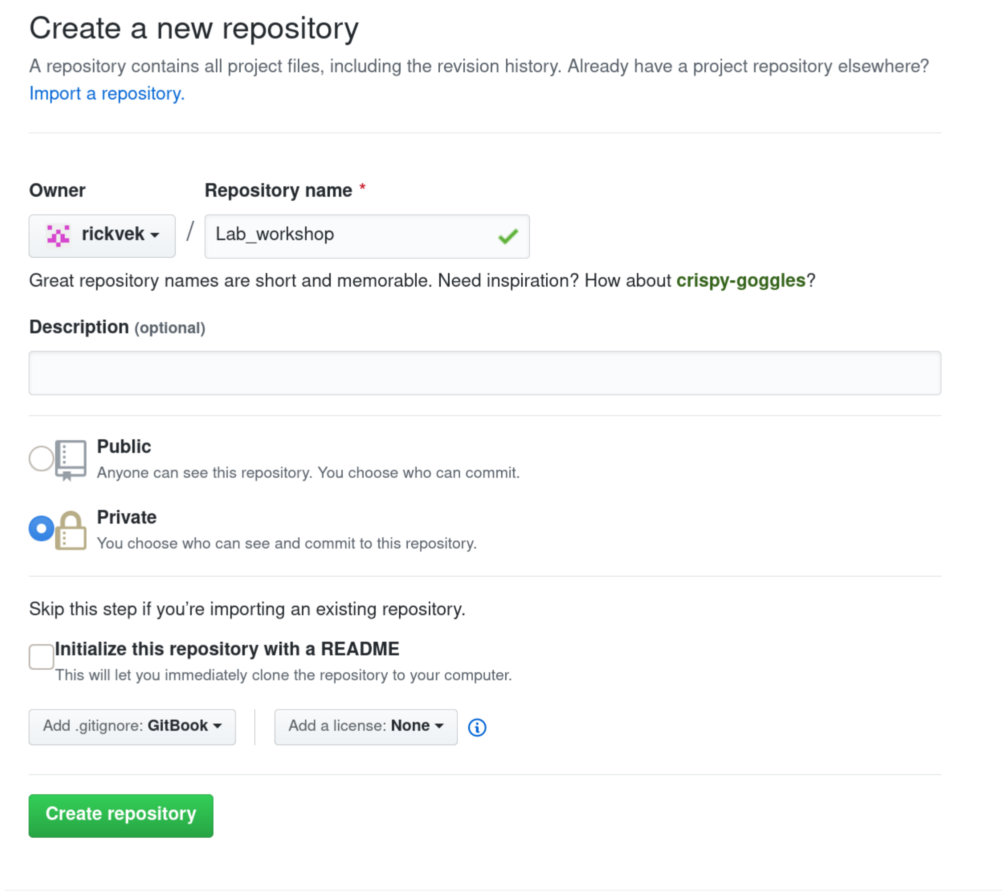
Lab belonging to GIT workshop

Start with downloading the documentation from github :

```
git clone https://github.com/rickvek/Workshop-GIT.git
```

Exercise :

Go to https://github.com/ and create an account, if you do not have this already. Account can be deleted afterwards, so even temporarily it will do.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: rickvek / Repository name *: Lab_workshop

Great repository names are short and memorable. Need inspiration? How about **crispy-goggles**?

Description (optional)

○ **Public** Anyone can see this repository. You choose who can commit.

