Source code management with Git

Fabien Badeig

Institut Henri Fayol

2020

Table of contents

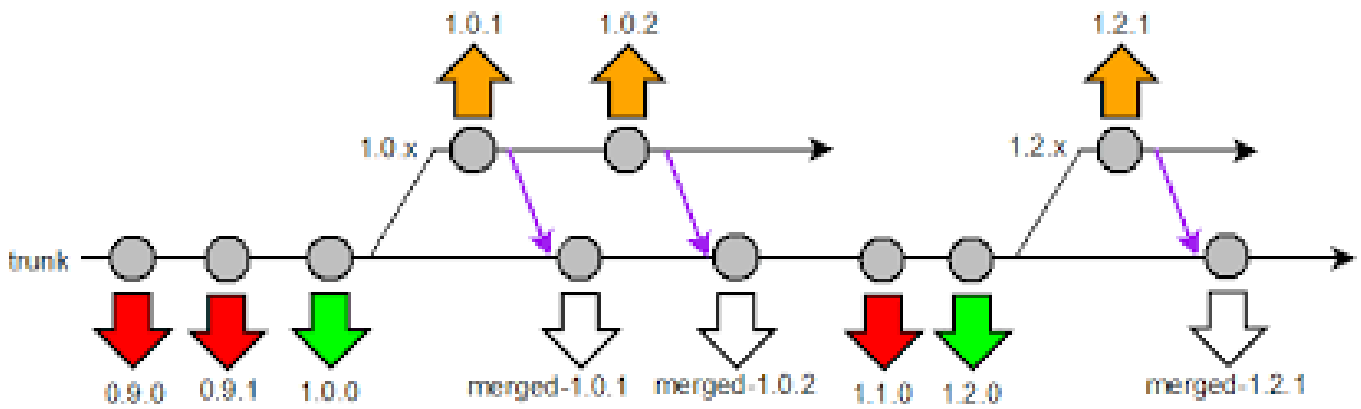
Definition

Git is a distributed version control system. It is a tool to work :

Git

- with others: concurrent access
- over time: modification history, tag management, no code loss, ...
- remotely: with customized local environments and keeping track of the modifications

on a set of files composing a project.

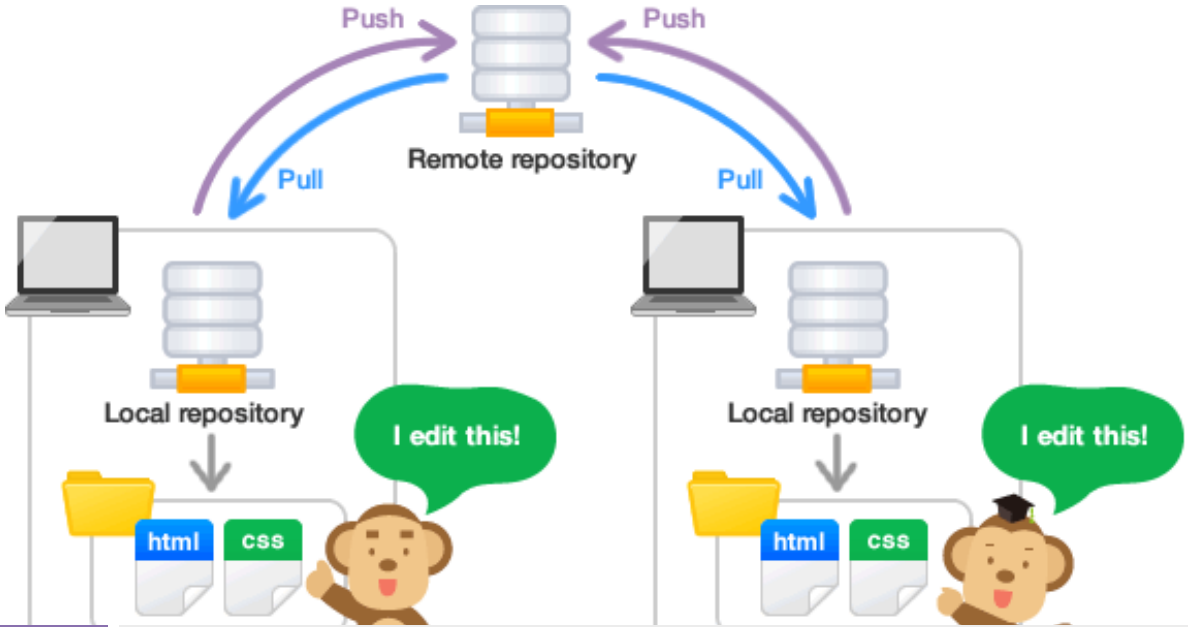


How it works

Structure

Git is organized in three layers

1. remote repository (on a server)
2. local repository (on your machine)
3. working directory (on your marchine)



