INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Framework de Segurança para Kubernetes

DIOGO DA COSTA BARBOSA
Setembro de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Kubernetes Security Framework

Diogo Barbosa

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Informatic Engineering

Advisor: Prof. Luís Nogueira

Porto, September 26, 2025

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, September 26, 2025

Dedicatory

I want to dedicate this thesis to my parents, who always pushed me through my objectives, and never let me give up on them. Thank you professor Luís Nogueira, for always being there for me, even when I didn't demonstrate the desire level of interest on the project, and thanks ISEP for all the years I've been studying here and all the things I've learned here.

Abstract

In this document, we can follow the entire process of developing the Master's Thesis written for the completion of the Master's Degree in Computer Engineering at the Higher Institute of Engineering of Porto. Considering that the Master's degree specializes in Cybersecurity and Systems Administration and that one of the most significant emerging technologies in recent years is Kubernetes, which is closely linked to the current systems administration scenario, it made perfect sense to develop something in the area of Kubernetes security.

In recent years, more and more companies have adopted other styles of system development, as conventional monolithic systems have begun to show some limitations. Thus, new architectures for computer systems have emerged, one of the most prominent in the market in recent years being Microservices. This architecture aims to divide systems into small services, whose operation does not depend on any other service. However, adopting this has been quite complicated, until solutions appeared that could work according to the principles of Microservices.

The solution for better implementing Microservices lies in containers. These allow for a higher level of abstraction of the system where they run, managing to simulate different types of systems on the same machine. Containers complement Microservices, as each of the independent services developed will be executed in different containers. This achieves another level of independence, as technological dependencies practically cease to exist, since each container has a different execution.

However, when large companies began their journey to migrate to Microservices, they noticed that the larger the system used, the greater the number of containers required, and since each container has a different execution and even different containers running the same service (different instances), managing these containers was becoming quite difficult. As a result, container management tools began to be developed, with some companies managing to develop their own internal solutions. The best example is Google, which currently has two internal container management tools, which it used as a basis to create the largest open-source container management tool.

Kubernetes was launched by Google in 2014 and immediately received strong support from the entire community, which readily contributed its knowledge to improvements that would come over the next eleven years. Despite the keen interest of a large community and many companies, it is still considered a recent technology. Kubernetes brought a new vision of containers, since in reality it is not containers that are executed, but Pods, which can be considered improved containers.

The basic operation of Kubernetes groups several Nodes, which can be virtual machines, physical machines, or even an instance in the cloud. One of the Nodes will be the main one where the Control Plane will be hosted, which is considered the brain of a Kubernetes cluster. This consists of several components with different tasks, including all communication between Pods and even with the outside world, which passes through the API Server.

Another component monitors all running Pods and compares them with the desired state for the cluster, which in turn is stored in another component of the Control Plane.

Although it seems like an excellent solution for migrating to microservices, companies are beginning to fear some security flaws that may exist. But even more worrying for these same companies is knowing whether they are maintaining the same level of security maturity in their system. This problem is the main focus of this work, since the intended result is to provide a security checklist to be applied to clusters in order to determine the cluster's security maturity level and how it can be improved.

Throughout the document, an in-depth analysis of Kubernetes will be carried out in order to understand the critical points of a Kubernetes cluster that must be analyzed in depth to ensure its total security. With this analysis completed, a checklist will be drawn up, which will provide information on how to perform the verification and, if necessary, an explanation of how to improve these points.

Keywords: Kubernetes, Security, Framework, Assessment

Resumo

Neste documento podemos acompanhar todo o processo de desenvolvimento da Tese de Mestrado desenvolvida em prol do término do Mestrado em Engenharia Informática do Instituto Superior de Engenharia do Porto. Tendo em conta a especialização do Mestrado ser Cibersegurança e Administração de Sistemas e uma das maiores tecnologias emergentes dos últimos anos ser o Kubernetes que está extremamente conectado ao atual cenário de administração de sistemas, fez todo o sentido desenvolver algo na área de segurança em Kubernetes.

Nos últimos anos, cada vez mais empresas têm adotado outros estilos de desenvolvimento de sistemas, já que os convencionais sistemas monolíticos começaram a apresentar algumas limitações. Deste modo, novas arquiteturas para sistemas informáticos, sendo que uma das que mais destaque ganhou no mercado, nos últimos anos foi a de Microserviços. Uma arquitetura que pretende dividir os sistemas em pequenos serviços, em que o seu funcionamento não tenha qualquer dependência de outro qualquer serviço. Porém, adotar isto teve bastantes complicações, até aparecerem as soluções que conseguissem funcionar, de acordo com as bases dos Microserviços.

A solução para melhor implementar Microserviços reside em contentores. Estes permitem um maior nível de abstração do sistema onde correm, conseguindo simular diferentes tipos de sistemas na mesma máquina. Ora, os contentores complementam-se com os Microserviços já que cada um dos serviços independentes desenvolvidos irão ser executados em contentores diferentes. Deste modo, outro nível de independência é atingido, já que praticamente deixam de existir dependências tecnológicas, já que cada um dos contentores tem uma execução diferente.

Porém, as grandes empresas quando começaram a sua jornada de migração para Microserviços notaram que quanto maior era o sistema usado, maior o número de contentores necessários, e sendo cada container uma execução diferente e até diferentes contentores a correr o mesmo serviço (instancias diferentes), a gestão destes mesmos contentores começava a tornar-se bastante difícil. Assim sendo, ferramentas de gestão de contentores começaram a ser pensadas, sendo que algumas empresas conseguiram desenvolver as próprias soluções internas. O maior exemplo será mesmo a Google que tem até hoje duas ferramentas de gestão de contentores, internas, nas quais se baseou para criar a maior ferramenta de gestão de contentores open-source.

O Kubernetes foi lançado pela Google em 2014, e desde logo teve bastante apoio de toda a comunidade, que de pronto deu bastante conhecimento para melhoramentos que viriam nestes onze anos. Apesar do grande interesse de uma grande comunidade e de muitas empresas, ainda é considerado uma tecnologia recente. O Kubernetes veio trazer uma nova visão de containers já que na realidade não são executados containers, mas sim Pods, que podem ser considerados containers com melhoramentos.

O funcionamento básico do Kubernetes agrupa vários Nodes, que podem ser máquinas virtuais, físicas ou até uma instância na cloud. Um dos Nodes será o principal onde estará

hospedado o Control Plane, que é considerado o cérebro de um cluster de Kubernetes. Este é composto por vários componentes com diferentes tarefas, entre as quais toda comunicação entre Pods e até mesmo com o exterior passa pelo API Server. Outro dos componentes pretende monitorizar todos os Pods que estão em execução, e comparar com o estado que é pretendido para o cluster, que por sua vez está guardado noutro componente do Control Plane.

Apesar de parecer todo ele uma excelente solução para a migração para Microserviços, as empresas começam a ter receio de algumas falhas de segurança que possam existir. Mas ainda mais preocupante, para estas mesmas empresas, é saber se estão a manter o mesmo nível de maturidade de segurança do seu sistema. Este problema é o grande mote para este trabalho, já que o resultado pretendido é fornecer uma checklist de segurança para ser aplicada aos clusters de forma a saber o nível de maturidade de segurança do cluster e como este pode ser melhorado.

Ao longo do documento irá ser feita uma profunda análise ao Kubernetes de forma a perceber quais os pontos críticos de um cluster Kubernetes que devem ser analisados em profundidade de forma a garantir a total segurança do mesmo. Com essa análise realizada, a elaboração de uma checklist será feita, e que irá fornecer informação de como realizar a verificação, e, se necessário, a explicação de como melhorar esses mesmos pontos.

Contents

List of Figures

Chapter 1

Introduction

1.1 Contextualization

The industry worldwide has been constantly changing throughout human history, either by adopting new ways of work, using new tools, or thinking in another way. 100 years ago to send a message, we would ask someone to deliver that message written on paper to a destiny, 40 years ago we would send a fax typed on a mechanical keyboard and now we simply need to type a message directly on the screen and hit the send button, to deliver an e-mail to someone around the globe.

With a clear digitalization of the world happening every day, companies are relying on the digital world more than ever, which means that the digital infrastructure of a company has become one of its most important and valuable assets. However, with new technologies emerging every day, a company needs to quickly adapt so it does not fall behind the competition. This necessity has pressured Information Technologies (IT) companies to deliver new features as fast as possible, and for that, it was necessary to adopt new cultures and ways of working.

DevOps is a concept from 2008 [PSP17], which quickly became a culture of software development, and since then, companies have been adopting DevOps more and more in their projects. Gathering the Development and Operations teams to work together and constantly deliver many small pieces, are two of the main objectives of DevOps. Delivering small pieces at the earlier stage of DevOps, was not an easy task since the majority of the companies were residing their digital infrastructure on monolith solutions. So the first step to achieving a good DevOps implementation is to divide monolith projects into small pieces with independent responsibilities from each other, and this process leads to Microservices architecture.

While adopting Microservices companies started to note that each service could be developed and managed by a smaller team, specialized in the technology used by that service, and capable of migrating it to a new one with no effort. Companies also noted that independent services could also be, platforms independent from each other, with no need to use the same technology in every service. Different technologies have different needs, which means that the same machine might not be able to support the different services developed. To handle this problem, a way to solve it is to use the same hardware but have a different runtime, and that solution is called containers. By turning each service into a container, which is packaging the developed service and their dependencies together, each one will be deployed and monitored by itself. One of the main container technologies in the market is Docker, launched in 2013 as an open-source Platform as a Service (PaaS).

With the systems getting more and more complex every day, to achieve all the demands of a company, more small components are introduced to systems, and with a constant growth of users of the system more containers will be necessary either for new features or more instances of existing services. For big companies that reside in a big infrastructure with many small parts, managing such a large number of containers, can easily become an impossible mission, and so companies started to use container orchestration tools, to guarantee better availability and consistency of the system. Over the years some tools have arrived on the market, with Kubernetes being one of the most used container orchestration platform in the market, at the moment. The function of a container orchestrator can be defined as a group of containers that ease the management of that group and make it work as expected. This group of containers is named cluster.

