Talentua: university environment within the context of virtual campus and the company-student matching (part III)

Dyhagho Briceño Pizarro

Resum– Talentua és un projecte que es consolida al juny de 2018 amb l'objectiu de ser un punt de trobada per estudiants de l'Escola d'Enginyeria de la Universitat Autònoma de Barcelona (UAB). El pas dels mesos i la participació en el concurs Startup Lab UAB 2019 fan que el projecte guanyi dimensió, esdevenint una plataforma LMS (Learning Management System) enriquida i una segona plataforma destinada a fer matching entre estudiants i empreses.

L'ús de tecnologies punteres en el desenvolupament de les plataformes web així com la utilització de tècniques d'intel·ligència artificial per a millorar els serveis oferts són els pilars fonamentals la vesant tècnica de Talentua.

Paraules clau– Cloud computing, DevOps, Docker, Jenkins, Amazon Web Services, LMS, talent, emprenedoria, seguretat, Machine Learning, reclutament

Abstract– Talentua is a project that was consolidated in June 2018 to be a meeting point for students of the Universitat Autònoma de Barcelona (UAB) School of Engineering. With the passing of the months and the participation in the UAB Startup Lab 2019 competition make the project gain in size, becoming an enriched LMS (Learning Management System) and a second platform for matching between students and companies.

The use of cutting-edge technologies in the development of web platforms as well as the use of artificial intelligence techniques to improve the services offered are the fundamental pillars of the technical aspect of this project.

Keywords– Cloud computing, DevOps, Docker, Jenkins, Amazon Web Services, LMS, talent, entrepreurship, security, Machine Learning, recruitment

1 Introduction

Talent management (TM) is a set of Human Resources (HR) processes to attract, develop, motivate, and retain employees to increase their performance. These processes are important for any organization to succeed. Julie Brandt points out that "TM real value allows organizations to identify high performers and future leaders, track and evaluate employee performance, and identify and address skill gaps with targeted training and development" [1].

The inability for detecting and retaining their university graduates and highly skilled professionals made Spain one of the most damaged European countries by the brain drain effect during the economic crisis [2]. This fact is reinforced by figure 1, where it can be seen that around 90,000 Spaniards with higher education emigrated to other EU countries.

• Contact e-mail: dyhagho.briceno@gmail.com
• Specialization: Information Technologies
• Project supervised by: Daniel Franco Puntes (DACSO)
• Academic year 2019/20

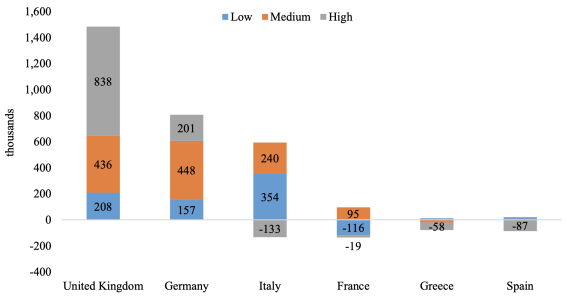

Fig. 1: Net migration within the EU by education level over the period 2007-2017. From [3]

Moreover, today almost half of their skilled workers end up emigrating abroad. And, as a result of this exit, those who emigrate have more educational level than those who stay. The investigation [3] emphasises that Spain, Italy and Greece face a potential decline in their workforce in the coming years due to the ageing population.

2 Objectives

The main objective should be to detect, retain and motivate the talent. In educational institutions, teachers should be the ones in charge to reach these. While in the companies, this job should be done by the HR team. In both cases, the solutions requires investing time, money and human resources.

Since the solution above solution requires a lot of limited results; an alternative solution that saves some of these resources should be planned. In the following section, a technological solution is proposed.

3 Proposed solution

To solve the detected problem it was proposed, in conjunction with two engineering students (Alan Fusté Rodríguez and Xavier Velasco Llauradó), to create a web platform using the knowledge acquired during their period in the university. As observed in figure 2, this platform is divided into two different application:

As can see in figure 2, the technologies for the web development of the project are React, Bootstrap and Django framework; on the AI Core uses Python for ML; and *PostgreSQL* for storing the information.

- The first tool is a LMS platform named as Smart Campus. The word "smart" comes from the fact that it takes some defined indicators to detect and motivate talent using Machine Learning. This platform should be modular to easily add or remove extra functionalities.
- The other tool is a website platform to do a matching between students and companies that are keen on finding potential talent. A simplified idea of how this website works would be: a company entering a set of desired skills in that the candidates should have and after clicking the searching button it will show the best profiles for the company necessities. Moreover, this second tool ease the job of TM that have the HR team.

