Log1000 \_Demo1

**1ère partie : installation de Git**

1. Rendez-vous au site officiel de Git et téléchargez l’installateur : <https://git-scm.com/downloads>.

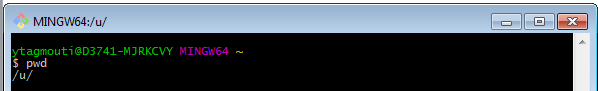
**2ième partie :**

Nous allons travailler en mode Console



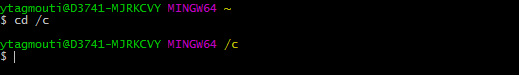
Tapez « **PWD** » suivi de **enter**.

Pour Connaître notre position



PWD veut dire **P**RINT **W**ORKING **D**IRECTORY

Tapez : «CD /C»



CD veut dire «Change Directory». Remarquez ce qu’il y **a** en jaune, dans l’invite de commande. C’est le répertoire en cours.

Pour visionner le contenu d’un répertoire, on effectue soit la commande «**dir**», «**ls**», ou

Tapez : «**LS -L**»

**Création de notre premier projet et dépôt Git**

Nous allons maintenant créer un petit projet web tout simple dans le but d’en activer le contrôle de version avec Git.

1. Créez un répertoire nommé «**Simpleprojet**» dans le répertoire «Mes Documents».

Tapez : **MKDIR** SimpleProjet

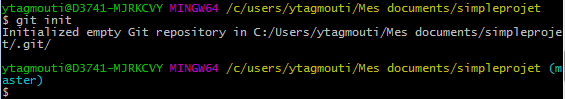
1. Entrez dans le projet

Tapez : **CD** SimpleProjet

1. Et, «**git init**»

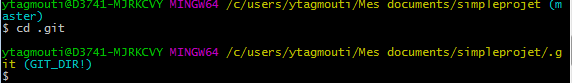
Tapez **git init**

Créer un dossier «.git». Ce dossier est la base de données qui contiendra toutes les versions de tous les fichiers associés au répertoire de travail ainsi que tous les sous-répertoires de travail!



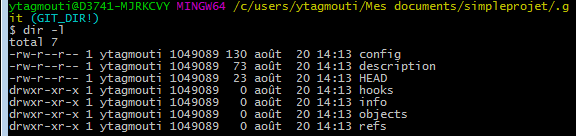
1. Allez dans le répertoire .git :

Tapez **CD .GIT**



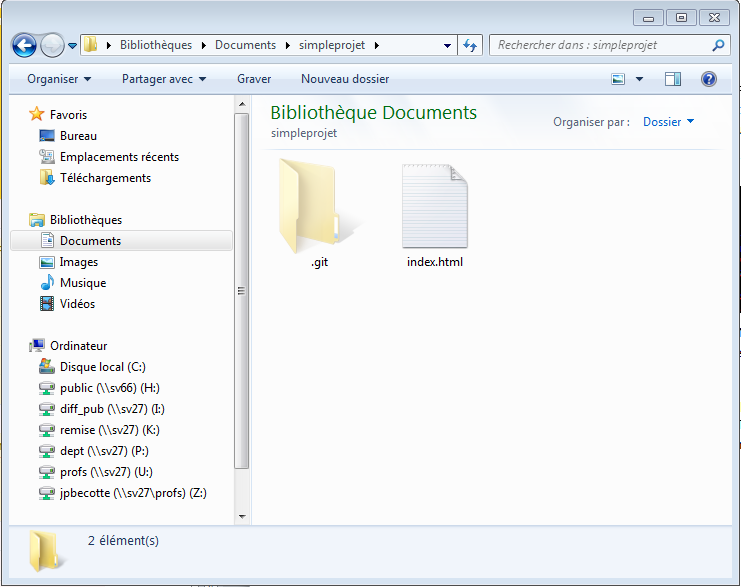
**nous ne sommes pas supposé changer les fichiers ni la structure du répertoire .git**. Il faut donc être prudent!

Que se passe-t-il si nous regardons la structure du répertoire .git?



Il n’est pas utile de détailler chacun de ces fichiers ni répertoires. Notons simplement que nous pouvons configurer les options de Git dans le fichier config; et que la base de donnée se trouve dans le répertoire «objects».