1.2 Problem

With the industry changing so fast and the demand to deliver new things as quickly as possible, the migration to container technology has become one of the most clear paths to follow. With complex systems that reside on monolith architectures, migrating an enormous number of services to different containers, and managing them becomes a pretty difficult task. To help manage containers adopting a container orchestration tool has become a must for every container project.

The objective of migrating to containers is to achieve new features faster to keep up with the market, and for that, it is wise to use the best technology in the market, that's why brands such as IBM, Adidas, Spotify and many others are adopting Kubernetes to manage their clusters and the other companies are following the same path.

Even though containers are looking to be the best solution for the near future in software development, companies are still reticent in adopting it, because being such a new technology can bring many zero-day attacks, as long as, new ways of attacking a digital infrastructure. By migrating to this technology, companies want to guarantee that the same maturity level of security is maintained. But what defines security in Kubernetes? Do the companies know their clusters are meeting their security expectations? Even if they are, for how long can it be true?

1.3 Objectives

After gathering the necessary level of knowledge about Kubernetes and understanding how it should correctly work, an analysis to every aspect of it is necessary to understand the different security needs of each part. In addition to that, the actual scenario of Kubernetes vulnerabilities is also explored.

With all the information and analysis of Kubernetes done, a checklist will be developed, with the objective of guide developers, engineers, and operators on the evaluation of the maturity level of security of clusters. Based on the answers provided by the user, it is expected to get some suggestions of actions to improve the security level of their clusters in the most affected areas. This actions will always be related to the cluster itself and never about the applications running on the clusters.

After developing the checklist, an evaluation of the solution is necessary. The best way to get results from the usage of the solution is to apply it on a real-world environment, and if

possible on a situation of migration to microservices and Kubernetes. There is a company that offered his infrastructure to test the solution developed in this project. The company is related to the telecommunication area, but one of the sub-objectives is not to turn the checklist focused only on one market, but as general as possible.

With the conclusion of the project it is expected to have a solution capable of entering any type of Kubernetes project, and help the companies to reach the desired maturity level of security in a much faster way. This is achieved by giving the correct information on how to verify each point of the checklist to be developed and what to do after getting the result of the evaluation and improve the security maturity level.

1.4 Methodology

For this project the chosen methodology to better approach this problem is the Case Study methodology. This methodology is the correct one to follow, since one of the main objectives of this research is to prove the gains of the implementation of a developed security checklist into a Kubernetes cluster. The methodology is the correct one since there is the possibility to take on a Kubernetes cluster, already in production, take the actual security scenario of the cluster in consideration. After acquiring the actual scenario, the suggestions from the checklist should be followed in order to understand, how easy it is to take a security evaluation over the cluster, and how improved the security maturity level can be.

1.5 Structure

Along the read of this report you will notice that it will be divided into 4 major parts. The first part is the Introduction where you will find the propose of this project and the objectives of it, which methodology was used to perform the research and the activities plan for the project.

The second part of this report is the state of the art where a compilation of theoretical information about Kubernetes, where a deep dive is made into the Kubernetes technology and how it is supposed to work. It will be divided in the different parts of Kubernetes, but it also will have some information about containers and Microservices, in order to understand one of the main reasons why the companies are adopting Kubernetes.

The third part of the report will be the analysis of Kubernetes in order to understand the points that compose the whole security scene of Kubernetes, but beside that, it also will have the analysis of how the security points are checked or where are they defined in the start of a Kubernetes cluster.

The last part of the report is the conclusions and analysis of results where after the usage of the final result of these research, the checklist, it is decided if it is really useful or not and what advantages it brings to the comapnies in using it.

1.6 Planning

To approach this research a good planning is extremely important, in order to perform the necessary tasks in the correct order and determine what is considered the conclusion of the task. But planning a project is way more than just determine tasks and predict how long do they take to be completed.

⊟Planning	10 dias	**16-12-2024 8:00**	**27-12-2024 17:00**
Project Charter	5 dias	16-12-2024 8:00	20-12-2024 17:00
WBS	5 dias	19-12-2024 8:00	25-12-2024 17:00
Plan	7 dias	19-12-2024 8:00	27-12-2024 17:00
Form Plan	2 dias	26-12-2024 8:00	27-12-2024 17:00
⊟State of Art	45 dias	**28-12-2024 8:00**	**28-02-2025 17:00**
Microservices	2 dias	28-12-2024 8:00	31-12-2024 17:00
Containers	3 dias	31-12-2024 8:00	02-01-2025 17:00
Kubernetes	42 dias	02-01-2025 8:00	28-02-2025 17:00
Form Publication	43 dias	01-01-2025 8:00	28-02-2025 17:00
⊟Analysis	21 dias	**01-03-2025 8:00**	**31-03-2025 17:00**
Kubernetes Security	21 dias	01-03-2025 8:00	31-03-2025 17:00
How to verify	5 dias	10-03-2025 8:00	14-03-2025 17:00
How to improve	5 dias	17-03-2025 8:00	21-03-2025 17:00
Form results	5 dias	24-03-2025 8:00	28-03-2025 17:00
⊟Development	22 dias	**01-04-2025 7:00**	**30-04-2025 17:00**
Maturity Level	13 dias	01-04-2025 7:00	17-04-2025 17:00
Parameters	13 dias	01-04-2025 7:00	17-04-2025 17:00
How to	10 dias	17-04-2025 7:00	30-04-2025 17:00
Helper	10 dias	17-04-2025 7:00	30-04-2025 17:00
⊟Implementation	23 dias	**01-05-2025 7:00**	**02-06-2025 17:00**
Production use	22 dias	01-05-2025 7:00	30-05-2025 17:00
Interviews	4 dias	28-05-2025 7:00	02-06-2025 17:00
⊟Conclusions	15 dias	**02-06-2025 7:00**	**20-06-2025 17:00**
Improvements analysis	15 dias	02-06-2025 7:00	20-06-2025 17:00
Users opinion	4 dias	09-06-2025 7:00	12-06-2025 17:00
Final conclusions	5 dias	16-06-2025 7:00	20-06-2025 17:00

Figure 1.1: Activities planning

To start the planning a good clarification of the problem and the definition of the objectives to it, must be done. Beside the objectives the result of the planning should be understood to, which in the case is a security checklist for Kubernetes cluster. As stated before this project want to provide a solution as general as possible and with that a stakeholder definition doesn't make much sense. But since the solution will be tested on telecommunication company environment, it can be a more directed to the telecommunication company.

In figure 1.1 it is possible to see the activities plan for this research. And being a project of more investigation, with almost none code development, but with more analysis to a solution that exists, the type of tasks defined are more theoretical. In all those tasks there are a few that to be performed will take more time, such as the State of the Art of Kubernetes because being such an enormous tool, it takes a long time to discover all the small details abou all the components of Kubernetes. The security analysis to Kubernetes is another important task because it will determine what parameters will the final checklist have. Other long tasks to be performed is the implementation because it will require people from the telecomunication company to use the developed checklist, retrieve data about how improved the security maturity level was, after using it. More data will be retrieved, by doing interviews to the people who used it.

In Appendix A there is the Gant Chart (Figure A.1 and A.2) associated to the table shown in figure 1.1. There a more visual perception of the plan of activities with the respective predict of dates to be performed. Each line of the Gant Chart is aligned with the lines of the table presented.

1.7 Motivation

The project idea came from start of my usage of Kubernetes and understand how powerful this technology can be. But there were some concerns on the safeness of it, and for that reason, why don't start a security baseline for kubernetes through a framework. Something that can be improved along the time, and can be used by anyone. But just because one person thinks it will be worthy, it doesn't mean it is really necessary. For that reason a survey

was shared with some people that work daily with Kubernetes. This survey showed that a majority of users doesn't have a security checklist for their clusters, and some even say that a new one will be good. To have a wider range of users and different contexts, persons from Development, Operations, Engineering and Management. In the image appendix it is possible to find the graphics that show the answers to the questions of the survey.

Chapter 2

State of Art

2.1 Microservices

By November 2024, according to the TIOBE Index, [Lin+22] Python, C++ and Java are the most popular programming language. Nevertheless, these languages were developed to create applications as one big artifact, also known as, monolith applications. This means that every component of the application is relying on the same infrastructure and runtime. By adopting this software architecture, when an update is necessary, the whole application will be restarted, even that just a minor bug is corrected.

The adoption of this software architecture caused some problems to companies. Down time while performing updates to their systems, was one of the most important since it impacted a lot on the user experience of the product. The development of an application could be forced to use a certain type of technology, due to dependencies caused by other parts of the system. With these and other problems in sight, developers started working on a possible solution.

In 2011, the term "microservices" was first introduced at an architectural workshop and since then its adoption has been growing.[Dra+17] A possible definition to Microservices is divided in to 2 parts: "

- - "Microservice: A microservice is a cohesive, independent process interacting via messages.
- - Microservices Architecture:"a distributed application where all its modules are microservices"" [Dra+17]

In the last thirteen years, companies and their developers started to adopt this architecture for their systems, even though some companies have developed approaches close to microservices. So, corporations started to divide each module of their system into an independent service that will take actions by receiving instructions via messages, from the other modules of the system.

By reaching this level of abstraction between modules of the application, developers noticed that they can use different technologies for each of the modules since they are not dependent on anything. Nevertheless, not all the technologies have the same needs regarding the hardware or the ambient they are running on, which means that there were still dependencies to resolve in order to turn the microservices approach the most flexible as possible. The solution to this was, somehow, to abstract the runtime of the necessary system of the hardware in which it is running.

2.1.1 Containers

The solution to abstract the software from the hardware reside on containers. They can be defined as " technologies that allow the packaging and isolation of applications with their entire runtime environment " [Red]. In a more practical way, this means that an application that is developed to run on a Linux environment, can be run on a Windows computer by using a the physical part to run a virtualization of the Linux and its application.

With companies starting to adopt or migrating to microservices their big systems started to decompose into various services. Having a service turned into a container means that, as many services as you have, the more containers will be part of the whole system. Many containers are not easy to manage, so the need for container management tools was starting to exist in the IT world.

