git

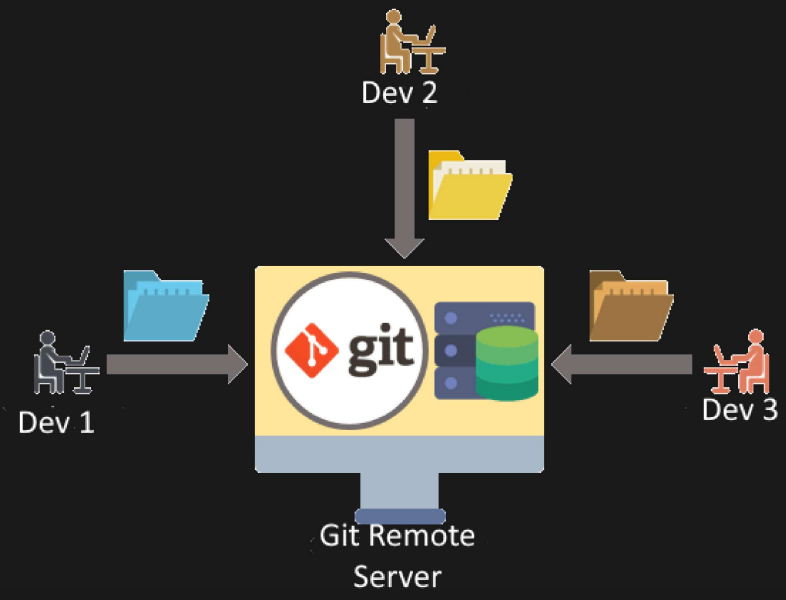
QU'EST-CE QUE C'EST ?

C'est un système de gestion de versions, c'est-à-dire de suivi des modifications apportées à un code, un dossier, un projet.

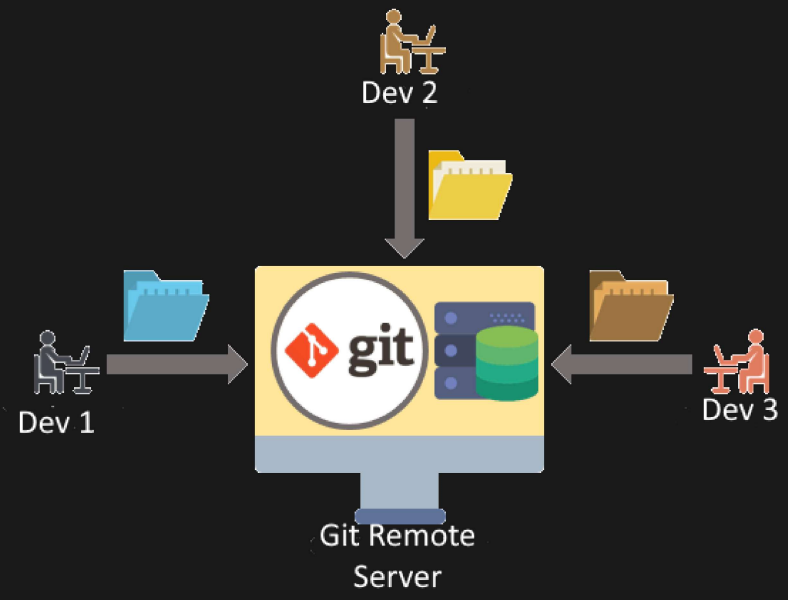
A ne pas confondre

Git c'est le système, GitHub c'est une **plateforme** qui héberge des dépôts Git, il y en a d'autres (GitLab par exemple) et en utiliser une n'est pas obligatoire quand on bosse seul, mais il ne faut pas confondre.

POURQUOI UTILISER GIT?

