

[Docs](#) » [User guide](#) » [Source Code Management](#) » [Code review](#) » Pull requests

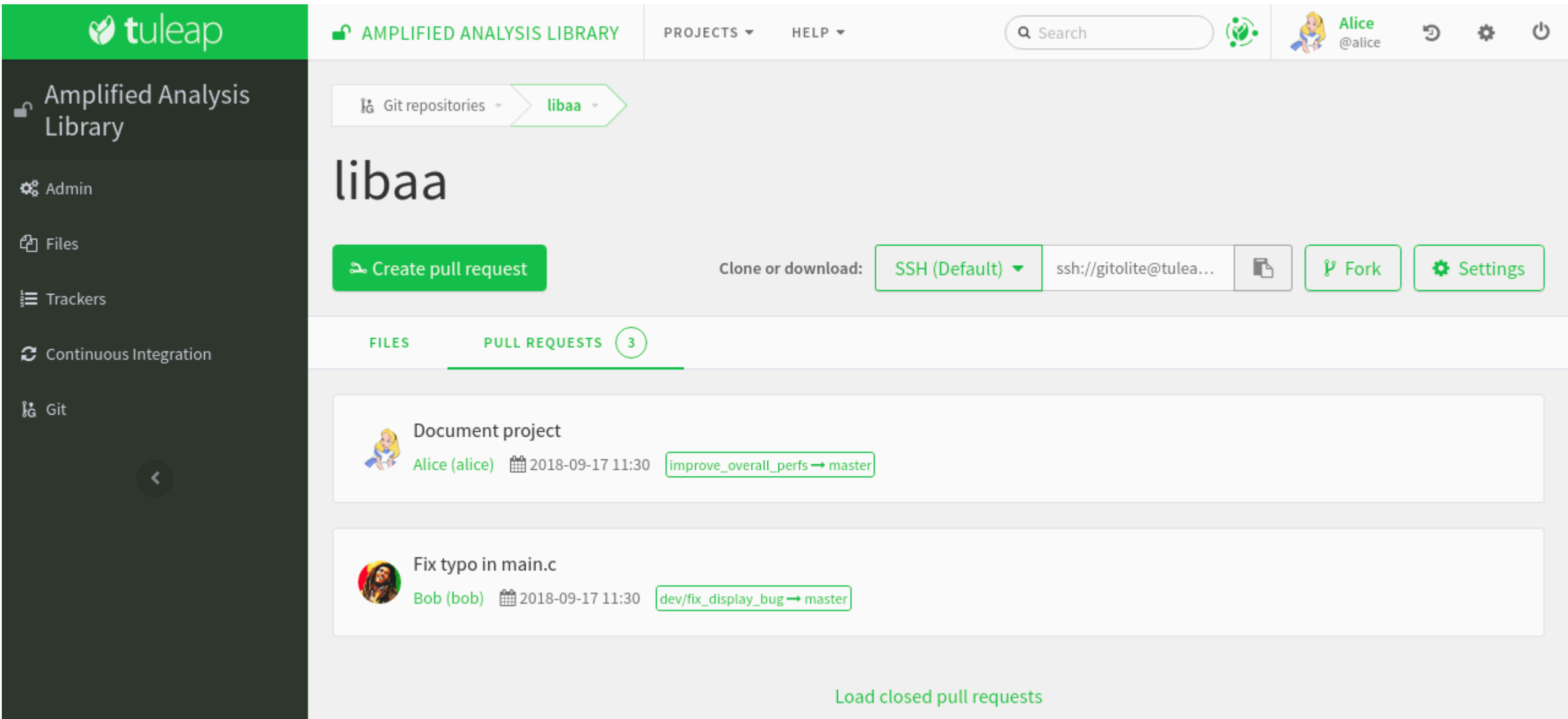
Pull requests

Tuleap pull requests (aka PR) are built on top of Git. They provide an easy way to do code review and integration workflow.

Tuleap also supports code review with [Gerrit](#).

Features:

- Create requests across branches in the same repository
- Create requests from a [personal fork](#)
- Comment in files reviewed
- Comment requests globally
- (cross)-reference requests from any point of Tuleap
- Integrate with Jenkins to know if tests passed on the code to integrate
- Dedicated dashboard to follow-up pull requests status



Ways of working

There is not a single way to use pullrequests. The way you will use it depends on the size of your team, the knowledge team members have with git and the workflow you are already used to.

In this documentation we will present two possible workflows that will allow to demonstrate all supported features. Keep in mind that you can define your own.

