KubeWatch Documentation

SE Project
Documentation

KubeWatch

Semester: Spring 2022

SE PROJECT

Version: XX.XX
Date: 2022-03-02 20:29:44+01:00
Git Version: a15da46-dirty

Project Team: Benjamin Plattner
Jan Untersander
Olivier Lischer
Pascal Lehmann
Petra Heeb

Project Advisor: Laurent Metzger

OST
Eastern Switzerland
University of Applied Sciences

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Contents

Instructions for using this template

1. The main aim of this document template is to allow you to start working on your project documentation as quickly as possible.

2. In addition to recommending a rough structure for your project documentation, this template also contains instructions related to the content and execution your project. Instructional text is typeset within the LaTeX `instructions` command defined in `custom.tex`. Make sure that your final submission does not contain any instructional texts.

3. The basic defaults that LaTeX provides have been used whenever possible in order to keep the template as simple as possible for beginners. One can do better. LaTeX also provides additional features such as glossaries, abbreviation lists and indexes that you may find useful. Feel free to customise the typesetting and content of this document as per your requirements, tastes and wishes as you progress with the project.

4. Here is an example of a bibliography reference: [Lar04]

Part I

Management Summary

Management Summary

Even though this is the first chapter of your document, it is typically the last one filled with content. The *Management Summary* is a **brief and high-level summary** of your project. It should give any reader unfamiliar to the project an overview of the contents included later in the document.

A common structure is:

- What is the problem we wanted to solve?
- How did we solve the problem?
- Does your solution solve the problem in a successful way?
- Will there be consecutive projects based on your work?

Diagrams and images work very well in this chapter, especially screenshots of your software.

One final remark: a well written management summary is a good starting point for your **Project Presentation**, as you will address a similar audience there.

Part II

Product Documentation

Chapter 1

Vision

Describe the vision for the product on 1 page as covered in the SEP1 module.

Chapter 2

Requirements

Describe the functional and non-functional requirements as covered in the SEP1 and SEP2 modules.

2.1 Functional Requirements

2.2 Non-Functional Requirements

Chapter 3

Domain Analysis

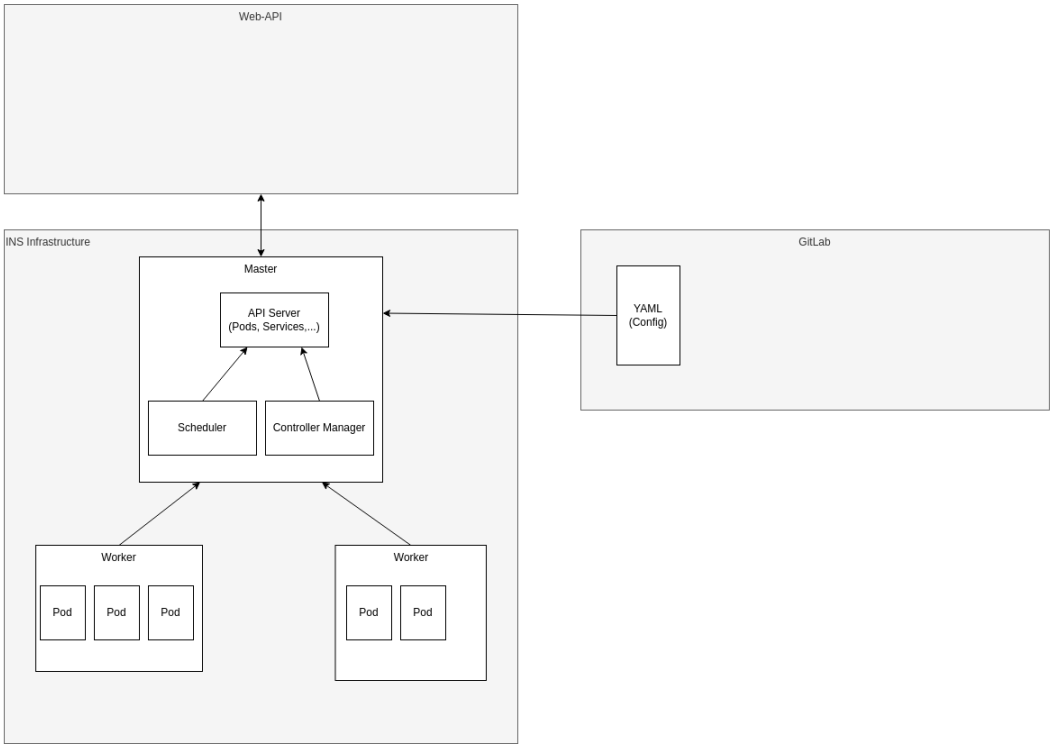
Describe the problem domain using a domain model as covered in the SEP1 module.

