

Introduction to Git — Fall 2023

Lecture 6: Working with remotes



UMEÅ
UNIVERSITET



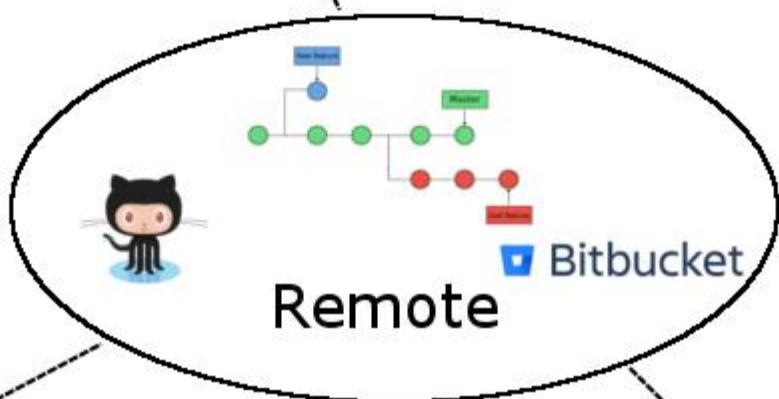
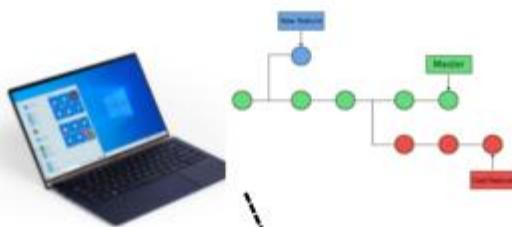
Slides: <https://hackmd.io/@git-fall-2023/L6-remotes#/>

Concepts

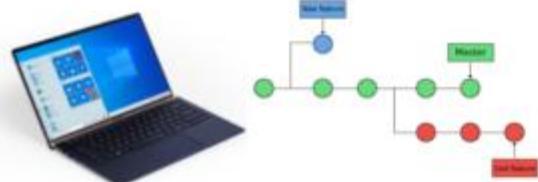
A remote repository is a version of the project which can be hosted in your local machine, some network, or over the internet (Pro Git, 2nd. Ed., Scott Chacon and Ben Straub) where you and your collaborators can push or pull code modifications.

In addition to this, a remote is a way to backup your repository.

