R-Ladies Montpellier

Version control with GITHUB

Criscely Luján

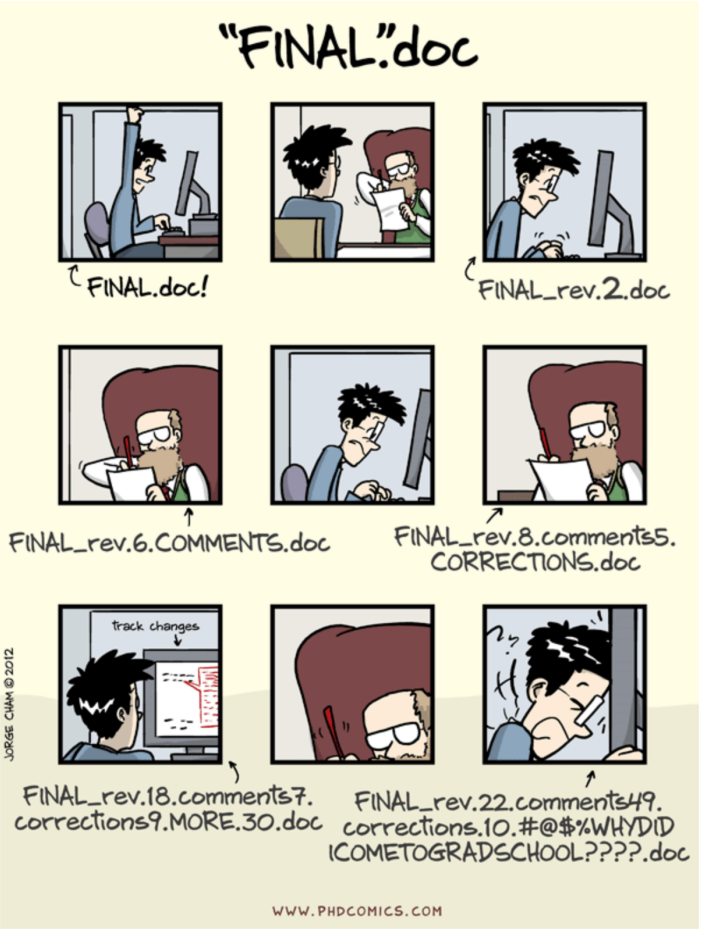
February 12, 2019

Keep in touch

Personal email: criscelylujan@gmail.com
Profesional email: criscely.lujan@ird.fr
Twitter: @CriscelyLP
GitHub: @CriscelyLP

Get start! YEAH!

Why is important version control?



phdcomics.com

Why is important version control?

Why is important version control?

The version control is important for:

Why is important version control?

The version control is important for:

- Storing version (properly)

Why is important version control?

The version control is important for:

- Storing version (properly)
- Restoring previous versions

Why is important version control?

The version control is important for:

- Storing version (properly)
- Restoring previous versions
- Collaborations

Why is important version control?

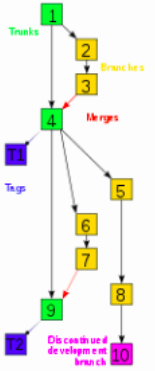
The version control is important for:

- Storing version (properly)
- Restoring previous versions
- Collaborations
- Save time!

Version control software

Version control software

V • T • E	Version control software	[hide]
Years, where available, indicate the date of first stable release. Systems with names *in italics* are no longer maintained or have planned end-of-life dates.		
Local only	**Free/open-source**	RCS (1982) · SCCS (1972)
	Proprietary	PVCS (1985) · *QVCS* (1991)
Client–server	**Free/open-source**	CVS (1986, 1990 in C) · CVSNT (1998) · QVCS Enterprise (1998) · Subversion (2000)
	Proprietary	AccuRev SCM (2002) · ClearCase (1992) · *CMVC* (1994) · Dimensions CM (1980s) · *DSEE* (1984) · Endevor (1980s) · Integrity (2001) · Panvalet (1970s) · Perforce Helix (1995) · SCLM (1980s?) · Software Change Manager (1970s) · StarTeam (1995) · Surround SCM (2002) · Synergy (1990) · Team Concert (2008) · Team Foundation Server (2005) · Visual Studio Team Services (2014) · Vault (2003) · *Visual SourceSafe* (1994)
Distributed	**Free/open-source**	*ArX* (2003) · BitKeeper (2000) · *Codeville* (2005) · Darcs (2002) · *DCVS* (2002) · Fossil (2007) · Git (2005) · *GNU arch* (2001) · GNU Bazaar (2005) · Mercurial (2005) · Monotone (2003) · Pijul (2015) · *SVK* (2003) · Veracity (2010)
	Proprietary	*TeamWare* (1990s?) · Code Co-op (1997) · Plastic SCM (2006) · Team Foundation Server (2013) · Visual Studio Team Services (2014)
Concepts	Baseline · Branch · Changeset · Commit · Data comparison · Delta compression · Fork (Gated commit) · Interleaved deltas · Merge · Repository · Tag · Trunk	
Category · Comparison · List		



