Introduction to Open & Reproducible Science (IORDS)

Michael Dayan, Data Scientist Manager

Methods & Data facility
Human Neuroscience Platform
Foundation Campus Biotech Geneva

Virtual machine info

To get an IP, please fill the form at:
`https://tinyurl.com/IORDS2021-IP-git2`

START RDP CLIENT (as instructed in email / Slack):

- *Remote Desktop Connection* on Windows
- *Remote Desktop App* on Mac OS
- *Remmina* on Linux distributions (e.g. Ubuntu)

Connecting your:	WIFI SSID	WIFI Password
Laptop (no phones)	NIDS_course	reproduciblescience
Phone	CAMPUS_VISITORS	welcomecampus

PLEASE CONNECT TO THE VM
→ Login: brainhacker
→ Password: brainhack!

Connect to Slack and download the exercise slides

ANY PROBLEM? Please raise your hand or ask questions on Slack: channel **`#git`**

On site support (including coding):

Maël

Louis

Remote support (including coding):

Serafeim

HNP METHODS & DATA

LECTURE OBJECTIVES

GIT lectures objectives (you should be able to…):

- Know what GIT is useful for and identify some famous actors in the Git ecosystem
- Understand the “promotion model” and “commits” at the heart of GIT
- Describe the typical workflow to record code changes
- Examine the code history and compare different code versions
- Understand the concepts of branches and branch merging

GIT Part 1

- Know how to collaborate with Git using Github
- Understand the concepts of remote repository / remote branches
- Synchronize code changes between local and remote repositories
- Distinguish between different kinds of merge

GIT Part 2

- Collaborate on public projects with Github
- Know advanced merging techniques and deal with conflicts
- Debugging with GIT

GIT Part 3

→ Track your code changes with GIT and collaborate on your own or public projects with Github

COLLABORATING (INCLUDING WITH ONESELF) WITH GIT

A crucial `git` feature is that it is distributed. A repository can be accessed from anywhere for:

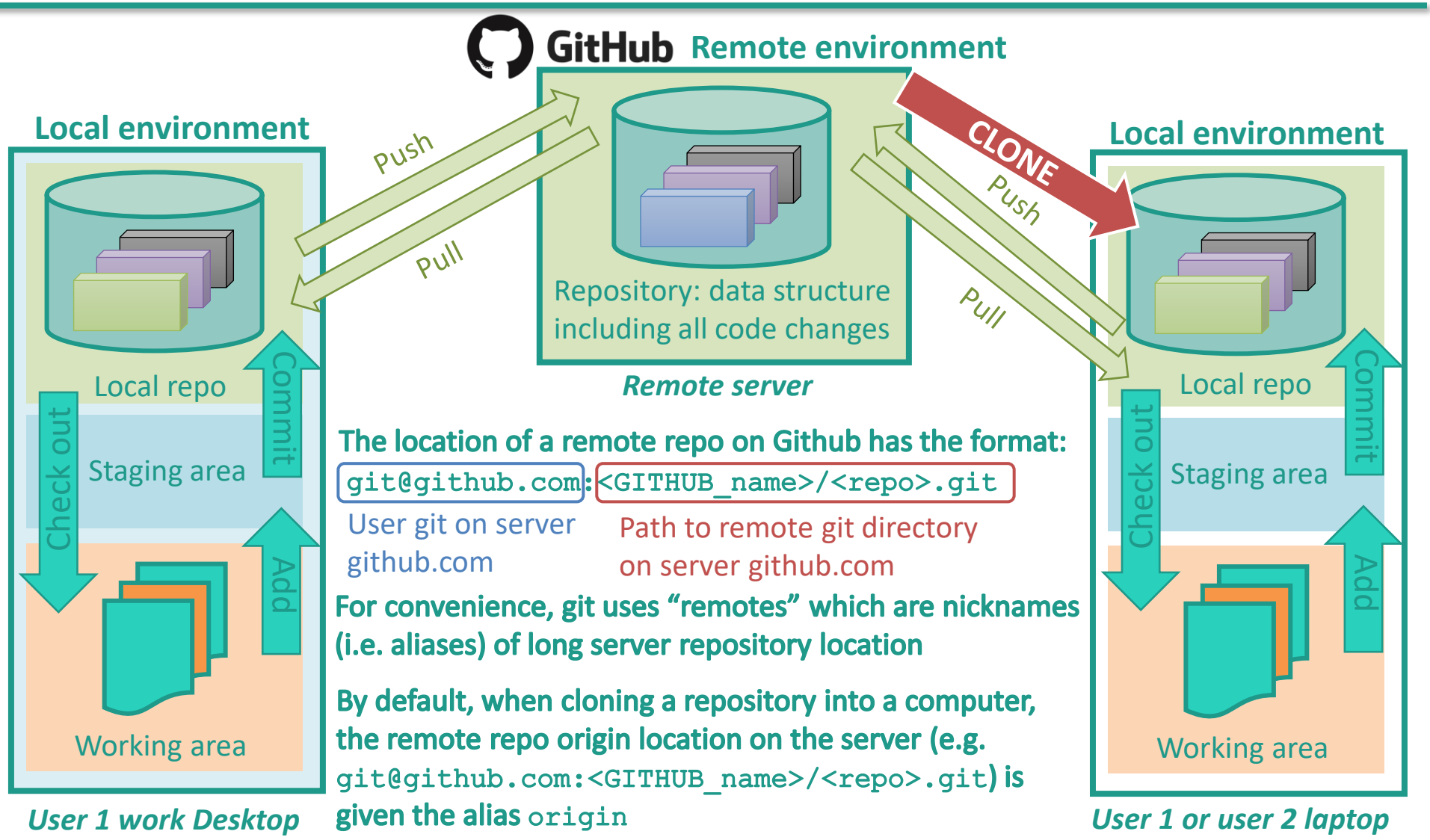
- "self-collaboration" (~ Dropbox)
- team collaboration

A remote repository is only meant for tracking and synching (no working or staging area)

It is accessed by multiple users and reached via a network protocol such as SSH or HTTPS

Each user can:

- send updates from the local to the remote repo with `git push`
- receive updates using:
  - `git clone` the 1st time the local repo is created
  - `git pull` any other time



The location of a remote repo on Github has the format:

`git@github.com:<GITHUB_name>/<repo>.git`

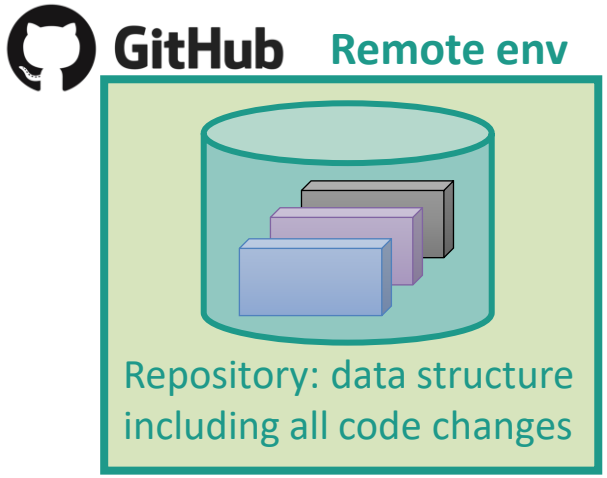
User git on server github.com

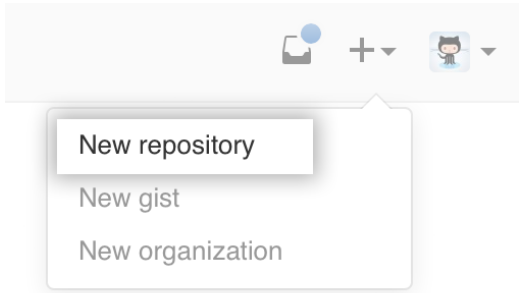
Path to remote git directory on server github.com

