Final Project

T-404-LOKA, Sleep DevOps, 2022-1

Reykjavik University, Menntavegi 1, IS-101 Reykjavík, Iceland

Ægir Máni Hauksson

aegir19@ru.is

May 13, 2022

Supervisors

Assistant Professor María Óskarsdóttir

Benedikt Hólm Þórðarson

Examiner

Assistant Professor Anna Sigríður Íslind

Dept. of Computer Science,
Reykjavík University

Abstract

This thesis describes the analysis, research and implementation process of introducing DevOps methodology into a software development project. The core aspects of the paper focus on the delivery of task automation in the form of continuous integration and continuous delivery, workflow improvements and remote development.

Contents

Table 1: Abbreviations and acronyms

Abbreviation	Full form
RU	Reykjavík University
CI/CD	Continuous Integration & Delivery
VPN	Virtual Private Network
IDE	Integrated Development Environment
VM	Virtual Machine
IP	Internet Protocol address
SSH	Secure Shell Protocol

1 Introduction

1.1 Terminology

DevOps is a set of practices that combines software development and IT operations. It aims to shorten the systems development life cycle and provide continuous delivery with high software quality [20]. It covers a broad range of automation strategies and practices applied to software development to ensure fast paced progression from development to production.

CI/CD is short for *continuous integration and continuous delivery*. It is a term often used in the realm of DevOps engineering. This term is directly related to the automation strategies incentivized by the DevOps methodology. Continuous integration refers to automation of software development tasks such as testing and verification. Continuous delivery represents automation of IT operations such as building and deploying software to production environments. These automation procedures require underlying software to operate, often referred to as *runners* [4].

Reykjavík University is a science, engineering and law school situated in Iceland's capital region, hereafter also referred to by the shorthand *RU* and not to be confused with the abbreviation for the Russian Federation which is not insinuated within this report.

Sleep Revolution [17] is a project funded by the European Union. Among other things this project aims to transform current diagnostic methods for sleep-disordered breathing [1] with machine learning technology. For this purpose they have set up a data gathering center at Reykjavik University to store sleep diagnostic data from across Europe. This cloud solution is hosted within a virtual private network.

Arora is a project that aims to deliver a new Python [15] package for working with sleep data from different sources. It is developed by a team of undergraduate students at the University of Reykjavik which are referred to in this paper as *Developers* and *Arora team*. The product delivered by the Sleep DevOps project, as described in this paper, is to be developed alongside, and integrated with the Arora project.

1.2 The Problem

The Arora project team aims to provide several data manipulation methods to researchers working with sleep diagnostic data gathered by the Sleep Revolution project. This is done by applying a multitude of algorithms or scripts on the data to alter its format and structure. Their project is therefore heavily dependent on data accessibility during development and testing.

The data in question is confidential and possibly protected by data protection laws [13] as it contains personal information. Any data gathered by the Sleep Revolution project cannot be exported from the data gathering center and is not readily available to the Developers through any means other than the RU cloud host. Without the proper tools and strategies implemented, this makes development particularly tedious. Further, a conventional CI/CD integration with version control is not possible, as the data required for testing is not locally accessible.

1.3 Project Goals

Our goal is to enable DevOps practices in projects related to Sleep Revolution. We aim to implement a continuous integration tool which allows secure access to data sets within Reykjavik University's cluster. We will provide guidance on it's usage and further suggest adjustments that can help with remote development.

The Arora project team will be our subject throughout the course of the project. We will analyze their working environment and find appropriate improvements to bottlenecks in productivity. This includes suggesting workflow improvements, introducing new software or features as well as automating tasks with a continuous integration tool.

A subset of goals have been defined by the product owners. We want to implement a secure testing automation strategy using the sleep diagnostic data and we want to implement release automation for the Python packages developed by the Arora team which will be made available on PyPi [14].

2 Methodology

2.1 Approach

The project was developed under agile principles. The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. [23] Once the work begins, the team cycles through a process of planning, executing, and evaluating.

The Arora project developers chose to follow the scrum methodology. Due to the nature of our relationship with the Developers and their project, we chose to follow a similar development methodology.

2.2 Meetings

On Tuesdays and Thursdays, the Developers' stand-up meetings were attended. Any detailed discussions regarding things not related to the status of the projects were discussed with the Developers individually in private conversation via communication channels such as slack and in person.

On Mondays, we held status meetings with the product owners where we discussed the progress of the project as well as any potential issues and concerns. Any take-away from these meetings was documented.

2.3 Project management tools

All project management was organized through GitLab. GitLab was the tool of choice as it offers an all-in-one project management platform. Version control [22] as well as tasks can be managed through their web interface. Issues can be tracked and created through GitLab boards [5], which operates in a similar fashion to project management tools such as Jira and Asana.

One of the main advantages of using GitLab is that development tasks can be linked directly to code. This gives a great overview of the project status as the relevant code versions can be attributed to the tasks (backlog) created. Having everything in one place also makes things easier to manage. Moreover, milestones [6] such as sprint deadlines can be assigned to each task and a summary of each milestone's achievements can then be reviewed and evaluated before upcoming sprints.

2.4 Sprints

There were 4 sprints, each one roughly 3 weeks long and built on the work achieved in the previous sprint. Table 2 represents the sprint structure. This structure was followed to the best of ability. Adjustments were made at the occurrence of any unforeseen events that hindered the project from proceeding with the initial plan.

Table 2: Planned sprint structure

Sprint No.	Time frame	Goal
1	31.1.22 - 20.2.22	Implement a proof of concept CI/CD solution
2	21.2.22 - 20.3.22	Produce first viable CI/CD product
3	21.3.22 - 10.4.22	Reiterate CI/CD product and add features. Improve working environment of Developers
4	25.4.22 - 13.5.22	Reiterate CI/CD product, add features and finalize Analyze and improve workflow of Developers

The first sprint aims to identify viable solutions for a CI/CD integration and testing software by integrating with a mock project. This gives us a general idea of how the product can be implemented.

The second sprint implements the first potentially viable product in the form of a cloud based CI/CD server. This is heavily dependant on finding viable existing software solutions for our use case in the first sprint.

Any further sprint aims to produce a new version of a potentially viable product. In these sprints, additional features are added and existing ones re-iterated upon if found to be necessary. Furthermore, complementary development features are integrated with the Arora Project repository. Some of the planned features are listed below.

- Workflow guideline documentation

- Pipeline overview
- Test coverage
- Project release versions and artifacts
- DevOps documentation

Additionally, potential improvements to the Arora team's working environment as well as working methodology are evaluated. This included things such as git workflow, project restrictions, IDE improvements and helping with test generation.

2.5 Work Hours

Table 3 lists the hours that members of the project strived to achieve. Some flexibility had to be taken into account since other courses had heavier load at times. At some points we ended up needing to make up for shortcomings of other weeks where such events occurred. Since the team consists of only one member, the work schedule was quite flexible. For improved productivity, similar work hours as the Arora Project team were preferable, especially in cases where blocking questions arose.

Table 3: Weekly target work hours

School week	Work hours
1-12	15h
13-15	40h
Total	300h

All work was documented. This included hours spent and what work was done in that time. The summary of this documentation was presented at the weekly meetings with the product owners. A template was set up for this in the repository and was updated daily when working on the project.

3 Risk assessment

The risk assessment discusses the primary risks we accounted for during project development. It defines the control measures taken at all times in order to both minimize the probability of these risks occurring and to minimize their negative impact on the project, affiliated persons, equipment and digital assets in the event of their occurrence. Table 4's enumeration denotes the sections below.

Table 4: A high level risk assessment overview

No.	Asset	Value	Risk	Probability
1.	Sleep Diagnostic Data	High	Data breach/leak	Low
2.	DevOps Server	Medium	Unavailability	High
3.	Personnel	High	Unavailability	Medium
4.	Product	Medium	Service downtime	Medium
5.	Arora team	Medium	Delayed Progress	Medium

3.1 Data breaches

Some inherent and unavoidable risks are involved when working with sensitive data in development environments. One of the most probable events that can lead to data breaches or data leaks, especially in our case is by the hands of an uninformed user of our product, namely the developers of the Arora project. The protocol which must be followed when using our product must thus be clearly and concisely documented. For the same reasons it is important that the aforementioned personnel be made fully aware of the possible consequences of failure to comply with the given protocol.

