Process Guide

Group 12

by

Samuel, Trine, Ali, Matin

IKT206
DevOps

Faculty of Engineering and Science
University of Agder

Grimstad, may, 2022

Process guide

for

Auby & Brinch Finance

A step by step process guide

Contents

1 Introduction **1**

2 Building a Team **2**
2.1 Structure 2
2.1.1 Team Lead 2
2.1.2 Architect 2
2.1.3 Developer 2
2.1.4 UI/UX Designer 2
2.1.5 Computer network engineer/DevOps 2
2.1.6 Tester/QA 3
2.1.7 Scrum master 3
2.2 5-person team build 3
2.3 10-person team build 3

3 Tools **5**
3.1 The Scrum methodology 5
3.1.1 Sprint 5
3.1.2 Product Backlog 5
3.1.3 Sprint Backlog 5
3.1.4 Stand-up meetings 5
3.1.5 Weekly procedure Scrum 6
3.2 Gitlab 7
3.2.1 CI/CD Pipeline 7
3.2.2 Adding team-members 8
3.2.3 Gitlab as Scrum software 12
3.2.4 Gitlab as versioncontrol 14

4 Resources **18**

List of Figures

1 Illustration Scrum workflow 6
2 Illustration Gitlab Pipeline 7
3 Adding user to the project 8
4 “menu”, and then “admin” 9
5 “Users” under “Overview” 9
6 “New user” button 10
7 Giving users access to the GitLab instance 10
8 The SSH key 11
9 Paste SSH key 11
10 Create milestone 12
11 Crate issue 13
12 Backlog 13
13 Boards 14
14 Milestones 14
15 Merge request 15
16 Automated tests 15
17 Submitted review 16
18 Approve merge 17
19 Merged 17

1 Introduction

This document is meant as a guide for optimizing the day-to-day workflow in a development team at Auby & Brinch Finance. This includes a new team build with new roles, a new set of technical tools to aid in the workprocess as well as an introduction to the scrum framework for managing the workflow. We will start off with team structure and roles, and progress to Gitlab as a workflow tool and as version-control. Basic knowledge on how to use Git Bash is assumed and will not be covered in detail.

2 Building a Team

2.1 Structure

Team structure, skill and communication are some of the most important factors to consider when building a high-performing software development team[1]. In this section we will look at how to best ensure your team is effective and result-driven.
First, lets break it down to team roles and take a look at different areas of responsibility and what skills they should possess.

2.1.1 Team Lead

The team lead is in charge of the overall performance, motivation, and organization of their team. A developer often assumes the role of team lead. Directs the designers/architects to work towards a common goal. They make sure team members work well together and are on track to deliver their software solutions on time. They're sometimes referred to as engineer managers and are also responsible for the development and learning of their team members. A team lead should have strong leadership and communication skills.

2.1.2 Architect

A software architect is responsible for describing the complete architecture system of a project. Architects need to have a clear vision of how the product should work. They define the technical and functional architecture of the entire system and writes specifications and requirements. They are responsible for the "big picture".

2.1.3 Developer

Developers are in charge or writing code and developing the project based on specifications. They can also write initial automatic tests to verify functionality. A developer should have a strong eye for detail and have solid knowledge of the needed programming languages as well as version control.

2.1.4 UI/UX Designer

UX and UI designers are responsible for the user interface and experience. They should be creative and have a strong design focus. They typically need to be analytical and think outside of the box.

2.1.5 Computer network engineer/DevOps

Responsible for operating and maintaining the server and backup server. Also has deep knowledge of the software development process and lifecycle. Needs to have experience with DevOps and docker. Will work on deployment of the code, monitoring and handling infrastructure and continuous integration. Takes the work done by developers and creates a new software version. Together with QA ensures the quality of the process.

[1] `https://projectcor.com/blog/fundamental-roles-on-a-software-development-team/`

2.1.6 Tester/QA

Testers form and execute test cases to detect bugs or deficiencies. Tests are based on the requirements set by the architect. They will detect problems before the whole project is staged for release and provide feedback about the results to take corrective actions. Testers will go through results from automatic tests, but also do manual testing. Testers should be curious and organized as they test different codes when creating the software product. They should have experience using Docker and Gitlab. Quality Assurance(QA) also involves preparing tools that allow for automating processes which verify software quality and maintaining the CI/CD pipeline. Their objective is to ensure quality at the process level in projects. Generally overseas what goes into the next release.

2.1.7 Scrum master

The scrum master is responsible for how the framework is followed in the team. Their role involves removing obstacles, creating a productive environment and helping the team work well together. Although technically a leader, they hold no authority over other team members. A Scrum master should have strong leadership, coaching and organizational skills[2].

2.2 5-person team build

For smaller teams the team-members often carry more then one role. As the team gets 2 more members, a few more roles will be implemented. We are not adding all at once, as this will give the team some time to adjust. The first new hire should be someone with a network operations backround and DevOps knowledge. The second one should be someone experienced with writing testing and QA as well as being familiar with DevOps. This is important to start integrating DevOps and start writing tests.
Suggested roles for a development team of 5:

- Role 1: Team Lead/Scrum master
- Role 2: DevOps/Network operations
- Role 3: Tester/QA
- Role 4 and 5: Developer

2.3 10-person team build

As the team expands, the roles can be broken down a bit more and we can add a couple of new ones. Assuming its appropriate in relation to the project to only have one development team of 10 instead of two 5-person development teams. Having more than 10 team-members is not recommended as it gets harder to manage. For 5 new hires we can now add an architect, a UI/UX designer, another tester/QA, a scrum master that is also a developer and one more developer. Here are suggested roles for a development team of 10:

[2] `https://www.hexacta.com/infographic-software-development-project-roles-and-responsibilities/`

- Role 1: Team Lead
- Role 2: Scrum Master/Developer
- Role 3: Architect
- Role 4: UI/UX Designer
- Role 5 and 6: Tester/QA
- Role 7: DevOps/Network operations
- Role 8, 9, 10: Developer

3 Tools

The purpose of the following tools are primarily to make information available and streamline the workflow as much as possible. In order for the team to work as effectively as possible you need to make sure that every team-member has access to the same updated information. Making data available means greater transparency in the progress of the project. This allows team members to see where progress is being made and where improvements may be necessary which in turn will help ensure an adaptable and agile team behind a high-standard product. By automating processes, like quality-checking and version-control, you can save a lot time and advance quicker in your project[3].

