web Services, Inc. Available at: <<https://aws.amazon.com/lambda/features/>> [Accessed 29 June 2020].

(AWS-s3 2020) AWS-s3, 2020. *Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3)*. [online] Amazon Web Services, Inc. Available at:

<https://aws.amazon.com/s3/?sc_channel=PS&sc_campaign=acquisition_PT&sc_publisher=google&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CStorage%7CS3%7CPT%7CEN%7CText&sc_content=s3_e&sc_detail=aws%20s3&sc_category=Storage&sc_segment=293614596600&sc_matchtype=e&sc_country=PT&skwid=AL!4422!3!293614596600!e!!g!!aws%20s3&ef_id=CjwKCAjwrcH3BRApEiwAxjdPTUo2AN28ft76b6GT8cZ4zY3alM_cFuCiVoyW_D8-ux8RaoNBdop0jxoC_twQAvD_BwE:G:s&skwid=AL!4422!3!293614596600!e!!g!!aws%20s3> [Accessed 22 June 2020].

(Bamboo 2020) Bamboo (2020). *Configuration as Code in Bamboo | Atlassian*. [online] Atlassian. Available at: <https://www.atlassian.com/company/events/summit-europe/watch-sessions/2017/build-deploy/configuration-as-code-in-bamboo> [Accessed 23 Jan. 2020].

(Bamboo - Features 2020) Bamboo - Features (2020). *Bamboo - Features | Atlassian*. [online] Atlassian. Available at: <https://www.atlassian.com/software/bamboo/features> [Accessed 23 Jan. 2020].

(Bamboo-Specs 2020) Bamboo-Specs (2020). *Bamboo Specs Reference*. [online] Docs.atlassian.com. Available at: <https://docs.atlassian.com/bamboo-specs-docs/6.10.4/specs.html?java#introduction> [Accessed 23 Jan. 2020].

(Bitrise-Capterra 2020) Bitrise-Capterra (2020). *Bitrise Reviews 2020 - Capterra*. [online] Capterra.com. Available at: <https://www.capterra.com/p/178004/Bitrise/reviews/> [Accessed 21 Jan. 2020].

(Bitrise-CLI 2020) Bitrise-CLI (2020). *Bitrise Docs*. [online] Bitrise Docs. Available at: <https://devcenter.bitrise.io/bitrise-cli/index/> [Accessed 20 Jan. 2020].

(Bitrise-infrastructure 2020) Bitrise-infrastructure (2020). *Bitrise Docs*. [online] Bitrise Docs. Available at: <https://devcenter.bitrise.io/infrastructure/infrastructure-index/> [Accessed 21 Jan. 2020].

(Bitrise - Mobile Continuous Integration and Delivery. 2020) Bitrise - Mobile Continuous Integration and Delivery. (2020). *Bitrise - Mobile Continuous Integration and Delivery*. [online] Available at: <https://www.bitrise.io> [Accessed 20 Jan. 2020].

(Bitrise-virtual-machines 2020) Bitrise-virtual-machines (2020). *Bitrise Docs*. [online] Bitrise Docs. Available at: <https://devcenter.bitrise.io/infrastructure/virtual-machines/> [Accessed 21 Jan. 2020].

(Blog.jetbrains.com 2020) Blog.jetbrains.com. (2020). *Configuration as Code, Part 1: Getting Started with Kotlin DSL | TeamCity Blog*. [online] Available at: <https://blog.jetbrains.com/teamcity/2019/03/configuration-as-code-part-1-getting-started-with-kotlin-dsl/> [Accessed 15 Jan. 2020].

(Bouchereau and Rowlands 2020) Bouchereau, V. and Rowlands, H. (2020). [online] Available at: https://www.researchgate.net/profile/Hefin_Rowlands/publication/235276199_Methods_and_techniques_to_help_quality_function_deployment_QFD/links/55dadfe308aeb38e8a8a2676.pdf [Accessed 11 Feb. 2020].

