Devenez un expert de Git et GitHub

6 heures Moyenne

Mis à jour le 27/06/2022

Personnaliser votre usage de Git et GitHub en fonction de vos besoins

Bravo ! Vous avez réussi cet exercice !

Compétences évaluées

Personnaliser votre usage de Git et GitHub en fonction de vos besoins

Question 1

Vous voilà à la tête d'un nouveau projet open source qui implique des collaborateurs éparpillés aux 4 coins du monde. C'est un projet d'envergure où la précipitation n'est pas souhaitée.

Vous devez mettre en place des workflows Git pour faciliter la collaboration sur le projet. Lesquels choisissez-vous ?

Attention, plusieurs réponses sont possibles.

- [ ] Le workflow centralisé
- [x] Le workflow basé sur les branches de fonctionnalités
- [x] Le workflow GitFlow

- ✗ ☑ Le développement basé sur le tronc
- ✓ ☑ Le workflow de duplication

Au vu de la dimension importante du projet, il se peut que les collaborateurs aient des compétences très variées. Il est donc préférable d'avoir un contrôle strict de leurs contributions. Le workflow GitFlow est le plus approprié dans ce cadre. De même, le contexte de l'open source implique de mettre en place le workflow de duplication pour éviter qu'un collaborateur ne crée de problèmes irrémédiables sur le code originel du projet.

Le workflow centralisé et le développement basé sur le tronc sont à écarter, car la branche principale risque rapidement d'être ingérable au vu du nombre de collaborateurs simultanés.
Le workflow basé sur les branches de fonctionnalités aurait pu être un bon candidat, mais il est moins structuré que le workflow GitFlow.

Question 2

Matt, un de vos collaborateurs, suggère de mettre en place un workflow Git pour accélérer l'intégration des nouvelles fonctionnalités, et donc la mise à disposition des nouvelles versions du produit. Quel workflow suggérez-vous dans ce cas ?

- ○ Le workflow centralisé
- ○ Le workflow basé sur les branches de fonctionnalités
- ○ Le workflow GitFlow
- ✓ ◉ Le développement basé sur le tronc
- ○ Le workflow de duplication

Le développement basé sur le tronc (Trunk Based Development) a pour caractéristique de réduire les lenteurs induites par de nombreuses branches à gérer et à intégrer. On atteint ce résultat grâce à l'intégration rapide des nouveaux développements sur la branche principale (trunk). Les itérations sur le trunk sont donc bien plus fréquentes qu'avec un workflow comme GitFlow.

Question 3

Vous commencez à utiliser GitFlow pour votre projet. Vous initialisez le repository avec la commande git flow init. Quel sera le résultat de la commande git status ?

- ○

```
$ git status
On branch master
nothing to commit, working tree clean
```

✓ ◉
```
$ git status
On branch develop
nothing to commit, working tree clean
```

○
```
$ git status
On branch feature
nothing to commit, working tree clean
```

Lors de l'initialisation du repository, GitFlow crée 2 branches : la branche principale (master) et la branche pour l'intégration des fonctionnalités (develop). Ensuite, il nous positionne sur la branche develop.

Question 4

Alors que vous venez d'initialiser votre projet avec git flow init, vous exécutez le scénario suivant avec un terminal bash :

```
$ git flow feature start issue1
$ echo “Bienvenue !” > readme.md
$ git add readme.md
$ git commit -m “#1 Creation du fichier readme.md”
$ git flow feature finish issue1
```

Quel sera le résultat ?

- ○ Le scénario échoue dès la première commande. Il aurait fallu faire un **git checkout issue1** avant **git flow feature start issue1**.
- ○ Le scénario échoue lors de la dernière commande (git flow feature finish issue1) car avant cela, il fallait faire un **git checkout develop** et un **git merge issue1**.
- ○ La branche main contient la nouvelle fonctionnalité. La branche issue1 a été supprimée.
- ✓ ◉ La branche develop contient la nouvelle fonctionnalité. La branche issue1 a été supprimée.

