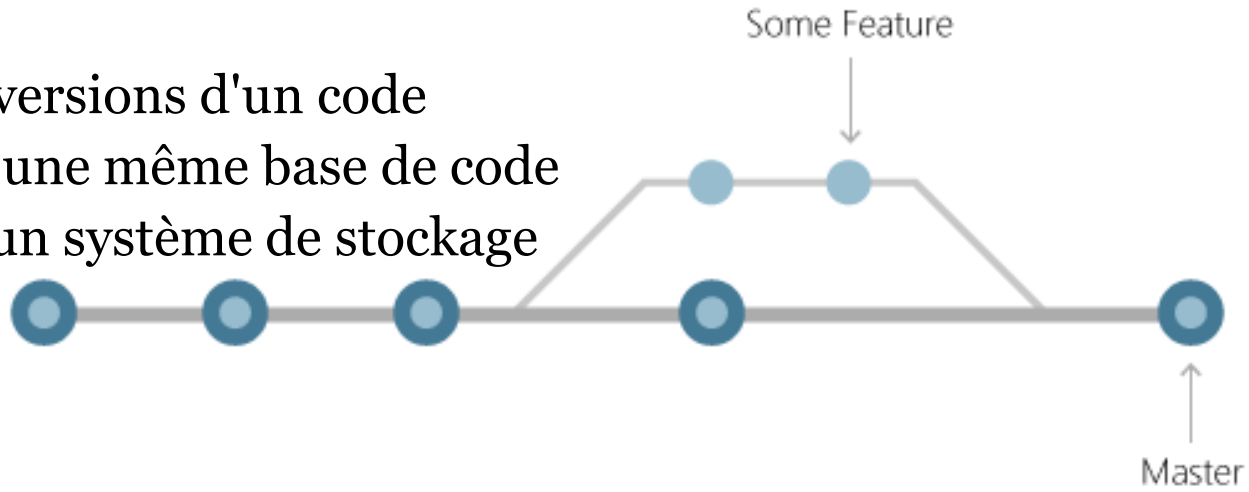
INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Git & Gitlab

Les fondamentaux de Git

