



FORMATION

IT - Digital - Management



m2iinformation.fr



Git & GitHub

Versionner son code et travail collaboratif



Nehemie “Alfred” BALUKIDI
CTO/Software Engineering & Data|ML/AI



Votre formateur

- ❏ Nehemie “Alfred” BALUKIDI
- ❏ 7 ans d’exp en development
- ❏ 3 ans d’exp en formation
- ❏ Expertise Python
- ❏ Backend, MLOps
- ❏ Judo



Et Vous?

- ☐ Noms
- ☐ Entreprise et poste
- ☐ Parcours
- ☐ Vos attentes par rapport à cette formation ?
- ☐ Etes-vous familier avec la ligne de commandes?
- ☐ Les droits admin sur vos postes
- ☐ Etes vous sympa ?



Déroulé de la formation

- ❑ Début tous les jours à 9h00
- ❑ Pause le matin de 15 min
- ❑ Pause déjeuner de 12h30 à 13h30'
- ❑ Pause l'après midi de 15 min
- ❑ Fin à 17h

Chaque jour, avant de commencer et en revenant de la pause dej vérifier que les feuilles d'émargement sont signées et on fera les évaluations formateur (obligatoire avant 15h30).



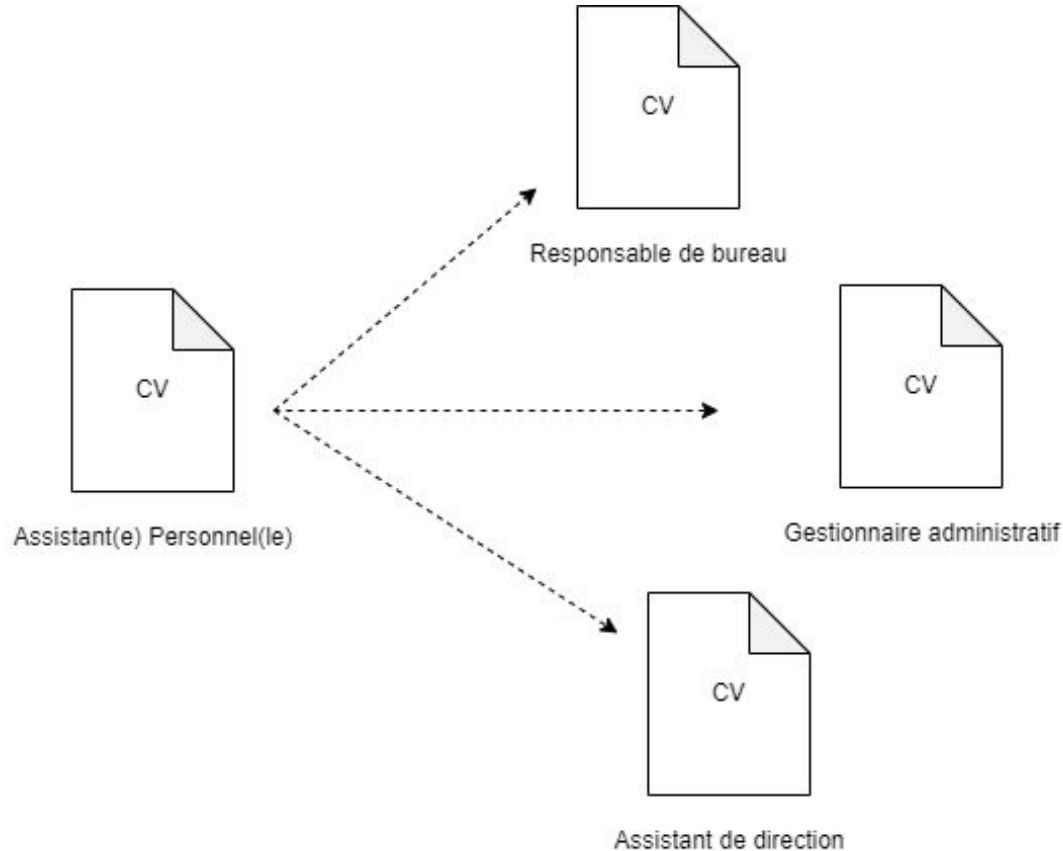
Objectifs du cours

- ❏ Comprendre les bases des outils de gestion de version, leur utilité, et pourquoi les utiliser.
- ❏ Maîtriser Git pour la gestion de version.
- ❏ Utiliser GitHub avec Git.
- ❏ Collaborer sur du code avec ces outils.



Gestion de version?

Le cauchemar des versions du CV

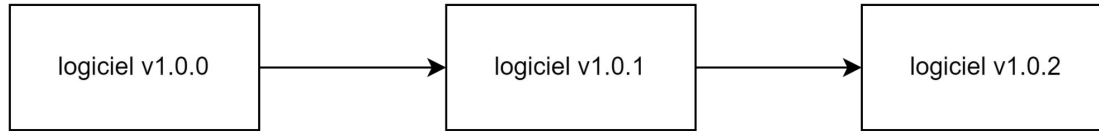




Un problème
généralisé.



