Introduction to Gitlab

What is GitLab

GitLab is a complete DevOps platform, delivered as a single application. This makes GitLab unique and creates a streamlined software workflow, unlocking your organization from the constraints of a pieced together toolchain. Learn how GitLab offers unmatched visibility and higher levels of efficiency in a single application across the DevOps lifecycle.
GitLab started as an open source project to help teams collaborate on software development. GitLab's mission is to provide a place where everyone can contribute. Each team member uses our product internally and directly impacts the company roadmap. This exceptional approach works because we're a team of passionate people who want to see each other, the company, and the broader GitLab community succeed and we have the platform to make that possible. From about gitlab.

Gitlab project

A GitLab project has at its center a git repository and a few services that evolve around said repository. Such as an issue tracker, contributions from other developers via **`merge requests`** and to do lists among others.

Create a new project

In the following steps a new personal public repository is created. The process of creating a new project is assisted and straight forward.
The creation starts with clicking on the boxed plus sign left to the search field.

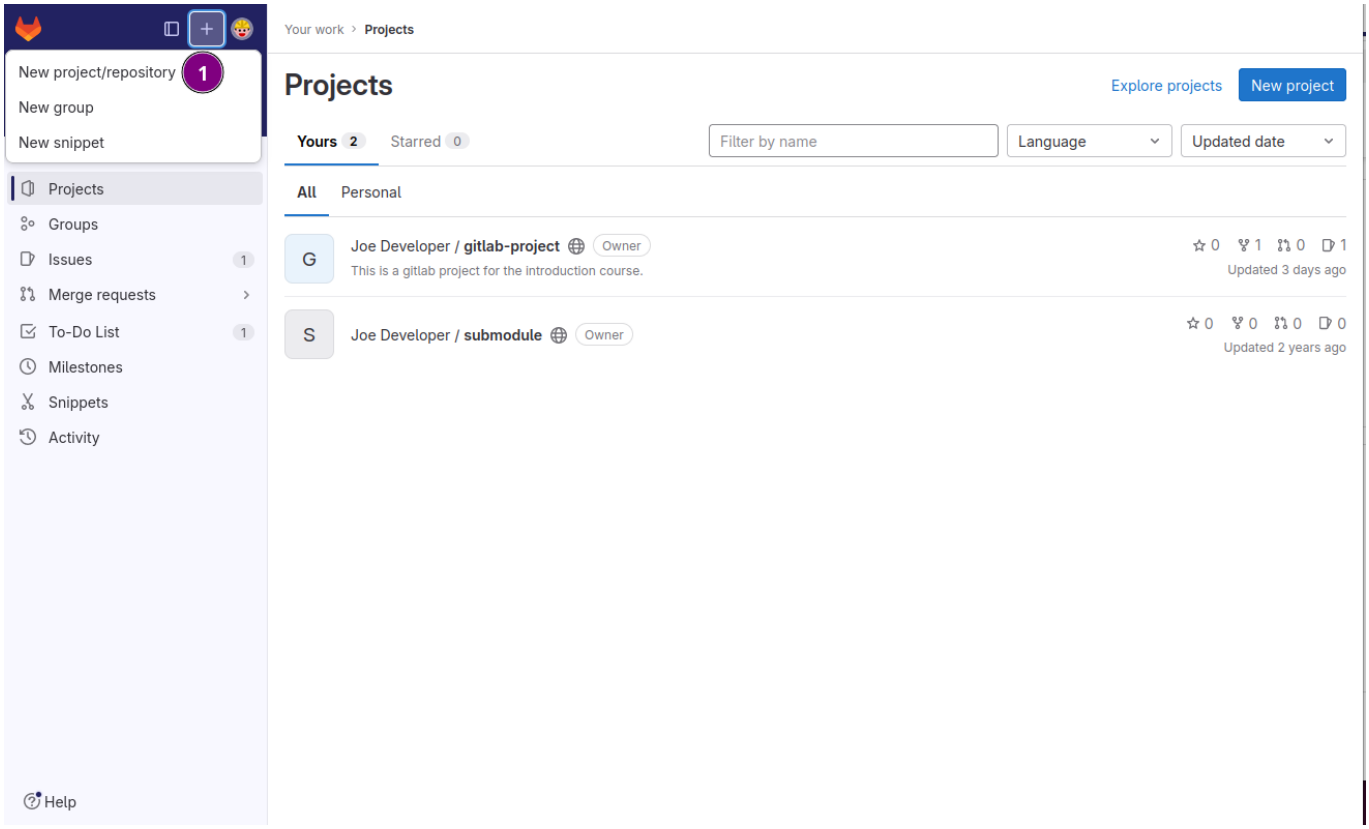

Figure 1. Create a new project from the top navigation bar

1 Choose **`New project/repository`** to open the initial starting point.

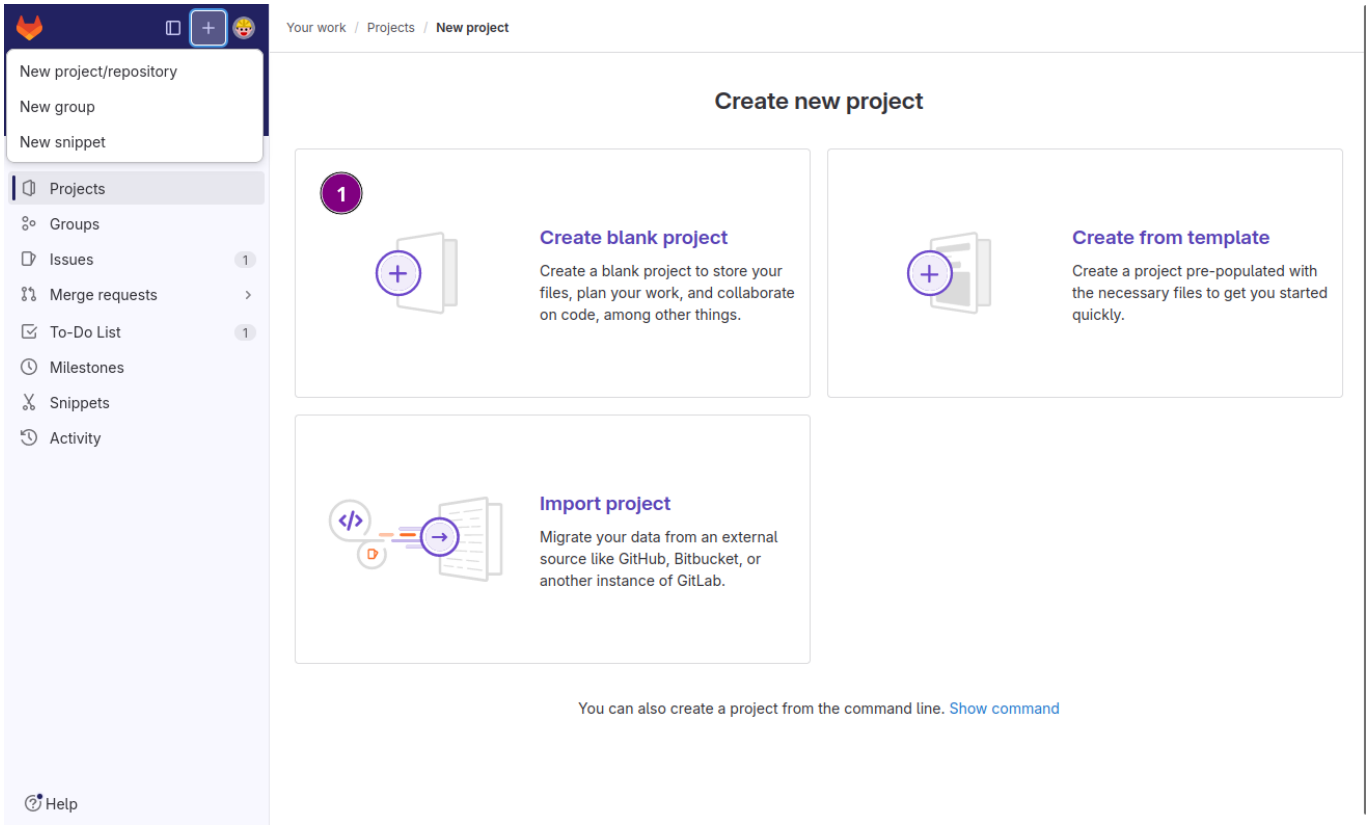

Figure 2. Choose project's starting point

1 Select **`Create blank project`** to open the creation form.

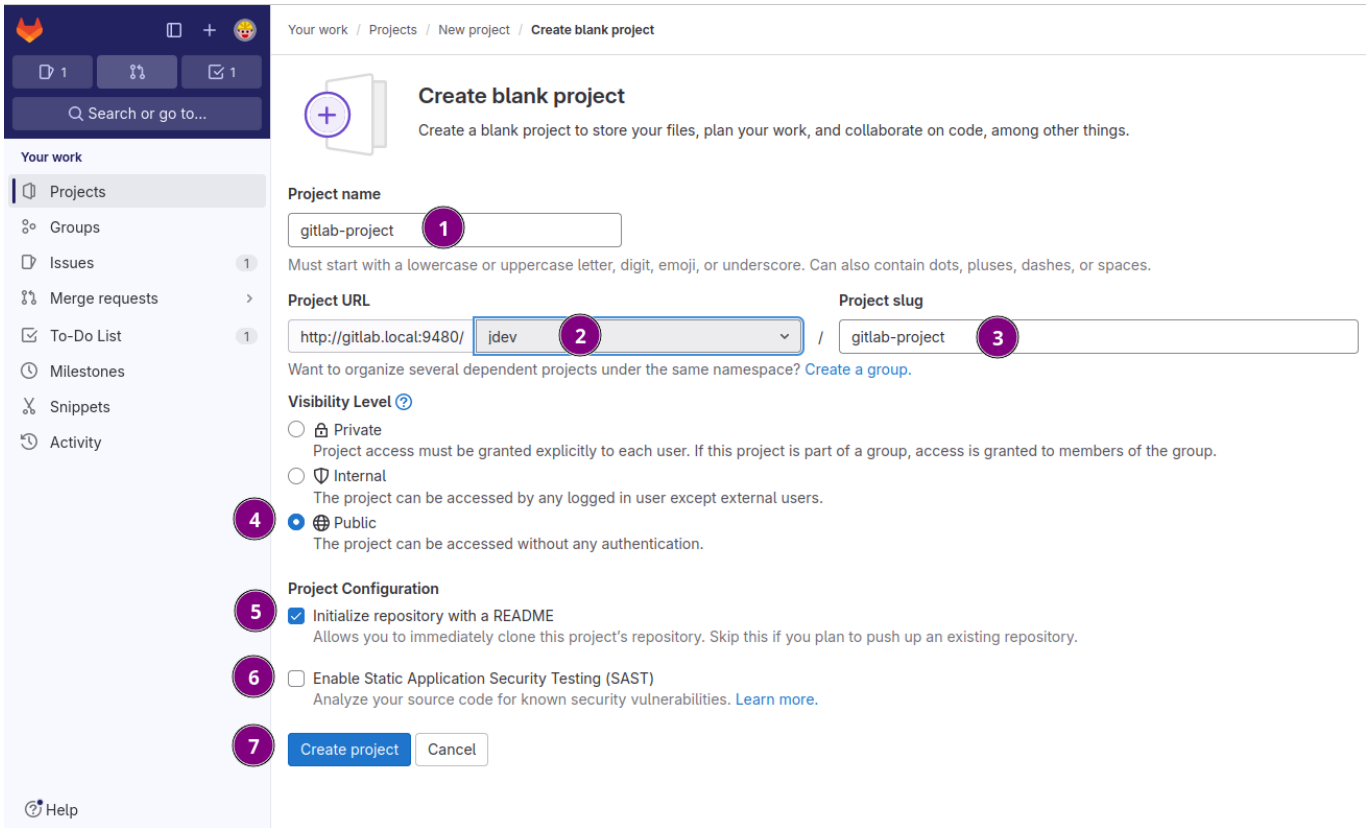

Figure 3. Fill in the required information

1 Use gitlab-project for **`Project name`**

2 Is the base URL for accessing the repository, can't be modified.

3 Project slug is pre-filled base on the project name. May be modified but is usually left as is.

4 Choose **`Public`** as the visibility layer.

5 Check **`Initialize repository with a README`** this creates a bare bone **`README.md`** file in the root of the repository.

6. Enable Static Application Security Testing (SAST). This checks the code for vulnerabilities and plain text passwords among other things.
7. To finish hit the [Create project] button.

Navigate new project

Right after the project is created the next screen is the project overview. For a first time visitor the screen has quite a few elements and is probably confusing at first sight.
Here the various elements are explained shortly to provide and initial overview where to find which functionality.

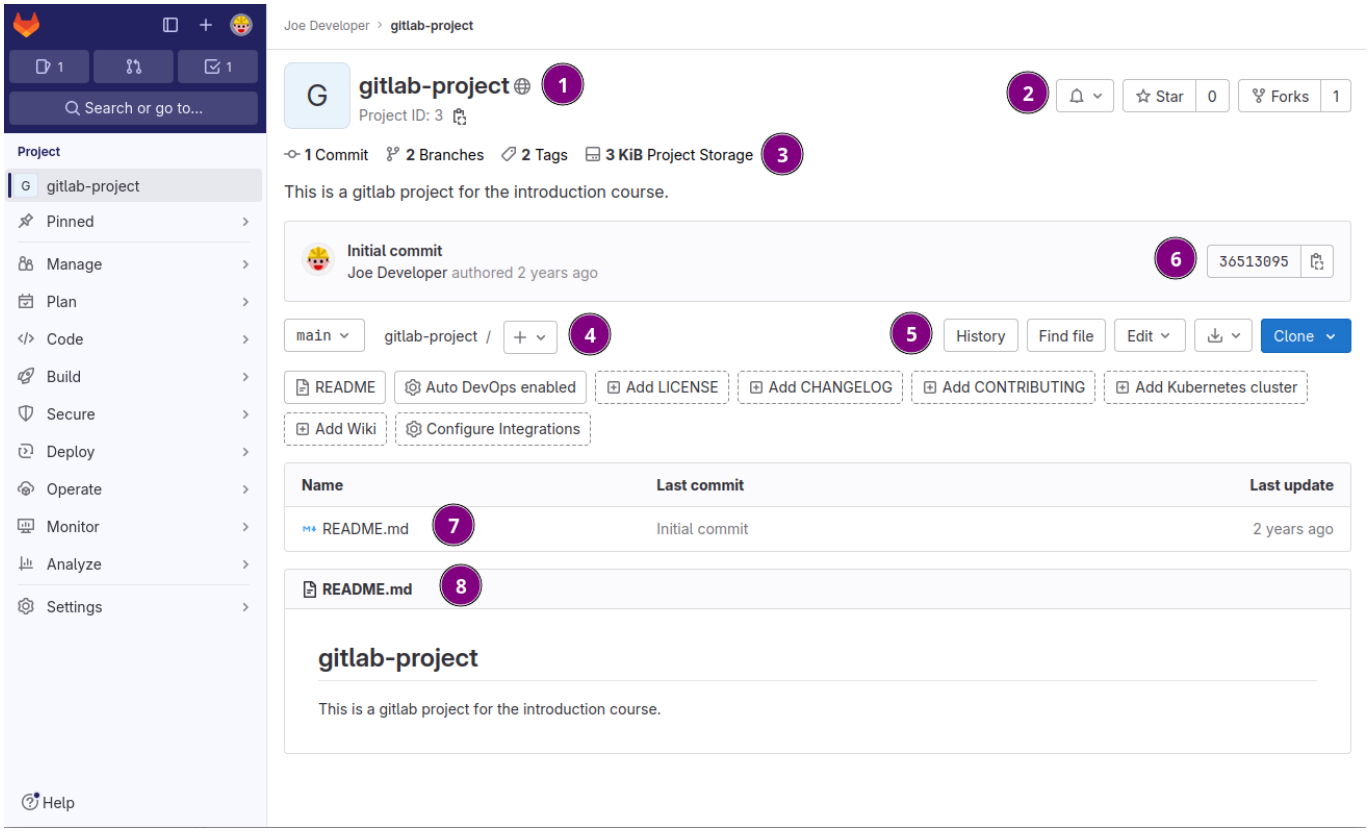

Figure 4. Initial project overview screen.

1. Project name and visibility level marked by icon on the right hand side of the project name.
2. Notification, star and fork buttons for other developers interested in the project.
3. Git repository statistics with regards to number of commits, branches tags and storage used.
4. Branch navigation and file, branch and tag creation.
5. Miscellaneous functionally such as file modification with Web IDE, archive downloads and clone addresses for the repository.
6. Current git revision number in shortened SHA1 format.
7. File browser area, list and navigate files and directories in the project.
8. HTML rendered content of **`README.md`**.
9. Navigation of further functionality and access to project settings.

Clone project

GitLab has quite a few tools allowing for changes to the git repository data via the web interface. But for most purposes the content of the git repository is worked on with an IDE or on the command line externally.
In this exercise the previously created **`gitlab-project`** is cloned from the command line.