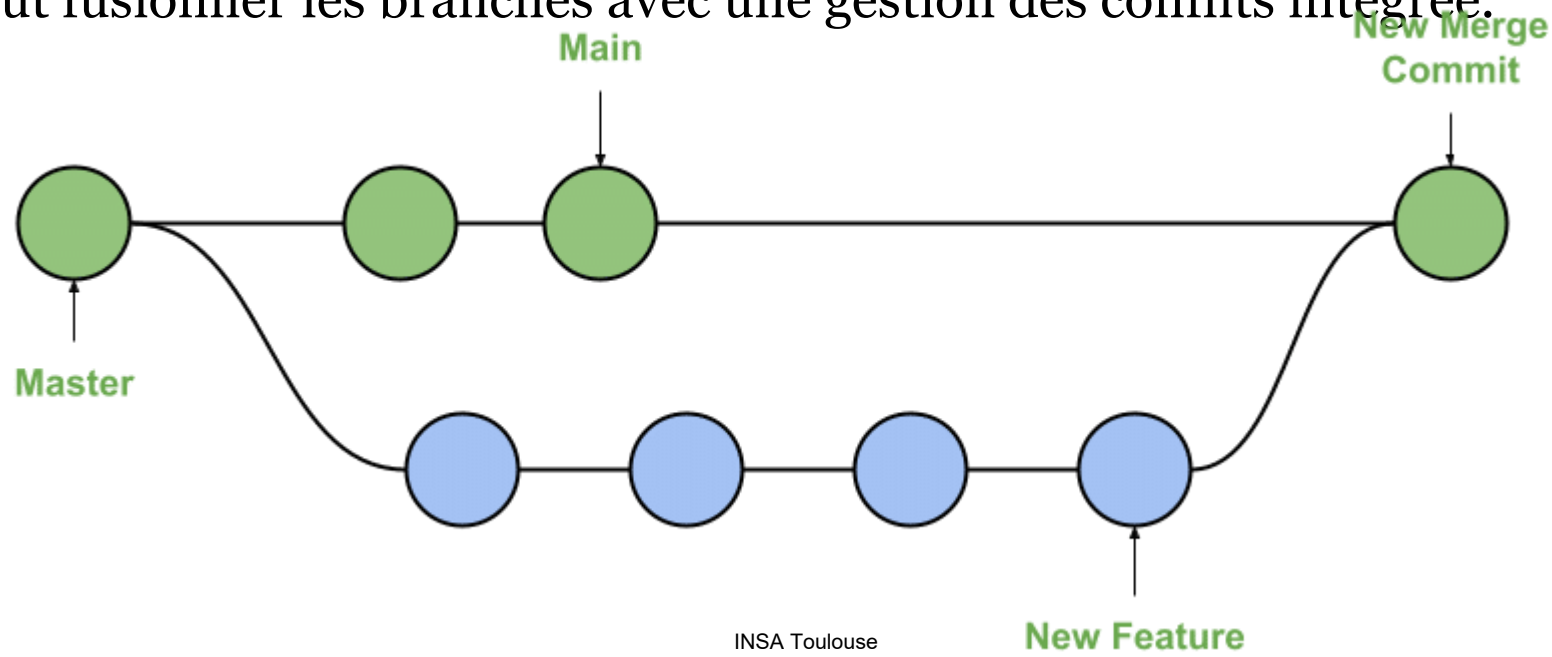
- Qu'est-ce que git ?
  - C'est un outil de gestion de version d'arborescences
  - Git vs Github, Gitlab