SIMPLE WORKFLOW

Search docs

HOW-TO GUIDES

USER GUIDE

- Introduction
- User
- Writing in Tuleap
- Project Management
- Backlog
- Kanban
- Program management
- Baseline
- Trackers
- Source Code Management
- Code versioning
- Code review
- Gerrit
- Pull requests
- Continuous Integration
- Test Management
- Documents and files
- Integration
- OAuth2 and OpenIDConnect
- Tuleap Query Language (TQL)
- Miscellaneous
- Legal Notice

INSTALLATION GUIDE

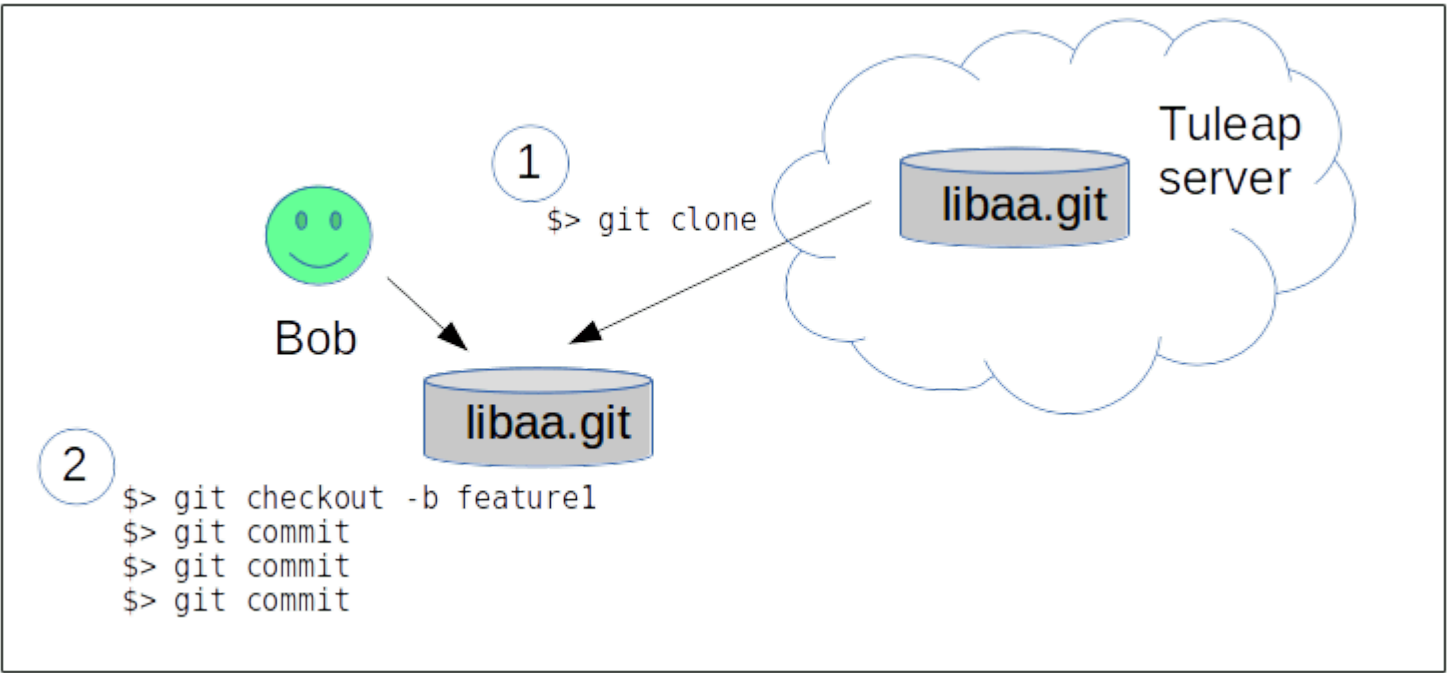
ADMINISTRATION GUIDE

DEPLOYMENT GUIDE

DEVELOPER GUIDE

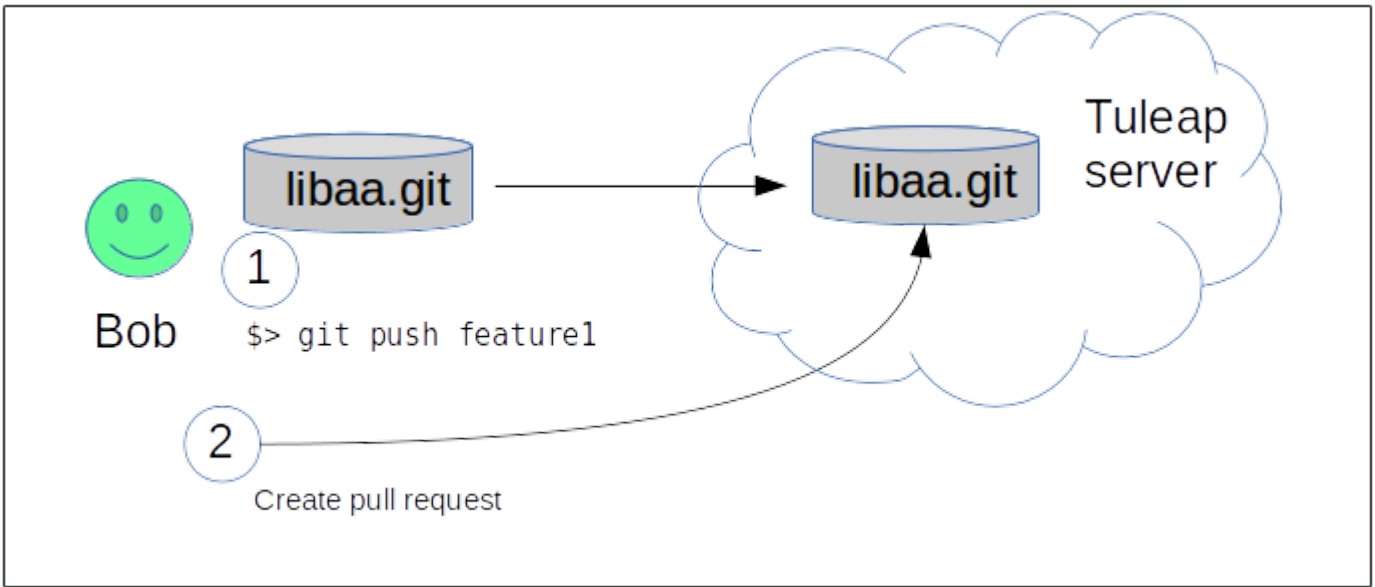
produced by Bob.