● **Private** You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

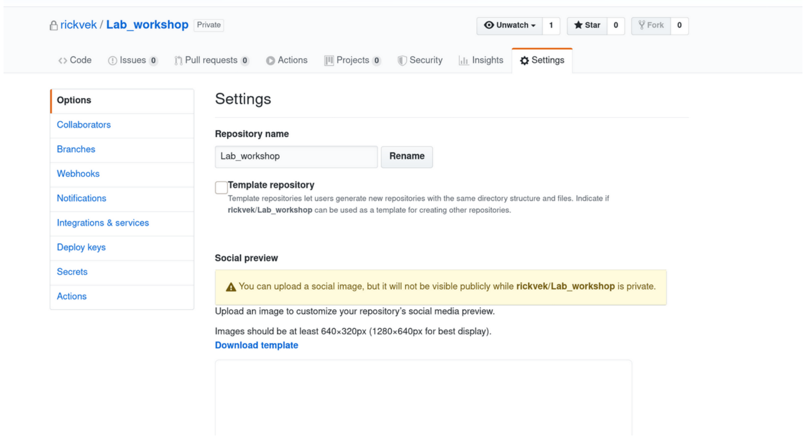
☐ **Initialize this repository with a README** This will let you immediately clone the repository to your computer.

Add .gitignore: **GitBook** | Add a license: **None**

Options chosen :

.gitignore, privat, Repository Name
 !note do not select .gitignore if the repository initially be filled with your local repository.

Setting up the Repository :



Settings

Repository name: Lab_workshop

Template repository
Template repositories let users generate new repositories with the same directory structure and files. Indicate if **rickvek/Lab_workshop** can be used as a template for creating other repositories.

Social preview

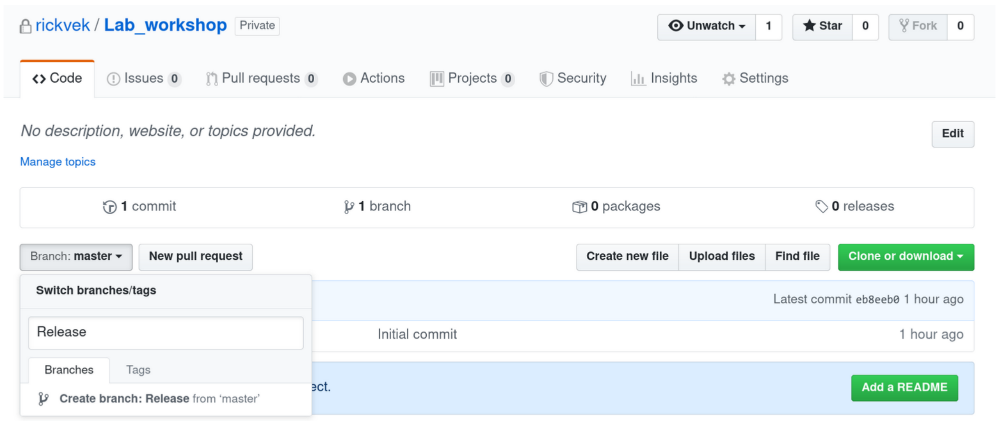
You can upload a social image, but it will not be visible publicly while **rickvek/Lab_workshop** is private.

Upload an image to customize your repository's social media preview.

Images should be at least 640×320px (1280×640px for best display).

Under branches the master can be locked (paid version). This is what usually is done for the master. Prevents accidental overwriting releases with development files. For this exercise we pretend this is done.
Section Notifications and Actions are also worth while to look into.

Now there are two options, we continue to set it up and the clone this to the workstation. Or we can start working on the workstation and push everything to the remote repository. For the second option continue with the 'create local repository' section below. (your branch needs to be empty, so no .ignore file selection)



No description, website, or topics provided.

