Bachelor Thesis

Documentation

LTB Operator

Semester: FS 2023

Version: 1.0
June 30, 2023

Students: Jan Untersander
Tsigereda Nebai Kidane

Advisors: Urs Baumann
Yannick Zwicker

INS | Institute for Network and Security

OST Eastern Switzerland University of Applied Sciences

Department of Computer Science
OST - Eastern Switzerland University of Applied Sciences

1. Abstract

Lab Topology Builder (LTB) is an application developed by the Institute for Networked Security (INS) for research and teaching purposes. It allows the creation of emulated network topologies, comprised of interconnected nodes, such as servers, switches, routers, etc. LTB serves as a vital tool in various courses at the OST, enabling students to gain hands-on experience and understanding of cloud computing, networking, and security concepts. Currently, LTB relies on a ReactJS frontend and a Django Python backend, utilizing KVM/Docker for lab deployment. However, due to accumulated technical debt and the lack of documentation, maintaining the application has become increasingly challenging.

The primary goal of this project is to create an LTB-inspired Kubernetes operator - short LTB Operator, which simplifies the deployment and management of LTB labs. Furthermore, the labs should run completely inside a Kubernetes cluster, which enables effortless scaling, orchestration and monitoring. This is possible by leveraging the capabilities of Kubernetes including automation, dynamic resource allocation, and seamless integration with a wide range of tools.

The LTB Operator is implemented using the Go-based Operator-SDK framework, that provides a set of tools and libraries for building Kubernetes operators. This allows for a streamlined development process and integration with the Kubernetes API. The deployment of VMs in Kubernetes is provided by the KubeVirt project, which extends Kubernetes with virtualization capabilities. KubeVirt allows the deployment and management of KVM based VMs as Kubernetes resources, accessible via the Kubernetes API.

The LTB Operator has the ability to deploy labs consisting of pods (containers) and KubeVirt VMs defined using a YAML file. Lab templates, lab instances, and node types are utilized as custom resources (CRs) to define the lab topology, configuration, and available node options. A simple layer 3 network between the nodes is implemented using Multus-CNI. In addition to that, out-of-band management access to the lab nodes is provided using a web-based terminal and a freely configurable port. The lab's status and remote access details can be obtained via a command-line interface (kubectl). Even though access control is not implemented in this project, the current implementation lays the groundwork for future development, such as defining basic Role-based Access Control (RBAC) policies, by deploying labs within their own namespace.

In conclusion, the LTB Operator offers users a streamlined approach to deploy and effectively manage network emulation labs within a Kubernetes cluster. This integration bridges the gap between network emulation and container orchestration, enhancing the flexibility and scalability of lab environments for research and teaching purposes.

2. Vision

The vision of the LTB Operator project is to revolutionize the deployment and management of emulated network labs by leveraging the power of Kubernetes. We aim to provide students and researchers with a robust and user-friendly platform that simplifies the creation, configuration, and orchestration of complex lab topologies. We strive to empower users to effortlessly scale their labs, automate resource allocation, monitor and easily deploy them by leveraging Kubernetes's rich ecosystem of tools and services.

Through this project, we aspire to bridge the gap between network emulation and container orchestration, enhancing the teaching and research experience in cloud computing, networking, and security domains. Our ultimate goal is to enable efficient, scalable, and extensible lab deployments, fostering innovation and advancing knowledge in the field.

3. Management Summary

3.1. Initial Situation

Lab Topology Builder (LTB) is an application developed by the INS and is used for research and teaching purposes. It can create networking labs, which are emulated network topologies with multiple interconnected nodes (servers, switches, routers, etc.). It is a key component of multiple courses at the OST and is used by students to practice and learn about cloud, networking and security concepts.

Currently, the application is based on a ReactJS frontend and a Django Python backend in combination with KVM/Docker for the deployment of the labs. This solution has grown organically over the years, with multiple contributions from term projects and bachelor theses. Maintaining the application has become increasingly challenging, because of technical debt and missing documentation. Therefore, making it open source would require a substantial refactoring effort.

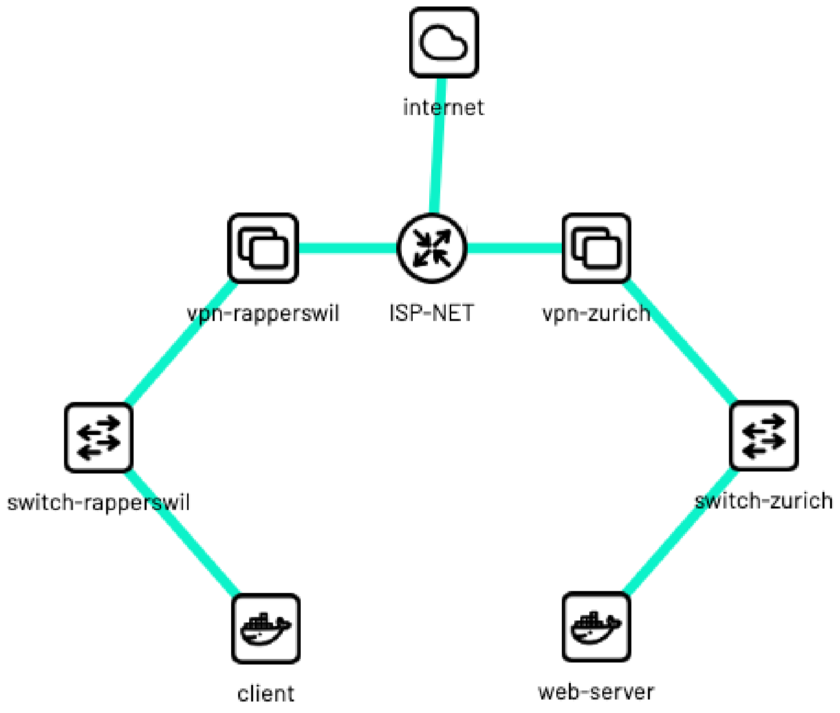


Figure 3.1.: Example of a Lab Topology [1]

3.2. Objective

With the overall goal of open sourcing the application, the objective of this project is to create an LTB-inspired Kubernetes operator in order to replace the existing backend and deployment mechanism of LTB. This Kubernetes operator will be responsible for deploying and managing the aforementioned labs on a Kubernetes cluster. Using Kubernetes offers simple automation, dynamic resource allocation, and seamless integration with a wide range of tools. Thus, making it ideal for managing complex applications while enabling effortless scaling and orchestration.

The LTB Operator should be able to manage labs consisting of pods (containers) and KubeVirt virtual machines and should accept a lab definition in the form of a YAML file. The different lab nodes should be able to communicate with each other via a layer 3 connection. The LTB Operator should also provide a feature to query the status of a lab. Additionally, the lab nodes should support out-of-band management through various protocols. Optionally, access to a lab should be restricted to specific users using access control policies.

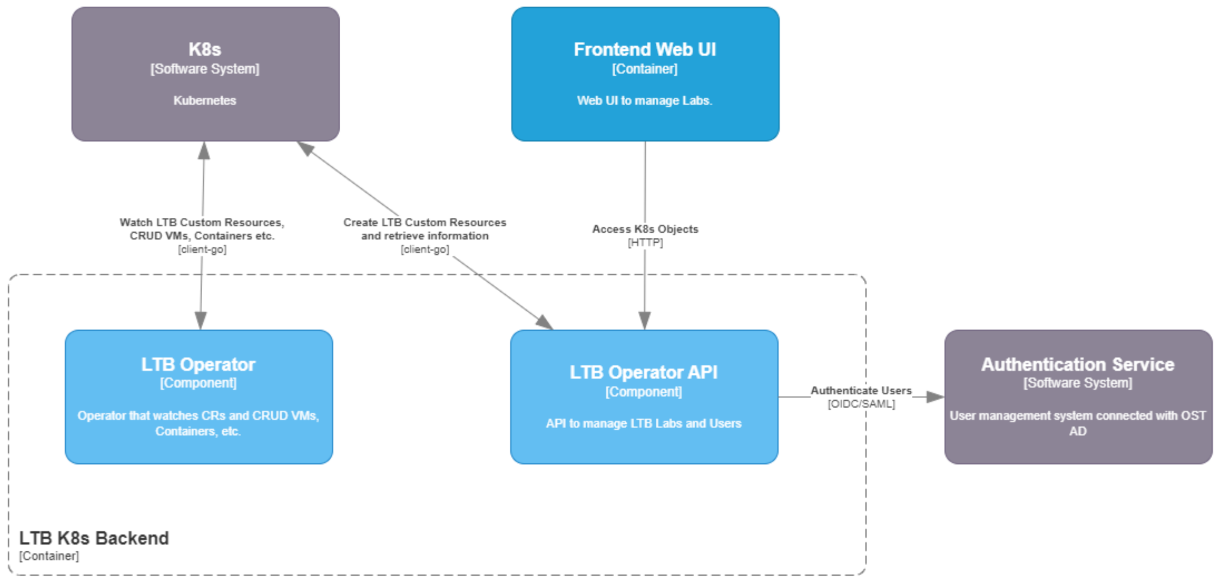


Figure 3.2.: LTB Operator Components and Connection to External Systems

3.3. Results

The project resulted in a Kubernetes operator, that takes three different custom resources (CRs) as input: lab templates, lab instances, node types.

The lab templates define the lab topology i.e. the configuration of the nodes and the connection between them. The node types define the different types of nodes that can be used in a lab. Finally, the lab instances are the actual labs that are deployed. Lab instances reference a lab template, that in turn references node types.

There is support for IP-based networking between the lab nodes limited to a single physical Kubernetes node. Access to the out-of-band management of the lab nodes is provided via a web-based terminal or a freely configurable port, that will be forwarded to the lab node. Information regarding the lab's status and remote access details can be obtained via a command-line interface (kubectl).

Access control has not been implemented yet, but it is anticipated to be easily implemented in the future since the current implementation has already been designed with this in mind.

In summary, the LTB Operator enables users to deploy and efficiently manage labs within a Kubernetes cluster.