Wikipedia

Why git?

Why git?

Git is a distributed version-control system for tracking changes in source code during the development of software.

Why git?

Git is a distributed version-control system for tracking changes in source code during the development of software.

Why git?

Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups

Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups
- Popular and Successful
  - Active development
  - Shiny new tools
  - Fast

Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups
- Popular and Successful
  - Active development
  - Shiny new tools
  - Fast
- Tracks any type of file
  - Works best with text

Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups
- Popular and Successful
  - Active development
  - Shiny new tools
  - Fast
- Tracks any type of file
  - Works best with text
- Branching
  - Smarter merges

Why git?

- Distributed
  - Work online or offline
  - Collaborate with large groups
- Popular and Successful
  - Active development
  - Shiny new tools
  - Fast
- Tracks any type of file
  - Works best with text
- Branching
  - Smarter merges

Slide taken from Mine Cetinkaya-Rundel.

GitHub Inc.

GitHub Inc.

A web-based hosting service for version control using Git.

GitHub

Download logo

Download mark

Download Octocat

GitHub Inc.

- Access to the control and collaboration features for every `project`.

GitHub Inc.

- Access to the control and collaboration features for every `project`.

- Work with public and private `repositories`.

GitHub Inc.

- Access to the control and collaboration features for every `project`.
- Work with public and private `repositories`.
- Develop a `networking`.

GitHub Inc.

- Access to the control and collaboration features for every `project`.
- Work with public and private `repositories`.
- Develop a `networking`.
- `Plans` for enterprise, teams, pro and free accounts.

GitHub Inc.

- Access to the control and collaboration features for every `project`.
- Work with public and private `repositories`.
- Develop a `networking`.
- `Plans` for enterprise, teams, pro and free accounts.
- Is the `largest host of source code` in the wordl! (28 million users and 57 million repositories until June 2018).

Register a GitHub account

Register a GitHub account

- Register an account with GitHub is free!

Register a GitHub account

- Register an account with GitHub is free!

- Free private repos
  - Students, faculty, and educational/research staff: GitHub Education
  - Official nonprofit organizations and charities: GitHub for Good

Register a GitHub account

- Register an account with GitHub is free!

- Free private repos
  - Students, faculty, and educational/research staff: GitHub Education
  - Official nonprofit organizations and charities: GitHub for Good

- Pay for private repos
  - $7/month: GitHub Pricing

Register a GitHub account

- Register an account with GitHub is free!

- Free private repos
  - Students, faculty, and educational/research staff: GitHub Education
  - Official nonprofit organizations and charities: GitHub for Good

- Pay for private repos
  - $7/month: GitHub Pricing

- Tips about the name acount:
  - Use your actual name!
  - Shorter is better than longer!
  - Be as unique as possible!
  - Reuse your name from other context, e.g. `Twitter` or `Slack`!

Git is already installed?

Git is already installed?

To check that go to shell (terminal / command line / console) and enter `which git` to request the path to your Git executable:

Git is already installed?

To check that go to shell (terminal / command line / console) and enter `which git` to request the path to your Git executable:

```
which git
```

```
## /usr/bin/git
```

Git is already installed?

To check that go to shell (terminal / command line / console) and enter `which git` to request the path to your Git executable:

```
which git
```

```
## /usr/bin/git
```

Then enter `git --version` to see its version

```
git --version
```

```
## git version 2.14.5
```

Git is already installed?

