

Git - Gérer le versioning

GIT-VER

m2information.fr



Déroulement de la formation

- Jour 1
 - Présentation de Git
 - Comprendre les principes de Git
 - Prise en main
 - Travailler en équipe
 - Gestion des branches
- Jour 2
 - Compléments

PRÉSENTATION DE GIT

The background is a solid red color. In the lower-left corner, there are stylized silhouettes of three people standing in a row. The person in the middle is slightly taller than the two on either side. Overlaid on the middle silhouette is a large, bold, dark red number '2'.

2

Présentation de Git (1/3) - Présentation et utilité

- Logiciel de gestion de versions
 - Permet de gérer l'évolution du contenu d'une arborescence via une architecture client/serveur
 - Sous licence GNU (libre et open-source)
- Pourquoi l'utiliser ?
 - Suivre les changements d'un projet
 - Gérer les conflits d'édition
 - Réaliser des sauvegardes régulières

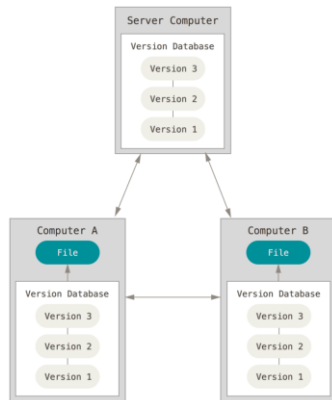
Présentation de Git (2/3) - Comparaison avec subversion (SVN)

GIT

Logiciel de gestion de versions décentralisé

« copie locale »
dépôt à part entière

Permet à ce titre de faire des
« commits » locaux

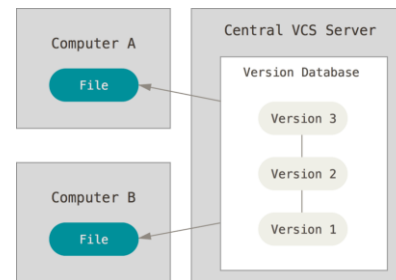


SVN

Logiciel de gestion de versions centralisé

« copie locale »
copie en lecture du dépôt

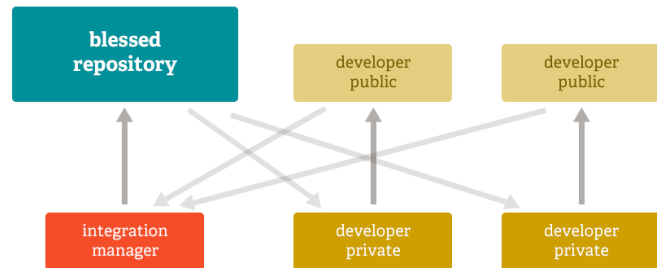
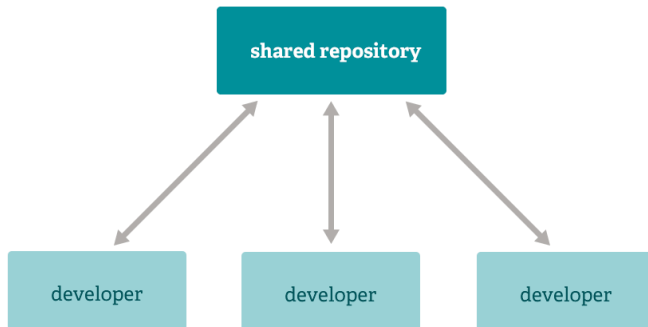
Les commits sont envoyés
directement au serveur



Présentation de Git (3/3) - Aperçu des flux de travaux possibles

« Centralisé »

