PROYECTO FIN DE GRADO

TITLE: DevOps. CI/CD implementation

AUTOR/A: Gabriel Enrique González López
TITULACIÓN: Grado en Ingeniería Telemática

TUTOR/A: Javier Martín Rueda
DEPARTAMENTO: Ingeniería Telemática y Eléctrica

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Rafael Delgado López

TUTOR/A: Javier Martín Rueda

SECRETARIO/A: Esther Gago García

Fecha de lectura: Jueves 21 de Septiembre de 2023

Calificación:

El Secretario/La Secretaria,

I. Resumen

Este proyecto de fin de grado presenta las prestaciones y mejoras que aporta la metodología DevOps a una organización. La metodología DevOps es una forma de trabajo que integra el desarrollo y la operación de software, con el fin de mejorar la calidad, la eficiencia y la satisfacción del cliente. Para ello, se ha implementado dicha metodología en un entorno Cloud para impulsar aún más sus ventajas, mostrando las capacidades de la automatización, la mejora de la seguridad y la innovación continua.

La infraestructura de la nube elegida ha sido Azure por su amplio abanico de servicios y ofertas beneficiosas para el desarrollo de este proyecto.

En este proyecto, la propuesta tiene en cuenta un diseño orientado a contenedores, entornos aislados y portátiles que pueden ejecutarse de manera consistente en cualquier sistema operativo. Se lleva a cabo la implementación de un flujo de trabajo de desarrollo continuo e integración continua (CI/CD) utilizando Azure DevOps como orquestador de dicho flujo. Se ha desarrollado para ser implementado con Kubernetes, una plataforma de orquestación de contenedores que permite su fácil administración, y al utilizarse la plataforma Cloud Azure, se utiliza su solución Azure Kubernetes Service (AKS).

Este proyecto ejecuta una implementación de infraestructura como código donde las aplicaciones pueden ser desplegadas en cualquier plataforma Kubernetes con mínimas modificaciones.

Se han implementado todas las fases de una metodología DevOps con su correspondiente solución software, proveniente de Azure o software libre. Para demostrar dicho flujo se han utilizado dos desarrollos con entornos y tecnologías dispares para demostrar la flexibilidad de la metodología. Un desarrollo de Frontend con JavaScript y el otro de una aplicación en Python Flask con base de datos MySQL.

Para testear dicho código de manera automática se usa la herramienta SonarQube, el cual informa de vulnerabilidades o bugs encontrados en el código analizado. También se monitoriza los desarrollos e infraestructura con los servicios Azure Log, Azure Monitor for Prometheus y Grafana.

II. Abstract

This final degree project presents the benefits and improvements that the DevOps methodology brings to an organisation. The DevOps methodology is a way of working that integrates software development and operation in order to improve quality, efficiency and customer satisfaction. To this end, the methodology has been implemented in a Cloud environment to further boost its benefits, showcasing the capabilities of automation, improved security and continuous innovation.

The cloud infrastructure chosen was Azure for its wide range of services and beneficial offers for the development of this project.

In this project, the proposal takes into account a container-oriented design, isolated and portable environments that can run consistently on any operating system. The implementation of a continuous development and continuous integration (CI/CD) workflow is carried out using Azure DevOps as the workflow orchestrator. It has been developed to be implemented with Kubernetes, a container orchestration platform that allows easy administration, and by using the Cloud Azure platform, its Azure Kubernetes Service (AKS) solution is used.

This project executes an infrastructure-as-code implementation where applications can be deployed on any Kubernetes platform with minimal modifications.

All phases of a DevOps methodology have been implemented with its corresponding software solution, originated from Azure or free software. To demonstrate this CI/CD flow, two developments have been used with different environments and technologies to demonstrate the flexibility of the methodology. One was a Frontend development with JavaScript and the another was a Python Flask application with a MySQL database.

To test this code automatically, the SonarQube tool is used, which reports vulnerabilities or bugs found in the code analysed. The developments and infrastructure are also monitored with the Azure Log, Azure Monitor for Prometheus and Grafana services.

Table of Contents

I. Resumen 1
II. Abstract 2
III. Table of Figures 5
IV. List of acronyms 6
Chapter 1. Introduction 7
1.1 Motivation 7
1.2 Objectives 8
1.3 Document structure 8
Chapter 2. Definition and Technical Framework 9
2.1 The need for Devops 9
2.2 Devops Definition 9
2.3 Devops Life Cycle 11
2.4 Devops and Agile Methodologies 13
2.5 Types of Architectures 14
2.6 Kubernetes 14
2.7 Azure 15
2.8 Code Version Control 16
2.9 Helm 16
2.10 Software Framework 17
Chapter 3. Design specifications and constraints 18
Chapter 4. Description of the proposed solution 19
4.1 Environment Explanation 19
4.2 Devops tools used 19
4.2.1 Azure Devops 19
4.2.2 Azure Kubernetes Service (AKS) 20
4.2.3 GitHub 22
4.2.4 Sonarqube Community Edition 22
4.2.5 Azure Monitor for Prometheus 23
4.2.6 Azure Managed Grafana 24

4.3 Technologies used to develop the applications 24
4.3.1 NodeJS 24
4.3.2 Flask 25
4.3.3 MySQL 25
4.4 Devops Platform Design 25
4.5 Kubernetes Microservice Architecture Design 27
4.5.1 Kubernetes Services 29
4.5.2 Kubernetes Deployment 30
4.5.3 Kubernetes StatefulSet 32
4.6 JavaScript Calculator App 32
4.6.1 The Frontend 33
4.6.2 The backend 34
4.6.3 Deployment 34
4.7 Python Flask Blog App 35
4.8 Azure Devops Pipeline 40
4.9 Testing Phase 43
4.9.1 Azure Devops Project 43
4.9.2 SonarQube Analysis 45
4.9.3 Executing the test on the AKS 47
4.10 Monitoring Phase 47
Chapter 5. Budget 51
Chapter 6. Impact of the Project 54
Chapter 7. Conclusions 56
7.1 Meeting Objectives 56
7.2 Problems Encountered 56
7.3 Future Work 57
Chapter 8. References 58

III. Table of Figures

Figure 1. Devops Life Cycle and Tools 11
Figure 2. Agile Methodology 13
Figure 3. Kubernetes Container Deployment 15
Figure 4. AKS Overview 22
Figure 5. CI/CD Chain 27
Figure 6. Microservices Architecture on AKS 29
Figure 7. Kubernetes Service Concept 30
Figure 8. Calculator app 34
Figure 9. MySQL Configuration 37
Figure 10. Python Blog App 38
Figure 11. Python Blog Register view 38
Figure 12. Python Blog App with User logged in 39
Figure 13. Python Blog App Editing Page 39
Figure 14. Azure Devops Pipeline Summary 41
Figure 15. Azure Devops Pipeline Jobs 42
Figure 16. Test Artefacts on Azure Devops 43
Figure 17. Test Results 43
Figure 18. Test Case 44
Figure 19. Mark Test as Passed 44
Figure 20. SonarQube project Overview 45
Figure 21. SonarQube Issues 46
Figure 22. SonarQube Quality Gate 46
Figure 23. Azure Monitor Workspace 47
Figure 24. Prometheus Monitoring Rules 48
Figure 25. Azure Logs 49
Figure 26. AKS Insights 49
Figure 27. Grafana Dashboard 50
Figure 28. Azure Costs 51
Figure 29. AKS CI/CD Microsoft approach 57

IV. List of acronyms

Nomenclature	Description
AKS	Azure Kubernetes Service
VM	Virtual Machine
CI	Continuous Integration
CD	Continuous Deployment/Delivery
QA	Quality Assurance
AWS	Amazon Web Services
VCS	Version Control System

Chapter 1. Introduction

1.1 Motivation

DevOps is a set of practices that aims to improve the collaboration and communication between software development and IT operations teams. DevOps also strives to automate and streamline the software delivery and infrastructure changes, resulting in a faster to deliver, more reliable, and secure software products.

One of the main motivations to show and implement DevOps methodologies is to increase the business value and customer satisfaction. By adopting DevOps, organizations can reduce the time to market, enhance the quality and performance of their software, and respond quickly to changing customer needs and feedback. DevOps also enables continuous learning and improvement, fostering a culture of innovation and experimentation.

Another motivation to show and implement DevOps methodologies is to optimize the use of resources. DevOps can help organizations eliminate waste, inefficiencies, and silos in their software development and delivery processes. DevOps can also leverage cloud computing, automation tools, and monitoring systems to scale up or down their infrastructure according to demand, saving money and energy. DevOps can also improve the security and compliance of their software products, avoiding potential risks and penalties.

Furthermore, using Cloud with Devops applies to achieve faster and more reliable software delivery. Cloud computing provides scalable and flexible infrastructure that can support Devops practices. By using Cloud with Devops, organizations can benefit from lower costs, higher performance, and greater innovation of the services that the Cloud provides.

If an organization wants to have the most innovative, efficient and modern methodology, it must have Devops. In this project we will see an example of how it can be implemented.

1.2 Objectives

The main objective of the project focuses on designing, implementing and configuring a series of tools that allow the application of the DevOps methodology for the development, testing, deployment and maintenance of a Web Application in Python and JavaScript language.

Specific objectives:

- The design of the architecture should be easy to maintain and easy to extend.
- The use of capabilities of Cloud services to create the platform.
- The development of both web applications with different needs to show Devops standards.
- The automation of building, testing and deployment of the applications source code.
- The monitoring and alerting of the web applications deployed on the cloud environment.

1.3 Document structure

The first Chapter of the project explains the motivation, objectives and how the document has been structured. Chapter 2 serves to explain the state of the art and a general explanation of the technologies used. In Chapter 3 the project specifications are explained.

The most important chapter is Chapter 4, as this is where the explanation of the project is elaborated. First we start by explaining in detail each of the technologies used, then the design of the DevOps platform, followed by the key concepts of Kubernetes, then an explanation of the applications and finally how the platform tools have been implemented.

The costs for the implementation of the project are indicated in Chapter 5. In Chapter 6 it will be highlighted the social, environmental, economic, technological and industrial implications related to the project, as well as the possible contribution to the SDGs (Sustainable Development Goals). To conclude, the Chapter 7 will summarise the conclusions extracted from the project.

Chapter 2. Definition and Technical Framework

2.1 The need for Devops

DevOps originated from the need to address the challenges and limitations of traditional software development approaches, which were characterized by soloed teams, manual processes, and lengthy release cycles. In traditional development approaches, development and operations teams worked in isolation, leading to miscommunication, lack of alignment, and delays in software delivery [1]. Some problems originated from that lack are:

- Poor communication and collaboration between developers and operations teams, resulting in delays, errors, and conflicts.
- Inefficient and manual processes for testing, deploying, and monitoring software, leading to low quality, high costs, and frequent failures.
- Lack of visibility and feedback on the performance and reliability of software, making it hard to identify and fix issues quickly and proactively.
- Difficulty in adapting to changing customer needs and market conditions, due to rigid and complex software architectures and processes.

2.2 Devops Definition

DevOps is a methodology that aims to bridge the gap between software development and operations teams by promoting collaboration, communication, and integration. It is an iterative and incremental approach that emphasizes agility, automation, continuous integration and delivery, and monitoring and feedback. DevOps combines the best practices of agile development, lean manufacturing, and systems thinking to create a culture of collaboration and innovation that enables organizations to deliver high-quality software at a faster pace.

DevOps is based on a set of principles that guide the methodology's implementation. These principles include continuous delivery, continuous testing, continuous monitoring, and continuous improvement.

- **Continuous delivery** involves automating the software delivery process, from development to production, in order to minimize errors and reduce the time required for software release.

- **Continuous testing** ensures that software is thoroughly tested throughout the development lifecycle, reducing the risk of defects and improving the quality of the software.
- **Continuous monitoring** involves tracking the performance of the software in production, in order to identify issues and take corrective action.

- **Continuous improvement** involves identifying areas for improvement and implementing changes to the software development process to increase efficiency and effectiveness.

DevOps also involves the use of a range of practices and tools:

- **Agile development** involves an iterative and incremental approach to software development, emphasizing collaboration, communication, and flexibility.

- **Lean manufacturing** involves the elimination of waste and the optimization of processes in order to improve efficiency and quality.

- **Infrastructure as code** involves the use of automated scripts to provision and manage infrastructure, enabling the automation of the entire software delivery process.

- **Configuration management** involves the automation of configuration changes to infrastructure and applications, ensuring consistency and minimizing errors.

- **Containerization** involves the use of containers to package and deploy applications, enabling greater flexibility and portability.

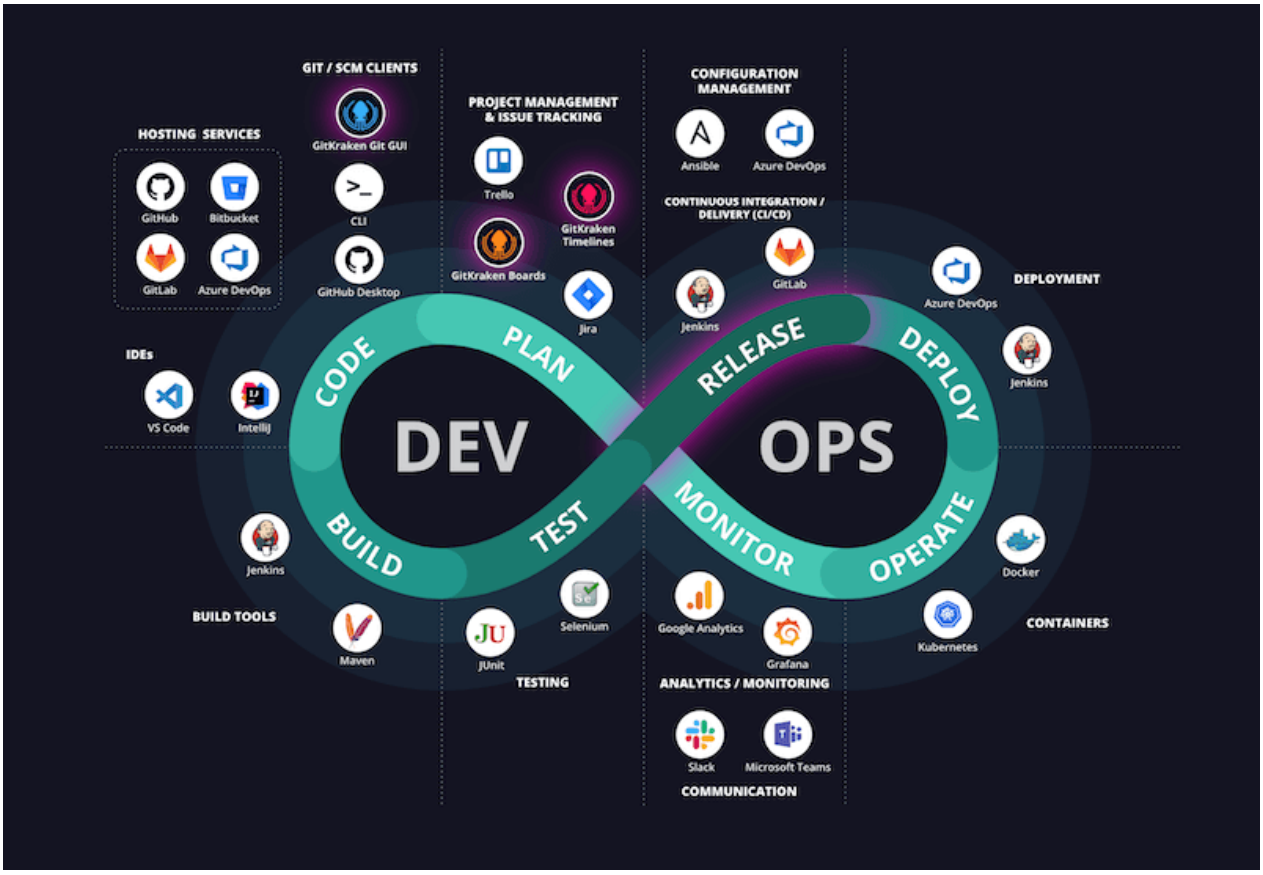

Figure 1. Devops Life Cycle and Tools [2]

The benefits of DevOps are numerous and far-reaching. However, the adoption of DevOps is not without challenges. One of the biggest challenges is cultural resistance, as DevOps requires a significant shift in mind-set and working practices. Another challenge is the complexity of the software development process, which can make it difficult to implement DevOps principles and practices. Finally, there is the challenge of selecting the right tools and technologies, as there are numerous options available, each with its own strengths and weaknesses [3].

