# Exercices - Git

Pour ces exercices, en plus de taper dans le terminal vos commandes, veuillez me rendre le répertoire de votre dépôté zipé.

Vous m'enverrez via WhatsApp vos exercices.


- Exercice n°1
- Exercice n°2


## Exercice n°1 - Creation / indexation / validation

Le nom du fichier zip aura la structure suivante**prenom_ex1.zip**. Par exemples: **johnny_ex1.zip** ou encore **bruno_ex1.zip**.

### 1.1 - Création d'un dépôt

1. Créez un répertoire nommé PremierDepot
2. Faites-en un dépôt à l'aide d'une commande git bien entendu.
3. Quelle commande utiliser pour vérifier que le répertoire est bien devenu un dépôt ?

### 1.2 - Modification (Ajouter de fichiers)

Pour éviter des alertes de sécurité de Chrome, j'ai nommé les fichiers avec l'extension .txt. Nous les renommerons plus tard avec la bonne extension.

1. Dans le répertoire que vous venez de créer, copiez les 3 fichiers suivants:

- index.txt
- genius.txt
- display.txt

2. Faites un git status. (Que vous dit/signale git ?)

### 1.3 - Indexation

1. Indexez le fichier index.txt avec git.
2. Indexez les autres fichiers.
3. Vérifiez avec la bonne commande git si les 2 indexations se sont bien passées. (Comment le voyez-vous ?)

### 1.4 - Validation/Commit

1. Validez les fichiers présents dans la zone d'index avec le message suivant:"Commit initial" (C'est souvent le premier message que l'on donne pour le tout premier commit)
2. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
3. Tapez la commande git tag v1 (Je vous expliquerai plus tard dans le cours l'utilité de git tag. Donc, comme un robot, exécutez les "ordres" et tapez git tag v1 :-) )

## 1.5 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.6 - Modifications (Renommer des fichiers)

Ici les modifications vont porter sur le fait que l'on va renommer nos fichiers.

Pour rappel, voilà comment renommer un fichier en ligne de commandes:

1. Windows: ren fichier1 fichier2 (va renommer le fichier1 avec le nom fichier2)
2. Mac Os/Linux: mv fichier1 fichier2 (va renommer le fichier1 avec le nom fichier2)

Renommez les fichiers de la manière suivante:

1. index.txt devient index.html
2. genius.txt devient genius.js
3. display.txt devient display.js
4. Évidemment vérifiez avec dir (windows) ou ls (mac/linux) s'ils sont bien renommés.

### 1.7 - Indexation

1. Faites un git status
2. Indexez les fichiers modifiés.
3. Refaites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.8 - Validation/Commit

1. Faites un git status
2. Validez/Commitez vos fichiers en ajoutant comme message “Renommage d’index, genius et display”
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v2

## 1.9 - Affichage de l’historique des commits

Affichez l’historique des commits à l’aide de la commande: git log

Vous verrez l’ensemble des commits effectués avec la date, l’heure, l’auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.10 - Modification (de code)

Explication des différents fichiers:

- index.html: il sert à afficher une page qui affiche le résultat de calculs.
- genius.js: c’est le fichier qui contient nos fonctions javascript de mathématique: add et substract.
- display.js: il contient une fonction display qui permet d’afficher le résultat d’un calcul à l’écran. Vous ne devez pas y toucher !

L’exercice:

Vous devrez ajouter une fonction qui multiplie deux nombres dans genius.js et l’utiliser dans index.html. Le code devra être fonctionnel. C’est à dire qu’à l’affichage dans index.html, votre fonction calcule bien la multiplication de deux nombres et affiche à l’écran le résultat. Donc index.html permettra de vérifier à l’affichage de la page web si votre code est bon.

1. Editez le fichier genius.js
2. Ajoutez la fonction multiply qui retourne (return) la multiplication des deux nombres donnés en paramètre dans la signature de la fonction multiply.
3. Editez le fichier index.html
4. Appelez la fonction display pour qu’elle affiche la multiplication de a par b (a * b).
5. Pour l’utilisation de display dans index.html, inspirez-vous des deux lignes précédentes qui font la soustraction et l’addition de a et b.
6. Testez votre programme en lançant index.html et voyez le résultat.
7. Si votre programme fonctionne allez au point suivant sinon cherchez votre erreur.
8. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.11 - Indexation

1. Faites un git status
2. Indexez les fichiers modifiés.
3. Refaites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.12 - Validation

1. Faites un git status
2. Validez/Commitez vos fichiers en ajoutant comme message “Ajout de la fonction multiply et adaptation du fichier index.html”
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v3

## 1.13 - Affichage de l’historique des commits

Affichez l’historique des commits à l’aide de la commande: git log

Vous verrez l’ensemble des commits effectués avec la date, l’heure, l’auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 1.14 - Modification (de code)

Vous devrez ajouter une fonction qui divise deux nombres dans genius.js et l’utiliser dans index.html. Le code devra être fonctionnel. C’est à dire qu’à l’affichage dans index.html, votre fonction calcule bien la division de deux nombres et affiche à l’écran le résultat. Donc index.html permettra de vérifier à l’affichage de la page web si votre code est bon.

