CENTRE ESPECÍFIC D´EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

UNIT 5.
VERSION CONTROL SYSTEM

Web Applications Deployment
CFGS DAW

Author: Carlos Cacho López

Reviewed by: Lionel Tarazón Alcocer

lionel.tarazon@ceedcv.es

2019/2020

License

Nomenclature

During this unit we are going to use special symbols to distinct some important elements.

This symbols are:

Important

Attention

Interesting

INDEX

1.Introduction...4
2.Repositories...6
3.Commit the changes...7
4.Remote repositories...7
5.Branches...8
6.Bibliography...9

UT05. VERSION CONTROL SYSTEM

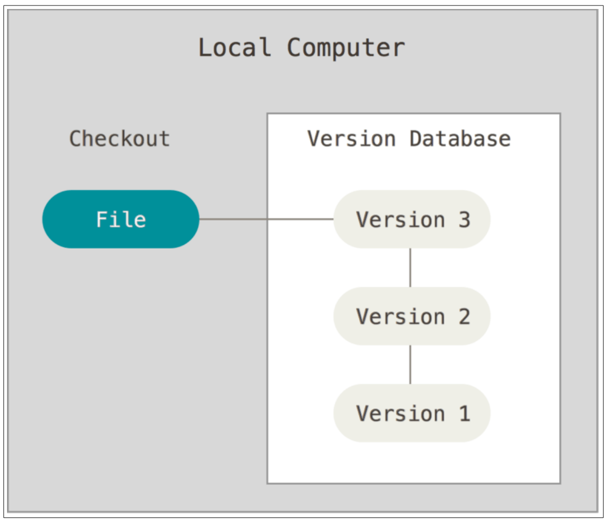
1. INTRODUCTION

"Git Pro"(Chapter 1: About Version Control) by Scott Chacon, Ben Straub licensed under CC BY-NC-SA 3.0

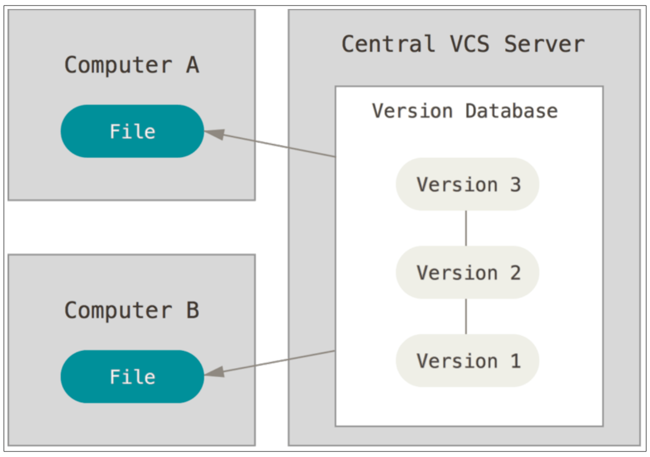
Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

There are different types:

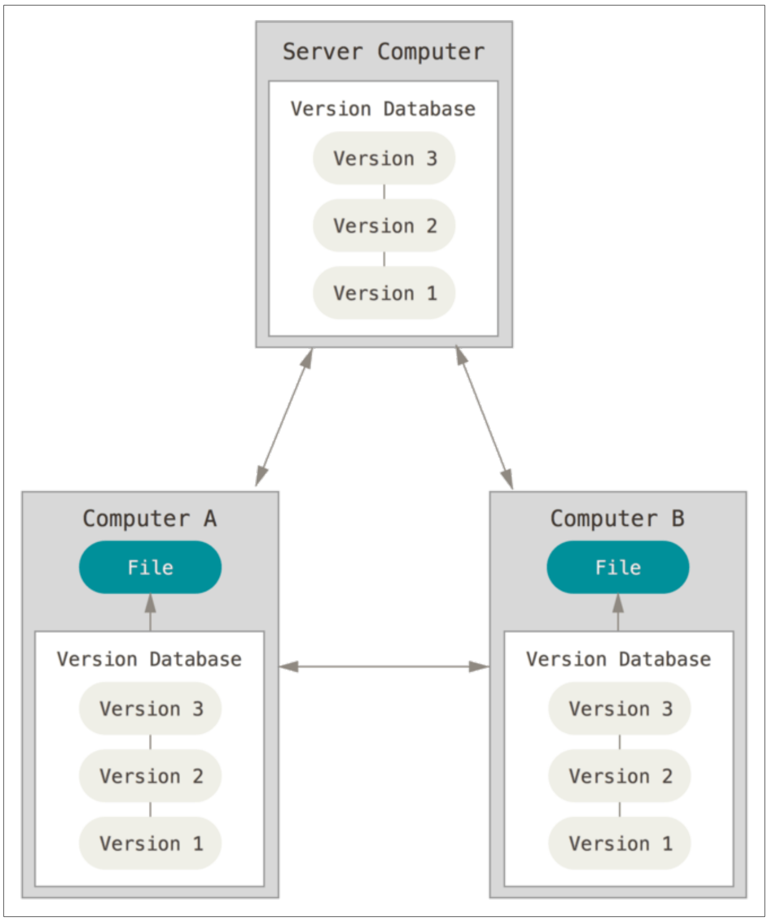
- ***Local Version Control Systems*** where programmers have a simple database that kept all the changes to files under revision control.