2.1.2 Containers Management

To simplify the management of a big group of containers, containers-management systems started to be developed. Google started with their own solutions. Borg was the first system capable of managing long-running services and batch jobs. (Referencia a Borg, Omega e k8s) It was a simple solution at the beginning but with the time passing by it turned a lot more complex, but it were the developers who came with new solutions for it

2.2 Kubernetes

Kubernetes is the most recent container orchestration developed by Google. Launched in 2013 [Ste], partly caused by the high interest of the Linux community about containers [Bur+16]. The open-source system has been growing in the market and with the help of an enthusiastic community, many improvements have been made all the time. Across this subchapter, Kubernetes will be broke down to the smaller part that composes it, in order to later do an analysis regarding the most security points as possible.

At an high-level Kubernetes is a container orchestrator that basically covers them inside it's own structure. A Kubernetes cluster is a group of nodes that can be physical or virtual machines, or just a simple Raspberry Pi or many other options. There are two types of nodes:

- Control Plane
- Worker nodes

The control plane is responsible for managing the worker nodes by ensuring that they are running as expected, through a saved desired state, and for exposing an API to all the worker nodes. The worker nodes are supposed to run the different containers that compose the application. In the most cases a pod only runs one container, but it is possible to run different containers in a single pod. In figure 2.1 it's possible to see an high-level diagram of a Kubernetes cluster.

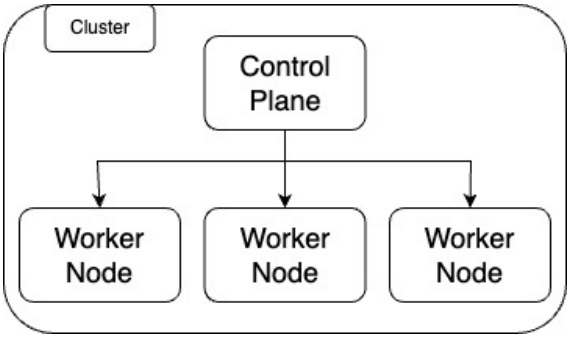

Figure 2.1: Kubernetes High Level diagram

2.2.1 Kubernetes Architecture

A cluster in Kubernetes has the main component, the Control Plane. It is normally wise to have multiple replicas, in order to guarantee a higher level of availability. As mentioned before the Control Plane has as main task, the management of the worker nodes, which in Kubernetes is called Pods.

Pods are the smallest unit in the Kubernetes universe, and their function is to have a container running inside and provision it with more capabilities to work perfectly in a Kubernetes cluster. This is one of the motivation that development teams are taking to have a more Kubernetes oriented development. In this way, the developed microservices are already prepared for taking Kubernetes and using it with better performance.

Generally, each pod is deployed under a specific controller, and its type is defined by, taking into consideration, the utility of the Pod. It is the controller that defines some characteristics of the pod. On the figure 2.2 the diagram is a more depth version of the high-level and more Kubernetes oriented.

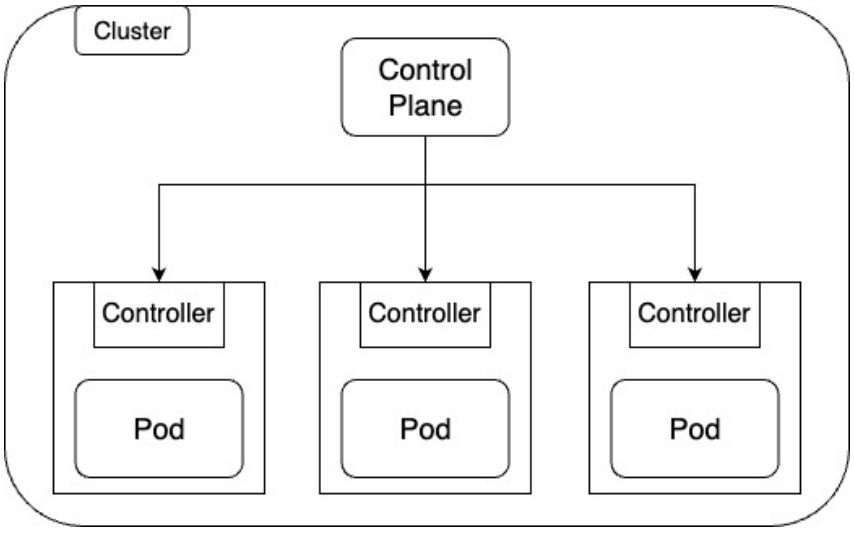

Figure 2.2: High Level diagram with K8's naming

2.2.2 Kubernetes Control Plane

The Control Plane is considered by many people the brain of Kubernetes cluster. In fact, this analogy is quite correct since without it a cluster might have the pods running, but they will clearly not work as expected, if they work at all. On the simplest implementation, it is normal to see only one instance of the control plane in the cluster, but the more complex the structure is, the more replicas should be deployed. Replicas in Kubernetes are instances of pods with the same container running inside.

The control plane has many functions inside a cluster. One of the most important, is the exposure of the API that allows the communication between pods and the communication between the control plane and the pods. Another important functionality is the storage of the desired state of the cluster in the cluster store. The controllers manager is responsible for managing the controllers that are being used in the cluster. To assign tasks to pods, there is the scheduler, which does much more than that in the background through a complex algorithm. In addition to all of this, there is also a cloud controller manager that eases the integration of the cluster into a cloud provider.

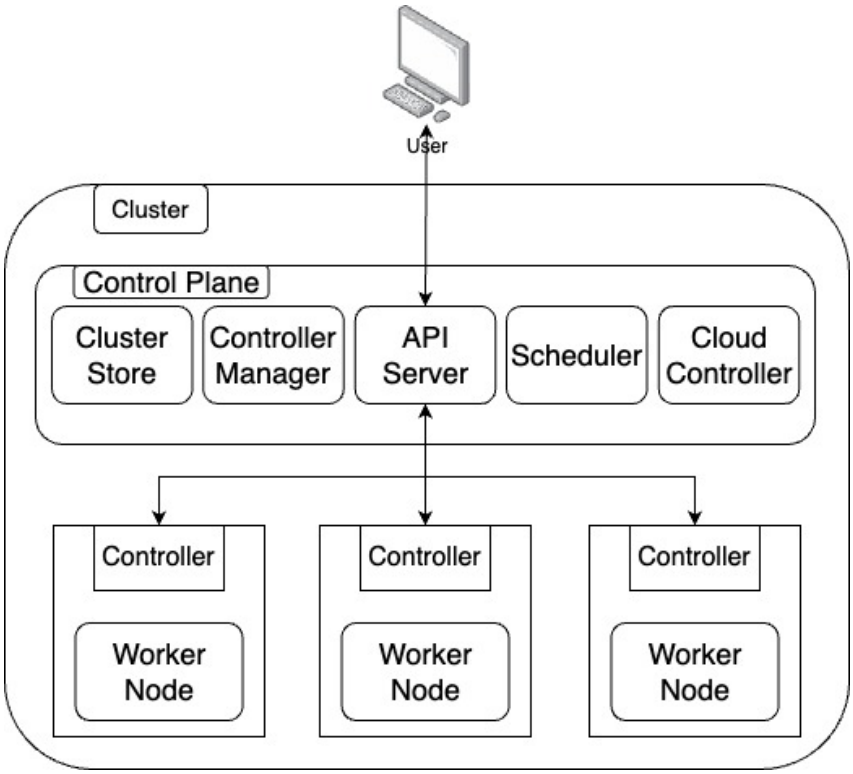

Figure 2.3: Control Plane diagram

API Server

The API Server is considered by many people, as the main part of a Kubernetes cluster. All the internal communication is made through the API and only it can open communication to the outside. All of these is achieved through a RESTfull interface that uses HTTPS, which provides more security to the cluster. The server normally exposes port 443 or 6443, but

it can be configured manually. Beside providing communication channels, it is also through the API Server, that all the changes are applied to the cluster such as creation or deletion of a Pod. All API calls are authorized.

To work properly a serialization is made, which is the process of transforming an object into a string of characters, while desirilization is the process of transforming the string into an object. This is also known as serialized state, which is saved in the cluster store. Kubernetes supports 2 types of serialization schema:

- JSON
- Protobuf

The JSON schemas are, in general, used for the communication with the users trough HTTP requests, while the Protobuf is more common in the internal communication of Pods with each other.

"The API is where all Kubernetes resources are defined. It's large, modular and RESTfull" [Pou21] .At the earlier stages of Kubernetes the API was designed with a monolith approach, but as new features were coming through the time the need of diving it in groups was becoming clear. These groups can be divided into two main groups:

- Core group
- Named groups

The core group is mainly composed by the components that are part of Kubernetes since the beginning such as Pods, Nodes or Services. The API exposes this group through the path */api/v1* and followed by the necessary resource. If the object belongs to a Namespace, then before the reference to the object it is necessary to insert the correct namespace, as showed in the following figure.

Figure 2.4: Generic API path for a resource

The named groups are for new features that Kubernetes got along the years, which are divided accordingly to its functionality. Groups such as networking, storage or authorization are some examples of them. One of the main differences is that the API Path does not start with /api but with /apis/group.

This new features that are added to Kubernetes along the time, always go through a process of approval. This process is composed by three states:

- Alpha
- Beta
- Stable

All the features are available to be used in Kubernetes through different API Paths, but only the features in production ensure the total security of it and the best optimization possible. The features in alpha are available through the path /apis/group/v1alpa1 or /apis/-group/v1alpa1. To access the features on the beta state the path us /apis/group/v1beta1

or /apis/group/v1beta2, features that graduate to beta state usually graduate to the Stable state.

Cluster Store

Another important piece of a Kubernetes cluster is the cluster store, and his importance comes from the responsibility of persisting the cluster configuration as long as the state of the cluster. This is one of the reasons why it is wise to always have multiple replicas of the control plane so the state can be recovered in most cases.

At the moment the Cluster Store is based on the distributed database *etcd* that is present on the majority of the Linux distributions. *"On the topic of availability, etcd prefers consistency over availability"* [Pou21] which means that configurations that are bad applied to the cluster and it will remain in a state of nom changed configuration. When writing to the cluster store, if two events collide in the writing of the same object, then the RAFT consensus algorithm is used to achieve a good outcome.