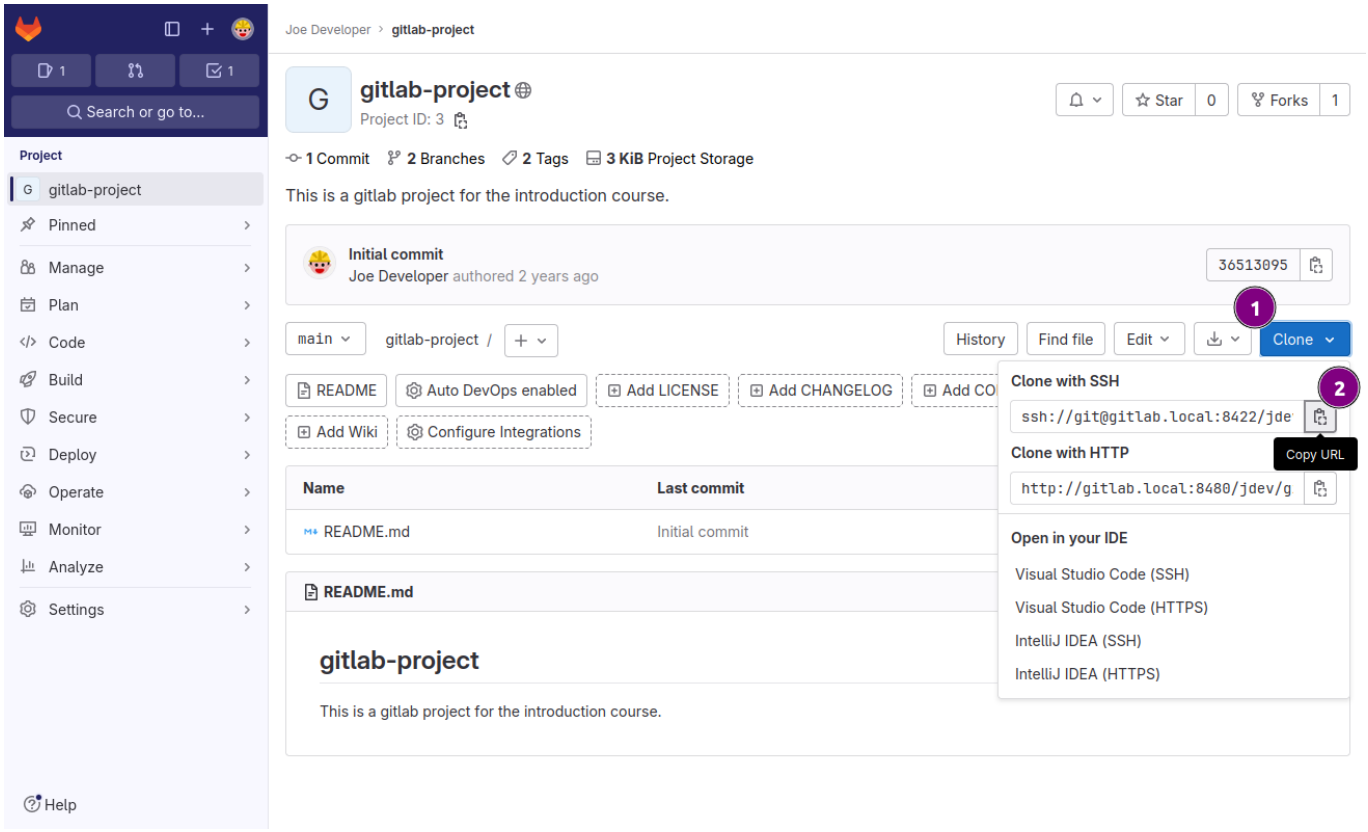

Figure 5. Copy project address for git clone operation.

(1) Press the [Clone] button.

(2) Copy the ssh address.

Change to terminal window and execute

```
$ git clone ssh://git@gitlab.local:8422/jdev/gitlab-project.git (1)
Cloning into 'gitlab-project'...
X11 forwarding request failed on channel 0
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.


$ cd gitlab-project (2)
$ git log (3)
commit 3651309571108055dcafb3dacaf9678e90c91291 (HEAD -> main, origin/main,
origin/HEAD) (4)
Author: Joe Developer <joe.developer@gitlab.local>
Date:   Mon Nov 9 12:10:30 2020 +0000

    Initial commit
```

(1) Clone the repository with the copied URL.

(2) Chang into the freshly created **`gitlab-project`** directory.

(3) Inspect the commit history with **`git log`**.

(4) Verify the commit hash matches with the one displayed in the project overview.

Navigate plan

Besides the managing repositories Gitlab also sports tools to plan like manage issues and kanban boards among others for the project.

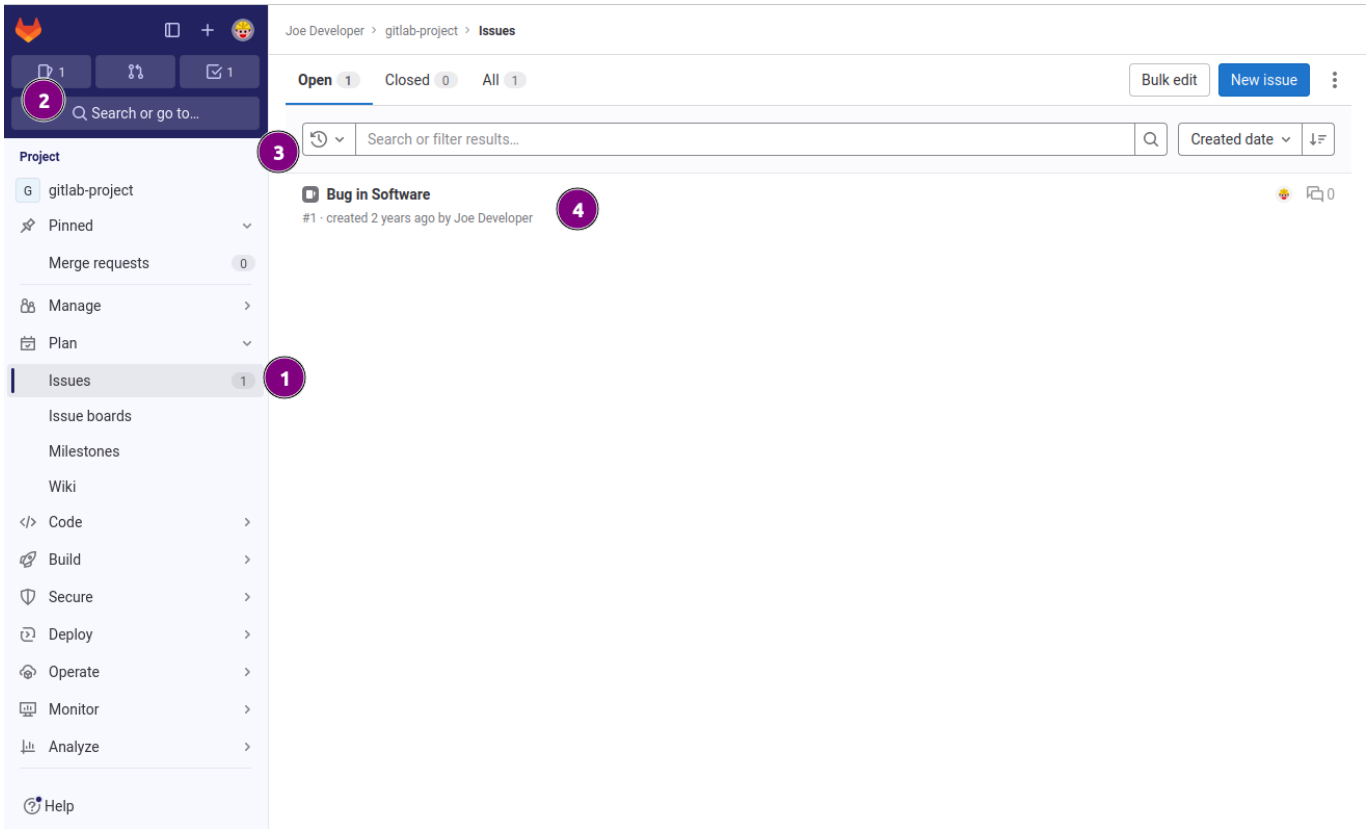

Figure 6. Issue List view

1. Navigate to **`Issues`** for overview.
2. Issue indicator for own account.
3. Filter and search controls.
4. List of issues.

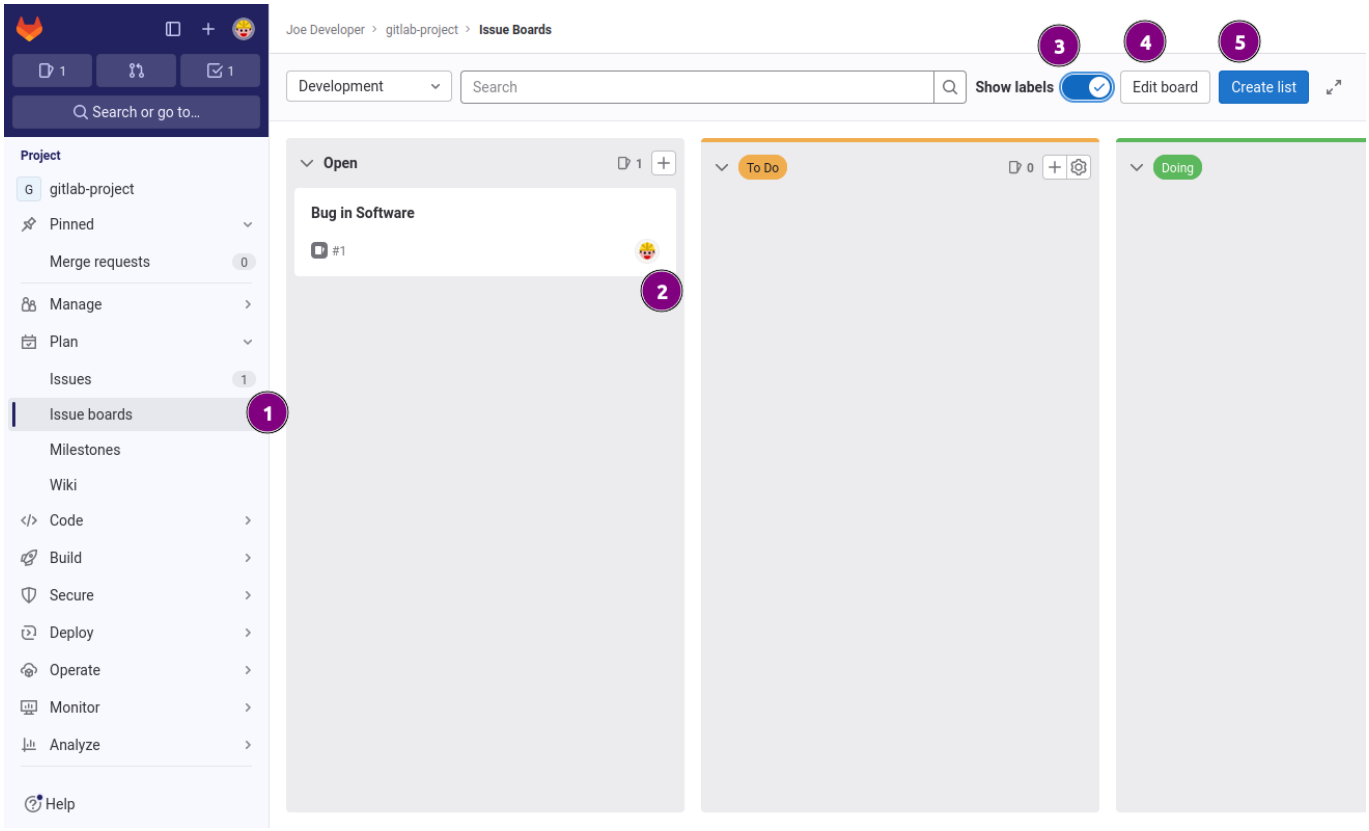

Figure 7. Issue Boards view

1. Navigate to **`Issue boards`** for overview.
2. Open issue assigned to user.
3. Toggle show assigned labels other than from the defined lanes.

4. Change the name and a few basic settings of the current board.
5. Add a new lane to the board.

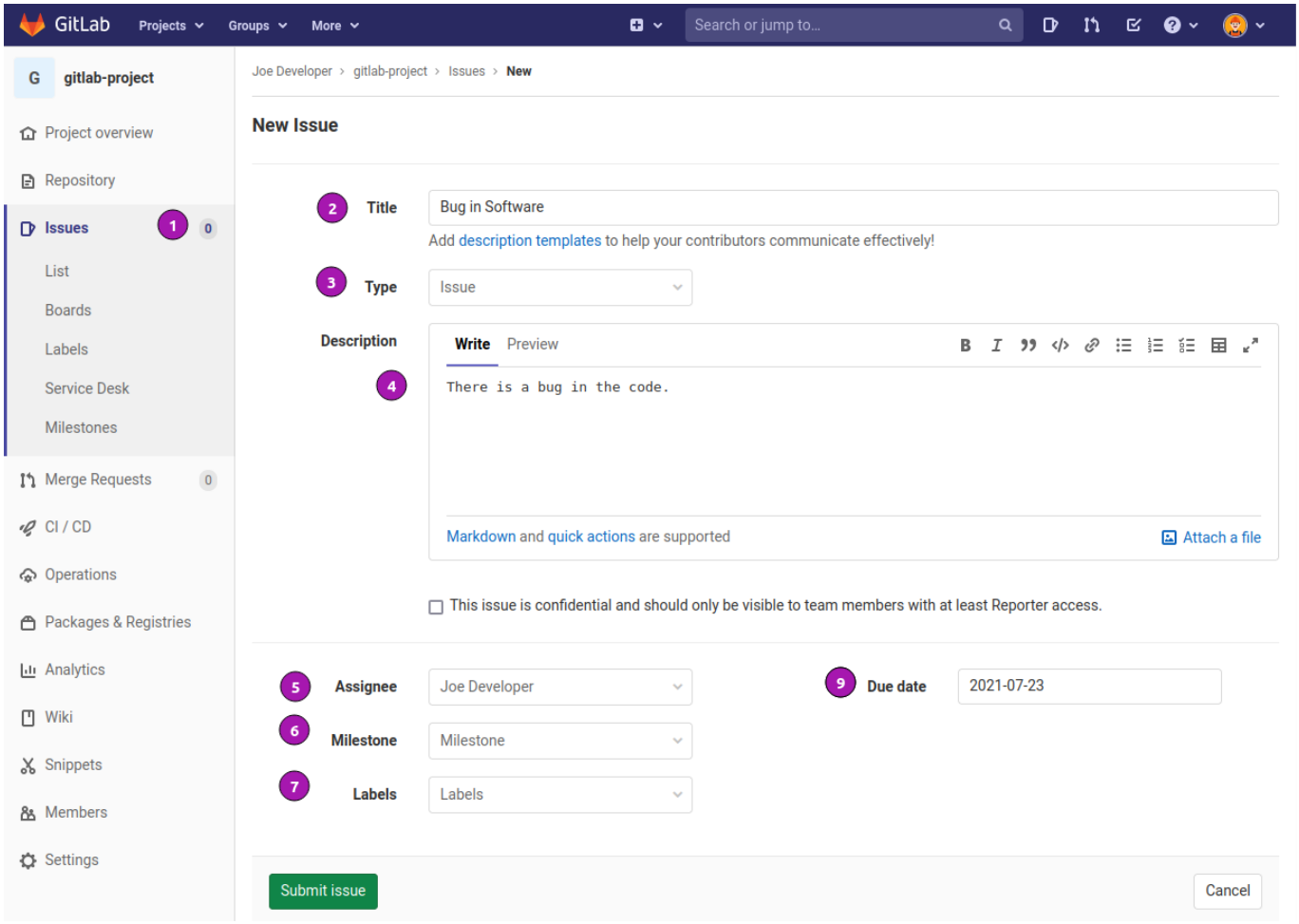

Figure 8. Create new issue

1. Navigate to **`Issues`** for overview.
2. Enter title.
3. Choose type **`Issue`** or **`Incident`**.
4. Provide a description.
5. Choose **`Assignee`**; Optional.
6. Assign or create **`Milestone`**; Optional.
7. Assign or create **`Label`**; Optional.
8. Choose **`Due date`**; Optional.

Navigate code

There are a few tools to get a better insight of what is happening with the code both in and around the repository.

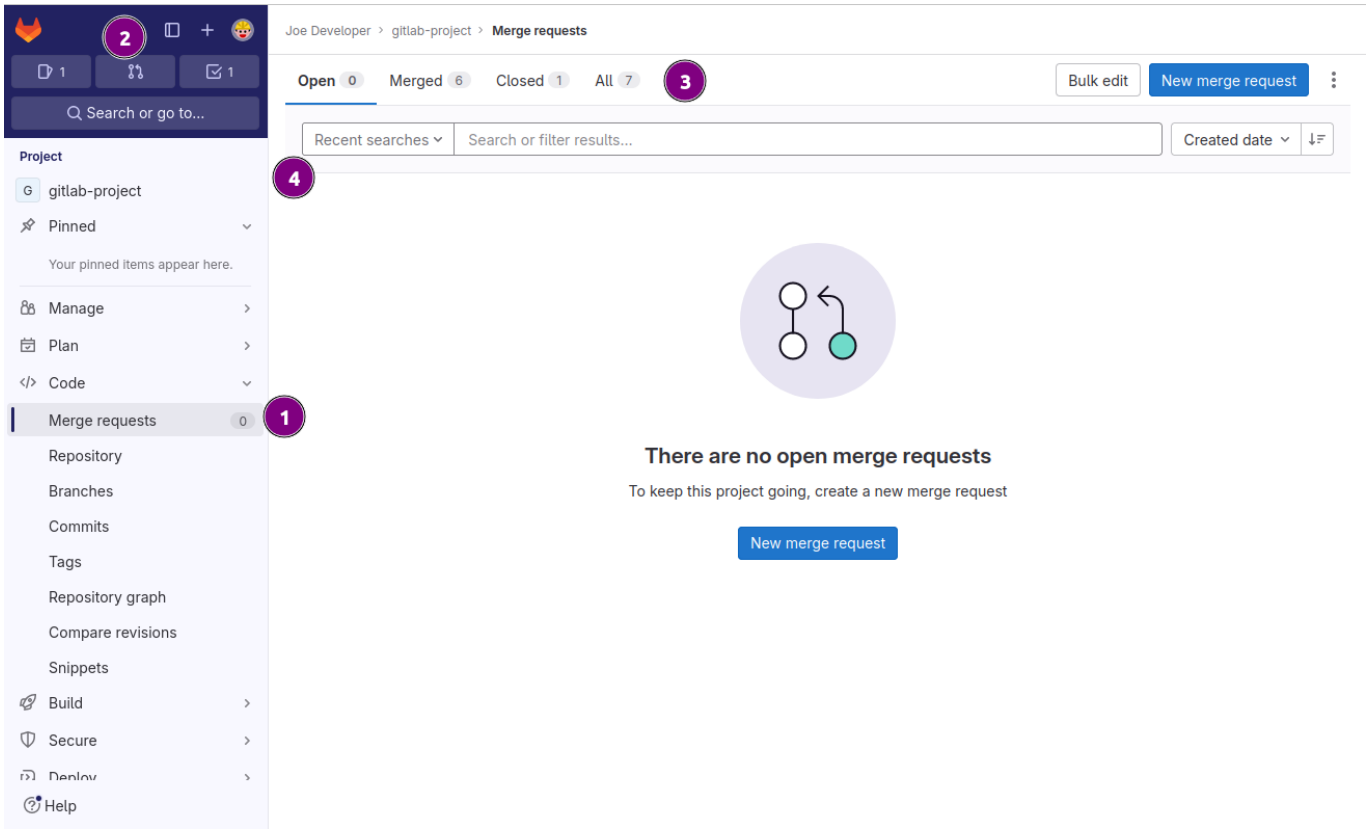

Figure 9. Merge requests overview

1. Menu item for viewing **`Merge requests`** for current project.
2. Global indicator for all assigned merge request.
3. Quick filter for current and previous merge requests.
4. Search to filter merge request by author and labels among other criteria.