Yet another possible security risk comes from third parties with ill intentions. Especially those with expert knowledge of network security whom one can categorize as *Black Hat Hackers* [19]. This risk applies to software used or developed in this project. Choosing reliable and secure software solutions, as well as following industry standards when it comes to implementing custom solutions will play an important role in minimizing the risk of intrusions.

3.2 DevOps server reliability

There are a multitude of factors that are at play when creating and configuring a server. Internet, power (electricity), software and hardware are components of a server that must be available at all times in order for it to be accessible and usable. Additionally we will be installing new software and trying out configurations on various components of the system. This without a doubt at some point in time results in a malfunction that will inevitably require a complete or partial system reconfiguration.

Preventing these events is close to impossible during development. It is therefore of utmost importance that every step taken during server configuration be extremely well documented and reproducible. Once a configuration has been determined, automating the reproduction steps will help significantly in shortening downtime.

Furthermore, the DevOps server for the project is currently hosted as one instance on a virtual machine. This server is neither externally managed by the project participants (configuration outside the virtual machine), nor does it have a backup in case of failures. In case of external server configuration or server failures, a request must be made, which in turn results in waiting for the operator of that server to complete that request. This can cause a significant productivity decrease, as these requests can come quite frequently and can take several days to complete. Therefore, a locally managed server, mocking the behavior of the cloud based development server should be set up. This allows addressing any concerns immediately using a local development

server while waiting for requests to be completed by the server operator. This also enables storing backups of the servers locally, allowing to quickly get back to work after a system failure.

3.3 Project personnel

One cannot ignore the risks of losing personnel temporarily nor permanently. We do however not concern ourselves with the permanent case in this writing as that would lead to complete termination of this project. Various factors can induce temporary unavailability of personnel during the 15 weeks of this project and there is no need to list them all here. Although we emphasize that things such as *"my dog ate my homework"* do not fall in this category.

As mentioned briefly before, other courses will demand more attention during certain periods of this semester and must be attended to respectively. Sickness is a rare occurrence among project participants, yet in such a case, it shall be reported to the project committee and resolved accordingly. Any downtime in project development must be made up for, and is done so in accordance to the minimal sum of hours required to be worked each week (15 Hours in the 12 week period, 40 hours in the 3 week period).

Unavailability is especially critical in this case, as there is only one person working on this project. Any unavailability must be accommodated for by the sole member of the project as soon as possible. Luckily this member has reached an elite status of overworking tolerance, and does not have any problem with putting in extra hours when the need presents itself.

3.4 Product downtime

Unexpected things can happen which induce downtime of service. The same can apply to the product delivered by this project, mainly the operational versions of the CI/CD server. Apart from being an annoyance to the development team of the Arora package this does not bring any critical risks to the project. We can however rest assured that development will progress more slowly for both parties (Developers and participants in this project) if these events occur frequently. Therefore before any integration is made with any version of the product delivered by this project, it must be well tested and assured to a high degree to be stable.

For this the Arora project can be forked and tested extensively under different pipeline loads until it can be assured that the product delivered is reliable.

3.5 Arora project progress

The success of this project relies on the Arora team's deliverables. Any delays on their behalf may delay the progress of our project. Understanding the goals of their project is therefore a highly valuable resource in these scenarios. Having this knowledge, a minimal replica (mock-up [21]) of their project can be created, which will allow continued work on certain aspects without relying on Arora source code entirely. This is of course not sufficient in all cases, but then there is not much that can be done other than waiting for the respective deliverables, figuring out what else can be done and altering the plan accordingly.

3.6 Other

Many other details are relevant risks, yet avoidable with commonly applicable knowledge of an amateur computer scientist. We therefore opt out of listing them in the risk assessment table and will instead describe the most notable ones very briefly below.

- Version control code: Prevent loss of progress
- Continuously document project: Prevent loss of information
- Try to stay on schedule: Prevent time crunching
- Keep up to date with relevant persons: Prevent miscommunication
- Keep a work log and document meetings: Prevent disorganization

4 Requirement Analysis

In our quite unconventional software development project, our efforts were concentrated on many different subject matters. This means our project implemented multiple features into the Developers' workflow that in some cases were completely unrelated to one another and are not contained within a single software. In fact, some of the "implementations" were not software related at all, and focused solely on improving working methodology or assisting with technical problems.

The requirement analysis was therefore mainly based on aspects of the project that required software configuration or software development to take place. This pertained mostly to the implementation of the different automation aspects of the CI/CD solution, but we left all options open for future additions of other software related requirements.

4.1 Target audience

Our target audience (user group) is the Arora team, which consists of 4 developers. A user group definition was therefore already established. Seeing as we had the whole team available to us, we created a short survey to understand some core aspects of their personal developer background and prior experience with technology that would likely play an important role in integrating DevOps features to their platform. The results of the survey can be viewed in Table 5

Table 5: Developer Survey

Developer No.	Programming Skills	Technological Experience	Learning Competence
1	Python: 8 Test creation: 6	Linux: 7 Docker: 1 CI/CD: 5	9
2	Python: 9 Test creation: 8	Linux: 4 Docker: 8 CI/CD: 8	10
3	Python: 8 Test creation: 8	Linux: 6 Docker: 7 CI/CD: 5	9
4	Python: 8 Test creation: 8	Linux: 7 Docker: 8 CI/CD: 5	9

From this quick survey we concluded that most of the developers felt confident in their programming skills. Their experience with CI/CD specifically suggested that it might be optimal for us to set up the initial pipeline configuration in the repository on their behalf. Seeing as they are all quite competent learners, we referred them to documentation for any further adjustments to the pipeline.

People may have a tendency to overestimate their abilities. Therefore we monitored their progress instead of taking the results at face value, and made adjustments when needed.

4.2 Requirements list

The requirements list can be viewed in Table 6. The requirements were updated throughout the course of the project. The requirements are ordered by priority and are assigned a type of either functional (F) or non-functional (NF).

Table 6: Requirement analysis

No.	Requirement	Priority	Type
1	A custom server hosts the CI/CD solution for the Arora project	A	F
2	Tests can be ran using data stored on the server	A	F
3	A python package can be built and deployed	A	F
4	Pipelines can be automatically triggered through source control	A	F
5	The Developers can configure the pipeline tasks without accessing the server	A	F
6	Pipeline status and logs are visible to the Developers	A	F
7	Instructions on pipeline configurations and usage is available to the Developers	A	F
8	Remote development can be performed in the same manner as local development by the Developers	A	F
9	Best practice workflow documentation is available to the Developers	B	F
10	The data sets used in tests can be configured by the Developers	B	F
11	Releasing new versions of the Arora Package can be done with a tag commit	B	F
12	A preliminary backup option exists in case of server downtime	B	F
13	Package release versions are stored in the git repository via package registry or container registry	B	F
14	Test coverage status of repository is updated via the pipelines	C	F
15	Automated semantic versioning is applied to releases	C	F
17	The documentation is clear and concise. All the Developers understand it.	A	NF
18	The product is easy to use. All the Developers know how to use it.	A	NF
19	Up-time of the released product is at a minimum of 95% for any released version	B	NF

5 Results

In this chapter we outline the solutions explored to enable optimal DevOps practices in the Arora project. We solve the major problems with data availability in development discussed in section 1.2 and introduce a highly practical CI/CD implementation for the Arora team.

5.1 CI/CD Software

5.1.1 Function

The CI/CD software is a task execution software designed to run jobs for GitLab projects. It was configured to run on the Sleep Revolution cluster hosted by Reykjavík University to allow authorized members of the Sleep Revolution project access to sleep diagnostic data in CI/CD pipelines. The extent of the software's application is not limited to Sleep Revolution projects, as it is configurable for any Linux operating system supporting the apt package manager and for any data, if any is required.

