Python programming

for biologists

02.09.2023

Our python course team

Anton Sidorin
SPbSU

Nadezhda Pavlova
MSU

Alexandra Kasianova
Skoltech

Nikita Vaulin
Skoltech
vaulin@ro.ru, tg @nvaulin

Course structure

Meetings

- 13 lectures - sat 11:45 MSC
- Extra meetings (QA, additional topics) - weekdays evenings

Quizzes

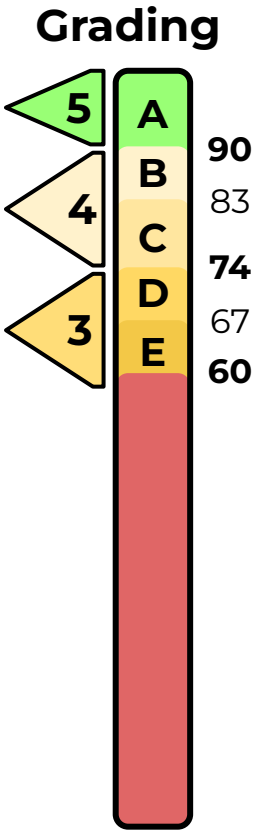
- 6 quizzes (at the beginning of the lectures)
- max 2 points (0: 0%, 1: <50%, 2: >50%)

Homeworks

- 11 homeworks
- Each - max 10 points
- Deadline: next sat 23:59 AOE
- Submissions **after deadline - half of the points**
- There are assignments for extra points

Interview with life coding

- Last week only for top 5 students (who wants)



Course program

	1.	**Intro. Git, GitHub**
Basic python 	2.	Python recap, data types
	3.	Functions
	4.	Modules and libraries
	5.	Files
	6.	IDEs
	7.	Virtual environments
	8.	Regular expressions
Scientific data analysis 	9.	Numpy
	10.	Pandas
	11.	Visualisation
	12.	Statistics
	13.	Discussion

next semester...

Advanced python

- OOP, classes
- Decorators
- Iterators & generators
- Web scraping

Tools development

- Parallel programming
- Profiling, performance
- Open source
- SQL

Sources:

Our

- GitHub course page
- Notion course page
- Stepik Git course

External

- Stepik python course (BI)
- Stepik python course (BEEGEEK)
- Python tutor
- LearningGitBranching
- Hexlet Git course

Soft

- Python 3.11
- Jupyter
- Git
- PyCharm / VSCode

Any questions?

Assignments?

Course program?

Grading?

Future plans?

Lecture 1

Version control systems
Git

02.09.2023

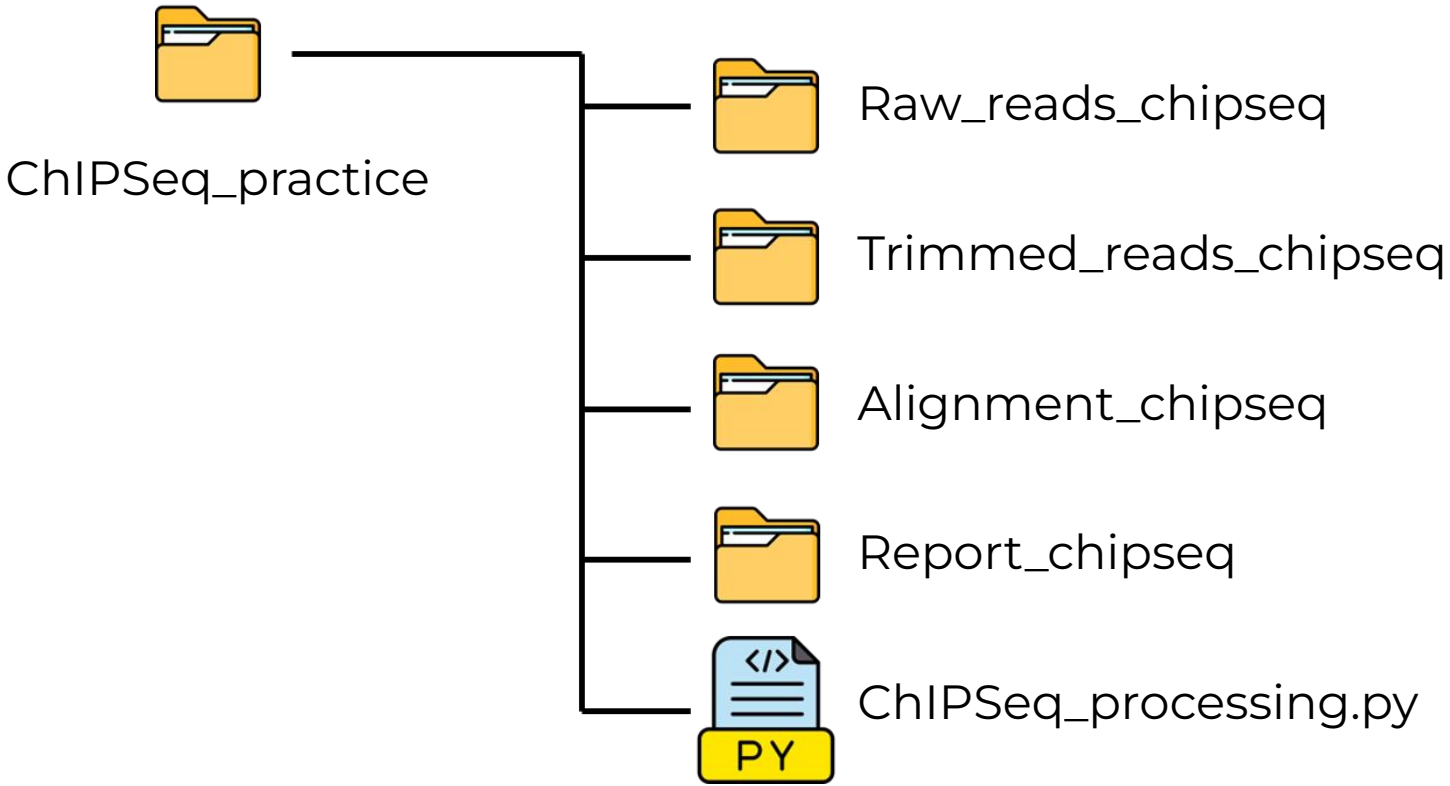
What do you know about Version Control Systems?

