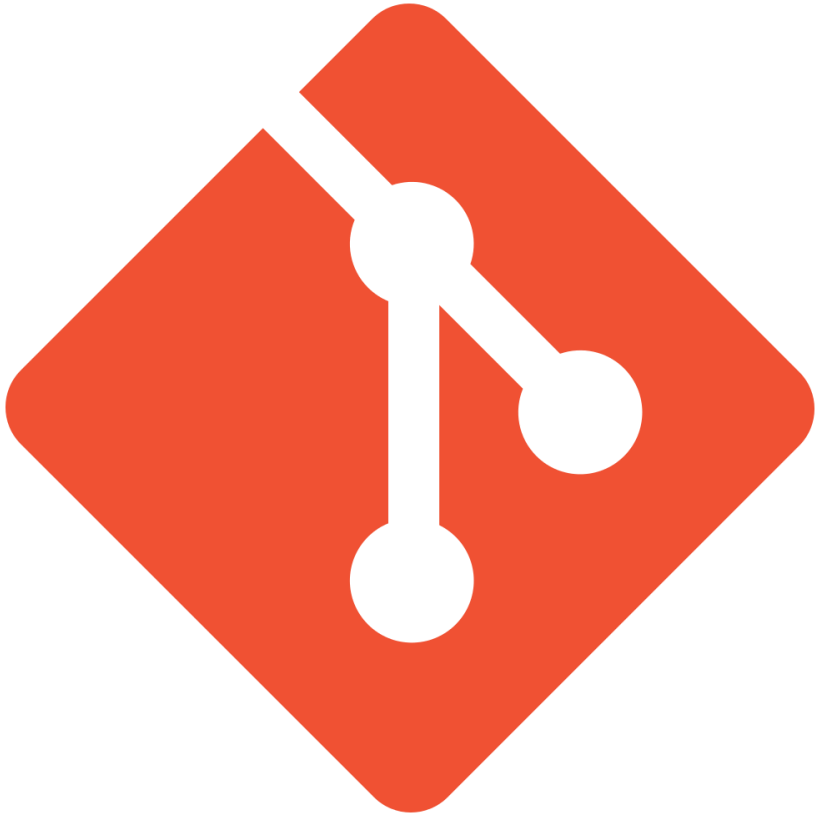
Introduction to git



Léonard & Martin

GNU Generation

Your situation

You want to …

- Work with other people on the same project
  - Share files with others easily
  - Work on the same file as someone else
- Go back in time

Current “solution”

- Share files via email, Dropbox, USB stick, etc.
- Make multiple copies of files/folders
- Keep commented code in source files
- Make frequent backups

Git is what you need!

- Easily share files with your teammates
- Keep history of modifications
  - Can restore older versions of files
  - See who did what, when and on which files
- Automatically merge modifications
- Easily access your code online

Online project hosting

- Offered by third parties
  - github.com
  - bitbucket.org
  - gitlab.gnugen.ch (what we will use today)
- You can also host your own

Create a new project

- Log in using Tequila on gitlab.gnugen.ch
- Create a new project
- Add your teammate(s) to the project

Install git on your machine

- Install git using your package manager
  - On Debian/Ubuntu

    ```
    sudo apt install git
    ```

  - On Fedora

    ```
    sudo dnf install git
    ```

Configure git

- Minimal configuration

```
git config --global user.name 'John Doe'
git config --global user.email 'jd@epfl.ch'
```

Get started with your project

- Clone your repository

```
git clone "https://<you>@gitlab.gnugen.ch/<your project>.git"
```

