



# Python programming

## for biologists

07.09.2024



# Our python course team



**Никита Ваулин**

MedUni Wien, BI Bio 22-23

[vaulin@ro.ru](mailto:vaulin@ro.ru)

T: @nvaulin

I: @nvaulin

X: @vaulin2

## Consultants

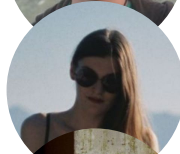
**Антон Сидорин**

SPbSU, BI Bio 20-21



**Надежда Павлова**

MSU, BI Bio 22-23



**Айгуль Нугманова**

WashU St. Louis, BI Algo 22-23



## Assistants



**Артем Ершов**

MGI, BI Algo 20-21



**Игнат Сонец**

IGB/SBM, BI Bio 20-21



**Иван Козлов**

SBM, BI Algo 21-22



**Рина Сытник**

Mislab, BI Bio 18-19



# Course structure

## Meetings

- 13 lectures - **sat 14:00** MSK
- Extra meetings (QA, additional topics) - **wed** evenings

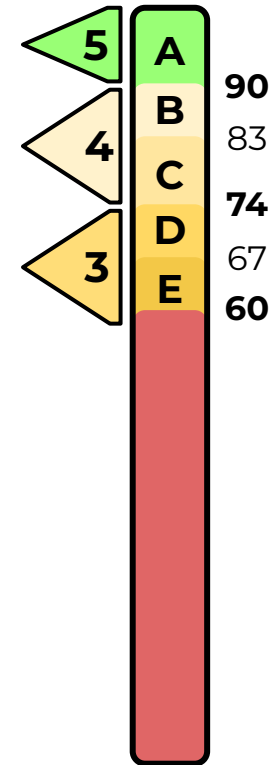
## Quizzes

- 3 quizzes (at the beginning of the lectures)
- max 10 points (0: 0%, 5: <50%, 10: >50%)

## Homeworks



- 11 homeworks
- Each - max 100 points
- **Deadline: next sat 23:59 AOE**
- Submissions after deadline - half of the points
- **Final final deadline: December 7**
- There are assignments for extra points

## Grading





# Course program

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

*next semester...*



## Advanced python

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

## Tools development

- Parallel programming
- Profiling, performance
- SQL



# Sources:

---

## Our

- [GitHub 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?







# Version control systems

## Git

07.09.2024



# What do you know about Version Control Systems?

What are the VCS?

What systems do you know?

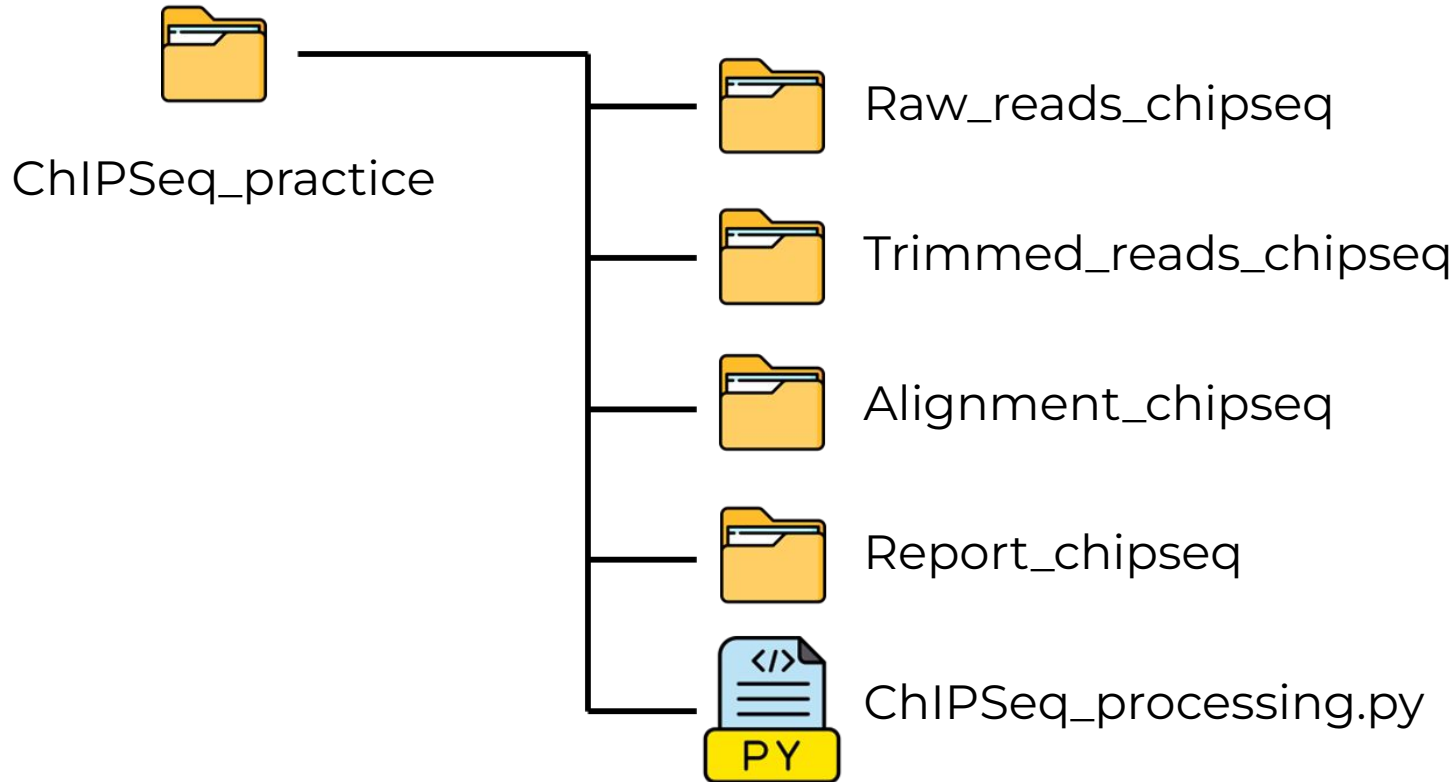
Do you use any of them?





