Découverte de GIT

C.BENSARI

Introduction

- Un projet est un ensemble de fichiers (*.php, *.html, *.sql, ..)
- En développement informatique nous sommes amenés à développer en équipe et chaque développeur est amené à modifier des fichiers du projet que porte l'équipe
- Il est nécessaire d'avoir l'historique des modifications d'un fichier
- Il est nécessaire de pouvoir gérer les modifications de plusieurs personnes sur le même fichier (gestion de conflits)
- **Git** est un outil de versionning qui va s'occuper de gérer l'historique des modifications et les conflits et encore plus !
- Il existe d'autres outils de versionning comme SVN, CVS, ..

À Quoi sert GIT ?

- Tracer l'historique des modifications des fichiers
- Revenir à une version antérieure d'un fichier
- Récupérer les modifications faites par d'autres personnes tierces (PULL)
- Envoyer les modifications (PUSH)
- Créer des branches pour des fonctionnalités spécifiques et avoir différentes version du projet
- Gestion des conflits

Installation de GIT

- Aller à l'url https://git-scm.com
- Télécharger et installer Git
- Lancer un terminal **git bash** et vérifier si Git a bien été installé grâce à la commande: **git --version**
- Configurer Git en lui fournissant le nom de la personne qui va l'utiliser sur cette machine:

git config --global user.name "nom_personne"

Utiliser Git

- Créer un répertoire avec le nom de votre choix (éviter les caractères spéciaux et les espacer)
  - Sur le terminal avec la commande: **mkdir** nom_repertoire
- Se déplacer dans le répertoire (commande **cd** nom_repertoire) créé et exécuter la commande **git init** : permet d'initialiser le dossier sous Git
- Exécuter la commande **git status**

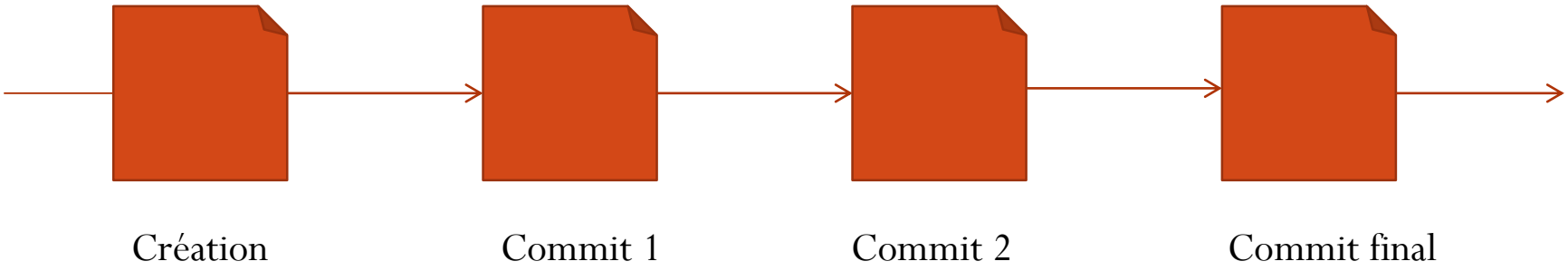
  Nothing to commit => répertoire vide
- Créer un fichier HTML/PHP dans le répertoire avec un contenu basique
- Exécuter la commande **git add** nom_fichier : permet d'ajouter le fichier dans le système Git
- Exécuter la commande **git commit –m** ``ma modif porte sur ..`` : permet d'enregistrer l'état des modifications dans le système Git

Utiliser Git

- Modifier le fichier HTML ensuite faire la même opération

(**git add** .. ensuite **git commit** ..)

- Exécuter la commande **git log** : permet d'afficher l'historique des modifications du répertoire
- Workflow (Cycle de vie) d'un fichier sur Git