« Responsable »
ayant autorité



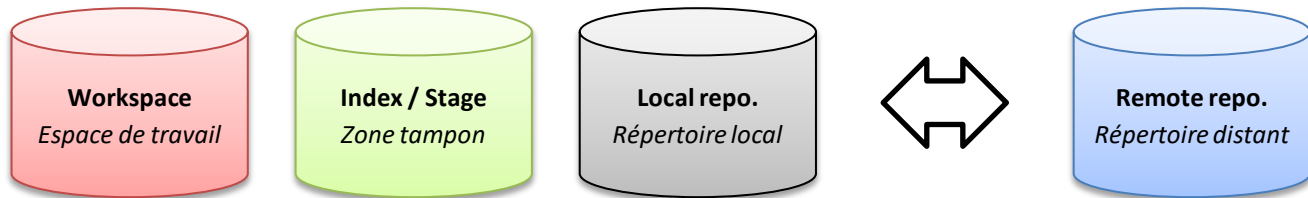
COMPRENDRE LES PRINCIPES DE GIT

The background is a solid red color. At the bottom, there are stylized silhouettes of three people in a darker red shade. The person in the middle has a large number '2' on their chest. The person on the right has a large number '1' on their chest. The person on the left is partially cut off.

2

Présentation des concepts de Git - Les différents flux et la décentralisation

- **Remote repository** Historique du projet sur le serveur (*dossier .git*)
- **Local repository** Historique du projet sur le client (*dossier .git*)
- **Index / Stage** Liste des fichiers indexés
- **Workspace** Liste des fichiers modifiés et/ou supprimés



Présentation des concepts de Git - Aperçu des flux de travaux possibles

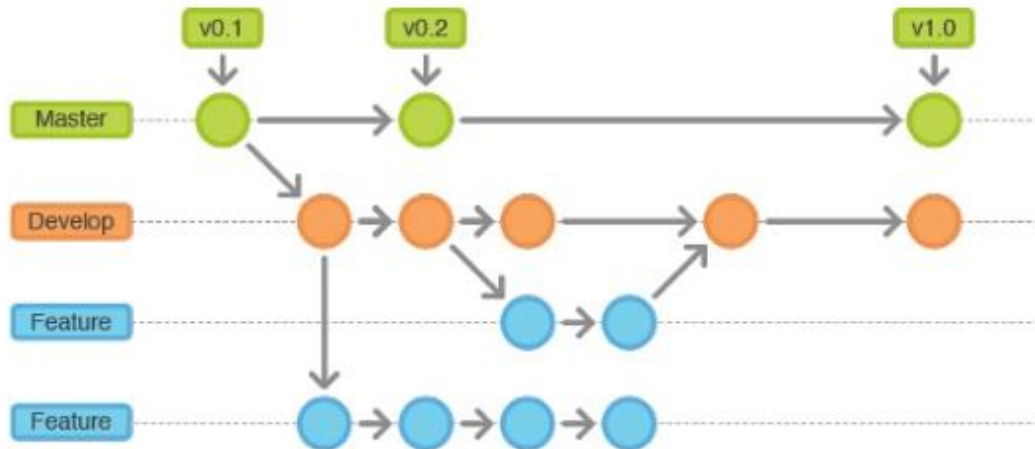
- **untracked** Nouveaux fichiers (*non connus de Git*)
- **modified** Fichiers connus de Git et **modifiés** (*dans le workspace*)
- **staged** Fichiers connus de Git, modifiés et **indexés**
- **stable** Fichiers connus de Git et **non modifiés**
- **deleted** Fichiers connus de Git et **supprimés** (*dans le workspace*)

HEAD : pointeur sur la référence de la branche actuelle, qui est à son tour un pointeur sur le dernier *commit* réalisé sur cette branche



Présentation des concepts de Git - Les branches

- Permet de créer des sous-espaces de travail
 - Chaque branche peut évoluer de manière séparée
 - On peut basculer d'une branche à l'autre « à tout moment »



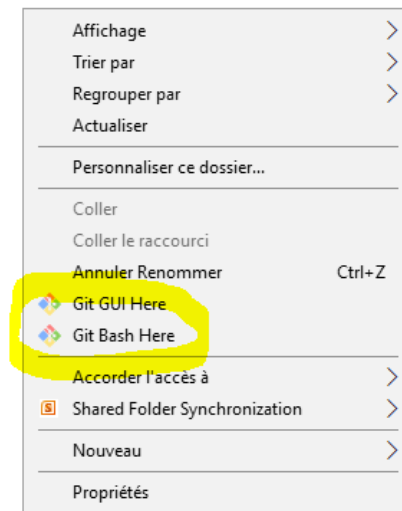
PRISE EN MAIN

A graphic at the bottom of the slide featuring three stylized red silhouettes of people. The central figure is slightly taller and has a large, bold, dark red number '2' superimposed on its torso. The two flanking figures are shorter and positioned slightly behind the central one.

