

Git Workshop

by Laura Żuchowska 🙌

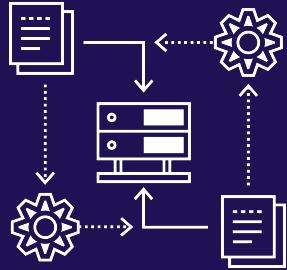


Agenda

Hello 🙋

1. What's Git even? 🤔
2. Basics 🏫
3. Pull Request how-to 📜
4. Practice! 😱





What's Git even?

How does git work?

Collaborations on trees

- has a tree-like structure
- every developer has a local copy on their machine
- you can change your local history
- can't handle big files

Branches

git log --all --decorate --oneline --graph 🎨

```
  |/
  | * aa65ac5 (origin/prefect_extract_data) ✘ Added ability to automatic change date_range if flows have failed in t
e last days
  | * 0d681bf ✘ Added class arguments and code to run() method
  | * 9ea987a ✘ Added PrefectExtract task to SupermetricsToADLS
  | * 8202644 ✘ Added new task for Prefect
  |
  |/
  * e37e4fa Merge pull request #261 from dyvenia/create_dirs_in_to_csv
  |
  | 5ebabc4 (origin/create_dirs_in_to_csv) ✘ Removed unnecessary bits
  | b256777 ✘ `df_to_csv` now creates dirs if they don't exist
  |
  * c4512b8 Merge pull request #256 from dyvenia/fix_utils_docstring
  |
  * d679697 ✘ Fixed utlis docstring
  |
  * 2af7b64 Merge pull request #255 from dyvenia/pin_prefect
  |
  | 260a609 (origin/pin_prefect) ✘ Added entry on Prefect version pinning
  | 232ddd2 ✘ Pinned Prefect version
  |
  |/
  | * 21fcfd8f (angelika233/kv_c4c_angelika, kv_c4c_angelika) ✘ Fixed kv c4c when credentials are None
  | * e838407 ✘ Removed unnecessary code
  | * e08d23b (0r) ✘ Added key vault support c4c
  | * befcfb1 (origin/dependabot/pip/mkdocs-material-8.1.10) Bump mkdocs-material from 8.0.1 to 8.1.10
```

Common misconceptions

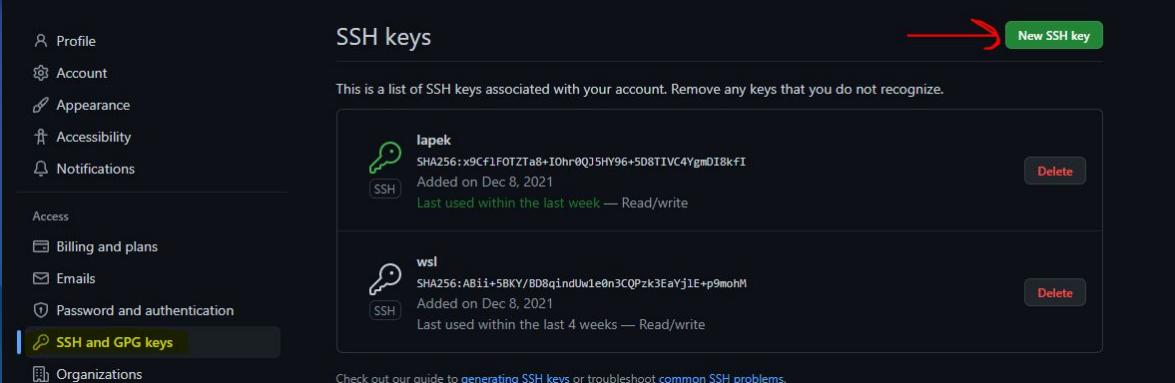
👀 Fear not 👀

-
1. You will break the git repository with one push
 2. You only commit when the feature is production-ready
 3. Merge/rebasing is scary

SSH

ssh-keygen -t rsa -b 4096 -C "tag-of-your-ssh"

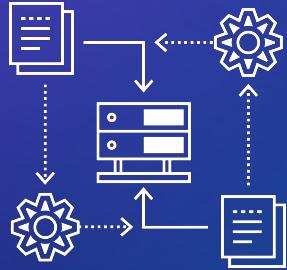
1. Create an SSH key
2. Save it to a file
3. Write passphrase
4. ssh-add -K /path/to/your/file
5. Add SSH on git



The screenshot shows the GitHub 'SSH keys' page. On the left, there's a sidebar with links: Profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, **SSH and GPG keys** (which is highlighted with a yellow bar), and Organizations. The main area is titled 'SSH keys' and contains the following information:

- lapek**
SHA256:x9Cf1F0TZTa8+IOhr0QJ5HY96+5D8TIVC4YgmDI8kFI
Added on Dec 8, 2021
Last used within the last week — Read/write
[Delete](#)
- wsl**
SHA256:Abii+5BKY/BD8qindUw1e0n3CQPzk3EaYj1E+p9mohM
Added on Dec 8, 2021
Last used within the last 4 weeks — Read/write
[Delete](#)

At the bottom, there's a note: 'Check out our guide to [generating SSH keys](#) or troubleshoot common SSH problems.'



Basics

Basic commands

**add**

```
git add [file/directory]
```

Stage files for commit.

Good practice:
Don't use *git add .*

**commit**

```
git commit -m [message]
```

Commit changes with
message.