- Create a new text file and “add” it

```
git add <your_file>
```

- Commit your modifications

```
git commit -m "some message"
```

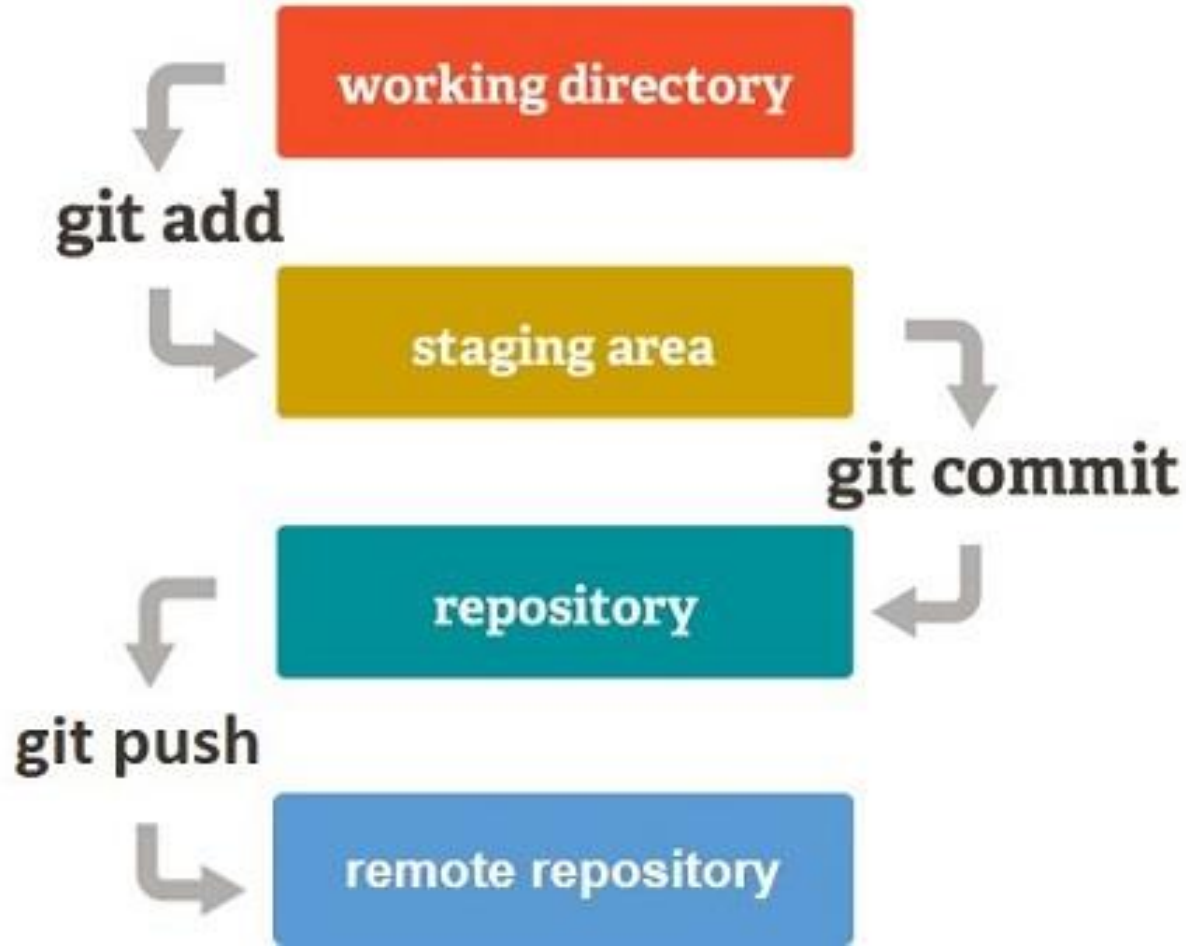
- Push your commit

```
git push
```

What did we just do?

- Downloaded the repository from Gitlab
- Told git to track the new text file
- Created a “snapshot” containing the modifications
- Uploaded our modifications to Gitlab

