GIT Collaboratif

Utiliser Git efficacement en équipe

Créer un dépôt commun

A partir de github, ou Gitlab, ou Bitbucket, créer deux dépôts :

- frontend,
- backend

Le Dictateur déposera la branche “main” et poussera une branche “devel” (à ce stade identique à la branche “main”)

Inviter les collaborateurs sur chacun des dépôts

Cette opération doit être réalisée par un seul et unique membre de l’équipe, le “Dictateur”, les autres seront les “Lieutenants”.

Pensez à “protéger” la branche “main” contre les fusions inopinées, la branche “main” doit être stable à chaque instant.

Pour les développeurs

Acceptez l’invitation du Dictateur à collaborer.

Clônez les dépôt localement :

```
git clone <lien-vers-le-depot-git>
```

Une fois le clône réalisé localement, il faudra récupérer la branche de développement :

```
git fetch origin devel:devel
```

En tant que collaborateur

Initialiser un “git flow” :

```
git flow init
```

Choisissez “main” pour la branche contenant la version de production,

Choisissez “devel” comme branche de prochaine “release”

Au quotidien

Vous allez traiter une US, vous devez donc créer une branche de fonctionnalité, si vous utilisez “git flow” :

```
git flow feature start myNewFeature
```

Si vous n’utilisez pas “git flow” :

```
git checkout -b myNewFeature devel
```

Au quotidien

Vous allez commiter fréquemment, ne commitez que les parties de fonctionnalités abouties et veillez à libeller correctement vos messages de commit.

Choisissez les fichiers que vous voulez commiter, par exemple, éviter de faire un commit avec des fichiers “package.json, package-lock.json” et des composants ou des templates. Préférez faire deux commits séparés.

`git add myFile` (ou `git add .` si vous êtes sûr de ce que vous faites)

Et commitez :

```
git commit -m “My beautifull commit message”
```

Terminer une fonctionnalité

Lorsque votre fonctionnalité est terminée (sans les console.log, sout, …) et testée (avec tests unitaires, ou tests fonctionnels).

Avant de terminer, nettoyer votre chambre en utilisant un git rebase -i HEAD~n où “n” représente le nombre de “commits” que vous souhaitez réecrire à partir du premier. Fusionnez les commits en vous laissant guider par les commandes possibles. Puis…

Si vous utilisez “git flow” :

```
git flow feature finish myFeature
```

Si vous n’utilisez pas “git flow” :

```
git checkout devel

git merge myFeature

git branch -m myFeature
```

Préparer le push vers le dépôt distant

Avant de “pousser” votre branche “devel”, vous devez “tirer” la branche “devel” distante :

```
git pull origin devel
```

(Vérifiez que vous êtes bien situé dans la branche “devel” locale avant : `git branch`)

Corrigez les éventuels conflits de fusion (vous pouvez utiliser git lens de VSCode) et surtout, ne corrigez les conflits qu’avec les principaux concernés (pas seul donc) et poussez la branche “devel” :

```
git push origin devel
```

Vous pouvez désormais repartir sur une nouvelle fonctionnalité. N’oubliez pas de prévenir les autres collaborateurs que vous avez poussé vos modifications !

Intégrer “devel” dans “main”

Cette opération doit être effectuée par le “Dictateur”, qui est le seul à pouvoir mettre à jour la branche “main”.

Avec “git flow” :

`git flow release start M.m.m` (M : version majeure, m : version mineure, m : micro version) (Par exemple : git flow release start 1.1.0)

Mettez à jour : RELEASE-NOTE.md et si frontend “package.json” pour mettre à jour la version. Le fichier RELEASE-NOTE.md doit contenir les notes de mises à jour :

https://www.freecodecamp.org/news/a-beginners-guide-to-git-what-is-a-changelog-and-how-to-generate-it/

Terminer la release

Avec “git flow” :

Créer le dernier commit avec le changelog et les fichiers de configuration à jour, puis :

```
git flow release finish M.m.m
```

Une fois l’opération terminée, le Dictateur peut pousser la branche “main”, la branche “devel” et la release (`git push M.m.m`)

C’est reparti pour la prochaine “release” en reprenant à partir des branches de fonctionnalité

Sans “git flow”

Créez une branche à partir de la branche “devel” :

```
git checkout -b release-M.m.m devel
```

Faites les modifications dans cette branche (bump de version et génération du changelog.md)

Taggez la branche :

```
git tag M.m.m
```

Fusionnez la branche de release dans “main” :

```
git checkout main && git merge release-M.m.m
```

Back merger “main” dans “devel” :

```
git checkout devel && git merge main
```

Supprimez la branche de release : git branch -d release-M.m.m et poussez “main”, “devel” et la version taggée