(buddy-action 2020) buddy-action (2020). *Builds & testing | Docs | Buddy: The DevOps Automation Platform*. [online] Buddy: The DevOps Automation Platform. Available at: <https://buddy.works/docs/pipelines/builds-and-testing> [Accessed 6 Jan. 2020].

(buddy build-apple 2020) buddy build-apple (2020). *The buddybuild team is now part of Apple!*. [online] Buddybuild.com. Available at: <https://www.buddybuild.com/blog/buddybuild-is-now-part-of-apple> [Accessed 20 Jan. 2020].

(Buddybuild.com 2020) Buddybuild.com. (2020). *Mobile Continuous Integration & Deployment for iOS / Buddybuild*. [online] Available at: <https://www.buddybuild.com> [Accessed 20 Jan. 2020].

(BuddyBuild-Release 2020) BuddyBuild-Release (2020). *A 100% automated build & release process for iOS apps / buddybuild*. [online] Buddybuild.com. Available at: <https://www.buddybuild.com/blog/automating-ios-releases> [Accessed 20 Jan. 2020].

(buddy-review 2020) buddy-review (2020). *Buddy Reviews 2020 - Capterra*. [online] Capterra.com. Available at: <https://www.capterra.com/p/157673/Buddy/reviews/> [Accessed 6 Jan. 2020].

(Buddy: The DevOps Automation Platform 2020) Buddy: The DevOps Automation Platform. (2020). *Buddy: The DevOps Automation Platform*. [online] Available at: <https://buddy.works> [Accessed 6 Jan. 2020].

(Business.qld.gov.au 2020) Business.qld.gov.au. (2020). *New product development | Business Queensland*. [online] Available at: <https://www.business.qld.gov.au/running-business/growing-business/becoming-innovative/developing-products/new-products> [Accessed 28 Jan. 2020].

(checkstyle 2020) checkstyle, 2020. *Checkstyle – Checkstyle 8.33*. [online] Checkstyle.org. Available at: <https://checkstyle.org/index.html> [Accessed 20 June 2020].

(CircleCI. 2020) CircleCI. (2020). *Continuous Integration and Delivery*. [online] Available at: <https://circleci.com> [Accessed 16 Jan. 2020].

(CircleCI-config.yml 2020) CircleCI-config.yml (2020). *Sample 2.0 config.yml Files - CircleCI*. [online] Circleci.com. Available at: <https://circleci.com/docs/2.0/sample-config> [Accessed 21 Jan. 2020].

(CircleCI-macos 2020) CircleCI-macos, 2020. *Choosing An Executor Type - Circleci*. [online] Circleci.com. Available at: <https://circleci.com/docs/2.0/executor-types/#using-macos> [Accessed 12 June 2020].

(CircleCI-start 2020) CircleCI-start (2020). *Getting Started Introduction - CircleCI*. [online] Circleci.com. Available at: <https://circleci.com/docs/2.0/getting-started/#section=getting-started> [Accessed 20 Jan. 2020].

(COCUS AG 2019) COCUS AG. (2019). *Your Solution provider for digitalisation - COCUS AG*. [online] Available at: <https://www.cocus.de/en/home/> [Accessed 30 Dec. 2019].

(Code Signing - Support - Apple Developer 2020) Developer.apple.com. 2020. *Code Signing - Support - Apple Developer*. [online] Available at: <https://developer.apple.com/support/code-signing/> [Accessed 21 June 2020].

(DeployPlace. 2020) DeployPlace. (2020). *Bamboo vs Jenkins vs Travis - Full Comparison of Continuous Integration Tools - DeployPlace*. [online] Available at: <https://deployplace.com/blog/bamboo-vs-jenkins-vs-travis/> [Accessed 23 Jan. 2020].

(Despa 2020) Despa, V. (2020). *YouTube*. [online] Youtube.com. Available at: https://www.youtube.com/watch?v=QhncdC_Q58c [Accessed 16 Jan. 2020].

(detekt 2020) Mindorks.com. 2020. *Detekt*. [online] Available at: <<https://mindorks.com/android/store/Static-Analysis-Tools/arturbosch/detekt>> [Accessed 20 June 2020].