En développement logiciel



- ❑ Suivi des modifications
- ❑ Collaboration
- ❑ Sauvegarde
- ❑ Déploiement

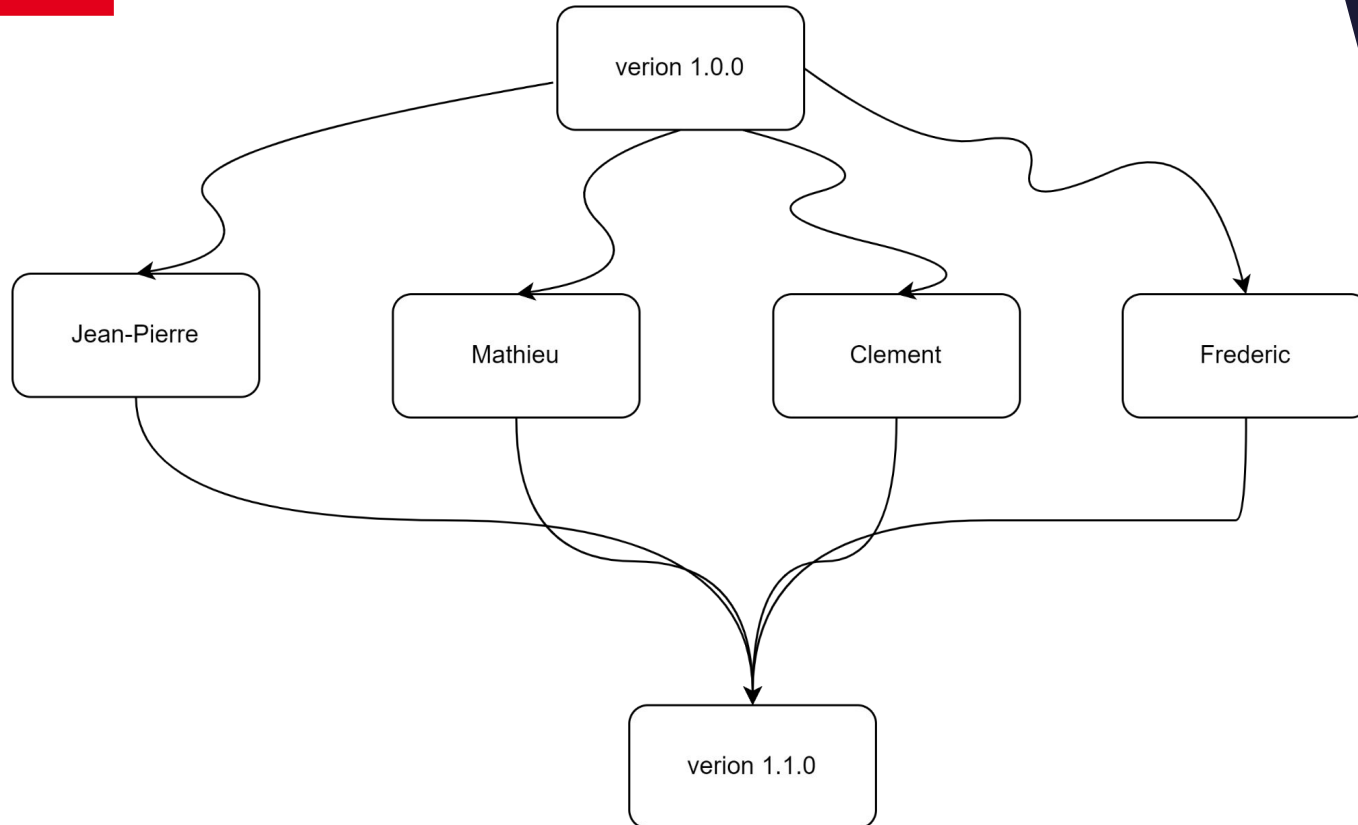
- ❑ Tests et validation
- ❑ Gestion des bogues
- ❑ Documentation
- ❑ Historique



Git

- ❑ Système de gestion de version.
- ❑ Créé par Linus Torvalds.
- ❑ Open source.
- ❑ Suivi des changements.
- ❑ Distribué.
- ❑ Collaboration.

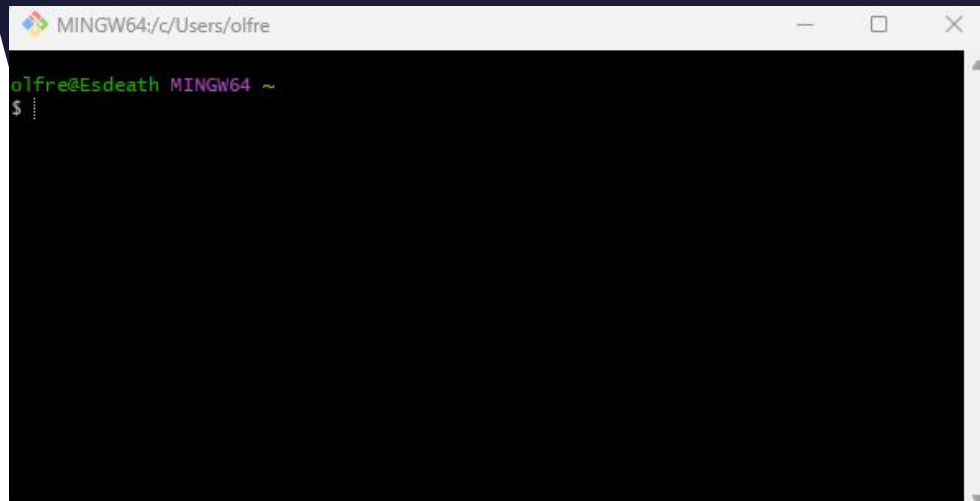
A propos de la collaboration





Démarrer avec Git

Télécharger sur <https://git-scm.com/>





La ligne de commande Git Bash

- ❑ Interface en ligne de commande
- ❑ Exécute des commandes textuelles
- ❑ Utilisé pour l'automatisation
- ❑ Répertoire de commandes
- ❑ Pas d'interface graphique



La ligne de commandes Git Bash(suite)

- ❑ **cd <chemin>** : Permet de naviguer dans notre file system
- ❑ **ls** : Elle permet de lister le contenu d'un répertoire
- ❑ **mkdir <nom_du_repertoire>** : permet de créer un répertoire ou dossier
- ❑ **rm <nom_du_fichier>** : Supprimer un fichier
- ❑ **rm -d <nom_du_dossier>** : Supprimer un dossier vide
- ❑ **rm -r <nom_du_dossier>** : Effectuer une suppression récursive
- ❑ **touch <nom_fichier>** : Permet de créer un fichier
- ❑ **cat <nom_fichier>** : Permet de voir ou lire le contenu d'un fichier
- ❑ **pwd** : Permet de voir le chemin absolu vers le dossier où l'on se trouve
- ❑ **du -h** : Taille occupé par les dossiers(l'option -h c'est pour avoir un format d'affichage facile à comprendre).