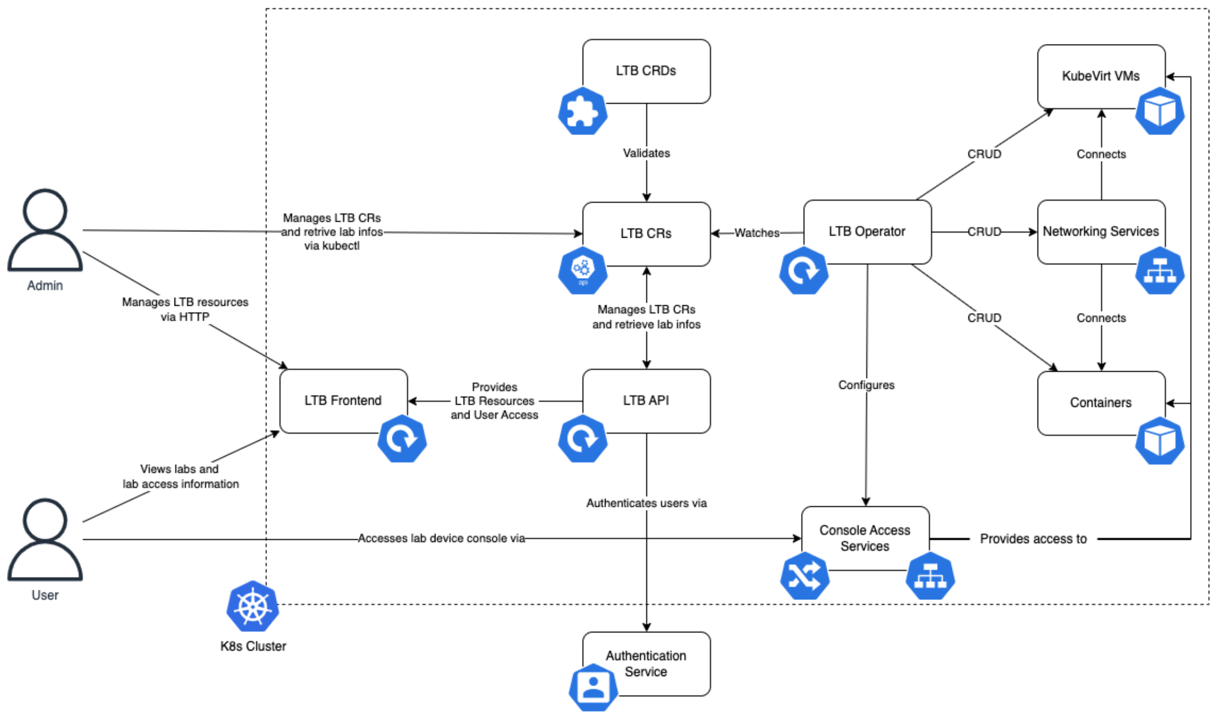


Figure 3.3.: LTB Operator Architecture

4. Acknowledgements

We would like to thank our supervisors Urs Baumann and Yannick Zwicker for their invaluable guidance and support throughout the course of this project. Their expertise, insightful feedback, and dedication have played an instrumental role in refining our ideas and enhancing the overall quality of this thesis. We would also like to thank our friends Dominik Ehrle, Tim Trachsel, Samson Kifleyesus Fanuel, Andreas Yosef and Anwar Hithnawi for proofreading the project documentation and providing us with valuable feedback. Lastly, we would like to thank Luzia Kündig and Olivier Lischer for the exchange of ideas and insight into their bachelor thesis.

Contents

1. Abstract **2**

2. Vision **3**

3. Management Summary **4**
3.1. Initial Situation 4
3.2. Objective 4
3.3. Results 5

4. Acknowledgements **7**

I. Technical Documentation **10**

1. Overview **11**

II. Project Documentation **12**

1. Requirements **13**
1.1. Goals of the project 13
1.2. Functional Requirements 13
1.2.1. Personas 13
1.2.2. User stories 14
1.3. Non-Functional Requirements 16

2. Preliminary Work **17**
2.1. KVM/Docker based Lab Topology Builder 17
2.2. K8s Lab Topology Builder 17

3. Quality Measures **18**
3.1. Conventions 18
3.1.1. Git Workflow 18
3.2. Quality Tracking 18
3.2.1. Organizational 18
3.2.2. Functional Requirements 18
3.2.3. Non-Functional Requirements 18

4. Results **19**
4.1. In Scope Requirements 19
4.1.1. Functional Requirements (User Stories) 19
4.1.2. Non-Functional Requirements 19
4.2. Implementation Out of Scope 20
4.3. Out of Scope 20

5. Conclusion **21**

6. Project Planning **22**
6.1. Project Plan . . . 22
6.1.1. Purpose . . . 22
6.1.2. Organization . . . 22
6.1.3. Timeline . . . 22
6.1.4. Processes . . . 23
6.1.5. Meetings and Sprints . . . 23
6.1.6. Roles . . . 24
6.1.7. Work Items and Releases . . . 24
6.1.7.1. Work Items . . . 24
6.1.7.2. Releases . . . 24
6.2. Planning Tools . . . 24
6.2.1. Documentation . . . 25
6.2.2. Communication . . . 25
6.3. Risk Management . . . 25
6.3.1. Risk Identification and Definition . . . 25
6.3.2. Risk Analysis Matrix - Initial Version . . . 26
6.3.3. Risk Assessment and Mitigation . . . 27
6.3.4. Risk Analysis Matrix - Final Version . . . 28

7. Time Tracking **30**

III. Appendix **31**

List of Figures **32**

List of Tables **33**

Glosary **34**

Abbreviations **35**

Bibliography **36**

Personal Report **37**

Meeting Minutes **46**

Time Tracking **50**

LTB Operator Goals **55**

Part I.

Technical Documentation

1. Overview

As mentioned in the management summary, the project will become open source in the near future, thus the documentation is written in a way that it facilitates the open sourcing process of the project. We separated the documentation into a technical and project documentation, since a bachelor thesis typically requires the inclusion of specific project management aspects that may not align with an open source documentation format, and it is also stated in our thesis assignment. We have decided to publish the technical documentation in GitHub pages.

Technical Documentation

Part II.

Project Documentation

1. Requirements

Definition of Done (DoD)

- **Code Review**: Work is reviewed and approved by at least one other team member.
- **Functional Validation**: The functionality of the work is verified and validated against the defined requirements.
- **Unit Testing**: Code is accompanied by comprehensive unit tests with high coverage.
- **Deployment Readiness**: The work is ready for deployment, with necessary build artifacts, configurations, and dependencies properly managed.
- **Continuous Integration**: The work is integrated into the continuous integration system and passes all automated checks and tests.
- **Documentation**: Sufficient and up-to-date documentation is provided, including code comments, and user guides.

1.1. Goals of the project

The project goals, defined by the product owner, outline the desired outcomes and objectives we aim to achieve. By clearly understanding and aligning ourselves with these goals, we can focus our efforts on delivering a successful project that meets the expectations of the product owner. The following list shows the goals that we were given by the product owner, more details could be found in the appendix LTB-Operator-Goals.

G-1 Analysis Architecture separation Orchestrator/UI vs Operator/K8s

G-2 Document how lab scheduling/resource reservation could be implemented (implementation out of scope)

G-3 Create K8s Operator

G-4 Deployment/Destruction of lab instances

G-5 Remote access to VMs/Containers

G-6 Status of lab instances query

G-7 Access control (Optional)

1.2. Functional Requirements

1.2.1. Personas

The following personas are defined for the project:

Admin A person who wants to provide a multitude of different lab environments of interconnected routers, switches, servers, etc. to the users. An admin can also be a user. An admin typically has advanced technical knowledge and permissions to configure the lab environment.

User A person who wants to use a lab environment, which requires a topology of interconnected routers, switches, servers, etc.

1.2.2. User stories

In this section, we define the user stories for the project. These user stories are written from the perspective of the previously defined personas. Three scopes are used to categorize the user stories: (1) in scope, user stories that will be implemented in this project; (2) implementation out of scope, user stories that won't be implemented in this project, but the way they could be implemented will be documented; (3) out of scope, which means neither implementation nor documentation of the user stories will be done in this project, but their implementation should not be prohibited by in this project.

U-1	As an Admin, I want to have an operator which manages the resources for me, so that I don't have to do the management manually.	In Scope
U-2	As an Admin, I want to be able to create a lab template of a network topology of interconnected nodes, so that the configuration can be reused for multiple lab instances.	In Scope
U-3	As an Admin, I want to be able to delete no longer needed lab templates, so that I can keep a tidy workspace.	In Scope
U-4	As a User, I want to be able to deploy a lab instance from a lab template, so that I can use the lab environment to improve my networking and security knowledge.	In Scope
U-5	As a User, I want to to be able to delete my lab instances and all the resources related to them, so that I can clean up no longer needed lab instances.	In Scope
U-6	As an Admin, I want to be able to delete any lab instance and all the resources related to it, so that I can free resources, remove no longer needed, broken or orphaned lab instances.	In Scope
U-7	As an Admin, I want to be able to modify (update) any lab instance, so that I can make sure resources are used properly or fix broken lab instances.	Out of Scope
U-8	As a User, I want to be able to stop a lab instance, so that I can save resources while a lab is not needed and continue working from the current state at a later time.	Out of Scope
U-9	As an Admin, I want to be able to create node types in different versions (ex. Ubuntu 20.04 and Ubuntu 22.04), so that I can choose which version I want to use in a lab template.	In Scope
U-10	As an Admin, I want to be able to create node types which are based on containers or virtual machines, so that I can provide a wider range of node types (ex. Cisco CSR 1000v).	In Scope

