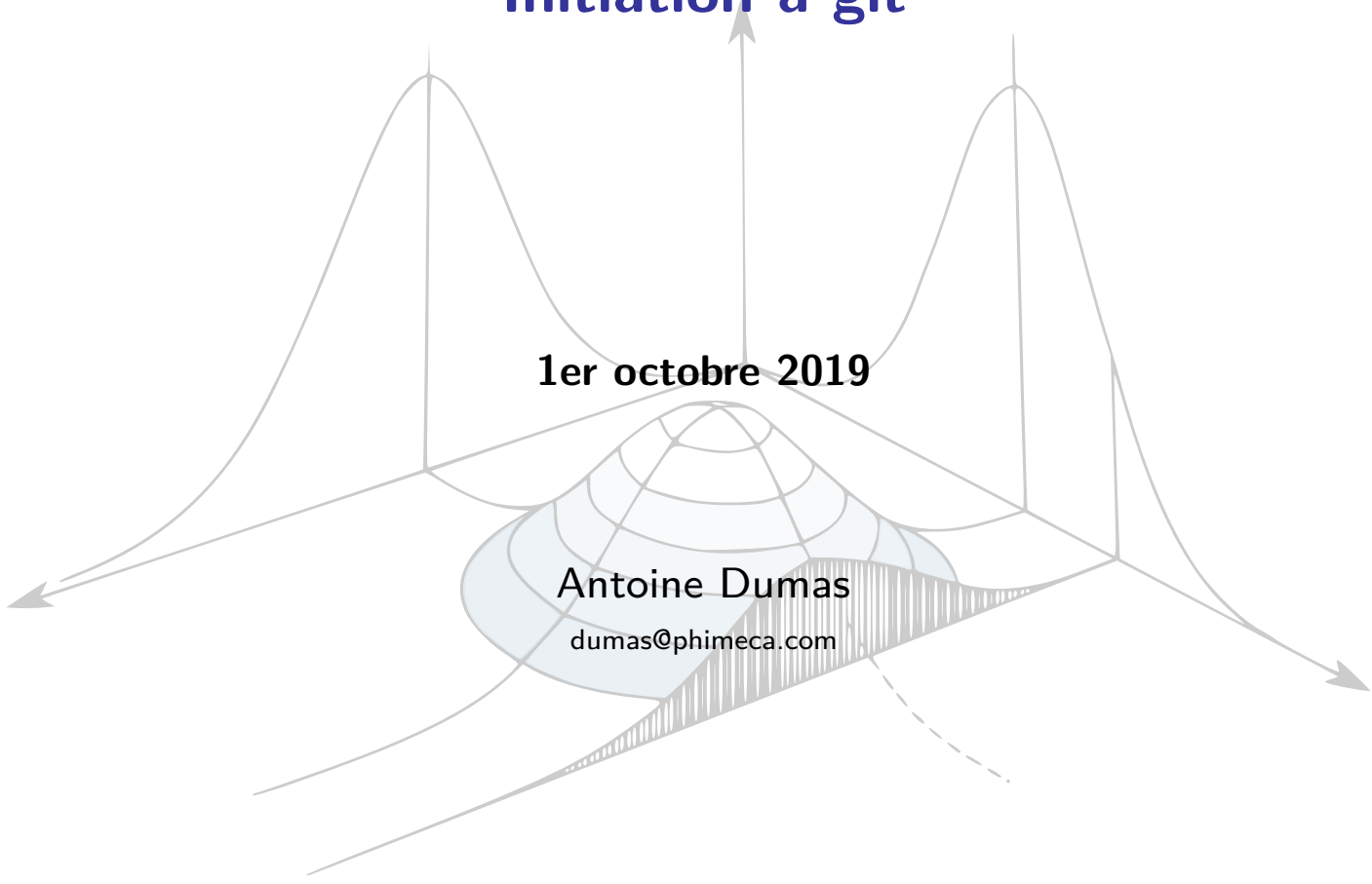
PHIMECA

Solutions for robust engineering

Initiation à git



1er octobre 2019

Antoine Dumas

dumas@phimeca.com

Sommaire

Sommaire

Git

- Logiciel de versionnement (VSC : *Version Control System*)
- But : Conserver l'historique d'un ensemble de fichier en le versionnant
  - pas de copie de copie
  - permet de retourner à des versions précédentes
  - permet de documenter les modifications (auteurs, date, message)
  - facilite la collaboration (permet de travailler en parallèle))
