Git

Goals

- Create a history of changes in your project
- Publish your projects
- Collaborate on projects

Before we start...

- Go to **https://git-scm.com/downloads**
- **Download** the installer
- Don't open it yet!

What is versioning control?

- Way to keep track of changes to files
- Between multiple authors
- History of changes through time
- Helps you:
  - Preventing mistakes
  - Remembering changes
  - Collaborating

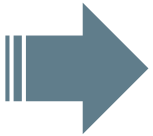
Git

- Created during development of Linux
- Open source

Changes

index.html

```
<html>
<head>
</head>
<body>
 Hallo Welt
</body>
</html>
```



index.html

```
<html>
<head>
</head>
<body>
 Hello world
</body>
</html>
```

What	Hallo Welt → Hello world
Where	index.html, line 5
When	26/04/2017, 1pm CET
Why	Change text language
Who	Leonardo

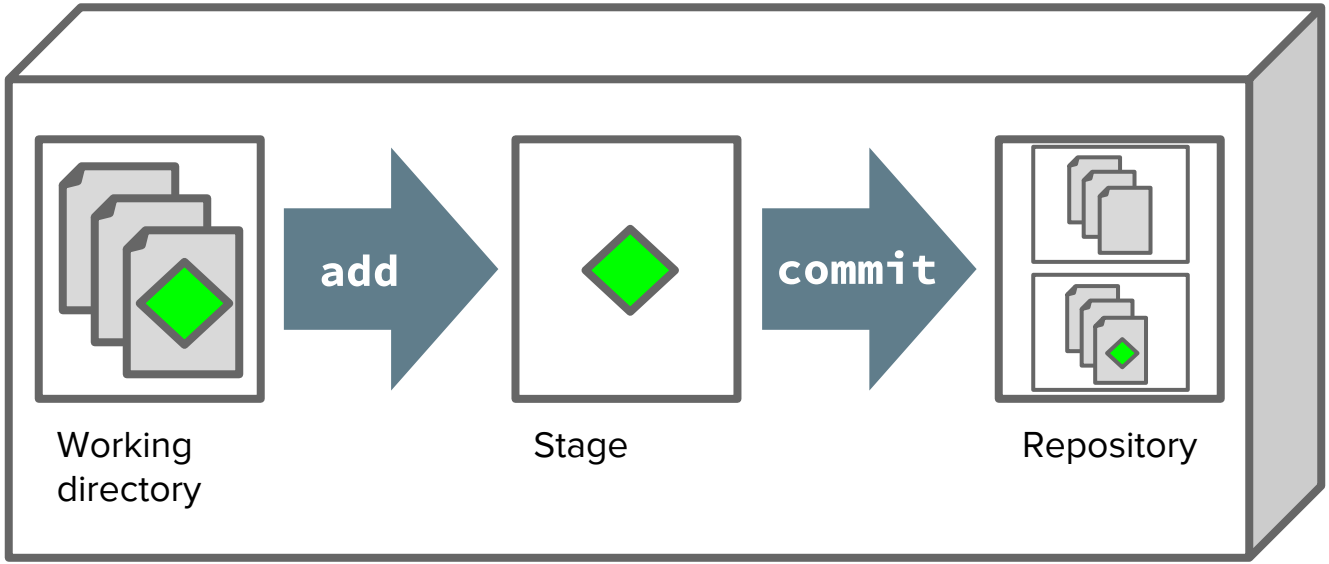
id

The action of "taking a snapshot" is called commit

Repository (repo)

is a collection of snapshots through time

Repository (repo)



Install git

- Select option “Use Git from Git Bash only”
- Select option “Use the OpenSSL library”
- Select option “Checkout Windows style”
- Select option “Use MinTTY”

- When the installation is finished, you can open the GitBash software and try to ask for the version installed:
  - `git version`

First steps

- **GitBash** works as a terminal console
- You'll be giving the commands inline
- Git commands will start with `git`

- Create a new directory
  - `mkdir NAME_OF_YOUR_PROJECT`
- Enter the directory
  - `cd NAME_OF_YOUR_PROJECT`

Use whatever name you prefer

- Your name
- Your favorite fruit
- Your favorite Disney film
- Your favorite pizza
- ...