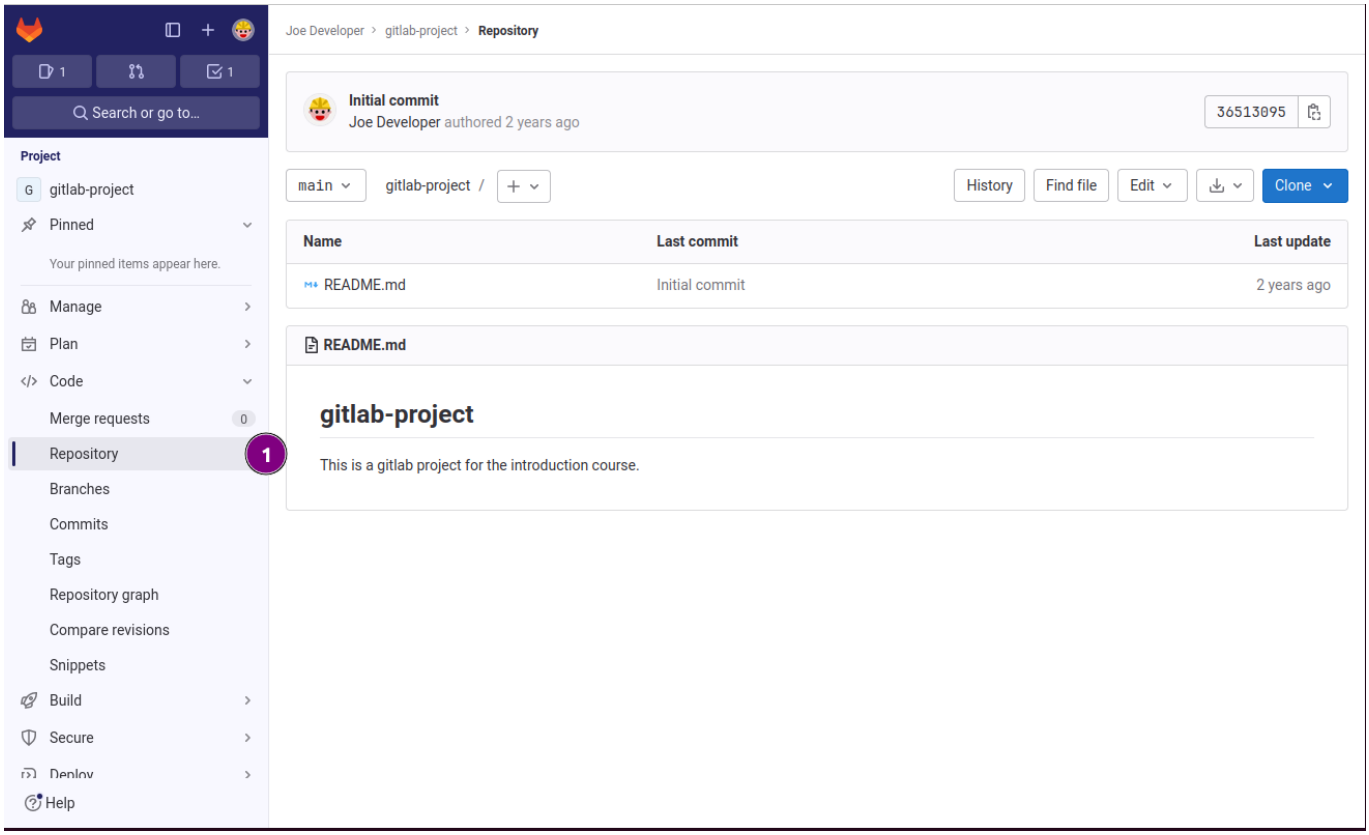

Figure 10. Repository view

1. **`Repository`** presents a similar layout as the project overview but in a slightly more compact format.

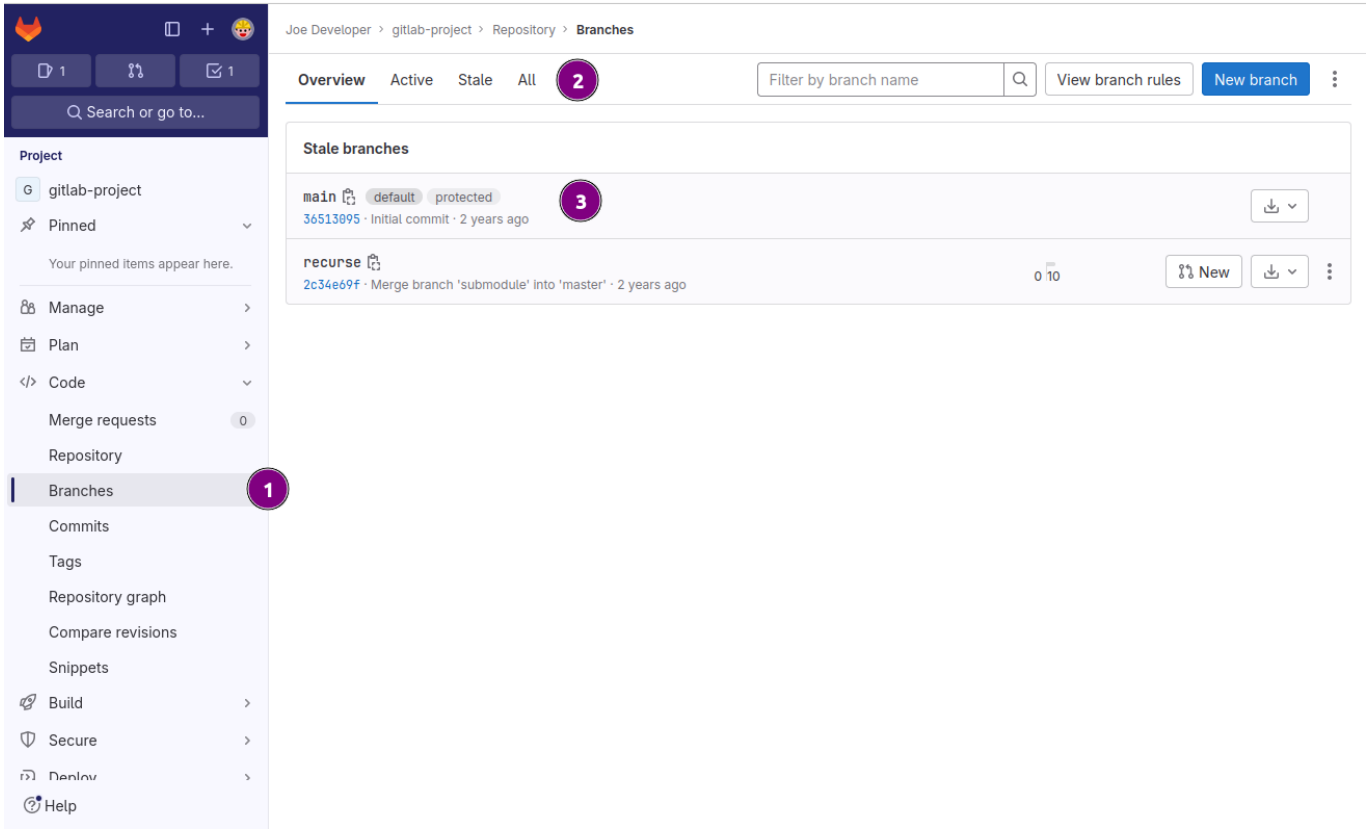

Figure 11. Branches view

1. Navigate to **`Branches`** for overview.
2. Filter branch by name.
3. List of branches with labels and actions.

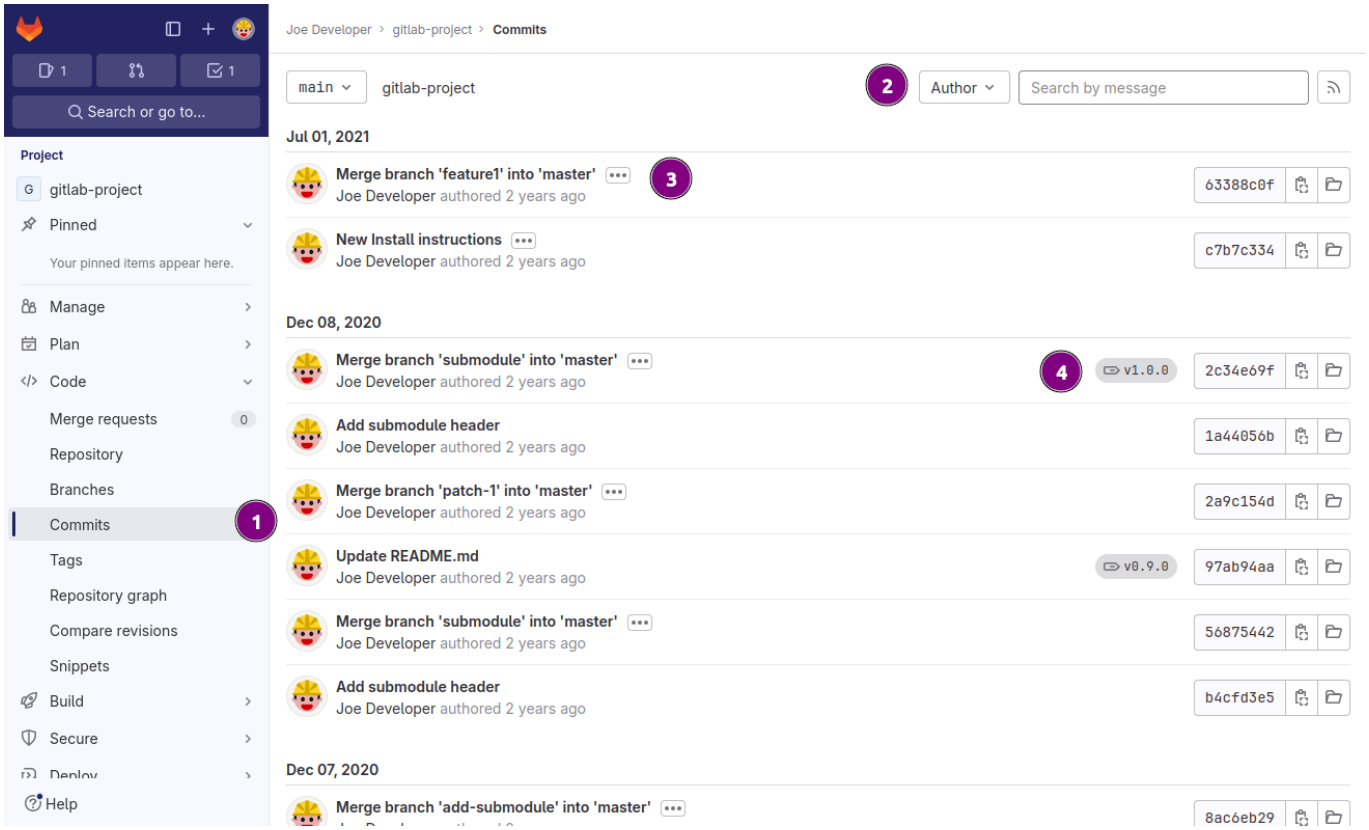

Figure 12. Repository Commits view

1. Navigate to **`Commits`** for overview.
2. Filter by one or more authors.
3. List of commits in default branch **`main`** equivalent of **`git log`** on the command line.
4. Display tags assigned to commit.

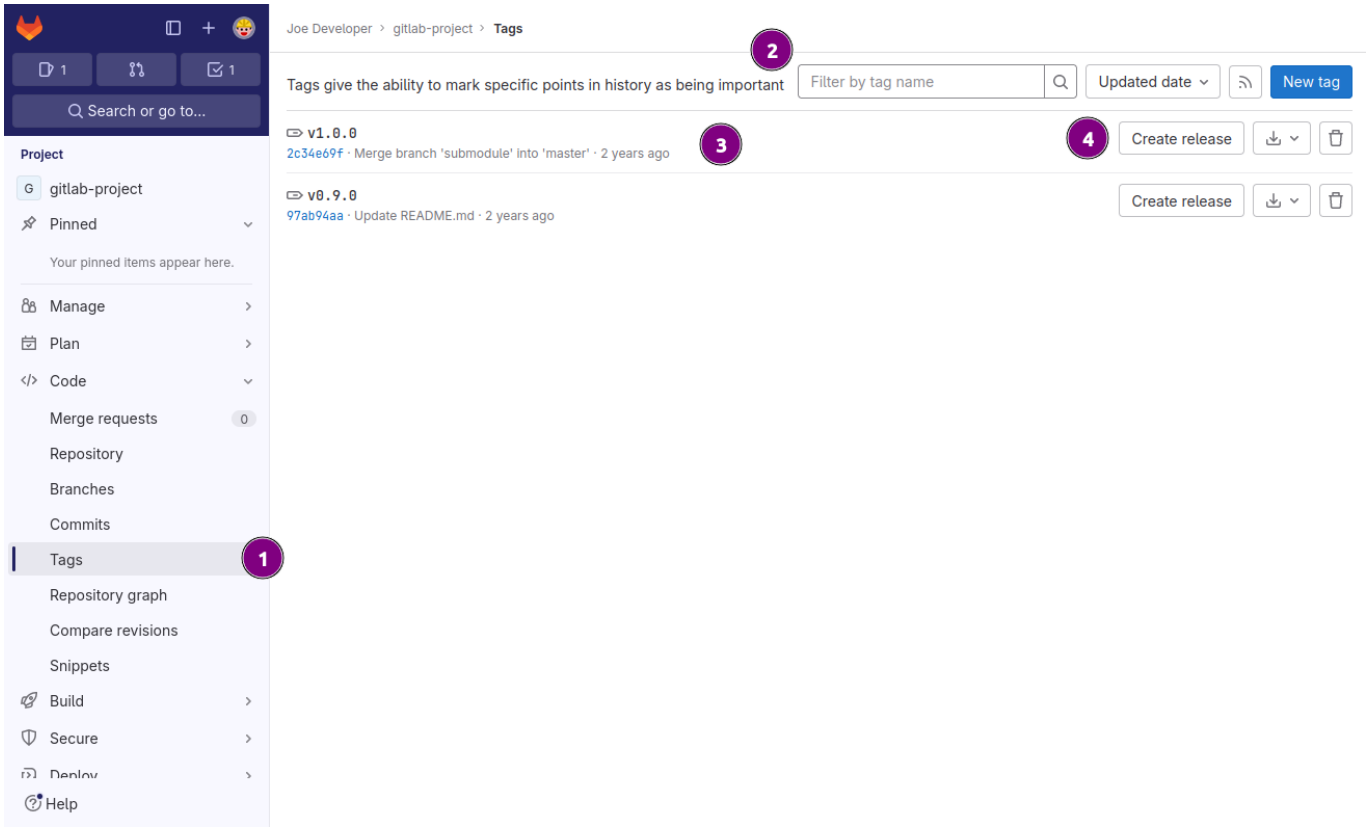

Figure 13. Tags view

1. Navigate to **`Tags`** for overview.
2. Filter tags by name.
3. List of tags with actions.
4. Create a release from tag.