Premiers pas avec Git: identification

- ❏ Configurer(globalement) votre identité

```
git config --global user.name "Robin des bois"
```

```
git config --global user.email "robin.dbois@gmail.com"
```

- ❏ Voir toutes configurations en place

```
git config -list
```




Débuter avec un project Git

- ❑ C'est le dossier contenant les fichiers à versioner.
- ❑ Git surveil le contenu de ce dossier
- ❑ Git sait dire si le contenu du dossier a changé
- ❑ Donne accès aux commandes Git
- ❑ L'initialisation se fait avec la commande: **git init**
- ❑ **Assurez-vous d'être dans le bon répertoire de travail avant d'initier la commande**

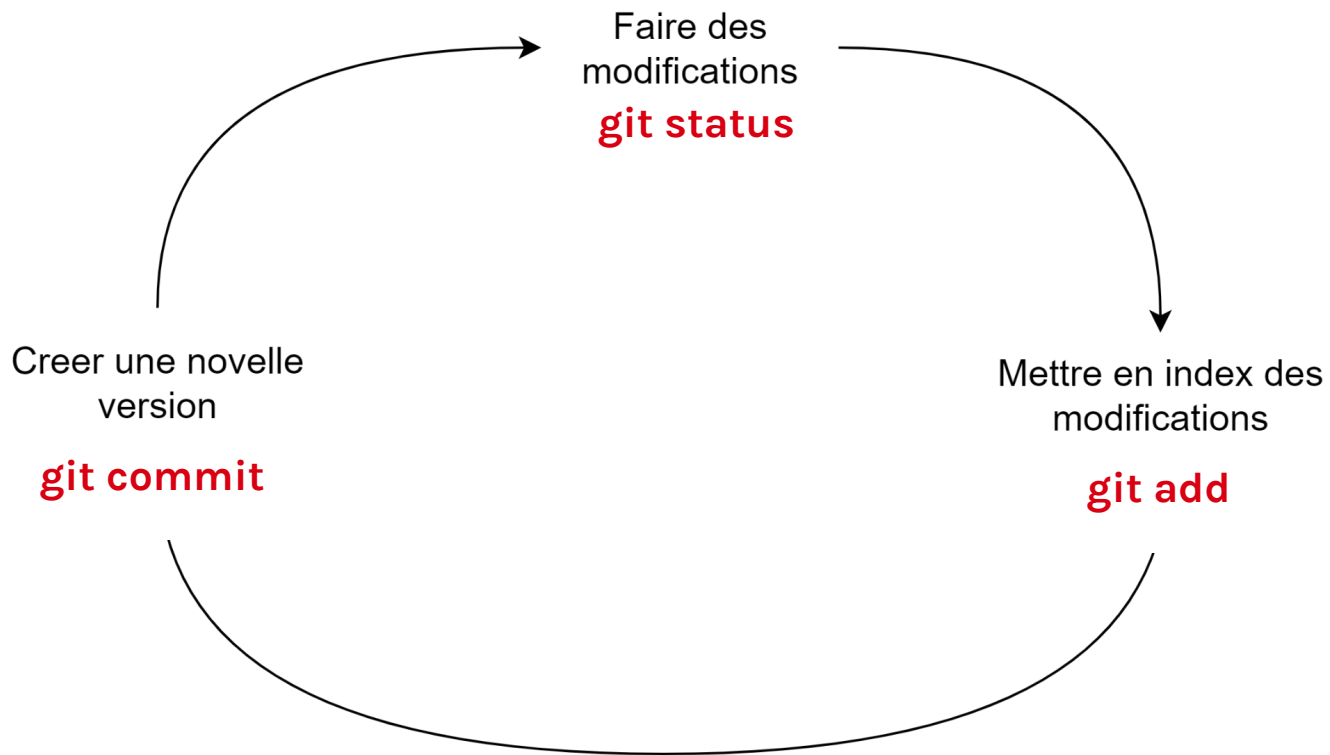


Le dépôt Git

- ❑ C'est le dossier caché **.git** dans un projet Git.
- ❑ Contient toute l'information nécessaire à, et recueillies par Git
- ❑ NE JAMAIS SUPPRIMER CE DOSSIER