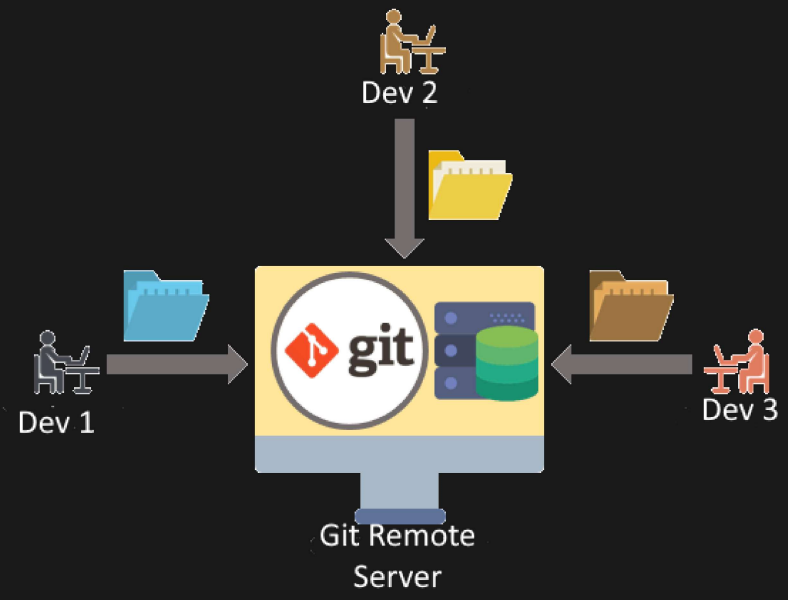


POURQUOI UTILISER GIT?



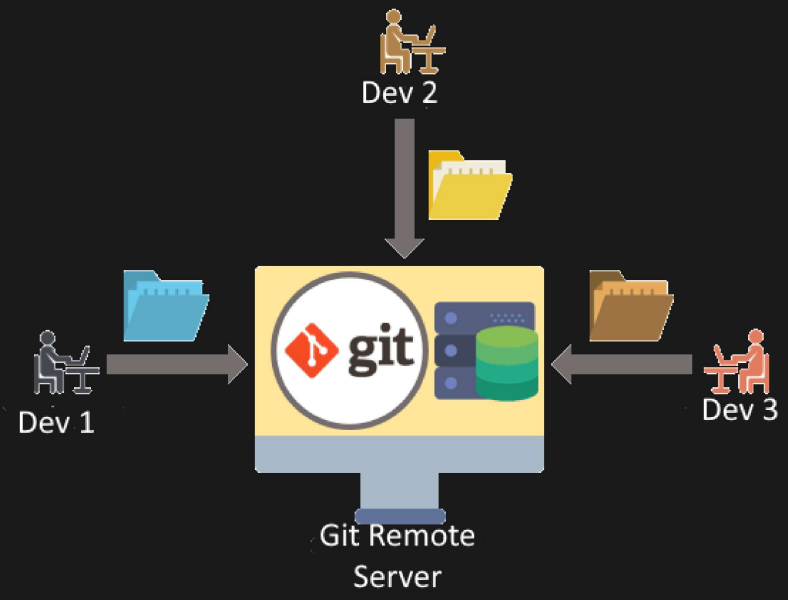
- Pour travailler à plusieurs

POURQUOI UTILISER GIT?



- Pour travailler à plusieurs
- Pour conserver un programme fonctionnel à tout instant

POURQUOI UTILISER GIT?

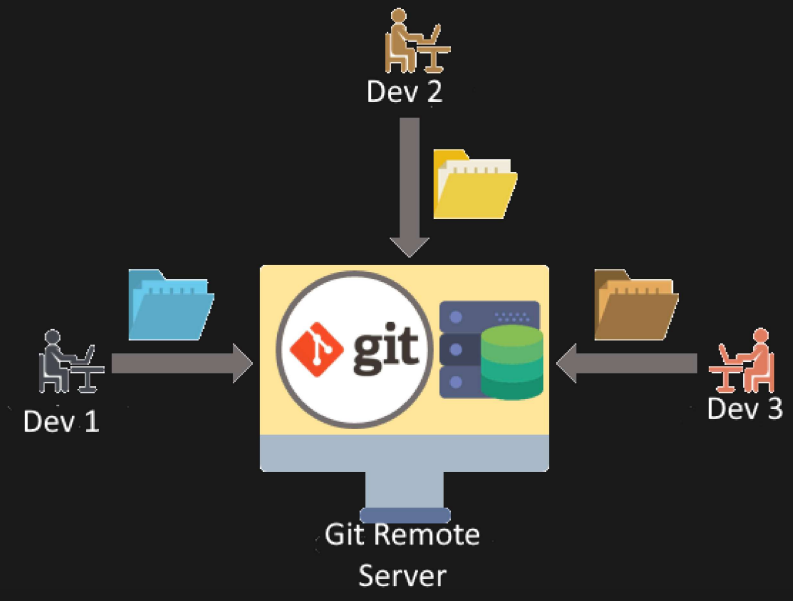


- Pour travailler à plusieurs
- Pour conserver un programme fonctionnel à tout instant
- Pour partager son dépôt à des utilisateurs distants

