Git , Workflow basique des projets

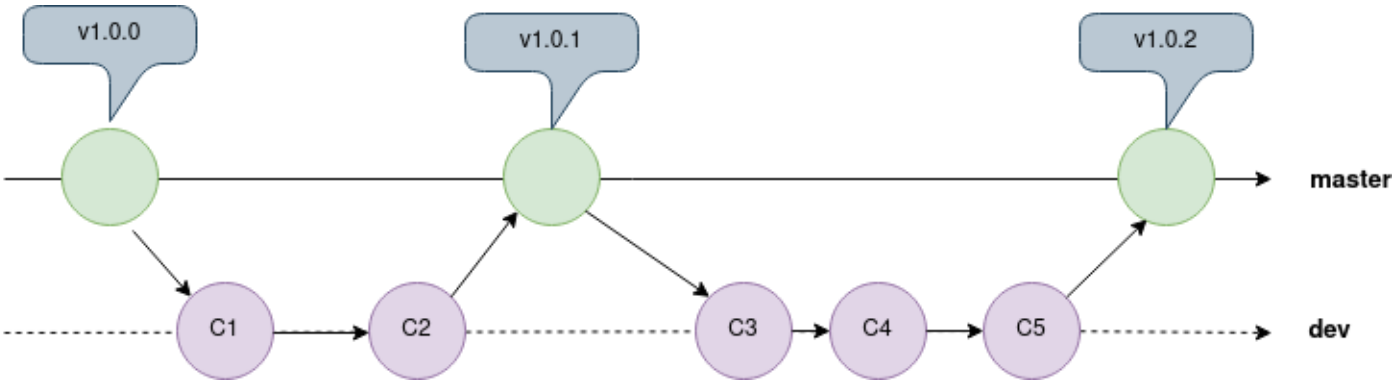
Il est temps de découvrir deux notions indispensables de git, les **tag** et les **branches**.

- Un tag est une version de logiciel stable, étiqueté par un numéro de version.
- Une branche vous permet de diverger de la ligne principale de développement et de travailler sans impacter celle-ci.

Il faut bien comprendre que l'ensemble des versions des fichiers reste présente sur votre poste (et sur le serveur). Vous pouvez naviguer d'une branche à l'autre, d'une version à l'autre et connaître les changements apportés à chaque commit.

Workflow

Le workflow décrit ci-dessous est un workflow "basique" pour une gestion de projet avec un seul développeur sur le projet. Nous étudierons dans votre formation un workflow plus évolué pour travailler à plusieurs développeurs sur le même projet.



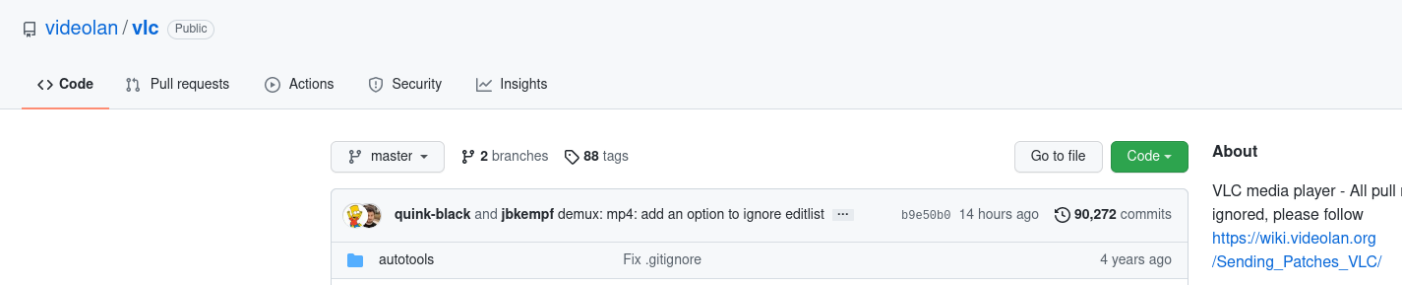
L'image présente deux lignes de développement, **master** et **dev**, appelées `branches` pour git. On voit aussi que les commit sur la branche master sont les seuls à posséder des numéros de versions.

Le principe général est que :

- La branche `master` contient toujours une version stable du logiciel
- Tous les développements de nouvelles fonctionnalités sont fait sur la branche `dev`
- Lorsque la fonctionnalité est complètement stable, on **fusionne** la branche `master` avec la branche de `dev`
- A ce moment, on crée un `tag` pour la nouvelle version stable

Remarque : Le cycle de développement décrit ci-dessus est répété pour chaque nouvelle fonctionnalité

Remarque : Ce workflow (tag et branch) est celui suivi par la majorité des projets utilisant git (comme VLC ci-dessous)



Gérer les tags

Les tags vous permettent d'avoir des numéros de versions de vos développements. Ils sont à créer pour chaque version stable.