To check that go to shell (terminal / command line / console) and enter `which git` to request the path to your Git executable:

```
which git
```

```
## /usr/bin/git
```

Then enter `git --version` to see its version

```
git --version
```

```
## git version 2.14.5
```

- **If git is not installed YET**: See Install git to follow the correct steps to install git according your system operative! :)

Introduce yourself to Git

Let `git` know how you are following this simple configuration steps!

Introduce yourself to Git

Let `git` know how you are following this simple configuration steps!

```
$ git config --global user.name "Criscely Lujan"
$ git config --global user.email "criscelylujan@gmail.com"
$ git config --global core.editor vim
$ git config --global --list
```

Introduce yourself to Git

Let `git` know how you are following this simple configuration steps!

```
$ git config --global user.name "Criscely Lujan"
$ git config --global user.email "criscelylujan@gmail.com"
$ git config --global core.editor vim
$ git config --global --list
```

- `user.name` can be your GitHub username. Your commits will be labelled with this name, so this should be informative!

Introduce yourself to Git

Let `git` know how you are following this simple configuration steps!

```
$ git config --global user.name "Criscely Lujan"
$ git config --global user.email "criscelylujan@gmail.com"
$ git config --global core.editor vim
$ git config --global --list
```

- `user.name` can be your GitHub username. Your commits will be labelled with this name, so this should be informative!
- `user.email` must be the email that you use to sign up for GitHub.

Introduce yourself to Git

Let `git` know how you are following this simple configuration steps!

```
$ git config --global user.name "Criscely Lujan"
$ git config --global user.email "criscelylujan@gmail.com"
$ git config --global core.editor vim
$ git config --global --list
```

- `user.name` can be your GitHub username. Your commits will be labelled with this name, so this should be informative!
- `user.email` must be the email that you use to sign up for GitHub.
- There are diverse options of Git editor.

Git Clients

Git Clients

- Git and Git client `are not` the same! Like R and RStudio is not the same thing! 💟

Git Clients

- Git and Git client `are not` the same! Like R and RStudio is not the same thing! 💟

- Git client:
  - Integrated development environment!
  - Make the experience more pleasant providing a richer visual representation!

Git Clients

- Git and Git client `are not` the same! Like R and RStudio is not the same thing! 💗

- Git client:
  - Integrated development environment!
  - Make the experience more pleasant providing a richer visual representation!

- Some Git clients:
  - SourceTreen
  - GitKraken
  - GitUp
  - SmartGit
  - git-cola
  - ... others...
  - **RStudio**

Almost ready!

How authenticating yourself with GitHub?

How authenticating yourself with GitHub?

There are two options of protocols for secure communication working over a computer network!

How authenticating yourself with GitHub?

There are two options of protocols for secure communication working over a computer network!

- Hypertext Transfer Protocol Secure (HTTPS)

How authenticating yourself with GitHub?

There are two options of protocols for secure communication working over a computer network!

- Hypertext Transfer Protocol Secure (HTTPS)

If you plan to push/pull using HTTPS protocol, you can follow Cache credentials for HTTPS for more information.

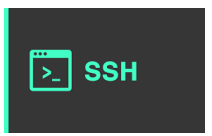
How authenticating yourself with GitHub?

There are two options of protocols for secure communication working over a computer network!

- Hypertext Transfer Protocol Secure (HTTPS)

If you plan to push/pull using HTTPS protocol, you can follow Cache credentials for HTTPS for more information.

- Secure Shell (SSH)



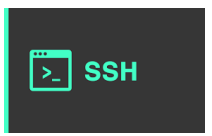
How authenticating yourself with GitHub?

There are two options of protocols for secure communication working over a computer network!

- Hypertext Transfer Protocol Secure (HTTPS)

If you plan to push/pull using HTTPS protocol, you can follow Cache credentials for HTTPS for more information.

- Secure Shell (SSH)



If you plan to push/pull using SSH protocol, you can follow Set up keys for SSH for more information.