FONCTIONNEMENT DE GIT

```
philemon.prevot@bigpus:/nfs/NAS7/SABIOD/METHODE/QHB_Tools$ git log
commit cebdba11284ad88acd0c194e47a48c47cd2111b6 (HEAD -> main, linux_fork/main)
Author: valentinbarchasz <valentin.barchasz@gmail.com>
Date:   Thu Jun 27 14:08:19 2024 +0200

    Added QHB_V3 Documentation and scripts

commit bc2d5de274cadf300647177df64e96d000a5c10f
Merge: ebc61c4 c0e945f
Author: valentinbarchasz <valentin.barchasz@gmail.com>
Date:   Mon Jun 24 14:54:34 2024 +0200

    Merge pull request #6 from PhilemonPrevot/main

    Creating a README.md

commit c0e945fae2743ddfe1f089c7f9a928ec5b2c3931
Merge: 39d5cd8 ebc61c4
Author: PhilemonPrevot <128744613+PhilemonPrevot@users.noreply.github.com>
Date:   Mon Jun 24 14:48:30 2024 +0200

    Merge branch 'valentinbarchasz:main' into main
```

REMOTE ET LOCAL REPOSITORIES

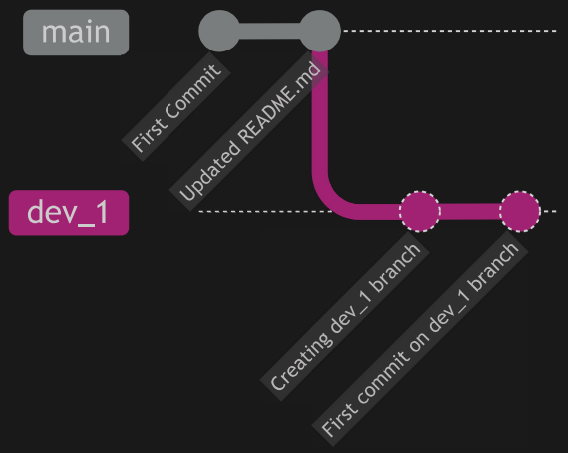


GITGRAPH ET BRANCHES

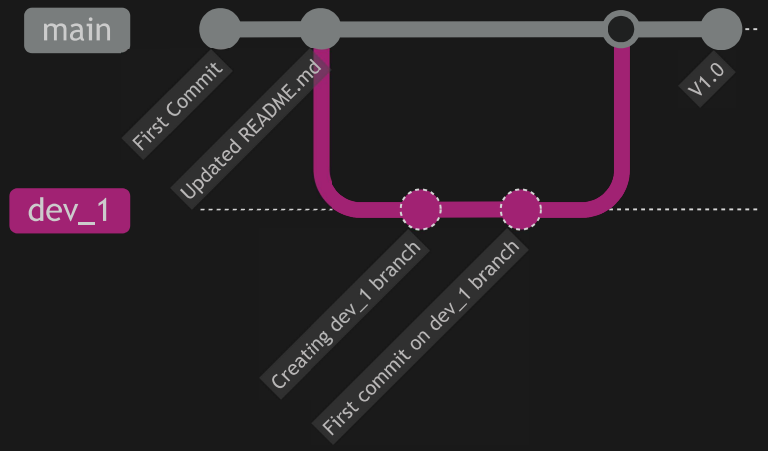
```
philemon.prevot@bigpus:/nfs/NAS7/SABIOD/METHODE/QHB_Tools$ git log
commit cebdba11284ad88acd0c194e47a48c47cd2111b6 (HEAD -> main, linux_fork/main)
Author: valentinbarchasz <valentin.barchasz@gmail.com>
Date:   Thu Jun 27 14:08:19 2024 +0200

    Added QHB_V3 Documentation and scripts

commit bc2d5de274cadf300647177df64e96d000a5c10f
Merge: ebc61c4 c0e945f
Author: valentinbarchasz <valentin.barchasz@gmail.com>
Date:   Mon Jun 24 14:54:34 2024 +0200

    Merge pull request #6 from PhilemonPrevot/main

    Creating a README.md

commit c0e945fae2743ddfe1f089c7f9a928ec5b2c3931
Merge: 39d5cd8 ebc61c4
Author: PhilemonPrevot <128744613+PhilemonPrevot@users.noreply.github.com>
Date:   Mon Jun 24 14:48:30 2024 +0200

    Merge branch 'valentinbarchasz:main' into main
```

```
philemon.prevot@bigpus:/nfs/NAS7/SABIOD/METHODE/QHB_Tools$ git log --graph
* commit cebdba11284ad88acd0c194e47a48c47cd2111b6 (HEAD -> main, linux_fork/main)
| Author: valentinbarchasz <valentin.barchasz@gmail.com>
| Date:   Thu Jun 27 14:08:19 2024 +0200
|
|     Added QHB_V3 Documentation and scripts
|
*   commit bc2d5de274cadf300647177df64e96d000a5c10f
|\  Merge: ebc61c4 c0e945f
| | Author: valentinbarchasz <valentin.barchasz@gmail.com>
| | Date:   Mon Jun 24 14:54:34 2024 +0200
| |
| |     Merge pull request #6 from PhilemonPrevot/main
| |
| |     Creating a README.md
| |
| *   commit c0e945fae2743ddfe1f089c7f9a928ec5b2c3931
| |\  Merge: 39d5cd8 ebc61c4
| |/  Author: PhilemonPrevot <128744613+PhilemonPrevot@users.noreply.github.com>
|/|   Date:   Mon Jun 24 14:48:30 2024 +0200
| |
| |       Merge branch 'valentinbarchasz:main' into main
| |
```

