Working with Git and Github

Why Git?

- When working with document we usually develop an *ad hoc* versioning, for example, when editing a paper, we could save different edits with different dates: my_paper_16apr2023, my_paper_20apr2023, my_paper_pi_edited, my_paper_final, etc.

- This system may become cumbersome and unreliable when working on large projects, like developing a body documents, creating software, or analyzing a large bioinformatics project.

- Such projects require tracking modifications for multiple files and multiple version of these files.

- It is difficult to share large project with your collaborators when using an *ad hoc* system

- It is also almost impossible for multiple team members to collaborate on the same project without automated and well-defined **versioning system.**

- Software engineers created **version control systems (VCS)** to manage different options of collaboratively developed code**.**

Why Git?

- Git is the most popular VCS used by 70% developers in the world

- Git was originally developed to facilitate collaboration of thousands of programmers working on creating of Linux kernel

- **Github** https://github.com/ is a massive online repository of code linked to **git**

- What can git do for you?

→ Create a snapshot of a project at the points the documents or code where modified and revert bask to these snapshots
→ Get line-by-line differences in code between different version
→ Git is essential part of proper documentation that can be shared with collaborators and public
→ Stay up to date with documentation or code development
→ Keeps software or documents organized and available after people leave
→ Allow multiple people to participate in development in a well-organized manner

Install and start using Git

Install git

```
$ git -h # check if git is installed, your Linux distro most likely has it pre-installed
```

If there is no git on the system, we can install it as follows

```
$ sudo apt-get install git
```

Create a directory **test_repo/** we will use to practice git

```
$ mkdir test_repo
$ cd test_repo
```

Configure user and email

```
$ git config --global user.name “Some Name”
$ git config --global “User email”
```

Enable terminal colors

```
$ git config --global color.ui true
```

Using Git

Git only manages the files in a directory initialized as a repository

We can initialize the repository in the existing directory, or we can download an existing repository from Github using **git clone**

Let's initialize git repository in **test_repo/** directory

```
$ git init
```

Copy some fastq files we previously created from **scripting/** directory

```
$ cp ../scripting/*_R*.fastq .
$ ls -la # note .git/ hidden directory that contains all git data
```

Create some files we will be working with

```
$ touch README
$ touch run_fastqc.sh
```

Now git is aware of all the files in the directory, but it's not tracking them

```
$ git status
```

Using Git

Tracked files – git is aware of these files, and they are added to the repository

Untracked files – git is aware of them, but they are not added to the repository

We did not add any files to the repository yet, so all of them are untracked

In generally we don't want to track data files (fastq, sam, bam, vcf, etc)

We will add the fastq files to ignore list
$ touch .gitignore # create hidden file .gitignore

Add fastq files to .gitignore
$ echo *.fastq .gitignore

Check git status again
$ git status

Using Git

Add files to **staging environment**

```
$ git add README
$ git add run_fastqc.sh
$ git status
```

Staged files are ready to be **committed** to the repository

We can add more than one file as follows **git add –all**

Let's modify the files and make first commits

Add some content to README

```
$ echo 'This is a fake README file for a test repo' > README
```

Commit the change

```
$ git commit -m 'Added content to README file' # we must always add a message with every commit
```

Using Git

Create a script that will run fastQC on all of the fastq files

Add a shebang line **#! /bin/bash** to run_fastqc.sh

```
$ git commit -m 'added shebang line’
```

Add a loop construct to run_fastqc.sh:

```
for file in ./*.fastq
do

done
```

Commit the change

```
$ git commit -m 'added loop construct'
```

Using Git

Add fastqc line to run_fastqc.sh

```
$ git commit -m 'added fastqc command line’
$ git status
```

Run the script

```
$ git status
```

Ignore fastqc output

```
$ echo ‘*.html’ > .gitignore
$ echo ‘*.zip’ > .gitignore
$ git status
```

The changes are not yet committed at this point, we need to **stage** files again to commit changes

```
$ git add README
$ git add run_fastqc.sh
```

Using Git

Add fastqc line to run_fastqc.sh

See status is short form

$ git status --short

Short status flags are:

- ?? - Untracked files
- A - Files added to stage
- M - Modified files
- D - Deleted files

Update and stage at the same time

```
git commit -a -m "Update file with a new line"
```

View commit log

$ git log

Using Git

View help

```
$ git commit –help
$ git help --all
```

We will not go over **branches** and **merging**

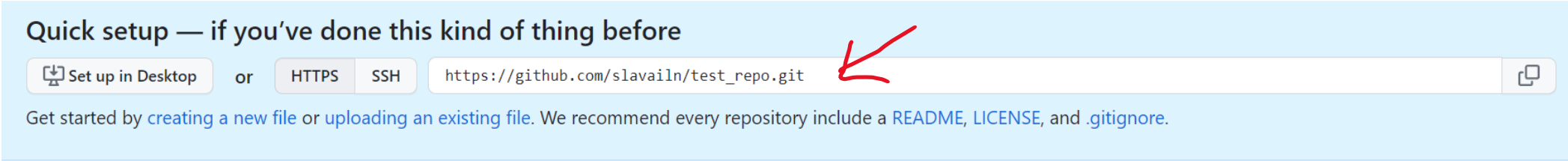
We will store our test_repo/ in remote **Github** repository

