.

Dedication

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Dedication

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

.

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

.

Abstract (TL;DR)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Table of Contents

List of Figures

List of Tables

End of Studies Project

Anis, Benna
anis.benna@incedo.com

Firas, Jaber
firas.jaber@incedo.com

March 2022

General Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Chapter 1

Context of the work

Introduction

This chapter introduces the general context of this report. We start by presenting the frame of the project as well as the host company. Then comes the enumeration of the problems which led to the realization of the project. We wrap it up by defining the methodology we've followed to carry out our work.

1.1 General framework of the internship

This project was carried out within the frame of obtaining a bachelor's degree in Computer Science at the Higher Institute of Technological studies of Nabeul. The internship took place fully remotely at Incedo Services GmbH for five months starting from the 1st of February 2022 to the 30th of June 2022 with the purpose of further developing an existing project of the company as well as migrating it to a micro-services architecture deployed on a self-managed instance of Kubernetes.

1.2 Company overview

This section introduces the host company **Incedo Services GmbH** and as well as the services it offers.

1.2.1 About Incedo

"We are a young software development and consulting firm located in Stuttgart. We help our clients to develop exciting digital products and solutions and we also love to bring our own ideas into life from time to time." [7]

Figure 1: Logo of Incedo Services GmbH

1.2.2 Incedo's services

Consulting

Incedo helps its clients overcome two main challenges:

- Develop a product that somebody actually needs and that creates enough value to form a profitable business case.
- Ensuring that (large) IT-projects are finished in time & budget with a result that satisfies the actual users/clients of the developed software solution.

Development

Incedo develops software for clients, as well as for the company itself.

Figure 2: Selection of Incedo clients

1.3 Stating the problem

Incedo -having ambitious goals to grow over the next 2 to 4 years- wants to win new clients and strategically develop the existing ones. Winning new clients starts with generating new leads. Previously, Incedo has worked with an Austrian start-up (motion group) that provides automated lead generation through LinkedIn at the cost of 0.85 € per requested contact. Although one big project was closed and several leads were generated, Incedo started using the LinkedIn Sales Navigator with a more targeted approach, but nevertheless wanted to automate the approach and increase its outreach. Having launched the first version of the ILG (Incedo Lead Generator) as a SaaS (Software as Service), Incedo was satisfied for a while. Over the time however, as more and more clients were interested in the ILG, the current architecture couldn't handle the load properly. Therefore, it was our task to re-design and implement a new scalable architecture instead of the old one.

1.4 Assessment of the case

1.4.1 Describing the work procedure

The work on any project must first of all be preceded by a thorough study of the existing ones which undermines the strengths and weaknesses of the current system, as well as the business decisions that should be taken into account during the conception as well as the realization.

1.4.2 Criticizing the current state

After studying the existing, we can determine its limitations:

- Bugs always tend to happen whenever the LinkedIn website changes (due to scraping).
- Since cron jobs are running for the whole day, bugs are hard to respond to fast enough because we can only deploy once at the end of day.
- It is hard to test the whole workflow because of how the app works (looking for certain changes in the LinkedIn interface after certain buttons are clicked for example) meaning that the dev environment is lacking.
- It cannot scale well enough since the whole application is deployed on a single server.

1.4.3 Proposed solution

The solution to these problems is refactoring the whole application to separate sub applications (microservices) where the automation and scraping processes can be scaled independently of the other parts of the application. This way, it will be easier to maintain and scale the different codebases as well as respond faster to bugs and know exactly what caused them in the first place.

1.5 Development Methodology

1.5.1 Agile methodology

Agile is a structured and iterative approach to project management and product development. It recognizes the volatility of product development, and provides a methodology for self-organizing teams to respond to change without going off the rails.

1.5.2 Scrum methodology

Scrum teams commit to completing an increment of work, which is potentially shippable, through set intervals called sprints. Their goal is to create learning loops to quickly gather and integrate customer feedback. Scrum teams adopt specific roles, create special artifacts, and hold regular ceremonies to keep things moving forward.

1.5.3 Kanban methodology

Kanban is all about visualizing your work, limiting work in progress, and maximizing efficiency (or flow). Kanban teams focus on reducing the time a project takes (or user story) from start to finish. They do this by using a kanban board and continuously improving their flow of work.

1.5.4 The choice for ILG

Kanban is based on a continuous workflow structure that keeps teams nimble and ready to adapt to changing priorities. Work items—represented by cards— are organized on a kanban board where they flow from one stage of the workflow(column) to the next. Common workflow stages are To Do, In Progress, In Review, Blocked, and Done. And since this project is very susceptible to changes from outside (LinkedIn), Kanban offered more flexibility than Scrum so that's why the team went with it instead.

Conclusion

In this chapter, we presented not only the host organization but also the general context of the project and why it's needed. We then criticized the current state of the ILG application and discussed the possible development methodologies. Lastly, we picked the most ideal choice for us which is Kanban.

Chapter 2

State of the art

Introduction

This chapter will present and study various concepts such as browser automation, web scraping, version control and DevOps with an emphasis on key DevOps terminologies. We will talk about the most used tools in the market as well as which ones we settled on after making a comparison.

2.1 Automation and Web Scraping concepts

2.1.1 Automation

At its core, automation is leaving menial and recurring tasks to the automaton (or machine) in order to do more meaningful work as a human in the meantime. Implementing automation improves the efficiency, reliability and the speed of tasks that previously took humans a lot of time.

2.1.2 Web Scraping

Web scraping is the process of automatically extracting content and data from a website. Although data extraction is done in a brute way by reading texts from HTML elements, it is used in a variety of legitimate digital businesses like search engines, price comparison sites and market research companies. So contrary to what some might think, web scraping is completely legal.

2.1.3 Relationship between the two

Web scraping simplifies the process of extracting data and the automation process helps repeating the recurring task of extraction. As a result, the combination of scraping data, storing it and automating the whole process is getting very popular, especially with new technologies and tools arising in order to do just that.

2.2 DevOps

2.2.1 Definition

DevOps is the set of practices, techniques and tools used to speed up the Software Development Lifecycle (SDL) by bringing together two historically separate functions of development and operations.

Development refers to writing code and software, whereas operations refer to provisioning servers, configuring them as well as deploying the apps to them, amongst other things.

DevOps teams focus on automating all of the above. The key terminologies around DevOps are Continuous Integration, Continuous Delivery and Infrastructure As Code and automation.

2.2.2 Lifecycle

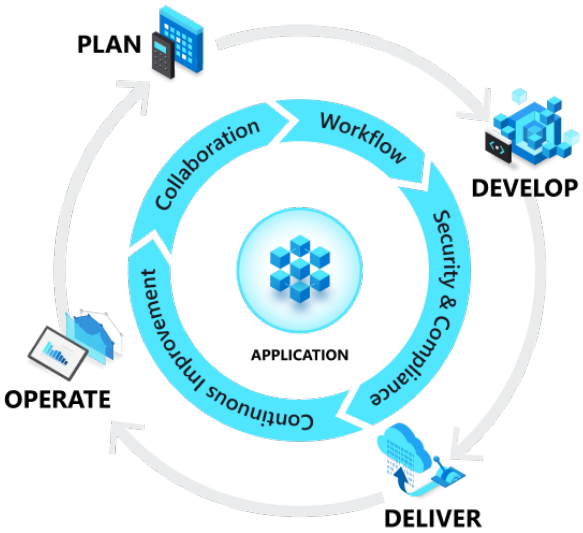


Figure 3: DevOps and the application lifecycle

DevOps -according to Azure- is the main influencer of the application lifecycle throughout its plan, develop, deliver, and operate phases. All of the phaes rely on one another and they are not role-specific. In a true DevOps culture, each role is involved in each phase to some extent. [1]

- **Plan:** In this first stage, DevOps teams define and describe the main features of the system being created. They accomplish that by many ways. Some of which are creating backlogs, tracking bugs, using Kanban boards and visualizing progress with dashboards.
- **Develop:** This step includes all aspects of coding, testing, reviewing and code integration. It also includes creating build artifacts that can be deployed into various environments. Automation of mundane and manual tasks plays a heavy role in this phase through automated testing and continuous integration.

- **Deliver:** This is the process of deploying application into various environments such as staging or production consistently and reliably. It also includes deploying and configuring the infrastructure that makes up the environments. The DevOps team defines a release management process with automated gates or checkpoints the move the applications between stages in order to delploy in an efficient, scalable, repeatable and controllable way.

- **Operate:** The operate phase involves maintaining, monitoring and troubleshooting the application in the production environment. The purpose of this phase is ensuring high availability, system reliability and zero downtime. This requires alerting and full visibility into the application.

2.2.3 Container management

Containers have become an integral part of DevOps over the past couple of years but what is a container exactly and how do we manage them?

What is a container ?

Simply said; container is a packaging format for software applications that are akin to a very lightweight virtual machine which always executes in an isolated environment [2]. What this implies is that said containers can be easily copied to and run on different machines with high reliability, lower costs and high efficiency.

What is container management ?

Container management is the process for automating the creation, deployment and scaling of containers [4]. Container management tools such as Docker and Podman facilitate the addition, replacement and organization of containers.

2.2.4 CI/CD pipelines

A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation. [10]

2.3 Microservices

2.3.1 Definition

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs. These services are owned by small, self-contained teams. [14]

Microservices architectures make applications easier to scale and faster to develop, enabling innovation and accelerating time-to-market for new features.

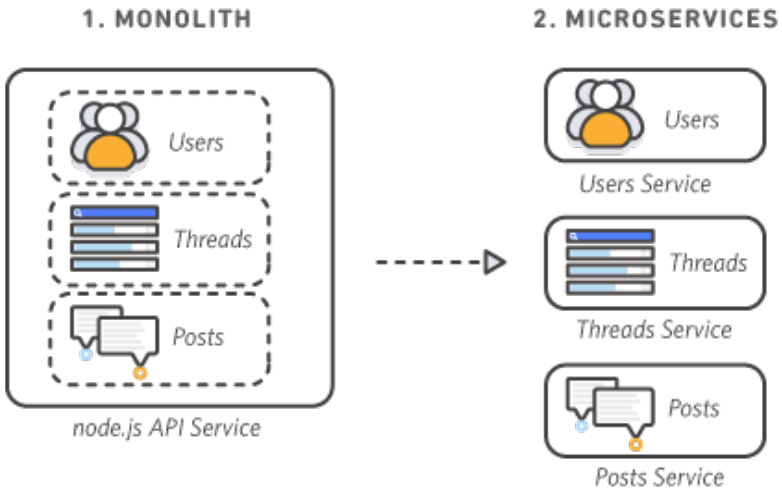


Figure 4: Breaking a monolithic application into microservices

2.3.2 Characteristics of Microservices

- **Autonomous :** Each component service in a microservices architecture can be developed, deployed, operated, and scaled without affecting the functioning of other services. Services do not need to share any of their code or implementation with other services. Any communication between individual components happens via well-defined APIs.
- **Specialized :** Each service is designed for a set of capabilities and focuses on solving a specific problem. If developers contribute more code to a service over time and the service becomes complex, it can be broken into smaller services.

2.3.3 Benefits of Microservices

- **Agility :** Microservices foster an organization of small, independent teams that take ownership of their services. Teams act within a small and well understood context, and are empowered to work more independently and more quickly.

- **Flexible Scaling :** Microservices allow each service to be independently scaled to meet demand for the application feature it supports. This enables teams to right-size infrastructure needs, accurately measure the cost of a feature, and maintain availability if a service experiences a spike in demand.