First steps

- Configure git:
  - `git config --global user.name "YOUR NAME"`
  - `git config --global user.email YOUR EMAIL`
- Check configuration:
  - `git config user.name`
  - `git config user.email`
- Initialize repository: **(in your working directory!)**
  - `git init`
- Check the status
  - `git status`

Your first commit

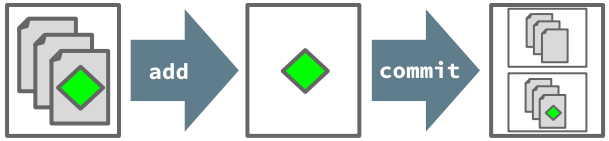
- Open the File Explorer on your computer
- Open your working directory (the one you created)
- In your working directory, **create a file**

index.html

```
<html>
<head>
</head>
<body>
 Hallo Welt!
</body>
</html>
```

Your first commit

- **Check the status** of your working directory
  - `git status`
- **Add** the file to stage
  - `git add FILENAME`
- **Check the status** again: what changed?
- **Commit** what's on the stage
  - `git commit -m "COMMIT MESSAGE"`
- **Check the status** again



Your second commit

- Create a new file (**`newfile.html`**)
- Add the file to stage
- Make some changes to the previous file (**`index.html`**)
- Commit what's on the stage

What happens to **`index.html`**?

Your third commit

- Add and commit the changes to `index.html`

Show the history

- Show list of past commits
  - `git log`
- Show only last 2 commits
  - `git log -n2`

USE THE NUMBER YOU WANT!