Bob has a local working copy of libaa and made a new contribution “feature 1”. He thinks the feature is ready to be integrated inside the public repository.



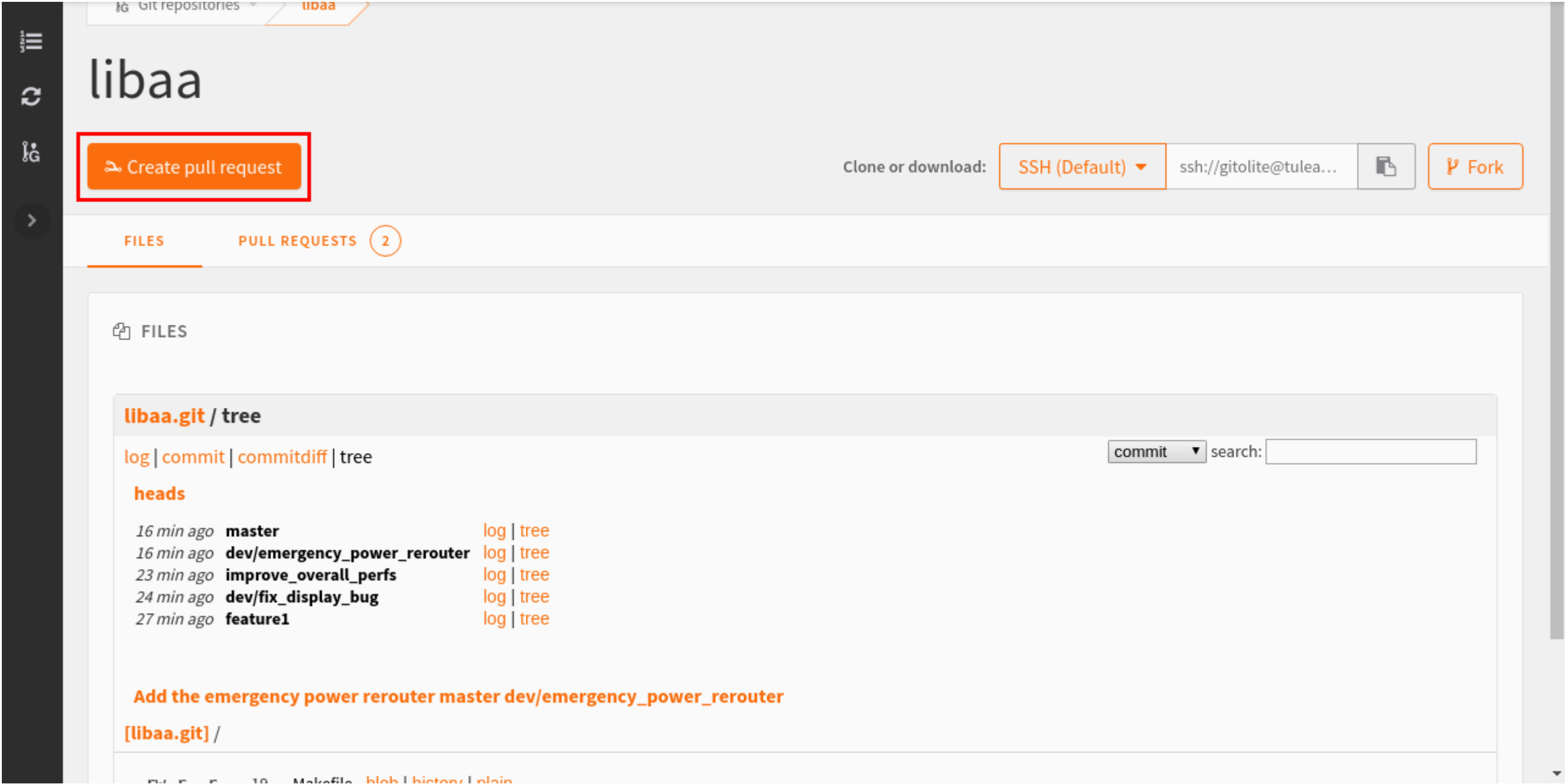
CREATE A PULL REQUEST

Bob needs to push his development to the Tuleap server and then generate a pull request.p

