



git



GitHub

Git ou GitHub ?

Mohamed BEN SAAD



git

Sommaire



Git

- A quoi sert le versionning ? Dans quels cas l'utiliser ?
- Installation de Git sur Windows
- Les lignes de commande, quels outils pour Windows ? Cmd.exe et GitBash
- Configurer Git en ligne de commande. Indiquer son nom et son adresse
- Initialiser un dépôt Git
- Commandes Git
- Réaliser son premier commit

A quoi sert le versionning ?



- **VCS** = Version Control System
- Gestion des versions et historiques de fichiers
- Gestion des branches
- Gestion des tags
- Gestion des conflits / merges
- Suivre les étapes de modification d'un programme
- Tester un changement complexe et pouvoir revenir facilement en arrière
- Travailler à plusieurs sur un projet
- Inviter des collaborateurs sur un projet

Dans quels cas l'utiliser ?



git

- Garder une trace de ce qui a été fait
- Revenir à une phase précédente
- Faciliter la résolution des erreurs et la correction d'autres erreurs de développement
- Noter les changements dans chaque version
- Aider les membres de l'équipe à distinguer ce qui est terminé et ce qu'il reste à accomplir

Attention : Git est le mieux adapté à une utilisation individuelle

Lexique



git

Git possède un vocabulaire spécifique,

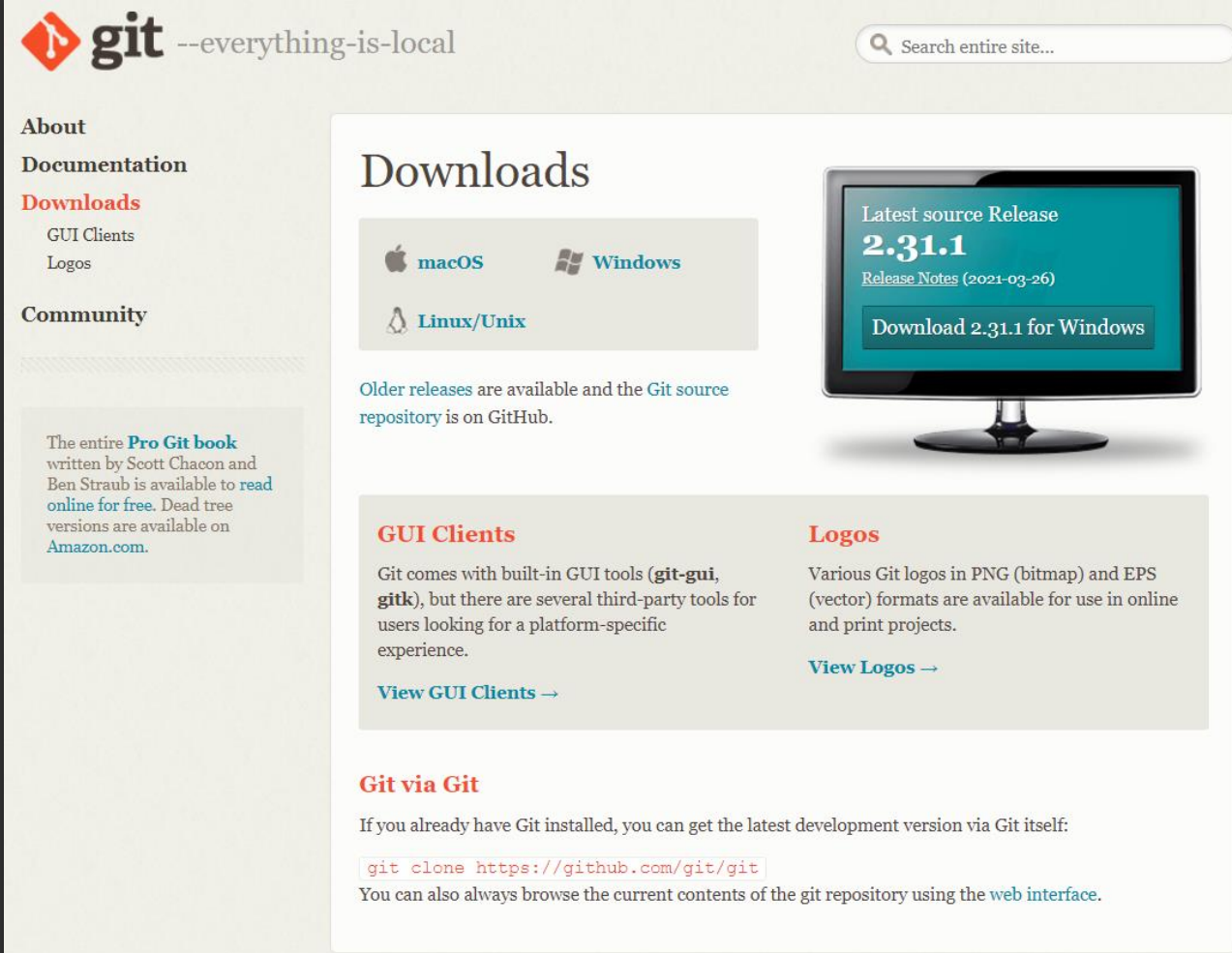
Voici un petit lexique des mots les plus courants :

- **Repository** (*Repot* ou “*dépôt*” en français) : espace de stockage que vous allez créer sur le serveur pour y déposer les fichiers sources
- **Push** (*pousser*) : envoyez des fichiers sur le serveur
- **Pull** (*tirer*) : téléchargez la dernière version des fichiers sur votre ordinateur à partir du serveur
- **Clone** (*cloner*) : copiez pour un projet sur votre ordinateur à partir du serveur.
- **Commit** (*valider*) : valider des modifications avant envoi
- **Branch** (*branche*) : créer de nouvelles branches pour développer une fonctionnalité en parallèle
- **Master** (*maître*) : branche principale du projet
- **Merge** (*fusionner*) : ajouter/fusionner une branche à la principale (master)

Installation de Git sur Windows

Étape 1 : Installer git et ajouter un dépôt

Lien : <https://git-scm.com/>



The screenshot shows the Git website homepage. At the top, the Git logo is followed by the tagline "--everything-is-local". A search bar is located in the top right corner. The left sidebar contains links for "About", "Documentation", "Downloads" (highlighted in red), "GUI Clients", "Logos", and "Community". The main content area features a "Downloads" section with buttons for macOS, Windows, and Linux/Unix. To the right of these buttons is a monitor graphic displaying the latest source release "2.31.1" and a button to "Download 2.31.1 for Windows". Below the download buttons, a text box states that older releases are available and the source repository is on GitHub. Further down, there are sections for "GUI Clients" and "Logos". The "GUI Clients" section mentions built-in tools like git-gui and gitk, and lists third-party tools. The "Logos" section mentions various Git logos in PNG and EPS formats. At the bottom, a section titled "Git via Git" provides instructions on how to clone the Git repository using the command `git clone https://github.com/git/git` and mentions the web interface.

git --everything-is-local

Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

macOS Windows Linux/Unix

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.31.1
[Release Notes \(2021-03-26\)](#)
Download 2.31.1 for Windows

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).

Installation de Git sur Windows

Étape 2 : Installer git

Git 2.31.1 Setup

Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?

☒ **Let Git decide**
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☐ **Override the default branch name for new repositories**
NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

This setting does not affect existing repositories.

<https://gitforwindows.org/>

☒ Only show new options

Git 2.31.1 Setup

Choose the default behavior of `git pull`
What should `git pull` do by default?

☒ **Default (fast-forward or merge)**
This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

☐ **Rebase**
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

☐ **Only ever fast-forward**
Fast-forward to the fetched branch. Fail if that is not possible.

<https://gitforwindows.org/>

☒ Only show new options

Git 2.31.1 Setup

Choose a credential helper
Which credential helper should be configured?

☒ **Git Credential Manager Core**
(NEW!) Use the new, [cross-platform version of the Git Credential Manager](#). See more information about the future of Git Credential Manager [here](#).

☐ **Git Credential Manager**
(DEPRECATED) The [Git Credential Manager for Windows](#) handles credentials e.g. for Azure DevOps and GitHub (requires .NET framework v4.5.1 or later).

☐ **None**
Do not use a credential helper.