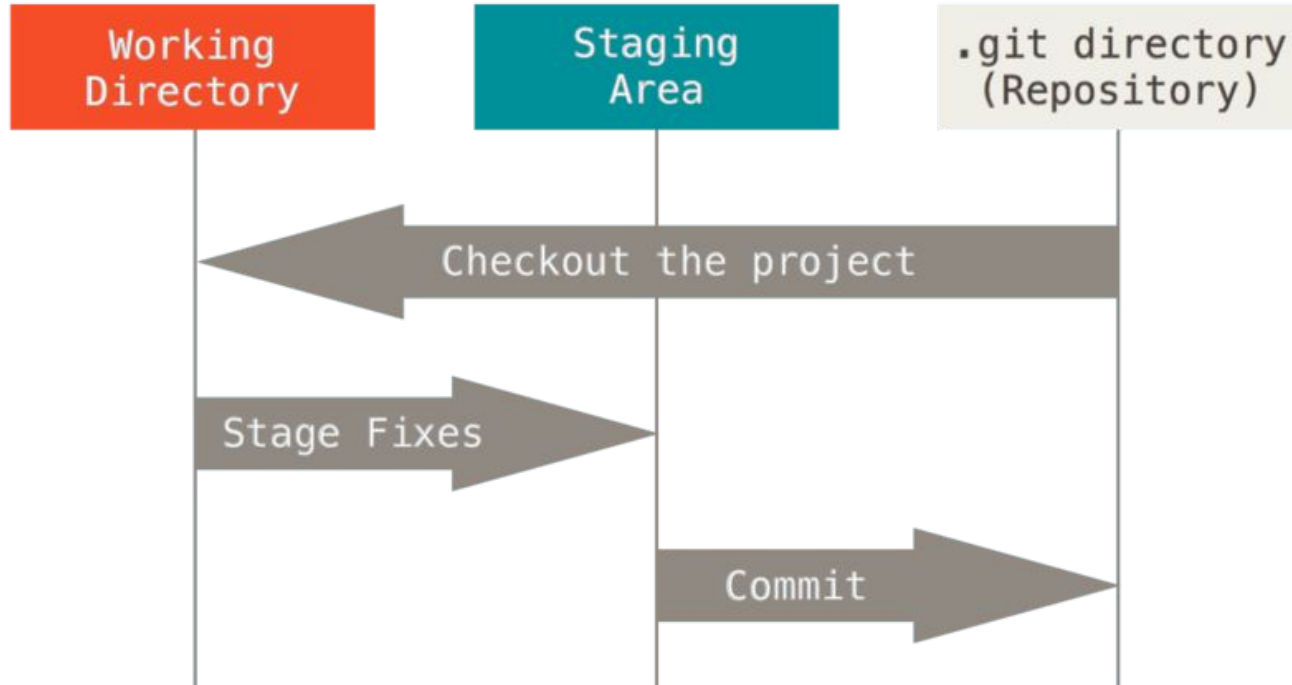


Le cycle de travail avec Git





Les trois états





Indexer des changements

- ❏ Tout considerer
`git add --all | -A`
- ❏ Par fichier ou sous-dossier
`git add nom_des_fichiers | nom_des_sous-dossies`
- ❏ Fileglob Regex(Expressions régulières)
`git add . | * | *suffixe | prefixe*`



Inspecter le dépôt

- ❏ L'état du dépôt:
`git status`
- ❏ Difference entre références
`git diff`
- ❏ Historiques des commits(versions)
`git log`



Créer une nouvelle version

- ❑ Utilisant l'index
`git commit`
- ❑ Depuis l'index avec message de validation
`git commit -m "message"`
- ❑ En indexant automatiquement les modifications
`git commit -a`



Annuler des changements

- ❑ Annuler un changement introduit par une version ou commit, entraînant la création d'une nouvelle version:
`git revert`
- ❑ Retourner le dossier de travail dans un certain état selon une certaine version:
`git reset`
- ❑ Mettre de côté temporairement les modifications non validées
`git stash`



Ignorer des fichiers et dossiers

- ❑ Ceci est possible grâce au fichier **.gitignore**
- ❑ Peut inclure des commentaires servant de documentation
- ❑ Chaque ligne de text est un chemin relatif à l'élément à ignorer
- ❑ Exemple contenu:

```
# ignorer tout fichier .txt (ceci est un commentaire)  
*.txt
```

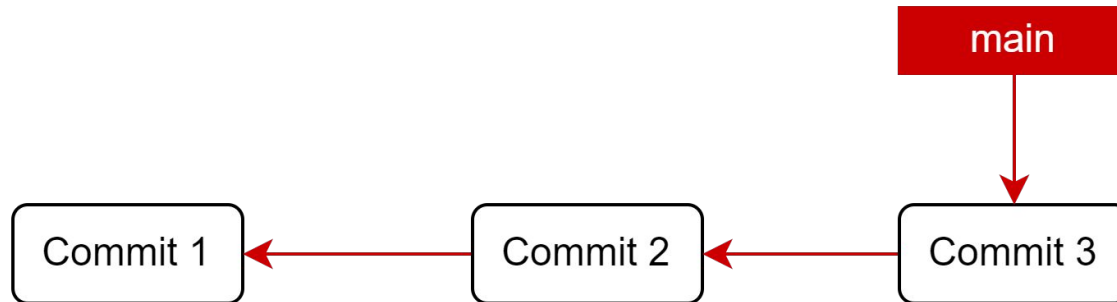
```
# ignorer tout dossier commençant par secret-  
/secret-*
```



