Source Code Management (Git)

Mohamed Amine Ben Rhouma

Plan

- Source Code Management
- Git
- Création d’un Repository Github
- Configuration
- Les commandes Git
- QUIZ

Source code management

CLEVORY TRAINING

- Contrôle de version
- Travailler en équipe
- Stockage centralisé de votre code
- Open Source
- Améliorer votre code
- Montrez-vous

Source code management

CLEVORY TRAINING

- Les modifications sont généralement identifiées par un code numérique ou alphabétique.
- Les révisions peuvent être comparées, restaurées et, avec certains types de fichiers, fusionnées
- SCM implique le suivi des modifications (Tracker) apportées au code.
- Le suivi des modifications facilite le développement et permet à différentes versions d'un même fichier de coexister.
- Savoir quel changement a été réalisé et qui l'a réalisé

Source code management

CLEVORY TRAINING

- Gestion des modifications de :
  - Documents
  - Programmes informatiques
  - sites Web
  - Autres collection d'informations…

Source code management

CLEVORY TRAINING

- Fournir un historique courant du développement
- Aider à résoudre les conflits lors de la fusion des contributions plusieurs sources.
- Les outils logiciels SCM sont parfois appelés :
  - Systèmes de gestion de code source (SCMS)
  - Systèmes de contrôle de version (VCS) ou simplement "dépôts de code"
  - Systèmes de contrôle de révision (RCS)

Source code management

Contrôle de version centralisé :

- Avoir une seule copie « centrale » de votre projet sur un serveur.
- centralisation du code source lié au projet
- Valider les modifications sur cette copie centrale
- Ne jamais avoir une copie complète du projet localement
- CVS, SVN (Subversion)

Contrôle de version distribué :

- Le contrôle de version est mis en miroir sur l'ordinateur de chaque développeur.
- Permet de gérer automatiquement les branchements et les fusions.
- Capacité à travailler hors ligne (permet aux utilisateurs de travailler de manière productive lorsqu'ils ne sont pas connectés à un réseau)
- Solution : Git, Mercurial.

Source code management

Contrôle de version centralisé :

- la source du code du projet est hébergé sur un serveur distant central et les différents utilisateurs doivent se connecter à ce serveur pour travailler sur ce code

Contrôle de version distribué :

- le code source du projet est toujours hébergé sur un serveur distant mais chaque utilisateur est invité à télécharger et à héberger l'intégralité du code source du projet sur sa propre machine.

Source code management: Contrôle de version distribué

- Copier l'intégralité du contenu du serveur pour travailler en local sur sa machine.
- L'historique des modifications faites par chaque développeur afin que chacun ait accès aux avancées des autres
- Synchroniser son dossier local avec le serveur

- Le modèle distribué a été popularisé par Git et présente différents avantages notables par rapport au modèle centralisé :
  - Simplicité / flexibilité du travail : Comme chaque utilisateur peut héberger le code du projet, on n'a plus besoin d'être constamment connecté à un serveur central et on peut donc travailler en ligne sur sa propre machine ;
  - Sécurité : Comme chaque utilisateur possède le code complet d'un projet, on peut utiliser la copie du projet d'un utilisateur comme back-up en cas de corruption du serveur central.

Git

Git

- Git est un logiciel de versionnement distribué créé en 2005 par Linus Torvalds, le créateur de Linux.

  - Gestion de projets

  - conserver un historique des modifications effectuées sur un projet

  - Travail collectif en équipe sur le même code

  - Eviter le conflits qui peuvent se produire entre

  - version actualisée pour chaque développeur du projet pour tester et implémenter ses fonctionnalités.

  - Savoir sur quoi travaillent les autres développeurs afin de travailler correctement et éviter la redondance

Git

- Dans le langage des systèmes de gestion de version, la copie de l'intégralité des fichiers d'un projet et de leur version située sur le serveur central est appelé un dépôt.