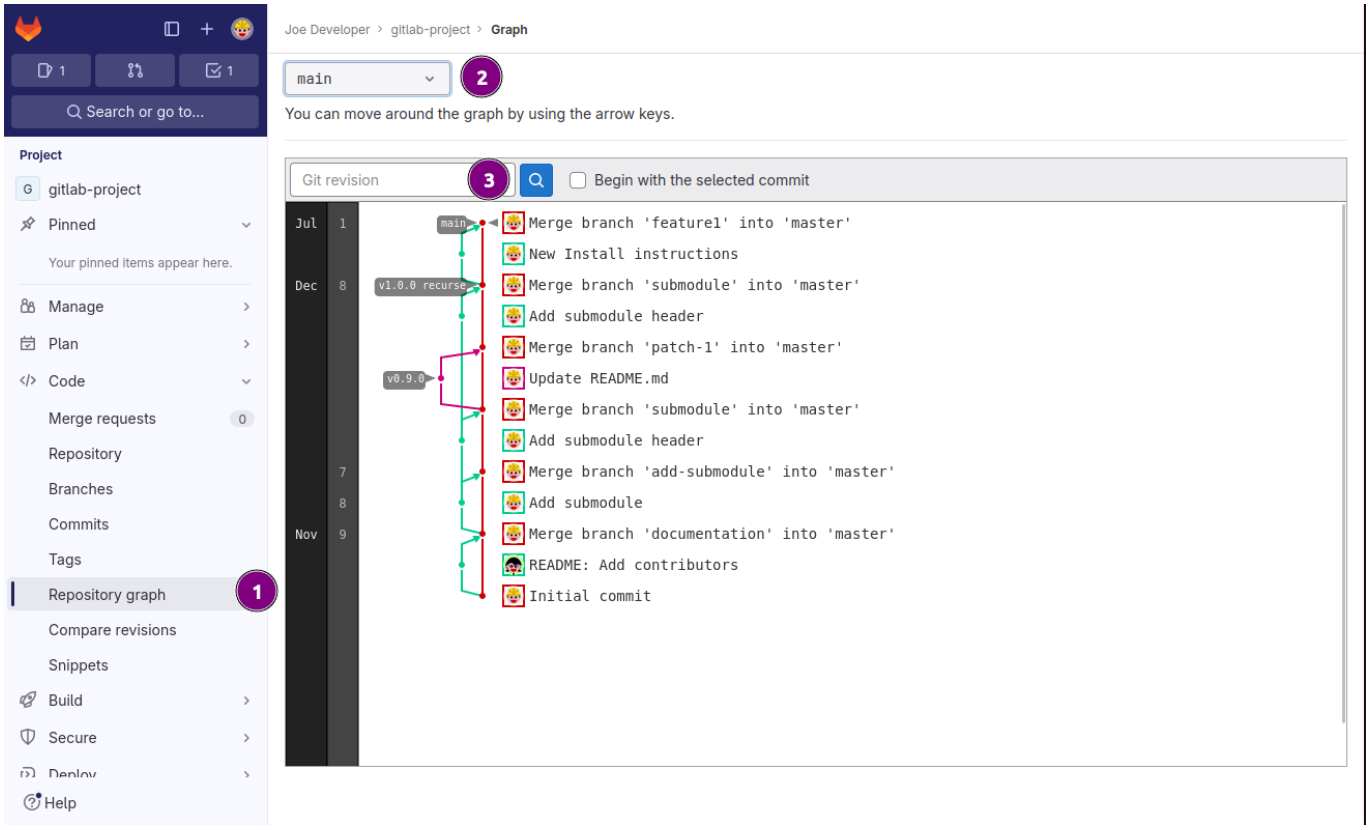

Figure 14. Repository graph view

1. Navigate to **`Graph`** for overview.
2. Choose branch or tag to display.
3. Filter by specific commit hash.

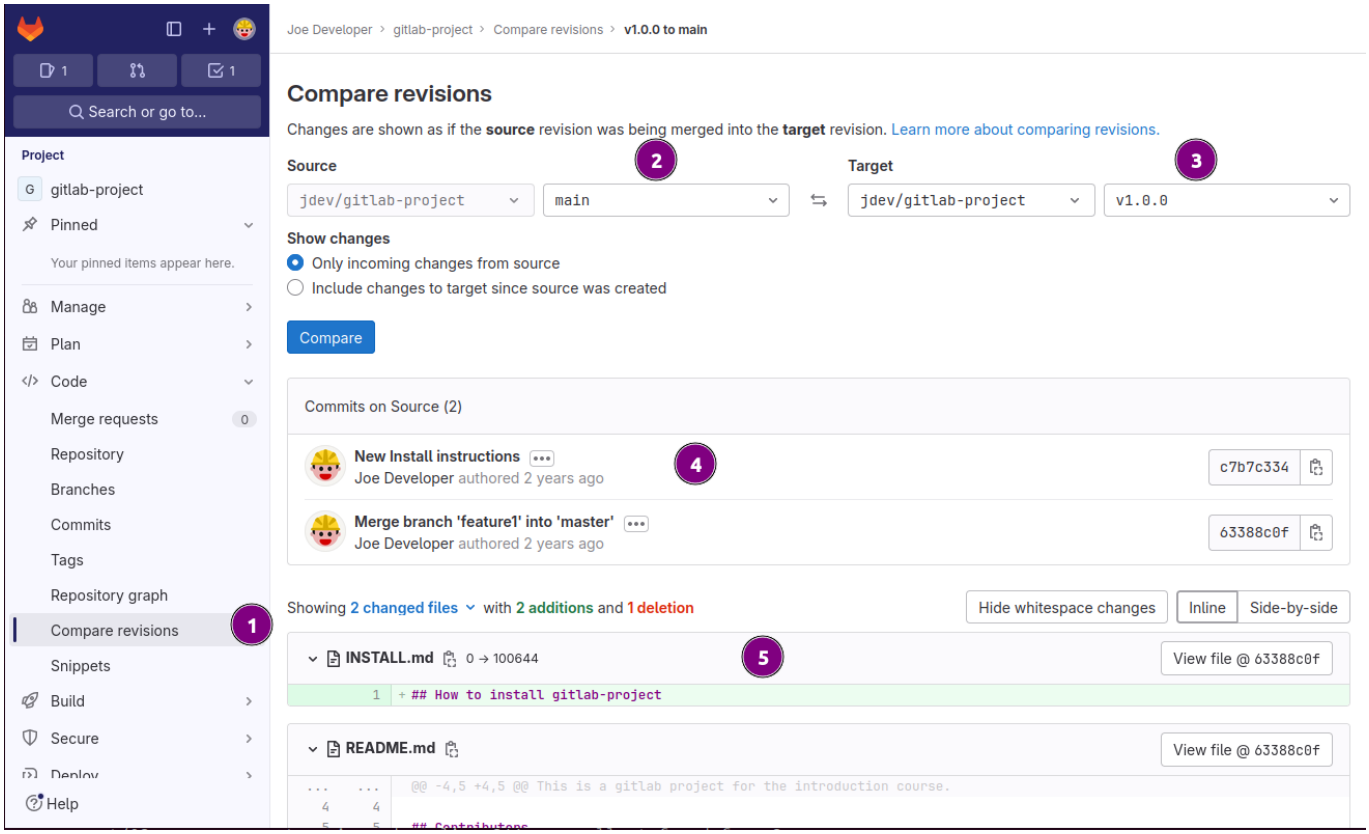

Figure 15. Compare view

1. Navigate to **`Compare`** for input mask.
2. Select source repository branch or tag.
3. Select target repository branch or tag.
4. List of commits only found in source.
5. File diffs of the changes.

Repository protection

Among the many settings available for a GitLab project at this stage the one that really matters is the protection of the main branch.
Per default only the maintainer can push and as such also force push changes to the **`main`** branch. This can be further restricted and no one make changes to main without proper evaluation of the changes.

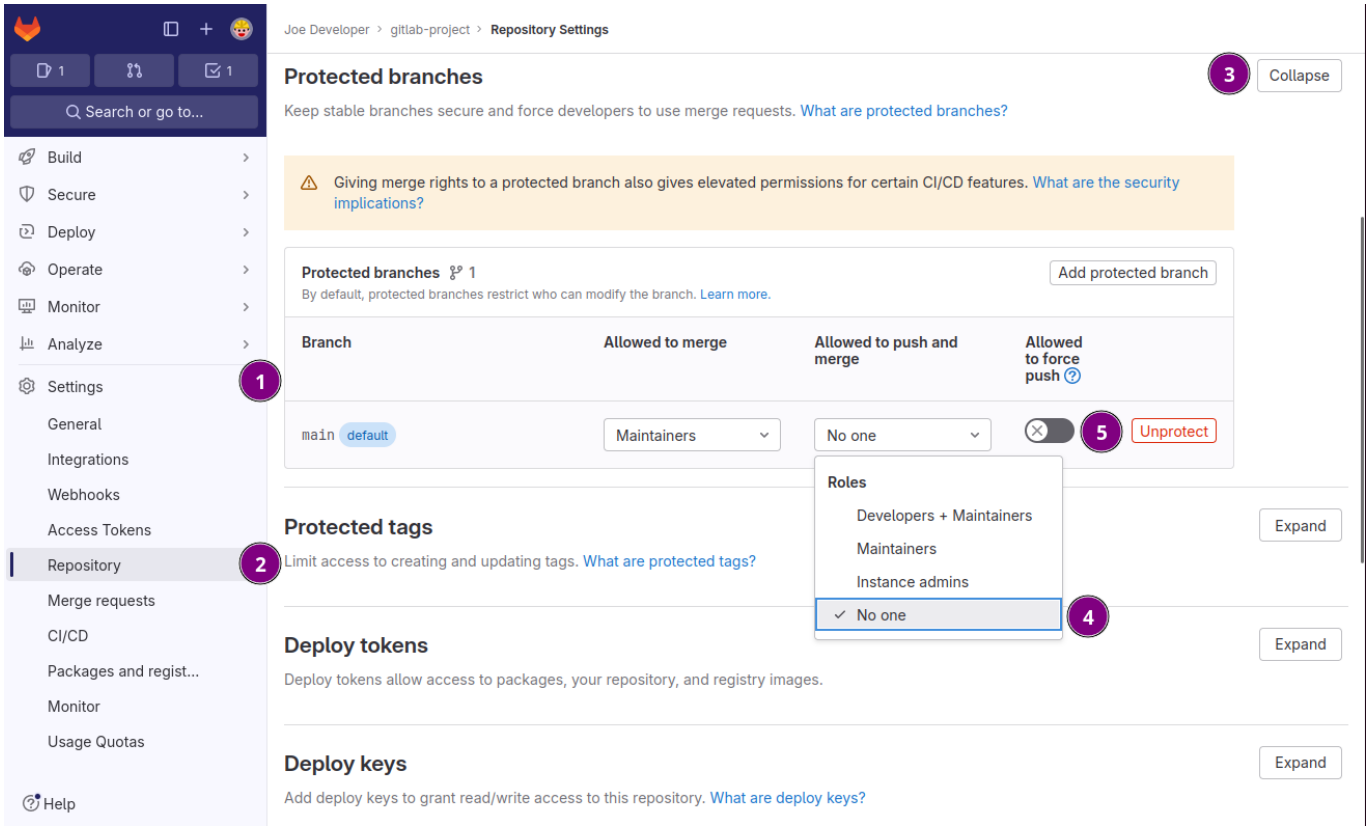

Figure 16. Restrictive main branch settings.

1. In the project navigate to **`Settings`**
2. Click on **`Repository`**.
3. Navigate to **`Protected Branches`** and hit the [Expand] button.
4. Go to the box for **`main`** under allowed to push choose **`No one`**.
5. Toggle for preventing **`force pushes`** to the branch.

By protecting the main branch from direct pushes one can prevent fat-fingered git pushes where main gets accidentally overwritten with content from another branch. As a disclaimer this never happened to the instructor guiding you currently, ever!!!

Gitlab group

A GitLab group is a way to structure and manage one or more related projects. Groups can also be compared to folders on a file system.
Groups inherit permission to their subgroups and/or projects. Say if someone has access to a group, they also get access to all the projects in the group.
This also includes the issues and merge requests for the projects in the group, and analytics for the group's activity.
Groups allow to communicate with all of the members at once.

Create a new group

In the following steps a new group is created. The process of creating a new group is assisted and straight forward.
The creation starts with clicking on the boxed plus sign left to the search field.

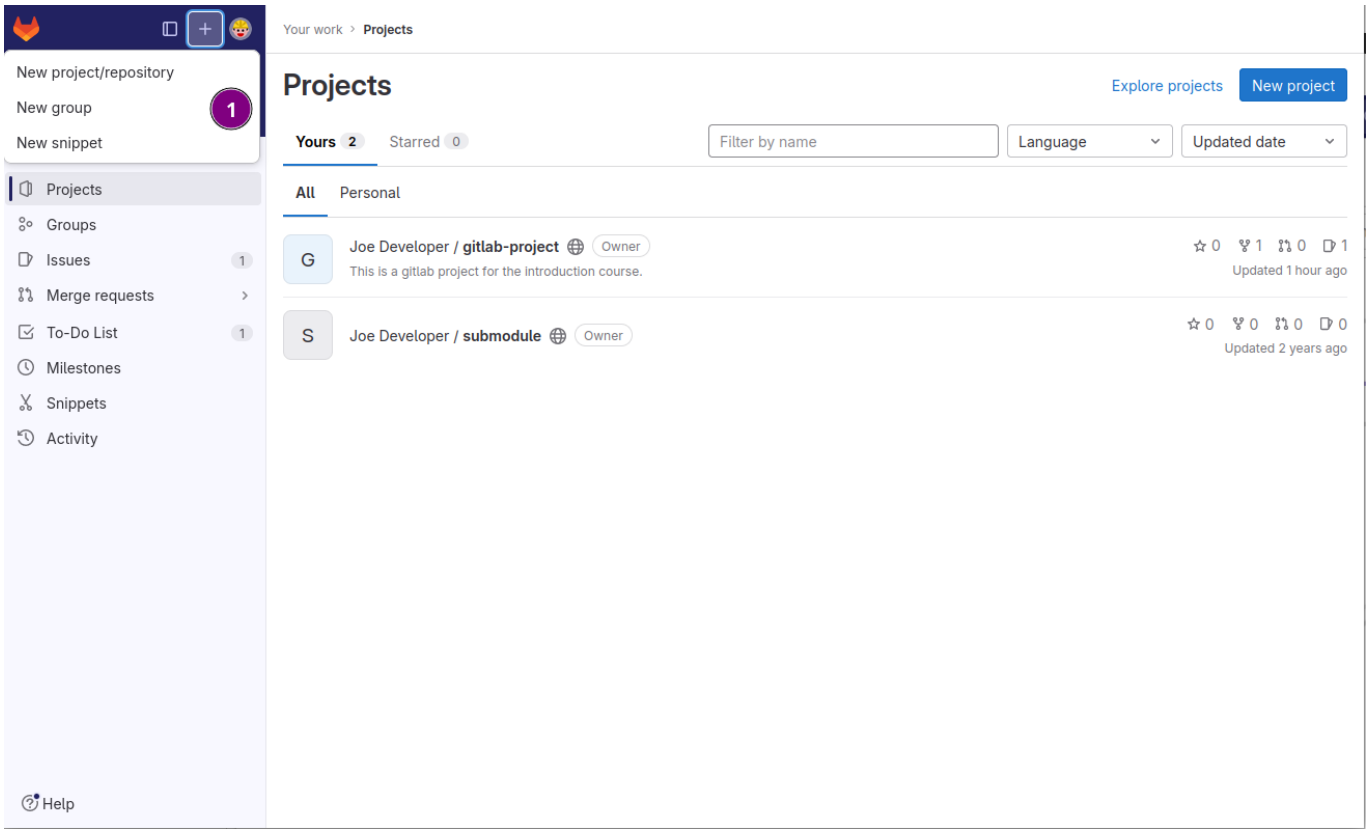

Figure 17. Create a new group from the top navigation bar
1 Choose **`New group`** to open the creation form.