(Dewulf, 2013) Kristel Dewulf (2013). *Sustainable Product Innovation: The Importance of the Front- End Stage in the Innovation Process*. INTECH Open Access Publisher.

(Disputesoft.com. 2020) Disputesoft.com. 2020. *Diving Into Code Quality: Factors Affecting Code Quality – Disputesoft*. [online] Available at: <<https://www.disputesoft.com/diving-into-code-quality-factors-affecting-code-quality/>> [Accessed 20 June 2020].

(Docker-installing 2020) Docker-installing, 2020. *Get Docker*. [online] Docker Documentation. Available at: <<https://docs.docker.com/get-docker/>> [Accessed 4 June 2020].

(Droids Roids - Poland 2020) Droids Roids - Poland (2020). *How to Configure Bitrise Workflows for Android Project*. [online] iOS & Android Mobile App Development Company - Droids On Roids - Poland. Available at: <https://www.thedroidsonroids.com/blog/how-to-configure-bitrise-workflows-for-your-android-project> [Accessed 21 Jan. 2020].

(Drone-installation 2020) Drone-installation (2020). *Overview / Drone*. [online] Docs.drone.io. Available at: <https://docs.drone.io/installation/overview/> [Accessed 21 Jan. 2020].

(Drone.io 2020) Drone.io. (2020). *Drone CI – Automate Software Testing and Delivery*. [online] Available at: <https://drone.io> [Accessed 21 Jan. 2020].

(drone-reviews 2020) drone-reviews (2020). [online] Available at: <https://www.g2.com/products/drone-io/reviews> [Accessed 21 Jan. 2020]

(ecr-aws 2020) ecr-aws, 2020. *Amazon ECR | Amazon Web Services*. [online] Amazon Web Services, Inc. Available at: <<https://aws.amazon.com/ecr/>> [Accessed 28 May 2020].

(Edureka Community 2020) Edureka Community. (2020). *what is the key difference between Declarative pipeline and scripted pipeliine*. [online] Available at: <https://www.edureka.co/community/54705/difference-between-declarative-pipeline-scripted-pipeliine> [Accessed 2 Jan. 2020].

(Ezugbaya 2019) Ezugbaya, A., 2019. *Continuous Deployment For Cross- Platform Mobile Application*. [online] Available at: <<https://www.theseus.fi/bitstream/handle/10024/227482/Thesis%202.pdf?sequence=2&isAllowed=y>> [Accessed 14 April 2019].

(fastlane 2020) fastlane, 2020. *Fastlane - App Automation Done Right*. [online] fastlane. Available at: <<https://fastlane.tools>> [Accessed 20 June 2020].

(Forrester.com 2020) Forrester.com. (2020). *The Forrester Wave™: Cloud-Native Continuous Integration Tools, Q3 2019*. [online] Available at: <https://www.forrester.com/report/The+Forrester+Wave+CloudNative+Continuous+Integration+Tools+Q3+2019/-/E-RES148217> [Accessed 20 Jan. 2020].

(Forsgren 2015) Forsgren, N 2015, *2015 State of DevOps Report, from ResearchGate website* <https://www.researchgate.net/publication/302566896> 2015 State of DevOps Report

(Frokjaer 2000) Frokjaer, E., 2000. 345. [online] Available at: <https://www.researchgate.net/profile/Erik_Frokjaer/publication/200553152_Measuring_usability_are_effectiveness_efficiency_and_satisfaction_really_correlated/links/0fcfd512924c960fe0000000/Measuring-usability-are-effectiveness-efficiency-and-satisfaction-really-correlated.pdf> [Accessed 6 April 2000].

(GitHub-drone 2020) GitHub-drone (2020). *drone/drone*. [online] GitHub. Available at: <https://github.com/drone/drone> [Accessed 21 Jan. 2020].

(GitLab. 2020) GitLab. (2020). *GitLab Demo*. [online] Available at: <https://about.gitlab.com/demo/> [Accessed 23 Jan. 2020].