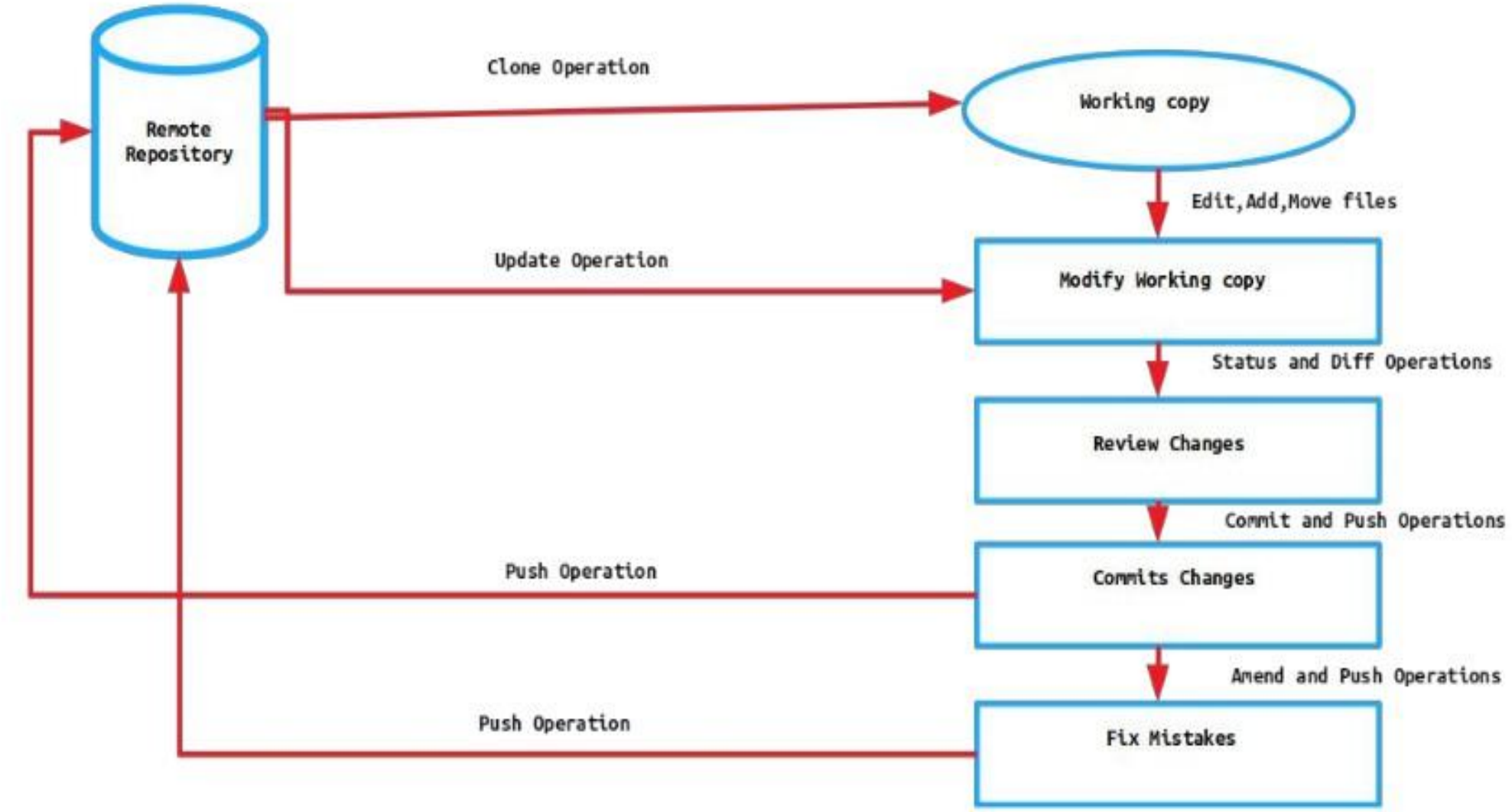
- Git appelle également cela "repository" ou "repo" en abrégé.

- Avantages :
  - Gratuit et open source
  - Simple et léger
  - backup
  - Sécurisé : SHA1 renommage et identification des objets .
  - Simple branchement