Scheduler

The scheduler is the piece of the control plane that defines which pod does. When a task arrives to the cluster, it needs to be assigned to a pod so it can be executed. This is achieved through a ranking system. This ranking system will get the nodes capable or running the job, and that group of pods will be ranked accordingly to a set of characteristics and the highest-ranked pod will be selected. If there is no pod available capable of performing the task, then that same task will be held in the state of pending.

Controller Manager

The controller manager is the part of the control plane, that manages and monitors the controller components of the cluster. Each controller has different functions, that when combined makes the cluster run as expected. Controllers are in constant observation of the API Server looking for changes to be applied. The controller manager has the responsibility of monitoring all the controllers to guarantee that the observed state of the cluster matches the desired state saved on the cluster store. This monitoring process is shown in figure 2.5.

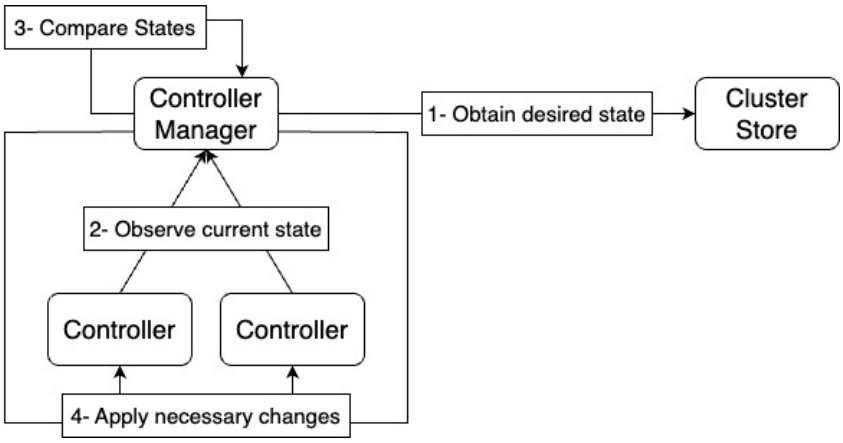

Figure 2.5: Controller Manager process of monitoring

By making each controller specialized in a small piece of the cluster, it means that no responsibility is over other controller, which allows to create more isolation. This isolation is very important in order to achieve a good Microservices implementation and eases that process.

2.2.3 Kubernetes Pod

In Kubernetes, a Pod is the atomic unit of scheduling, which means that in a Kubernetes cluster no containers are deployed, but Pods. Pods are deployed into Nodes, which can be virtual or physical machines, as well as other options. Inside a Pod there will be a container running, but the Pod will provide the container inside with more features. In general a single Pod will a run only one container, but it is possible to run multiple containers in the same Pod, which are called multi-container Pods. On this scenario, the deployed containers are not the same, but complementary containers, which have dependencies. These dependencies are ensured by making the containers share the same environment created by the Pod.

Pod Deployment

Pods can be deployed into Nodes by two different methods:

- Via Pod manifest
- Via higher-level controller

If a Pod manifest is used, then the Pod will become a static Pod, which means that it wont have the enhancements that a controller can provide to a Pod, such as auto-scaling or automatic restarts. This is because a static pod is only monitored by the kubelet service running locally, which means that if the node where the Pod is running fails, nothing can bring the Pod back because, the kubelet process will not be running as well.

When a Pod is deployed through an higher-level controller it will be gifted with a much more features than a static Pod. These features will differ from Pod to Pod, depending on the chosen controller. Different controls became pretty useful since different Pods, have different needs, so choosing the correct controller is a pretty important task. By being deployed by a controller it will be constantly monitored by the controller manager in the control plane. This allows the restart of the Pod in almost any scenario, even if the Node, where it was originally, is not available, and so it is deployed in another Node.

The deployment of a Pod is an atomic operation, which means that it happens totally or it doesn't. If something in the node, when deployed is not accordingly to the desired state in the cluster store then it will be deleted. If a running Pod starts to work abnormally, it will not be restarted or restored and it can't be updated with new things. It is eliminated and a brand new Pod will be deployed, with new IP, new memory, and storage.

Pod Scalability

When it comes to scalability, Pods are extremely scalable but different from the conventional architectures, the scalability is horizontal and not vertical, more Pods are added and not containers to a single Pod.

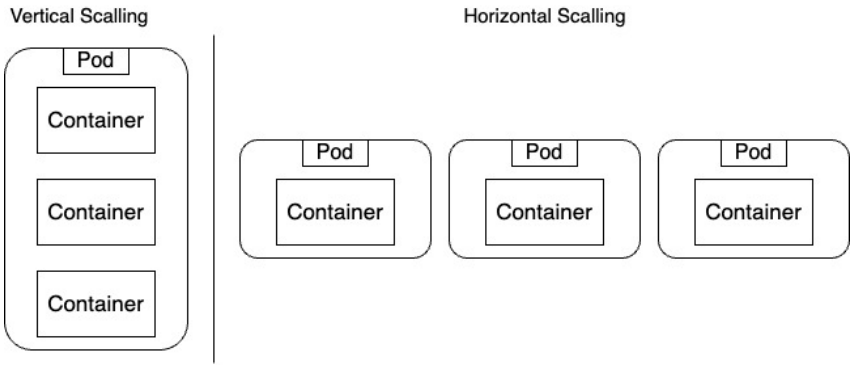

Figure 2.6: Vertical vs Horizontal scalling

Long-lived and Short-lived Pods

Pods can also be divided into two other different types. The short-lived pods are created to perform single tasks that isn't supposed to be running indefinitely, such as batch jobs. Normally these types of Pods are not meant to have a restart policy. In the opposite side, long-lived Pods are meant to always stay running, and if it´s process is broken, usually this type of Pods have a well-defined restart policies.

2.3 Security in Kubernetes

Kubernetes is a technology, from what has been seen so far, that is capable of hosting an entire system, which seems a lot powerful but, with a lot of aspects to cover can be difficult to guarantee the safeness of them all, since it covers all the areas of IT.

Nevertheless, to host and manage a cluster, in the most of the scenarios it is wise to use external services capable of doing changes into the cluster, or even deploy it all, but also services to store the images that will be used on the pods, and analyze them to guarantee the safeness of them. So, this means that security in Kubernetes, goes about a wide range of topics from different areas.

For all these reasons, compiling the important points of security to take in consideration on a cluster will help in guaranteeing that the security of the cluster is on the desired level. For that, one of the first steps to take is to divide the main topics to have in consideration for a cluster and its safeness. Taking in consideration the general knowledge of the IT world and generalizing to all the types of architectures, those topics can be six:

- Network
- Data
- Authentication & Authorization
- Performance
- Development
- Deployment

This six topics can have an overview of an entire systems, since in each one there are may points to take in consideration.

2.3.1 Network

Network in IT is the part that is related to the communication that a system needs to work as expected, from the type of connections that are established, on how those are established, to what ports are opened or not, what is closed to the exterior and all the things related to connecting to the cluster and communicating with it. Transporting the general knowledge of the IT world to a Kubernetes cluster, most of the terms and techniques are the same, with some changes. For example to start, one of the most notable differences is that, on a monolithic approach the application is running on one or more machines and only the physical configuration needs to be applied to the machine, but taking it to Kubernetes, beside the machines that are hosting the nodes of the cluster, the pods will be configured to, and need to be created always as that.

Another important technique that is used to protect systems is the implementation of a VPN on the company, so the system os not exposed to the outside and its only accessible inside of it. Not only by hiding the system from the exterior but also acting as Access controller, because if the accesses are logged, then it is easier to identify possible causes of some error that might occur on the system.

Also important when discussing about network, is how the communication is done to the system. if it is a service to be available to clients, it certainly has to expose part of the system to the exterior, so the users can access it. The type of connections that the company uses is up to the needs of the system, nevertheless, there should always be some type of service analyzing the connections, to verify if there is some type of unusual patterns, that might indicate possible attacks.

2.3.2 Data

For a company one of the more valuable assets that exists is the data that is stored, either could it be important documents, as long as, data from clients that needs to be stored. Keeping this data secure is one of the main objectives when talking about security on an IT system, and if it is not secured, and the data is exposed and used by unauthorized people, the company can suffer severe legal consequences.

One of the most usual things to do and implement in a system, is normally the encryption of data that needs to flow through the network, because this way if someone is listening to the network and can reach to the information flowing on it, if it is encrypted, it will not see the information directly, and would likely need to find the encryption method even before of starting the decryption.

When the data is received by the system it needs to be stored, on the most of the scenarios a database will be used for this, and in the case of a Kubernetes cluster, it is possible to use almost any type of database service. Part of the cluster will be hosting the necessary databases for the service, and it is really important, for the safeness of the cluster, to isolate this part as much as possible, since it allows much less possible attacks.

2.3.3 Authentication & Authorization

The security of any system starts on the persons that use it and manage it, and for that reason it is really important to manage the users carefully, so nobody has more permissions on the system than it is supposed. But the term users can apply to different services, and different types of users.

In the general there will be 2 types of users, the client user, that is a real person that controls it, and service accounts that will be used by automatic tools, to perform changes into the cluster. The client users, should normally by managed in groups, because it is gives more control in the attribution of permissions, since it makes unnecessary to provide permissions one by one, but to give those to groups and make the users part of the desired group. The service accounts, works a bit different because, it will be created an account for each different service that needs one, and the permissions of each one, will differ from service to service.

Service accounts, has being used in external services, there is need to provide those to the system, but the credentials must never be exposed in any type of script or automation declaration, but in the service itself with only the authorized people, able to reach those credentials if it necessary to perform nay change to the account.

2.3.4 Performance

In any service that is delivered by system, its availability is an important point for the clients that uses the service, and nobody wants the service down. So, making the service as available as possible becomes a concerning for any service provider, and since it is a topic that allows a better up-time and service, it becomes a security concern.

With this the performance of a system, on the project scenario, more looking into the Kubernetes world, as a wide range of points that can be discussed in order to achieve security on this. The main points performance is the delivery of hardware that is hosting the cluster, either it is deployed on-prem, with physical machines prepared and optimized for the purpose, either the cluster is deployed on the cloud, where it is necessary to configure the correct infrastructure to deploy the cluster.