The idea is to unify Talentua and Smart Campus through an internal server called AI Core. This server will collect a series of indicators extracted from both platforms and apply Machine Learning (ML) algorithms to generate the processed interesting data to serve. Considering that each member of the team has to do two end-of-degree projects, the project has been divided into six parts to have a Minimum Value Product (MVP) after finishing all six degrees projects. The first phase (parts I-III) of the project are developed between the last quarter of 2019 and January 2020. This parts are I, II and III. The parts I and II are focused on the development of the Smart Campus web platform. Alan Fusté and Xavier Velasco were in charge of this job, tutored by Rubén Rubio Barrera.

To what concerns to part III, mentored by the Professor Daniel Franco Puntes (from now on, the tutor), the idea was to create a development environment that eases the process of adding new functionalities to the platform. This was achieved by building a Cloud-based pipeline using some set of *DevOps* tools that will be discussed later on.

After finishing this first phase, it is planned that the second phase begins the February's third week of 2020 and concluding on the second trimester of the same year. This second phase it would be mainly focused on the development of different features for the IA Core.

The first version of this platform, which is estimated to be ready with the conclusion of the second phase summer of 2020, would not be a marketable version but a version to be tested under a supervised environment.

4 State of Art

4.1 Virtualization

Virtualization aims to isolate applications and its dependencies in self-contained units that are able to be executed anywhere.

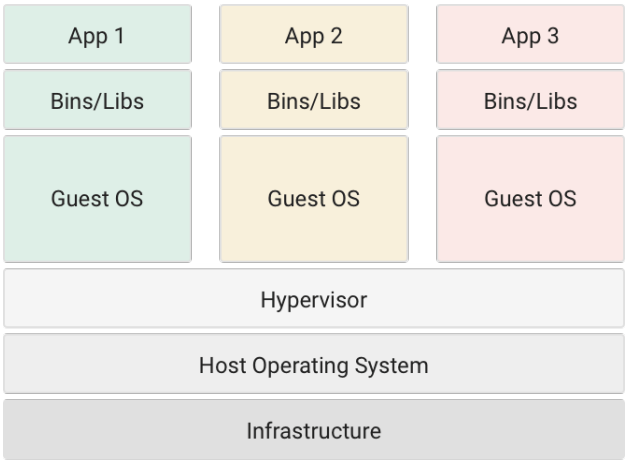

Fig. 3: Virtual Machine Stack. Adapted from Google Documentation [4]

Virtual machines (VMs) are a type of virtualization that are executed on a physical or *host machine* simulating the behaviour of a real computer. A particular VM has

