

Version Control Systems

ALGORITHMICS

Vicente García Díaz
garciavicente@uniovi.es

Basic concepts

Git

Github



VERSION CONTROL
SYSTEMS

Basic concepts



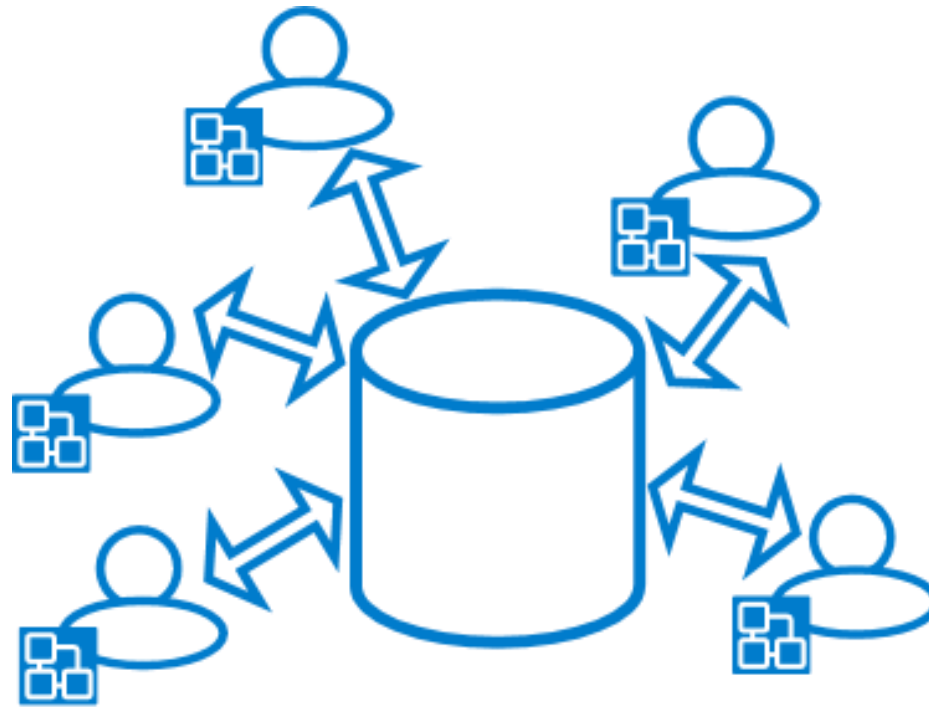
What is a VCS?

- Multiple revisions of the same unit of information



What is the idea?

- People collaborating and sharing files

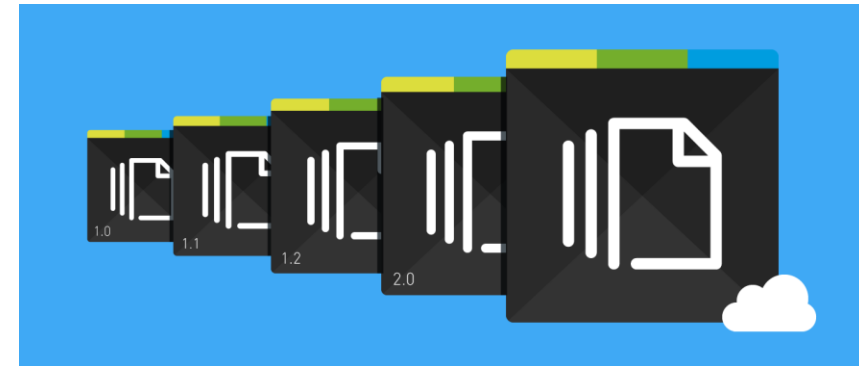


Why we need a VCS?

➤ Enable collaboration



➤ Store versions of a document



Why we need a VCS? (II)

➤ Maintain historical information

Author: [Janusz Białobrzeski <jbialobr@o2.pl>](#)
Date: 23 hours ago (13-Apr-18 3:57:05 PM)
Committer: [GitHub <noreply@github.com>](#)
Commit hash: 1cfd6820ad2306baf67330a6ae0cf30be99c7d5b
Children: [f988cbc1be](#)
Parent(s): [08a69150e3](#) [2d740b9fee](#)



Merge pull request #4823 from jbialobr/jb/fix/4818

Cancel background tasks on disposing.

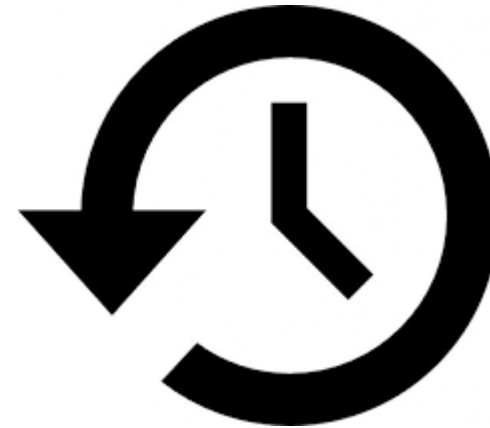
Related links: [Issue 4823](#), [View on GitHub](#)

Contained in branches: [upstream/master](#), [upstream/HEAD](#),
[addGitDescribeToCommitHeader](#)