Go to https://github.com/ and create your account

Log in into your account and create new repository called **test_repo**

Using Git

Now we will **push** our local repository to Github



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

```
echo "# test_repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/slavailn/test_repo.git
git push -u origin main
```

Copy the URL pointed by a red arrow

Now add a remote github repository with the specified URL to our local repo

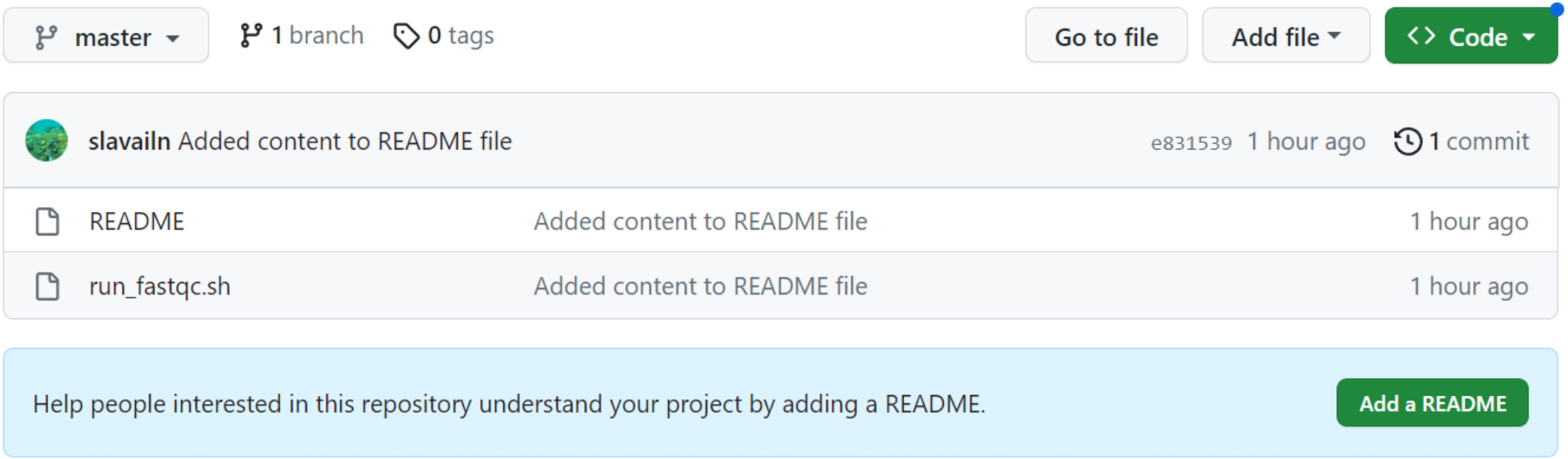
$ git remote add origin https://github.com/slavailn/test_repo.git

Using Git

Git will ask you for username and password

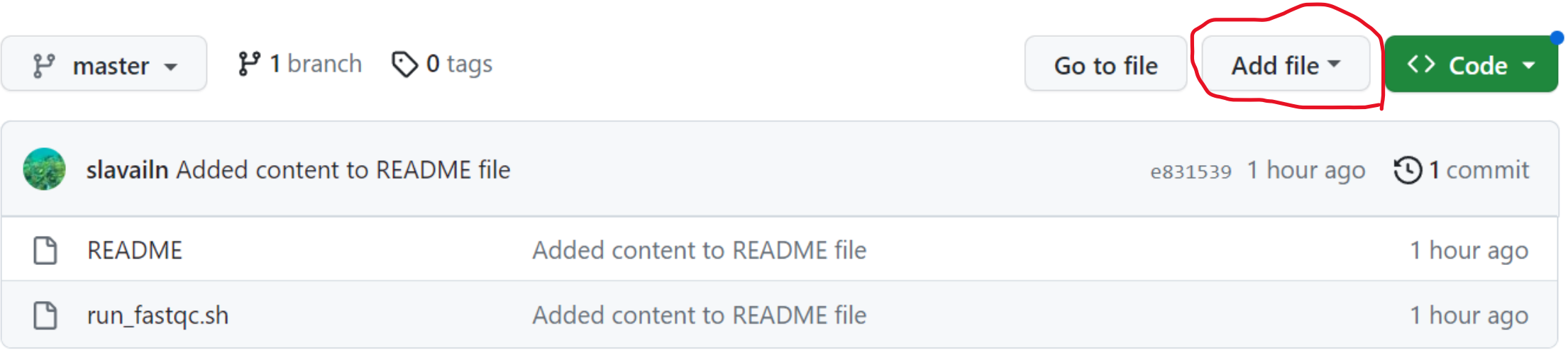
Github disables password access and now uses tokens

Enter the token instead of the password, the token can be found in **profile → settings → developers settings → personal access tokens**



Using Github

Add and edit files directly on github



Click on any of the files, the edit button appears