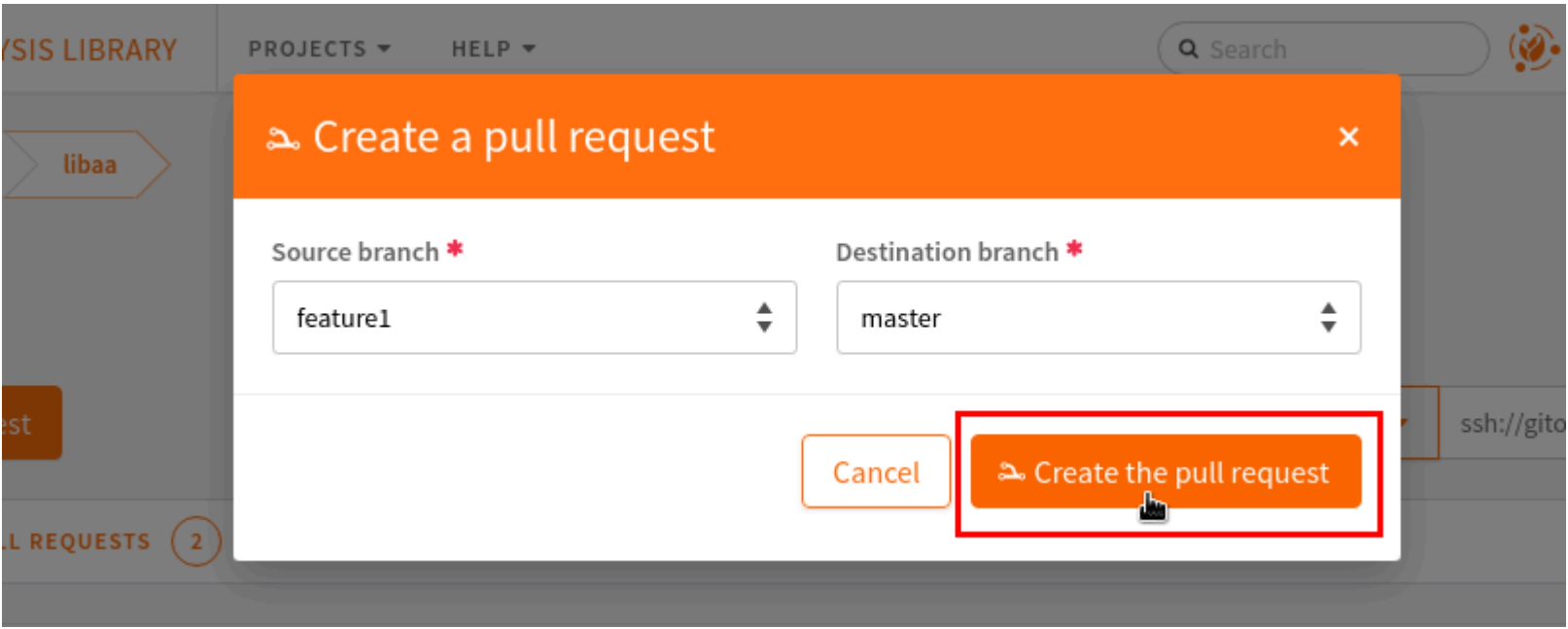


Once the code is on the server, Bob goes to the Tuleap web interface, in the repository (git service).

From there he can create a pull request by selecting the source and target branches.



Source branch is where the work was done, target is where it should be integrated.



Bob is redirected on the pull request screen where he can quickly see the major information about his work.

The pull request's summary is automatically extracted from the first line of the first commit message in the branch. The description is the rest of the commit message.

This information can be edited directly from the web browser.



Create pull request

Clone or download: SSH (Default) ssh://gitolite@tulea... Fork

FILES **PULL REQUESTS** 3

Add makefile target

OVERVIEW CHANGES

Edit message Checkout

Bob (bob)

2018-09-17 11:50

Metadata

+26 -0

How to test:

- run "make main"
- The program should be built

Commit message (editable)

a8878d79dbab4971ccdbd2686b8cfe63ca889d70

feature1 → master

Last CI Status

Unknown

Merge Abandon

Discussion has not started yet

Say something...