1. Créons maintenant une simple page web HTML dans le répertoire Simpleprojet. (Vous devrez utiliser l’explorateur de fichiers Windows). **ATTENTION! Le fichier ne doit pas se trouver dans le répertoire .git! !!Le fichier doit s’appeler «index.html».**



Notez que si vous avez spécifié vos options de dossier correctement, vous pourrez voir le dossier .git! !!

<!DOCTYPE html>

<html>

<body>

<h1>Voici une simple page web</h1>

<p>Bonjour!</p>

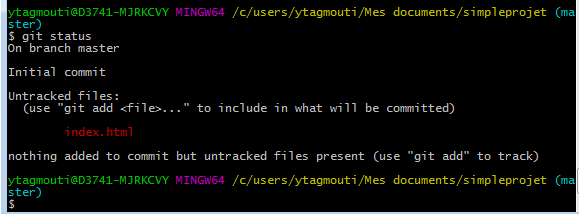
</body>

</html>

1. Voyons ce que Git à dire

**Tapez GIT STATUS**

Qui permet d’afficher l’état d’un fichier, rappelez-vous des sections principales d’un projet Git.



Ici, Git nous indique que **index.html** est «**untrack**». Cela veut dire que le fichier ne se trouve même pas dans la base de données.

1. Plaçons le fichier sur le «**staging**» qui se résume à la zone d’index.

Tapez **GIT ADD .**

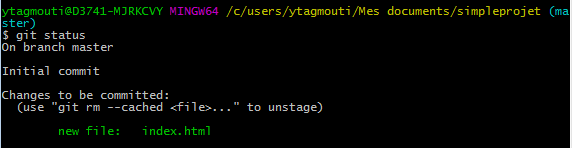


**«git ad**d» Permet de suivre c'est-à-dire indexer un fichier, ce qui veut dire en d’autres mots que le fichier fera partie du prochain instantané

**«git add»** introduit une confusion. Ici, «Add» veut dire «ajoute les fichiers et répertoires suivants dans le staging c'est-à-dire la zone d’index» et non «ajoute le fichier dans la base donnée». Bien entendu, le fait d’ajouter dans le staging ajoutera le fichier à la base de données éventuellement, mais c’est une nuance importante à comprendre. (Car l’opération inverse de «git add» est «git rm»… git rm n’efface pas le fichier du répertoire de travail… seulement du staging.)

1. Que se passes-t-il maintenant si on fait un «git status»?

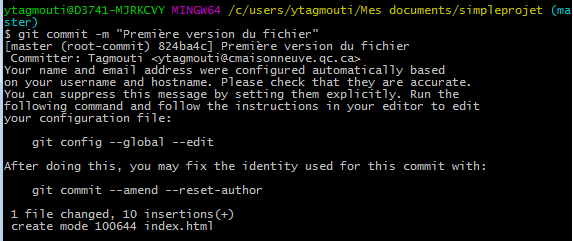
Tapez **GIT STATUS**



Git nous informe que notre fichier est bel et bien sur le staging!

1. Une fois notre *staging* prêt, on peut faire un commit! Avec la commande **git commit,** l’option –m permet d’ajouter un commentaire à notre commit

Tapez **GIT COMMIT -m ‘Première version du fichier’**



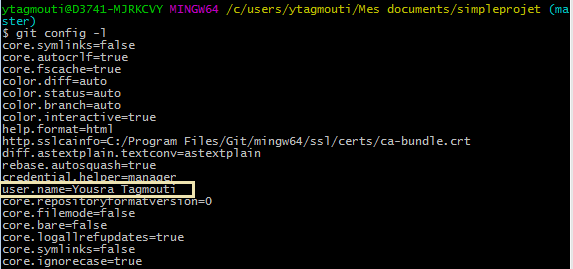
1. J’aimerais tout de même que mon nom apparaisse. Voici comment :

Tapez **GIT CONFIG** --global user.name "Yousra Tagmouti"



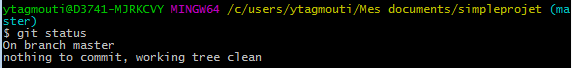
1. J’aime bien m’assurer que Git a bien pris mes changements en compte. Voici comment :

**Tapez GIT CONFIG -L**



1. Et maintenant, vérifions avec un autre «git status» :

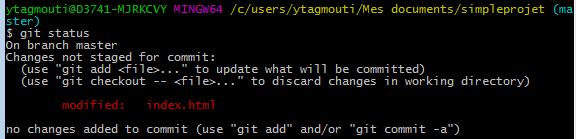
**Tapez GIT STATUS**



Qu’est-ce que cela signifie? En fait, Git nous dit que notre répertoire de travail est «clean» donc exempt de modifications! Modifions un peu les choses.