- **Easy Deployment :** Microservices enable continuous integration and continuous delivery, making it easy to try out new ideas and to roll back if something doesn't work. The low cost of failure enables experimentation, makes it easier to update code, and accelerates time-to-market for new features.

- **Technical Freedom :** Microservices architectures don't follow a "one size fits all" approach. Teams have the freedom to choose the best tool to solve their specific problems. As a consequence, teams building microservices can choose the best tool for each job.

- **Reusable Code :** Dividing software into small, well-defined modules enables teams to use functions for multiple purposes. A service written for a certain function can be used as a building block for another feature. This allows an application to bootstrap off itself, as developers can create new capabilities without writing code from scratch.

- **Resilience :** Service independence increases an application's resistance to failure. In a monolithic architecture, if a single component fails, it can cause the entire application to fail. With microservices, applications handle total service failure by degrading functionality and not crashing the entire application.

2.4 Comparative Analysis

In order to get started with our project, we need a wide range of tools that deal with the following areas; version control, orchestration between containers, browser automation as well as scheduled automation. For that, we did a comparative study on some of the tools the market provides.

2.4.1 Version Control: Git vs SVN

The most used tools for version control are Git and SVN. Here is a table comparing both tools.

Table 1: Comparative study between Git and SVN

	Description	Advantages
Git	Git is a distributed version control system which means that when coloning a repository, you get a copy of the entire history of that project.	Git has what's called a staging area. This means that even if you made over a 100 changes, they can be broken down to 10 commits each with their own comments and description.
SVN	SVN or Subversion is a centralized version control system. Meaning that there is always a single version of the repository that you checkout.	SVN has one central repository – which makes it easier for managers to have more of a top down approach to control, security, permissions, mirrors, and dumps.

At the end, both are great options. However, we will be using Git which is the VCS the company uses.

2.4.2 Container management: Docker vs Podman

Although Docker is widely popular, Podman is also taking its share from the market. Especially on Redhat systems.

Table 2: Comparative study between Docker and Podman

Aspect	Docker	Podman
Definition	Docker and Podman are both container management technologies used to build container images and store said images in a registry to then run them as containers in a target environment.	
Technology	Docker uses the containerd daemon which does the pulling of images then hands over the creation process to a low-level runtime named runc.	Podman uses a daemon-less approach using a technology named conmon which does the heavy lifting. It also delegates the container creation to a low-level container runtime.
Specificity	Docker Desktop is a great feature for Docker which provides an easy way to build and distribute containers amongst developers.	The smallest unit in Podman is the pod. A pod is the organizational unit for containers and is directly compatible with Kubernetes.

As a result, we will also be sticking to Docker in this project.

2.4.3 Browser automation: Puppeteer vs Playwright

Both are Node.js libraries for browser automation. The table below compares these two on different levels.

Table 3: Puppeteer vs Playwright highlights

Category	Puppeteer	Playwright
Overview	Puppeteer makes it easy to get started with browser automation. This is in part because of how it interfaces with the browser.	Playwright is very similar to Puppeteer in many respects. The API methods are identical in most cases, and Playwright also bundles compatible browsers by default.
Community	Has a large community with lots of active projects.	Small but active community.

Since the difference is pretty minor, we will be sticking to the more recent Playwright.

2.4.4 Database: MongoDB vs MySQL

The old architecture of ILG uses MySQL but we have the choice to use a different database as well, so we did the following study.

Table 4: Comparative study between MongoDB and MySQL [6]

Category	MongoDB	MySQL
Overview	An open-source NoSQL that stores data in JSON-like documents.	An open-source relational database system.
Storage	Stores data in documents that belong to a particular class or group.	Data is stored in tables where each table corresponds to an entity.
Type	NoSQL, meaning that data in a collection can have different structures.	Uses SQL (Standard Query Language) meaning that the schema of data cannot be changed once defined.

Since the old architecture uses MySQL, we will be sticking to that in the new architecture as well for an easier migration.

2.4.5 Database: GraphQL vs Rest

The most used tools for version control are Git and SVN. Here is a table comparing both tools.

Table 5: Comparative study between GraphQL and REST [5]

	Description	In depth
GraphQL	An open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data.	Describe only what's needed in queries, so no under or over fetching. It is also Schema and Type safe so it's less prone to bugs.
REST	REST (Representational State Transfer) is an architectural style that conforms to a set of constraints when developing web services.	It was introduced as a successor to SOAP APIs, follows the REST standards and is not constrained to XML.

Since the old architecture uses MySQL, we will be sticking to it in the new architecture.

2.4.6 Deployment: Docker Swarm vs Kubernetes

The old deployment uses Docker Compose, but we had to to a comparative study instead of the available tools on the market since docker-compose is not very optimized for production. Our research led us to both Docker Swarm and Kubernetes.

Table 6: Comparative study between Docker Swarm and Kubernetes

Aspect	Docker Swarm	Kubernetes
Overview	Docker Swarm is a lightweight, easy-to-use orchestration tool with limited offerings compared to Kubernetes. In contrast, Kubernetes is complex but powerful and it provides self-healing and auto-scaling capabilities out of the box.	
Advantages	• Straightforward to install • Takes less time to learn • Works with the Docker CLI	• Can sustain and manage large architectures and complex workloads • Has a self-healing capacity that supports automatic scaling. • Supports every operating system. • Has a large open-source community with Google's backup. • The most popular distributed system orchestrator in the world.
Disadvantages	It has been abandoned by Docker Inc. and is no longer maintained.	It has a steep learning curve and requires separate CLI tools.

Therefore, we will be deploying our application stack to Kubernetes since the potential is huge compared to the deprecated Docker Swarm.

2.4.7 Deployment management: Kubectl vs Helm

Since we're using Kubernetes, we have the option to deploy our microservices either with kubectl or with helm.

Table 7: Comparative study between Kubectl and Helm

Aspect	Kubectl	Helm
Overview	Kubectl is the official Kubernetes command-line tool that allows running commands against Kubernetes clusters.	Helm -in a nutshell- is the package manager for Kubernetes.
In depth	Kubectl can be used to deploy applications, inspect and manage cluster resources and also view logs. It also provides many other advanced functionality such as node tainting or setting up strategies for load balancing.	Helm helps in defining, installing and upgrading the most complex Kubernetes applications. Helm templates also provide a very clean way to avoid code duplication between similar resources in applications, such as services or deployments.

Thouroughly studying our use case, we agreed to create a uniform Helm package to ship our microservices. We also deemed it necessary to have some common configuration that we simply apply with kubectl such as Ingress rules for convenience.

Conclusion

In this chapter, we discussed the main concepts relevant to our project such as web scraping, automation and DevOps. Then we talked about some of the tools in the market and discussed some of our choices.

Chapter 3

Analysis and specification of requirements

Introduction

In this chapter, we are going to analyze the specification and the requirements of the application, mostly this chapter will contains the product main features and their conception so we will have our expectation for the end result.

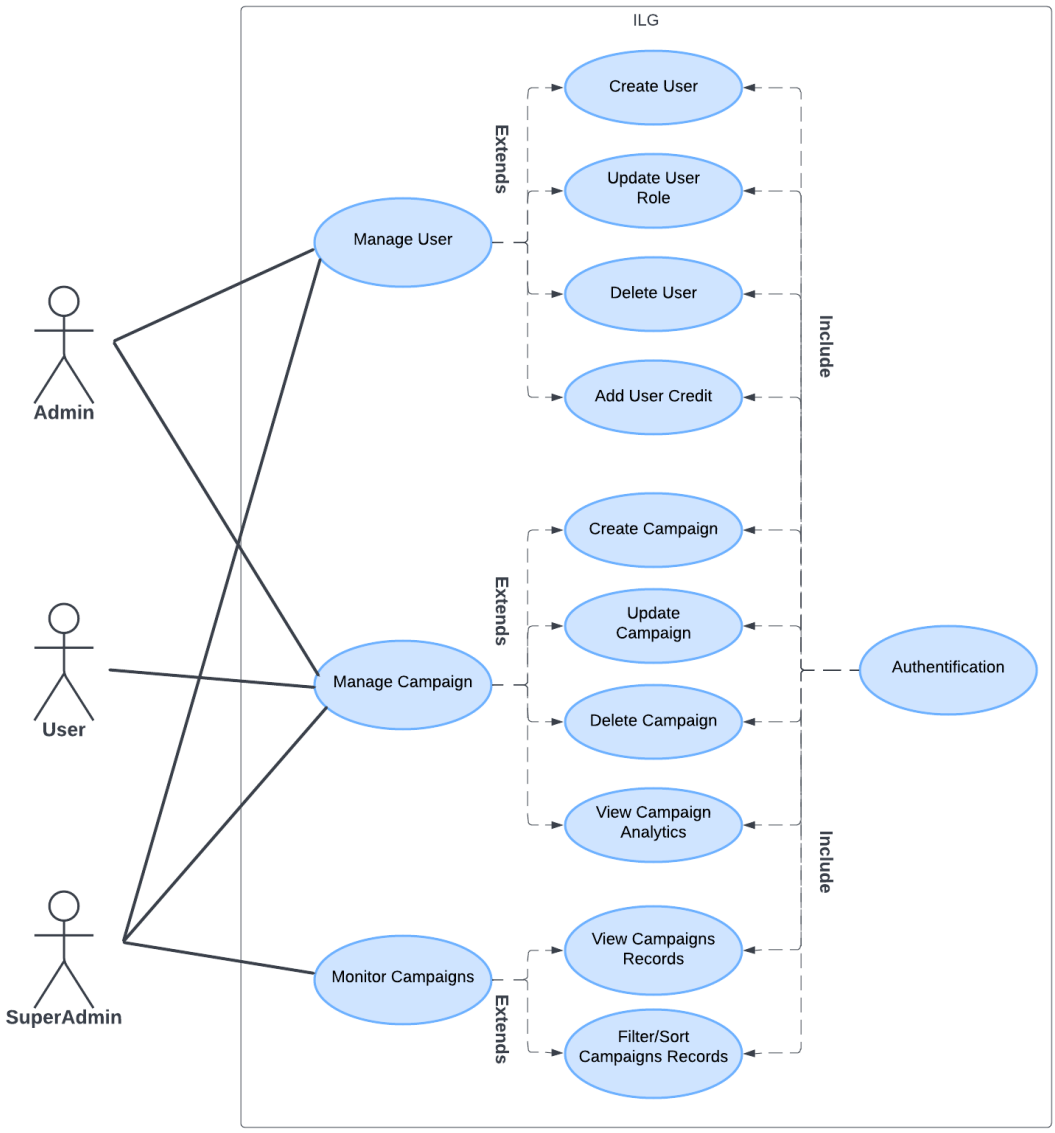
3.1 Requirements

Our requirements are divided into two parts, the clients portal of the application called Dashboard, and the lead generating and handling happening in the background.

3.1.1 Dashboard

3.1.1.1 Requirements

Here is a global use case diagram that specifies the dashboard requirements :



3.1.1.2 Actors

Mainly, there are 3 types of actors of the system:

- **Super Admin**: the main administrator of the system.
- **Admin**: the administrator of the system but with lower priviledges then the SuperAdmin.
- **User**: the main user of the system, who can benefits from the different features the system offers, usually a represontative from the client.

3.1.1.3 User management

Here are the main requirements of user management for different type of actors:

- As a **Super Admin**, i can create, modify or delete users and admins accounts.
- As an **Admin**, i can create, modify or delete user's accounts (clients).
- As an **Admin**, i can add or remove credit from user's accounts (clients).
- As an **Admin**, i can view my user's accounts, their used and bought credits.