Chapter 4

Architecture

Describe the architecture of your software as covered in the SEP2 module. The main goal of this chapter is describing the **technical implementation** in a way that a new team member can start working on the product as fast as possible.

- Use an existing template as a starting point (`arc42`, `C4 model`, ...)
- Focus on stable, high-level concepts rather than details
- Cover different views (static, dynamic, deployment, ...)
- Prefer diagrams over text (ideally UML)
- Explain the reasons behind your actions: *Why did we build it like this?*



Master: Each Kubernetes cluster contains at least one master node which runs the Kubernetes control plane. These control plane consists of an API server, scheduler, controller manager, etc.

Worker/Node: Each cluster includes at least one worker node, this node runs containerized applications. Each worker node host *Pods*.

Pod: A pod is a components of the application workload.

Chapter 5

Quality Measures

Describe the quality measures applied in your project as covered in the SEP2 module. Things that might be included in this chapter:

- Organizational means like *Merge Requests*, *Definition of Done*, etc.
- Tools used to assess the quality of your product (linter, metrics, ...)
- Tools used to build and deploy your product (CI/CD)
- The *Test Concept* used for testing your product

Try to avoid duplication with other chapters such as the *Project Plan*. Work with cross-references when appropriate.

Part III

Project Documentation

Chapter 6

Initial Project Proposal

Project name: KubeWatch

Team Members

1. Petra Heeb (petra.heeb@ost.ch)
2. Benjamin Platter (benjamin.plattner@ost.ch)
3. Olivier Lischer (olivier.lischer@ost.ch)
4. Jan Untersander (jan.untersander@ost.ch)
5. Pascal Lehmann (pascal.lehmann@ost.ch)

Availabilities

Time slot	Mon	Tue	Wed	Thu	Fri
08h00-09h00	XR	-	-	-	-
09h00-10h00	XR	-	-	-	-
10h00-11h00	-	-	-	(XO)	-
11h00-12h00	-	-	-	(XO)	-
12h00-13h00	XR	XR	XR	XO	XO
13h00-14h00	-	-	-	-	-
14h00-15h00	-	-	-	-	-
15h00-16h00	-	-	-	(XO)	(XO)
16h00-17h00	-	-	-	(XO)	(XO)
17h00-18h00	-	-	-	XO	XO
18h00-19h00	-	-	-	XO	XO

Please mark ALL time slots during which every team member can attend review meetings.

XO	Slot available online
XR	Slot available in Rapperswil
(XO)	Slot available online, but not optimal
(XR)	Slot available in Rapperswil, but not optimal
-	Slot not available

Having a slot available on campus automatically implies that this slot is also available online. The more slots that are available, the more likely it is that you will get an optimal slot and coach assignment. You are required to have a minimum of 5 slots available during regular working hours (08h-12h, 13h-17h) to be eligible to form a team.

Project Idea

KubeWatch is a monitoring application for Kubernetes (K8S), intended for technical users. It keeps track of multiple K8S nodes, records performance data over time and generates visualizations from the aggregated data. It can detect when a node goes down, which triggers a notification to the person in charge. We chose this topic because it combines multiple technical aspects, so that each team member can get something out of it.
Ideas for extending the project scope:

- Notifications through multiple channels like Email, SMS, IM.
- Running an analysis on the aggregated data to detect anomalies like a DDOS attack.
- Visualizing the K8S pods and nodes in a graph.