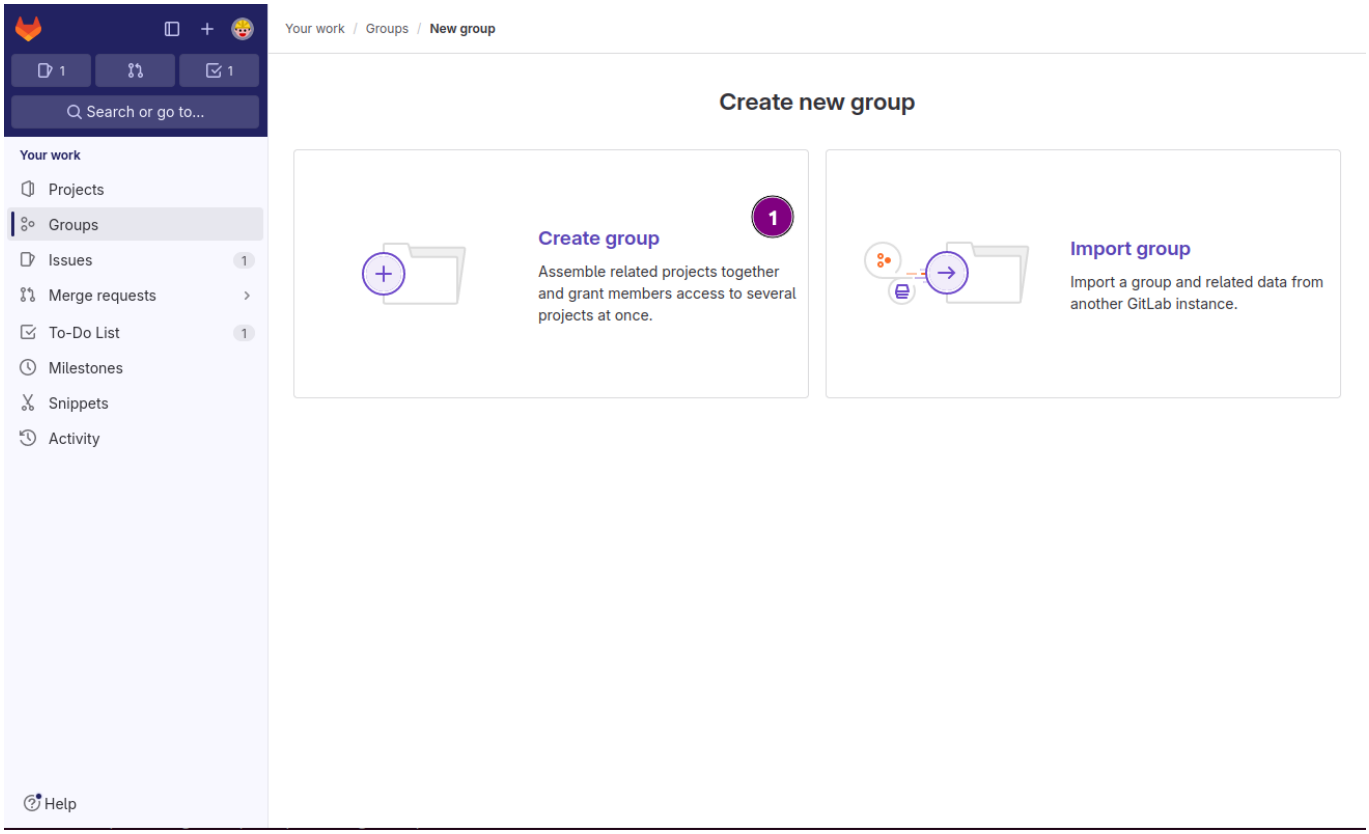

Figure 18. Choose create options

1 Choose **`Create group`** to create a new empty group.

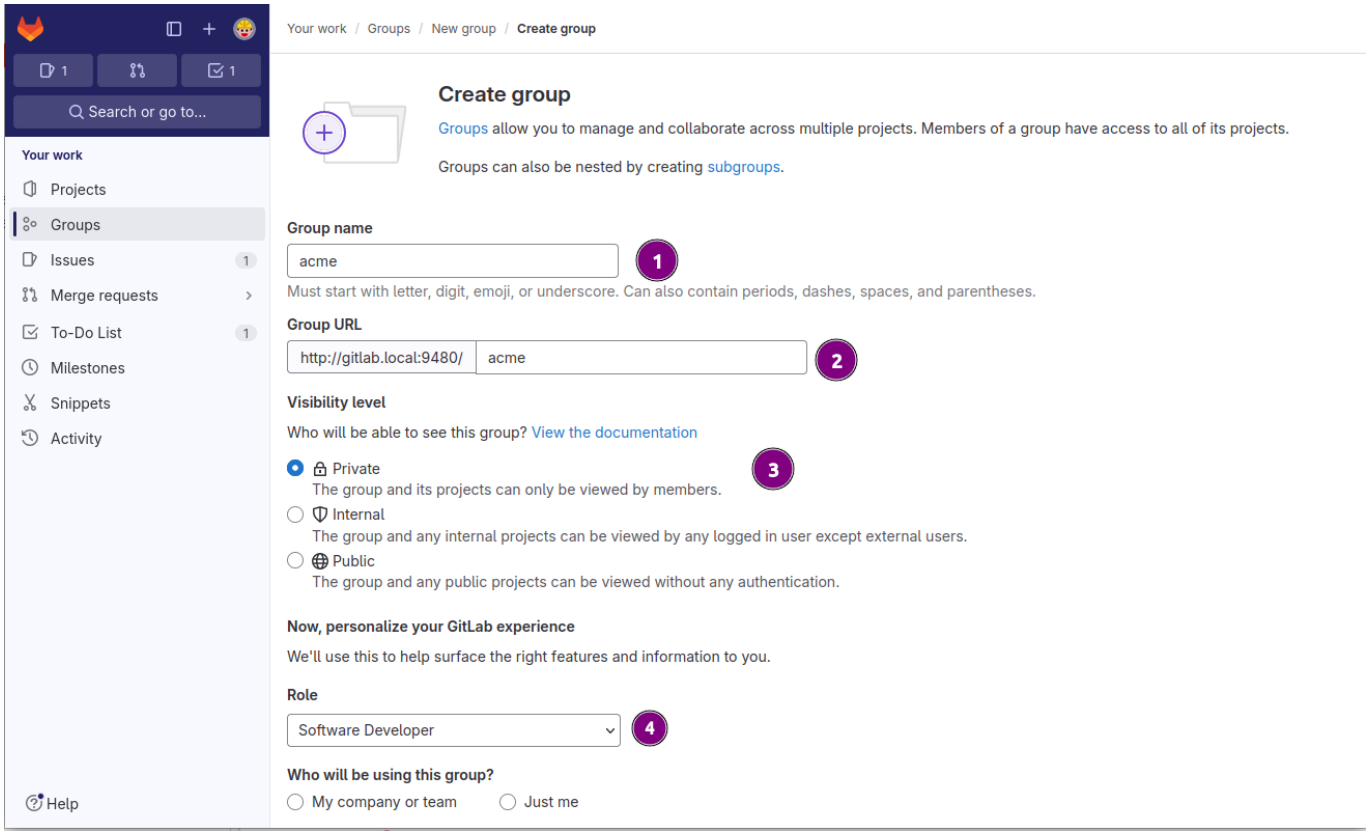

Figure 19. Fill in the required information

1 Use acme as **`Group name`**

2 Specify the **`Group URL`**, usually derived from the **`Group name`**.

3 Specify the **`Visibility level`** of the group. Defaults to **`Private`**.

4 Further optional information for the creation of the group. More questions can be found when scrolling down.

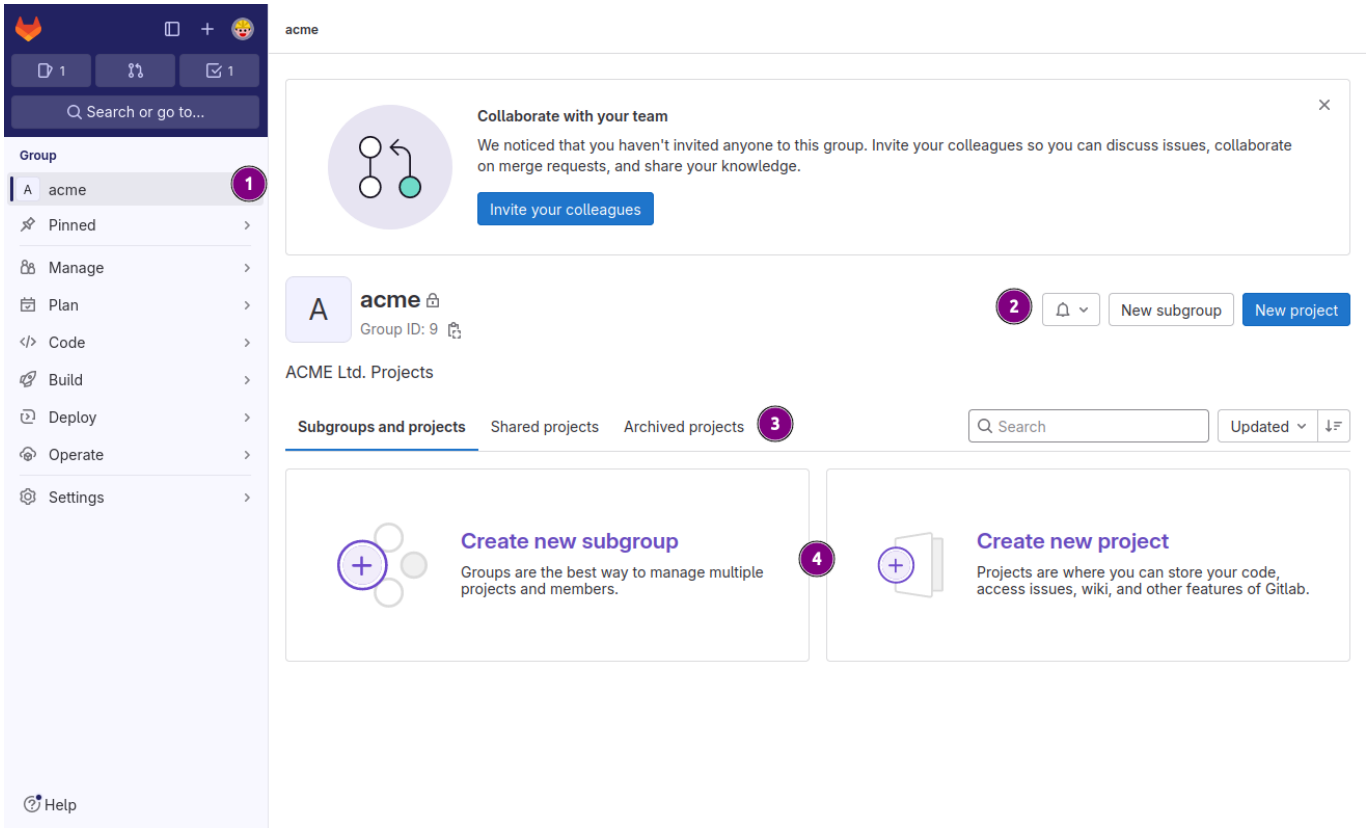

Figure 20. Initial group overview

1. Group navigation and configuration.
2. Notification and new project or subgroup creation.
3. Subgroup and projects tab navigation.
4. Creation of new projects or subgroups.

Group members

An integral part of groups is membership management.
The task is very straight forward but has a couple of Gitlab specific settings which need further explanation.

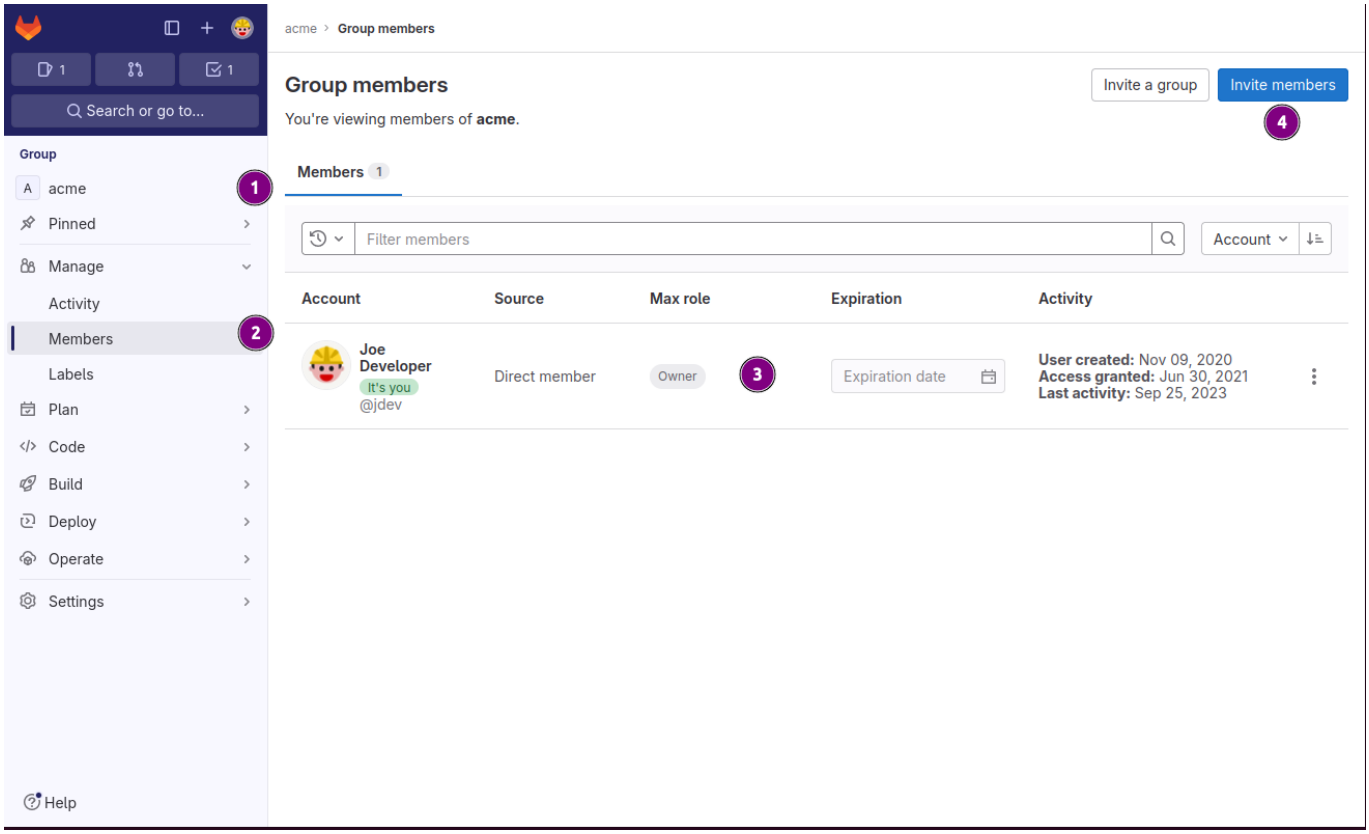

Figure 21. Manage group members

1. Group name.
2. Member list under **Manage**.
3. List of group members.
4. Button to invite new users.

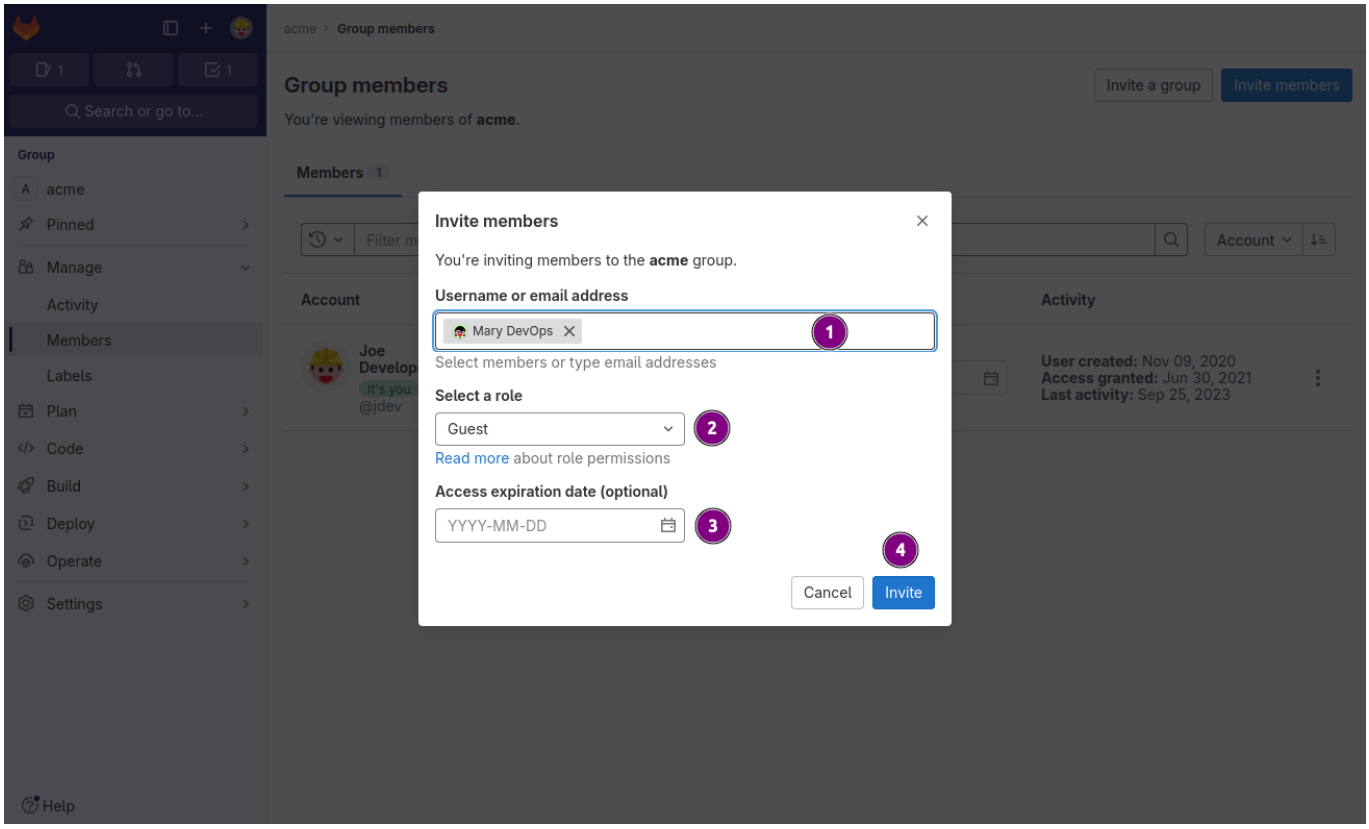

Figure 22. Manage group members

1. List of members to invite.
2. Role the members will fill in the group.
3. Expiration date. E.g. how long the member will have access to the group.

4. Button to send the invitation.

Merge requests

GitLab's statement about merge requests:
Merge requests are a place to propose changes you've made to a project and discuss those changes with others
Interested parties can even contribute by pushing commits if they want to.

Goals

Learn how to collaborate, review and evaluate changes to a repository before adding the changed to the **`main`** branch. After a merge request has been put forward for review the process is shown of how evaluate the change comment on particulars in the change and request amendments and finally merge the code.

- Create a merge request from a feature branch.
- Submit a merge request.
- Review changes in a merge request.
- Amend a merge request.
- Merge the request.

Create merge request

Initiate merge request

There is a few ways to initiate a merge request under GitLab. The variations and locations here are listed below.
For a limited time after pushing changes to a repository a message with a button is display in the upper half of the overview page.

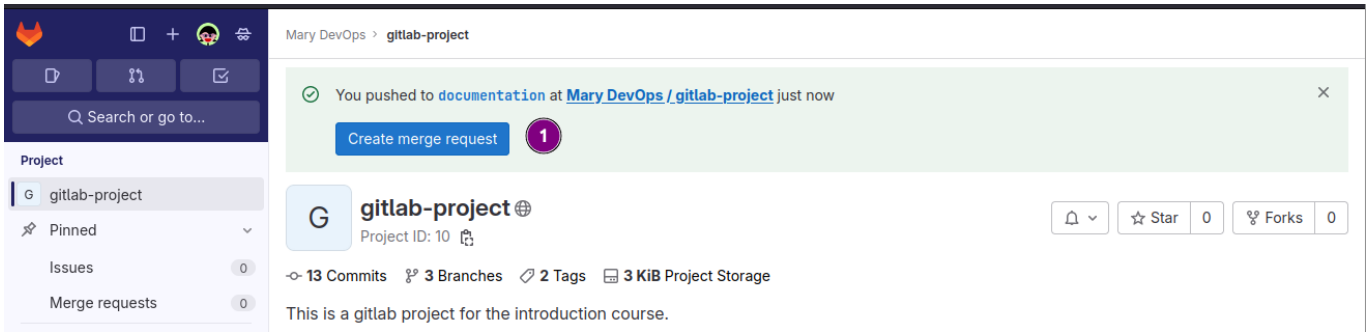

Figure 23. After pushing a branch notification appears on the overview page.
1 Click the [Create merge request] button to initiate.

