memo-tp3-branches-depot-unique

Collaboration sur Git avec des branches - BUT Info 1 - 2023

Alice et Bob collaborent à l'aide de branches sur un dépôt unique

le problème

- Supposons qu'Alice et Bob veulent collaborer sur le projet super-proj
- Alice possède le code initial qu'elle publie sur son dépôt gitlab : https://gitlab.com/alice/sup
  proj.git

Dépôt initial et invitations

Alice invite Bob sur son projet en tant que developer (droit de lecture et écriture) ou maintainer (tous les droits).

Bob clone le projet d'Alice

- Bob va faire un clone du dépôt d'Alice sur Gitlab, pour obtenir une copie locale
  :

```
git clone https://gitlab.com/alice/super-proj.git
```

- chacun vérifie qu'il a un remote origin en tapant :

```
git remote -v
```

(qui doit afficher 2 lignes)

Le travail collaboratif

Bob choisit une feature à développer Bob crée une branche nommée en conséquence, par exemple ici readme et puis l'utilise :

```
git branch readme
git switch readme
```

ou il peut le faire aussi en une seule ligne :

```
git switch -c readme
```

Bob fait le travail Puis Bob lance ses tests, vérifie qu'ils passent et commite !

```
git commit -am "Intégration Readme.md"
```

Bob pousse sa branche sur Gitlab Bob pousse sur son remote origin sa branche readme.

```
git push origin readme
```

Bob fait un merge request Bob se rend sur le dépôt d'Alice et y fait un Merge Request ou MR (Gitlab) ou Pull Request ou PR (sur Github ou Bitbucket)

Alice accepte directement le Merge Request de Bob Alice peut accepter automatiquement le Merge Request de Bob si celui-ci est très simple, directement sur le Gitlab. Dans ce cas elle n'a plus qu'à faire un simple pull (git pull) de son main pour que sa copie locale soit à jour aussi.

Alice teste le code de Bob Alice récupère la branche de Bob :

```
git fetch origin readme
```

ou

```
git fetch origin readme --tags
```

pour aussi récupérer des tags éventuels.

ou Alice télécharge directement la branche concernée depuis Gitlab.

Alice consulte la branche readme réalisée par Bob :

```
git switch readme
```

(pour lister toutes les branches : `git branch -av`)

Alice revient dans son main :

```
git switch main
```

Alice traite le Merge Request de Bob Alice se rend sur son dépôt Gitlab, puis refuse ou accepte le Merge Request. Si elle l'accepte, met à jour sa copie locale :

```
git pull
```

Bob se met à jour Il reste à Bob à récupérer le main mis à jour par Alice pour se mettre aussi à jour.

Bob se met à jour avec un fetch et un merge :

```
git fetch origin main
git merge origin/main
```

Bob se met à jour directement avec un pull :

```
git pull origin main
```

ou simplement `git pull`.

Réalisation d'autres fonctionnalités par Bob

- Bob choisit une fonctionnalité ou feature à réaliser (item de backlong)
- Crée une branche feature1 dans son projet et y implémente le code nécessaire : puis `git switch -c feature1`, puis codage ...
- Bob implémente les tests correspondants et vérifie que ça échoue
- puis implémente les fonctionnalités suffisantes pour que le test passe
- puis commite son travail dans sa branche : `git commit -am \"Message de commit\"`
- et pousse sa branche sur son dépôt Gitlab: `git push origin feature1`
- Il visite alors la page de son projet (ou celle du projet d'Alice) et crée un Merge Request pour Alice

Alice traite le nouveau Merge Request de Bob

Cela peut se faire :

- Entièrement en ligne mais le code est seulement consulté dans ce cas et pas réellement testé. A réserver aux cas très simples ou lorsqu'Alice vient de travailler directement avec Bob !
- En récupérant le code de la branche feature1 de Bob et en le consultant et le testant. Ceci peut se faire en téléchargeant directement l'archive de la branche correspondante sur Gitlab ou en ligne de commande
- Alice vérifie que le code de Bob est correct
- Dans l'affirmative, Alice accepte le Merge Request (MR) de Bob sur Gitlab

- Dans le cas contraire, refuse le MR et Bob en est informé pour qu'il repropose une version modifiée selon les remarques d'Alice

Alice met à jour son dépôt local Si le MR a été accepté, le dépôt distant d'Alice fusionne automatiquement dans le main, la branche feature1 de Bob. Le dépôt distant d'Alice est donc à jour. Il lui reste à mettre à jour son dépôt local, ce qu'elle peut faire avec :

```
git fetch origin main
git merge origin/main
```

ou directement avec `git pull`

Bob se met à jour sur le nouveau main d'Alice et pousse les modifs dans son dépôt distant Bob revient dans son main :

```
git switch main
```

puis se met à jour des dernières modifications d'Alice qui intègrent sa branche :

```
git pull
```

Puis Bob met à jour son dépôt distant avec le code ainsi récupéré :

`git push`

Puis detruit si besoin sa branche locale feature1 :

```
git branch -d feature1
```

etc.

Commandes utiles sur les branches

Renommage de branche

Pour renommer une branche :

```
git branch -m <oldname> <newname>
```

Switcher sur des branches

Pour basculer sur une branche existante :

```
git switch <feature>
```

ou sur une nouvelle branche :

```
git switch -c <newname>
```

ou pour revenir à l'ancienne banche :

```
git switch -
```

Modification d'une branche distante

Si vous voulez mettre à jour une branche distante mabranche :

```
 git push origin mabranche --force
```

A utiliser si d'autres n'ont pas déjà intégré votre branche par exemple ...

Suppression d'une branche distante

Pour supprimer mabranche distante :

```
git push origin --delete mabranche
```