- Créer un tag (exemple v1.0)

```
git tag -a v1.0 -m "message de version"
```

- Voir les tags du projet

```
git tag
```

- Pousser le tag sur le serveur distant

```
git push origin vx.x
```

- Supprimer un tag en local

```
git tag -d vx.x
```

- Supprimer un tag sur le serveur distant

```
git push origin --delete vx.x
```

Travailler avec les branches

Créer une branche signifie diverger de la ligne principale de développement et continuer à travailler sans impacter cette ligne. Dans un premier temps, vous aurez deux branches à gérer `master` et `dev`.

Créer une nouvelle branche pour votre projet

- Créer une branche locale sur votre PC

```
git checkout -b dev
```

- Pousser la branche locale vers le dépôt distant

```
git push -u origin dev
```

Remarque : La branche dev est créée automatiquement sur le serveur.

Travailler avec des branches locales (sur le PC de développement)

- Voir les branches locales

```
git branch
```

- Voir les branches locales et distantes

```
git branch -a
```

- Basculer sur une autre branche

```
git checkout nom_branche
```

Remarque :

- la commande `checkout -b` crée une branche si elle n'existe pas.
- la commande `checkout` permet de basculer sur une autre branche (elle doit exister)

Fusionner une branche

Lorsque la fonctionnalité est `développée`, vous devez fusionner (`merge`) la branche `master` et `dev`. Cela va ajouter toutes les modifications développées sur `dev` à `master`.

- Se positionner sur la branche à fusionner

```
git checkout master
```

- Fusionner dev à master

```
git merge dev
```

- Pousser le travail sur le serveur

```
git push origin master
```

Remarque :
Juste après une fusion, c’est le moment de créer le tag de version.

Travailler avec des branches distantes (sur le serveur)

Par défaut, lorsque vous clonez un projet, seule la branche master est téléchargée sur votre poste. Si vous voulez récupérer une autre branche, il faut le préciser :

- Soit lors du clone du projet
- Ou si c’est un projet déjà cloné, avec un checkout de la branche distante.

CAS DU CLONE D’UN NOUVEAU PROJET

- Cloner une branche spécifique (ici version1)

```
git clone --branch version1 https://git.vainsta.fr/projet
```

Cas d’un projet déjà cloné, récupération d’une branche supplémentaire

- Voir les branches distantes

```
git remote show origin
```

- Récupérer une (autre) branche distante(ici origin/version2) dans son dépôt local

```
git checkout -b version2 origin/version2
```

Renommer une branche

Cela arrive lors d'une erreur de manipulation ;O

- Renommer une branche locale

```
git branch -m new_branch_name
```

- Renommer une branche (locale et distante)

SubModules

Les sous modules vous permettent de gérer les dépendances. C'est à dire des bibliothèques utiles (ou indispensables) au fonctionnement de votre projet dont vous n'êtes pas responsables (en tant que développeur)

Un sous module va permettre d'intégrer le code de la bibliothèque à votre projet et de simplement conserver un lien vers celle-ci.

Le sous module sera présent dans votre répertoire local. Seul le lien vers le sous module apparaîtra sur votre dépôt distant.

- Ajouter un sous module

```
git submodule add adresse_du_depot
```

- Récupérer les modifications d'un sous module chargé

```
git submodule update --remote
```

- Lister tous les sous modules

```
git submodule foreach
```

- Réinitialiser les changements dans les sous modules (solution 1)

```
git submodule foreach git reset --hard
```

- Réinitialiser les changements dans les sous modules (solution 2)

```
git submodule deinit -f .
git submodule update --init
```

Intégration aux IDE

Utilisation de GIT dans l’IDE QtCreator

Git est intégré à QTCreator. Si un dépôt Git est déjà initialisé, QtCreator le détecte automatiquement.

Attention : Je ne conseille pas d’utiliser les commandes ci-après avant de maîtriser l’outil en ligne de commande

VERSIONNER UN FICHIER

- Ajouter le fichier (équivalent de git add)

```
Outil → Git → FichierCourant → Ajouter main.c au staging pour commit
```

- Commiter le fichier (équivalent de git commit)

```
Outil → Git → Dépôt local → Commit
```

- Pousser sur le dépôt distant

```
Outil → Git → Dépôt distant → push
```

- Vérifier que les modifications apparaissent sur le serveur.

Utilisation de GIT dans l’IDE VsCode

La documentation n’est pas détaillée, mais de manière générale, tous les IDE modernes détectent automatiquement git et les autres systèmes de gestion de version.

Revision #3
Created 28 May 2024 12:26:58 by Nicolas
Updated 9 February 2025 15:11:07 by Nicolas