Les branches Git



Qu'est ce qu'une branche



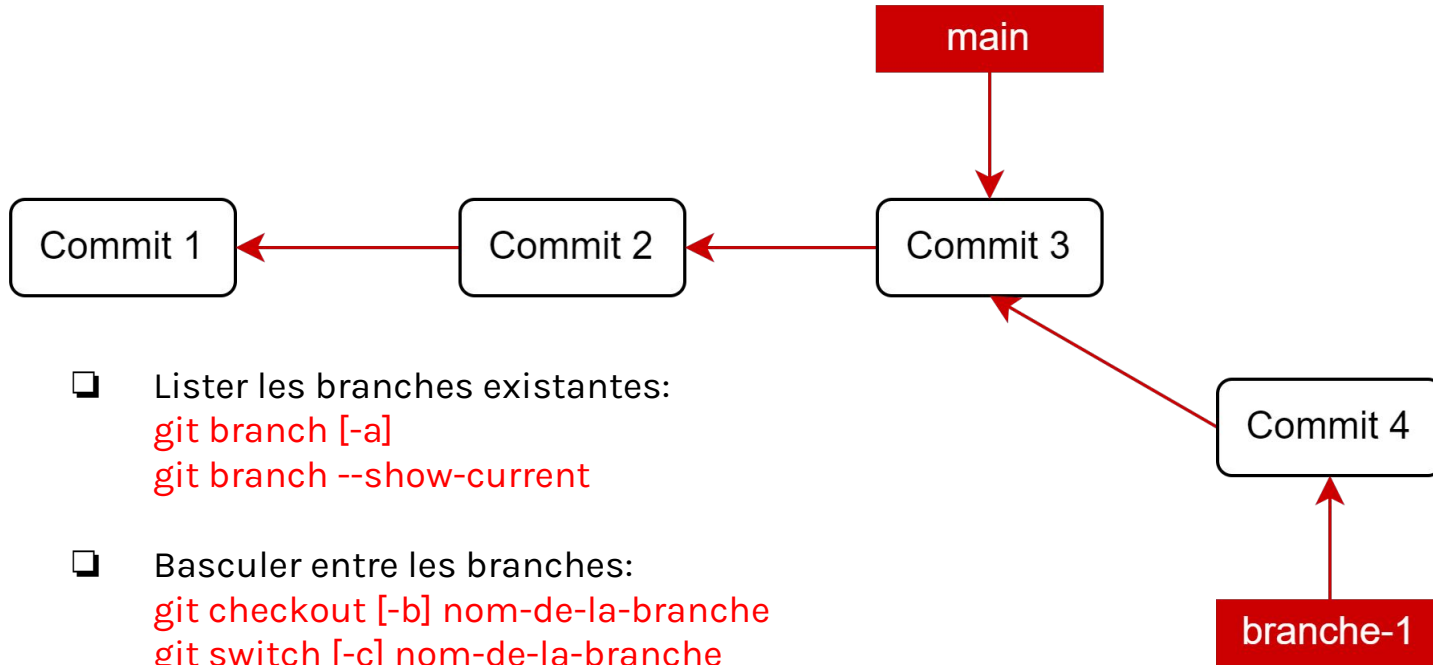


Travailler sur une branche

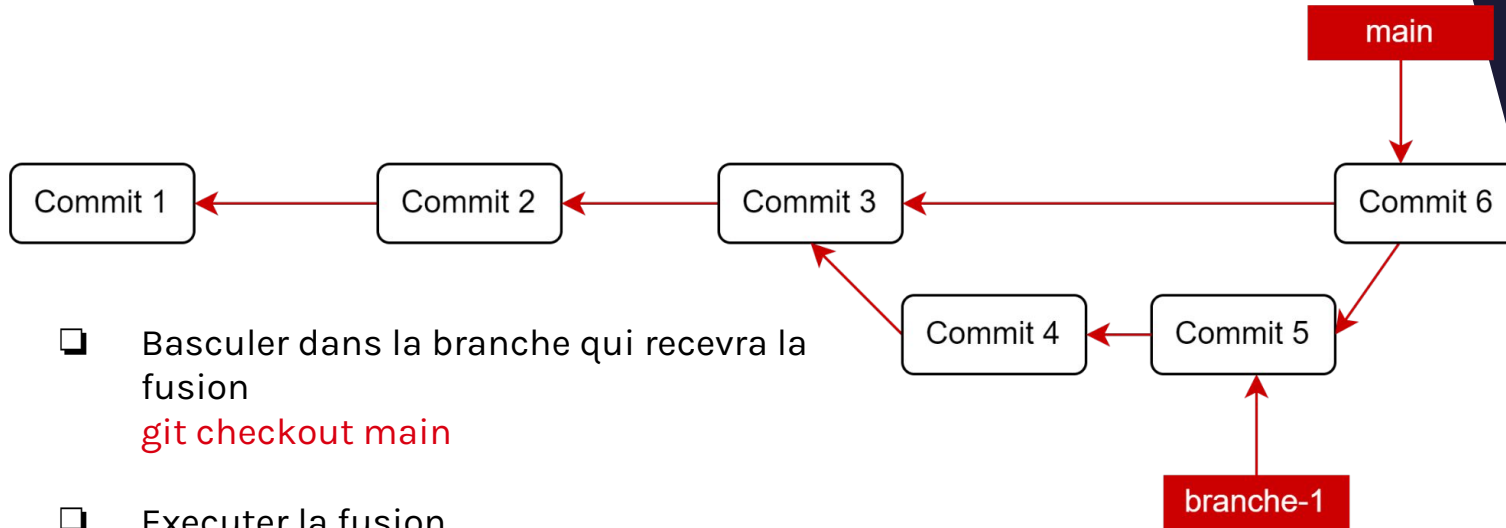
- ❏ Créer une branche
`git branch [nom]`
- ❏ Renommer une branche
`git branch -m [nouveau_nom]`
- ❏ Supprimer une branche
`git branch -D [nom]`