U-11	As an Admin, I want to be able to modify a node type's configuration (example, upgrade to the latest software version for that image), so that I can provide up-to-date images.	In Scope
U-12	As an Admin, I want to be able to delete a node type, so that I can free storage space and remove no longer needed node types.	In Scope
U-13	As a User, I want to have a L3 connection between lab nodes, so that I can improve my knowledge of interconnected systems.	In Scope
U-14	As a User, I want to have a L2 connection between lab nodes, so that I can improve my knowledge of interconnected systems on a lower level.	Out of Scope
U-15	As a User, I want to be able to create a reservation, so that I can plan a lab's deployment ahead of time and reserve the resources needed for my lab.	Implementation out of Scope
U-16	As an Admin, I want to be able to create a reservation, so that I can plan a lab's deployment ahead of time and reserve the resources needed for a lab.	Implementation out of Scope
U-17	As a User, I want to be able to update my reservations, so that I don't have to delete and recreate them.	Implementation out of Scope
U-18	As an Admin, I want to be able to update any reservations, so that I don't have to delete and recreate them.	Implementation out of Scope
U-19	As an Admin, I want to be able to delete my reservations, so that I can make the resources available for others to use.	Implementation out of Scope
U-20	As an Admin, I want to be able to delete any reservations, so that I can make the resources available for others to use.	Implementation out of Scope
U-21	As a User, I want to be able to see the status of my lab instance, so that I can see if my lab instance is running or not.	In Scope
U-22	As a User, I want to have remote access to the lab nodes (container and VMs) via freely configurable ports (example, SSH, HTTP), so that I can use them for educational and testing purposes (academic exercises, testing of networking features/designs).	In Scope
U-23	As an Admin, I want be able to restrict users to only have access to their own lab instances, so that users can't access other users' lab instances and resources (containers, VMs, etc.).	In Scope (optional)
U-24	As a User, I want to be able to capture traffic between any two nodes in my lab instance using Wireshark, so that I can analyze the traffic sent between the nodes.	Out of Scope

1.3. Non-Functional Requirements

This section defines the non-functional requirements. The same scopes as in the functional requirements (1.2.2) are used. As the functional requirements were the main focus of the advisors and product owner, the non-functional requirements are kept to a minimum.

NFR-1	**Security**: The application follows the Least Privilege principle and Kubernetes operator security best practices.	In Scope
NFR-2	**Performance**: The lab instances of a size of up to 10 nodes, should be deployed and running within 10 minutes.	In Scope

2. Preliminary Work

There already exist a KVM/Docker based Lab Topology Builder, built by the INS and multiple contributions of students via term projects and bachelor theses. It has a rich feature set, but has become difficult to maintain and extend. This is due to technical debt and a lack of documentation.

2.1. KVM/Docker based Lab Topology Builder

The architecture and components of the predecessor and their functionalities are described in the technical documentation of this project.

2.2. K8s Lab Topology Builder

The main goal of the K8s Lab Topology Builder is to fully replace the backend and deployment mechanisms of the KVM/Docker based LTB and update its frontend. In this thesis we aim to implement the deployment mechanism. Kubernetes is used as a new technology with the goal of providing a more scalable and more extensible version of the LTB. The architecture and components of K8s LTB and their functionalities are described in the Kubernetes LTB architecture of the technical documentation.

To sum up, there won't be any new features added to the KVM/Docker based LTB. The K8s LTB will be a replacement for the currently used KVM/Docker based LTB and this thesis will lay the foundation for it.

3. Quality Measures

Some of the quality measures are documented in the technical documentation of the project, whereas the rest are defined in this document. The reason for separating the quality measures into two documents is to make the quality measures that are important for open sourcing the project available to the public, whereas the rest of them are only important for the bachelor thesis.

3.1. Conventions

3.1.1. Git Workflow

The project utilizes the following approach for the git workflow across two repositories: GitLab repository for the project documentation; and GitHub repository for the source code and technical documentation.

- Branch **main** should be used for merging finished features.
- A new branch should be created for each feature and deleted after the feature is merged to the main branch. It should be named after the work item in Jira.
- A merge request should be created after the feature is done and the work is reviewed by the other team member. If the merge request is approved, it can be merged into the main branch and the team member who approved the merge request should set the work item in Jira to "done".
- If a merge request is set to draft, it means that the work is not done and it should not be merged into the main branch yet.

3.2. Quality Tracking

3.2.1. Organizational

Topics discussed in the weekly meetings are recorded in the meeting minutes (Meeting-Minutes). Weekly meetings are held with the advisors to discuss the progress of the project and to get feedback on the work done.

3.2.2. Functional Requirements

The quality of the functional requirements will be tracked on our Jira board. The following guidelines describe the process of implementing the requirements:

- A task or work item is created in Jira out of the user stories.
- The task is assigned to one of the team members.
- A pull request is created after implementing the feature (task), the other team member reviews the work and approves the pull request. Then the task will be set to "done".

3.2.3. Non-Functional Requirements

The quality of the non-functional requirements will also be tracked regularly.

- **NFR-1, Security**: Kubernetes security best practices are followed to ensure that this requirement is met.
- **NFR-2, Performance**: Is tested periodically using lab instances of different sizes.

4. Results

In this chapter we will present an overview of the achieved results in relation to the defined requirements (see Requirements). The following sections outline what has been accomplished and what could be improved or added in the future.

More detailed information on how the LTB Operator is used can be found in the user guide of the technical documentation.

4.1. In Scope Requirements

All the user stories of category "in scope" are implemented. The following list provides a brief overview of how the user stories were implemented.

4.1.1. Functional Requirements (User Stories)

- **U-1**: The LTB Operator is created and capable of managing lab resources.
- **U-2**: The admin can create a YAML file describing the lab template and apply it using the kubectl command. Then, the created lab template can be used by multiple lab instances.
- **U-3**: The admin can delete the lab template by using the kubectl command.
- **U-4**: The user can provide a YAML file describing the lab instance and reference a lab template in it. After applying the file using the kubectl command, the lab instance is deployed by the LTB Operator.
- **U-5**: The user can delete the lab instance by using the kubectl command. Then, the LTB Operator will delete all the resources related to the lab instance.
- **U-6**: The admin can delete any lab instances by using the kubectl command. Then, the LTB Operator will delete all the resources related to those lab instances.
- **U-9**: The admin can create a YAML file describing the node type and specify which version should be used. After applying the file using the kubectl command, the node type is created by the LTB Operator.
- **U-10**: The admin can provide a YAML file describing the node type and apply it using the kubectl command. Then, the node type is created by the LTB Operator.
- **U-11**: The admin can edit the node type or add a new node type, but the edited node type is only applicable to newly deployed lab templates.
- **U-12**: The admin can delete the node type by using the kubectl command, and all the resources related to it will be deleted by the LTB Operator.
- **U-13**: A L3 connection will be created between the lab nodes of each lab instance.
- **U-21**: By running a kubectl "get <lab-instance-name>" command, the user can see the status of the lab instance, including the number of nodes (pods/VMs) running in the lab instance.
- **U-22**: Using the provided web-based terminal, the user can access the console of a lab node. Other freely configurable ports for OOB management access can be provided in the lab template. Information regarding the remote access can be retrieved via kubectl.
- **U-23**: The lab instance will be created in its own namespace, which simplifies the creation of network policies to restrict traffic. In the future, the LTB Operator could also create the required network policies.

4.1.2. Non-Functional Requirements

- **NFR-1**: The LTB Operator is currently running in cluster scope, with RBAC rules to limit its access to the required resources. The CRs in the other hand are running in namespace scope,

which provides even more security.
- **NFR-2**: The lab instances are deployed and running within a few seconds and we have not encountered any issues with the performance.

4.2. Implementation Out of Scope

How the user stories of category "implementation out of scope" could be implemented is documented in the Kubernetes LTB architecture of the technical documentation.

4.3. Out of Scope

The current project should enable the later implementation of all out of scope user stories, and we anticipate only minor refactoring to be required.

5. Conclusion

The LTB Operator project aimed to address the challenges associated with maintaining the existing KVM/Docker-based LTB application and provide an enhanced solution for deploying and managing network emulation labs. Through the development of the LTB Operator, significant progress has been made in achieving these goals. The LTB Operator is capable of effectively deploying and managing emulated network labs within a Kubernetes cluster. We have successfully accomplished all required goals, as outlined in this (Goals of the project) section.

The project began with the analysis of the KVM/Docker-based LTB to determine which components should be retained and which should be replaced. This analysis resulted in the decision to fully replace the backend of the KVM/Docker-based LTB. Subsequently, the LTB Operator was developed to facilitate the deployment and management of lab instances. Various features were implemented, including remote access capabilities to the nodes (VMs/containers) and the ability to query the status of lab instances. To support these functionalities, three custom resources definitions (CRDs) were created: LabTemplate, LabInstance, and NodeType. Further information on the CRDs can be found in the technical documentation, accessible at concept.

We have laid some groundwork for the optional goal of implementing access control (**G-7**) and therefore further work will be required to fully implement this feature. Specifically, network policies will need to be implemented to restrict access to the lab instances, which should be relatively straightforward as all the lab instances are deployed into their own namespace. Additionally, an API which supports authentication and authorization of users is needed.

The features implemented in this bachelor thesis demonstrate a significant progress, although there are still more features needed to replace the KVM/Docker-based LTB. In the technical documentation, we have outlined additional features that could be incorporated in a future work.

In conclusion, this bachelor thesis has laid a solid foundation for a more scalable and extensible version of the LTB application. The implementation of features categorized as "implementation out of scope" (1.2.2) should be relatively straightforward and require minimal refactoring of the LTB Operator.

6. Project Planning

6.1. Project Plan

6.1.1. Purpose

The purpose of this document is to provide a detailed description of the plan for the project, LTB Operator. The project plan describes how the project is executed, monitored, controlled and finalized.

6.1.2. Organization

People taking part in the project

The table below 6.1 shows the people taking part in the project and their roles.

Table 6.1: People taking part in the project

Project	LTB Operator
Contributors	Jan Untersander & Tsigereda Nebai Kidane
Product Owner	Institute for Networked Security (INS)
Advisors	Urs Baumann & Yannick Zwicker
Proofreader	Mitra Purandare
Co-Examiner	Philip Schmid

6.1.3. Timeline

Important Dates and Figures

In the table below 6.2 the important dates and figures of the project are shown, such as when the project starts and ends, the resources available and the working days.