<https://gitforwindows.org/>

☒ Only show new options

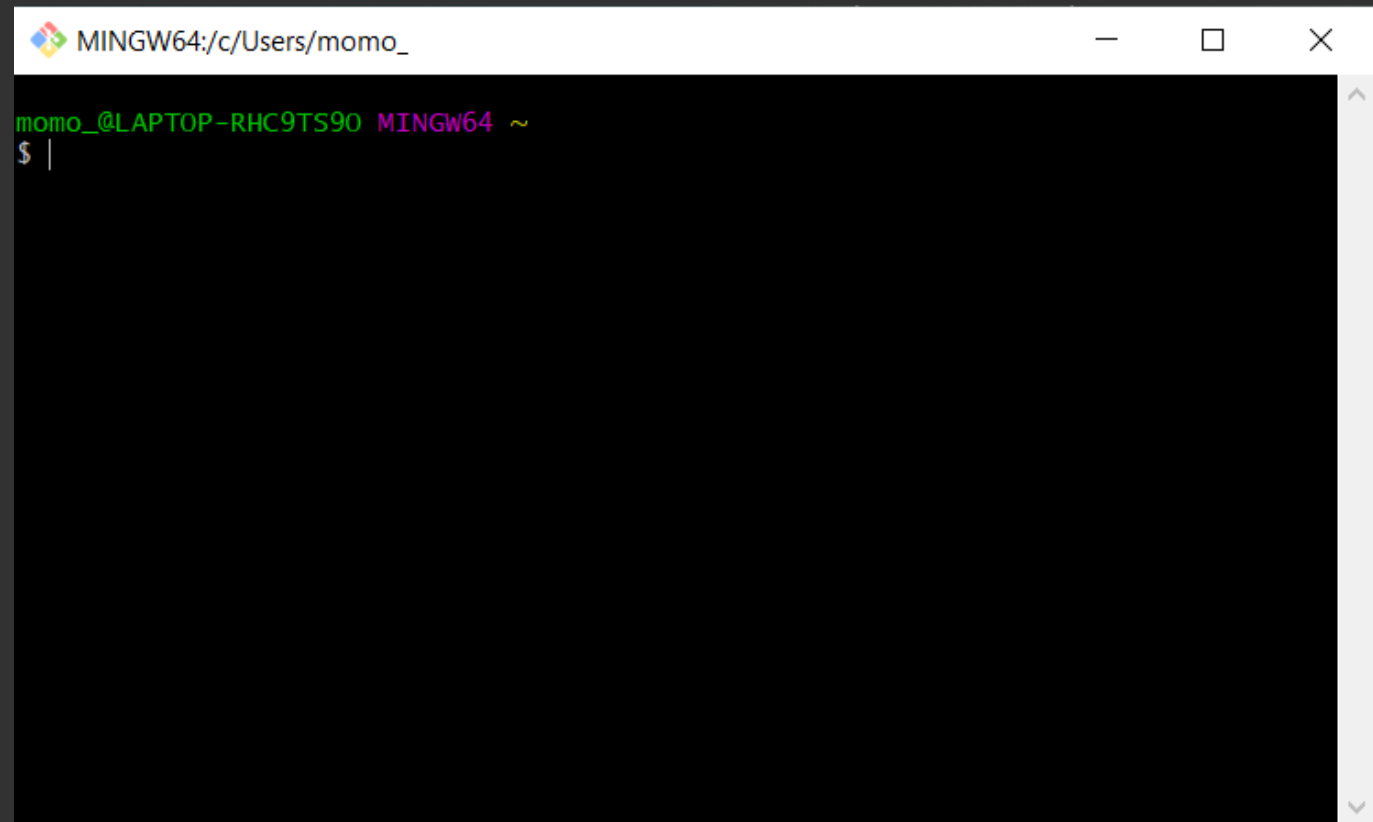
GitBash



Cmd.exe ou Git Bash ?

Peut importe, c'est en fonction de vos préférences

Commandes **Windows** ou **Linux**

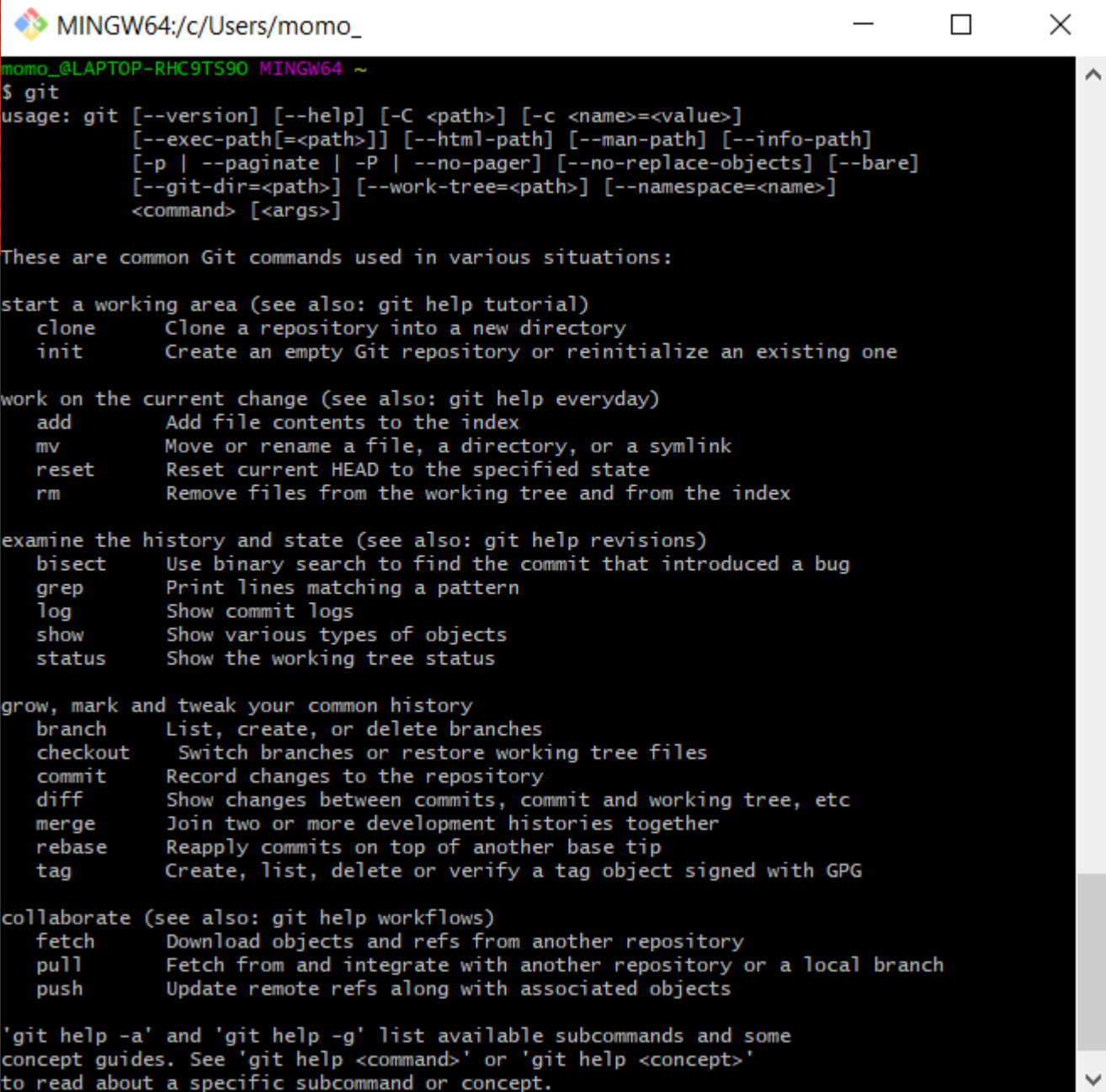


Les lignes de commande

```
# Format strict
git <command> --<option>=<param>
# Format light
git <command> --<option> <param>
```

○ Voici les commandes principales :

```
git config --global user.name 'Name'
git config --global user.email 'Mail'
git help commande
git init
git add <file> <otherfile>
git commit
```



```
MINGW64:/c/Users/momo_
momo_@LAPTOP-RHC9TS90 MINGW64 ~
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path=<path>] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --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.
```

Git & Options



59 possibilités pour la gestion de ses projets

```
MINGW64:/c/Users/momo_
momo_@LAPTOP-RHC9TS90 MINGW64 ~
$ git
Display all 59 possibilities? (y or n)
add          clean          gitk          pull          send-email
am           clone           grep          push          shortlog
apply        commit         gui           range-diff    show
archive     config          help          rebase        show-branch
bisect      describe       init          reflog        stage
blame       diff            instaweb      remote        stash
branch     difftool        lfs           repack        status
bundle     fetch           log           replace       submodule
checkout   flow            merge         request-pull  tag
cherry     format-patch   mergetool     reset         whatchanged
cherry-pick fsck            mv            revert        worktree
citool     gc              notes         rm
```

Créer un dépôt



- Ouvrir GitBash
- Aller jusqu'à la destination avec
`cd <chemin>`
- Créer un dépôt git avec
`git init <dossier de travail>`

Si le dossier de travail indiqué n'existe pas, elle va le créer.

