Thesis of the end-of-study project

To obtain the State Engineering Diploma

Major: Innovation & AMOA

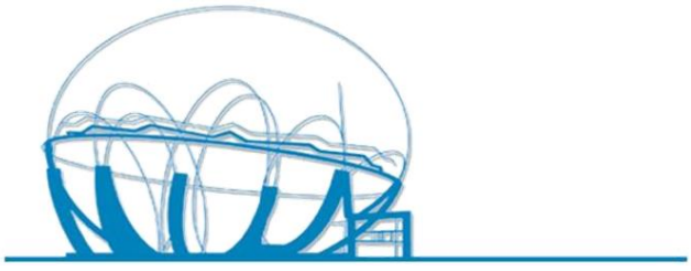
Green IT Integration in DevOps

Authored by :

Mr. Imad EL BOUHATI

Defended on July XX, 2024, before the jury composed of :

Mr. / Mrs. Xxxx XXXX	INPT	- Examiner
Mr. / Mrs. Xxxx XXXX	INPT	- Examiner
Mrs. Charifa HANIN	INPT	- Supervisor
Mr. Amine AJA	Orange Business	- Supervisor



Agence National de Réglementation des Télécommunications

Institut National des Postes et Télécommunications

Class of : 2023/2024

Dedicated to

I humbly dedicate this work to my dear mother, who gave me life, love, courage, and a reason to live. I also wish to express my gratitude to my father for his support and sacrifices.

A big thank you to my two dear sisters, Alae and Samia, whom I love so much, to all my friends with whom I have shared moments of joy and happiness, as well as to my entire extended family and to all those who love me.

Imad

Acknowledgments

Thank your supervisors (INPT and company), the managers and HR, the teachers at INPT and the jury.

Résumé

Ce rapport reflète le travail réalisé chez Orange Business Maroc dans le cadre de mon projet de fin d'études pour le diplôme d'ingénieur en Télécommunications et Technologies de l'Information.

Dans un contexte où les architectures distribuées et les pratiques de développement agile dominent la conception des applications, intégrer les principes de Green IT dans les pratiques DevOps est devenu essentiel. Cette évolution témoigne de l'importance croissante accordée à la durabilité environnementale dans le secteur de la technologie.

En se concentrant sur la surveillance de la consommation d'énergie et des émissions de CO_2 de l'infrastructure, nous mettons en lumière une préoccupation croissante pour l'empreinte écologique de nos systèmes informatiques. Cette prise de conscience a conduit à l'adoption de mesures proactives telles que l'utilisation de Kepler, Prometheus et Grafana pour surveiller et visualiser ces métriques environnementales.

Mon projet de fin d'étude vise à mettre en place Kepler pour surveiller la consommation énergétique et les émissions de CO_2 de Kubernetes, et à l'intégrer dans la chaîne CI/CD.

Mots-clés : Green IT, DevOps, Kepler, Prometheus, Grafana, Kubernetes, CI/CD.

Summary

This report reflects the work carried out at Orange Business Morocco as part of my final project for the engineering degree in Telecommunications and Information Technologies.

In a context where distributed architectures and agile development practices dominate application design, integrating Green IT principles into DevOps practices has become essential. This evolution underscores the increasing importance placed on environmental sustainability in the technology sector.

By focusing on monitoring energy consumption and CO_2 emissions of the infrastructure, we highlight a growing concern for the ecological footprint of our computer systems. This awareness has led to the adoption of proactive measures such as using Kepler, Prometheus, and Grafana to monitor and visualize these environmental metrics.

My final project aims to implement Kepler to monitor energy consumption and CO_2 emissions of Kubernetes, and integrate it into the CI/CD pipeline.

Keywords: Green IT, DevOps, Kepler, Prometheus, Grafana, Kubernetes, CI/CD.

ملخص

يعكس هذا التقرير العمل الذي قمت به في أورانج بيزنس المغرب في إطار من مشروعي الختامي للحصول على شهادة مهندس دولة في الاتصالات وتكنولوجيا المعلومات.

في سياق تهيمن فيه الهياكل الموزعة وممارسات التطوير السريعة على تصميم التطبيقات، أصبح دمج مبادئ تكنولوجيا الخضراء في ممارسات DevOps أمراً ضرورياً. يؤكد هذا التطور على الأهمية المتزايدة التي تحظى بها الاستدامة البيئية في قطاع التكنولوجيا.

وقد أدى هذا الوعي إلى اعتماد تدابير استباقية مثل استخدام Kepler و Prometheus و Grafana لمراقبة هذه المقاييس البيئية وتصورها.

يهدف مشروعي الختامي إلى تطبيق Kepler لمراقبة استهلاك الطاقة وانبعاثات ثاني أكسيد الكربون في Kubernetes و دمجه في سلسلة CI/CD .

الكلمات المفتاحية: Green IT, DevOps, Kepler, Prometheus, Grafana, Kubernetes, CI/CD

List of Figures

List of Tables

Listings

Contents

General Introduction

Modern applications are typically designed with distributed architectures and developed using agile methodologies. These continuous integration and delivery (CI/CD) practices play a crucial role in efficiently managing the software lifecycle in terms of speed and effectiveness. However, despite this efficiency, security is often overlooked in the CI/CD workflow, particularly regarding concerns related to environmental footprint.

After deploying an application in a real production environment, additional security measures are needed to ensure its protection. The use of Green IT practices, such as integrating metrics for energy consumption and CO_2 emissions into the CI/CD pipeline, is becoming increasingly important to ensure the environmental sustainability of technological infrastructures.

My thesis project fits into this context, aiming to automate the monitoring of energy consumption and CO_2 emissions in DevOps infrastructures using Kepler.

This report synthesizes the work done throughout my internship period, structured into four chapters covering various aspects of the project.

The first chapter introduces the project context, objectives, and the approach and planning followed.

The second chapter focuses on the project's requirements analysis, while the third chapter addresses the technological choices.

Finally, the fourth chapter presents the project's final structure in detail, along with an overview of the achievements. The report concludes with a summary and a list of bibliographical references.

Chapter 1

General Project Context

1.1 Introduction

This chapter aims to conduct an initial exploration of the project, in order to better understand the system requirements, and to position the project within its organizational and contextual environment. It provides an overview of the hosting organization Orange Business Morocco, then reveals the general theme of the project by introducing its intentions and specifications, followed by the various approaches applied, including the recommended methodology for project management and the resulting planning.

1.2 Presentation of host organization

1.2.1 Company Overview

Figure 1.1: OBS logo

In 2000, following the opening of the telecommunications market to competition in Europe, France Télécom, the historic French operator, acquired the British brand Orange. At the same time, the Senegalese group Sonatel was created in the late 1980s, resulting from the merger of the Office des Postes et Télécommunications and TéléSenegal. Sonatel was privatized in 1997 and integrated into France Télécom's capital.

In France, France Télécom, quickly becoming a leader thanks to the quality of its networks, gradually unified all the group's subsidiaries under the name Orange. By 2009, Orange had a presence in more than 30 countries and served over 123 million customers.

On July 1, 2013, following a vote at a general meeting, France Télécom officially changed its name to Orange. This transformation marked a new chapter for the company while retaining a cultural heritage inherited from the public service. France Télécom's historic installations played an important role in the French telephone network, paving the way for the rise of mobile telephony and the beginning of a new revolution: the Internet.

Orange operates in several regions, notably in Europe, Africa, and the Caribbean. In 2019, it was considered the leader or second operator in 75

1.2.2 Vision of the Orange Group

In its strategy of development and evolution, the ambition of the Orange Group is based on five action levers and a dynamic of an efficient and responsible digital group. These objectives, which govern the daily work of all its employees, are as follows:

- Ensure enriched, more efficient connectivity at all levels and more eco-friendly, all without borders.
- Give a new look to customer relations by further personalizing them, transforming sales spaces, and digitizing customer interactions.
- Develop a digital and human employer model and offer a unique employee experience fueled by digital and fostering engagement.
- Assist in the transformation of client companies, suggesting new working methods and leveraging technology for transformation projects.
- Diversify by leveraging its assets and future growth markets, such as mobile banking services and connected objects.

1.2.3 Introducing Orange Business - Customers and figures

Figure 1.2: OBS logo

Orange Business operates in 166 countries and serves its customers in 220 countries and territories (Table 1.1).

Date of Establishment	2006
Legal Form	Legal Form
Management	Christel Heydemann (CEO, Orange S.A) Helmut Reisinger (CEO, Orange Business Services)
Headquarters	Paris, France
Parent Company	Orange
Activity	Telecommunications Information Technology Services
Subsidiaries	Orange Application for Business, Business et Decision
Website	http://www.orange-business.com

Table 1.1: Orange Business Information

Orange Business Services (Fig. 1.2) aims to be the trusted partner for the digital transformation of its clients, with the objective of transitioning from the network operator model to the service model. It brings together the Business-to-Business activities of the Orange group carried out by various subsidiaries in France and abroad. OBS specializes in the design and development of application services and system integration in various domains for complex enterprises. Orange Business Services 28,500 employees are dedicated to French and multinational companies on five continents, accompanying them daily in their digital transformation (Fig. 1.3). It acts as both an infrastructure operator, technology integrator, and value-added service provider. OBS offers digital solutions to companies for their employees (collaborative spaces and mobile workstations), for their customers (customer relations and development of new services), and for their projects (enhanced connectivity, flexible IT infrastructures, cyber defense) (see Fig. 1.4). In 2016, OBS once again obtained the "Top Employer Europe 2016" certification. This certification recognizes the best policies and practices in terms of human resources programs. Orange Business Services aims to strengthen its global footprint in information technology services by leveraging expertise in areas such as the Internet of Things (IoT), cloud, data, and artificial intelligence (AI), application development, and cybersecurity, following an innovation-based approach to help its clients generate more value.

OBS relies on an executive committee composed of 18 members whose ambition is to:

- Design solutions for today and tomorrow, in response to customer needs and expectations.
- Provide daily support by advising, proposing, and facilitating the implementation of selected services, and providing after-sales support worldwide.
- Establish the organizations and processes of Orange Business Services for a strategy of operational excellence serving customers.

Figure 1.3: Orange Business key figures

While developing its activities, OBS IT ensures to maintain a wide diversification of its clientele and the sectors it addresses, in order to contain the risk of concentration on a limited number of clients.

Figure 1.4: OBS IT customers and partners.

1.2.4 Global Presence, Local Service

OBS IT teams are present in more than 65 countries to deploy and supervise networks and digital solutions on a daily basis (see Fig. 1.5).