- Pourquoi l'utiliser ?
  - Tracer les modifications
  - Travailler sur plusieurs versions d'un code
  - Travailler en équipe sur une même base de code
  - Attention, ce N'est PAS un système de stockage



Les fondamentaux de Git

- Comment ça marche ?
  - Chaque modification est appelée un "commit", tout est versionné,
    - Chaque commit est identifié par un hash, ex: cfe47ffb
  - Git peut gérer plusieurs "histoires" (branches) parallèles, correspondant à N développeurs, ou N fonctionnalités en cours de développement,
  - Git peut fusionner les branches avec une gestion des conflits intégrée.

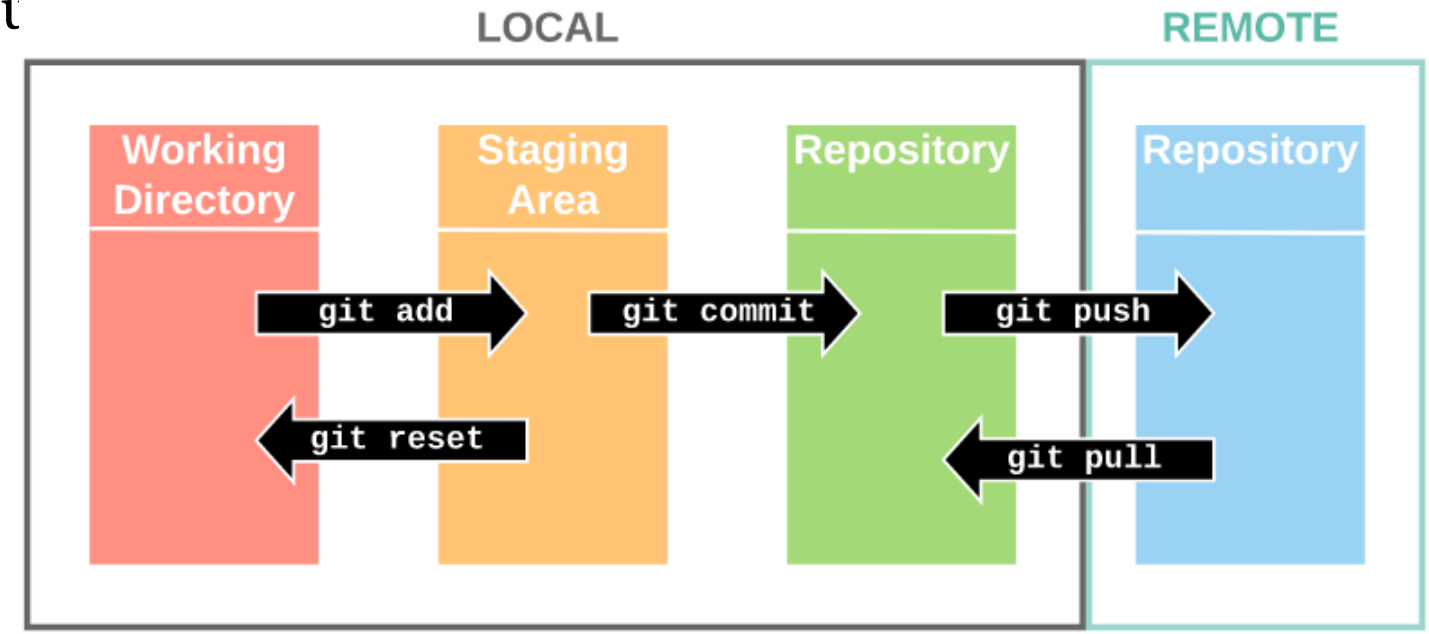


Sur votre poste

- Debian / Ubuntu : apt install git
- Windows: git for windows, TortoiseGit (dans self-service WAPT)
- Intégré dans VSCode (probablement le plus simple)

Commandes pratiques à maîtriser

- git status : Vérifier l'état des fichiers (modifiés, ajoutés, non suivis).
- git diff: Voir les différences par lignes avant mise dans la staging area
- git add <fichier ou dossier> : Ajouter les modifications à la zone de transit.
- git commit -m "Ajout d'u ons
  dans l'historique local.



Exemple

- Norbert développe son application Java
  - Il corrige régulièrement des bugs sur la version plus récente de son application
  - En parallèle il développe une nouvelle fonctionnalité

  - Deux histoires pour deux branches ! Une fois la fonctionnalité en développement terminée, il peut "merger" dans l'histoire principale

  - git checkout –b devNewFeature
    <développement>
    git add NewFeature.java
    git commit -m "Ajout nouvelle fonctionnalité"