*Ce scénario ne contient pas d'erreur et il n'échoue pas. L'exécution de la commande **git flow feature finish issue1** déclenche un merge sur la branche develop du contenu de la branche issue1. Ensuite, la branche issue1 est supprimée.*

Question 5

Vous travaillez avec le workflow GitFlow, et de nouvelles fonctionnalités ont déjà

été intégrées dans la branche develop. Vous exécutez le scénario suivant :

```
$ git flow release start 0.0.1
$ git flow release finish '0.0.1'
```

Quelles commandes Git ont été exécutées ?

- ○
```
git checkout main
git merge develop
```

- ✓ ◉
```
git checkout develop
git checkout -b release/0.0.1
git checkout main
git merge release/0.0.1
git branch -D release/0.0.1
```

- ○
```
git checkout develop
git merge release/0.0.1
git checkout main
git merge release/0.0.1
```

*La commande **git flow release start 0.0.1** déclenche la création d'une nouvelle branche à partir de la branche develop. Les commandes sont donc :*

```
git checkout develop
git checkout -b release/0.0.1
```

*La commande **git flow release finish '0.0.1'** déclenche le merge de la branche release/0.0.1 sur la branche main, puis supprime la branche de release. Les commandes sont donc :*

```
git checkout main
git merge release/0.0.1
git branch -D release/0.0.1
```

Question 6

Vous décidez d'augmenter les contrôles sur les opérations de commit réalisées par vos collaborateurs. Vous fournissez à chacun d'eux le fichier prepare-commit-msg.sample. Quelle procédure fournissez-vous à vos collaborateurs pour la mise en fonction du hook ?

- ○ Copier le fichier dans le répertoire .git

- Copier le fichier dans le répertoire .git et le renommer en prepare-commit-msg
- Copier le fichier dans le répertoire .git/hooks
- ✓ Copier le fichier dans le répertoire .git/hooks et le renommer en prepare-commit-msg

Le répertoire dédié aux hooks par défaut est .git/hooks. Néanmoins, il faut retirer l'extension .sample pour utiliser le hook.

Question 7

Vous voulez empêcher l'envoi au serveur d'un push de commits qui ne respectent pas un critère défini côté client. Quel hook choisissez-vous ?

- post-commit
- ✓ pre-push
- pre-receive

C'est le hook pre-push qui peut empêcher l'opération de push.

Le hook post-commit sera exécuté après une exécution valide de git commit, mais ne peut pas bloquer l'opération de push.

Le hook pre-receive est quant à lui exécuté côté serveur lorsqu'une opération de git push est reçue.

Question 8

Le hook post-checkout suivant est en place sur votre repository Git :

```
#!/bin/sh

if [[ "$3" == "1" ]]; then
        echo "Changement de branche"
fi
```

Sachant que la documentation indique : *"The hook is given three parameters: the ref of the previous HEAD, the ref of the new HEAD (which may or may not have changed), and a flag indicating whether the checkout was a branch checkout (changing branches, flag=1) or a file checkout (retrieving a file from the index, flag=0)."*

Que se passera-t-il lors de l'exécution de la commande git checkout main ?

- ✓ (●) La console affichera le message Changement de branche en plus de la sortie console habituelle.
- ( ) La console affichera la sortie console habituelle, le hook n'a pas d'effet.
- ( ) La console affichera une erreur car le paramètre $3 n'existe pas.
- ( ) L'opération de checkout sera annulée, et la console affichera Changement de branche.

*Comme le révèle la documentation, le troisième paramètre est un flag indiquant s'il s'agit d'un checkout de branche ou de fichier. Ici, la commande **git checkout main** correspond à un changement de branche, le paramètre aura donc la valeur **1** et le script affichera alors le texte **Changement de branche**.*

À noter que ce hook est un "post" hook, il ne peut donc pas annuler l'opération ciblée (ici le checkout).

‹ Automatisez des traitements sur vos projets Git grâce aux hooks

Prenez en main un client GUI Git ›

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