3.1 The Scrum methodology

The Scrum methodology is a work strategy that manages a project by breaking it down into several stages. It is primarily designed for teams of 10 or less members. The team will cycle through a planning, executing and evaluation process. A scrum cycle is called a sprint.

3.1.1 Sprint

A sprint is a longer interval of time, usually between 1-4 weeks, depending on the goal of the team, where the team works toward a delivery/iteration goal. The sprint-duration is set and should be the same throughout the project. A sprint meeting is held at the beginning of each sprint where the team decides what tasks should be included. The Scrum-master leads these meetings. At the end of a sprint, the team comes together to discuss what was completed, what worked, and how you can improve moving forward.

3.1.2 Product Backlog

The product backlog contains features that might be implemented later in development. When starting new sprints, issues from the backlog can be moved to the sprint backlog. These are often user-stories, tasks or bugs that need fixing.

3.1.3 Sprint Backlog

The sprint backlog contains the tasks the team has committed to for the following sprint. This can be tasks from the backlog or new ones. The titles of a task should be as descriptive and precise as possible. Tasks can be assigned to a specific one or more team-members. Instead of estimating how many hours a task will take, we estimate workload depending on the issue size. We can then assign story-points to that issue which is a value based on workload. Over time estimates will be more accurate.

3.1.4 Stand-up meetings

These are short daily meetings, no more than 15 minutes. The meeting time should be determined at the first sprint meeting and should be the same time every day. Here team members can update each other on what they're working on, progress, or if they need assistance/can be of assistance. This helps the team to stay informed and focused on hitting the sprint goals. If a team-member is

[3]`https://projectcor.com/blog/fundamental-roles-on-a-software-development-team/`

stuck on a task of any kind, the Scrum-master is responsible for delegating the needed resources for solving the issue. This way the project-progress will not halt for longer periods of time. Below is a 30-day sprint illustration[4].

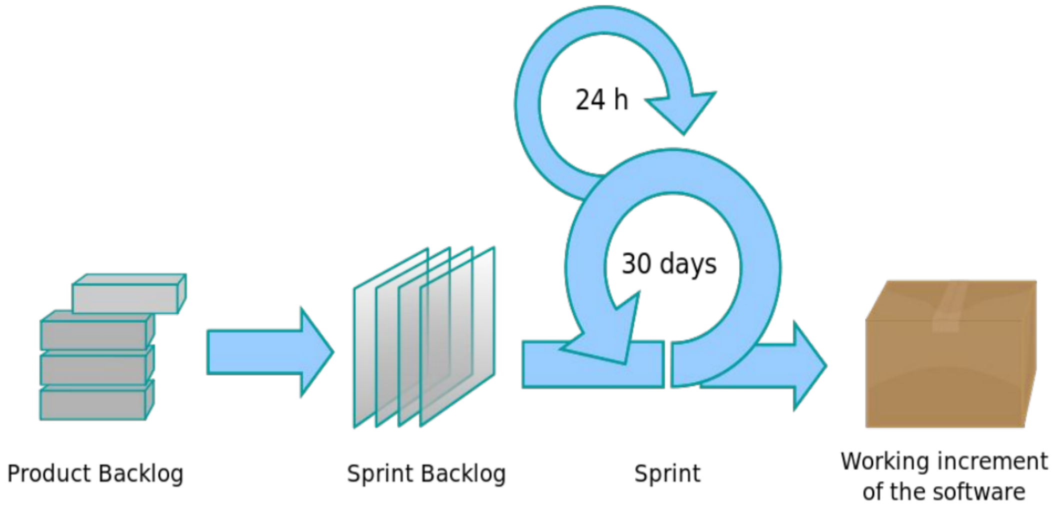


Figure 1: Illustration Scrum workflow

3.1.5 Weekly procedure Scrum

We recommend starting off with shorter sprints as its easier for getting used to a new methodology. Described below is a suggested procedure using Scrum for a one-week sprint:

Sprint meeting on Monday 08:00

- Step 1: Retrospective
  Discuss what worked and what could be better from the last sprint. Note suggested improvements to be implemented.

- Step 2: Review
  Review the finished work/remaining work that still needs to be done. Unfinished work must be transferred to the next sprint. Check project progress. Close sprint.

- Step 3: Update the backlog
  Add new issues to the product backlog if needed. Keep the backlog sorted with the most important issues at top. Estimate issues with story-points.

- Step 4: Create new sprint
  Add new issues to this sprints backlog from the product backlog, including remaining issues from the previous sprint if any.

[4]https://en.wikipedia.org/wiki/Scrum_(software_development)/

Daily stand-up meeting Tuesday-Friday 08:00

- Team members briefly update each other on progress and daily goal. Keep it short, 10-15minutes. Scrum-master is responsible for any needed delegation.

3.2 Gitlab

Gitlab is an all-in-one DevOps software used for developing, securing, and operating software. It is an open source code repository and collaborative software development platform for large DevOps projects[5]. This includes features well suited for managing the Scrum workflow and for version control. Version control allows software teams to track changes in code, while enhancing communication and collaboration between team members and creating a centralized location for code[6]. Version control is important for reproductability, to make it possible to build or rebuild any version of the software. Next we will look at the continuous integration/continuous deployment pipeline provided by Gitlab.