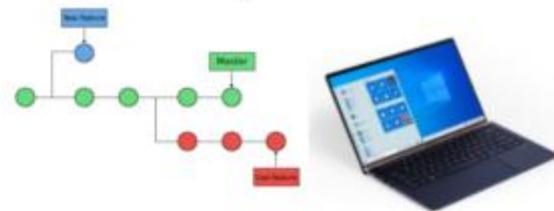
Mirko



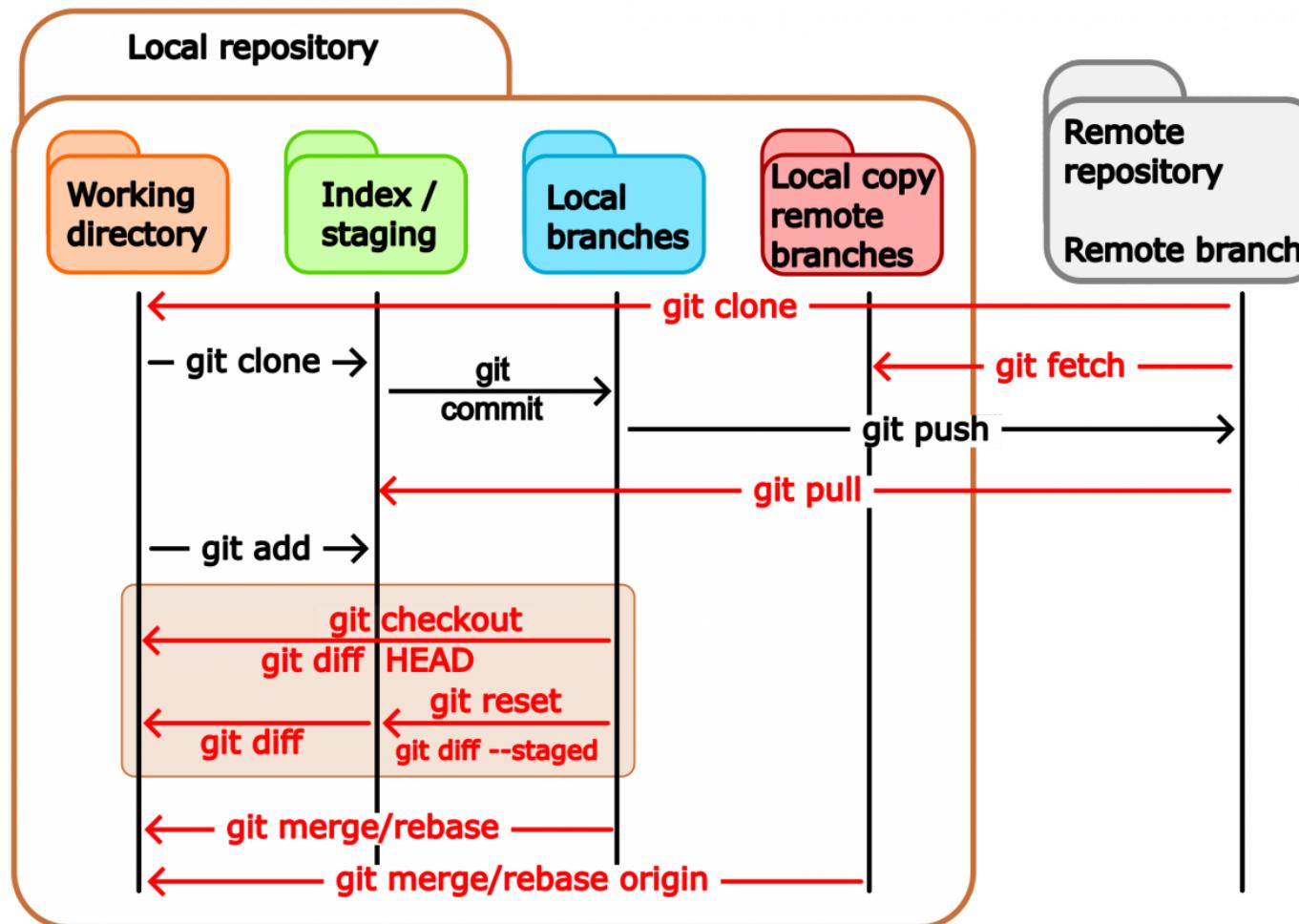
Pedro



Birgitte



Updated scheme for file stages



Concepts cont.

The command

```
$ git remote -v  
origin  git@bitbucket.org:arm2011/gitcourse.git (fetch)  
origin  git@bitbucket.org:arm2011/gitcourse.git (push)
```

displays the remotes that are already set up where you can *fetch* and *pull* changes. In this case there is only a single remote called **origin**.

```
$ git graph
* 2e56d0a (HEAD -> main, origin/main, origin/HEAD) text of ex
* 22a7316 Adding yet more lectures
* 0ddb791 Adding some more of the lectures
* 3ff9f8f Adding some of the lectures
```

Adding remotes

A remote repository can be added manually with the command

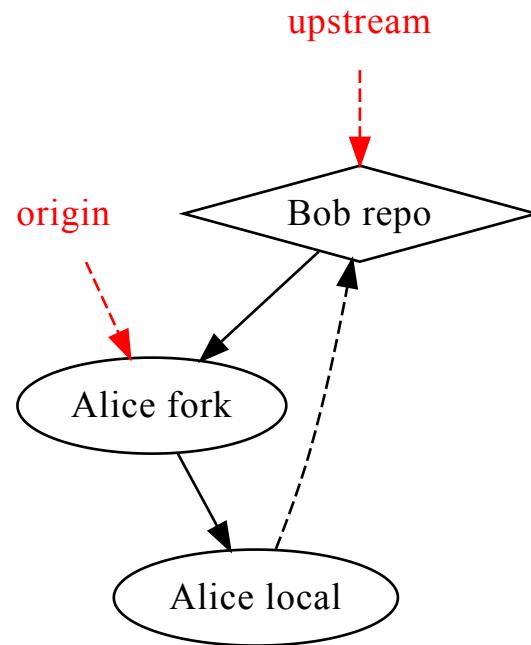
```
$ git remote add remote_name location  
$ git remote add remote_name git@github.com:aliceuser2020/my-first-project.git  
$ git remote -v  
remote_name git@github.com:aliceuser2020/my-first-project.git (fetch)  
remote_name git@github.com:aliceuser2020/my-first-project.git (push)
```

where the location of the remote can be an URL or the path if that is in your local machine.

