Day 3, part 3. Version control and collaboration: Git and GitHub

Digital Skills for Research

12 May 2022

Contents

Familiarity with Git/GitHub is required by employers in translation industry: e.g. from a job advert for a project/product manager at Smartling:

> Do you have Git experience? Please give us 3 examples where you used GitHub.
> What do you find more challenging with GitHub?

This session offers a practical and systematic explanation of core functionality of Git and GitHub.

1 Git: Keeping track of changes

For software developers, Git allows **parallel maintenance of older and newer versions** of the product. Each released version is stored as a separate branch of the project.
For a researcher, Git is a way to **collaborate and keep track of contributions** from participants as well as **making your work available** to other researchers.

Important features:

- by Linus Torvalds (2005), a Finnish-American software engineer, who is the main developer of the Linux kernel. First Linux prototype by released in 1991 when Torvalds was 22.
- OSS (open-source soft)
- de facto standard in academia and industry
- Distributed Version Control System (full history of changes is kept locally and remotely)

Figure 1: Linus Torvalds

1.1 Is Git installed in your OS?

- By default, Git is installed on Linux and macOS computers as a command line option.
- How to install Git on Windows

1.2 HowTo

Start tracking a local project folder

1. Go into the `demo` directory containing the project in the terminal ("Open in terminal/command line/cmd")
2. create README**.md** and **.**gitignore files in the root of the project
3. `git init`
4. `git add README.md .gitignore` to add **relevant** files
5. `git commit -m 'first commit'`

Each time you want to update the history of changes in the project, pass `git add` and `git commit` commands (don't forget a descriptive but short message!)

Boom! If you look at hidden files in the tracked folder, you will see a .git/

2 GitHub: Local and remote, push and pull, auth

2.1 Connect to remote URL at GitHub server

Git associates a remote URL with a name, and your default remote is usually called origin.
You can only push to two types of URL addresses:

- **(default)**: An HTTPS URL like https://[hostname]/user/repo.git
- (not discussed here) An SSH URL, like git@[hostname]:user/repo.git

Connect a tracked local project to GitHub

- Log in to your GitHub account
- Click the new repository button in the top-right
- Click the "Create repository" button
- It is easier to give it the same name as the name of folder to connect
- Decide whether you are ready to go public (default!) with your project
- run the following commands from the local tracked folder in terminal

```
git remote add origin https://github.com/kunilovskaya/demo.git
git push -u origin master
```

master means main branch of the project.
origin master is main branch of the remote repo (on the server).

2.2 Auth: username and password/access token/SSH key [and a passphrase])

Select username wisely! e.g. ssharoff, TharinduDR, ltgoslo, torvalds

How passwords and tokens are used:

- **website password** is required to create/delete repositories and add changes to them **in the browser**
- access token is needed to push local content to a remote repo with HTTPS URL **from the command line**
- SSH key [or a passphrase to it] is needed to push content to a remote repo with SSH URL

See detailed and official HowTo

NB! Password-based authentication for Git has been removed on August 13, 2021.

Permanently authenticating with Git repositories:
Locally stored access keys apply to all projects.

1. create a config file

```
git config credential.helper store
git push https://github.com/kunilovskaya/dskills_workshop.git
```

2. give your username and the new key to save it remotely
3. set a longer cache timeout than the default 15 mins (e.g. 2 hours or 5 days=7200 min) to avoid accessing the txt file with the unencrypted password stored on your local disk each time you push

```
git config --global credential.helper 'cache --timeout 7200'
```

Instructions here

2.3 Typical workflow

1. publish local changes in files (or `git rm -r folder-name`) to the website
   - `git add my_file1.py my_file2.py`
   - `git commit -m 'added export to tsv'`
   - `git push` (if you have set up automatic authentification, you will not be asked for username and password)
2. get changed made on the remote
   (if you know that someone might have pusshed to your repo, pull changes first to avoid conflicts)
   - `git pull`

2.4 PyCharm-Git(Hub) intergration

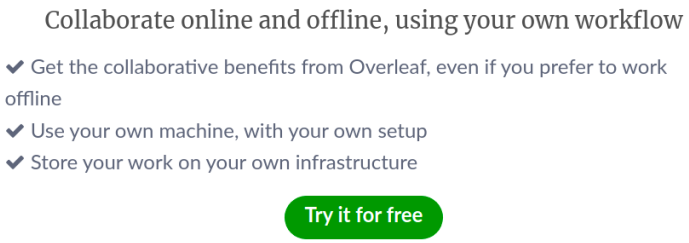
PyCharm 2021.3 (Professional Edition)

File → Settings → Version control → Github

- Log in via Github
- Log in with Token
- (a tickbox) clone git repo using ssh

2.5 Overleaf-Git(Hub) intergration

GitHub Sync is a premium feature



3 Markdown and arranging repos

1. don't push everything: ~~git add .~~ even if .gitignore exists
2. don't push data and output (mind 2MB limit for one file upload)
3. create a clear description in README (using markdown); see my example

markdown (.md) is a markup language for creating formatted text using a plain-text editor.
One useful Markdown Cheatsheet

How to format:

- headings
- emphasis
- line breaks
- lists
- links
- tables
- code listings

````
```python
        s = "Python syntax highlighting"
        print(s)
```
````

- horizontal rule

Task 3-8. Start a private repo and add 'kunilovskaya' as a collaborator

- set up Git tracking for a local folder (with a README);
- push it to GitHub;
- invite a collaborator;
- make changes on the server (in browser)
- pull remote changers
- make local changes and push them (don't forget to refresh the page!)