Initiation à Git & Github

Chapitre 1 - Les commandes de bases

Section 1 - Hello Git

Git est un logiciel qui vous permet de garder la trace des modifications apportées à un projet au fil du temps. Git fonctionne en enregistrant les modifications que vous apportez à un projet, en stockant ces modifications, puis en vous permettant de les référencer si nécessaire.

Nous allons apprendre Git en l'utilisant. Nous allons nous en servir pour nous aider à écrire un scénario intitulé Harry Programmer and the Sorcerer's Code.

Nous allons commencer par jeter un coup d'œil au projet de scénario.

Dans scene-1.txt, ajoutez ce texte :

```
Harry Programmeur et le Code du Sorcier : Scène 1
```

Section 2 - Git init

Maintenant que nous avons commencé à travailler sur le scénario, transformons le répertoire sorcerers-code en un projet Git.

Nous faisons cela avec les commandes suivantes :

```
git init
```

Le mot init signifie initialiser.

La commande met en place tous les outils dont Git a besoin pour commencer à suivre les modifications apportées au projet.

Instructions

Dans le terminal, initialisez un nouveau projet Git.

Remarquez la sortie :

```
Initalized empty Git repository in
/home/ccuser/workspace/sorcerers-code/.git/
```

Section 3 - Git Workflow

Joli ! Nous avons un projet Git. Un projet Git peut être considéré comme ayant trois parties :

1. **Un répertoire de travail** : où vous ferez tout le travail : création, édition, suppression et organisation des fichiers.
2. **Une zone de transit** : où vous répertoriez les modifications apportées au répertoire de travail.
3. **Un référentiel** : où Git stocke de manière permanente ces modifications sous forme de différentes versions du projet.

Le flux de travail de Git consiste à éditer des fichiers dans le répertoire de travail, à ajouter des fichiers à la zone de transit et à enregistrer les modifications dans un dépôt Git.

Dans Git, nous enregistrons les modifications avec un commit, que nous allons découvrir dans cette leçon.

Section 4 - Git Status

Au fur et à mesure que vous écrivez le scénario, vous allez modifier le contenu du répertoire de travail.

Vous pouvez vérifier le statut de ces changements avec :

```
git status
```

Instructions

Depuis le terminal, vérifiez le statut du projet sorcerers-code.

Dans la sortie, remarquez le fichier en rouge sous les fichiers non suivis.

Untracked files signifie que Git voit le fichier, mais n'a pas encore commencé à suivre les changements.

Section 5 - Git Add

Pour que Git commence à suivre scene-1.txt, le fichier doit être ajouté à la zone de transit.

Nous pouvons ajouter un fichier à la zone de transit avec :

```
git add filename
```

Le mot "filename" fait ici référence au nom du fichier que vous éditez tel que scene-1.txt.

Instructions

1. Ajoutez scene-1.txt à la zone de transit dans Git. Rappelez-vous que vous devrez identifier le fichier par son nom
2. Vérifier l'état du projet

Dans la sortie, remarquez que Git indique les changements à commettre avec "nouveau fichier : scene-1.txt" en texte vert.

Git nous indique ici que le fichier a été ajouté à la zone de transit.

Section 6 - Git Diff

Bon travail ! Vous savez maintenant comment ajouter un fichier à la zone de transit.

Imaginons que nous tapions une autre ligne dans scene-1.txt.

Puisque le fichier est suivi, nous pouvons vérifier les différences entre le répertoire de travail et la zone de transit avec :

```
git diff filename
```

Ici, le nom du fichier est le nom réel du fichier. Si le nom de mon fichier était changes.txt, la commande serait la suivante :

```
git diff change.txt
```

Instruction

1. Dans l'éditeur de code, ajoutez ce texte à scene-1.txt :

```
Dumblediff : J'aurais dû savoir que vous seriez là, Professeur McGonagit.
```

Cliquez sur Exécuter.

2. Dans le terminal, utilisez la nouvelle commande pour vérifier la différence entre le répertoire de travail et la zone de transit.

Remarquez la sortie :

"Harry Programmer et le code du sorcier : Scene 1" se trouve dans la zone de transit, comme indiqué en blanc.

Les modifications apportées au fichier sont marquées d'un + et sont indiquées en vert.

3. Ajoutez les modifications à la zone de transit dans Git. Rappelez-vous que vous devrez identifier le fichier par son nom.

Section 7 - Git Commit

Un commit est la dernière étape de notre workflow Git. Un commit stocke de façon permanente les changements de la zone de transit dans le dépôt.

Git commit est la commande que nous allons utiliser ensuite. Cependant, un bout de code supplémentaire est nécessaire pour un commit : l'option -m suivie d'un message.

Voici un exemple :

```
git commit -m "Complete first line of dialogue"
```

Conventions standard pour les messages d'engagement :

- Ils doivent être entre guillemets
- Ils sont écrits au présent
- Ils doivent être brefs (50 caractères ou moins) lorsqu'on utilise -m

```
git diff change.txt
```

`Instruction`

1. Faites votre premier commit ! Dans le terminal, tapez la commande avec un message de validation. Le message doit décrire le but de la livraison.

Si vous avez du mal à trouver un bon message de validation, réfléchissez à la façon dont le projet a changé depuis son lancement.

Section 8 - Git Log

Souvent avec Git, vous aurez besoin de vous référer à une version antérieure d'un projet.

Les commits sont stockés chronologiquement dans le dépôt et peuvent être visualisés avec :

```
git log
```

`Instruction`

Depuis le terminal, enregistrez une liste de vos commits.

Dans la sortie, remarquez :

- Un code de 40 caractères, appelé SHA, qui identifie de manière unique le commit. Il apparaît en texte orange.
- L'auteur du commit (vous !)
- La date et l'heure de la livraison
- Le message de la livraison

Section 9 - Généralisation

Vous avez maintenant été introduit au flux de travail fondamental de Git.

Vous avez beaucoup appris ! Prenons un moment pour généraliser :

Git est le système de contrôle de version standard de l'industrie pour les développeurs web.

Utilisez les commandes Git pour garder la trace des modifications apportées à un projet :

- git init crée un nouveau dépôt Git.
- git status inspecte le contenu du répertoire de travail et de la zone de transit.
- git add ajoute des fichiers du répertoire de travail à la zone de transit.
- git diff montre la différence entre le répertoire de travail et la zone de transit.
- git commit stocke définitivement les changements de fichiers de la zone de transit dans le référentiel.
- git log montre une liste de tous les commits précédents.