Systèmes d'exploitation 420-W12-SF

GIT : Le contrôle de version

Jean-Pierre Duchesneau, Automne 2021

Les cours

1. Intro aux infrastructures informatiques et les composantes internes du PC
2. Les composantes internes du PC et réseau
3. Système d'Exploitation
4. Virtualisation de clients et de serveurs
5. Disque dur, partition et système de fichier
6. La ligne de commandes (Shell) et les scripts
    1. CMD
    2. Linux
    3. Bash
    4. Les scripts
7. **Git, le contrôle de version**
8. WAMP

La gestion de version, pourquoi ?

Le besoin

- Pouvoir restaurer une version antérieure d'un fichier

La solution

- Les systèmes de gestion de version(SGV)

Comment

- Visualiser les changements au cours du temps
- Comparer
- Qui a modifié quoi
- Déterminer quel changement a causé des problèmes
- Revenir en arrière
- Fusionner (collaborer)
- Etc.

Les premiers Systèmes SGV

SGV locaux

SGV centralisés (SVN- CVS)

SGV distribués (GIT et autres)

- Suppression du point unique de panne
- possibilité de collaborer sur le même projet avec plusieurs équipes en simultané.

Git :

*G*it est un logiciel de gestion de versions décentralisé.

C'est un logiciel libre créé par Linus Torvalds en 2005.

Documentation complète en français :
https://git-scm.com/book/fr/v2

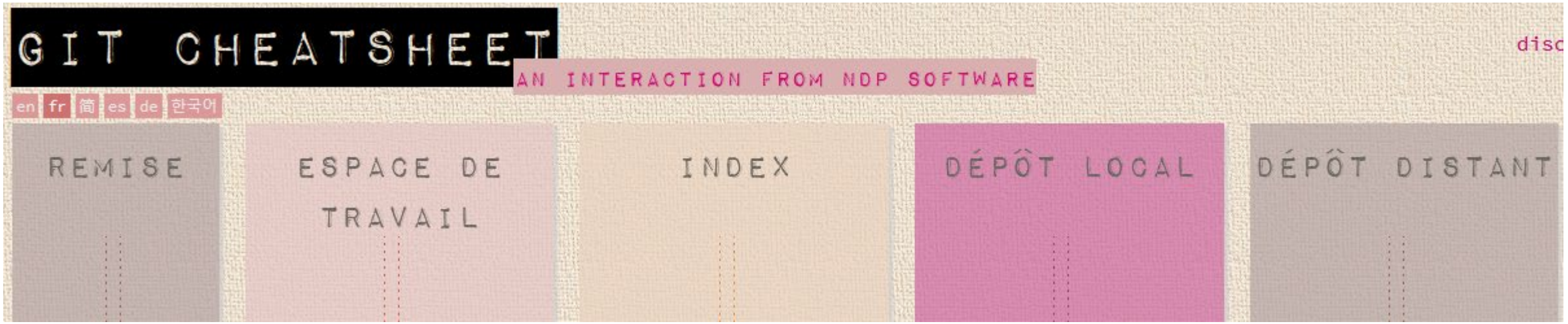
Serveur hébergeant le code source :

- GitHub
- Azure DevOps (Mircrosoft), à partir de la session 3.
- GitLab (OpenSource)
  - Cégep : https://dfcgit.cegep-ste-foy.qc.ca
  - Utilisé pour les travaux et les examens (Confidentialité)

Rudiment de GIT

Git gère trois états dans lesquels les fichiers peuvent résider : **modifié, indexé et validé**.

- **Modifié** signifie que vous avez modifié le fichier, mais qu'il n'a pas encore été validé.
- **Indexé** les changements sont enregistrés, le fichier peut être validé.
- **Validé** signifie que les données sont stockées en sécurité dans votre base de données locale ou votre dépôt local.