- Show changes too
  - `git log -p`

USE SPACEBAR TO SCROLL

Commit

Who

When

Where

What

ID

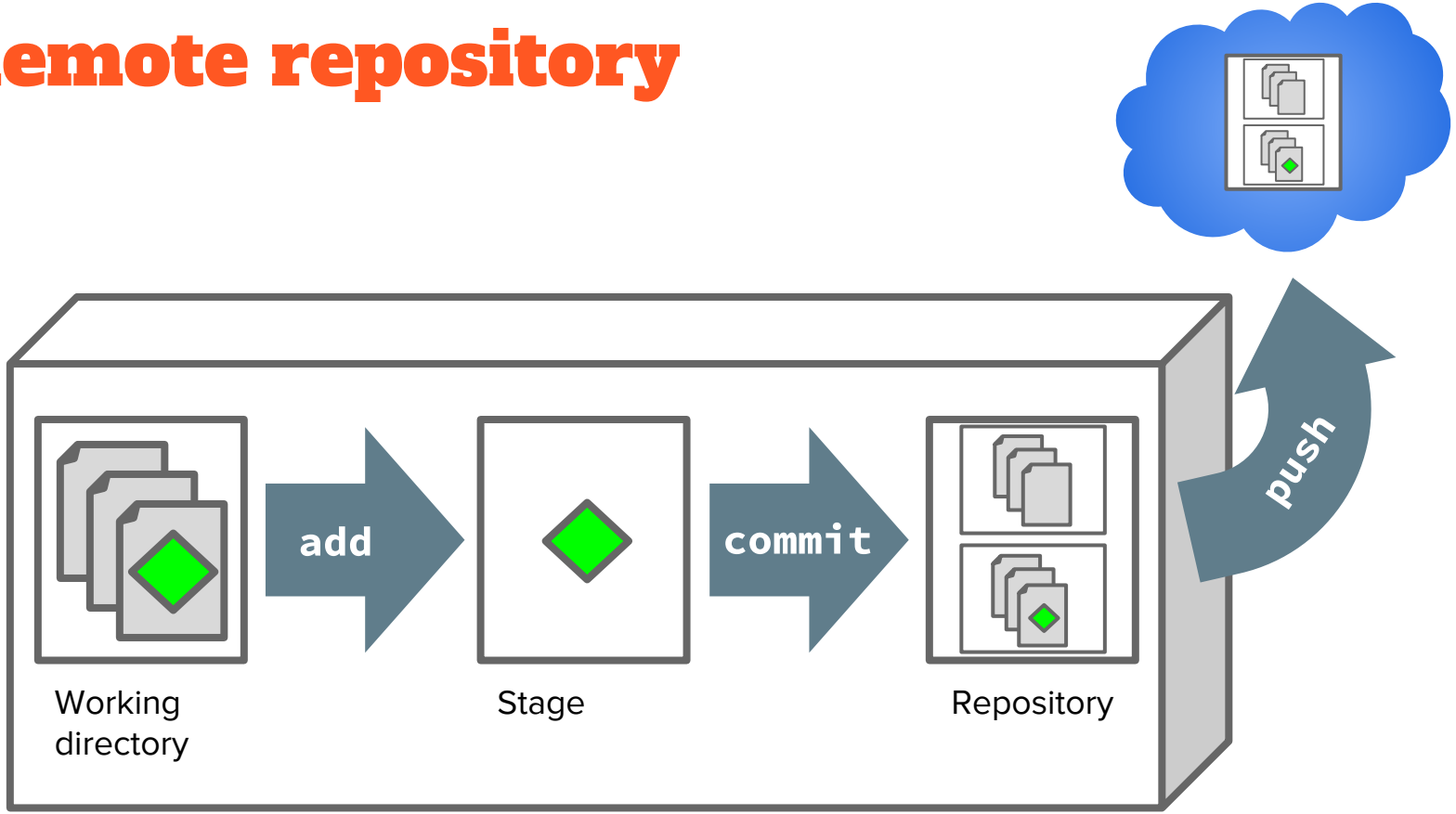
Why

```
commit 777e5923458503ffa8cb81d2de3107b2df4afc36
Author: Leonardo
Date:   Fri Mar 24 12:18:04 2017 +0100

    changed text from German to English

diff --git a/index.html b/index.html
index ff4202b..43f7e90 100644
--- a/index.html
+++ b/index.html
@@ -2,6 +2,6 @@
 <head>
 </head>
 <body>
- Hallo Welt!
+ Hello World!
 </body>
 </html>
```

Time-traveling

- You can go back to a previous state!
  - `git checkout ID`
- To return to last state:
  - `git checkout master`