For convenience, git uses "remotes" which are nicknames (i.e. aliases) of long server repository location

By default, when cloning a repository into a computer, the remote repo origin location on the server (e.g. `git@github.com:<GITHUB_name>/<repo>.git`) is given the alias `origin`

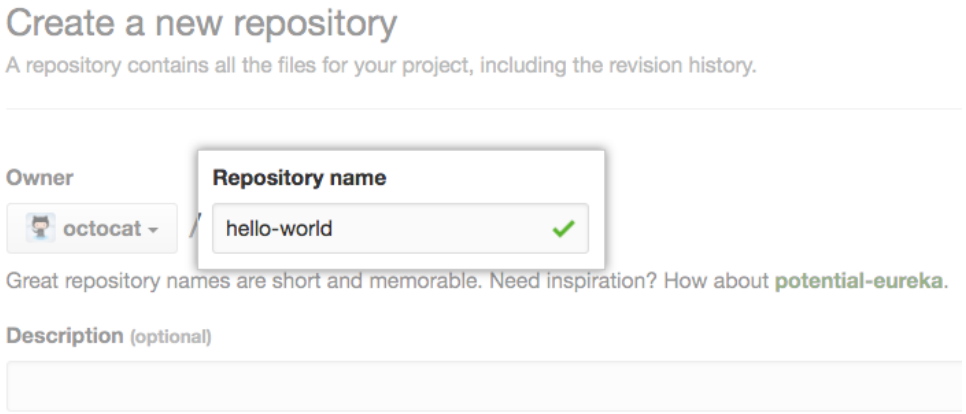
How to push an existing local repository to GitHub ?

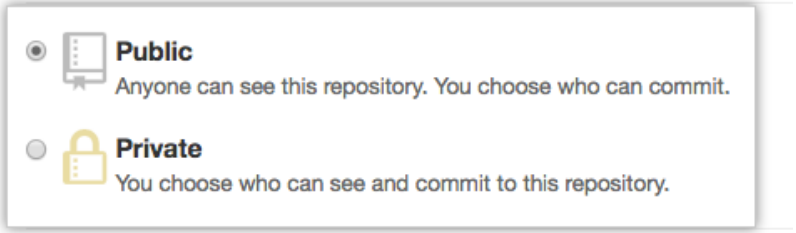




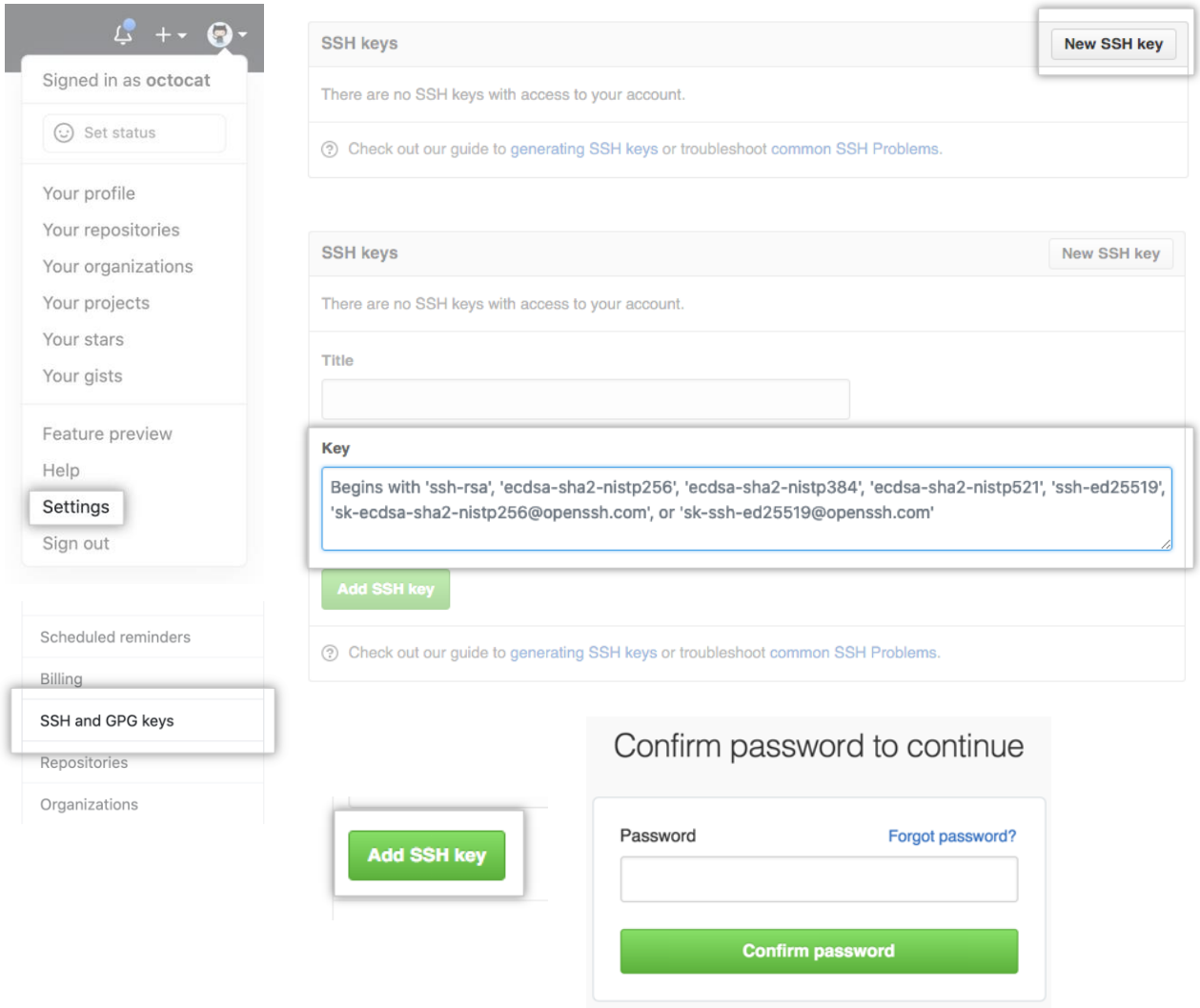
1. On GitHub, create an empty repository

- Click on the "+" symbol on the top right and choose "New repository"
- Choose a name for the new repository, e.g. `greeter`
  - Optionally provides a description
- Set the project to "Public"
- Do not initialize the repository with any files since the project will be pushed to GitHub
- Click on the "Create Repository" button





How to push an existing local repository to GitHub ?