- Centralise le code dans un dépôt distant (Github, Gitlab, Bitbucket)
- Les fichiers à versionner : (script, code, fichiers de configurations, tous types de documents textes,)
- Les fichiers mal gérer : (les gros fichiers binaires, documents Microsoft Office, base de données).
  Solution possible : les ignorer (fichier *.gitignore*), utiliser *git lfs*.

Sommaire

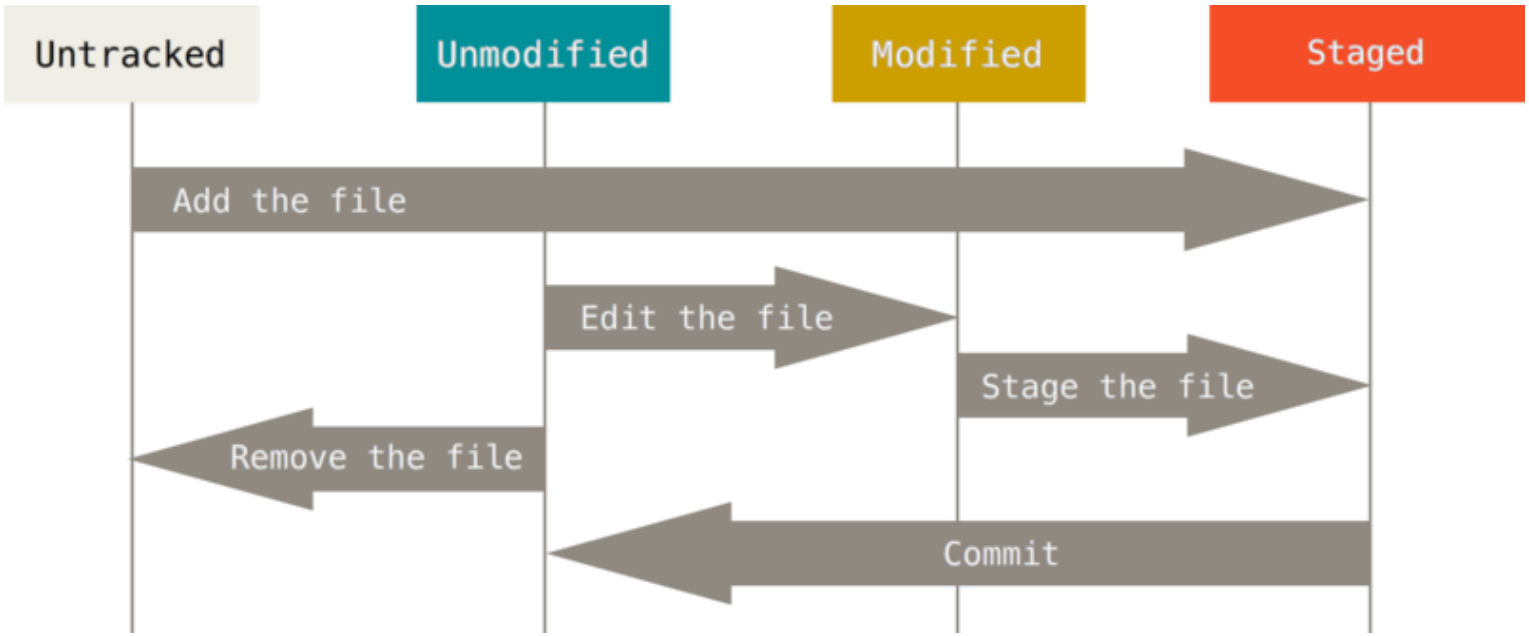
Git en ligne de commande

Git fonctionne avec de nombreuses commandes mais des applications graphiques existent aussi.

