The very basics of Git

`git` for the non-programmer

Xavier Sánchez Díaz

sax@tec.mx

Research Group with Strategic Focus on Intelligent Systems
Tecnológico de Monterrey, Campus Monterrey

Outline

Everything is a file

The UNIX philosophy

- Source code is stored in files.

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.
- HDDs and USBs are files.

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.
- HDDs and USBs are files.

Everything

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.
- HDDs and USBs are files.

Everything is

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.
- HDDs and USBs are files.

Everything is a

Everything is a file

The UNIX philosophy

- Source code is stored in files.
- Word documents are files.
- Photos are files.
- Videos are files.
- HDDs and USBs are files.

Everything is a file.

Everything is a file

The UNIX philosophy

Since everything is a file, it makes sense to keep track of the changes of your projects using `git`, no matter what you're working on!

Changes are labeled with a unique identifier, a message and then linked to the user (both name and email) who submitted the modification.

A user can also look and revisit any state of the project throughout its lifetime, and see the differences between two versions of a file at a given state.

What is `git`?

And why should I use it?

`git` is a free and open source distributed version control system.

What is `git`?

And why should I use it?

`git` is a free and open source distributed version control system.

Version control system

A system that allows easy management project versions.

What is `git`?

And why should I use it?

`git` is a free and open source distributed version control system.

Version control system

A system that allows easy management project versions.

Distributed

Not centralized. Everyone on the team has their own local copy.

What is `git`?

And why should I use it?

`git` is a free and open source distributed version control system.

Version control system

A system that allows easy management project versions.

Distributed

Not centralized. Everyone on the team has their own local copy.

Free and Open Source

Git is distributed according to a GPLv2 License.

Free speech, not free beer

What is `git`?

Git is free software since it respects the four essential freedoms of the user:

Free speech, not free beer

What is `git`?

Git is free software since it respects the four essential freedoms of the user:

- Freedom to use the program as you wish, for any purpose.

Free speech, not free beer

What is `git`?

Git is free software since it respects the four essential freedoms of the user:

- Freedom to use the program as you wish, for any purpose.
- Freedom to modify the software as you want. The source code is there for you to look at and modify at will.

Free speech, not free beer

What is `git`?

Git is free software since it respects the four essential freedoms of the user:

- Freedom to use the program as you wish, for any purpose.
- Freedom to modify the software as you want. The source code is there for you to look at and modify at will.
- Freedom to share or redistribute copies as you wish to help your neighbor.

Free speech, not free beer

What is `git`?

Git is free software since it respects the four essential freedoms of the user:

- Freedom to use the program as you wish, for any purpose.
- Freedom to modify the software as you want. The source code is there for you to look at and modify at will.
- Freedom to share or redistribute copies as you wish to help your neighbor.
- Freedom to share or redistribute copies of your modified versions to others.

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

- What changed since the last version?

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

- What changed since the last version?
- When was the last modification?

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

- What changed since the last version?
- When was the last modification?
- Who was the last person on the team to modify this file?

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

- What changed since the last version?
- When was the last modification?
- Who was the last person on the team to modify this file?
- Which version is the one I'm looking for?

What is `git`?

And why should I use it?

A version control system tracks changes on projects. It is a useful tool to keep track of the following:

- What changed since the last version?
- When was the last modification?
- Who was the last person on the team to modify this file?
- Which version is the one I'm looking for?

So you can avoid having filenames like
`Projectfinal3lastrev18.doc` or
`Projectfinal(this_is_the_one).pdf`.

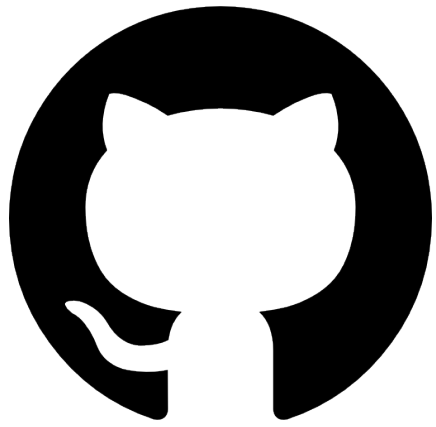
`git` is not GitHub I

Let's get it right

Git is the software: a technology and a workflow methodology.