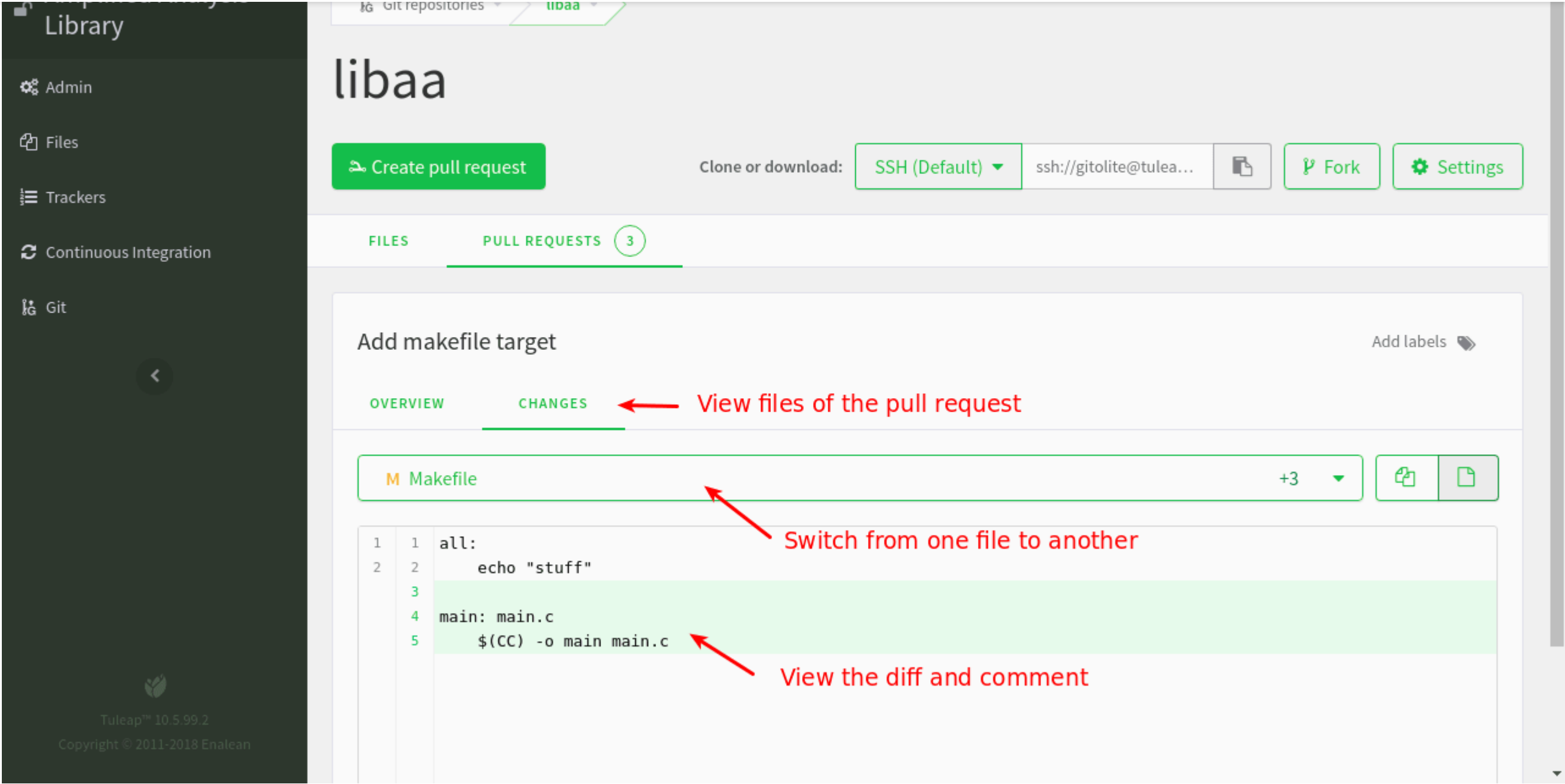
Comment

REVIEW THE CODE

Once Bob has created the pull request, Alice can review it. The majority of the work is available in the “Files” tab.