RStudio and GitHub

RStudio and GitHub

Setup Git + RStudio + GitHub

RStudio and GitHub

Setup Git + RStudio + GitHub

- Tools / Global Options / Select Git/SVN tab.
- Create the RSA key.
- Click, View public key, and copy the displayed public key.
- Save the key on your GitHub account: Settings / SSH key / Add SSH key.

Step by step here: Connecting RStudio and GitHub

Now we can really have fun!

Make a new repo

- Go to https://github.com and login. Click in the green box called `New repository`.

Make a new repo

- Go to https://github.com and login. Click in the green box called `New repository`.
- If you are on your own profile page, go to the section `Repositories`, then click the green box called `New`.

Make a new repo

- Go to https://github.com and login. Click in the green box called `New repository`.
- If you are on your own profile page, go to the section `Repositories`, then click the green box called `New`.
- Assign a name for the `repository` and include a `description` (this is optional but is recommended!).

Make a new repo

- Go to https://github.com and login. Click in the green box called `New repository`.
- If you are on your own profile page, go to the section `Repositories`, then click the green box called `New`.
- Assign a name for the `repository` and include a `description` (this is optional but is recommended!).
- Public or private.

Make a new repo

- Go to https://github.com and login. Click in the green box called `New repository`.
- If you are on your own profile page, go to the section `Repositories`, then click the green box called `New`.
- Assign a name for the `repository` and include a `description` (this is optional but is recommended!).
- Public or private.
- Initializa the repository using the `README`.

Create a new repository

A repository contains all project files, including the revision history.

Owner **Repository name** *

CriscelyLP /

Great repository names are short and memorable. Need inspiration? How about **stunning-system**?

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

Clone a repo

Using RStudio

- File -> New Project

Clone a repo

Using RStudio

- File -> New Project
- Version control

Clone a repo

Using RStudio

- File -> New Project
- Version control
- Git

Clone a repo

Using RStudio

- File -> New Project
- Version control
- Git
- Fill `Repository URL` and project name (what you want the folder to be called locally).

Clone a repo

Using terminal

- Create a new directory, open it and perform `git init` to create a new git repository:

```
git init
```

Clone a repo

Using terminal

- Create a new directory, open it and perform `git init` to create a new git repository:

```
git init
```

- Clone the repository running the command `git clone` plus the path:

```
git clone /path/to/repository
```

Clone a repo

Using terminal

- Create a new directory, open it and perform `git init` to create a new git repository:

```
git init
```

- Clone the repository running the command `git clone` plus the path:

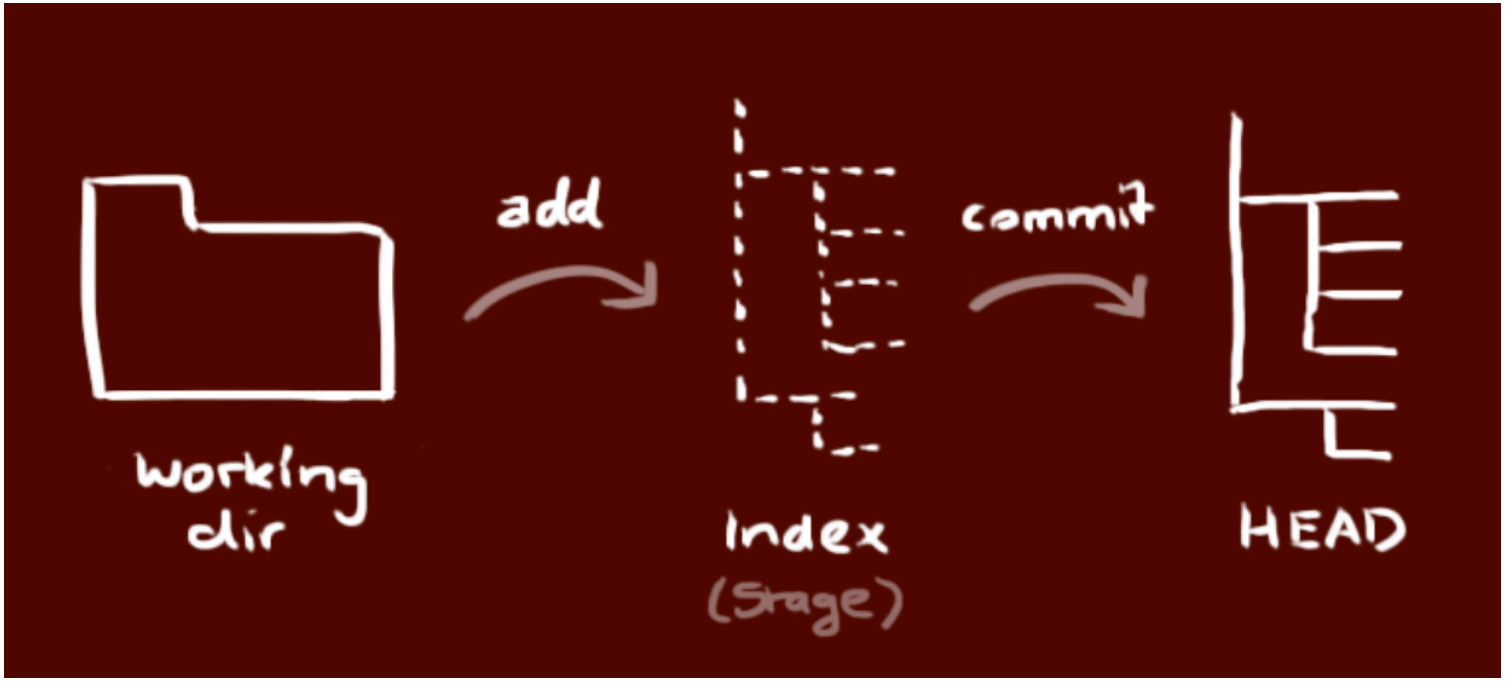
```
git clone /path/to/repository
```

- When you are using a remote server, your command will be: `git clone username@host:/path/to/repository`
- For example:

```
git clone git@github.com:r-ladies-montpellier/20181212-RLadiesMontpellier-Meetup1.
```

Workflow

Your local repository consists of three "trees" maintained by git.



Add and commit

- After clone the repository already clone on your computer, you can make changes and propose changes using `add` and `commit`:

For specific file:

```
git add <filename>
```

Add and commit

- After clone the repository already clone on your computer, you can make changes and propose changes using `add` and `commit`:

For specific file:

```
git add <filename>
```

Or adding all changes:

```
git add .
```

Add and commit

- After clone the repository already clone on your computer, you can make changes and propose changes using `add` and `commit`:

For specific file:

```
git add <filename>
```

Or adding all changes:

```
git add .
```

Commit!

```
git commit -m "Commit message"
```

Add and commit

- After clone the repository already clone on your computer, you can make changes and propose changes using `add` and `commit`:

For specific file:

```
git add <filename>
```

Or adding all changes:

```
git add .
```

Commit!

```
git commit -m "Commit message"
```

Now the file (or changes in general) is committed to the `HEAD`, but not in your remote repository YET!

Pushing changes

Pushing changes

When your changes are in `HEAD`, your are ready to sent those changes to your remote repository, executing:

Pushing changes

When your changes are in `HEAD`, your are ready to sent those changes to your remote repository, executing:

```
# If your are not working in "master", change the branch
git push origin master
```

Pushing changes

When your changes are in `HEAD`, your are ready to sent those changes to your remote repository, executing:

```
# If your are not working in "master", change the branch
git push origin master
```

If you dont know that is a `branch`, the next slide is for YOU! 

Branches are used to develop features ISOLATED from each other!

Branching

Branching

- The `master` branch is the default branch when you create a repository.

Branching

- The `master` branch is the default branch when you create a repository.

- We can create other branch for the development of other features (example: `develop` branch).