5.1.2 Technology stack

Figure 1: CI/CD solution tech-stack

Figure 1 shows the technology stack of the CI/CD solution. GitLab is used as a version control provider and GitLab CI/CD is used to register and manage runners for projects and groups. Docker is the executor of all tasks configured in pipelines and is managed by GitLab Runner which acts as the task scheduler. The solution is integrated on a Ubuntu operating systems.

5.1.3 Design

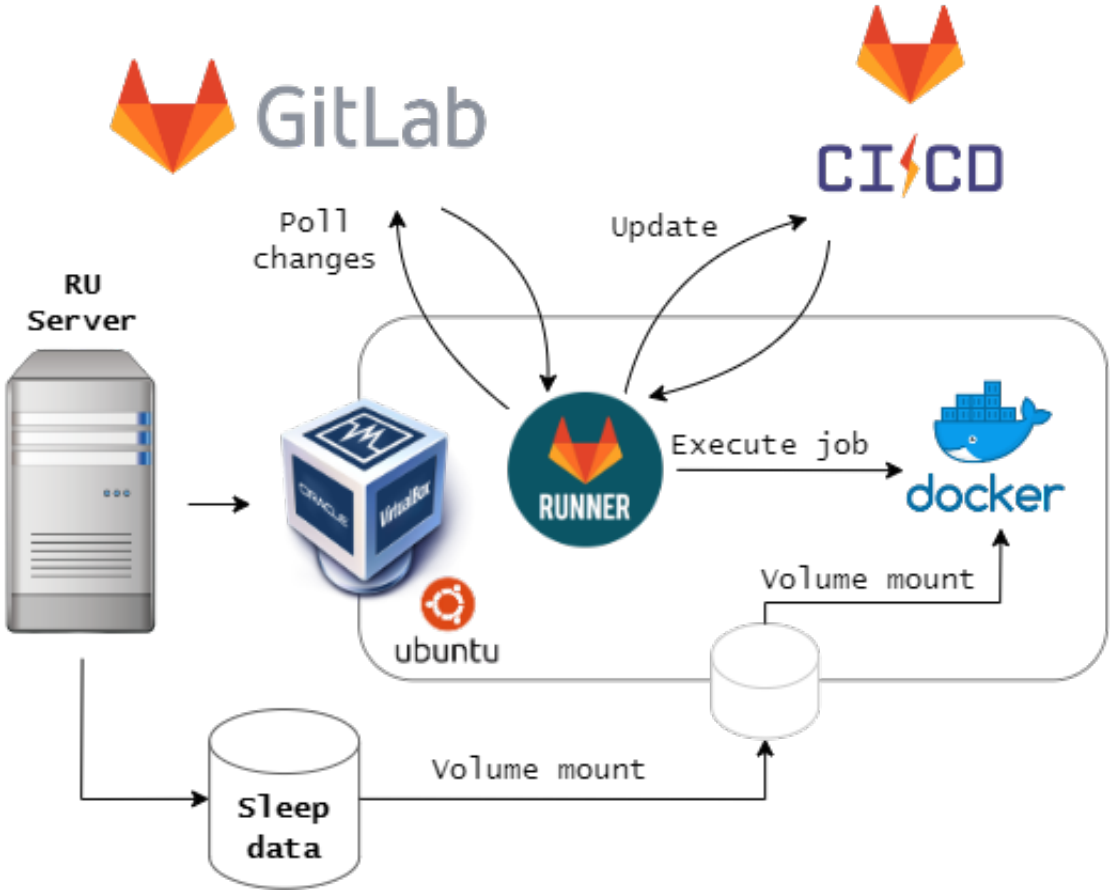

Figure 2: CI/CD solution design

Figure 2 describes the communication between the different technologies. GitLab Runner polls the GitLab repository for changes on a set interval. If changes are detected, GitLab CI/CD is contacted and any tasks configured to run for the current changes are scheduled. The runner can schedule tasks to run both in parallel and in a sequence and moves tasks to execution through docker environments. Once tasks are scheduled, live updates of each individual execution is sent to the GitLab CI/CD interface.

Sleep diagnostic data hosted on the Reykjavík University cluster is attached as a volume mount to the virtual machine server running the pipelines. Further, the sleep diagnostic data is mounted as a read-only volume to all docker execution environments within the virtual machine through runner configuration.

5.2 Arora Project Pipeline

5.2.1 Function

The Arora project pipeline is a task automation configuration that represents a CI/CD implementation carried out on the delivered CI/CD software described in section 5.1. It was created to automate tests, builds and deployments with the goal of shortening the development life-cycle for the Arora development team.

5.2.2 Technology stack

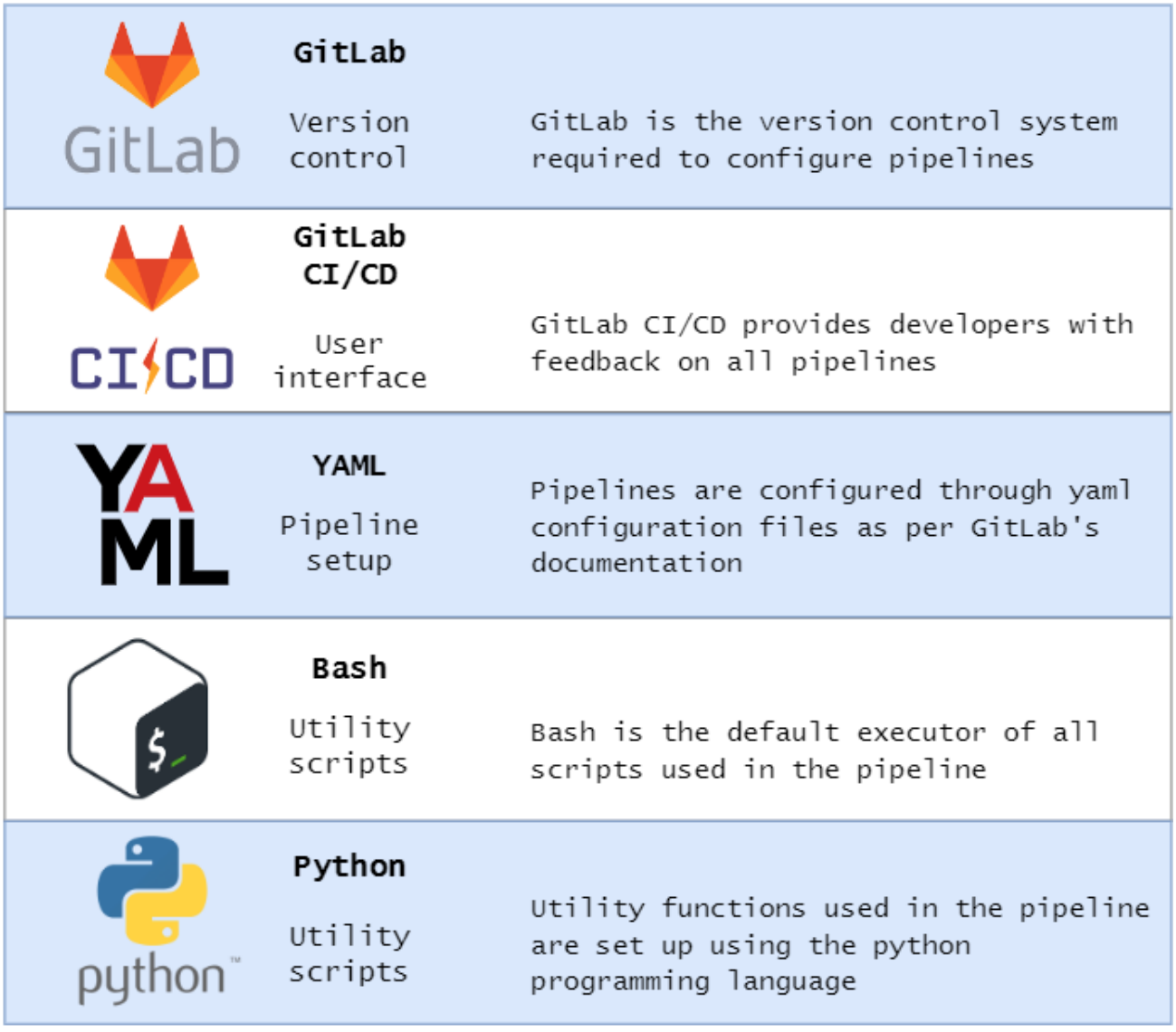

Figure 3: Arora project pipeline tech-stack

Figure 3 shows the technology used for the Arora project pipeline. GitLab is the required version control system to configure pipelines. This means the project must be within a GitLab repository. The user interface for pipelines is provided by Gitlab CI/CD which is accessible from within GitLab projects. YAML is the configuration language used to configure pipelines and additional pipeline utilities are provided through Bash as well as Python scripts.