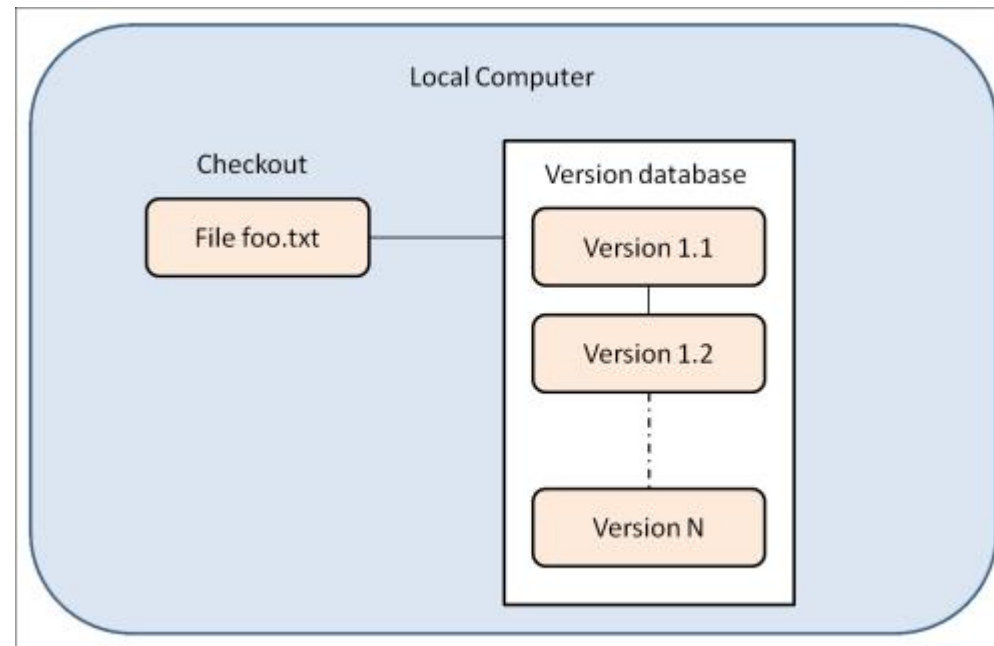
Contained in no tag

Derives from tag: [v2.51](#) + 211 commits

➤ Recover from accidental edits



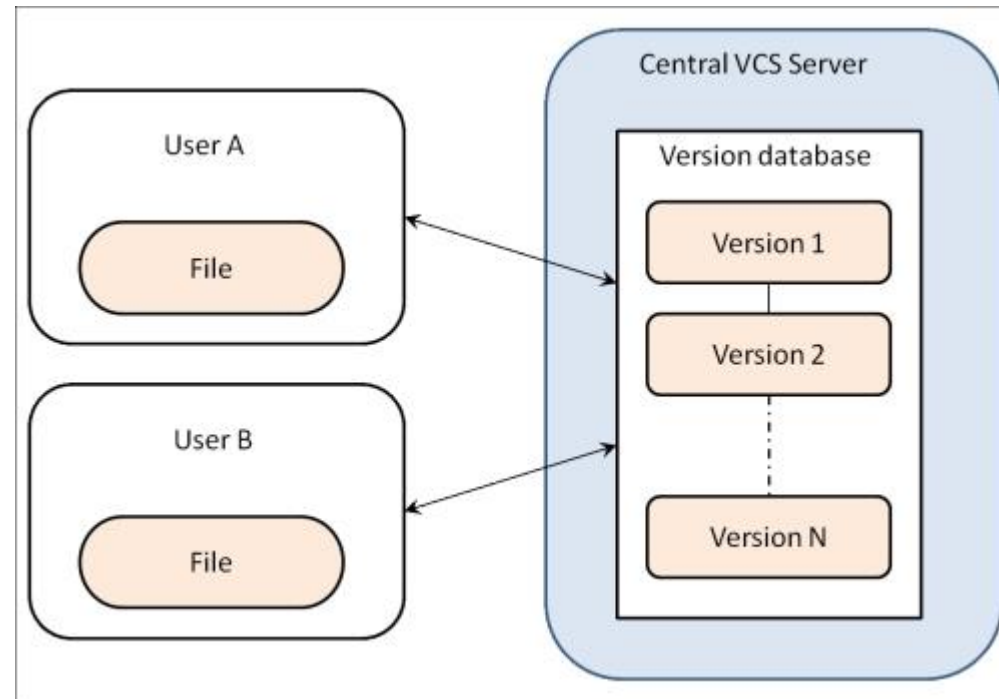
Local version control system



RCS

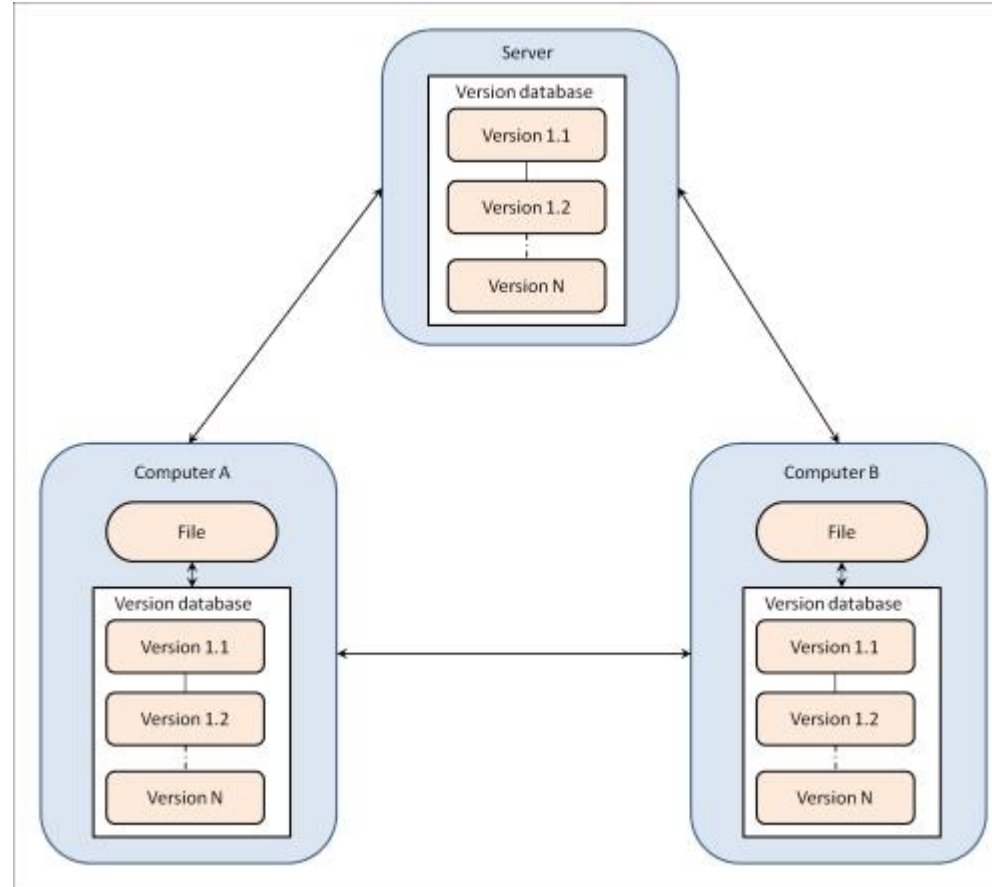
https://subscription.packtpub.com/book/application_development/9781849517522/1/ch011v11sec12/types-of-version-control-systems

Centralized version control system



https://subscription.packtpub.com/book/application_development/9781849517522/1/ch011v11sec12/types-of-version-control-systems

Distributed version control system



https://subscription.packtpub.com/book/application_development/9781849517522/1/ch011v11sec12/types-of-version-control-systems

VERSION CONTROL SYSTEMS

Git



Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases for all major platforms.



Community

Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.43.0
[Release Notes \(2023-11-20\)](#)
[Download for Windows](#)

Get help

- git help
- A 'must' command

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status


grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```


To set your Git information

- `git config --global user.name [GITHUB_USER_NAME]`
- Save the name of the user for Git
- `git config --global user.email [GITHUB_USER_EMAIL]`
- Save the email for Git

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git config --global user.name "Vicente García Díaz"

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git config --global user.name
Vicente García Díaz

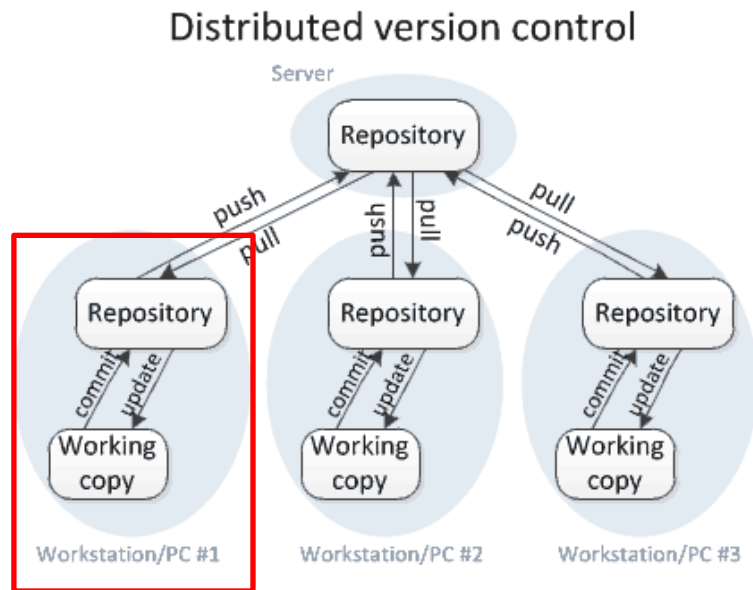
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git config --global user.email vicegd@gmail.com

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git config --global user.email
vicegd@gmail.com
```

Create a new repository

- `git init`
- It is the first step to work with Git