2

Prise en main (1/2) - Installation et configuration

- Installation sous Windows <https://git-scm.com>
 - Git Bash *émulateur de console Unix*
 - Git GUI *interface graphique*
 - Intégration automatique dans Windows
- Configuration
 - **git config** --global user.name "votre_pseudo"
 - **git config** --global user.email moi@email.com
 - **git config** --list



Prise en main (1/2) - Nouveautés

- (v2.25) Nouvelles commandes : **git restore** / **git switch**
 - Effacer des modifications
 - **git checkout** *file* **git restore** *file*
 - Changer de branche
 - **git checkout** *branch* **git switch** *branch*
 - **git checkout** -b *branch* **git switch** -c *branch*
- Revenir en arrière dans l'historique
 - **git checkout** *ab12c3*

Prise en main (2/2) - Commandes principales

- Commandes pour démarrer
 - **git init** *Création du dépôt dans le répertoire courant*
 - **git init mon-projet** *Création du dépôt dans le répertoire mon-depot*
 - **git clone https://...** *Clonage d'un dépôt depuis un serveur distant*
 - **git status** *Affiche l'état du dépôt*

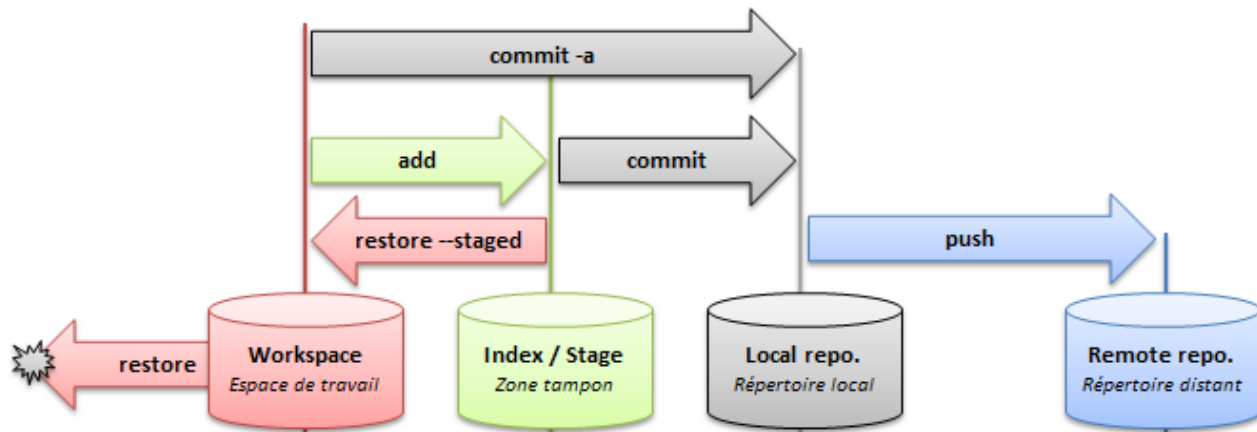
```
F2000@F2000-PC MINGW64 /d/www/formation (master)
$ git status
On branch master

Initial commit

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

Prise en main (2/2) - Commandes principales

- Commandes pour envoyer des modifications
 - **git add** *Ajoute des fichiers dans l'index local*
 - **git restore --staged** *Retire des fichiers de l'index local*
 - **git rm [--cached]** *Supprime des fichiers de l'index local*
- **git commit** *Compacte l'index local au sein d'un « commit »*
- **git push** *Envoie les « commits » locaux sur le serveur*



Exercice

Créer un premier dépôt en local.

Créer un premier fichier
« hello.txt » contenant
« Hello world ! »

Ajouter et intégrer ce fichier
dans un « commit ».

Tester la commande « *git push* »

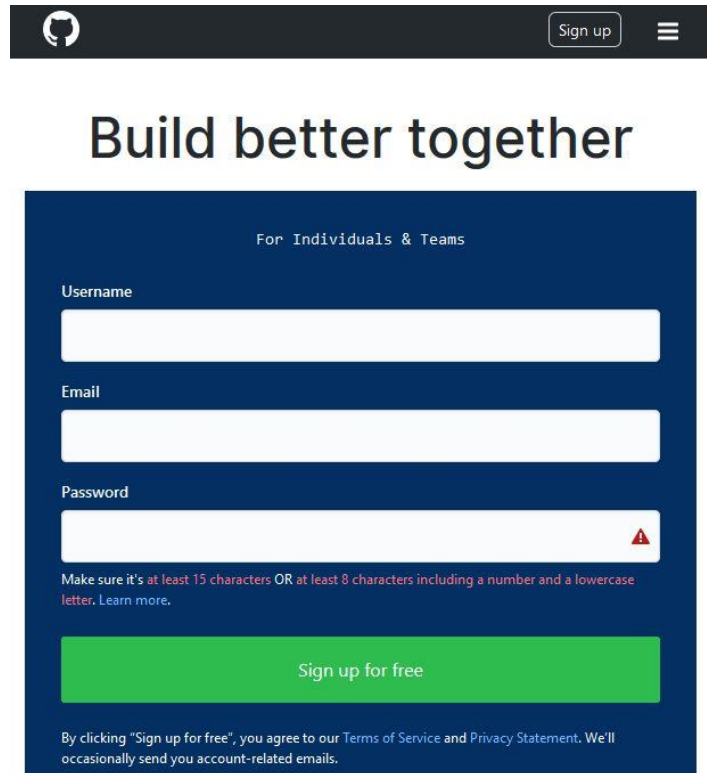


Service web

Hébergement et de gestion de projets / code source (via Git)