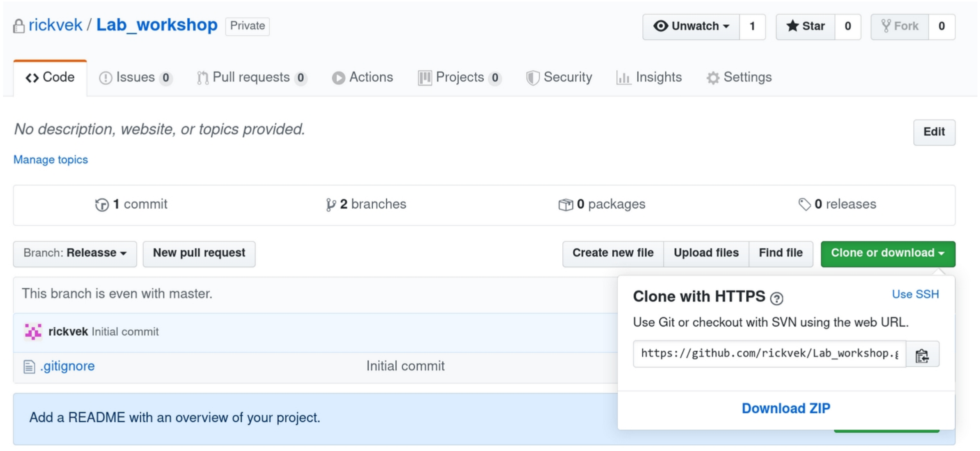
Release

Create branch: Release from 'master'

Latest commit eb8eeb0 1 hour ago

Initial commit 1 hour ago

We create our Release branch, we will be pushing the development branch which will be merged here with Release, then we can merge with master.



No description, website, or topics provided.

This branch is even with master.

rickvek Initial commit

.gitignore Initial commit

Add a README with an overview of your project.

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

```
https://github.com/rickvek/Lab_workshop.g
```

Clone repository

We copy the string from the remote repository and look for a suitable location on the local disk. Following steps will be done from the command line.

```
[Workshop-GIT]$ pwd
/home/rickvek/temp/lab/Workshop-GIT
[Workshop-GIT]$ cd ..
[lab]$ git clone git@github.com:rickvek/Lab_workshop.git
Cloning into 'Lab_workshop'...
Tunnel device open failed.
Could not request tunnel forwarding.
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.
```

We wil be asked for password as we have not set up a ssh key, in the example above the account already had a ssh key configured.

Setting up our development branch and project branches. Jump to section ‘Setup branches’

Create local repository

Now we create a local repository which we later will push to the remote repository. Which will give us a starting point.