```
Vicen@VGD-VM-DEV MINGW64 ~/Project  
$ git init  
Initialized empty Git repository in C:/Users/Vicen/Project/.git/
```



Disco local (C:) > Usuarios > Vicen > Project > .git

Nombre

- hooks
- info
- objects
- refs
- config
- description
- HEAD

Check the status of the repo

- `git status`
- It is a key command to see what is happening
- There are no **commits**
- The main **branch** is called **master**

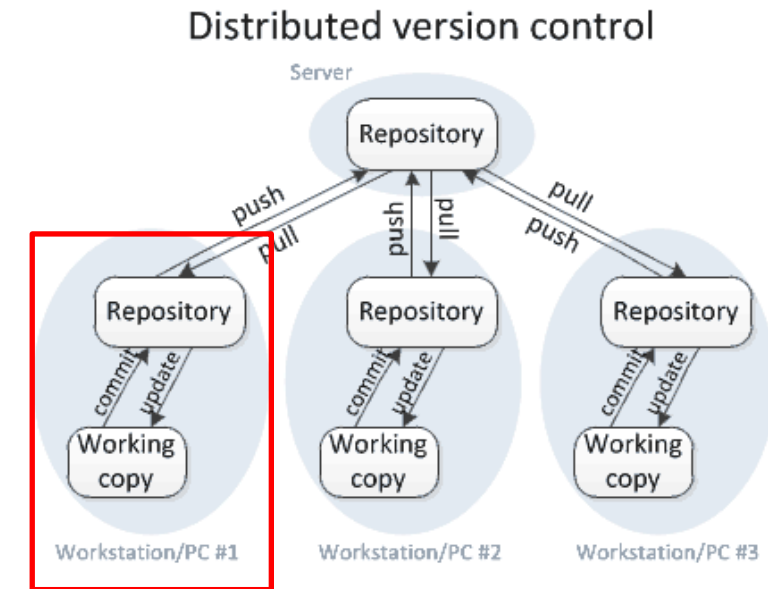
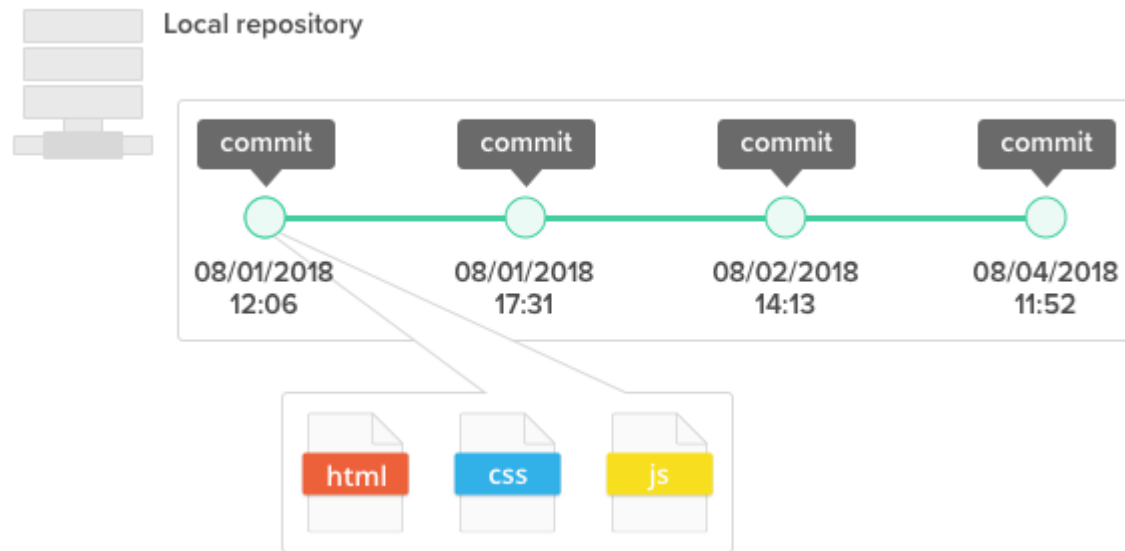
```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git status
On branch master

No commits yet

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

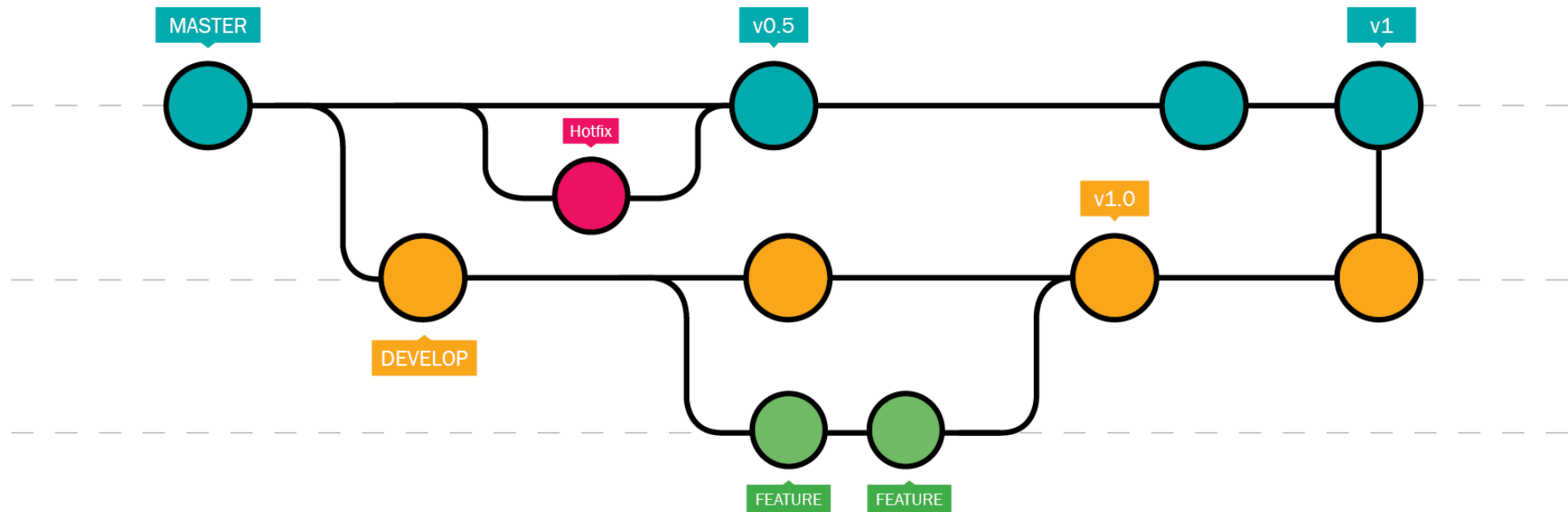
Commits

- Commits are used to save changes to the **local repo**
 - Core **building block** units of a Git project timeline
 - **Snapshots** or milestones along the timeline
 - Capture the **state of a project** at that point in time



Branches

- Independent paths that can come together again



Add a file to the repo

- `git add [FILE] | .`
- It is not enough to copy/add files to the folder
- The file is still **untracked**

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ echo "hello everybody" > file.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ ls
file.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git status
On branch master