Protocols:

- local -> git clone /opt/git/project.git
- SSH -> git clone ssh://user@server:project.git
- HTTP -> git clone
<http://example.com/gitproject.git>
- Git

Why do we need more than one remote?



```
$ git remote add upstream git@github.com:bob/my-first-project.git  
  
$ git remote -v  
origin  git@github.com:aliceuser2020/my-first-project.git (fetch)  
origin  git@github.com:aliceuser2020/my-first-project.git (push)  
upstream  git@github.com:bobuser2020/my-first-project.git (fetch)  
upstream  git@github.com:bobuser2020/my-first-project.git (push)
```

```
$git graph
* 2e56d0a (HEAD -> main, upstream/main, origin/main, origin/HEAD) text of exercise
* 22a7316 Adding yet more lectures
* 0ddb791 Adding some more of the lectures
* 3ff9f8f Adding some of the lectures
```

Working with remotes

One can push or fetch/pull to or from remotes by

```
$ git push  remote_name branch_name  
$ git fetch remote_name branch_name  
$ git pull  remote_name branch_name
```

In case you obtained the repository by cloning an existing one you will have the **origin** remote. You can do push/fetch/pull for this remote with

```
$ git push origin master  
$ git fetch origin master  
$ git pull origin master
```

or

```
$ git push  
$ git fetch  
$ git pull
```

because the remote *origin* and the *master* branch are configured for pushing and pulling by default upon cloning.

The command:

```
$ git pull
```

brings all the changes (branches) that are in the remote and tries to merge them with your local repo. The default behavior of *git pull* is in the *\$GIT_DIR/config* file:

```
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
```

In fact, *git pull* is a combination of two commands:

```
$ git fetch remote_name branch_name  
$ git merge remote_name/branch_name
```

or

```
$ git fetch  
$ git merge
```

Advanced

The command

```
$ git push
```

will send all the changes (branches) to the remote by default. This can be changed by applying:

```
git config --global push.default matching(default), current,
```

If you have a brand-new branch called **new**, then it is recommended to push it the first time with the command:

```
git push -u origin new
```

which is equivalent to

```
git push origin new  
git branch --set-upstream new origin/new
```

then, you will be able to push/pull the changes in the branch by simply typing **git push/pull**

Displaying remote information

```
$ git remote show origin
* remote origin
  Fetch URL: git@bitbucket.org:arm2011/gitcourse.git
  Push  URL: git@bitbucket.org:arm2011/gitcourse.git
  HEAD branch: master
  Remote branches:
    experiment      tracked
    feature         tracked
    less-salt       tracked
    master          tracked
    nested-feature  tracked
  Local branches configured for 'git pull':
    feature        merges with remote feature
    master         merges with remote master
    nested-feature merges with remote nested-feature
  Local refs configured for 'git push':
    feature        pushes to feature        (fast-forwardable)
    master         pushes to master        (up to date)
    nested-feature pushes to nested-feature (up to date)
```