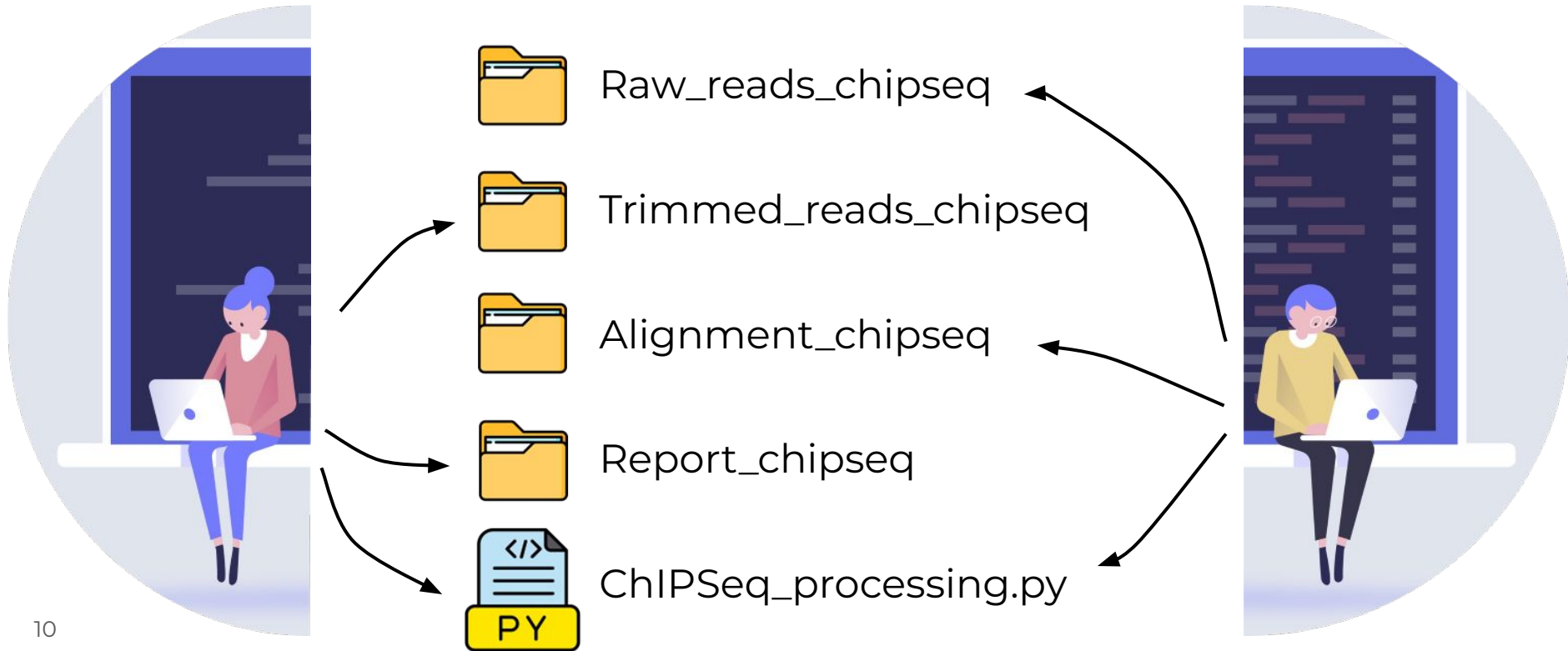


# Some bio project





# Some bio project



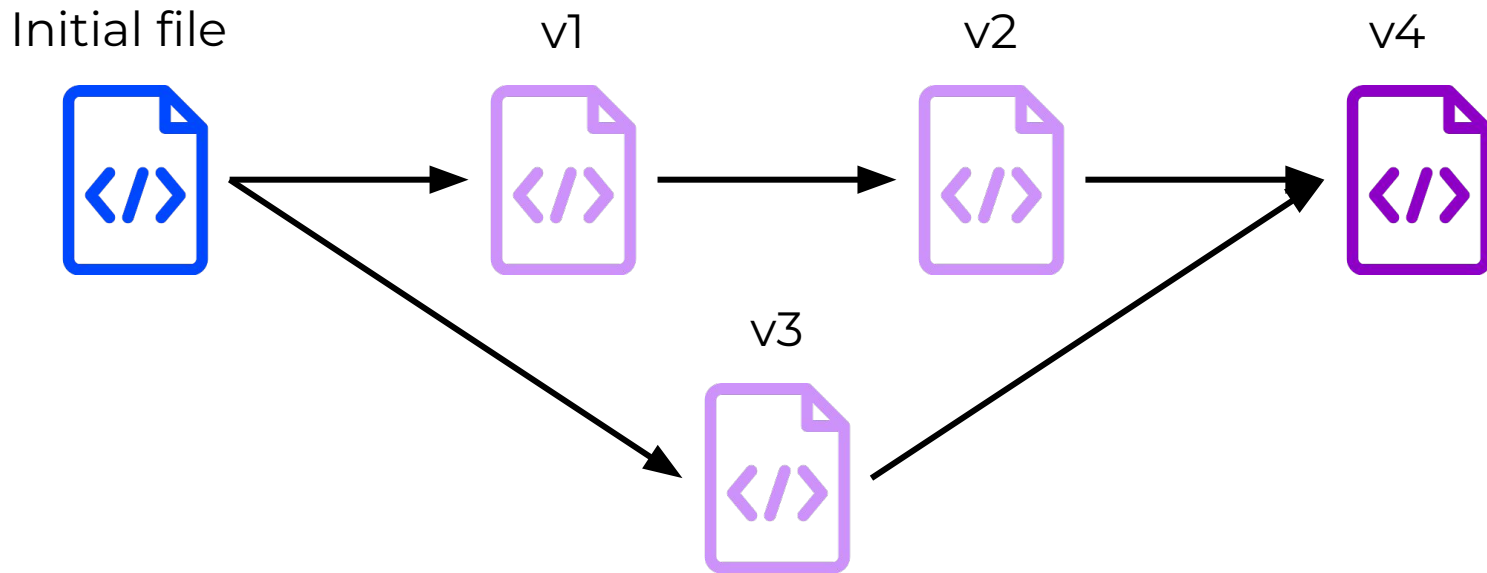


# Two main VCS purposes



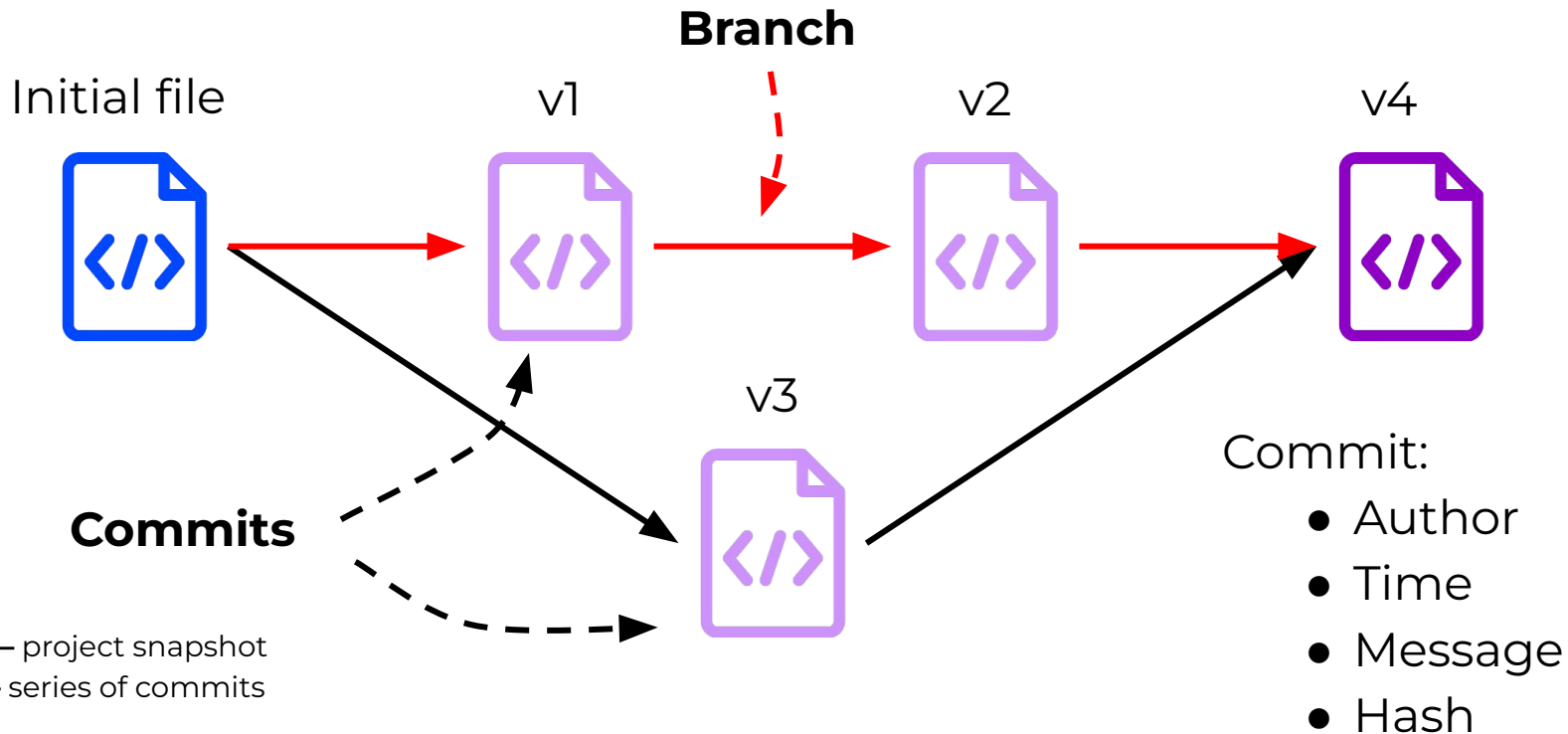


# Version controlling





# Version controlling

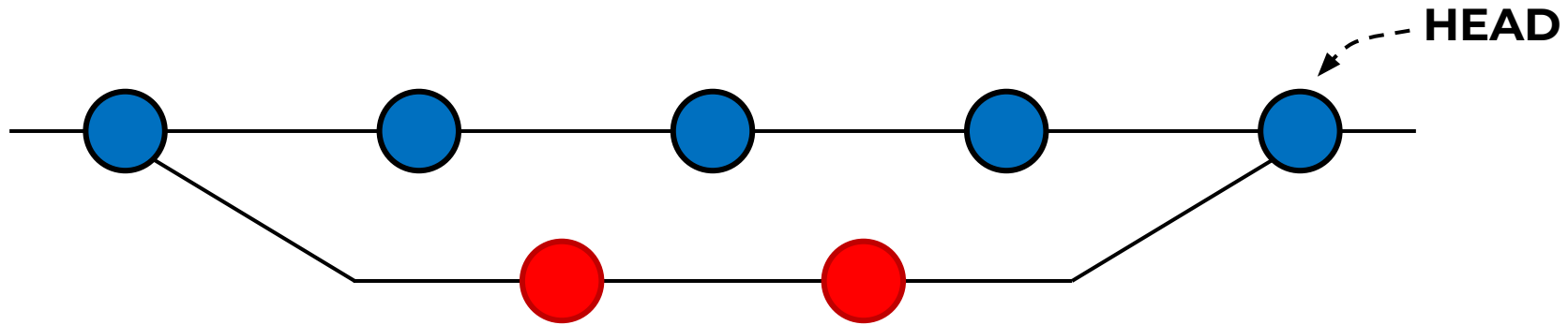