Sans paramètre, la commande crée un dépôt git pour le dossier courant.

`git -help <commande>`

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git
$ git init ExempleGit
Initialized empty Git repository in D:/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit/.git/

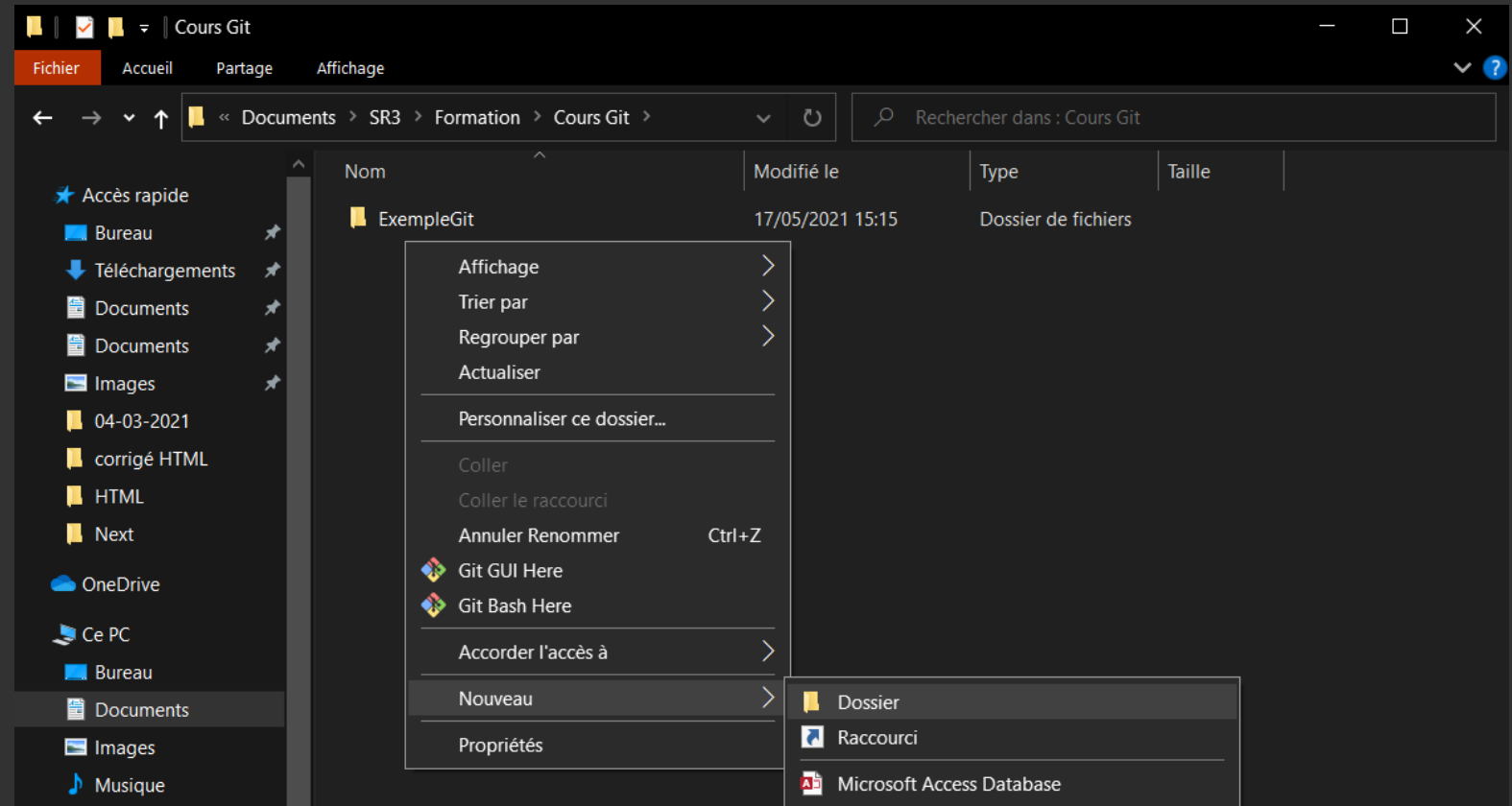
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git
$ git init ExempleGit
Reinitialized existing Git repository in D:/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit/.git/

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git
$ |
```

Alternative



- Créer un Nouveau Dossier

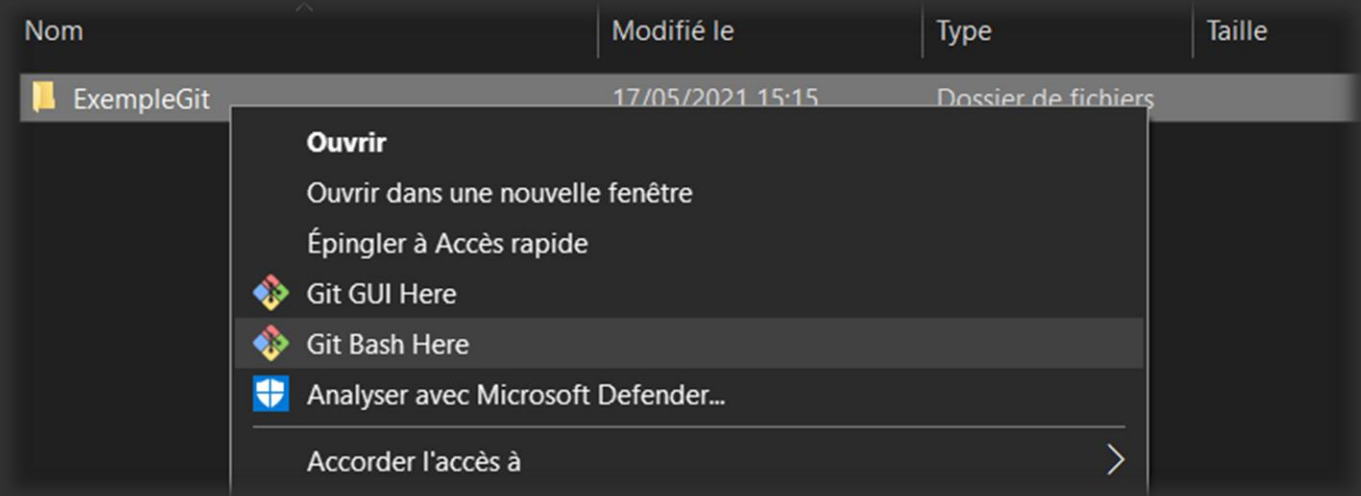


Alternative



○ Ouvrir Git Bash

Tapez : `git init`

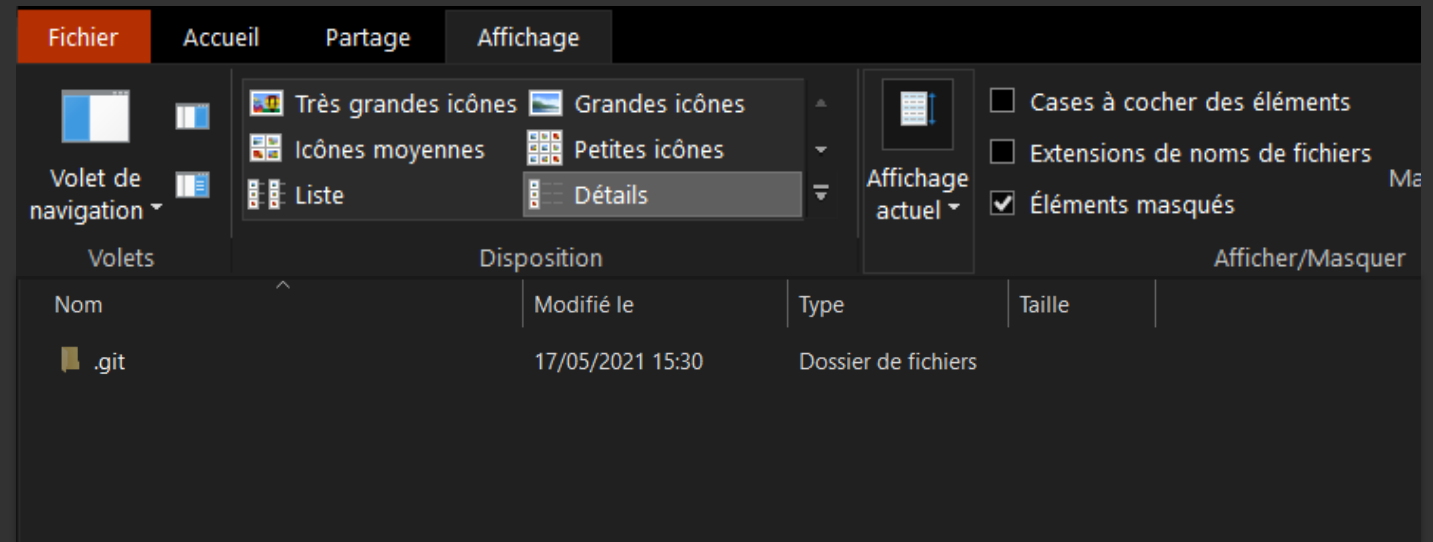