Table 6.2: Important Dates and Figures

Project Start	22.02.2023
Interim Presentation	28.04.2023
Project Submission	30.06.2023
Final Presentation	25.08.2023
Resources	720h (2 Members * 360h)
Working days	Primarily Wednesday, Thursday and Friday and in the semester break, on every working day

The figure 6.1 shows the timeline of the project in Jira. It will be regularly updated according to the actual status of the project.

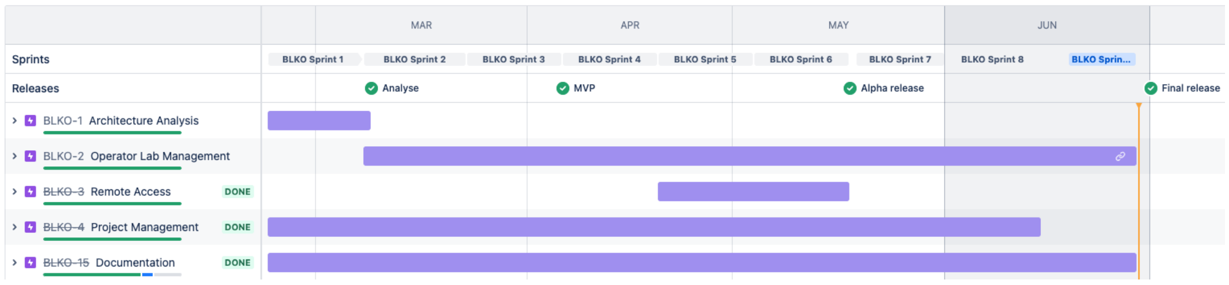


Figure 6.1.: Project Timeline

6.1.4. Processes

The project is managed and structured using the agile development methodology, Scrum. Scrum has been chosen for its flexibility and effectiveness in adapting to changing conditions. By implementing Scrum, we can embrace iterative and incremental development, enabling us to respond to evolving requirements and deliver value in a dynamic environment. In addition to this, Scrum enables us to collaborate efficiently, receive feedback regularly, and continuously improve our work.

6.1.5. Meetings and Sprints

We primarily work on the project on Wednesdays, Thursdays and Fridays together, which enables us to discuss any issues that might occur during the development of the project.

Weekly Meetings Team members and advisors will meet weekly on Friday 10:45 - 11:45 to discuss the progress of the project and any issues that might occur.

- **Jan Untersander** leads the meetings.
- **Tsigereda Nebai Kidane** takes notes.

Moreover, we discuss the inputs from the advisors after every meeting and update the project plan accordingly. The notes from the meetings are attached in the appendix (Meeting Minutes).

Sprint Sprints are two weeks long and tracked in Jira. We have a total of 9 sprints.

Daily Daily stand-ups take place only on Wednesdays, Thursdays and Fridays as the team members are working together on those days. The team members discuss the tasks they have completed, the tasks they are working on, and any issues that are blocking them.

Sprint Planning The next sprint is planned and workloads are estimated using story points. The sprint planning is held after the sprint review and retrospective.

Sprint Review The sprint review is held on the end of each sprint. The team members discuss the tasks they have completed and which tasks they were not able to complete.

Sprint Retrospective The sprint retrospective takes place after the sprint review and the team members reflect the sprint and discuss the things that need improvement for the next sprint.

6.1.6. Roles

We are not using any specific roles in the project, as the team size is small and the tasks are distributed equally among the team members. Each team member is responsible for their assigned tasks.

6.1.7. Work Items and Releases

6.1.7.1. Work Items

Detailed work items and their current status is tracked in Jira.

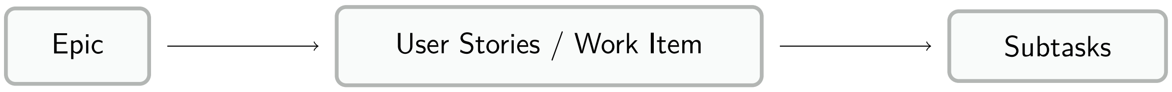
We organize the work items in the following categories:

Epic High level goals of the project, that can be broken down into smaller user stories and tasks.

User Stories are tasks belonging to user-focused features.

Work Items are small actionable units of work, with no direct value to the user, but are necessary for a successful project.

Subtasks Subtasks either belong to work items or User Stories that can be completed in a few hours.



6.1.7.2. Releases

Table 6.3 shows the planned four releases of the project, more about the releases can be found in Jira.

Table 6.3: Project Releases

Release Name	Release Date	Description
Analyse	08-Mar-2023	Current architecture and initial architecture design are analyzed
MVP	05-Apr-2023	Important features are implemented
Alpha Release	17-May-2023	More features are implemented
Final Release	20-Jun-2023	All the features are implemented

6.2. Planning Tools

The following tools are used to plan, manage and track the project.

- **Jira**: is used to track the project issues and project management. We use Jira because it is a popular tool and we wanted to expand our experience with it.
- **toggltrack**: is free and offers simple but efficient task tracking, tagging and reporting.
- **GitLab** & **GitHub**: are used to store the project documentation, and technical documentation and source code respectively. The reason for using two different platforms is that there is a plan to make the project an open source and GitHub is the most popular platform for open source projects. Moreover, it was also suggested by the advisors to use GitHub for the source code and technical documentation. As the project documentation is not part of the open source project, it is stored in a private repository on GitLab.

6.2.1. Documentation

The project documentation is written in Latex and stored on GitLab. This way, the project documentation is versioned and protected from loss. As with the technical documentation, it is written with MkDocs and stored together with the source code on GitHub. Merge requests are used for both documentation and source code changes, to ensure a high quality standard and correctness.

6.2.2. Communication

Team members work together on the above mentioned days, which means they are communicating directly in-person, but for the other working days Microsoft Teams is used.

6.3. Risk Management

6.3.1. Risk Identification and Definition

The following risks may occur in the project. They are sorted according to their over all impact on the project.

R-1 Operator-SDK

- **R-1.1 Learning curve** : We don't have any experience with the Operator-SDK and creating operators. Therefore, it will take some time to learn the tool and create operators.
- **R-1.2 Dependencies**: The Operator-SDK depends on Kubernetes and other related technologies. If there are any issues with the dependencies, then it will be difficult to work with the Operator-SDK.
- **R-1.3 Compatibility & Versioning**: Ensuring compatibility between different versions of the Operator-SDK, Kubernetes, and related dependencies can be challenging.
- **R-1.4 Documentation & Community Support**: To learn the Operator-SDK, it is important to have good documentation and community support. If at least one of them is not good, then it will be difficult and time consuming to learn and use the tool.

R-2 KubeVirt, cloud-init, zero-touch provisioning:

- **R-2.1 Complexity**: To be able to deploy the VMs in Kubernetes clusters, we have to use KubeVirt, cloud-init, and zero-touch provisioning. We are not sure how complex it will be to use these technologies.
- **R-2.2 Dependencies**: The operator will depend on KubeVirt for the virtual machine management, which means that if KubeVirt is not maintained anymore, then the operator will not be able to manage virtual machines.
- **R-2.3 Documentation & Community Support**: At least one of them should be good, otherwise it will be difficult to learn the technologies.

R-3 Connection between nodes: We do not know how the connection between the nodes will be established and if it will be possible to establish a simple connection between the nodes. This is a risk because we might spend a lot of time trying to find a solution.

R-4 Remote access to nodes: Supporting multiple protocols for remote access to the nodes could be challenging because we do not know how we could implement it.

R-5 Deployment: The plan is to deploy the operator in operatorhub.io, but we don't know how easy it will be to deploy it there or if the deployment will be even possible.

R-6 **Understand LTB**: The current LTB doesn't have a documentation, which means we will have to read the code to analyze the application and decide which features need to be replaced. This is a risk because it might take a lot of time to understand the application.

R-7 **Project Management**: We will be using GitHub and GitLab to manage the project, which could be challenging to manage it on both platforms.

R-8 **GitLab and GitHub Outage**: Based on past experiences, GitLab or GitHub outages may occur, so this might somehow affect the progress of the project.

R-9 **Team member absence**: Team members might be absent due to illness or other reasons, which means that the project's progress will be affected.

R-10 **Conflict among team members**: Misunderstanding could happen among team members, which could affect the project's progress.

6.3.2. Risk Analysis Matrix - Initial Version

The following diagrams show the initial risk analysis matrix for the above mentioned risks. The data used in the diagrams is attached in the appendix Risk analysis.

