

CORBEILLE HISTORIQUE GIT

Tout afficher + Tout réduire -

SUJET

✓ Énoncé

EXERCICE : CORBEILLE HISTORIQUE GIT

Suite à vos travaux réalisés lors de la séquence de préparation, votre environnement de gestion de version Git est opérationnel.

Vous êtes prêt à démarrer les premières manipulations de vos fichiers tout en maîtrisant leurs différentes versions ou historiques.

Dans la suite de cette corbeille, nous vous invitons à utiliser le mode commande (CLI) pour toutes les manipulations de fichiers qui vous sont proposées.

Ici nous allons nous intéresser à un ensemble de recettes de sandwiches que nous allons faire évoluer tout en maîtrisant les historiques de chacune des différentes recettes.


Question

1. Créons une première version d'un fichier

- Créez un dépôt vide `mesSandwichs` `git init mesSandwichs`
- Cette commande initialise un dépôt Git dans le répertoire `C:/users/«username»/mesSandwichs`
- Ce répertoire définit alors un espace de travail et un dépôt Git dans `.../mesSandwichs/.git`
- Créez le fichier `burger.txt` avec les ingrédients `steak, salade, tomate, cornichon` et `fromage` (un ingrédient par ligne) dans l'espace de travail du dépôt `mesSandwichs`.
- Le fichier texte `burger.txt` est créé dans le répertoire `C:/users/«username»/mesSandwichs`
- Vérifiez l'état dans lequel se trouve votre dépôt. Commentez.
- En mode commande, `git status`
- Ajoutez `burger.txt` à votre projet.
- En mode commande, `git add burger.txt`
- Vérifiez l'état dans lequel se trouve votre dépôt. Commentez.
- Déclarez une première version du fichier `burger.txt` dans votre projet. Quel est le vocabulaire Git utilisé pour ce type d'action ?
- En mode commande, `git commit -m « livraison de la première version de burger.txt »`
- Vérifiez l'état dans lequel se trouve votre dépôt. Commentez.
- En mode commande, `git status`
- Affichez la liste des changements et historiques correspondant aux différentes actions effectuées dans votre dépôt Git. Commentez.
- En mode commande, `git log`

2. Manipulons plusieurs fichiers et faisons des modifications

- Créez quelques autres sandwiches `hotdog.txt`, `jambonbeurre.txt`, ... et modifiez des sandwiches déjà créés en changeant leur liste d'ingrédients.
- Effectuez alors plusieurs livraisons successives où chaque livraison doit contenir une et une seule création de fichier et une et une seule modification de fichier.
- Analysez les différences entre les différents états de votre dépôt au fur et à mesure de vos livraisons.
- Analysez les historiques de votre dépôt au fur et à mesure de vos livraisons.
- Commentez.

- Sur la base de vos différentes manipulations sur les fichiers de vos sandwiches, familiariser vous avec les différentes commandes de git telles de celles-ci sont listées sous <https://git-scm.com/docs/git/2.9.4> .

Question

Tout le travail effectué jusqu'à présent a été réalisé dans une et une seule branche, la branche principale.

Nous allons maintenant nous concentrer sur la notion de branche qui permet, dans un projet ... ou la gestion d'une liste de sandwiches ... d'introduire des modifications (nouvelle fonctionnalité, correction d'anomalie, ...). Ceci dans un premier temps, sans toucher à la version officielle stable ... ou votre liste de sandwiches officielle !

On propose de changer la liste des ingrédients du sandwich burger pour expérimenter une nouvelle recette pour ce sandwich. Une fois la recette « validée » par les clients, celle-ci sera intégrée dans la liste officielle des sandwiches.

Avant toute évolution dans la recette du sandwich burger.txt, on vérifiera bien que l'on se trouve dans la branche principale. Ensuite, au fur et à mesure des manipulations, on gardera bien à l'esprit de toujours savoir dans quelle branche on se trouve.

A partir de la branche principale, créez une nouvelle branche `nouvelrecetteburger`

```
1 git branch nouvelrecetteburger
```

Changez la copie ou branche de travail pour vous trouver sur la branche `nouvelrecetteburger`

Commentez

```
1 git checkout nouvelrecetteburger
2 git status
3 git log
```

Dans cette branche, corrigez le contenu du fichier `burger.txt`.

Faites toutes les manipulations nécessaires pour que le nouveau fichier `burger.txt` puisse faire partie de la branche `nouvelrecetteburger`.

Commentez

```
1 git add burger.txt
2 git commit -m « nouvelle recette burger »
3 git status
4 git log
```

Retournez dans la branche principale et livrez votre nouvelle recette du sandwich burger dans la branche principale.

```
1 git checkout master
2 git merge « livraison nouvelle recette du sandwich burger »
3 git log
```

Question

Jusqu'à présent nous nous sommes placés dans des scénarii dans lesquels les livraisons dans la branche principale sont simples, c'est-à-dire qu'il n'y a pas de conflits avec deux fichiers qui n'ont pas été modifiés de la même manière dans deux branches différentes. Par exemple, la nouvelle recette du sandwich burger proposée dans la branche `nouvelrecetteburger` n'est pas la même que celle qui a été proposée dans une autre branche.

Implémentez un scénario de conflit entre une nouvelle recette du sandwich burger proposé par Pierre et une autre nouvelle recette pour ce même sandwich burger proposé par Jacques.

Créez les branches Pierre et Jacques, effectuez les modifications pour chaque nouvelle recette dans la branche qui va bien, faites les livraisons dans la branche principale.

Commentez

A NOTER : Nommer des branches par rapport à des noms de personnes n'est pas une pratique courante et recommandé. En effet, dans la pratique, une branche reçoit le nom de la version pour laquelle elle a été créée pour permettre à l'équipe de développement d'en assurer la version.

Question

Vous aurez noté que jusqu'à présent nous nous sommes placés dans des contextes de démarrage projet « from scratch ».

Pour aller plus loin :

- Explorez les possibilités de git et notamment la récupération de projet existants.

- Rejouez les manipulations (effectuées jusqu'à présent en mode commandes) en mode graphique : Utiliser Git GUI ou Tortoise Git (recommandé).