3.2.1 CI/CD Pipeline

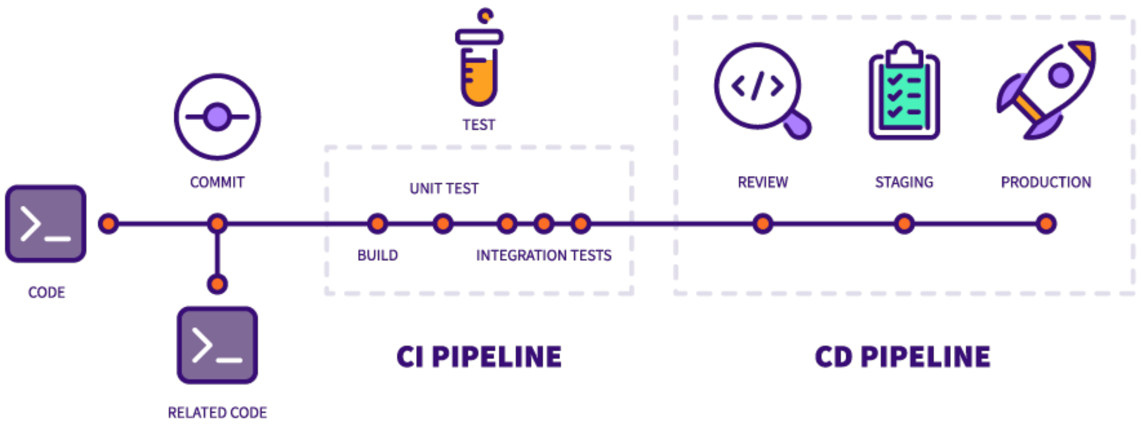


Figure 2: Illustration Gitlab Pipeline

The CI/CD pipeline (as shown in Figure 2[7]) can be seen as a series of steps. We start at the beginning, to the left, with the familiar process of writing source code, commit our changes, and git push our source code. The changes will trigger the CI pipeline to build and run unit and/or integration tests automatically. It can then proceed to the CD pipeline for a manual review. After reviewing we enter the staging step where it can be sent to a server for further testing with a QA team, and in the end be put in production to run live on our final server. The following chapters will describe this process step-by-step in detail.

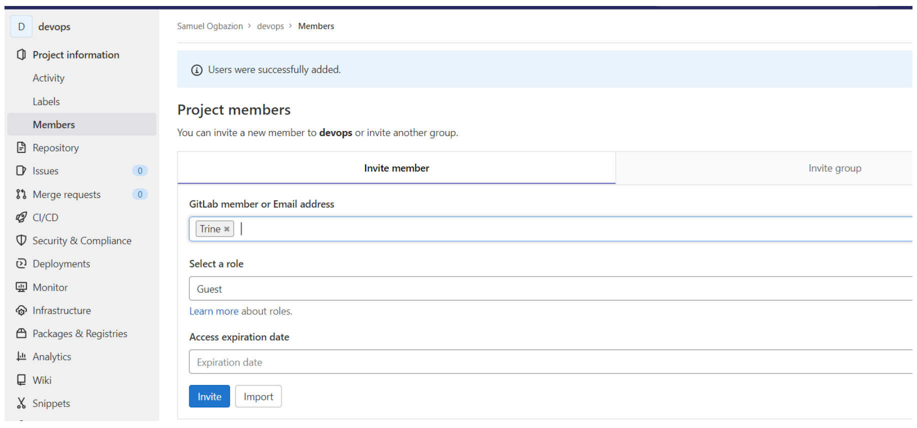
[5] `https://about.gitlab.com/stages-devops-lifecycle/`
[6] `https://about.gitlab.com/topics/version-control/`
[7] `https://about.gitlab.com/handbook/marketing/strategic-marketing/competitive/cicd/`

3.2.2 Adding team-members

Giving access to team members: To be able to give the team members access to the project you have to first create the project in GitLab. This is usually done by the DevOps engineer. One of the last steps in the setup of the project in GitLab is to both add the new members to the project, and to give them access to the server.

To do this you have to do the following:

1. Once you're inside the project in GitLab, click on "Project information"
2. Then click on "members"
3. Under "Invite member", fill out the information needed for the new member and click "Invite"


Samuel Ogbazion › devops › Members

Users were successfully added.

Project members

You can invite a new member to **devops** or invite another group.

GitLab member or Email address: Trine

Select a role: Guest

Learn more about roles.

Access expiration date: Expiration date

Figure 3: Adding user to the project

The invited members are now invited to the project in GitLab.

Then you have to give the invited members their own user accounts to be able to login to the same server:

1. Click on “Menu” next to the GitLab logo in the upper left corner and then click on “Admin”

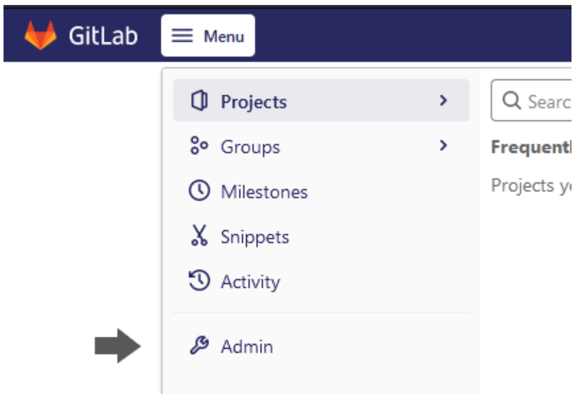


Figure 4: “menu”, and then “admin”

2. Under “Overview” on the left side, click on “Users”

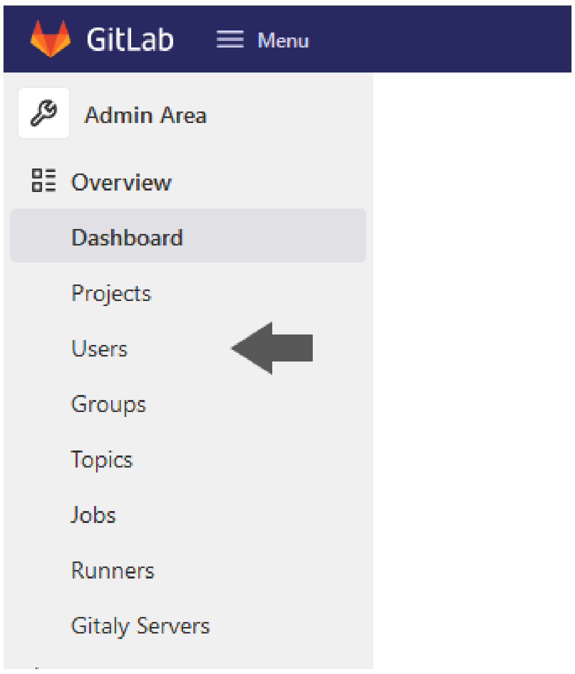


Figure 5: “Users” under “Overview”