3.1.1.4 Campaign management

Here are the main requirements of user management for different type of actors:

- As a **User**, i can create my campaigns specifiying the LinkedIn account credentials to be used, the campaign list, salutatiom messages, invite messages, follow up messages and the number of invites per day.
- As a **User**, i can update my campaigns details such as the LinkedIn account credentials, the campaign list, salutatiom messages, invite messages, follow up messages and the number of invites per day.
- As a **User**, i can delete my campaigns.
- As a **User**, i can activate my campaigns.
- As a **User**, i can stop my campaigns.
- As a **User**, i can export informations about the scraped leads as a CSV file.

3.1.1.5 Campaign monitoring

- As a **Super Admin**, i can view all the campaigns records, on day by day basis or in a specific range of dates with informations such as number of invites sent, number of follow ups sent and the number of withdrawn invites.
- As a **User**, i can view my campaigns analytics with informations such as requested, connected, replied leads, and convertion rates.

3.1.2 Lead generating

3.1.2.1 Requirements

The lead generating is the process of **scraping** the LinkedIn accounts and **generating** leads by sending theme customized invite and follow ups messages depending on the user's campaign preferences. So here is the requirements of both parts :

3.1.2.2 Scraping leads

- Each day, the system generate a defined number leads profiles from a campaign (search) list.
- Each day, Scrape and save a defined number of leads profiles to get all their available professional informations.

3.1.2.3 Generating leads

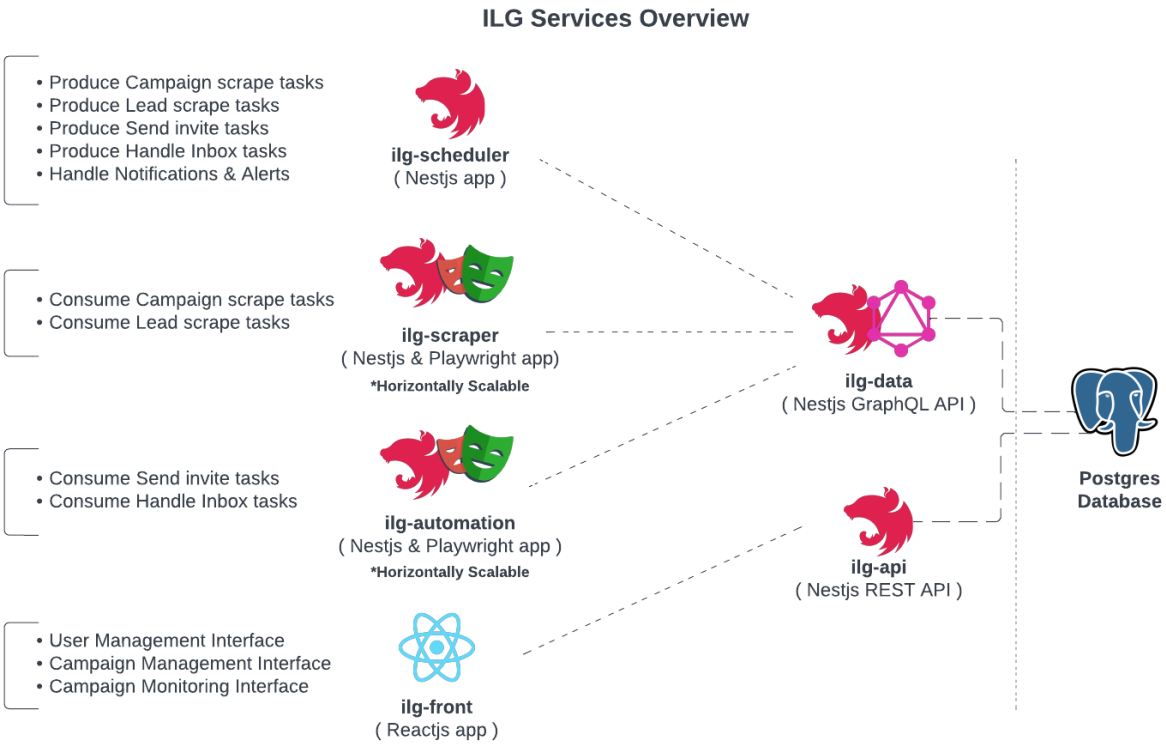
- Each day, The system sends a defined number from the user preference range of connection invites to the scraped leads.
- The system sends out first follow up message to leads that accepted the connection request and did not yet reply, the first follow up message is sent the earliest 24 hours after the connection request has been sent.
- Second follow up message is sent out on the day after the first follow up message was sent if the lead stil not yet replied.

3.2 Conception

In this section, we will dive deep into the conception and the logic behind the parts of the application :

3.2.1 Architecture

Here is an overview of the different services of the application before we dive into the details of each service :



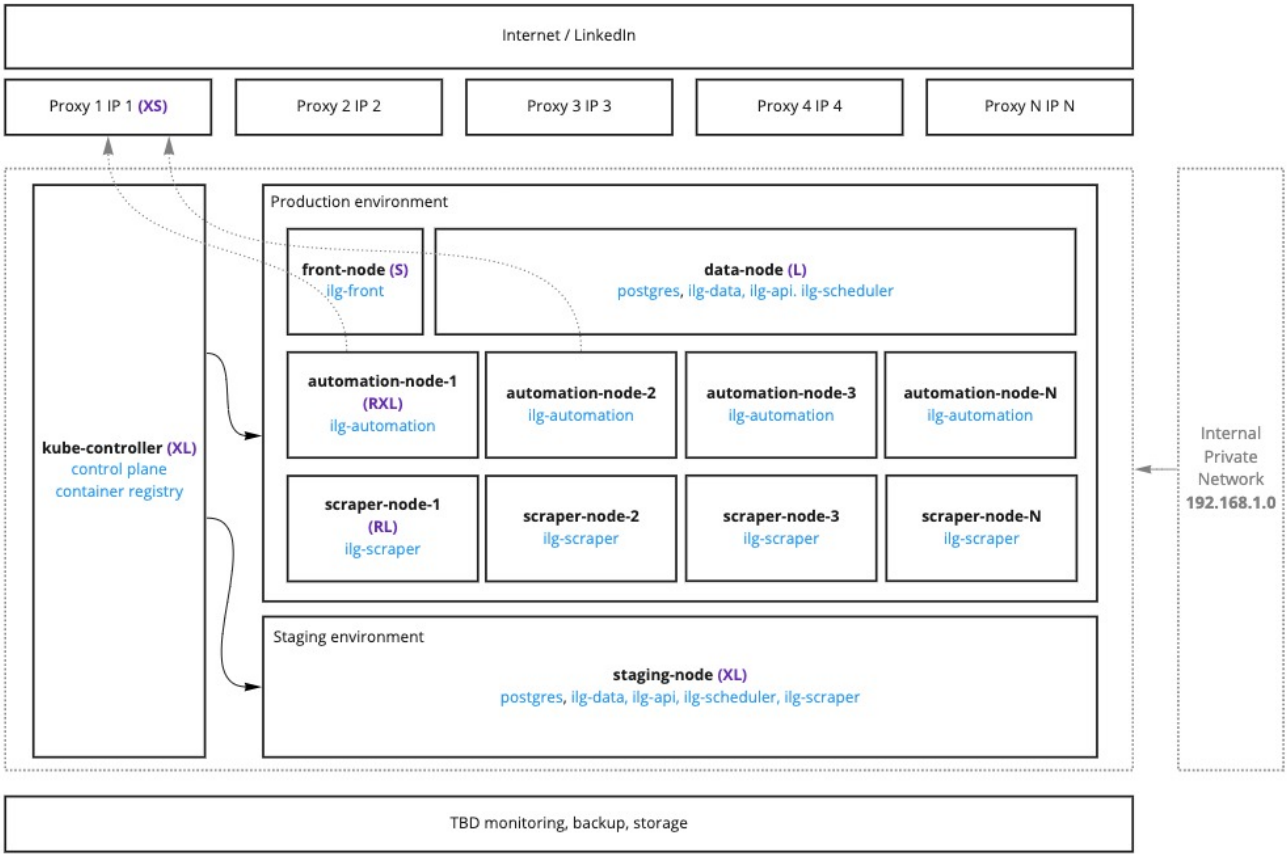
3.2.1.1 Microservices

Table 8: All the application's Microservices

Name	Description	Scalable
ilg-data	GraphQL Database access layer used to feed all needed data for other services from the Postgres DB.	No
ilg-api	RESTful api to serve data to the React app dashboard.	No
ilg-front	a React app dashboard application served throw NGINX.	No
ilg-scheduler	Service thats responsible for scheduling and orchestrating the tasks queues and notifications.	No
ilg-scraper	Service thats responsible for scraping leads of campaigns links from LinkedIn Sales Navigator.	Yes
ilg-automation	Service thats responsible for handling LinkedIn accounts inboxs/threads and sendings invites to leads.	Yes

3.2.1.2 Cloud Infrastructure

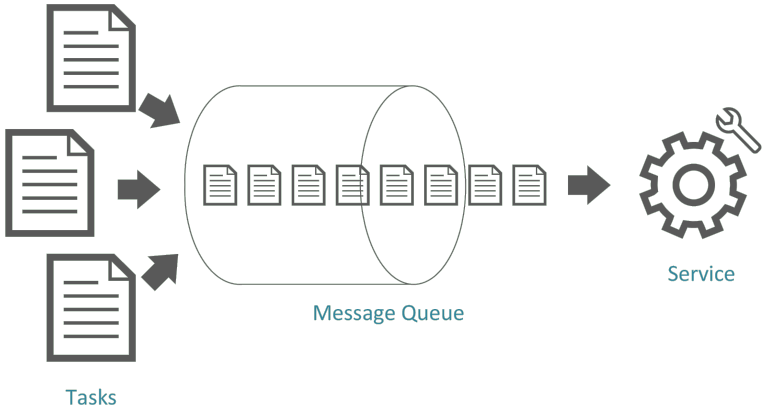
Since we are using a Kubernetes Cluster for our Cloud Infrastructure, we decided to distribute our different microservices into their corresponding nodes (VMs) depending on their shared components and scalabity. So here is an overview of the current Cloud Infrastructure of the application :



3.2.2 Tasks & Queues

3.2.2.1 Thoery of tasks and queues

Task queues let applications perform work, called tasks, asynchronously outside of a user request. If an app needs to execute work in the background, it adds tasks to task queues. The tasks are executed later, by worker services. The Task Queue service is designed for asynchronous work. All the lead generating features are implemented as a task queue.