Remote repository



GitHub

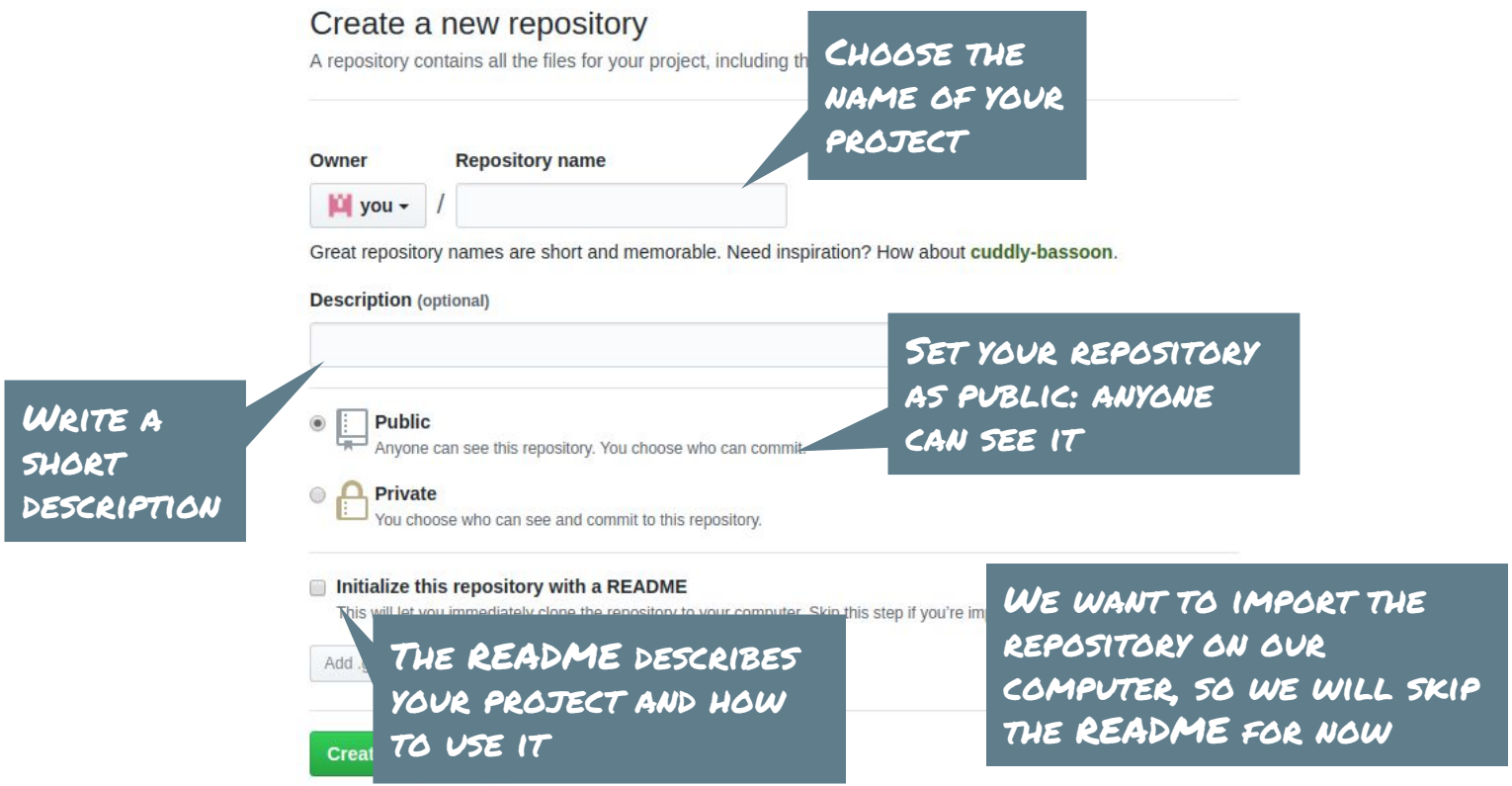
- Web-based Git repository
- 14 million users
- 85 million repositories
- Social network to share your work
- It has a free plan

Create a GitHub account

- Go to **github.com/join**
- Choose your **username**
- Insert a **password** and a valid **email address**
- Choose the **free plan**