3. Then click on blue “New user” button on the right side

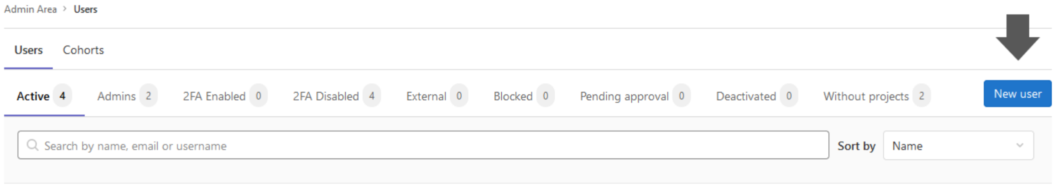
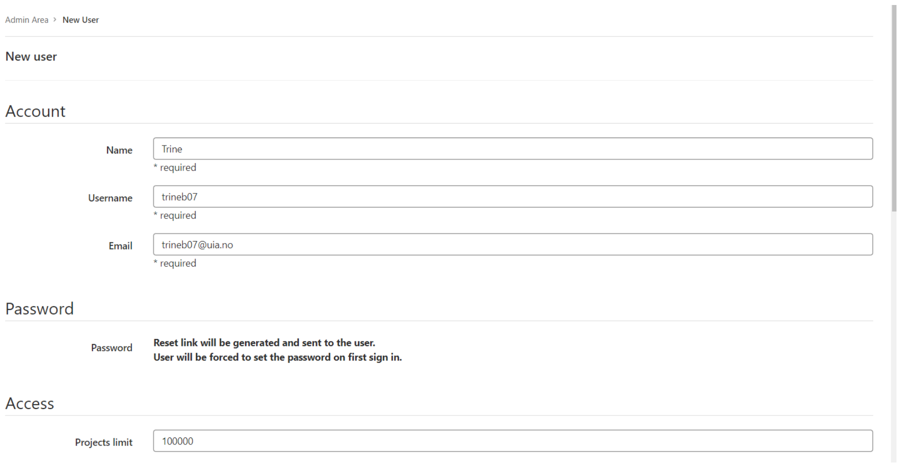


Figure 6: "New user" button

4. Fill out the required fields and click "Create user" on the bottom


Admin Area › New User

New user

Account

Name: Trine
* required

Username: trineb07
* required

Email: trineb07@uia.no
* required

Password

Password: Reset link will be generated and sent to the user.
User will be forced to set the password on first sign in.

Access

Projects limit: 100000

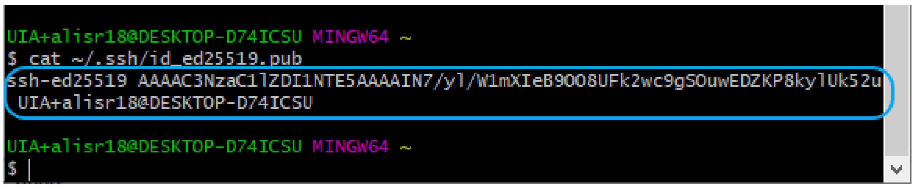
Figure 7: Giving users access to the GitLab instance

You have now created a user that have access to both the server, and the project. NOTE: Its important to give the right kind of roles to each employee for them to be able to use the functionalities they need.

Connect your local machine to GitLab with SSH To be able to pull and push codes and changes, in a secure way, to GitLab, you need to set up a connection between your local machine and GitLab. Assuming you already have installed Git Bash on your computer, do the following:

1. **Create a SSH key in Git Bash** - start up "Git Bash" and enter the following commando: "`ssh-keygen -t ed25519`". NOTE: Just press enter until it's done.

2. **Copy the SSH key you generated** - enter the following commando to show the generated key: "`cat~/.ssh/id_ed25519.pub`".
Copy the whole output as shown in the picture below.

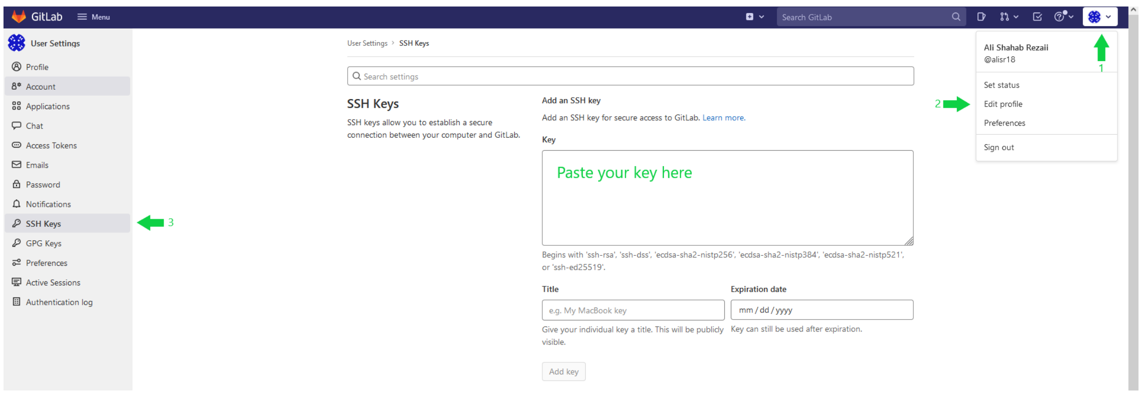


```
UIA+alisr18@DESKTOP-D74ICSU MINGW64 ~
$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIN7/yl/W1mXIeB9OO8UFk2wc9gSOuwEDZKP8kylUk52u
 UIA+alisr18@DESKTOP-D74ICSU

UIA+alisr18@DESKTOP-D74ICSU MINGW64 ~
$ 
```

Figure 8: The SSH key

3. **Paste the SSH key in GitLab -** We need to paste the copied SSH key in GitLab in order to get access from our local machine. Open GitLab, login and click on “Edit profile” under your avatar in the upper right corner. Locate “SSH key” on the left side and click it. Paste your code here, enter a title for your key and an expiration date, and click “Add key” (see figure 19). You have now access to GitLab from your local machine.



SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

Add an SSH key for secure access to GitLab. Learn more.

Key

Paste your key here

Begins with 'ssh-rsa', 'ssh-dss', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', or 'ssh-ed25519'.

Title: e.g. My MacBook key

Give your individual key a title. This will be publicly visible.