Initialiser un dossier avec GIT

```
jpduches@Bilbo MINGW64 ~/Desktop
$ cd test/

jpduches@Bilbo MINGW64 ~/Desktop/test
$ ls

jpduches@Bilbo MINGW64 ~/Desktop/test
$ git init
Initialized empty Git repository in C:/Users/jpduches/Desktop/test/.git/

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ ls -al
total 8
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 ./
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 ../
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 .git/

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$
```

Première utilisation :

git config

```
jpduches@Bilbo MINGW64 /d/OneDrive - Cégep de Sainte-Foy/Cours/420-W44-SF (maste
r)
$ git config --global user.name "Jean-Pierre Duchesneau"

jpduches@Bilbo MINGW64 /d/OneDrive - Cégep de Sainte-Foy/Cours/420-W44-SF (maste
r)
$ git config --global user.email jpduchesneau@csfoy.ca

jpduches@Bilbo MINGW64 /d/OneDrive - Cégep de Sainte-Foy/Cours/420-W44-SF (maste
r)
$
```

git help

```
jpduches@Bilbo MINGW64 /d/OneDrive - Cégep de Sainte-Foy/Cours/420-W44-SF (maste
r)
$ git config -h
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-regex]
    --get-all             get all values: key [value-regex]
    --get-regexp          get values for regexp: name-regex [value-regex]
```

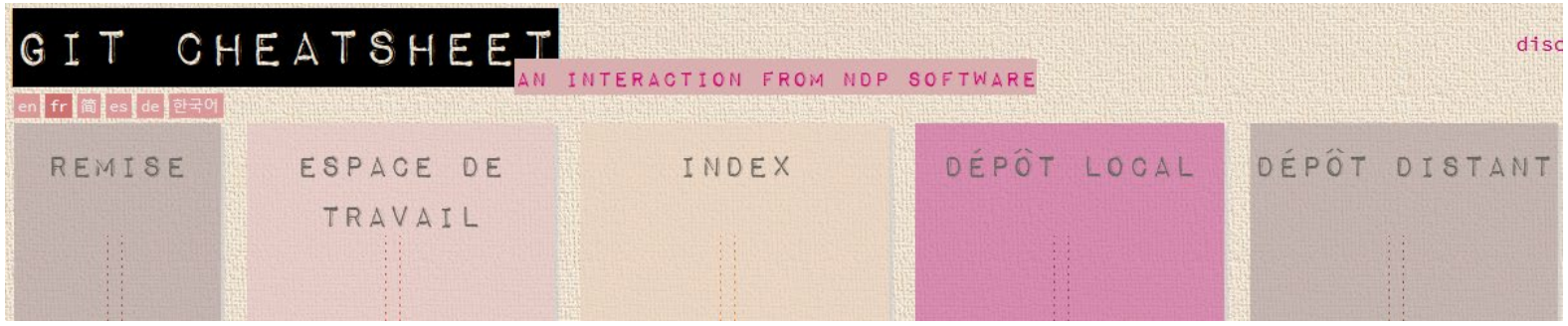
Modifier des fichiers

Indexer le fichier

```
total 8
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 ./
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 ../
drwxr-xr-x 1 jpduches 197609 0 avr.   6 09:27 .git/

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ touch toto.txt

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        toto.txt
```

```
jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ git add toto.txt

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ status
bash: status: command not found

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   toto.txt
```