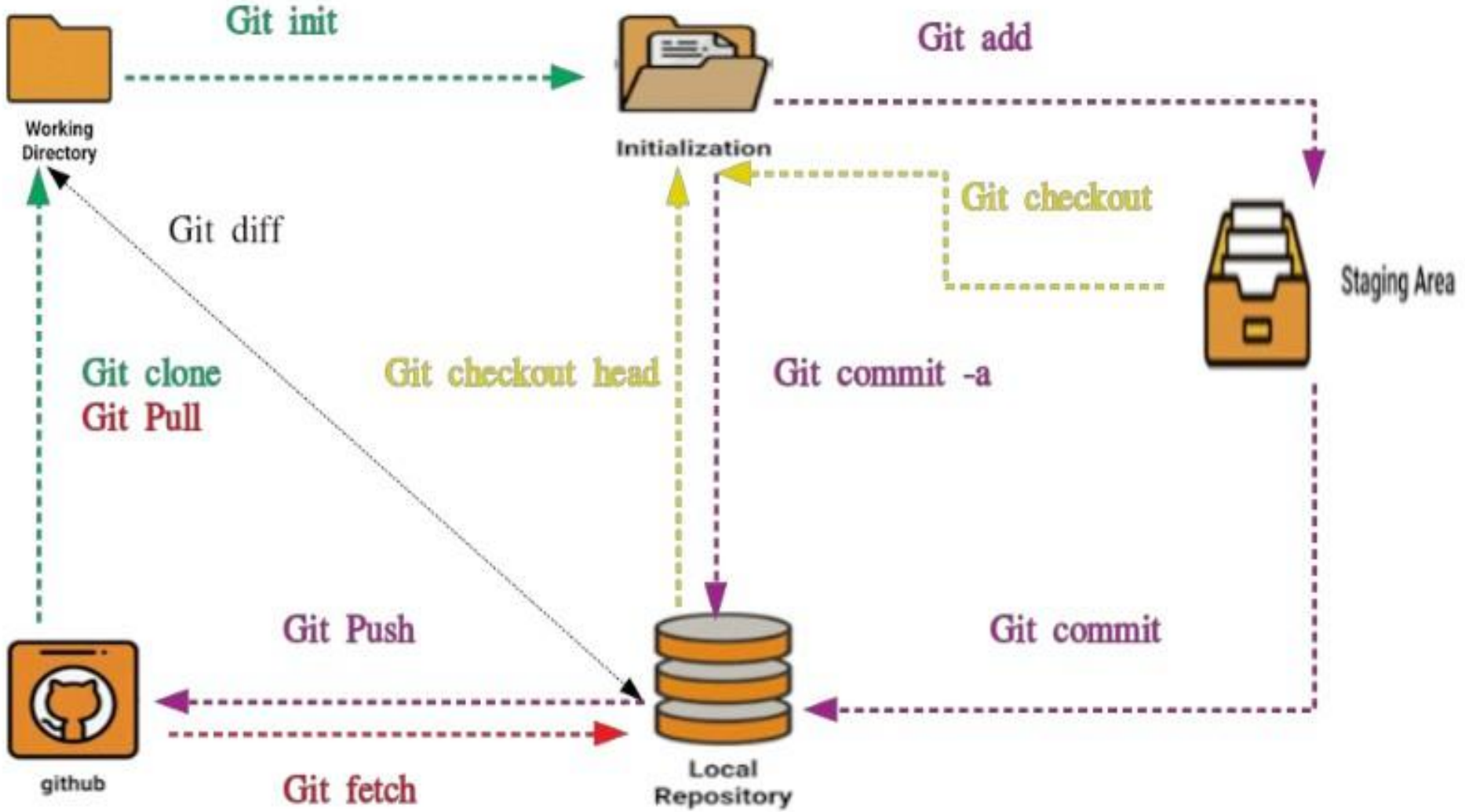
Git

- General workflow :
  - cloner le repository Git en tant que copie de travail.
  - modifie la copie de travail en ajoutant/éditant des fichiers.
  - Si nécessaire, mettre également à jour la copie de travail en prenant les modifications d'autres développeurs.
  - Vérifier les modifications avant de valider.
  - Valider les modifications.
  - Si tout va bien,pousser les modifications vers le remote repository.
  - Après avoir validé, si quelque chose ne va pas, alors vous corrigez le dernier commit et poussez le modifications apportées au repository.

Git cycle de vie



Git cycle de vie



GitHb

- Git est le système de contrôle de version,
- GitHub est un service en ligne d'hébergement de dépôts ou repo Git qui aide les gens à collaborer sur l'écriture de logiciels.

- le plus grand hébergeur de dépôts Git du monde.
- Une grande partie des dépôts hébergés sur GitHub sont publics, ce qui signifie que n’importe qui peut télécharger le code de ces dépôts et contribuer à leur développement en proposant de nouvelles fonctionnalités.

Création d'un RepositoryGithub

GitHub

CLEVORY TRAINING

- Un “dépôt” ou repository correspond à la copie et à l’importation de l’ensemble des fichiers d’un projet dans Git. Il existe deux façons de créer un dépôt Git :

  - On peut importer un dépot déjà existant dans git ;
  - On peut cloner un dépôt git déjà existant.
  - Création de Token

Git

CLEVORY TRAINING

- Un fichier peut avoir deux grands états dans git :
  - il peut être sous suivi de version
  - non suivi.

- Ensuite, chaque fichier suivi peut avoir l'un de ces trois états :
  - Modifié ("modified") ;
  - Indexé ("staged") ;
  - Validé ("committed").

installation

- https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-Installation-de-Git
- https://github.com/git-guides/install-git

Configuration

GitHub

CLEVORY TRAINING

- L'outil git config permet de définir des variables de configuration.
- Git stocke toutes les configurations globales dans le fichier .gitconfig qui se trouve dans votre répertoire personnel.
- gitconfig configuration globale de Git
- Pour définir ces valeurs de configuration comme globales, ajoutez l'option –global
- Sans --global , vos configurations sont spécifiques au repository git actuel.
- Vous pouvez également définir une configuration à l'échelle du système utiliser le -- système
- Git stocke ces valeurs dans le fichier /etc/gitconfig, qui contient le configuration pour chaque utilisateur et repository sur le système.
- Pour définir ces valeurs, vous devez disposer des droits root.

Git : exemple de configuration

CLEVORY TRAINING

- Setting username:
  - git config --global user.name « Ahmed Mokaddem"
- Setting email id:
  - git config --global user.email "ahm.moka@gmail.com"
- Color highlighting:
  - git config --global color.status auto
  - git config --global color.ui true
  - git config --global color.branch auto

Git : exemple de configuration

- Setting default editor:
  - git config --global core.editor vim
- Setting default merge tool:
  - git config --global merge.tool vim
- Check Configuration:
  - git config --list

Les commandes Git

git config

- **Configuration:**

  git config –gloabal username "nom"

  git config –gloabal useremail « @Email »

git init

- Initialise le repository
- Cela crée un nouveau dossier nommé .git qui contient tous les fichiers nécessaires au dépôt

git add <nom> / git rm <nom>

- Ajouter les fichiers à tracké et versionnés
- Commande :
  - git add <nomfichier>
  - git add -A or
  - git add .
  - git add -u(Modified and deleted files),
  - git add *.go

- Supprimer le fichier de l’espace de staging
  - git rm <nomfichier>

git commit

- Validation de l'environnement et des fichiers et placer en local repository
  - git commit – m "le message du comit"
  - git commit - a
  - git commit amend merger plusieurs commit
  - git commit – am

- HEAD pointeur sur l'environnement du dernier commit de la branche courante
- git tag : Marquer les commits spécifiques