Beside deciding which environment will be used and what type of machines will host the cluster, another part of performance of a system is planning the future, by knowing how the systems is supposed to run, and what are the needs for that, but not only the necessary should be used. There must be extra room to host more instances of services. This is important in the scenario, where an abnormal number of accesses is done, and the system is not prepared.

2.3.5 Development

As any other system in the IT world, it needs to be developed. In the Kubernetes world the development can be shaped in many forms. As the first, there is the image development, where in some cases the company might decide to develop the images from the zero, in other cases companies decide to get images from external vendors. In both cases, there are security concerns that need to be taken in consideration.

The development of a Kubernetes cluster is mainly focus on the images that will be running on the cluster pods, and those should be verified before being used on the cluster, to guarantee that there is no possibility of getting attacks that are obviously, from a security breach of the image. To verify images one of the first things possible is to verify the vendor of images, and if it does the image verification before being published. But also by external services that can be used in the IT infrastructure of the company.

Continuing in the images of the pods, another important topic its how they are stored and installed on the cluster, since to get a higher level of security it is wise to "hide" the cluster

as much as possible, and so the images shouldn't be installed directly from the vendor. A mid-term platform is necessary to download and analyze the images before using them in the cluster.

2.3.6 Deployment

As the last topic of security in Kubernetes, there is the Deployment. The deployment of a cluster is one of the most important moments, and it needs to be precise so it doesn't make the cluster run unexpectedly. The deployment can be performed manually or automatic, but if it is automatic a service needs to have access to the cluster, if it is manual, it can be an intense process and exposed to human errors.

This way, most companies are implementing automatic ways of deploying the cluster automatically. This is normally achieved through an external services that allow to define pipelines that will perform the desire steps in order to deploy the cluster in the correct place. This is a security concern because it is a tool, that will have power to perform creation and deletion of objects in the machines that are hosting the cluster. So maintaining the credentials that will serve the external service, secure is a really important point of security.

Complementing with the performance topic, more related with the up-time of the service, it is important that the deployment doesn't affect the availability of it. That's why a structured plan of how the machines are turned up and down to receive the deployment, needs to be defined to always keep part of the service running.

Chapter 3

The Framework - Analysis

Returning to one of the focus of the project, that is defining what security is on Kubernetes and try to find a solution to acquire a security level of a Kubernetes cluster.

3.1 The Problem

As already said before, Kubernetes is a technology emerging and capable of hosting entire systems, and providing a company with all their needs. Nevertheless, with all this power in one tool, there will always be some concerns in using it, regarding possible security problems it might bring. To make it easier to prevent possible security breaches and flaws, a checklist with some general and common points in any Kubernetes cluster can help. And with this list, basic practices and ideas can be defined and bring to the world of Kubernetes, a first glance on how to guarantee the safety of a cluster.

3.2 The Analysis

Approach this problem looking to Kubernetes as one big piece is probably not the best option, as that, dividing Kubernetes in different topics is wise, so each of the topics can be analyzed individually and more in depth. As discussed before on this document, those topics can be applied in clusters as in normal systems. So following the basis, it is possible to say that, security in Kubernetes can be divided in 6 different topics:

- Network
- Data
- Authorization and Authentication
- Performance
- Development
- Deployment

Taking in consideration the enumerated topics, the exploration of each one is the next step, because inside of each topic there will be some points to check on the clusters in order to guarantee the safety of them. So a diagram that shows an high-level idea, of what it is supposed to acquired knowledge and result in security for a cluster, is in the image 3.1.

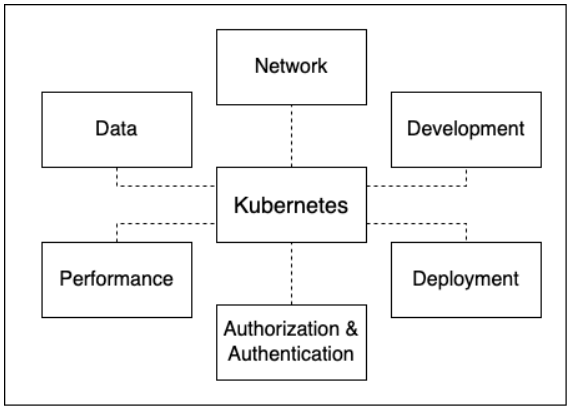

Figure 3.1: Project high-level idea

Now that the topics are defined, what is supposed to do, is deep diving in each one taking always in consideration the implementation of a Kubernetes cluster. With the knowledge to describe what are the more important points in each topic, that lead to a safe cluster.

3.2.1 Network

As in any IT infrastructure and project, some of the most critical points to guarantee a safe infrastructure come from the **network**, which is a big is most likely present in all components of Kubernetes cluster, since pods communicate with each other through the API, as long as all the data that will be used by the application or service that is running on the cluster. Beside all this internal communication, nowadays most of the application or services that are developed, are supposed to be available to a client, which means that connections to the internet are necessary. With these needs the concern of exposing too much and unnecessary information are real and it becomes worse since in a cluster there are different machines working with different purposes with different configuration, and so maximum isolation is very important.

3.2.2 Data

As other important topic about security is the **data** and it becomes a bigger concern on Kubernetes because it flow through the API to be transfered from pod to pod. Once again, the isolation must be at the maximum so the least connections are made to the pods that will store the data, and if possible don't have a direct connection to a pod that is connected to the internet, in order to have a better security, since in the major of the cases the stored information is highly confidential.

3.2.3 Authorization and Authentication

As the most "see-able" point in the security specturm there is the **Authorization and Authentication**, since it is directly associated to accesses to the cluster. Normally there are considered 2 types of accounts:

- User accounts
- Service accounts

The users accounts are associated to physical users that will directly access the cluster. This accounts must be precisely managed and preferably organized in groups so the permissions management becomes a lot easier. In the other hand there are the accounts of the clients that are supposed to use the service, which occurs through the internet. For this reason the authentication service and the data storer of the necessary information, are as isolated as possible. The Service accounts are destined to be used by external services that need to perform actions directly on the cluster, such as CI/CD platforms that can deploy the whole cluster, if well configured, and the credentials must be secure when stored on this platforms, guaranteeing that nobody that isn't expected to do so. Inside the cluster these account must only have permissions for the necessary actions because they are exposed to higher risk of exploit.

3.2.4 Performance

Another point that is directly related to the security of any IT infrastructure is the **performance**, because a good performance of a system is a step further into a secure system, since the majority of system breakdowns are avoided. Kubernetes already helps solving one of the problems in performance, with the automatic replication of pods, but the main question is how and when to activate it, so the performance of the system is not affected, and beside that, how long it takes to put the new pod to work. Nevertheless, the pods need physical machines to run on, and choosing hardware to a Kubernetes cluster turns pretty different when compared to a monolith solution, since it can be divided through different machines and those can be different in order to adapt to needs of the part of the system that it will host. Beside the hosting and running the services, it is necessary to guarantee that the service is running smoothly and for that, it is necessary to implement monitoring tools so the state of the services and servers are controlled and analyzed and a better service is provided. Cupulate to the monitoring normally comes the alarms, that are used to advise the necessary people that a certain server or service is working out of normal.

3.2.5 Deployment

Now we take on the **Deployment** as next point of security. Normally a Kubernetes cluster is composed by several number of pods, and deploying them manually it's not wise since it can lead to a several number of errors. Following a DevOps mentality, as it is recommended, the deployment process must be as automated as possible in order to avoid human caused errors and the process of deployment becomes a lot faster providing a better product quicker. To do it, it is recommended to use the Kubernetes manifest that will declare the desired pods and the number of replicas, as long as many other cluster characteristics. Once again, the store of the manifest must be done with precaution because it declares all the cluster, and so, only a few people are allowed to see it.

3.2.6 Development

For the last security point for this project, there is the **Development**. Going deep diving this topic into a Kubernetes cluster it will more focus on the images used by the pods on its runtime. With Kubernetes being an open-source tool, it was a matter of time to see the rise of the number of images providers and not all of them are trust able, as long as, the images that the trust able providers make available. Normally trustable sources have a marker for the safe images that are reviewed by them, nevertheless it is truly important

to have an image scanner on the own infrastructure. In order to keep all the infrastructure safer, the images should be downloaded from an isolated service that will download the images from the internet and send them to the artifactory repository that should be part of the infrastructure, which will provide the images to be installed in the pods.

Chapter 4

Implementation

Now that the knowledge about Kubernetes is acquired, the problem is understood, the objectives are clear, and the security objectives, regarding the different topics of Kubernetes system are defined, it is possible to develop the way of providing an evaluation of the cluster security level.

4.1 Design

With a clear path of what is expected to do to achieve a good security level in each of the topics, to start the implementation of the solution, its design is the first step. The design of the solution starts with the definition of the points that are supposed to be validated in a cluster. Each of the topics will be evaluated individually resulting in a final percentage according to the number of points that go accordingly to what is expected.

With all the checklist verified and the sub-results of each topic, to generate the final security level of the cluster, all the results is taken from the average percentage of all the topics, which will result in a final percentage. Taking the final percentage the cluster will be evaluated in 5 possible levels of security:

- On Risk <25%
- Low 25%<= & <50%
- Medium 50%<= & <75%
- Low 75%<= & <90%
- Excellent 90%<=

The company, when having the final results, can look into the cluster with a perspective of improving the worst points on the actual scenario of the cluster.

4.1.1 Network

Are the necessary Services declared?

Services are the Kubernetes method of exposing a pod to the internet. Services use a selector to determine which pods will be affected by the Service, and so labeling correctly the pods will be necessary, so a clear separation of pods is created. It might be recommendable the use of Ingresses if the workload comes via HTTP to have a more controlled accesses.

Are the pods correctly labeled?