3.2.2.2 Tasks queues types

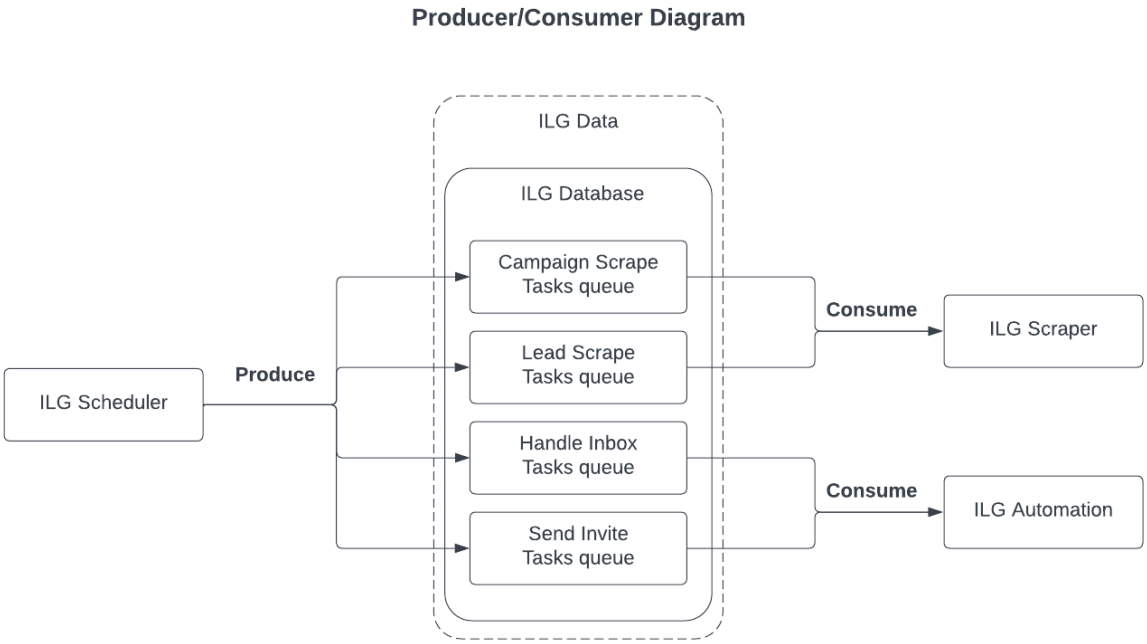
Here are the different types of the system tasks :

Table 9: All the application's tasks/jobs

Name	Description	Producer	Consumer
campaign-scrape	Scrape list of leads from campaign search list	ilg-scheduler	ilg-scraper
lead-scrape	Scrape lead information and save them in the database	ilg-scraper	ilg-scraper
handle-inbox	Hanle conversation threads of leads and send follow ups if needed	ilg-scheduler	ilg-automation
send-invite	Send connection invite to leads	ilg-scheduler	ilg-automation

3.2.2.3 Scheduling tasks

Scheduling the different types of tasks is mainly done by the ilg-scheduler service periodically on daily basis. and we use our database thats acceciabble by ilg-data, as the store of those task queues with all of their corresponding informations and state.

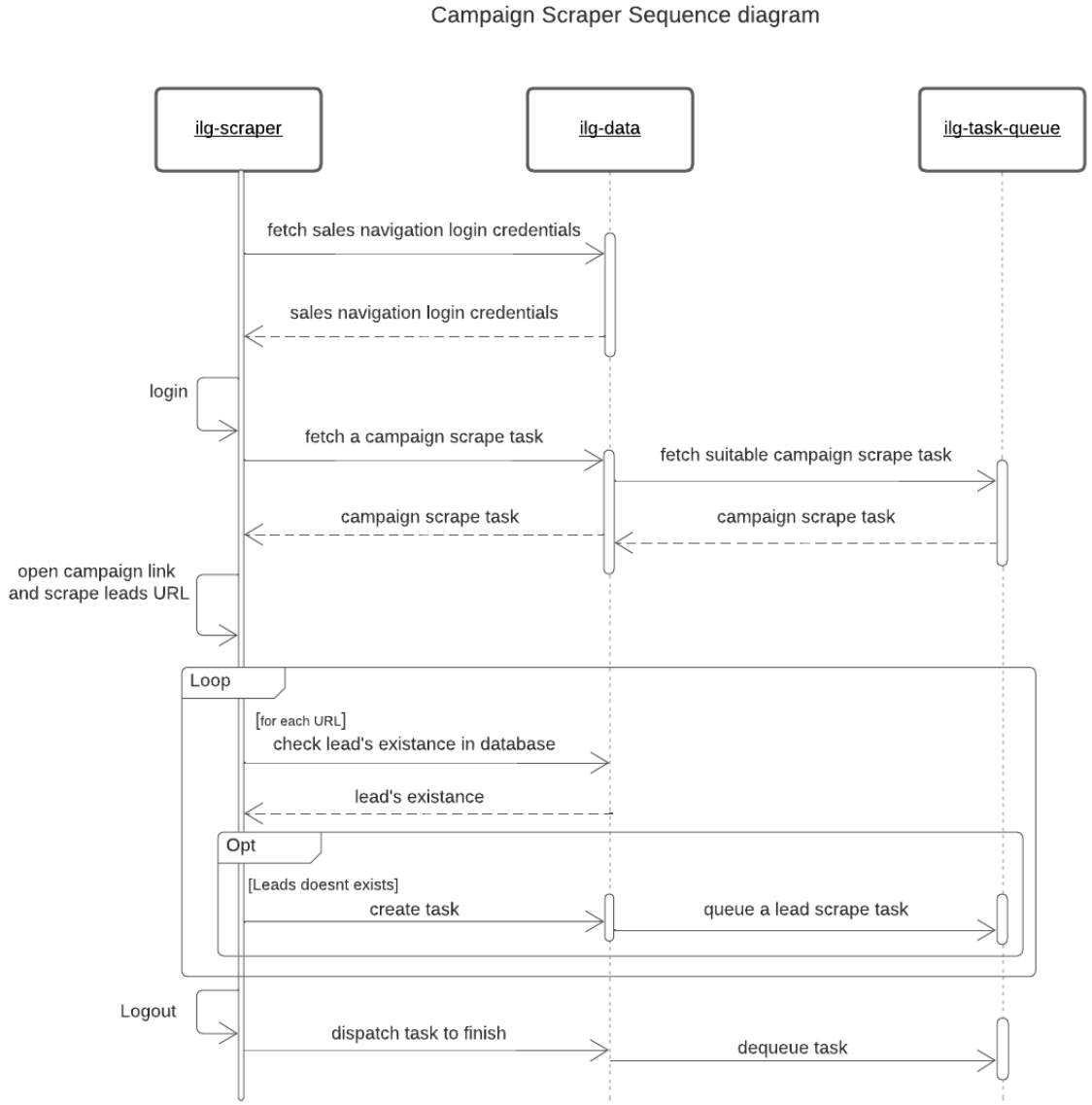


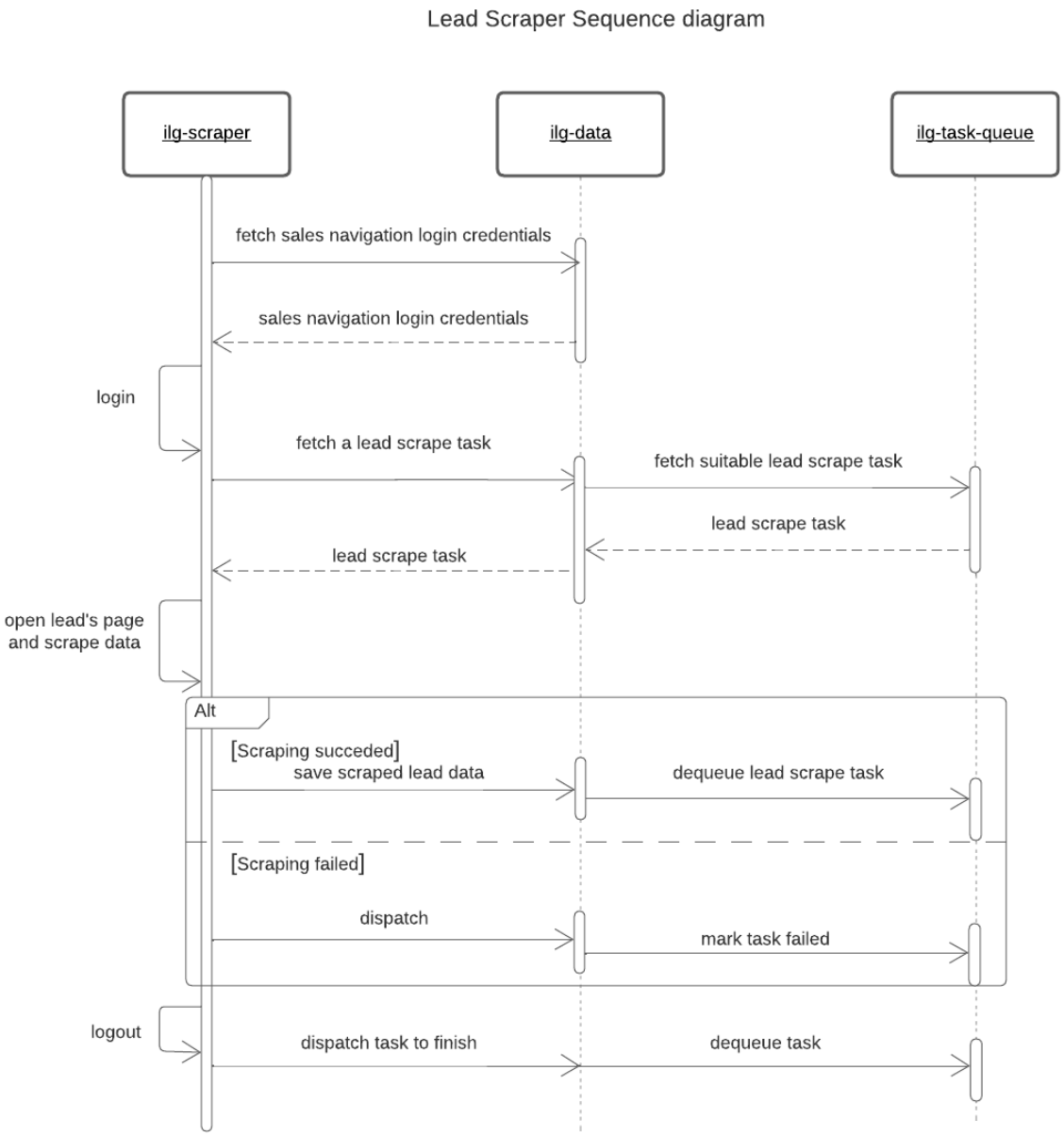
3.2.3 Scraper & Automation

Both the scraper and the automation services are responsible for consuming the tasks of the system on daily basis for generating the leads of a campaign.