**push**

```
git push [from] [to]
```

Push changes to a
branch.

**pull**

```
git pull [to] [from]
```

Pull changes from a
branch.

Basic commands

**rebase**`git rebase [branch]`

Update branch based on changes made.

Use if you're the contributor on official repo.

**merge**`git merge [branch]`

Update branch based on changes made.

Use if you're working on a forked repo.

**checkout**`git checkout [branch]`

Change branch.

Add -b to create a new branch.

Add -d to delete a branch.

**status**`git status`

Show current list of files with changes.

Useful commands

stash

git stash show - show latest stash
git stash list - show list of stashes
git stash - save changes into stash (removes from workspace!)
git stash apply/pop - apply stashed changes (pop removes from stash list)

fetch

git fetch [to][branch] - saves a remote work in your local repo (without making actual changes, so you have to git checkout)

restore

git restore [file] - reverse the changes made on a file

Forking

On github repository page,
click ‘fork’. Clone the forked
repository with:

```
git clone <repository>
```

Add the original repository to
your remote list

```
git remote add <name> <url>
```

**Always pull changes from
dev before working!**

Create a new branch
`git checkout -b <name>`

~do stuff~

Add and commit
`git add <things>`
`git commit -m “message”`

Check for changes (and pull if there are)
`git fetch`
`git pull <remote> <branch>`

Push changes and merge
`git push origin <name>`
`git checkout <where do you want to merge>`
`git merge <branch to merge>`

Internal user

Clone the forked repository with:

```
git clone <repository>
```

No need to add to remote list

Always pull changes from dev before working!

Create a new branch
git checkout -b <name>

~do stuff~

Add and commit
git add <things>
git commit -m "message"

Check for changes (and pull if there are)
git pull <remote> <branch>

Push changes and merge
git checkout <branch>
git rebase <where to>
git push <origin> <branch>

Merge conflicts

Chill, you're not breaking anything

```
C:\Users\Laura\Documents\Work\git_workshop\forkd\git-workshop>git
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (5/5), 1001 bytes | 66.00 KiB/s, done.
From https://github.com/dyvenia/git-workshop
 * branch           main      -> FETCH_HEAD
   548f0af..5fa8e12  main      -> dyvenia/main
Auto-merging code/welcome.py
CONFLICT (content): Merge conflict in code/welcome.py
Automatic merge failed; fix conflicts and then commit the result.
```

Merge conflicts

Open your VSCode

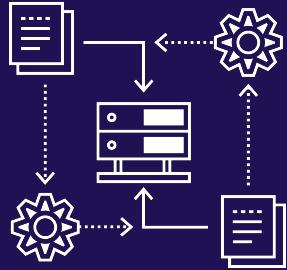
```
        '
You, 3 minutes ago | 2 authors (You and others) | Accept Current Change | Accept Incoming Change | Accept Both Changes | C
<<<<< HEAD (Current Change)
print("Welcome to the git program!")
=====
print("Welcome to the workshop program!")
>>>>> 5fa8e122651f8755ea4cb0fb18bcfe43c1fe3c05 (Incoming Change)
print("I'm not very useful")
```

Merge conflicts

Where am i?

Add, commit, push to your remote, merge again.

```
C:\Users\Laura\Documents\Work\git_workshop\forkd\git-workshop>git status
HEAD detached from 548f0af
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)
```



Pull Request how-to

Never push to main

⚠️ Closed

Izuchowska wants to merge 1 commit into `main` from `improve_flow_of_flows_list_handling` ⚙️

Don't be like this gal

Keep it simple

One PR should implement one functionality

Don't try to push as much as possible - if you notice a bug, have an idea for a new component get a sudden urge to write a test for something not directly connected to what you're working on - leave it for next PR. Open an issue about it.

Keep the amount of commits per PR as low as possible. Reading through 100 commits is not the most pleasurable thing to do. But so is reading one commit with 20.000 lines of code. It's all about the balance in the universe.

Naming conventions

:emoji: Verb + functionality

Always begin with a gitemoji (VSCode extension, each emoji specifies a function).

Use verbs such as “Added” or “Fixed”, capitalized.

Specify the functionality, try to keep it short.

If you get lost - naming conventions are usually written down in the documentation.

	COMMENT	DATE
O	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O	ENABLED CONFIG FILE PARSING	9 HOURS AGO
O	MISC BUGFIXES	5 HOURS AGO
O	CODE ADDITIONS/EDITS	4 HOURS AGO
O	MORE CODE	4 HOURS AGO
O	HERE HAVE CODE	4 HOURS AGO
O	AAAAAAA	3 HOURS AGO
O	ADKFJSLKDFJSDFKLJF	3 HOURS AGO
O	MY HANDS ARE TYPING WORDS	2 HOURS AGO
O	HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Examples

Bad

- Fixed a bug
- 🍴 Added a test for *my-thing*
- anna_commit
- ✨ Added the possibility to parse a letter from a non-latin alphabet - a mandarin Chinese symbol for 'data', I've discovered those by reading a book once mentioned from my father whom...

Better

- 🐛 Fixed a bug in *name* for empty df
- ✓ Added a test for my thing
- 🎨 Updated code structure for *name*
- ✨ Added the possibility to parse a letter from a non-latin alphabet

PRACTISE
TIME

Thanks!

Contact us:

ul. Czyżówka 14/0.3,
30-526 Kraków
Poland

hello@dyvenia.com
www.dyvenia.com