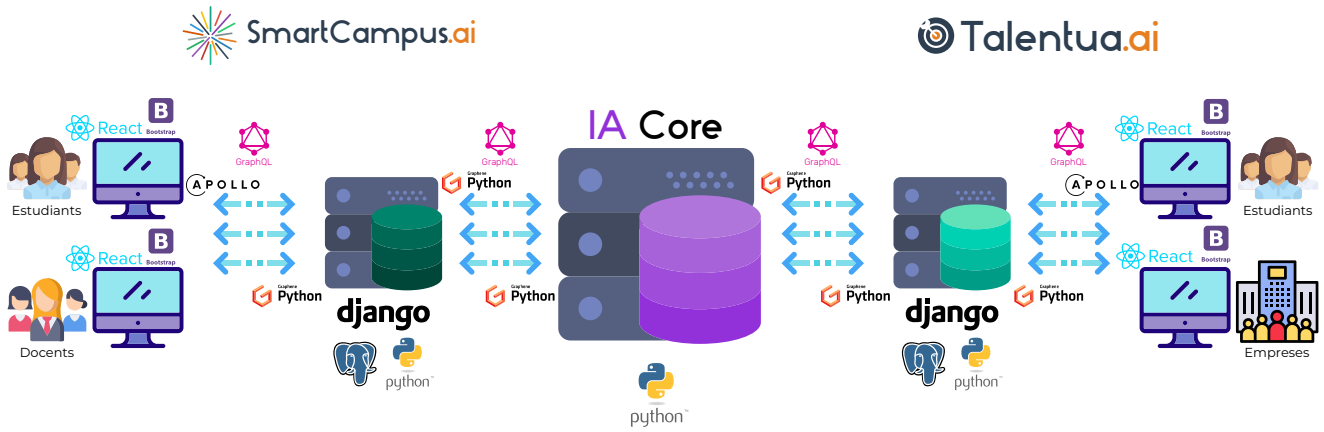

Fig. 2: Web Architecture

its own OS and its own virtual hardware (CPU, virtual network adapter, memory, etc.). Achieving this level of virtualization it is only possible thanks to the *Hypervisor*. This tool, placed between the host and guests machines, is the one in charge of the management and execution of VMs. Finally, as shown in Figure 3, a particular host can run multiple of these VMs (also known as *guests machines*).

On the other hand, containers are presented as an alternative to VMs. The first remarkable difference between VMs and containers is that containers share the kernel's host with other containers while each VM has its own kernel. As can be seen in Figure 4, containers only encapsulate the *userland*. Userland has its own processing space, its network interface, and even its own file system. The fact that containers do not package up the entire OS makes them use less memory than a traditional virtual machine and have a quicker startup process. All this translates into resource savings.

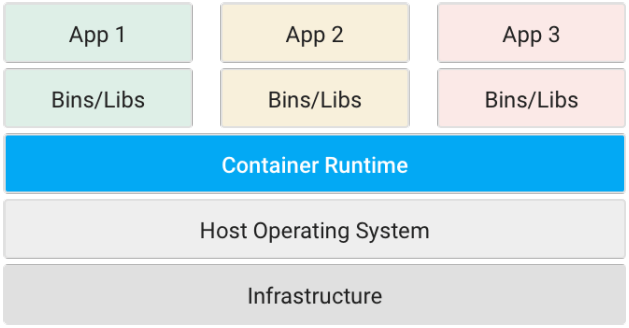

Fig. 4: Container Stack. Adapted from Google Documentation [4]

The process to create a container is the following: first, a template file is created with multiple instructions that define the environment needed; next, the *Container Runtime* generates an image from the template file; and finally, a container is created from the built image. Any container that uses this template will behave the same in every host executed. This compatibility is relevant to companies since every person in the team will use the same environment. From the developer perspective, it saves a lot of time on setting up the work environment and one is able to focus on the code. Another reason why companies have been using containers during the last years is because of their efficiency in Continuous Integration/Continuous Deployment (CI/CD). Being able to use the same image in integration than in production, makes the testing and debugging processes easier. A scenario where that is useful would be a bug in production. It is possible to pull the production environment to debug the issue, fix it, and then deploy the application with the bug-free version. Since the addition or subtraction of containers can be done pretty quickly for a specific environment, companies see scalability as another key aspect of adopting them over virtual machines.

4.2 The Cloud

Cloud computing have gained popularity during the last years. This approach consist on a cloud service provider taking the the responsibility of the administration a part of the infrastructure:

- **Infrastructure as a Service (IaaS)**, the cloud service provider administrates the low-level infrastructure. This system is suitable for development teams that prefer to manage the infrastructure.

- **Platform as a Service (PaaS)**, on this kind of service, the provider manages the security, OS, the server software, and the backups. The user only have to worry about developing, managing and distributing its applications.

- **Software as a Service (SaaS)**, providers offer the user software and application through subscription. With this approach, users do not have to manage nor install software since everything is managed by the provider.

On the other hand, there are companies that still prefer using a **traditional IT infrastructure**. This approach consists in a regular self-hosted data centre. With this approach, the team is responsible for every layer of the architecture. In other words, the team provide all of the infrastructure necessary to run their applications.
On figure 5 can be seen a visual representation of all the alternatives commented above.

Fig. 5: Stack architectures. From IBM Documentation [5]

The main differences between cloud hosting and traditional web hosting are:

- **Elasticity and resilience**: the data and applications hosted in the cloud are distributed evenly across the provider servers. The advantage over the traditional approach is that if one server fails, the downtime is avoided and there is no data lose. Moreover, the storage and computing power offer by cloud services is cheaper than having an in-house system.

- **Flexibility and Scalability**: the on-demand virtual space of cloud computing has unlimited resources. This makes the servers scale up or down depending on the necessities of the application.

- **Automation**: this is the biggest difference between both approaches. The Cloud service provider takes care of all the hardware, making the user focus on its business application. On the other hand, traditional infrastructure requires a dedicated team to maintain and monitor the data centre.