5.2.3 Design

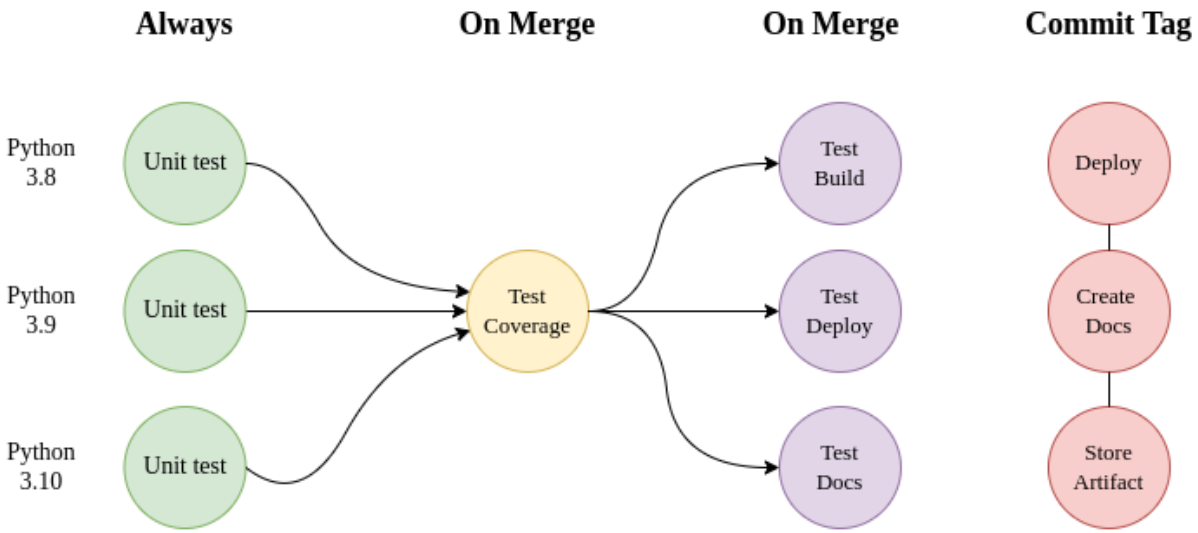

Figure 4: Arora project pipeline design

Figure 4 shows an overview of pipeline tasks available. The arrows denote dependencies and execution order of tasks. The pipeline runs unit tests on all versions of python that the Arora Package aims to support. These test jobs are executed on any change in the repository. The test coverage job is executed only on commits linked to merge requests and produces metadata on source code covered with unit tests. The coverage reached is displayed to the developer in the pipeline status of the merge requests. Test builds, test deployments and test uploads of documentation are also executed on commits linked to merge requests. They make sure that a deployment to production with any new addition to the main branch will always succeed.

Production deployment pipelines are triggered either via manual uploading of commit tags to the repository, or by a deployment script which abstracts the process. In this pipeline, unit tests are also executed for all supported python versions. Artifacts are created in the project's private GitLab registry which serves as a backup for any deployed version of the python package. New documentation is created and published under GitLab pages for the new package.

Any job that fails will terminate the remaining pipeline. The GitLab repository is configured to disallow merging any changes where pipelines have failed. To enforce this even further, direct commits to the production branch are disallowed such that to integrate any new changes, a merge request must be made from a different branch.

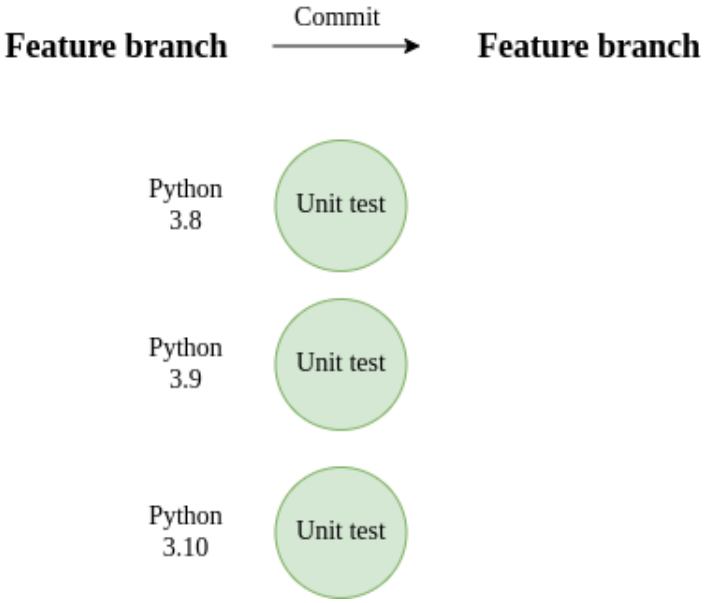

Figure 5: Committing changes to a feature branch

Figure 5 shows the tasks executed when commits are pushed to a feature branch.

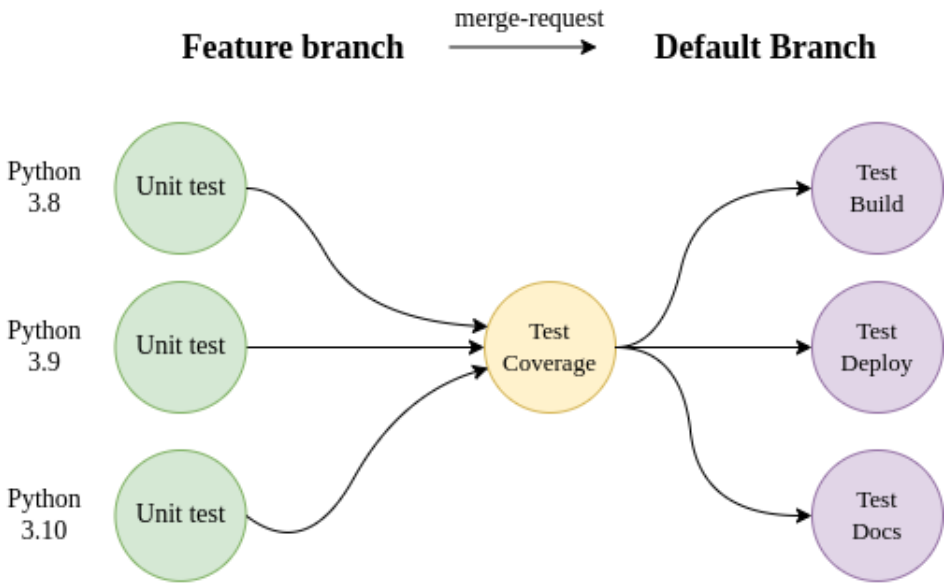

Figure 6: Committing to a merge request

Figure 6 shows the tasks executed when commits related to merge requests are pushed to remote. This is any commit that either adds changes to, or creates a merge request.

