# Using GitHub

Daniel Mateos, Antonio Almagro, Igor Arambasic

# Git

Do we all have a Github account?

Creating a version control repository/directory:

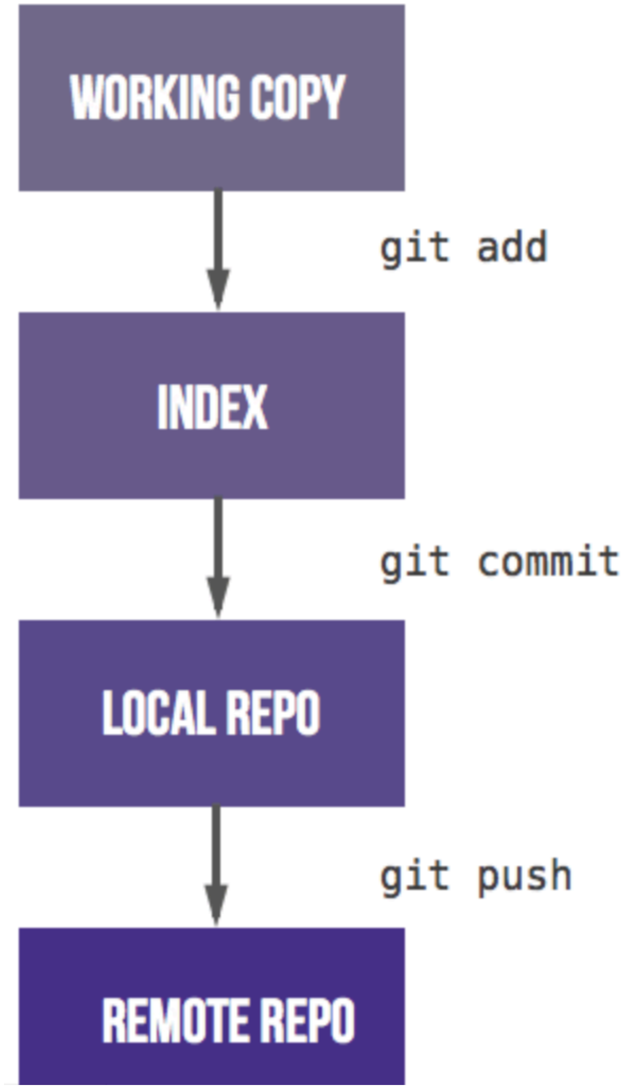
- @remote directory (github)
- @local directory

Git basic commands:

- clone - Clone a repository into a new directory
- init - Create an empty Git repository or reinitialize an existing one
- status - Show the working tree status
- diff - Show changes between commits
- add - Add file contents to the working tree and the index
- rm - Remove files from the working tree and from the index
- commit - Record changes to the repository
- fetch – retrieves objects and their metadata from the remote repo
- push - Update remote refs along with associated objects
- pull – fetch + merge  changes

# What is git and how to use it???

- Version control
- Only text files
  - max file size 100Mb
  - No binary!!!!!
    - .doc
    - .docx
    - .xlsx
    - .zip
    - .exe
    - …
    - …



# Git configuration

Change the content of

- ~/.gitconfig
- git config --global user.name "Igor A"
- git config --global user.email “ia@some.com"

Without global flag it would set repository specific information

- git config --unset user.name
- cat .git/config

Look inside the

- ~/.git-credentials -> I will NOT show this file on the screen!!!