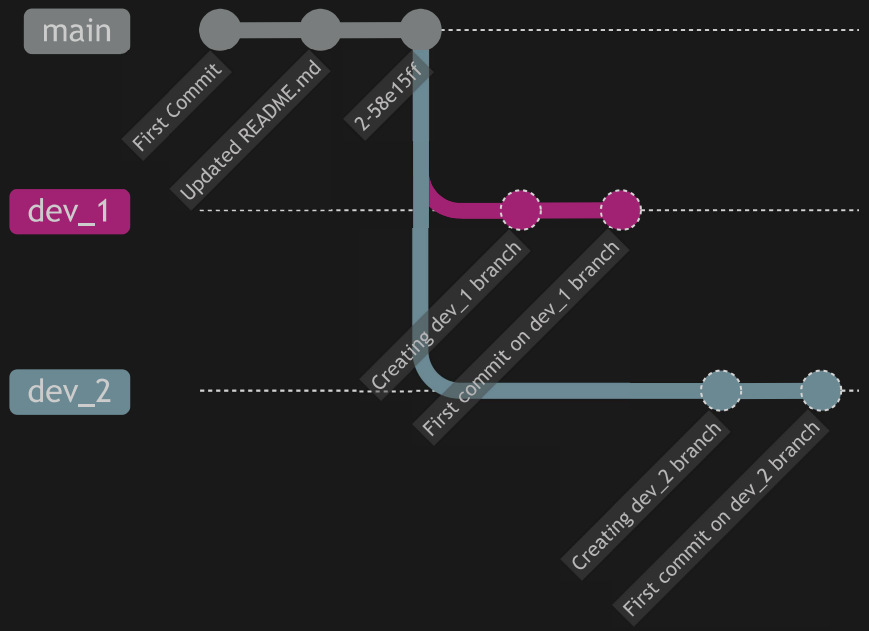
BRANCHES MAIN ET FEATURE/DEV



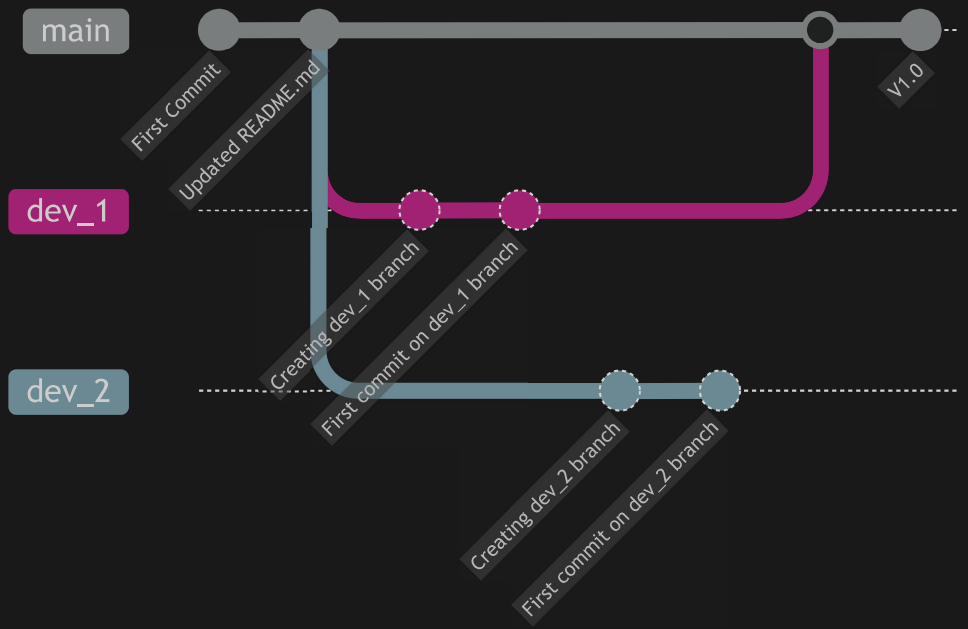
TERMINER UN DÉVELOPPEMENT : MERGE



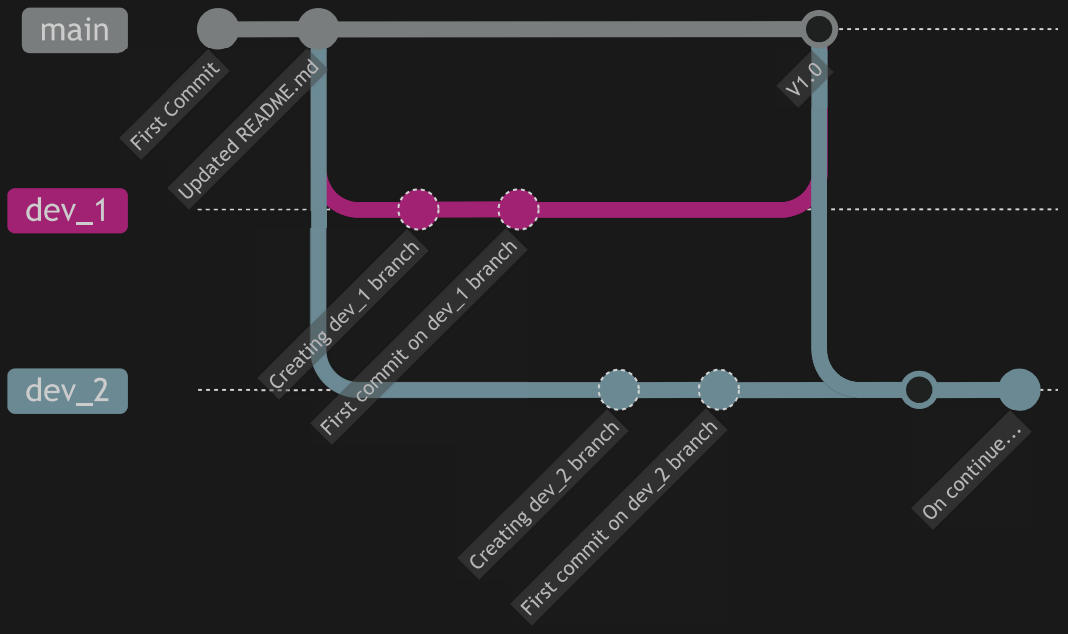
TRAVAILLER À PLUSIEURS



TRAVAILLER À PLUSIEURS



SUIVRE LE DÉVELOPPEMENT DES AUTRES : REBASE



COMMENT ON FAIT?

INSTALLER GIT

VÉRIFIER SI VOUS L'AVEZ DÉJÀ

```
PS C:\Users\PREVOT Philemon> git version
git version 2.45.0.windows.1
```

SINON :

- Pour Linux

```
sudo apt-get install git
```

- Pour Windows

```
https://git-scm.com/downloads
```

Et télécharger la dernière version pour Windows

COMMANDES DE BASE

COMMANDES DE BASE

- Clone

COMMANDES DE BASE

- Clone
- Add

COMMANDES DE BASE

- Clone
- Add
- Status

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit
- Pull

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit
- Pull
- Push

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit
- Pull
- Push
- Branch

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit
- Pull
- Push
- Branch
- Checkout

COMMANDES DE BASE

- Clone
- Add
- Status
- Commit
- Pull
- Push
- Branch
- Checkout
- Merge

GIT CLONE

```
git clone <ssh or url repository> <folder name>
```

GIT CLONE

```
git clone <ssh or url repository> <folder name>
```

Copier un dépôt Git en ligne dans un dossier afin de travailler dessus.