**Commit** – project snapshot

**Branch** – series of commits



# Version controlling



**Commit** – project snapshot

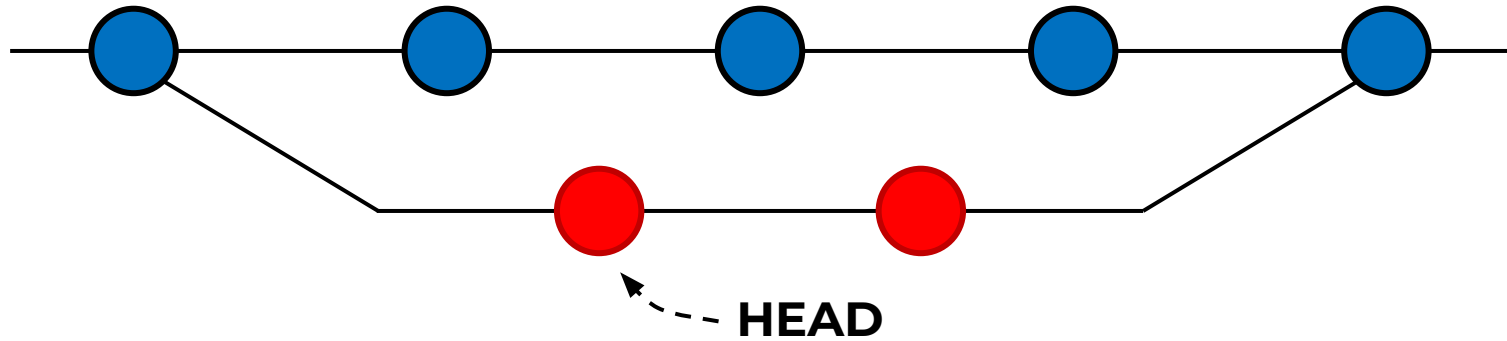
**Branch** – series of commits

**HEAD** – commit you are currently on





# Version