No commits yet

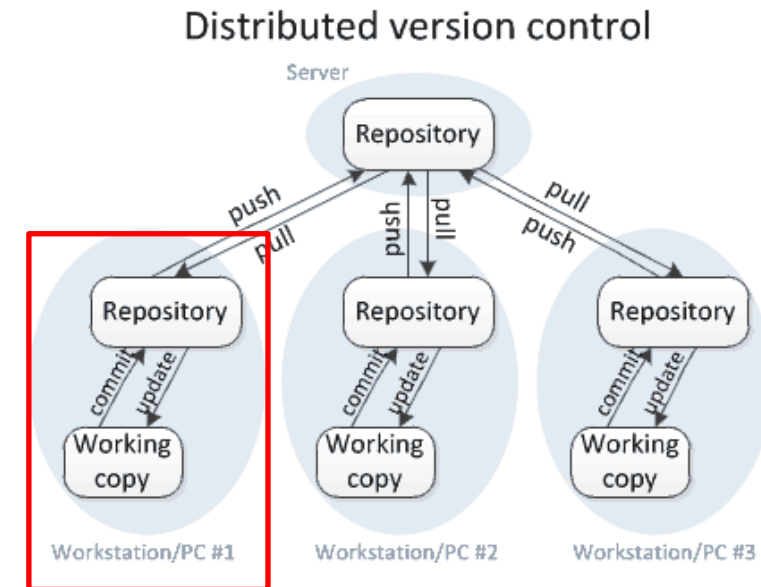
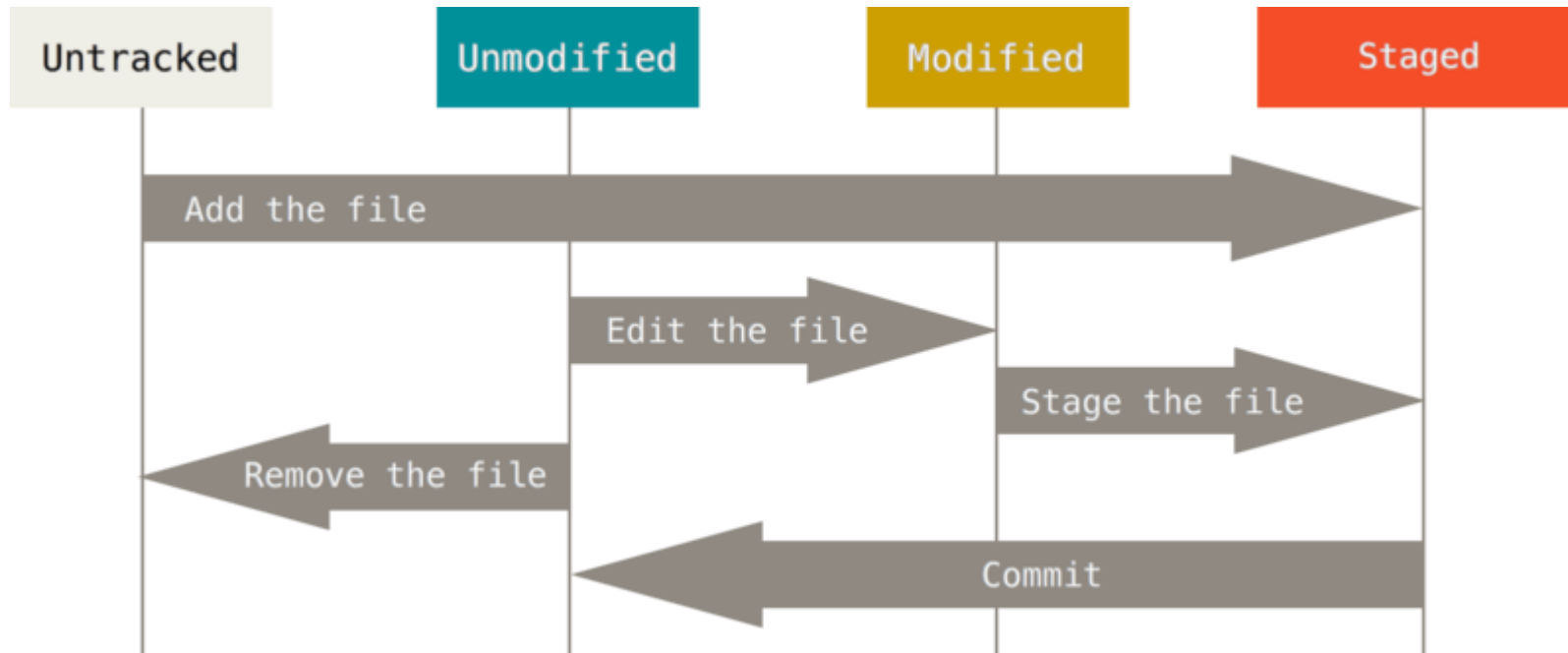
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file.txt

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

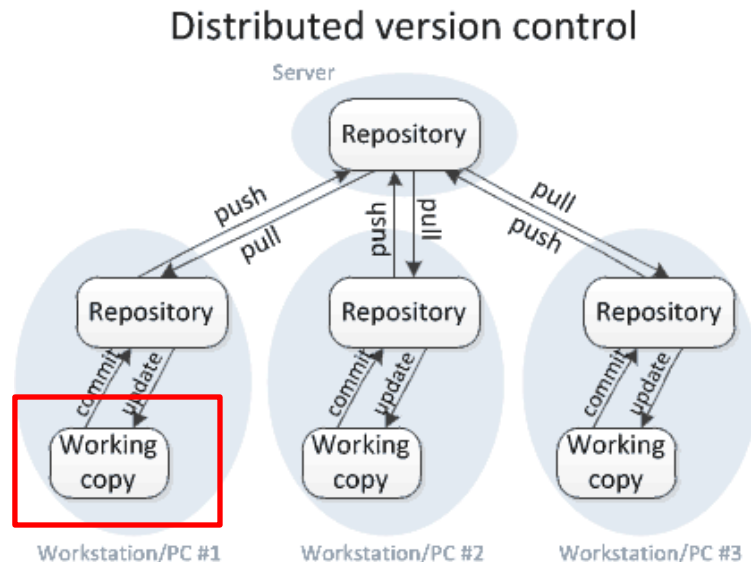

Lifecycle of the status of the files

- Tracked files are the files being managed in the repo



Correctly...add a file to the staged area

- `git add [FILE] | .`
- Don't confuse working directory with local repo



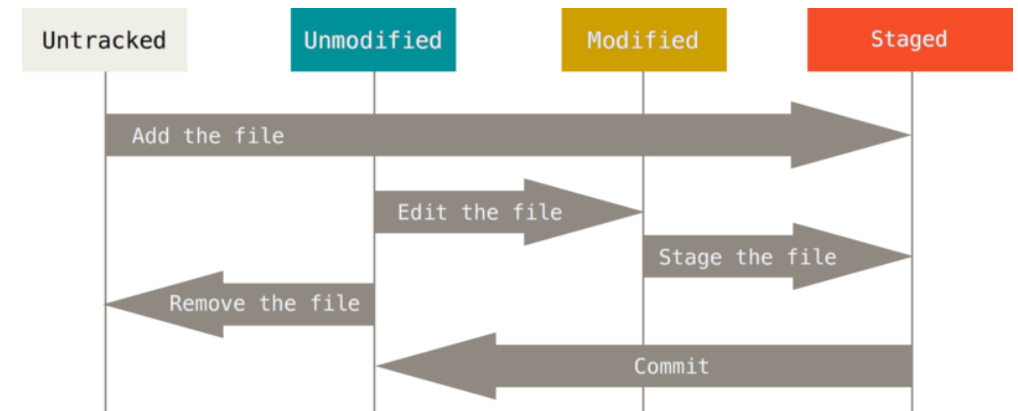
```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git add file.txt
warning: LF will be replaced by CRLF in file.txt.
The file will have its original line endings in your working directory.

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git status
On branch master

No commits yet

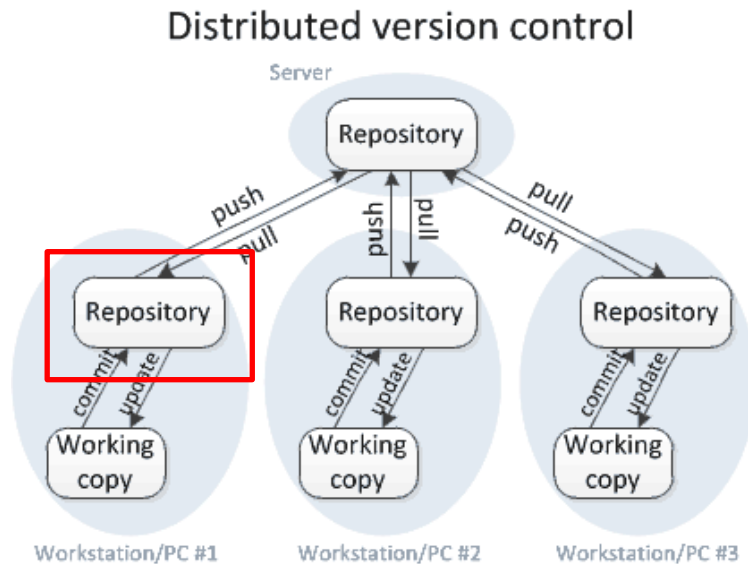
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   file.txt
```