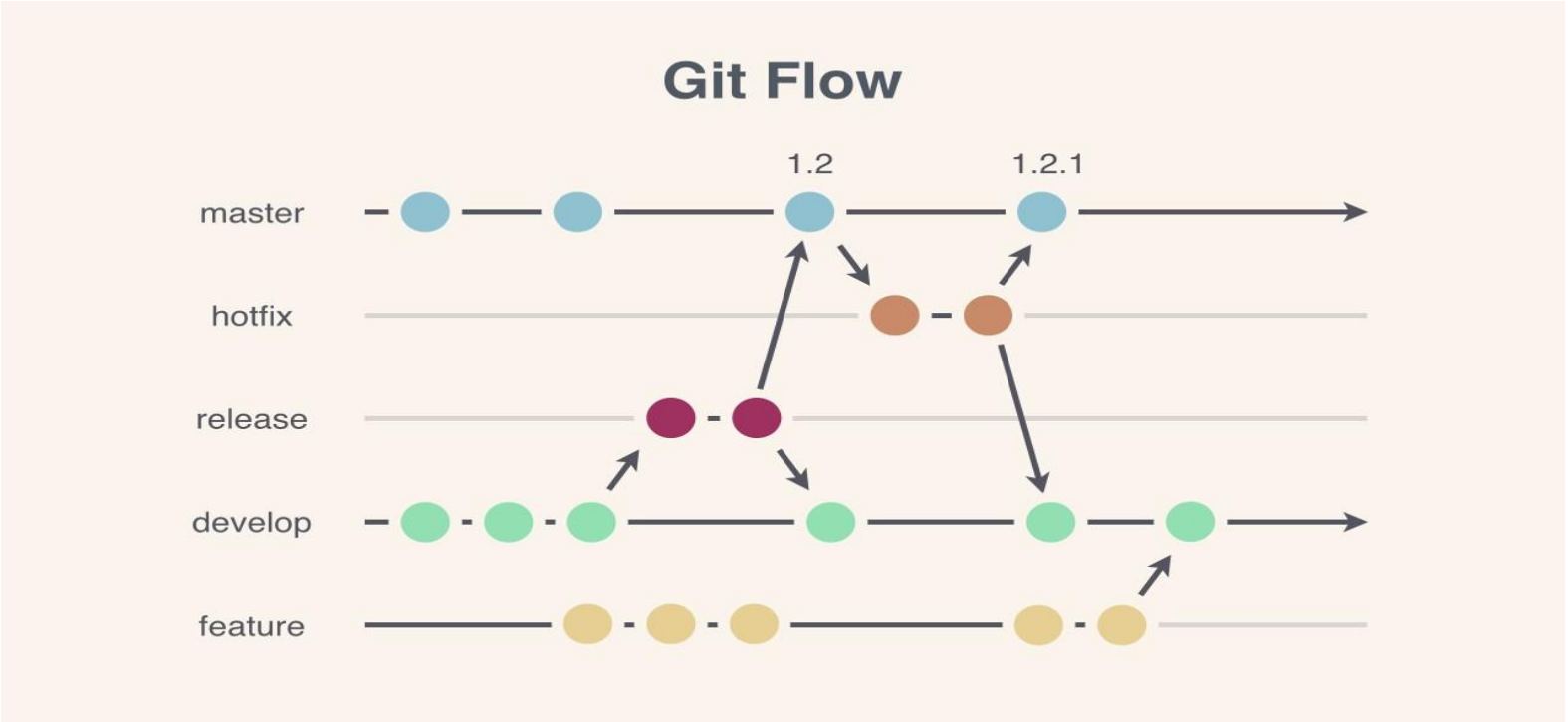
    git checkout main
    git merge devNewFeature

Commandes à maîtriser : gestion des branches

git branch : Lister les branches existantes.

git checkout -b <nom> : Créer et basculer sur une nouvelle branche.

git merge <nom-branche> : Fusionner une branche dans la branche actuelle.