2.3 Devops Life Cycle

The life cycle of a project applying the DevOps philosophy includes 6 distinct phases that are repeated iteratively during the life of the project [4]. Each of these steps is explained below:

- **Planning Phase**: In this first step, the requirements that the service or product to be developed in the project must meet are identified and prioritised. It also identifies which tasks must be carried out, and in what order, in order to carry them out according to what is most relevant for the end user.

- **Build Phase**: In this step, the requirements are already identified, and the development of the product code begins. As the name of the phase indicates, this is the moment when the software starts to be built.

- **Continuous Integration Phase**: This stage comes hand in hand with the build phase. In this phase, which should be carried out periodically or even daily, the newly built code is integrated with the existing code, as well as the new code with other code developed by the rest of the team members. In this phase, all the code is compiled, unified in a joint repository, and quality and security tests are carried out on the developments.

- **Deploy Phase**: One of the many advantages of using this philosophy is that the deployment of new components can be automated or programmed. Not only does this phase talk about software deployments in a production environment, but there are usually two environments prior to the final environment. The first environment, commonly referred to as the integration environment, is a test environment where deployments are usually performed to functionally test the developments once they have been integrated with the previous code. Another environment is the pre-production or staging environment. This environment is more similar to production, where more complete, or realistic, testing of developments can be done before a production deployment is made. This helps to detect and reduce possible errors that may occur in the production environment after deployment to end users.

- **Operate Phase**: In this phase, the performance of the new functionalities is monitored. Not just performance in terms of physical memory measurements, load times, etc. But monitoring of the incidents that may arise once the software has been deployed, such as those related to possible conflicts between components, or unresolved dependencies. Also this phase is called "*Monitoring"*.

- **Continuous Feedback Phase**: Finally, before starting the cycle again, it is important to keep in mind that communication channels should be kept open with all the stakeholders of the project to receive their opinions about the new functionalities. In this way, the plan for future improvements or modifications to the system can be revisited.

2.4 Devops and Agile Methodologies

DevOps and Agile are two methodologies that are closely related and often work together in software development projects.

Agile is a software development methodology that emphasizes flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto [5], which values individuals and interactions, working software, customer collaboration, and responding to change. Agile promotes iterative and incremental development, with frequent feedback loops between the development team and the customer or end user.

Figure 2. Agile Methodology [6]

The relationship between DevOps and Agile is that they are complementary methodologies that work well together. DevOps can help to support Agile principles by improving collaboration and communication between the development and operations teams. By automating the deployment and delivery processes, DevOps can help to speed up the delivery of new features and updates, which is a key component of Agile development.

In addition, DevOps can help to ensure that Agile development is successful by providing feedback loops and monitoring capabilities that can help to identify and address issues quickly. By improving communication and collaboration between the development and operations teams, DevOps can help to ensure that the software is delivered on time and with high quality.

2.5 Types of Architectures

Currently and previously Devops, there were several software architecture archetypes. The main ones are the following [7]:

- **Monolithic model**: All functionalities are coupled in a single application. Its advantage is that it is usually simpler to develop in the first instance, but increases in complexity depending on how much the system is intended to scale.

- **Three-layer architecture, or monolithic 2.0**; Split into 3 monolithic layers the presentation layer, the application server and the database layer. More modular than the previous, but scaling is still insufficient.

- **Modular model, or, Microservices**: Divides the system into small, independent units, making each unit an application in itself. Each of these units is simple, its implementation and scaling is independent of the others. There are other disadvantages, such as the number of integrated units being too large. Another disadvantage is that it requires more dependency management. But it provides much more flexibility than in the two previous cases.

The system designed in this project is based on a Microservices architecture.

2.6 Kubernetes

Kubernetes, also known as K8s, is an open-source container orchestration platform initially developed by Google [8]. It is designed to automate the deployment, scaling, and management of containerized applications. Kubernetes provides a unified API that allows you to manage and deploy applications across multiple clouds and servers, whether they are on-premises or in the cloud.

Kubernetes is built on the concept of **Containers**, which are a lightweight, portable method for packaging and deploying software applications. Containers isolate software from the underlying infrastructure, allowing it to run consistently and reliably across different environments. Containers allow developers to easily package an application and all its dependencies into a single, self-contained unit that can be easily deployed and run anywhere, regardless of the underlying infrastructure.

Kubernetes simplifies container deployment and management by automating tasks such as container scheduling, scaling, and orchestration. Kubernetes allows you to define how your containers should run, including the resources they need, how they should be deployed, and how they should communicate with each other. Kubernetes also provides features for automatic scaling, rolling updates, and self-healing, making it easier to manage and maintain containerized applications.

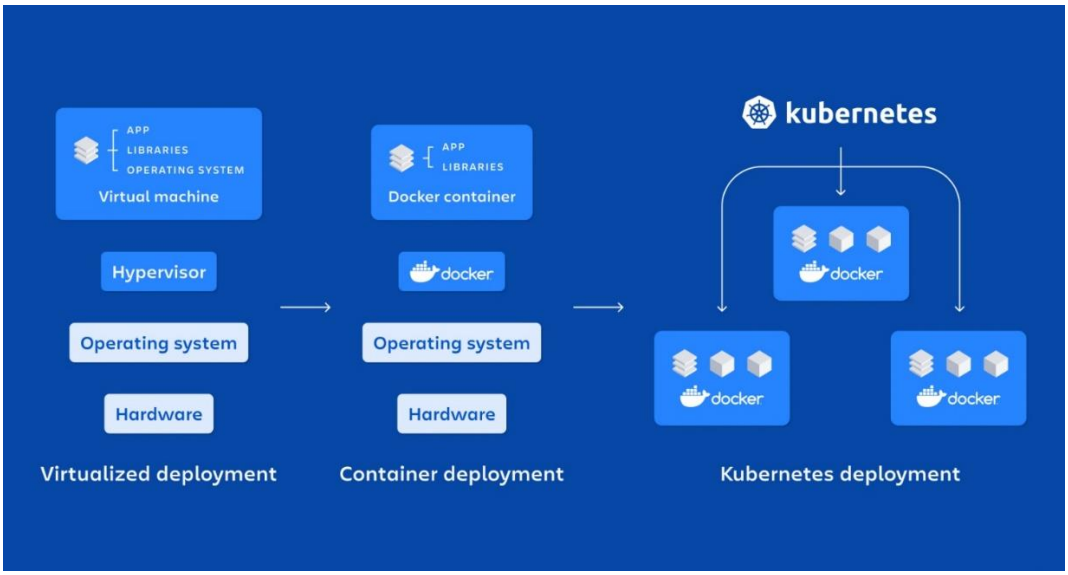

Figure 3. Kubernetes Container Deployment [9]

Overall, Kubernetes provides a powerful and flexible platform for deploying, scaling, and managing containerized applications in production environments. Its popularity has grown rapidly in recent years, and it has become a standard for container orchestration in the industry.

2.7 Azure

Azure is a cloud computing platform and a collection of services provided by Microsoft to build, deploy, and manage applications and services through Microsoft-managed data centers. It offers a wide range of services, including virtual machines, databases, developer tools, storage, and analytics. Azure supports a variety of operating systems, programming languages, frameworks, tools, databases, and devices, making it a flexible and scalable solution for businesses and organizations of any size. It also provides tools for monitoring, security, backup, and disaster recovery, among others, to ensure the availability and reliability of applications and services hosted on the platform [10].

An **Azure subscription** is a type of account that allows users to access and use Azure cloud computing services. It provides users with access to various Azure resources, such as virtual machines, storage accounts, and databases.

2.8 Code Version Control

Code version control, also known as version control system (VCS), is a software tool used by developers to manage changes to source code over time. It is a critical component of software development, enabling teams of developers to collaborate on code and track changes made by different team members over time [11].

With version control, changes to code are tracked by creating a "version" or "commit" of the code every time a change is made. These versions are stored in a central repository, which can be accessed by all members of the development team. Version control systems also allow developers to track who made changes to the code and when they were made, making it easier to identify and resolve conflicts that may arise during development.

Code version control helps development teams work more efficiently by enabling multiple developers to work on the same codebase at the same time. It also provides a safety net for code changes, allowing developers to easily roll back to previous versions of code if necessary.

Some popular version control systems include Git, SVN, and Mercurial. These systems are widely used in open-source and enterprise software development, and many cloud-based code hosting platforms, such as GitHub and Bitbucket, offer version control tools as part of their platform.

2.9 Helm

Helm is an open-source package manager for Kubernetes that simplifies the process of deploying and managing applications on a Kubernetes cluster. It enables developers and system administrators to define, install, and upgrade complex Kubernetes applications using simple configuration files called charts [12].

Charts are packages of pre-configured Kubernetes resources such as deployments, services, and ingress rules that can be easily installed and configured using Helm. Charts are versioned, and their source code is maintained in a Git repository, making it easy to track changes and manage updates.

Helm also provides a templating system that allows developers to parameterize their Kubernetes resources and generate dynamic configurations for each installation. This can be useful when deploying applications to multiple environments, such as development, staging, and production, with different configurations for each environment.

2.10 Software Framework

A software framework is a set of pre-written and reusable code libraries that provides a structure for developers to build applications [13]. It is a collection of tools, libraries, and conventions that work together to simplify the development process and allow developers to focus on the application's specific functionality.

Frameworks provide a set of building blocks for developers to use, making it easier to create software applications by providing pre-built functionality. For example, a web application framework like Flask or Node.js provides tools to handle requests from clients, manage sessions, and render dynamic HTML pages.

Frameworks are designed to be flexible and modular, allowing developers to choose which components to use based on their specific needs. This modular approach also allows for easier testing and debugging since the code is organized into smaller, more manageable units.

Some popular web application frameworks include Ruby on Rails, Django, Flask, and Node.js. These frameworks are widely used in web development, providing developers with powerful tools and structures to build web applications efficiently.

Chapter 3. Design specifications and constraints

The project will be deployed in a Kubernetes environment equivalent to that which may exist in an Azure infrastructure: A GitHub repository to upload the developers' code; an AKS (Azure Kubernetes Service) in charge of orchestrating the containerisation of the tools used, and be reachable by the public Internet.

The tools to be used for the implementation of the project will be the following and the latest available version of these will be used:

- Azure Devops: responsible for automating the build, testing and deployment phases.
- Sonarqube: application capable of scanning the code and assessing its quality.
- Azure Container Registry: repository of container images created by developers.
- Grafana: dashboard for the representation of relevant metrics for developers.
- Prometheus: in charge of collecting logs and sending them to Grafana for representation.

For the Blog App we will create it with the Python Flask code framework. It is a blog in which users upload posts on different topics. It will have a database to store the posts and the different users. The application will have user-password authentication and password management. A version without user management will be made beforehand, and a later version with the management implemented to demonstrate the methodology. This app has been heavily improved from a public code repository with a python flask blog [14].

As for the JavaScript App we will deploy an interactive calculator with NodeJS hosting the service. The source code of the app to use as example is in this repository [15].

Both examples can show the universal use of our platform and automation approach regardless of the technologies that the developers use.

Chapter 4. Description of the proposed solution

4.1 Environment Explanation

The aim of the project was initially to simulate a Devops environment that could be used by several development teams within an organisation. This was initially planned in a Cloud environment with the AWS public cloud, but the cost of maintaining the infrastructure was too high. Even with the resources off, there were costs with just adding the resource pool.

Therefore, it was decided to try to simulate such an environment with virtualisation running on the student's computer. However, due to the high resource requirements it was not possible to carry out the simulation. Some micro services were able to be virtualized, but not all of them at the same time.

Finally, it was decided to use an Azure for Students free account [16], which offers a wide range of free, although limited, services. In addition, to showcase one of the strengths of Azure Cloud, it has been decided to use as many of the available services as possible, demonstrating their unification and saving costs and resources. The Azure for Students provided the student an Azure Subscription with 100$ to use on their cloud. But there are some restrictions, for example, the number of public IPs, the number of Virtual Machines CPU cores to reserve or Networking services with extra costs or unavailable. For that reason, in the following section it will be explained the tools used and their restrictions for this environment.

4.2 Devops tools used

Within the different phases that make up the development lifecycle of a project applying DevOps philosophy, different tools can be recognised that are used in each of them. It is possible to find that the same tool is used in different phases of this life cycle, given that they may have functionalities that allow this. These are the different tools used for this project:

4.2.1 Azure Devops

Azure DevOps is a comprehensive suite of collaborative software development tools provided by Microsoft as part of the Azure cloud computing services. Azure DevOps is designed to

enable software development teams to plan, build, test, and deploy software faster and with higher quality.

With Azure DevOps, software development teams can streamline their development processes by using a single platform to manage all aspects of software development, from ideation to deployment. This allows teams to collaborate more effectively, track progress more easily, and deliver high-quality software with greater speed and efficiency.

