2 - Premier pas sous Git

Vous trouverez ci-dessous une série de commandes avec des descriptions pour chacune d'entre elles.

Si vous vous posez des questions je vous conseille de vous rendre sur la documentation officielle de git qui est très complète : https://git-scm.com/book/fr/v1/D%C3%A9marrage-rapide

2.1 Configurer git

Commande	git config
Description	Permet de configurer git sur son poste. Par défaut, la configuration ne pose pas de problème, il faut juste spécifier son nom d'utilisateur et son adresse e-mail afin de signer les modifications que vous effectuerez auprès des autres utilisateurs.
Exemple	$> git config --global user.name "Mon nom" $> git config --global user.email "mon@email.fr"
Retour de la commande	

2.2 Initialiser un projet git

Commande	git init
Description	Permet d'initialiser le dossier courant pour un faire un dépôt ou répo (pour repository en anglais) git. Le dossier courant devient ainsi le répertoire de travail (workspace en anglais) de votre nouveau projet git. Un dossier caché nommé « .git » devrait apparaître une fois la commande effectuée. Ne le supprimez surtout pas, il contient toutes les informations nécessaires au bon fonctionnement de git.
Exemple	$> cd /home/user/my-first-repo $> git init
Retour de la commande	Initialized empty Git repository in /home/user/my-first-repo/.git/

2.3 Clôner un dépôt existant

Commande	git clone
Description	Ceci téléchargera le référentiel « .git » sur votre ordinateur et extraira l'instantané actuel du référentiel (tous les fichiers) dans votre répertoire de travail. Par défaut, tout sera sauvegardé dans un dossier portant le même nom que le repo. L'URL que vous spécifiez ici s'appelle l'origine distante (c'est l'endroit où les fichiers sont téléchargés à l'origine). Ce terme sera utilisé plus tard.
Exemple	$> cd /home/user $> git clone https://github.com/github/gitignore.git
Retour de la commande	

2.4 Ajouter un fichier à l'index

Commande	git add
Description	Cette commande permet d'ajouter un ou plusieurs fichiers à l'index des fichiers à sauvegarder (git commit).
Exemple	*Ajouter le fichier mon_premier_fichier* $> git add mon_premier_fichier
	Ajouter le fichier README.md et tous les fichiers .txt du dossier app $> git add README.md app/*.txt
	*Ajouter tous les fichiers **unstaged*** $> git add .
Retour de la commande	

2.5 Supprimer un fichier de l'index

Commande	git rm
Description	Cette commande permet de supprimer un ou plusieurs fichiers de l'index des fichiers à commiter. Attention cette commande supprimera définitivement les fichiers de votre ordinateur, pour supprimer uniquement les fichiers de l'index sans les supprimer il faut utiliser l'option --cached
Exemple	*Supprimer totalement le fichier « mon_premier_fichier »* $> git rm mon_premier_fichier
	Supprimer le fichier « mon_premier_fichier » de l'index $> git rm --cached mon_premier_fichier
Retour de la commande	rm mon_premier_fichier

2.6 Voir le statut de votre projet

Commande	git status
Description	Affiche quelques informations de base comme la branche courante, les fichiers en cours de modifications, ou ceux ajoutés récemment. Vous devriez utiliser cette commande dès que vous avez un doute dur l'état d'un de vos dépôts.
Exemple	$> git status
Retour de la commande	On branch master Changes to be committed: (use "git reset HEAD ..." to unstage) deleted: toto Untracked files: (use "git add ..." to include in what will be committed) tata

2.7 Valider les modifications courantes d'une branche

Commande	git commit ou git commit -m "Add a new function"
Description	Par défaut la commande « git commit » va ouvrir votre éditeur de texte par défaut (vim, nano, emacs...) et va vous demander de confirmer votre commit avec un message explicatif concernant les changements que vous avez effectués. Dès que vous aurez sauvegardé votre message, le commit sera stocké localement. Vous pouvez directement passer le message à la commande en utilisant le flag -m. Vous avez également la possibilité de ne commiter que certains fichiers en passant leurs noms à la fin de la commande
Exemple	$> echo `date` > date.txt $> git add date.txt $> git commit -m "Add a date" date.txt
Retour de la commande	[master d8668c0] Add a date 1 file changed, 1 insertion(+) create mode 100644 date.txt