- **Security**: since the data storage and software delivery are placed on external servers, it is considered less secure than hosting the data on in-house systems. Thus, it is important to choose a cloud provider that is completely transparent in its hosting platforms.

4.3 DevOps

Tech developments must achieve some characteristics so companies can remain in the market. Two important characteristics should be considered: the first one is **speed** and it is the time that takes to make a new functionality for a product, from the time where the development begins until it is available to the end-user. The second characteristic is **stability**, that is the reliability that users have to be able to use a new functionality on an application without failures and its availability.

The problem that had the majority of development projects was that the team were not on the same page. While the goal of developers was speed, the goal for system administrators was stability. The problem of this approach is that changes may cause system errors.

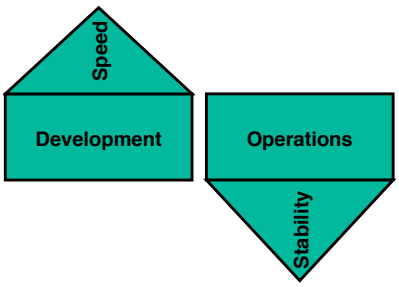

Fig. 6: Traditional project development environment issues

DevOps is a software engineering practice that aims at unifying software development (Dev) and software operations (Ops). The goals that have this practice are:

- To have shorter development cycles
- Increase the code deployment frequency
- Immediate recovery from failures

Summarising, DevOps can ensure the speed of implementing new features and making them available to end-users, and still maintaining product stability.

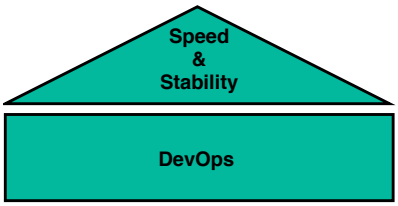

Fig. 7: DevOps project environment

4.4 CI Tools

Companies are making their moves toward DevOps practices and Agile culture to accelerate the delivery speed and ensure product quality. In DevOps, a continuous and automated delivery cycle is the backbone that makes fast and reliable delivery possible.
Continuous integration (CI) is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error. This method has a defined work-flow: Developers write code and commit changes to the shared repository; after that, the CI server monitors the repository and evaluates all the changes; CI builds the system and conduct integration and unit tests; the server releases deployable artefacts; the continuous integration server assigns a build tag to the version and building code; then the CI server reports the team about the successful build. If the tests fail, the server alerts about the event to the development team, and the team will fix the issues as quickly as is possible.
One of the most popular CI tools available in the market is **Travis CI**, which is mainly used for build and test projects. This CI tool can easily be integrated with the most common cloud repositories like Github and Bitbucket. Moreover, this tool does not need dedicated servers to run since is hosted in the cloud. Some of the main features that have Travis CI are: automatic integration with Github; repository access to build pull requests; pre-installed builds

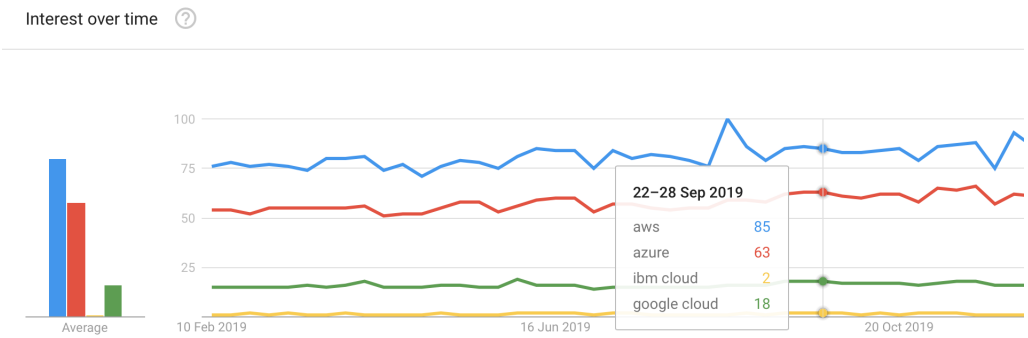

Fig. 8: Comparison of the interest between cloud-service providers from Feb. 2019 to Feb 2020. From Google Trends.

and test tools; deployment to multiple cloud services. Finally, this tool is free for open-source projects, otherwise, an enterprise plan must be purchased.