controlling



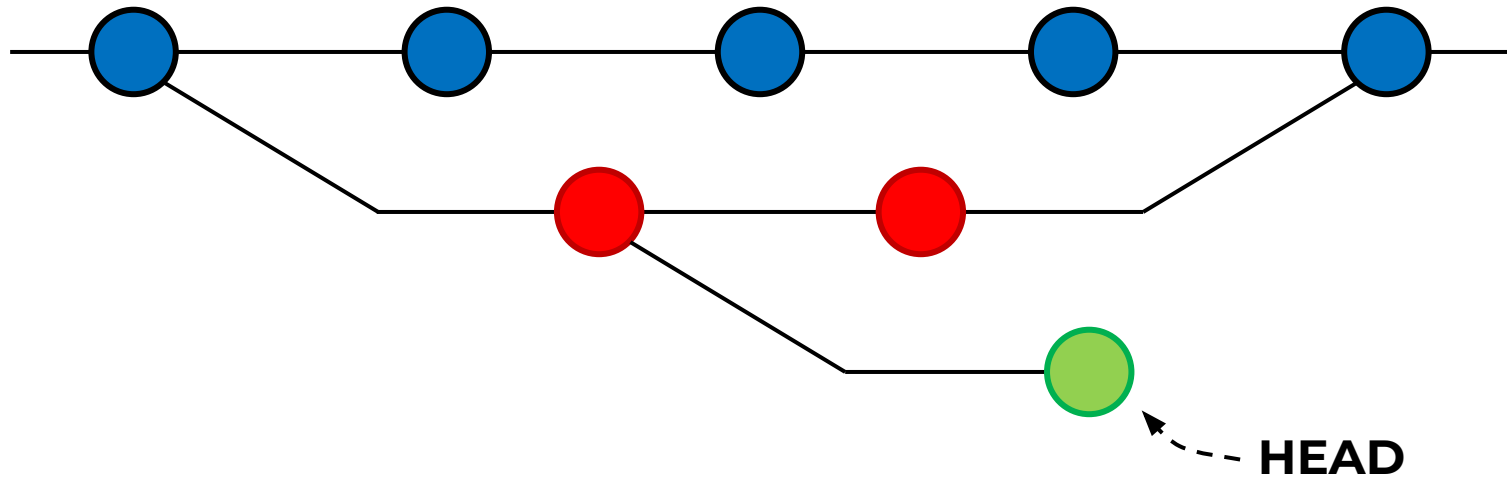
**Commit** – project snapshot

**Branch** – series of commits

**HEAD** – commit you are currently on



# Version controlling



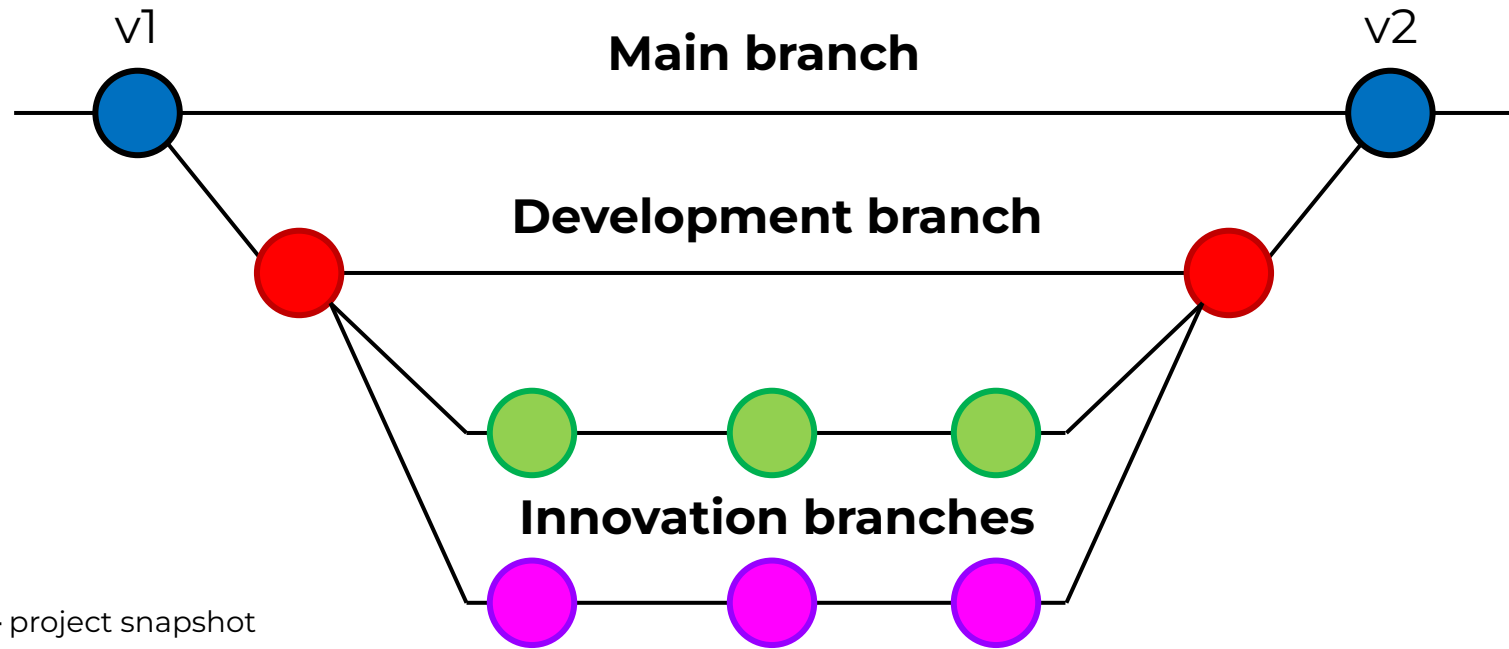