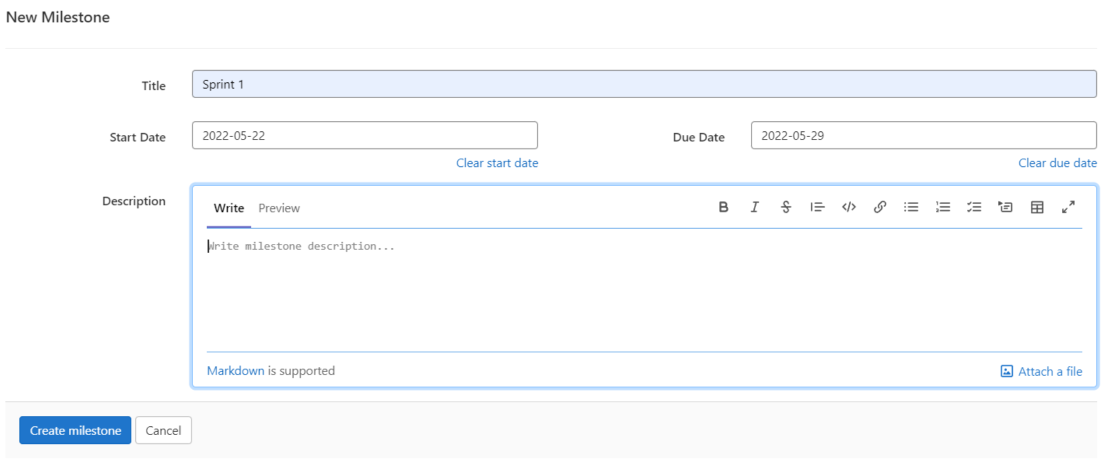
Expiration date: mm / dd / yyyy

Key can still be used after expiration.

Figure 9: Paste SSH key

3.2.3 Gitlab as Scrum software

The first thing to do when starting a new sprint is to create a new milestone. Gitlab milestones are the equivalent to sprints. You can assign a start-date and due data to the milestone(i.e 1 week), and you can then put the desired issues into that upcoming sprint.

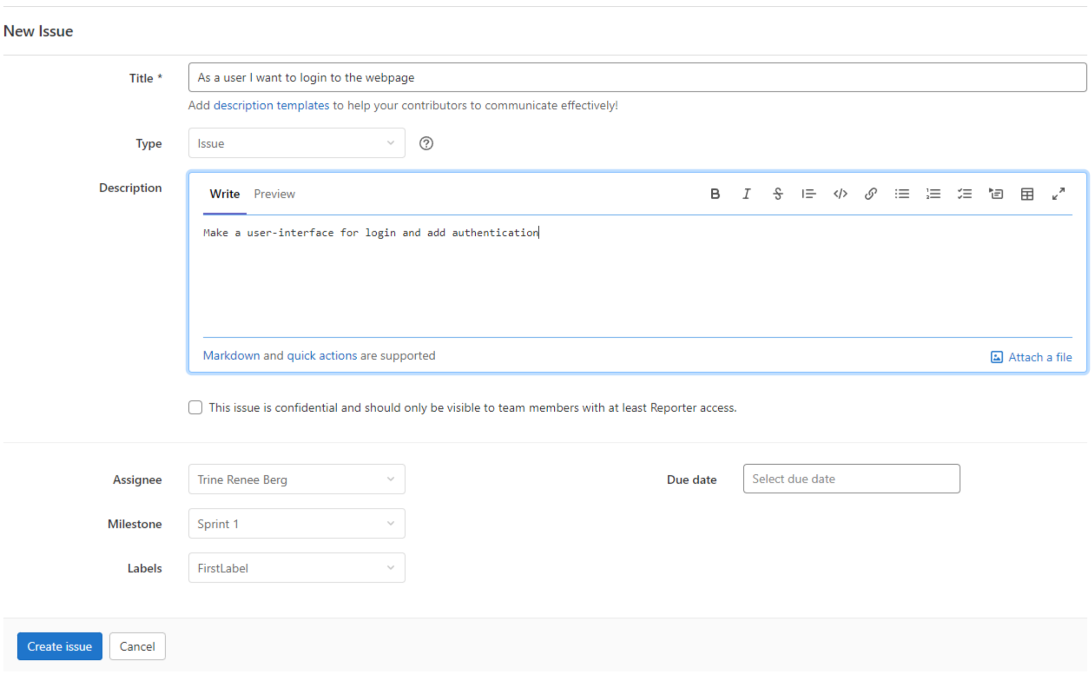


New Milestone

Title	Sprint 1
Start Date	2022-05-22
Due Date	2022-05-29
Description	Write milestone description...

Figure 10: Create milestone

Next you can add issues. You can simply click `Issues` –> `New issue` to add a new issue.


New Issue

Title * As a user I want to login to the webpage

Add description templates to help your contributors to communicate effectively!

Type: Issue

Description:

Make a user-interface for login and add authentication

Markdown and quick actions are supported

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee: Trine Renee Berg

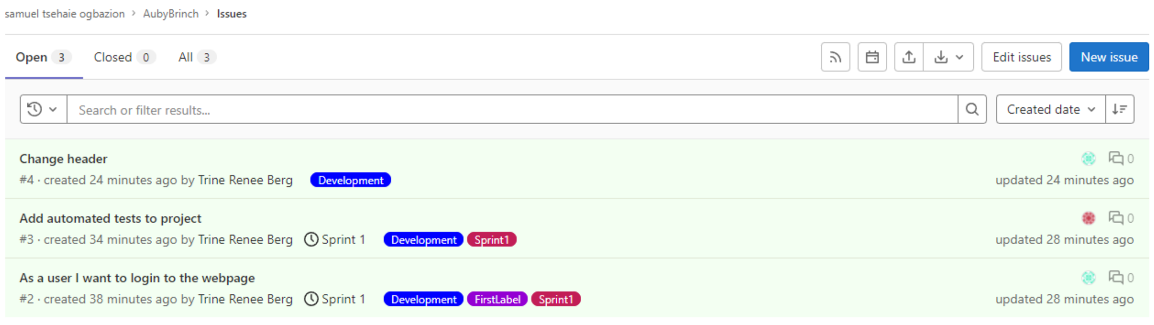
Milestone: Sprint 1

Labels: FirstLabel

Due date: Select due date

Figure 11: Crate issue

You can assign the issue to the created milestone. The issue will then appear in that milestones(sprint) backlog as shown below. The issue can be assigned to a team-member or be left unassigned. Labels can be created and assigned to individual issues. You can then filter the issue list by one or more labels for more flexibility. All issues will be visible under `List`. You can view the issue list to track the backlog. To be able to assign story-points you need to use labels or add story-points in the description as the Gitlab-version of story-points(weight) is only available in Enterprise Gitlab.