Basculer entre les branches



Fusionner les branches



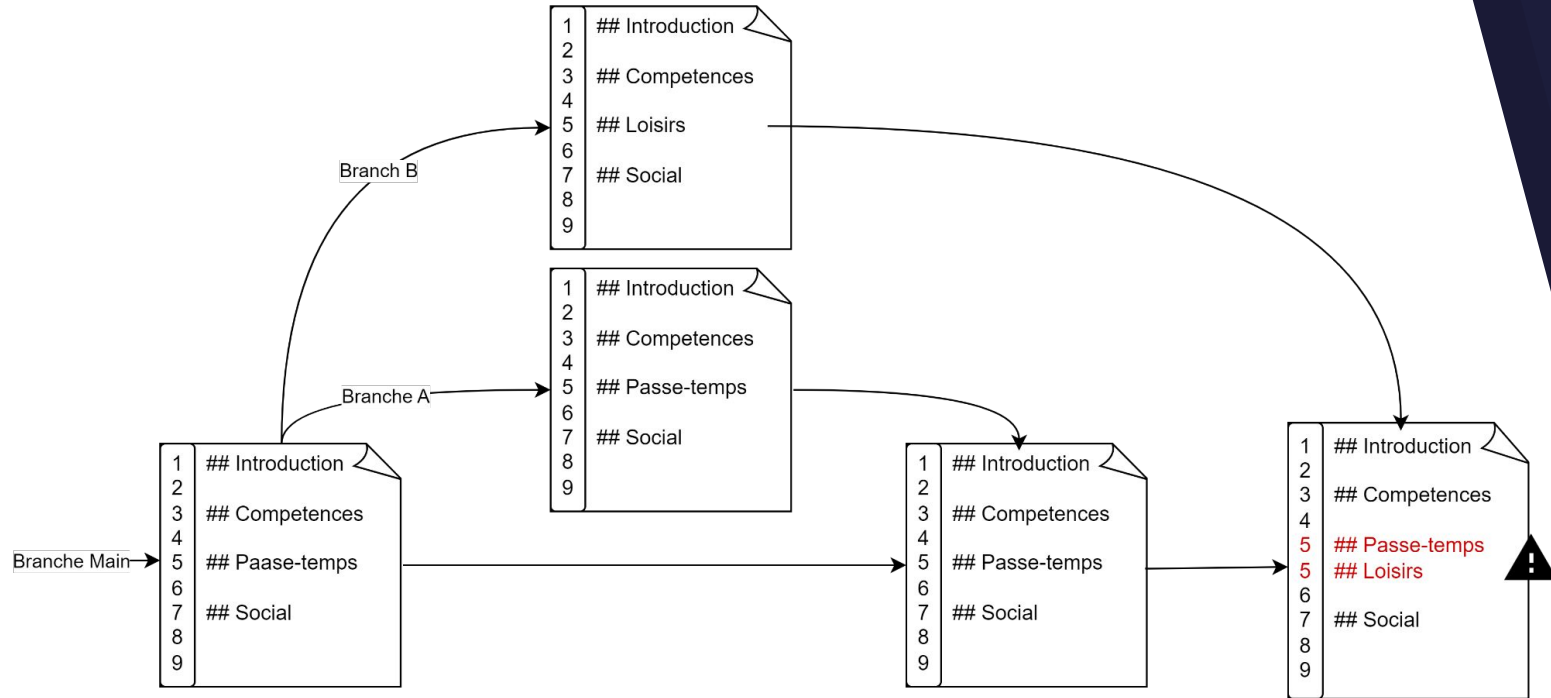
- ❑ Basculer dans la branche qui recevra la fusion

`git checkout main`

- ❑ Executer la fusion

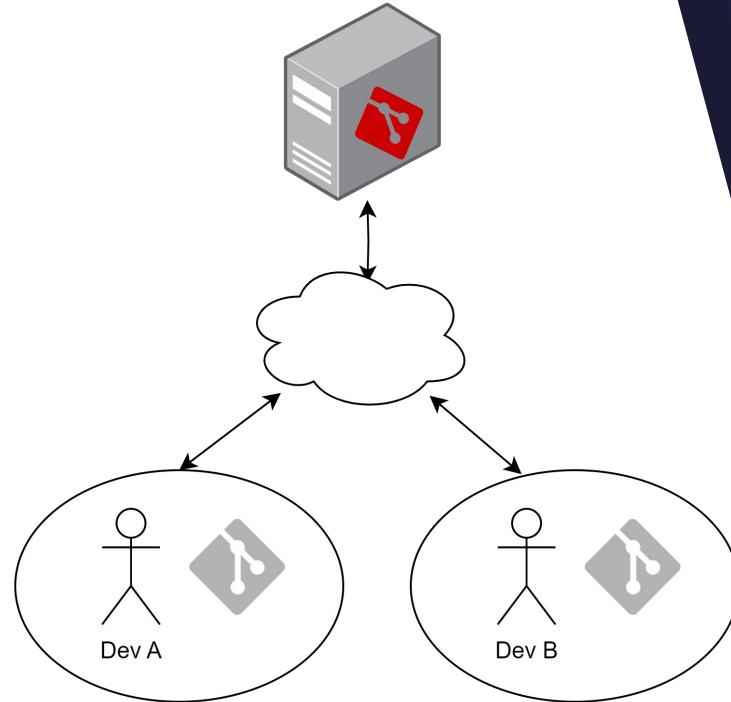
`git merge branche-1`

Conflits lors de la fusion des branches



Les dépôts distants

- ❑ Stockage en ligne
- ❑ Collaboration à distance
- ❑ Synchronisation
- ❑ Suivi des versions
- ❑ Plateforme d'hébergement:
GitHub, Bitbucket, GitLab, etc..





A propos de GitHub

- ❑ Plateforme de développement collaboratif
- ❑ Hébergement de code
- ❑ Contrôle de version
- ❑ Communauté
- ❑ Intégration continue et Déploiement continu (CI/CD)



Créer un compte GitHub

<https://github.com/signup>



Exercice 1: Créer son compte GitHub

- ❑ Naviguer sur <https://github.com/signup> et créer votre compte GitHub
- ❑ Une fois le compte créé, créer un dépôt (spécial) ayant le même nom que votre nom de compte GitHub
- ❑ Modifier le contenu de ce dépôt à votre guise et valider les changements