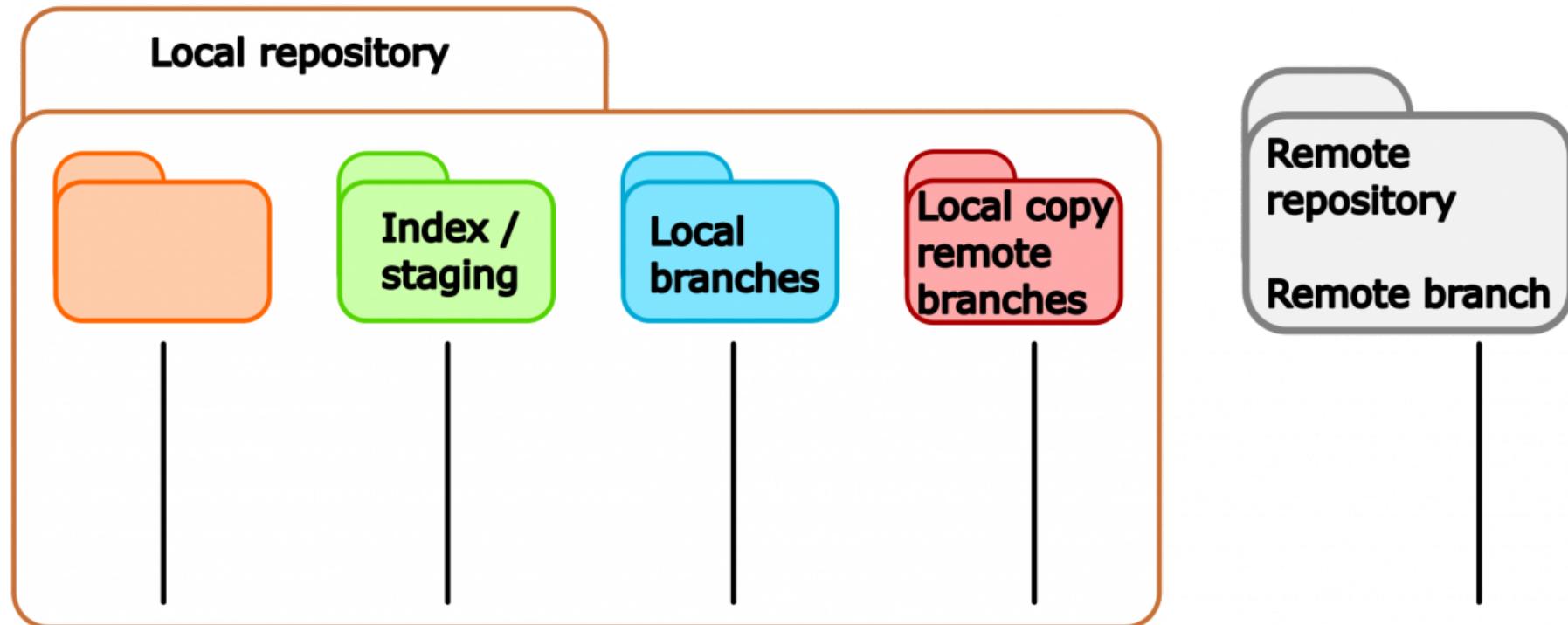
Renaming remotes

```
$ git remote rename initial_name new_name
```

Deleting remotes

```
$ git remote remove remote_name
```

Bare repositories



A bare repository is a repository with no working directory.

Creating a bare repository

```
$ mkdir bare.git && cd bare.git  
$ git init --bare
```

Cloning a bare repository cont.

```
$ git clone --bare location
```

Using GitHub

The screenshot shows the GitHub homepage with a dark theme. On the left, there's a large white banner with the text "Built for developers" and a paragraph about GitHub's mission. On the right, there's a sign-up form for new users. The form includes fields for "Username", "Email", and "Password". Below the password field is a note: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)". At the bottom of the form is a green button labeled "Sign up for GitHub". The top of the page features the GitHub logo, navigation links like "Why GitHub?", "Team", "Enterprise", "Explore", "Marketplace", and "Pricing", a search bar, and user account links for "Sign in" and "Sign up".