2. Authorize your computer to access GitHub

This will be done by copying your local user public SSH key to the "authorized keys" on GitHub

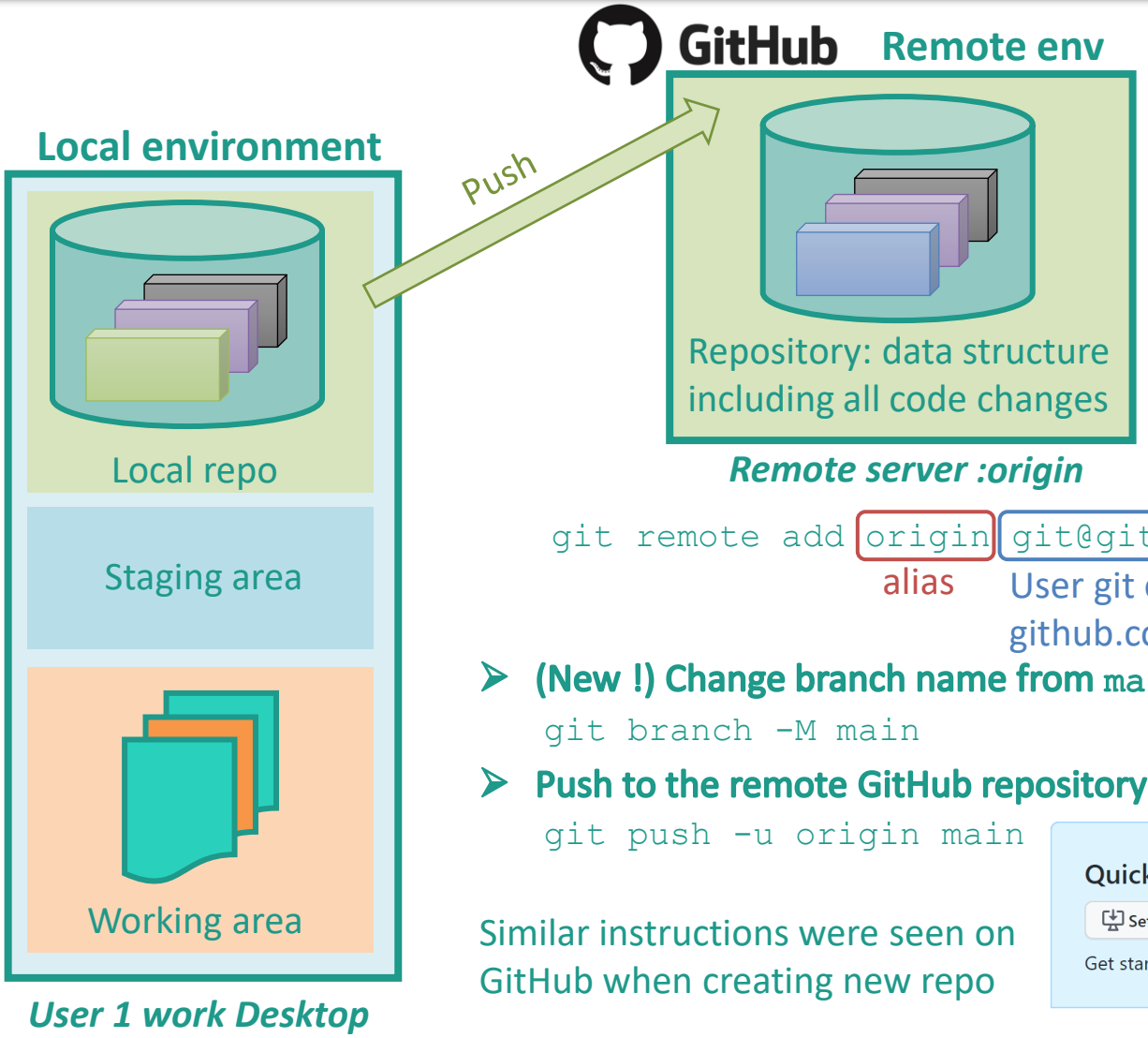
- On your local computer, display the content of your public key on the terminal and copy it (you may need create a public/private key pair if you don't have one already)

```
cat ~/.ssh/id_rsa.pub
```

  To copy it you could select the output with your mouse, right click and select copy (or use the SHIFT + CTRL + C shortcut)

- On GitHub:
  - Click on your profile photo then on Settings
  - Click on "SSH and GPG keys" in the sidebar
  - Click on "New SSH key" or "Add SSH key", and
    - Add a label for your computer (e.g. "Home Laptop")
    - Paste the content of your public key in the "Key" field
  - Click on "Add SSH key"
  - Confirm your GitHub password if asked

How to push an existing local repository to GitHub ?



3. On your computer, push your local repo to your newly created GitHub repo

➢ Add an alias `origin` to point to the repo location on GitHub servers

Note: an alias is just a shortcut to avoid typing the whole remote location path when using git commands. Such an alias is called a `remote`

You can check all the remote definitions with `git remote:`

```
> git remote -v
```

```
git remote add origin git@github.com:<GITHUB_username>/<reponame>.git
```

- `origin`: alias
- `git@github.com`: User git on server github.com
- `<GITHUB_username>/<reponame>.git`: Path to remote git directory on server github.com

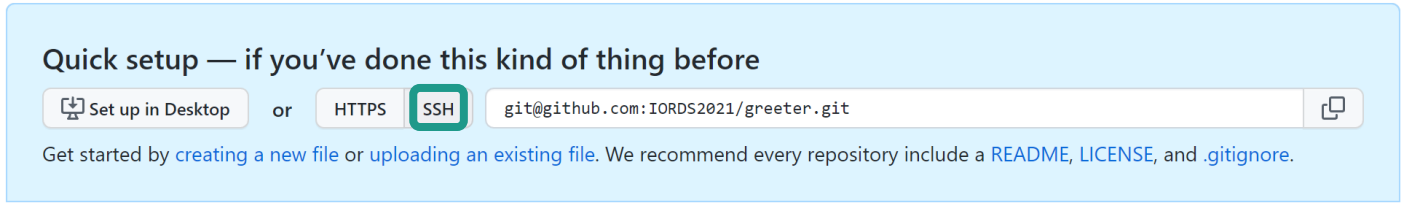
➢ (New !) Change branch name from `master` to more neutral `main` name

```
git branch -M main
```

➢ Push to the remote GitHub repository (located at `origin`) your local branch `main`

```
git push -u origin main
```

Similar instructions were seen on GitHub when creating new repo



HNP METHODS & DATA

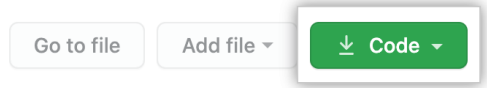
campus biotech

Fondation Campus Biotech Geneva

How to get a GitHub repo to have it on a local computer ?