(GitLab-analytics 2020) Gitlab-analytics, 2020. [online] Available at: <<https://docs.gitlab.com/ee/ci/pipelines/#pipeline-success-and-duration-charts>> [Accessed 17 May 2020].

(Gitlab-ci.yml 2020) Gitlab-ci.yml (2020). *lib/gitlab/ci/templates/Gradle.gitlab-ci.yml · master · GitLab.org / GitLab FOSS*. [online] GitLab. Available at: <https://gitlab.com/gitlab-org/gitlab-foss/blob/master/lib/gitlab/ci/templates/Gradle.gitlab-ci.yml> [Accessed 27 Jan. 2020].

(Gitlab-Doc 2020) Gitlab-Doc, 2020. [online] Available at: <https://docs.gitlab.com/ee/user/gitlab_com/> [Accessed 24 May 2020].

(Gitlab-Forrester 2020) Gitlab-Forrester (2020). *GitLab Continuous Integration named a Leader in the Forrester Wave™*. [online] GitLab. Available at: <https://about.gitlab.com/blog/2017/09/27/gitlab-leader-continuous-integration-forrester-wave/> [Accessed 26 Jan. 2020].

(Gitlab-Helm Chart 2020) Gitlab-Helm Chart, 2020. [online] Available at: <<https://docs.gitlab.com/runner/install/kubernetes.html>> [Accessed 22 June 2020].

(Gitlab-Kubernetes 2020) Gitlab-Kubernetes, 2020. [online] Available at: <https://docs.gitlab.com/ee/user/project/clusters/add_eks_clusters.html> [Accessed 1 June 2020].

(Gitlab-pipeline 2020) Gitlab-pipeline (2020). [online] Available at: <https://docs.gitlab.com/ee/ci/pipelines.html> [Accessed 27 Jan. 2020].

(Gitlab-runner 2020) Gitlab-runner (2020). [online] Available at: https://docs.gitlab.com/ee/ci/quick_start/README.html#configuring-a-runner [Accessed 27 Jan. 2020].

(Gitlab-Runner-Cache 2020) Gitlab-Runner-Cache, 2020. [online] Available at: <<https://docs.gitlab.com/runner/configuration/advanced-configuration.html#the-runnerscache-section>> [Accessed 22 June 2020].

(GitLab-runner-doc 2020) Gitlab-runner-doc, 2020. [online] Available at: <<https://docs.gitlab.com/runner/>> [Accessed 21 June 2020].

(Gitlab-Register-Runner 2020) Gitlab-Register-Runner, 2020. [online] Available at: <<https://docs.gitlab.com/runner/register/>> [Accessed 24 May 2020].

(Gitlab-values.yml 2020) GitLab-values.yml, 2020. *Values.Yaml · Master · Gitlab.Org / Charts / Gitlab Runner*. [online] GitLab. Available at: <<https://gitlab.com/gitlab-org/charts/gitlab-runner/blob/master/values.yml>> [Accessed 22 June 2020].

(Google Forms: Free Online Surveys for Personal Use 2020) Google.com. 2020. *Google Forms: Free Online Surveys For Personal Use*. [online] Available at: <<https://www.google.com/forms/about/>> [Accessed 17 May 2020].

(Guru99.com. 2020) Guru99.com. (2020). *Jenkins vs Travis-CI: What is the difference?*. [online] Available at: <https://www.guru99.com/jenkins-vs-travis.html> [Accessed 8 Jan. 2020].

(Haksever, Chaganti and Cook 2004) Haksever, C., Chaganti, R. and Cook, R. (2004). A Model of Value Creation: Strategic View. *Journal of Business Ethics*, 49(3), pp.295-307.

(Installing Helm 2020) Helm.sh. 2020. Installing Helm. [online] Available at: <<https://helm.sh/docs/intro/install/>> [Accessed 22 June 2020].

(Isoconsultantpune.com 2020) Isoconsultantpune.com. (2020). [online] Available at: <http://isoconsultantpune.com/wp-content/uploads/2015/04/Example-of-QFD.pdf> [Accessed 11 Feb. 2020].