https://github.com

... 🌐 ⭐️ Search

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub

Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

Username

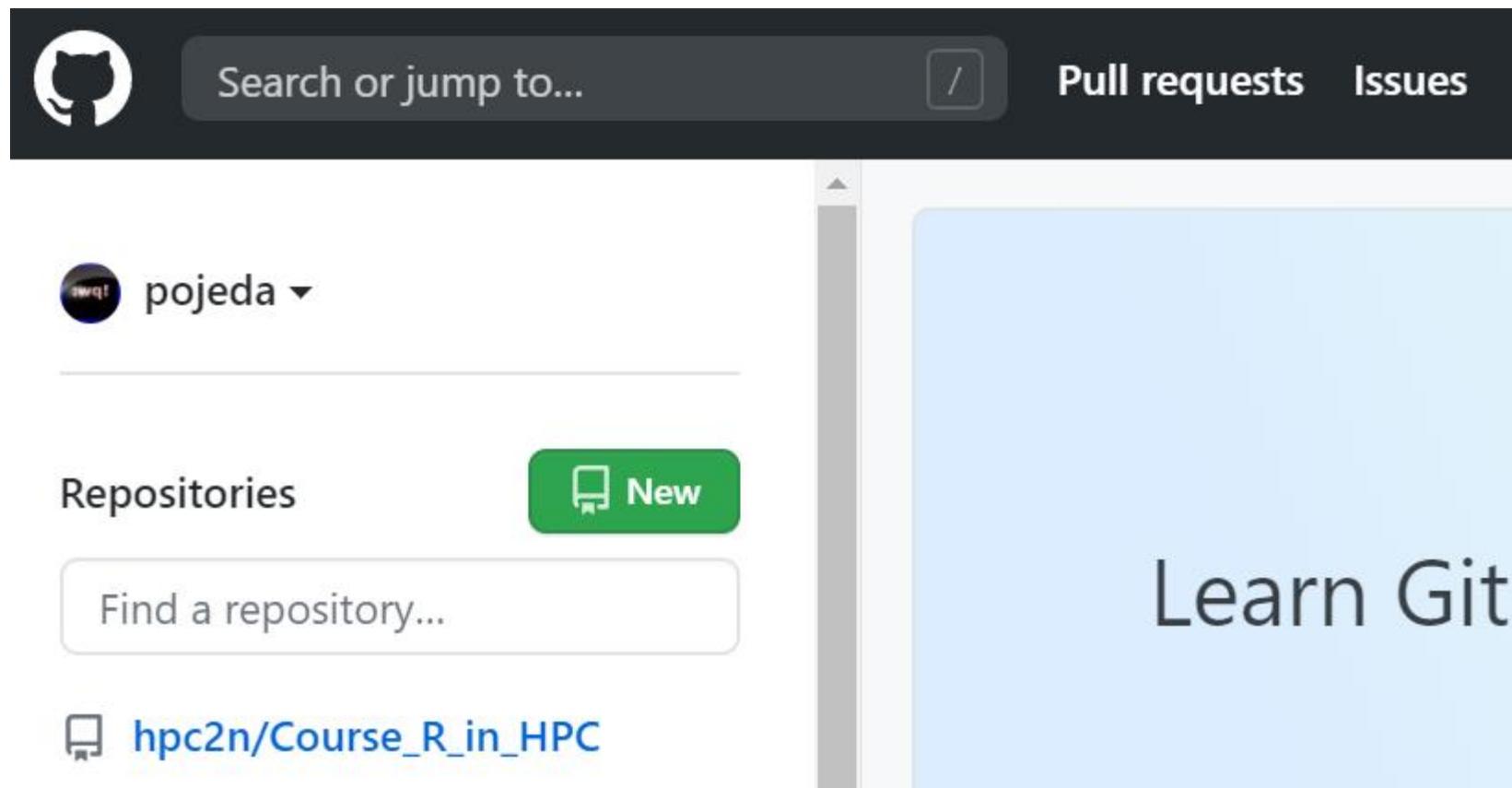
Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

Upon login into your GitHub account you will see the following option to create a new repository



Here, you can choose the type of repository that is appropriate to your needs (public/private), if you want to add *README* and *.gitignore* files and also the type of license for your project,

Create a new repository

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

[Import a repository.](#)

Owner *

Repository name *



pojeda ▾



/ my-first-repo



Great repository names are short and memorable. Need inspiration? How about [expert-waddle](#)?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

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

GitHub will suggest some steps that you can take for your brand-new repository:

The screenshot shows the GitHub interface for creating a new repository. At the top, there are three cloning options: 'Set up in Desktop' (buttoned), 'or', 'HTTPS' (selected), and 'SSH'. Below this, the repository URL is displayed as `git@github.com:pojeda/my-first-repo.git`. A note below the URL encourages users to create or upload files, and recommends including `README`, `LICENSE`, and `.gitignore`.