Create a commit

- `git commit -m [MESSAGE]`
- We already have a snapshot of the project



```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git commit -m "This is my first commit"
[master (root-commit) 9222e19] This is my first commit
1 file changed, 1 insertion(+)
create mode 100644 file.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

How are things going?

Project - master - gmaster [GC: 27MB Committed: 111MB]

Repositories Project master Push Pull Fetch

Commit Branch Explorer Commit History Remotes

Search: last 3 months current branch

Commit history

Filter

master This is my first commit

9222e19 13/10/2020 07:19:44 by Vicente García Díaz

Properties

Sha: 9222e19 | Date: 13/10/2020 07:19:44

Author: Vicente García Díaz<vicegd@gmail.com>

This is my first commit

Changed files

First commit: 1 item(s)

Added: 1 item(s)

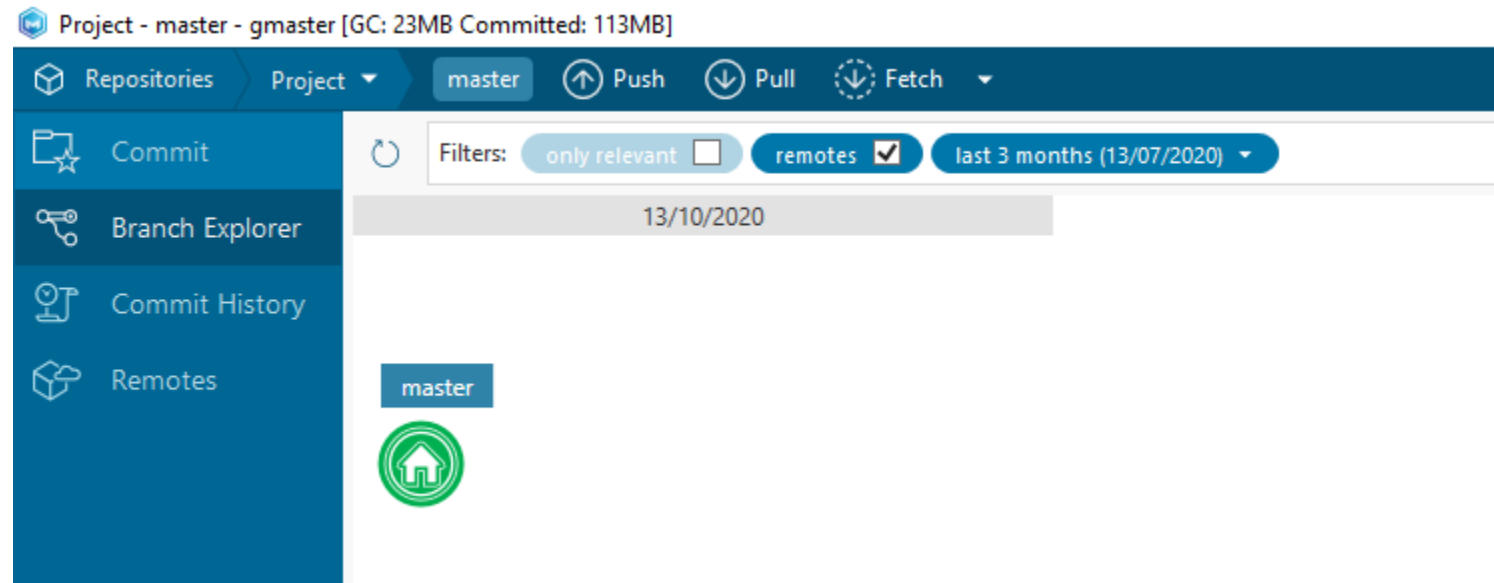
file.txt 1 file.txt

Differences

Content of file.txt

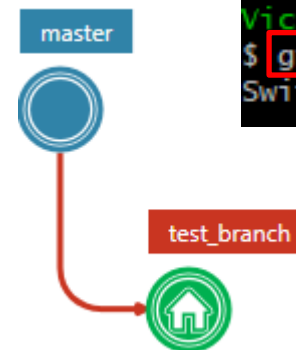
```
1 hello everybody
2
```

How are things going? (II)



Create a branch

- To see the current branch
 - `git branch`
- To create a branch
 - `git checkout -b [NEW_BRANCH]`
- To switch branch
 - `git checkout [BRANCH]`



```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git branch
* master

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git checkout -b test_branch
Switched to a new branch 'test_branch'

Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ git branch
  master
* test_branch

Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ git checkout master
Switched to branch 'master'
```


Let's add a file to the new branch

- The **master** branch is not affected at all
- When we work on a branch, the other branches did not notice it until we **merge**

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git checkout test_branch
Switched to branch 'test_branch'

Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ echo "My file in the new branch" > file2.txt

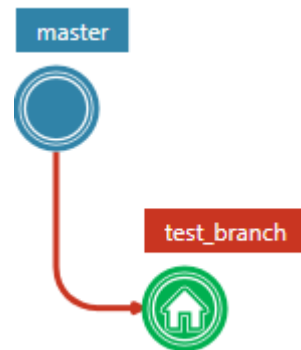
Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ git add file2.txt
warning: LF will be replaced by CRLF in file2.txt.
The file will have its original line endings in your working directory.

Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ git commit -m "Adding my second feature"
[test_branch da9fb07] Adding my second feature
1 file changed, 1 insertion(+)
create mode 100644 file2.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ ls
file.txt file2.txt
```

What is the content of the master branch?

- The **master** branch did not change at all
- The **test_branch** can evolve by itself independently

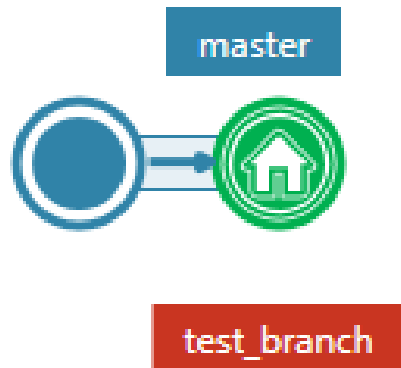


```
Vicen@VGD-VM-DEV MINGW64 ~/Project (test_branch)
$ git checkout master
Switched to branch 'master'

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ ls
file.txt
```

Merge the two branches

- `git merge [BRANCH]`
- We mix the contents of the two branches



```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git merge test_branch
Updating 9222e19..da9fb07
Fast-forward
 file2.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ ls
file.txt  file2.txt
```

How are things going?