Git Inspecter les changements

CLEVORY TRAINING

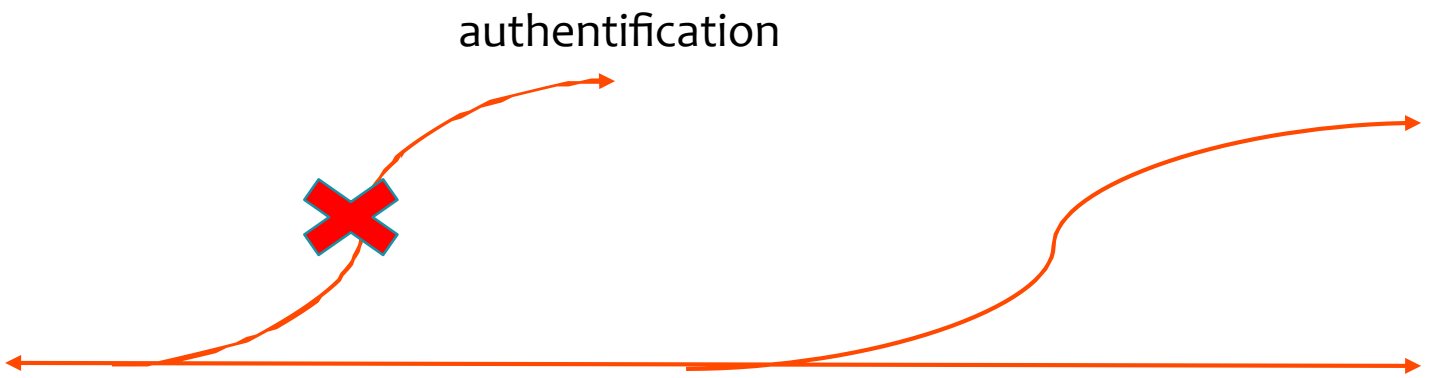
- git status
  - L'état des fichiers du repository de la zone staging
  - Affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore ajoutés ou validé
- git fetch
  - Afficher les changements entre the remote and the local dépôt
- git diff :
  - Afficher les différences entre les commits
- git log
  - Show commit logs et l'historique
- git blame
  - Afficher les modifications sur le repository et celui qui est à l'origine

Branche

- Les branches sont utilisées pour développer des fonctionnalités isolées les unes des autres
- La branche master est la branche "par défaut" lorsque vous créez un repository
- Utilisez d'autres branches pour le développement et fusionnez-les avec la branche principale une fois terminées

git branch dev
git checkout dev

Branche



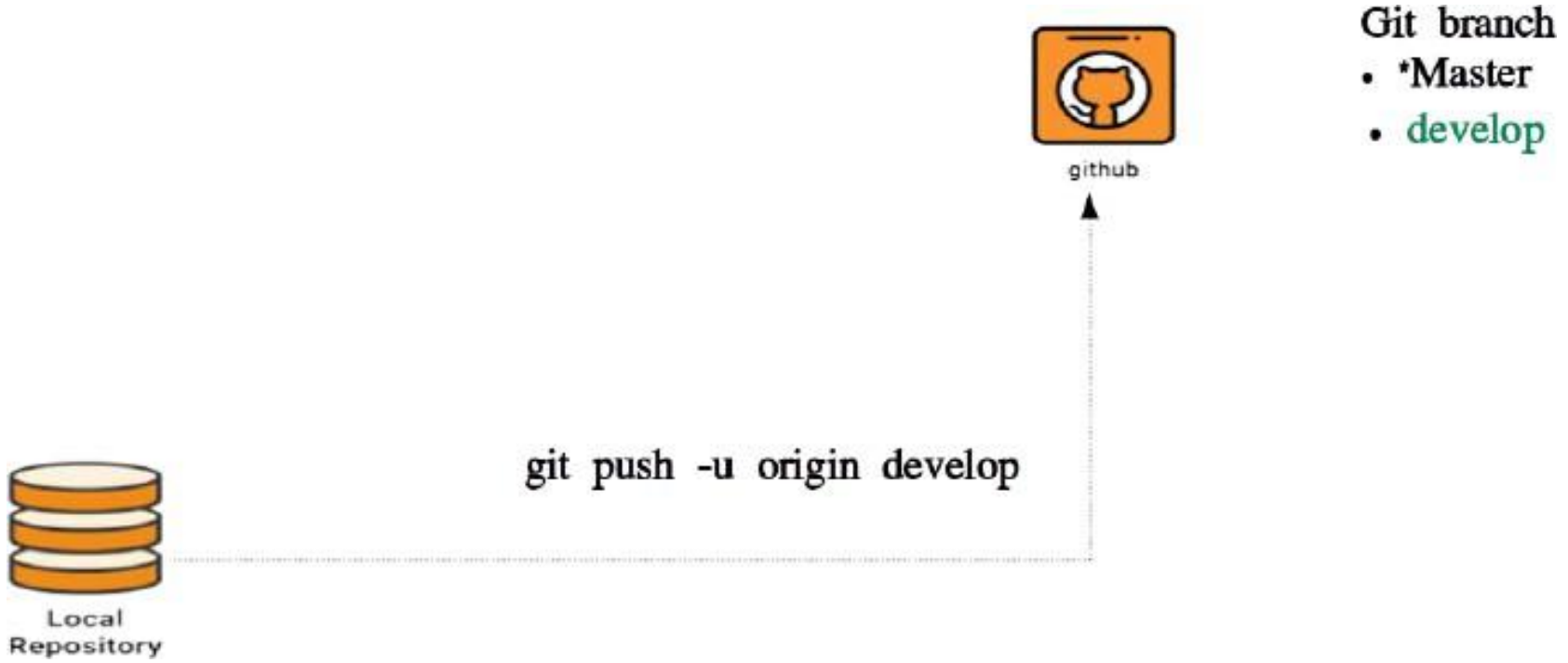
git branch –d authentification

git branch --delete authentification

git branch –m dev develop

git branch --move dev develop

Push branch au remote repository



git cherry-pick

- Parfois, vous ne voulez pas fusionner une branche entière dans une autre et vous n'avez besoin que d'un ou deux commits spécifiques.
- Ce processus s'appelle cherry-pick
- Choisir le ou les commit à merger
  - git cherry-pick