1. On GitHub, get the path location of the remote repo on GitHub servers

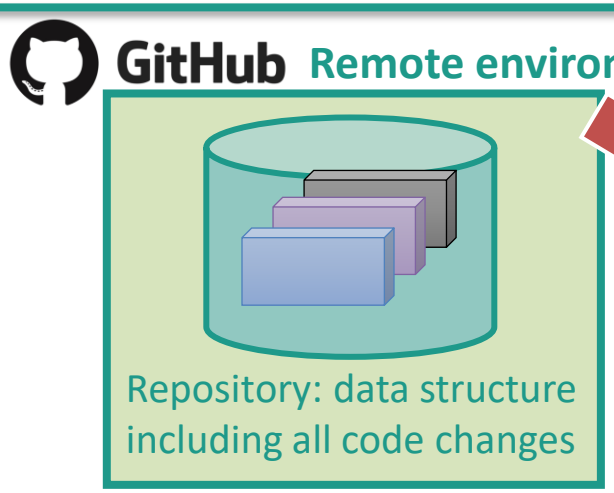
- Go to the main page of your existing GitHub repository
- Click on the "Code" button

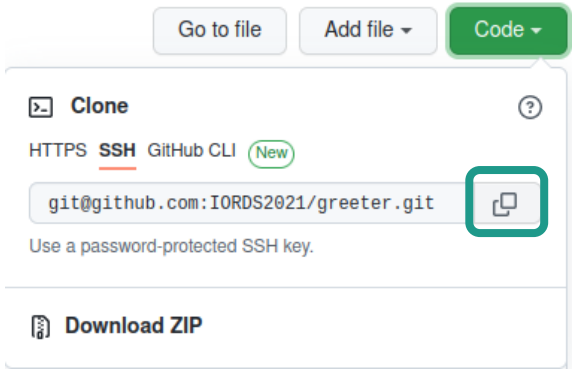


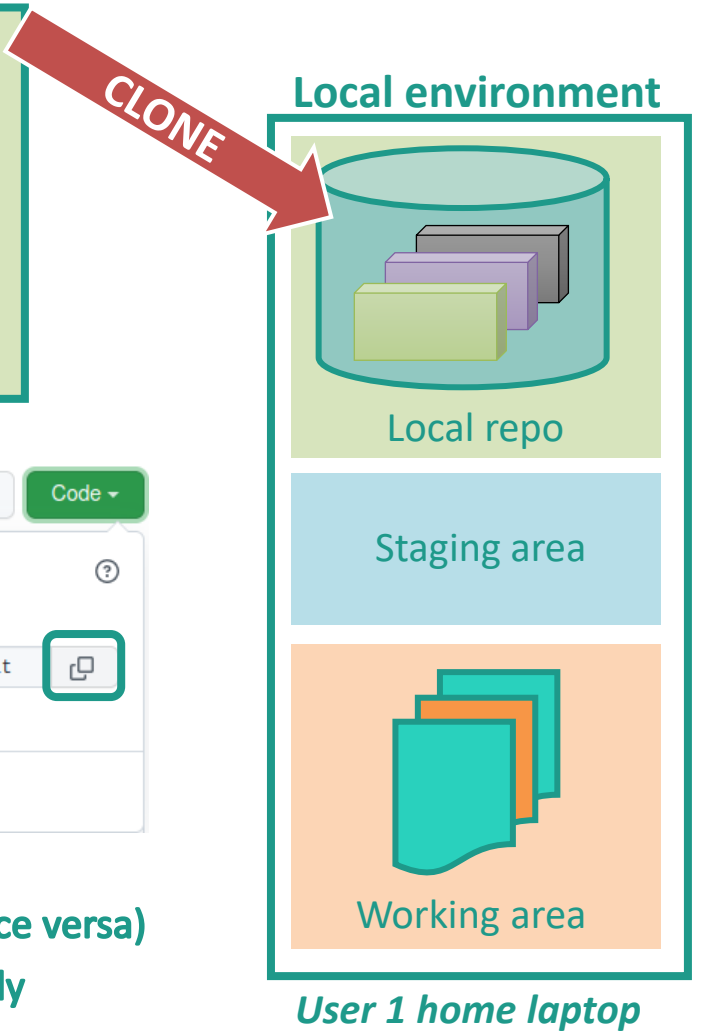
- To clone with SSH click on the copy symbol (double squares) under "Clone / SSH"

2. On your machine, "clone" the GitHub repository into a new local git repository

- Go inside the directory where you put your different coding projects
- Then type `git clone` and paste the link you just copied:
  `git clone git@github.com:<GITHUB_username>/greeter.git`
- Go inside the cloned repo (let's assume it is called `greeter`)
  `cd greeter`
- Check everything is fine (e.g. with `git status`)







Now for example one local repository can push changes to `origin` with: `git push origin main` ,
while the other local repository can pull changes from `origin` with: `git pull origin main` (and vice versa)
Note: no need to define a remote after cloning a GitHub repo, a remote `origin` is created automatically

GIT LAB 1 – working from multiple places

Your lab now allows for remote working. As a result, before leaving work, you decided to create a remote GitHub repository to be able to also work from home on your personal laptop on the code you developed on your work computer.

Make sure you are in the `test_hello` directory

Follow the instructions on the previous slides to create a GitHub repo called `greeter` and pushing your code there

Check your GitHub repo page to make sure all your code is there on the `main` branch, including all your commits.

To simulate another computer, create the directory `/home/brainhacker/home_laptop`

Go inside the `home_laptop` directory

Follow the instructions on the previous slides to clone your `greeter` repo inside the `home_laptop` directory

Change the `hello` code by replacing “Hi” with “Hello”