What actually happened



Useful commands

- See the state of your local repository

```
git status
```

- See the logs of the last commits

```
git log
```

- See the unstaged modifications

```
git diff
```

Get up to date

- Update your local version

```
git pull
```

Ignoring files

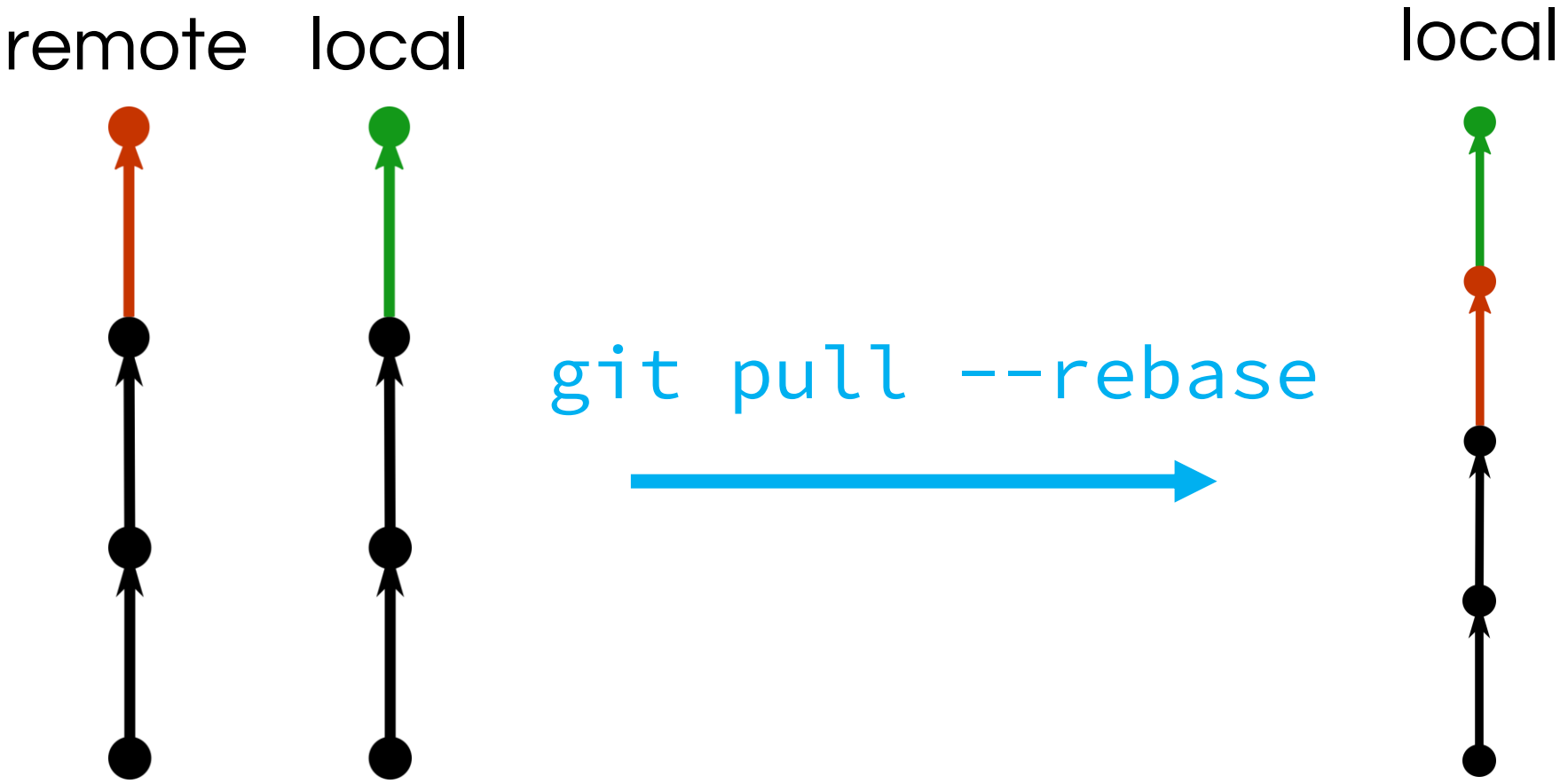
- Create a file named `.gitignore` in your repository
- Add to it the names of the files and folders to ignore
- You can uses patterns in the file names
  - `*.out` will ignore all files with extension `.out`
  - `<dir_name>/**` matches all files inside the directory

Working with others

Possible issues:

- Someone did a push before you
- Pull with uncommitted modifications

Someone pushed before you



Pull with local modifications

- Throw away all your local modifications

`git reset --hard` (caution!)

- Keep your local modifications

`git stash save` (put your modifications aside)

`git stash pop` (reapply your modifications)

- Commit your modifications, then rebase

Boom!

- Attempting to merge modifications of the same line(s) of a file will cause a conflict
- Problematic commands

```
git pull
git stash pop
```

Resolving conflicts

- During a stash pop:
  - Resolve conflict
  - `git stash drop`
- During a pull rebase:
  - Resolve conflict
  - `git rebase --continue`

common →
remote →
local →

```
Hello World!
<<<<<<< Updated upstream
Lorem Ipsum
=======
Dolor Sit Amet
 >>>>>>> Stashed changes
```