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# my-first-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin git@github.com:pojeda/my-first-repo.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:pojeda/my-first-repo.git  
git branch -M master  
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

Setting ssh-keys

1. ssh-keygen -t rsa -b 4096 -C
“pedro@gemail.com”
2. eval \$(ssh-agent -s)
3. ssh-add ~/.ssh/id_rsa
4. clip < ~/.ssh/id_rsa.pub (it copies the ssh key
that has got generated)

5. Go to your remote repository on [github.com](#) and then **Settings** -> **SSH and GPG keys** ->**new SSH key** -> write a title and paste the copied SSH key and save it
6. check if the key was properly set on [github/bitbucket](#)

```
$ ssh -T git@bitbucket.org  
$ ssh -T git@github.com
```

SSH keys / Add new

Title

Key

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

Add SSH key

Network visualization

Pulse

Contributors

Traffic

Commits

Code frequency

Dependency graph

Network

Forks

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

The repository network shows the 100 most recently pushed forks. Do you need to see more forks? Please [give us feedback](#) on your usage of this feature.

pojeda

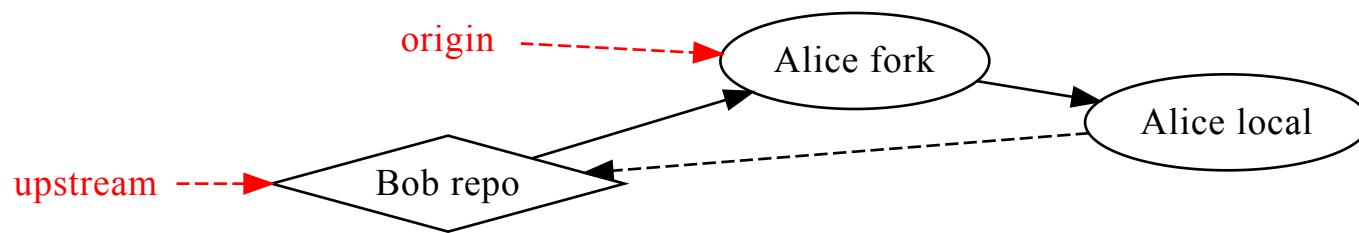
ct

6

master

Pull requests

In the following scenario, a developer, Bob, has its repo on GitHub. Another developer, Alice, finds it useful and forks it. After doing some changes, Alice push them and do a “pull request”



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base repository: bobuser2020/my-first-project ▾

base: master ▾



head repository: aliceuser2020/my-first-project ▾

compare: master ▾

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

[Create pull request](#)

-o 1 commit

1 file changed

0 comments

1 contributor

Commits on Sep 28, 2020

Update README.md

Verified

a8902e2

Showing 1 changed file with 3 additions and 1 deletion.

Unified Split

v 4 README.md

↔ ↗ ⋮

... ... @@ -1 +1,3 @@

Then, Bob receives an email with the pull request information about Alice modifications. On the GitHub site he sees the request:

bobuser2020 / my-first-project

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Update README.md #1

[Open](#) aliceuser2020 wants to merge 1 commit into `bobuser2020:master` from `aliceuser2020:master`

Conversation 0 Commits 1 Checks 0 Files changed 1

aliceuser2020 commented 11 minutes ago First-time contributor ...
change from Alice