## Creating  local repository by cloning (an empty) from remote repo

## Creating  local repository by cloning (an empty) from remote repo

Lets try it (go to ~Repositories):

- Prepare the repository at github webpage (with file inside)
- **git clone** _______*LINK_TO_REPO* __________________ (link C/P from webpage)
  cd *REPO_NAME*

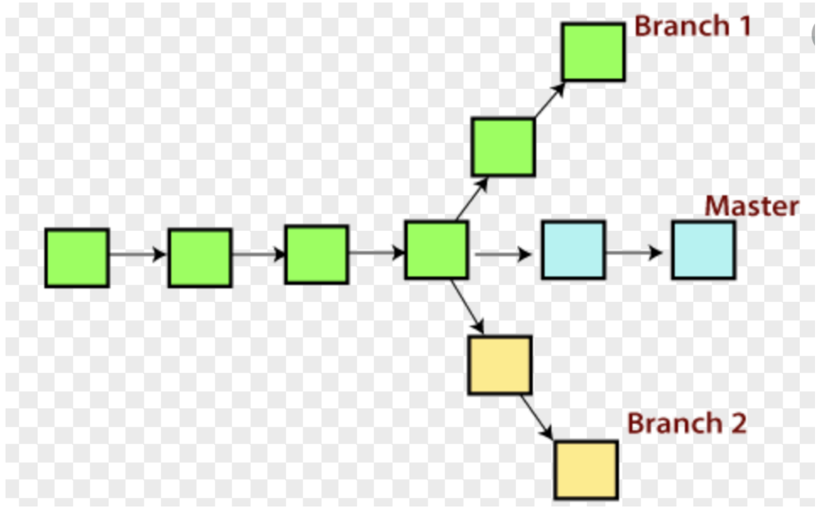
We see master in green. This means we are on master branch which is default branch name.

## Creating  local repository by cloning (an empty) from remote repo

What is a branch? Branches are like "Save as..." on a directory witht the difference that common files are only stored once -> no wasted space

You can say they are as "virtual directories" in the .git folder.

While inside a physical directory, you move through  virtual directories with a **git checkout**.



What is a remote? A remote (branch) is a branch on a remote location (in most cases origin).

By default each new clone maintains a link back to its parent repository via a remote called origin!

## Creating  local repository by cloning (an empty) from remote repo

Lets edit a file inside the local repository…

```
kwrite readme.txt
git add readme.txt
```

we see 2 numbers and the master is not green!

Why? What’s happen?

We have un-staged changes…

first number corresponds to the number of added lines wrt last committed version

second  number corresponds to the number of delete lines wrt last committed version

```
git diff
git status
```

## Creating  local repository by cloning an empty from remote repo

Moving on….

**git commit**

Commit is used to record changes to repository.
We always have to describe what has been changed so a log file opens automatically…

master has changed its color in prompt again !
Why? What's happen?
We have committed changes ready to be pushed to repository
We see only one number and it corresponds to the number of committed changes ready to be pushed

**git status**

## Creating  local repository by cloning an empty from remote repo

Lets contine working on our file…

```
kwrite readme.txt
git add readme.txt
```

we see 3 numbers now and the master is not green again!

## Creating  local repository by cloning an empty from remote repo

Moving on….

    git push -u origin master (push to which remote from which local branch)

-u, --set-upstream
For every branch that is up to date or successfully pushed, add upstream (tracking) reference,

master in prompt in green again!

What is origin/master? Where did we push?

```
git remote –v
git remote show origin
```

- See the repository at github webpage

## Creating  local repository by cloning an empty from remote repo

## Initializing local repository and pushing it to remote repo

Lets try this(go to ~Repositories):

```
mkdir master3_second; cd master3_second
echo “my name is not important" >> README.md
git init
git add README.md
git commit -m "first commit“
```

Now create new (**empty**) repository @github web page

And then….

```
git remote add origin https://github.com/Your_Account_name/Your_repository_name.git
git push -u origin master
```

## Git

```
alias gs="git status“
alias gc="git commit -m “
```

…

and many more☺

# Git

File classifications in Git

- Tracked – any file already in repo or staged for commit
- Ignored – file explicitly declared as invisible or ignored inside .gitignore file
- Untracked – any file not found in previous two categories

git ls-files – show all files being tracked

cat .gitignore

Let’s make .gitignore …. And commit it!

echo "*~" >.gitignore

## Git Useful commands

- git add .    DON'T DO THIS IN YOUR HOME DIRECTORY!!!