GIT CLONE

```
git clone <ssh or url repository> <folder name>
```

Copier un dépôt Git en ligne dans un dossier afin de travailler dessus.

Petit exemple :

GIT CLONE

```
git clone <ssh or url repository> <folder name>
```

Copier un dépôt Git en ligne dans un dossier afin de travailler dessus.

Petit exemple :

```
https://gitlab.lis-lab.fr/philemon.prevot/test_repo.git
```

GIT PULL

```
git pull <remote name> <branch name>
```

GIT PULL

```
git pull <remote name> <branch name>
```

```
git pull origin main
```

GIT PULL

```
git pull <remote name> <branch name>
```

```
git pull origin main
```

Télécharger les modifications ajoutées à un dépôt en ligne par d'autres développeurs.

GIT ADD

```
git add .
```

```
git add <file name>
```

Ajouter des fichiers et des modifications au suivi Git.

GIT STATUS

```
git status
```

Pour visualiser quelles modifications ont été ajoutées au prochain commit et lesquelles ne seront pas prises en compte.

GIT COMMIT

```
git commit -m "Short description of the changes"
```

Sauvegarder les modifications localement, créer une "version".

GIT PUSH

```
git push origin master (ou main)
```

Envoyer les versions (commits) sur le dépôt distant (origin) pour les globaliser.

GIT BRANCH

```
git branch <branch name>
```

Créer une nouvelle branche.

GIT CHECKOUT

```
git checkout <branch name>
```

Changer de branche active.

QUELQUES EXEMPLES

COPIER UN DÉPÔT EXISTANT POUR TRAVAILLER DESSUS

- Clone

COPIER UN DÉPÔT EXISTANT POUR TRAVAILLER DESSUS

- Clone
- Pull

CRÉER UN NOUVEAU DÉPÔT À PARTIR D'UN DOSSIER EXISTANT

- Créer sur Gitlab un dépôt et lui donner le nom souhaité (nom du dossier du dépôt)

CRÉER UN NOUVEAU DÉPÔT À PARTIR D'UN DOSSIER EXISTANT

- Créer sur Gitlab un dépôt et lui donner le nom souhaité (nom du dossier du dépôt)
- Dans le dossier du dépôt en local : `git init` pour créer un repo local, puis `git remote add origin <url ou ssh du nouveau dépôt git>` pour ajouter le repo distant en tant qu'origine au repo local.

CRÉER UN NOUVEAU DÉPÔT À PARTIR D'UN DOSSIER EXISTANT

- Créer sur Gitlab un dépôt et lui donner le nom souhaité (nom du dossier du dépôt)
- Dans le dossier du dépôt en local : `git init` pour créer un repo local, puis `git remote add origin <url ou ssh du nouveau dépôt git>` pour ajouter le repo distant en tant qu'origine au repo local.
- Add

CRÉER UN NOUVEAU DÉPÔT À PARTIR D'UN DOSSIER EXISTANT

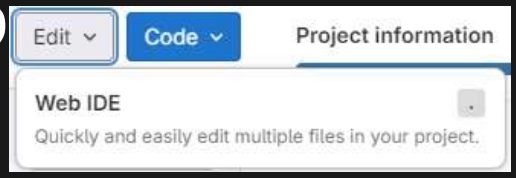
- Créer sur Gitlab un dépôt et lui donner le nom souhaité (nom du dossier du dépôt)
- Dans le dossier du dépôt en local : `git init` pour créer un repo local, puis `git remote add origin <url ou ssh du nouveau dépôt git>` pour ajouter le repo distant en tant qu'origine au repo local.
- Add
- Commit

CRÉER UN NOUVEAU DÉPÔT À PARTIR D'UN DOSSIER EXISTANT

- Créer sur Gitlab un dépôt et lui donner le nom souhaité (nom du dossier du dépôt)
- Dans le dossier du dépôt en local : `git init` pour créer un repo local, puis `git remote add origin <url ou ssh du nouveau dépôt git>` pour ajouter le repo distant en tant qu'origine au repo local.
- Add
- Commit
- Push

