Une fois que vous vous êtes placés dans votre nouveau dossier grâce à la commandecd de votre Terminal, créez un nouveau dossier 'monPremierRepo' en lançant la commande suivante :

mkdir monPremierRepo

 Pour activer un dossier comme repository Git, il suffit de se placer dans ce dossier avec le Terminal puis d'utiliser la commande **git init**.

 Pour gérer un repository, Git génère un index de tous les fichiers dont il doit faire le suivi. Lorsque vous créez un fichier dans un repository, vous devez donc l'ajouter à l'index Git à l'aide de la commande **git add nomDeVotreFichier.extension**. Par exemple :

git add checklist-vacances.md

Pour gagner du temps, vous pouvez ajouter tous les fichiers dans le répertoire courant en tapant

git add .

 Lorsque vous modifiez votre repository, vous devez demander à Git d'enregistrer vos modifications en faisant un **git commit**. L'option-m vous permet de lui envoyer un message décrivant les modifications effectuées, ce qui s'avèrera très utile pour vous par la suite, you'll see! :)Par exemple :

git commit -m "Ajouté ma checklist-vacances.md (woohoo!)"

Comment vous y retrouver dans l'historique de vos commits ?

Grâce à la commande  git log qui vous affiche la liste de tous les commits que vous avez réalisés !

Le sens de lecture du log est le suivant : on part du commit le plus récent en haut de la liste, au commit le plus vieux en bas de la liste.

Et bien, si vous ne faites que mettre à jour un fichier que vous aviez déjà ajouté à l'index, vous pouvez condenser ces deux étapes de la façon suivante :

git commit -a -m "Ajouté itinéraire dans checklist-vacances.md"

L'option   -a demande à Git de mettre à jour les fichiers déjà existants dans son index.

Lorsque vous effectuez une série de commits sur un projet, il peut vous arriver de vouloir remonter dans le temps à la recherche d'erreurs éventuelles par exemple. Pour vous positionner sur un commit donné de votre historique, il vous suffit d'utiliser la commande **git checkout** de la façon suivante :

git checkout SHADuCommit

Pour revenir à votre branche principale (au commit le plus récent), on utilise la même commande :

git checkout master

Je vous présente une de ces options : vous pouvez "revert" un commit, c'est-à-dire créer un nouveau commit qui fait l'inverse du précédent, avec la commande suivante :

git revert SHADuCommit

Sinon, si vous voulez simplement modifier le message de votre dernier commit, vous pouvez utiliser la commande suivante :

git commit --amend -m "Votre nouveau message"

... mais attention, vous ne pouvez modifier ce message que si vous n'avez pas encore pushé votre commit sur l'origine !

Je n'ai pas encore fait mon nouveau commit, mais je veux annuler tous les changements que je n'ai pas encore commités. Possible ?

 Possible avec un reset !

git reset --hard‌

À partir de GitHub, vous pouvez copier un repository sur votre machine. Pour cela, il vous suffit de rechercher le repository qui vous intéresse sur GitHub, de vous y placer, puis d'utiliser l'option "clone URL" en bas à droite de l'écran.

Cette option vous propose un lien SSH, HTTPS ou Subversion. Ici, nous allons choisir un lien HTTPS, le copier, puis coller ce lien en utilisant la commande **git clone** dans le dossier que vous aurez choisi sur votre machine :

git clone lienFourniParGitHub

Dans cette vidéo, nous avons cloné le repo de la librairie [React.js](https://github.com/facebook/react), une librairie créée par Facebook et qui permet de mieux gérer ses interfaces graphiques grâce à JavaScript. Nous avons donc effectué sur notre machine la commande :

git clone https://github.com/facebook/react.git

Vous avez clôné votre repo GitHub sur votre machine. Comment faire pour synchroniser les modifications que vous faites dans votre repo sur votre machine avec votre repo sur GitHub ?

* *À vos marques...* Ouvrez votre terminal et placez-vous dans votre repo local.
* *Prêts...* Faites un/des commit(s) des modifications que vous avez ajoutées sur ce repo.
* *Pushez!*Envoyez ces modifications sur votre repo GitHub en utilisant la commande git push:

git push origin master

Dans le chapitre précédent, vous avez vu comment envoyer vos modifications locales vers votre repo GitHub avec **git push**. Mais si vous modifiez votre repo GitHub en ligne, ou si vous travaillez avec d'autres personnes dessus et qu'elles envoient leurs modifications locales sur le repo en ligne, votre code local ne sera plus à jour.

Pour récupérer en local les dernières modifications du repo GitHub, il vous faut utiliser la commande **git pull** :

git pull origin master