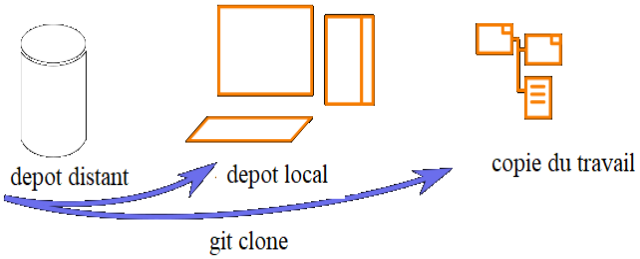
Université Mohammed V de Rabat
Ecole Nationale Supérieure d'Arts et Métiers
Filière INDIA (3eme année) A.U. : 2025-2026 Pr. A. El Qadi

TP n°1 en DevOps

Gestion de version : Git/GitHub

1. Installer le logiciel **Github Desktop** (https://desktop.github.com/download/) qui contient le logiciel **git**.
   Chaque personne se connecte à son compte Github.
   Configurer : nom d'utilisateur et l'adresse e-mail dans GitHub : https://github.com/nom_du_compte
2. Création d'un dépôt local

Un dépôt, ou repo, sera le centre du projet. Il peut s'agir d'un fichier ou d'une collection de fichiers contenant du code, des images, du texte ou tout autre élément.



Ex1. Gestion de projet avec Git et GitHub

1. Créer un nouveau dépôt sur GitHub (ex. : my_first_depot)
   - Ouvrir un navigateur web et aller sur GitHub
   - Connecter à votre compte (https://github.com/nom_du_compte).
   - Cliquer sur le bouton **"New"** ou **"Créer un dépôt"**.
   - Remplir le formulaire :
     - **Nom du dépôt** :my_first_depot
     - **Description** : (facultatif) Ajouter une description à ce projet.
     - **Public/Privé** : Choisir si le dépôt sera public ou privé.
     - **Ne pas initialiser avec un README** (décochez l'option).
2. Créer un nouveau répertoire sur votre machine : tp_DevOps
3. Cloner le dépôt « my_fisrt_depot » sur votre machine locale (tp_DevOps).
4. Afficher la version de git
5. Créer une branche de fonctionnalité nommée demo
6. Dans le dossier local, créer un fichier demo.txt.
   `echo "# My first git" > demo.txt`
7. Ajouter ce fichier à Git et faire un premier commit
8. Pousser la branche demo vers GitHub
9. Sur GitHub, créer une pull request depuis demo vers main :
   - Aller dans l'onglet Pull requests,
   - Cliquer sur New pull request,
   - Sélectionner develop comme branche source,
   - Ajouter un titre et une description, puis créer la PR.
10. Mettre à jour votre copie locale régulièrement avec git pull.
11. Explorer l'historique et vérifier l'état des fichiers

Ex2. Travailler avec des branches

Git permet de créer des branches de fonctionnalités : chaque développeur peut travailler de façon isolée sans impacter la branche principale (main). Une fois les modifications validées, la branche peut être fusionnée (merge) dans main

1. Utiliser GitHub
   a. Créer un nouveau dépôt sur GitHub (**sans README**).
   - **Nom du dépôt** :my_second_depot

   b. Lier le dépôt local au dépôt GitHub.
   `git remote add origin https://github.com/nom_du_compte/my_second_depot`
   c. Pousser (commit) vers GitHub.

2. Gérer les conflits
   a. Créer deux branches distinctes à partir de la branche principale, et modifier le même fichier dans les deux branches et fais un commit dans chacune.
      - Retourner à la branche principale
      - Créer la première branche « branche1 » et basculer dessus
      - Créer un fichier devops.txt et ajouter du contenu au fichier.

      ```
      echo "Ceci est un fichier pour la branche branche1." > devops.txt
      ```

      - Effectuer un commit.
      - Modifier un fichier (par exemple, devops.txt), puis effectuer un commit :

      ```
      echo "Modification dans branche1." >> devops.txt
      git add devops.txt
      git commit -m "Modification dans branche1"
      ```

      - Retourner à la branche principale
      - Créer la deuxième branche et basculer dessus
      - Modifier le même fichier (par exemple, devops.txt) de manière différente, puis effectuer un commit :

      ```
      echo "Modification dans branche2." >> devops.txt
      git add devops.txt
      git commit -m "Modification dans branche2"
      ```

   b. Fusionner les branches dans la branche principale et résoudre les conflits.
      - Retourner à la branche principale
      - Fusionner la première branche (branche1) dans la branche principale :
      - Fusionner la deuxième Branche

      À ce stade, un message de conflit, car les deux branches ont modifié le même fichier (devops.txt).
   c. Résoudre les Conflits
      - Ouvrir le fichier en conflit (par exemple, devops.txt) dans votre éditeur de texte. Vous verrez des sections marquées avec

      ```
      "Ceci est un fichier pour la branche branche1."
      <<<<<<< HEAD
      "Modification dans branche1."
      =======
      "Modification dans branche2."
      >>>>>>> branche2
      ```

      - Résoudre le conflit en choisissant quelle modification garder ou en combinant les deux. Par exemple :

      ```
      Modification dans branche1.
      Modification dans branche2.
      ```

      - Ajouter le fichier à l'index
      - Effectuer un commit pour finaliser la fusion :