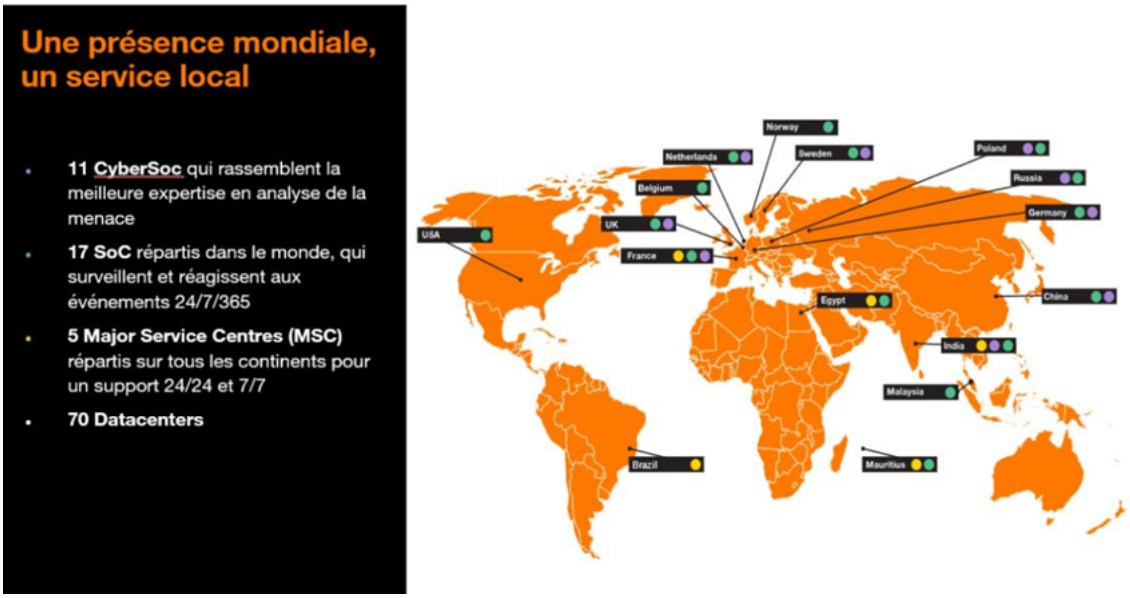


Figure 1.5: OBS Global Presence.

- 11 CyberSocs bringing together the best expertise in threat analysis.
- 17 SoCs distributed worldwide monitoring and responding to events 24/7/365.
- 5 Major Service Centers (MSC) spread across all continents for 24/7 support.
- 70 Data Centers.

1.2.5 Orange Business Morocco

Orange Business Morocco is a newly established entity in 2018 in Rabat, specializing in the design and development of application services and system integration in various domains. It works on behalf of various expertise of Orange Business Services. Several companies have provided application services for Orange Business Services, such as Almerys, a health third-party payer operator. OBS IT Morocco is founded to take control of its applications, optimize and rationalize costs, bring development teams closer to business directions, and improve IT solution delivery times for TTM. By following an Agile way of working to deliver functionalities regularly that create the most value and adopting a simple and automated development and production environment to focus on the essentials. OBS IT Morocco uses the cloud, DevSecOps tools, and automated testing and feedback utilization to enrich applications while ensuring a user-centered approach. By guaranteeing versatile, autonomous, and committed teams to manage all activities on their applications (collecting requirements, drafting user stories, development, testing, integration, deployment, support).

1.2.6 Organization and Organizational Chart

OBS Morocco, overseen by a director Mrs. Rym Sahnoun who is the highest authority, is organized into operational poles whose goal is to focus its energies on a daily basis towards customer satisfaction. Teams are organized by Orange business services. I completed my internship in the OBS IT entity managed by Mr. Aarab Abderrahim and specifically Customer Marketing Innovation (Fig. 1.6) in the DevSecOps/TAAS team attached to Mr. Moustachi Mouhcine, IT RUN Manager (see Fig. 1.7).

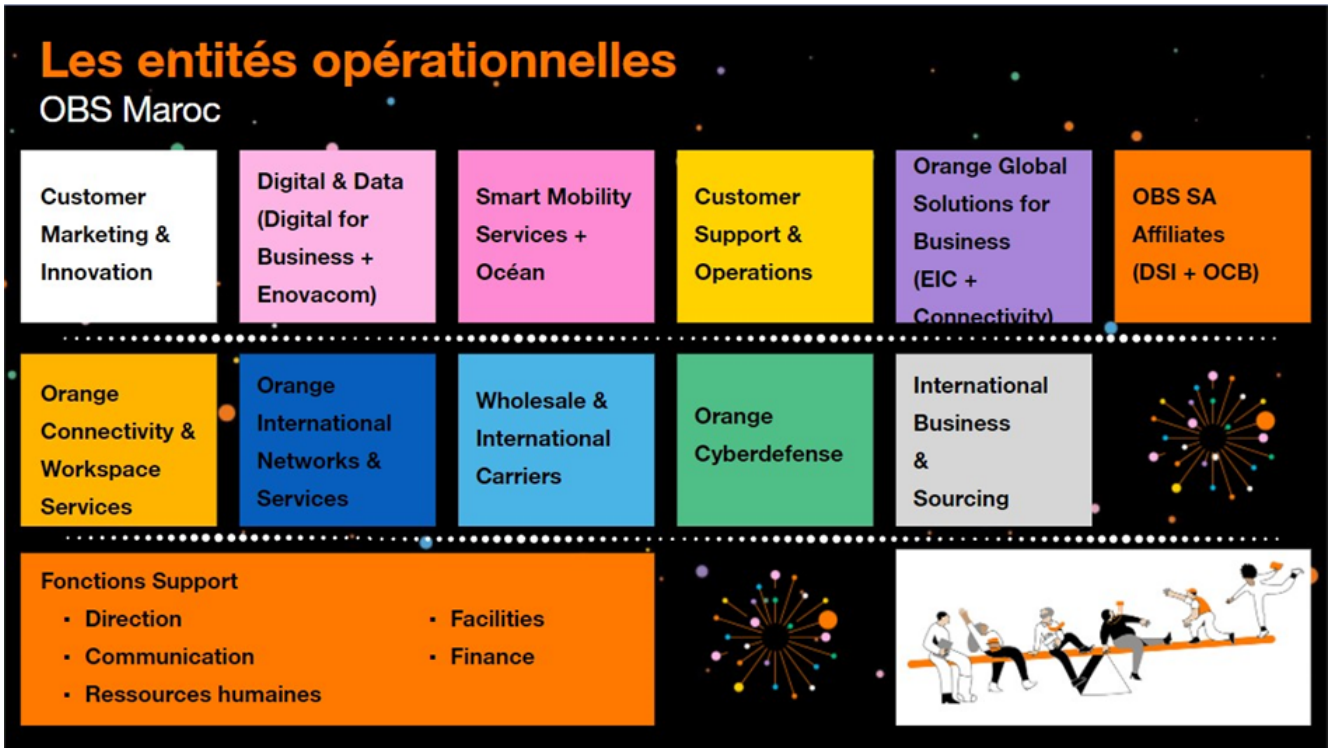


Figure 1.6: OBS IT Morocco's operating entities

CMI: This is the IT Department of the Orange Group. Its goal is to create efficient software development, entity-oriented, based on the best practices of a software factory (digital factory model). This entity adopts an agile delivery model, possesses a simple and automated development and production environment. It is a versatile, autonomous, and committed team to manage all activities related to the applications entrusted to it: from collecting customer needs and requirements, to writing user stories, through development, testing, and integration, before reaching deployment and support.

- Enrich the relationship between companies and their customers through a digital approach and a 360° customer journey.
- Exploit machines and communicating objects that transmit real-time data to create new sources of revenue and differentiation.
- Make use of the data produced by companies and their customers to innovate and personalize their products and services.

OCWS: This entity is composed of 3 poles with different missions. The first center, internationally / in Morocco, manages the operational part. A pre-sales team provides technical assistance to commercial engineers to help them negotiate contracts. A BUILD team's missions include deploying network

solutions for customers and offering them the best possible connectivity experience through various technologies such as SDWAN, WIFI, LAN, etc.

OCD: This is the entity that designs cybersecurity services that support customers in the Maghreb and West Africa, throughout the life cycle of threats that can impact client companies. They have 5 main missions:

- Anticipate the latest threats.
- Identify the assets and critical data of customers, prepare the security strategy, and ensure its proper functioning.
- Deploy appropriate technology to defend the organization and manage it continuously.
- Monitor, qualify, and analyze security alerts to confirm if there has been an incident.
- Qualify, contain, and remedy attacks. Thanks to their sectoral expertise, they can offer tailor-made services to meet customer challenges.

Orange Connectivity: Entity responsible for marketing OBS Connectivity data offers (Internet, Ethernet, VPN).

Missions in Morocco: Management of pricing for offers marketed in France and internationally (definition and update of standard prices, calculation of custom prices).

OINIS: Orange International Networks Infrastructures & Services' mission is to design, deploy, and operate reliable, secure, and high-quality international network infrastructures and services for businesses, wholesale customers (operators), and subsidiaries.

WIN IC: This is the commercial arm of OINIS. This entity provides international connectivity services to operators and internet service providers worldwide.

The Business Unit EIC (Enriched Interactions and Collaboration): Straddling digital & data and connectivity, its goal is to design and bring to market communication and collaboration offerings from Orange Business Services: voice solutions, conference solutions, unified communications suites, and contact center solutions.

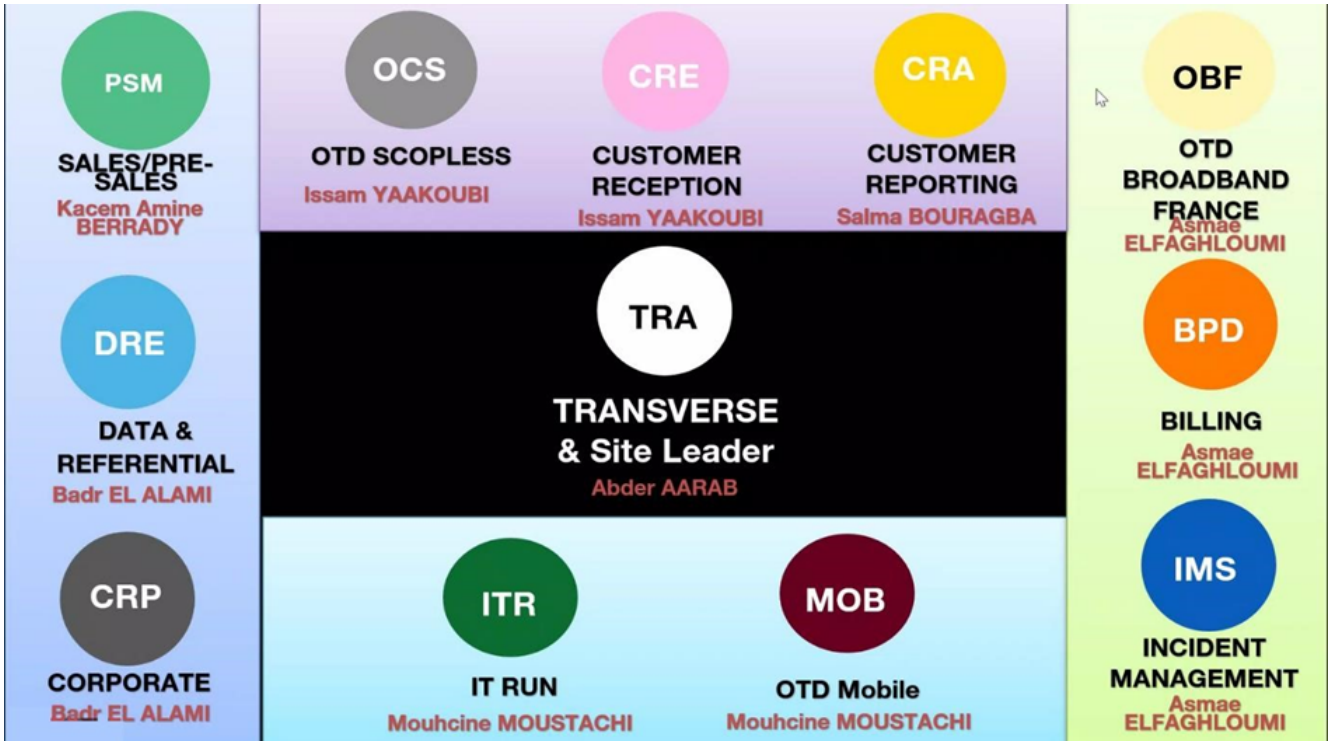


Figure 1.7: OBS IT organization chart

1.3 Introducing the DevSecOps team

DevSecOps is a team recently created within the IT RUN entity (Fig. 1.7) to fulfill two main missions:

- Cloud DevOpSec: Design, build, and implement Cloud services and DevSecOps tools (Fig. 1.9).
- REaCT: The goal is to implement the DevOps mindset within the company and support development projects (Fig. 1.8).

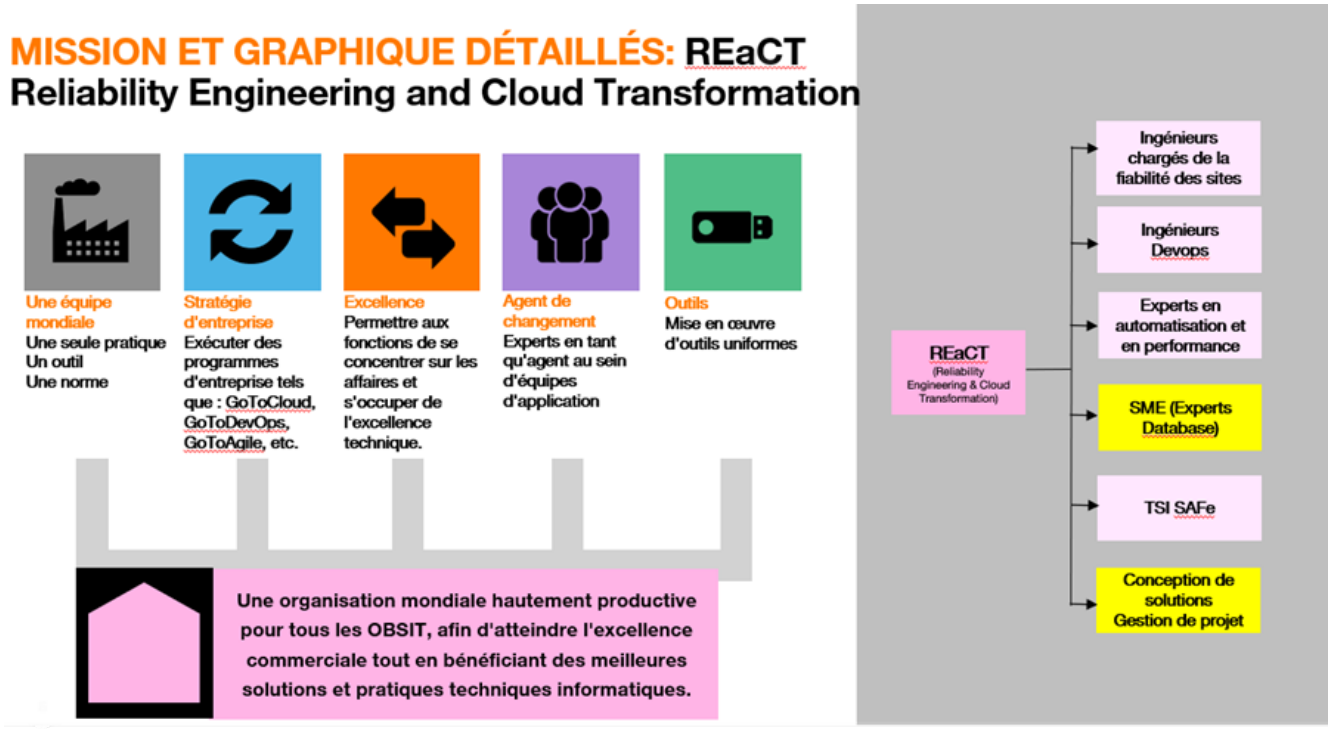


Figure 1.8: React missions

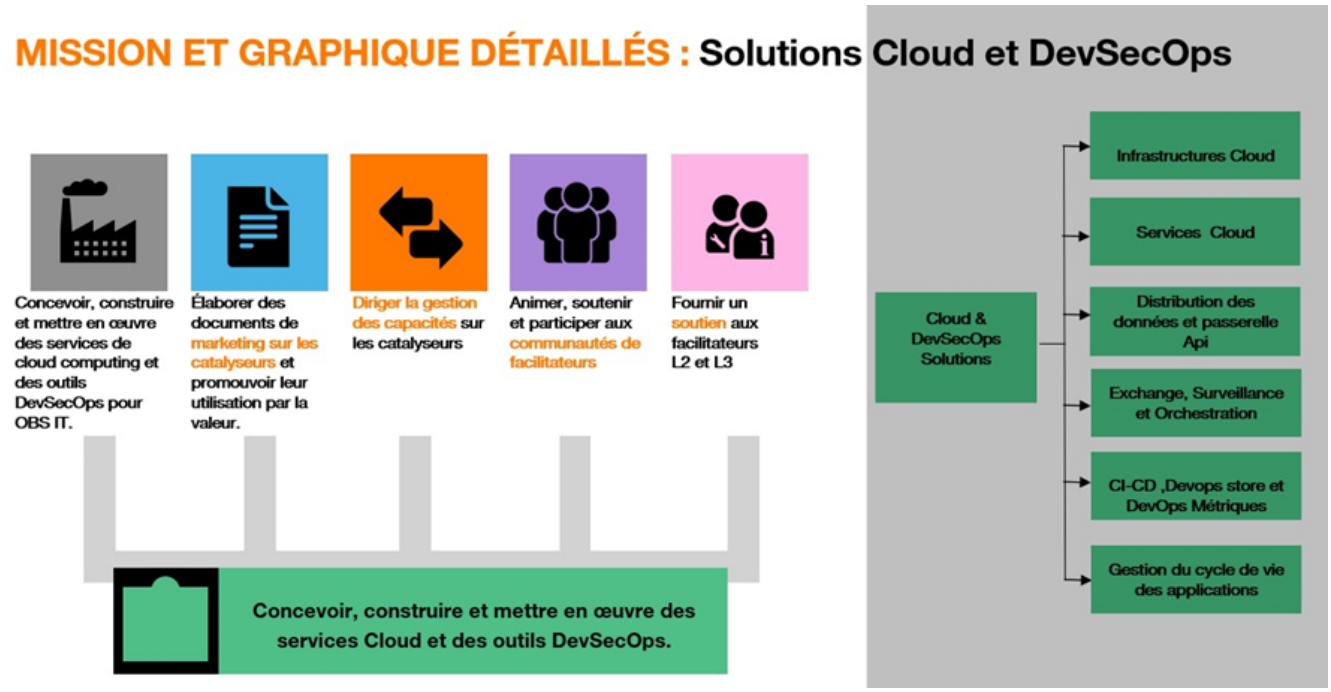


Figure 1.9: Cloud & DevSecOps missions

1.4 Project presentation

1.4.1 General project context

In today's world, ensuring the sustainability of IT practices is paramount. When it comes to IT professionals, it is imperative to prioritize eco-friendly strategies from the outset of the development process.

As a developer, you may not be required to master the intricacies of environmental impact assessments, as there are several reliable tools available to assess the carbon footprint of IT solutions. By integrating these assessments into the CI/CD pipeline and implementing effective remediation strategies, organizations can significantly reduce the environmental cost associated with IT operations.

This includes not only the direct costs of energy consumption but also the broader environmental impact of resource depletion and pollution. Consequently, IT teams have embraced a new approach known as Green IT, which promotes the integration of sustainability principles into IT practices and encourages close collaboration between environmental experts, developers, and IT engineers.

This approach ensures that environmental considerations are addressed at every stage of the development and delivery process, thereby fostering a more sustainable IT ecosystem.

1.4.2 Problematic

While Orange Business Services Morocco has established efficient CI/CD pipelines using GitLab, there is a lack of focus on environmental sustainability and energy efficiency within their DevOps processes. The current approach prioritizes swift software delivery but overlooks the environmental impact of development and deployment activities. This neglect of Green IT principles hinders efforts to reduce energy consumption and carbon emissions associated with the organization's IT infrastructure, particularly its Kubernetes clusters. The challenge lies in integrating Green IT considerations seamlessly into existing DevOps workflows without compromising efficiency or productivity. Thus, the problematic revolves around balancing the need for rapid software delivery with environmental responsibility, requiring the development of strategies and tools to monitor and optimize energy usage and carbon footprint throughout the CI/CD pipeline

1.5 Project planning and management

1.5.1 Method adopted

To ensure optimal progress of our project, we have chosen to use the agile Scrum method (illustrated in Fig. 1.10).

