Devenez un expert de Git et GitHub

6 heures Moyenne

Mis à jour le 27/06/2022

Collaborer efficacement grâce à Git et GitHub

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

Collaborer efficacement grâce à Git et GitHub

Question 1

Vous voulez contribuer à un projet open source : Open Bureautique. Vous allez donc devoir forker le projet. Quel sera le résultat de cette opération ?

- Le projet sera cloné sur votre poste de travail.
- Le projet sera dupliqué sur le dépôt de Open Bureautique.
- ✓ Le projet sera copié sur votre dépôt.

L'opération de fork copie le projet ciblé sur le dépôt de celui qui exécute l'opération.

Question 2

Parmi les situations suivantes sur le projet Open Bureautique, lesquelles peuvent être à l'origine d'un conflit ?

Attention, plusieurs réponses sont possibles.

- [ ] Vous avez corrigé une erreur sur le fichier index.html et commité cette mise à jour puis fait un **git push**.
- [x] Vous avez travaillé sur l'issue n° 13, ce qui a amené à créer différents commits. Vous faites un **git pull**.
- [ ] Lors de votre dernier commit vous avez oublié d'ajouter un fichier à l'index. Vous exécutez la commande **git commit --amend**.
- [x] Vous travaillez sur 2 branches différentes, vous faites un **git merge** de la branche 2 vers la branche 1.

Un conflit est déclenché lorsque Git réalise une opération de merge. La commande git merge entre 2 branches peut donc bien évidemment être à l'origine d'un conflit. La commande git pull va déclencher un merge et peut donc aussi créer un conflit.

En revanche, la commande git push peut être rejetée mais ne déclenche pas de conflit. La commande git commit --amend permet de corriger le dernier commit.

Question 3

Vous exécutez la commande git status afin de résoudre ce conflit. Quelle conclusion tirez-vous du résultat suivant ?

```
$ git status
On branch feature
Your branch and 'origin/feature' have diverged,
and have 1 and 1 different commits each, respectively.
 (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
 (fix conflicts and run "git commit")
 (use "git merge --abort" to abort the merge)

Unmerged paths:
 (use "git add <file>..." to mark resolution)
    both modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

- ( ) Le conflit sur la branche main a déjà été corrigé, d'où la mention "modified".
- ( ) La commande git merge --abort stoppera le merge et résoudra le conflit.
- (x) Le conflit sur la branche feature concerne le fichier index.html. Il n'a pas encore été résolu.

Le texte “both modified: index.html” signifie que le fichier index.html a été modifié par 2 sources distinctes. Ainsi, il existe désormais un conflit qui n’est pas encore résolu.

Le conflit n’est pas corrigé ; preuve en est, la sortie console nous encourage à le faire avec (fix conflicts and run “git commit”) ou bien à annuler le merge qui est à l’origine du conflit avec (use “git merge --abort” to abort the merge).

La commande git merge --abort permettra d’annuler l’opération de merge mais elle ne résout pas le conflit, elle revient simplement en arrière.

Question 4

Suite au conflit précédent, vous exécutez la commande *git checkout --ours index.html*. Quel est son résultat ?

- ✓ (●) La commande réussit, et la version locale du fichier index.html est conservée.
- ( ) La commande échoue, git checkout sert à changer de branche.
- ( ) La commande réussit, et la version distante du fichier index.html est conservée.

git checkout --ours [fichier] permet de conserver la version locale du fichier.

Question 5

Plusieurs collaborateurs travaillent en ce moment sur Open Bureautique. De ce fait, vous créez une branche nommée issue-3, et travaillez sur cette dernière. Quel sera le résultat de la suite de commandes git checkout issue-3 puis git rebase main ?

- ✓ (●) Les commits de la branche issue-3 sont placés à la pointe de la branche main.
- ( ) Les commits de la branche main sont placés à la pointe de la branche issue-3.
- ( ) La commande échoue car vous n’êtes pas positionné sur la branche main.

git checkout issue-3 nous positionne sur la branche issue-3. git rebase main permet de rebaser la branche issue-3 en la plaçant à la pointe de la branche main.

Question 6

Quel est le résultat de l’exécution de la commande git commit -m “update index.html” à la fin du scénario suivant ?

```
$ git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Le fichier index.html est modifié.

$ git add index.html

$ git stash
Saved working directory and index state WIP on main: 8ac2dd4 first commit
```

- ○
  ```
  $ git commit -m “update index.html”
  [main 93ea0d5] u
  1 file changed, 1 insertion(+), 1 deletion(-)
  ```

- ✓ ◉
  ```
  $ git commit -m “update index.html”
  On branch main
  Your branch is up to date with ‘origin/main’.

  nothing to commit, working tree clean
  ```

- ○
  ```
  $ git commit -m “update index.html”
  Everything up-to-date
  ```

La commande git stash met de côté le travail en cours, ainsi il n’y a rien dans l’index éligible au commit. Le message “nothing to commit, working tree clean” sera affiché.

Question 7

La commande git revert [numéro de commit] va :

- ○ supprimer le commit concerné de l’historique de commits.
- ○ dupliquer la branche du commit concerné sans y copier ce dernier.
- ✓ ◉ créer un nouveau commit qui contient l’inverse du commit concerné, pour annuler son effet à la pointe de la branche.
- ○ créer un nouveau commit qui contient l’inverse du commit, juste après le commit concerné et avant le commit suivant.

git revert est une commande type “undo”. Elle sert à annuler l’effet d’un commit. Pour atteindre cet

objectif, elle crée un nouveau commit à la pointe de la branche.

Question 8

Quel est le résultat du scénario suivant ?

- 3 fichiers sont modifiés : index.html, style.css et script.js.
- git add index.html, style.css.
- git commit -m “Mis à jour de la page d’accueil du site web de Open Bureautique”.
- git add script.js.
- git commit --amend --no-edit.

- ○ L’opération de commit échoue.
- ○ Un nouveau commit est ajouté à l’historique de commits, avec les modifications du fichier script.js.
- ✓ ◉ Le dernier commit est modifié, et intègre les modifications du fichier script.js en plus des modifications des fichiers index.html et style.css.

La commande git commit --amend --no-edit permet de modifier le dernier commit. Dans notre situation, cela ajoutera donc les modifications apportées au fichier script.js qui été ajouté à l’index après le premier commit.

‹ Corrigez l’historique du projet au fil de vos développements

Structurez la collaboration grâce à GitFlow ›

Le professeur

Romain Sessa

Développeur et Architecte Java/JavaEE. Je suis aussi Enseignant et Mentor étant passionné par la transmission de connaissances/compétences.

AIDE

FAQ étudiants

FAQ employeurs

LANGUE

Français

NOUS SUIVRE

Télécharger dans l'App Store

OPENCLASSROOMS

Entreprise

Cette entreprise respecte des normes sociales et environnementales élevées.

Certifiée

Mentions légales Conditions générales d'utilisation Politique de protection des données personnelles

Cookies Accessibilité