Another popular CI tool is **Jenkins**. This open-source automation tool allows monitoring executions of deployment cycles. The fact that Jenkins is Java-based, means that can run on any OS where Java runs. It has a simple and user-friendly interface; its easy to install, upgrade, and configure; is extensible with a huge community that contribute on plugin resources; compatible with Docker, Kubernetes, and many other technologies; supports distributed builds; and also, supports notification on the build status.

5 Methodology

Since the project kickoff, agile SCRUM methodology has been followed, with meetings every 15 days. On those meetings, the team and the tutors (acting as product owner) checked the progress made between each sprint. Furthermore, the tutors gave feedback and mentoring on The use of that methodology was agreed by both the development team and the tutors, since it fits well on the kind of project that was being developed.

Besides the sprint meetings, the three members agreed to made daily reunions where only the team take part in. The purpose of these reunions was for checking the progress made and share useful resources that might help to complete a specific task or solve an issue found in development. The meetings were done via WhatsApp or Skype.

6 Project Development

It was agreed with the tutor, to divide the project on a set of tasks:

- Analyse and choose the Cloud provider that will host the platform.
- Provide an infrastructure that guarantee security due to information sensitivity.
- Analyse which DevOps tools are available in the market to ease the platform development on parts I and II.
- Build a stack using DevOps tools for automating test and deployment process in different environments.
- Document the infrastructure and DevOps stack.

7 Results

7.1 Choosing the *Cloud* service provider

The number of options to compare was reduced to the first and second market leaders in public cloud-service providers, Amazon Web Services and Microsoft Azure. The reason behind that decision was because both cloud providers can be accessed using the UAB institutional mail, allowing to use their services for free for learning purpose. The fact that the services offered by the top cloud-service providers have become very similar in the last few years, makes them easy to compare.

Both cloud-services offer largely similar basic capabilities around flexible compute, storage, networking, and ML support. Also, AWS and Azure share the common elements of a public cloud:

- Self-service and instant provisioning
- Autoscaling
- Plus security
- Compliance
- Identity management features

The key difference between the analysed cloud-providers comes with the breadth and depth of the services offered by AWS. The number of options provided by the market leader makes it win on developer functionality.

Even though the idea of this sub-section is to compare the technological capabilities of both cloud-services, a cost comparison has been made since Talentua aspires to become a startup. The analysis done by Kim Weins [6] points out that Azure has low costs price in almost every scenario while AWS is a middle-priced option.

TABLE 1: CLOUD DISCOUNTING OPTIONS. ADAPTED FROM [6]

	AWS	Azure
Type of discount	RI	RI
Length of commitment	1 or 3 years	1 or 3 years
Range of discounts levels	Up to 75%	Up to 72%

As shown in the above table, AWS and Azure offer discounts in exchange for committing using their cloud-provider for one or three year period.

Another interesting indicator to compare, is the cloud community engagement that each service provider have. The comparison showed in the following figure (figure 8) shows a search comparison between AWS, Azure, Google Cloud, and IBM Cloud during one year period. As can be seen, the most searched cloud-provider is AWS, followed by the Microsoft cloud service. The metrics shown by this figure are not absolute, but constantly being the more searches could give the idea that AWS community is bigger than the other top cloud services communities.

The possibility to create tailored instances, their community, and because of their experience providing IaaS since 2008, were the reasons to choose Amazon Web Service over Microsoft Azure.

7.2 Planning the infrastructure

Before implementing the *DevOps* stack in AWS, it seemed necessary to plan the platform's infrastructure. The main goals to achieve in this milestone were security, and scalability.

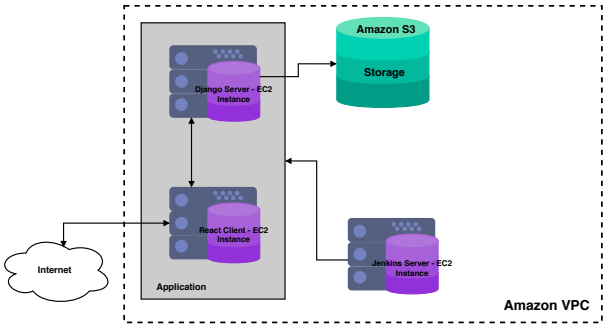

Fig. 9: Talentua infrastructure

The first model approach of this infrastructure can be seen in the above figure (figure 9). As can be observed, this model has four main components:

- **Storage Server**: on this instance, is where all the data is stored.
- **Jenkins Server**: this is the instance in charge of doing the CI/CD process. In the following subsection, its behaviour will be described.
- **React Client**: frontend instance that provides the views to be rendered in the browser.
- **Django Server**: backend instance that acts as a middleware between the database server storage and the frontend instance.