It was used the Azure Devops Board to list the tasks to implement in the Devops project and also simulate an Agile approach. Also Azure Devops Test is used to report the testing results related to the Deploy phase of the Devops Life Cycle.

The key tool of this suite is Azure Pipelines. **Azure Pipelines** is a cloud-based continuous integration and continuous delivery (CI/CD) service provided by Microsoft Azure. It allows developers to automate the build, testing, and deployment of their applications across multiple platforms and environments. With Azure Pipelines, developers can create and manage pipelines to build, test, and deploy their code, and also benefit from a wide range of features such as support for multiple languages and platforms, integration with popular development tools and services, and advanced automation capabilities.

In this project it is used a free tier cloud-based service. This means Azure Devops put at our disposal a VM with a wide but limited range of pipeline run minutes, setting the limit in 1800 min/per month [17]. However, is more than enough for this project.

4.2.2 Azure Kubernetes Service (AKS)

AKS stands for Azure Kubernetes Service, which is a fully managed container orchestration service provided by Microsoft Azure. It simplifies the process of deploying, scaling, and managing containerized applications on a Kubernetes cluster. AKS offers features such as automated updates, self-healing, horizontal scaling, and integrated security. With AKS, developers can focus on building and deploying applications while the underlying infrastructure is managed by Azure [18].

The solution of this project consist of a limited version available for users of Azure for Students but replicable in a real production environment. Some characteristics of this Kubernetes cluster are:

- Kubernetes version is 1.23.12 even though there is a higher version (1.24) to allow Azure Devops connectivity. There is not a correct working integration with Azure Devops and AKS at the moment, the reason is that Kubernetes changes how to handle technical users authentication in the higher version [19]. The Azure Devops platform didn't implemented this change, so the cluster couldn't authenticate the technical user of Azure Devops when deploying or doing tasks on the cluster.

- The cluster is created with 2 nodes (infrastructure Virtual Machines of Kubernetes) that are torn down and rebuilt each time the cluster is stopped and restarted. The size of each VM is 2 CPUs, 8 GiB of RAM and 16 GiB of temporal storage.

- The cluster has a domain and public IP that make it accessible from the public Internet. The cluster has loadbalancing to redirect the incoming requests and using the public IP as entrypoint to our applications. The service is limited to use HTTP instead of HTTPS given that the certificates of the applications are not trustworthy for the rest of Internet. If needed, there is another public IP available to use from the Azure Subscription.

- The access to the cluster is managed by the Kubernetes platform, it is not controlled by Azure user management. Giving more freedom and flexibility but less security due to the fact that Azure Active Directory have more security capabilities that the Role-Based Access Control of the cluster. For example, an administrator of the cluster can create the user and grant him permissions regardless of the access Azure manages. While using Active Directory, the user must originate from that service and manage the permissions in the cluster by that service, rather than locally on the cluster.

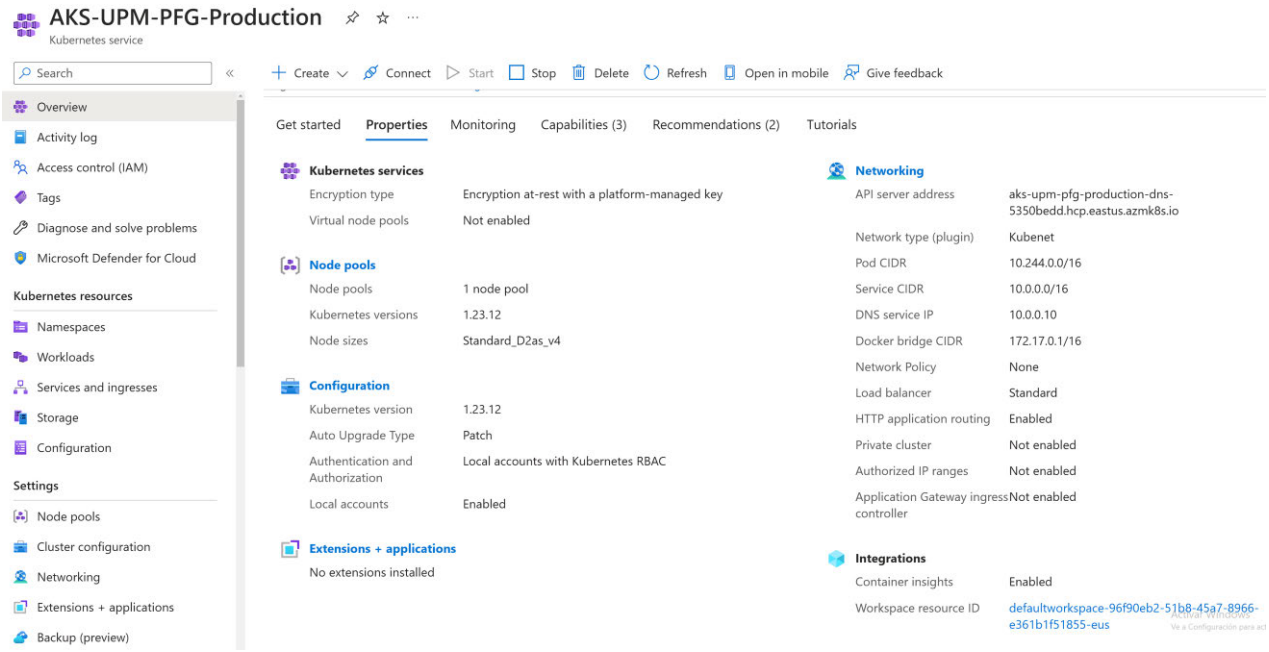

Figure 4. AKS Overview

4.2.3 GitHub

GitHub is a web-based platform that provides a collaborative environment for software developers to work on code projects. It allows developers to host, share, and review code, as well as track and manage changes to code over time using version control tools such as Git.

In this project this VCS was chosen because it is compatible with Azure Devops, Visual Studio (IDE for the developers) and CI/CD capabilities. For each application there is code repository and one additional for the Microservices configuration.

4.2.4 Sonarqube Community Edition

SonarQube Community Edition is an open-source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities in a wide range of programming languages. It allows developers to create quality reports and track the evolution of code quality over time. The Community Edition of SonarQube is free to use and is primarily designed for small to medium-sized teams.

For this project the platform is used with Azure Devops making use of and Azure Devops Extension available in their Marketplace. The SonarQube is deployed on the Kubernetes cluster and configured with Helm.

4.2.5 Azure Monitor for Prometheus

Azure Monitor is a comprehensive monitoring and analytics platform offered by Microsoft for cloud applications and infrastructure. It allows users to monitor the performance and availability of their applications and infrastructure, as well as identify and diagnose issues through metrics, logs, and other telemetry data.

Azure Monitor also includes a managed service for Prometheus, which enables users to scrape Prometheus metrics from their Kubernetes clusters running in Azure Kubernetes Service (AKS) or other cloud environments. With this service, users can collect and visualize their Prometheus metrics data in Azure Monitor, providing a unified monitoring experience across their entire application stack.

The Azure Monitor managed service for Prometheus supports the standard Prometheus query language and allows users to use a variety of built-in monitoring solutions for analysis and alerting, such as log analytics, metric alerts, and dashboards. Users can also integrate their Prometheus metrics with other Azure services and tools, such as Azure Functions, Azure Stream Analytics, and Power BI.

For that reason, the Azure Monitor managed service for Prometheus provides users with a scalable and reliable solution for monitoring their Kubernetes clusters and applications, with the added benefits of integration with the Azure ecosystem and support from Microsoft.

In this project we make use of this platform instead of the another most common solution of deploying Prometheus inside the Kubernetes Cluster. Saving resources from the cluster and having and easier management with the same capabilities and configuration.

4.2.6 Azure Managed Grafana

Azure Managed Grafana is a fully managed service offered by Microsoft Azure that allows users to create, explore, and share interactive dashboards based on various data sources, such as Prometheus, Azure Monitor, and Application Insights. With Azure Managed Grafana, users can easily monitor and analyse data from different sources in a centralized location.

Azure Managed Grafana offers various features, such as built-in Azure Active Directory integration, automatic updates and scaling, and support for plugins and extensions. It also offers enhanced security features, such as role-based access control and encryption at rest for data storage.

Azure Managed Grafana can be used for various use cases, including infrastructure monitoring, application performance monitoring, and business intelligence. It provides a user-friendly interface that allows users to create and customize their own dashboards and visualizations, making it a powerful tool for data analysis and decision-making.

This service has a cost, but can be implemented with our Azure Monitor managed service for Prometheus. This service could also have been deployed on the AKS, but this option is preferable as it allows for efficient resource utilization from the cluster and use Azure framework.

4.3 Technologies used to develop the applications

This section describes the technologies used to develop the applications that have been used as examples on the Devops platform.

4.3.1 NodeJS

Node.js is a free, open-source, cross-platform, back-end JavaScript runtime environment that runs on the Chrome V8 engine. It enables developers to run JavaScript on the server-side and build scalable network applications with ease. Node.js provides an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for building real-time applications that require a large amount of data to be processed and transmitted over the network. Node.js has a large and active community, and it has become one of the most popular and widely-used technologies in web development.

This software framework is used to build and host the calculator app of this project.

4.3.2 Flask

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has a small core and provides a range of extensions that add functionality to the application. Flask is mainly used for building web applications, APIs, and Microservices.

In our case, Flask is used to build and host the blog app written in Python.

4.3.3 MySQL

MySQL is a popular open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for managing and manipulating data. It is widely used in web development for managing data in web applications and is known for its scalability, security, and ease of use. MySQL also offers a range of tools and APIs for developers to integrate and interact with the database. It is often used in conjunction with programming languages like PHP, Python, and Java to build robust web applications.

For this project MySQL is used as Database for our blog app, to store the information, like users or posts written by them.

4.4 Devops Platform Design

Having outlined the concepts and technologies specific to this project, we will now describe the proposed design. There will be three sections, one explaining the design of the Devops platform, another one for the calculator app and finally, one for the blog app.

As mentioned above, the aim of this project is to show the capabilities and particularities of the Devops model in a Cloud environment. For that reason, the implementation between Azure services has been used as much as possible.

As shown in Figure 5, this would be the design and CI/CD chain used. The steps followed on the figure are the following ones:

1. A push to the master branch on the GitHub repository triggers the Azure Devops Pipeline. This pipeline runs on the Azure Devops environment. If any of the steps of the pipeline fails, the pipeline does not continue.

2. Then the CI part of the pipeline starts on the creation and publishing of a container image in the Azure Container Repository and continues to the test stage. In an environment with more developers, this phase would also merge the code from the branches that had a Pull Request accepted.

3. In this CI part of the pipeline the code is analysed with the Sonarqube service running on the AKS and tested with Unit tests. If the results of the tests are correct and passes the Quality Gate set on SonarQube, then the flow continues. The QA team can also check the results on the Azure Devops Test platform.

4. The completion of the CI stage triggers the CD stage.

5. The deployment of the app on AKS requires secrets, so this pipeline gets those secrets from Azure Key Vault.

6. The CD pipeline automatically deploys the Kubernetes objects with the configuration of the app to the AKS environment. The QA team can then perform acceptance tests against the environment to validate the deployment.

7. The automated deployment specifies the container image from the Azure Registry built on the previous stage of the pipeline.

8. Container Insights forwards performance metrics, inventory data, and health state information from the containers to Azure Monitor for Prometheus.

9. Azure Monitor collects observability data such as logs and metrics so that an operator can analyse health, performance, and usage data. Application Insights collects all application-specific monitoring data, such as traces. Azure Log Analytics is used to store all that data. Azure Managed Grafana helps to survey this data with useful dashboards.

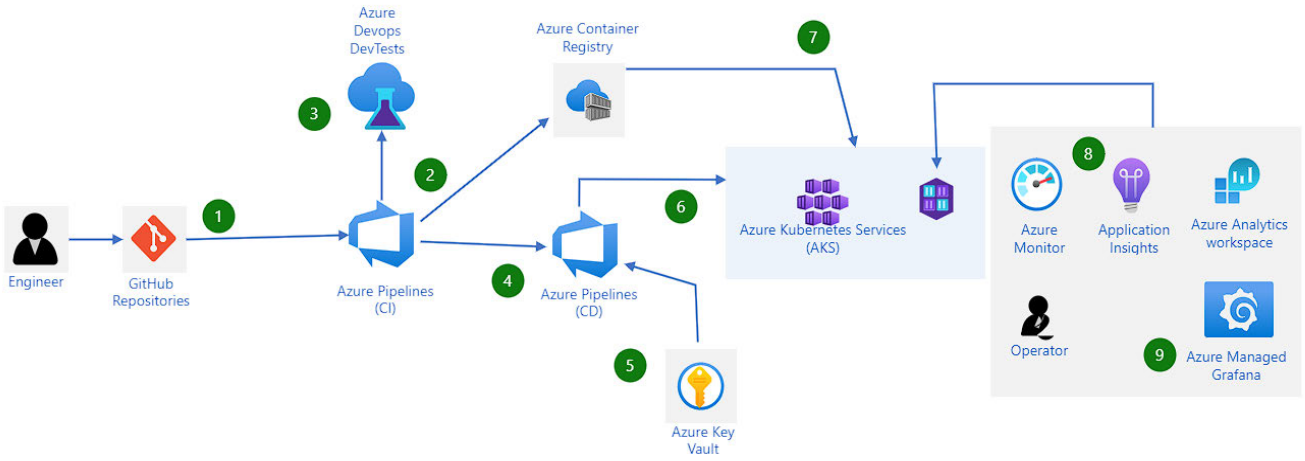

Figure 5. CI/CD Chain

4.5 Kubernetes Microservice Architecture Design

A Microservice architecture is a method of designing software applications as a collection of small, independent services. Each service is self-contained, loosely coupled, and can be developed, deployed, and scaled independently. When it comes to deploying microservices on Azure Kubernetes Service (AKS), it requires a set of specific steps to ensure the services are communicating properly, and the system is secure and stable.

To expose the services to the outside world, AKS uses an **Ingress Controller**. The ingress controller acts as a reverse proxy and exposes HTTP routes from outside the cluster to services within the cluster. This means that external requests to the services can be routed through the ingress controller. The ingress controller can also be used to apply security and routing policies, enabling secure communication between the external clients and the services. External clients can route to the cluster thanks to the Azure DNS service that hosts the domain of the services running on the cluster.

Another aspect of AKS is **Namespaces**. A namespace is a logical boundary within a Kubernetes cluster that allows developers to segregate resources and services. This helps to avoid naming conflicts and resource contention between services. The applications deployed on the AKS have their own namespace with the different Kubernetes components necessary for make them work.