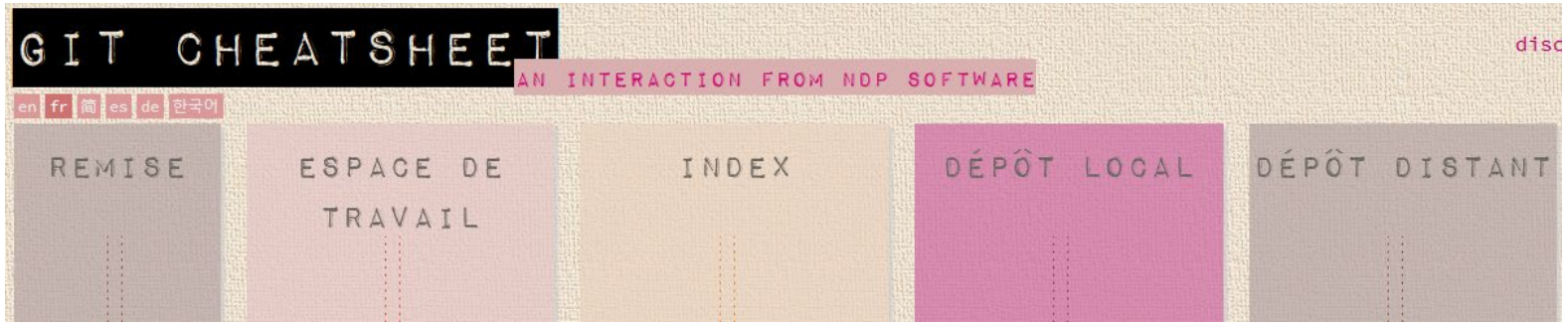
- Je prends de l'espace de travail et je le place dans l'index.
- Git vas garder une trace.



Faire les commit

```
jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$ git commit -am "Mon premier fichier"
[master (root-commit) 22108d1] Mon premier fichier
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 toto.txt

jpduches@Bilbo MINGW64 ~/Desktop/test (master)
$
```

Je prends de la zone index et je le place dans le dépôt local



Si j'avais un dépôt distant je pourrais faire la commande :
git push

Et ainsi j'enverrais tout ce qui est dans mon dépôt local dans mon dépôt distant.

Les étapes ET les commandes :

$git config : Exécuté que lorsqu'on veut modifier la configuration de git.

$git init : Initialisation d'un dépôt dans un répertoire existant.

$git clone [Adresse] [Destination] : Cloner un dépôt existant.

Seule l'une des deux commandes peut être utilisée.

$git status : Vérifier l'état des fichiers qui différent entre l'index et la tête du commit courant…

$git add : Indexer des fichiers modifiés

$git diff : Pour visualiser ce qui a été modifié, mais pas encore indexé.

$git commit [-am] : Valider vos modifications

$git rm fichier1 : Effacer le fichiers1

$git mv nom_origine nom_cible : Déplacer un fichiers

$git log : Énumère en ordre chronologique inversé les commits réalisés.

-p : montre les différences introduites entre chaque validation

-2 : limite le sortie de la commande aux deux entrées les plus récentes.

Les étapes ET les commandes (suite) :

$gitk	: interface graphique pour visualiser l'historique.
$git commit --amend	: Annuler ou modifier le dernier commit
$git reset HEAD fichier	: désindexer le fichier.
$git checkout --file	: ramène le fichier à son état du dernier instantané.

Travailler avec les dépôts distants

Origin : : C’est le nom par défaut que git donne au serveur à partir duquel vous avez cloné

$git remote : Visualisé les serveurs distants

-v : montre l’url que Git a stockée pour chaque nom court.

$git remote add : ajouter un dépôt distant.

$git fetch [Nom court] : Cette commande s'adresse au dépôt distant et récupère toutes les données de ce projet que vous ne possédez pas déjà. Après cette action, vous possédez toutes les références à toutes les branches contenues dans ce dépôt, que vous pouvez fusionner ou inspecter à tout moment