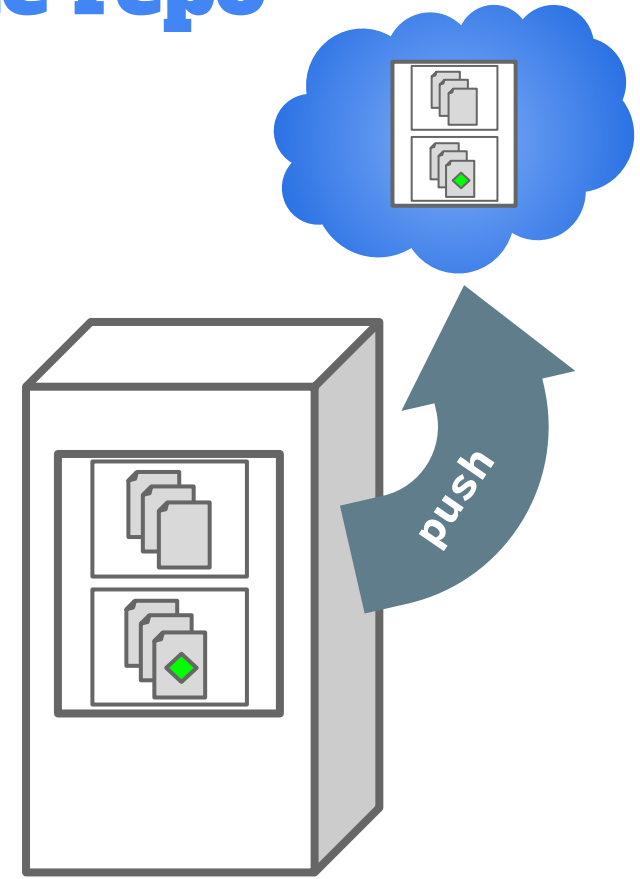
Be creative!
You can have a look at
github.com/trending/developers
to see some "famous" developers
and their usernames

Create a new repository



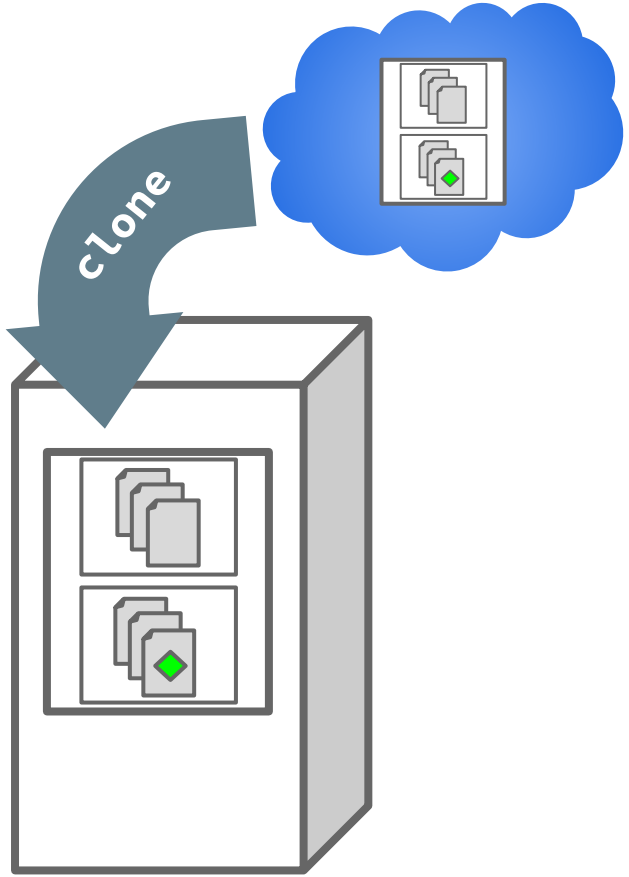
Link local repo with remote repo

- Set the remote repository
  - `git remote add origin https://github.com/USER/REPO.git`
- Verify
  - `git remote -v`
- Push the master (the latest status)
  - `git push -u origin master`
- The next time, you can use just
  - `git push`
- Now go to github.com and look at your profile!



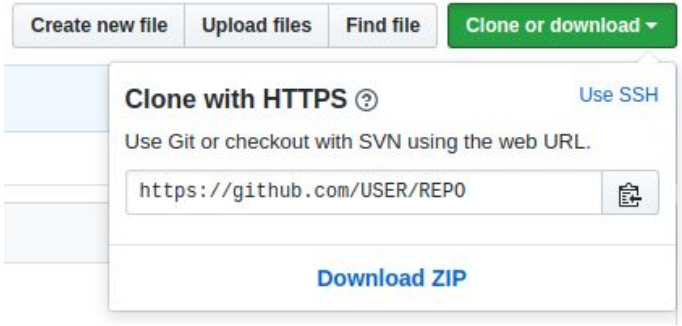
Cloning

- Every (public) repository on GitHub can be downloaded
- The action of download a remote repository to your local machine is called **cloning**
  - `git clone https://github.com/USER/REPO`