**Commit** – project snapshot

**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
```

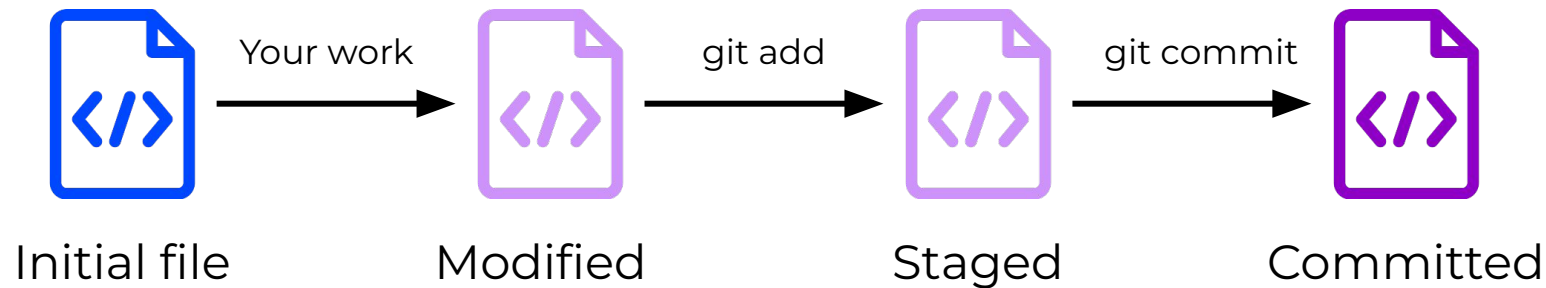
Check once again

What do you see?





# Life of the file





# 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

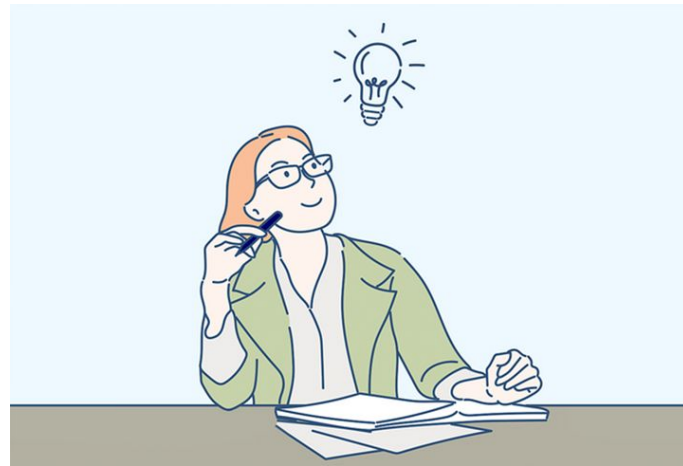