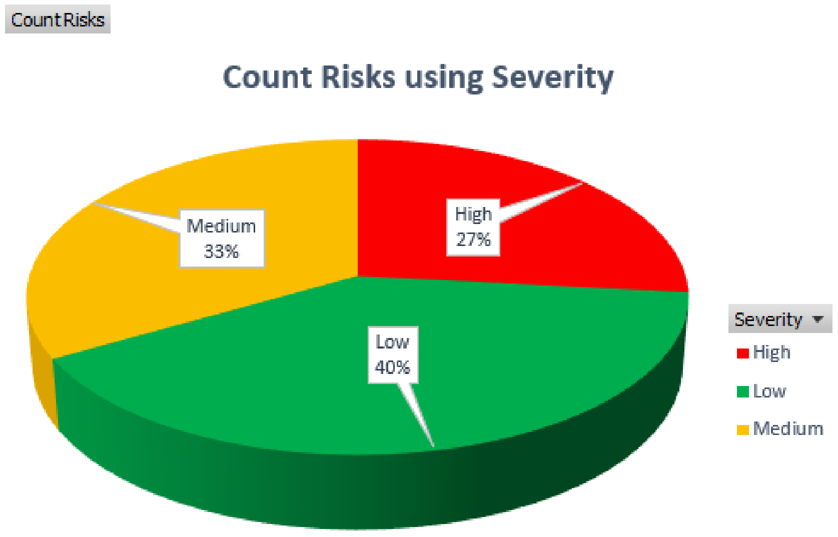


Figure 6.2.: Risk Analysis Matrix 1-1

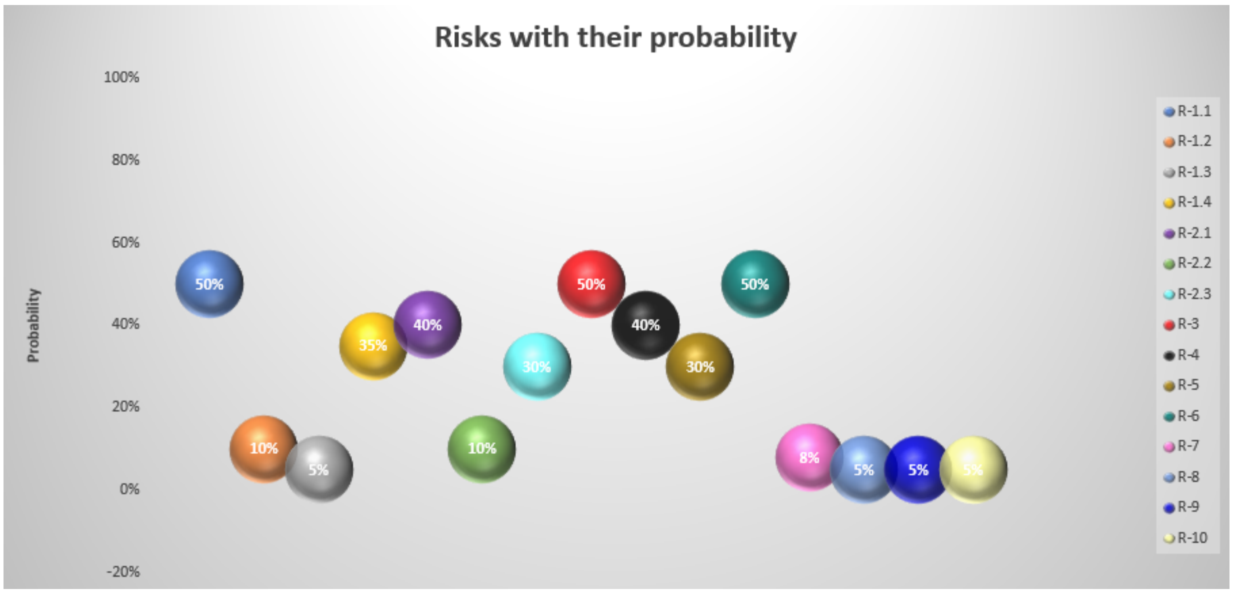


Figure 6.3.: Risk Analysis Matrix 1-3

6.3.3. Risk Assessment and Mitigation

In this section, the risks that were identified in the previous section will be assessed and mitigated, if possible. The risks are sorted the same way as in the previous section.

R-1:

R-1.1: We were able to mitigate this risk in the second sprint by learning the basics of Operator-SDK and operator development concepts through the provided documentation and tutorials. We created sample operators to learn the tool and its components.

R-1.2: For the duration of this project, the technologies we are using are stable and maintained, so we don't have to worry about this risk.

R-1.3: The versions we are using are compatible with each other.

R-1.4: The documentation of Operator-SDK is not really bad, but there are some parts, that are not documented at all or not clear. This risk can not be fully mitigated, but we were able to reduce it through research.

R-2:

R-2.1: This risk was mitigated in sprint 3. We are able to provide KubeVirt VMs with cloud-init configuration and deploy them in Kubernetes clusters.

R-2.2: KubeVirt is well maintained, so we don't have to worry about this risk.

R-2.3: KubeVirt is not well documented, but we mitigated this issue by using the Go package documentation.

R-3: We were informed by the advisors that we don't need to spend much time on this feature, we just need to have a simple connection between the nodes because there is another term project that will be be providing this feature. We mitigated this risk in sprint 4 by using Multus-CNI to create a simple L3 connection between the nodes.

R-4: We mitigated this risk in sprint 4 by creating a web-based terminal to access the nodes.

R-5: Deploying the operator in operatorhub.io is complicated as expected because the documentation is not good and there are some requirements that need to be fulfilled, such as making a pull request on the operatorhub.io repository. We created the necessary files and setups, so that we can deploy the operator there, but the pull request will be made in a later stage.

R-6: We were able to mitigated this risk in sprint 1 by asking the advisors to help us understand all the features of LTB, and where in the code we can find them.

R-7: This risk hasn't occurred, and we could surely say that it won't be a problem because each one of us makes sure to update the project's progress on both platforms.

R-8: GitLab outages has not occurred, but GitHub partial outage occurred once (April 2023), but it wasn't a big problem because it didn't affect the project's progress.

R-9: This risk has occurred once.

- An injury sustained by one of the team members impaired their ability to work on the project at a standard pace, resulting in a delay of the project's progress. In response to the setback, the team requested a deadline extension of two weeks, which was granted by the advisors and the university. This means the project submission deadline is now on the 30th of June 2023 instead of the 16th of June 2023.

R-10: We communicate with each other openly.

6.3.4. Risk Analysis Matrix - Final Version

The following diagrams show the final version of the risk analysis matrix. The data used in the diagrams is attached in the appendix Risk Analysis - Final Version.

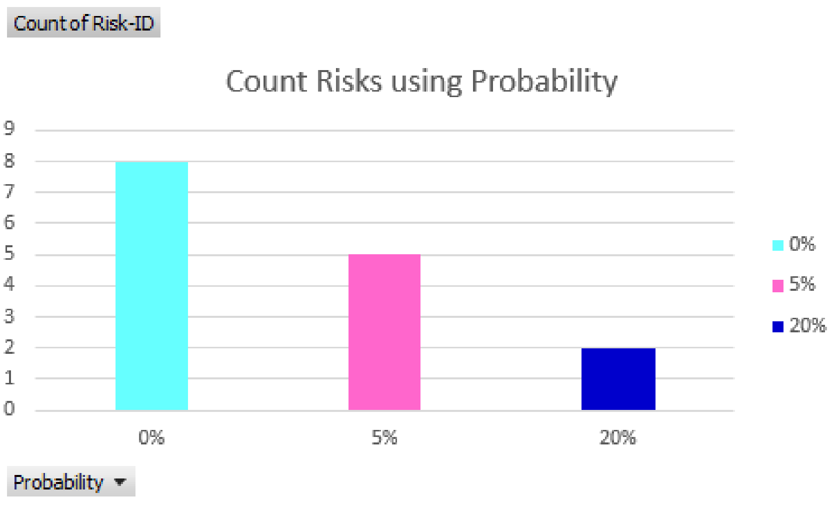


Figure 6.4.: Risk Analysis Matrix 2-1

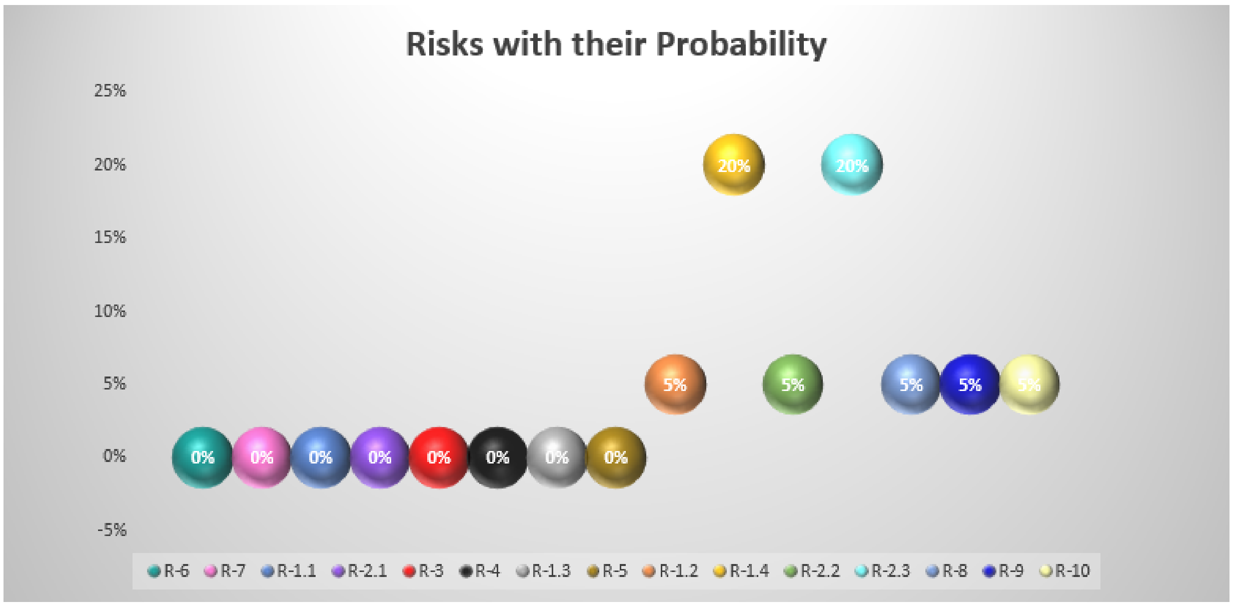


Figure 6.5.: Risk Analysis Matrix 2-3

7. Time Tracking

This chapter shows a summarized overview of the hours spent on the project.

Table 7.1: Time Tracking

Project	**Jan Untersander**	**Tsigereda Nebai Kidane**
BA Coding	211	155
BA Design	20	13
BA Documentation	122	104
BA Meeting	38	37
BA Project Management	18	29
BA Research	0.0	27
Total	**409**	**365**

The detailed time tracking report can be found in the appendix Time tracking.

Part III.