To make the platform scalable, the frontend and the backend have been split in different Amazon EC2 instances. The benefit of using this approach is that when some of the components described needs to auto-scale, e.g due to high CPU load, only the affected instance will be scaled without touching anything in the rest of the components.

Since the processed data is sensitive information, it is prime to limit the access from where this data can be accessed. This is another benefit of this architecture. To deploy a secure platform, it was created an Amazon Virtual Private Cloud (Amazon VPC). This service makes possible to logically isolate a section of the AWS Cloud by launching resources in a private network fully customizable. Thanks to this service, the only instance that has access to the data is the Django Server, since the backend and the databases are in a private-facing subnet, while the frontend instances are in the private and public subnets.

A page request from a user will do the following: first, the user request arrives through the public interface of the React EC2 instance; the React application then communicates via the private subnet with the Django EC2 instance; if the request made by the client needs some data stored the Django instance will fetch the data; after that Django instance will return the requested resources; and finally the React instance will respond to the user client.

7.3 Implementing the DevOps pipeline

Instead of using the tools that AWS provides, the team opted for using open-source tools. This way Talentua is not tied with any technologies offered by companies, which would smooth the transition to another cloud provider service. For this reason, and also because of the high community that has, the team decided to implement Jenkins as the CI/CD automation tool.

The EC2 instance observed in the previous subsection (figure 9), is the one in charge of detecting changes from the repositories; downloading and testing the modified code; notifying the administrators the build status; and deploying the new application build if all tests were successful. To successfully build the pipeline described, it was necessary to install some plugins, e.g a mail plugin to notify the team the job status.

After having all the necessary resources, the pipeline seen in figure 10 was build by using a *Jenkinsfile*. The pipeline is composed of four total stages:

- **Checkout (Stage 1):** on the first stage, is where the code is fetched from the Github repository. To complete this task it was necessary to install on the Jenkins server a *git webhook* plugin, and after that configure the Github repositories to allow the CI tool detect changes.
  When there are changes on the repositories, a new job

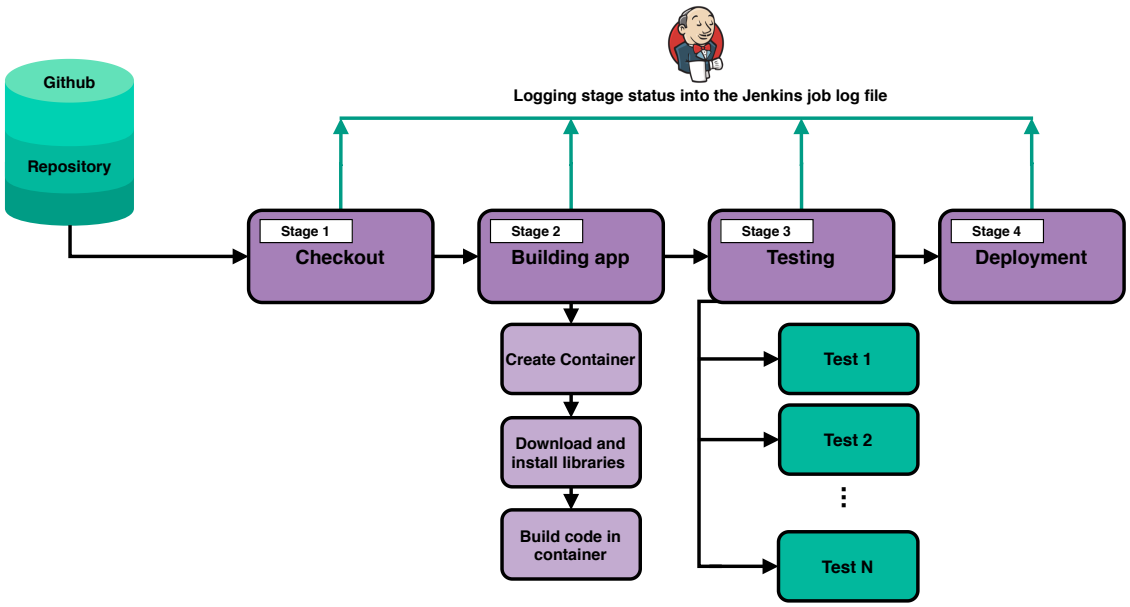

Fig. 10: CI/CD pipeline using Jenkins

starts and a copy of the source code is saved on the Jenkins server.