Fonctionnalités annexes

- « Bugtracker »
- « Wiki »



The screenshot shows the GitHub sign-up interface. At the top, there's a dark header with the GitHub logo on the left, a 'Sign up' button in the center, and a hamburger menu icon on the right. Below the header, the main heading 'Build better together' is displayed in a large, bold font. Underneath this, the text 'For Individuals & Teams' is shown in a smaller font. The sign-up form consists of three input fields: 'Username', 'Email', and 'Password'. The 'Password' field has a red warning icon on the right. Below the 'Password' field, there is a line of text: 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.' At the bottom of the form is a large green button labeled 'Sign up for free'. Below the button, there is a small line of text: 'By clicking "Sign up for free", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account-related emails.'

Exercice

Créer un dépôt distant
grâce à Github.

Envoyer le commit
précédent sur le dépôt
distant.

Astuce :

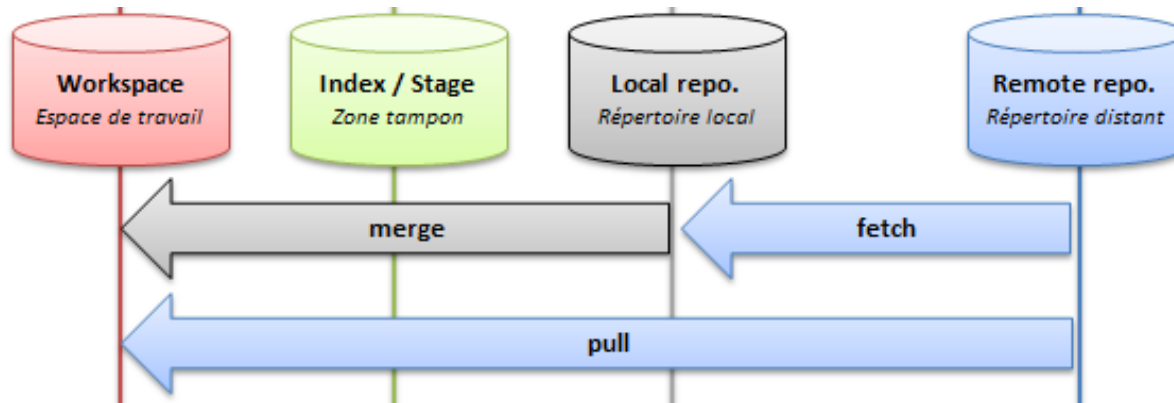
git remote add ...

Pour ajouter un dépôt
distant

The screenshot shows the GitHub homepage with the header navigation bar. The main heading is "Build better together". Below it, a subheading states: "From [open source](#) to [business](#), you can host and review code, manage projects, and build software alongside 31 million developers." The sign-up form is divided into two sections: "For Individuals & Teams" and "For Enterprises". The "For Individuals & Teams" section contains input fields for "Username", "Email" (with the placeholder "F2000-FR"), and "Password" (with a strength indicator). Below these fields is a green "Sign up for free" button. The "For Enterprises" section is titled "Get started with Enterprise" and includes the text: "Take collaboration to the next level with security and administrative features built for businesses." It also features links for "Start a free trial of Enterprise Server" and "Get started with Enterprise Cloud", along with a blue "Contact Sales" button.