- git <commande> `--option`
- Première configuration :
  - git config `--global user.name 'Antoine Dumas'`
  - git config `--global user.email 'dumas@phimeca.com'`
- Exemple de commandes : init, status, log, diff, add, rm, commit, checkout, clone, pull, push, fetch, merge, branch, rebase, tag...

Les quatre statuts des fichiers suivis

- Le cycle de vie d'un fichier suivi avec Git



(Source: Pro Git book `http://git-scm.com/book`)

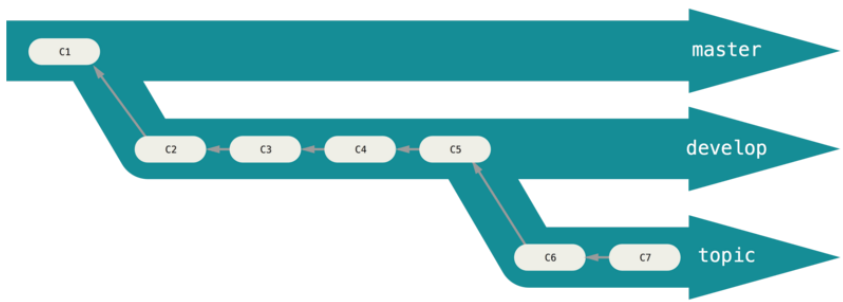
- Les fichiers qui ne sont pas des sources (fichiers objets, fichiers de compilations, exécutables, etc.) peuvent être ignorés.
  - Créer à la racine du dépôt le fichier *.gitignore* : lister les noms de dossier ou fichier devant être ignorés (*.pyc, build/, ...)

Les premières commandes

- Initier un dépôt avec versionnement
  - git init
- Vérifier l'état du dépôt
  - git status
  - git diff
  - git log
- Versionner des fichiers
  - git add : ajouter des fichiers au statut *staged*
  - git commit : enregistrer les modifications dans le dépôt (local)
- Annuler changements
  - git reset <fichier> : annuler l'état *staged* d'un fichier
  - git checkout <fichier> : récupérer la dernière version commitée d'un fichier

Les branches

- Git permet de créer et de fusionner très facilement des branches
- Un système de branches permet de préserver une version stable (branche *master*, par défaut à la création du dépôt) sans limiter les développements en parallèle ou non (branche *dev*, *develop*)
- Les branches sont particulièrement utiles pour le travail collaboratif et par sujet
- La réintégration des modifications se fait avec un opérateur de fusion (*git merge*)