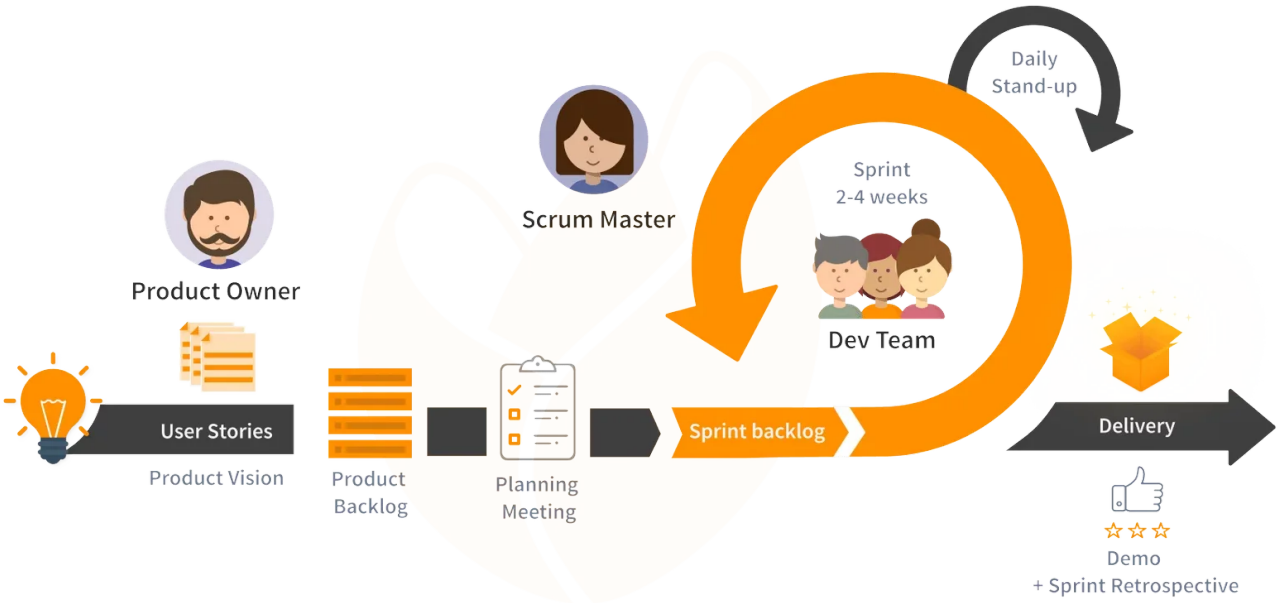


Figure 1.10: Scrum methodology

This approach will enable us to best meet the needs of the company while minimizing the risk of delays and ensuring the satisfaction of all stakeholders.

Scrum is an agile project management method that was first introduced in an article by Hirotaka Takeuchi and Ikujiro Nonaka in 1986, published in the Harvard Business Review under the title "The New Product Development Game". It was developed in the 1990s and formalized by Ken Schwaber and Jeff Sutherland in 1995.

The fundamental principle of this method is to iteratively focus on the features to be implemented. The project is divided into functional modules that are developed, tested, and delivered in iterative sequences called "sprints". Each sprint aims to achieve a specific goal, from which the features to be implemented are chosen.

We have adapted the Scrum method to our project by dividing it into sprints of two to three weeks. At the end of each sprint, a meeting is organized with the Scrum Master and the Product Owner to review the work done during the period and set the tasks and objectives for the next sprint. If differences are noted during the inspection, we adapt the process in question. To do this, we have set up weekly

meetings led by the Scrum Master, during which each team member presents the status of their tasks, reports any encountered blockers, and outlines the actions to be taken for the next period as well as future prospects. This approach has allowed us to clearly define the objectives for each increment and adapt them as needed, while also allowing us to complete or modify the list of features to be implemented for the upcoming sprints.

1.5.2 Project Planning

Planning is an essential step in project management, as it allows for defining the work to be done, setting objectives, coordinating actions, managing resources, reducing risks, tracking ongoing actions, and reporting the project's progress. In our case, we started by breaking down the project into tasks, then proceeded with a risk assessment before establishing a Gantt chart to have an overview of the project's progress.

1.6 Conclusion

In this chapter, we first introduced Orange Business Morocco, the organization involved in the project. Then, we established the general framework of the project before presenting the specific problem to be addressed. We also explained the working method we adopted and provided an overview of the schedule that was followed throughout the project. In the next chapter, we will move on to the functional and non-functional analysis of our project, as well as detailed design. This phase will define the specifications necessary for the development of the overall project architecture.

Chapter 2

Specification and Needs Analysis

2.1 Introduction

The success of a project heavily relies on the quality of its initiation. Therefore, the functional study phase is crucial for a successful start to the project. This chapter encompasses requirement specifications, including needs analysis involving stakeholder identification, system use cases, and the textual scenarios associated with these use cases.

2.2 Study of the Current State

Orange Business Services Morocco has embraced an approach known as CI/CD to efficiently manage the development and delivery of its projects. They utilize the GitLab platform, which facilitates swift creation, compilation, testing, delivery, and deployment of software.

In their existing solution, they have established continuous integration and delivery pipelines using GitLab CI, Nexus, Docker, and Kubernetes. Here's how it operates:

- Code management and updates are performed using GitLab. Developers push their code changes to the GitLab repository.
- The application is containerized using Docker, a technology enabling the creation of isolated and portable environments for applications.
- The GitLab Runner conducts a check to ensure the YAML code is correct and clean. Then, it triggers pipeline steps such as Docker image generation and their submission to the Nexus repository.
- Kubernetes, aided by Helm (a package manager for Kubernetes), handles the application deployment across various environments. It follows a "Push" deployment model.

This approach enables Orange Business Services Morocco to efficiently manage the development and delivery of their projects by automating processes using GitLab, Docker, Nexus, and Kubernetes. It empowers them to deliver high-quality software more rapidly.

2.3 Analysis and criticism of the current state

While utilizing GitLab accelerates software development and deployment, there's a clear need for enhancements focused on sustainability and reducing environmental impact. An exhaustive study of the existing system was conducted to refine our project's scope and expected functionalities, aiming for a more reliable system than the previous one. Identified areas for improvement include

- The current CI/CD approach relies heavily on resource-intensive technologies like Docker and Kubernetes, which contribute to increased energy consumption and carbon emissions.
- While effective for streamlining development workflows, the pipelines lack mechanisms for tracking and optimizing energy usage.
- Without visibility into the energy consumption and carbon footprint of the Kubernetes cluster, targeted strategies for reducing environmental impact are challenging to implement.

2.3.1 Functional needs

Functional requirements gathering is a crucial step in the project. This stage produces the functional specifications document, during which the expected functionalities are formalized along with all governing management rules. To address the issues identified in the existing study, the principle is to integrate monitoring tools such as Kepler (Kubernetes-based Efficient Power Level Exporter), Prometheus, and Grafana into the CI/CD pipeline. This integration aims to measure the energy consumption and CO_2 emissions of Orange's Kubernetes cluster, providing valuable insights for optimizing Green IT practices within the DevOps workflow.

2.3.2 Non-Functional needs

Non-functional requirements play a crucial role in system design, because they define the attributes, constraints and restrictions to be taken into account.

These requirements, also called system qualities, guarantee the user-friendliness and overall efficiency of the system. If any of these requirements are not met, the system risks not meeting the internal needs of the company, users or the market.

- **Accuracy**: The system must provide accurate measurements of energy consumption and CO_2 emissions to support informed decision-making.

- **Scalability**: Ensure scalability to handle increasing data volumes as the size of the Kubernetes cluster grows.
- **Reliability**: The solution should be reliable, with minimal downtime, to maintain continuous monitoring and data collection.
- **Security**: Implement robust security measures to safeguard sensitive energy consumption and emission data from unauthorized access or tampering.
- **Performance**: Ensure efficient performance to deliver timely insights and visualizations, even during peak usage periods.
- **Ease of Use**: Design an intuitive user interface that allows administrators to easily configure monitoring settings and interpret data visualizations.
- **Compatibility**: Ensure compatibility with existing infrastructure and tools used within Orange Business Services Morocco, such as Prometheus, Grafana, and Kepler.

2.4 DevOps appraoch

The word DevOps is a combination of the terms development and operations, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams. The DevOps methodology aims to shorten the systems development lifecycle and provide continuous delivery with high software quality. These characteristics help ensure a culture of building, testing, and releasing software that is more reliable and at a high velocity.

2.5 Definition of CI/CD

Continuous Integration (CI) is a software development practice where developers frequently merge their code changes into a central repository, typically multiple times a day. Each merge triggers an automated build and testing process. The primary goal of CI is to detect and address issues early in the development cycle, which helps in maintaining code quality and reducing integration problems. By integrating frequently, developers can identify bugs early, facilitating easier and quicker fixes.

Continuous Delivery (CD) extends CI by automatically preparing code changes for a release to production. It ensures that the software can be reliably released at any time, with the deployment process being automated but requiring manual approval. Continuous Deployment is a further extension

of CD where every change that passes all stages of the production pipeline is automatically released to customers without human intervention. This practice reduces the lead time for delivering new features and fixes, thus providing rapid feedback to developers and stakeholders.

2.6 Definition of Green IT

Green IT, also known as Green Information Technology, refers to environmentally sustainable computing practices. It encompasses a broad range of strategies aimed at minimizing the environmental impact of IT operations. This includes optimizing energy consumption, reducing electronic waste, and promoting the use of eco-friendly technologies.

The goal of Green IT is to create more efficient and sustainable systems by implementing practices such as energy-efficient hardware, virtualization, and effective cooling systems in data centers. It also involves the adoption of cloud computing to optimize resource use and the implementation of robust recycling programs for electronic devices.

2.7 Green DevOps:

Green DevOps extends DevOps principles, focusing on resource efficiency while staying agile. It emphasizes sustainability, reducing environmental impact, and integrating eco-friendly strategies into software development. The goal is to balance technological innovation with ecological responsibility.

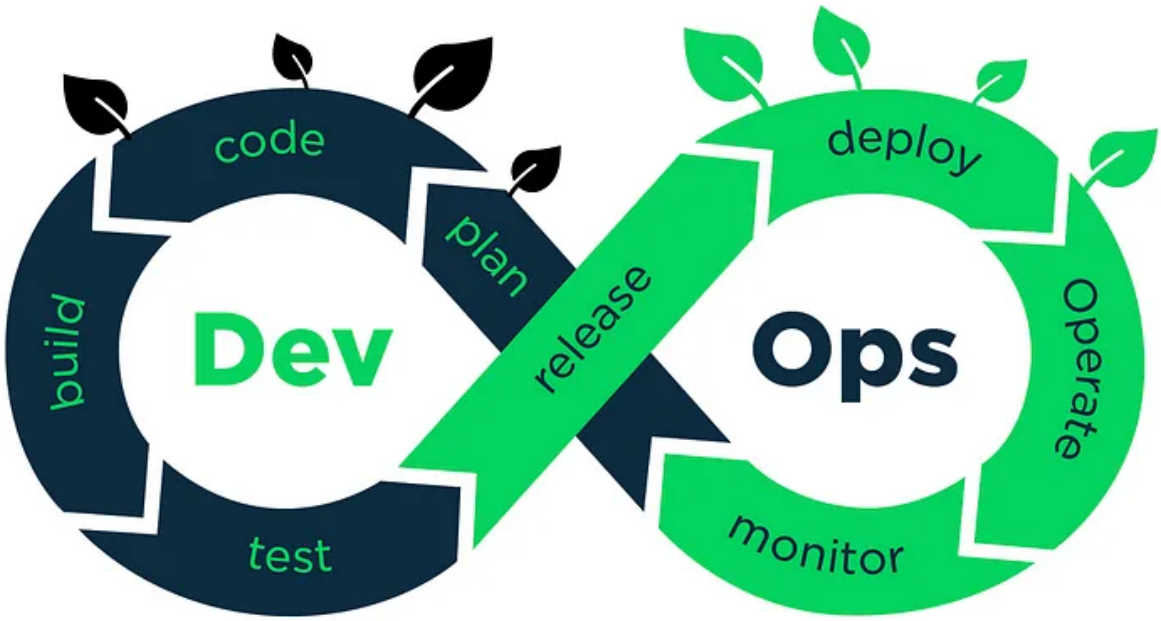


Figure 2.1: Green DevOps

2.7.1 The Fundamental Principles of Green DevOps

Green DevOps is nothing more than an extension of DevOps. It shouldn't be viewed as an upheaval of all DevOps principles. We retain all DevOps principles but adjust them as finely as possible to optimize resource usage.