`git` is not GitHub II

Let's get it right



GitHub is a website that offers 'free' (as in free beer) hosting of open source projects. GitHub uses the Git methodology. There are plenty of alternatives to GitHub.

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes
2. Prepare and stage those changes

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes
2. Prepare and stage those changes
3. Accept and label the changes

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes
2. Prepare and stage those changes
3. Accept and label the changes
4. Submit your labeled changes

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes
2. Prepare and stage those changes
3. Accept and label the changes
4. Submit your labeled changes
5. Your changes are accepted and incorporated to the project

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes
2. Prepare and stage those changes
3. Accept and label the changes
4. Submit your labeled changes
5. Your changes are accepted and incorporated to the project

6. Everyone updates their copy to the newest version

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes to your files
2. Prepare and stage those changes in your computer
3. Accept and label the changes in your computer
4. Submit your labeled changes
5. Your changes are accepted and incorporated to the project

6. Everyone updates their copy to the newest version

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes to your files
2. Prepare and stage those changes in your computer
3. Accept and label the changes in your computer
4. Submit your labeled changes to the master repository
5. Your changes are accepted and incorporated to the project on the master repository
6. Everyone updates their copy to the newest version

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes to your files
2. Prepare and stage those changes in your computer
3. Accept and label the changes in your computer
4. Submit your labeled changes to the master repository
5. Your changes are accepted and incorporated to the project on the master repository
6. Everyone updates their copy to the newest version on their computers

How does it work?

The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

1. Make some changes to your files
2. Prepare and stage those changes in your computer
3. Accept and label the changes in your computer
4. Submit your labeled changes to the master repository
5. Your changes are accepted and incorporated to the project on the master repository
6. Everyone updates their copy to the newest version on their computers

How does it work?