Travailler avec un dépôt distant

- https://gitlab-rech.insa-toulouse.fr/

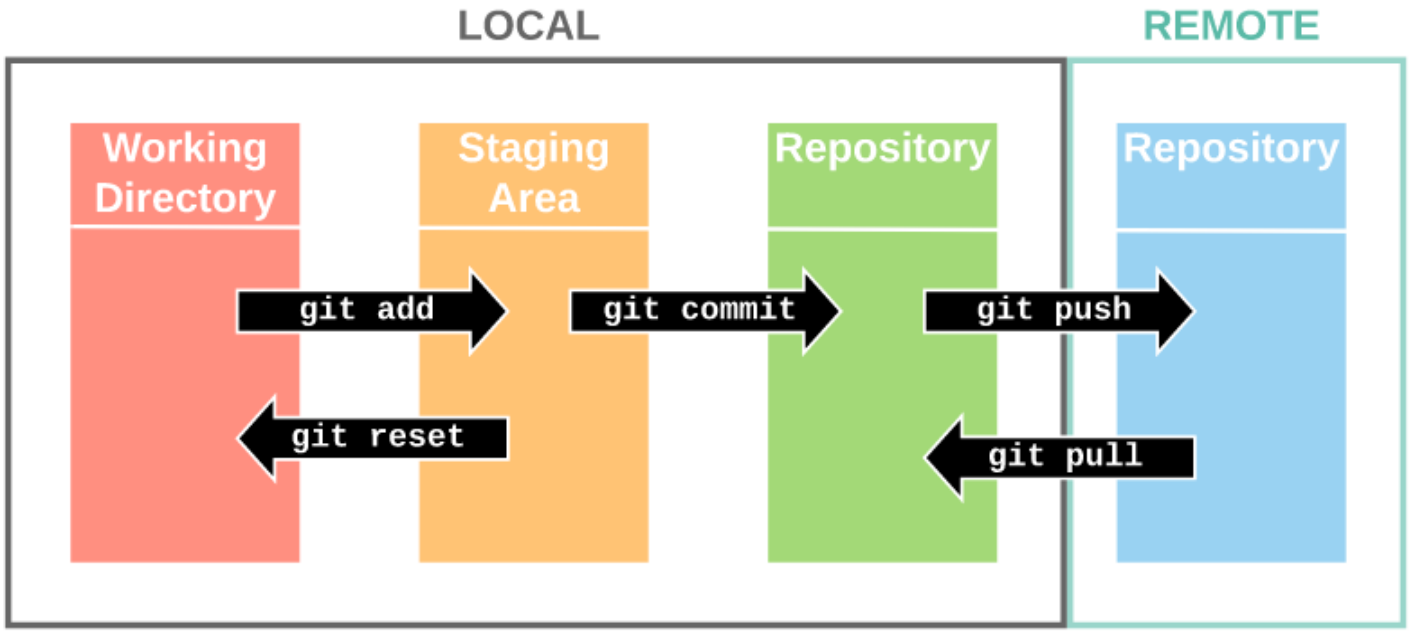
- L'INSA fournit un dépôt pour la recherche.
  - Basé sur la plateforme Gitlab
  - Intègre un editeur en ligne (basé sur VSCode)
  - Possibilité de mettre en place de l'intégration continue (tests et déploiement automatique)

Commandes à maîtriser : travail avec un dépôt distant

git clone <url dépôt> : Récupérer un dépôt distant en local

git push origin <nom-branche> : Envoyer sa branche locale vers GitHub/GitLab.

git pull : Récupérer et fusionner les dernières modifications du serveur distant.



Travailler sur un dépôt distant

- Plusieurs manières de communiquer avec le dépôt:
  - Via SSH, utilisation d'une paire de clé : pas de mot de passe (clé publique sur le serveur)
  - Via HTTPS, authentification utilisateur mot de passe régulièrement demandé
  - En ligne avec WebID

- Gestion des conflits:
  - Git résout pour vous et de manière transparente la plupart des conflits
  - Sinon c'est à vous de résoudre !