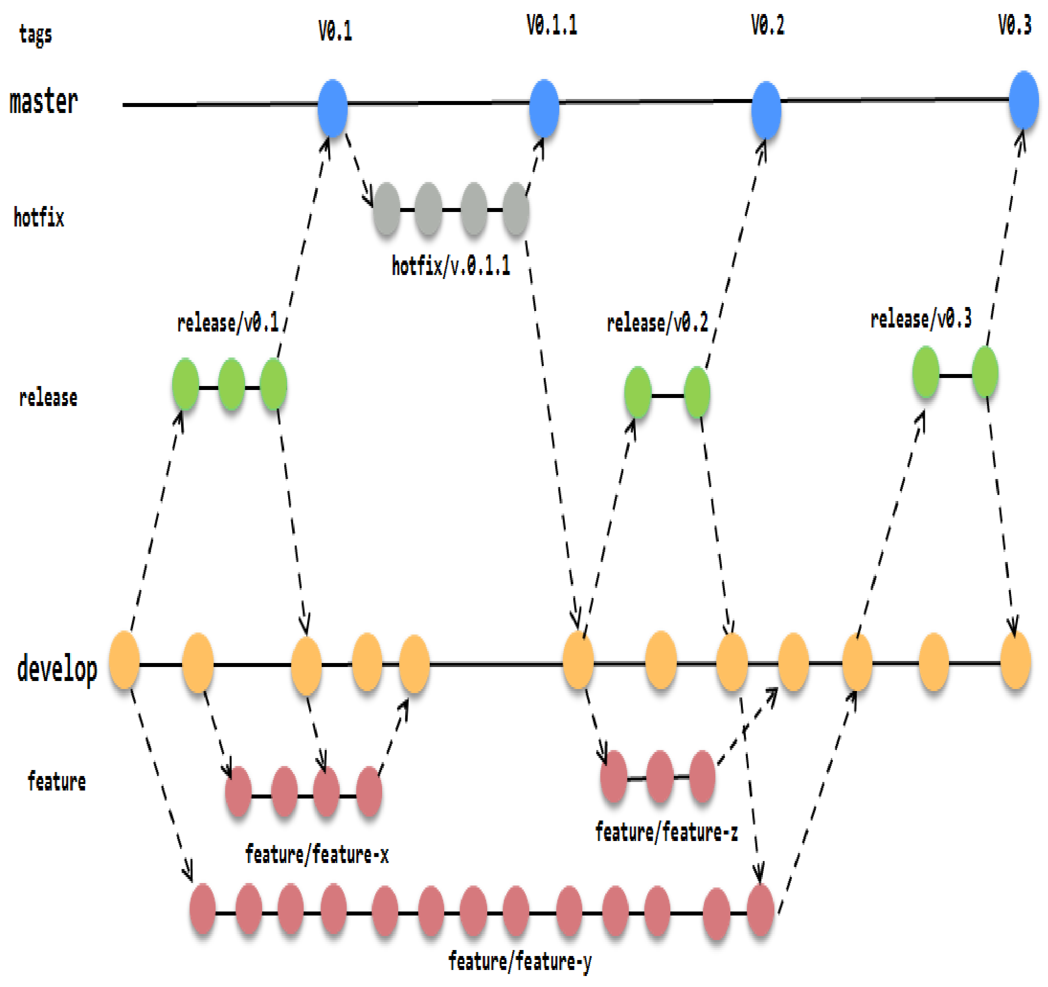
TBD

Setting up branches.

We now setting up the branches we are going to work with.

```
[lab]$ cd Lab_workshop/
[Lab_workshop]$ git checkout -b Release
Switched to a new branch 'Release'
[Lab_workshop]$ git checkout -b Development
Switched to a new branch 'Development'
[Lab_workshop]$ git checkout -b Proj1
Switched to a new branch 'Proj1'
[Lab_workshop]$ git branch
  Development
* Proj1
  Release
  master
  remote
```

Now we have our environment setup, time to do some work. Remember this one?



You can now replay this example by yourself or follow the chapter below. Check your status with following commands :

```
git branch
git status
git log –oneline –graph
git diff <branch>
```

There are no tags in the example, recommendation is to do this. Feel free to fill in some tags and trace them….

```
NAME
       git-tag - Create, list, delete or verify a tag object signed with GPG

SYNOPSIS
       git tag [-a | -s | -u <keyid>] [-f] [-m <msg> | -F <file>] [-e]
               <tagname> [<commit> | <object>]
       git tag -d <tagname>...
       git tag [-n[<num>]] -l [--contains <commit>] [--no-contains <commit>]
               [--points-at <object>] [--column[=<options>] | --no-column]
               [--create-reflog] [--sort=<key>] [--format=<format>]
               [--[no-]merged [<commit>]] [<pattern>...]
       git tag -v [--format=<format>] <tagname>...
```