Undo stuff

- Undo changes to a file

```
git checkout <file>
```

- Remove file from staging

```
git reset <file>
```

- Undo a commit

```
git reset <commit>
```

- Undo a commit if already pushed

```
git revert <commit>
```

Branches

- Create a new branch

```
git branch <branch_name>
```

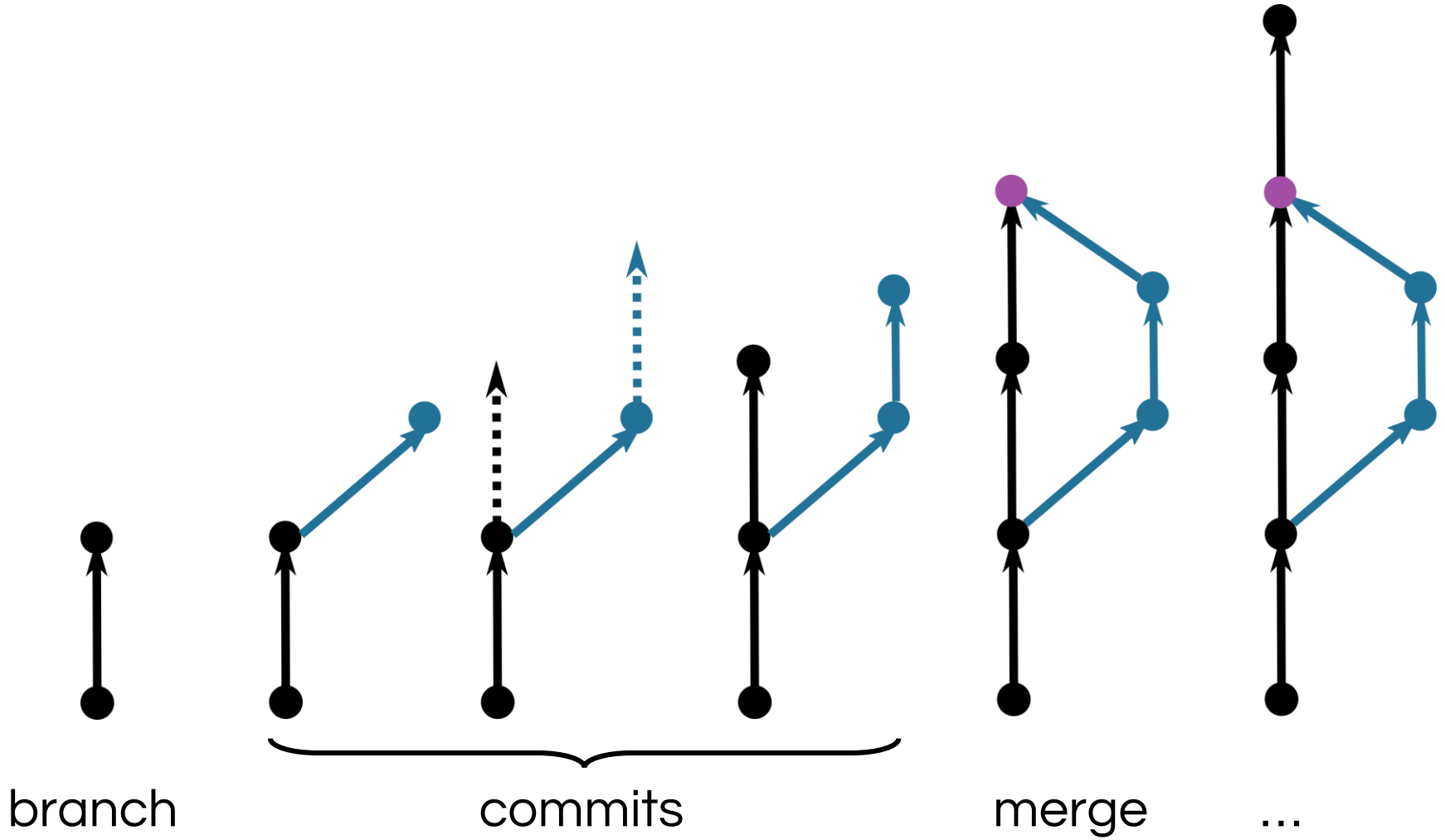
- Switch to a branch

```
git checkout <branch_name>
```

- Merge a branch into the current branch

```
git merge <branch_name>
```

Branch lifecycle



Other useful commands

- Create a repository in a local directory

```
git init
```

```
git remote add origin <url>
```

Do some commits

```
git push -u origin master
```
(-u only the first time)

Other useful commands (2)

- Interactive staging

```
git add -i
```

- Interactive history rewriting

```
git rebase -i <commit>
```

- History of checkouts

```
git reflog
```

Other useful commands (3)

- Selectively merge commits from another branch

```
git cherry-pick <commit>
```

- Show authors of each line of a file

```
git blame <file>
```

- Display diff of a commit

```
git show <commit>
```

Thanks for participating!