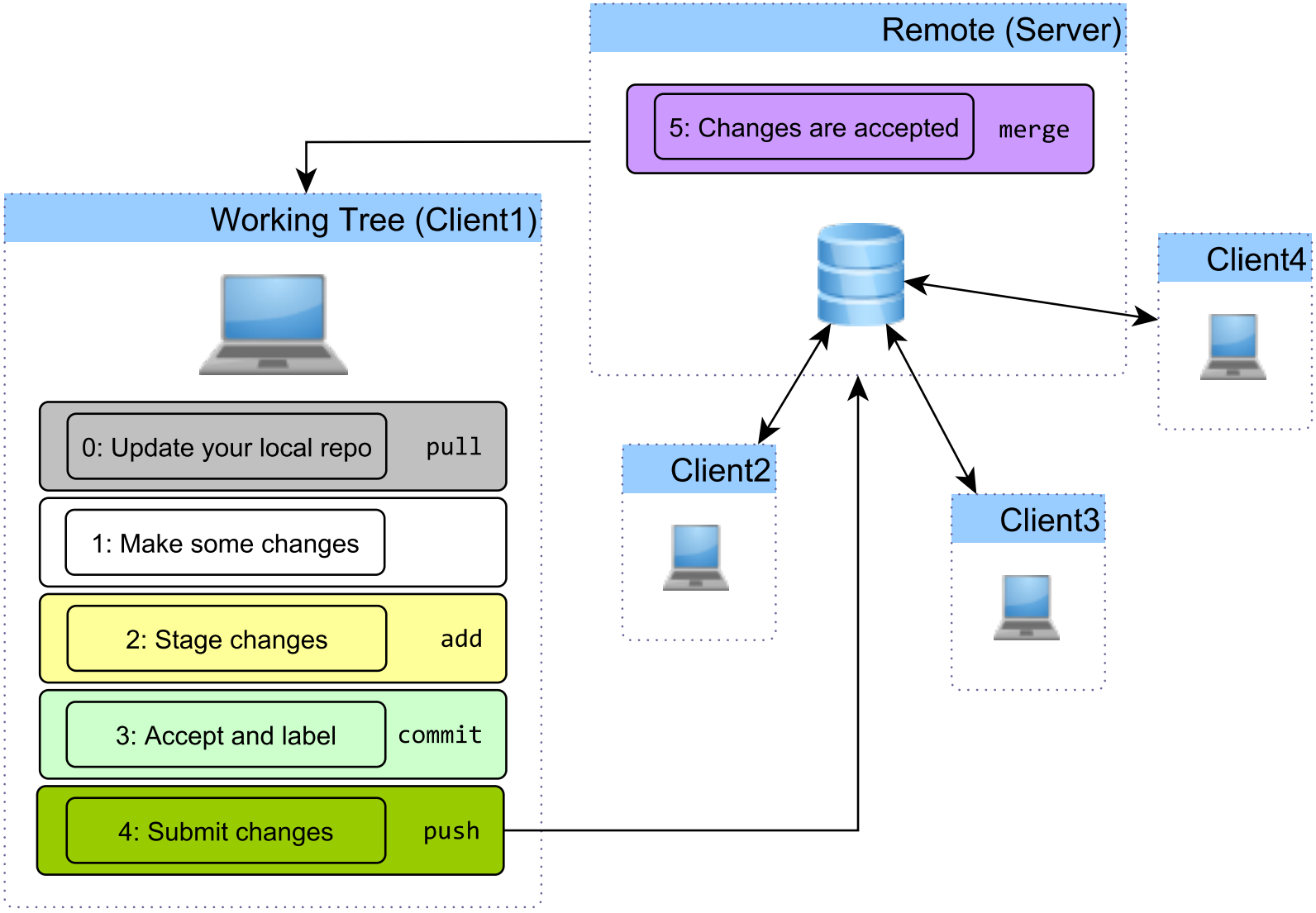
The Git workflow

Assume you're in charge of a feature. Then, the process would be similar to the following:

0. Update your copy to the newest version on your computer
1. Make some changes to your files
2. Prepare and stage those changes in your computer
3. Accept and label the changes in your computer
4. Submit your labeled changes to the master repository
5. Your changes are accepted and incorporated to the project on the master repository

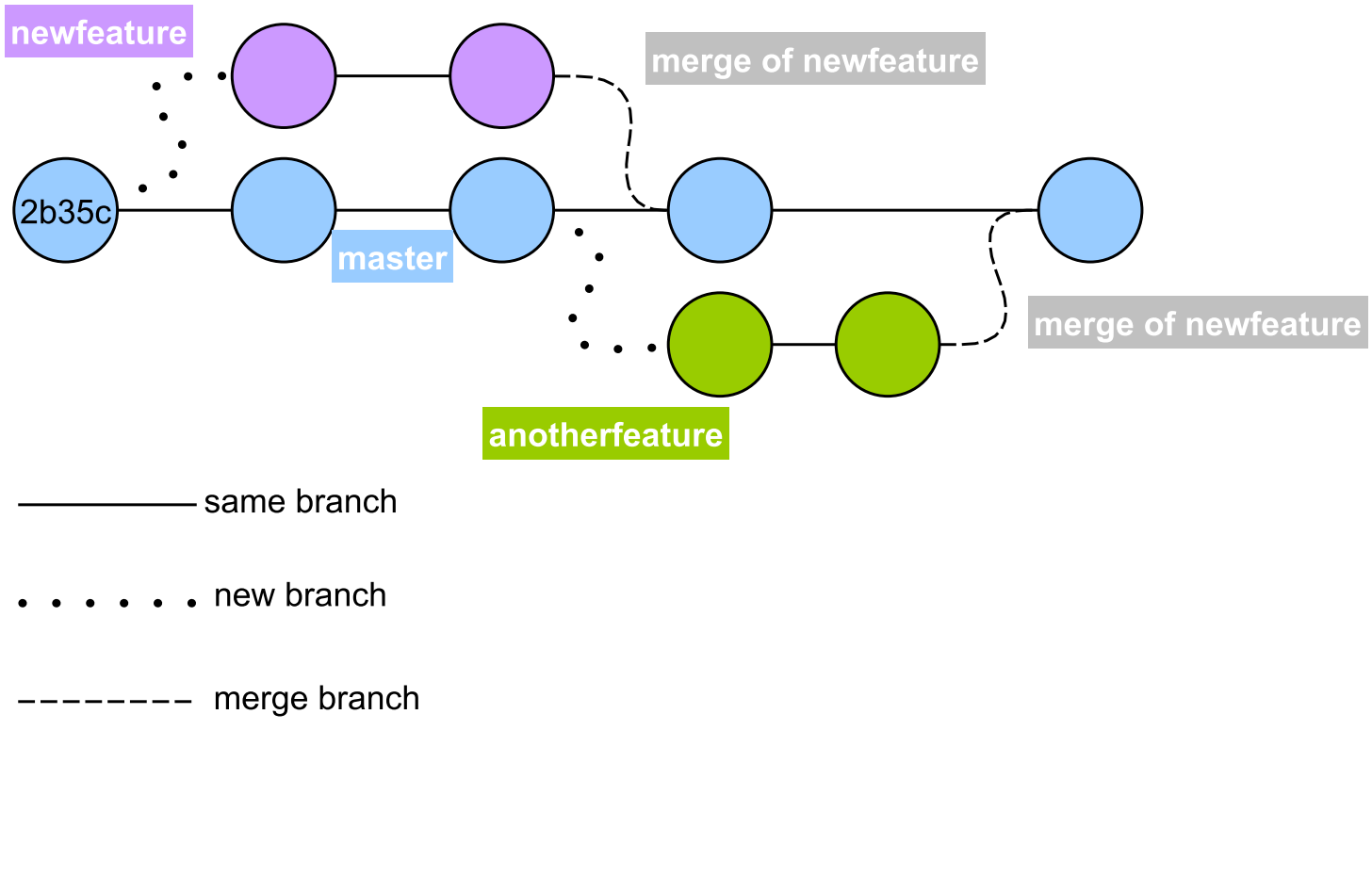
What and where?