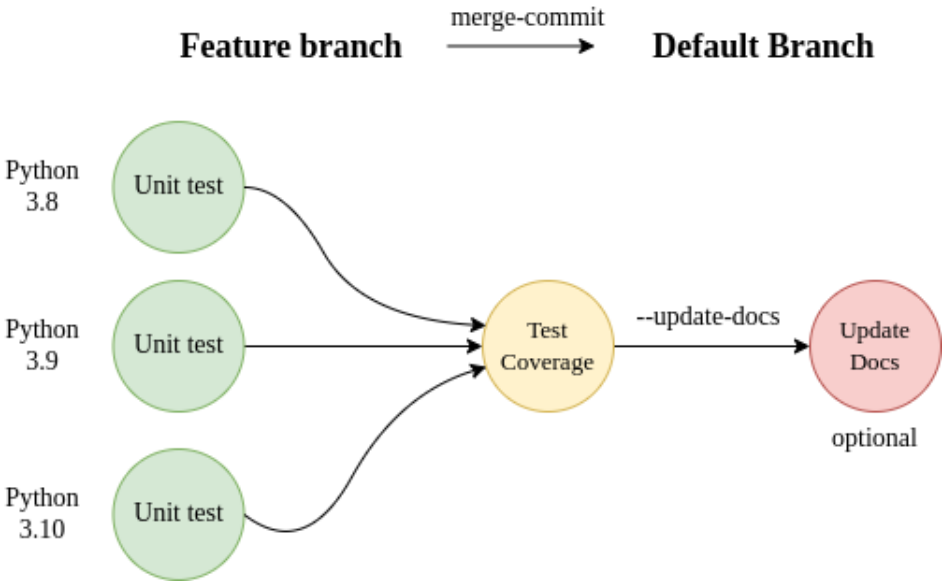

Figure 7: Merging changes into the default branch

Figure 7 shows the tasks executed when changes are merged into the mainline. The `Update Docs` task is an optional feature only triggered if the command `--update-docs` is found in the commit message.

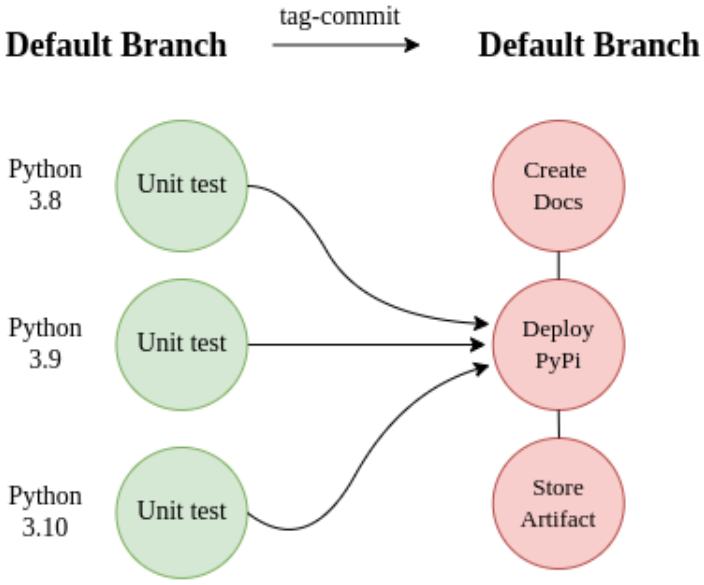

Figure 8: Pushing a commit tag to the default branch

Figure 8 shows the tasks executed in the deployment pipeline. The deployment pipeline is triggered by tag commits with semantic versions are pushed to the remote.

5.3 Remote Development

5.3.1 Function

Sleep revolution projects are dependent on remote machines for access to sleep diagnostic data in development. The remote development solution was implemented to allow projects to maintain normal development practices using the popular PyCharm IDE [11] while developing directly on a remote machine. The solution we define is for Python based projects. JetBrains [10], the provider of PyCharm, offers similar IDE's for multiple other programming languages, thus a similar implementation is possible for other types of projects.

5.3.2 Technology stack

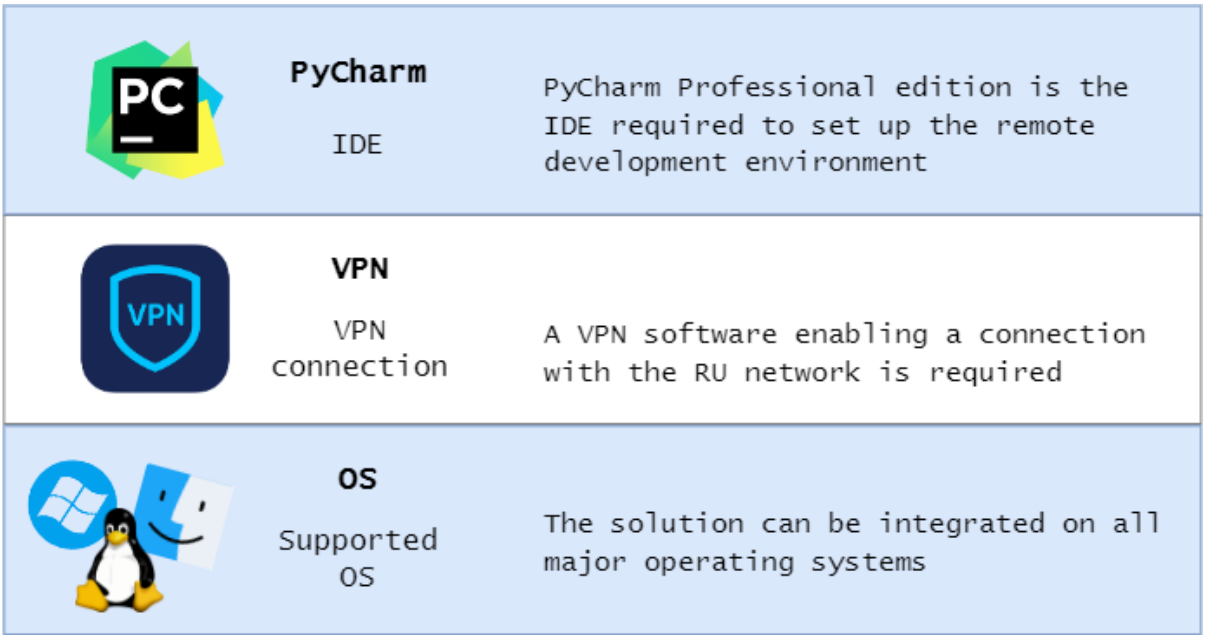

Figure 9: Remote development tech-stack

Figure 9 shows the technology used for remote development. PyCharm is the IDE used for project development and is configured to execute code remotely. A VPN client is used to connect to the Reykjavík University VPN. The solution is supported on all major operating systems.

5.3.3 Design

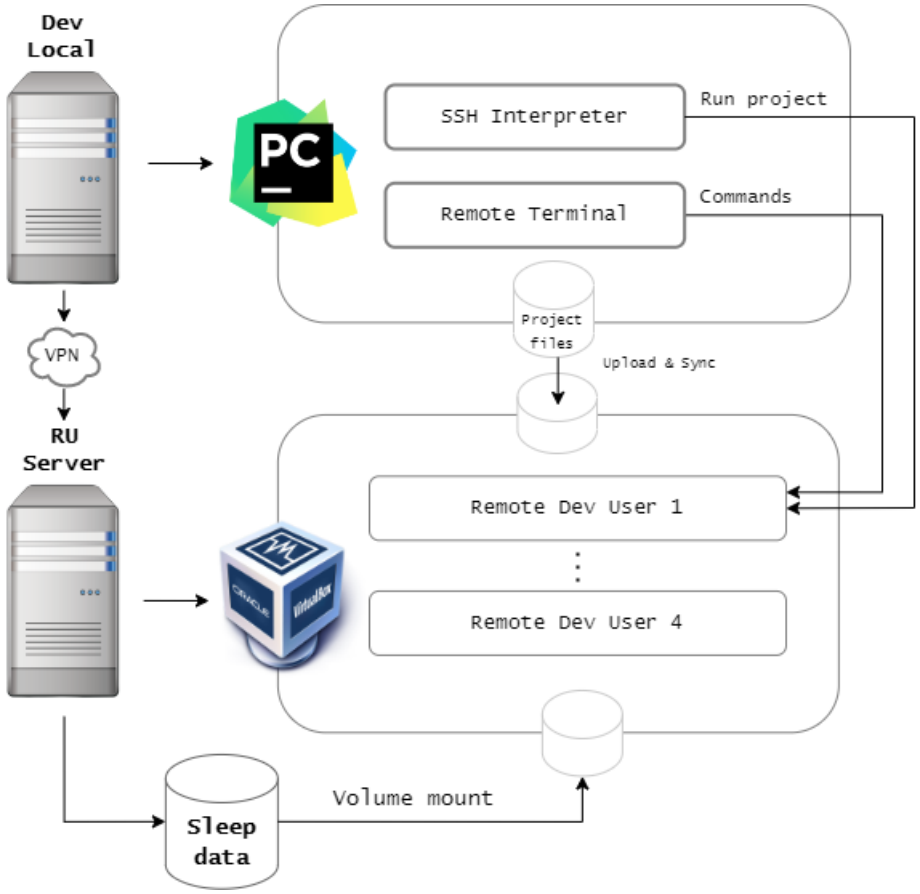

Figure 10: Remote development design

Figure 10 describes the behavior and communication between technologies used in the solution. A locally available project is opened in the PyCharm IDE. Through PyCharm's configuration, a remote interpreter is initialized and connected with an authorized user on the Reykjavík University cluster. Additionally an SSH terminal is configured to allow the user to execute terminal commands on the remote host. Once a connection is established, local project files and directories are uploaded and synced to the remote environment.