git checkout

CLEVORY TRAINING

- Basculer entre différentes versions d'une entité cible.
- La commande opère sur trois entités distinctes : les fichiers, les commits et les branches. ...

- Peut être utilisé pour afficher les anciens commits.
- Quitter une branche ou se positionne sur une branche
  - git checkout <nom branche>
  - git checkout –b <nom branche> créer une branche et se positionner sur la branche

git revert <ID commit>

CLEVORY TRAINING

- Revenir à un commit par son ID

  - Avoir le hash à l'aide de la commande

    - git log

  - git revert <ID du commit>

git reset

- git reset : git reset HEAD~ --hard
- Supprime le dernier commit de la branche master
- Pour annuler et revenir en arrière corriger un commit
  - git reset HEAD~ --hard
  - git reset --hard

Git fork

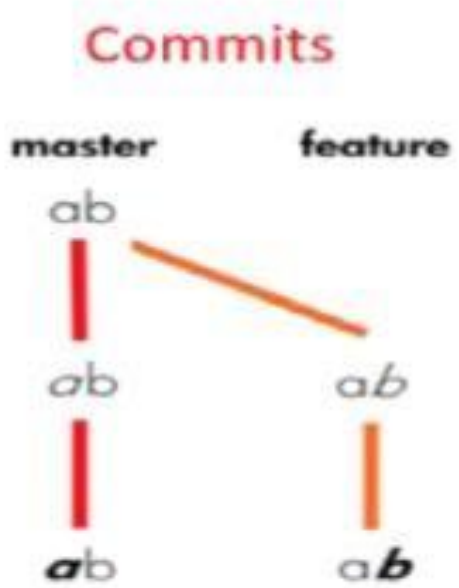
- Un fork est une copie d'un repository.
- Forker un repository vous permet d'expérimenter librement des modifications sans affecter le projet initial.

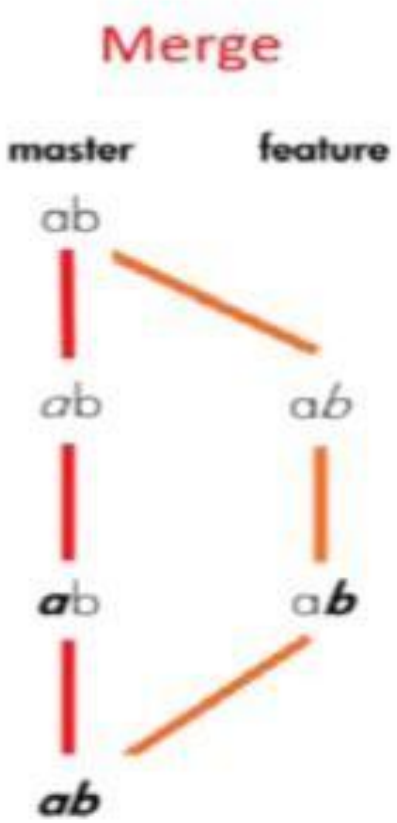
git merge et git rebase

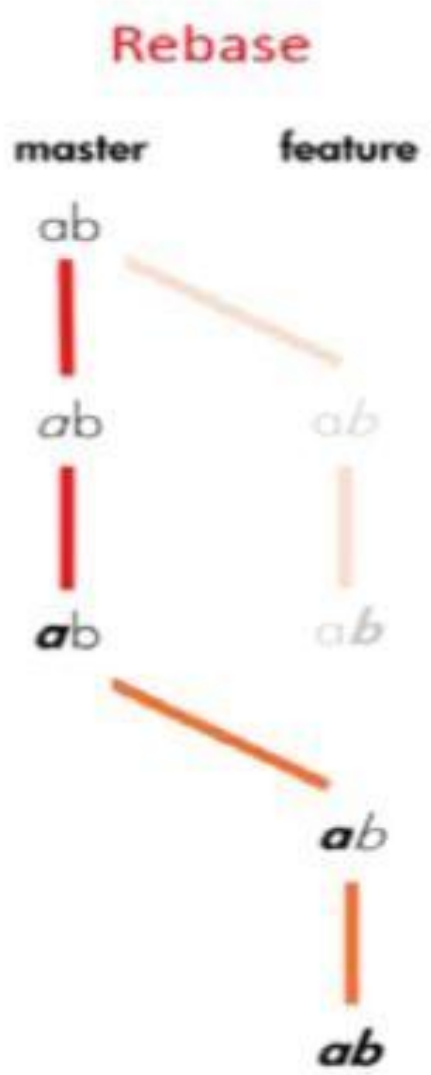
- merger et rebase intègre tous les deux une branche dans une autre mais de façon différentes
- rebase ajout tous les changements dans la branche à partir du dernier commit de la branche master