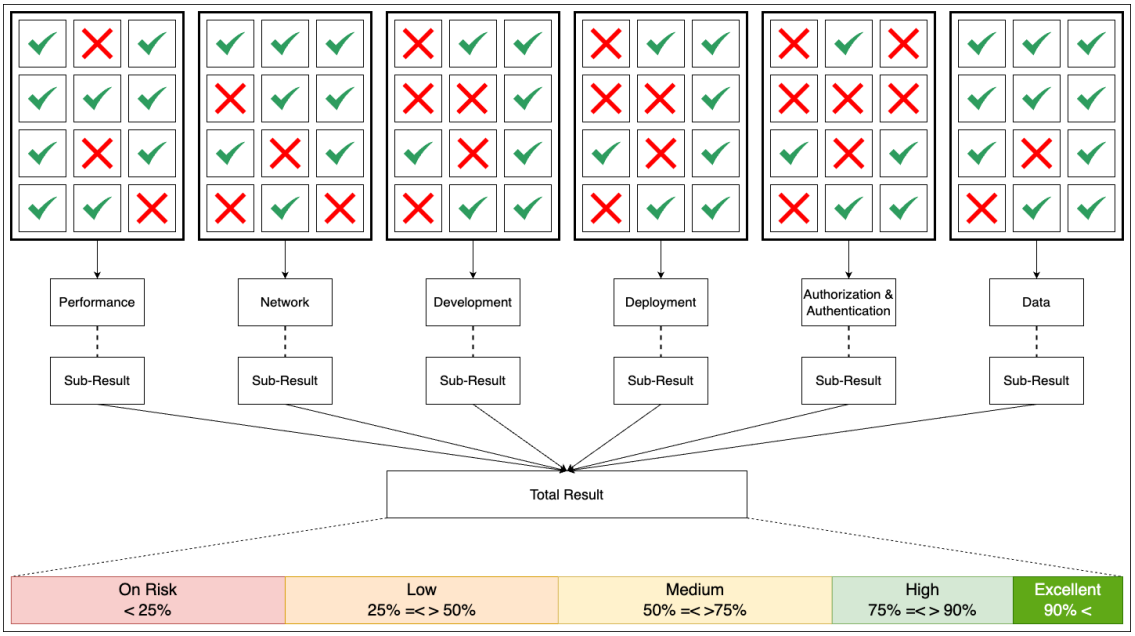

Figure 4.1: Framework workflow

The labeling of pods is an essential part to acquire the isolation that turns a cluster safe. By using labels on pods definition it is possible to manipulate pods as a group. One of the main usage examples is precisely in the Services, because using a selector in the Service it will only take effect into the pods that share the label in the selector.

Are the necessary Ingress created?

If the application running on the cluster works with HTTP or HTTPS, then it is wise the usage of Ingresses, since they are cappable of creating routes to services using DNS subdomains and also load balances the traffic to the Pods that are used by Service. Beside this, Ingresses allow to declare the ports to be used in the Service, which allows to have a better control of the open ports of Pods.

The usage of an Ingress implies the creation of an Ingress Controller that needs to be deployed into the cluster and it is the responsible of load balancing the traffic that incomes to the Ingresses controlled by it.

Is an Ingress Controller implemented on the cluster?

If the use of Ingresses on the cluster will exist then it is necessary to chose one of the controllers that are recommended by Kubernetes and is more appropriated to the system. The Ingress Controller will work as load balancer for the pods that will be affected by the Ingress.

Are the unnecessary pods isolated from the internet?

To avoid unwanted accesses to the cluster one of the measures in any system with or without Kubernetes, to improve the security is hiding as much information and services from the internet. So in Kubernetes it is necessary to block the external access to the unnecessary pods, so those only communicate internally to other pods.

Are only the necessary network ports open in each pod?

In the other hand to improve the security on the pods that need to be exposed to the internet, the less exposed the more secure it is, and for that reason by opening only the necessary number of network ports in each pod is essential.

Is the incoming traffic logged?

Another major concern of the teams about their systems are the attacks that can be suffered by a system. And to prevent or found a possible attack, is important to have logs of the traffic that is coming or tying to do so. By having the logs available, with its analysis is possible to detect possible attack tries due to unusual traffic from a single source, as many other options.

Is the API traffic logged?

Beside the traffic that comes from the outside in order to use the platform, the whole system working with the pods communicating with each other through the Kubernetes API. To ease the process of detecting possible errors on the runtime, or detect possible data loss, logging the Kubernetes API traffic becomes a must.

Is a load balancer service installed?

To guarantee that no pod is under to much workload, the load balancer service will ensure that the workload is well distributed between all the pods available for the required task.

Is a VPN service installed?

In order to turn the cluster safer, it is wise to enclosure it under a VPN because this way only computers connected to it can access directly the cluster. For that reason a VPN service with double factor of authentication should be running.

Is the artifactory repository isolated?

The artifactory repository is one important part of development phase of a cluster and should be present since its usage raises the security of any system, especially in a Kubernetes system that uses images to run in the pods that might come from the internet.

Is the artifactory repository available only through the VPN?

Just like the cluster the artifactory repository is a really important piece of the system and if not safe and exposed it can culminate in some undesired data being upload there, such as infected images. So by enclosing it inside the VPN prevents a lot of undesired connections.

Is the connection between the cluster and CI/CD platform safe?

As the CI/CD has the responsibility of deploying the cluster, it is necessary that the connection between both is extremely secure so nothing unwanted gets inside this connection in order to perform some type of malicious action.

Is the CI/CD accessible only through VPN?

Being such an important tool as the CI/CD is, it is necessary to get it as much protected as possible. One of the first steps for that is the enclosure inside the VPN so only users inside the VPN van access it.

4.1.2 Data

Are the credentials of the clients stored in an isolated service?

In a security perspective, the data from the clients is one of the most important assets that company has and stores. This data is accessible by the clients when using the correct credentials. So, the storage of those must be as isolated as possible with the minimum possible accesses. This includes only the authorization service.

Is the access to the storage logged?

In some cases it might be necessary to directly access the pods that are hosting the storage and to be able to track down some type of problem it is necessary to log the accesses to those pods and store them for a period of time.

Are the logs stored isolated from the cluster?

Since the major objective of the logs is to track down problems or bad function of the system, it is wise to store, at least, part of them outside the cluster, because if something stops the system it wont be possible to access the logs and investigate.

Are the database pods only accessible by the backend pods?

In the most of the systems it will be necessary a database to store the data of it. And in order to get a safe cluster, the pods that will host the databases should be as isolated as possible, and that means that only the backend pods that need to read and write to them, should have access.

Is the pipeline declaration stored safely?

Another important digital asset of an automated deployable cluster, is the pipeline declaration file, that will contain all the process of deployment of the cluster. As it seems, it really is an important asset and so its storage is very important, so only authorized people can access it, and edit it.

4.1.3 Authorization and Authentication

Are the users organized in groups?

In much of the systems there will be many users that will need different accesses, to different zones of the system, and managing them one by one can easily turn into impossible task, which can lead to a security breach by providing to much access to an unwanted user. For this reason, it is a must to organize users in groups and provide the permissions to the groups.

Is the control plane accessible only by the necessary people?

As one of the most important pieces of any Kubernetes cluster, the access to the control plane needs to be reduced to the minimum necessary. For this to be correctly applied, the easiest way is to provide to a reduced group of users that permission and remove it from any other group.

Is a multi-factor authentication service installed?

To improve the security of any system nowadays, it is a must to have a multi-factor authentication tool working at some point of access of the system, either at the access of the vpn or any other point.

Are the credentials stored encrypted?

As in any system, the encryption of credentials and some data is truly important, since it provides a new level of security if something fails, as for example, an unwanted access to the system or permissions that weren't supposed to be given. So having an encryption method for the most critical data is important.

Are the users accesses logged?

Another measure to track down possible errors on the systems, in this case, that might have been caused by human error, is the logging of accesses to the system, since it allows to see who has been at the system when the error occurred.

Are the accesses to the manifest made by authorized people?

One more asset that needs to be stored safely, is the manifest, since it contains all the infrastructure of the cluster and information from where images are retrieved and what images are used in the system. If this information is exploited then it might lead to strangers using the images vulnerabilities to enter in the system.

Is the manifest stored safely?

As a main asset of the cluster, since it describe the all system, the manifest should be stored in a safe place with the access extremely controlled. Every edit to the file should be reviewed for a second person.

Does the CI/CD have controlled access?

Being one of the most used and important tools in the whole infrastructure, since it will deploy automatically the cluster into the machines, the CI/CD platform should only be accessible by authorized people, and only available inside the VPN of the organization.

Are the Service Accounts with the minimum permissions?

The Service Accounts are meant to perform automatic actions as a user, and to avoid possible attacks with the use of this accounts, they must only have the permissions to perform the necessary actions, and nothing else.

Do the service accounts have a random password?

To avoid someone knowing the password of a service account, it should have a random password generated to it, and if possible being changed from time to time.

Are the service accounts credentials stored safely?

The service accounts will need to have their credentials stored in order to the platforms they serve can use it. It is extremely important that the least amount of people can access those credentials, because in the most of the cases it will have permissions to perform changes in the cluster.

4.1.4 Performance

Are the machines improved for the service they are running?

A cluster might be formed by multiple machines, and since different pods will be distributed between those machines, and there are different types of pods that do different tasks. So in order to achieve a better performance to the service it is wise to choose proper machines to the predestinated pods, that will run the service in the best way possible.

Is a monitoring service installed?

To control the state of the cluster and know that it is running as expected, it is a must, as in any other type of system beside Kubernetes, to have a monitoring service installed so the performance is reviewed from time to time.

Is pod replication configured?

As one of the best features of Kubernetes, replication of pods is a must configuration. This allows to put new similar pods working of some of then stop working, or if the resources are not enough for the demand, but it is necessary to have limits associated to the replication. This way the system knows when to start a new pod.

Is alarm service installed?

To guarantee the service is running as expected at any moment of the day, an alarm system should be running so alarms are sent to people so they are aware of the problems happening at the moment in the system.

Are defined values for the performance of the machine defined?

To be possible to implement the alarms the performance must be described to the point it is expected to work. The alarms will have 2 functions, to alert people of bad functioning and to inform the system that might be necessary more pods.

Are the resources allocated enough to guarantee a smooth service?

When creating a system it is necessary to know his needs, but it is not good to buy just what is needed, Basically if a system needs 100 GB of memory the smartes move is to implement 150 GB so at least more 50 per cent of the necessary is available.