3.2.3.1 Scraper flow

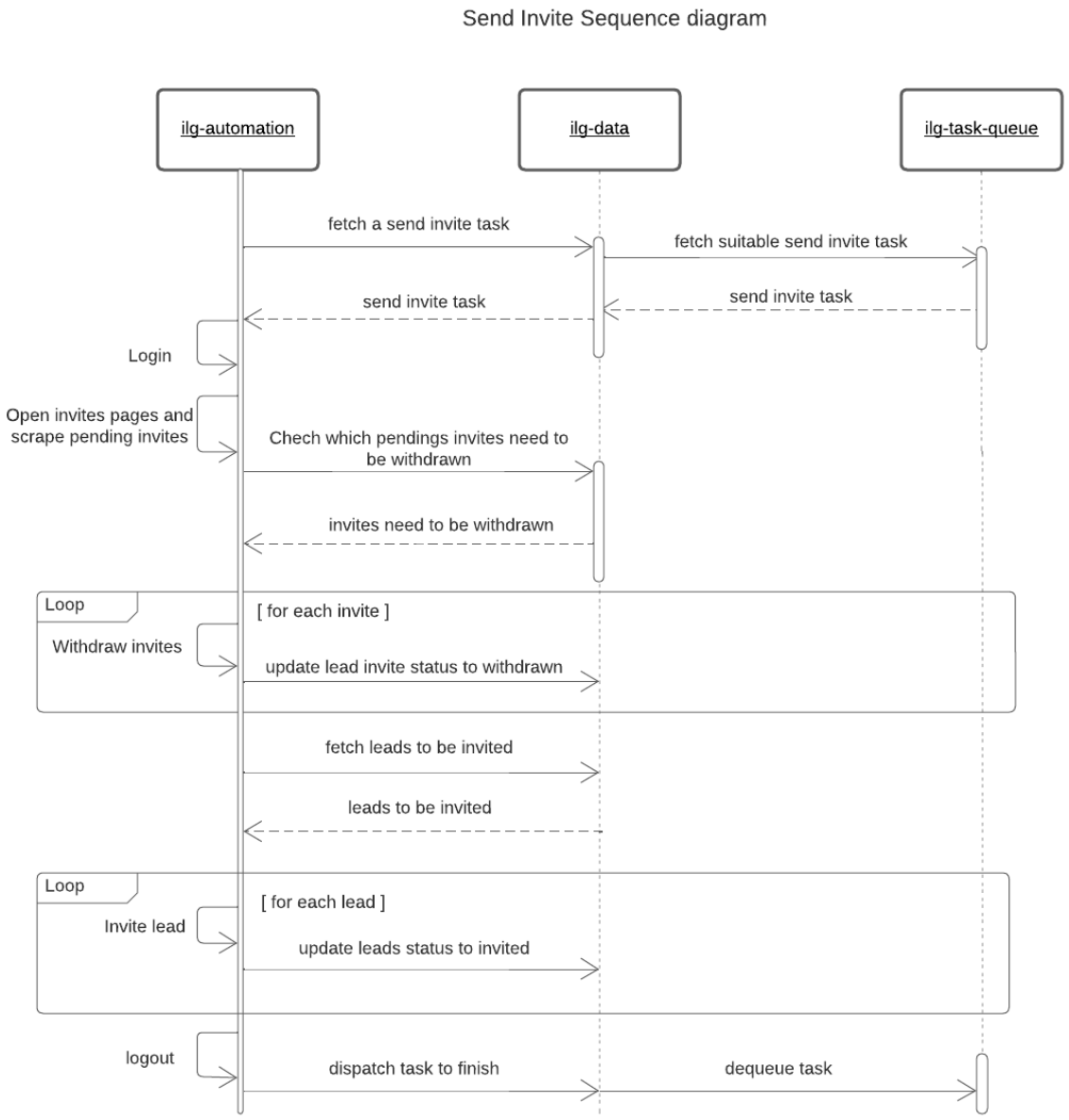
Here is the sequence diagrams of the **campaign-scrape** and **lead-scrape** tasks that the scrape service is responsible for :

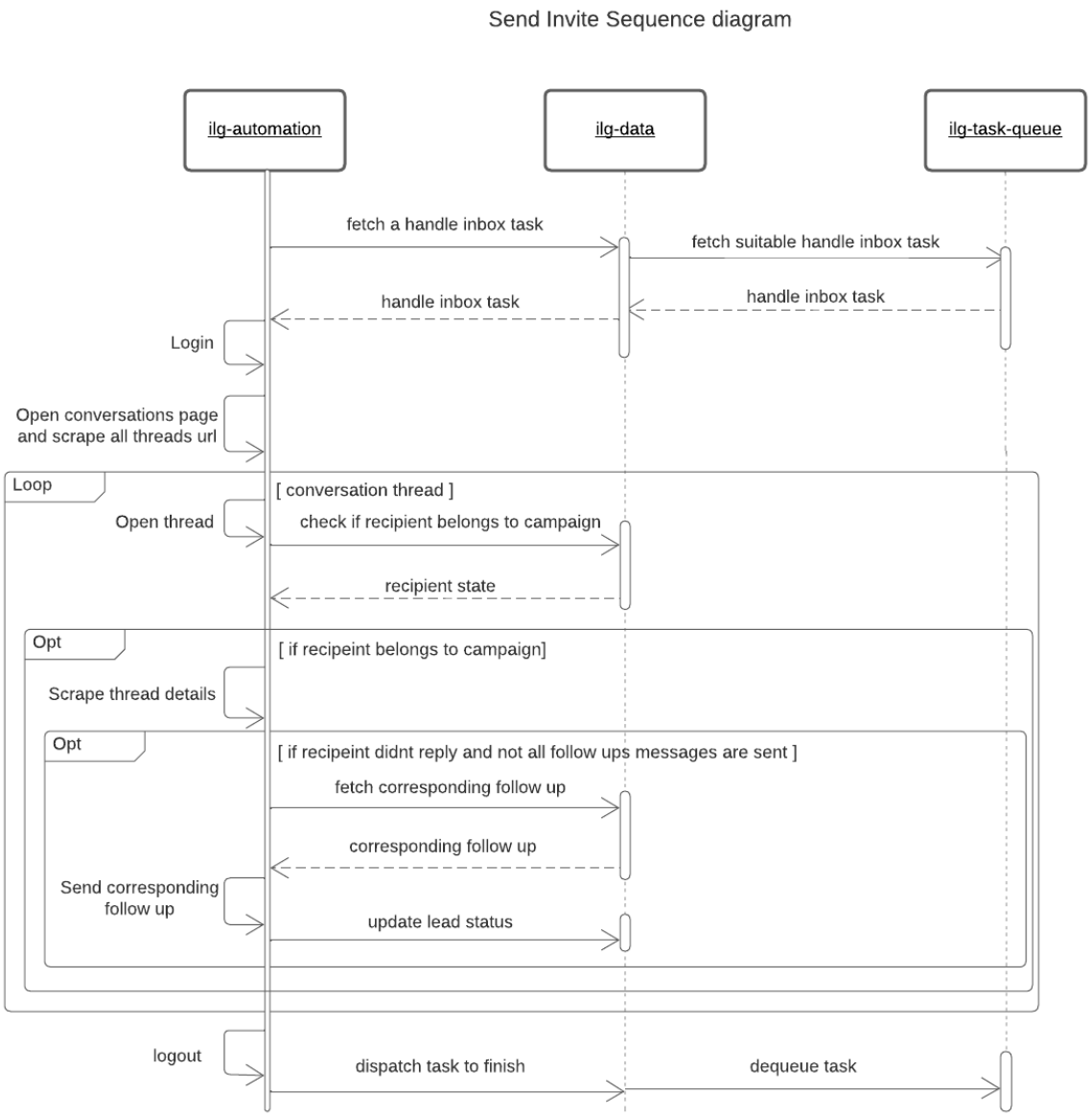



Lead Scraper Sequence diagram
ilg-scraper
ilg-data
ilg-task-queue
fetch sales navigation login credentials
sales navigation login credentials
login
fetch a lead scrape task
fetch suitable lead scrape task
lead scrape task
lead scrape task
open lead's page
and scrape data
Alt
[Scraping succeded]
save scraped lead data
dequeue lead scrape task
[Scraping failed]
dispatch
mark task failed
logout
dispatch task to finish
dequeue task

3.2.3.2 Automation flow

Here is the sequence diagrams of the **send-invite** and **handle-inbox** tasks that the automation service is responsible for :




Send Invite Sequence diagram
ilg-automation
ilg-data
ilg-task-queue
fetch a handle inbox task
fetch suitable handle inbox task
handle inbox task
handle inbox task
Login
Open conversations page and scrape all threads url
Loop
[conversation thread]
Open thread
check if recipient belongs to campaign
recipient state
Opt
[if recipeint belongs to campaign]
Scrape thread details
Opt
[if recipeint didnt reply and not all follow ups messages are sent]
fetch corresponding follow up
corresponding follow up
Send corresponding follow up
update lead status
logout
dispatch task to finish
dequeue task

3.2.4 Data Layers

3.2.4.1 Schema (Entity Relationship Diagram)

Here is the entity relationship diagram of the application database schema :

3.2.4.2 Different interfaces

Mainly we have two interfaces to access the database from all other services of the application :

- **GraphQL API Layer**: this interface is used by all the internal services of the application to access the data. so this layer is only reachable within the **internal private virtual network** and **not exposed for public**.
- **RESTful API Layer**: this interface is used by the frontend application to access the data. so this layer is **exposed for private** but of course secuired by am authentifition and authorization systems.

Conclusion

In this chapter, we discussed the main features of the application and its different layers and services. so now with everything clear and explained, we can start discussing the realization part and our final products in the next chapter.

Chapter 4

Realization

Introduction

In this chapter, we will finally discuss the long awaited realization. This is the implementation phase where all of the work done in the previous chapters comes into picture. This section will **heavily focus on DevOps** since our primary objective has been migrating the ILG application to a new scalable architecture. We will start with the new architecture hardware and software setup, present some of the difficulties we encountered and then list all of the technologies we've used for our work.

4.1 Hardware setup

Since we agreed to proceed with the self-managed Kubernetes deployment as mentioned in **Chapter 2: State of the art**, we had to create own cluster nodes or virtual machines. The nodes in question can be separated into the core nodes that are created once, and the scalable nodes that can be created indefinitely.

4.1.1 Cloud provider

To provision the hardware resources that we need for the deployment, we're going to use the IONOS Cloud provider. [16, Ionos is a web hosting company founded in Germany in 1988 and is currently owned by United Internet.]

Figure 5: Logo of IONOS

IONOS Cloud provides -through the Ionos cloudpanel service- an easy way to create and manage virtual machines, firewall rules, private networks and various other infrastructure components.

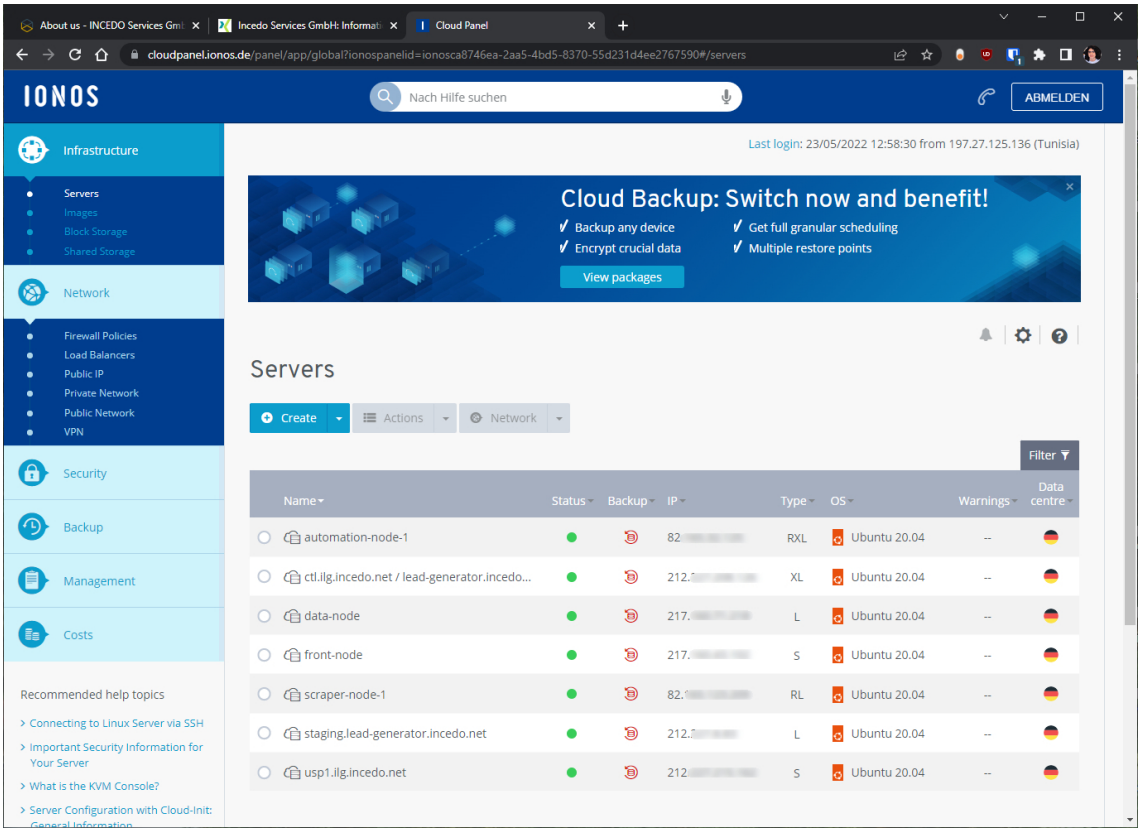


Figure 6: Ionos cloudpanel dashboard

The two images below show the available VM sizes in the Ionos cloudpanel dashboard that we can pick for our VMs. Note that will be referenced in the tables below.