Proposed Realisation

The K8S nodes are polled with the K8S API and the data ist stored in a database, since the K8S API doesn't deliver historical data. All interaction with the application happens through a web interface, which is backed by a Node.js server. We use Handlebars for templating and TypeScript for typesafe programming.

We discussed our proposal briefly with Thomas Kälin and he approved of the direction we're going.

Chapter 7

Project Plan

Describe the project plan as covered in the SEP2 module. A project plan typically consists of the following topics:

- Processes, meetings and roles
- Phases, iterations and milestones
- A **rough** list of things to be done (work items)
- Risk management
- Planning Tools (issue tracker, time tracker, ...)

You should **not** describe your **technical solution** in this chapter. It is all about organizing your project.

7.1 Processes

For the long-term planning we use RUP (7.4). For the short-term planning we use Scrum. The Scrum roles and other assignments are made in 7.2. The Scrum Events (Sprint Planning, Sprint Review, ...) are declared in 7.3. How we implement the Scrum artifacts is defined in 7.1.1.

7.1.1 Scrum

For the short term planning iteration we use the Scrum method. In the following we describe how we want to implement the Scrum process using Gitlab.

Scrum	**Gitlab**
User Story / Feature	Issues with Project Backlog label
Work Item	Issue linked to the User Story issue
Prioritization	Prioritized Labels
Scrum Board	Issue Board

To build a Scrum Board the following labels are required:

- ToDo
- Work in Progress / WIP
- Done

7.2 Roles

Role	Person
Session Chair:	changes every week
Product Owner (PO):	The whole team
Developer:	Benjamin Plattner, Olivier Lischer, Pascal Lehmann
Network:	Jan Untersander, Petra Heeb

The classification in *Developer* and *Network* shows only the primary strengths. Members of the *Developer* group sometimes also work on the network part and vice versa.

7.3 Meetings

We have two weekly meetings. A team internal meeting on each Monday. This meeting is used to implement:

- Sprint Planning
- Sprint Retrospective

The second meeting is on each Tuesday with the advisor Laurent Metzger. In this meeting the *Sprint Review* is done. The *Daily Scrum* take places during the week at lunch or during breaks. One Sprint last between two and four weeks.

7.4 Phases, iterations and milestones

In our project we work with the four project phases which are defined also in the RUD model which we used for our rough project plan. The four phases are:

- Inception
- Elaboration
- Construction
- Transition

7.4.1 Inception

The first phase is the *Inception* phase. In this phase we start the new project and use to define the following things to plan our project:

- Approximate vision
- Defining the scope
- Rough estimates for efforts

7.4.2 Elaboration

The second phase is called *Elaboration*. This phase is used to start the practical part of the project and the goal of this phase is to eliminate potential risks. There are some parts you need to handle in this phase:

- Identification of most requirements
- Iterative implementation of core architecture
- Resolution of high risks
- More realistic estimates for efforts

7.4.3 Construction

The third and biggest phase is the *Construction* phase. In this phase the team need to make the project around the risk parts of the project. In this phase the risk parts should be already solved. The contents of this phase are:

- Iterative implementation of functionality
- Resolution of lower risks
- Preparation for deployment