samuel tsehaie ogbazion › AubyBrinch › Issues

Open 3 Closed 0 All 3

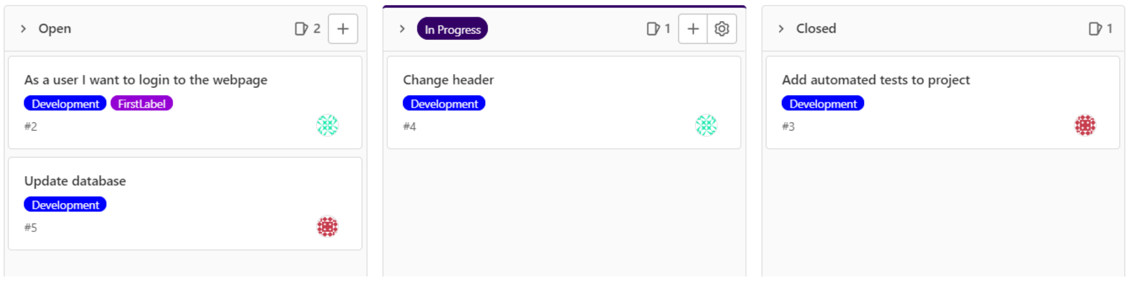
Change header
#4 · created 24 minutes ago by Trine Renee Berg Development — updated 24 minutes ago

Add automated tests to project
#3 · created 34 minutes ago by Trine Renee Berg Sprint 1 Development Sprint1 — updated 28 minutes ago

As a user I want to login to the webpage
#2 · created 38 minutes ago by Trine Renee Berg Sprint 1 Development FirstLabel Sprint1 — updated 28 minutes ago

Figure 12: Backlog

By clicking **`Boards`** you'll get the main view to manage progress during the sprint. Here you can see all ongoing issues as well as completed issues. When an issue is completed, simply drag it over to the completed-list. To add a “In progress” -list we can click `create list` and name it “In Progress” . Issues can be dragged over to this list as members start working on them. This will automatically add the “In progress”-label to that issue.

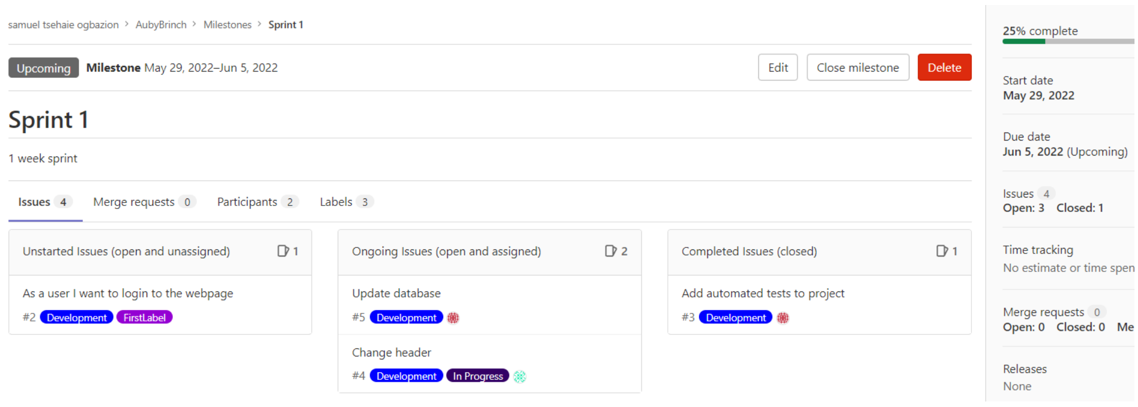

Open 2
As a user I want to login to the webpage
Development FirstLabel
#2
Update database
Development
#5

In Progress 1
Change header
Development
#4

Closed 1
Add automated tests to project
Development
#3

Figure 13: Boards

To track progress, you can click `Milestones` and select the current sprint. This will show the ongoing issues(sprint backlog) and completed issues as well as a completion-percentage-bar to the right.


samuel tsehaie ogbazion › AubyBrinch › Milestones › Sprint 1

Upcoming **Milestone** May 29, 2022–Jun 5, 2022 Edit Close milestone Delete

Sprint 1

1 week sprint

Issues 4 Merge requests 0 Participants 2 Labels 3

Unstarted Issues (open and unassigned) 1
As a user I want to login to the webpage
#2 Development FirstLabel

Ongoing Issues (open and assigned) 2
Update database
#5 Development
Change header
#4 Development In Progress

Completed Issues (closed) 1
Add automated tests to project
#3 Development

25% complete

Start date
May 29, 2022

Due date
Jun 5, 2022 (Upcoming)