What are the VCS?

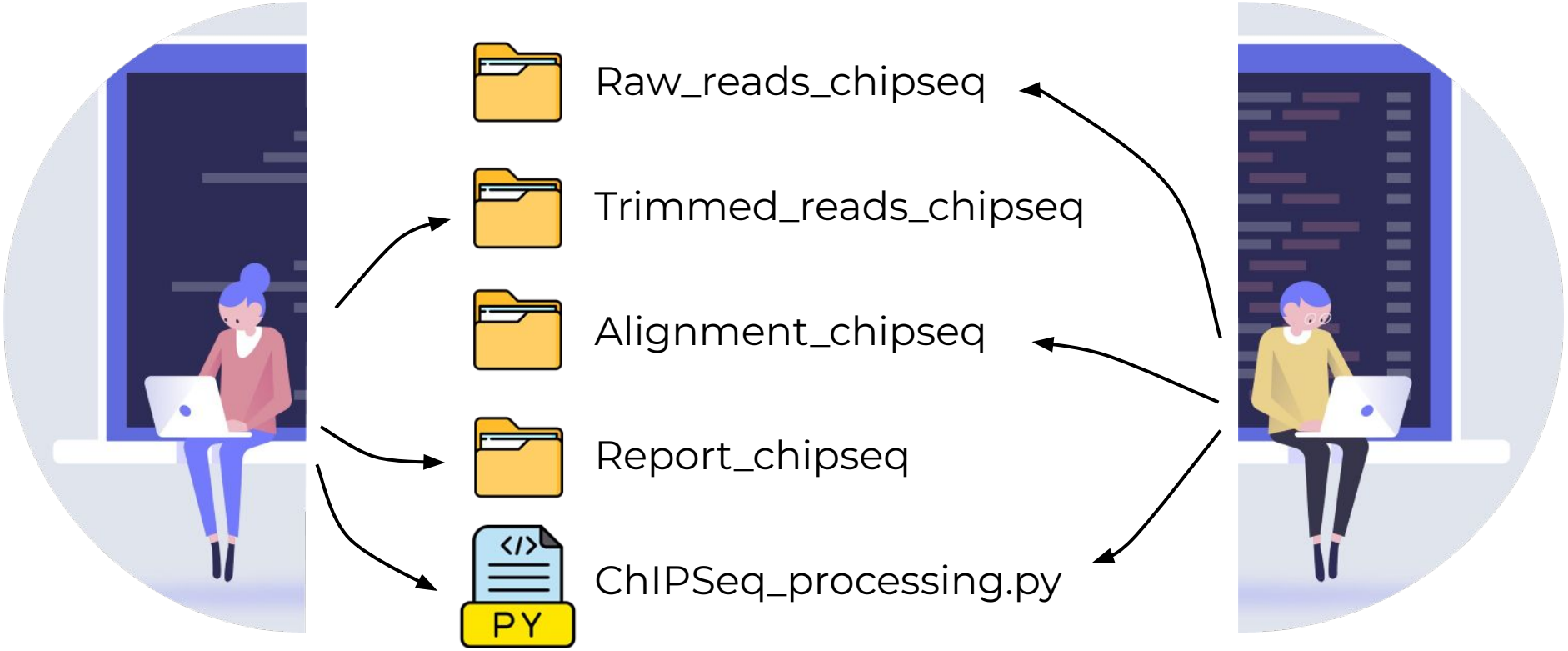
What systems do you know?

Do you use any of them?