7.4.4 Transition

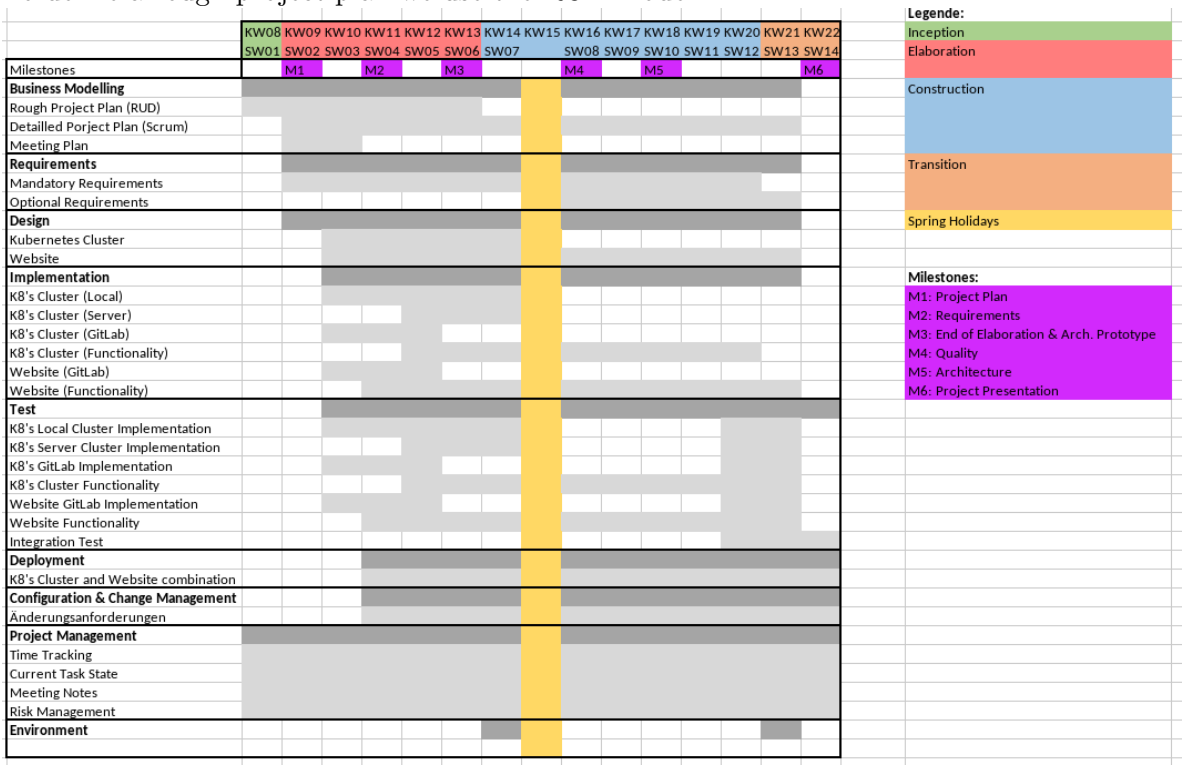
The last project phase is to finish the project and to test the project with the whole environment and called *Transition* phase. The basic points of this finish phase are:

- Beta Tests
- Deployment
- Tie up any loose ends

7.5 Project Plan

7.5.1 RUD - Rational Unified Process

To define a rough project plan we use the RUD model.



7.6 Risk management

7.7 Planning Tools

For planning our work, for issue handling and time tracking we use the GitLab tool which is hosted by the OST themself.

Chapter 8

Time Tracking Report

The main goal of this chapter is to show your stakeholders that your project is on track, i.e. you will be able to finish it within the remaining time. The chapter should cover two topics:

1. Describe **how you track time** in your project. Try to avoid duplication with the *Project Plan*.
2. The **current time tracking statistics** for your project.

For the latter one, keep the information on a high-level:

- How much time did we invest in total yet? How much is remaining?
- How much time did we invest in iteration 1, 2, 3, ...?
- How much time did we invest in which topic?

Diagrams work very well to communicate these information. Low-level information (e.g. *how much time did we spend on task XYZ?*) is best kept within the chosen time tracking tool.

Chapter 9

Personal Reports

Before the final submission, **personally reflect** your work in this project:

- What things did go well?
- Which areas could we improve?
- What were your personal highlights?

The information gathered in this chapter will be **very** useful for all your future projects.

Chapter 10

Meeting Minutes

Add your meeting minutes here. As usual, try to keep them short and concise.

Chapter 11

Guidelines

For our project we need to define some guidelines.