The Git workflow



Branching

The Git workflow



`git` command syntax

The git workflow

Git is usually run in a UNIX-like console. As such, you invoke commands like this:

`git` command syntax

The git workflow

Git is usually run in a UNIX-like console. As such, you invoke commands like this:

```
$ git <command> [arguments]
```

`git` command syntax

The git workflow

Git is usually run in a UNIX-like console. As such, you invoke commands like this:

```
$ git <command> [arguments]
```

The most common commands are `init`, `clone`, `add`, `commit`, `push`, `pull`, `checkout`, `branch`, `log`, `remote` and `diff`.

`git` command syntax

The git workflow

Git is usually run in a UNIX-like console. As such, you invoke commands like this:

```
$ git <command> [arguments]
```

The most common commands are `init`, `clone`, `add`, `commit`, `push`, `pull`, `checkout`, `branch`, `log`, `remote` and `diff`.

All commands have different arguments (or **options**). You can check the available options (and the syntax) using the `--help` option for any of the available commands in git:

`git` command syntax

The git workflow

Git is usually run in a UNIX-like console. As such, you invoke commands like this:

```
$ git <command> [arguments]
```

The most common commands are `init`, `clone`, `add`, `commit`, `push`, `pull`, `checkout`, `branch`, `log`, `remote` and `diff`.

All commands have different arguments (or **options**). You can check the available options (and the syntax) using the `--help` option for any of the available commands in git:

```
$ git add --help
```

Now with commands!

Recap

Update your copy to the newest version with `git pull <repository> [branch]`.

Make some changes by editing and saving your work. You have some **unstaged** changes by now.

Stage those changes using `git add <file>`. Your files are now **staged** and ready to be labeled.

Label the changes by using `git commit [options]`, your work is now **committed** with a hash (SHA-1) and linked to your user.

Submit your labeled changes to the master repository by using `git push [options]`.

Your changes are incorporated to the project by merging the branch you worked on, with `git merge [options]`. This is usually done by an administrator.

That's it!

For more information

Be sure to check out the documentation for `git` in its official webpage. They also have a free book!

You can also take a look at the GitHub, GitLab and Bitbucket `git` tutorials.