Appendix

List of Figures

3.1. Example of a Lab Topology [1] 4
3.2. LTB Operator Components and Connection to External Systems 5
3.3. LTB Operator Architecture 6

6.1. Project Timeline 23
6.2. Risk Analysis Matrix 1-1 26
6.3. Risk Analysis Matrix 1-3 27
6.4. Risk Analysis Matrix 2-1 28
6.5. Risk Analysis Matrix 2-3 29

1. Risk Analysis - Initial Version 55
2. Risk Analysis - Final Version 55

List of Tables

6.1. People taking part in the project 22
6.2. Important Dates and Figures 22
6.3. Project Releases 24

7.1. Time Tracking 30

Glossary

Django Is a Python-based free and open-source web framework that follows the model-template-views architectural pattern. 2, 4

layer 3 Is IP-based network communication, where packets are routed between different networks using IP addresses. 2, 4

out-of-band management Is the process of managing and monitoring network devices remotely via a dedicated management channel. 2

Scrum Is an agile framework for developing, delivering, and sustaining complex products. 23

Abbreviations

API Application Programming Interface. 2

BA Bachelor Arbeit (Bachelor Thesis). 30

CNI Container Network Interface. 2, 27

CR custom resource. 19

CSR Cloud Service Router. 14

HTTP Hypertext Transfer Protocol. 15

INS Institute for Networked Security. 2, 4, 17, 22

K8s Kubernetes. 13, 17

KVM Kernel-based Virtual Machine. 2, 4, 17, 21

L2 Layer 2. 15

L3 Layer 3. 15, 19, 27

LTB Lab Topology Builder. 2, 4, 5, 17, 19–22, 26, 28

OOB out-of-band. 19

OST Ostschweizer Fachhochschule (Eastern Switzerland University of Applied Sciences). 1, 2, 4

RBAC Role-based Access Control. 2, 19

SDK Software Development Kit. 2, 25, 27

SSH Secure Shell. 15

UI User Interface. 13

VM Virtual Machine. 2, 13, 15, 21, 25

YAML YAML Ain't Markup Language. 2, 4, 19

Bibliography

[1] INS. *Example of a Lab Topology from the KVM/Docker-based LTB frontend.*

Personal Report

Jan Untersander

This project has been an enriching journey that has greatly increased my knowledge and interest in this field. It felt like this project was tailor-made for me, as it combined my passion for Kubernetes, computer networks and software development. Furthermore, I had the chance to broaden my skill set by learning the Go programming language and some of the inner workings of the Kubernetes API, which will undoubtedly be useful in my future career. Collaboration with Tsigereda has been a pleasure and I am very proud of what we have accomplished together. Her dedication and hard work have left a lasting impression on me and I would be happy to work with her again in the future.

Tsigereda Nebai Kidane

Working on this project has been an incredible learning experience for me. I am immensely grateful for the opportunity to work with Jan. Throughout the project, I had the privilege of learning numerous valuable lessons from him. Jan consistently demonstrated his problem-solving expertise, providing innovative ideas and approaches. His patience and willingness to assist me were truly commendable. I am delighted with the outcome of our project and incredibly proud of what we have accomplished together. Moreover, the collaborative work environment was enjoyable, fostering a sense of camaraderie and making the entire experience even more fulfilling.

Meeting Minutes

Kickoff Meeting 22.02.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker

- Receive Bachelor Thesis task description
- Discuss project idea
- Inform that we will use k3d and okteto for development
- Inform that we found Kubebuilder and Operator SDK, which will probably help us develop the operator
- Wireshark feature out of scope
- Make a trakt list for a meeting before one day
- Write vSwitch Openflow flows
- Should labs be distributed across multiple nodes or deployed in one node in a cluster

Questions:

- MkDocs to pdf, can we hand in MkDoc page, instead of pdf? Yes, no Problem from Urs, ask revisors (Gegenleser) if they are ok with it
- Where can we find the current LTB logo? INS-Data Sharepoint

Advisor Meeting 03.03.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker

Progress report:

- Project plan / risks
- Status of dev environment
- Created user stories and NFRs

Planned next steps:

- Refinement of user stories and NFRs
- Finish setup of dev environment
- Create initial version of architecture
- Start implementation of operator

Discussion:

- Possible role distribution: operator(everything else) <-> backend (user management, authentication, connection to frontend?)
- Difference between lab topology and lab template
- Lab topology and lab templates store them as two different CRDs or store "templates" separately somewhere else?

Discussion results:

- Architecture should represent final design, the architectural goal for the bachelor thesis should be defined separately
- Architecture should be based on greenfield approach
- Add Risks for implementation of operator
- Add potential problems or technical difficulties for user stories
- Sort user stories into three categories: in scope, out of scope implementation, out of scope analysis
- Container-lab can be used as a reference for the design and naming of CRDs and their fields
- Potential new name for *device types* could be *kinds*

Advisor Meeting 10.03.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Architecture drafts, (rename of project, consolidation of services)
- State of technical documentation
- Start of implementation moved to this sprint, because there are some dependencies that need to be finished first

Planned next steps:
- CI/Build image creation and initialization of go k8s operator
- Test concepts
- Specification of CRD for lab instances
- Start implementation of the first user stories involving lab instances
- documentation update (design decisions, architecture, user stories, NFRs, risks)

Discussion:
- Open for input

Discussion results:
- LTB K8s Operator and LTB API why separate? - They are the same service
- Rename LTB K8s Operator to LTB K8s Backend - ok for Yannik
- C4 model: mark it as an external service or may be remove it (suggestion from Yannik) - we need to look into it
- Concepts: change the order of the concepts - lab instance, lab deployment, lab template
- Documentation: Make a developer section - Coding conventions, naming conventions, etc. will be inside this section
- OpenVswitch - don't spend too much time on it - it is not a priority
- Remote access - every pod has an interface, so that could be used for remote access
- May be take another issue from the backlog, and work on connections later.
- Research - spec, deploymentTemplate, check how others do it
- ApplicationSet - ArgoCD
- Pair programming maybe an idea in the beginning

Advisor Meeting 17.03.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Operator status (CRD, Controller)
- Demo operator-sdk setup
- Updated architecture
- Updated risks
- Test concepts

Planned next steps:
- Deploy sample labInstance
- General update of documentation (user stories, ADR, etc.)

Discussion:
- Postpone BA hand-in (30.06.2023), presentation (last exam week)
- CRD implementation details
- Other inputs

Discussion results:
- BA hand-in (30.06.2023), presentation (last exam week) - ok for Yannik and Urs
- Meeting next week (24.03.2023) change to Thursday (23.03.2023) 13:30 - 14:30

- Reservation - may be in the generators part
- May be configMap for the templates
- Editable lab instance templates
- Consider other better ways to implement the lab instance templates

Discussion after meeting (Jan and Tsigereda) - 17.03.2023

- Create labTemplates and labInstances separately
- Reference the labTemplate in the labInstance
- LabInstance without the details of template spec.
- Create new apis for labTemplates and labInstances

Advisor Meeting 23.03.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Operator status(CRD)
- Updated user stories
- Updated risks
- Test concepts

Planned next steps:
- Deploy sample labInstance
- Update ADRs (Design decisions)
- Send the user stories to Yannick and Urs for review

Discussion:
- Admin / user: should the user be able to create a labInstances?
- Inputs CRDs
- Mirko Stocker granted a two-week extension for the hand-in deadline, but the poster must still be submitted within the original time frame.

Discussion results:
- Admin / user: should the user be able to create a labInstances? - yes
- Inputs CRDs - Moving connections to be independent of hosts would be a good idea for future extensibility and reduction of repeated definitions
- Crossplane implements a way to reference templates, which could be a good reference for us
- The deadline extension granted, because of the injury of Jan should be documented
- We could create a JSON schema for the labInstance CRD to be used in a code editor like vs code for input validation and hints
- Urs will organize the pre-presentation meeting

Advisor Meeting 31.03.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Operator status(CRD) - Deployment, Status of labs, etc.
- Change in some namings - Host -> Node, etc.

Planned next steps:
- Delete VMs
- Connection between containers/VMs
- General update of documentation (Risks, etc.)

Discussion:
- CRDs for Devices, aka. Nodes
- Other inputs

Discussion results:
- List all advantages and disadvantages of the different approaches, how the device types could be handled.
- Check other better solutions, e.g. for XRD
- See all the different types of the devices and check which one of them are special cases and which fields need to be treated specially.
  - XRD: The env/interfaces => how to that should be implemented?
  - XR - the first interface is for management, the second and third are empty, the rest could be looped for the configuration.
  - Configs: NetPlan, cloud-init, etc.
- (Options): May be create a CR for every device type (XR, XRD, etc.), or create a go type for every device type.

Advisor Meeting 06.04.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Operator status(CRD) - Deployment, Status of labs, etc.

Planned next steps:
- Connection between containers/VMs
- Design decision for node types (XR, XRD, etc.)
- Remote Access
- Tests
- General update of documentation

Discussion:
- Inputs to current progress

Discussion results:
- Namespaces - may be add prefix, let users decide the namespace they want to use
- For the moment, it is ok how it is implemented
- Remote access - check the github repo (sshdocker - INSRapperswil/sshdocker), may be use it.

Advisor Meeting 21.04.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Operator status(CRD) - Deployment, Status of labs, etc.
- Connection between containers/VMs
- Tests
- Identified challenges for node types (XR, XRD, etc.)

Planned next steps:
- Prototype implementation in pure Go and as CR for node types (XR, XRD, etc.)
- Remote Access
- Update of project planning and technical documentation

Discussion:
- Inputs to current progress

Discussion results:

- Presentation
  - Try to impress them in this presentation
  - Takes 20-30 minutes
  - Prepare the documentation in case they want to see it
- Identified challenges for node types (XR, XRD, etc.)
  - How you pass the config is different with XR and IOS
  - XR: the first interface is for management
- Tests
  - Document that the tests would be updated in the future (for open sourcing it)

Advisor Meeting 28.04.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:

- Node types status
- Connection between containers/VMs
- Remote access status

Planned next steps:

- Node types implementation
- Remote Access implementation
- General update of documentation

Discussion:

- Node types (decision)
- Feedback to planned implementation of Remote Access

Discussion results:

- Node types
  - Where does merging the config happen?
  - The CRD option is a good option - easily extendable
  - Container disk images
    - No backing disk: after stopping the container, the changes are lost
    - Download image from a registry
      - Give a http link to the image - will be always downloaded
      - Or caching
    - The one we have looks good (incase of multiple labs deployment)
    - Validation check - make a mapping to valid images
    - Document what happens if we modify the file, and the pod is stopped, what happens next (Will the old configuration be reused, or the pod be recreated with the newest configuration). The same with the VMs too.
    - Final decision: use CRD for node types instead of using go implementation
- Remote Access
  - The labtemplate has a field where ports can be chosen to be exposed
- Access control
  - As a user, I should be the only one who have access my lab (console, etc.)
  - Document that K8s API gateway could be used later (cilium)

Advisor Meeting 05.05.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:

- Remote access
- Documentation (Design decisions, user guide, etc.)