# Commit message rules

- 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

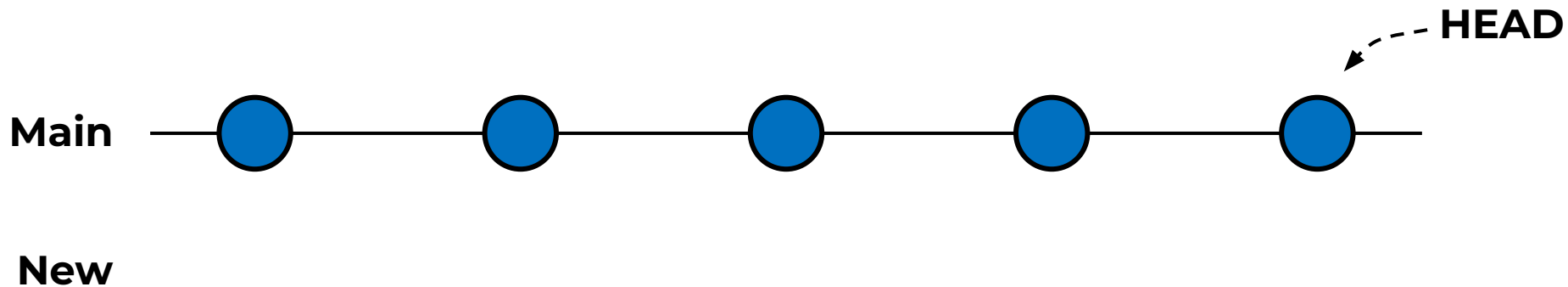
`git blame`

Who edited the file?



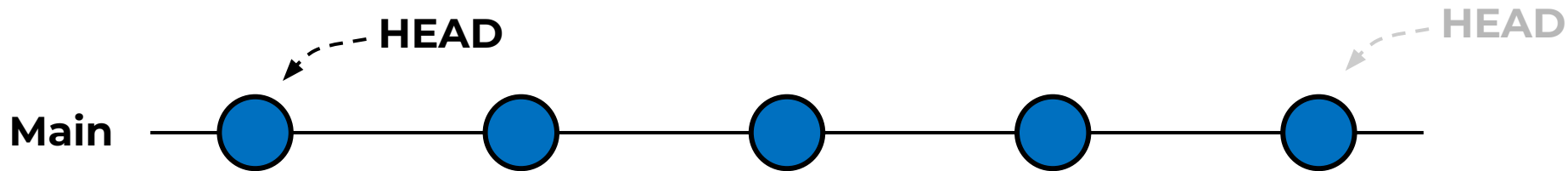


# Branching





# Branching

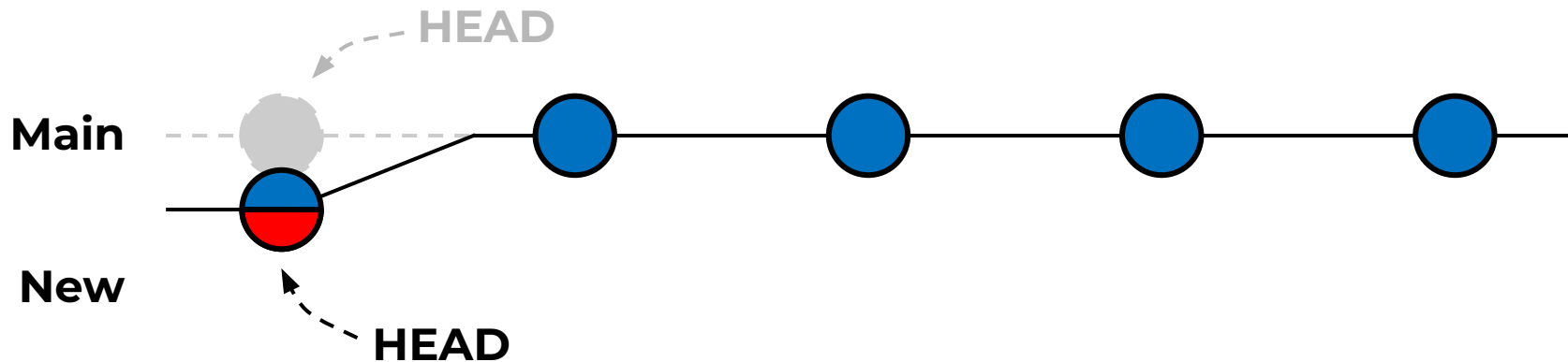


New