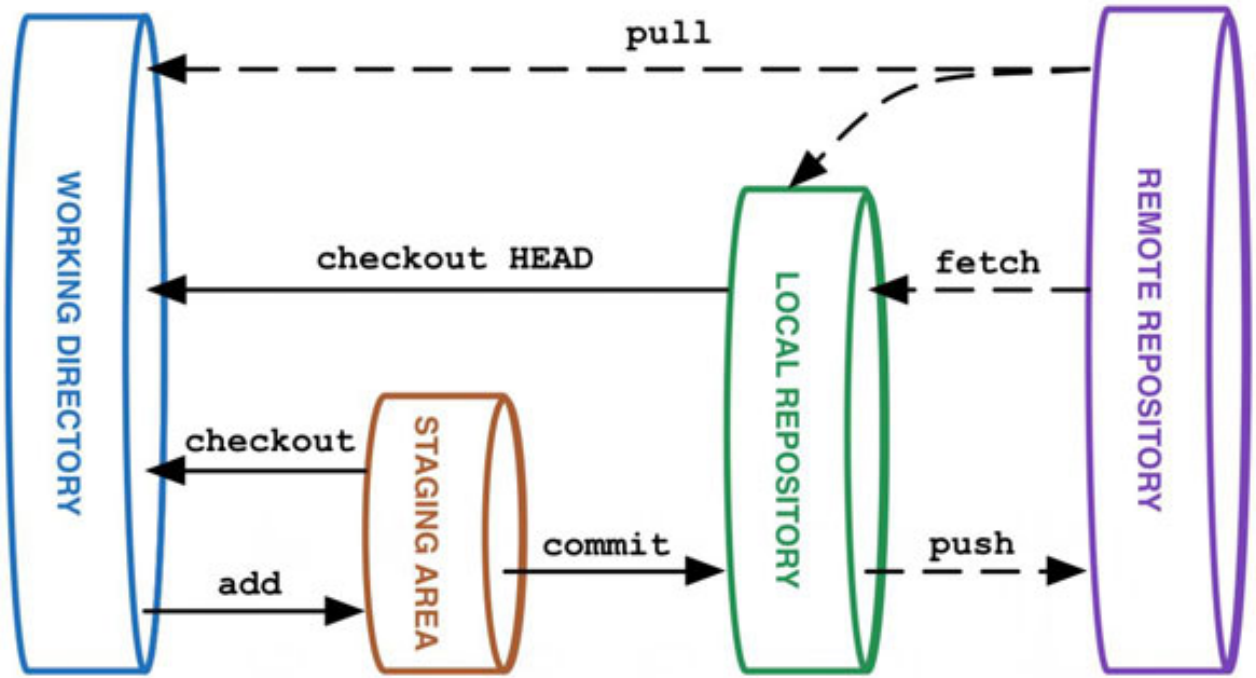


Sommaire

Le travail collaboratif : le dépôt distant

- Notion de *remote* : version du dépôt stocké dans un dossier local, une autre machine ou un serveur internet
  - https: https://github.com/lbeaucourt/SIGMA-algo-python.git
  - ssh : git@git.phimeca.lan:dumas/catalogue_formation.git
  - Répertoire local : /home/dumas/git/mon-depot-bare
- Gestion des remotes
  - git remote -v : lister les remotes avec leurs adresses
  - git remote add <name> <address>: ajouter un remote
- Convention : le dépot distance avec lequel vous synchronisez votre dépôt local s'appelle *origin*.

Le travail collaboratif : le dépôt distant



(Source : https://gnss-sdr.org/docs/tutorials/using-git/)

Le travail collaboratif : récupérer et partager les données

- Initier un dépôt git et ajouter un remote
  - git init : créer un dépôt de travail
  - git remote add <remote> <url>: ajouter un remote
- Cloner un dépôt distant
  - git clone <url du remote> : créer une copie local du dépôt distant
- Synchroniser le dépôt local et le dépôt distant
  - git push <remote> <branch> : pousser une branche vers un remote
  - git fetch <remote> : télécharger les changements et les branches du dépôt distant
  - git pull <remote> <branch> : télécharger la branche du remote et la fusionner avec le dépôt local

Le travail collaboratif : les serveurs

Comparaison des services d'hébergement Git en version gratuite

Github

- ✓ très gros projet (138+ millions dépôts, 600 employés)
- ✓ Grande interopérabilité avec d'autres outils
- ✗ dépôts publics uniquement
- ✗ pas d'instance privée

Bitbucket

- ✓ très gros projet
- ✓ Git & Mercurial
- ✓ intégration dans les produits Atlassian
- ✗ 5 utilisateurs max/dépôt
- ✗ pas d'instance privée

GitLab

- ✓ projet pérenne et dynamique (132 employés, 33 pays)
- ✓ pas de limite dans la version hébergée
- ✗ des lenteurs sur le site gitlab.com
- ✓ instance privée opensource
- ✓ outils d'intégration continue natifs

(novembre 2016, d'après `http://comparegithosting.com`)

Seul gitlab permet d'héberger une instance privée.

Sommaire

Travaux pratiques

- Se créer un compte sur `https://github.com/`
  - cloner le dépôt du cours : https://github.com/lbeaucourt/SIGMA-python.git
  - créer un dépôt pour votre code développé lors de ce cours.
- Configurer votre Git (dans une console)
- Synchroniser le dépôt local et distant (dans une console)
  - initialiser un dépôt git local (*git init*)
  - ajouter les fichiers (*git add .*)
  - commiter les fichiers (*git commit -m "mon premier commit"*)
  - ajouter le remote du dépôt distant (*git remote add origin <address>*)
  - pousser les changements sur le dépôt distant (*git push origin master*)
- Intégrer de nouvelles modifications
  - faire des modifications et les visualiser (*git status*)
  - commiter les nouvelles modifications
  - continuer à commiter lors de vos futurs développements !!