Planned next steps:

- Implementation CRDs for node types
- Code refactoring
- Improvement remote access

Discussion:

- Remote access
- Inputs to current progress

Discussion results:

- Remote access
  - ttyd - Terminal resizing bug
  - Using a load balancer and clusterIp - ok for Urs
- Inputs to current progress
  - Concept documentation: Write the difference between Lab and LabInstance clearly
  - Document the dependencies' versions of the project
- Reserve IP address for the load balancer.
- For beta version
  - Inform the user how the dependencies and LTB kubernetes operator can be installed
- For final release
  - Use helm charts to install the dependencies
- Test how local installation works, example with miniKube/k3d/kind

Advisor Meeting 12.05.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:

- Current Status (Refactoring, prototype CRDs for Node types)
- Documentation (Design decisions, user guide, etc.)

Planned next steps:

- Implementation of CRDs for node types

Discussion:

- Inputs to current progress

Discussion results:

- CRD for Node types' prototype
  - Create some sample node types
- Requirements documentation - leave it in the project documentation for now

Meeting with Philip 17.05.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Philip Schmid
Discuss results:

- Out of band management: how we plan to implement it
- Document what could be implemented in the future, or how that could be implemented (eg. Prometheus)

- Operator: which namespace is it running on? It would be good to deploy in its own namespace for RBAC reasons
- For deployment of the operator, helm chart is a good option
- RBAC: asked a question about network policies
- Important points of documentation: structure, order, easy to read
- The kubernetes concepts: may be keep those till the end of the project, and then reference it to the kubernetes documentation later on.

Advisor Meeting 19.05.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker
Progress report:
- Current Status (Implementation of CRD for Node types)
- Refactoring of the code
- Meeting with Philip (Deployment of the operator will be separate (dependencies should be installed by the user))

Planned next steps:
- Write unit tests
- API implementation
- Abstract, A0 poster, etc.
- Test how local installation works

Discussion:
- Inputs to current progress

Discussion results:
- Refactoring of the code, looks good
- Node types:
  - Planned way of error handling is ok
  - We're not directly using YAML because templating is needed
  - Cloud-init:
  - Make the input as base64 encoded string
  - ConfigMap - reference in nodeTypes, gets the config from there
  - Test the things with the other nodes
  - VM Network should stay within the controller
  - VM Interface configuration can also be made in controller but allow for additional empty interfaces in nodetype
  - We only need to know the number of interfaces in nodetype
  - VM Default Interface will be configured in controller, document this
- API
  - Focus on conceptual idea and analysis of implementation options
  - It should be fully separated from the operator to keep it separated and because Kubernetes is a well defined interface between API and operator
  - Only make a small prototype to test implementation options
- Deployment of the operator will be separate (dependencies should be installed by the user) - ok for Urs too

Advisor Meeting 02.06.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker Progress report:
- Abstract, A0 Poster

- Tests

Planned next steps:
- Finalize tests
- API implementation
- Diagrams (State and update LTB diagram)
- Config node types

Discussion:
- Inputs to current progress

Discussion results:
- Deployment needs to be updated
- Config for node types are important and needs to be prioritized
- License: Apache 2.0 when all the dependencies use the same license - Done
- Meeting of next Friday postponed to Monday 12th of June

Advisor Meeting 12.06.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker Progress report:
- Abstract, A0 Poster
- Tests (Test coverage)

Planned next steps:
- Node types config input
- Simplify deployment
- Code cleanup

Discussion:
- Inputs to current progress

Discussion results:
- A0 poster looks good
- Abstract: Update the text in the tool and upload pictures
- Tests: Looks ok.
- Record a video as a backup for the final presentation if the live demo doesn't work, we can then link the video in the github repo (Tool:asciinema)
- Meeting of next Friday postponed to Monday 19th of June

Advisor Meeting 19.06.2023

Attending: Jan Untersander, Tsigereda Nebai Kidane, Urs Baumann, Yannick Zwicker Progress report:
- Code Cleanup
- Config Input for VMs and Container
- Operator Deployment using OLM
- Tests (Test coverage)

Planned next steps:
- Documentation

Discussion:
- Inputs to current progress

Discussion results:
- Deployment of the operator via OLM - looks good for both of them
- Code cleanup and config input
  - For the moment ok, test with a switch
  - Test for xrd -> INS-Infrastructure repo/container registry

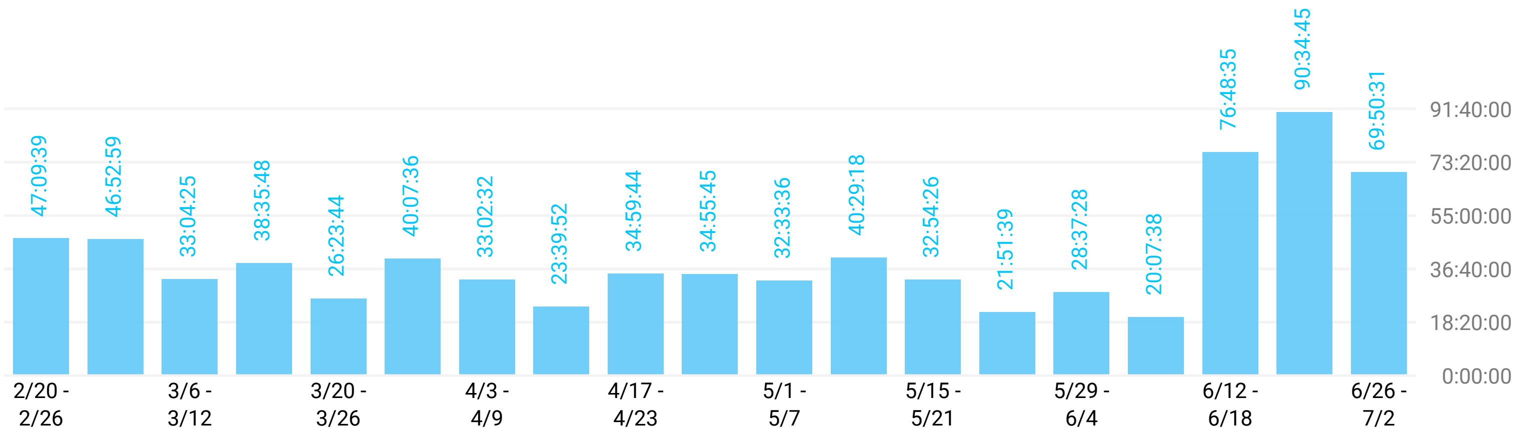
- Test coverage - ok for both of them

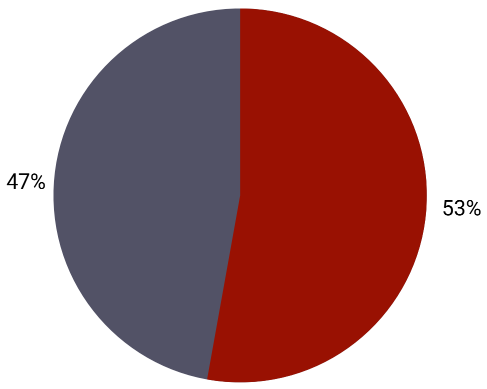
Time Tracking

Summary Report

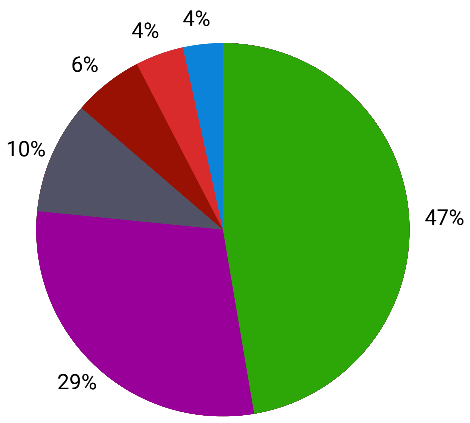
02/20/2023 – 06/30/2023

TOTAL HOURS: 772:40:00





USER	DURATION
JA Jan	408:36:58
TN Tsigereda Nebai	364:03:02



PROJECT	DURATION
● BA Coding	365:42:11
● BA Documentation	226:29:49
● BA Meeting	74:56:53
● BA Project Management	46:14:47
● BA Design	32:12:48
● BA Research	27:03:32

USER - PROJECT	DURATION	PERCENTAGE
JA Jan	408:36:58	52.88%
● BA Coding	211:06:22	27.32%
● BA Design	19:35:18	2.54%
● BA Documentation	122:20:59	15.83%
● BA Meeting	37:55:09	4.91%
● BA Project Management	17:39:10	2.28%
TN Tsigereda Nebai	364:03:02	47.12%
● BA Coding	154:35:49	20.01%
● BA Design	12:37:30	1.63%
● BA Documentation	104:08:50	13.48%
● BA Meeting	37:01:44	4.79%
● BA Project Management	28:35:37	3.7%
● BA Research	27:03:32	3.5%

LTB Operator Assignment

INS | Institute for Network and Security

Bachelor Thesis Assignment

K8s Lab Topology Builder Operator

Version 1.0
March 2, 2023
Institute for Network and Security

1 Assignment

1.1 Supervisor and Expert

This student project will be developed for the Institute for Network and Security at OST internally. It will be supervised by Yannick Zwicker (yannick.zwicker@ost.ch) and Urs Baumann (urs.baumann@ost.ch), OST.

1.2 Students

This project is conducted in the context of the module "Bachelorarbeit" in the department "Informatik" by:

- Tsigereda Nebai Kidane
- Jan Untersander

1.3 Introduction

You have been tasked with designing and developing an open-source tool for deploying and managing network labs in Kubernetes (K8s) using YAML files. The Institute for Network and Security has developed a solution based on React and Django that will serve as inspiration for your work.