```
git checkout <hash>
```



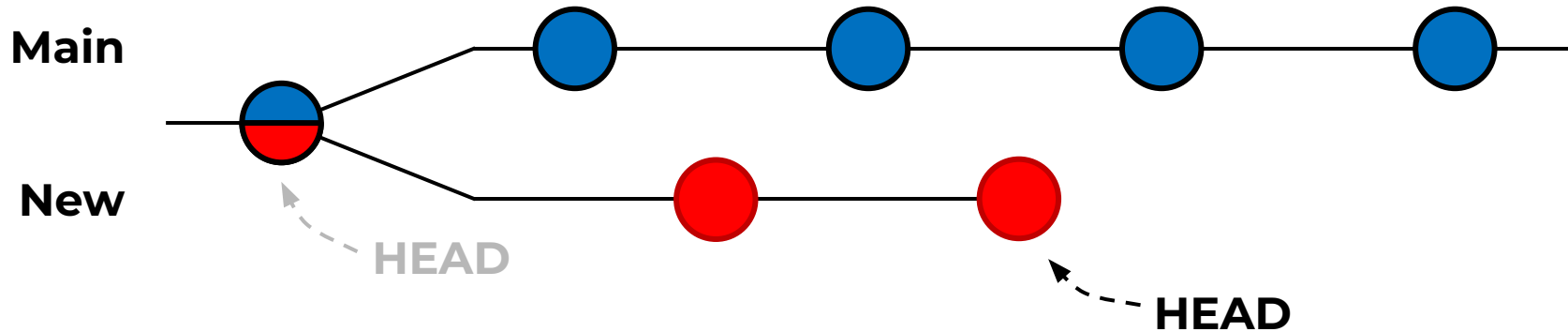
# Branching



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



# Branching

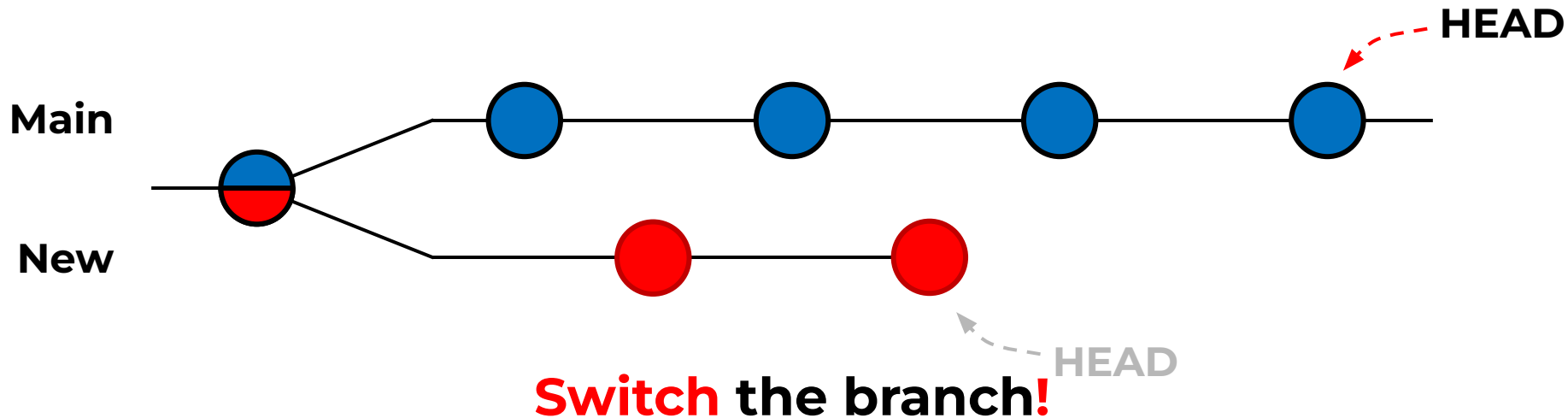


```
git add && git commit
```





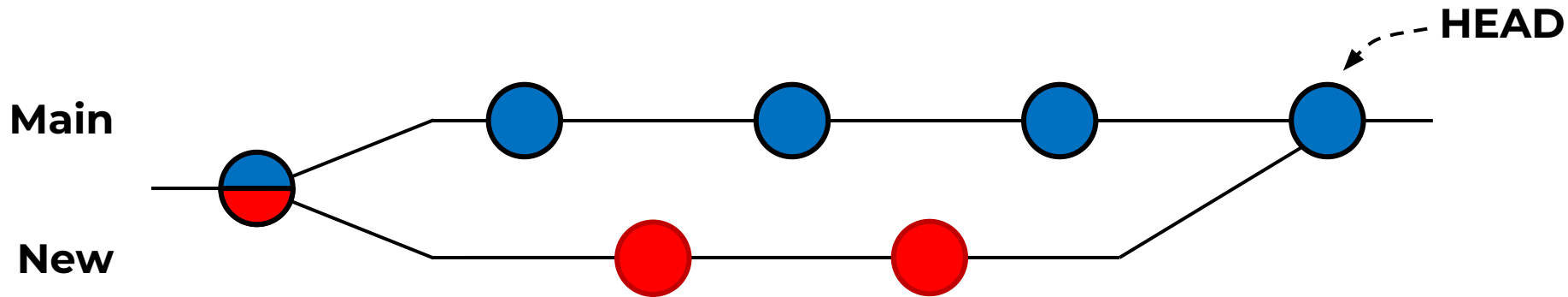
# Branching



```
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





# GitHub

07.09.2024



# Remote repositories



You can use them for:

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



GitLab



GitHub



# 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
```







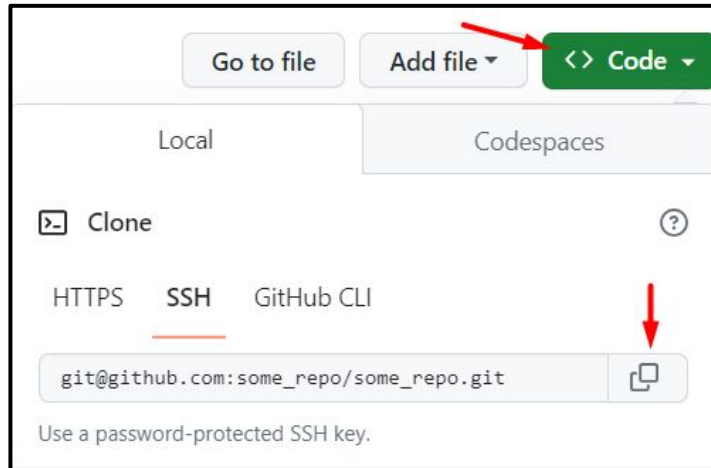
# Get remote repo as local

—— 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`

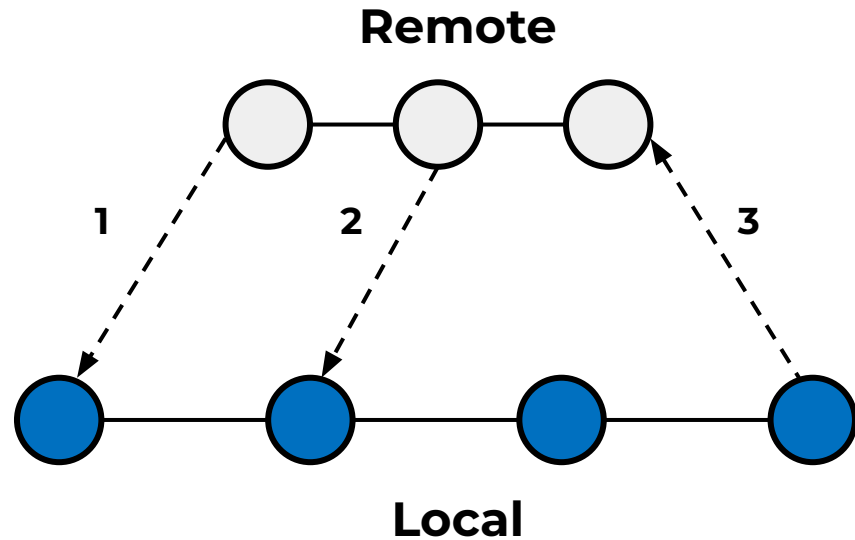
1. **Download** the repo

`git pull`

2. **Update** the repo

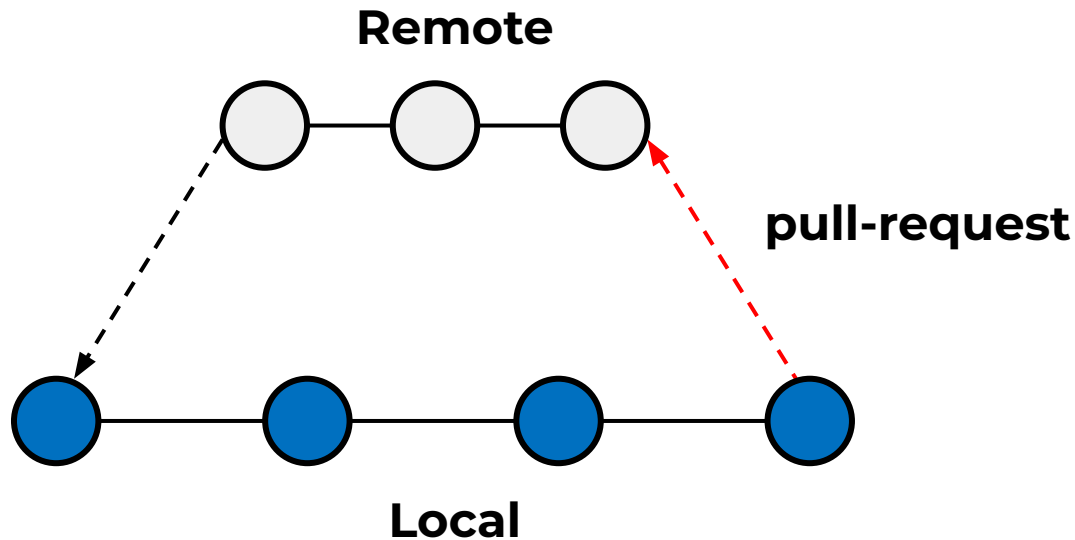
`git push`

3. **Upload** the repo





# Merging changes to remote





# Troubleshooting

07.09.2024



# 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](https://stackoverflow.com)



# Undo, undo, undo!

## 2.4 Git Basics - Undoing Things

### Trouble:

"Last commit is wrong and I need to remove it".

### Solutions:

```
git commit --amend
```

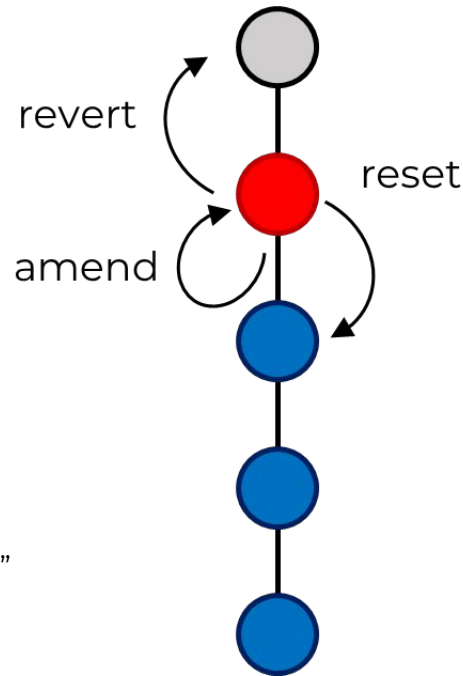
Add more files to commit

```
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?