Secrets are Kubernetes objects that store confidential information, like passwords or usernames. These secrets are created and deployed on the namespace during the Pipeline Phase from the information stored on the Azure Key Vault service.

Also the applications need storage, so the **PVC (Persistence Volume Claim)** is another object to configure how the data of the applications are stored and accessed. In this case, Azure Disk is used as solution for our applications.

Another component are **Pods**. A Pod is a logical host for one or more containers that run together on a node. Each Pod has its own unique IP address within the Kubernetes cluster and can be managed, scheduled, and scaled independently. Pods are not meant to be long-lived entities, and they are designed to be created, run, and destroyed as needed. When a Pod is destroyed, all the containers within the Pod are also terminated. To maintain the desired state of the application, Kubernetes uses a declarative approach to manage the Pods, which means that the desired state of the application is defined in the Pod specification, and Kubernetes ensures that the current state matches the desired state. Fulfilling one of the requirements of DevOps methodology to maintain the infrastructure as code.

A **ConfigMap** in Kubernetes is an API object used to store configuration data in key-value pairs. It provides a way to separate configuration from application code, allowing you to manage and modify configuration independently of the application itself.

A ConfigMap can be created using various sources such as literal values, environment variables, or files. It can be used to store configuration settings, command-line arguments, environment variables, or any other configuration data required by the application.

Once a ConfigMap is created, it can be mounted as a volume or injected as environment variables into a pod, making the configuration data accessible to the application running within the pod. This allows the application to read and use the configuration values during runtime.

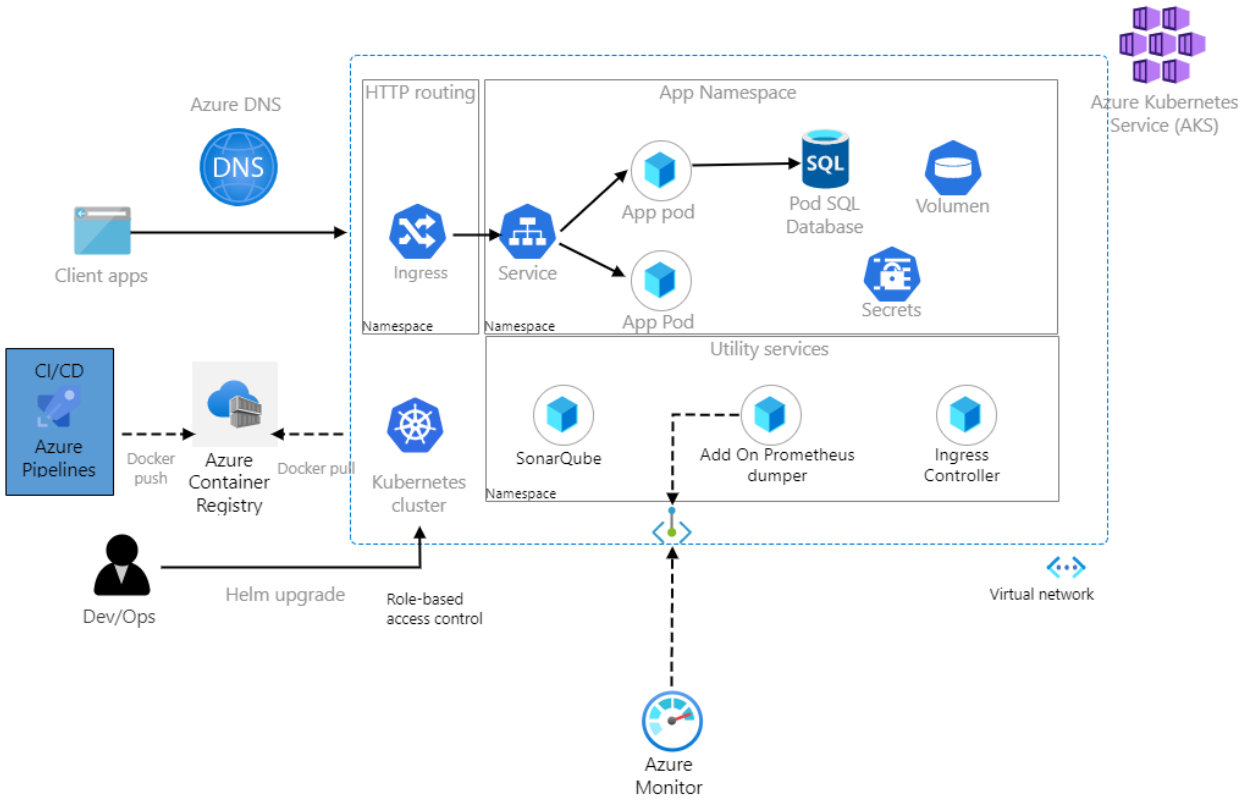

Figure 6. Microservices Architecture on AKS

4.5.1 Kubernetes Services

Kubernetes Services is an abstraction layer in Kubernetes that allows decoupling of microservices or applications from the underlying network. Services provide a consistent way to access and communicate with a group of pods, abstracting the internal network details of the pods.

Kubernetes Services allow pods to be discovered by their labels. By assigning labels to a pod, Services can then select and forward traffic to those pods. This makes it possible to load balance traffic across multiple pods and provide a stable endpoint for communication with a group of pods.

Kubernetes Services can also be used to communicate between different namespaces within a cluster. By using the Service DNS name, other pods can communicate with a Service in a different namespace as if it were in the same namespace. This enables decoupling of applications and microservices, and allows for better isolation and scalability of different parts of an application.

The following diagram shows the conceptual relation between services and pods. The actual mapping to endpoint IP addresses and ports is done by kube-proxy, the Kubernetes network proxy.

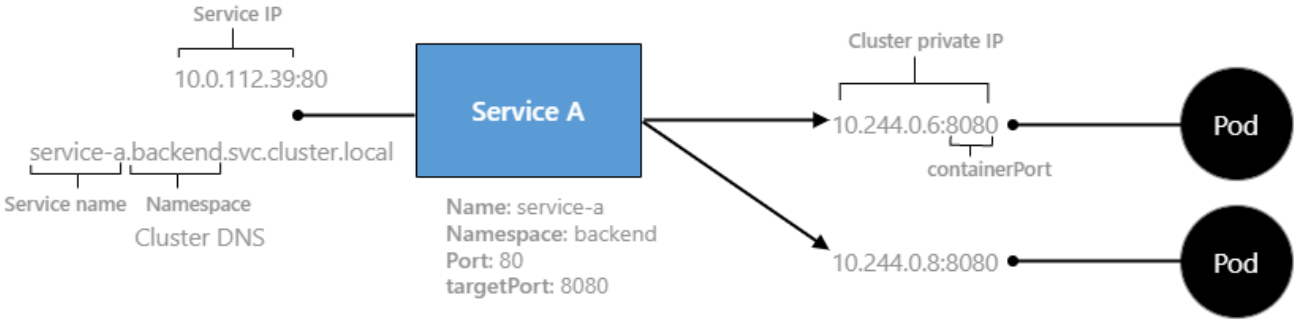

Figure 7. Kubernetes Service Concept

4.5.2 Kubernetes Deployment

A Kubernetes Deployment is an object in Kubernetes that defines the desired state and specifications for running a set of identical pods. It is a higher-level abstraction that allows you to manage and control the lifecycle of your application containers.

A Deployment ensures that a specified number of pod replicas are running and maintains the desired state by continuously monitoring the health of the pods. If a pod fails or is terminated, the Deployment automatically creates a new pod to replace it, ensuring that the desired number of replicas is always maintained.

Key features and components of a Kubernetes Deployment include:

- Pod Template: A Deployment specifies a template for creating pods. This template defines the container image, environment variables, volume mounts, and other configurations for the pods.

- ReplicaSet: The Deployment uses a ReplicaSet to manage the actual pod replicas. The ReplicaSet ensures that the desired number of replicas are running at all times and performs scaling operations when needed.

- Rolling Updates: Deployments support rolling updates, allowing you to update your application without downtime. During a rolling update, the Deployment creates new pods with the updated configuration and gradually replaces the old pods, ensuring a smooth transition.

- Rollback: If an update or new version of your application introduces issues or failures, Deployments provide a rollback mechanism to revert to a previous stable version. This allows you to quickly recover from errors and maintain application availability.

- Scaling: Deployments allow you to scale your application horizontally by adjusting the number of replicas. You can manually scale up or down based on resource requirements or set up auto-scaling rules to dynamically adjust the number of replicas based on metrics such as CPU utilization.

By using Deployments, you can easily manage the lifecycle of your application, ensure high availability, and perform seamless updates and rollbacks in a Kubernetes cluster.

For our project it is used Kubernetes Deployments to manage the pods that run the Calculator App and the frontend of the Python Blog App.

4.5.3 Kubernetes StatefulSet

A StatefulSet is a Kubernetes object used to manage the deployment and scaling of stateful applications in a cluster. It provides guarantees about the ordering and uniqueness of pods and their corresponding network identities.

Unlike a Deployment, which manages stateless applications, a StatefulSet manages stateful applications that require stable network identities and persistent storage. Some examples of stateful applications include databases, key-value stores, and messaging systems.

Key features and components of a StatefulSet include:

- Ordered Pod Creation: StatefulSets create pods in a predictable and ordered manner. Each pod is assigned a unique ordinal index, starting from 0, which is reflected in the pod's hostname and network identity. This ordering ensures that pods are created and scaled in a controlled sequence.

- Stable Network Identities: StatefulSets provide stable network identities for pods. Each pod in a StatefulSet has a stable hostname based on the pod's ordinal index. This enables applications to rely on consistent network addresses, making it easier to communicate with specific pods.

- Persistent Storage: Each pod in a StatefulSet can have its own dedicated volume, allowing data to be retained even if the pod is rescheduled or deleted.

As showed above it is recommended to use StatefulSet to manage databases. We use a StatefulSet object to manage the pods that hosts the Python Blog app MySQL database.

4.6 JavaScript Calculator App

The application consists of a server that listens for incoming HTTP requests and responds with the result of the requested operation.

The backend of the application is built using Node.js and Express, which provides the API for the frontend to communicate with. The frontend of the application is built using React and displays the user interface of the calculator.

The application is divided into two main components: the frontend and the backend. The frontend is responsible for displaying the calculator user interface and sending requests to the backend API to perform calculations. The backend is responsible for handling the API requests, performing the calculations, and sending the results back to the frontend.

4.6.1 The Frontend

The index.html file is the front-end code for a pocket calculator that makes requests to the API for performing arithmetic operations.

The script in the HTML file defines several variables and functions for managing the calculator's behaviour. The calculator uses a state machine approach to handle user input. There are five possible states: start, operand1, operator, operand2, and complete. The state is set and updated based on the user's input.

The calculator sends requests to the API using XMLHttpRequest and handles the response to display the result or error messages. The calculator's display format is customized to show numbers in exponential notation when they are too large or too small to fit in the display. The HTML is dynamically updated to show the calculated result.

The calculator also has features for clearing the current entry, clearing the entire calculation, and calculating the inverse of the current number.

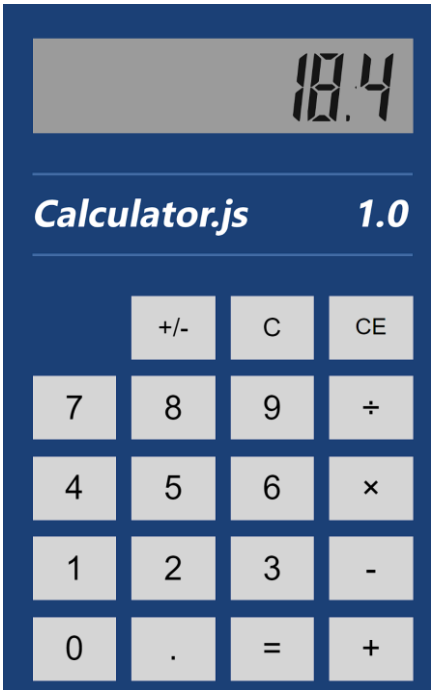

Figure 8. Calculator app

4.6.2 The backend

In the app it is set the route to '/arithmetic' endpoint that perform the arithmetic operation based on the parameters provided in the GET request. The route uses the 'calculate' function which handles errors and return the results as a JSON response.

4.6.3 Deployment

The application is deployed on the AKS cluster using Kubernetes manifests that define the **Deployment, Service,** and **Ingress** resources. The deployment resource specifies the number of pod replicas and the container image to use for the application, while the service resource creates a stable endpoint for accessing the application. The ingress resource defines the rules for routing incoming traffic to the appropriate service based on the requested URL.

4.7 Python Flask Blog App

To show the functionalities of the DevOps methodology, the application has been divided into 2 iterations, as in a normal development phase, with the first iteration having only the basic blog and the second iteration adding user authentication.

The app is a Flask web application for a blog with a MySQL database backend. The database backend is used to store user account information, blog posts, and comments.

The main functionality of the app includes:

- Registering and logging in as a user
- Creating, editing, and deleting blog posts
- Viewing and commenting on blog posts
- Editing user profile information
- The session is saved

The Python Flask app is deployed using a containerization approach. The deployment of the app is done through a series of YAML configuration files that describe the various components of the app and their deployment characteristics.

The main components of the app are the Flask web server and the MySQL database, which are deployed as separate containers. The app is designed to be scalable, with the ability to run multiple replicas of the Flask server and the MySQL database. The configuration files also include various environment variables that are used to configure the app, such as the database host and credentials, the app secret key, and the port on which the Flask server listens. Once the deployment configuration is applied, the app can be accessed through the public IP address or domain name of the Kubernetes cluster, using the specified port and URL paths.