```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit (master)
$ git init
Reinitialized existing Git repository in D:/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit/.git/
```


Fichier Git



- Afficher le fichiers Masqué
- Visualisation du fichier **.git**





Dossier de travail VS dépôt



- Votre dossier de travail contient vos fichiers et dossiers composant votre projet. Git peut modifier ces fichiers pour les mettre à jour ou vous les présenter à différentes versions du projet grâce à votre dépôt.
- Votre dépôt git contient tout l'historique de votre projet. Toutes les versions des fichiers, toutes les modifications, etc. C'est le dossier **.git** dans votre dossier de travail.

Premier Pas



Définir son identité

Votre identité sera associée aux modifications que vous enregistrerez dans vos dépôts

```
$ git config --global user.name "Votre nom"
$ git config --global user.email votre@mail
```

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/Exemp
leGit (master)
$ git config --global user.name "Mohbenz"

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit (master)
$ git config --global user.email moh_benz@outlook.fr
```

Commandes Git



- La commande `git status` affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés.
- `touch <nom>` est une commande Unix pour créer un fichier

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit (master)
$ git status
On branch master

No commits yet

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

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit (master)
$ touch fichier.html

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit (master)
$ git status
On branch master

No commits yet

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

        fichier.html

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

Commandes Git



- La commande git add peut être utilisée pour ajouter des fichiers à l'index :
 - `git add <nom>`
 - ou
 - `git add .` (pour tout le dossier courant)
- La liste du status change et le fichier est en attente de validation

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
$ git add .

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
$ git status
On branch master

No commits yet

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

        new file:   fichier.html
```

Commandes Git



- La commande git add peut être utilisée pour ajouter des fichiers à l'index :
 - `git add <nom>`
 - ou
 - `git add .` (pour tout le dossier courant)
- La liste du status change et le fichier est en attente de validation

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
$ git add .

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours Git/ExempleGit
$ git status
On branch master

No commits yet

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

        new file:   fichier.html
```


Commandes Git



- Git rm peut être utilisé pour supprimer des fichiers de l'index et du répertoire de travail.

```
git rm nomfichier.txt
```

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/Exem...
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
it/ExempleGit (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   fichier.html

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
it/ExempleGit (master)
$ git rm --cached fichier.html
rm 'fichier.html'

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
it/ExempleGit (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    fichier.html

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

        fichier.html
```

Commandes Git



- La commande `git commit` permet de valider les modifications apportées au HEAD.

```
git commit -m "Description du commit"
```

- La liste change de nouveau après la validation.

Notez que tout commit ne se fera pas dans le dépôt distant.

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/Exem...
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
$ git commit -m "Ajout fichier.html"
[master (root-commit) 8b71821] Ajout fichier.html
1 file changed, 12 insertions(+)
create mode 100644 fichier.html

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
$ git status
On branch master
nothing to commit, working tree clean

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3/Formation/Cours G
$
```



Historique des modifications Git

- La manière la plus simple de consulter l'historique des modifications Git est d'utiliser la commande : `git log`. Cette commande affiche la liste des commits réalisés du plus récent au plus ancien
- `git show [--stat]` : montre le contenu d'un objet
- `git diff id_commit` : diff entre working copy et commit
- `git diff id_commit1 id_commit2` : diff entre deux commits

```
MINGW64:/d/Mohbenz/Documents/CURSUS/M1/S2/Solveur7erreurs

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/CURSUS/M1/S2/Solveur7erreurs
(master)
$ git log
commit 317f290b597e6b86fcdccbf6a6be06da16a8fc55 (HEAD -> master, origin/master)
Merge: 226cf7b 62d1953
Author: Greyman93 <40568634+Greyman93@users.noreply.github.com>
Date: Thu Apr 18 15:34:10 2019 +0200

    Merge branch 'master' of https://github.com/Mohaabenz/Solveur7erreurs

# Conflicts:
#   app/src/main/java/com/example/solveur7erreurs/Page3.java

commit 62d195390944ba3a05be2050d55661097d05604e
Merge: 84cc6f0 2336a5b
Author: Wyyvel <abdelli.elias@gmail.com>
Date: Thu Apr 18 17:11:42 2019 +0200

    Merge branch 'master' of https://github.com/Mohaabenz/Solveur7erreurs

commit 84cc6f06ff5ec20a0417cb7f8f99852c403a36f6
Author: Wyyvel <abdelli.elias@gmail.com>
Date: Thu Apr 18 17:11:35 2019 +0200

    suite detection(2)
```



Editeur de Texte

- Utilisation d'une bonne IDE est indispensable pour une utilisation optimale.
- Voici une liste d'IDE gratuits les plus connus : Visual Studio Code, Atom, Sublime Text...

Choisissez selon vos préférences et vos habitudes

```
Fichier  Edition  Sélection  Affichage  Atteindre  Exécuter  Terminal  Aide  fichier.html (non suivi) - ExempleGit - Visual Studio Code
```