https://docs.tuleap.org/user-guide/code-review/pullrequest.html

4/11



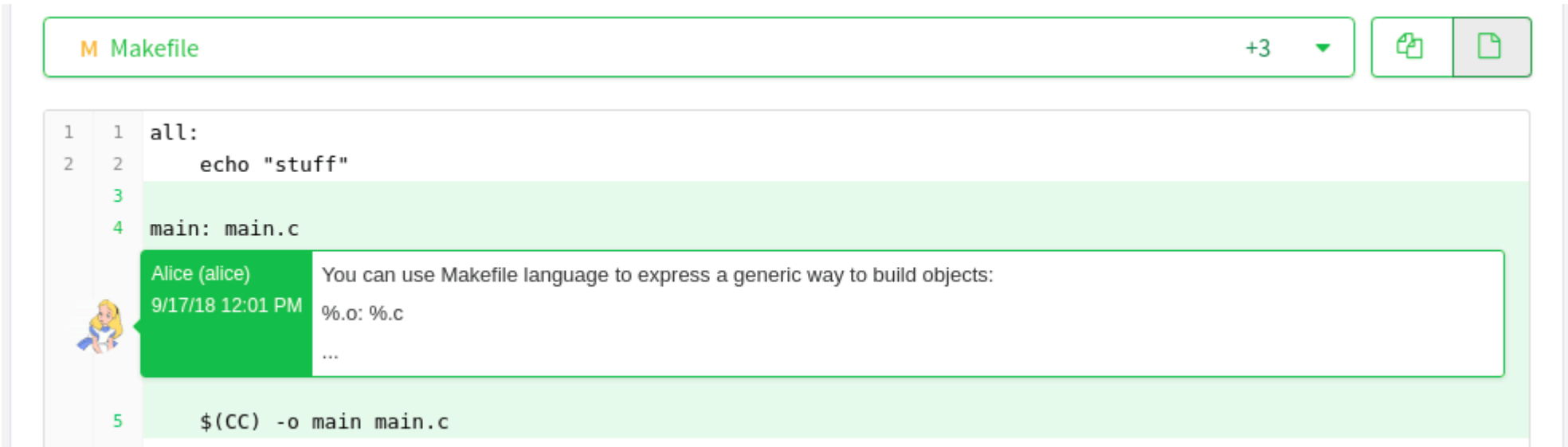
Attention

The diff is generated between the latest commit and the base of the branch (common ancestor).

If master evolved in the mean time, you won't see those changes (and potential conflicts) in this view.

To have an accurate review, you should rebase your work locally with the tip of the branch you want to push in before creating the pull request.

Alice can then comment lines of the diff by clicking on the line number.



UPDATE A PULL REQUEST

Bob can see all of Alice's comments from the discussion view.



OVERVIEW

CHANGES

Edit message

Checkout

Bob (bob)

2018-09-17 11:50

+26 -0

How to test:

- run "make main"

- The program should be built

a8878d79dbab4971ccdbd2686b8cfe63ca889d70

Last CI Status

Unknown

Merge

Abandon

Add labels

Alice (alice) 5 minutes ago

Makefile

You can use Makefile language to express a generic way to build objects:

%o: %.c

...

Alice (alice) a minute ago

main.c

You should put some content here

Alice (alice) a few seconds ago

This is not good to go for me, please have a look at my comments.

Say something...

Comment

Then he should go back to work and update the branch. When he pushes his work again, the pull request will be automatically updated.

AMPLIFIED ANALYSIS LIBRARY

PROJECTS

HELP

Search

Alice @alice

Amplified Analysis Library

Admin

Files

Trackers

Continuous Integration

Git

Add makefile target

Add labels

OVERVIEW

CHANGES

Edit message

Checkout

Bob (bob)

2018-09-17 11:50

+30 -2

How to test:

- run "make main"

- The program should be built

a11e0a57218d3d7a70cdcefb83834c33a28bbd1

Last CI Status

Unknown

Already merged

Alice (alice) 2 hours ago

Makefile

You can use Makefile language to express a generic way to build objects:

%o: %.c

...

Alice (alice) 2 hours ago

main.c

You should put some content here

Alice (alice) 2 hours ago

This is not good to go for me, please have a look at my comments.

Bob (bob) 4 minutes ago

Has updated the pull request.

This is a known limitation when lines change from one diff to another. The comment was placed on Makefile L4 at the first review but in the second review L4 was changed (the first 2 lines of the Makefile were removed).

Add makefile target

OVERVIEW

CHANGES

M

 Makefile

1

2

1

2

3

4

5

6

7

all:

echo "stuff"

all: main

%.o: %.c

\$(CC) -c \$<

main: main.o

\$(CC) -o \$<

MERGE THE REQUEST

The work is now done, Alice can click on the “Merge” button and the code will be integrated inside master.

tuleap

AMPLIFIED ANALYSIS LIBRARY

PROJECTS ▾ HELP ▾

Amplified Analysis Library

Admin

Files

Trackers

Continuous Integration

Git

Tuleap™ 10.5.99.2
Copyright © 2011-2018 Enalean

Edit message Checkout ▾

Bob (bob) 2018-09-17 11:50

+30 -2

How to test:

- run "make main"

- The program should be built

a11e0a57218d3d7a70cdcefb83834c33a28bbd1

feature1 → master

Last CI Status

Unknown

Already merged

Alice (alice) 2 hours ago

main.c

You should put some content here

Alice (alice) 2 hours ago

This is not good to go for me, please have a look at my comments.

Bob (bob) a minute ago

Has updated the pull request.

Alice (alice) a few seconds ago

LGTM

Alice (alice) a few seconds ago

Has merged the pull request.