(Jane 2020) Jane (2020). *Ci/CD platform with drone and gogs*. [online] Slideshare.net. Available at: <https://www.slideshare.net/PolJane2/cicd-platform-with-drone-and-gogs> [Accessed 21 Jan. 2020].

(Jenkins 2020) Jenkins. (2020). *Jenkins*. [online] Available at: <https://jenkins.io> [Accessed 2 Jan. 2020].

(Jenkins-parameters 2020) Jenkins-parameters, 2020. *Pipeline Syntax*. [online] Pipeline Syntax. Available at: <<https://www.jenkins.io/doc/book/pipeline/syntax/#parameters>> [Accessed 9 June 2020].

(JetBrains 2020) JetBrains. (2020). *Build Management & Continuous Integration - Features | TeamCity*. [online] Available at: <https://www.jetbrains.com/teamcity/features/> [Accessed 14 Jan. 2020].

(Jira 2020) Jira (2020). Agile tools for software teams - Jira Software | Atlassian. [online] Atlassian. Available at: <https://www.atlassian.com/software/jira/agile> [Accessed 13 Feb. 2020].

(Katalon Solution 2020) Katalon Solution. (2020). *Best 14 CI/CD Tools You Must Know | Updated for 2019*. [online] Available at: <https://www.katalon.com/resources-center/blog/ci-cd-tools/> [Accessed 2 Jan. 2020].

(Koen et al., 2001, p.47) Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'Amore, et al. (2001). Providing clarity and a common language to the "fuzzy front end." *Research- Technology Management*, 44(2), 46-55. Retrieved from <http://www.ingentaconnect.com/content/iri/rtm/2001/00000044/00000002/art00009>.

(Kubernetes 2020) Kubernetes, 2020. *Building Large Clusters*. [online] Kubernetes.io. Available at: <<https://kubernetes.io/docs/setup/best-practices/cluster-large/>> [Accessed 28 May 2020].

(Kubernetes-aws 2020) Kubernetes-aws, 2020. *Kubernetes On AWS | AWS*. [online] Amazon Web Services, Inc. Available at: <<https://aws.amazon.com/kubernetes/>> [Accessed 28 May 2020].

(Mcleod 2020) Mcleod, S., 2020. *Likert Scale Definition, Examples And Analysis | Simply Psychology*. [online] Simplypsychology.org. Available at: <<https://www.simplypsychology.org/likert-scale.html>> [Accessed 18 May 2020].

(Migrating from GitLab - CircleCI 2020) Migrating from GitLab - CircleCI (2020). *Migrating from GitLab - CircleCI*. [online] Circleci.com. Available at: <https://circleci.com/docs/2.0/migrating-from-gitlab/> [Accessed 20 Jan. 2020].

(Mode 2020) Mode, D. (2020). *What is REST – Learn to create timeless REST APIs*. [online] Restfulapi.net. Available at: <https://restfulapi.net> [Accessed 16 Jan. 2020].

(Overview - CircleCI 2020) Overview - CircleCI (2020). *Overview - CircleCI*. [online] Circleci.com. Available at: <https://circleci.com/docs/2.0/about-circleci/> [Accessed 16 Jan. 2020].

(pipeline-examples 2020) pipeline-examples (2020). *Deployment pipeline | Docs | Buddy: The DevOps Automation Platform*. [online] Buddy: The DevOps Automation Platform. Available at: <https://buddy.works/docs/pipeline-examples/deployment-pipeline> [Accessed 6 Jan. 2020].

(Pipeline Syntax 2020) Pipeline Syntax. (2020). *Pipeline Syntax*. [online] Available at: <https://jenkins.io/doc/book/pipeline/syntax/#declarative-pipeline> [Accessed 2 Jan. 2020]

(product - CircleCI 2020) product - CircleCI (2020). *Continuous Integration Product and Features*. [online] CircleCI. Available at: <https://circleci.com/product/> [Accessed 20 Jan. 2020].

(Ramalho 2018) Ramalho, F. (2018). *Application at Glintt*.