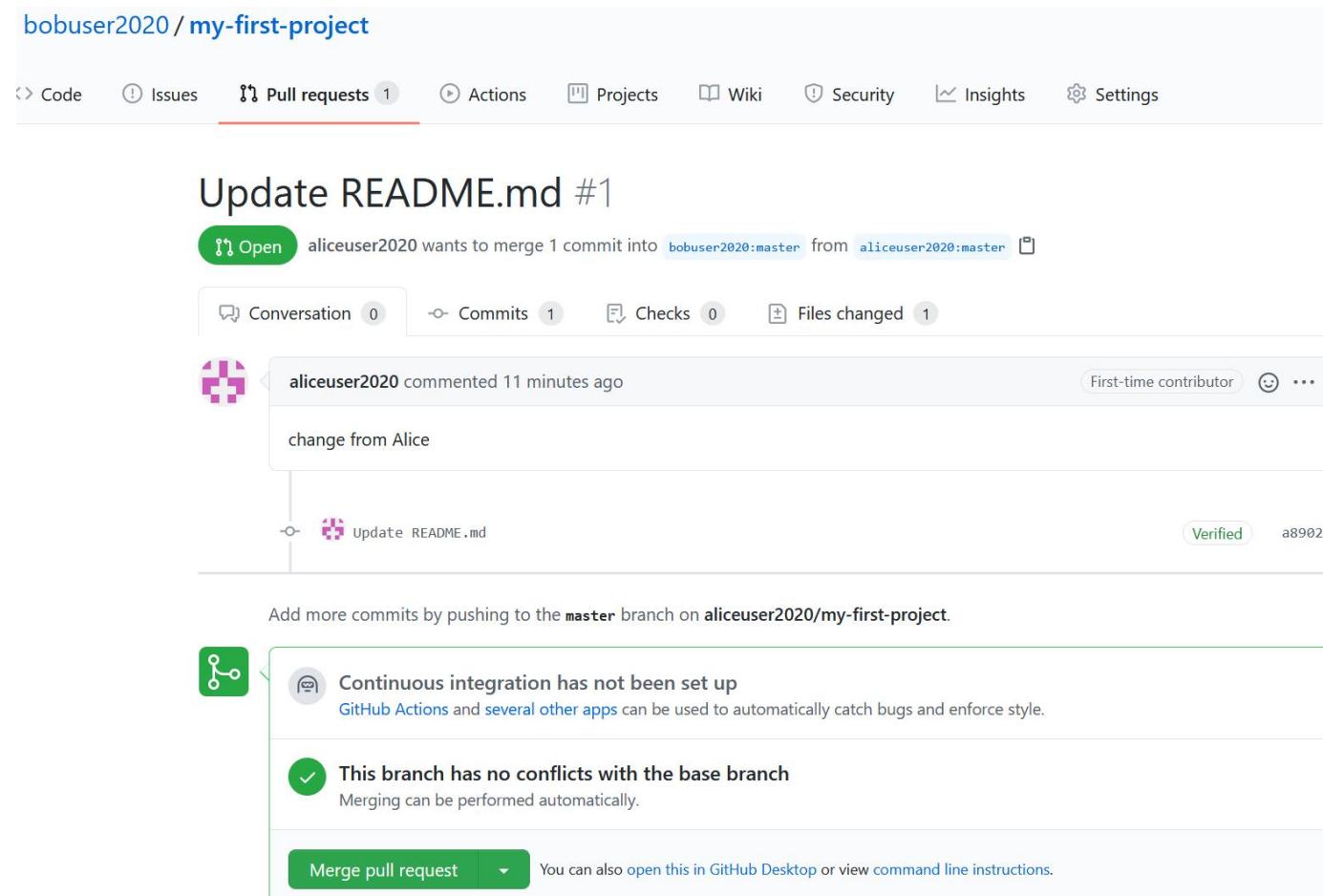
Update README.md Verified a8902e2

Add more commits by pushing to the `master` branch on [aliceuser2020/my-first-project](#).

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.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.



Because Bob finds the changes from Alice useful and there are no conflicts he can merge them straight away,

bobuser2020 / my-first-project

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Update README.md #1

Merged bobuser2020 merged 1 commit into bobuser2020:master from aliceuser2020:master 5 minutes ago