Some bio project



Some bio project

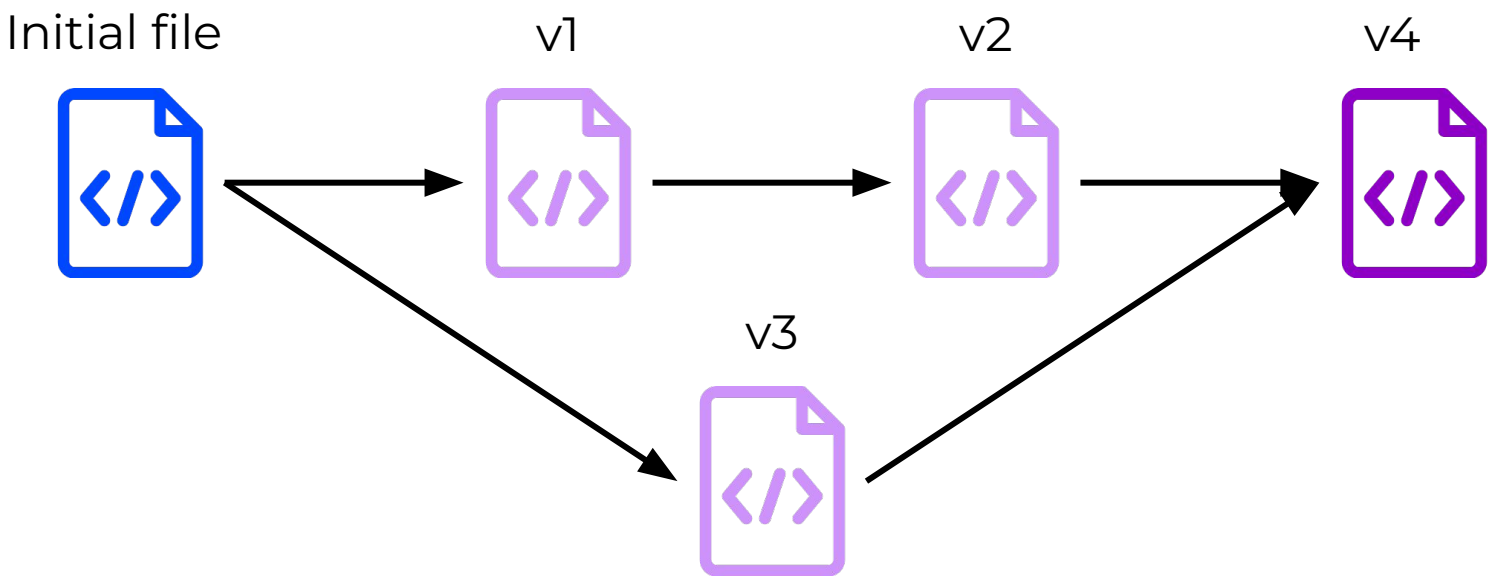


Two main VCS purposes

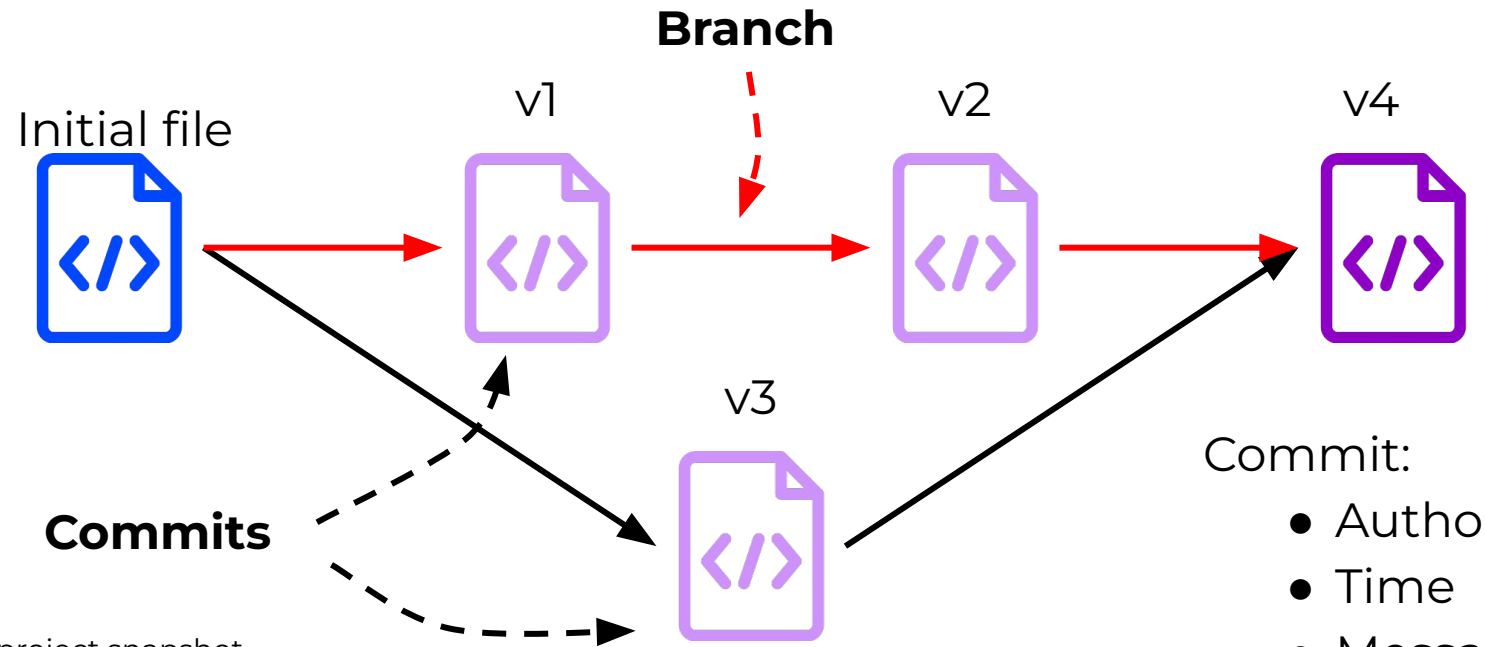
Tracking history

Collaborative development

Version controlling



Version controlling

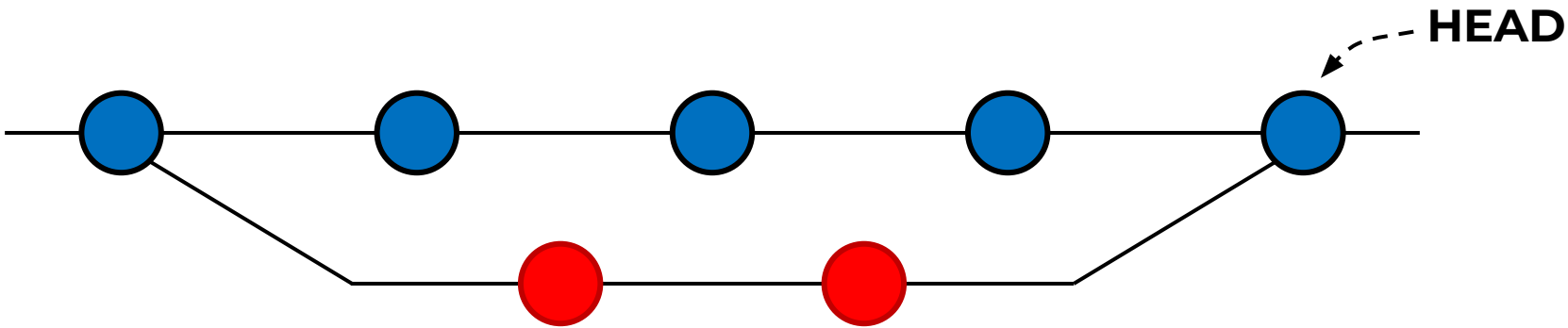


Commit – project snapshot
Branch – series of commits

Commit:

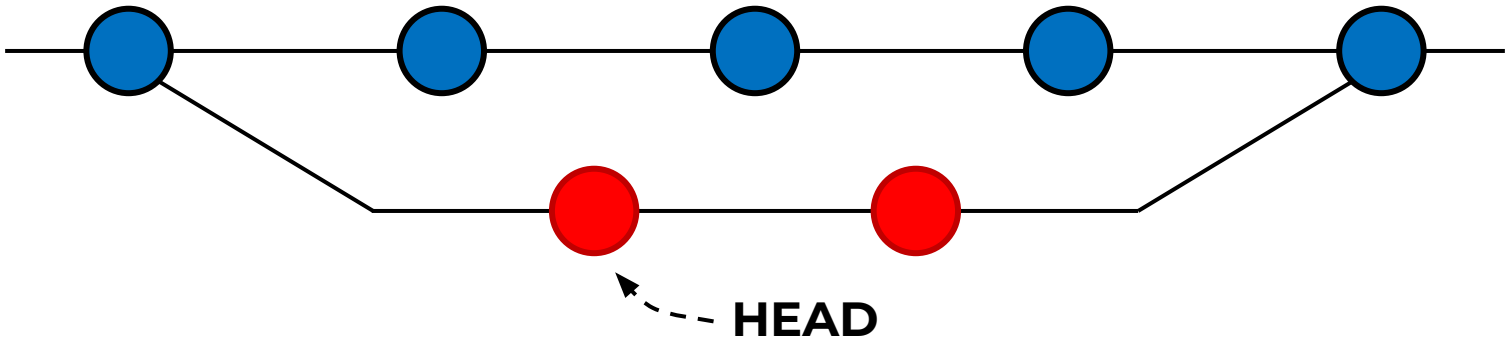
- Author
- Time
- Message
- Hash

Version controlling



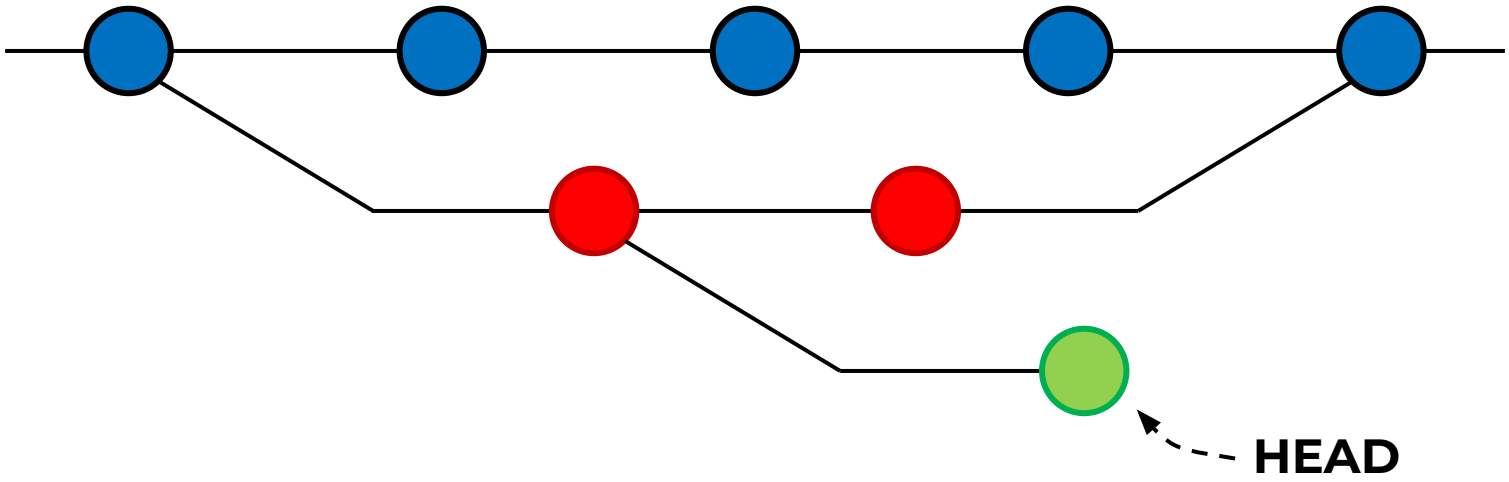
Commit – project snapshot
Branch – series of commits
HEAD – commit you are currently on

Version controlling



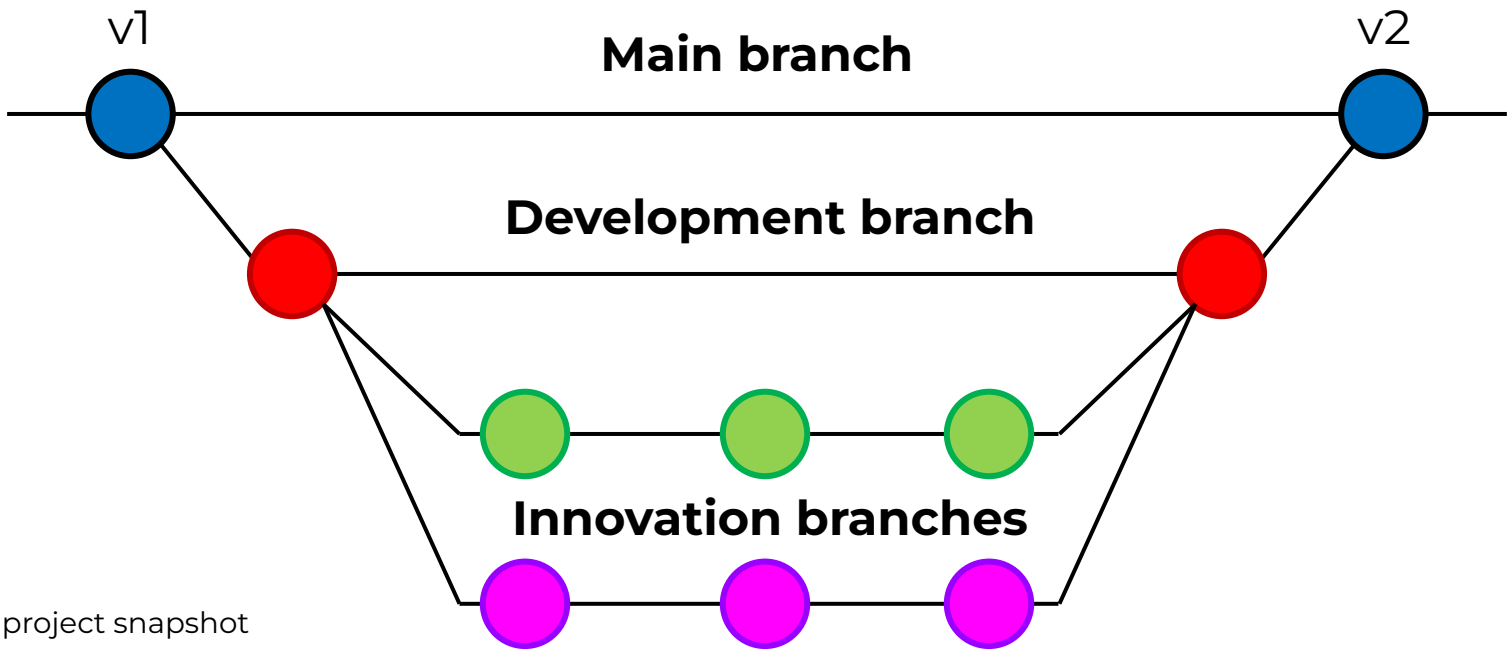
Commit – project snapshot
Branch – series of commits
HEAD – commit you are currently on