- ***Centralized Version Control Systems*** where there is a single server that contains all the versioned files, and a number of clients that check out files from that central place. For many years, this has been the standard for version control.



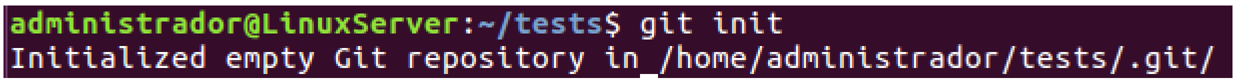
- ***Distributed Version Control Systems*** where clients do not just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.



2. REPOSITORIES

A repository is where we store our files and change them with version control.

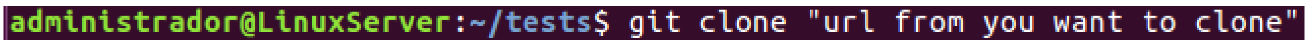
The command to initialize a new repository in a directory that exits is:



```
administrador@LinuxServer:~/tests$ git init
Initialized empty Git repository in /home/administrador/tests/.git/
```

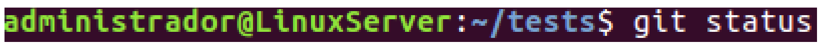
This command create a *.git* subdirectory in the current directory with all the necessary repositories files to work.

And the command to clone an exiting repository from an url is:



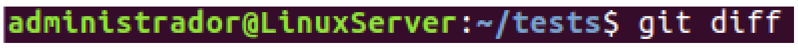
```
administrador@LinuxServer:~/tests$ git clone "url from you want to clone"
```

To see the status of the files of the repository use the command:



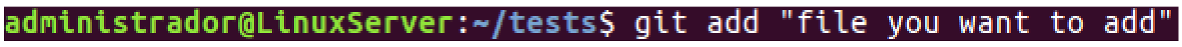
```
administrador@LinuxServer:~/tests$ git status
```

To see also what is changed in the content of the files use the command:



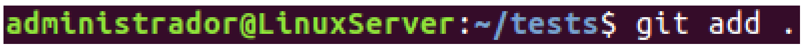
```
administrador@LinuxServer:~/tests$ git diff
```

To add a file in the repository we use the command:



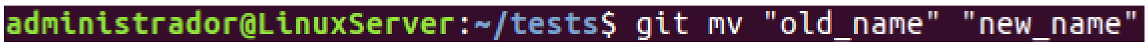
```
administrador@LinuxServer:~/tests$ git add "file you want to add"
```

If you want to add all files of the directory you can use the dot (.):



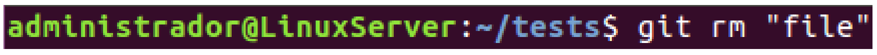
```
administrador@LinuxServer:~/tests$ git add .
```

To rename a file we use:



```
administrador@LinuxServer:~/tests$ git mv "old_name" "new_name"
```

And to delete a file:

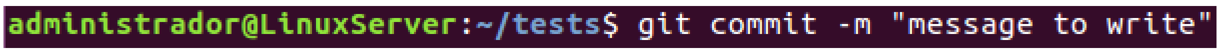


```
administrador@LinuxServer:~/tests$ git rm "file"
```

3. COMMIT THE CHANGES

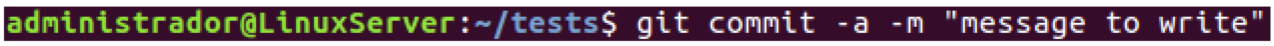
Once added all the files to the staging area we can commit the changes. Only the files added will be committed.