Basic commands

Commands

- Copy repository on your machine:

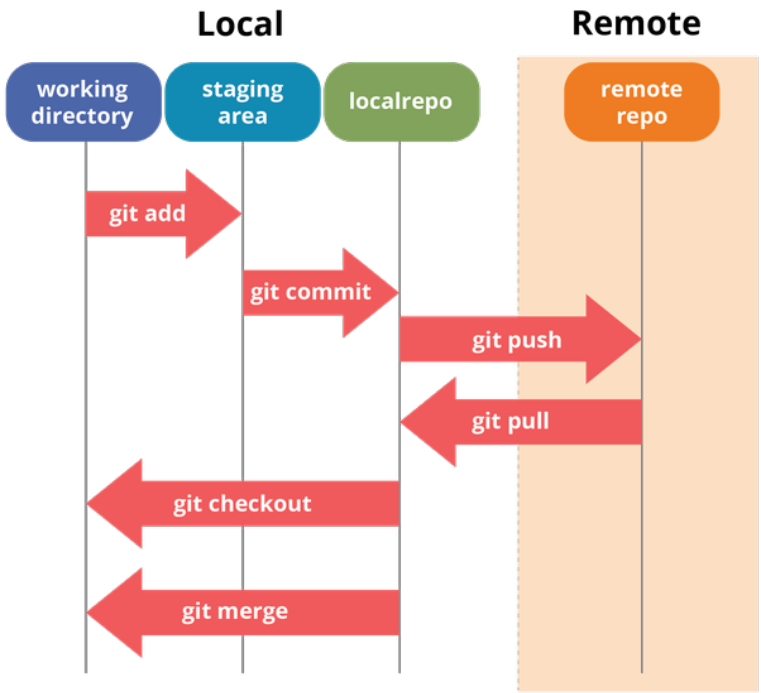
```
>$ git clone <url>
```

- Update remote repository:

```
>$ git add <files>
>$ git commit <files>
   -m "<description>"
>$ git push <server>
            <branche>
```

- Update local and work repository:

```
>$ git pull
```



Projects

Context

- Several developers
- One instance for test
- One instance for production ("stable release")
- Updates (bug fixes)
- New features

Rules

1. **ONLY ONE** maintainer who manages the releases
2. **NEVER** commit on the master branch
3. **NEVER** rebase from the master branch to another branch
4. Respecting the development workflow

Management

Workflow: bug fix/new features

- Create a branch from master
- Commit its modifications to its branch
- Check that its code still works with the master code with a rebase from its branch to master

```
>$ (master) git checkout -b featureA
>$ (featureA) git commit -a -m "featureA part 1"
>$ (featureA) git rebase master
```

Maintainer tasks

Updating master after validation of a new feature

- (1-2) Move to the master branch and merge the featureA branch
- (3-4) Delete the featureA branch if the merge was successful

```
1$ (XXX) git checkout origin master
2$ (master) git merge --no-ff featureA
3$ (master) git branch -d featureA
4$ (master) git push origin :featureA
```

Useful commands

Commands

- Repository status :

```
>$ git status
```

- Comparing the contents of a file on the working copy with its version on the repository:

```
>$ git diff <file>
```

- Log history:

```
>$ git log
```

New branch

Create a new branch

- The option -u tells git to create all the information needed to make the branch traceable.
- The command checkout positions the project on the new branch.

```
$ git git checkout -b <the new branch>
$ git push -u origin <the new branch>
```

Cloning in a non-empty directory

Cloning in a non-empty directory

- To just download the repository

```
$ cd <path/folder>
$ git init
$ git pull <url to my repo git>
```

- To grab all information related to the repository

```
$ cd <path/folder>
$ git init
$ git remote add origin <git@url:yourname/yourproject.git>
$ git fetch
$ git checkout master
$ git pull
```

Graphic GIT tools (client side)

For windows

- GitBash : `https://gitforwindows.org/`
- GitHub : `https://desktop.github.com/`
- TortoiseGit : `https://tortoisegit.org/`

For linux

- *git-gui* + *gitk* (graphic tool of history, managing git log and git grep)
- GitKraken : `https://www.gitkraken.com/`