Version controlling



Commit – project snapshot
Branch – series of commits
HEAD – commit you are currently on

Ideal project



Commit – project snapshot
Branch – series of commits
HEAD – commit you are currently on

Git practice

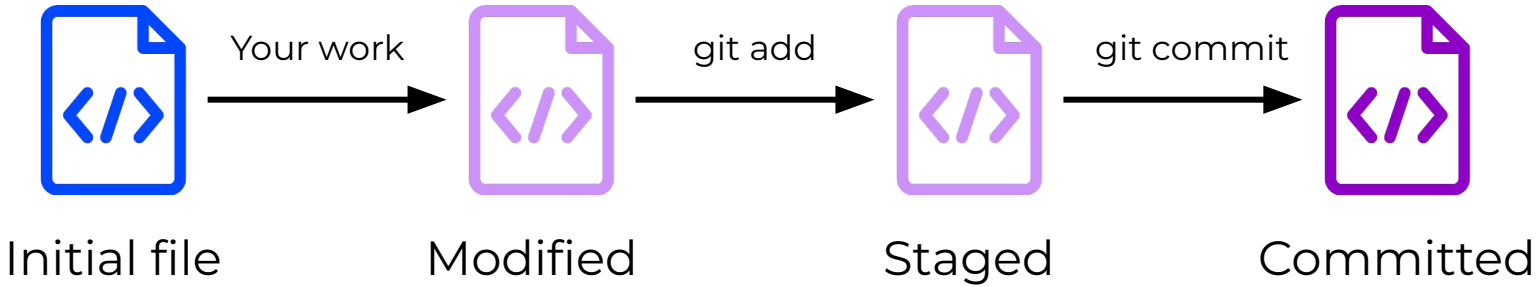
Hands on your keyboard

Starting new project

`mkdir test_project`	Make new empty folder
`cd test_project`	Move into it
`ls -al`	Check it is empty
`git init`	Initialize git repository
`ls -al`	Check once again

What do you see?

File life


Your work
git add
git commit
Initial file
Modified
Staged
Committed

Git practice

Hands on your keyboard

Working with files

`echo '...' > file`	Edit the file
`git add file`	Track the file
`git commit`	Save changes
`git config`	Set up some git things

Commit message rules

- Use **imperative** mood
- Start with **Capital** letter
- Line size **< 50** symbols
- Be **informative**
- Do not end with a period
- Use the body if you need

```
“Add read_fasta function”
```

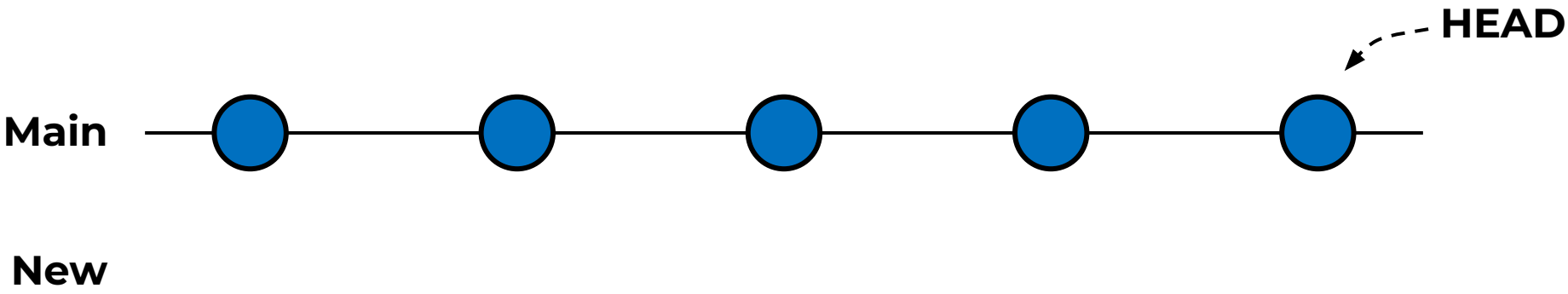
Git practice

Hands on your keyboard

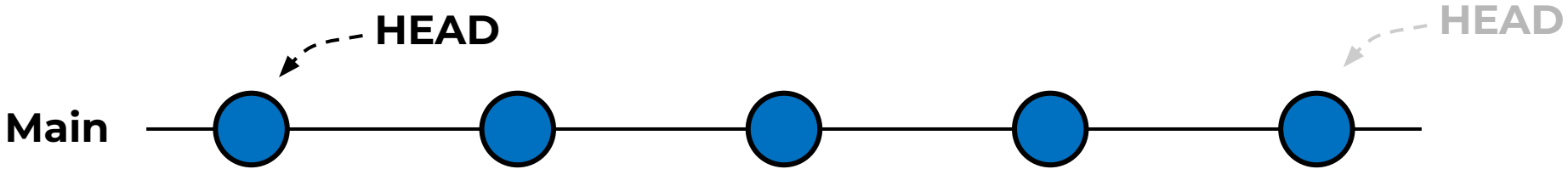
Getting information

`git show`	List of the files
`git status`	List of the files and their statuses
`git log`	Commits history
`git reflog`	List of operations
`git blame`	Who edited the file?