Issues 4
Open: 3 Closed: 1

Time tracking
No estimate or time spen

Merge requests 0
Open: 0 Closed: 0 Me

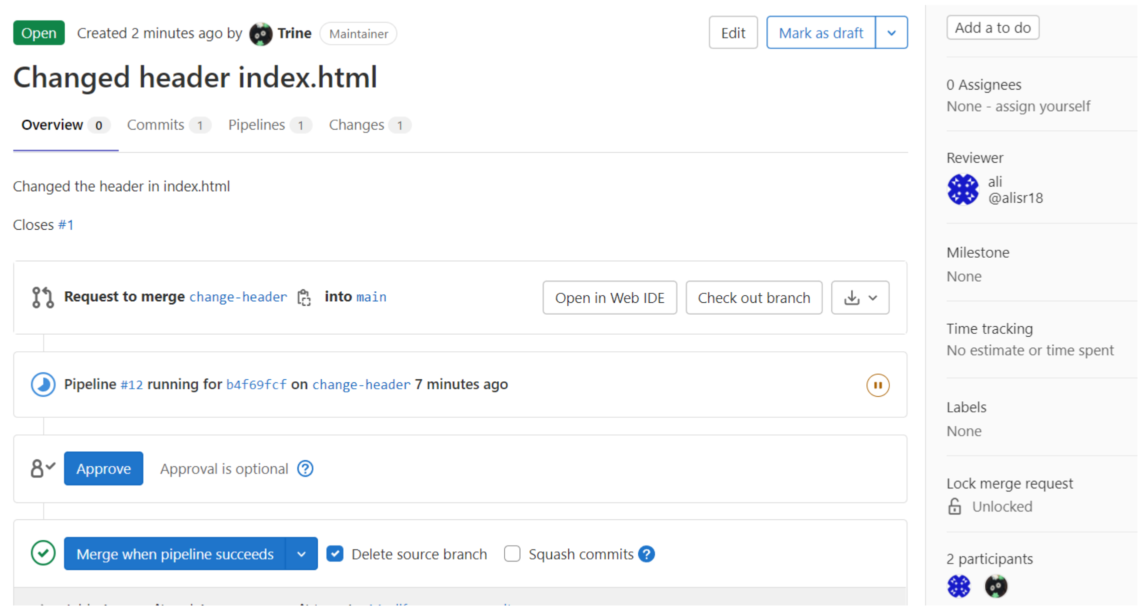
Releases
None

Figure 14: Milestones

3.2.4 Gitlab as versioncontrol

Step 1: Clone project
Clone project using the SSH clone URL found in the project.
Use the git command `git clone` and paste the SSH clone URL.
Step 2: Use Branches
When working on a task, always work from a branch. Use the command git command `git checkout -b branch-name` Now any changes to the code will only be in your branch. You can add the changes with `git add .` and commit the changes with `git commit -m "commit message here"` and push with `git push origin branch-name`

Step 3: Trigger Pipeline
Now we can see the newest commit in the project on gitlab. And we can see the pipeline running.



Changed header index.html

Changed the header in index.html

Closes #1

Request to merge change-header into main

Pipeline #12 running for b4f69fcf on change-header 7 minutes ago

Approval is optional

Delete source branch
Squash commits

0 Assignees
None - assign yourself

Reviewer
ali
@alisr18

Milestone
None

Time tracking
No estimate or time spent

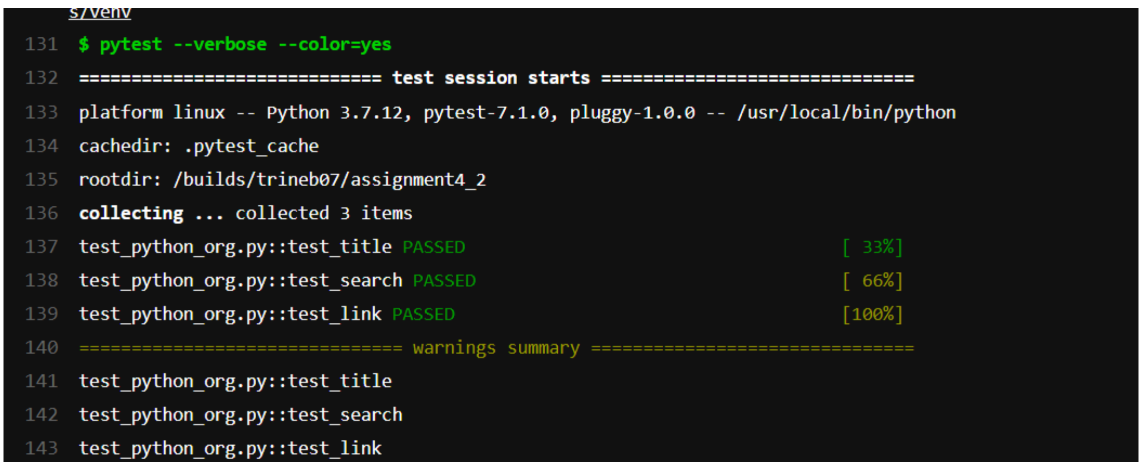
Labels
None

Lock merge request
Unlocked

2 participants

Figure 15: Merge request

Any automated tests will run here as shown below. When the pipeline is complete without issues we will get a "passed".

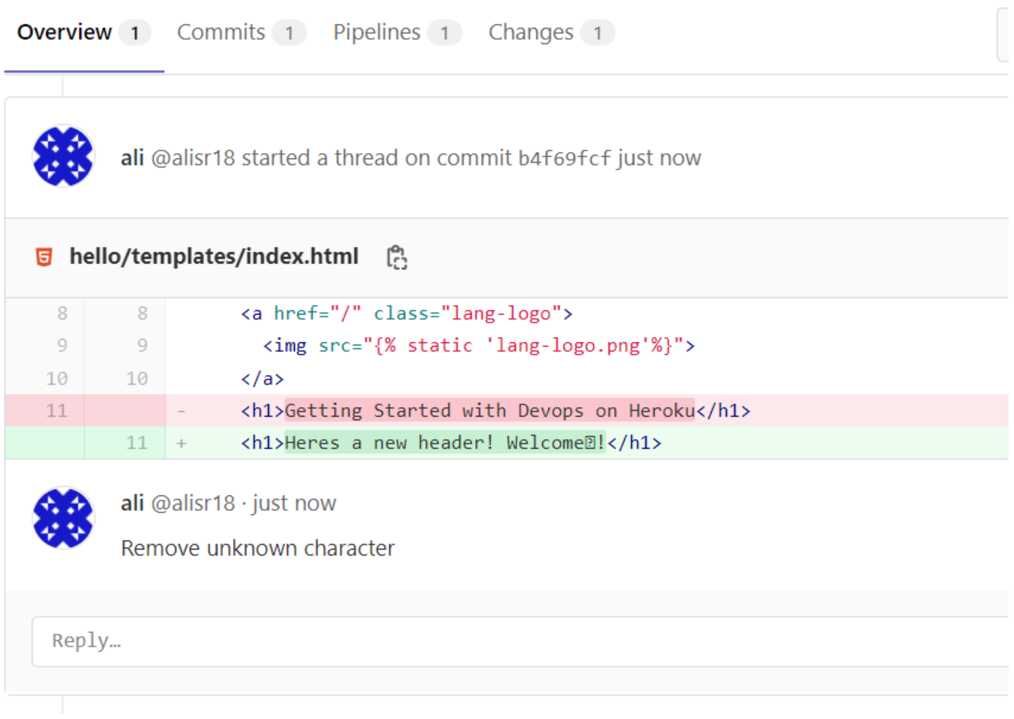


```
$ pytest --verbose --color=yes
============================= test session starts ==============================
platform linux -- Python 3.7.12, pytest-7.1.0, pluggy-1.0.0 -- /usr/local/bin/python
cachedir: .pytest_cache
rootdir: /builds/trineb07/assignment4_2
collecting ... collected 3 items
test_python_org.py::test_title PASSED                                    [ 33%]
test_python_org.py::test_search PASSED                                   [ 66%]
test_python_org.py::test_link PASSED                                     [100%]
=============================== warnings summary ===============================
test_python_org.py::test_title
test_python_org.py::test_search
test_python_org.py::test_link
```