The tool should support the deployment and destruction of lab components, remote access to Out-of-Band (OOB) management of lab devices using multiple protocols, and status queries for deployed labs. The tool should also support access control (optional) so that each user can only access their own labs.

The technical documentation for the tool should be written in MkDocs, and all documents should be in English. If time permits, the project should be open-sourced to allow for contributions from the community.

Your solution should adhere to best practices for security, reliability, and scalability.

1.4 Goals of the Project

- Analysis Architecture separation Orchestrator/UI vs Operator/K8s
  - Provide an analysis of the architecture separation between Orchestrator/UI and Operator/K8s for the lab deployment and management tool.
- Lab scheduling/resource reservation (implementation out of scope)
  - Describe how the tool will handle lab scheduling and resource reservation. Explain the considerations and challenges involved in implementing this feature.
- K8s Operator

  - Write an operator that can deploy and destroy the lab components specified in a YAML file. The operator should be able to create pods with containers, Kubevirt VMs, and establish connections between lab components at Layer 3.

- Goal01: Deployment / Destroy
  - The tool takes a YAML schema as input and deploys the lab components defined in the schema as Kubernetes resources, including pods with containers, kubevirt virtual machines, and connections between components at Layer 3. When the user decides to destroy the lab, the tool removes all associated Kubernetes resources. The output includes the created Kubernetes resources and established connections.
- Goal02 Remote access
  - Develop a remote access feature that allows users to access the OOB management of devices using multiple protocols. The feature should also provide a programmable interface that can be used by the UI or CLI to retrieve information about each lab.
- Goal03 Status query
  - Develop a status query feature that allows users to check the deployment status of labs. The status should be one of Finished|Pending|Running|Failed.
- Goal04 Access Control (optional)
  - If time permits, implement an access control feature that restricts user access to their own labs only.

1.5 Documentation

This project must be documented according to the guidelines of the "Informatik" department. This includes all analysis, design, implementation, project management, etc. sections. All documentation is expected to be written in English. The project plan also contains the documentation tasks. All results must be complete in the final upload to the archive server. There is no need to print out the documentation

1.6 Important Dates

⚠ Official documents

Check the official documents and relegments

Date	Event
20.02.2023	Start of the student project
12.06.2023	Hand-in of the abstract using the online tool abstract.rj.ost.ch
16.06.2023 17:00	Final hand-in of the report using the online tool avt.i.ost.ch
TBD (until 31.08.2023)	Presentation

1.7 Evaluation

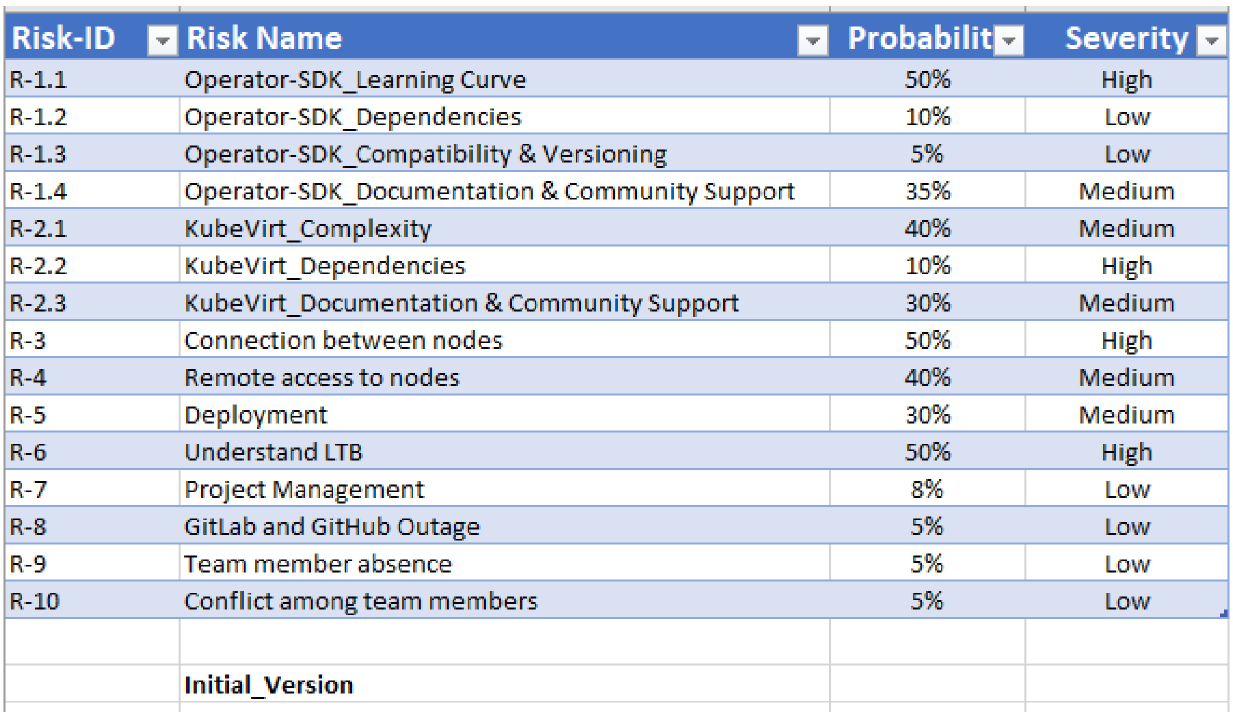
⚠ Official documents

Check the official documents and relegments

Criterion	Weight
Organization and implementation	10%
Formal quality of the report	10 %
Analysis, design and evaluation	20 %
Technical implementation	40 %
Presentation	20 %

Risk Analysis

Initial Version

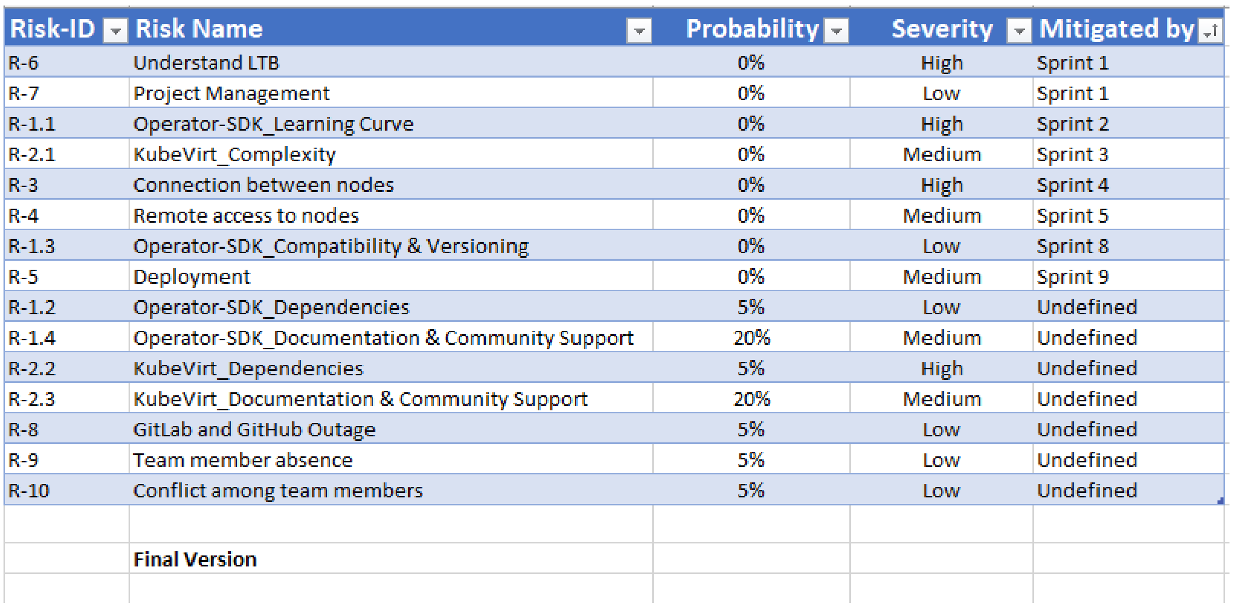


Risk-ID	Risk Name	Probabilit	Severity
R-1.1	Operator-SDK_Learning Curve	50%	High
R-1.2	Operator-SDK_Dependencies	10%	Low
R-1.3	Operator-SDK_Compatibility & Versioning	5%	Low
R-1.4	Operator-SDK_Documentation & Community Support	35%	Medium
R-2.1	KubeVirt_Complexity	40%	Medium
R-2.2	KubeVirt_Dependencies	10%	High
R-2.3	KubeVirt_Documentation & Community Support	30%	Medium
R-3	Connection between nodes	50%	High
R-4	Remote access to nodes	40%	Medium
R-5	Deployment	30%	Medium
R-6	Understand LTB	50%	High
R-7	Project Management	8%	Low
R-8	GitLab and GitHub Outage	5%	Low
R-9	Team member absence	5%	Low
R-10	Conflict among team members	5%	Low

Initial_Version

Figure 1.: Risk Analysis - Initial Version

Final Version



Risk-ID	Risk Name	Probability	Severity	Mitigated by
R-6	Understand LTB	0%	High	Sprint 1
R-7	Project Management	0%	Low	Sprint 1
R-1.1	Operator-SDK_Learning Curve	0%	High	Sprint 2
R-2.1	KubeVirt_Complexity	0%	Medium	Sprint 3
R-3	Connection between nodes	0%	High	Sprint 4
R-4	Remote access to nodes	0%	Medium	Sprint 5
R-1.3	Operator-SDK_Compatibility & Versioning	0%	Low	Sprint 8
R-5	Deployment	0%	Medium	Sprint 9
R-1.2	Operator-SDK_Dependencies	5%	Low	Undefined
R-1.4	Operator-SDK_Documentation & Community Support	20%	Medium	Undefined
R-2.2	KubeVirt_Dependencies	5%	High	Undefined
R-2.3	KubeVirt_Documentation & Community Support	20%	Medium	Undefined
R-8	GitLab and GitHub Outage	5%	Low	Undefined
R-9	Team member absence	5%	Low	Undefined
R-10	Conflict among team members	5%	Low	Undefined

Final Version

Figure 2.: Risk Analysis - Final Version