Cloning

- Form pairs
- Go to your partner's GitHub profile
- Open the repository page
- Click on **Clone or download**
- **Copy** the HTTPS url for the repo
- **Switch folder** in the terminal!!
  - `cd ..`
- **Clone** the repo with git
  - `git clone URL`



Cloning

- In the cloned repo, add or **modify** a file
- **Commit** the changes
- Now try to **push** your commit to the remote. What happens?

Collaborating

- **Every user** can download your repository
- They won't be able to push commits
- You can invite other users to also **contribute** to your repository

Collaborating

In your pair, choose who will be the "inviter" and who will be the "collaborator"

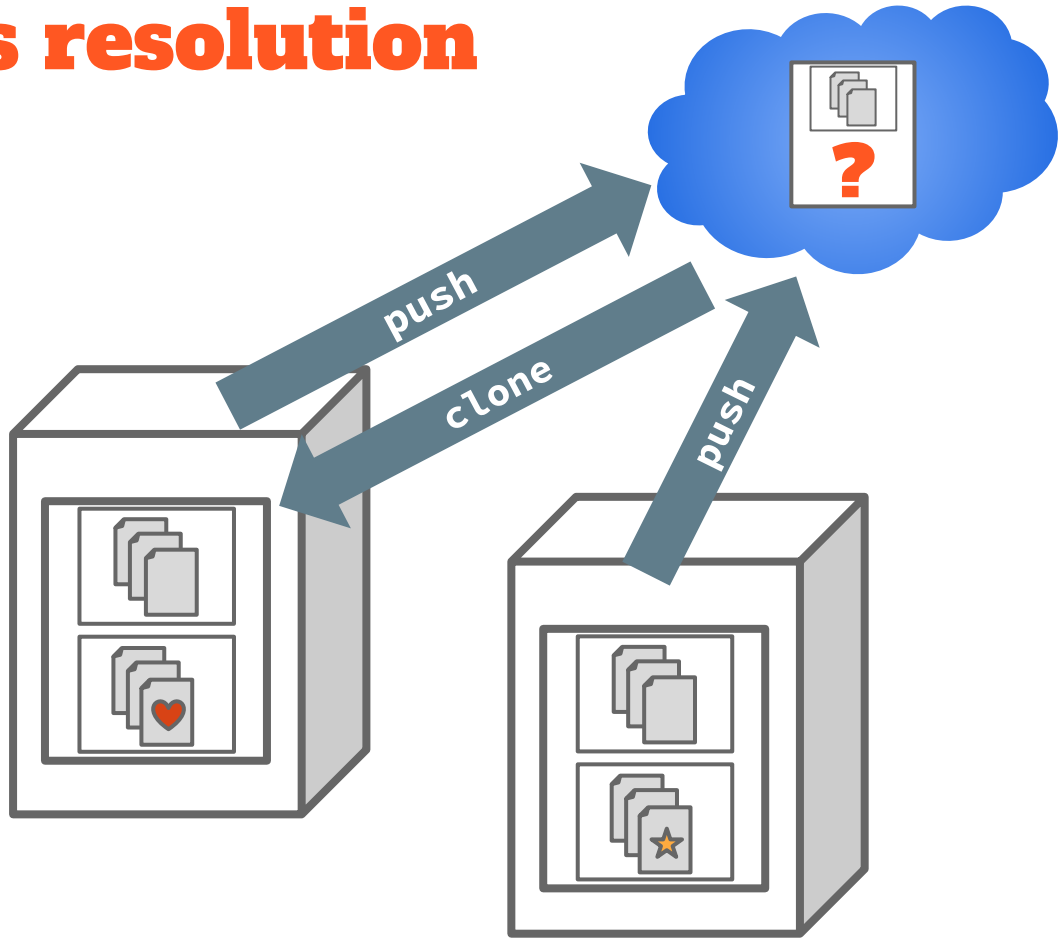
Inviter

- In your repository page:
- Go to **Settings**
- Select **Collaborators**
- **Search** for your invitee's username
- **Add** them as collaborator