Branching

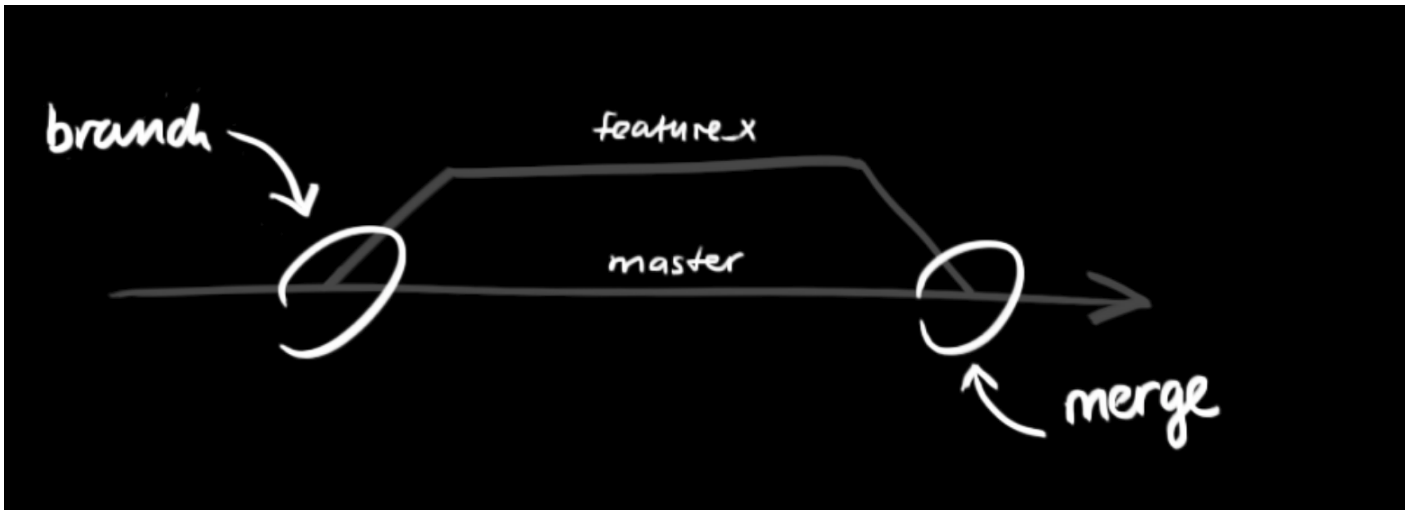
- The `master` branch is the default branch when you create a repository.
- We can create other branch for the development of other features (example: `develop` branch).
- We can merge branches.

Branching

- The `master` branch is the default branch when you create a repository.

- We can create other branch for the development of other features (example: `develop` branch).

- We can merge branches.



Branching

Create a new branch named `feature_x` and switch to it using :

```
git checkout -b feature_x
```

Branching

Create a new branch named `feature_x` and switch to it using :

```
git checkout -b feature_x
```

Switch back to `master`:

```
git checkout master
```

Branching

Create a new branch named `feature_x` and switch to it using :

```
git checkout -b feature_x
```

Switch back to `master`:

```
git checkout master
```

To delete the brach `feature_x`:

```
git branch -d feature_x
```

Branching

Create a new branch named `feature_x` and switch to it using :

```
git checkout -b feature_x
```

Switch back to `master`:

```
git checkout master
```

To delete the brach `feature_x`:

```
git branch -d feature_x
```

IMPORTANT!!!!! Any branch in your local repository is available to others users unless you push the branch to the remore repository, doing:

```
git push origin feature_x
```

Update and merge

Update and merge

To update your local repository to the newest commit, execute:

```
git pull
```

Update and merge

To update your local repository to the newest commit, execute:

```
git pull
```

To merge another branch into your active branch (for example: master), use:

```
git merge <branch>
```

Update and merge

To update your local repository to the newest commit, execute:

```
git pull
```

To merge another branch into your active branch (for example: master), use:

```
git merge <branch>
```

Before marging changes in branch you can also see the differences!

```
git diff <source_branch> <target_branch>
```

Update and merge

To update your local repository to the newest commit, execute:

```
git pull
```

To merge another branch into your active branch (for example: master), use:

```
git merge <branch>
```