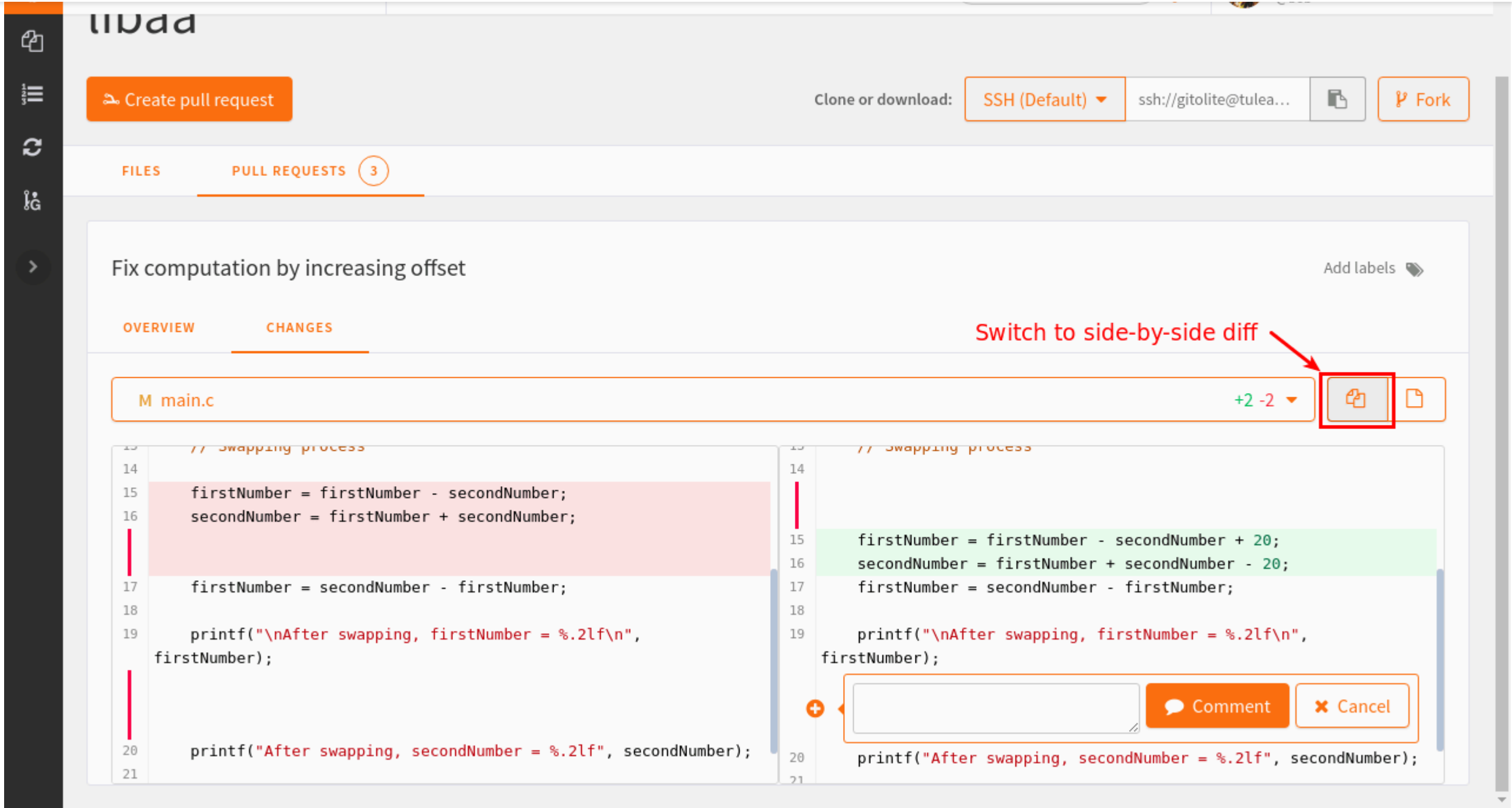
Comment

Alice can also merge “by hand” in her own working copy and then push to the repository, the end result will be the same.

SEE SIDE-BY-SIDE DIFF

https://docs.tuleap.org/user-guide/code-review/pullrequest.html

7/11



ADVANCED WORKFLOWS

In the previous example, we followed a basic “feature branch” model where only basic git features are involved. It’s the easiest way to start with code reviews because it basically change nothing to developers workflow (they create branches, commit within and when the work is ready, merge in master).

For developers with more git skills there are two popular practices:

- Rebase
- Rebase and squash

Those two practices happen at the end of the review cycle, when the branch is “ready to go”.

REBASE

As already said, the diff under review in a PR is a difference of the branch itself. It doesn’t reflect the changes that were done on the target branch (typically master).

So when a feature is ready, integrator might ask for a rebase. Developer would then run on its working copy:

```
$> git fetch origin
$> git checkout dev/feature1
$> git rebase origin/master
$> git push -f origin dev/feature1
```

For this to work, developer must be granted the “rewind” permission on the given branch.

Attention

Be very careful with “rewind” permission. People granted to rewind can completely erase the repository if they want to.

If you want to generalize the rebase pattern we strongly suggest that you either:



REBASE & SQUASH

As you might expect, rebase and squash is a variation of the previous one. In addition to rebasing your work with the target, squash means that you will rewrite the branch history to only keep what is relevant as history steps.