(Reddit-drone 2020) Reddit-drone (2020). *[Looking for Opinions] ConcourseCI vs Drone.io : devops*. [online] Reddit.com. Available at: https://www.reddit.com/r/devops/comments/7oyah8/looking_for_opinions_concourseci_vs_droneio/ [Accessed 21 Jan. 2020].

(React Native 2020) React Native, 2020. *React Native · A Framework For Building Native Apps Using React*. [online] Reactnative.dev. Available at: <https://reactnative.dev> [Accessed 11 June 2020].

(Releases 2020) Releases, G., 2020. *Gradle Build Tool*. [online] Gradle. Available at: <https://gradle.org> [Accessed 20 June 2020].

(REST API-Teamcity 2020) REST API-Teamcity (2020). *REST API - Help | TeamCity*. [online] JetBrains.com. Available at: <https://www.jetbrains.com/help/teamcity/rest-api.html> [Accessed 16 Jan. 2020].

(Rich and Holweg 2000) Rich, N. and Holweg, M. (2000). *VALUE ANALYSIS*. [ebook] pp.2, 3. Available at: https://www.urenio.org/tools/en/value_analysis.pdf [Accessed 28 Jan. 2020].

(Salminen 2020) Salminen, H., 2020. [online] Trepo.tuni.fi. Available at: <https://trepo.tuni.fi/bitstream/handle/123456789/27193/Salminen.pdf?sequence=4&isAllowed=y> [Accessed 2 February 2019].

(Savvy Apps. 2020) Savvy Apps. (2020). *How-to Guide: Setting Up Continuous Integration for iOS with Bitrise*. [online] Available at: <https://savvyapps.com/blog/continuous-integration-ios-bitrise> [Accessed 21 Jan. 2020].

(Shanker 2012) Shanker, A. (2012). *Q&A: What Is Customer Value and How Do You Deliver It?*. [online] Timreview.ca. Available at: <https://timreview.ca/article/525> [Accessed 30 Jan. 2020].

(Slack-Bamboo 2020) Slack-Bamboo (2020). *Slack Task for Bamboo*. [online] Atlassian Marketplace. Available at: <https://marketplace.atlassian.com/apps/1217681/slack-task-for-bamboo?hosting=server&tab=overview> [Accessed 23 Jan. 2020].

(Slant and Pipelines 2020) Slant, C. and Pipelines, B., 2020. *Slant - Circleci Review*. [online] Slant. Available at: <https://www.slant.co/options/625/~circleci-review> [Accessed 3 July 2020].

(Slideshare.net. 2020) Slideshare.net. (2020). *Drone CI/CD Platform*. [online] Available at: <https://www.slideshare.net/appleboy/drone-cicd-platform> [Accessed 21 Jan. 2020].

(SonarQube 2020) SonarQube, 2020. Documentation | Sonarqube Docs. [online] Docs.sonarqube.org. Available at: <https://docs.sonarqube.org/latest/> [Accessed 3 July 2020].

(SourceTree 2020) SourceTree. (2020). *Sourcetree | Free Git GUI for Mac and Windows*. [online] Available at: <https://www.sourcetreeapp.com> [Accessed 13 Feb. 2020].

(StackShare 2020) StackShare. (2020). *What are the best Continuous Integration Tools?*. [online] Available at: <https://stackshare.io/continuous-integration> [Accessed 8 Jan. 2020].

(StackShare-Teamcity 2020) StackShare-Teamcity (2020). *Why developers like TeamCity*. [online] StackShare. Available at: <https://stackshare.io/teamcity#pros> [Accessed 16 Jan. 2020].

(Startupcraft.io 2020) Startupcraft.io. (2020). *React Native DevOps made easy. Part 1. Android*. [online] Available at: <https://startupcraft.io/blog/react-native-devops-made-easy-part1-android> [Accessed 21 Jan. 2020].

(Startupcraft.io-IOS 2020) Startupcraft.io-IOS (2020). *React Native DevOps made easy. Part 2. iOS*. [online] Medium. Available at: <https://medium.com/startupcraft/react-native-devops-made-easy-part-2-ios-767c6f904fe1> [Accessed 21 Jan. 2020].