1. Editez le fichier genius.js
2. Ajoutez la fonction divide qui retourne (return) la division des deux nombres donnés en paramètre dans la signature de la fonction divide. N'oubliez pas que l'on ne peut diviser par 0 en mathématiques. Donc il faudra tester que le diviseur est bien différent de 0 avant de retourner le quotient.
3. Editez le fichier index.html
4. Appelez la fonction display pour qu'elle affiche la division de a par b (a / b).
5. Pour l'utilisation de display dans index.html, inspirez-vous des trois lignes précédentes qui font la soustraction, l'addition et la multiplication de a et b.
6. Testez votre programme en lançant index.html et voyez le résultat.
7. Si votre programme fonctionne allez au point suivant sinon cherchez votre erreur.
8. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.15 - Indexation

1. Faites un git status
2. Indexez les fichiers modifiés.
3. Refaites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)

### 1.16 - Validation

1. Faites un git status
2. Validez/Commitez vos fichiers en ajoutant comme message "Ajout de la fonction divide et adaptation du fichier index.html"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v4

## 1.17 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

# Exercice n°2 - Creation / indexation / validation

### 2.1 - Création d'un dépôt

1. Créez un répertoire nommé Recettes
2. Allez dans le répertoire Recettes
3. Faites-en un dépôt à l'aide d'une commande git bien entendu.
4. Quelle commande utiliser pour vérifier que le répertoire est bien devenu un dépôt ?

### 2.2 - Ajout d'une recette et Indexation

1. Allez dans le répertoire Recettes
2. Copiez le fichier omelette.txt dans votre répertoire Recettes.
3. Faites un git status, vous verrez que git a bien détecté notre nouveau fichier.

### 2.3 - Indexation

1. Indexez notre recette à l'aide d'une commande git.
2. Vérifiez à l'aide d'une commande git que celle-ci est bien indexée.

### 2.4 - Validation

1. Faites un git status
2. Validez/Commitez votre fichier en ajoutant comme message "Ajout de la recette de base de l'omelette"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v1

## 2.5 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 2.6 - Modification

Vous vous sentez l'âme d'un grand cuistot. Et vous avez trouvé que votre recette est meilleure avec du gruyère.

1. Ouvrez le fichier omelette.txt
2. Ajoutez la ligne suivante: 30g de gruyère râpé
3. Ajoutez la ligne suivante: une pincée d'origan.
4. Vous sauvegardez votre recette modifiée.

### 2.7 - Indexation

1. Faites un git status et vous verrez que git détecte bien qu'il y a eu modification dans votre recette.
2. Indexez notre recette modifiée.
3. Refaites un git status.

### 2.8 - Validation

1. Faites un git status
2. Validez/Commitez votre fichiers en ajoutant comme message "Ajout du gruyère et l'origan dans la recette de l'omelette"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v2

## 2.9 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 2.10 - Modification

Votre grand-mère est de passage chez vous et goûte votre super omelette. Vous la voyez faire la grimace et vous dire:"Ne me dis pas que tu fais tes omelettes sans mettre un petit peu de lait dedans ? Tu ne savais pas que c'était meilleur ?!"

1. Ouvrez le fichier omelette.txt
2. Ajoutez la ligne suivante: un petit peu de lait.
3. Vous sauvegardez votre recette modifiée.

### 2.11 - Indexation

1. Faites un git status et vous verrez que git détecte bien qu'il y a eu modification dans votre recette.
2. Indexez notre recette modifiée.
3. Refaites un git status.

### 2.12 - Validation

1. Faites un git status
2. Validez/Commitez votre fichiers en ajoutant comme message "Ajout un peu de lait dans la recette de l'omelette"
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v3

## 2.13 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

### 2.14 - Modification

Aujourd'hui, vous sentez que vous êtes parti à faire beaucoup de recettes de cuisine. C'est en vous, vous êtes un futur chef ! :)

Mais, comme tout chef, il faut être organisé avec vos recettes. C'est pourquoi chaque recette sera placée dans un répertoire de l'ingrédient principal.

1. Créez un répertoire oeufs.
2. Déplacez votre recette omelette.txt dans le répertoire oeufs.

Vous avez deux manières de déplacer ce fichier: - soit via la commande Windows move omelettes.txt oeufs ou sous Mac mv omelettes.txt oeufs - soit via l'explorateur de fichiers

### 2.15 - Indexation

1. Faites un git status et vous verrez que git détecte bien qu'un répertoire a été créé et que votre recette omelette.txt a été déplacée dans le répertoire oeufs.
2. Indexez toutes les modifications en une seule commande.
3. Refaites un git status.

### 2.16 - Validation

1. Faites un git status
2. Validez/Commitez votre fichiers en ajoutant comme message “Déplacement de la recette de l'omelette dans le répertoire oeufs”
3. Faites un git status (vous voyez ce qui a changé depuis le point 1 ? Quoi ?)
4. Tapez la commande git tag v4

## 2.17 - Affichage de l'historique des commits

Affichez l'historique des commits à l'aide de la commande: git log

Vous verrez l'ensemble des commits effectués avec la date, l'heure, l'auteur et le message de commit.

Le nouveau commit que vous venez de faire doit y figurer.

---