1. Dans le fichier index.html, ajoutez le nom de votre animal préféré dans la balise <p> et refaire un git status

Tapez **GIT STATUS**



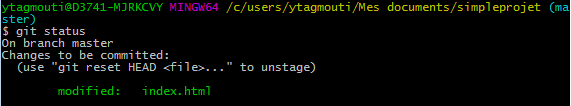
Maintenant, Git nous indique que notre fichier a été modifié. Nous devons le placer sur le staging. Quelle commande allons-nous utiliser ?

Tapez **GIT ADD** .



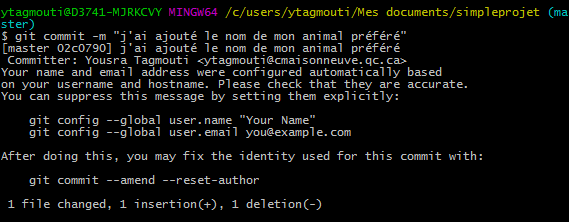
1. Et ensuite encore un **git status** :

Tapez **GIT STATUS**



1. Voilà qui est bien. On peut maintenant ajouter les changements à la base de données.

Tapez **GIT COMMIT -M** "j'ai ajouté le nom de mon animal préféré"

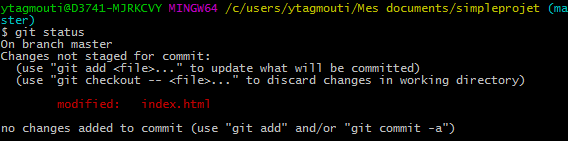


1. Nous avons effectué deux versions! Il est maintenant le temps d’en faire une troisième. **Ajoutez un dossier CSS dans lequel vous placerez une feuille de style. Votre page doit s’afficher sur fond bleu grâce à cette feuille de style.**
2. Ensuite, ajoutez tout cela avec git. Cette fois-ci, je vous laisse aller!

Nous commençons par regarder le statut

Tapez **GIT STATUS**

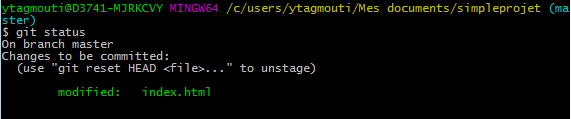
Git nous dit que notre fichier a été modifié, nous devons donc le placer sur le staging (la zone d’index) avec **git add**



Tapez **GIT ADD .**

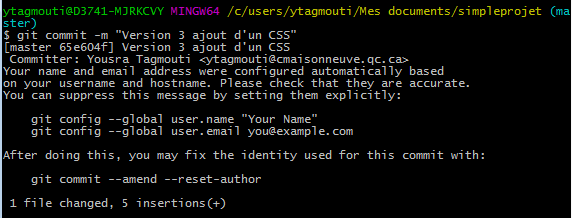


Et nous vérifions encore une fois le statut



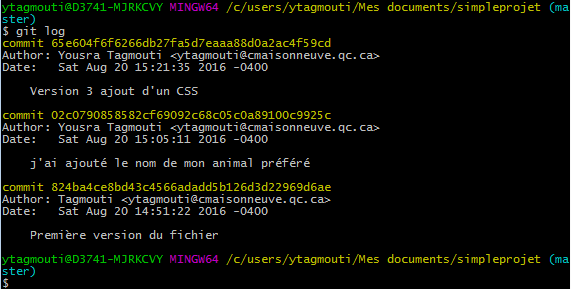
Nous pouvons maintenant ajouter les changements dans le dépôt de git

Tapez : **GIT COMMIT -M** "ajout d'un fichier CSS"



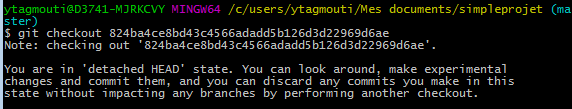
1. Maintenant, voyons l’historique!

Tapez **GIT LOG**



1. Vu que Git conserve un instantané des fichiers à chaque commit, on peut même revenir en arrière! Pour cela, on utilise «**git checkout** SHA1» où SHA1 représente la clé de hashage du commit. Voici un exemple :

Tapez **GIT CHECKOUT** SHA1



1. De quoi est fait le contenu HTML du fichier index.html?
2. Revenons à la dernière version.

