After this lesson you will be able to:

- Understand what a version control system is
- Understand the advantages of using Git
- Create a new repository
- Clone a repository
- Check the status of a repository
- Add files to a commit
- Commit files
- Push files into a remote repository
- See and understand the git log
- Create an account in GitHub
- GitHub create account, create repo, clone repo, add files, commit files, push files, git log

What is Git?

Web development projects usually require the effort of more than one programmer. During web applications life cycle a lot of changes occurs: new features, bugs management, re-factorization of the code… The result?

New releases, code that has to be organized and accesible for everyone while we track changes and a easy and practical way of putting new code together without breaking what is already there.

Git is a free and open source distributed version control system. This means Git can do all of the tasks needed above.

A version control system is software that helps developers to track changes and distribute their code.

Git advantages

Distributed architecture

Git is an example of a DVCS (Distributed Version Control System). This means that rather than have one single place for the full version history of the software, every developer gets to have his own working copy of the code and this copy is also a repository with the full history of all changes ever done to that version of the code.

Flexibility

Git allows us to have various development workflows, so it adapts to any project size. It also provides compatibility with a lot of existing systems and protocols.

Everyone knows Git

Git has the functionality, performance, security and flexibility that most teams and individual developers need, that's why everyone uses it. Also, a lot of developers already have Git experience, so it is a de facto standard.

Git is open source

Git is an open source project with more than ten years of evolution. The project maintainers have shown balanced judgment and a mature approach to meeting the long term needs of its users with regular releases that improve usability and functionality.

This also means Git community is huge and a lot of good quality documentation is available on the Internet, through books, tutorials and dedicated websites.

Let's practice a little bit with Git!

Creating a Repository

Git manages different projects with different repositories. We generally call them `repos` for short :)

To create a new repo in your project, open your terminal and go to **your project folder**. Right in this folder, at the top of your project, type:

Copy

```
$ git init
```

```
Initialized empty Git repository in [your .git directory path]
```

 You don't need to type the `$` symbol. It is provided by your terminal to let you know where you should type your commands.

This command creates a hidden directory `.git` in that folder. This hidden directory is where Git operates and stores it's data, so right now, it has an empty repository in it.

Type `ls -la` in your terminal to see the new folder.

Checking the Status

Next up, let's type the `git status` command to see what the current state of our project is:

```
$ git status

nothing to commit (create/copy files and use "git add" to
track)
```

```
nothing to commit (create/copy files and use "git add" to
track)
```

 It’s healthy to run git status often. Sometimes things change and you don’t notice it.

Adding and committing changes

Create a file called index.html

```
touch index.html
```

Run the git status command again to see how the repository status has changed:

```
git status
```

```
On branch master
```

```
Initial commit
```

```
Untracked files:


  (use "git add <file>..." to include in what will be
committed)


              index.html


nothing added to commit but untracked files present (use "git
add" to track)
```

As you can see, Git is now detecting changes -in this case, a new file. Also, it detects that this new file is not being tracked and Git suggests how to add this file in the next commit.

A Git commit works a lot like *taking a picture*. It takes a snapshot of the files added to the commit.

This is a very important concept to understand. Imagine it's your brother's birthday and you are the unofficial photographer. You get to choose which family members are going to be included in each picture. You select some of them, ask them to stay still somewhere in the house and then you take a picture. With Git, family members are your project files, when you ask them to stay in a certain place you're adding and when you take the picture you're committing the changes.

You are not making copies of the files you're tracking. You are just creating a snapshot of the files you added.

Adding the file

Now, let's add index.html to the tracked group.

Copy

```
$ git add index.html
```

```
$ git status
```

```
On branch master
```

```
Initial commit
```

```
Changes to be committed:
```

```
  (use "git rm --cached <file>..." to unstage)
```

```
            new file:   index.html
```

If you need to add every file in your filesystem, you can type:

Copy

```
$ git add .
```

As you can see in your Git status output, you can now also remove the file from the tracking group. This is known as *unstage* the changes (to add them is to *stage* the changes to the commit).

Committing the files

To commit your changes simply type

Copy

```
$ git commit -m "Add the index.html file"
```

```
[master (root-commit) d100e63] Add the index.html file
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 index.html
```

You could type `git status` again to check your repository status now.

Git log

So we've made a commit. But if we have several commits, we would like to be able to see the history of changes we've done. Think of Git's log as a journal that remembers all the changes we've committed so far, in the order we committed them. Try running it now:

Copy

```
$ git log
```

The output will be an opened txt file something like this

Copy

```
commit d100e63a4266ccaf1c0fe0e0c8e65b40440e4327

Author: fontcuberta <isafontcu@gmail.com>

Date:   Sat Oct 22 12:42:22 2016 +0200

    Add the index.html file
```

The log file will open in your terminal with the less program and to exit simply type q.

Remote repositories

To collaborate on any Git project, you need to know how to manage your remote repositories.

Remote repositories are versions of your project that are hosted on the Internet or network somewhere. Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work.

GitHub

GitHub is a code-hosting platform. It lets you and others work together on projects from anywhere.

Before anything else, if you haven't done it yet, you should go now to github.com and create an account.

GitHub is NOT **Git**. Git is a version control software that sets repositories. GitHub is a hosting platform that understands Git. Git runs on your local machine, whereas GitHub runs on the Internet. Keep in mind some people consider GitHub as a Social Network to share code.

Clone a GitHub repository

To clone a GitHub repository is to copy the entire project to your local computer. Basically, it is a combination of:

- `git init` (create the local repository)
- `git remote add` (add the URL to that repository)
- `git fetch` (fetch all branches from that URL to your local repository)
- `git checkout` (create all the files of the main branch in your working tree)

For now, forget about the fetching and the checkout. We will learn about branches in another lesson.

But you should know that `git clone` already initializes your local repository and adds the URL of the remote repository so the system knows where to put the changes and from where get the changes while you work.

Let's clone a repository from Ironhack's account. To do this, move to the directory where you want to have your project folder and type this:

```
git clone https://github.com/EcaCosca/lab-
htmlbuildingpractice.git
```

Summary

In this LU you've learned what Git and GitHub are, how to initialize repositories, stage and unstage existing files, commit changes, and read the log. Also, you have your GitHub account already created and you cloned a real project repository in your local computer.

Extra Resources

- Cheat Sheet - This cheat sheet features the most important and commonly used Git commands for easy reference.

Steps:

1. FORK
2. Check name, Clone > Click Clipboard
3. On Terminal
   cd Desktop\Tuwaiq\labs
4. git clone <paste url from the clipboard>
5. cd <name of cloned folder >
6. code .
7. Change something in the file
8. git status
9. Git add .
10. Git commit -m "First Commit"
11. git push origin master
12. Pull requests > create new pull request > (fill name with:
    [FS102021] Your Name
13. Leave a comment > create pull request