Figure 16: Automated tests

Step 4: Code Review
Now you can create a merge request and add a reviewer from the team. To create a merge request

you can simply copy the URL we got when you pushed the code. Now the reviewer can look through the changes made to the code and add a message if its needed. For only one comment you can click `Add comment now` or `Start review` if there's more to comment. The comment will be displayed in the overview as shown below.



ali @alisr18 started a thread on commit b4f69fcf just now

hello/templates/index.html

```
      <a href="/" class="lang-logo">
        <img src="{% static 'lang-logo.png'%}">
      </a>
-     <h1>Getting Started with Devops on Heroku</h1>
+     <h1>Heres a new header! Welcome�!</h1>
```

ali @alisr18 · just now

Remove unknown character

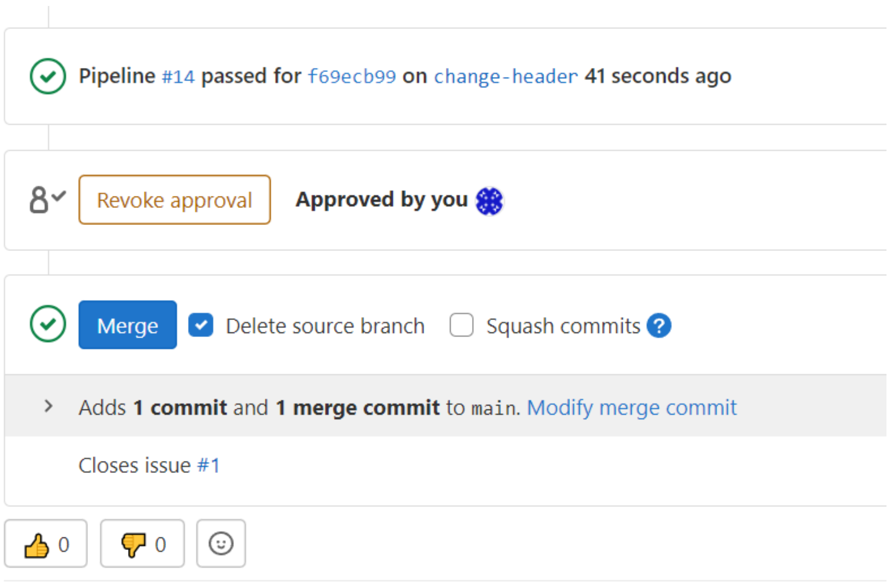
Figure 17: Submitted review

Step 4: Make changes to pushed code

Based on the received review you can now correct the code. The changes can be made locally through the preferred framework or by opening the file through git and make changes directly. When the changes have been made, the code must be committed again. This time its important to use the command `git commit --amend` in order to fix a previous commit and not create a new one. This keeps the history clean. When pushing you use `git push origin branch-name --force`. This is done because we work from a branch. It must never be done directly to main.

Step 4: Approve merge request

The reviewer can now approve the change that had been made so that the code can be merged.


Pipeline #14 passed for f69ecb99 on change-header 41 seconds ago

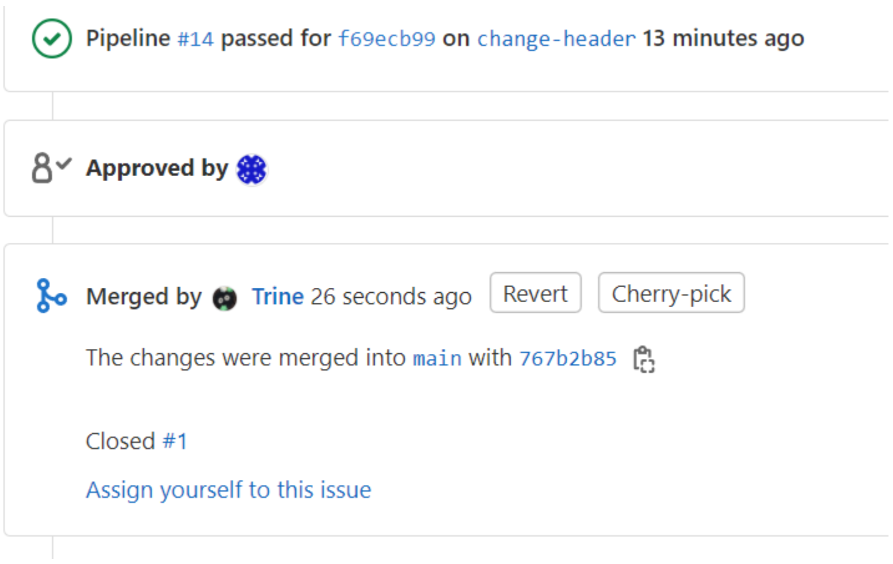
Approved by you

Adds **1 commit** and **1 merge commit** to `main`. Modify merge commit

Closes issue #1

Figure 18: Approve merge

When the code is approved by the assigned reviewers you can simply click the `Merge` button and the merge will begin.


Pipeline #14 passed for f69ecb99 on change-header 13 minutes ago

Approved by

Merged by Trine 26 seconds ago

The changes were merged into `main` with `767b2b85`

Closed #1

Assign yourself to this issue

Figure 19: Merged

4 Resources

Scrum: `https://www.scrum.org/resources/what-is-scrum`

The Scrum-master role: `https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-master`

Team Roles:
`https://www.hexacta.com/infographic-software-development-project-roles-and-responsibilities/`

Documentation Gitlab: `https://docs.gitlab.com/`

Using Gitlab with scrum:
`https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/`

Onboarding:
`https://www.nuclino.com/articles/employee-onboarding-process#onboarding-process-steps`