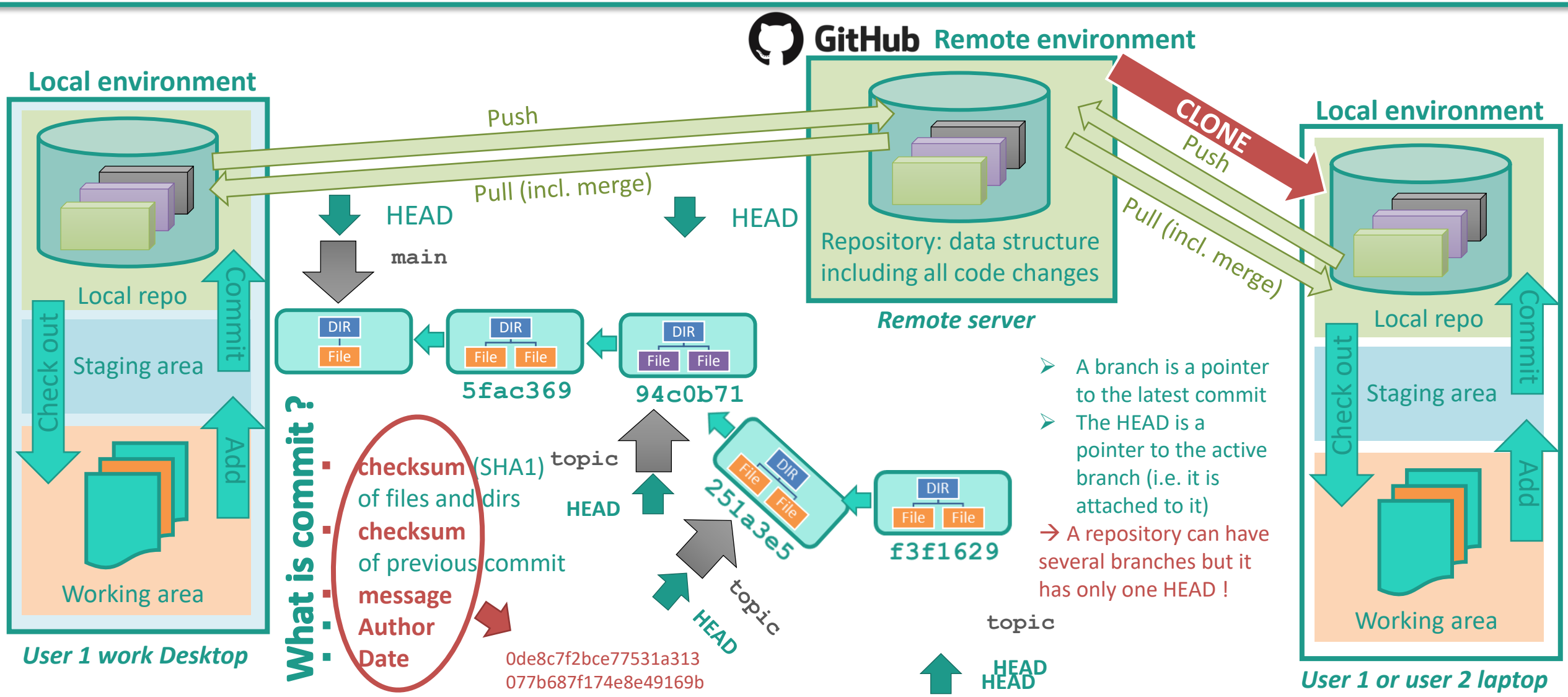
Stage the modified file, then commit it

Push your changes to the remote repo, and check GitHub

```
pwd # print path of current directory
cd DIR # change to directory DIR
ssh-keygen -t rsa -b 4096 # create an SSH key pair
git remote add ALIAS git@github.com:<user>/<repo>.git
# set alias ALIAS to a Github remote repository location
git branch -M main # rename current branch to main
git push -u ALIAS BRANCH
# push to remote ALIAS the local branch BRANCH
Note: click on  3 commits  button on GitHub to see commits
mkdir DIR # create the directory DIR
git config --global user.name NAME  # use quotes for NAME
git config --global user.email EMAIL
git clone git@github.com:<user>/<repo>.git # clone GitHub repo
git status # give git status of local environment
git add FILE # stage FILE
git add . # stage all eligible files
git commit # commit staged files to local repo
           # message will be written in an editor
git commit -m MESSAGE # commit with message MESSAGE
git push ALIAS BRANCH # push to repo ALIAS the branch BRANCH
git pull ALIAS BRANCH # pull from repo ALIAS the branch BRANCH
```

Go to the initial `test_hello` directory and check you have the old code. Pull the changes from the remote. Check you have the new code.

SUMMARY: HOW DOES GIT WORK ?



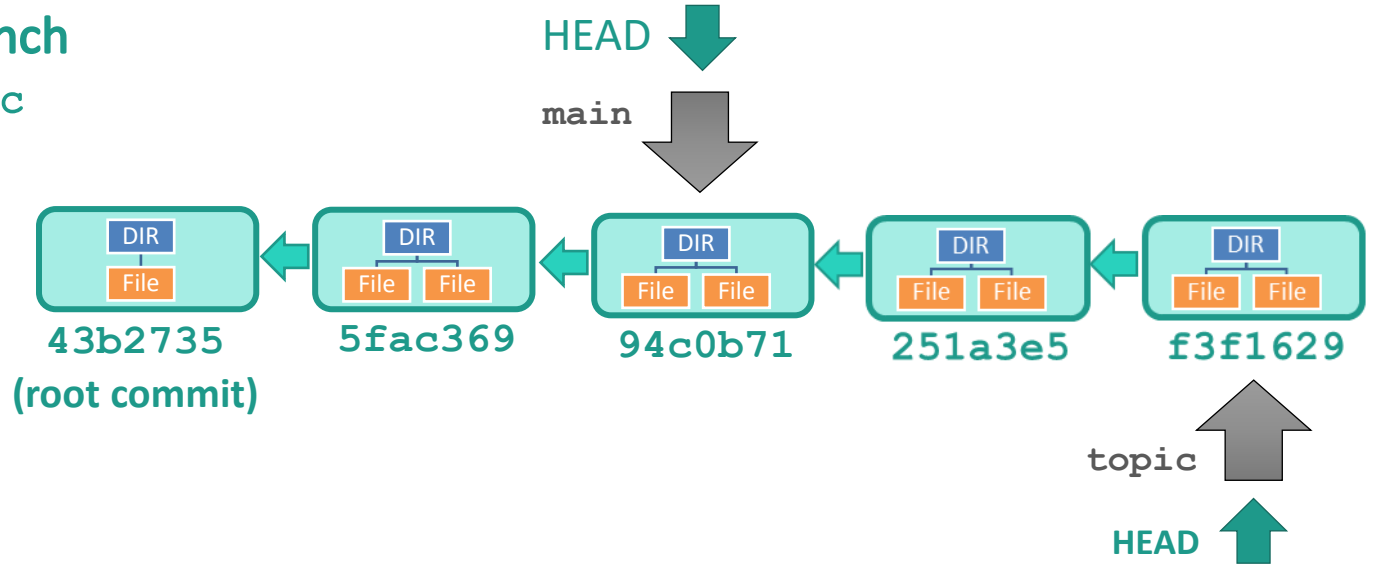
GIT BRANCHES – THE IMPORTANCE OF BRANCH UPDATES

How to merge a branch (e.g. called `topic`) into the `main` branch?

Simplest situation: no changes made on `main` branch

- → **the `main` branch is "paused" compared to the `topic` branch (it is a subset of the `topic` branch)**
- → **the `main` branch pointer can simply be fast-forwarded to the `topic` branch pointer**

```
> git checkout main
> git merge topic
```



GIT BRANCHES – THE IMPORTANCE OF BRANCH UPDATES

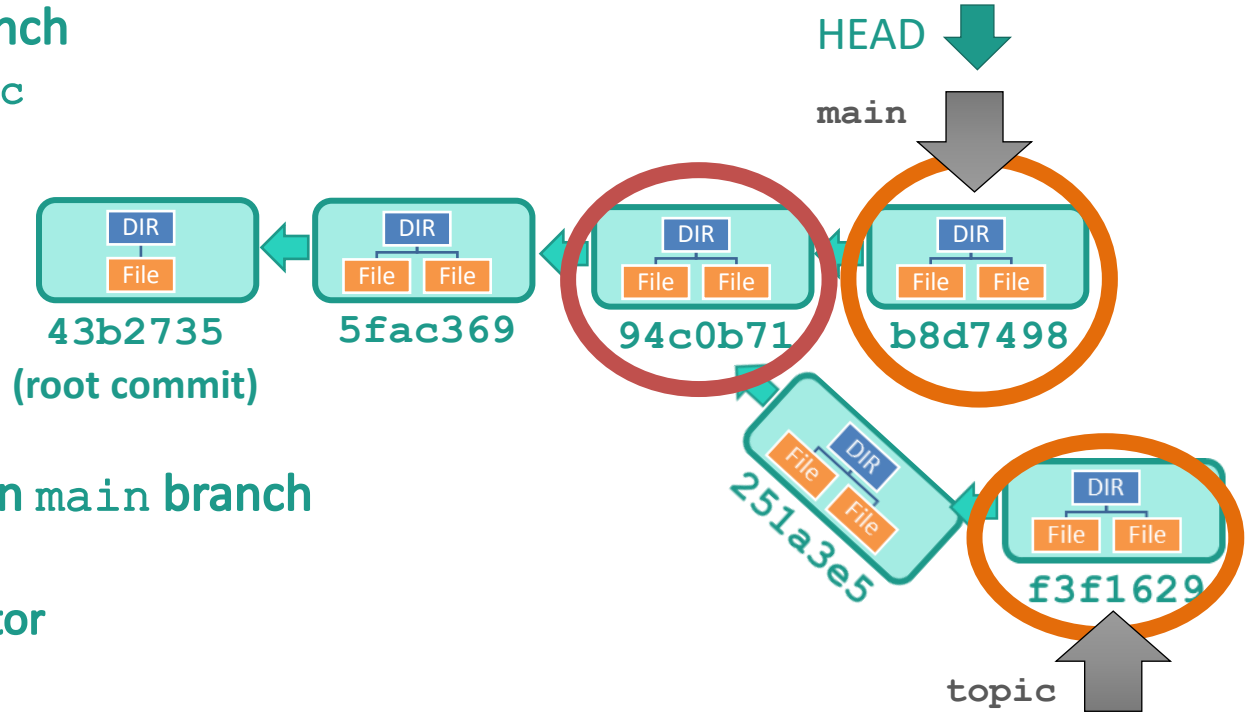
How to merge a branch (e.g. called `topic`) into the `main` branch?