- **Building the app (Stage 2):** the Jenkins server open a communication via SSH with the affected instance (the frontend or the backend instance) and copy the source code. After that, thanks to a *Dockerfile*, a new Docker instance will be created. On this instance is where the application with the new features will be tested and deployed. Then, the Docker instance, download and install the necessary libraries to compile the code. Finally, the application is build inside the Docker container.

- **Testing the build (Stage 3):** After the code is built, inside the Docker instance a set of tests are checked. This tests are done in parallel. In the case all the tests are successful, the pipeline will continue to the last Jenkins stage. Otherwise, the job would fail, causing that Jenkins will notify the interested by mail.

- **Deploying the build (Stage 4):** The last stage, overwrites the last Docker built for the new one and set the new instance to production. Finishing all this stages, would mean that the Jenkins job was successful and the new features are already implemented in production. Finally, a notification is sent to the interested.

7.4 Merging the cloud platform with Parts I and II

On the first iterations of the pipeline implementation, the source code used were mainly snippet codes such as 'Hello World'. This was done with the purpose of testing that Jenkins job were failing because a bad system configuration and not from development failures.

Once the infrastructure and the pipeline were fully functional, the source code developed by Alan and Xavi was uploaded to the AWS servers.

8 Conclusions

Much of the work done in recent months has been based on studying the current state of art and then choosing the proper tools to use. The decision proposed for the different technologies, and to solve problems in parts I, II and III have been made by the three members of the team which we believe made a huge difference when compared with a project where the decisions are made individually.

The chosen methodology was a complete success, since it helped the three parts of the project to advance thanks to the feedback and mentoring provided by both professors.

The objective proposed for this part was to build a cloud based pipeline using DevOps tools which would allow new functionalities to be implemented on the platform as quickly and conveniently as possible. So, to what concerns to this part, the main objective has been achieved, since the team can push new changes to the repository without worrying to do build the new features on the servers nor doing the sysadmin job. Even though the main objective was reached, there are still unfinished tasks to be completed or even started.

9 Future work

Related with this part, there is still work to do. The fact of already having set the infrastructure and a DevOps pipeline, makes it really easy to extend this base. Some to-do tasks, would be:

- Change the way Jenkins build and deploy the code to the application. Since the pipeline it is already working, this is not a priority task but maybe would be interesting on pushing the new commits inside a centralised instance.

- Document the infrastructure

- Integration with Kubernetes

- Test an external open-source monitoring tool

Beyond the work to finish on this part, there is still the development of the parts IV, V and VI. These parts will be focused on implementing machine learning algorithms in the backend.

It is estimated that after finishing these parts, an MVP of the platform will be achieved.

ACKNOWLEDGEMENTS

To begin with, I would like to thank my partners, Alan Fusté and Xavi Velasco, for giving me the opportunity of working with them in this project. Furthermore, I would also like to thank them for their support during the development of this first phase.
Also, I would like to express my gratitude to Daniel Franco and Rubén Rubio for their time and mentoring during this lasts months. Finally, I would like to thank my fathers and brothers for their constant support to every decision I made.

REFERENCES

[1] Brandt, J. “Transforming Education with Talent Management.” School Business Affairs 77.1 [2011]: 30-31

[2] The Local. “Spain’s brain drain ‘worst in Western Europe’ ”. Internet: https://www.thelocal.es/20140901/spanish-professionals-leave-spain-brain-drain [Feb 7th, 2020]

[3] Alcidi, C. and Gros D. “EU Mobile Workers: A challenge to public finances?.” CEPS Special Report [2019]

[4] Google. “Containers at Google”. Internet: https://cloud.google.com/containers [Feb 1st, 2020].

[5] IBM Cloud Education. “IaaS (Infrastructure-as-a-Service)”. Internet: https://www.ibm.com/cloud/learn/iaas [Feb 3rd, 2020].

[6] Weins, K. “Comparing Cloud Instance Pricing: AWS vs Azure vs Google vs IBM”. Internet: https://www.flexera.com/blog/cloud/2017/11/comparing-cloud-instance-pricing-aws-vs-azure-vs-google-vs-ibm/ [Feb 7th, 2020]