All development moving forward can now be done in an identical fashion to local development while also having access to the sleep diagnostic data sets.

5.4 Documentation

5.4.1 Function

The documentation describes all implemented development features. Many developers are completely unfamiliar with the DevOps methodology, and therefore additional, complementary development practices are outlined and best practice workflow is discussed to enhance the DevOps experience.

5.4.2 Technology stack

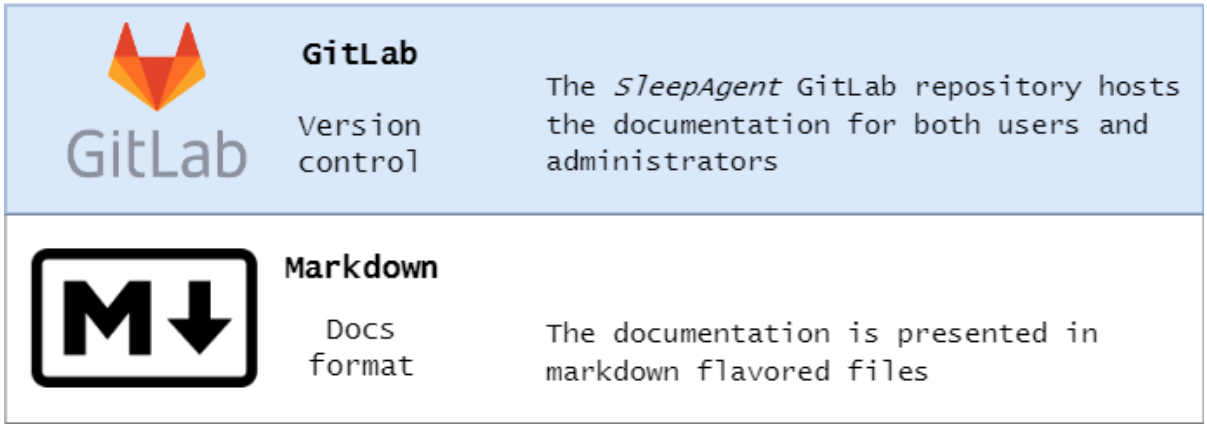

Figure 11: Documentation tech-stack

Figure 11 displays the technologies used for documentation. The documentation is hosted on a GitLab repository within the Sleep Revolution GitLab organization and written in Markdown.

5.4.3 Design

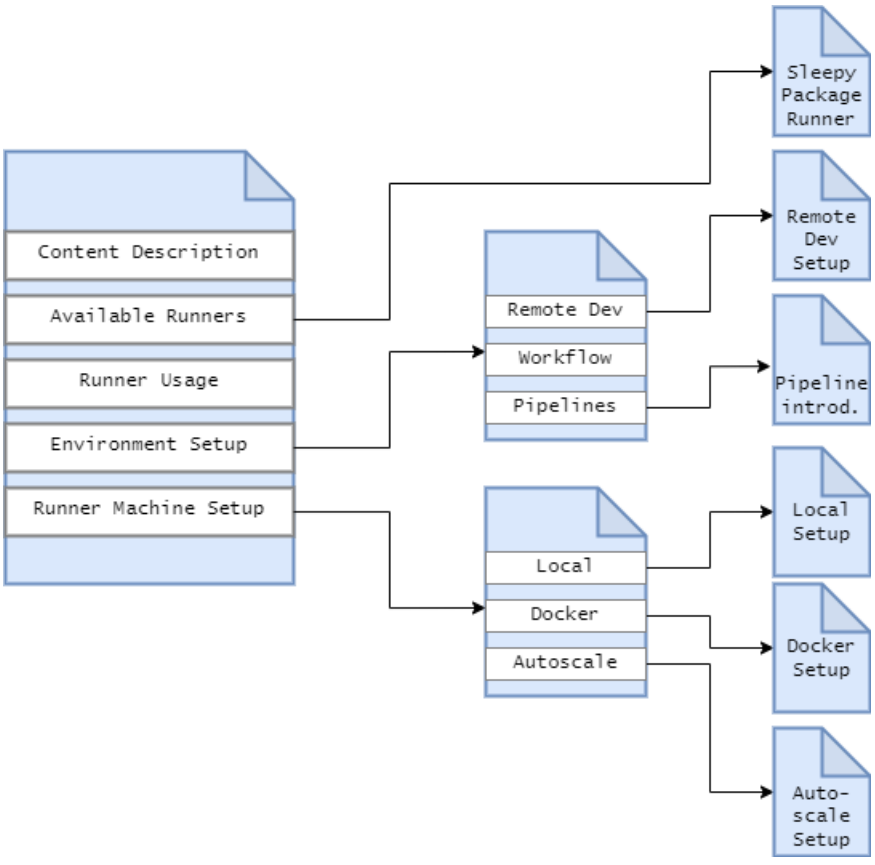

Figure 12: Documentation design

Figure 12 shows the structure, navigation hierarchy and topics of the available documentation.

6 Process

By virtue, a large portion of this project evolved through analytical exploration. This chapter therefore outlines discoveries, gives reason to decisions taken in terms of design and implementation throughout development and demonstrates the steps taken to implement DevOps methodology into a software development project.

6.1 Sprint 0

Realizing the project scope was the first step of the development process. At this stage the main goal was to be able to categorize the needs of our target audience, as well as analyzing the potential benefits from different features that could be implemented within this project. The predefined requirements were analyzed and assessed by researching industry standard implementation strategies. The initial requirements can be seen in Table 7.

Table 7: Initial requirements

No.	Requirement
1	Implement a secure and self hosted DevOps server
2	Run tests on sensitive data hosted on DevOps server
3	Build and deploy python packages on DevOps server
4	Implement other industry standard DevOps features

An initial impression of this research, lead to the conclusion that in a perfect world, the best practice approach to working with sensitive data in development was to sanitize it, minimize it's size and keep it stored in version control. This would cover the main problem this project aims to solve, mainly implementing a secure and self hosted CI/CD server. Having the data in version control would eliminate the need for this type of server, allowing all CI/CD operations to be done through readily available cloud solutions provided by today's most popular version control services such as GitHub or GitLab.

This realization brought some worries about the project's scope, as this would mean setting up configuration files in the Arora project would suffice. After having discussed the matter with the project owners they concluded that this possibility was out of the picture, and all data, in any shape or form, must remain on the RU hosted cluster at all times.

This lead us to the requirement definitions we have today, as presented in Table 6.

6.2 Sprint 1

In the first sprint we focused our efforts on researching available CI/CD solutions. We were looking for a software provider that allowed executing tasks for a version control repository via an external source. It must be externally hosted (self managed), because we must be able to mount sleep diagnostic data from the operating system to the software's execution environment.

6.2.1 Analytical

An initial take on our research resulted in four viable options to be used as underlying software for a self hosted solution. Not to be mistaken with other mentions of GitLab, the options below list the nominated CI/CD providers in no particular order.

