Tout sur git

Commandes Git de base

Pour utiliser Git, les développeurs utilisent des commandes spécifiques pour copier, créer, modifier et combiner du code. Ces commandes peuvent être exécutées directement à partir de la ligne de commande ou à l'aide d'une application comme GitHub Desktop. Voici quelques commandes courantes pour l'utilisation de Git :

- `git init` initialise un tout nouveau dépôt Git et commence à suivre un répertoire existant. Elle ajoute un sous-dossier masqué dans le répertoire existant qui héberge la structure de données interne requise pour la gestion de versions.
- `git clone` crée une copie locale d'un projet qui existe déjà à distance. Le clone inclut tous les fichiers, l'historique et les branches du projet.
- `git add` indexe un changement. Git effectue le suivi des modifications apportées au codebase d'un développeur, mais il est nécessaire d'indexer et de prendre un instantané des modifications pour les inclure dans l'historique du projet. Cette commande effectue l'indexation, la première partie de ce processus en deux étapes. Toutes les modifications qui sont indexées feront partie de l'instantané suivant et de l'historique du projet. L'indexation et le commit donnent séparément aux développeurs un contrôle complet sur l'historique de leur projet sans modifier la façon dont ils codent et travaillent.
- `git commit` enregistre l'instantané dans l'historique du projet et termine le processus de suivi des modifications. En bref, un commit fonctionne comme la prise d'une photo. Tout ce qui a été indexé avec `git add` fera partie de l'instantané avec `git commit`.
- `git status` affiche l'état des modifications comme non suivies, modifiées ou indexées.
- `git branch` montre les branches en cours de traitement localement.
- `git merge` fusionne les lignes de développement. Cette commande est généralement utilisée pour combiner les modifications apportées sur deux branches distinctes. Par exemple, un développeur fusionne quand il souhaite combiner les modifications d'une branche de fonctionnalité dans la branche principale pour le déploiement.
- `git pull` met à jour la ligne de développement locale avec les mises à jour de son équivalent distant. Les développeurs utilisent cette commande si un collègue a effectué des commits sur une branche d'un dépôt distant et qu'ils souhaitent refléter ces modifications dans leur environnement local.
- `git push` met à jour le dépôt distant avec les commits effectués localement sur une branche.

Pour plus d'informations, consultez le guide de référence complet des commandes Git.

Exemple : Contribuer à un dépôt existant

```
# download a repository on GitHub to our machine
# Replace `owner/repo` with the owner and name of the repository to clone
git clone https://github.com/owner/repo.git

# change into the `repo` directory
cd repo

# create a new branch to store any new changes
git branch my-branch

# switch to that branch (line of development)
git checkout my-branch

# make changes, for example, edit `file1.md` and `file2.md` using the text editor

# stage the changed files
git add file1.md file2.md

# take a snapshot of the staging area (anything that's been added)
git commit -m "my snapshot"

# push changes to github
git push --set-upstream origin my-branch
```

Exemple : Démarrer un nouveau dépôt et le publier sur GitHub

Tout d'abord, vous devez créer un dépôt sur GitHub. Pour plus d'informations, consultez « Hello World ». **N'initialisez pas** le dépôt avec un fichier Lisez-moi, .gitignore ou de licence. Ce dépôt vide attend votre code.

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"

# provide the path for the repository you created on github
git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY-NAME.git
```

```
# push changes to github
git push --set-upstream origin main
```

Exemple : Contribuer à une branche existante sur GitHub

Cet exemple suppose que vous avez déjà un projet appelé `repo` sur l'ordinateur et qu'une nouvelle branche a été poussée vers GitHub depuis les dernières modifications locales.

```
# change into the `repo` directory
cd repo

# update all remote tracking branches, and the currently checked out branch
git pull

# change into the existing branch called `feature-a`
git checkout feature-a

# make changes, for example, edit `file1.md` using the text editor

# stage the changed file
git add file1.md

# take a snapshot of the staging area
git commit -m "edit file1"

# push changes to github
git push
```