      ```
      git commit -m "Résolution des conflits entre branche1 et branche2"
      ```

Ex3. GitHub et collaboration entre les membres des équipes

1. Chaque groupe réalise les étapes suivantes, en utilisant le compte d'**un seul membre du groupe**.
   a. se connecter au site github.com et s'identifier (sign in)
   b. Créer un nouveau dépôt, en prenant soin :
      - Nom de dépôt : **INDIA**
      - Avec un fichier **README.md**
      - Choisir la licence GPLv3 ou BSD (au choix)
      - Choisir une **visibilité privée**
   c. Ajouter les comptes des autres membres du groupe à ce dépôt
      - Aller dans l'onglet "Settings" (paramètres) de votre nouveau dépôt.
      - Dans le menu latéral, cliquer sur "Manage access" (gérer l'accès).
      - Cliquer sur le bouton "Invite teams or people" (inviter des équipes ou des personnes).
      - Entrer les noms d'utilisateur GitHub des autres membres du groupe.
      - Cliquer sur "Add" pour les inviter.
   d. Chacun des autres membres valide cet ajout en se connectant à son compte.

2. Création d'une copie locale
   a. Faire une copie locale du dépôt créé précédemment avec le bouton **Clone repository**, choisir un dossier dans votre répertoire personnel « my_third_depot », qui sera le dossier contenant tous les fichiers.

```
git clone https://github.com/<coordinateur>/INDIA.git my_third_depot
```

   b. Chaque membre du groupe, créer un fichier d'extension « **votre_nom.md** » dans le dossier de travail du dépôt, contenant : Votre nom, parcours, formation, Format Markdown (.md)
   c. Ecrire un message qui décrit l'ajout du fichier : git add votre_nom.md
   d. Ajouter ce fichier à la copie locale du dépôt : git add puis git commit
   e. Soumettre cette modification au dépôt distant : git push
   f. Mettre à jour votre copie locale depuis le dépôt distant :git pull.
3. Création des supports de cours. Chaque étudiant·e choisit un module de sa formation, et différent de celui de ses camarades
   a. Créer un dossier correspondant dans la copie de travail du dépôt : mkdir -p cours/mon_module
   b. Ajouter un fichier au format « .md », contenant le titre de ce module, et le nom des intervenants
   c. Créer un fichier cours/mon_module/intervenants.md avec : Titre du module, et Nom des enseignants
   d. Ajouter ce fichier au dépôt local, puis le soumettre au dépôt distant
   e. Récupérer les fichiers des autres membres du groupe
4. Simulation des attaques et résolution
   a. Chaque étudiant·e modifie le fichier d'un·e autre étudiant·e de son groupe
   b. Le propriétaire du fichier corrige l'erreur et pousse la correction.
   c. Utiliser git log et git blame pour identifier qui a fait quoi.
5. Création de branches individuelles
   a. Chaque étudiant créé une branche de travail sur son dépôt local (git branch) nommé suivant un des cours de sa formation. Chaque groupe s'arrange pour que les étudiants du groupe n'aient pas de cours en double.

   On peut utiliser git checkout avec le nom d'une branche pour changer la copie de travail de branche, et ainsi travailler sur l'une ou l'autre des branches disponibles dans le dépôt local.

   b. Faire une description de la branche

   Chaque étudiant créé un fichier « .md » décrivant le cours qu'il a choisi, puis l'ajoute à la branche du dépôt local avec git add.

```
echo "# Cours : [Nom]" > cours/nom_du_module/contenu.md
```

   c. Chaque étudiant édite le fichier README.md pour y ajouter une courte description de ce cours récemment ajouté.

   Ces modifications sont ajoutées au dépôt local par un git commit. N'oublier pas d'indiquer au moment du ***commit*** un message de description explicite de la contribution**.**

```
- [Nom du cours](cours/nom_du_module/contenu.md)
```

   d. Chaque étudiant soumet sur le dépôt distant sa branche, en une nouvelle branche distante. Vérifier que vos branches existent bien sur le dépôt serveur de github.
6. Fusion de branche :
   a. Chaque étudiant récupère toutes les branches : git fetch origin
   b. Chaque étudiant peut voir les branches distantes : git branch –r
   c. Chaque étudiant fusionne SA propre branche dans main :

```
git checkout main
git pull origin main          # synchroniser
```

```
git merge nom_du_module       # fusionner sa branche
# Résoudre les conflits si nécessaire
git push origin main          # pousser vers le dépôt partagé
```

7. Récupération du projet complet, chaque étudiant peut récupérer en local la version complète du projet (git pull).

```
git checkout main
git pull origin main
```

8. Structuration du dépôt, chaque étudiant déplace avec git son fichier markdown dans un dossier cours afin de structurer le dépôt (git mv).

```
git mv votre_prenom.md membres/
git mv cours/... cours/  # si besoin
```

```
git commit -m "Structuration du dépôt"
git push origin main
```

9. Améliorer le contenu des notes, On peut poursuivre :
   - en créant un lien hypertexte dans le fichier README.md pour chacun des cours
   - en améliorant le contenu des notes de cours, en intégrant des illustration au sein des notes, améliorer la structure hiérarchique des dossiers, par exemple en séparant les cours par sous-dossiers correspondant aux différents master (et partie commune).

A chaque fois, n'oublier pas de partager ces améliorations sur le dépôt du serveur.