Working local with GIT

```
[lab] git clone git@github.com:rickvek/Lab_workshop.git
[lab]$ cd Lab_workshop/
[Lab_workshop]$ git checkout -b Release
[Lab_workshop]$ git checkout -b Development
[Lab_workshop]$ git checkout -b Proj1
[Lab_workshop]$ git branch
[Lab_workshop]$ for no in 1 2 3 4 5 6 7 8 9 10
 do
  echo file${no} >> file${no}.txt
 done
[Lab_workshop]$ ll
[Lab_workshop]$ git add file*.txt
[Lab_workshop]$ git commit -m "Added filesXX.txt"
```

```
[Lab_workshop]$ git checkout Development
[Lab_workshop]$ git checkout -b Proj2
[Lab_workshop]$ for no in 1 2 3 4 5 6 7 8 9
 do
  echo Proj2_file${no}.txt >> Proj2_file${no}.txt
 done
[Lab_workshop]$ ll
[Lab_workshop]$ git add Proj*
[Lab_workshop]$ git commit -m "Added Proj2 files"
[Lab_workshop]$ git checkout Development
[Lab_workshop]$ git merge Proj2
```

```
[Lab_workshop]$ git log --oneline --graph
[Lab_workshop]$ git checkout Release
[Lab_workshop]$ git checkout -b HF
[Lab_workshop]$ for no in 1 2 3 4 5 6 7 8 9
 do
 echo HF_file${no}.txt  >> HF_file${no}.txt
 done
[Lab_workshop]$ git add HF*
[Lab_workshop]$ git commit -m "Hot Fix done"
[Lab_workshop]$ git checkout  Release
[Lab_workshop]$ git merge HF
[Lab_workshop]$ git branch -d HF
```

```
[Lab_workshop]$ git checkout Development
[Lab_workshop]$
[Lab_workshop]$ git checkout -b  Proj3
[Lab_workshop]$ for no in 1 2 3 4 5 6 7 8
 do
   echo Proj3_file${no}.txt >> Proj3_file${no}.txt
 done
[Lab_workshop]$ git add Proj*
[Lab_workshop]$ git commit -m "added Proj3 file "
[Lab_workshop]$ git checkout Development
[Lab_workshop]$ git merge Proj3
[Lab_workshop]$ git branch -d Proj3
```

```
[Lab_workshop]$ git checkout Proj1
[Lab_workshop]$ ls
[Lab_workshop]$ for no in 1 2 3 4 5 6 7 8 9
 do
  echo Proj1_file${no}.txt  >>  Proj1_file${no}.txt
 done
[Lab_workshop]$ git add Proj1_file*
[Lab_workshop]$ git commit -m "added Project1 files"
[Lab_workshop]$ git checkout Development
[Lab_workshop]$ git merge Proj1
[Lab_workshop]$ git branch -d Proj1
```

```
 [Lab_workshop]$ git checkout Release
 [Lab_workshop]$ git rebase Development
or
 [Lab_workshop]$ git merge  Development
```

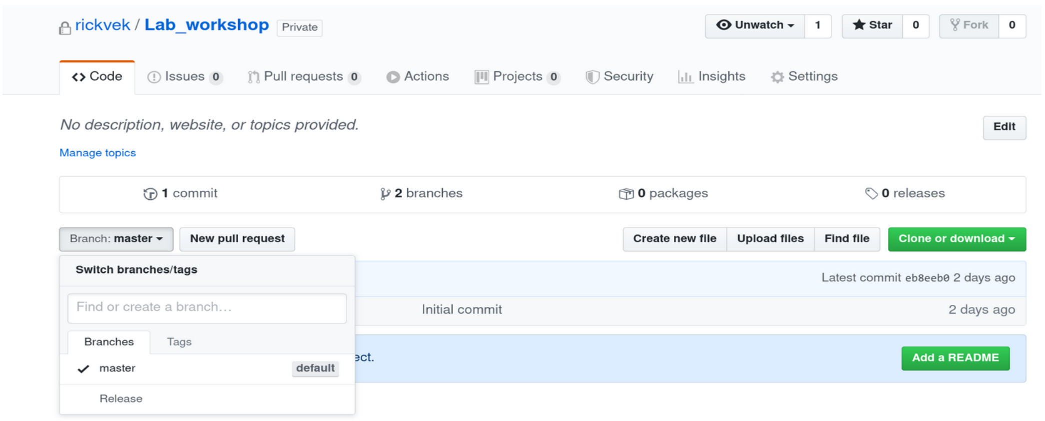
Last step, push it to the remote repository.

```
[Lab_workshop]$ git push origin Release
 Tunnel device open failed.
 Could not request tunnel forwarding.
 Enumerating objects: 58, done.
 Counting objects: 100% (58/58), done.
 Delta compression using up to 4 threads
 Compressing objects: 100% (12/12), done.
 Writing objects: 100% (57/57), 3.25 KiB | 302.00 KiB/s, done.
 Total 57 (delta 6), reused 0 (delta 0)
 remote: Resolving deltas: 100% (6/6), done.
 remote:
 remote: Create a pull request for 'Release' on GitHub by visiting:
 remote:      https://github.com/rickvek/Lab_workshop/pull/new/Release
 remote:
 To github.com:rickvek/Lab_workshop.git
  * [new branch]      Release -> Release
```

Done and now to the remote server to finish it.