Collaborator

- Go to: github.com/INVITER/REPO/invitations (you'll also receive an email with this link)
- **Accept** the invitation
- **Push** the commit you created before

Conflicts resolution



Conflicts resolution

- In the repo of the inviter, both change the same part of the **same file**
  - The inviter will need to switch folder:
    `cd ../FOLDER NAME`
- Both **commit**
- Both **push** the changes

What happens?

index.html

```
<html>
<head>
</head>
<body>
 Hola mundo
</body>
</html>
```

index.html

```
<html>
<head>
</head>
<body>
 Ciao mondo
</body>
</html>
```

Pull

- The second commit is rejected!
- Before pushing, the second person must pull
  - `git pull`
- Pulling: getting the changes from remote
- Pulling changes to files that were also changed in the working directory creates **conflicts**

Conflicts resolution

- **Pull** for changes
- You'll be warned about a **conflict** in the file you both have changed
- Open the file and **solve** the conflict

Try to include both changes

index.html

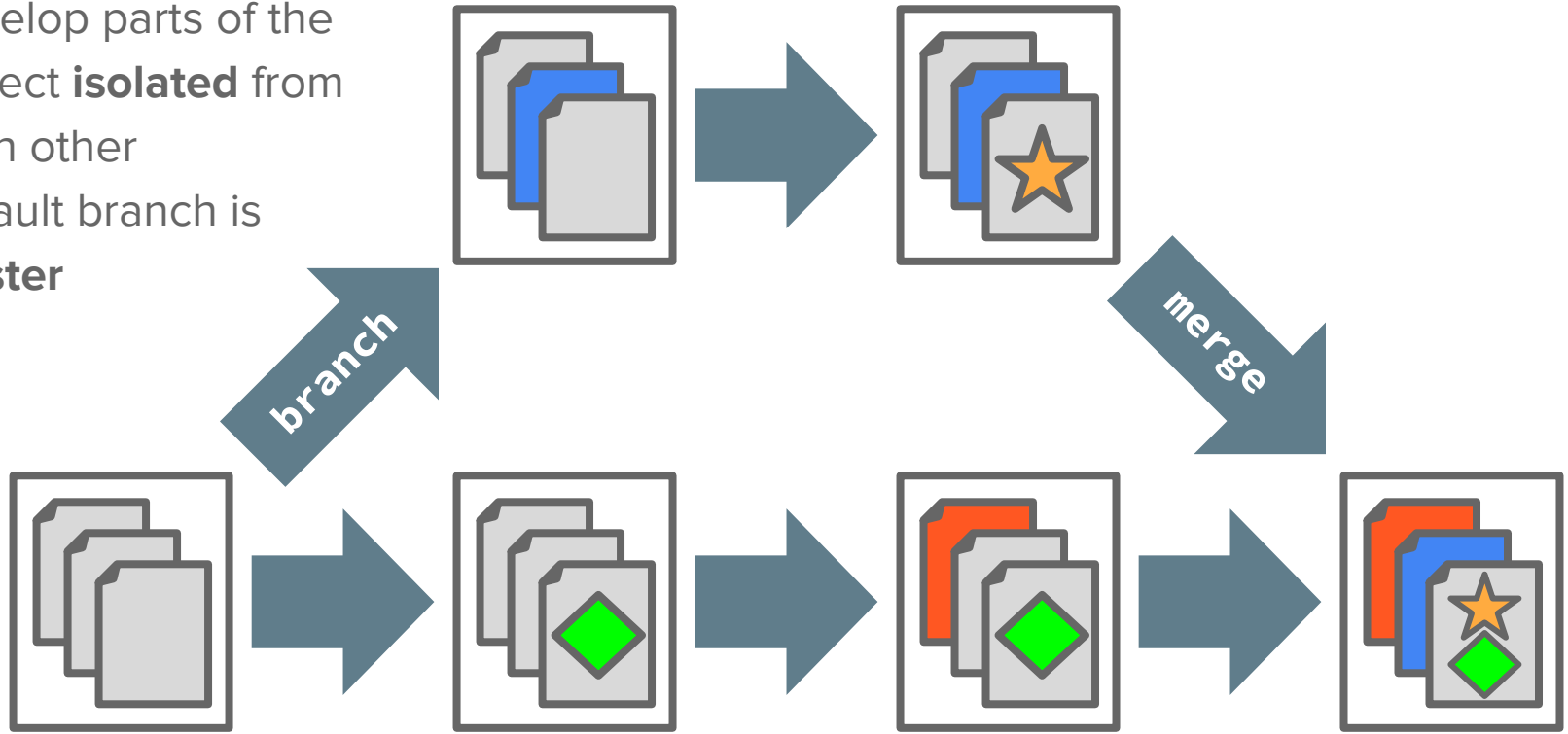
```
<html>
<head></head>
<body>
<<<<<<< HEAD
Hola mundo
=======
Ciao mondo
>>>>>>> 13a24f22f8ac7fc
</body>
</html>
```

