

Le gestionnaire de versions



Sana EL ATCHIA

Plan

Introduction

Installer et configurer Git et GitHub

Utiliser les commandes de base de Git

Corriger les erreurs courantes sur GitHub

Introduction

Problématique :

- ❖ Difficulté de gérer l'historique des fichiers de projets

But :

- ❖ Gérer et déployer les projets informatiques avec git

Gestionnaire de versions

- est un programme qui permet de conserver un historique des modifications et des versions de tous les fichiers.
- est appelé aussi **système de contrôle de versions**
- permet de garder en mémoire les modifications sur chaque fichier et par qui
- Fonctionnalités :
 - Revenir à une version précédente du code en cas de problème.
 - Suivre l'évolution du code étape par étape.
 - Travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs.
- Exemple : Git, Mercurial

Git ou GitHub

Git est un gestionnaire de versions est utilisé pour créer un **dépôt local** et gérer les versions de vos fichiers.
GitHub est un service en ligne héberge les dépôts Git. On parle de **dépôt distant**.



Dépôt local vs Dépôt distant

Dépôt (Repository)

C'est comme un dossier qui conserve un historique des versions et des modifications d'un projet.

Il peut être local ou distant.

Dépôt local :

C'est un entrepôt virtuel du projet.

C'est l'endroit où on stocke, sur la machine, une copie du projet

Il permet d'enregistrer les versions de votre code et d'y accéder au besoin.

Dépôt distant :

Il permet de stocker les différentes versions du code afin de garder un historique hébergé sur Internet ou sur un réseau

Remarques : Le dépôt local est un clone de dépôt distant.



Plateforme d'hébergement du code



Projet avec GitHub

- Créer un compte gitHub :

- ❑ Voici le lien : <https://github.com/>

- Interface repository

- ❑ C'est l'emplacement où vous pourrez créer et retrouver vos dépôts existants.

- ❑ Pour mettre votre projet sur GitHub, on doit créer un repository.
=> Créer le dépôt « LGC_DEV3_G1 »:

Créer le dépôt local- deux méthodes

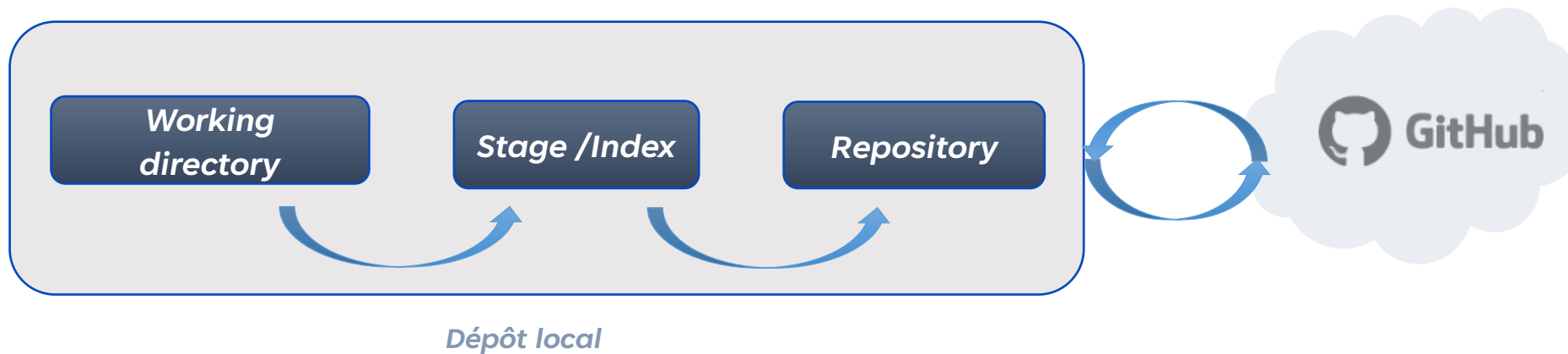
- **Créer un dépôt local vide** pour accueillir un nouveau projet
- **Cloner un dépôt distant**, c'est-à-dire rapatrier l'historique d'un dépôt distant en local

Installer et configurer git- Créer un dépôt local vide

- Télécharger et installer git:
 - ❑ Voici le lien : <https://git-scm.com/downloads>
- **Lancer Git Bash** : C'est une interface qui permet d'utiliser Git en ligne de commande
 - ❑ Accéder à ce dossier : **cd Documents/**
 - ❑ Créer un dossier un dossier dans lequel toutes vos modifications seront enregistrées : **mkdir projet_dev3**
 - ❑ Accéder à ce dossier : **cd projet_dev3**
 - ❑ Configurer GIT : git config permet de réaliser la configuration pour tous les projets git
 - **Configurer l'identité avec le nom et l'email :**
 - **git config --global user.name 'Sanny'**
 - **git config --global user.email 'sanaeelatchia@gmail.com'**
 - **Configurer les couleurs afin d'améliorer la lisibilité des différentes branches**
 - **git config --global color.diff auto**
 - **git config --global color.status auto**
 - **git config --global color.branch auto**
 - **Configurer l'éditeur :**
 - **git config --global core.editor vim** ou **git config --global core.editor notepad++**
 - **git config --global core.tool vimdiff**
- ❑ Initialisation du projet git : **git init**