Project - master - gmaster [GC: 28MB Committed: 119MB]

Repositories Project master Push Pull Fetch

Commit Branch Explorer Commit History Remotes

Search: last 3 months all branches

Commit history

- test_branch, master Adding my second feature
da9fb07 13/10/2020 07:45:37 by Vicente García Díaz
- This is my first commit
9222e19 13/10/2020 07:19:44 by Vicente García Díaz

Properties

Sha: da9fb07 | Date: 13/10/2020 07:45:37
Author: Vicente García Díaz <vicegd@gmail.com>
Adding my second feature

Changed files

- Parent sha 9222e19: 1 item(s) A 1
- A Added: 1 item(s)
file2.txt 1 file2.txt

Differences

Content of file2.txt

```
1 My file in the new branch
2
```

See what happened in the repo

- `git log`
- It shows main information of all the commits in the repo
- It shows the last commit of the branches

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git log
commit da9fb076172406b8d1ddb804b76d545d1983e9da (HEAD -> master, test_branch)
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:45:37 2020 +0200

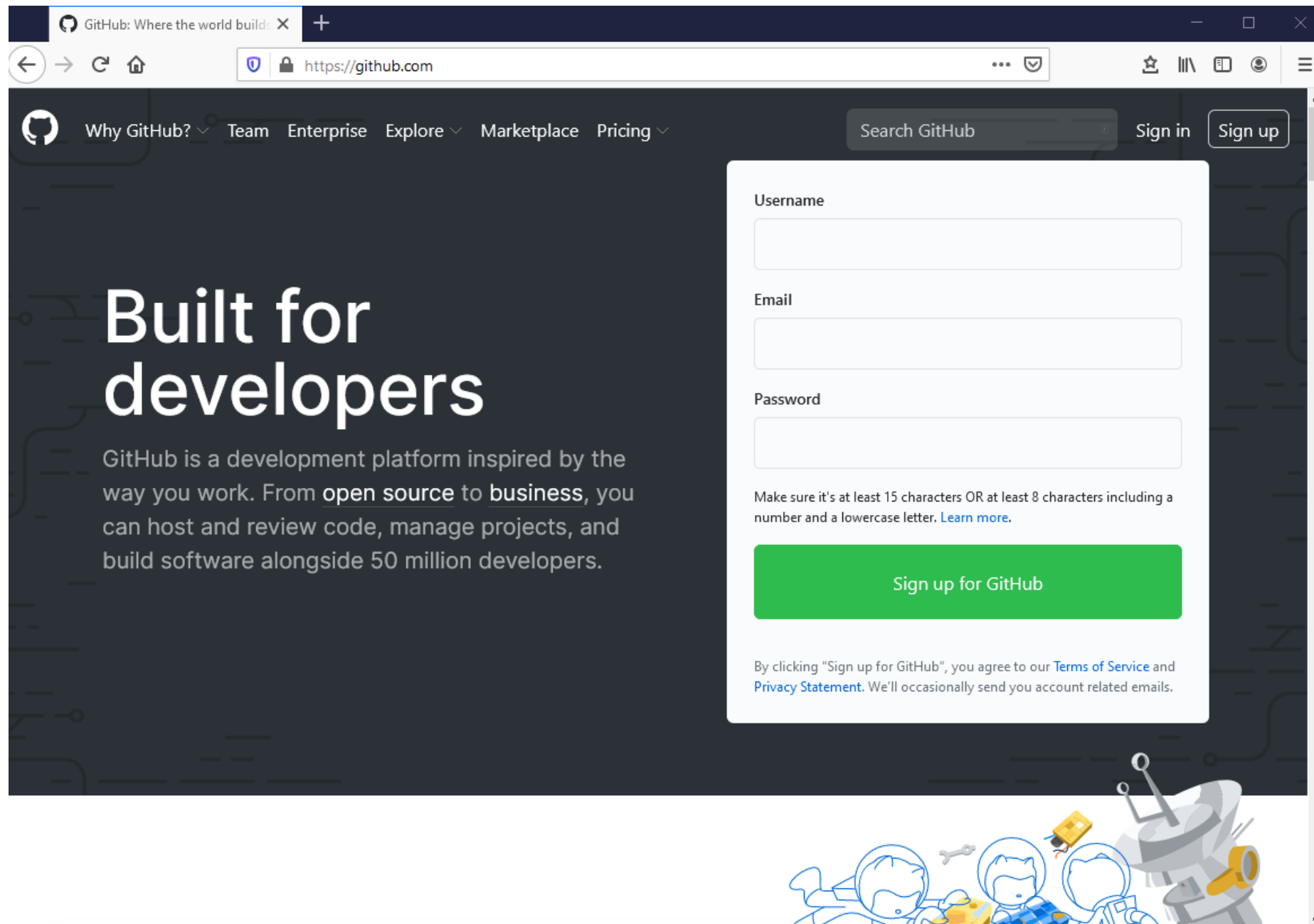
    Adding my second feature

commit 9222e1971989e850537c8a584d36a066647acbcd
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:19:44 2020 +0200

    This is my first commit
```

VERSION CONTROL
SYSTEMS

Github



The image is a screenshot of the GitHub website's homepage. At the top, there is a dark navigation bar with the GitHub logo on the left and links for 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. On the right side of the navigation bar, there is a search bar labeled 'Search GitHub', and buttons for 'Sign in' and 'Sign up'. Below the navigation bar, the main content area has a dark background with a subtle pattern of code and icons. On the left, the headline 'Built for developers' is written in large white text. Below it, a paragraph states: 'GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.' On the right side, there is a white sign-up form with three input fields labeled 'Username', 'Email', and 'Password'. Below the 'Password' field, there is a note: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)'. Below the form is a large green button labeled 'Sign up for GitHub'. At the bottom of the form, there is a disclaimer: 'By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.' At the bottom right of the page, there is a cartoon illustration of three people in space suits working on a satellite or space station.

GitHub: Where the world builds

https://github.com

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.



Find a Campus Advisor

Advisors support their fellow faculty members with all things GitHub. Find out if your school has a Campus Advisor who can help.



Create a new repository



Owner * Repository name *

vicegd / my_testing_project ✓

Great repository names are short and memorable. Need inspiration? How about [cautious-guide](#)?

Description (optional)

This is just a test project

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Java ▼

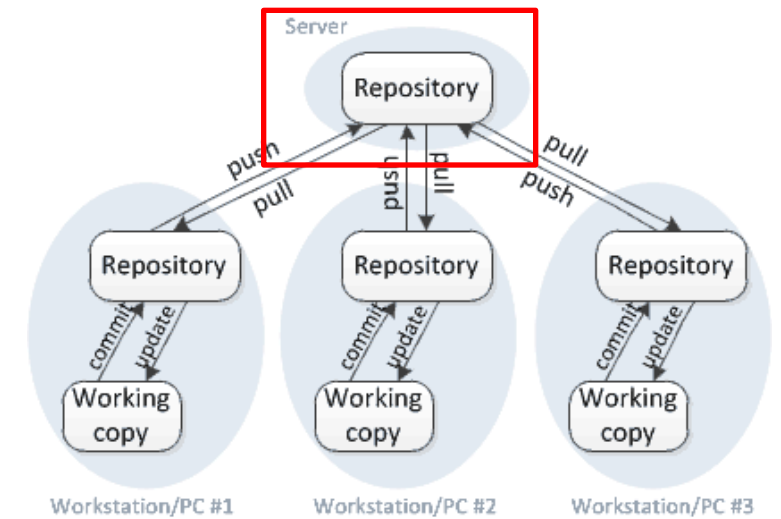
☒ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼


This will set **main** as the default branch. Change the default name in your [settings](#).

Create repository

Distributed version control



New repository created

 [vicegd / my_testing_project](#)

Unwatch 1

Star 0

Fork 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights


Settings




main 1 branch 0 tags

Go to file

Add file

Code

 **vicegd** Initial commit 13c5e89 now 1 commits

 .gitignore	Initial commit	now
 LICENSE	Initial commit	now
 README.md	Initial commit	now

README.md

my_testing_project

This is just a test project

About

This is just a test project

Readme

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Set your Github credentials

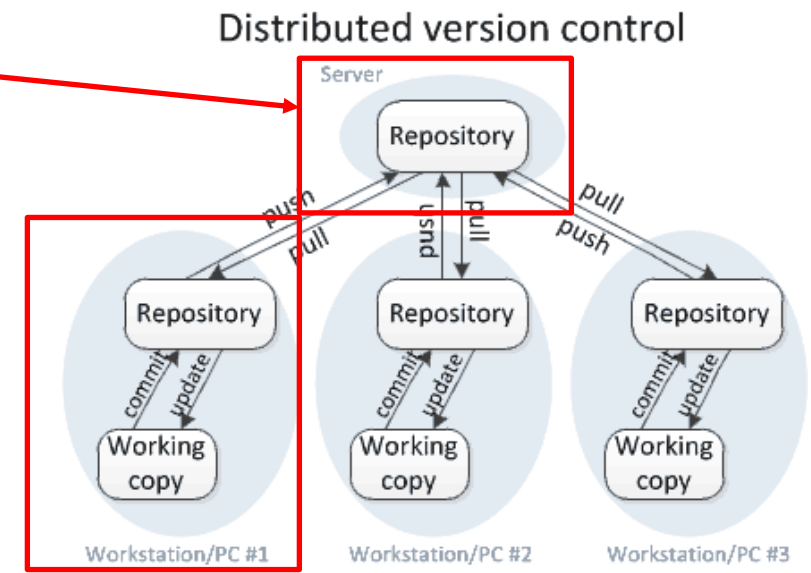
➤ `git config --global credential.helper store`

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)  
$ git config --global credential.helper store
```