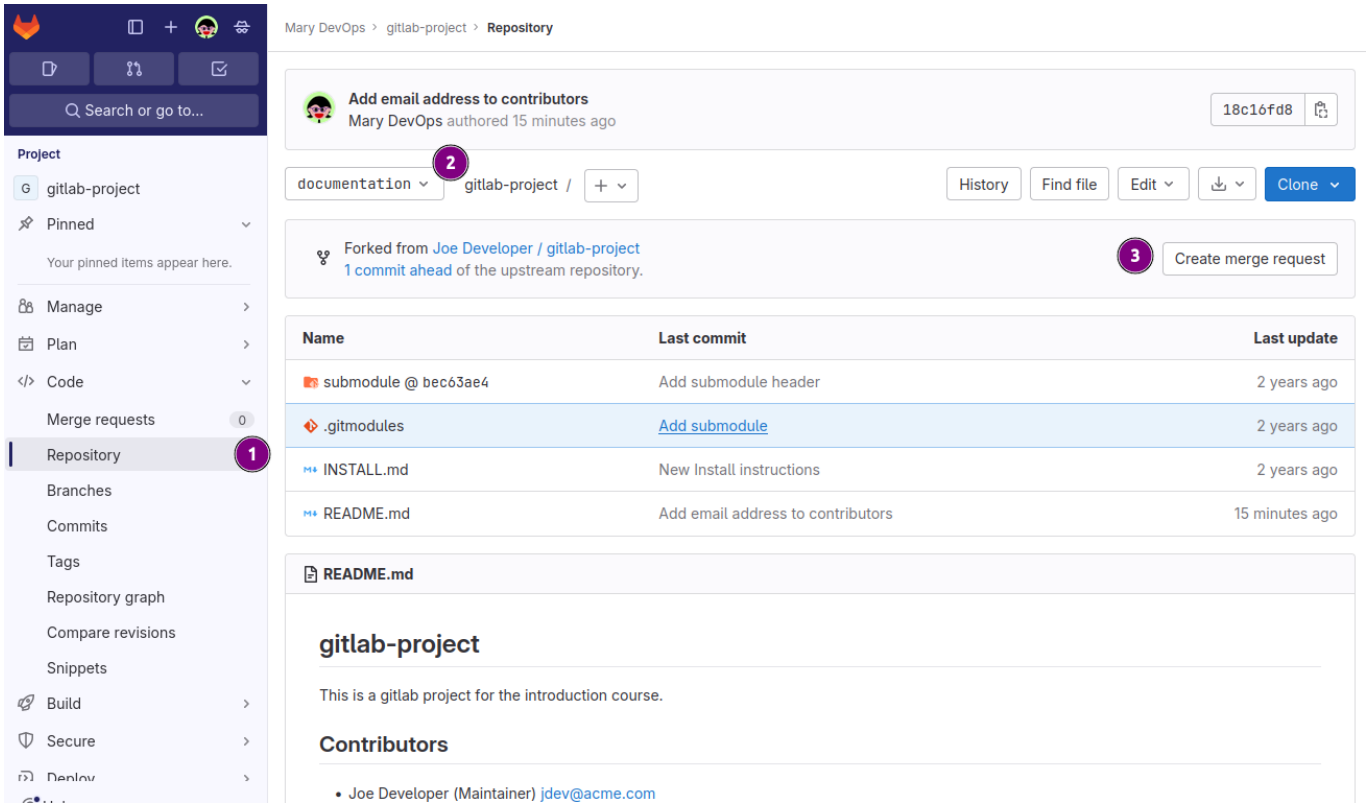

Figure 24. Create merge request via **`Code → Repository`**

1. Navigate to **`Code → Repository`**
2. Select the branch e.g. **`documentation`**.
3. Initiate via the [Create merge request] button.

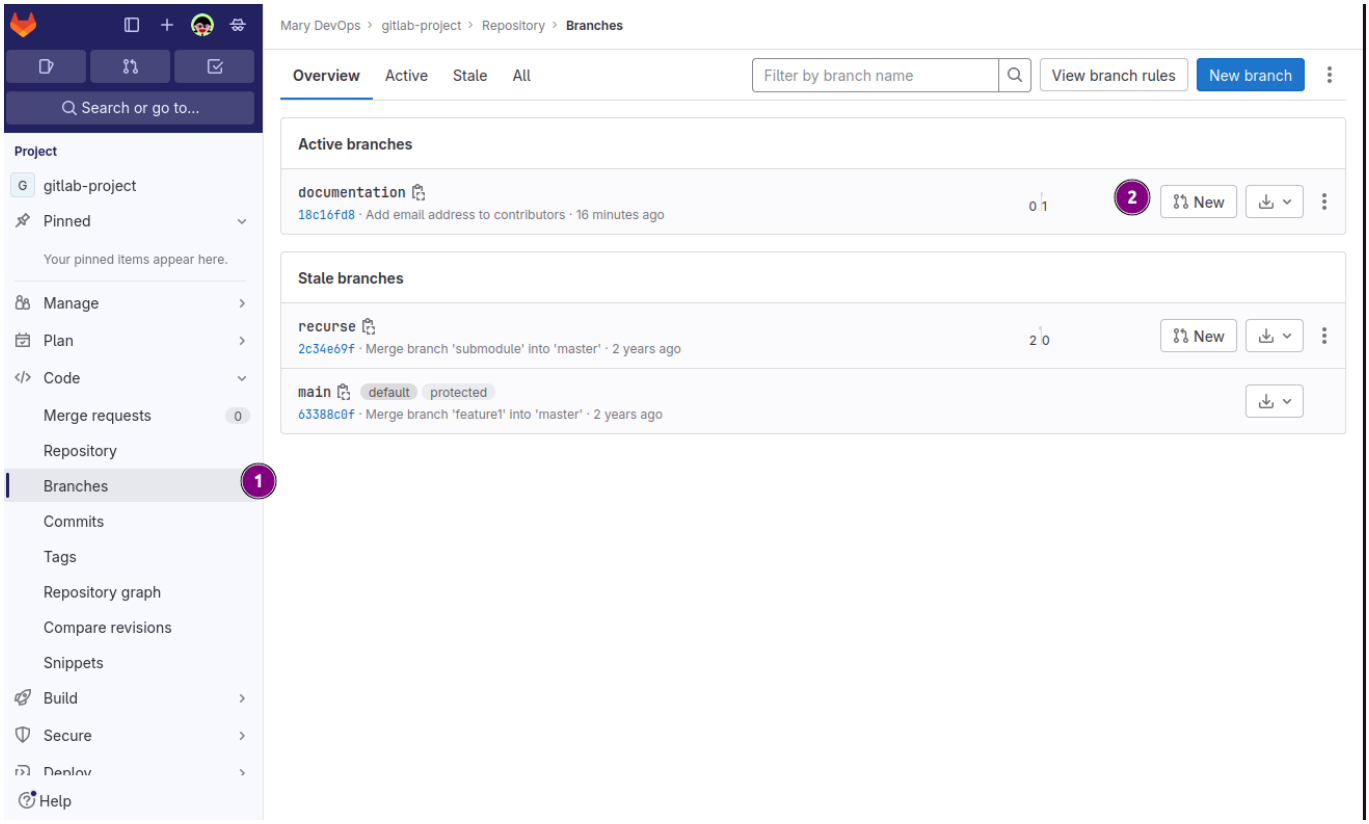

Figure 25. Create merge request via **`Code → Branches`**

1. Navigate to **`Code → Repository`**
2. Click on [New] button of the branch to be merged.

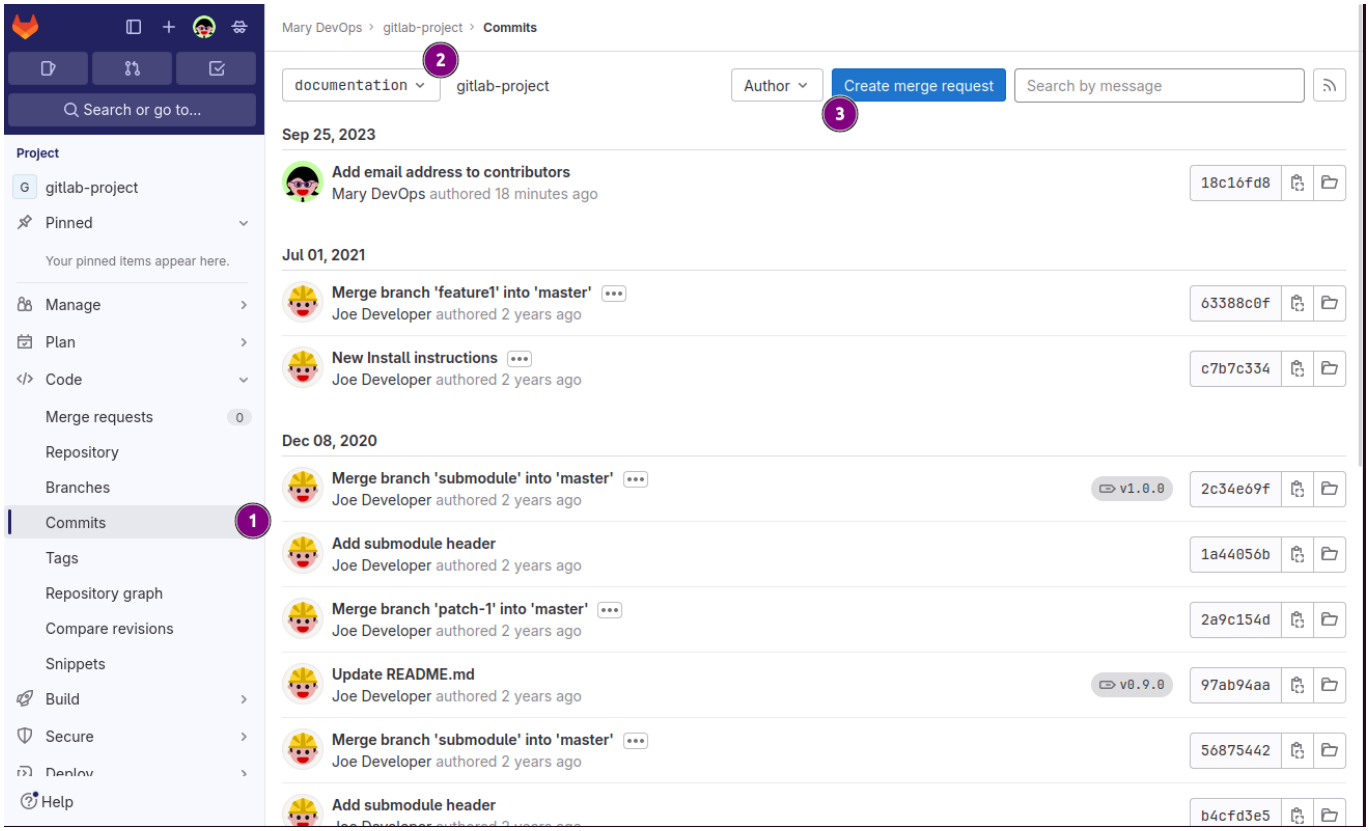

Figure 26. Create merge request via **`Code → Commits`**

1. Navigate to **`Code → Commits`**
2. Select the branch e.g. **`documentation`** to be merged.
3. Initiate via the [Create merge request] button.

Submit merge request

To submit the merge request there are a few questions to be answered generally the defaults are fine most of the time.

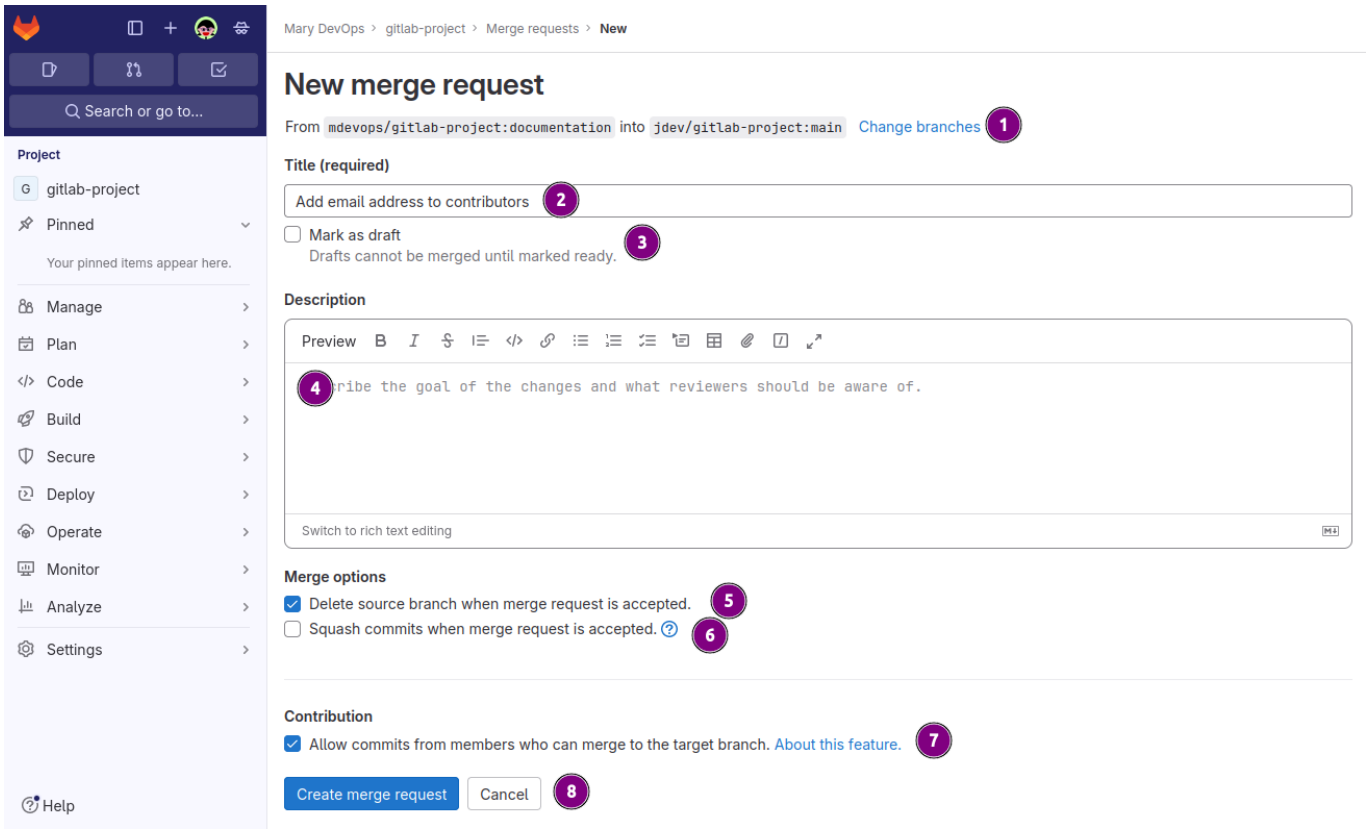

Figure 27. Create merge request via **`Branches`**

1. Change the source or / and destination branch for the merge request.
2. Change the title of the merge request. By default the title is the first line of the commit message.
3. Mark as draft to prevent them from being merged before review.
4. Add a description. Per default it is filled from the commit message starting at line 2 and greater.
5. Deletes the source branch after the merge is done. This is checked by default and prevents out of date branches lingering around too long.
6. When a merge request contains more than one commit the commits are squashed into a single commit before applying to the target branch.
7. Allows members of the target project to submit adjustments to the request before merging.
8. Submit the merge request to the target branch owner.

Review merge requests

View merge request

Once the merge request is submitted the other members of the project can view, comment and give suggestions for minor code changes.

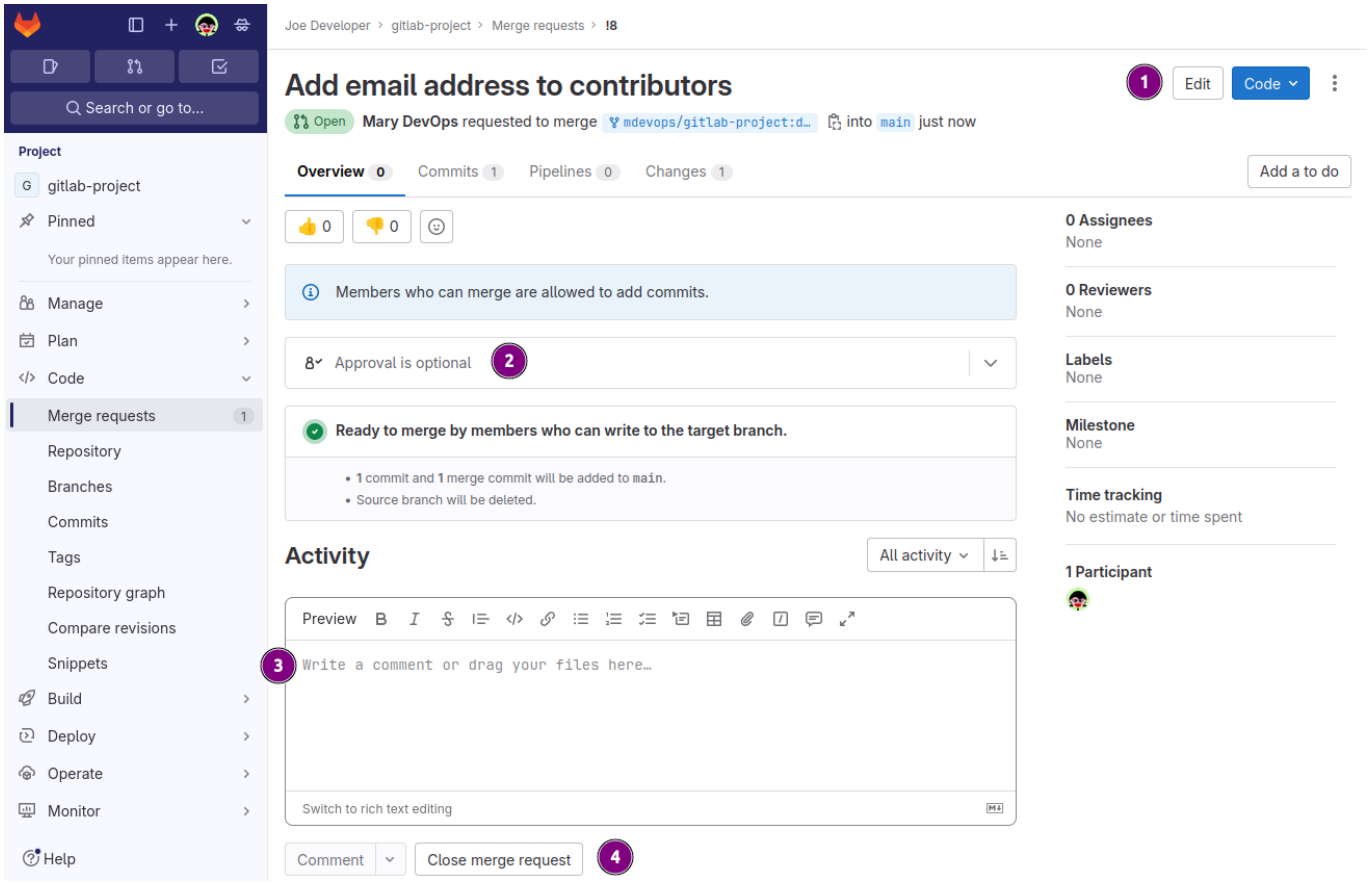

Figure 28. Merge request right after submission.

1. With [Edit] the title and message can be amended.
2. Used to [Approve] the change. Under the GitLab community edition this is gratuitous. The enterprise edition can enforce approval from stakeholders.
3. Add comments to the request.
4. [Close merge request] is another term for canceling the request. Note the request stays in history and can be reopened if required.

Add comment

Other users in the group are encouraged to review the changes and add comments. Commenting under the [Changes] tab a reviewer can mark the lines of code the comment is referring to.