Prise en main (2/2) - Commandes principales

- Commandes pour recevoir des modifications
 - **git pull ...** *Récupère des modifications depuis le serveur et les applique sur le répertoire de travail*
 - **git fetch ...** *Récupère des modifications depuis le serveur*
 - **git merge** *Applique les modifications sur le répertoire de travail*



Exercice

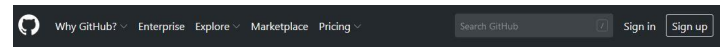
Sur Github, créer un fichier README.md et créer un commit.

Essayer les commandes suivantes :

- **git status**
- **git push**

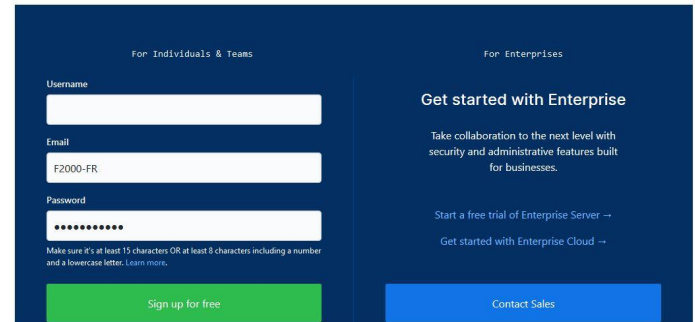
Puis dans un second temps

- **git fetch**
- **git status**
- **git merge**



Build better together

From [open source](#) to [business](#), you can host and review code, manage projects, and build software alongside 31 million developers.

The image shows the GitHub sign-up form. It is divided into two sections: 'For Individuals & Teams' and 'For Enterprises'. The 'For Individuals & Teams' section has input fields for 'Username', 'Email' (with the example 'F2000-FR'), and 'Password' (with a masked password '*****'). Below these fields 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.' and a green 'Sign up for free' button. The 'For Enterprises' section has the heading 'Get started with Enterprise', a description: 'Take collaboration to the next level with security and administrative features built for businesses.', and two links: 'Start a free trial of Enterprise Server ->' and 'Get started with Enterprise Cloud ->'. At the bottom is a blue 'Contact Sales' button.

TRAVAILLER EN ÉQUIPE



Voir les différences : **git diff**

- Lorsque l'on modifie plusieurs fichiers, il peut être utile de réafficher les modifications effectuées
 - **git diff**
 - affiche les modifications des fichiers modifiés, non indexés
 - **git diff --cached**
 - affiche les modifications des fichiers modifiés et indexés