(Statista 2019) Statista. (2019). *Number of daily Android app releases worldwide 2018 | Statista*. [online] Available at: <https://www.statista.com/statistics/276703/android-app-releases-worldwide/> [Accessed 30 Dec. 2019].

(Teamcity-Build Configurations 2020) Teamcity-Build Configurations (2020). *Creating and Editing Build Configurations - Help | TeamCity*. [online] JetBrains.com. Available at: <https://www.jetbrains.com/help/teamcity/creating-and-editing-build-configurations.html> [Accessed 15 Jan. 2020].

(TeamCity Reviews, 2020) TeamCity Reviews, 2020. *Teamcity Reviews And Pricing - 2020*. [online] Capterra.com. Available at: <https://www.capterra.com/p/136011/Teamcity/> [Accessed 3 July 2020].

(Terraform by HashiCorp 2020) Terraform by HashiCorp. 2020. *Terraform By Hashicorp*. [online] Available at: <https://www.terraform.io> [Accessed 29 June 2020].

(Tobies 2020) Tobies, V. (2020). *20 Best Travis CI Alternatives | Reviews | Pros & Cons - Alternative.me*. [online] Alternative.me. Available at: https://alternative.me/travis-ci#Pros_&_Cons_of_Travis_CI [Accessed 8 Jan. 2020].

(Tobies-Teamcity 2020) Tobies-Teamcity, V. (2020). *16 Best TeamCity Alternatives | Reviews | Pros & Cons - Alternative.me*. [online] Alternative.me. Available at: <https://alternative.me/teamcity> [Accessed 16 Jan. 2020].

(Tobies-BuddyBuild 2020) Tobies-BuddyBuild, V. (2020). *20 Best buddybuild Alternatives | Reviews | Pros & Cons - Alternative.me*. [online] Alternative.me. Available at: https://alternative.me/buddybuild#Pros_&_Cons_of_buddybuild [Accessed 20 Jan. 2020].

(Tools and Reviews 2020) Tools, B. and Reviews, J. (2020). *Jenkins Reviews & Ratings 2020 | TrustRadius*. [online] TrustRadius. Available at: <https://www.trustradius.com/products/jenkins/reviews> [Accessed 2 Jan. 2020].

(Tarvis-build 2020) Tarvis-build (2020). *Travis CI Documentation*. [online] Docs.travis-ci.com. Available at: <https://docs.travis-ci.com/user/build-stages/> [Accessed 8 Jan. 2020].

(Travis CI Enterprise 2020) Travis CI Enterprise. (2020). *Travis CI Enterprise - On Premises Continuous Integration and Deployment Platform*. [online] Available at: <https://enterprise.travis-ci.com> [Accessed 8 Jan. 2020].

(Travis-example 2020) Travis-example (2020). *Travis CI Documentation*. [online] Docs.travis-ci.com. Available at: <https://docs.travis-ci.com/user/build-stages/> [Accessed 8 Jan. 2020].

(TrustRadius-Bamboo 2020) TrustRadius-Bamboo (2020). *Bamboo Reviews & Ratings 2020 | TrustRadius*. [online] TrustRadius. Available at: <https://www.trustradius.com/products/bamboo/reviews> [Accessed 23 Jan. 2020].

(Visual Studio 2020) Visual Studio. 2020. *Visual Studio App Center FAQ | Visual Studio - Visual Studio*. [online] Available at: <https://visualstudio.microsoft.com/app-center/faq/> [Accessed 20 June 2020].

(Woodall 2003) Woodall, T. (2003). *Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis*. [ebook] Available at: https://www.researchgate.net/publication/228576532_Conceptualising_27Value_for_the_Customer_27_An_Attributional_Structural_and_Dispositional_Analysis [Accessed 30 Jan. 2020].

(Xcode 2020) Xcode, 2020. *Xcode - Apple Developer*. [online] Apple Developer. Available at: <https://developer.apple.com/xcode/> [Accessed 3 July 2020].