Gérez du code avec Git et GitHub

6 heures Facile

Mis à jour le 19/01/2024

Corriger les erreurs courantes

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

Corriger les erreurs courantes sur GitHub

Question 1

Je suis sur ma Branch1 et je viens d'ajouter un fichier "File2.txt".
Je change d'avis : je veux finalement ajouter mon fichier à une nouvelle branche, "BranchFile". Que dois-je faire ?

- ```
  git status
  git branch BranchFile
  git checkout BranchFile
  git status apply
  ```

- ```
  git status
  git stash
  git branch BranchFile
  ```

```
git checkout BranchFile
git status apply
```

✓ ◉

```
git status
git stash
git branch BranchFile
git checkout BranchFile
git stash apply
```

Nous abordons ici la remise. Nous allons devoir remiser les changements faits sur Branch1 pour les appliquer ensuite sur la nouvelle branche, "BranchFile". `git stash` *permet de remiser des modifications et* `git stash apply` *permet d'appliquer une remise sur une branche. Vous allez devoir remiser vos modifications de la branch1 à la branchFile en utilisant plusieurs commandes dans un ordre précis :*

- git status pour vérifier l'état de vos fichier

- git stash pour remiser vos modifications

- git branch branchFile pour créer une nouvelle branche

- git checkout BranchFile pour basculer sur cette branche

- git stash apply pour appliquer les modifications

Question 2

`git commit --amend -m "Test"`

Cette commande permet de :

- ○ revenir au commit précédent
- ✓ ◉ modifier le message du commit précédent
- ○ créer un nouveau commit avec le message "Test"

`git commit --amend -m` *modifie le message du commit précédent. C'est le -m qui indique que nous allons modifier le message.*

Question 3

Je souhaite revenir sur le commit précédent et modifier son contenu en ajoutant un fichier, que dois-je faire ?

- ```
  git commit --amend --no edit
  git add monFichierOublié
  git commit
  ```

- ✓ ```
  git add monFichierOublié
  git commit --amend --no-edit
  ```

- ```
  git commit --amend
  git add monFichierOublié
  git commit
  ```

- ```
  git add monFichierOublié
  git commit --amend
  ```

- `git add monFichierOublié` *permet d'ajouter mon fichier.*
- `git commit --amend --no-edit` *permet de modifier le commit sans changer le message.*

Question 4

La commande git reset peut être appliquée de 3 façons différentes. Lesquelles ?

- ✓ `git reset --soft` / `git reset --mixed` / `git reset --hard`
- `git reset --down` / `git reset --middle` / `git reset --up`
- `git reset --safe` / `git reset --regular` / `git reset --risky`
- git reset n'existe pas.

`git reset` *peut être appelée de 3 façons différentes, qui correspondent aux arguments de*

ligne de commande --soft, --mixed et --hard.

Question 5

Quelle est la différence entre `git revert` et `git reset` ?

- ○ `git revert` ne réinitialise qu'un commit alors que `git reset` réinitialise tout.
- ✓ ◉ `git revert` crée un nouveau commit alors que `git reset` , non.
- ○ `git reset` crée un nouveau commit alors que `git revert` , non.

`git reset` *ne crée jamais un nouveau commit alors que* `git revert` *, oui.*

Question 6

Je souhaite savoir qui a touché à une ligne en particulier dans le fichier test.html. Quelle commande vais-je devoir exécuter ?

- ○ `git log`
- ○ `git reflog`
- ✓ ◉ `git blame`
- ○ `git cherry pick`

Seule la commande git blame permet ***d'examiner le contenu d'un fichier ligne par ligne*** *et de déterminer la date à laquelle chaque ligne a été modifiée, et le nom de l'auteur des modifications .*

Question 7

Vous avez deux branches dans votre projet : branch1 et branch2.

Sur branch1, dans le fichier “monfichier”, ligne 10, nous avons “titre”.
Sur branch2, dans le fichier “monfichier”, ligne 10, nous avons “titre !”.

Vous avez voulu fusionner ces deux branches et cela a provoqué un conflit. Que devez-vous faire ?

- ✓ (●) Résoudre ce conflit
- ( ) Forcer la fusion
- ( ) Supprimer une branche
- ( ) Demander à Git de faire un choix

Dans cette situation, il faut résoudre le conflit. Nous avons 2 versions différentes pour la même ligne de code, la seule solution possible est donc de faire un choix et de décider laquelle garder.

Question 8

Qu'est ce qu'un SHA ?

- ( ) Un SHA est un numéro aléatoire permettant de se connecter à GitHub
- ✓ (●) Un SHA est un identifiant pour les commits et autres actions gardés en mémoire par Git
- ( ) Un SHA est un petit animal qui ronronne

Le SHA, c'est ce grand code qui vous permettra de revenir à un commit exact. C'est l'identifiant de votre action !

Allez plus loin dans votre utilisation de

Git et GitHub

Les professeurs

Luc Bourrat

Mentor et développeur web

Tiffany Lestroubac

Développeur Fullstack / Mentor / Evaluatrice / Rédactrice

Kassandre Pedro

Backend Engineer at Pexels | Ruby on Rails Developer

Mila Paul

I am a researcher and teacher with a PhD in Cyber Operations. I also love trekking, scuba diving and traveling!

POUR LES EMPLOYEURS

Solutions de formations et recrutement

Recruter en alternance

Développer les connaissances

Booster les compétences

Blog employeurs

OPENCLASSROOMS

Qui sommes-nous ?

Nous rejoindre

Devenir mentor

Devenir coach carrière

Boutique

AIDE

FAQ étudiants

FAQ employeurs

OPENCLASSROOMS