Imagine your branch `git log` after Alice review:

- 2c74d67ae fix after review 1
- c8658adbc fix after alice comment
- 676b89ac3 typo
- 9792c7bed request #2314: fix OutOfMemory exception in core

Most of this history doesn't really make sense and will polluate master. You can group all those commit into one:

```
$> git fetch origin
$> git checkout dev/feature1
$> git rebase -i origin/master
```

At this step, your favorite text editor will pop-up and present a “menu” of changes:

```
pick 9792c7bed request #2314: fix OutOfMemory exception in core
pick 676b89ac3 typo
pick c8658adbc fix after alice comment
pick 2c74d67ae fix after review 1

# Rebase 9792c7bed..2c74d67ae onto 274b801 (4 command(s))
#...
```

You can dig into `git help rebase` menu to understand all the possible commands but if you want to only have one commit that groups the 4 changes, you'll need to update and save the file like:

```
pick 9792c7bed request #2314: fix OutOfMemory exception in core
fixup 676b89ac3 typo
fixup c8658adbc fix after alice comment
fixup 2c74d67ae fix after review 1

# Rebase 9792c7bed..2c74d67ae onto 274b801 (4 command(s))
#...
```

After save, the rebase will be applied (you might have to solve some conflicts) and then, if you issue `git log` again you will see only one commit:

- 2de53ac74 request #2314: fix OutOfMemory exception in core

And you can push the result to the branch:

```
$> git push -f origin dev/feature1
```

Note

If you are using pull requests in a repository relying on fine grained permissions and the Git plugin based on `tuleap-plugin-git` you will not have access to the merge and abandon buttons in the web UI. To make it works, you will need to migrate to Gitolite3 (`tuleap-plugin-git-gitolite3`). You can find more information on how to do that in the administration guide.

Reference pull requests

One of the key feature of Tuleap is to be able to reference anything from anywhere and having a back reference automatically created on the other end. Pull requests make no exceptions and follow this pattern.

From the PR, either in description, in global comments or directly within the diff you can reference any Tuleap element (artifact, document, file release, ...). The example below is a reference to an artifact in PR description:

Add makefile target

This is an attempt to fix [bug #1](#)

feature1 → master

c4b43c7a4ac4b66a5cf5aa7

Artifact ID: 1

Summary: Issue when filling the input with 0x00 char

Status: New



Note: in this example, the bug n°1, automatically got a link back to pull request:

References

Cross references

List of items referenced by or referencing this item.

Git Pull Request

 [pullrequest #11](#) 

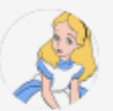
Title: Add makefile target

Status: Merged

Created on: 2016-08-11 15:57

Permissions

From any element in Tuleap, you can reference the pull request using `pr` or `pullrequest` keywords:

 [Alice \(alice\)](#) - less than a minute ago

See the code pushed in [pr #1](#)

Permissions on artifact set

Title: Base

Status: Abandoned

Created on: 2016-07-19 15:25

Mail notifications

A mail notification is sent to the reviewers of a pull request, the creator of the pull request and to the users having updated the pull request when one of the following actions happen:

- the pull request is updated (new commit are pushed into the pull request branch)
- the pull request is merged
- the pull request is abandoned
- a new global comment is added
- a new inline comment is added

User being added to list of reviewers of a pull request also receive a notification.

Notifications are not sent to the user doing the action, i.e. a user will not receive a mail for a comment she has posted.

Notification by @ mention

While writing comments or editing the pull request’s description, you can mention somebody by typing `@` and their Tuleap username. For example: `@admin`. After typing three or four letters, an auto-completer appears and suggests users matching what was written after the `@`. When you submit the comment, each mentioned user will receive an e-mail notification, unless they do not have permission to see the pull request.

This is a one-time notification, mentioned users will not receive updates for new comments or changes of the pull request.

Integrate with Jenkins



The integration is a two step process:

- first you need to configure your repository to trigger builds on Jenkins whenever there is a commit in your repository
- then, in the Jenkins job definition, you must add an extra step to feed tuleap back with job status (success or failure).

CONFIGURE TULEAP TO JENKINS TRIGGER

You need to configure Jenkins webhook as described in the [git documentation section](#).

Note

If you are using pullrequests across repositories, you must ensure that the CI job is properly configured to use the target repository.

To trigger a new build each time a pull request is created or updated, set the name to `origin`, the refspec to `+refs/tlpr/*:refs/remotes/origin/pr/*` and the branch specifier to `**`.

SCM

Git

Repositories

Repository URL

ssh://gitolite@tuleap.example.com/projectname/repowithpr.git

Credentials

gitolite (SSH key for aci_agent on tuleap.example.com)

Ajoute

Name

origin

Refspec

+refs/tlpr/*:refs/remotes/origin/pr/*

Add Repository

Branches to build

Branch Specifier (blank for 'any')

**

Add Branch

CONFIGURE JENKINS TO TULEAP FEEDBACK

The Jenkins to Tuleap feedback is possible thanks to [Tuleap API](#) plugin which can be installed via the Jenkins plugins manager. The configuration of this plugin is [here](#)

Previous

Next



Tuleap project and this website are sponsored by Enalean

© 2024 Enalean SAS



Trademarks

Security

Legal