EXPLORATEUR

ÉDITEURS OUVERTS

- fichier.html... M

EXEMPLEGIT

- fichier.html M

fichier.html (non suivi) M X

fichier.html > html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Document</title>
8 </head>
9 <body>
10-
11 </body>
12 </html>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <title>Document</title>
8 </head>
9 <body>
10+ <h1>Bienvenue sur mon site</h1>
11 </body>
12 </html>
```

You, 10 minutes ago • Ajout fichier.html

Questions ?

Lien utiles :

<https://www.hostinger.fr/tutoriels/commandes-git>

<https://www.youtube.com/watch?v=4o9qzbssflI>



GitHub

Sommaire



GitHub

- Présentation de la plateforme et son intérêt
- Inscription et Visite de la plateforme
- Expliquer les différences entre dépôt private et public
- Ajouter un fichier .readme
- Créer et configurer un répertoire Git.
- Créer un projet complet versionner via Git et « pousser » ses commits sur son dépôt distant
- Créer des branches
- Clôner un projet
- Forks et Pull Request

Présentation de la plateforme et son intérêt



- **GitHub** facilite la collaboration en utilisant git
- Une plateforme qui peut contenir des **dépôts de code**
- Un **stockage** dans le **cloud** afin que plusieurs développeurs puissent travailler sur un même projet
- Voir les **modifications** des autres en **temps réel**
- Le **plus grand hébergeur** de dépôts Git du monde.
- Une grande partie des dépôts hébergés sur GitHub sont **publics**, ce qui signifie que n'importe qui peut télécharger le code de ces dépôts et **contribuer** à leur développement en proposant de nouvelles fonctionnalités.
- En résumé, Git est un logiciel de gestion de version tandis que GitHub est un service en ligne d'hébergement de dépôts Git qui fait office de serveur central pour ces dépôts.

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub Sign in Sign up

Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

Email Sign up for GitHub

65+ million Developers	3+ million Organizations	200+ million Repositories	72% Fortune 500
---------------------------	-----------------------------	------------------------------	--------------------

Inscription



- Un Nom : de préférences Nom.prenom ou PseudoPro
- Un Email
- Un Mot de passe

Remarque : Très peu d'information demander pour s'inscrire

Why GitHub? Team Enterprise Explore Marketplace Pricing Search GitHub Sign in

Create your account

Username *

Email address *

Password *

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

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Veuillez résoudre cette énigme pour nous prouver que vous êtes une personne

Vérifier

Create account

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's

Inscription



- Repositories
- Contributions
- Followers
- Activity
- Project

The screenshot shows the GitHub profile of a user named 'Mohaabenz'. The profile includes a circular avatar with a teal and white geometric design, a bio, and statistics: 5 followers, 2 following, and 0 stars. The 'Popular repositories' section lists four repositories: 'iot-projects' (forked from institut-galilee/iot-projects, 1 fork), 'Solveur7erreurs' (Java, 1 fork), 'DataLogger' (forked from STRCWearlab/DataLogger, 2 forks), and 'utiliser-markdown' (forked from Simplonline-foad/utiliser-markdown). The 'Contributions' section shows a calendar for 2019 with 128 contributions. The 'Organizations' section shows one organization. The page has a dark theme and a navigation bar at the top with links to Overview, Repositories (6), Projects, and Packages.

Overview Repositories 6 Projects Packages

Popular repositories Customize your pins

- iot-projects**
Forked from institut-galilee/iot-projects
1 fork
- Solveur7erreurs**
Java 1 fork
- DataLogger**
Forked from STRCWearlab/DataLogger
Android application for reliable multidevice multisensor big data collection
Java 2 forks
- utiliser-markdown**
Forked from Simplonline-foad/utiliser-markdown
Un fichier vous expliquant comment écrire un fichier Readme avec les balises de la syntaxe Markdown.

Contributions 128 contributions in 2019 Contribution settings

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Mon
Wed
Fri

Learn how we count contributions Less More

Contribution activity 2021

Créer un Repot




- Un nom de repot et une description
- Public ou Privé ?
- README file ?
- .gitignore ?
- Licence ?

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 Mohaabenz ▾

Repository name *

/

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

Description (optional)



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.](#)



Choose a license

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

Create repository

Public ou Privé




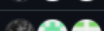

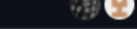


- L'acquisition de la plateforme pour développeurs GitHub par Microsoft fait partie des infos qui ont le plus marqué le monde de la tech en 2018. Et moins d'un an après ce rachat, GitHub modifie déjà ses offres.
- GitHub est gratuit pour les projet open-source ou publics. Pour collaborer sur des projets privés, il fallait souscrire à l'une de ses offres payantes.
- Mais dorénavant, la plateforme propose également les dépôts privés aux utilisateurs qui ne passent pas à la caisse.
- « GitHub Free inclut désormais un nombre illimité de dépôts privés », écrit la plateforme dans un billet de blog. « Pour la première fois, les développeurs pourront utiliser GitHub gratuitement pour leurs projets privés avec trois collaborateurs par dépôt. »
- En musclant son offre gratuite, GitHub compte satisfaire les développeurs qui veulent ouvrir un dépôt privé pour postuler à une offre d'emploi, pour les petits projets ou pour tester quelque chose en privé avant de rendre le code public.

Contributeurs



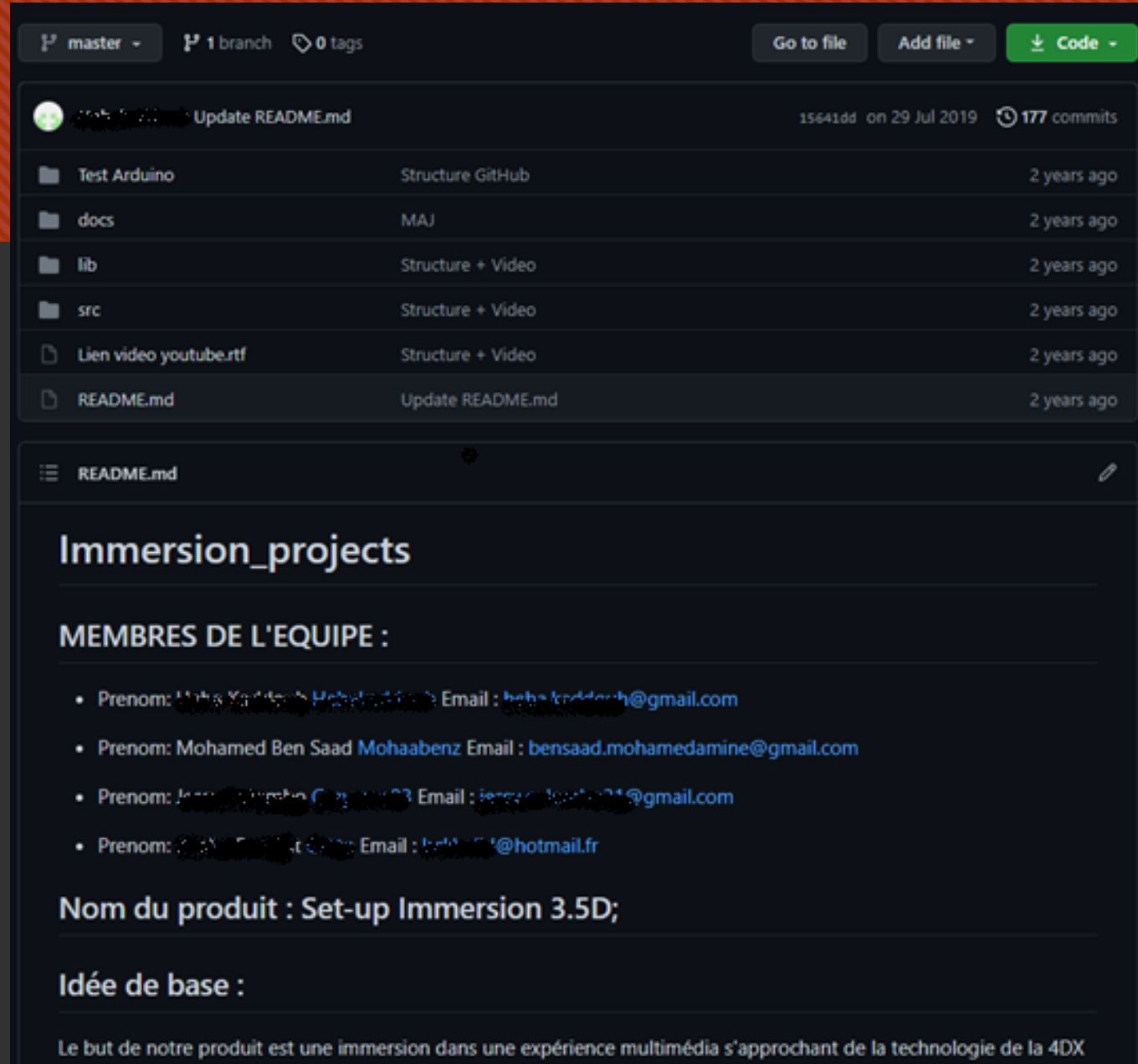
- Pour les repots public le nombre de contributeurs n'est pas limité
- Pour les repots privés le nombres de contributeurs est limité à trois pour la version gratuite

	5 members
	5 members
	5 members
	4 members
	5 members
	3 members
	5 members
	3 members
	4 members
	4 members
	4 members
	4 members
	3 members
	3 members
	3 members
	2 members
	2 members
	3 members
	4 members
	4 members

README file

Qu'est-ce que Markdown? (README.md)

- Markdown est un moyen de styliser du texte sur le Web. Vous contrôlez l'affichage du document; formater des mots en gras ou en italique, ajouter des images et créer des listes...
- Vous pouvez utiliser Markdown dans la plupart des endroits autour de GitHub:
 - Gists
 - Commentaires dans les problèmes et demandes d'extraction
 - Fichiers avec l'extension .md ou .markdown



The screenshot shows a GitHub repository interface. At the top, it indicates the current branch is 'master', there is 1 branch, and 0 tags. Navigation buttons include 'Go to file', 'Add file', and 'Code'. Below this, a commit history table is visible, showing a recent commit 'Update README.md' by user '15641dd' on 29 Jul 2019, with 177 commits in total. A table of repository files follows, listing folders like 'Test Arduino', 'docs', 'lib', 'src' and files like 'Lien video youtube.rtf' and 'README.md', each with its last update time (all 2 years ago). The main content area displays the 'README.md' file, which is titled 'Immersion_projects'. It lists the 'MEMBRES DE L'EQUIPE' (Team Members) with their names and email addresses. Below this, it states the 'Nom du produit' (Product Name) as 'Set-up Immersion 3.5D;' and provides an 'Idée de base' (Basic idea) stating the goal is a multimedia immersion experience approaching 4DX technology.

File	Update	Time
Test Arduino	Structure GitHub	2 years ago
docs	MAJ	2 years ago
lib	Structure + Video	2 years ago
src	Structure + Video	2 years ago
Lien video youtube.rtf	Structure + Video	2 years ago
README.md	Update README.md	2 years ago

Immersion_projects

MEMBRES DE L'EQUIPE :

- Prenom: Maha Kaddouch Email : maha.kaddouch@gmail.com
- Prenom: Mohamed Ben Saad Mohaabenz Email : bensaad.mohamedamine@gmail.com
- Prenom: Jeanne Louna Email : jeanne.louna@gmail.com
- Prenom: ... Email : ...@hotmail.fr

Nom du produit : Set-up Immersion 3.5D;

Idée de base :

Le but de notre produit est une immersion dans une expérience multimédia s'approchant de la technologie de la 4DX

Licences



Les quatre libertés garanties par les licences libres

- Les licences libres sont des contrats passés entre l'auteur d'un logiciel et un utilisateur, celui-ci accordant à celui-ci ces quatre libertés :
 - la liberté d'utiliser le logiciel ;
 - la liberté de copier le logiciel ;
 - la liberté d'étudier le logiciel ;
 - la liberté de modifier le logiciel et de redistribuer les versions modifiées.

Ce principe est commun à toutes les licences libres.

Les deux grandes familles de licences libres

- Il y a deux grandes familles de licences libres :
 - les licences dites « copyleft », qui garantissent que les versions redistribuées du logiciel (modifié ou non) sont publiées avec les mêmes garanties sur les libertés de l'utilisateur ;
La licence copyleft la plus connue est la licence GNU GPL (General Public license).
 - les licences dites « permissives », qui permettent à des versions redistribuées du logiciel (modifié ou non) d'être publiées sous des conditions ne garantissant pas les quatre libertés fondamentales de l'utilisateur.
Les licences permissives connues sont les licences BSD, MIT, Apache.



Liaison avec Git



Ligne de commandes sur le GitBash :

- Créer un nouveau git
- Liaison avec git existant
- Importer un code

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** SSH <https://github.com/> 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# tutoGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/gCKn/tutoGit.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/gCKn/tutoGit.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

Projets (Agile)



institut-galilee / Immersion_projects

Watch 1 Star 0 Fork 1

<> Code 1 Issues 5 Pull requests Actions Projects 1 Wiki Security Insights

Immersion 3.5D Updated on 6 May 2019

Filter cards + Add cards Fullscreen Menu

2 To do

Fauteuil

L'idée est de concevoir un fauteuil émettant des vibrations en fonctions de l'intensité du son produit par une action

Added by Mohaabenz

Immersion 3.5D

#3 opened by Mohaabenz

2 In progress

Rapport du projet

Added by Hebakaddouh

PowerPoint

Added by Hebakaddouh

17 Done

Detection des couleurs primaire RVB sur écran PC

Added by Mohaabenz

Calibrer l'arduino à partir de l'application

Added by Hebakaddouh

Ajout de seekbar

Added by Hebakaddouh

Vidéo

Added by Hebakaddouh

Arduino ESP32 ?

#1 opened by Mohaabenz

Composant LED

3 of 3

#6 opened by Mohaabenz

Maquette de l'application

Added by Greyman93

Recherche de l'existant

Added by Hebakaddouh

Test

Flasher l'ESP32 avec Arduino IDE

Added by Hebakaddouh

application

Added by Hebakaddouh

3 Logiciel

Android Studio

Le développement de l'application se fera sur Android Studio

Maquette de l'application

Front-end

Back-end

Added by Greyman93

Système d'exploitation

La conception du programme pourras se faire sur windows et linux

Added by Hebakaddouh

ARDUINO IDE

Nous allons coder notre esp32 grace a ce logiciel pour adapter la TCS3200 qui fonctionne sur arduino

Added by Hebakaddouh

2 Composant

Composants LED

Arduino

Bande Led couleur

TCS3200

Fil de connexion

Added by Mohaabenz

Composant Fauteuil

Housse

Moteur Vibrant

ESP32

Capteur sonore

Enceinte Bluetooth

Added by Mohaabenz

3 Liens utiles

TCS3200 Librairie :

<https://github.com/blascarr/TCS3200-ColorSensor>

Added by Mohaabenz

Seekbar :

<https://o7planning.org/fr/10507/tutorial-android-seekbar>

Added by Hebakaddouh

Bluetooth pour esp32 :

<https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/>

tuto recherche bluetooth :

<https://www.youtube.com/watch?v=DPbCFAQCfHE>

customisation d'une seekbar:

<http://www.zoftino.com/android-seekbar-and-custom-seekbar-examples>

<https://a-renouard.developpez.com/tutoriels/android/custom-progress-seek-bar/>

Added by Greyman93



Fetch, Pull & Push

- La commande `git fetch <nom-distant>` permet de récupérer toutes les données d'un dépôt distant qu'on ne possède pas déjà. C'est-à-dire de récupérer les nouvelles version à jour du projet.
- On peut également utiliser `git pull` pour récupérer les données d'une branche distante et les fusionner automatiquement avec la branche locale, dans le cas où on a créé une branche sur notre projet et pour suivre les avancées d'une branche distante
- Une fois qu'on a terminé nos modifications localement, on va les pousser vers le dépôt distant. On utilise pour cela la commande `git push [nom-distant] [nom-de-branche]`. Si on souhaite par exemple pousser les modifications faites sur notre branche master vers le serveur origin, on écrira `git push origin master`