Green DevOps relies on the following concepts:

- **Infrastructure Optimization:** Choosing eco-friendly data centers, energy-efficient hardware, and effective resource utilization are pillars of Green DevOps. The goal is to reduce energy consumption while maintaining performance.
- **Reduction of E-waste:** This involves minimizing over-provisioning of IT resources, properly managing electronic waste, and promoting hardware component recycling.
- **Automation and Process Optimization:** Automation is at the core of DevOps, but it holds even greater importance in the context of Green DevOps. Automating tasks reduces energy consumption by minimizing unused resources and enabling fine-grained resource management.
- **Carbon Impact Assessment:** To truly become "green," measurement is necessary. Green DevOps integrates tools and metrics to continuously assess and monitor the carbon footprint of projects. This helps identify areas needing improvement and track progress.
- **Reduction of Cycle Times:** Shorter development cycles mean less energy consumption and faster time to market. By reducing delays, Green DevOps contributes to the competitiveness of the business while minimizing its environmental impact.
- **Awareness:** DevOps teams need to be aware of environmental issues. Employee training and engagement are essential to fostering the adoption of sustainable practices.

2.7.2 The Benefits of Green DevOps

- **Cost Reduction**: Less wasted resources, time saved, and efficient resource utilization result in a significant reduction in operating and infrastructure costs.
- **Quality Improvement**: Green DevOps practices encourage automation, monitoring, and rigorous testing, leading to better software quality, fewer bugs, and an enhanced user experience.

- **Social Responsibility**: Companies adopting Green DevOps demonstrate their commitment to environmental sustainability, which can enhance their brand image and attract new customers and talents.
- **Regulatory Compliance**: With an increasing number of environmental regulations in place, Green DevOps helps companies comply with standards and avoid potential penalties.

2.8 DevOps Monitoring

DevOps monitoring is the process of tracking and measuring the performance of applications and systems in order to help software development teams identify and resolve potential issues more quickly. This is typically done via a manual or automated DevOps monitoring solution or a collection of continuous monitoring tools that gather data

2.8.1 DevOps Monitoring Use Cases

The main benefit of DevOps monitoring is its ability to define, track, and measure KPIs across all aspects of DevOps. Here are some specific use cases of DevOps monitoring:

- **Detect and Report Errors Earlier:** Flagging issues to DevOps teams more quickly means they can resolve them before they impact user experience. Early detection and reporting of errors allow for prompt resolution, minimizing disruptions and maintaining a smooth user experience.
- **Reduce System Downtime:** DevOps monitoring tools provide continuous oversight of databases, applications, and networks, enabling teams to resolve issues before system downtimes occur. By proactively identifying potential problems, teams can take preemptive actions to avoid outages and ensure system reliability.
- **Enhance Observability of DevOps Components:** Easily identify when various systems and applications in your DevOps stack degrade in performance, cost, security, or other factors to avoid problems down the road. Enhanced observability enables teams to maintain optimal performance and security by monitoring and analyzing system behavior continuously.
- **Uncover Root Cause of Issues Faster:** Continuous tracking of logs and metrics helps teams identify the root cause — where a problem started or occurred. This allows engineers to detect patterns in system behavior to anticipate and prevent future issues, improving mean time to detection (MTTD), mean time to repair (MTTR), and mean time to isolate (MTTI).

2.9 Needs Analysis

After identifying the functional and non-functional requirements of the project, it is essential to clearly define the stakeholders of the system and the different use cases. This step is crucial for preparing the design phase of the project.

2.9.1 Identification of Stakeholders

Identifying stakeholders is a fundamental step in any development project. It helps to understand the interactions and measure the influence of each group on the actions to be undertaken. But what exactly is a stakeholder? A stakeholder can be a physical person or a legal entity that participates in or is affected by the action or project in question. It is important to clearly specify the action or series of actions for which we seek to determine who the stakeholders are and what they represent.

In the context of integrating Green IT into DevOps practices at Orange Business Services Morocco, we identify the following stakeholders:

- **Platform Admin Team**: Responsible for setting up staging and production environments, managing cluster extension modules, and other cluster resources (controllers and admission webhooks), integrating development team repositories using customized GitRepository resources, and configuring how development team repositories are reconciled on each cluster.
- **Development Team**: Configures application definitions at the Kubernetes deployment level, manages Helm releases, configures how applications are reconciled across environments, and oversees the promotion of applications between environments.
- **Green IT Team**: Responsible for analyzing the energy consumption and CO_2 emissions of the Kubernetes clusters.
- **GitLab**: A secondary stakeholder that manages team access and coordinates the CI/CD processes.

By identifying these stakeholders and their respective roles, we can ensure a comprehensive approach to integrating Green IT into our DevOps practices, balancing rapid software delivery with environmental responsibility.

2.9.2 Use Case Diagrams

The role of use case diagrams in UML is to model the behavior of a system and capture the associated requirements. These diagrams highlight the high-level functions of the system and define its scope. They also help identify the interactions between the system and its stakeholders. However, it is important to note that use case diagrams focus on what the system does and how the stakeholders use it, without delving into its internal workings.

In this perspective, a use case represents a function that the system performs to achieve a specific goal for the user. It should generate an observable result that holds value for the system's user. On the other hand, a stakeholder represents the role of a user who interacts with the modeled system. This could be a human, an organization, a machine, or another external system.

The relationships between elements in use case diagrams play a crucial role. As UML model elements, these relationships establish semantic links that define the structure and behavior among different elements of the model.

In the context of integrating Green IT into DevOps at Orange Business Services Morocco, we present the global use case diagram of our solution in Fig. **??**. This diagram provides an overview of the interactions between the stakeholders and the various functionalities of the system.

2.10

Chapter 3

Technologies Used

The adoption of the DevOps approach largely relies on the automation of communication between the various required tools, thus allowing us to benefit from the previously described functionalities. The choice of tools for each stage of our DevOps pipeline, such as continuous integration and continuous deployment, depends on specific criteria related to our needs. Consequently, this comparative study aims to reinforce the selection of tools that we have implemented for the integration of Green IT into DevOps practices.

3.1 Application Containerization

In this section, we will delve into our problem by examining the fundamental principles of traditional virtualization and containerization. Our aim is to provide a clear description of these concepts and explain the relationships between them.

3.1.1 Virtualization

Virtualization involves creating a virtual version of a device or resource, such as an operating system, server, storage device, or network resource. It allows for the abstraction of physical resources, representing them independently of their physical equivalents. Each virtual element, such as a disk, network interface, local network, switch, processor, or memory, is associated with a physical resource in a computer system. Thus, virtual machines hosted by the host machine are considered applications that require allocation or distribution of the host's resources. There are various types of virtualization, each corresponding to a specific use case:

- **System Virtualization** is a server virtualization technique that uses a hypervisor to allow the host machine to run multiple virtual instances simultaneously. These virtual instances are commonly referred to as virtual machines (VMs), while the hypervisor is known as the virtual machine monitor (VMM), responsible for managing the VMs.
- **Application Virtualization** differs from system virtualization by using a virtualization layer in the form of an application. This application creates virtual instances and isolates them from the specifics of the host machine, such as its architecture or operating system. This approach allows developers to avoid creating multiple versions of their software to run in different environments. Common application virtual machines include the JVM (Java Virtual Machine) and containers.

3.1.2 Containerization

Container-based virtualization, also known as containerization, offers an alternative to system virtualization. This approach directly leverages kernel features to create isolated virtual environments called containers. Containers use control groups (cgroups) and namespaces provided by the operating system kernel. Namespaces allow for controlling and limiting the resources used by a process, while cgroups manage the resources of a group of processes. Thus, a container provides the necessary resources to run applications in isolation, as if they were the only processes running on the host machine's operating system.

Containers offer advantages such as scalable, modular, and loosely coupled application development. The need to create portable deliverables independent of development or production environments has become a standard. The term "container" emerged to refer to an application independent of the system, represented as a box containing all the dependencies necessary for its proper functioning.

3.1.3 Containerization vs Virtualization

Containers have an intrinsically smaller footprint than virtual machines and require less startup time. This means that more containers can run on the same computing capacity compared to a single virtual machine. This improvement in server efficiency reduces costs associated with servers and licenses. In simple terms, containerization allows applications to be developed once and run anywhere. This portability is crucial for streamlining the development process and ensuring compatibility with different service providers.

Figure 3.1 illustrates the distinction between container architecture and virtual machine architecture.

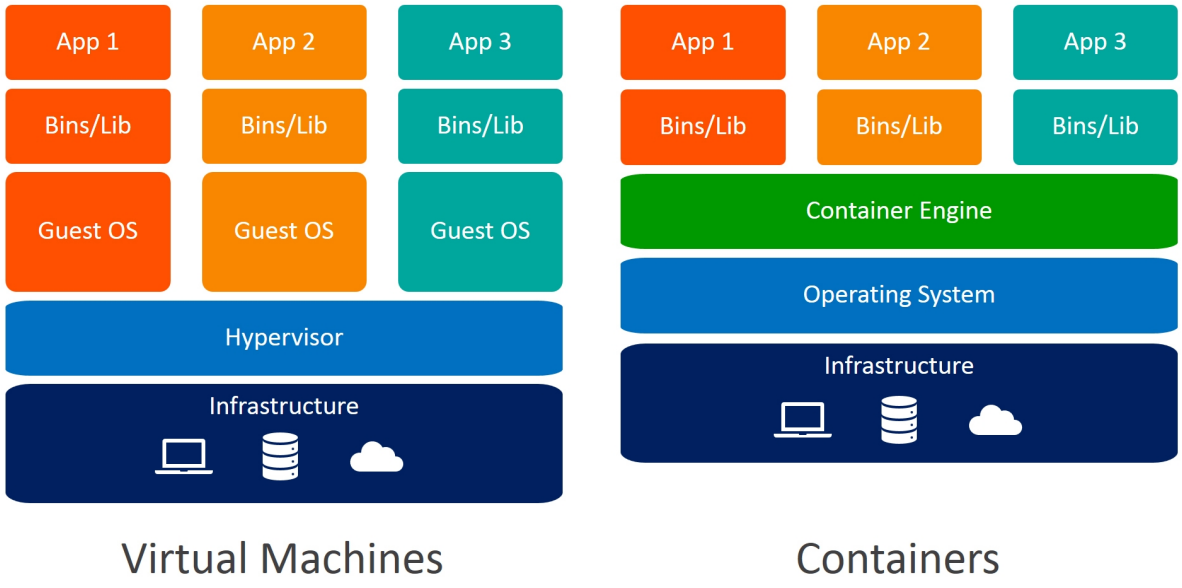


Figure 3.1: Comparison between Container and Virtual Machine Architectures

3.1.4 Containerization Technologies

There are several containerization tools available in the market. We conducted a comparison to select the most suitable one.