Branching

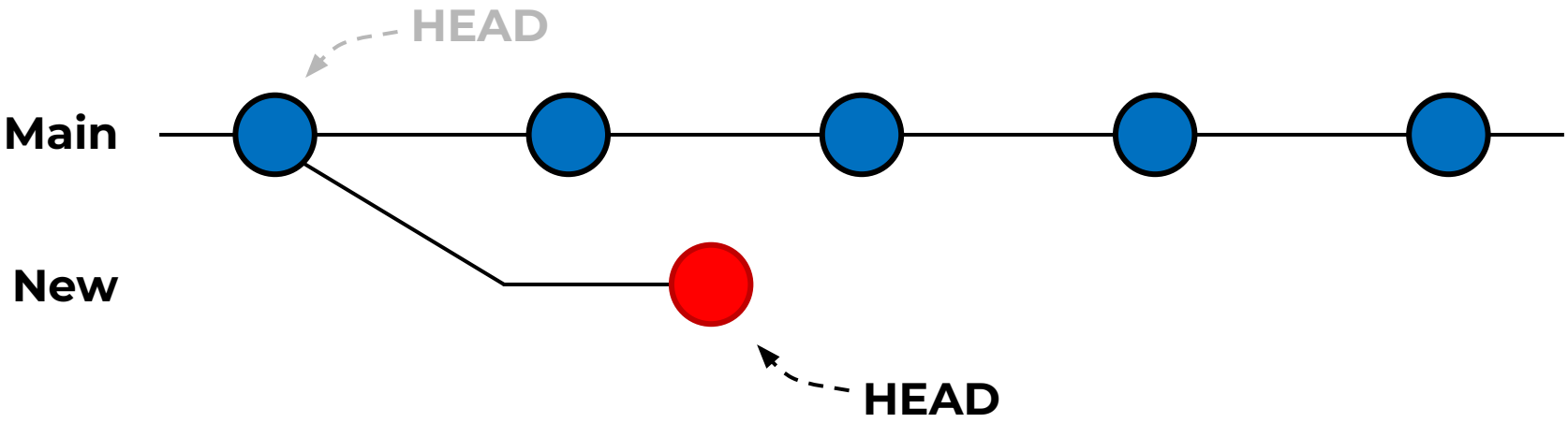


Branching



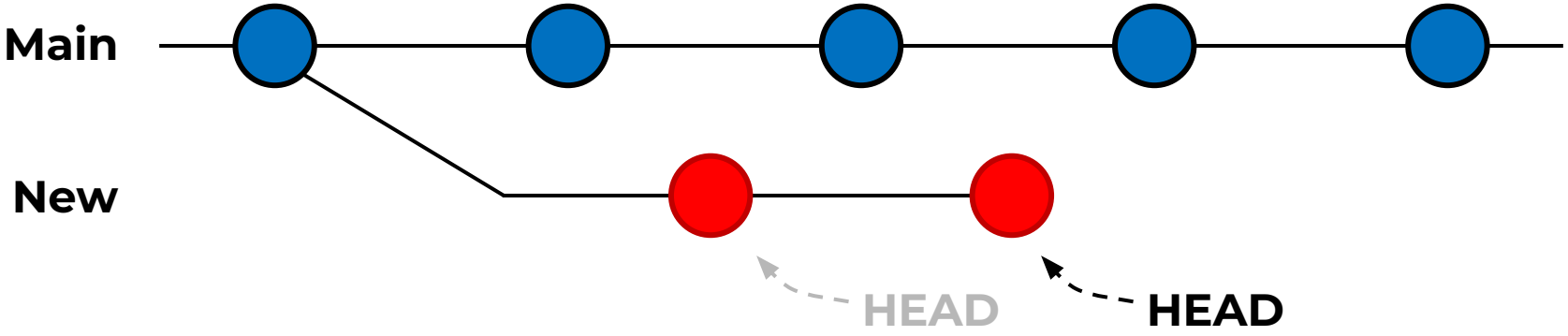

```
git checkout <hash>
```

Branching



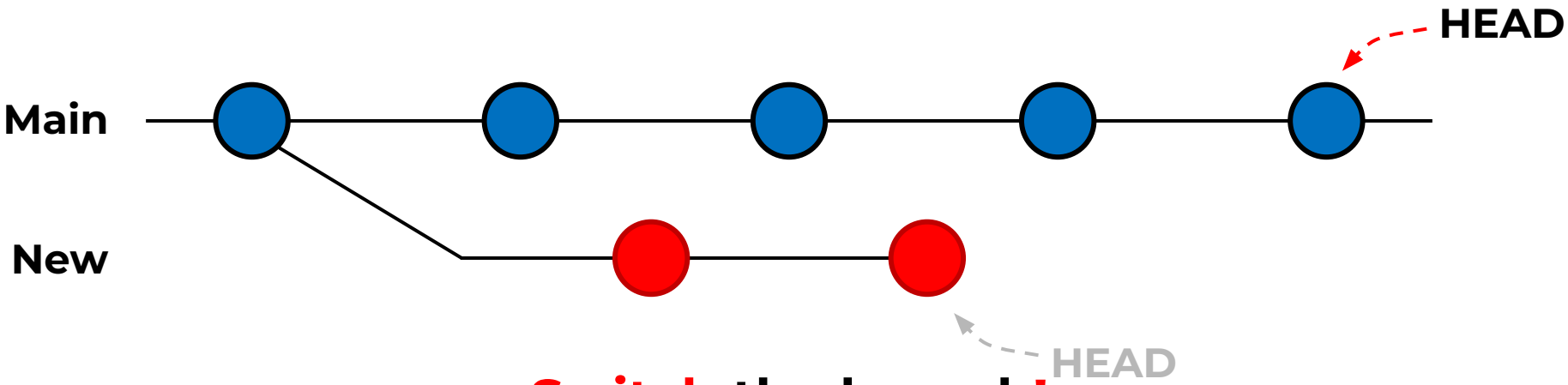

```
git checkout -b <branch name>
```

Branching




```
git add && git commit
```

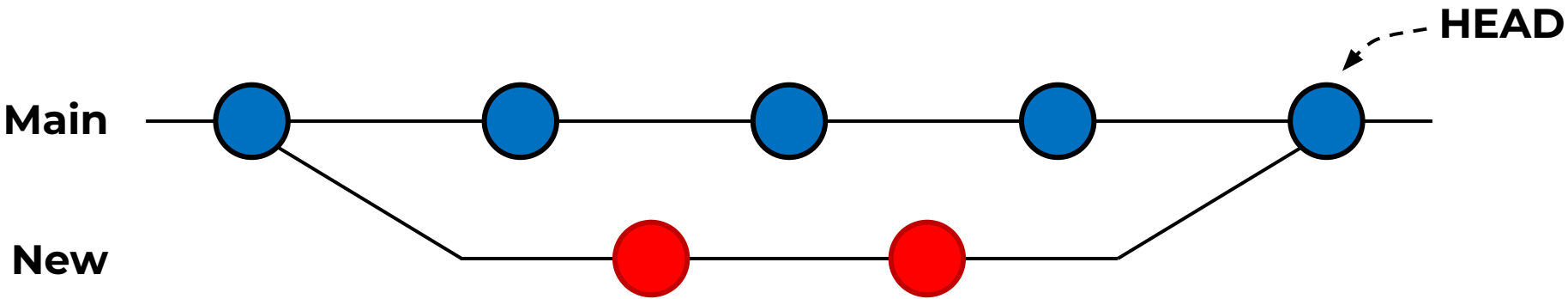
Branching



Switch the branch!

```
git checkout main
```

Branching




```
git merge <branch name>
```

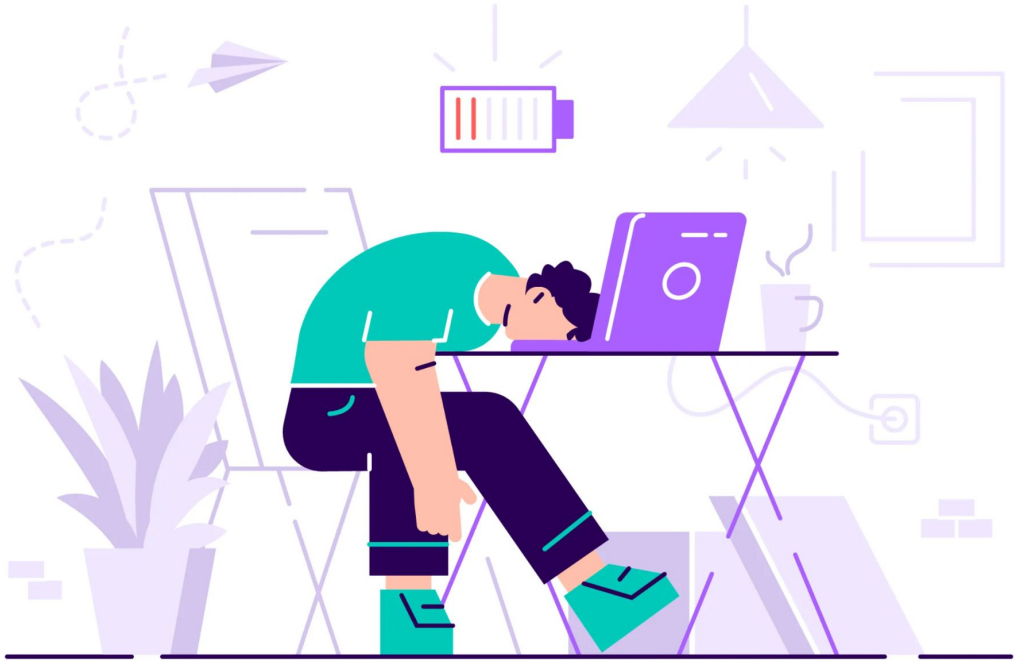
Git practice

Git branching

`git checkout`	Walk on the commits
`git checkout <hash>`	Switch to the commit
`git checkout <branch>`	Switch to the latest commit
`git branch`	List of the branches
`git branch <name>`	Create the branch
`git checkout -b <name>`	Create the branch and move on

`git merge <name>`	Merges the other branch into yours
