Git et GitHub : versions, branches et collaboration

Formation : BTS SIO — Option SISR

1. Rôle et fonctionnement

Git conserve l'historique local d'un projet. GitHub héberge des dépôts et fournit des outils de collaboration. Le répertoire de travail contient les fichiers modifiés ; l'index prépare le prochain commit ; le dépôt conserve les versions enregistrées.

2. Cycle de travail

```
git status
git diff
git switch -c documentation/dns
git add docs/documentation/services/dns.md
git diff --staged
git commit -m "Documenter le fonctionnement du DNS"
```

L'identité se configure avec `git config user.name` et `git config user.email`, au niveau du dépôt ou avec –`global`. Un commit reste local jusqu'à sa publication.

3. Dépôt distant et Pull Request

```
git remote -v
git fetch origin
git log --oneline --graph --all
git push -u origin documentation/dns
```

Une Pull Request permet de comparer une branche, discuter les changements et les fusionner après validation. `fetch` récupère l'historique distant sans fusionner ; `pull` récupère puis intègre selon la configuration. Une divergence n'entraîne pas systématiquement un conflit : celui-ci dépend des modifications concurrentes.

4. Annulation et conflits

`git restore –staged fichier` retire un fichier de l'index sans effacer son contenu de travail. `git restore fichier` rétablit le fichier depuis l'index et peut faire perdre les modifications non enregistrées : examiner `git diff` avant utilisation.

En cas de conflit, comprendre les deux versions, éditer le résultat, supprimer les marqueurs, tester puis enregistrer la résolution. `git revert` crée un commit inverse et convient souvent mieux qu'une réécriture de l'historique partagé.

5. Vérification

Avant publication, vérifier le statut, les différences préparées, la branche et le dépôt distant. Ne jamais versionner de secret, même dans un fichier supprimé au commit suivant : il resterait dans l'historique.

Le dépôt utilisé dans la situation est `Azuneer/CUB-Scripts-Administration`. Les commandes ci-dessus sont des exemples, pas une publication automatique du travail.

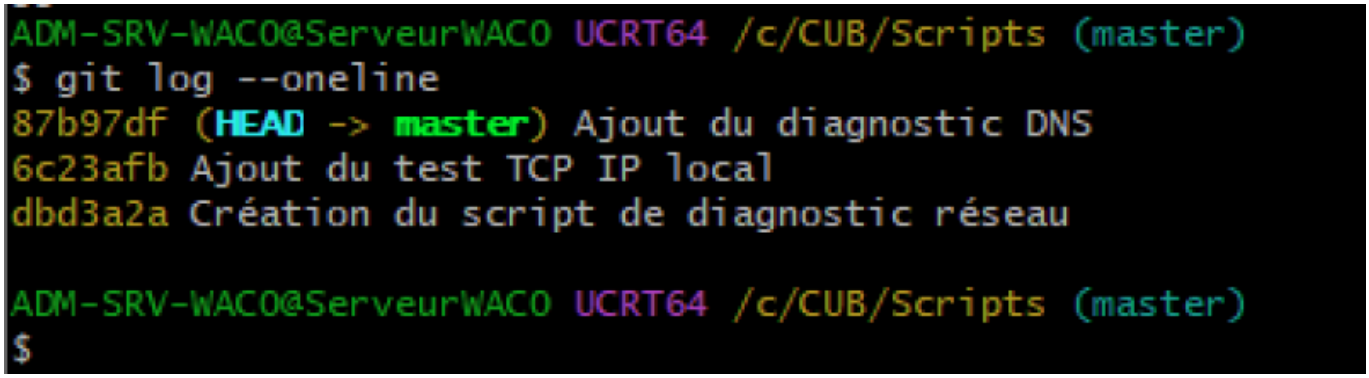
6. Exemple : versionner un script PowerShell

Dans un nouveau répertoire de travail :

```
git init
git status
git add diagnostic-reseau.ps1
git diff --cached
git commit -m "Ajouter le diagnostic réseau"
git log --oneline
```

`git init` crée un dépôt local. `git add` prépare le contenu du prochain commit ; `git commit` l'enregistre dans l'historique local. Aucun de ces deux appels ne publie le travail sur GitHub.

Sans argument, `git diff` compare le répertoire de travail à l'index. `git diff –cached` compare l'index au dernier commit.