Simplest situation: no changes made on `main` branch

→ the `main` branch is "paused" compared to the `topic` branch (it is a subset of the `topic` branch)

→ the `main` branch pointer can simply be fast-forwarded to the `topic` branch pointer

```
> git checkout main
> git merge topic
```



More problematic situation: new changes made on `main` branch after `topic` branch was created

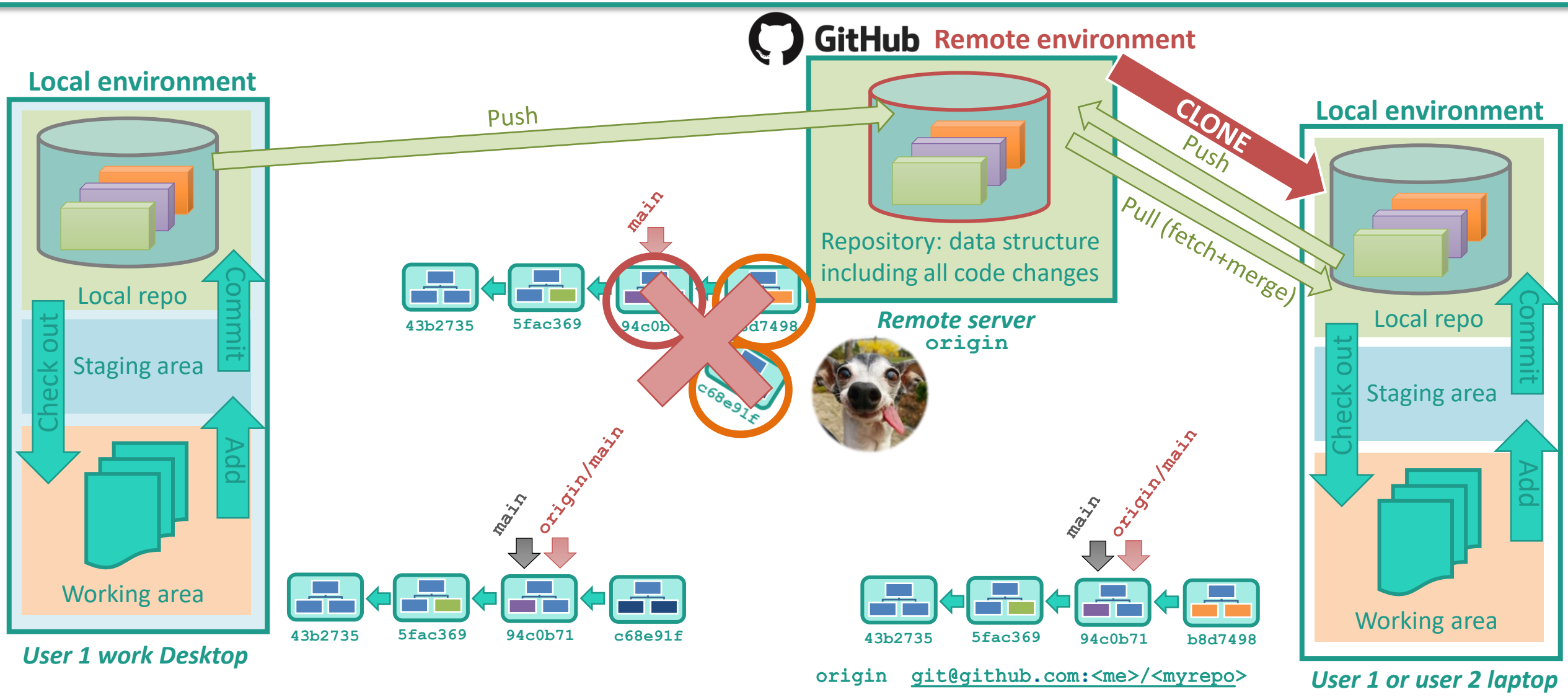
→ both branches have updates from a common ancestor

→ no fast-forward merge possible

What does that mean for using git ?

→ always make sure to have the latest version of the `main` branch before adding changes (otherwise problem such as impossibility to push to remote repo)

GIT BRANCHES – TRACKING REMOTE BRANCHES



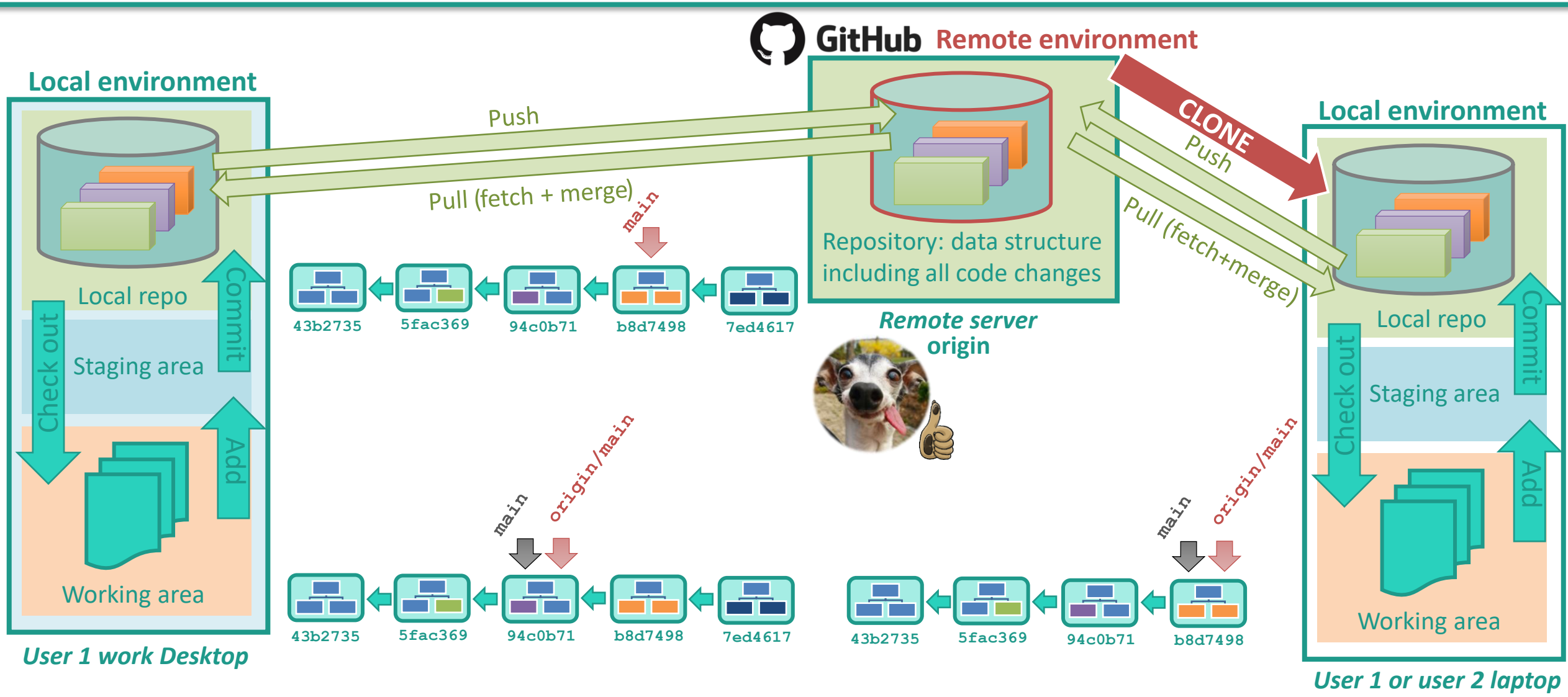

```
git remote add origin git@github.com:<me>/<myrepo>.git
git push -u origin main
```