Purging on the server

The next step is making a purge request.

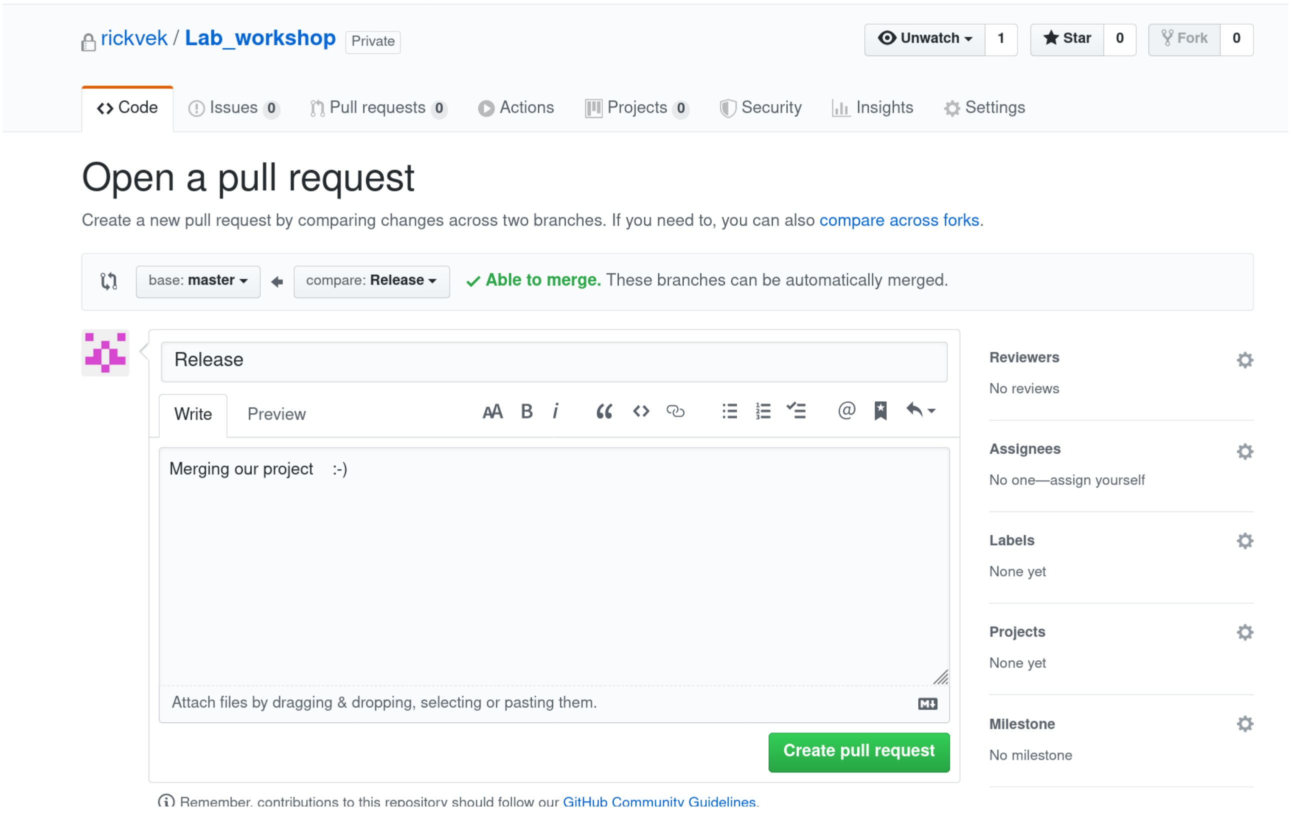


No description, website, or topics provided.