Is reserve hardware available if more service instances are necessary?

To guarantee a better resolution of problems, the company must buy replacement parts for the servers, for when something crashes there material available to replace the broken pieces of the servers.

4.1.5 Development

Are the images used in pods developed internally?

The images used in the pods of the cluster can be retrieved from a image repository on the internet, or being developed inside the company, which is normally is the safest way since the company knows how the image is working and how to solve its possible problems.

Is the image repository trust able?

If the company decides to use ready images from a repository, it is really necessary to check everything about the repository that is being used, to guarantee the safeness of the images.

Are the images used in pods verified by the vendor?

In general of the cases, official distributors of images, such like Docker hub, have internal methods to declare an image safe or not, and its normally mentioned on the image page.

Is an image scanner service used?

Beside the verification of the vendor of the image, it is really recommended to have a service capable of analyzing the images that are downloaded, inside the company infrastructure.

Is the image directly downloaded to the cluster?

It is not recommended at all that an image is directly downloaded to the machine it will be used on, and as said before, the most of the machines must not even have connection to the internet, which makes it impossible.

Is the image saved on an own artifactory repository?

One of the best practices when using Kubernetes is having an own artifactory repository, where all the necessary images for the pods are stored. This repository will be connected to the CI/CD server in order to provide the images for the deploy.

Are the images used optimized for the purpose?

The development or selection of an image to a pod, must be well done, and this means that it must only do the necessary and as light as possible so it consumes the least amount of the resources.

4.1.6 Deployment

Is the cluster defined on a manifest?

To have a centralized source of truth, regarding the declaration of the cluster, doing it through the manifest will help in maintaining the cluster, as long as, doing updates to it.

Is a CI/CD service installed?

One of the methods for avoiding human errors in the deployment of the cluster or the compilation of images, is having a CI/CD service on the infrastructure. This way the deployment becomes automatic through a pipeline.

Are only the necessary secrets stored in the CI/CD platform?

The CI/CD service will need to have access to the cluster as other parts of the whole infrastructure such as the artifactory. For that it will need some secrets in it but only the necessary ones.

Is the pipeline declaration free from secrets?

In the pipeline that will deploy the cluster, it is mandatory that no secret is explicit on the declaration of the pipeline, since it is one of the most exposed elements of the infrastructure.

4.2 Implementation

Now that the design of the tool is developed, which consists on what will be the items that compose the checklist. With this items it is supposed to provide a way to the user to answer to this questions and obtain a result regarding the level of security of it's Kubernetes cluster.

As decided before, there will be 5 different results for the analysis of the cluster, regarding different levels of security. In the general of the companies, the objective is to have an excellent level of security in their systems, but if is not well prepared the security could be so bad in the system that leaves it at risk. This way the 5 possible results are:

At Risk Is the lowest level possible, where the result are below 25%, which means that the majority of the points are not in conformation according to the list, and the cluster could be exposed to the majority of the known possible attacks, as long as, not going accordingly to the data protection rules.

Low Is the second level of security, where some improvements can be noted when compared to level below, but in the general of the cases there will still be lots unsafe points, that will normally be exposing information that it is not supposed to be exposed. This levels extends until the 50% the questions are accordingly.

Medium Goes just to the 75% and the safeness of the cluster is improving, and at this level the security points that might be missing, can be related to automatic processes and services that are supposed to provide to the cluster, which means that at this level the data and the majority of the cluster itself might be at an acceptable level of security, but should be improved.

High Is the second highest level, and maybe for some companies this level that goes up to 90% of the questions going accordingly, is really acceptable, because the missing points, might be related to external tools and investments that the company might not be willing to do.

Excellent Is the highest possible rating of the checklist, which should be the expected result for any company that is implementing Kubernetes on their systems since it will mostly prove that the cluster is safe and there is not much to improve on the actual implementation.

4.3 Development

To develop something, simple, user friendly, and capable of being used in almost any platform, that are on the market nowadays, an excel file was created. The different topics of questions presented in the Design section, are divided through different pages on the file, which are accessible through Main Page that should open with the document.

In the Main Page it will be possible to find the results for each topic separately, as long as the final result accordingly to the percentages explained before. There is also a button that redirects to the Help page, so anybody that uses the tool, can understand how the tool works.

Each page of each topic has the same structure with the respective number of questions. Each question has 3 possible answers:

- Yes
- No
- Not Applicable

If the cluster that is being analyzed is in concordance with the item of the checklist, the user must select the option "Yes". In the other hand, if the cluster is missing the point that is being analyzed, then the response should be "No". Nevertheless, not all the cluster are equal or have the exact same configuration, and for this reason in some cases the point might not even fit on the cluster and for that there is the "Not Applicable" option. If the point is not filled with an answer the tool will count it as "No". In each topic page there is a button to return to the Main Page.

To have the final result of the security evaluation, in the main page there will be done the necessary math operations to obtain the final percentage of security points covered by the actual state of the company cluster. On the following figure (3.2), it is possible to see the main page of the developed tool in which the perfect result is displayed.

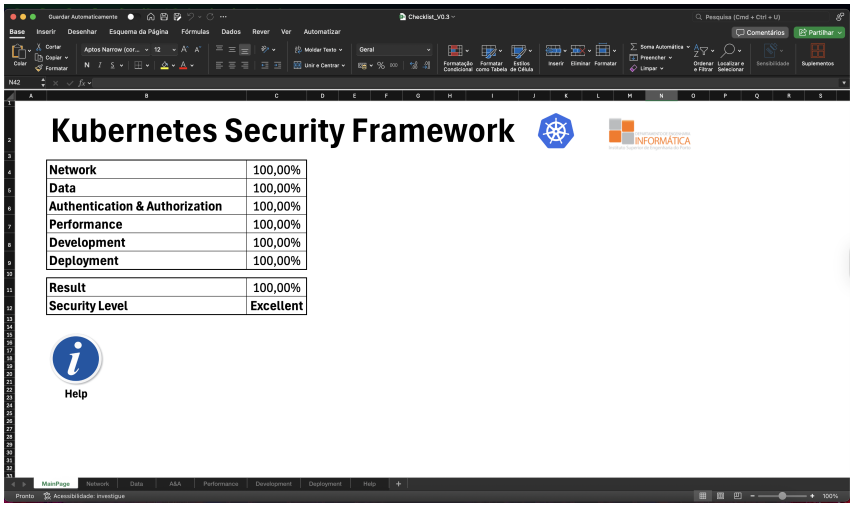

Figure 4.2: Main page of the tool developed

4.4 Testing

Any solution that is projected and brought to life is based on an idea, that was developed by someone, and that for the creators it will bring value to the community that the solution was designed for. But the true is that a possible solution to a problem can only be declared successful, when tested on real life situations and improvements can be made through that solution. In this project, one of the objectives is to provide an evaluation of the security level of a Kubernetes cluster. To verify that this objective and the developed tool, really brings value to the community and to the users, the best option is to make a company use it on their clusters, and let them understand what the results of the analysis can tell about the cluster.

As first tester, there was a company that offered to test the solution on their own clusters. The company acts on the telecommunications market, and provides many different types of services. At the moment of the development of this project, the company was in the process of migrating some of the services they host into Kubernetes clusters, but as the first experience with the technology for the company. The company hosts everything inside doors, which means that all the machines that will host the clusters, will be inside of their datacenter. This scenario is pretty good for testing the solution developed in this project, since the implementation is so fresh, and being the first experience with Kubernetes, it most likely to exists some flaws in the implementation, regarding the security of the clusters.

In the company there is a few number of people that directly works with the clusters, and has permissions to manage them. This group of people is precisely who should use the tool and verify the level of security of the clusters. Logically the first step was to introduce them to the tool and explain what it is, and its purpose and let them use taking in consideration what was implemented by the time. After a few days the results came from their usage, and some feedback was provided.

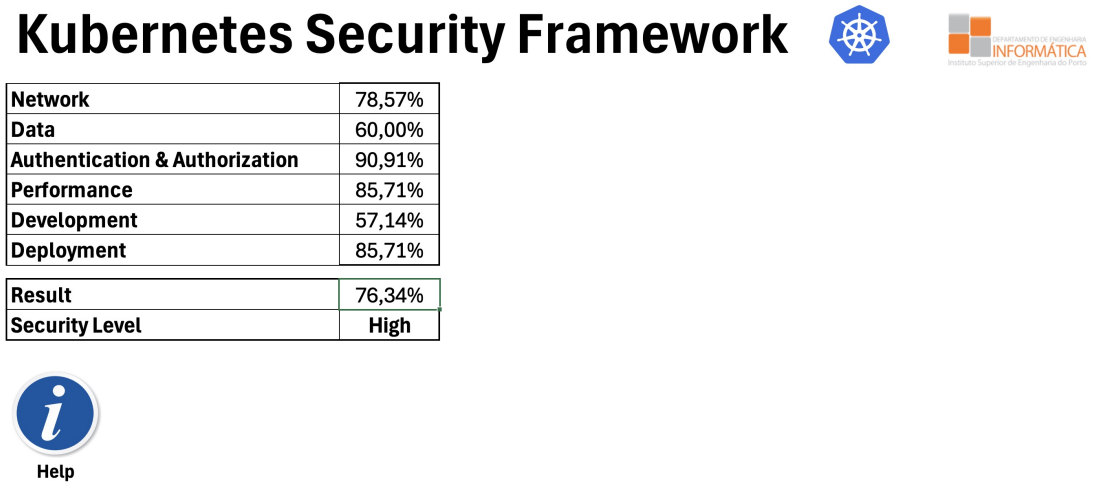

Figure 4.3: Test results

Firstly, regarding the interface and usability, the feedback was pretty positive, since in general all the users liked the interface for being simple and clear, but the help button was also pretty appreciated, even that they don't really needed it. They understood the choice of developing it, at this phase, using Microsoft Excel, but the idea of turning this solution more dynamic was a topic of discussion for future work.