- rebase est le fait merger tous les commit de la branche
- rebase déplace la base de la branche vers le master au point d'arrivée de la branche.
- merger le dernier commit
- git merge <nom branche>
- git rebase <nom branche>

git merge et git rebase







gitignore

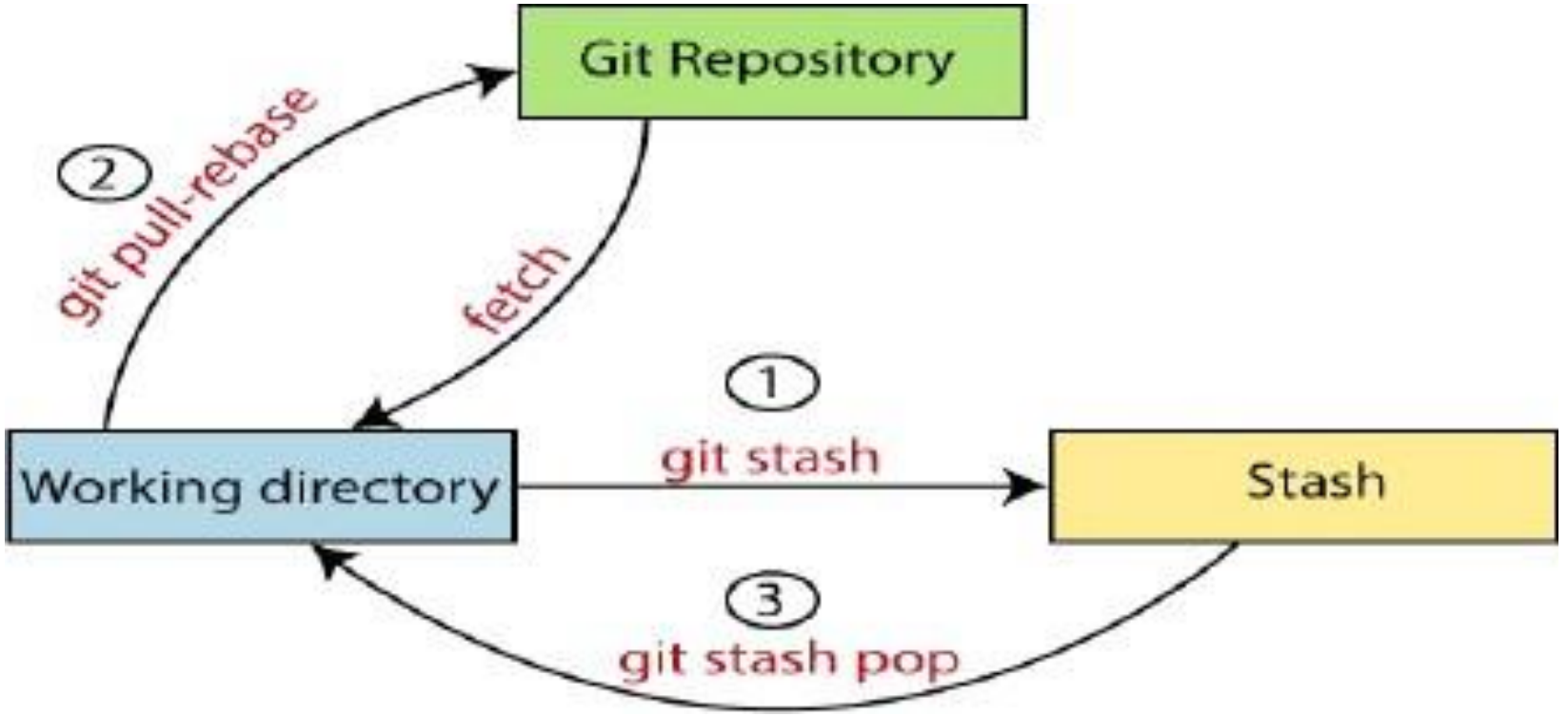
- Un fichier .gitignore est généralement placé dans le répertoire racine d'un projet.
- Le fichier qui contient le noms des fichiers à ne pas pousser en remote repository
- Vous pouvez également créer un fichier .gitignore global et tout les entrées de ce fichier seront ignorées dans tous vos référentiels Git.
- Exemple :
  - Ignore all texte files
    - *.txt