The MySQL database is deployed on the AKS cluster using Kubernetes manifests that define the **StatefulSet, Service, ConfigMap** and **Persistence Volume Claim**. The tables that uses the app to store the posts and users are created once the pod is deployed with the ConfigMap mysql-config. Also the authentication credentials to configure on the database is stored on the Azure Key Vault, to do that, they are extracted in the azure pipeline and the pipeline creates the secrets to store that information. Then on the StatefulSet is where this configuration it is set.

```
containers:
      - name: flask-blog-mysql
        image: registryupmpfg.azurecr.io/flask-blog-mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: flask-blog-mysql-secrets
              key: db_root_password
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              key: username
              name: flask-blog-mysql-auth
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              key: password
              name: flask-blog-mysql-auth
        - name: 'MYSQL_DATABASE'
          value: flask_blog
        ports:
        - containerPort: 3306
          name: db-container
```

```
volumeMounts:
          - name: mysql-persistent-storage
            mountPath: /var/lib/mysql
          - name: mysql-config
            mountPath: /docker-entrypoint-initdb.d
      volumes:
        - name: mysql-persistent-storage
          persistentVolumeClaim:
            claimName: mysql-pv-claim
        - name: mysql-config
          configMap:
            name: mysql-config
```

Figure 9. MySQL Configuration

To deploy the app on the AKS it is needed to define the **Deployment, Service** and **Ingress**. For the Flask app container, this configuration is also set in the Deployment, to allow the connection between the database and the app. To check the app, the Figures 10, 11, 12 and 13 show the different pages:

When there is no user logged in:

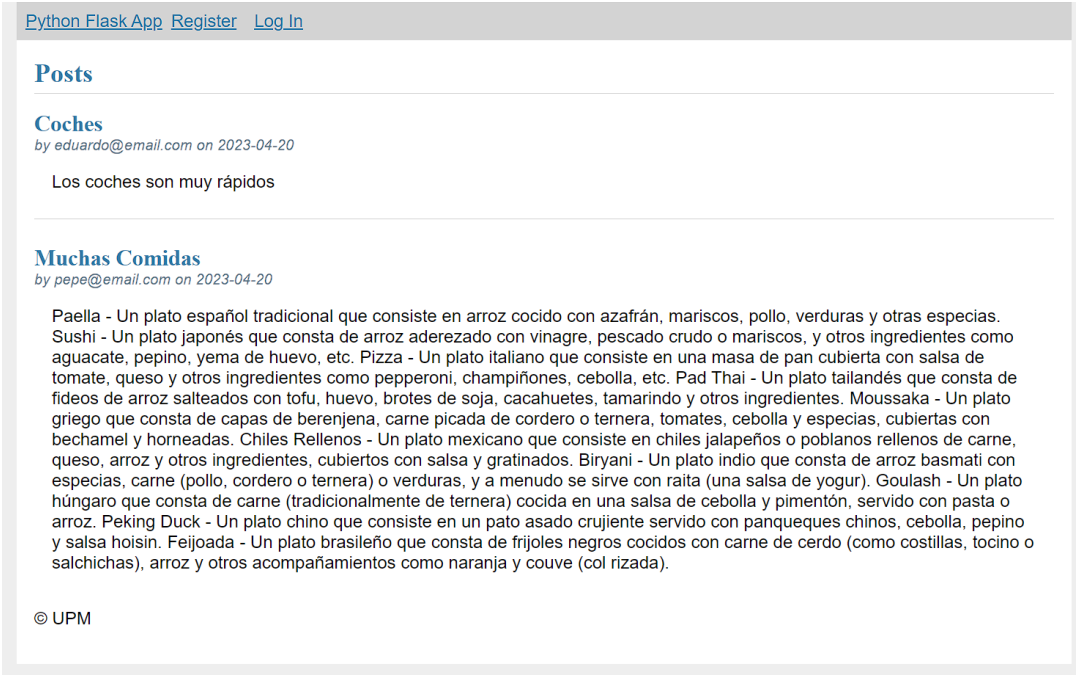

Figure 10. Python Blog App

The page to register:

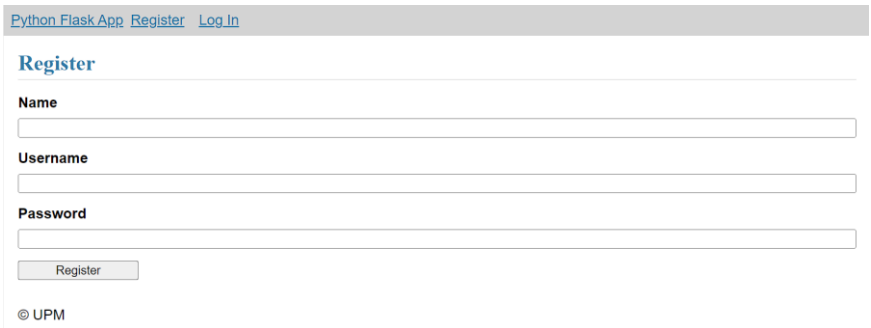

Figure 11. Python Blog Register view

When a user has logged in, he can edit the posts that he has created or delete them:

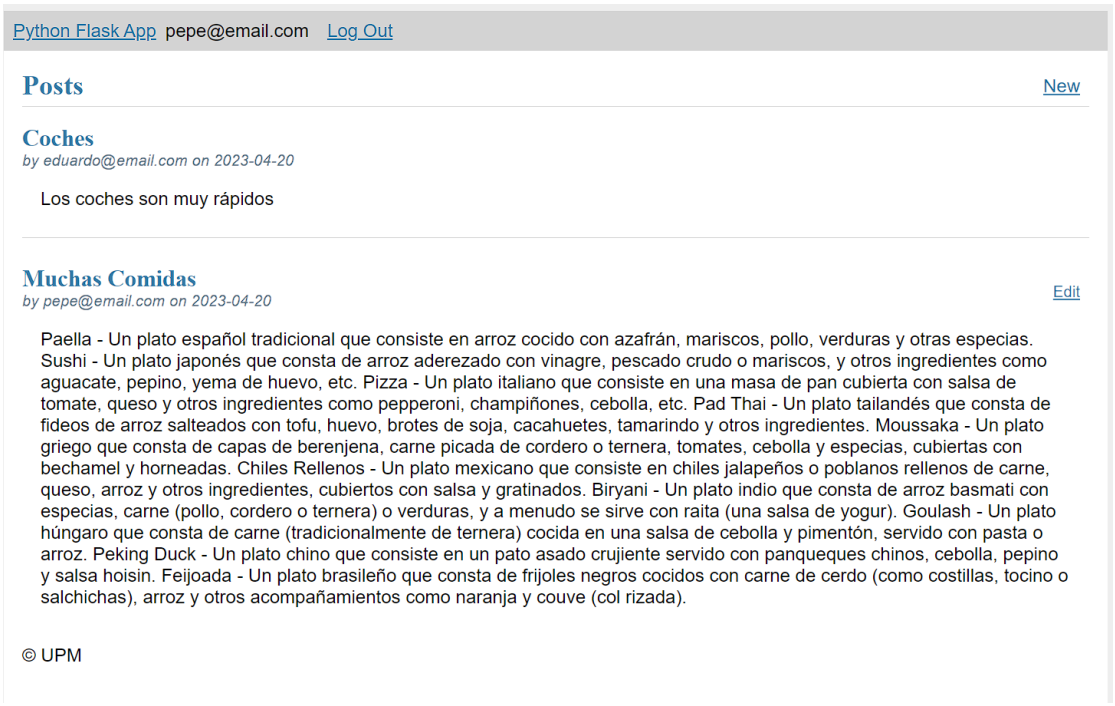

Figure 12. Python Blog App with User logged in

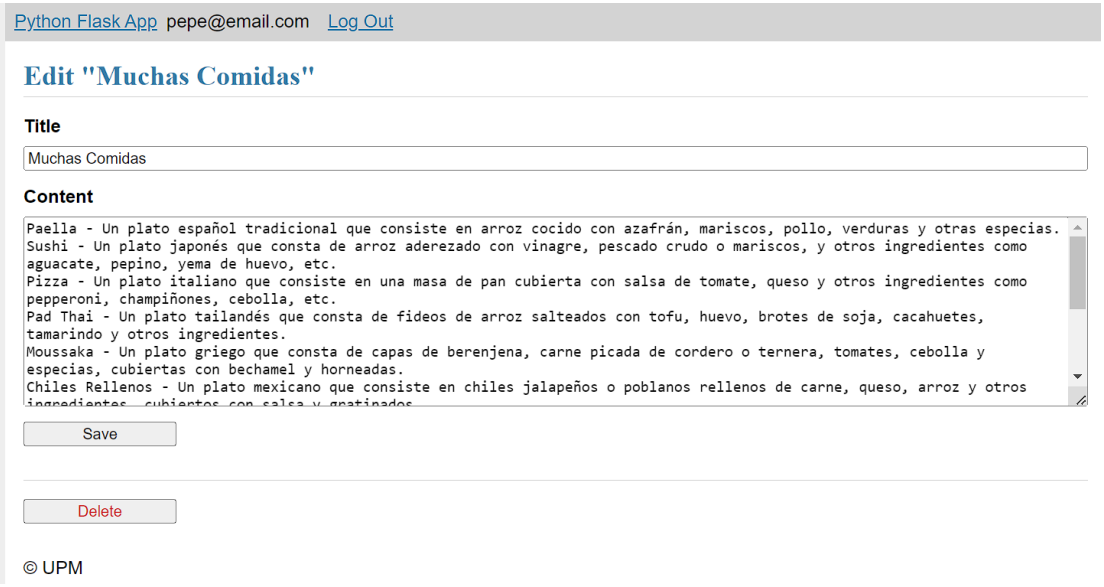

Figure 13. Python Blog App Editing Page

4.8 Azure Devops Pipeline

We have two main Pipeline that run the entire CI/CD phase for each app. They share in common that they are in the same Devops Project, they are connected with the different Azure Services explained previously and have similar structure to standardize as much as possible. The services connected to the pipelines are: Azure Container Registry, Azure Key Vault, the Azure Kubernetes Service and the SonarQube service running on it.

The pipeline configuration file defines the steps and stages required to build the application and deploy it to the AKS. It ensures that the application is built and deployed consistently whenever changes are made to the master branch of the code repository. The important aspects of the pipeline are the following:

- Trigger: the pipeline is triggered on every push to the repository's master branch.
- Variables: defines environment variables that can be used throughout the pipeline, such as the resource group name, container registry, and Kubernetes cluster details. These variables can be customized to match your Azure environment.
- Pool: specifies the pool of agents to be used for running the pipeline jobs. In our case a default Ubuntu VM.
- Jobs: defines the different stages and tasks of the pipeline. The pipeline consists of three stages: "Build", “Test” and “Deploy". Each stage has its own jobs (Figure 8) to run and are dependant from the previous one. For example, if the Test stage it is not successful, the Deploy stage won’t run, marking the pipeline run as failed.
- Pipeline Completion: the pipeline is marked as successful if all the defined jobs complete without errors.

To save space in the Azure Container Registry, a script was used that only keeps the last 10 images of each container. It is a script that launches commands with the Azure CLI client and deletes the leftovers. This runs on the Build stage of the pipeline.

When the container images are built, different Layers are created, each of them being a binary. To increase storage efficiency, Layers that have not changed are cached, making each image take up less space.