Looking at the real results of the experience, the users looked into the cluster and all got the same result. The cluster got a High security level as result but for a very low margin since the final percentage was 76,34%, which is 1.34% above the minimum to get the High level of security. Since the company is in the beginning of the implementation of Kubernetes, it was an expected result from them, and they could take some ideas to improve the security of the cluster. The results can be seen on the picture 4.2.

The worst topic on the company's cluster is the development, in much related to the artifactory and in the used images security. This results made them think about the possibility of having an own artifactory repository, because that way, beside of making the cluster safer, because of not using direct connections to the internet, the images can be stored internally and analyzed if necessary. Another weak topic is the Data, which let them concern since there is many data about clients that cannot be shared or exploited. The implementation of the databases and data protection will be reviewed and taken in consideration the points that are missing by now.

Even that the testing scenario might not be the best or the most extensive, from this first experiment it is reasonable to say that the checklist helped a company to understand their security level and in what areas are they missing the most. Which means that this list can be either improved on its content, as long as, being updated to a more dynamic form.

Chapter 5

Conclusion

It is now time to look at the whole project and understand what was achieved. Throughout the project, there were some challenges to overcome, but the main one was probably to combine the working of Kubernetes with the security baselines that are used daily on the IT world, so it can be possible to answer the Research Question of this project, that is "How is security defined in Kubernetes?".

This project never wanted to look into what is being used inside the cluster, but how the cluster is managed and deployed. This is a very important point regarding this project, since this way it is possible to have a more general overview of any type of cluster Kubernetes that might be giving service of any type. This means that by having a more general approach, it allows us to provide a tool capable of being used in a more wide range of companies.

Throughout the document it was notable that Kubernetes is a pretty powerful tool, but to take the maximum of it, it is really necessary a good configuration, because a simple mis-configuration all the cluster could be at risk. It is normally, not only dependent of itself, because some external tools or services will be necessary to configure and manage the cluster, avoiding human errors, which is one of the reasons of some security problems that might appear. Beside external tools and services, as it was said before, it is also necessary a different working and development mentality, because if the teams do work separately, it is probable that it will take longer to get the system up and running.

But with all this information, all the work done so far, it remains the question: How is security defined in Kubernetes? Well, across the project multiple points were taken into the perspective, reaching different areas of the IT world, and it is important to be like this, since Kubernetes could handle a whole complete system that can have multiple parts from multiple domains of IT. From the Network to Deployment and passing through Data, all the points are important when talking about Kubernetes.

With all this, according to this project, part of the RQ can be answered, because security in Kubernetes is not defined by a singularity, but a composition of factors and ideas. The security of a cluster starts "in the paper", by deciding how the cluster is declared, what will be hardware needs of the systems or if it will be hosted on the cloud. The hosting environment is an important point and the framework is designed to have the questions generalized enough so it is applicable to both environments.

After the basis is defined, comes the configuration of the cluster on the manifest, to guarantee that nothing is more exposed than what is intended, either on the network side or the data. The cluster has to be configured as it is planned so no pods read more data than what is truly necessary, But it is also important to limit the network access sf a pod, because if

it doesn't need network connectivity to perform it's necessary action, then it doesn't need access to network.

Nevertheless, the security of a cluster starts outside of it, because there are tools that will support the deployment of the cluster, as others will help in maintaining the used images as safe as possible. Because deploying a Kubernetes cluster can be a very complicated task, and where human errors can occur quickly and result in a bad configured cluster. This way a automation is necessary to do the deployment, which is normally acquired through a CI/CD server, that if possible, should be installed inside the company infrastructure.

But, all the points mentioned so far are not useful if the accesses are not controlled. That's why the users should be managed and controlled from time to time and organized in groups to have a proper management of permissions in the whole platform. But the accesses are applicable to every part of the platform that is directly connected to the cluster, from the CI/CD server to the VPN service used by the company.

After all this points being connected with each other, and making each other working on a safer manner. Nevertheless there is one thing that crosses all the points, because it is necessary for everything that happens inside and around the cluster, which is logging. Even if a system is considered safe, errors can always happen as long as zero-day attacks, that nobody expects. If the whole system is providing logs of the activities and accesses, it will ease the process of identification of the causes of the problem.

This is what this project brings, as defining security in Kubernetes, a lot of components that if configured in the correct way, with a limited access and as automated as much as possible can turn a cluster safe.

Looking in to the future, the next step for this project might be taking the tool out of an excel file and make executable on a standalone application, and looking even more into the future, maybe migrate it to an applicable service to Kubernetes, so it can automatically read the cluster and provide a document with the analysis of the cluster and where is should be improved.

Appendix A

Big Figures

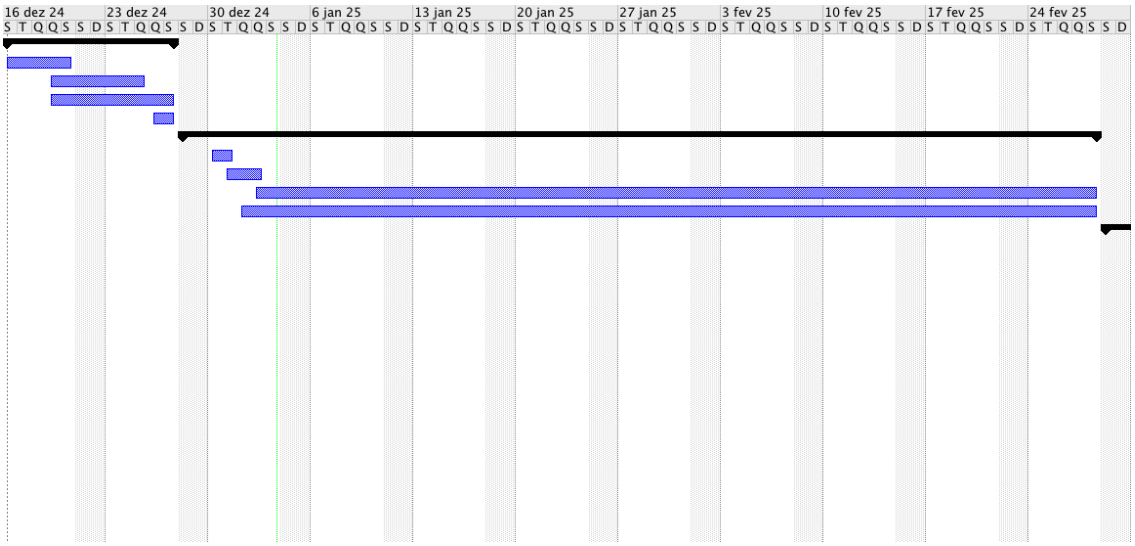

Figure A.1: Gant Chart (16/12/2024 - 02/03/2025)

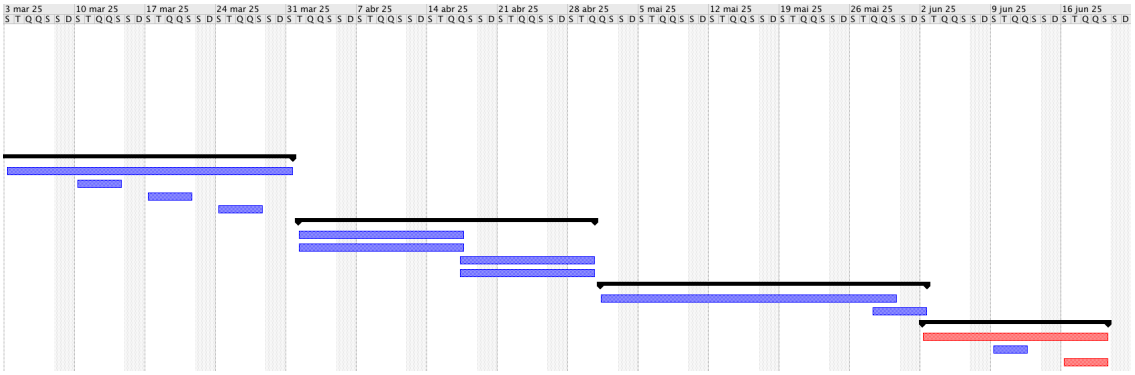

Figure A.2: Gant Chart (03/03/2025 - 22/06/2025)

Bibliography

[Thö15] Johannes Thönes. "Microservices". In: *IEEE Software* 32.1 (2015), pp. 116–116. doi: 10.1109/MS.2015.11.

[Bur+16] Brendan Burns et al. "Borg, Omega, and Kubernetes". In: *Commun. ACM* 59.5 (Apr. 2016), pp. 50–57. issn: 0001-0782. doi: 10.1145/2890784. url: https://doi.org/10.1145/2890784.

[Dra+17] Nicola Dragoni et al. "Microservices: Yesterday, Today, and Tomorrow". In: *Present and Ulterior Software Engineering* (2017), pp. 195–216. doi: 10.1007/978-3-319-67425-4_12.

[PSP17] Pulasthi Perera, Roshali Silva, and Indika Perera. "Improve software quality through practicing DevOps". In: *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)* (Sept. 2017), pp. 1–6. doi: 10.1109/icter.2017.8257807.

[IAR20] Md. Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, and Akond Rahman. "XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices". In: *2020 IEEE Secure Development (SecDev)*. 2020, pp. 58–64. doi: 10.1109/SecDev45635.2020.00025.

[Pou21] Nigel Poulton. *The kubernetes book*. Amazon, 2021.

[BOP22] Grzegorz Blinowski, Anna Ojdowska, and Adam Przybyłek. "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation". In: *IEEE Access* 10 (2022), pp. 20357–20374. doi: 10.1109/ACCESS.2022.3152803.

[Lin+22] Author: Paul Jansen Chief Executive Officer Follow Paul Jansen on LinkedIn et al. *Tiobe index*. June 2022. url: https://www.tiobe.com/tiobe-index/.

[Kub] Kubernetes. *Kubernetes Kubernetes Documentation?* https://kubernetes.io/docs/home/. Accessed: 2025-09-24.

[Red] RedHat. *RedHat Containers*. https://www.redhat.com/en/topics/containers. Accessed: 2025-09-22.

[Ste] Ian Smalley Stephanie Susnjara. *IBM What is Kubernetes?* https://www.ibm.com/think/topics/kubernetes. Accessed: 2025-09-23.