`git branch -d <name>`	Delete branch

Break!



Any questions?

Assignments?

Course program?

Grading?

Future plans?

Lecture 1

GitHub

02.09.2023

Remote repositories

GitLab

Bitbucket

GitHub

You can use them for:

- Code sharing
- Files sharing (even with no coding)
- Resume
- Look for others repo’s

Connecting GitHub with your PC

Hands on your keyboard

```
ls -al ~/.ssh
```

```
ssh-keygen -t ed25519 -C "your@email.com"
```

```
eval "$(ssh-agent -s)"
```

```
ssh-add ~/.ssh/id_ed25519
```

```
cat ~/.ssh/id_ed25519.pub
```

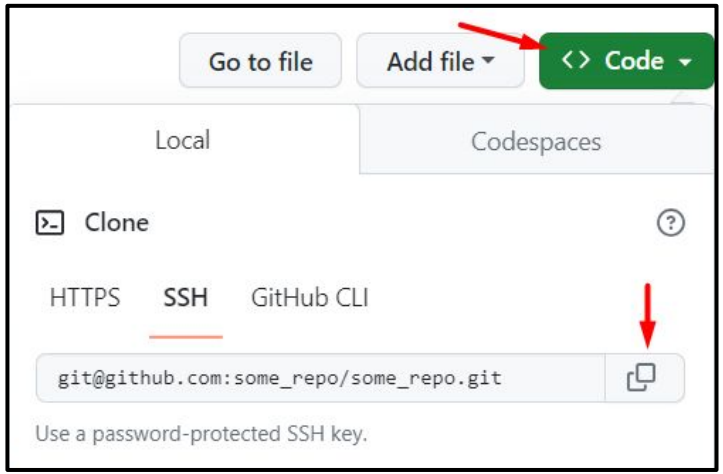
Get remote repo as local

Hands on your keyboard

```
git clone <SSH_URL>
```

Download the repo

```
cd <directory>
```



Get local repo as remote

Hands on your keyboard

```
git remote add origin <SSH_URL>
```

```
git remote -v
```