```
MINGW64:/d/Mohbenz/Documents/CURSUS/M1/S2/Solve
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Docume
(master)
$ git pull
warning: refname 'HEAD' is ambiguous.
warning: refname 'HEAD' is ambiguous.
warning: refname 'HEAD' is ambiguous.
warning: refname 'HEAD' is ambiguous.
Already up to date.

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Docume
(master)
$ git push
Logon failed, use ctrl+c to cancel basic creden
Username for 'https://github.com': Mohaaben
Everything up-to-date

momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Docume
(master)
$
```


Master & Branch

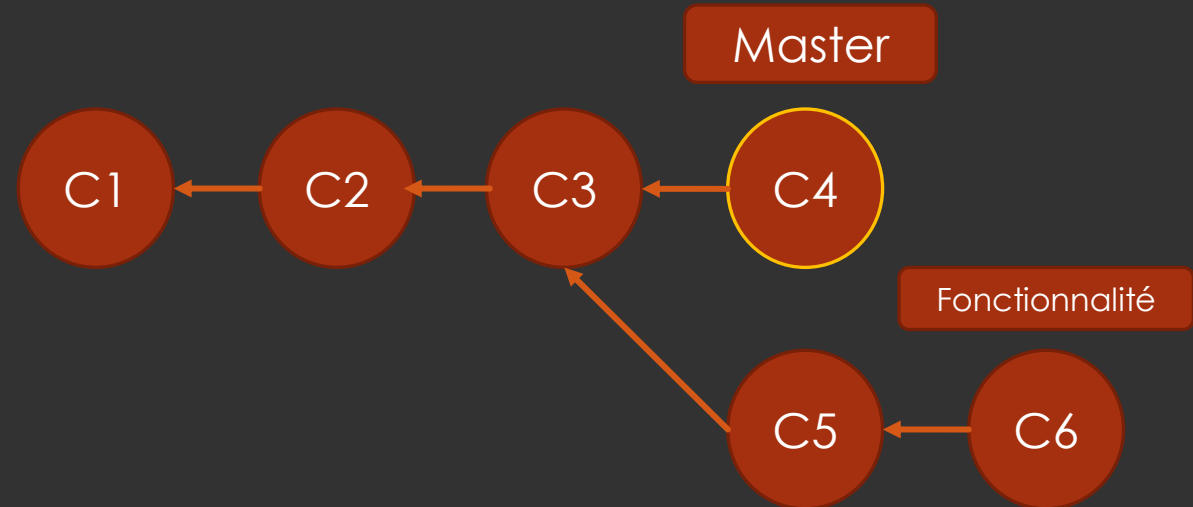


- Créer une **branche**, c'est en quelque sorte comme créer une "copie" de votre projet pour développer et tester de nouvelles fonctionnalités sans impacter le projet de base.
- Git rend la création de nouvelles branches et la **fusion** de branche très facile à réaliser. Une branche est simplement un pointeur vers un **commit**.
- La branche par défaut dans Git s'appelle **master**. Cette branche master va se déplacer automatiquement à chaque nouveau commit pour pointer sur le dernier commit effectué tant qu'on reste sur cette branche.
- Pour rappel, lorsqu'un commit est effectué, Git stocke un objet qui contient les **nom** et **prénom** de l'auteur, le **message** renseigné lors de sa création ainsi qu'un **pointeur** vers l'**index** et des pointeurs vers le(s) commit(s) précédant directement le commit courant.
- En résumé, une branche est un pointeur vers un commit en particulier. Ainsi, créer une nouvelle branche dans Git, c'est créer simplement un nouveau pointeur au lieu de recopier l'intégralité du projet.

Master & Branch



- Branche : pointeurs vers un commit
- Master : branche principale
- Head : branche courante
- Privilégier une fonctionnalité par nouvelle branche



Master & Branch



- Afficher les branches : `git branch`
- Créer une branche : `git branch <ma branche>`
- Déplacer le HEAD vers une branche : `git checkout <ma branche>`
- Créer et Déplacer en même temps : `git checkout -b <ma branche>`