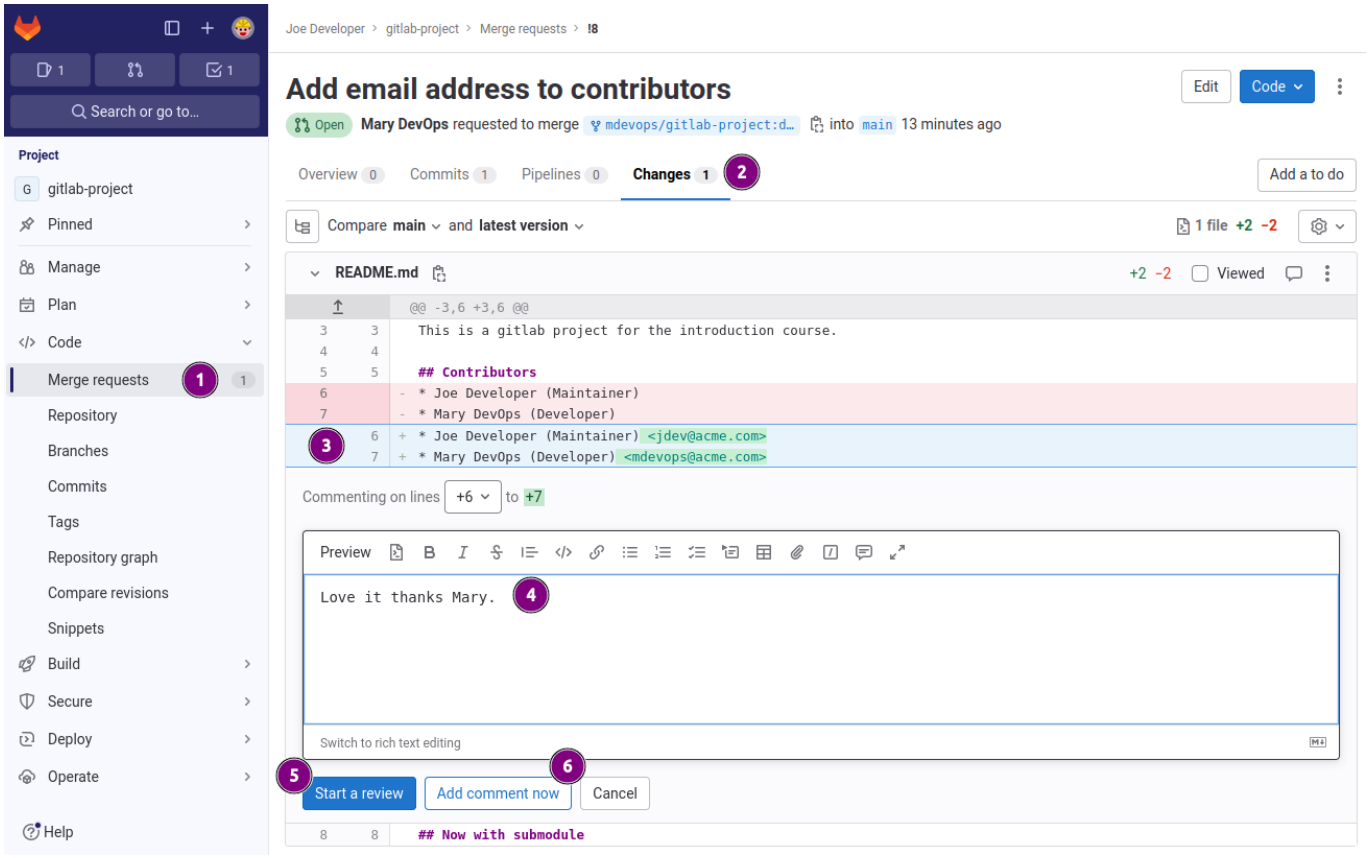

Figure 29. User Joe Developer adds a comment under changes.

1. Navigate to **`Code → Merge requests`**
2. Switched to tab [Changes].
3. Marking lines **`+6`** ~ **`+7`** as relevant to the comment.
4. Writing message about the previously marked lines.
5. If this change should be started as a review the [Start a review] button is clicked.
6. In this particular case it is a comment so [Add comment now] is clicked.

Once the comment is submitted the others can chime in. Since it is only a comment Mary DevOps is resolving the thread.

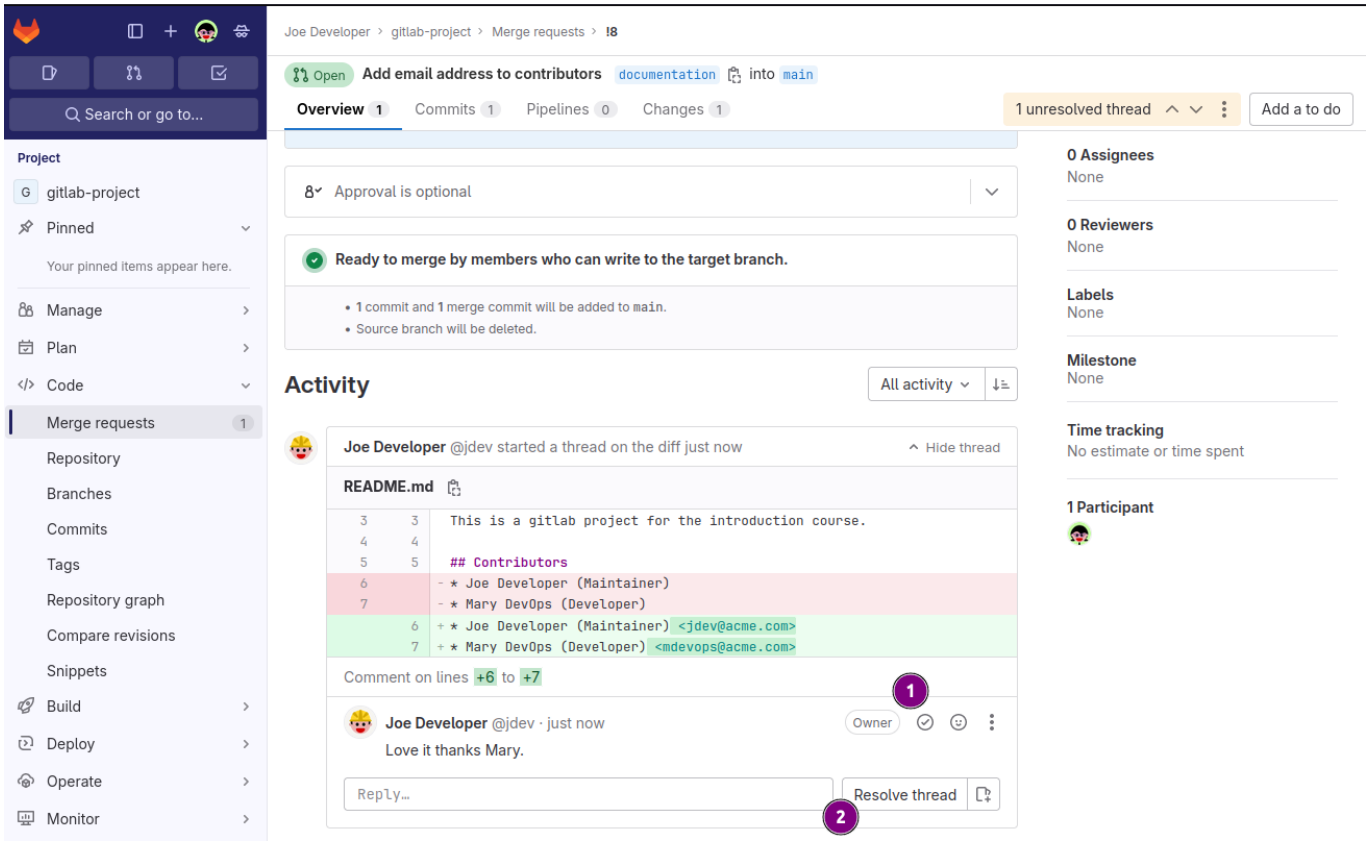

Figure 30. Mary DevOps views comment and resolves thread.

1 Resolving the thread via the circled check mark.

2 Alternatively via the button [Resolve thread].

Amend request

Maintainers have the option to make small amendments to the merge request. This can be done entirely within the merge request page.

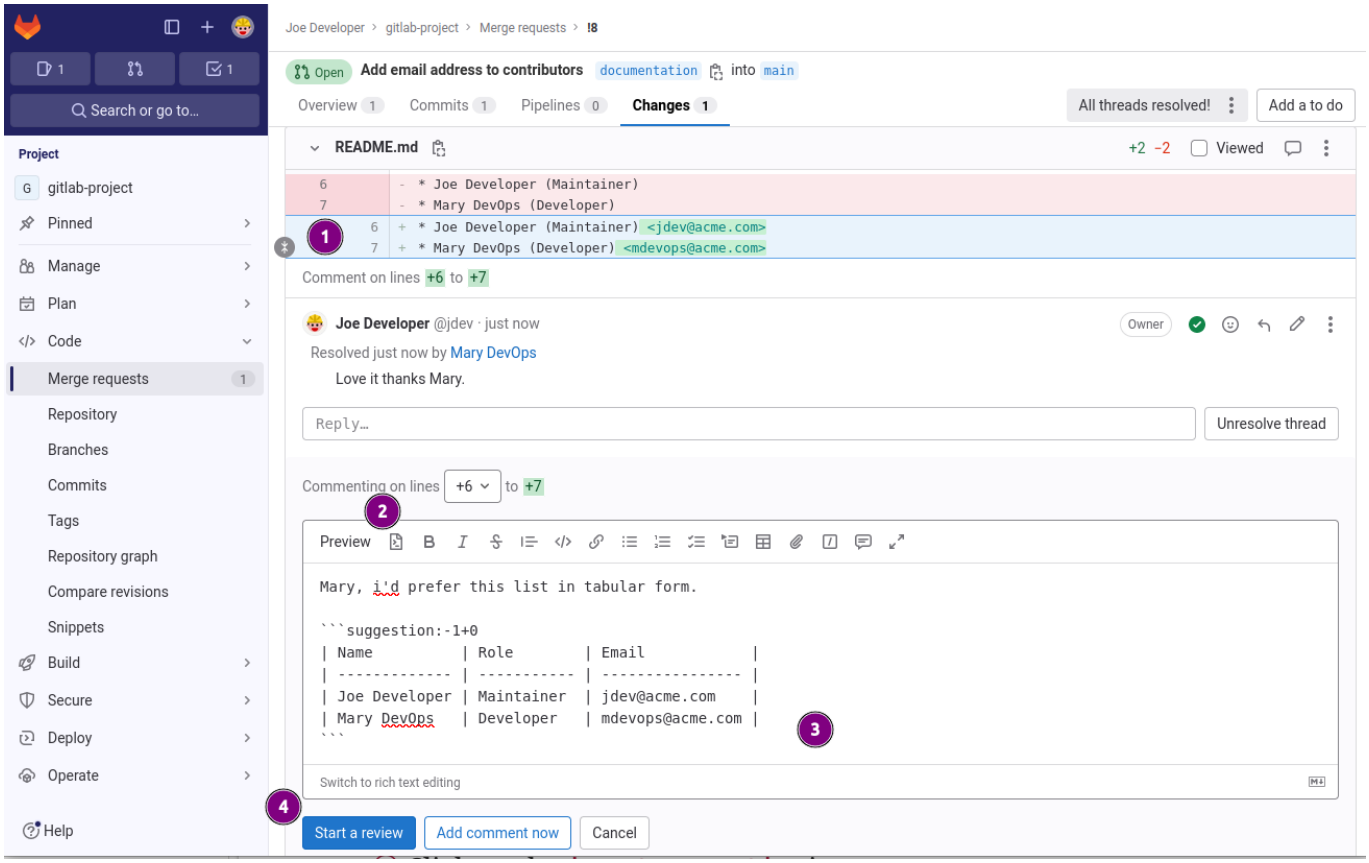

Figure 31. Create a suggestion

1. Select target lines.
2. Click on the **`insert suggestion`** icon.
3. Modify the selected code.
4. [Start a review] to submit the suggestion.

> Amendments are only available if the option for **`Allow commits from member who can manage the target branch`** has been checked during merge requestion submission.

Once the suggestion has been submitted other maintainers can further review the new amendment and comment.

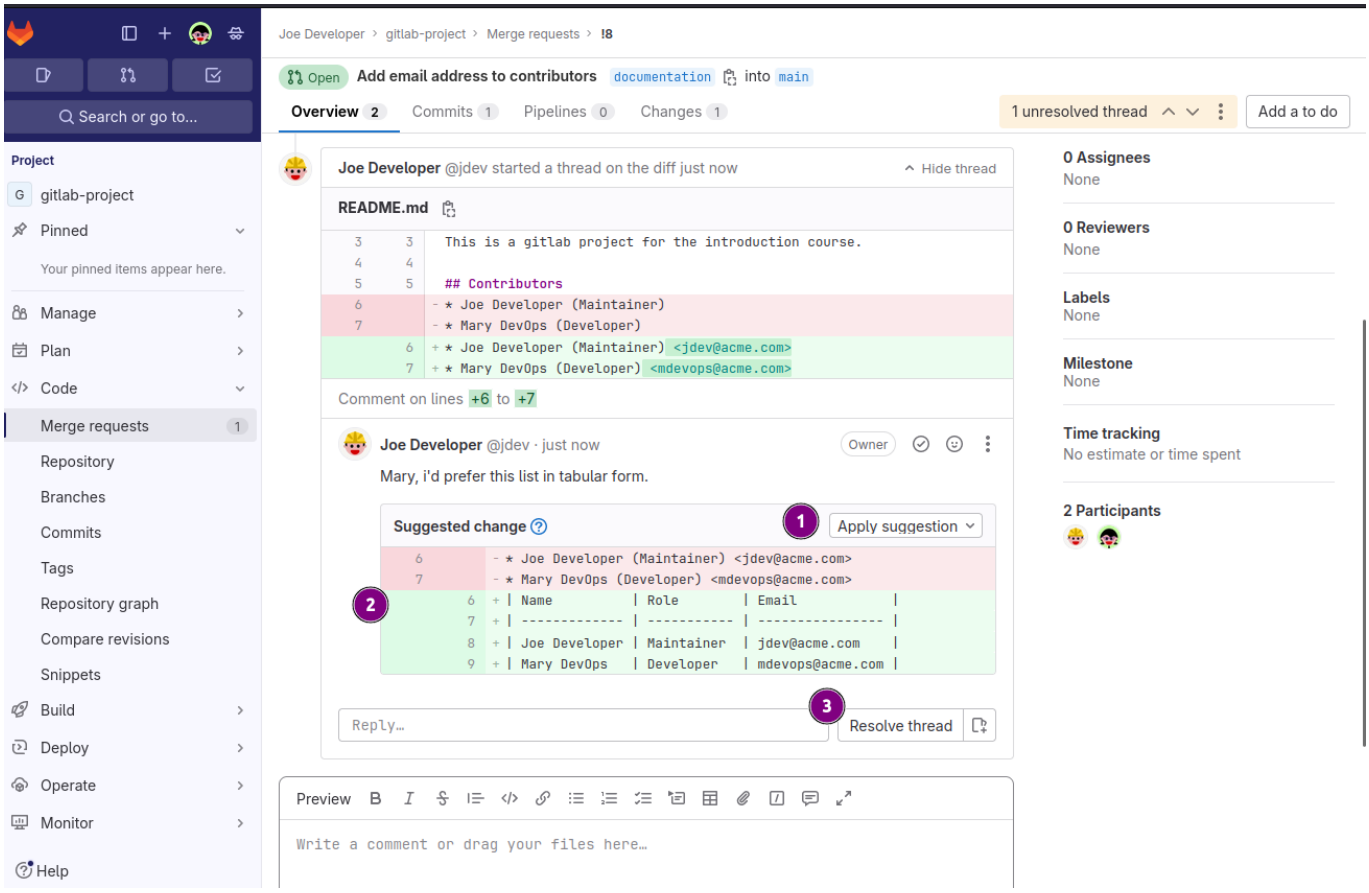

Figure 32. Suggestion ready for review or application.

1. Clicking on [Apply suggestion] will create a new commit to the merge request.
2. The diff of the suggestion is shown to make suggestion review easy.
3. Further comments can be appended to the suggestion.