git stash

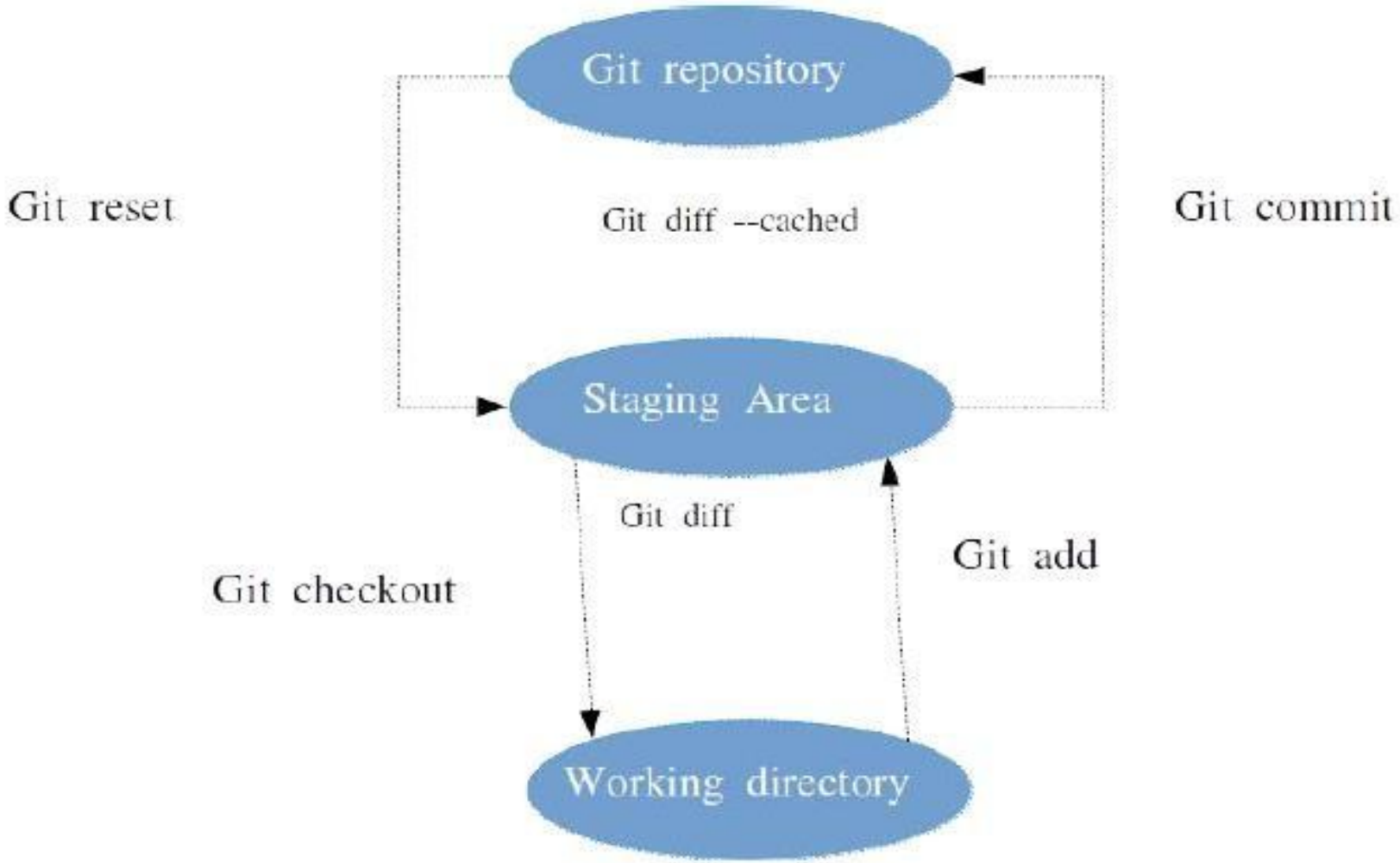
CLEVORY TRAINING

- Git stash: prend vos modifications non validées (non commited), les enregistre dans le cache pour un future usage.
- Pour mettre le fichier du repository dans le cache de git afin de ne plus les stagés et faire un autre trackage
  - git stash apply prendre sans effacer le stash
  - git stash pop prendre et supprimer le stash
  - git stash list affiche le stash

git stash


Git Repository
2
git pull-rebase
fetch
1
git stash
Working directory
Stash
3
git stash pop

git stash


Git repository
Git reset
Git diff --cached
Git commit
Staging Area
Git diff
Git add
Git checkout
Working directory

git clone <URL>

- Pour copier une version du remote repository en local si il n’existe pas

- git clone <Url du dépôt >

git remote add <nom> <url>

CLEVORY TRAINING

- Pour relier le repository local au rempote repository ayant l'URL dans Github
- Exemple :
  - git remote add origin <@remote repository>

Git fetch and Pull

CLEVORY TRAINING

- Regrouper les commit précédents en un seul.
- Squash : Excellent moyen de regrouper certains changements ensemble avant de les partager avec d'autres

- git Fetch:
  - Est une commande principale utilisée pour télécharger le contenu d'un autre repository distant (remote repository).
- git Pull:
  - Mettre à jour en local depuis le remote repository

git push

- git push:
- La commande est utilisée pour télécharger le contenu du repository local vers un repository distant.

- Transférer les commits de votre repository local vers un repository distant

Git flow manifest

- Master
  - releases seulement
- Develop
  - compile et réussit tous les tests
- Fonctionnalités des branches :
  - Là où la plupart des développements se produisent
  - Branchez-vous sur Develop
  - Fusionner dans Develop
- Libérer les branches
  - Branchez-vous sur Dvelop
  - Fusionner en Master et Develop
- Correctif
  - Débranchement du Master
  - Fusionner en Master et Develop
- Correction d'un bug
  - Branchez-vous sur Develop (développer)
  - fusionner dans Develop (développer)

Cycle Git d'une branche de feature

1) Activer le flux git pour le repository :
   - git flow init –d

2) Démarrer la branche features :
   - git flow feature start newstuff
   - Crée une nouvelle branche appelée feature/newstuff qui dérive de develop

3) Push it to GitHub pour la premiére fois :
   - Apportez des modifications et validez-les localement
   - fonctionnalité git flow publier des actualités

4) Additional (normal) commits and pushes as needed:
   - git commit –a
   - git push