The analysis presented in Table 3.1 highlights the significance of Docker in the market. Docker stands out as the most universal solution in terms of image compatibility and offers numerous additional

Technology	Principle	Market Position	Documentation/ Complexity
Docker	A container contains a single process; multiple containers can be connected to form an application.	Market leader, widely used by major online service providers (e.g., Netflix, Spotify)	Official documentation is available; community-driven tutorials and resources are abundant. Known for its simplicity.
Podman	A container contains a single process; multiple containers can be connected to form an application.	Increasingly popular, supported by major enterprises like Google and Red Hat. Acquired by Red Hat in May 2018.	Limited documentation, somewhat difficult due to lack of resources but highly modular and compatible with ancillary tools.
LXC/LXD	Isolation of an entire application within a complete Linux environment (similar to a VM).	Established since 2008, the only one closely resembling a VM. Isolated but with an active community.	Limited documentation; most information is posted on Ubuntu community forums.

Table 3.1: Comparison of Containerization Tools

advantages. It simplifies the creation and management of containers, facilitates the design and construction of images, and enables easy distribution and version control of these images. Thus, Docker plays a crucial role in streamlining and enhancing containerization processes.

3.2 Introduction to Docker

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.

- **Docker Daemon (*dockerd*):** Responsible for handling Docker API requests and managing various Docker objects such as images, containers, networks, and volumes. Listens for incoming requests and executes them accordingly. Can interact with other daemons to manage Docker services across different environments.

- **Docker Client (*docker*):** Primary interface for Docker users to interact with the Docker ecosystem. Users issue commands through the Docker client (e.g., `docker run`), which are communicated to the daemon for execution. Utilizes the Docker API for communication and can connect with multiple daemons for managing Docker instances.

- **Docker Registries:** Facilitate storage and distribution of containerized applications. Docker Hub is a public registry accessible to all users, while private registries can be set up for hosting proprietary or sensitive images. Commands like `docker pull` or `docker run` fetch required images from the configured registry, and `docker push` uploads images to the designated registry for sharing or backup purposes.

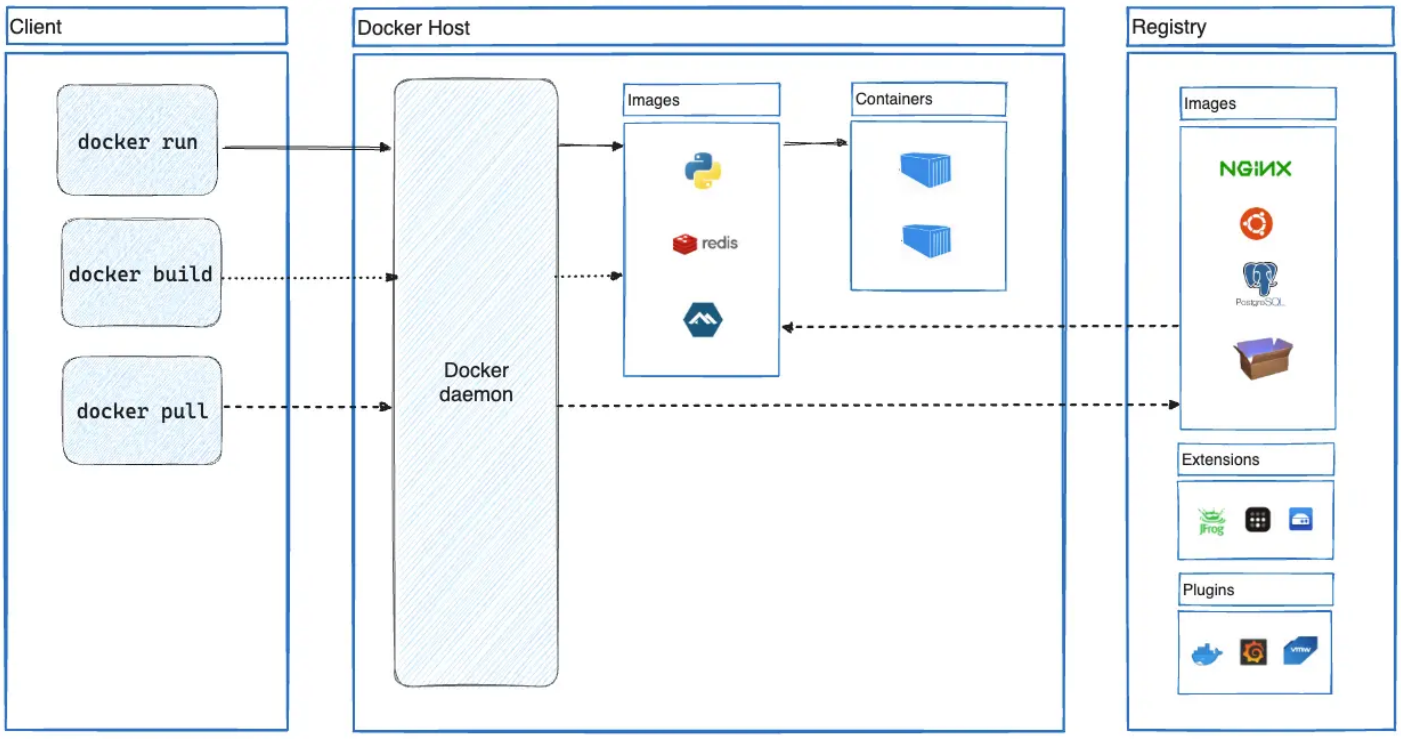


Figure 3.2: Docker Architecture

3.3 Container Orchestration

Container orchestration involves automating much of the operational tasks required to run containerized workloads and services. This encompasses various aspects that software teams need to manage the lifecycle of a container, such as provisioning, deployment, scaling (both up and down), network management, and load balancing. Container orchestration simplifies and optimizes container management by providing tools and features to effectively coordinate different container instances and associated resources. This facilitates the deployment and management of complex and scalable containerized applications in production environments.

3.3.1 Why Orchestration is Necessary

The ability to create containers has existed for several decades but became widely accessible in 2008 when Linux integrated container functionality into its kernel. However, widespread adoption of containers took off with the introduction of Docker, an open-source containerization platform, in 2013. Docker became so popular that the terms "Docker containers" and "containers" are often used interchangeably.

Containers have become the de facto computing units for modern cloud-native applications due to their small size, resource efficiency, and portability compared to virtual machines (VMs). Containerized microservices or serverless functions are particularly prevalent in these applications. Manually deploying and managing a small number of containers is relatively easy. However, in most organizations, the number of containerized applications is rapidly increasing, making large-scale management impossible without automation. This is where container orchestration comes into play.

Container orchestration provides essential automation for managing a large number of containerized applications. This significantly reduces the effort and complexity associated with managing this application fleet. By automating operations, orchestration facilitates an agile or DevOps approach, allowing development teams to rapidly deploy new features and capabilities.

The automation provided by orchestration enhances many inherent benefits of containerization. For example, it enables automated host selection and efficient resource allocation based on declarative configuration, thereby maximizing the utilization of computing resources. Additionally, orchestration provides automated monitoring of container health status and can move containers as needed to maximize availability.

In summary, container orchestration brings essential automation to effectively manage a large number of containerized applications. It facilitates rapid and iterative development cycles, allowing teams to

release new features and capabilities more quickly. Furthermore, orchestration can enhance and extend the benefits of containerization through advanced features such as resource management and automated monitoring.

3.3.2 Kubernetes Container Orchestration

Kubernetes is a powerful and popular container orchestration platform that enables enterprises to provide a highly productive PaaS for developing cloud-native applications. With Kubernetes, development teams can focus on coding and innovation, while container infrastructure and operations are managed automatically and efficiently. Kubernetes' advantages over other orchestration solutions (such as Docker Swarm, Apache Mesos, OpenShift, AWS Fargate) largely stem from its various more comprehensive and sophisticated features in several areas, including:

- **Container Deployment:** Kubernetes deploys a specified number of containers on a given host and maintains them in the desired state.
- **Rollouts:** A rollout is a modification to a deployment. Kubernetes allows launching, pausing, resuming, or canceling deployments.
- **Service Discovery:** Kubernetes can automatically expose a container to the internet or other containers using a DNS name or IP address.
- **Storage Provisioning:** Developers can configure Kubernetes to mount local or cloud-based persistent storage for containers as needed.
- **Load Balancing and Scaling:** When traffic to a container experiences a high spike, Kubernetes can employ load balancing and scaling to distribute it across the network to ensure stability and performance. (This also saves developers from having to configure a load balancer)
- **Self-healing for High Availability:** When a container fails, Kubernetes can automatically restart or replace it. It can also take containers out of service that do not meet our health check requirements.
- **Multi-cloud Provider Support and Portability:** Kubernetes enjoys broad support from all major cloud providers. This is particularly important for organizations deploying applications in a hybrid or multi-cloud environment.

3.4 Helm

Figure 3.3: Helm

Helm is a package manager for Kubernetes, facilitating the deployment, management, and scaling of applications within a Kubernetes cluster. Helm simplifies complex Kubernetes application deployments by providing a higher level of abstraction through Helm charts. These charts are pre-configured packages of Kubernetes resources.

3.4.1 Key Features of Helm

- **Charts:** Helm uses charts to define, install, and upgrade even the most complex Kubernetes applications. A Helm chart is a collection of files that describe a related set of Kubernetes resources.
- **Releases:** When a chart is deployed, it becomes a release. Helm manages these releases, enabling rollbacks and upgrades.
- **Repositories:** Helm charts can be stored and shared via repositories, making it easy to distribute applications.
- **Templates:** Helm charts support templates, which allow for dynamic and customizable Kubernetes manifests. Templates are written in Go templating syntax.
- **Lifecycle Management:** Helm provides commands to manage the entire lifecycle of applications, from installation and upgrades to rollbacks and deletion.

3.4.2 Benefits of Using Helm

- **Simplified Deployments:** Helm abstracts the complexity of Kubernetes configurations, making deployments more straightforward and repeatable.
- **Version Control:** Helm charts allow you to manage versions of your application, enabling easy rollbacks and upgrades.
- **Sharing and Reusability:** Helm charts can be shared across teams and organizations, promoting reusability and standardization.
- **Scalability:** Helm facilitates the management of applications at scale, handling multiple Kubernetes resources efficiently.

Helm significantly enhances the Kubernetes experience by providing a robust framework for managing the complexities of application deployment and lifecycle management. It empowers developers to create more consistent and reliable deployments while reducing the potential for human error.

3.5 Source Code Management Tools

Source Code Management (SCM) concerns how software changes are managed. Its primary goal is to accelerate the delivery of quality code changes by development teams. SCM tools enhance tracking, visibility, collaboration, and control throughout the software development lifecycle, allowing developers working on complex projects to be more creative, have more freedom, and options.