- Jenkins [9]
- TeamCity's self managed CI/CD solution [12]
- GitHub Actions using self hosted runners [3]
- GitLab CI/CD using self hosted runners [4]

Further research on each individual solution listed above, favoured GitLab's integrated CI/CD solution as a first choice. Some of many non-exclusively listed reasons for this were; it's reliability, modern approach, readily available documentation, large amounts of preexisting configuration options and user friendliness. To extend the list of reasons why we chose GitLab, it can also be mentioned that it completed all functional requirements of our CI/CD server with it's existing configuration [7] options.

We constrained ourselves to finding a solution that allows developers to modify the CI/CD pipeline behaviour without extensive knowledge of the underlying system or advanced prior experience in CI/CD configuration. This came as a result of the survey 5 taken during the requirement analysis which showed mediocre to low self-assessments when it came to experience with CI/CD. This constraint is also valuable when factoring in time consumed in configuring DevOps procedures, as we did not want this addition to the Developers workflow to lead to a decrease in productivity elsewhere in development. Table 8 represents a high level overview of the results of this research.

Table 8: CI/CD Provider ranking

Provider	Rank	Ease of use	Security	Reliability	Scalability	Learning curve
GitLab CI	1	High	High	High	High	Small
GitHub Actions	3	High	Unknown	High	High	Small
TeamCity	2	Low	High	High	High	Large
Jenkins	4	Very low	Depends	Low	Depends	Very Large

GitHub Actions was ranked as number 2 because of lacking documentation on runner configuration which makes a self hosted implementation more problematic and gives large uncertainty about security measures. TeamCity is ranked as the number 3 option and seems to be a large improvement over it's competitor Jenkins, however their solution would require a complete setup through their scripting language Koltin and has the downside of not being UI supported by the major version control platforms such as GitLab and GitHub. This would move any pipeline feedback to a different platform.

An important note was made about GitLab's security features during this research. Configuring the docker containers, which the GitLab Runner distributes tasks to, as *privileged* can lead to security hazards such as remote execution and session hijacking. All docker containers in the GitLab Runner must therefore be configured as *unprivileged* to prevent such attacks. Luckily this is easily configured.

6.2.2 Implementations

A GitLab source control repository was created and organized with an initial structure for the project including milestones and the product backlog.

An initial attempt at connecting GitLab's CI/CD software from the RU cloud host with a GitLab repository was successful. The software was configured to run GitLab pipelines using a docker-machine executor which houses docker containers inside virtual machines. The virtual machines are spawned on-demand with this setup (auto-scaling [8]). Configuring the software to work beneath the strict VPN rules of the RU cluster resulted in problems with virtual machine spawning because of failing IP assignments.

A local virtual machine replicating the cloud server was set up and the configuration process was documented. The local server was not behind a VPN and successfully imitated our desired goals, which were running python tests on data hosted by the runner's operating system and building a python package for a repository.

6.2.3 Conclusion

Countless attempts were made at fixing the IP assignment problem encountered on the cloud based server. No solutions were found. This problem caused some minor delays in progression as the CI/CD solution was not successfully used on the RU cloud host.

The delays were not critical nor did they require intervention. We were confident that once this issue had been resolved, our prior experience with GitLab's CI/CD features and DevOps would ensure fast paced progress in future sprints.

6.3 Sprint 2

The Arora team encountered delays in producing source code. As a result we were unable to integrate our solution which was the initial goal of the sprint. The sprint was re-planned and we used this opportunity to expand into other areas of DevOps such as improvements of development workflow and working environments.

6.3.1 Analytical

We analyzed the Developers' working environment to see whether improvements in that category could promote productivity.

The Developers require access to data only available on the RU cluster while they work on their project. Development with a modern IDE is therefore not possible unless a solution for remote execution environments is supported. The Developers had resorted to using tools such as Jupyter Notebook for remote work. This is a sub-optimal solution as Jupyter Notebook is a tool for scientific analysis and is not an IDE for project development. The code snippets produced in the Notebooks have to be copied into source code by the Developers where it can no longer be tested for correctness. Apart from being a tedious task, this likely resulted in low confidence in the product.

The code had not been made available in version control as Jupyter does not promote the use of such features by default. The Arora team informed us that they were resorting to sharing notebook files via communication channels such as Facebook Messenger, as it was more time efficient than uploading it to a version control system. Teamwork and organization had therefore been very low, as code written by individual developers had not been readily available to other participants of the project. We suspect this had a significant impact on their progress and we view it as a contributing culprit of their delays. The core concepts of software development in a team had not been present. A new and improved solution was in order.

Research on the topic of remote development quickly gave results. Many modern IDE's support remote development to some extent. It was eventually decided that PyCharm by Jetbrains might be the optimal solution. They support many IDE features for remote development, such as remote docker environments, remote terminals and most importantly, remote interpreters via SSH connection. An implementation of this was delayed until sprint 3.

6.3.2 Implementations

The CI/CD solution was modified to no longer run inside a docker-machine, but rather interact with docker directly. This not only eliminated the VPN problems we encountered in the previous sprint, but also increased execution time of pipeline tasks. The changes were deployed on the remote host and tested successfully against a mock project.

A GitLab group was created and the DevOps project, as well as all of its configuration, was ported over to a new repository within the group. The Arora team was also added to the group where they have their own designated repository. The reason for this move was the overwhelmingly enhanced DevOps experience available on GitLab when compared to GitHub. This also enables the repositories to stay private while using GitLab CI, since integrating GitLab CI with a GitHub repository requires it to be public. Another benefit of this is that the runners configured within the group can be shared and used among all projects contained within the group. This will allow future projects to quickly integrate their own CI/CD experience with our custom runners.

6.3.3 Conclusion

Unfortunately most of the planned tasks this sprint were canceled due to factors out of our control. However we would like to look at this on a positive note, as it widened our sphere of influence and give us an opportunity to have a positive impact on other areas of their development, which quite honestly, direly needed improving. Tasks previously planned for sprint 2 which were not completed were shifted to sprint 3 and we hoped that the Arora team would be able to produce some viable code for us to integrate with.

6.4 Sprint 3

The Arora team encountered another delay. Some code had been uploaded to their git repository, however it was not operational nor complete to any extent. We delayed an integration of our pipeline solution yet again and focused our efforts on providing the remote development solution and finding improvements to their workflow.

6.4.1 Analytical

Some observations were made about the general project structure that was uploaded to the Arora GitLab repository. It seemed to suggest that the team was generally disorganized as even the simplest of features were missing such as a .gitignore file and a well defined working directory for source code. Some files within the repository contained variations of the same methods which likely came as a result of the lack of a conventional IDE and git workflow.

Some of these issues would likely cease to exist once the remote development environment was integrated. To help the team organize we decided it would be best to create some documentation on ideal code structure, git workflow and general conventions for project development. We also decided to help introduce these changes and to refer them to the documentation once the changes were implemented.

6.4.2 Implementations

The PyCharm remote development environment discussed in the previous sprint was created. Members of the Arora team were given very detailed setup instructions which are available within the DevOps repository.

Additional documentation was added for important development features that may frequently get overlooked, such as project structure, git workflow, branch layout and repository restrictions.

6.4.3 Conclusion

The sprint pace was quite unsatisfactory compared to the previous sprints, mainly because of the lack of content to work with, but also because of upcoming exams and the pressure from course deadlines leading up to the final weeks of the 12 week period.

We would have liked to have our solution integrated, as it would have allowed the Developers to pick up their pace. However, what we were confident that what we had brought to the table so far would lead to a fast paced and productive final sprint for the Arora team.

6.5 Sprint 4

Not much had changed in the Arora team's GitLab repository at the very start of this sprint. The pipeline integration still needed tests and working source code and thus we decided to take the initiative in order to kick-start a productive and organized sprint for the Developers. The entirety of the first week of this sprint was therefore spent working together with the Developers.

6.5.1 Analytical

Halfway through the sprint, a minor blockade was reached with using readthedocs [16] as a public host for the package documentation created with Sphinx [18] was not working. The service requires projects to be publicly available, which is something we wanted to avoid. We proposed an alternative by publishing all documentation through GitLab Pages, a free website hosting service for projects provided by GitLab. This gave us more freedom with publishing our documentation which could also come in handy for additional pipeline features.