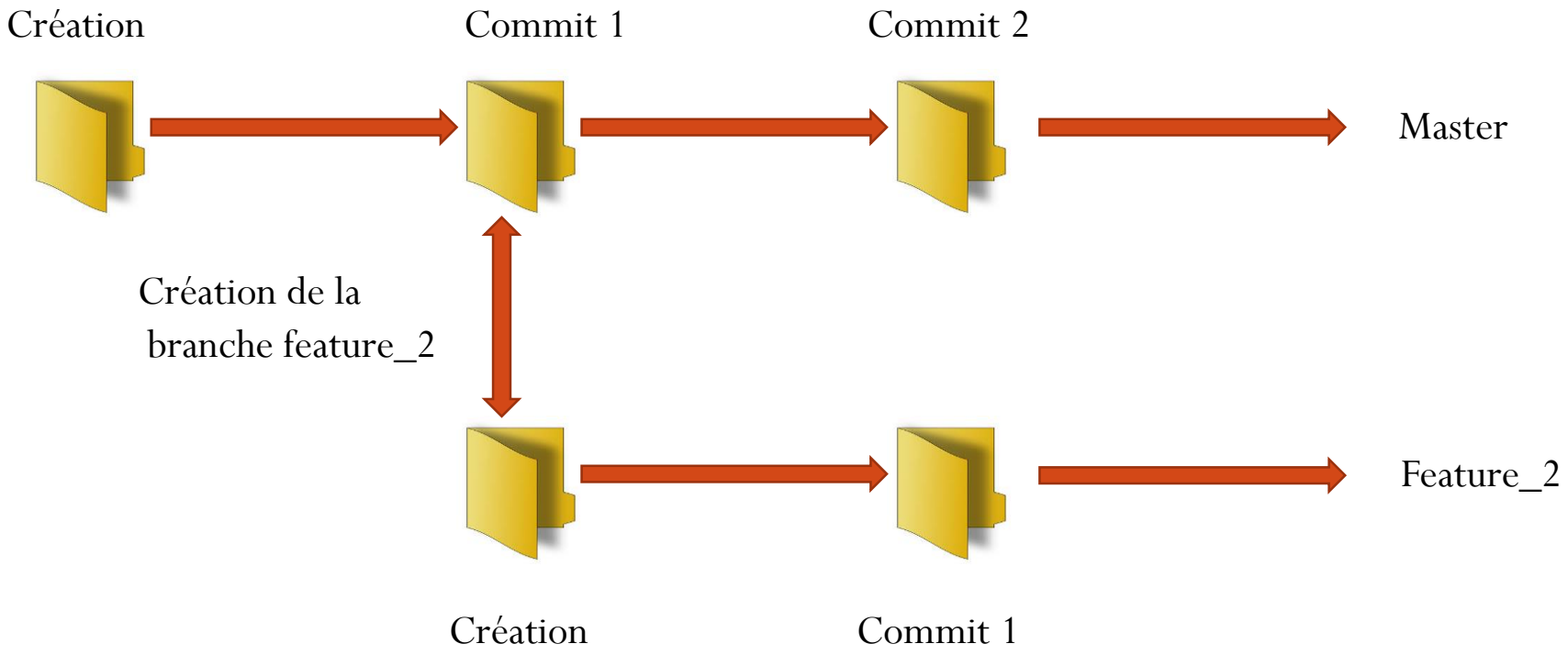
Gestion des branches

- Une branche est une copie du projet qui peut être gérée indépendamment de la branche principale
- La branche principale d'un projet sous Git s'appelle **master**
- Il est possible avec Git de créer d'autres branches qui peuvent être utilisées pour des évolutions du projet ou des correctifs d'une version de l'application déjà en Production
- Une branche peut être crée à partir de n'importe quelle version

Gestion de branches

- Pour créer une nouvelle branche, exécuter la commande:

  git branch nom_nouvelle_branche

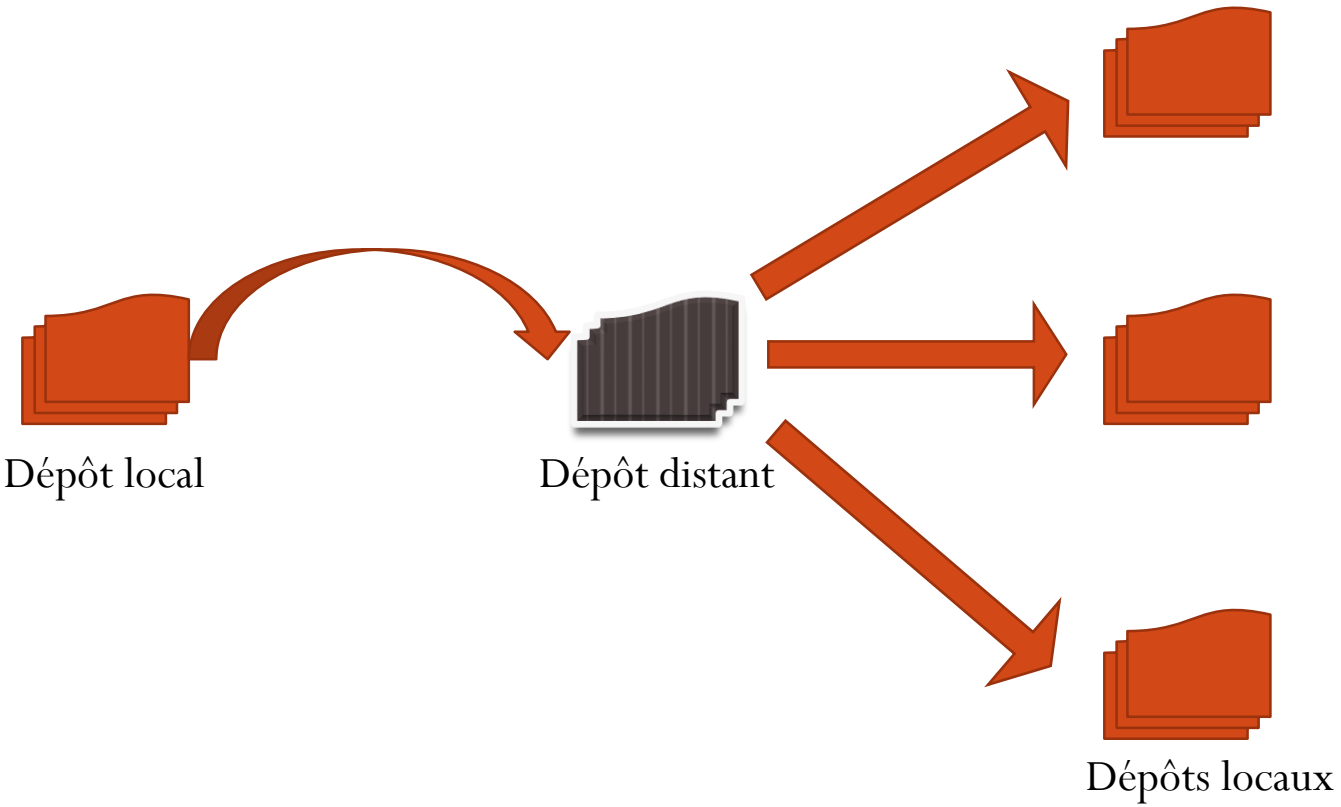


Gestion des branches

- Pour lister la liste des branches, utiliser la commande **git branch**
  - Lorsque Git liste les branches, il affichera la branche master avec un couleur différente et mettra un astérisque devant la branche actuelle
- Pour basculer d'une branche à une autre, utiliser la commande **git checkout** nom_branche. A partir de ce moment on peut faire des modifications et des commits sur cette branche
- Lors de l'exécution de la commande **git log** sur la nouvelle branche, Git affichera le même historique que celui de la branche initiale (Master)
- Pour rapatrier les modifications sur la branche principale (Master) :
  - Se positionner sur la branche Master (git checkout master)
  - Exécuter la commande : **git merge** feature_2
  - Supprimer (facultatif) la branche feature_2 avec la commande **git branch –d** feature_2

Et pour le travail d'équipe ?

- Avec Git, il est possible de partager un projet dans un dépôt distant où il peut être récupéré par d'autres collaborateurs



Le dépôt distant: GitHub

- Pour déposer un projet dans un dépôt distant, il faut créer un compte utilisateur sur le site https://github.com
- Créer un nouveau repository (dépôt) avec un nom court et mémorisable
- Pour créer un lien (pont) entre notre dépôt distant et le dépôt local, utiliser la commande:

  git remote add origin url_vers_depot_distant
- On peut vérifier à tout moment les dépôts distants du projet en utilisant la commande **git remote**.
- Une fois le lien établit, il sera possible d'envoyer le contenu du dépôt local vers le dépôt distant avec la commande

  git push origin branche_courente(master)

GitHub: gestion d'accès

- Afin d'autoriser d'autres personnes à modifier les fichiers du projet partagé, il faut suivre les étapes suivantes :
  - Aller sur l'onglet « settings » de la page du dépôt distant
  - Sur le menu « collaborateurs », saisir le nom d'utilisateur du collaborateur dans le champs input et cliquer sur le bouton « add collaborator »
- Pour récupérer le contenu du dépôt distant, le collaborateur peut:
  - Créer son dépôt local et créer un pont avec **git remote add** origin url_depot_distant et ensuite exécuter la commande **git pull --rebase** origin master afin de récupérer le contenu du dépôt distant (cette commande doit être utilisée par chaque collaborateur afin d'avoir une version à jour du projet)
- Les collaborateurs peuvent rapatrier leur modification sur le dépôt distant avec la commande **git push** origin master