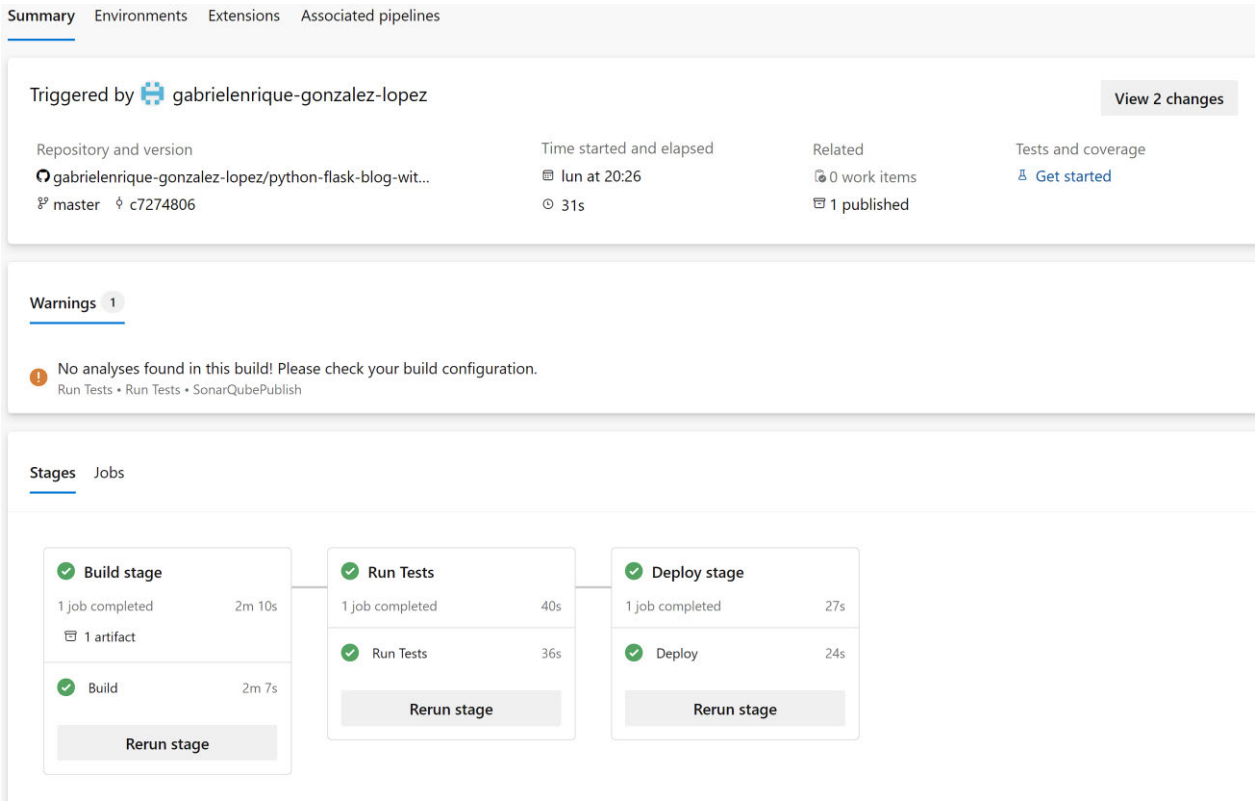

Figure 14. Azure Devops Pipeline Summary

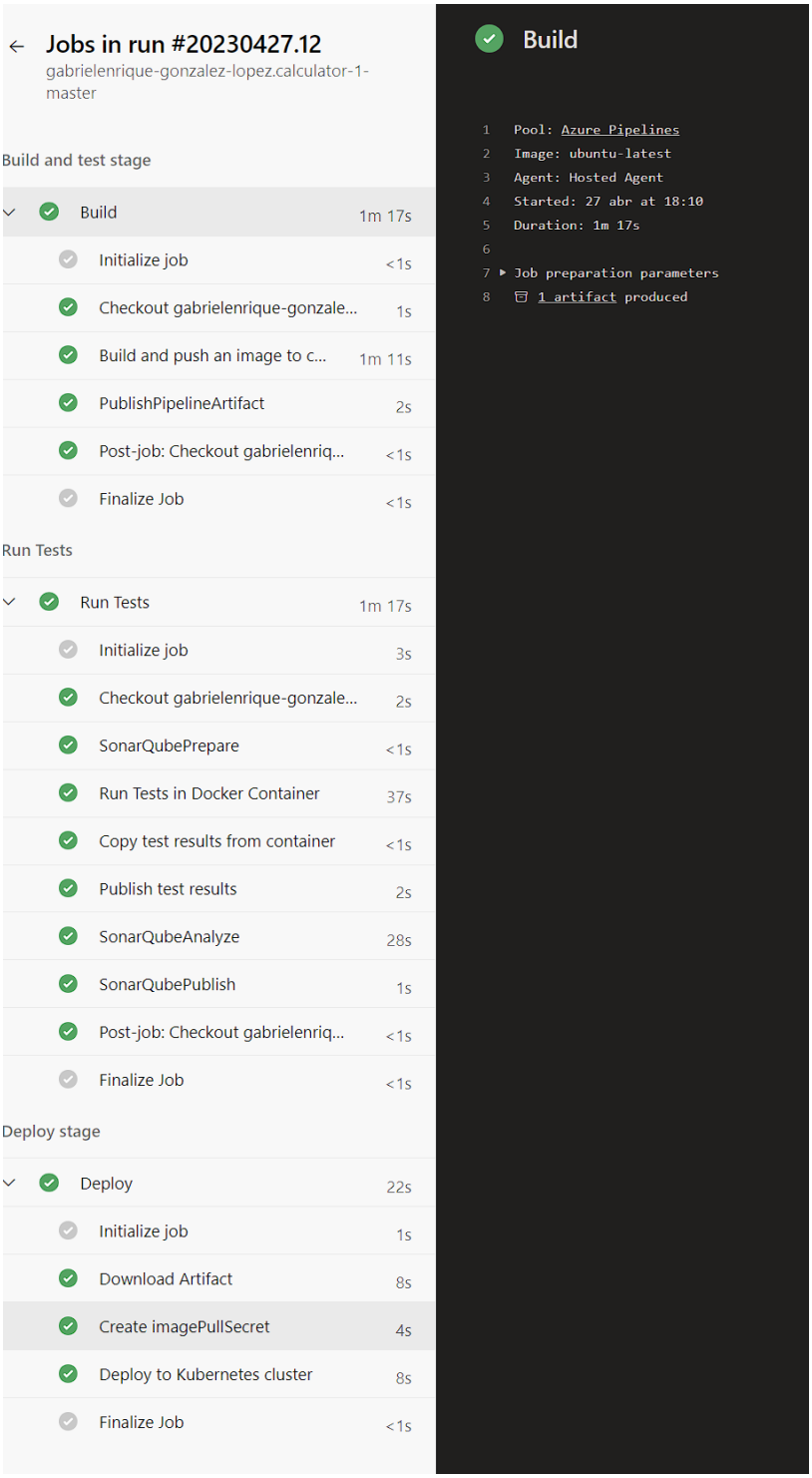

Figure 15. Azure Devops Pipeline Jobs

4.9 Testing Phase

For the testing and analysis phase, the code is studied in different ways. In the application code repository, the unit tests are added and configured. But depending on the application, the tests have to be run in different environments. Therefore, there are 3 specific tools that a QA team should access and use: Azure Devops, AKS and SonarQube.

4.9.1 Azure Devops Project

To avoid install dependencies on our Host VM running the pipeline, we run the tests once the container image is built. Then the image is downloaded from the Registry and launched the tests on the container running on the VM. The results of the tests can be checked on the Azure Devops project and download from the pipeline. This can also be linked with test tasks from the Azure Devops project section, allowing the QA team to report the results.

This is the case of the Calculator app that can be tested on the Host VM. The results are published as artefact on the test stage of the pipeline run.

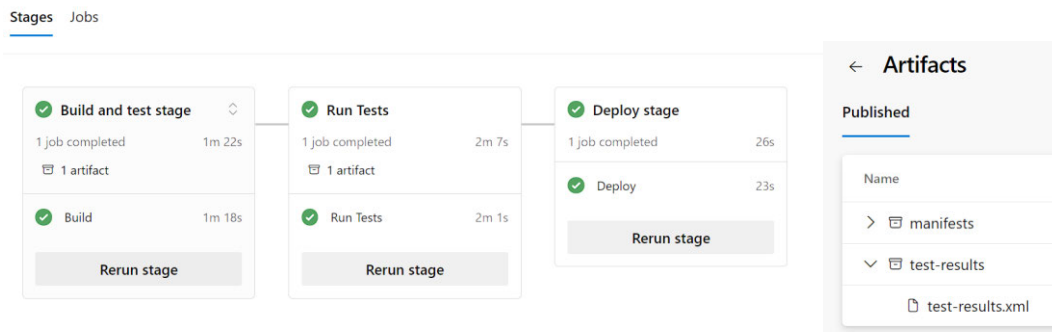

Figure 16. Test Artefacts on Azure Devops

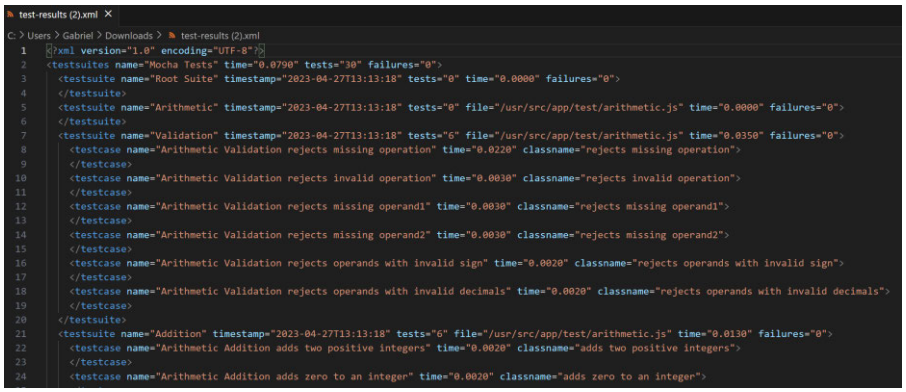

Figure 17. Test Results

This is example of a test case to refer the results of the tests:

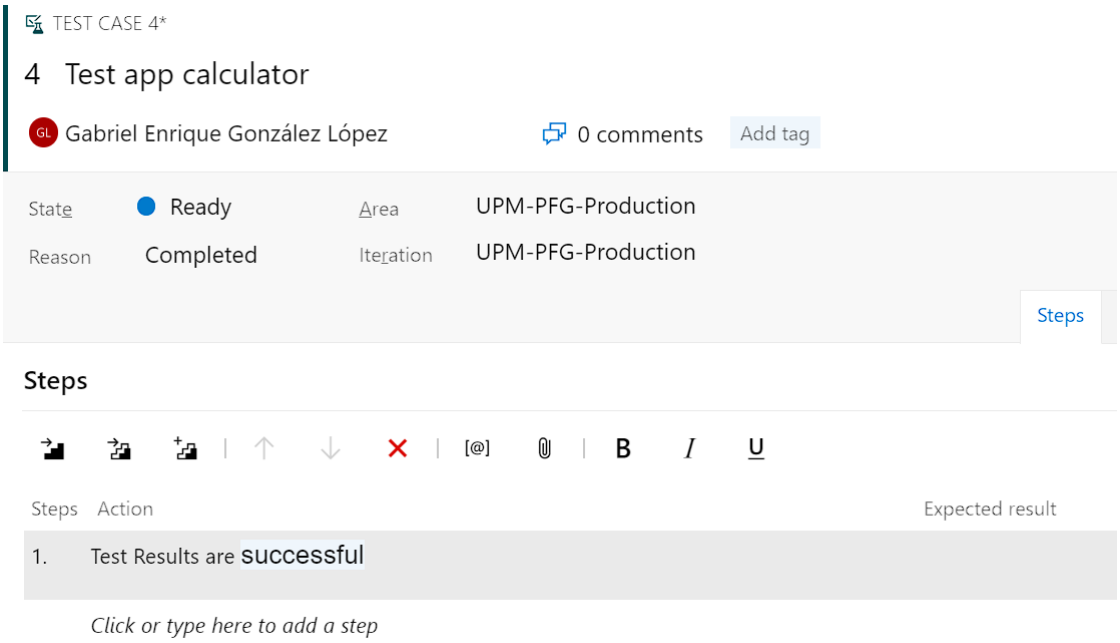

Figure 18. Test Case

And marking the tests as passed:

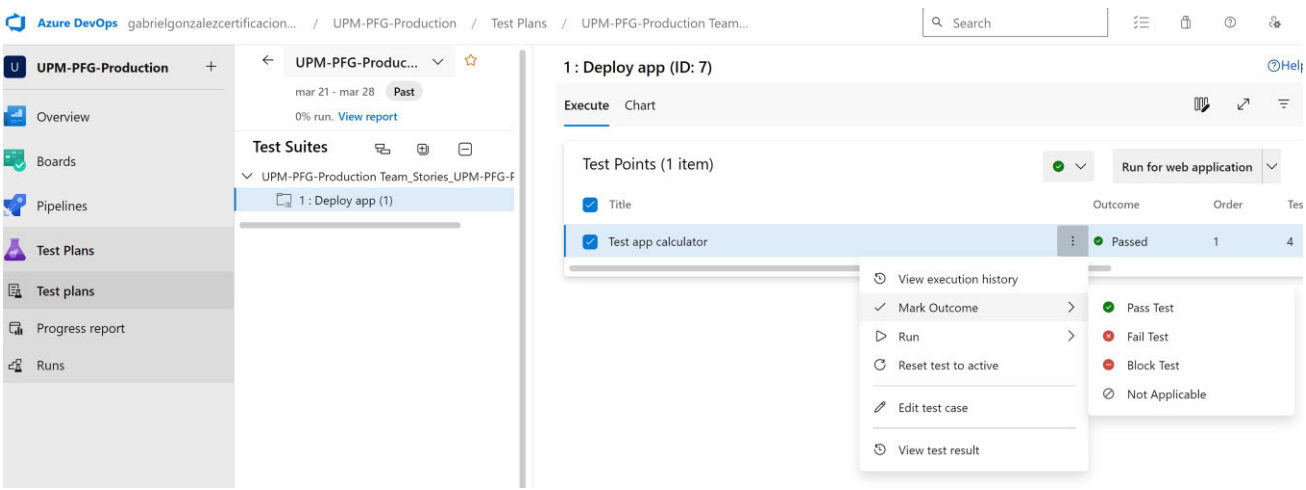

Figure 19. Mark Test as Passed

4.9.2 SonarQube Analysis

On the test stage of the azure pipeline, the code is analysed by SonarQube and generate a report with the results. We have two projects, one for the Calculator app and another one for the Python Blog app. We will use as example the calculator app project and show it report.

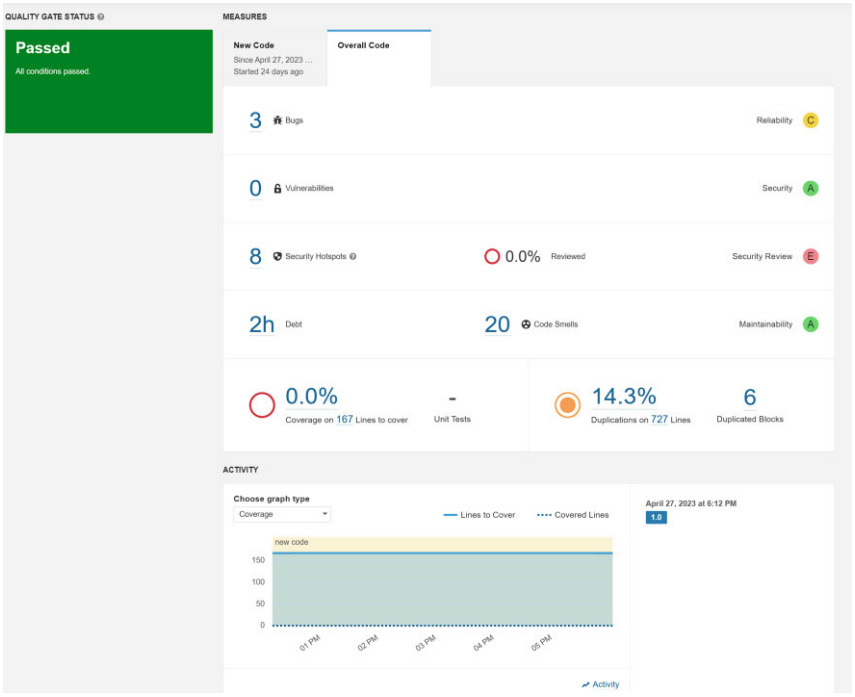

Figure 20. SonarQube project Overview

The following provides information and suggestions for our code. It also gives an estimate of how long it would take to fix these issues.

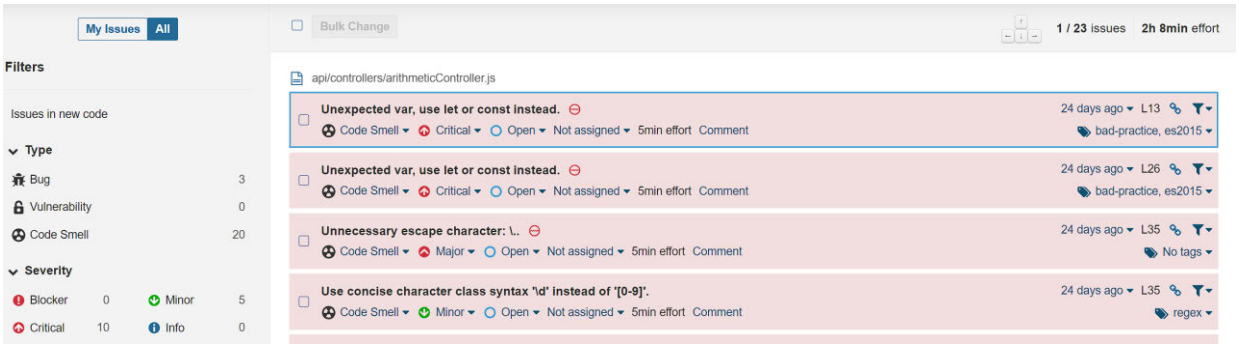

Figure 21. SonarQube Issues

When running the test stage on the pipeline there is a **SonarQube Quality Gate** setting if the code if enough well done to continue to be deployed. This Quality Gate is configured on the project page. In our case we use the default one, but it can be customized as the organisation needs.