6.5.2 Implementations

We reorganized the project to a standardized structure. The python package layout was set up and a separate testing module was added along with a gitignore file and a requirements management file that allowed all newer python versions (>=3.8) to install the needed dependencies. Since the tests were configured to run against remote hosted files, we introduced a set of sample tests and created a standard configuration in order for tests to run identically on the local machine and in the pipelines. This allowed the developers to use the sample tests as a template for further test implementations.

After restructuring the project and setting up the remote development environment in PyCharm, the Developers were encouraged to quickly fix any errors with the current source code which allowed us to start integrating the much awaited pipeline. The pipeline came together quite seamlessly. Additional bash scripts and python scripts were implemented for convenience to allow simple builds and deployments to both the test PyPi and the official PyPi package index.

By the end of the first week the pipeline was complete and running tests, builds as well as deployments for the Arora team. Documentation was added to their repository for current and future developers on the usage of those pipelines. More in depth documentation on pipelines such as pipeline customization and pipeline workflow was added to the DevOps repository.

The pipelines for documentation were added by the end of the second week. Each new release of the package creates a new documentation page that reflects the state of the current version. In addition, features were added for updating documentation for the latest released version.

At this stage most of our work had been completed, so we focused our efforts on making some final improvements to our software setup. We automated the complete setup of GitLab's software for running pipelines on an external host by the use of bash scripts and rigorously documented the setup process. The setup was now possible by one terminal command. After completion, we followed up with the Arora team and assisted them with any technical problems.

6.5.3 Conclusion

Productiveness for the Arora team had significantly increased since the beginning of the sprint. We would like to think that our development features contributed a major factor in this leap in productivity.

Moreover we would like to signify the importance of not only development features but also the importance of a more comprehensive relationship between the Developers and DevOps personnel. Without this tightly coupled relationship, the improvements in the development life-cycle in it's entirety would not have been possible.

At this stage we consider our project to be complete.

6.6 Work hour breakdown

Table 9 shows the weekly work hours achieved during the 15 weeks of the semester.

Table 9: Weekly work hour breakdown

Week	Date	Work hours
1	17.1.22 - 23.1.22	9h 50m
2	24.1.22 - 30.1.22	16h 0m
3	31.1.22 - 6.2.22	18h 0m
4	7.2.22 - 13.2.22	5h 0m
5	14.2.22 - 20.2.22	52h 20m
6	21.2.22 - 27.2.22	11h 30m
7	28.2.22 - 6.3.22	9h 0m
8	7.3.22 - 13.3.22	16h 30m
9	14.3.22 - 20.3.22	24h 0m
10	21.3.22 - 27.3.22	8h 40m
11	28.3.22 - 3.4.22	5h 0m
12	4.4.22 - 10.4.22	16h 0m
13	25.4.22 - 1.5.22	62h 30m
14	2.5.22 - 8.5.22	68h 30m
15	9.5.22 - 12.5.22	40h 10m
Total		363h 0m

7 Conclusion

7.1 Our contribution

The aim of this project was to explore the integration of DevOps practices into the existing workflow of the Sleep Revolution project. Through analytical exploration of a development team working within a Sleep Revolution project, we determined various aspects of the development life-cycle which could be improved upon by introducing technical solutions and workflow improvements.

The explored solutions are the backbone to more important subject matters such as a shortened development life-cycle, increased productivity and higher confidence in delivered product [2]. These benefits have undoubtedly been contributed to the Arora project and it's members. We hope our project can do the same for others.

7.2 Limitations

The subject used to reach conclusions about required features and implementations for optimal DevOps practices was a small development team consisting of only 4 participants working on one specific project. The extent to which our contribution is valuable to other Sleep Revolution projects is therefore still uncertain and should be evaluated further.

The pipeline implementation for the Arora project has limited value to other projects within Sleep Revolution. Other projects will have to implement their own pipelines specific to their use case. At most, the implementation can serve as guidance on how to integrate or approach specific features of a pipeline, given that future members of Sleep Revolution have access to the

Arora project's GitLab repository. Even at that, much of the information required for such an implementation will have to be gathered through experience as well as by reading the official GitLab documentation.

The integration of our pipeline to the Arora project was finalized no more than three weeks before the project deadline. Since then, only limited amounts of pipelines have been executed. Due to the late integration, potentially unforeseen issues might arise in the future.

The remote development environment solution has not been tested for projects not compatible with PyCharm. Using other IDEs' provided by JetBrains for different programming languages is therefore not guaranteed to work.

7.3 Future work

The explored solutions should be evaluated for their effectiveness against other Sleep Revolution projects. Doing so will allow improving the provided documentation, discovering features that were previously not required and eliminating any perceived bias obtained from the limited subject size. This will require configuring pipelines for each new subject, tailored specifically to their needs.

The DevOps implementations in the Arora project will have to be continuously maintained throughout the evolution of the project. This will in most cases be manageable by developers and should not require specialized personnel unless major technical reforms are to be made.

7.4 Special thanks

We would like to extend our gratitude to our supervisors María Óskarsdóttir and Benedikt Hólm Þórðarson, and examiner Anna Sigríður Íslind for their continuous support throughout with all aspects of the project.

References

[1] Islind A. S. Arnardottir E. S. and Óskarsdóttir M. The future of sleep measurements: A review and perspective. *Sleep medicine clinics*, 16(3):447–464, 2021.

[2] Buisness2Community. Key benefits of devops. https://www.business2community.com/business-intelligence/9-key-benefits-of-devops-02391855. Accessed: 2022-5-12.

[3] GitHub. Github - ci/cd software. https://docs.github.com/en/actions/hosting-your-own-runners/about-self-hosted-runners. Accessed: 2022-5-12.

[4] GitHub. Gitlab - ci/cd software. https://docs.gitlab.com/runner/. Accessed: 2022-5-12.

[5] GitLab. Gitlab - issue boards. https://docs.gitlab.com/ee/user/project/issue_board.html. Accessed: 2022-5-12.

[6] GitLab. Gitlab - milestones. https://docs.gitlab.com/ee/user/project/milestones/. Accessed: 2022-5-12.

[7] GitLab. Gitlab ci/cd documentation. https://docs.gitlab.com/ee/ci/. Accessed: 2022-5-10.

[8] GitLab. Runner - autoscaling. https://docs.gitlab.com/runner/configuration/autoscale.html. Accessed: 2022-5-12.

[9] Jenkins. Jenkins - ci/cd software. https://www.jenkins.io/. Accessed: 2022-5-12.

[10] JetBrains. Jetbrains - ide provider. https://www.jetbrains.com/. Accessed: 2022-5-10.

[11] JetBrains. Pycharm ide. https://www.jetbrains.com/pycharm/. Accessed: 2022-5-12.

[12] JetBrains. Teamcity - ci/cd software. https://www.jetbrains.com/teamcity/. Accessed: 2022-5-12.

[13] Persónuvernd. Iceland: Data protection act. https://www.personuvernd.is/information-in-english/greinar/nr/438. Accessed: 2022-1-31.

[14] PyPi. Pypi homepage. https://pypi.org/. Accessed: 2022-5-10.

[15] Python. Python homepage. https://www.python.org/. Accessed: 2022-5-10.

[16] ReadTheDocs. Readthedocs homepage. https://readthedocs.org/. Accessed: 2022-5-8.

[17] Sleep Revolution. Sleep revolution project. http://sleeprevolution.eu/en/home/. Accessed: 2022-5-12.

[18] Sphinx. Sphinx documentation. https://www.sphinx-doc.org/en/master/. Accessed: 2022-5-8.

[19] Wikipedia. Black hat hacker. https://en.wikipedia.org/wiki/Black_hat_(computer_security). Accessed: 2022-2-17.

[20] Wikipedia. Devops. https://en.wikipedia.org/wiki/DevOps. Accessed: 2022-1-31.

[21] Wikipedia. Mockups. https://en.wikipedia.org/wiki/Mockup. Accessed: 2022-5-12.

[22] Wikipedia. Version control systems. https://en.wikipedia.org/wiki/Version_control. Accessed: 2022-5-12.

[23] Wrike. Agile development. https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/. Accessed: 2022-1-31.