Fonctionnement de git

- **Créer un dépôt local vide** pour accueillir un nouveau projet
- **Cloner un dépôt distant**, c'est-à-dire rapatrier l'historique d'un dépôt distant en local



Fonctionnement de git – *add/commit/push/pull*

- Utiliser le dépôt git créé :

cd C:\Users\Iagra\Documents\projet_dev3

- Ajouter un fichier dans le dépôt git

touch index.html

- Le fichier est dans le working directory, on peut le voir avec :

git status

- Pour ajouter le fichier dans l'index, on tape :

git add index.html // le fichier est dans le **stage**

- Création d'une nouvelle version du projet et de passer le code vers le repository avec :

git commit -m "mettre un commentaire clair"

- Relier le dépôt local avec le dépôt distant :

git remote add origin https://github.com/Sana-El-Ath/LGC_DEV3_G1.git

git branch -M main

- Passer le code du repository vers le dépôt distant (gitHub)

git push -u origin main

- Mettre à jour le dépôt local

git pull origin main



Fonctionnement de git - Branch

- Une branche correspond à une copie du code principal à un instant donné.
- Créer une branche par fonctionnalité
- La branche principale est appelée la branche **main** ou **master** .
- La branche principale contient l'intégralité des modifications effectuées. Le but n'est donc pas de réaliser les modifications directement sur cette branche, mais de les réaliser sur d'autres branches, et après divers tests, de les intégrer sur la branche principale.

Fonctionnement de git - Branch


- Connaître les branches présentes dans le projet avec la commande:
git branch
- Créer une branche en local avec la commande :
git branch contrat
- Basculer d'une branche à une autre avec la commande :
git checkout contrat
- Appliquer des modifications sur la branche contrat, exemple :
touch contrat.html
git add contrat.html
git commit -m
git commit -m "ajout d'un contrat pro "
git push -u origin contrat
- Fusionner les deux branches : Intégrer l'évolution réalisée dans la branche « contrat » à la branche principale "main"
git checkout main
git merge contrat

Fonctionnement de git - Branch

- Supprimer une branche :
`git branch -d nomBranche`

Fonctionnement de git – Clone /Pull Request

- Accéder à un dépôt distant et le copier en local :
git clone https://github.com/Sana-El-Ath/LGC_DEV3_G1.git
- Une pull request(demande de pull) est une fonctionnalité de GitHub qui permet de demander aux propriétaires d'un repository l'autorisation de fusionner les changements sur la branche principale ou toute autre branche sur laquelle on souhaite apporter les modifications.
- Pour créer une pull request, il faut tout d'abord:
 1. Créé une nouvelle branche.
 2. Envoyé votre code sur cette même branche.

 pullR had recent pushes less than a minute ago

Compare & pull request



Correction d'erreur - stash

- ❑ **Objectif** : Effectuer des modifications sur une nouvelle branche
- ❑ **Problème** : Vous avez modifié la branche principale avant de créer une branche et que vous n'avez pas fait le commit
- ❑ **Solution** : Appliquer une remise (stash)

Correction d'erreur - stash

Stash permet de mettre les modifications de côté, le temps de créer une nouvelle branche et d'appliquer cette remise sur la nouvelle branche en utilisant la commande suivante : `git stash`

Récupérer le dernier stash

git stash apply : récupérer les modifications que vous avez rangées dans le dernier stash et les appliquer sur la nouvelle branche

Récupérer un stash particulier :

git stash list : la liste des stash

git stash apply stash@{0}

Correction d'erreur - log

- ❑ **Objectif** : Effectuer des modifications sur la branche master sans passer le commit
- ❑ **Problème** : vous avez passer le commit
- ❑ **Solution** : Analyser les derniers commits avec git log

Correction d'erreur - reset

- **git log** : lister les derniers commits
- **git reset --hard HEAD** : supprimer de la branche principale le dernier commit.
- **git branch brancheCommit** : créer une nouvelle branche
- **git checkout brancheCommit** : basculer sur cette branche
- **git reset --hard ca83a6df** : appliquer ce commit sur la nouvelle branche

Correction d'erreur - commit

- ❑ **Objectif** : Marquer les modifications effectuées dans le message descriptif du commit
- ❑ **Problème** : Message de commit erroné
- ❑ **Solution** : `git commit --amend -m "Votre nouveau message de commit"`

Merci