index.html

```
<html>
<head></head>
<body>
  Hola mundo & ciao mondo!
</body>
</html>
```

Branches

- Develop parts of the project **isolated** from each other
- Default branch is **master**



Branches

Inviter

- Create a new branch
  - `git branch NAME`
- Push the new branch
  - `git push origin NAME`

When you push a new branch, you need these two extra parameters

Collaborator

- Pull the changes
- Switch to the new branch
  - `git checkout NAME`

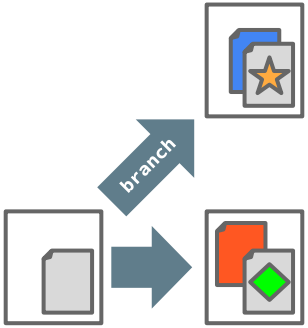
Branches

Inviter

- Checkout master
- Create a new file page2.html
- Create a link to it in the index.html
  - `<a href="page2.html">Link</a>`
- Commit this change

Collaborator

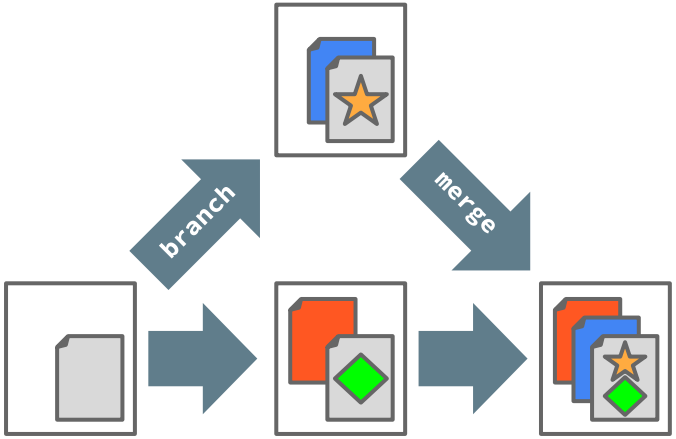
- In the new branch, create a new file page1.html
- Create a link to it in the index.html
  - `<a href="page1.html">Link</a>`
- Commit this change



Branches

Invitee

- **Merge** the new branch into master
  - `git merge NAME master`
- Solve the conflicts!
- Push



Slides

- Switch folder
  - `cd ..`
- **Clone** this repository:
  - `git clone https://github.com/leovugee/git-intro-devugees`

Thanks!