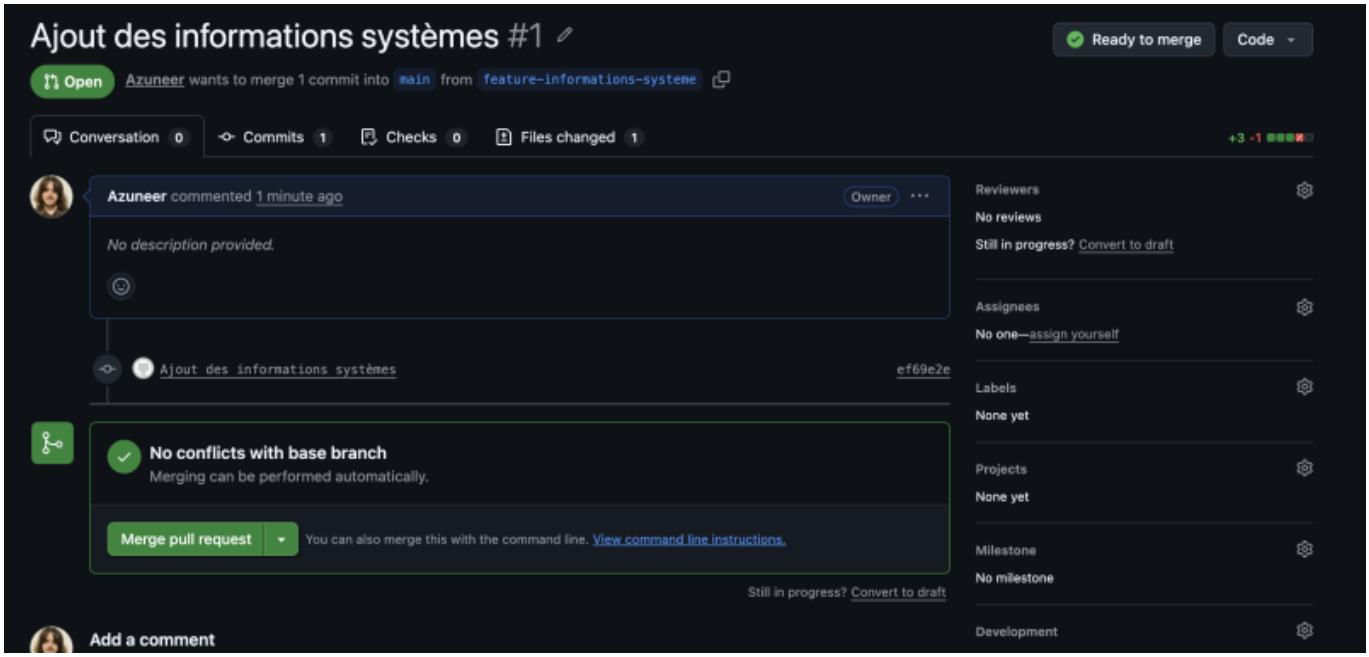


Historique de trois versions du script.

7. Travailler dans une branche

```
git switch -c feature-diagnostic-dns
# Modifier le script puis vérifier les différences
git diff
git add diagnostic-reseau.ps1
git commit -m "Ajouter le diagnostic DNS"
git push -u origin feature-diagnostic-dns
```

La dernière commande suppose qu'un dépôt distant `origin` a déjà été configuré. La publication de la branche permet ensuite d'ouvrir une pull request.



Proposition de fusion de la branche sur GitHub.

8. Fusion et conflits

Pour une fusion locale, partir d'un répertoire de travail propre et se placer sur la branche qui doit recevoir les changements :

```
git switch main
git merge feature-diagnostic-dns
```

En cas de conflit, lire les deux variantes, choisir le contenu cohérent avec les auteurs concernés puis supprimer les marqueurs de conflit. Tester le résultat avant d'enregistrer la résolution :

```
git add diagnostic-reseau.ps1
git commit
```

La résolution n'appartient pas automatiquement au dernier auteur ayant publié. La personne réalisant la fusion doit comprendre les modifications et, si nécessaire, échanger avec leurs auteurs.

9. Annuler une modification locale

```
git diff -- diagnostic-reseau.ps1
# Après vérification : abandonner les modifications non indexées de ce fichier
git restore -- diagnostic-reseau.ps1
```

Cette commande supprime les changements non indexés du fichier. `git revert` répond à un autre besoin : enregistrer un nouveau commit qui inverse un commit existant.

10. Références

- Git — branches et fusion
- Git — restore
- GitHub — Pull Requests

From:
https://wiki.ewengadonnaud.xyz/ - **Base de savoir réseaux/cyber/devops**

Permanent link:
https://wiki.ewengadonnaud.xyz/doku.php?id=devops:git

Last update: **2026/10/18 01:23**