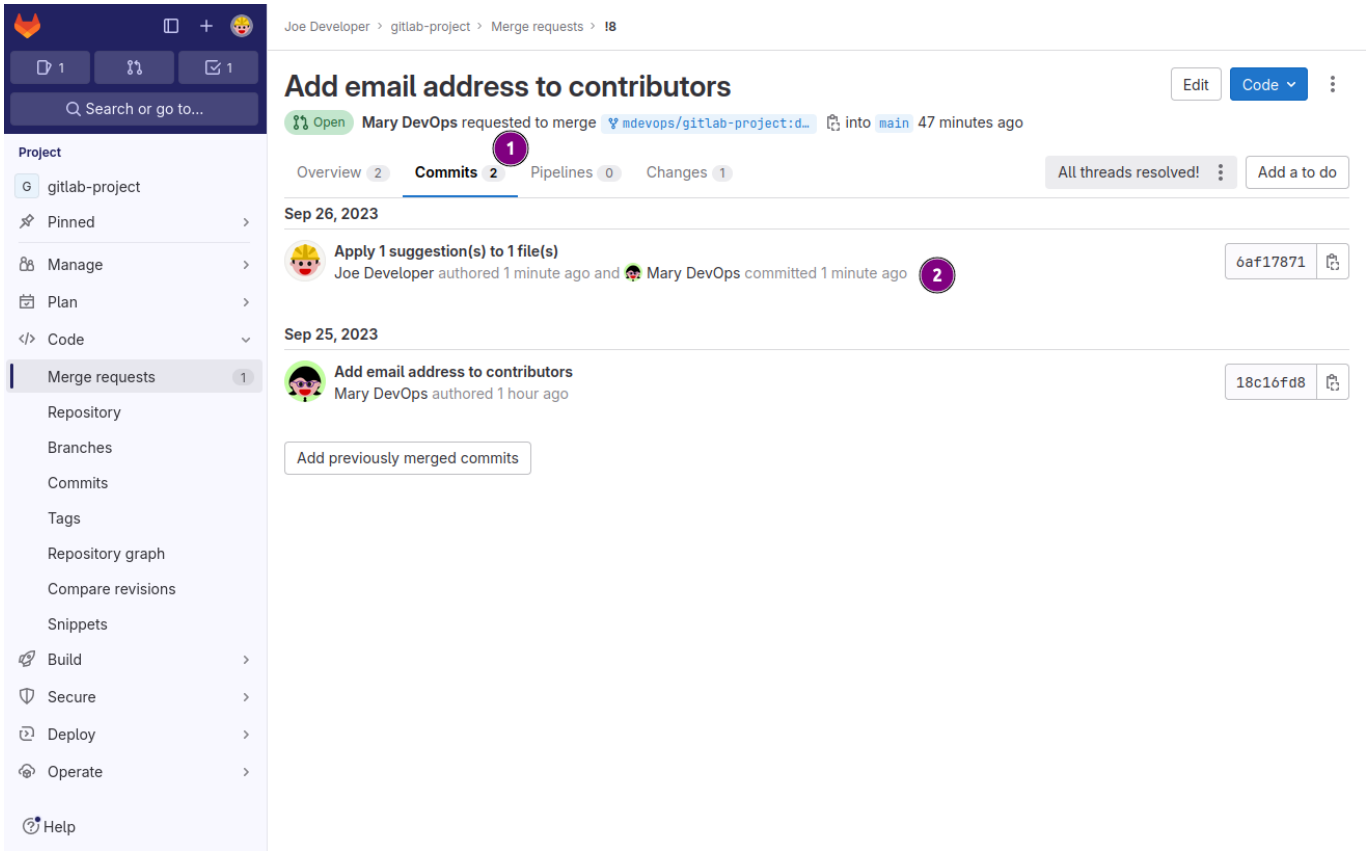

Figure 33. After suggestions have been applied two commits are present

1. The Commits tab shows now 2 commits in the indicator.
2. The addtional commit now listed.

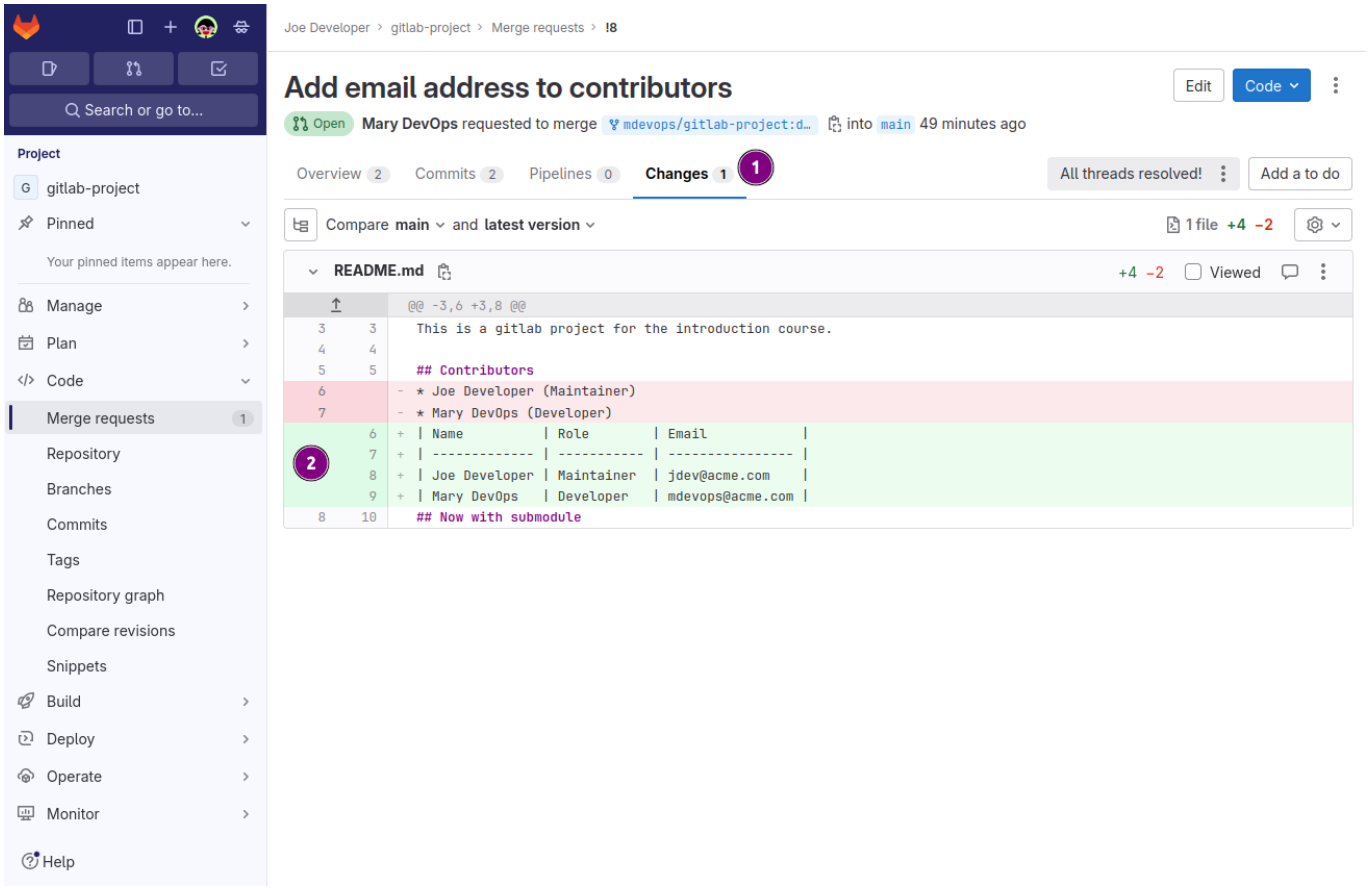

Figure 34. The suggestions is now part of the merge request

1. There is still only one changed file.
2. The markdown table has been applied.

Merge the request

Once the stakeholders are satisfied with the changes, suggestions and amendments the request can be merged by a maintainer.

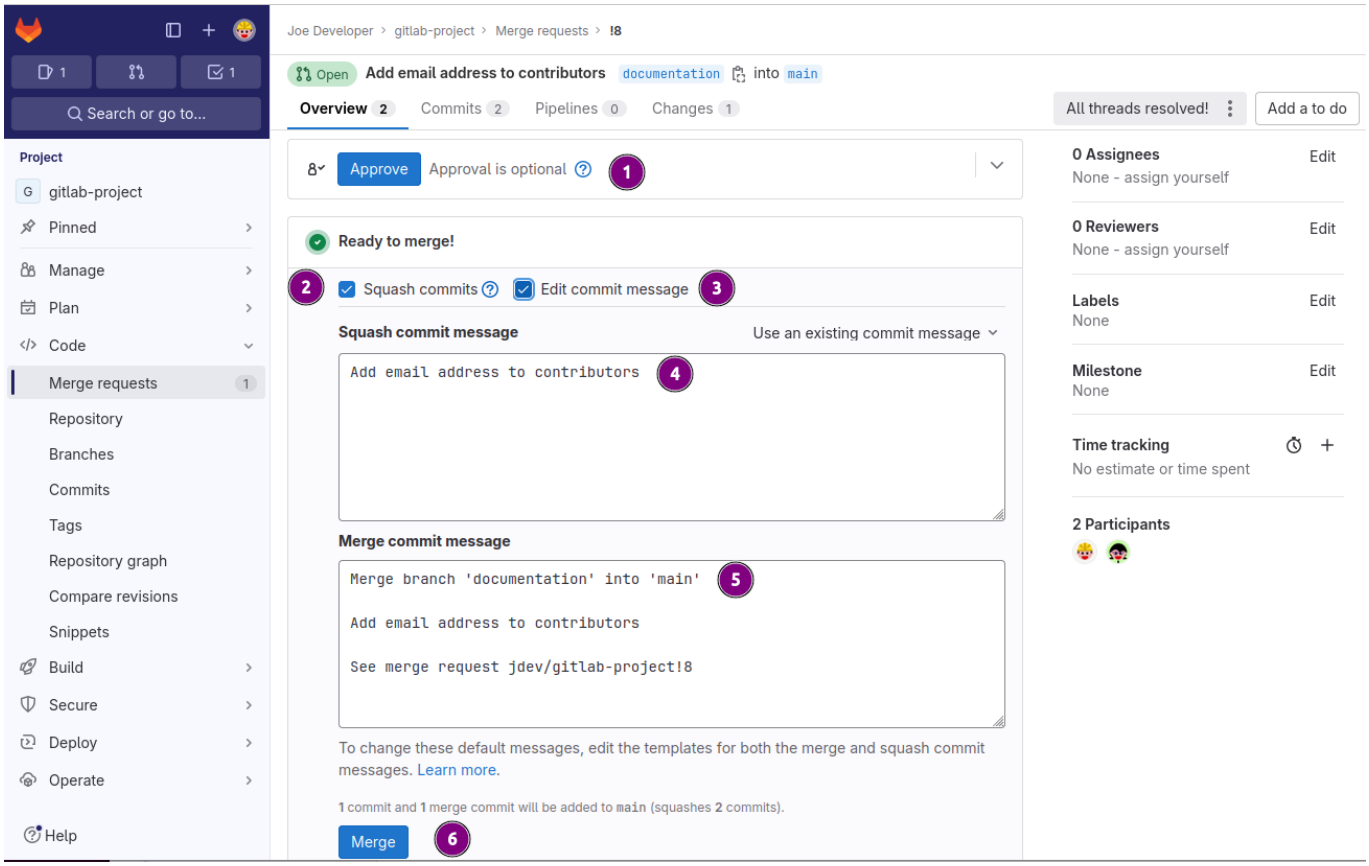

Figure 35. Merge the request

1. Approval of a merge request is not a requirement but can be a helpful tool to determine who has reviewed the change.
2. With the applied suggestion the **`Squash commits`** checkbox can be activated to only have one commit at merge time.
3. The commit message can be adjusted as well by checking **`Edit the commit message`**.
4. The textarea to adjust the commit message.
5. The textarea to change the merge messsage.
6. The [Merge] button to execute the merge.

Post merge review

Eventually the merge request was merged in this particular case with the **`Squash commits`** option checked.
It is now time to find out how the project repository changed.

Web UI post merge review

In the Web UI the changes are very subtle and the complexity of the process is masked.

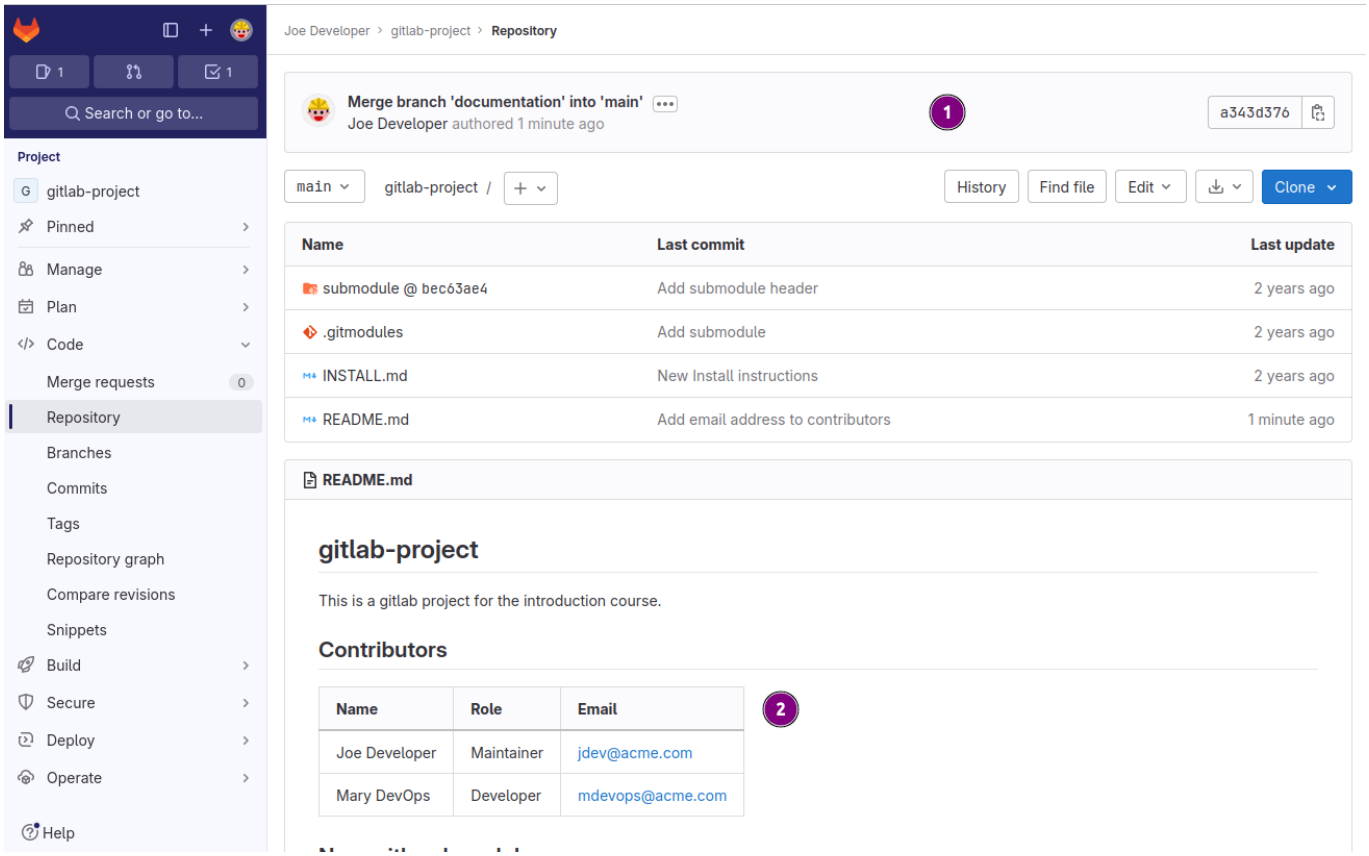

Figure 36. Project overview page post merge.

1. The commit SHA1 is different to the one from Mary DevOps when submitting the merge request.
2. The changes including the suggestion from Joe Developer have are displayed in the rendered **`README.md`**.

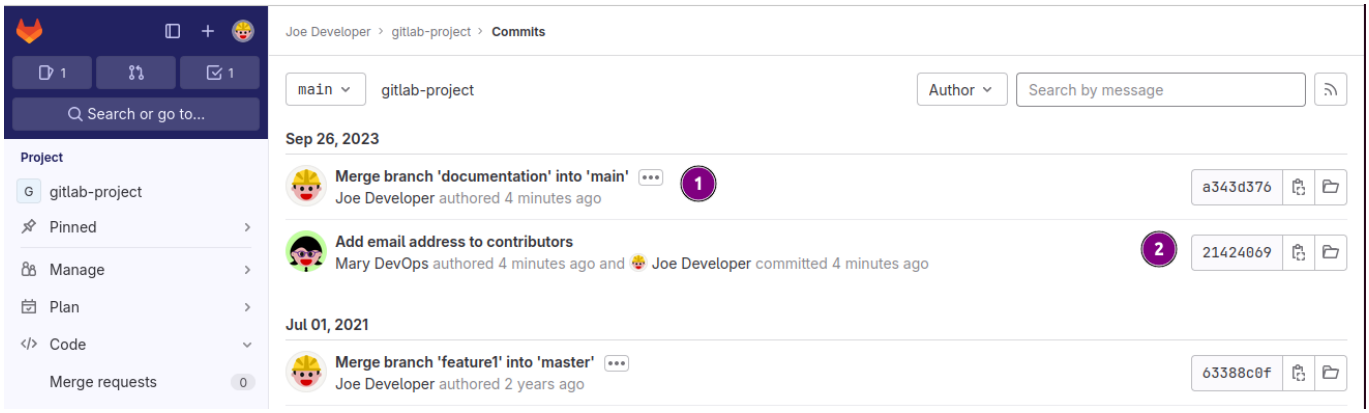

Figure 37. Review of the commits present.

1. A new commit with a commit message never entered during the process is now the **`HEAD`** of the repository.
2. The commit hash changed compared to the submitted one. What happened?

git command post merge review

It is sometime easier to review the changes that look confusing in the Web UI on the command line.

```
$ git pull origin main (1)
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 664 bytes | 664.00 KiB/s, done.
```

```
From https://gitlab.local:9480/jdev/gitlab-project
 * branch            main       -> FETCH_HEAD
   63388c0..a343d37  main       -> origin/main
Updating 63388c0..a343d37
Fast-forward
 README.md | 6 ++++❷
 1 file changed, 4 insertions(+), 2 deletions(-)



$ git log ❸
commit a343d37672c6c73efea85a7100340057ca7a4858 (HEAD -> main, origin/main,
origin/HEAD)
Merge: 63388c0 2142406 ❹
Author: Joe Developer <joe.developer@gitlab.local>
Date:   Tue Sep 26 00:40:53 2023 +0000

    Merge branch 'documentation' into 'main'

    Add email address to contributors

    See merge request jdev/gitlab-project!8

commit 214240696453302969ee67e364fa206d5dda5e65
Author: Mary DevOps <mary.devops@gitlab.local>
Date:   Tue Sep 26 00:40:53 2023 +0000

    Add email address to contributors

commit 63388c0fd88b803af36e8c158139edde98c05830
Merge: 2c34e69 c7b7c33
Author: Joe Developer <joe.developer@gitlab.local>
Date:   Thu Jul 1 09:55:50 2021 +0000

    Merge branch 'feature1' into 'master'

    New Install instructions 123

    See merge request jdev/gitlab-project!7

[.notes]
--
<1> Pull the changes from the GitLab project to the previously
    created repository.
<2> The file `README.md` is updated during this procedure.
<3> The `git log` command displays a bit more information about
    commit `5717883`. This is a merge commit with multiple parents.
<4> The merge line display all the parents this particular commit has.
    Merge commits can be prevented in under `Settings` -> `General` ->
    `Merge requests` -> (*) Fast-forward merge.
--

// vim: set colorcolumn=80 textwidth=80 spell spelllang=en_us :

// vim: set colorcolumn=80 textwidth=80 spell spelllang=en_us :

<<<
```