HNP METHODS & DATA

campus biotech

Fondation Campus Biotech Geneva

GIT BRANCHES – TRACKING REMOTE BRANCHES




```
git fetch origin main
git merge origin/main
```
same as `git pull origin main`

```
git add .
git commit -m "add feature"
```
`git push origin main`

GIT BRANCHES – TRACKING REMOTE BRANCHES

How to automatically track branches on remote ?

If no tracking, always have to do:

- For pushing changes:
  ```
  git checkout topic
  git push origin topic
  ```
- For pulling changes
  ```
  git checkout topic
  git pull origin topic
  ```

+ no information with `git status` to know if you are ahead or behind remote branches

Automated solution

```
git config --global push.default simple
```

Now the first time you push a branch, use

```
git push -u origin topic
```

Then to push and pull changes on the active branch simply do:

```
git push
git pull
```

To get a remote branch not yet existing locally (i.e. it was not pushed from your local repo):

```
git checkout <branch_name>
```

To show all local branches and check if they associated to a remote tracking branch use:

`git branch -vv` (`vv` is to have a "very verbose" output, i.e. lots of details)

have remote tracking branches

```
* bugfix 490d988 [origin/bugfix] correct bad bug
  main   edacdb5 [origin/main] improve user interface
  topic  490d988 add useful feature
```

does not have a remote tracking branch

(need `git push -u origin topic` to track it on GitHub)

To show all local and remote branches use:

```
git branch -avv
```

```
* bugfix 490d988 [origin/bugfix] correct bad bug
  main   edacdb5 [origin/main] improve prog
  topic  490d988 add useful feature
  remotes/origin/bugfix 490d988 correct bad bug
  remotes/origin/main   edacdb5 improve prog
  remotes/origin/testf  a952e6c add test function
```

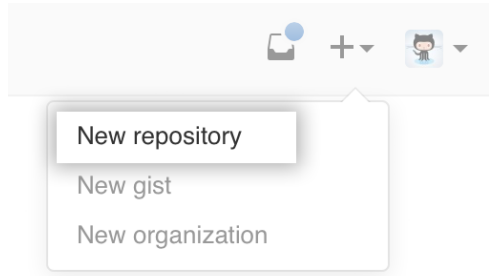
The remote branch `testf` does note exist locally. To work on it locally use `git checkout testf`

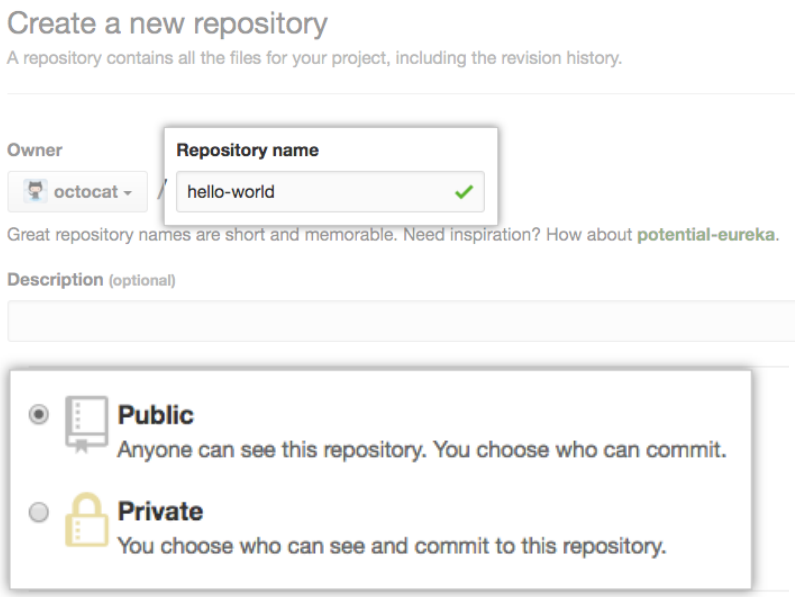
GIT LAB 2 – PART 1

You work on a program to automatically greet some Star Wars characters. You already ran `git init` in the directory `sw_greeter` for all your files to be tracked by git. Now you want to put your code on GitHub to work on it from any computer.

Create a public GitHub repository for your Star Wars greeter program

- On GitHub, click on the "+" symbol on the top right and choose "New repository"
- Choose a name for the new repository, e.g. `starwars_greeter`
- Set the project to "Public"
- Do not initialize the repository with any files since the project will be pushed to GitHub
- Click on the "Create Repository" button





Once you created a GitHub repo named `starwars_greeter`, keep for later its GitHub server location (`git@github.com:<user>/starwars_greeter.git`)

GIT LAB 2 – PART 2

You work on a program to automatically greet some Star Wars characters. You already ran `git init` in the directory `sw_greeter` for all your files to be tracked by git. Now you want to put your code on GitHub to work on it from any computer.

Upload your local computer code to the GitHub repository you created

- Go inside the directory `laptop1/sw_greeter` in your home dir
- Create the remote (i.e the alias) `origin` to point to the remote repository location you noted in part 1
- Check your alias was saved by git by listing your remote aliases
- Rename your current branch (`master`) to `main`
- Push the `main` branch to GitHub so that to track the remote branch locally (tip: use the `-u` flag in the push command)
- Double check it worked by listing your local branches (you should see the tracked remote branch `origin/main`)
- Verify you can use `git pull` and `git push` without arguments as git now knows which remote branches to pull from and push to

```
cd DIR # change to directory DIR (i.e. go inside DIR)
 git remote add ALIAS git@github.com:<user>/<repo>.git
 # set alias ALIAS to a GitHub remote repository location
git remote -v # list all remotes
git branch -M main # rename current branch to main
git push -u ALIAS BRANCH
# push to remote repo ALIAS the branch BRANCH
# and track the remote BRANCH locally (-u option)
git branch -vv # list local branches in very verbose mode
git push # push the current branch to Github
git push ALIAS BRANCH # push to repo ALIAS the branch
BRANCH (only required for untracked remote branches)
git pull # pull from the Github repo the branch having the
         # same name as the current active branch
git pull ALIAS BRANCH # pull from the repo ALIAS the branch
                      # BRANCH (only for untracked branch)
```

GIT LAB 2 – PART 3

Before going to go continuing working on another computer, you would like to fix an error in the crew member names.