Latest commit eb8eeb0 2 days ago

Initial commit 2 days ago

There are now two branches, next to it is the " New pull request" button..



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

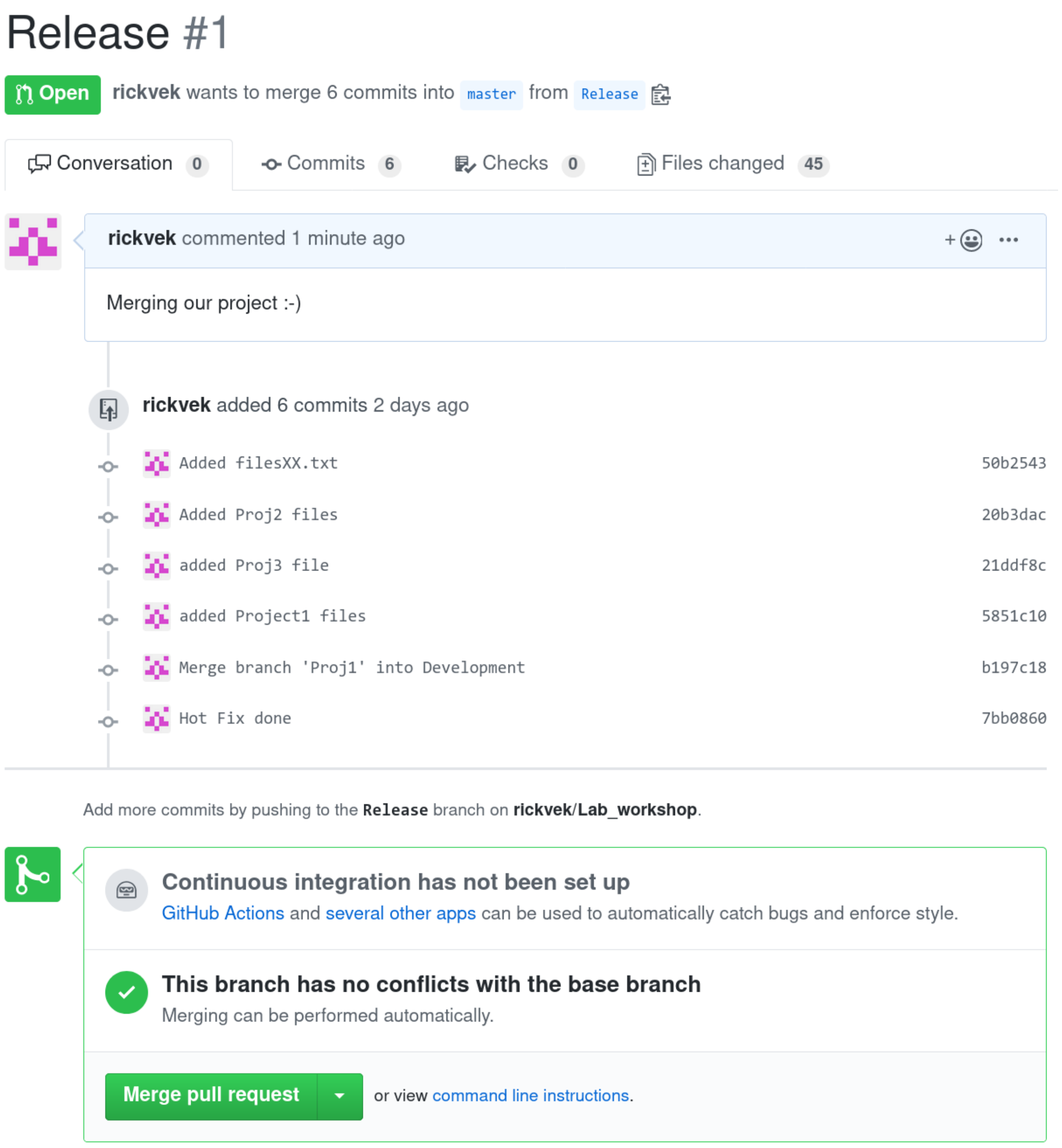
Able to merge. These branches can be automatically merged.