 - **git diff HEAD**
 - affiche les modifications par rapport au dépôt local
- On peut également afficher des modifications entre des branches ou des commits
 - **git diff *master origin/master***
 - affiche les modifications de la branche locale par rapport à la branche distante (« locale »)
 - **git diff *master develop***
 - affiche les modifications de la branche locale « master » par rapport à la branche locale « develop »
 - **git diff *hash1 hash2***
 - affiche les modifications entre les deux commits indiqués

Voir l'historique des changements : **git log** / **git show**

- Après de nombreux « *commits* », il peut être intéressant d'afficher l'historique des modifications
 - **git log**
 - affiche les différents commits effectués sur le dépôt
 - **git log -p**
 - affiche le détail des différents commits effectués sur le dépôt
 - **git show *hash***
 - Affiche le détail d'un commit spécifique grâce à son « hash »

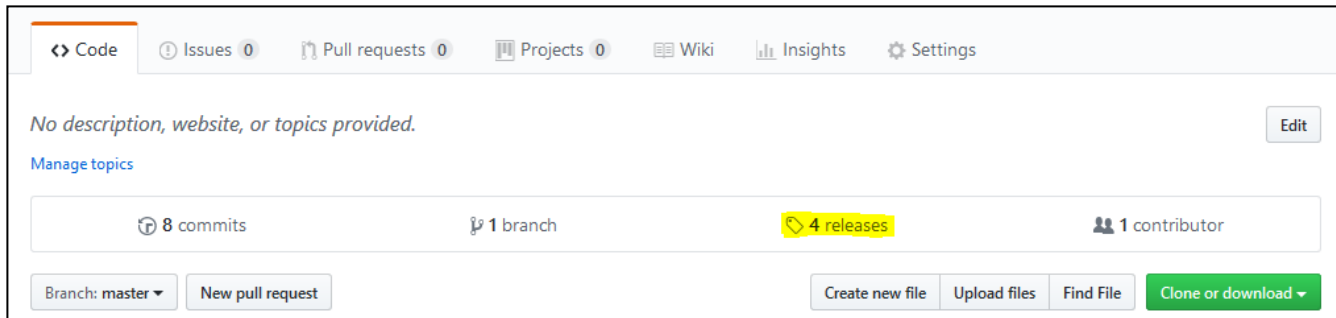
Etiqueter des versions : **git tag**

- De temps en temps, il peut être utile de « taguer » un état du projet (ex: v1, v2, etc.)
 - **git tag**
 - affiche les étiquettes existantes
 - **git tag v1 -m « Version 1 »**
 - crée l'étiquette « v1 », *sur le dernier commit*, avec comme message « Version 1 »
 - **git tag v1 hash -m « Version 1 »**
 - crée l'étiquette « v1 », *sur le commit spécifié*, avec comme message « Version 1 »
 - **git show v1**
 - permet d'afficher le détail de l'étiquette
 - **git push origin v1**
 - permet d'envoyer l'étiquette sur le serveur
 - « *git push origin --tags* » pour envoyer toutes les étiquettes

Exercice

Créer un tag localement et l'envoyer sur le dépôt distant (Github)

Y accéder ensuite sur Github via **Code > releases**



Gestion des conflits

- Survient dès lors qu'un même fichier a été modifié par des « *commits* » différents sur des lignes communes
 - Soit Git pourra corriger les conflits automatiquement
 - Soit Git vous donnera la main pour les corriger