By using SCM, source files are protected against anomalies, and all teams can identify who made which changes and at what stage of the process. This promotes transparency and facilitates collaboration among team members.

We conducted a source code manager study to consolidate the choice of the GitLab tool, which is already used by Orange Business, compared to its competitors BitBucket and GitHub.

Table 3.2 summarizes the comparison of the previously presented source code managers:

We chose GitLab without hesitation due to its open-source nature and free community version. This decision was motivated by the ability to integrate continuous delivery and issue tracking functionalities, which will allow us to expand our tool usage in the future.

Tool	Open Source	License	Version Control	CD integration
	Yes	Free for community version	Git	Already integrated or through other tools
	No	Paid	Git	Integration through other tools or GitHub Actions
	Yes	Free	Git	Integration through other tools or Jira

Table 3.2: Comparison of source code management tools

3.5.1 GitLab Platform

GitLab is now a comprehensive DevOps platform presented as a single application. This platform revolutionizes development, security, operations, and collaboration practices within teams. With GitLab, it is possible to create, test, and deploy software faster, using an integrated solution. GitLab includes the following features:

- **Source Code Management:** Facilitates version control, collaboration, and accelerates the delivery of high-performance software. Enables efficient task coordination, tracking, and verification of changes, as well as simplified code reviews.
- **Agile Project Management:** Improves visibility within the organization, helping to meet set deadlines and budgets. Promotes collaboration among different teams.
- **Continuous Integration and Continuous Delivery (CI/CD):** GitLab's CI/CD feature enables the creation of high-quality applications by automating integration and deployment processes.
- **Cost-Efficient Software Deployment:** GitLab enables faster software releases with reduced development costs.

3.6 Prometheus

Figure 3.4: Prometheus

Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability in modern, dynamic environments. It is part of the Cloud Native Computing Foundation (CNCF) and is widely used for monitoring containerized applications and microservices architectures. Prometheus provides a multi-dimensional data model with time series data identified by metric name and key/value pairs. It collects metrics from configured targets at specified intervals, evaluates rule expressions, displays the results, and can trigger alerts if specified conditions are met.

3.6.1 Prometheus Architecture

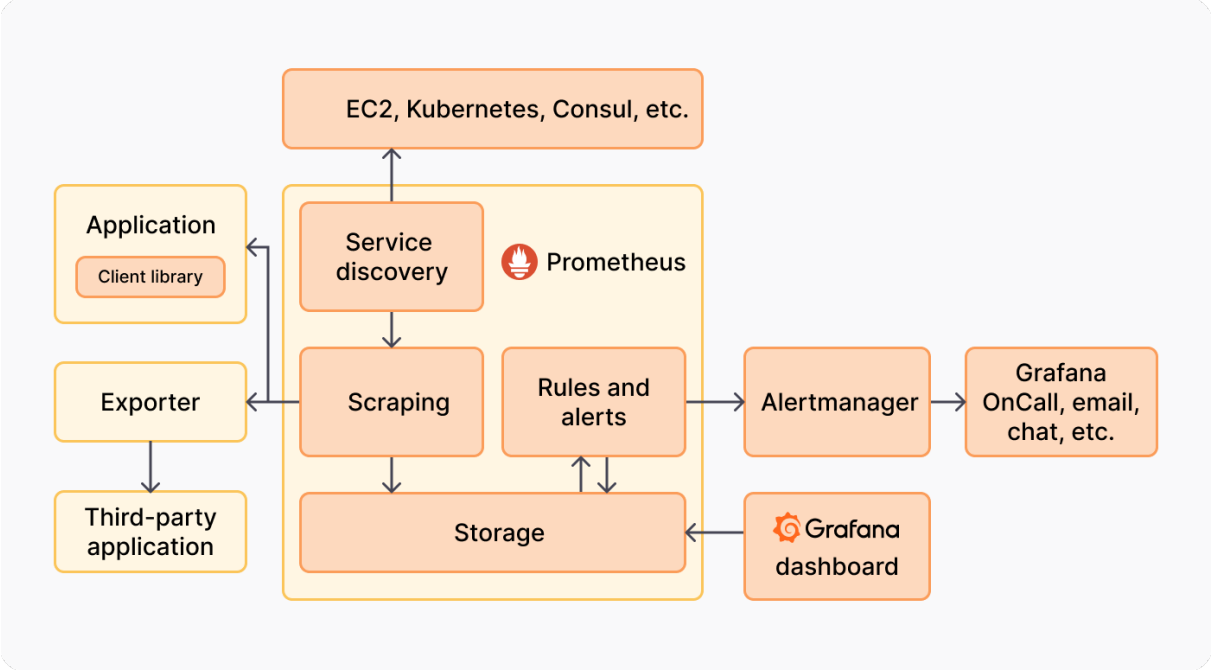


Figure 3.5: Prometheus Architecture

Figure 3.5 illustrates the Prometheus architecture. Prometheus discovers targets to scrape via service discovery, which can include instrumented applications or third-party applications accessed through exporters. Scraped data is stored and can be utilized for dashboards using PromQL or for sending alerts to the Alertmanager, which converts them into various notifications.

- **Client Libraries:** Client libraries allow for easy instrumentation of applications, typically requiring only a few lines of code. Prometheus provides official client libraries in Go, Python, Java/JVM, Ruby, and Rust, with third-party libraries available for other languages. These libraries handle thread safety, bookkeeping, and producing the Prometheus text format. They can automatically pick up metrics from dependencies and offer built-in metrics for CPU usage and garbage collection.

- **Exporters:** Exporters collect metrics from software that cannot be instrumented directly. They transform data from an application's metrics interface into the Prometheus exposition format. Exporters are widely available and can be easily extended if necessary.
- **Service Discovery:** Prometheus uses service discovery to locate and monitor applications and exporters. Integrations are available for Kubernetes, EC2, Consul, and more. Service discovery information is mapped to monitoring targets and their labels using relabeling.
- **Scraping:** Prometheus fetches metrics by sending HTTP requests called scrapes. The responses are parsed and stored. Scraping is configured to occur regularly, typically every 10 to 60 seconds per target.
- **Storage:** Prometheus stores data locally in a custom database, optimized for high performance. The storage system in Prometheus 2.0 can ingest millions of samples per second with efficient compression.
- **Dashboards:** Prometheus provides HTTP APIs for querying and displaying data. Grafana is recommended for creating advanced dashboards, supporting multiple Prometheus servers.
- **Recording Rules and Alerts:** Recording rules allow regular evaluation of PromQL expressions, storing the results. Alerting rules work similarly, generating alerts sent to the Alertmanager.
- **Alert Management:** The Alertmanager handles alerts from Prometheus servers, converting them into notifications via email, chat applications, and services like PagerDuty. Alerts can be aggregated, throttled, and routed based on different team requirements.
- **Long-Term Storage:** Prometheus stores data locally, limiting retention to available disk space. For long-term storage, remote read and write APIs enable integration with external systems for extended data retention.

3.7 Grafana

Figure 3.6: Grafana

Grafana is an open-source analytics and visualization platform commonly used alongside Prometheus for monitoring and observability purposes. Grafana provides a rich set of features for creating interactive dashboards and visualizations from various data sources, including Prometheus metrics, logs, and other time series databases.

- **Dashboards:** With Grafana, users can easily create and customize dashboards to visualize key performance indicators, monitor system health, and track application metrics in real-time.
- **Visualization Options:** Grafana supports a wide range of visualization options, including graphs, tables, heatmaps, and histograms, allowing for flexible and insightful data analysis.
- **Integration:** Grafana offers extensive integration capabilities with numerous data sources, including Prometheus, Elasticsearch, Graphite, and InfluxDB, making it a versatile tool for consolidating and visualizing data from different sources in a single interface.
- **Advanced Features:** Grafana provides advanced features such as alerting and annotations, allowing users to set up alerts based on predefined thresholds or conditions and annotate dashboards with contextual information for improved situational awareness.

3.8 Project Management Tool

With the rapid evolution of information and communication technologies, user needs are increasing and becoming more demanding, while the economic context is constantly changing. In this context, managing

IT projects has become a challenge for companies, as it is essential to master and successfully complete them, regardless of their size or type. For our DevSecOps team, which adopts the Agile Scrum method, Jira Software offers ready-to-use Scrum and Kanban boards. These boards serve as task management centers, where tasks are associated with customizable workflows. They ensure transparency of teamwork and provide visibility into the progress of each task. Time tracking features and real-time performance reports (Burnup/Burndown charts, sprint reports, velocity charts) allow the team to closely monitor its productivity over time.

Figure 3.7: Jira logo

3.9 Conclusion

Throughout this chapter, we have synthesized the selection of various tools necessary for the implementation of our project. By comparing the tools that correspond to our needs, we can detail the phases of implementing these solutions in the next chapter with the goal of achieving our objectives.

Chapter 4

Implementation

Introduction

The implementation chapter is a crucial step in our project. In this chapter, we will present in detail the various steps we followed to implement our solution. Overall, this implementation chapter will allow us to document in a detailed and transparent manner the process of executing our project. It will highlight the efforts made, the choices made, and the results obtained.

4.1 Creating a GitLab Template for Kepler

To successfully integrate Kepler with our GitLab CI pipeline and ensure a seamless, efficient, and automated workflow, we followed a series of steps that included configuring the necessary environments, creating precise configuration files, and integrating essential monitoring tools

Configuration of the Environment

The initial step was to set up the execution environment for GitLab CI. We configured the runners, which are the virtual machines or containers responsible for executing the CI jobs. It was crucial to ensure that all dependencies required to run Kepler, Prometheus, and Grafana were installed on these runners.

Creation of the CI Configuration File

We added a `template-ci-kepler.yml` file. This file plays a role as it defines the various stages of the CI pipeline and specifies the actions required to integrate Kepler into our CI process. By using this configuration file, we were able to clearly and systematically outline the tasks necessary for seamless integration into our pipeline.

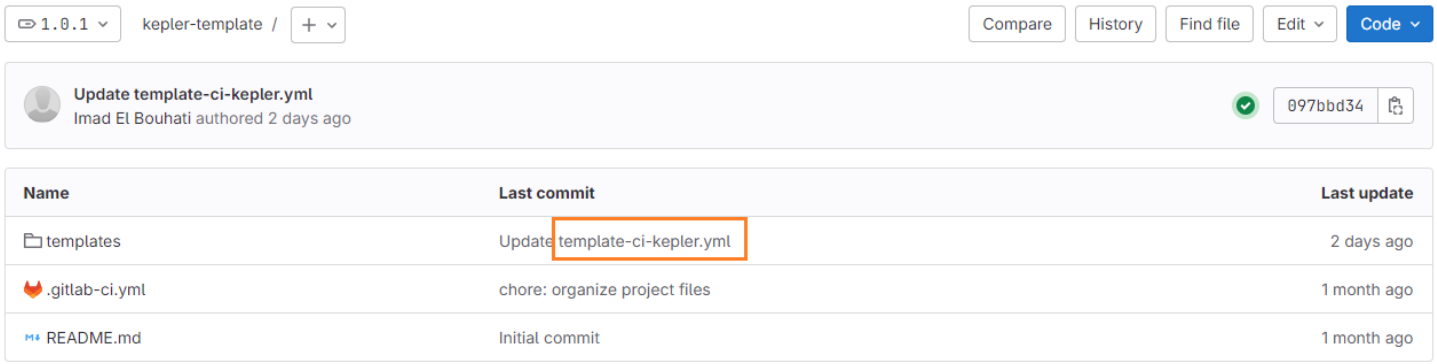


Figure 4.1: Kepler template repository

4.1.1 Integration of the Kepler Template

After creating the template, it can be used and integrated directly into a project. To achieve this, a few lines must be added to the `.gitlab-ci.yml` file.