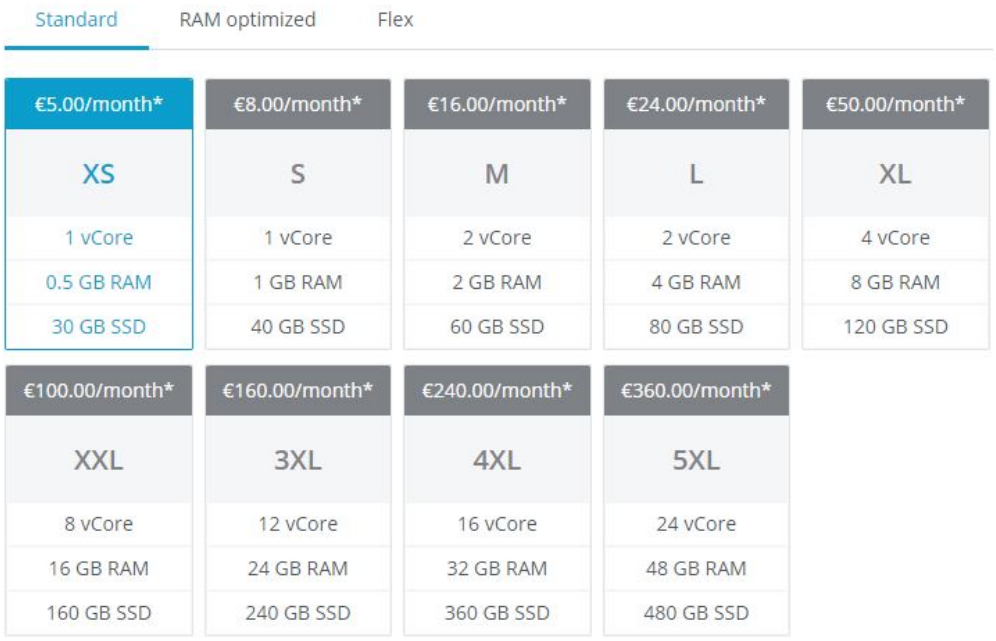


Figure 7: Standard VM sizes in IONOS

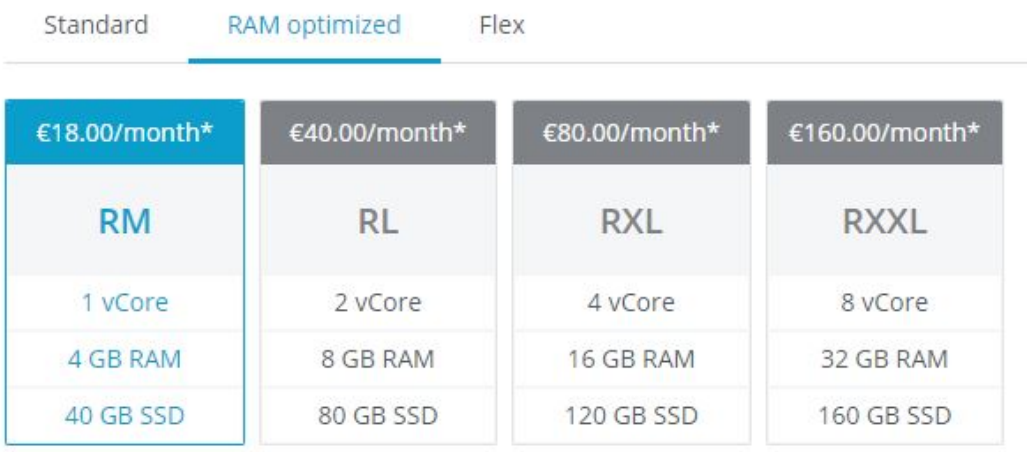


Figure 8: RAM Optimized VM sizes in IONOS

4.1.2 Kubernetes Cluster

Below are the necessary virtual machines needed to setup a self-managed Kubernetes Cluster. Please note that

- What each of the services represent is already mentioned in **Chapter 3: Analysis and specification of requirements**
- The size column which stands for the specification of the virtual machine is described in the previous section.
- The number of servers for **scraper**, and **automation** nodes will be scaled depending on the number of clients as mentioned in **Chapter 3: Analysis and specification of requirements**.

Table 10: Core cluster nodes

Name	Services	Size
kube-controller	The main Kubernetes cluster controller	XL
data-node	postgresdb, ilg-data, ilg-api, ilg-scheduler	L
front-node	ilg-front	S
staging-node	postgresdb, all other ilg services	L

Table 11: Scalable cluster nodes

Name	Services	Size
scraper-node-n	ilg-scraper	RL
automation-node-n	ilg-automation	RXL

4.1.3 Proxy Servers

Proxies are used to ensure all requests to LinkedIn come from a single IP address as mentioned in **Chapter X:** —

Table 12: List of proxy servers

Name	Services	Size
usp-n	Upstream proxy server running Squid	XS

4.2 Software setup

After setting up our hardware resources, we end up with raw Linux virtual machines with no configuration whatsoever. We have to create and manage our own Kubernetes instance on it to proceed any further. We will be using **MicroK8s**. But before that, we also need to setup some basic configuration.

4.2.1 Manual configuration

For a single virtual machine, we have to do the following setup:

- Create a user with sudo permissions
- Install the base packages we will be using
- Edit the hosts file so that the VM recognizes other VMs by their hostnames
- Install MicroK8s to have a single cluster node running on the VM
- Join the VM with the master node to have a multi-node cluster.

4.2.2 Automating the process

Manually configuring the above can be extremely tedious, unefficient and error prone. Therefore, we've created **Ansible** scripts or playbooks that do just that. In the end, we simply edit a configuration file and directly run the script. This has a huge advantange especially when

- We want to create a development environment for easy prototyping.
- We want to use the exact same setup for future projects.

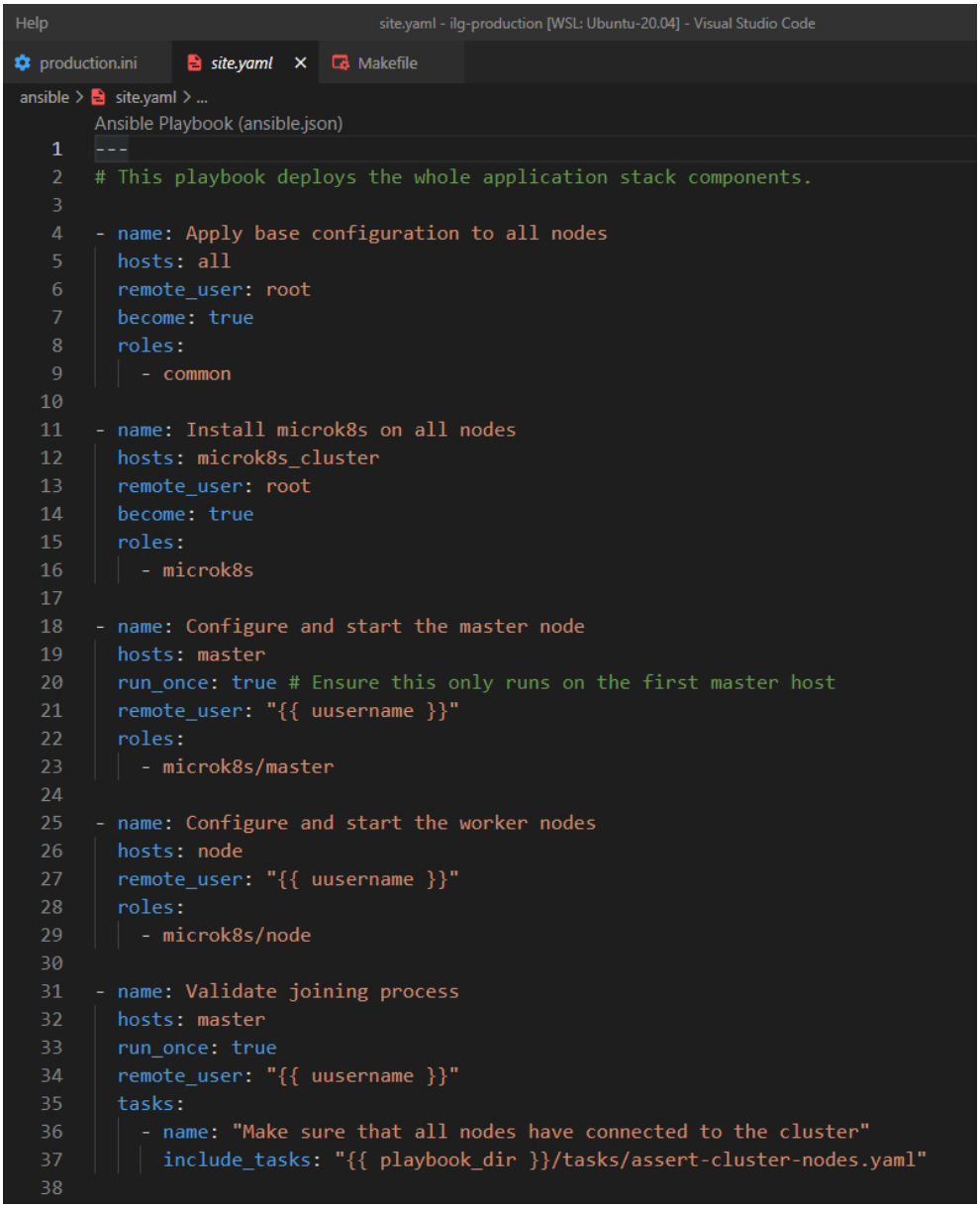


Figure 9: Extract from the Ansible playbook

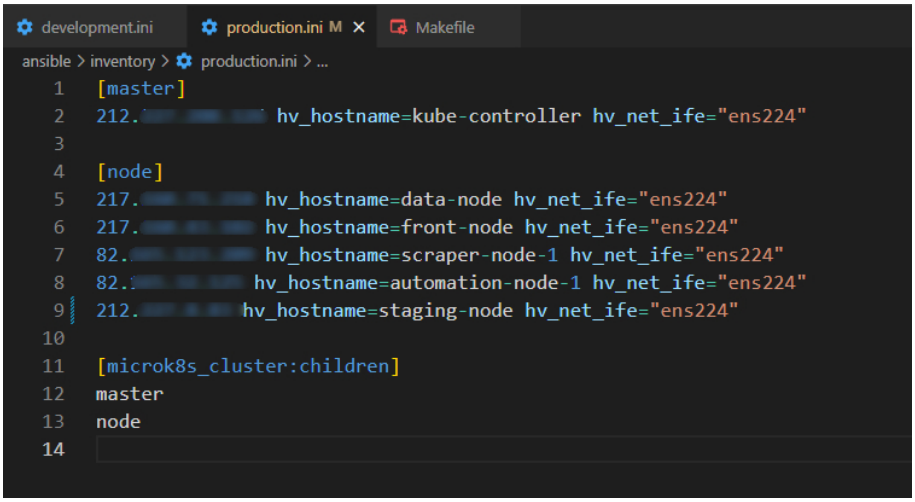


Figure 10: The Ansible hosts configuration

Note that we've also used a hosts file for development where we tested the setup extensively before trying it on production as seen from **Figure 10**.