```
F2000@F2000-PC MINGW64 /d/www/formation (master)
$ git merge
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

F2000@F2000-PC MINGW64 /d/www/formation (master|MERGING)
$ cat README.md
<<<<<<< HEAD
# Projet Git

Ceci est une formation M2i.
=====
# FooBar|

FooBar is a Python library for dealing with word pluralization.
>>>>>> refs/remotes/origin/master
```

Exercice

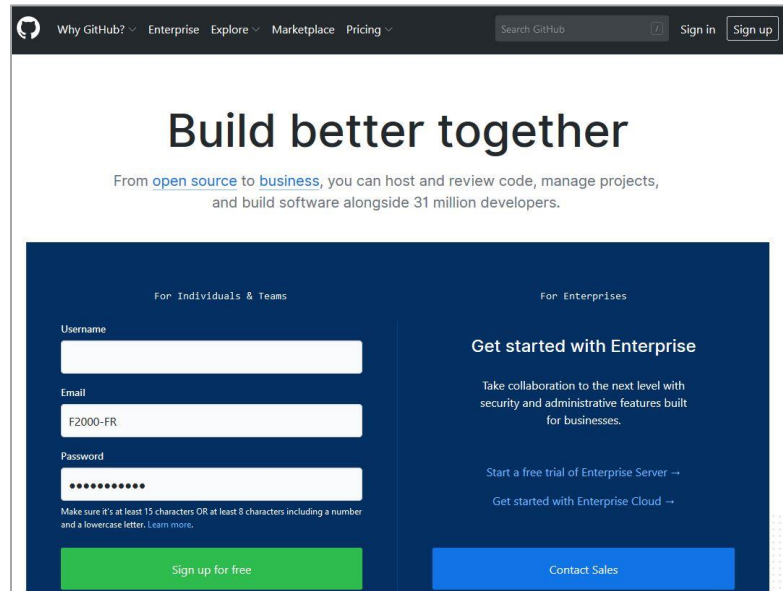
Sur Github, modifier le fichier README.md et créer un commit.

Modifier également le fichier README.md en local et créer un commit.

Effectuer les commandes suivantes :

- **git fetch**
- **git status**
- **git merge**

Corriger le conflit

The image shows the GitHub sign-up page. At the top, there's a dark navigation bar with links like 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. A search bar and 'Sign in'/'Sign up' buttons are on the right. The main heading is 'Build better together'. Below it, a subtext says 'From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.' The page is split into two columns. The left column, 'For Individuals & Teams', has a sign-up form with fields for 'Username', 'Email' (containing 'F2000-FR'), and 'Password' (with a strength indicator). A green 'Sign up for free' button is at the bottom. The right column, 'For Enterprises', has the heading 'Get started with Enterprise', followed by text about collaboration, security, and administrative features. It includes links for 'Start a free trial of Enterprise Server' and 'Get started with Enterprise Cloud', and a blue 'Contact Sales' button at the bottom.

Annuler des actions (1/2) : sur le dépôt local

- Modifier le dernier commit non propagé
 - La commande « **git commit --amend** » permet de modifier un commit local (sur le « local repository »)
- Annuler le dernier commit non propagé
 - La commande « **git reset HEAD~n** » permet d'annuler N commits locaux et remet les modifications dans le « workspace »
 - L'option « --soft » garde l'indexation précédemment réalisée
 - L'option « --hard » efface définitivement les modifications
- Désindexer un fichier
 - La commande « **git restore --staged** » ou « **git reset HEAD ...** » permet de désindexer les fichiers spécifiés (ou tout l'index courant pour « git reset HEAD »)
- Réinitialiser un fichier modifié
 - La commande « **git restore** » ou « **git checkout** » permet de réinitialiser toutes les modifications locales d'un fichier.

Exercice

- Tester l'amendement de commit
 - Créer un commit C1, puis le modifier en C1'
- Créer 2 nouveaux commits et les annuler
 - Créer C2 et C3, puis revenir à C1'
- Envoyer le résultat (C1') sur le serveur

Astuce : vérifier l'état courant via « *git log* »

Annuler des actions (2/2) : sur le dépôt distant

- Annuler le dernier commit propagé sur le serveur
 - « **git reset --hard HEAD~n** » revient en arrière de N commits
 - « **git push** » refusé par Git si les commits ont été propagés sur le serveur
 - « **git push -f** » permet de pousser « en force » mais est très dangereux à utiliser puisque cela écrase l'historique du serveur
- Bonne méthode : appliquer un commit « inverse »
 - La commande « **git revert hash** » permet d'annuler un commit présent sur le serveur (ou créant son commit inverse). Il faut ensuite propager ce commit sur le serveur.
 - « **git revert HEAD~3..HEAD** » permet d'annuler les trois derniers commits (et va créer 3 commits inverses)

Exercice

- Annuler le commit C1' précédemment envoyé sur le serveur (méthode 1)
 - **git reset --hard [...]**
- Refaire un commit C1' (et l'envoyer sur le serveur) puis l'annuler
 - **git revert [...]**