2.8 Créer une nouvelle branche

Commande	git branch
Description	La création d'une branche peut être apparentée à une copie à un instant « t » d'une branche existante, en anglais on parle de « snapshot ». Vous allez pouvoir donner un nom à cette branche pour la reconnaître facilement. Vous pourrez faire des modifications (git commit) sur la branche nouvellement créée de la même façon que vous avez effectué des modifications sur la branche initiale.
Exemple	$> git branch fonction_color_button
Retour de la commande	*none*

2.9 Rapatrier ou changer de branche

Commande	git checkout

Description	Changer de branche vous permettra de retourner vers une branche existante sur votre pc dans l'état exact où vous l'aviez laissée. Il faut bien prendre soin de valider les modifications (git commit) de la branche courante avant de changer de branche sinon vous ferez face à un message d'erreur vous indiquant que toutes les modifications n'ont pas été sauvegardées. Vous pouvez utiliser le flag -b pour créer une nouvelle branche et basculer dessus automatiquement.
Exemple	$> git checkout fonction_color_button
	$> git checkout -b fonction_color_field
Retour de la commande	Switched to branch 'fonction_color_button'
	Switched to a new branch 'fonction_color_field'

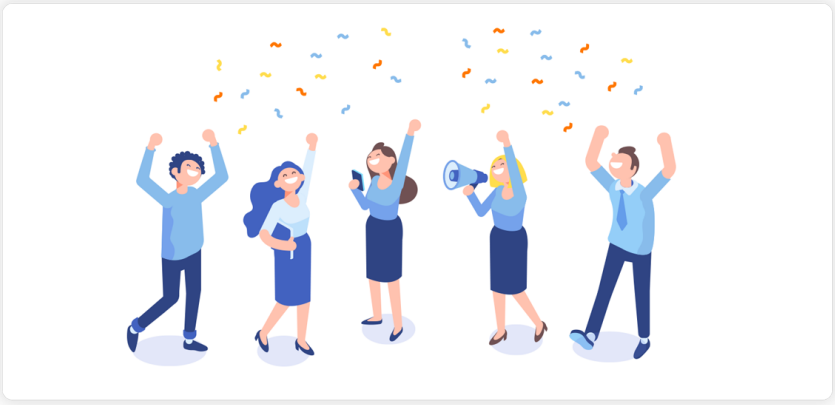
2.10 Comparer deux branches

Commande	git diff ou git diff
Description	Après avoir édité des fichiers et commité les modifications, vous pouvez facilement comparer deux branches pour en afficher les différences. Vous avez également la possibilité d'afficher les modifications faites sur la branche courante avant d'effectuer un commit. Pour cela il faut uniquement lancer la commande « git diff ». Pour chaque groupe de changement, vous verrez à quoi le fichier ressemblait (avec les suppressions en rouge avec un symbole -) et à quoi il ressemble maintenant (avec les ajouts en rouge avec un symbole +)
Exemple	$> echo "test1\ntest2" > my_first_test $> git commit -m "Add two test" my_first_test $> echo "test3" >> my_first_test $> git diff

Retour de la commande	diff --git a/my_first_test b/my_first_test index 055d7a6..eb49fe4 100644 --- a/my_first_test +++ b/my_first_test @@ -1,2 +1,3 @@ echo test1 echo test2 +echo test3

2.11 Pousser une branche sur dépôt distant

Commande	git push origin
Description	Cette commande va vous permettre d'envoyer votre branche locale sur le dépôt d'origine. Vous vous souvenez du répo que vous avez cloné lors de la commande git clone ? C'est lui votre dépôt d'origine. Une fois que vous aurez envoyé votre branche sur le répo d'origine, les autres développeur seront capables de la télécharger pour voir vos commits.
Exemple	$> git push origin new-branch-name
Retour de la commande	*none*



Maintenant, vous connaissez **toutes les commandes de base** pour bien utiliser git.

Je vous conseille de vous familiariser avec, car elles vous seront toutes utiles dans votre vie de DevOps.

Pour voir la fiche complète et les documents attachés, rendez-vous sur
https://elearning.26academy.com/course/play/5aa265f759a5ca3c65d5487b