- Create a branch `bugfix` and activate that branch
- Modify `hello` in VS Code to correct the name of the first crew member. (File→Open Folder, find `sw_greeter`). Stage, then commit this change.
- Create a new commit to correct the second crew member name.
- Create a final commit to correct the third crew member name.
- Print on the terminal the commit history to check all is fine.

You now have to rush to a different place, but thanks to GitHub you will be able to clone the repo on your other PC

- Change to the directory `laptop2` in your home directory
- Clone your GitHub repository inside the `laptop2` directory using the remote repository GitHub location you previously noted
- Go inside the `starwars_greeter` directory which appeared after cloning
- Modify the script `hello` so that to print "Hi everyone" at the very end.
- Stage, then commit the change you just made.

```
git branch BRANCH # create the branch BRANCH
git checkout BRANCH # activate the branch BRANCH
Note: the two commands above can be done in a single
command with "git checkout -b BRANCH"
For VS Code, use your own internet browser at
<the IP you received>:8080 (password: braincode!)
git status # give git status of local environment
git add FILE # stage FILE
git add . # stage all eligible files
git commit # commit staged files to local repo
           # message will be written via an editor
git commit -m MESSAGE # commit staged files to local
                      # repo with message MESSAGE
git log --oneline # get concise commit history
cd DIR # change to directory DIR (i.e. go inside DIR)
git clone git@github.com:<user>/<repo>.git # clone
git push # push the current branch to Github
```

Your day now ended. You need to push all laptop 2 changes to Github so that they are available from any computer.

- Check current status with `git status`
- Push the changes made to branch `main` to your Github repo

HNP METHODS & DATA

In VS Code use "File → Open Folder" and choose `laptop1/sw_greeter` Then click "OK"

campus biotech

Fondation Campus Biotech Geneva

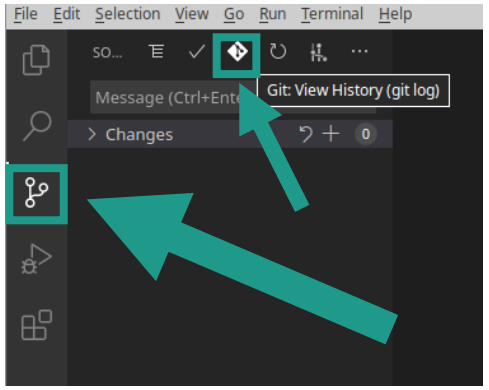
GIT LAB 2 – PART 4

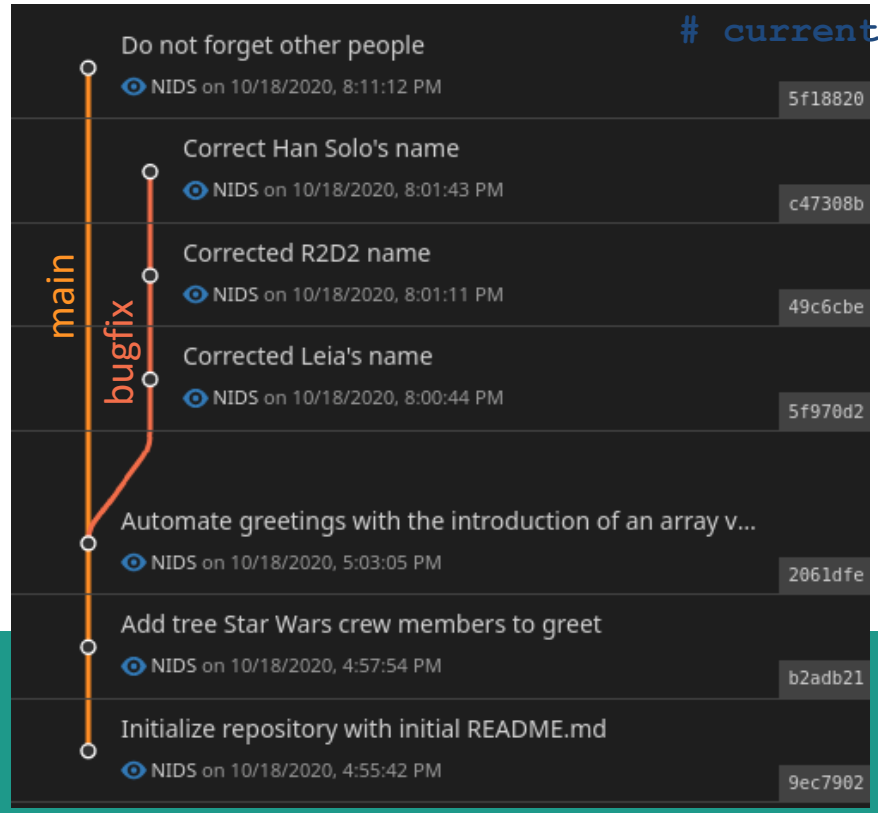
At this point you realize that while GitHub has the latest changes from laptop 2, it does not have the `bugfix` branch from laptop 1. You are back at laptop 1 today and decide to push to Github the `bugfix` branch you worked on yesterday

- Change to the directory `laptop1/sw_greeter` in your home directory
- Double check you are on the branch `main`, if not activate it
- Check git status to print the current status
- Update your remote tracking branches, then check git status
- Merge the latest changes on the remote `main` branch locally
- Check git status again
- Visualize graph history in Visual Studio Code

```
cd DIR # change to directory DIR (i.e. go inside DIR)
git branch # list branches (* shows the active branch)
git checkout BRANCH # activate the branch BRANCH
git status # give git status of local environment
git fetch # update all remote tracking branches
git pull # fetch and merge remote branch
git merge ALIAS/BRANCH # merge from remote repo ALIAS
                       # the branch BRANCH in the
                       # current active branch
```

Use "File → Open Folder" and choose the `laptop1/sw_greeter` folder





Merging `bugfix` with `main` now looks problematic...

HNP METHODS & DATA

campus biotech

Fondation Campus Biotech Geneva

COURSE SUPPORT

SLACK (iords2021.slack.com)

- Course main channel: `#general`
- Topic channels: `#linux`, `#linux-capstone`, `#git`, `#python`, `#full-example`, `#machine-learning`

→ Check regularly for course info (esp. pinned items)

→ Do not hesitate to ask questions (please reply "in thread")

1-to-1 OFFICE HOURS for course questions:

- 20-min slots every Friday morning between 9AM and 11AM

→ Book a time slot here: https://tinyurl.com/IORDS-office-hours

→ Do not hesitate to ask any kind of question, this is a beginner course !

EMAIL: methods@fcbg.ch ← Please whitelist!

Delete the SSH key you added to your GitHub repo during the lecture !

HNP METHODS & DATA

campus biotech Fondation Campus Biotech Geneva


Thank You!
Michael Dayan: methods@fcbg.ch
H N P
METHODS & DATA
campus biotech
Fondation Campus Biotech Geneva