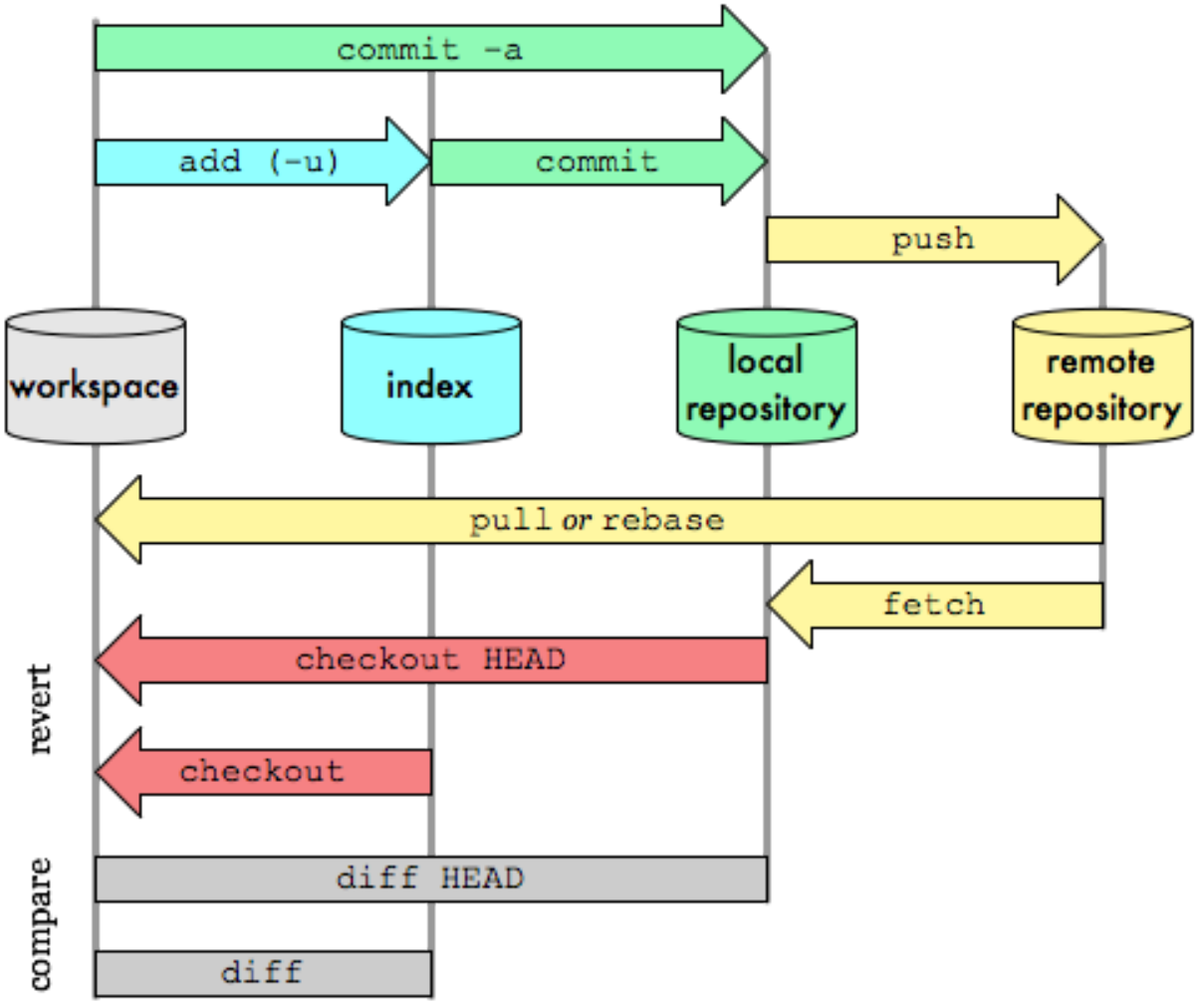
Before marging changes in branch you can also see the differences!

```
git diff <source_branch> <target_branch>
```

More information about git commands here: Git the Simple guide

Git Data Transport Commands

http://osteele.com



Exercise

- Create a repo in GitHub.
- Clone the repo in your computer.
- Make changes (editting also the README.md and .gitignore)
- Make commits, if is possible include emojis to have fun! :)
- Use: push / pull / merge / status / etc.
- Check the changes in you remore repository.
- Develop a new branch, delete it, play, enjoy and ...

...be in problems, we are here to help you!

Follow the R-Ladies Montpellier in GitHub.

Thanks!

- R-Ladies Global for the help and support.
- Jenny Bryan: Happy Git with R.
- Slides created via the R package **xaringan** with the **R-Ladies theme**
- Cowork'in Montpellier