    - Adds all the files inside the local repository and stages them for commit.

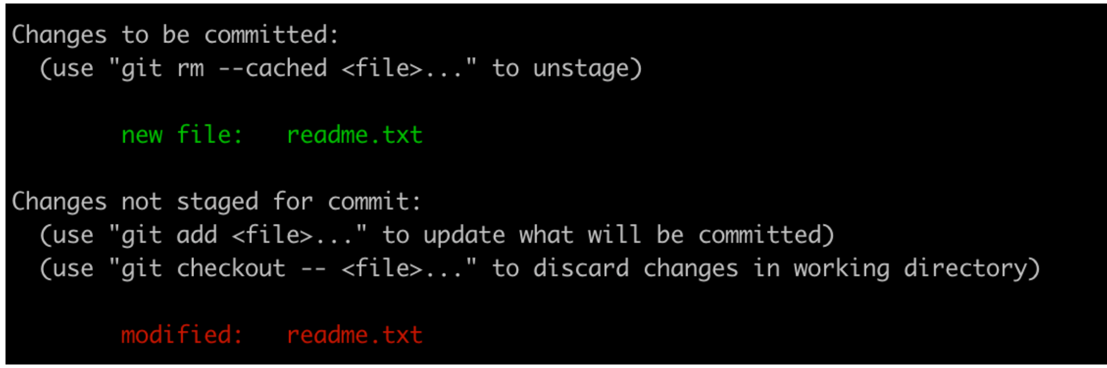
- git log - Show commit logs =(git log HEAD)
- git log + TAB

git config -l –  list the settings to all variables

# Dealing with “oh… sh*t… wait… too late”

- File prepared for commit with git add but you continue working without commiting…
  - git add readme.txt
  - git status

And here you continue working on readme.txt
  - git status
    - → you see readme.txt in 2 stages



```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   readme.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme.txt
```

- Solution
  - git add readme.txt → to update the file prepared for commit
  - git checkout -- readme.txt → to discard the new changes

# Dealing with “oh... sh*t... wait... too late”

- Committed but not pushed:
  - git add readme.txt
  - git commit -m "readme.txt with new text"
  - git reset origin/master

  or
  - git reset --hard origin/master

  or
  - git reset HEAD readme.txt

- hard = Any changes to tracked files in the working tree since <commit> are discarded.
- What is HEAD?
  - A pointer to the most recent commit on the current branch
  - HEAD^ = commit -1 (parent of the most recent commit)
  - HEAD^^= commit -2
  - HEAD~N=commit -N

## Dealing with “oh… sh*t… wait… too late”

- “Accidentally” deleted a file
  - rm readme.txt
  - git checkout readme.txt
- I don’t want this to be local git repository
  - rm -rf ./.git
- I have a typo in repository name
  - Change the repo name at webpage
  - git remote set-url origin https://github.com/user/NEW_NAME.git

  or
  - git remote rm origin
  - git remote add origin https://github.com/user/NEW_NAME.git

## Dealing with “what if…”

What if you want to change the name of the file in the repo?
What if you want to delete a file in the repo?
What if you want to delete a remote repo?

Use git commands:

- mv - Move or rename a file or a directory
  - Git doesn’t keep track of rename!!!
- rm - Remove files from the working tree and from the index

Don’t forget to push the changes!!

Try:

```
git mv old_name new_name
git rm file_name
```

## Dealing with “what if…”

If files a, b and c are changed, you SHOULD commit them separately!

However, we rarely do this so…

- git add -u = stage all tracked, modified files
- git commit -a -m “commiting all tracked, modified files”

## Dealing with “what if…”

- You just want to check the content from a specific commit:
- git log → get the commit hash
- git checkout commit_hash
    - See what you are looking for
- git checkout master → go back to last commit