4.3 Deployment process

This section will explain the approach we adopted to continously integrate, deploy and deliver our application as per the best practices of DevOps previoulsy mentioned in **Chapter 3: State of the art**. The deployment is currently supported on two different environments; the staging environment where we make sure our application works as intended, and of course the production environment.

4.3.1 Adopted strategy

Since we have multiple microservices, it would not be ideal to interact with our Kubernetes cluster from all of them whenever we make a change. For that reason we use the following steps to deploy our application.

- On code changes on any microservice, we create a new Docker image that we push to a custom container registry.
- Once the build is finished, we trigger a multi-project pipeline to a central repository we will call ilg-controller.

- The central repository will interact with our Kubernetes cluster to make the final deployment to the correct environment using a custom Helm package that we create.

This is further explained in detail in the following sub sections.

4.3.2 Linking the cluster to GitLab

Since we have a Kubernetes cluster and we also use GitLab, we will link the two using the GitLab Kubernetes Agent, or KAS for short.

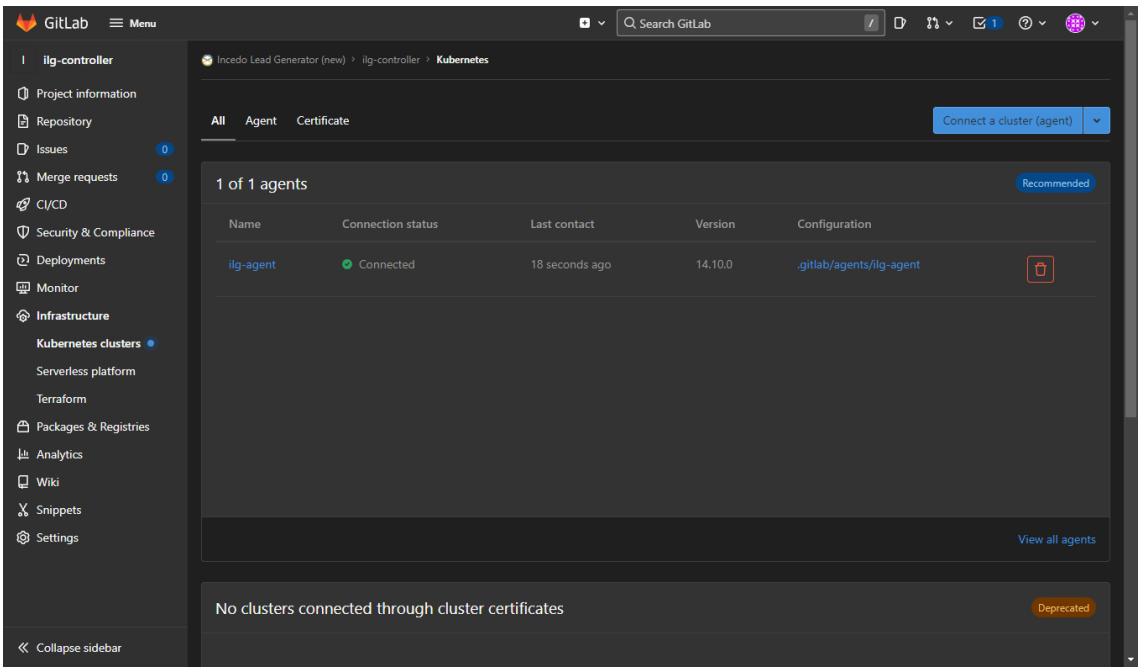
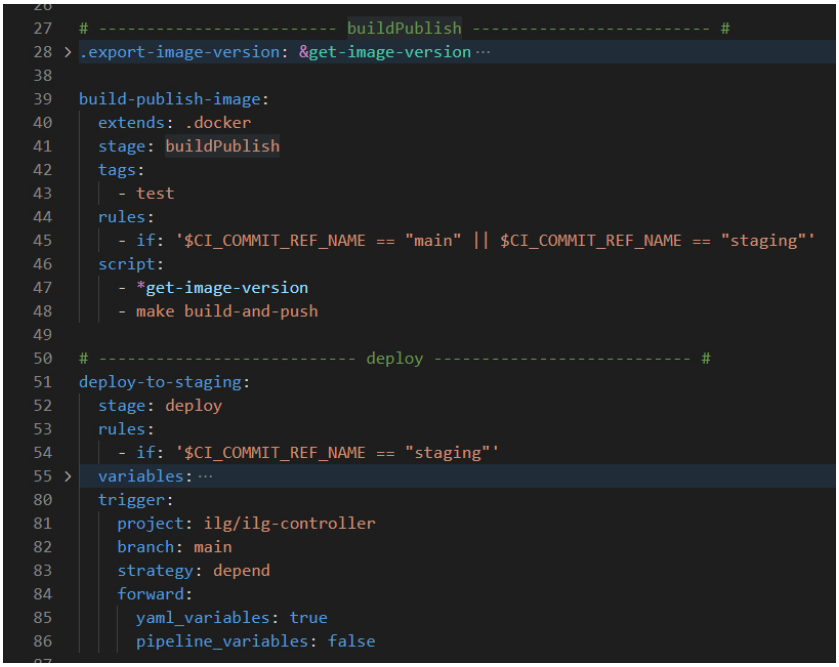


Figure 11: GitLab Kubernetes Agent

4.3.3 Continuous Integration

When developers make code changes to the main branch of any microservice, tests are automatically executed. If the tests are validated, a new Docker image that incorporates the changes the developers made will be pushed to the container registry and our deployment pipeline will be triggered. In other words, the new code will be integrated and that actually marks the end of this step.

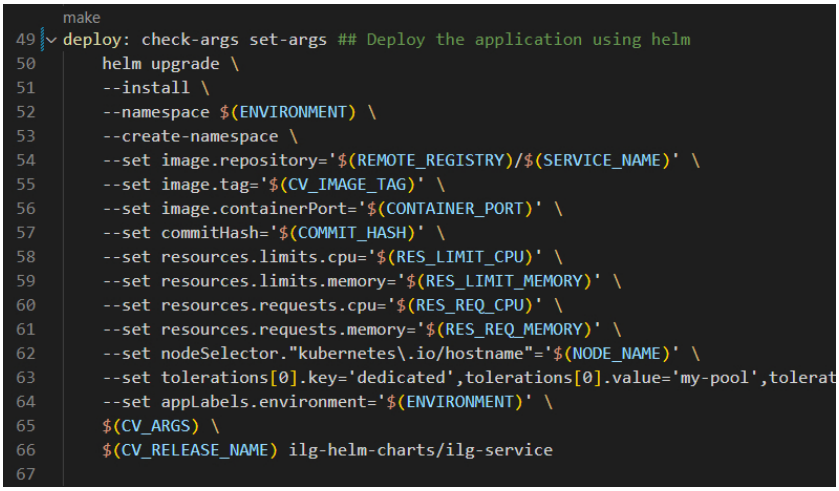


```
# ------------------------ buildPublish ------------------------ #
.export-image-version: &get-image-version ···

build-publish-image:
  extends: .docker
  stage: buildPublish
  tags:
    - test
  rules:
    - if: '$CI_COMMIT_REF_NAME == "main" || $CI_COMMIT_REF_NAME == "staging"'
  script:
    - *get-image-version
    - make build-and-push

# --------------------------- deploy --------------------------- #
deploy-to-staging:
  stage: deploy
  rules:
    - if: '$CI_COMMIT_REF_NAME == "staging"'
  variables: ···
  trigger:
    project: ilg/ilg-controller
    branch: main
    strategy: depend
    forward:
      yaml_variables: true
      pipeline_variables: false
```

Figure 12: Extract from the CI pipeline

4.3.4 Continuous Deployment

Our ilg-controller repository (where KAS is configured) will choose the correct microservice to deploy depending on the trigger input. It then sets the necessary environment variables for it and installs it from a custom Helm package we've created for the deployment.



```
make
deploy: check-args set-args ## Deploy the application using helm
    helm upgrade \
    --install \
    --namespace $(ENVIRONMENT) \
    --create-namespace \
    --set image.repository='$(REMOTE_REGISTRY)/$(SERVICE_NAME)' \
    --set image.tag='$(CV_IMAGE_TAG)' \
    --set image.containerPort='$(CONTAINER_PORT)' \
    --set commitHash='$(COMMIT_HASH)' \
    --set resources.limits.cpu='$(RES_LIMIT_CPU)' \
    --set resources.limits.memory='$(RES_LIMIT_MEMORY)' \
    --set resources.requests.cpu='$(RES_REQ_CPU)' \
    --set resources.requests.memory='$(RES_REQ_MEMORY)' \
    --set nodeSelector."kubernetes\.io/hostname"='$(NODE_NAME)' \
    --set tolerations[0].key='dedicated',tolerations[0].value='my-pool',tolerat
    --set appLabels.environment='$(ENVIRONMENT)' \
    $(CV_ARGS) \
    $(CV_RELEASE_NAME) ilg-helm-charts/ilg-service
```

Figure 13: Extract from the deployment Makefile

4.3.5 Custom Helm Package

Since we agreed to use Helm (refer to **Chapter 2: State of the art**), we've created a package -or in terms of Helm- a chart of our own to manage the Kubernetes cluster resources.