➤ To save the remote credentials when used

➤ It is needed when:

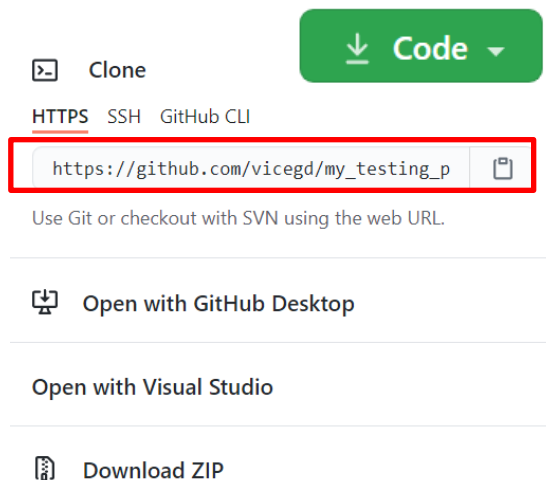
- You work with a private repo
- You try to send things to a public repo



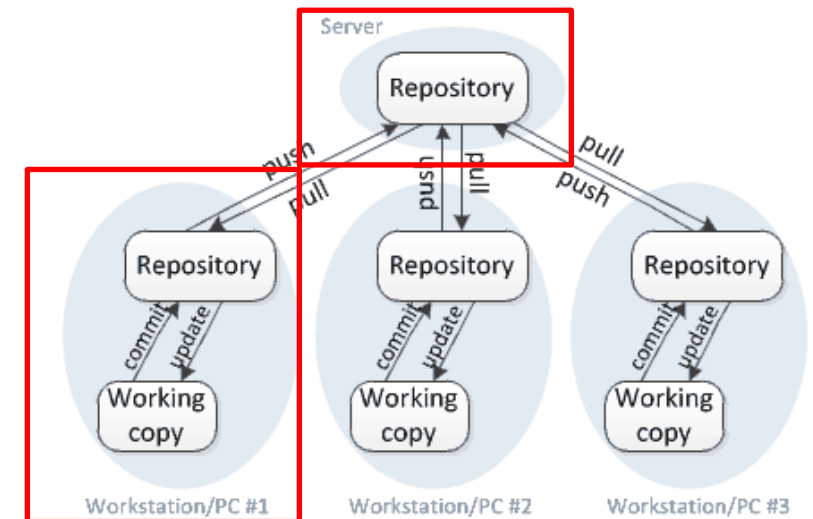
Link repositories

- `git remote add [NAME] [URI]`
- The idea is to have the complete path

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)  
$ git remote add origin https://github.com/vicegd/my_testing_project.git
```



Distributed version control

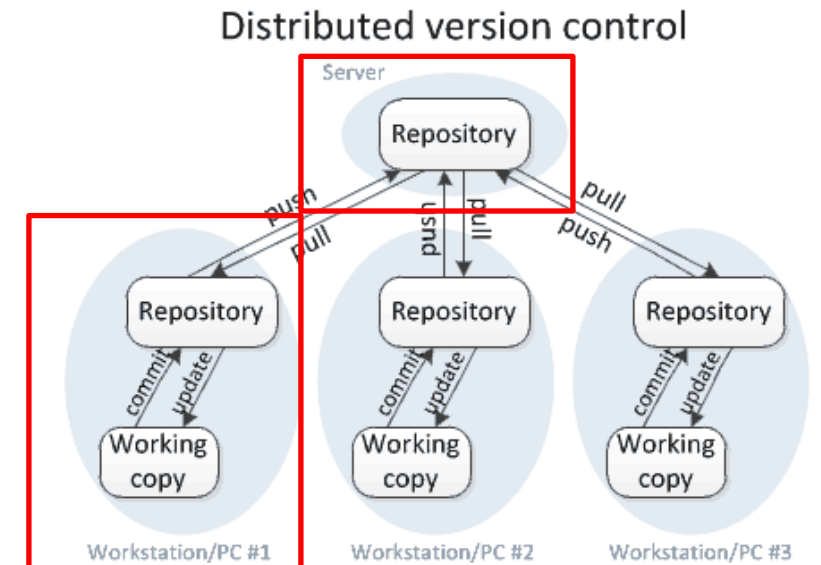


Get contents from the remote

- `git pull [REMOTE_NAME] [REMOTE_BRANCH] --rebase`
- **Pull** gets contents from the remote to the local repository

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git pull origin main --rebase
From https://github.com/vicgd/my_testing_project
 * branch          main      -> FETCH_HEAD
First, rewinding head to replay your work on top of it...
Applying: This is my first commit
Applying: Adding my second feature

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ ls
file.txt  file2.txt  LICENSE  README.md
```



See what happened in the repo

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git log
commit 88dc97a1357a690d72f3b9e8fbd9fe1de76fa96e (HEAD -> master)
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:45:37 2020 +0200

    Adding my second feature

commit 66aa404c0066c313b51c6ccc913271c1f02cf826
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:19:44 2020 +0200

    This is my first commit

commit 13c5e89dfb4c1d3d321223e08754b4472841cadd (origin/main)
Author: Vicente García Díaz <vicegd@users.noreply.github.com>
Date: Wed Oct 14 12:56:05 2020 +0200