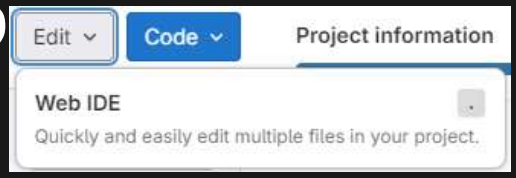
EFFECTUER DES MODIFICATIONS DIRECTEMENT DANS GITLAB

- Modifier son fichier avec l'IDE Web



EFFECTUER DES MODIFICATIONS DIRECTEMENT DANS GITLAB

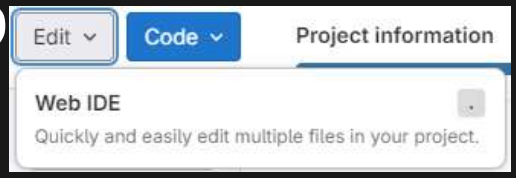
- Modifier son fichier avec l'IDE Web



- Commit (avec le bouton)

EFFECTUER DES MODIFICATIONS DIRECTEMENT DANS GITLAB

- Modifier son fichier avec l'IDE Web



- Commit (avec le bouton)

Pas besoin d'add, Gitlab le fait tout seul, et pas besoin de push non plus.

.GITIGNORE

.gitignore

```
.vscode/
data/
figure/
metadata/
psi-biom/
psibiom_project.venv/
methode/INDICES/
methode/short/
methode/small_detec/
methode/small_detec_multi/
methode/SPEC/
methode/YOLO/
methode/get_bounding_box.py
methode/vision_comparatif/test_dataset/
methode/vision_comparatif/yolov8/runs/
methode/vision_comparatif/yolov5/
methode/vision_comparatif/SSD/ssdlite_training_script/models/
methode/vision_comparatif/SSD/Yolo-to-COCO-format-converter
methode/vision_comparatif/SSD/pytorch_vision_code/
methode/vision_comparatif/SSD/ssdlite_training_script/__pycache__/
methode/mini_linux/tests/
methode/vision_comparatif/SSD/ssdlite_training_script/__pycache__/utils.cpython-39.pyc
methode/vision_comparatif/SSD/ssdlite_training_script/__pycache__/engine.cpython-39.pyc
methode/vision_comparatif/SSD/ssdlite_training_script/__pycache__/transforms.cpython-39.pyc
```

WORKFLOW

GIT FLOW

- Une branche **main** (ou master) qui est la branche déployée

GIT FLOW

- Une branche **main** (ou master) qui est la branche déployée
- Une branche **dev** qui est la branche sur laquelle les développeurs mergent leurs travaux