Travailler avec les dépôts distants

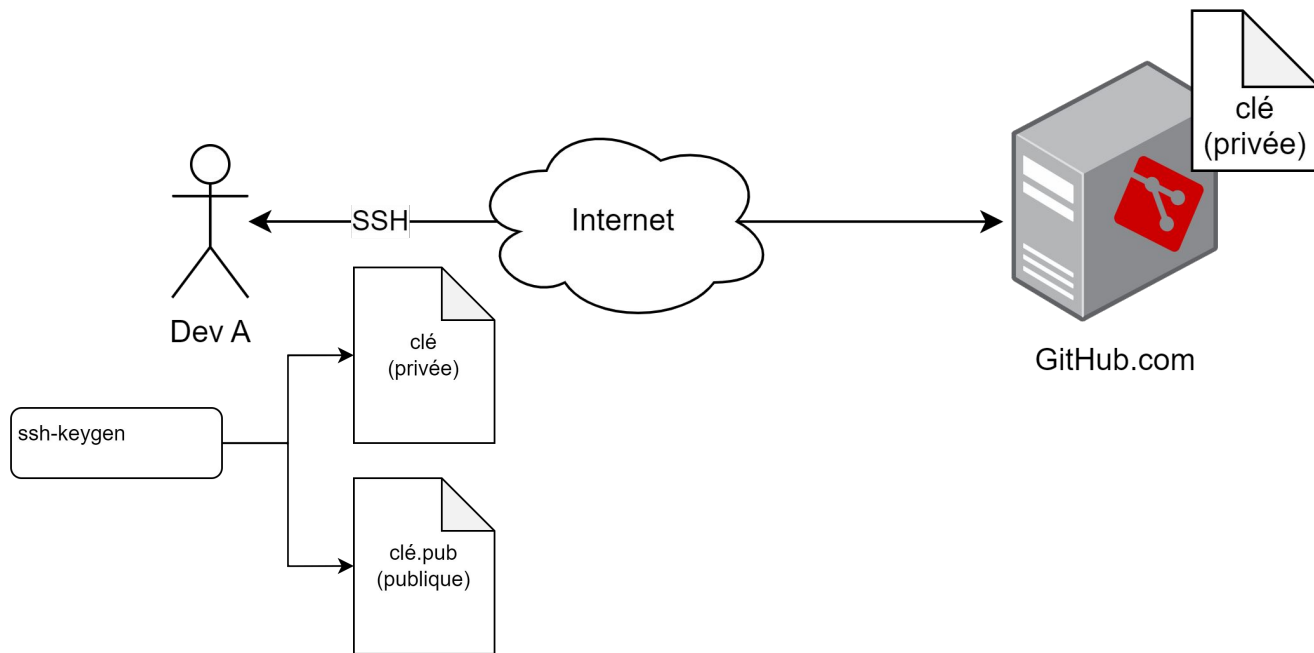
Cloner un dépôt distant

```
git clone url
```

```
cd nom-du-depot/dossier
```



Travailler avec les dépôts distants: l'authentification SSH





Travailler avec les dépôts distants(suite)

Les branches distantes:

`git branch --remote`

Met à jour un dépôt distant

`git push [-u nom branche]`



Exercice 2: Mettre à jour un dépôt distant GitHub

- ❑ Modifier le dépôt cloné lors de l'exercice 3
- ❑ Envoyer les modifications faites vers GitHub



Le GitHub Flow: Introduction

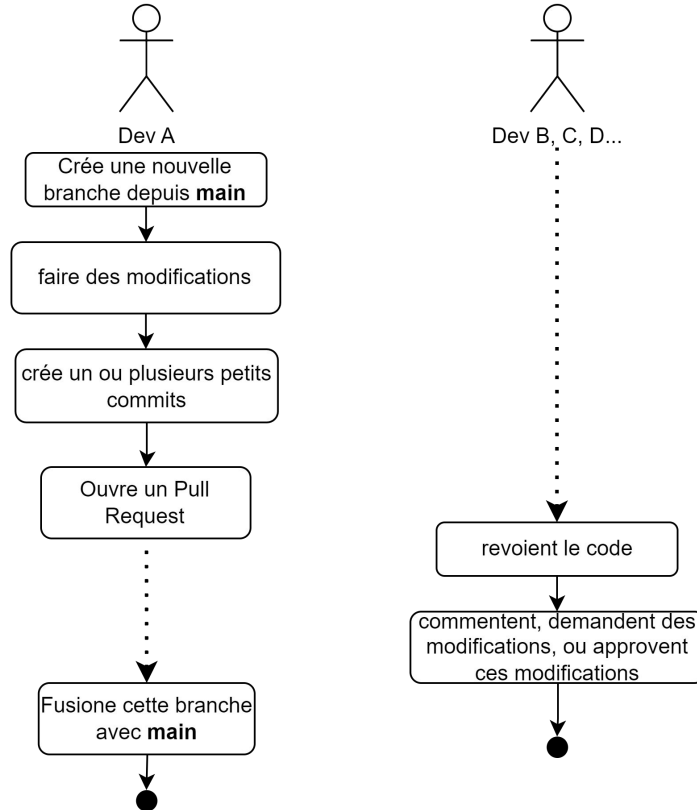
- ❑ Un flux de travail de développement
- ❑ Privilégie la simplicité, la collaboration et la livraison continue
- ❑ Pas Adapté à Tous les Projets, voir GitFlow Workflow



Le GitHub Flow: Principes clés

- ❑ Branche **main** ou **master** comme Production
- ❑ Branches de Fonctionnalités
- ❑ Commits Petits et Fréquents
- ❑ Pull Requests(Demandes de Tirage) et Révision de Code
- ❑ Intégration Continue et Déploiement après Fusion

Le GitHub Flow: Concretement





Exercice 3: Tirage sur GitHub

- ❑ Modifier le dépôt cloné lors de l'exercice 3
- ❑ Envoyer les modifications faites vers GitHub