Release

Merging our project :-)

Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.


Release #1

Open rickvek wants to merge 6 commits into `master` from `Release`

Conversation 0 · Commits 6 · Checks 0 · Files changed 45

rickvek commented 1 minute ago

Merging our project :-)

rickvek added 6 commits 2 days ago

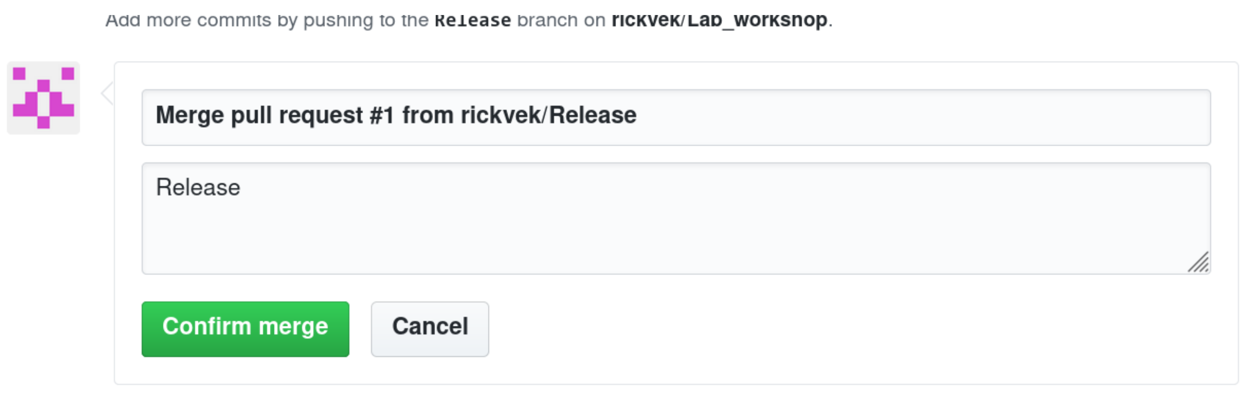
```
Added filesXX.txt                          50b2543
Added Proj2 files                          20b3dac
added Proj3 file                           21ddf8c
added Project1 files                       5851c10
Merge branch 'Proj1' into Development      b197c18
Hot Fix done                               7bb0860
```

Add more commits by pushing to the **Release** branch on **rickvek/Lab_workshop**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

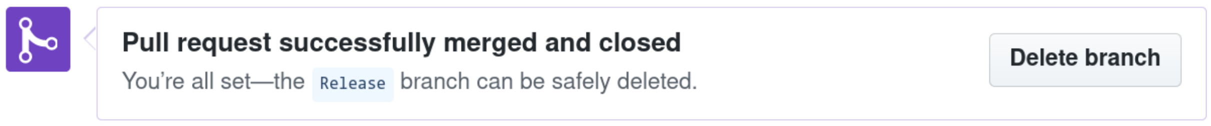
This branch has no conflicts with the base branch
Merging can be performed automatically.

Depending on the size the server is busy checking the merge conditions. When everything is ok, then the button for merging is green.


Add more commits by pushing to the **Release** branch on **rickvek/Lab_workshop**.

Merge pull request #1 from rickvek/Release

Release



Pull request successfully merged and closed

You're all set—the `Release` branch can be safely deleted.

When the merge is done and successful, then the branch can be deleted. Not needed any more so why keep it ….