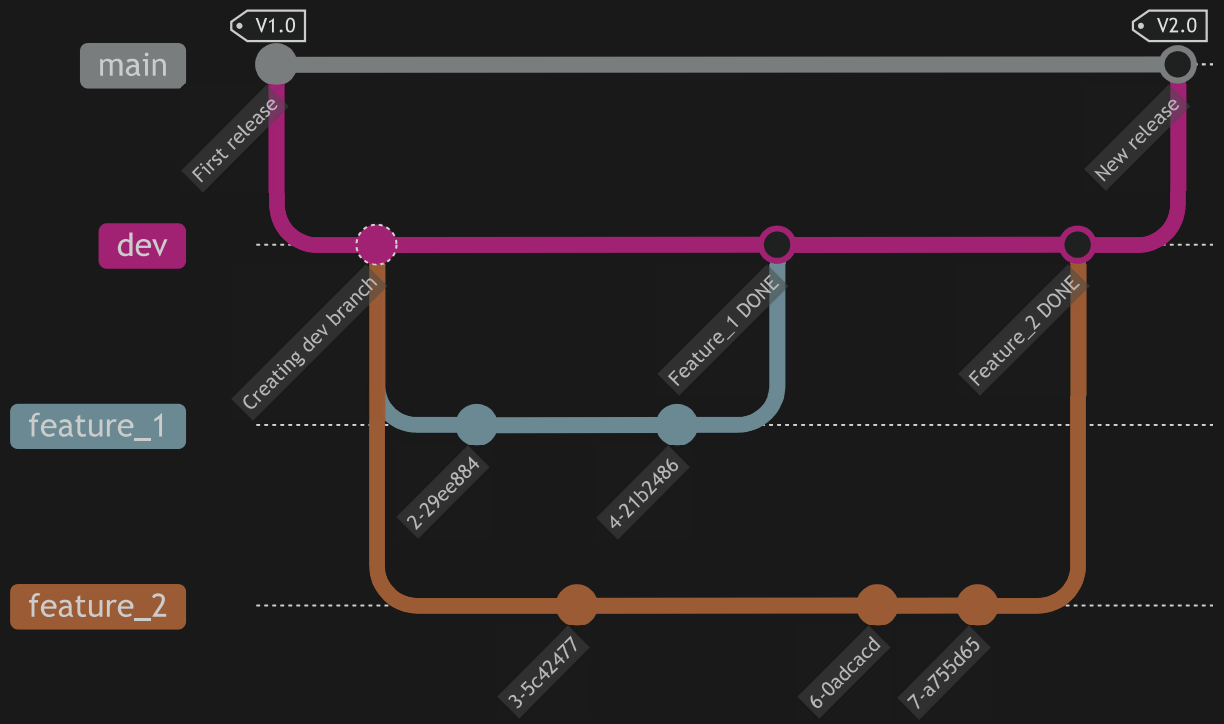
GIT FLOW

- Une branche **main** (ou master) qui est la branche déployée
- Une branche **dev** qui est la branche sur laquelle les développeurs mergent leurs travaux
- Des branches **feature** sur lesquelles les devs travaillent

GIT FLOW

Les branches **feature** sont merge dans **dev** à chaque fois que le développement de la feature en question est terminé. Mais la branche **dev** n'est merge sur **main** que lorsqu'on souhaite publier une nouvelle version du code. De plus elle est merge avec un nom de version.

GIT FLOW


V1.0
V2.0
main
First release
New release
dev
Creating dev branch
Feature_1 DONE
Feature_2 DONE
feature_1
2-29ee884
4-21b2486
feature_2
3-5c42477
6-0adcacd
7-a755d65

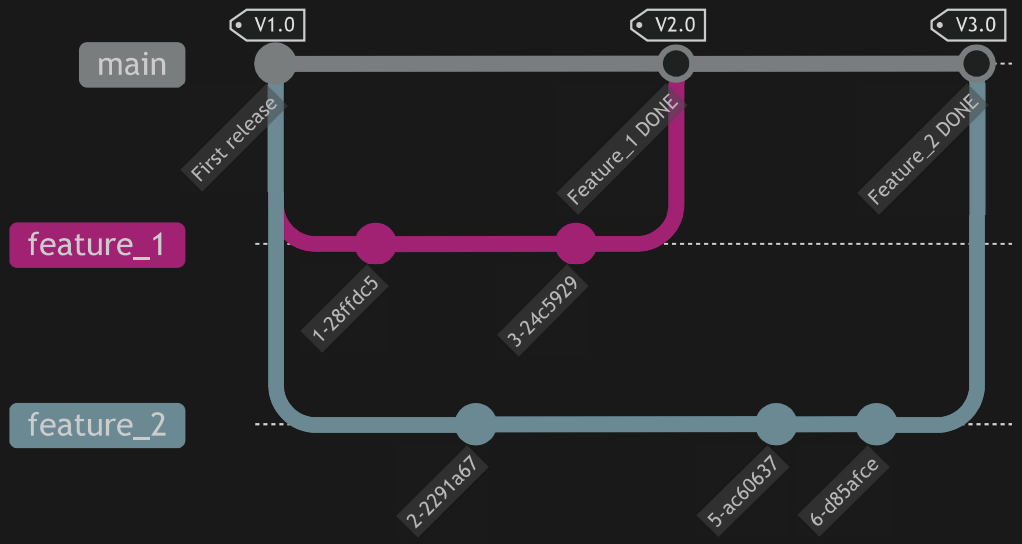
GITHUB FLOW

- Une branche **main** (ou master) qui est la branche déployée

GITHUB FLOW

- Une branche **main** (ou master) qui est la branche déployée
- Des branches **feature** sur lesquelles les branches travaillent et qui sont merge dans **main** quand les features sont terminées.

GITHUB FLOW


V1.0
V2.0
V3.0
main
First release
Feature_1 DONE
Feature_2 DONE
feature_1
1-28ffdc5
3-24c5929
feature_2
2-2291a67
5-ac60637
6-d85afce

MERCI POUR VOTRE ATTENTION!

Des questions?