La Collaboration sur GitHub

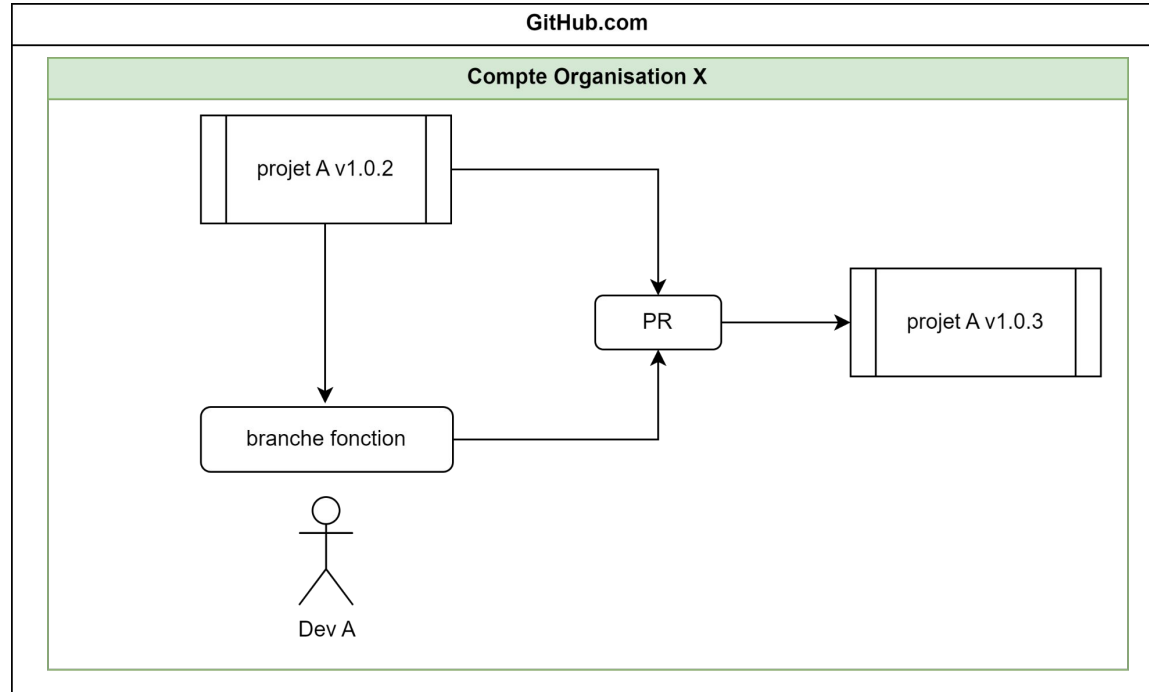
- ❑ **Suivre les directives du projet est requis:**

- ❑ Licence
- ❑ Lignes directrices de contribution

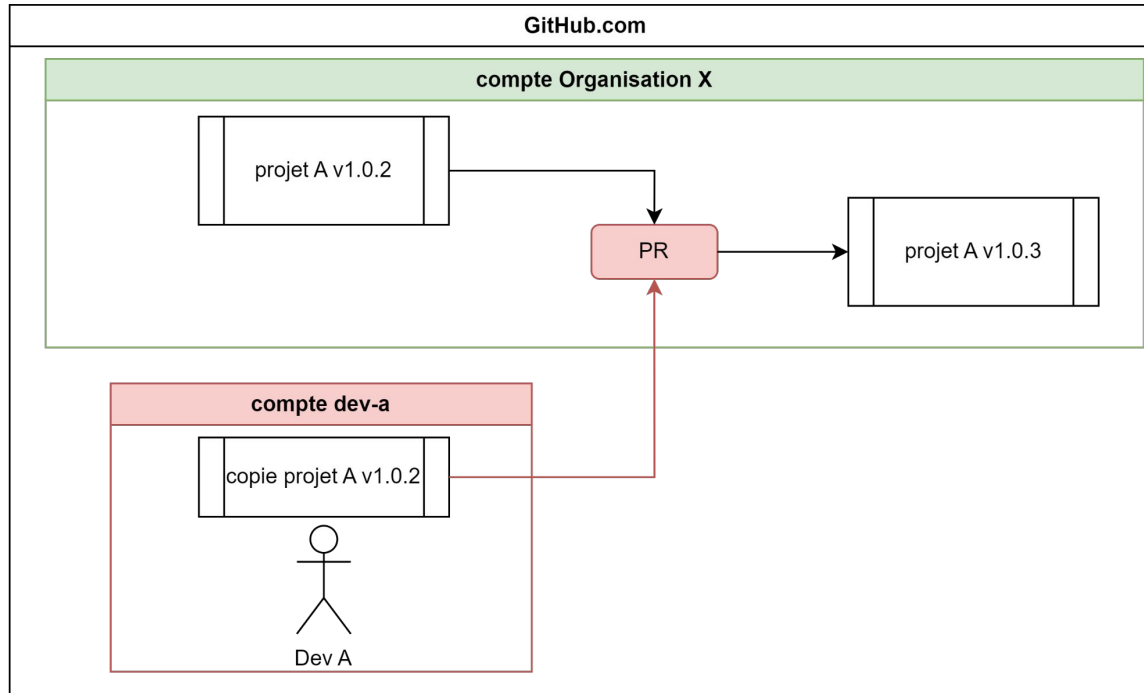
- ❑ **Deux modèles:**

- ❑ Modèle de dépôt partagé
- ❑ Modèle de Fork et Pull

Modèle de dépôt partagé



Modèle de Fork et Pull





Le Forking(suite)

- ❑ Une copie personnelle d'un dépôt public
- ❑ Les modifications faites n'affectent pas l'original
- ❑ Permet de collaborer sur le dépôt original au travers des PRs
- ❑ Synchronization bi-directionnelles



Démo: Forking

<https://github.com/signup>



Quelques bonnes pratiques

- ❑ Toujours vérifier le dossier de travail dans la ligne de commande
- ❑ Commiter régulièrement et pousser les changements vers le dépôt distant
- ❑ Utiliser des stratégies de branching comme GitHub Flow
- ❑ Documenter vos projets en utilisant des noms explicites (branches, messages des commits)
- ❑ Créer et maintenir un fichier **.gitignore**
- ❑ Être prudent avec les données privées
- ❑ Respecter les directrices de contribution



Exercice 4: Collaboration

- ❑ Forker le dépôt
<https://github.com/2023-10-04-POE-JAVA-NIORT/git-et-github>
- ❑ Modifier le pour inclure vos details sur la liste des participants dans le fichier README.md
- ❑ Créer une PR pour mettre à jour le dépôt original



Ressources

- ❑ Site officiel Git: <https://git-scm.com/>
- ❑ Documentation officielle Git (FR):
<https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-%C3%80-propos-de-la-gestion-de-version>
- ❑ Bien démarrer avec la documentation GitHub
<https://docs.github.com/fr/get-started>
- ❑ Syntaxe de base pour l'écriture et la mise en forme(GitHub)
<https://shrturl.app/Zlz8hZ>
- ❑ Collaboration à l'aide de demandes de tirage(GitHub)
<https://docs.github.com/fr/pull-requests/collaborating-with-pull-requests>