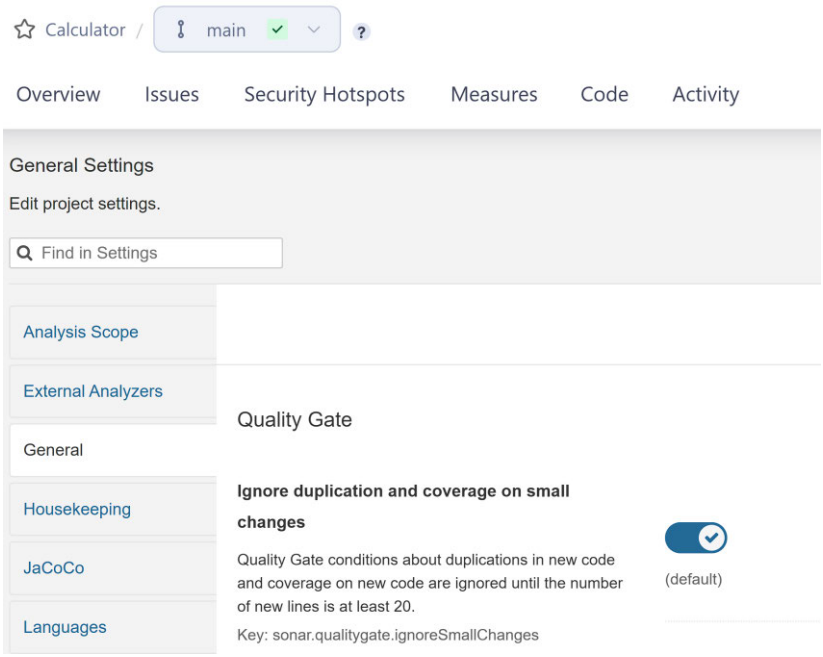

Figure 22. SonarQube Quality Gate

4.9.3 Executing the test on the AKS

For more complex applications, they cannot be tested running as containers on the Host VM. As it is the case of the Python Blog app, which is configured to be connected to the MySQL database, so simulating this scenario on the Host VM is unfeasible. That's why the test must be executed on the pods running the containers and extract the test with commands or automations done by the QA team.

4.10 Monitoring Phase

Azure Monitor managed service for Prometheus is a component of Azure Monitor Metrics, providing additional flexibility in the types of metric data that you can collect and analyse with Azure Monitor. Prometheus metrics share some features with platform and custom metrics, but use some different features to better support open source tools such as PromQL and Grafana.

The component is activated as an Add-on of the AKS and the feature manage the monitoring of the cluster. There are services running on the AKS that extract the data and is sent to the Azure Monitor workspace.

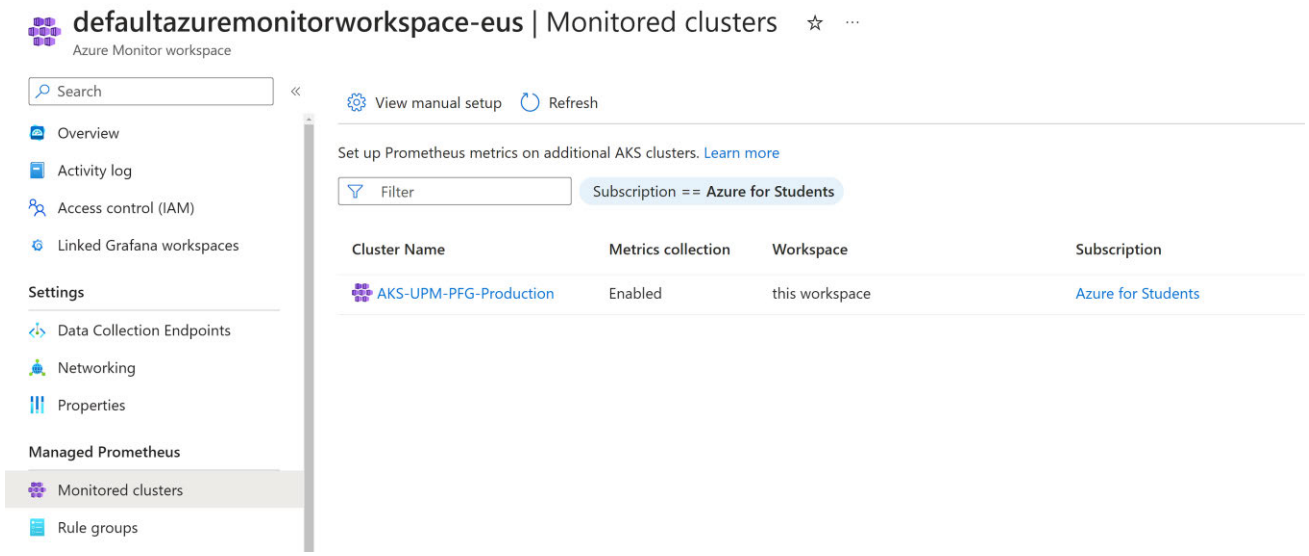

Figure 23. Azure Monitor Workspace

The rules to extract the monitored data (Figure 24) is configured with a file. This configuration file is passed to the feature as a Kubernetes ConfigMap object named container-azm-ms-aks-k8scluster. This file can be found on the helm values GitHub repository [20]. They can be modified also from this portal.

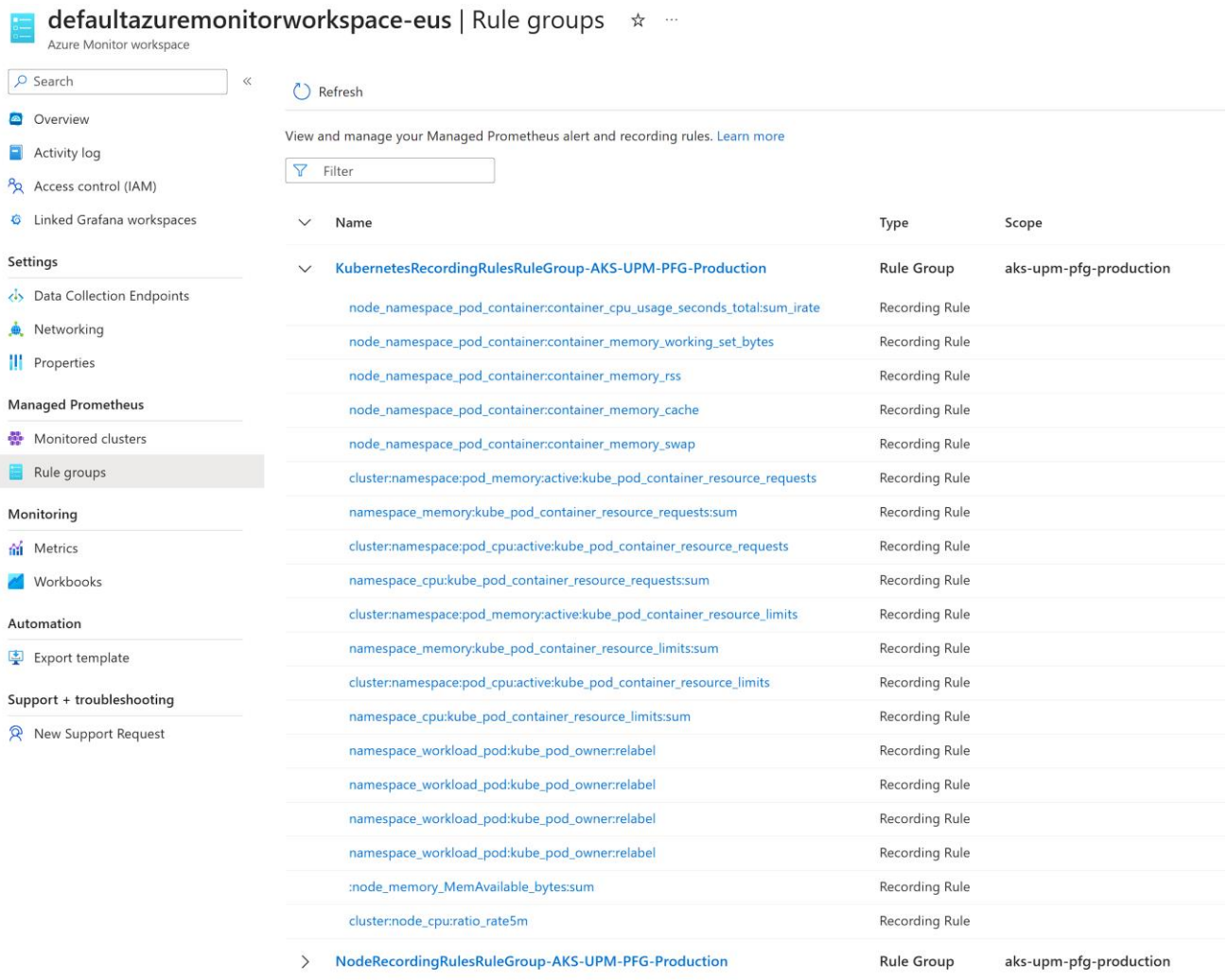

Figure 24. Prometheus Monitoring Rules

For debugging problems, we can utilize Azure Logs to monitor the reports from our apps deployed on the container, as well as the health of the cluster, pods, nodes or networking. We can execute Queries within the service and saved them for future use.

For example, the Figure 25 shows how Query helps us check any log that contains the string database:

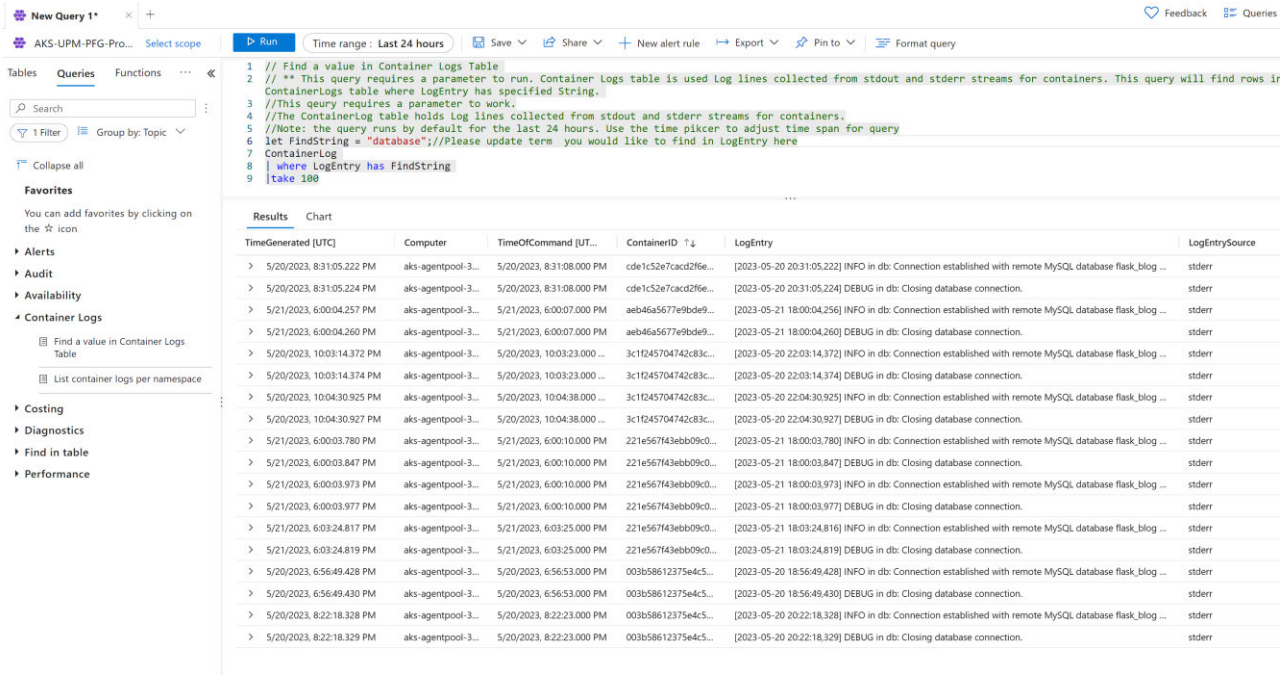

Figure 25. Azure Logs

There is data that can be visualized in the Insight section of the AKS page.

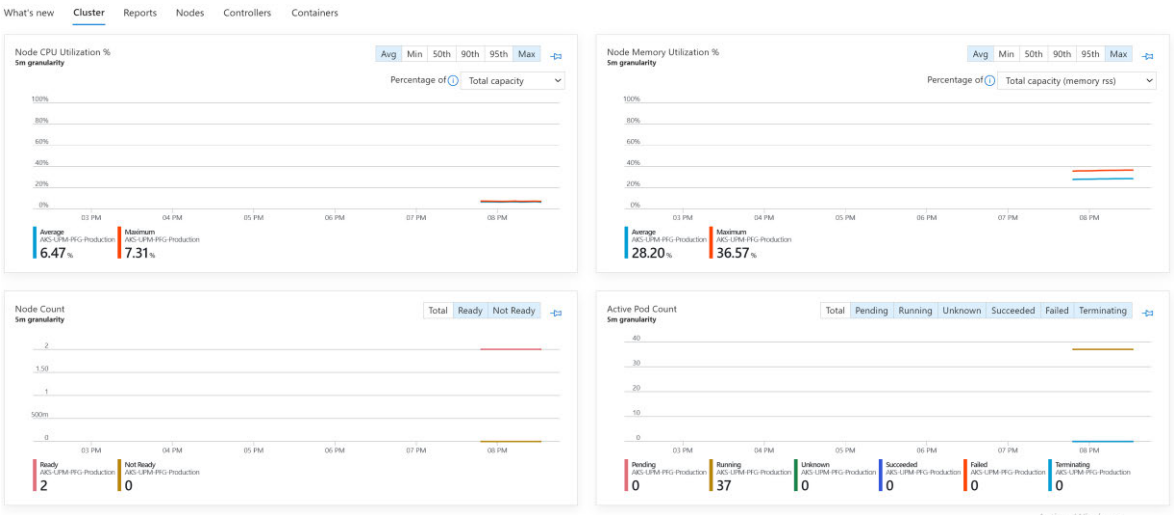

Figure 26. AKS Insights

For a more standardize environment we can use Grafana, where dashboards are configured with a JSON structure and can be imported or exported to other platforms. The JSONs used to create different dashboards can be found in the helm values repository.

Figure 27. Grafana Dashboard

Chapter 5. Budget

This section provides a breakdown of the costs involved in carrying out this project. As direct costs are the costs of **Cloud** and **Human** resources.

The cloud computing resources since November 2022 to June 2023 is shown in the Figure 28. The services were turn down when not used to try to stay in budget until the completion of the project.