Conversation 0 Commits 1 Checks 0 Files changed 1

aliceuser2020 commented 17 minutes ago
change from Alice

Contributor

Verified a8902e2

Update README.md

bobuser2020 merged commit ba421a0 into bobuser2020:master 5 minutes ago

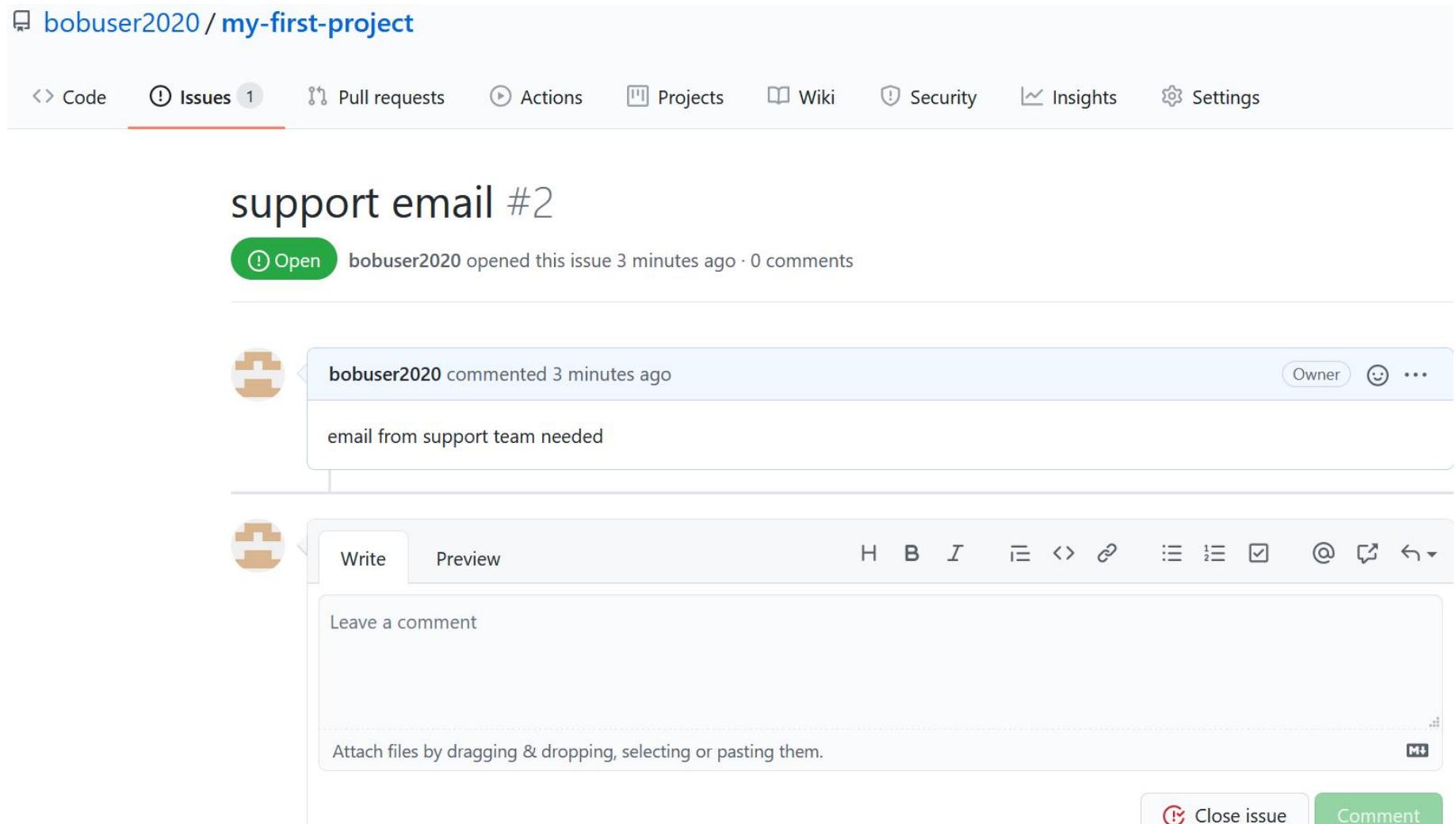
Revert

The screenshot shows a GitHub pull request merge history. At the top, it says "Merged bobuser2020 merged 1 commit into bobuser2020:master from aliceuser2020:master 5 minutes ago". Below this, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment from "aliceuser2020" is shown, stating "change from Alice". The commit itself is titled "Update README.md" and is marked as "Verified" with hash "a8902e2". At the bottom, it shows "bobuser2020 merged commit ba421a0 into bobuser2020:master 5 minutes ago". There is also a "Revert" button.

Issues

If you find some issues in the files/code you can open an “Issue” on GitHub

The screenshot shows the GitHub interface for a repository named "bobuser2020 / my-first-project". The "Issues" tab is selected, indicated by a red underline. A search bar at the top contains the text "support email". Below the search bar, there are two tabs: "Write" and "Preview", with "Write" currently active. The main area is a rich text editor with a blue border, containing the text "email from support team needed". Above the editor is a toolbar with various formatting icons: H, B, I, bold, italic, etc. At the bottom of the editor area, there is a placeholder text "Attach files by dragging & dropping, selecting or pasting them." and a small "M+" icon. At the very bottom of the screen, there is a note "Styling with Markdown is supported" and a green button labeled "Submit new issue".



You may also assign people to the issues that are more related to that topic.

In future commits you may refer to this issue by using the issue number, **#2** in this case. This will allow you to track the evolution of the issue on GitHub.

Best practices

- Talk with your colleagues.
- Some commands such as **git rebase** change the history. It wouldn't be a good idea to use them on public branches.
- Don't accept pull requests right away.