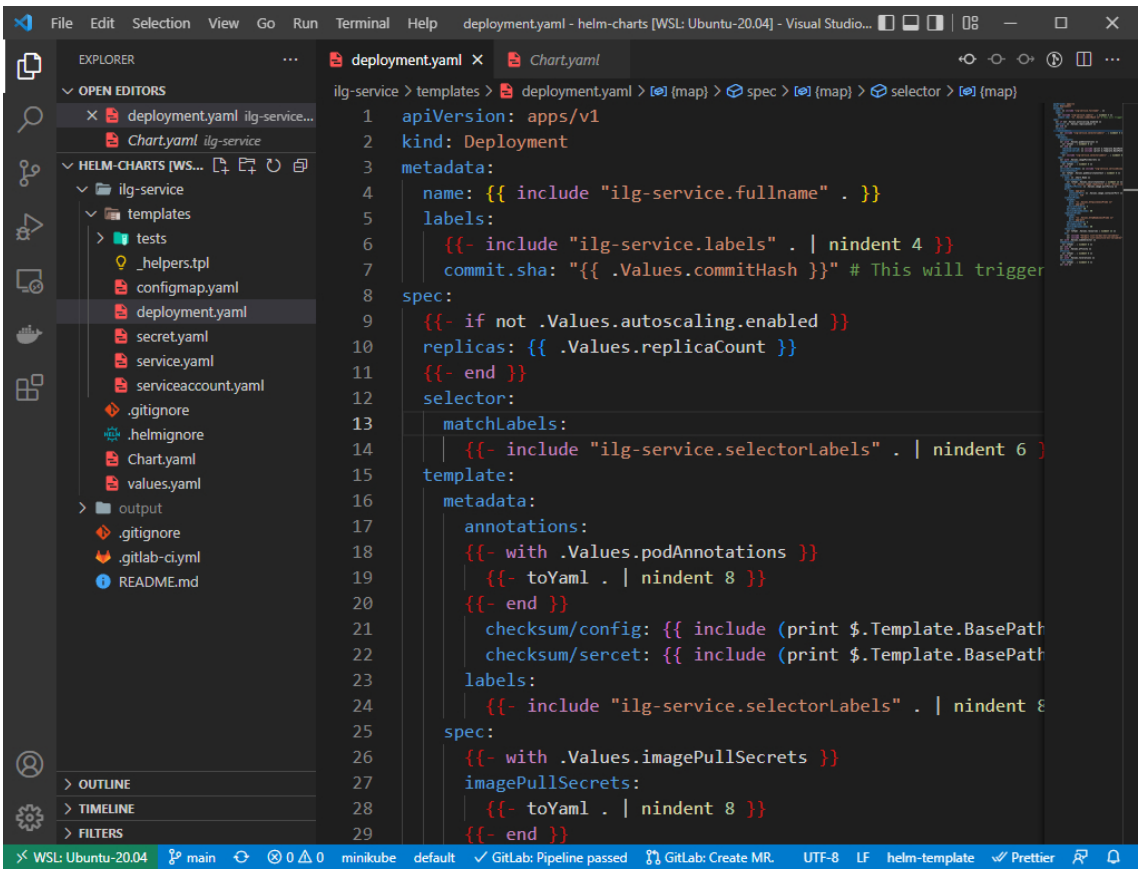


Figure 14: Extract from the custom Helm Chart

We even integrate a CI/CD pipeline for the Helm chart itself so that new packages are built automatically whenever we make changes to the manifest files.

```yaml
# -------------------- buildPackage ---------------------- #
build-ilg-service:
  extends: .helm
  stage: buildPackage
  tags:
    - test
  rules:
    # https://regexr.com/6im81
    - if: '$CI_COMMIT_REF_NAME == "main" || $CI_COMMIT_REF_NAME =~ /^v\d+\.\d+\.\d+-?[a-zA-Z0-9_.-]*$/'
  script:
    - *get-package-version
    - helm package ./ilg-service/ --destination ./output
  artifacts:
    expire_in: 1 week
    paths:
      - output
```

Figure 15: Extract from the custom Helm package's pipeline

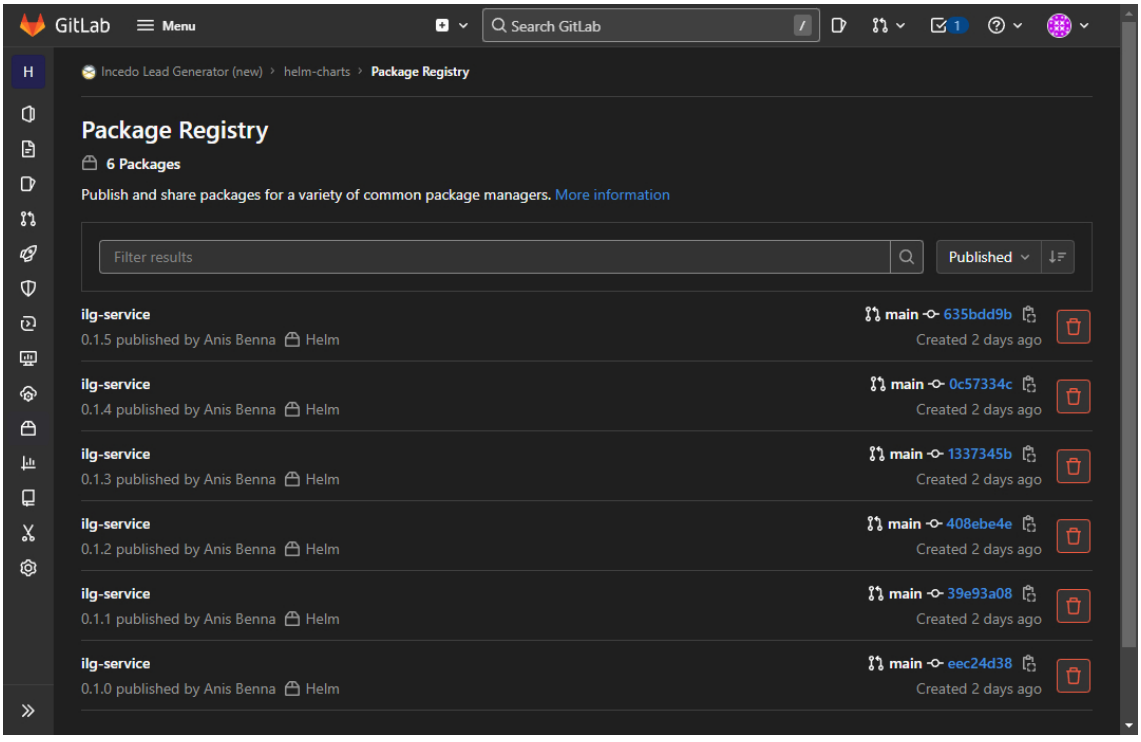


Figure 16: GitLab's package registry

4.4 Technologies

This part is reserved for the presentation of all of the software used in the realization of the project and includes but is not limited to; programming languages, frameworks, technologies, etc...

For a comparative analysis on some of our choices, see **Chapter 2: State of the art**.

- **Git:**

Figure 17: Logo of Git

- **Docker:**

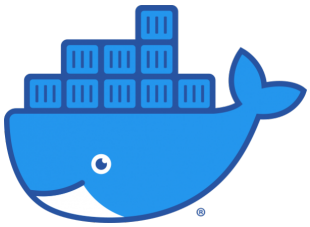


Figure 18: Logo of Docker

- **Playwright:**

Figure 19: Logo of Playwright

- **NestJS:**
  [12] A framework for building efficient, scalable Node.js web applications.

Figure 20: Logo of NestJS

- **ReactJS:**
  A front-end javascript library that's often used as a framework.

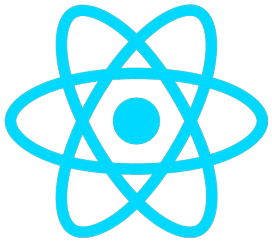


Figure 21: Logo of ReactJS

- **MySQL:**

Figure 22: Logo of MySQL

- **Sequelize:**
  [13] A modern TypeScript and Node.js ORM for SQL databases.

Figure 23: Logo of Sequelize

- **GraphQL:**
  [8] A query language for APIs and a runtime for fulfilling those queries with existing data.

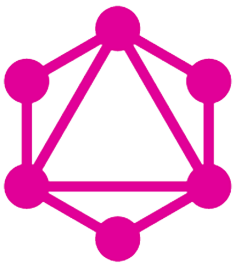


Figure 24: Logo of GraphQL

- **GitLab:**
  A DevOps software that allows secure collaboration and operations in a single application.

Figure 25: Logo of GitLab

- **Nginx:**
  [17] A multi-purpose web server can also be used as reverse proxy, load balancer, mail proxy and HTTP cache.

Figure 26: Logo of Nginx

- **Renovate:**
  A software that provides automatic dependency updates with support for multiple languages.

Figure 27: Logo of Renovate

- **Docker compose:**
  A helper tool to run applications with multiple Docker containers using a .YAML format. Used in development.

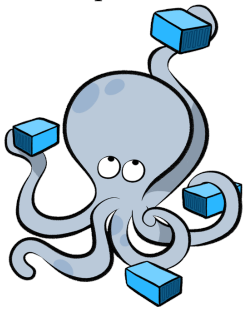


Figure 28: Logo of Docker Compose

- **Visual Studio Code:**
  A code editor that's used as an entire development environment with the help of extensions.

Figure 29: Logo of VSCode

- **Linux:**
  A Unix-like operating system for common daily use and more commonly for servers.

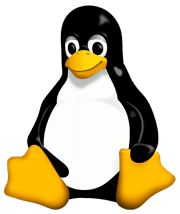


Figure 30: Logo of Linux

- **Make:**
  GNU Make is used extensively throughout the application to save useful commands for the ease of maintainability.

Figure 31: Logo of Makefile

- **MicroK8s:**
  [11] MicroK8s is a simple production-grade conformant K8s. It is Lightweight and focused.

Figure 32: Logo of MicroK8s

- **Kubernetes:**
  The most powerful tool for managing containerized workloads in the cloud.

Figure 33: Logo of Kubernetes

- **Ansible:**
  An open-source software for provisioning, configuring and managing infrastructure.

Figure 34: Logo of Ansible

- **Squid Proxy:**
  [3] Squid is a caching proxy for the Web that supports HTTP, HTTPS, FTP and more.

Figure 35: Logo of Squid Proxy

- **GitLab agent for Kubernetes:**
  [15] A secure and reliable way to attach a Kubernetes cluster to GitLab.

Figure 36: Logo of GitLab Agent for Kubernetes

- **Helm:**
  Helm is the most popular package manager for Kubernetes.

Figure 37: Logo of Helm

- **LaTeX:**
  [9] A high-quality document preparation and typesetting system for technical grade documents.

Figure 38: Logo of The LaTeX Project

4.5 Difficulties encountered

This not persay a technical difficulty as it is more of a limitation from the side of Ionos cloudpanel; the service of IONOS Cloud that we use the provision the needed resources. To have a full DevOps approach, even the infrastructure should be managed in code -also known as IAC (Infrastructure As Code)- using tools such as Terraform. However, Ionos cloudpanel does not provide that functionality. Therefore we had to scrape off the idea and simply create the virtual machines fron the UI everytime we need them.

Conclusion

A successfull project comes from a successfull development environment as well as a clean production environment. For this reason, we spent quite a good amount of time setting up the infrastructure in a clean and scalable way using various tools and script which all comform to the modern ways of how DevOps should be done.

General Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Webography

[1] Microsoft Azure. Devops overview, 2022. https://azure.microsoft.com/en-us/overview/what-is-devops/#devops-overview.

[2] Arthur Busser. What is a container?, 2020. https://www.padok.fr/en/blog/container-docker-oci.

[3] Squid Cache. The official website, 2022. http://www.squid-cache.org/.

[4] Beth Pariseau Emily Mell. What is container management?, 2021. https://www.techtarget.com/searchitoperations/definition/container-management-software.

[5] Ronak Ganatra. Graphql vs rest apis, 2018. https://graphcms.com/blog/graphql-vs-rest-apis.

[6] GeeksForGeeks. Mongodb vs mysql, 2018. https://www.geeksforgeeks.org/mongodb-vs-mysql/.

[7] Incdeo Services GmbH. About incedo, 2022. https://incedo.de/.

[8] GraphQL. The official website, 2022. https://graphql.org/.

[9] The LaTeX Group. The official website, 2022. https://www.latex-project.org/.

[10] Red Hat. What is a ci/cd pipeline?, 2022. https://www.redhat.com/en/topics/devops/what-cicd-pipeline.

[11] MicroK8s. The official website, 2022. https://microk8s.io/.

[12] Nest.js. The official website, 2022. https://nestjs.com/.

[13] Sequelize. The official website, 2022. https://sequelize.org/.

[14] Amazon Web Services. What are microservices?, 2022. https://aws.amazon.com/microservices/.

[15] Senior Product Manager at GitLab Viktor Nagy. A new era of kubernetes integrations on gitlab.com, 2021. https://about.gitlab.com/blog/2021/02/22/gitlab-kubernetes-agent-on-gitlab-com/.

[16] Wikipedia. Ionos, 2022. https://en.wikipedia.org/wiki/Ionos.

[17] Wikipedia. Nginx definition, 2022. https://en.wikipedia.org/wiki/Nginx.