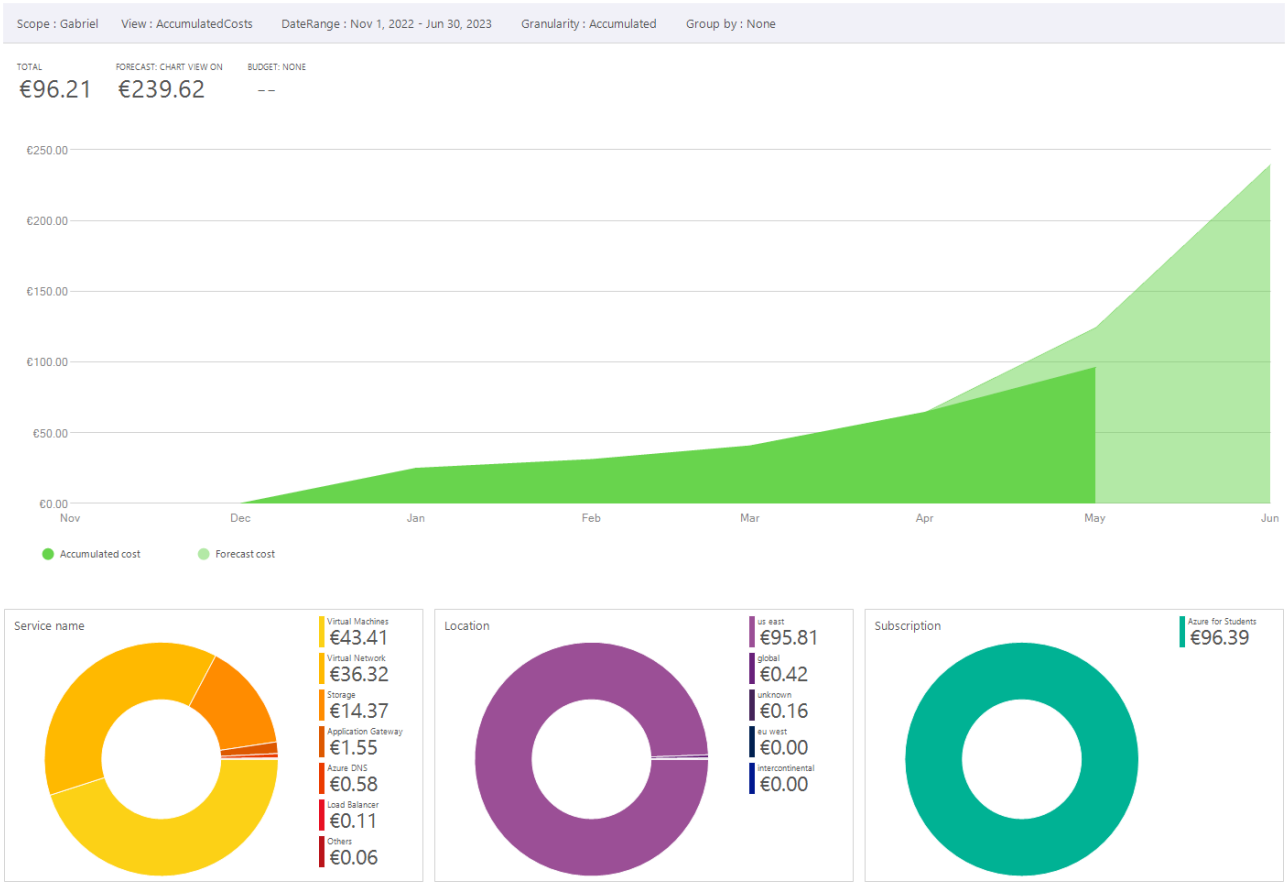

Figure 28. Azure Costs

Thanks to the capabilities of the Cloud environment it easy to pick and control the costs of our platform.

Breaking down the costs per service and resources we obtain the Table 1 imported from Azure. A lot of services were free due to the Azure Student Subscription. Half of the cost comes from the compute resources used, reflecting the time the AKS was running. The other costs for the project comes from maintaining the Virtual Networks reserved for the AKS.

TABLE 1. AZURE COSTS PER SERVICE

Service Name	Service Tier	Cost
Virtual Machines	Virtual Machines Dav4 Series	42,83 €
Virtual Network	IP Addresses - Standard IPv4 Static Public IP	36,70 €
Storage	Premium SSD Managed Disks - P10 LRS Disk	13,11 €
Storage	Standard SSD Managed Disks - Disk Operations	0,08 €
Azure Grafana Service	Azure Managed Grafana	1,56 €
Azure DNS	Azure DNS - Public Zone	0,61 €
Azure DNS	Azure DNS - Public Queries	0,00 €
Load Balancer	Load Balancer - Standard Data Processed, Free	- €
Load Balancer	Load Balancer - Outbound Rules, Free	- €
Bandwidth	Bandwidth Inter-Region	- €
Bandwidth	Standard Data Transfer Out, Free	- €
Key Vault	Key Vault	0,00 €
Container Registry	Container Registry	- €
Log Analytics	Log Analytics	- €

The limit of the budget was arrived at the end on May 2023 with all of the 100$ from the subscription used.

To simulate this project, it would be needed at least team of 4 members and their corresponding salaries during 6 months:

- A DevOps Engineer to design and implement the automation and the platform.
- Two Developers to develop each application, one specialized in JavaScript and another one in Flask.
- A Quality Assurance Engineer to define the tests of the app and monitor the applications.

Additional members could be added to complete a full team, like a Project Leader to guide the team or a Scrum Master to help implement agile methodologies.

Chapter 6. Impact of the Project

The adoption of DevOps methodology can have several positive impacts on both society and the environment. Here are some of the key benefits:

- Increased Efficiency and Productivity: DevOps practices promote automation, collaboration, and streamlined processes, leading to increased efficiency and productivity in software development and deployment. This enables organizations to deliver software faster, which can have a positive impact on society by enabling the timely release of innovative and useful applications and services.

- Improved Software Quality: DevOps emphasizes continuous integration, continuous testing, and continuous deployment, which helps identify and fix issues early in the development cycle. This results in higher-quality software products with fewer bugs and vulnerabilities. Improved software quality contributes to a better user experience, increased customer satisfaction, and reduced negative impacts on society caused by software failures or security breaches.

- Enhanced Collaboration and Communication: DevOps encourages close collaboration and communication among development, operations, and other stakeholders involved in the software development process. This improves teamwork, reduces silos, and fosters a culture of shared responsibility and accountability. Better collaboration leads to faster problem-solving, improved decision-making, and more effective utilization of resources.

- Rapid Response to Changes and Customer Feedback: DevOps enables organizations to quickly adapt to changes in user requirements, market demands, or technology advancements. Through practices like continuous delivery and feedback loops, DevOps teams can incorporate customer feedback and iterate on software features and improvements. This agility allows businesses to stay competitive and deliver products that better meet the evolving needs of society.

- Infrastructure Optimization and Resource Efficiency: DevOps emphasizes the use of infrastructure as code, automation, and cloud technologies. These practices enable efficient resource utilization, dynamic scaling, and optimized infrastructure management. By reducing the waste of computing resources and minimizing energy

consumption, DevOps contributes to a more environmentally friendly approach to software development and deployment.

- Continuous Improvement and Learning: DevOps promotes a culture of continuous improvement, learning, and innovation. Teams are encouraged to experiment, measure results, and make data-driven decisions. This culture of learning fosters professional growth, knowledge sharing, and the adoption of best practices. The collective expertise gained through DevOps practices can contribute to the advancement of the software industry as a whole.

Mostly, it would contribute to the sustainable development goals: affordable energy; decent work and economic growth; industry, innovation and infrastructure; quality education; and responsible production and consumption.

Chapter 7. Conclusions

This section explains the conclusions reached based on the objectives set at the beginning of this project. And following the conclusions, the steps that can be taken in the future to improve the system created.

7.1 Meeting Objectives

Referring to the prioritised objectives, they have been satisfactorily met. It is an easy to maintain and scale Cloud environment, following the requirements of a DevOps work cycle, in which there is code version control, a container orchestration platform, an automated continuous integration and deployment system implemented with different stages of code testing. Also a Monitoring feature has been implemented to check the state of the applications deployed.

Applications have been successfully adapted for deployment in the DevOps environment. It has been demonstrated that the environment can be used regardless of the type of application, making it flexible and can be used for the different projects of a company at least as a development environment.

The knowledge gained from this project has been broad and diverse. In fields such as application containerisation, the use of Cloud environments and services, application deployment in Kubernetes, pipeline creation to automate the CI/CD chain, application development and testing with JavaScript and Flask.

7.2 Problems Encountered

As previously mentioned, this project was initially intended to be deployed on AWS and subsequently virtualised, but due to lack of resources it has been deployed on a limited Azure account. Because the lack of resources was not detected until the project was at a more advanced point, it has been redone 2 more times and has had to be adapted for each environment, thus delaying the project.

7.3 Future Work

Due to the limited Azure account and amount of resources to be used, the project had to be cut back. For it to be a complete DevOps environment with the necessary value to be used by an organisation it would be necessary: An **Artefact Repository** to store the different build applications and its dependencies, not only containers; **Standard Deployment Blueprints** to different application life cycle to standardize and help developers to deliver their applications without paying attention to the platform itself.

But more importantly a change in the architecture of the DevOps platform design. It would be best to have:

1. Use of Pull Requests to modify the source code of the application and a first test stage for this code.

2. One additional Azure Container Registry to upload non production container images.

3. Another AKS cluster (Staging cluster) to use as a Pre-Productive environment to deploy and test the not productive images.

4. Once the image has been approved, they are promoted from the not productive to the productive one Azure Container Registry. Then they app is automatically deployed on the AKS productive cluster.

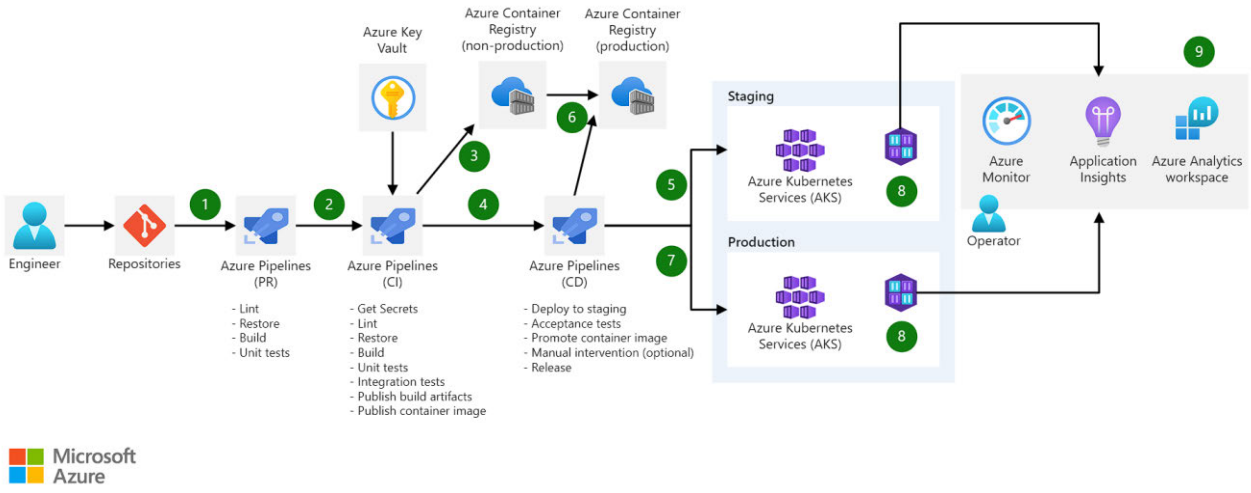

Figure 29. AKS CI/CD Microsoft approach [21]

Chapter 8. References

References

[1] R. S. a. I. P. P. Perera, «Improve software quality through practicing DevOps,» de *Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Colombo, Sri Lanka, 2017.

[2] R. Web, «Informe sobre herramientas DevOps 2020,» [En línea].

[3] T. C. a. H. Suo, «Design and Practice of DevOps Platform via Cloud Native Technology,» de *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2022.

[4] M. Virmani, «Understanding DevOps & bridging the gap from continuous integration to continuous delivery,» de *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, Galicia, Spain, 2015.

[5] M. B. A. v. B. A. C. W. C. M. F. K. Beck, «Manifesto for Agile Software Development,» Agile Alliance, [En línea]. Available: https://agilemanifesto.org/.

[6] S. Jaenhosjsaon, «Agile and Scrum Methodology,» [En línea]. Available: https://jaenhosjsaon.medium.com/agile-and-scrum-methodology-32a14ba3746f.

[7] A. O. a. A. P. G. Blinowski, «Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation,» IEEE Access, 2022.

[8] «Kubernetes - Production-Grade Container Orchestration,» [En línea]. Available: https://kubernetes.io/..

[9] L. Leahey, «Kubernetes vs Docker,» Atlassian, 2021. [En línea]. Available: https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker.

[10] M. Azure, «Cloud Computing Services,» [En línea]. Available: https://azure.microsoft.com/en-us/.

[11] T. Atlassian, «What is Version Control,» [En línea]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control.

[12] H. contributors, «Kubernetes Helm - The package manager for Kubernetes,» [En línea]. Available: https://helm.sh/.

[13] W. Kent, «Software Framework,» Wikipedia, 24 4 2021. [En línea]. Available: https://en.wikipedia.org/wiki/Software_framework.

[14] Tuts, «Create a Web App from Scratch using Python, Flask, and MySQL,» [En línea]. Available: https://github.com/tutsplus/create-a-web-app-from-scratch-using-python-flask-and-mysql/tree/main.

[15] Octodemo, «Calculator-1 Repository,» [En línea]. Available: https://github.com/octodemo/calculator-1.

[16] M. Azure, «Azure for Students | Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/en-us/free/students/.

[17] Microsoft, «Concurrent jobs in Azure Pipelines. Azure DevOps documentation,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/devops/pipelines/licensing/concurrent-jobs?view=azure-devops&tabs=ms-hosted#microsoft-hosted-cicd.

[18] M. Azure, «Azure Kubernetes Service,» [En línea]. Available: https://azure.microsoft.com/en-us/products/kubernetes-service/.

[19] S. Aslam, «Service Account Token Changes in Kubernetes Version 1.24,» LinkedIn Pulse, [En línea]. Available: https://www.linkedin.com/pulse/service-account-token-changes-kubernetes-version-124-shafeeque-aslam/.

[20] G. Gonzalez-Lopez, «Helm Values Repository,» [En línea]. Available: https://github.com/gabrielenrique-gonzalez-lopez/helm-values.

[21] Microsoft, «Azure Kubernetes Service (AKS) Continuous Integration and Continuous Deployment (CI/CD) using Azure Pipelines,» [En línea]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/aks/aks-cicd-azure-pipelines.