First, we add the `include` directive followed by the path to our directory containing the Kepler template into an existing pipeline.

.gitlab-ci.yml 3.68 KiB

```yaml
include:

  # Maven template
  - project: 'to-be-continuous/maven'
    ref: '3.2.3'
    file: '/templates/gitlab-ci-maven.yml'

  # Docker template
  - project: 'to-be-continuous/docker'
    ref: '3.5.2'
    file: '/templates/gitlab-ci-docker.yml'

  # GitLeaks template
  - project: 'to-be-continuous/gitleaks'
    ref: '2.2.0'
    file: '/templates/gitlab-ci-gitleaks.yml'

  # Kepler template
  - project: 'imad.elbouhati/kepler-template'
    ref: '1.0.1'
    file: '/templates/template-ci-kepler.yml'
```

Figure 4.2: Integration of the Kepler template

Next, we specify the stage name as `"deploy"`.

```yaml
stages:
  - check
  - build
  - display
  - test
  - package-build
  - package-test
  - publish
  - package-publish
  - deploy
```

Figure 4.3: Stage `"deploy"`

We define the following variables for the deployment:

- `HELM_RELEASE_NAME` as the name of our Helm release.
- `HELM_NAMESPACE` as the namespace for our Helm deployment.
- `KUBECONFIG_CONTENT` to store cluster authentication information for kubectl.

```
deploy-kepler:
  stage: deploy
  extends: .deploy_kepler_chart
  variables:
    HELM_RELEASE_NAME: kepler
    HELM_NAMESPACE: kepler
    KUBECONFIG_CONTENT: $KUBECONFIG_CONTENT
```

Figure 4.4: Deployment variables for Kepler

This configuration allows us to include the steps for deploying Kepler in our CI/CD pipeline using Helm and Kubernetes.

4.2 Deploying Prometheus & Grafana on Kubernetes

4.3 Configuring Rules and Alerts in Prometheus

4.3.1 Setting Up Alerting Rules

4.3.2 Configuring Alerts in Prometheus

4.4 Slack Integration with Prometheus Alerts

4.4.1 Configuring Slack for Prometheus Alerts

4.4.2 Testing Slack Integration

4.5 Conclusion

Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent nec dapibus justo. Donec sagittis vulputate ante sed porttitor. Suspendisse sit amet nisl massa. Curabitur nec nisl condimentum, egestas ex vitae, dapibus enim. Etiam iaculis, erat faucibus pellentesque sagittis, nisi justo sollicitudin nibh, et condimentum augue massa non turpis. Proin commodo enim fermentum suscipit condimentum. Maecenas molestie, dui nec vestibulum rhoncus, arcu nisl faucibus neque, a ornare nisi massa ac eros. Aenean id velit sit amet lacus mattis varius. Donec fringilla massa sed nisi eleifend, a aliquet mi tempus. Nunc posuere euismod est, nec tristique augue lobortis non. Sed sodales sem ut metus tempus ullamcorper.

Chapter 5

Chapter 5 title

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent nec dapibus justo. Donec sagittis vulputate ante sed porttitor. Suspendisse sit amet nisl massa. Curabitur nec nisl condimentum, egestas ex vitae, dapibus enim. Etiam iaculis, erat faucibus pellentesque sagittis, nisi justo sollicitudin nibh, et condimentum augue massa non turpis. Proin commodo enim fermentum suscipit condimentum. Maecenas molestie, dui nec vestibulum rhoncus, arcu nisl faucibus neque, a ornare nisi massa ac eros. Aenean id velit sit amet lacus mattis varius. Donec fringilla massa sed nisi eleifend, a aliquet mi tempus. Nunc posuere euismod est, nec tristique augue lobortis non. Sed sodales sem ut metus tempus ullamcorper.

5.1 Section1

5.1.1 Subsection1

5.1.2 Subsection2

5.1.2.1 Subsubsection1

5.1.2.2 Subsubsection2

5.1.2.2.1 Paragraph a

5.1.2.2.2 Paragraph b

5.2 Section2

5.2.1 Subsection1

5.2.2 Subsection2

5.2.2.1 Subsubsection1

5.2.2.2 Subsubsection2

5.2.2.2.1 Paragraph a

5.2.2.2.2 Paragraph b

Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent nec dapibus justo. Donec sagittis vulputate ante sed porttitor. Suspendisse sit amet nisl massa. Curabitur nec nisl condimentum, egestas ex vitae, dapibus enim. Etiam iaculis, erat faucibus pellentesque sagittis, nisi justo sollicitudin nibh, et condimentum augue massa non turpis. Proin commodo enim fermentum suscipit condimentum. Maecenas molestie, dui nec vestibulum rhoncus, arcu nisl faucibus neque, a ornare nisi massa ac eros. Aenean id velit sit amet lacus mattis varius. Donec fringilla massa sed nisi eleifend, a aliquet mi tempus. Nunc posuere euismod est, nec tristique augue lobortis non. Sed sodales sem ut metus tempus ullamcorper.

General Conclusion and Perspectives

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent nec dapibus justo. Donec sagittis vulputate ante sed porttitor. Suspendisse sit amet nisl massa. Curabitur nec nisl condimentum, egestas ex vitae, dapibus enim. Etiam iaculis, erat faucibus pellentesque sagittis, nisi justo sollicitudin nibh, et condimentum augue massa non turpis. Proin commodo enim fermentum suscipit condimentum. Maecenas molestie, dui nec vestibulum rhoncus, arcu nisl faucibus neque, a ornare nisi massa ac eros. Aenean id velit sit amet lacus mattis varius. Donec fringilla massa sed nisi eleifend, a aliquet mi tempus. Nunc posuere euismod est, nec tristique augue lobortis non. Sed sodales sem ut metus tempus ullamcorper.

Nullam fermentum id mauris suscipit varius. Quisque tristique tempor fringilla. Nam porta tincidunt orci in consectetur. Suspendisse nec sem nisi. Suspendisse potenti. Sed sodales aliquam libero, at dapibus ligula accumsan non. Quisque congue et urna a consectetur. Nullam maximus venenatis ornare. Donec in luctus urna, vel rhoncus lectus. Etiam lobortis, lacus nec gravida iaculis, magna nulla iaculis leo, quis congue nunc tortor pellentesque lorem. Sed fermentum vulputate dui, ac malesuada eros molestie sit amet. Proin fringilla elit justo, in posuere urna porttitor et. Curabitur dictum justo vitae metus pellentesque, eget feugiat sem feugiat.

Curabitur in mauris eu felis cursus auctor eget ut massa. Curabitur eleifend consectetur magna in ultrices. Etiam ut semper turpis. Morbi sed ipsum nec sem rutrum blandit sit amet vitae felis. Vestibulum vitae hendrerit diam. Aliquam pellentesque est mi, et tempus nisl laoreet in. Maecenas consequat augue a ante congue dignissim. In condimentum erat in volutpat tempus. Mauris vulputate, enim ut dignissim varius, lorem purus tristique sapien, lobortis accumsan dolor lorem quis eros. Duis vel imperdiet metus, non suscipit tortor.

Appendix A

Glossary

Telnet: Telnet is a network protocol that allows users to establish a remote terminal connection to a computer or network device over a network.

SSH (Secure Shell): SSH is a network protocol used for secure remote administration and secure file transfers.

SNMP (Simple Network Management Protocol): SNMP is a network protocol designed for managing and monitoring network devices and systems.

Appendix B

Acronyms

IP	Internet Protocol
CNS	Cloud and Network Services
MN	Mobile Networks
NI	Network Infrastructure
IoT	Internet of Things
NFV	Network Functions Virtualization
SDN	Software-Defined Network
TCP	Transmission Control Protocol
7750 SR	Nokia 7750 Service Router
OSPF	Open short Path First
BGP	Border Gateway Protocol
VPN	Virtual Private Network
MPLS	Multiprotocol Label Switching
EVE-NG	Emulated Virtual Environment - Next Generation
VM	Virtual Machine
SSH	Secure Shell
WinSCP	Windows Secure Copy
FTP	File Transfer Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
FTPS	FTP over SSL/TLS

SFTP	Secure File Transfer Protocol
SCP	Secure Copy Protocol
CLI	Command-line interface
MD-CLI	Model-Driven Command Line
API	Application Programming Interface
re	Regular Expression
IGP	Internal Gateway Protocol
IS-IS	Intermediate-System Intermediate-System
SPF	Shortest Path First
SF	Switch Fabric
CPM	Control Processing Module
IOM	Input/Output Modules
MDA	Media Dependent Adapters
QoS	Quality of Service
IPsec	Internet Protocol Security
VSR	Virtual Services Router
DDoS	Distributed Denial-of-Service
SROS	Service Router Operating System
NETCONF	Network Configuration Protocol
YANG	Yet Another Next Generation
MP-BGP	Multiprotocol Border Gateway Protocol
IPv6	Internet Protocol Version 6
AS	Autonomous Systems
iBGP	Internal Border Gateway Protocol
eBGP	External Border Gateway Protocol
LAG	Link Aggregation Group
VPRN	Virtual Private Routed Network
SNMP	Simple Network Management Protocol
Telnet	Telecommunication Network
YAML	Yet Another Markup Language
XML	eXtensible Markup Language

Appendix C

Some subject you want to expand on

You can add more appendices depending on the subjects.

```
[root@host ~]# You can change the language and caption.
```

Listing C.1: Bash example

```
# Solve the quadratic equation ax**2 + bx + c = 0

# import complex math module
import cmath

a = 1
b = 5
c = 6

# calculate the discriminant
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

```
20
```

Listing C.2: Python example

Column1	Column2
Element11	Element21
Element12	Element22
Element13	Element23

Table C.1: Table Example

Table C.2: Long table Example

Cell	Description
Element11	Element21
Element12	Element22
Element13	Element23
Element14	Element24
Element15	Element25
Element16	Element26
Element17	Element27
Element18	Element28
Element19	Element29
Element110	Element210
Element111	Element211
Element112	Element212
Element113	Element213
Element114	Element214

Bibliography

[1] Author name, Book name.

[2] *Title 1*, **Title 2**