```
MINGW64:/d/Mohbenz/Documents/SR3/Formation/Cours Git/Exem...
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3
it/ExempleGit (master)
$ git branch
* master

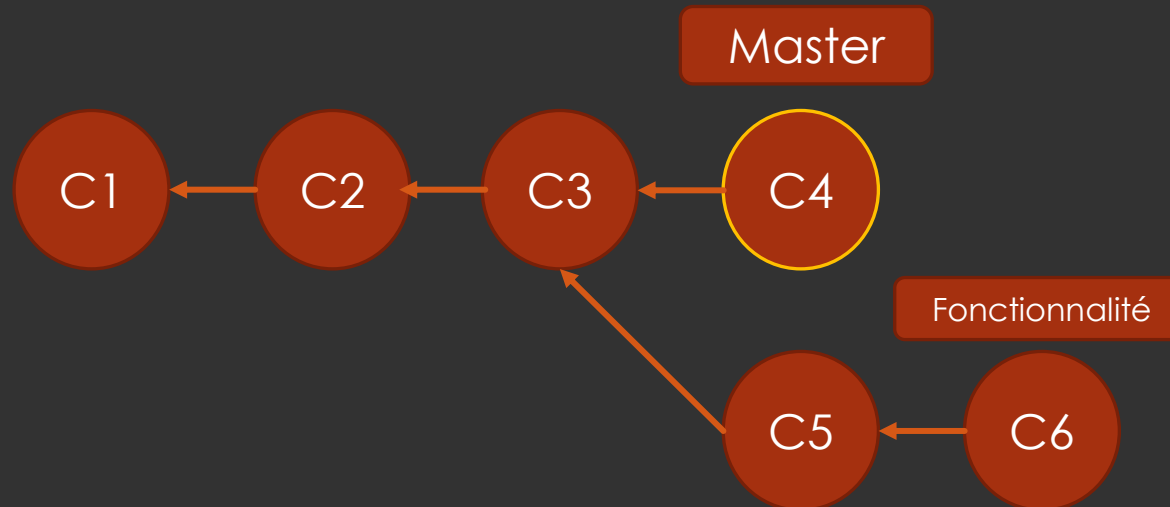
momo_@LAPTOP-RHC9TS90 MINGW64 /d/Mohbenz/Documents/SR3
it/ExempleGit (master)
$ git checkout master
Already on 'master'
D      fichier.html
```

Master & Branch



- Merge (fusion) : intégrer les modification de la branche vers la courante.

```
git merge <ma branche>
```

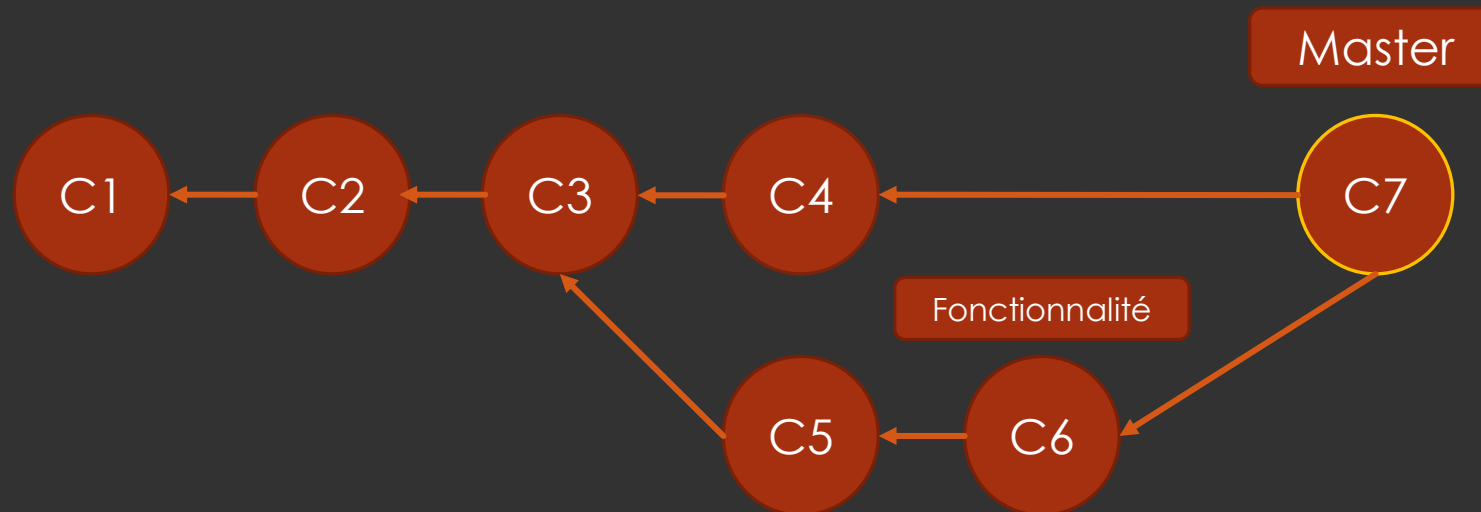


Master & Branch



- Merge (fusion) : intégrer les modification de la branche vers la courante.

```
git merge <ma branche>
```



Clone

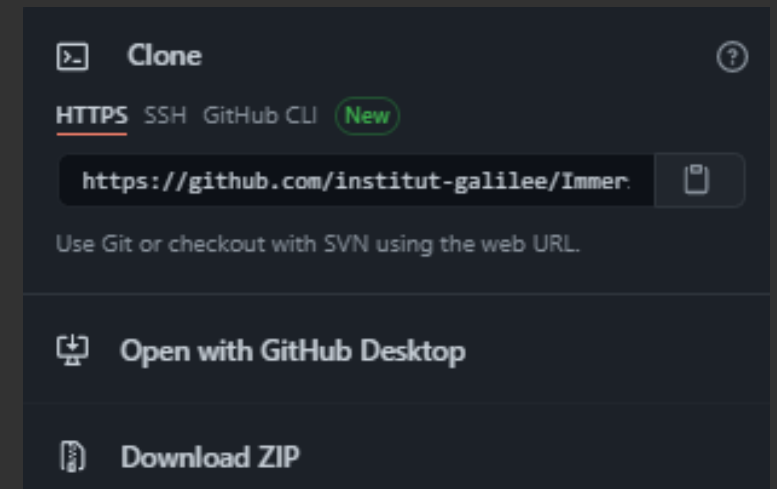


- La commande `git clone` est utilisée pour la vérification des dépôts. Si le dépôt se trouve sur un serveur distant, utilisez:

```
git clone serveur:/chemin/vers/dépôt ou adresse
```

- Inversement, si une copie de travail d'un dépôt local doit être créée, utilisez:

```
git clone /chemin/vers/dépôt
```





Fork & Pull Request

○ Fork :

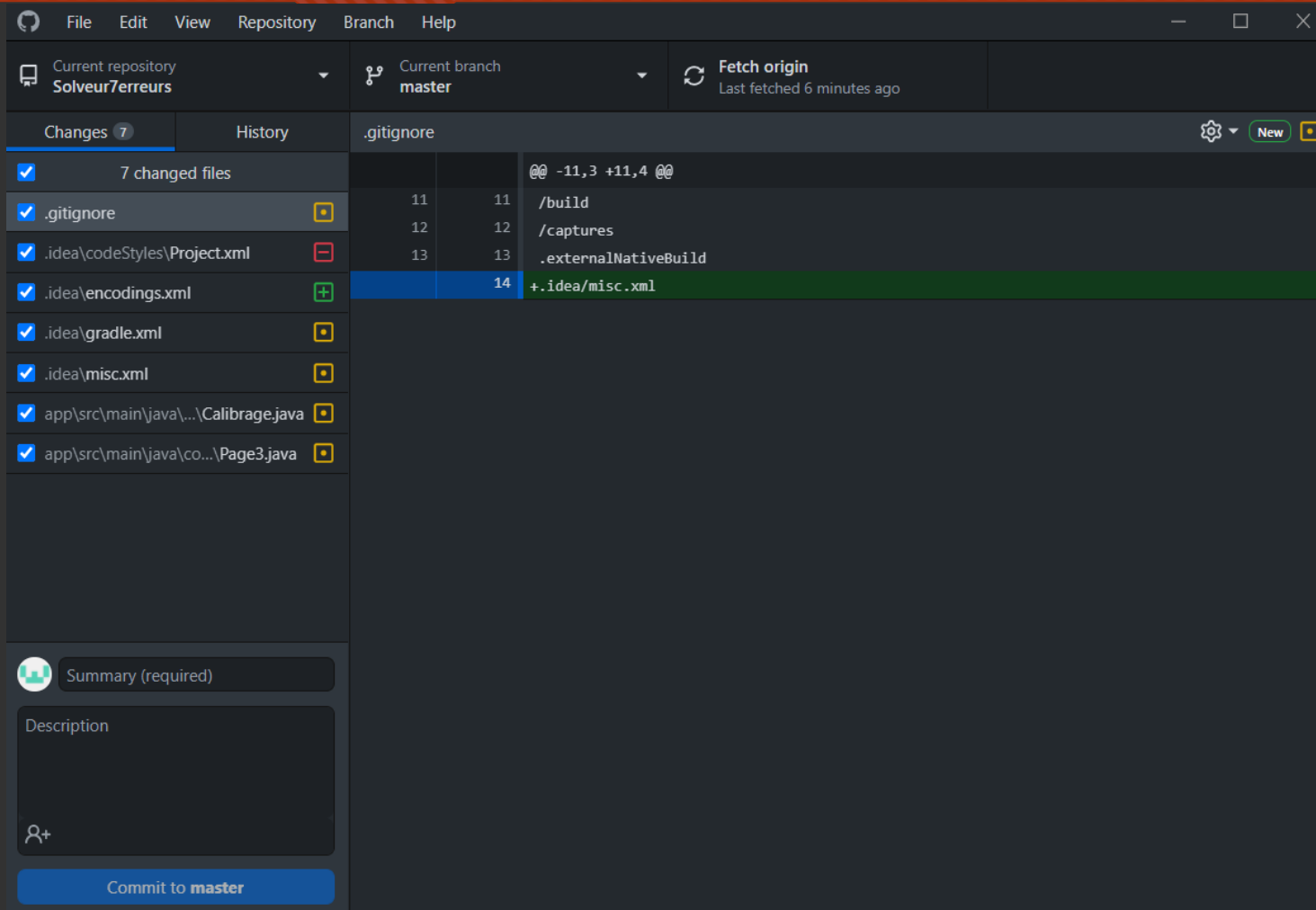
Un fork désigne une copie d'un dépôt. En effet, par défaut il n'est pas possible de faire de commit sur un dépôt qui ne nous appartient pas (heureusement sinon ça serait l'anarchie). Du coup, les services ont introduit cette notion de fork qui permet de se retrouver avec un dépôt sur lequel on aura la permission d'écriture

○ Pull request :

La notion de pull request va de paire avec le système de Fork. Une fois que l'on a travaillé sur notre fork on souhaite souvent proposer à l'auteur original nos modifications. On fera alors un pull request qui consiste tout simplement à demander à l'auteur original de merge nos modifications.

La différence entre un **fork** et un **clone** est que lorsqu'on fork une connexion existe entre notre fork (notre copie) et le projet original. Cela permet notamment de pouvoir très simplement contribuer au projet original en utilisant des **pull requests**, c'est-à-dire en poussant nos modifications vers le dépôt distant afin qu'elles puissent être examinées par l'auteur du projet original.

Github Desktop



- Pour Windows et Mac
- Connecté à vos compte Github
- Cloner facilement
- Créer Localement et pousser
- Plus besoin de GitBash

Github Desktop



File Edit View Repository Branch Help

Current repository: Solveur7erreurs | Current branch: master | Fetch origin (Last fetched 12 minutes ago)

Changes (7) | History

Select branch to compare...

Nettoyage du code

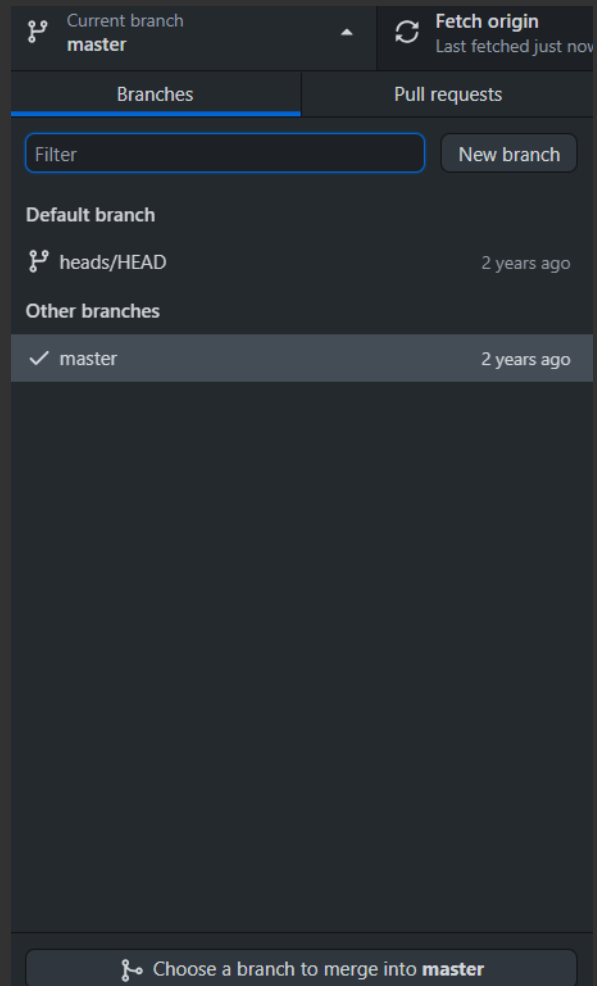
Mohaabenz · dc3f8b1 · 1 changed file · New

app\src\main... \Calibrage.java

Line	Diff	Code
49		}
50		
51		public static Calibrage recherche(Bitmap im1, Bitmap im2, int r1){
52	-	Bitmap zone2, zone3;
52	+	Bitmap zone2;
53	+	Bitmap zone3 = null;
53		double p = 100;
54	-	double p1, p2;
55	+	double p1;
56	+	double p2= 100;
55		int x1, y1, x2, y2, z1, z2, x11, y11, x12, y12, z11, z12;
56		Calibrage cal = new Calibrage(0,0,100,0, r1);
57		int r = 0;
58		while(r<=5)
59		/*for(int r=-1 ; r<=0 ; r++) */{
60		zone2=zone(rotateBitmap(im1,r));
61	-	x1 = ((im2.getWidth()/5));
62	-	y1 = ((im2.getHeight()/5));
63	+	if(r!=0) {
64	+	zone3 = zone(rotateBitmap(im1, -r));
65	+	}
66	+	x1 = ((im2.getWidth()/9));
67	+	y1 = ((im2.getHeight()/9));
63		z1 = zone2.getWidth();
64		z2 = zone3.getHeight();

- Historiques
- Envoi de Commit
- Différences

Github Desktop



- Créer, modifier ou supprimer des branches
- Fusionner des branches
- Fetch Origin
- Push
- Pull

Questions ?

Lien utiles :

<https://github.com/>

<https://www.pierre-giraud.com/git-github-apprendre-cours/>

<https://devstory.net/10283/utiliser-github-avec-github-desktop>

Merci...