To commit the changes we have to use the command:



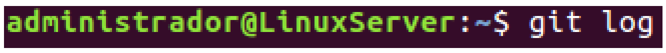
```
administrador@LinuxServer:~/tests$ git commit -m "message to write"
```

With the option *-a* we can include all changed files (as a *git add* command):



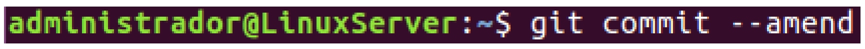
```
administrador@LinuxServer:~/tests$ git commit -a -m "message to write"
```

To see all the changes we did we have to use the command:



```
administrador@LinuxServer:~$ git log
```

And if we want to undo the commit done we have to use:



```
administrador@LinuxServer:~$ git commit --amend
```

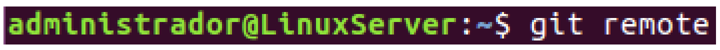
4. REMOTE REPOSITORIES

"Git Pro"(Chapter 2: Git Basics) by Scott Chacon, Ben Straub licensed under CC BY-NC-SA 3.0

“Remote repositories are versions of your project that are hosted on the Internet or network somewhere. You can have several of them, each of which generally is either read-only or read/write for you. Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work.”

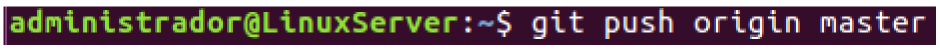
When we clone a repository, the command automatically adds that remote repository under the name *origin*.

To see which remote servers you have configured we have to run:



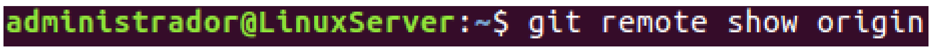
```
administrador@LinuxServer:~$ git remote
```

When we have our project at a point that you want to share, you have to push it upstream. For that we use (in this case we push our *master* branch to our *origin* server):



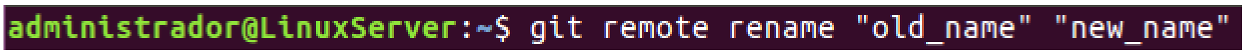
```
administrador@LinuxServer:~$ git push origin master
```

To get information about our remote repository we have to run:



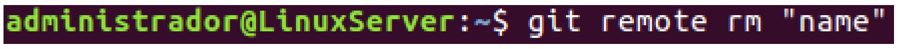
```
administrador@LinuxServer:~$ git remote show origin
```

If we want to rename the remote server:



```
administrador@LinuxServer:~$ git remote rename "old_name" "new_name"
```

And if we want to erase it:



```
administrador@LinuxServer:~$ git remote rm "name"
```

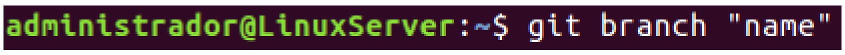
5. BRANCHES

"Git Pro"(Chapter 3: Git Branching) by Scott Chacon, Ben Straub licensed under CC BY-NC-SA 3.0

Branching means you diverge from the main line of development and continue to do work without messing with that main line.

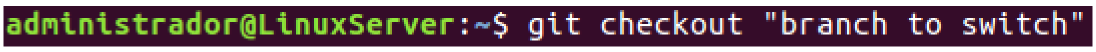
The default branch name in Git is *master*. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, it moves forward automatically.

If we want to create a new branch we have to run:



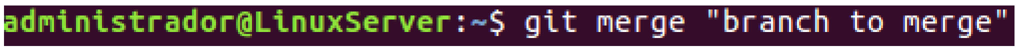
```
administrador@LinuxServer:~$ git branch "name"
```

We can switch to another existing branch running:



```
administrador@LinuxServer:~$ git checkout "branch to switch"
```

We can work with our branches and when we are ready we can merge the branches back into our *master* branch to deploy to production running:



```
administrador@LinuxServer:~$ git merge "branch to merge"
```

6. BIBLIOGRAPHY

[1] *Scott Chacon, Ben Straub. "Pro Git". Appress.*

[2] *http://docs.gitlab.com*