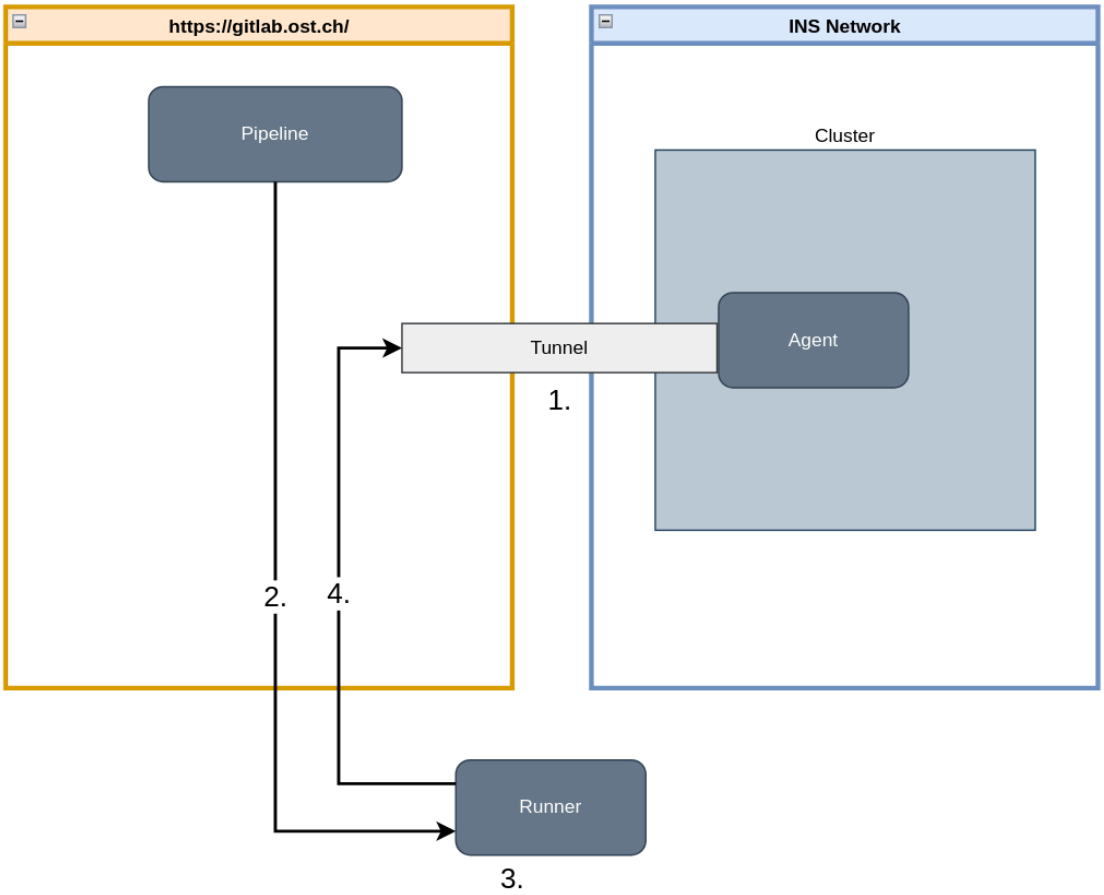
11.1 Kubernetes

Definition	Guideline
file names	snake_case
placeholders	angle brackets
document interface elements	bold
document filenames or directories	code style

11.2 Web-API

Chapter 12

Interface K8s Cluster and GitLab-OST



1. The *Agent* create a *Tunnel* from the INS network to the gitlab.ost.ch network.
2. The *Pipeline* trigger the *Runner* to run the cluster commands.

3. The *Runner* start the docker.

4. The *Runner* get via the gitlab.ost.ch network access to the *tunnel endpoint* and run the cluster commands now at the cluster.

The figure above shows you how the Kubernetes cluster and the GitLab are combined. The GitLab and the cluster are in two different networks, the GitLab is placed in the OST network and the Kubernetes cluster is deployed in the INS network.

Only the *Agent* inside the INS network can create the *Tunnel* to the gitlab.ost.ch network. Nobody from the gitlab.ost.ch network can deploy the *Tunnel* by themself.

The *Runner* is independent of the location, so it doesn't depend on the network you place the *Runner*.

Bibliography

[Lar04] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.