5) Bring it up to date with develop (to minimize big changes on the ensuing pull request)

6) Mettez-le à jour avec develop (pour minimiser les gros changements sur la pull request qui s'ensuit)
   - git checkout develop
   - git pull origin develop
   - git checkout feature/newstuff
   - git merge develop

7) Terminez la branche de fonctionnalité (n'utilisez pas la finition de la fonctionnalité git flow)
   - git remote update -p
   - git branch -d feature/newstuff

Cycle Git d'une branche de feature

CLEVORY TRAINING

- En utilisant la commande git add, on demande à Git de suivre certains fichiers, c'est-à-dire de prendre en charge la gestion de l'évolution de ces fichiers,
- Par la commande git commit, nous demandons à git d'enregistrer un snapshot, c'est-à-dire une photographie instantanée de l'état des fichiers suivis (plus exactement, des modifications qui ont été placées dans l'index, mais il s'agit là en fait d'une facilité),

- Une suite des commits constitue un historique

Cycle Git d'une branche de feature

CLEVORY TRAINING

- Git va alors voir votre répertoire de travail comme étant en fait composé de :
  - l'ensemble des fichiers suivis, dans leur version correspondant à UN SNAPSHOT COURANT (un commit)
  - PLUS les modifications existant sur ces fichiers par rapport à cette version spécifique
  - PLUS des fichiers non suivis (pour lesquels on n'a pas effectué de git add)
  - !!! Relisez bien cela, c'est un des secrets pour utiliser Git avec facilité !!!

- Pour désigner le 'SNAPSHOT COURANT', Git utilise une référence (c'est à dire, un pointeur vers un commit) qui s'appelle HEAD.

- Il existe une autre référence, que l'on va bientôt manipuler, et qui pointe vers la tête de l'historique. Sachez juste pour l'instant qu'elle s'appelle master, on verra plus tard pourquoi elle porte ce nom.

Influence sur le répertoire de travail

CLEVORY TRAINING

- Votre répertoire de travail est constitué des fichiers suivis dans leur version correspondant au commit référencé par 'HEAD'
- Par conséquent, git checkout 56cd8a9, par exemple, va déplacer 'HEAD' sur le commit 56cd8a9, puis va faire en sorte que votre répertoire de travail soit constitué des fichiers dans leur version correspondant au nouveau 'HEAD'.
- Les fichiers sont donc modifiés, ajoutés, ou supprimés, pour que votre répertoire de travail soit dans l'état qui était le sien au moment du commit 56cd8a9
- Vous comprenez pourquoi il n'est pas possible (sauf dans un cas bien précis) d'utiliser git checkout si il y a des modifications en cours : elles seraient perdues. C'est ce que Git vous répondra, en refusant d'effectuer le git checkout.

Influence sur le répertoire de travail

CLEVORY TRAINING

- Avec git checkout, vous pouvez aussi récupérer un état particulier d'un fichier donné. Cela va nous permettre de récupérer notre fichier hello.py, en utilisant (à adapter à votre historique) :
  - git checkout 081c474 hello.py
- Littéralement, cela signifie : met dans le répertoire de travail le fichier hello.py dans l'état où il était au commit 081c474.
- git status montre que la commande précédente a mis la modification dans l'index. Vous pouvez maintenant faire un nouveau commit pour restaurer complètement le fichier.

CLEVORY TRAINING

Quiz

- Commande pour savoir qui a modifié un code
  - Git commit
  - Git log
  - Git blame

CLEVORY TRAINING

Quiz

- Commande qui permet de mettre à jour un dépôt existant
  - git push origin master
  - git fetch
  - git pull
  - git clone

Quiz

- Pour ajouter les fichier python seulement pour les version quel commande ajouter
  - git add
  - git add .
  - git add *.py
  - git add –a

Quiz

- Vous devez fusionner une branche appelée my-feature dans votre branche principale et vous êtes actuellement sur la branche my-feature. Comment effectueriez-vous la fusion ?
  - git checkout master
  - git merge my-feature

Quiz

- Vous avez été transféré dans une nouvelle équipe de développement. Le développeur principal vous envoie un lien vers le repository GitHub. Quelle commande utiliseriez-vous pour copier le travail en local

CLEVORY TRAINING

Quiz

- Après avoir terminé votre travail de développement
- Quelles sont les commandes pour héberger votre code en GitHub

TP git

- Créer le répertoire projet contenant les fichiers index.html readme.txt et index.py
- Ajouter ces fichiers a versionner et tracké
- Créer un premeir commit d'ajout des fichiers avec message ; Ajout des fichiers
- Créer une branche Develop
- Se positionner dans la branche
- Modifier le fichier index.py pour afficher un message
- Inspecter les différances entre les versions ,
- merger la branche
- créer le commit « affichage des données »
- Revenir au premier commit
- Afficher l'historique
- Créer la branche test
- Afficher les branches
- Afficher différences entre commit
- Afficher le status du repoditory local
- Supprimer la branche test
- Push dans le remote repository sans le fichier readme.txt
- Placer dans le cache les fichiers puis les replacer
- Importer le dossier CoursDevops

CLEVORY TRAINING

Sitographie

- https://www.pierre-giraud.com/git-github-apprendre-cours/presentation-git-github/
- https://github.com/git-guides
- https://danielkummer.github.io/git-flow-cheatsheet/