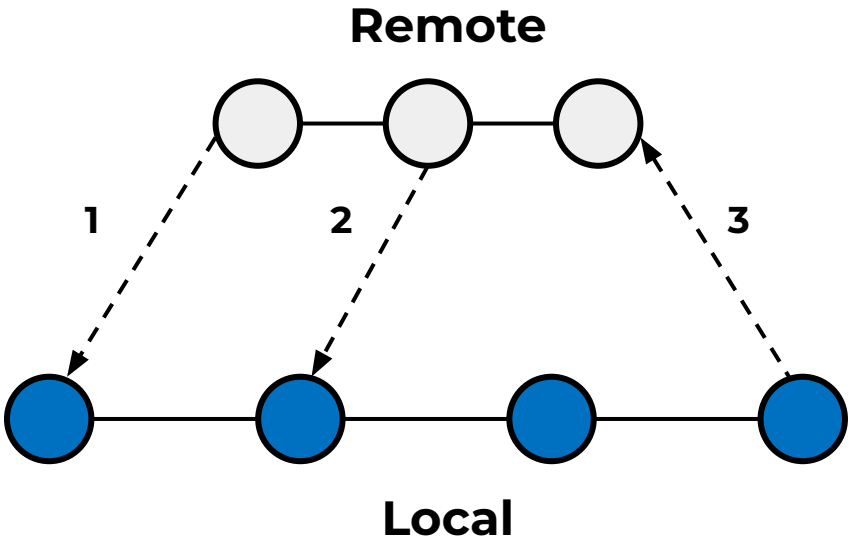
Check URL

Working with remote repo's

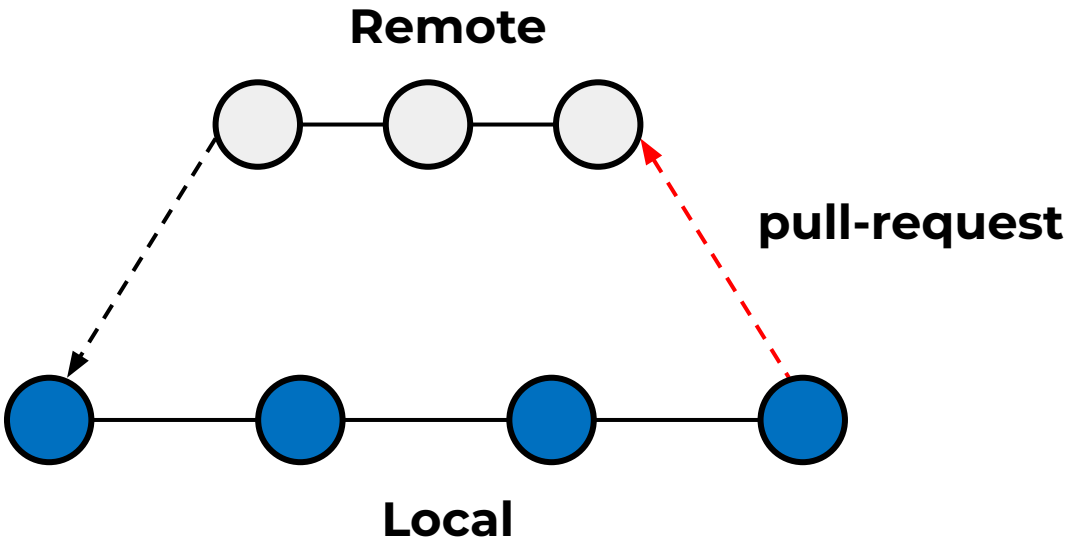
`git clone` **1. Download** the repo

`git pull` **2. Update** the repo

`git push` **3. Upload** the repo



Merging changes to remote



Lecture 1

Troubleshooting

02.09.2023

Git does not commit

Message:

```
# Changes not staged for commit
(some other text here)
```

Solution:

```
Pay attention to solution,
suggested by Git.
Only added files can be
committed.
```

The only correct order:

```
git add
```

```
git commit
```

Over-changing branches

Trouble:

```
Your branch is ahead of
'origin/master' by 1 commit
```

Solution:

Commit it with message “WIP” (work in progress) and do checkout. In this case you will not lose something.

```
git commit -m “WIP”
```

My branch is ahead of origin

Message:

“I have something modified and not ready for commit, but I need to change branch”

Solution:

It’s not an error ;)
Git noticed that you have 1 commit that it does not see on remote repository and suggest you to **push** the changes.

```
git push origin <branch>
```

My HEAD is detached

Message:

```
You are in 'detached HEAD'
state.
```

Solution:

Probably, you face with it while review your **old commit**. You can go back to the old commit (discard changes) or save them as new branch

Stackoverflow

Undo, undo, undo!

2.4 Git Basics - Undoing Things

Trouble:

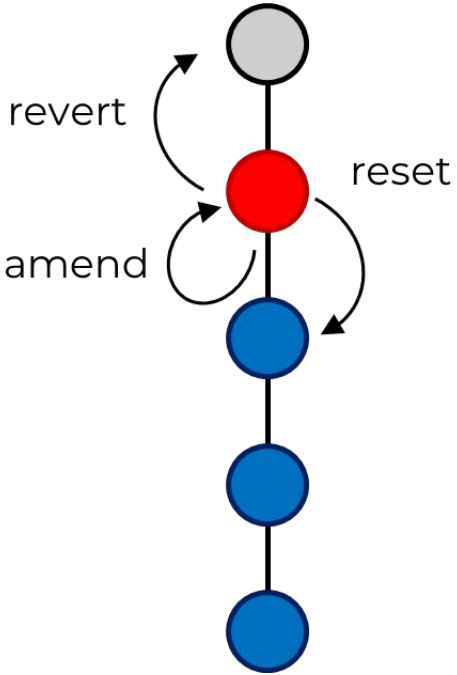
`“Last commit is wrong and I need to remove it”.`

Solutions:

`git commit --amend` Add more files to commit

`git reset HEAD~1` Move HEAD 1 commit back, has **soft** and **hard** modes

`git revert` Create new one, “anti-commit”



Our successes

- We now know what is Git and GitHub
- We can manage Git projects
- We can link local and remote repo’s
- We can себя показать и других посмотреть on GitHub

Any questions?

Course structure?

Git usage?

GitHub?