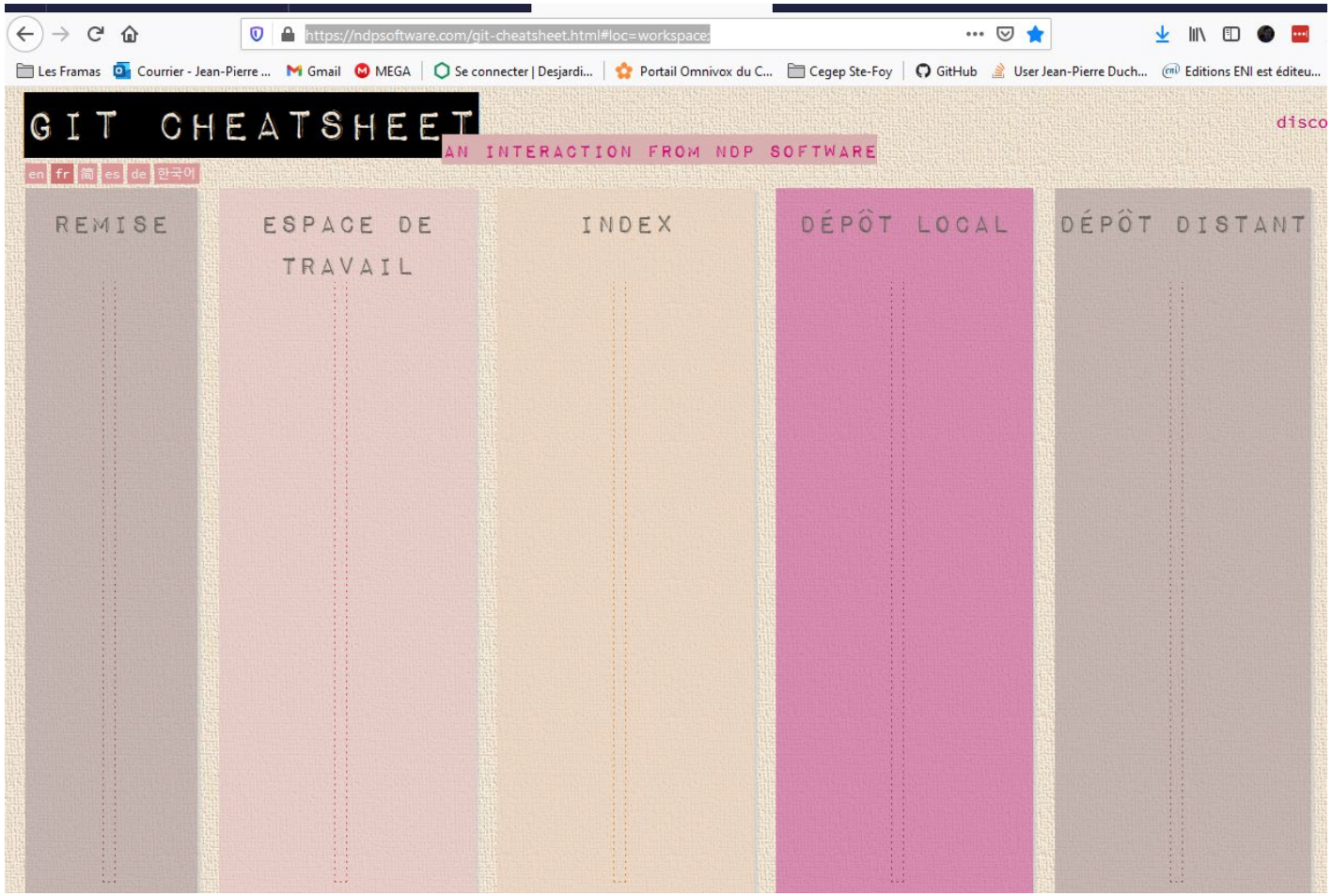
$git pull [nom branche] :suivre l'évolution d'une branche distante.

$git push [nom distant] [nom branche]:pousser son travail sur un dépôt distant.

$git remote show [nom distant] : visualiser plus d’informations à propos d’un dépôt distant particulier.

Amusez-vous sur le site :

https://ndpsoftware.com/git-cheatsheet.html#loc=workspace;



Travail à faire

Exercices

- Exercice 9 Visite guidée d'Ubuntu (**attention au nouveau compte Prof**)
- Exercice 10 Commandes de bases du Shell Linux
- Exercice 11 Gestion des usagers Linux
- Exercice 12 Git