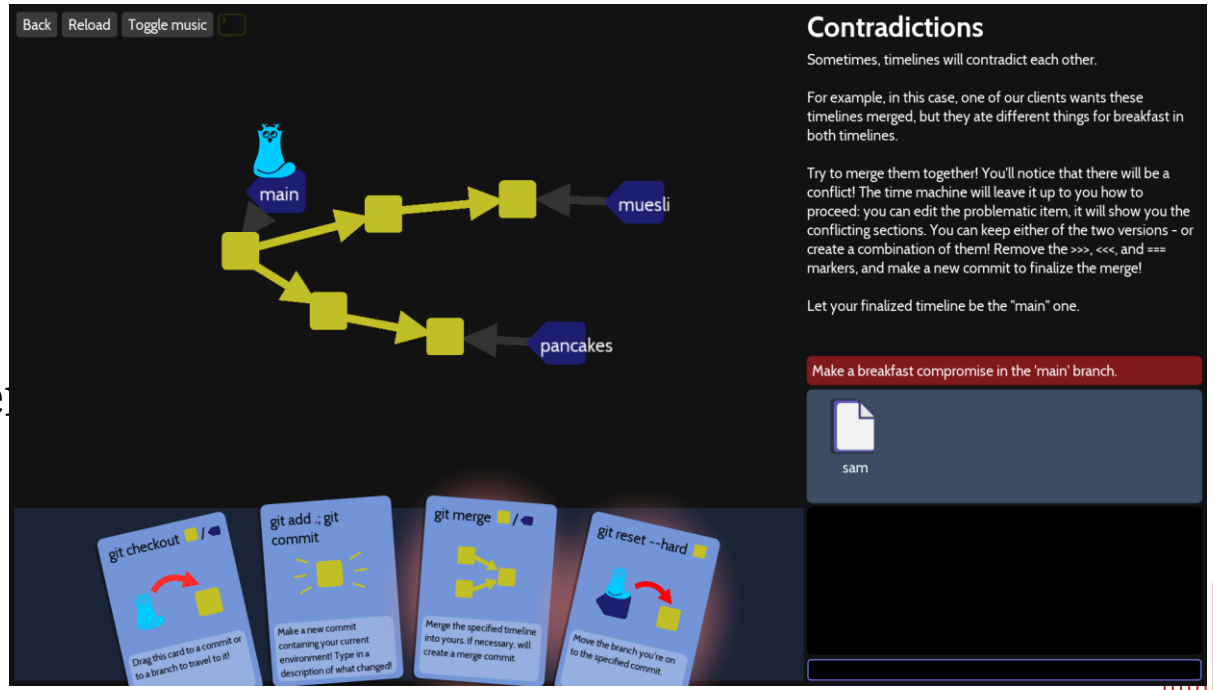
Autres petites astuces et vocabulaire

- Un commit = une modification consistante, avec un intitulé
- merge request
- Tags : nommer un "commit" (ex: v1.0, v2.1, et...)
- Ne pas mettre les dépendances dans le dépôt, uniquement les sources => .gitignore (venv pas dans le dépôt par ex)
- git blame => c'est à qui la faute ?
- git stash => mettre de côté des modifications temporairement

Quelques outils pour apprendre

- Oh My Git ! Sous forme de "jeu", avec des scénarios de gestion de branches simple à complexes
- Cours de Laurent Grangeau:

https://cours-ecetech.github.io/git/#/



Cheatsheet, git flow

https://gist.github.com/JamesMGree

Collaboration Gitlab : cas pratique sur la plateforme Gitlab Recherche de l'établissement

Générer votre clé SSH

Vous allez créer deux clés : une clé **privée** (qui reste secrète sur votre ordinateur) et une clé **publique** (que vous allez donner à GitLab).

- Ouvrez votre terminal (ou Git Bash sur Windows).
- Tapez la commande suivante en remplaçant l'email par celui de votre compte GitLab :
- ssh-keygen "votre_email@exemple.com"
- Appuyez sur **Entrée** à toutes les questions (ne pas mettre de *passphrase* si vous voulez que ce soit 100% automatique) .Étape 2 : Ajouter la clé publique sur GitLab (La serrure)

Il faut maintenant copier le contenu de votre clé publique pour la coller dans les paramètres de GitLab.

- Affichez votre clé publique dans le terminal avec cette commande :
- cat ~/.ssh/id_ed25519.pub
- Copiez tout le texte qui s'affiche (qui commence par ssh-ed25519).
- Allez sur GitLab, cliquez sur votre **Avatar** en haut à gauche > **Edit profile** (Éditer le profil) > **SSH Keys** (dans le menu de gauche).
- Cliquez sur **Add new key**, collez votre clé dans le grand champ "Key", donnez-lui un nom (ex: "Mon PC Portable") et cliquez sur **Add key**.

Pour vérifier que ça fonctionne, tapez ssh -T git@gitlab.com dans votre terminal. Si GitLab vous dit "Welcome to GitLab", c'est gagné !

Ajout et travail avec le dépôt distant

- Ajout du dépôt distant:

git remote add origin

- Publication des changements sur le dépôt

git push -u origin main

MERCI