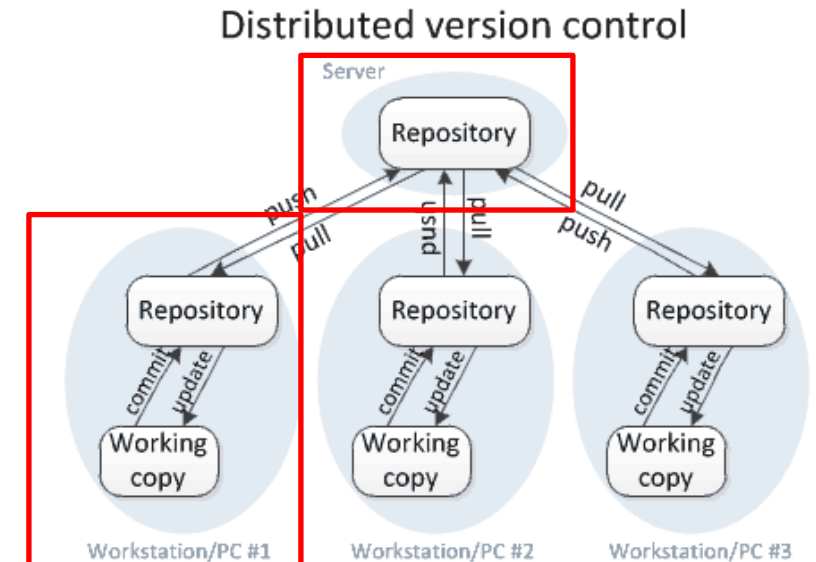
    Initial commit
```


Send contents to the remote

- `git push [REMOTE_NAME]`
`[LOCAL_BRANCH] : [REMOTE_BRANCH]`
- **Push** sends contents from the local to the remote repository

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git push origin master:main
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 648 bytes | 648.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/vicegd/my_testing_project.git
13c5e89..88dc97a master -> main
```

main 1 branch 0 tags	
vicegd Adding my second feature	
.gitignore	Initial commit
LICENSE	Initial commit
README.md	Initial commit
file.txt	This is my first commit
file2.txt	Adding my second feature



Compare what happened in the repo

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git log
commit 88dc97a1357a690d72f3b9e8fbd9fe1de76fa96e (HEAD -> master)
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:45:37 2020 +0200

    Adding my second feature

commit 66aa404c0066c313b51c6ccc913271c1f02cf826
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:19:44 2020 +0200

    This is my first commit

commit 13c5e89dfb4c1d3d321223e08754b4472841cadd (origin/main)
Author: Vicente García Díaz <vicegd@users.noreply.github.com>
Date: Wed Oct 14 12:56:05 2020 +0200

    Initial commit
```



```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git log
commit 88dc97a1357a690d72f3b9e8fbd9fe1de76fa96e (HEAD -> master, origin/main)
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:45:37 2020 +0200

    Adding my second feature

commit 66aa404c0066c313b51c6ccc913271c1f02cf826
Author: Vicente García Díaz <vicegd@gmail.com>
Date: Tue Oct 13 07:19:44 2020 +0200

    This is my first commit

commit 13c5e89dfb4c1d3d321223e08754b4472841cadd
Author: Vicente García Díaz <vicegd@users.noreply.github.com>
Date: Wed Oct 14 12:56:05 2020 +0200

    Initial commit
```

Let's create a conflict

➤ On Github

my_testing_project / log.txt Cancel

<> Edit new file

👁 Preview

1 This file is going to give us problems

Commit new file

Adding log.txt

Add an optional extended description...

☒ Commit directly to the `main` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file Cancel

On the local repository

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ echo "Hello" > log.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git add .
warning: LF will be replaced by CRLF in log.txt.
The file will have its original line endings in your working directory.

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git commit -m "I added a log file now to the local repository"
[master 088c9e6] I added a log file now to the local repository
1 file changed, 1 insertion(+)
create mode 100644 log.txt
```

Let's create a conflict (II)

➤ Nothing will work

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git pull origin main --rebase
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/vicegd/my_testing_project
* branch      main      -> FETCH_HEAD
   88dc97a..dca366e main    -> origin/main
First, rewinding head to replay your work on top of it...
Applying: I added a log file now to the local repository
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
Auto-merging log.txt
CONFLICT (add/add): Merge conflict in log.txt
error: Failed to merge in the changes.
Patch failed at 0001 I added a log file now to the local repository
Use 'git am --show-current-patch' to see the failed patch

Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
```

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master|REBASE 1/1)
$ git push origin master:main
To https://github.com/vicegd/my_testing_project.git
! [rejected]        master -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/vicegd/my_testing_project.git'
hint: Updates were rejected because a pushed branch tip is behind its remote
hint: counterpart. Check out this branch and integrate the remote changes
hint: (e.g. 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Let's fix the conflict

➤ We fix the file

log.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

<<<<<< HEAD

This file is going to give us problems

=====

I added a log file now to the local repository

>>>>>> I added a log file now to the local repository



*log.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Ok, let's put this in the file!

➤ We indicate that the conflict is solved

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master|REBASE 1/1)
$ git add log.txt
Vicen@VGD-VM-DEV MINGW64 ~/Project (master|REBASE 1/1)
$ git rebase --continue
Applying: I added a log file now to the local repository
```

➤ We PUSH the changes to the server

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git push origin master:main
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 327.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vicegd/my_testing_project.git
24bbb6c..acde98b master -> main
```

Remove a file from the repos

- `git rm [FILE]`
- **Rm** is used to stop a file from being processed by Git
- After executing the commands, both repositories will be synchronized

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ ls
file.log file.txt file2.txt LICENSE log.txt README.md

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git rm log.txt
rm 'log.txt'

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git commit -m "Removing logger"
[master 0ee6e31] Removing logger
1 file changed, 1 deletion(-)
delete mode 100644 log.txt

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git push origin master:main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 238 bytes | 18.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vicegd/my_testing_project.git
82641a6..0ee6e31 master -> main
```

Ignore not important files

- The file `.gitignore` contains the files that should not be tracked by Git
- We should ignore:
 - Log files
 - Files with API keys/secrets, credentials
 - Any sensitive information
 - Useless system files
 - Generated/compiled files
 - Dependencies which can be downloaded from a package manager

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git add file.log
The following paths are ignored by one of your .gitignore files:
file.log
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"

Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

`*.gitignore: Bloc de notas`

	Archivo	Edición	Formato	Ver	Ayuda
# Compiled class file					
*.class					
# Log file					
*.log					
# BlueJ files					
*.ctxt					
# Mobile Tools for Java (J2ME)					
.mtj.tmp/					
# Package Files #					
*.jar					
*.war					
*.nar					
*.ear					
*.zip					
*.tar.gz					
*.rar					

How to use .gitignore

Example
file.log
logs/
*.log
? .log
*.log !file.log
logs/ !logs/file.log
**/logs/*.log
logs/**/*.log

Clone a project

- `git clone [URI]`
- It is the easiest way to start working with a remote repository and be directly in sync with it

```
Vicen@VGD-VM-DEV MINGW64 ~/Project (master)
$ cd ..

Vicen@VGD-VM-DEV MINGW64 ~
$ git clone https://github.com/vicgd/my_testing_project.git
Cloning into 'my_testing_project' ...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 40 (delta 13), reused 27 (delta 8), pack-reused 0
Unpacking objects: 100% (40/40), 5.83 KiB | 7.00 KiB/s, done.

Vicen@VGD-VM-DEV MINGW64 ~
$ cd my_testing_project/

Vicen@VGD-VM-DEV MINGW64 ~/my_testing_project (main)
$ ls
file.txt  file2.txt  LICENSE  README.md
```