**Christophe Fessard**

**Contrôle de version avec git et github**

BDD

**Introduction au contrôle de version**

# ****Qu'est-ce qu’un gestionnaire de versions**** ****?****

Un gestionnaire de versions est un programme qui permet aux développeurs de conserver un historique des modifications et des versions de tous leurs fichiers.

Le gestionnaire de versions permet de tracer (persister en mémoire) :

* Une image contenant texte, diagramme, capture d’écran, croquis

  Description générée automatiquementchaque modification de chaque fichier ;
* pourquoi elle a eu lieu ;
* par qui !

Cet outil a trois grandes fonctionnalités :

1. Revenir à une version précédente de votre code en cas de problème.
2. Suivre l’évolution de votre code étape par étape.
3. Travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs.

Il existe différents types de modifications : ajouter, modifier, supprimer.

Deux types de contrôle de version : centralisé / distribué.

# ****Un peu de vocabulaire****

**Git est un gestionnaire de versions**. Vous l’utiliserez pour créer un dépôt local et gérer les versions de vos fichiers.

**GitHub est un service en ligne** qui va héberger votre dépôt. Dans ce cas, on parle de **dépôt distant** puisqu’il n’est pas stocké sur votre machine.

**Qu'est-ce qu’un Dépôt** **?**

Un dépôt est comme un dossier qui conserve un historique des versions et des modifications d’un projet. Il peut être local ou distant.

**Le dépôt local**

Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquementUn **dépôt local** est un entrepôt virtuel de votre projet. Il vous permet d'enregistrer les versions de votre code et d'y accéder au besoin. C’est sur votre dépôt local que vous ferez toutes vos modifications de code.

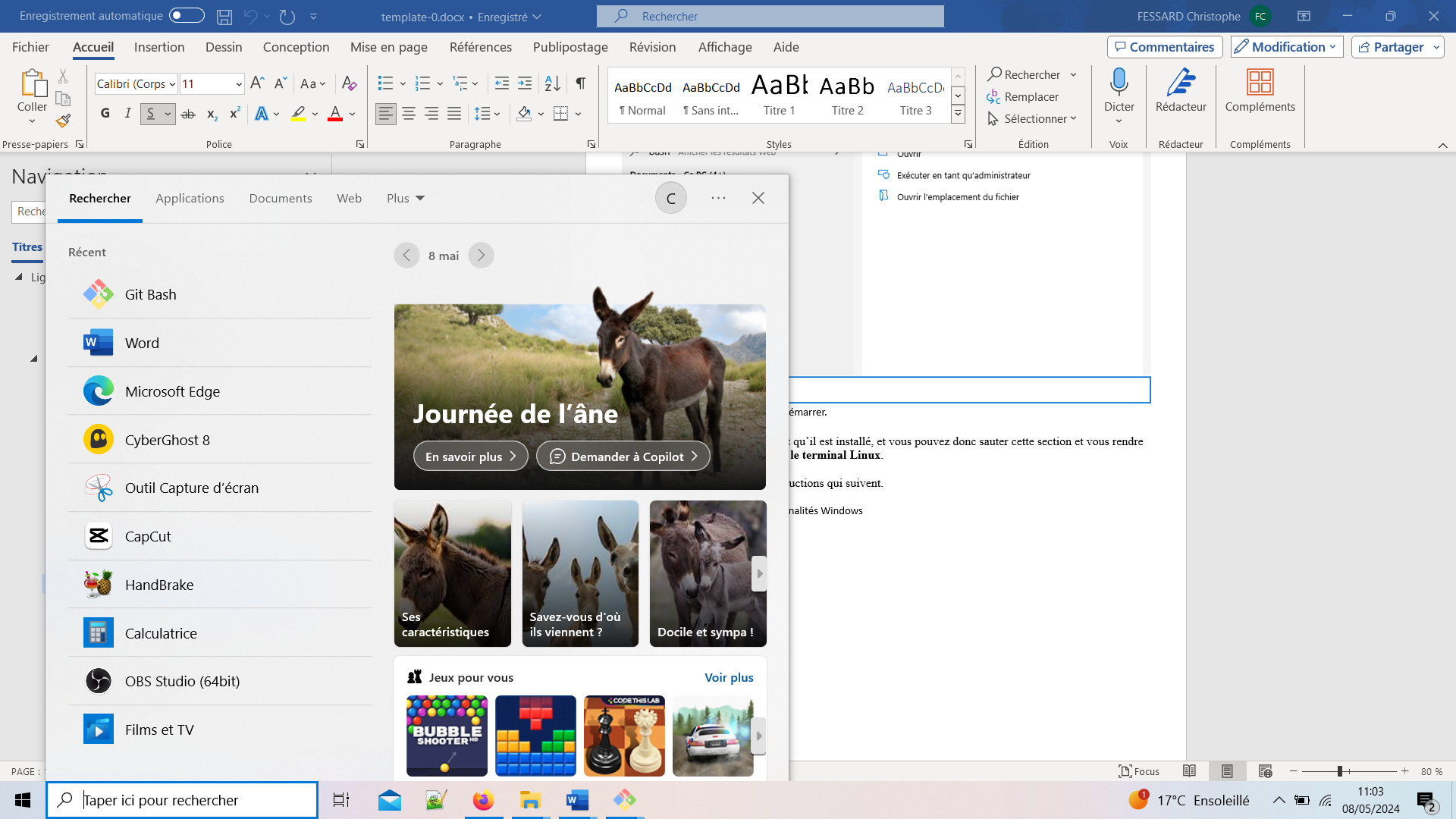
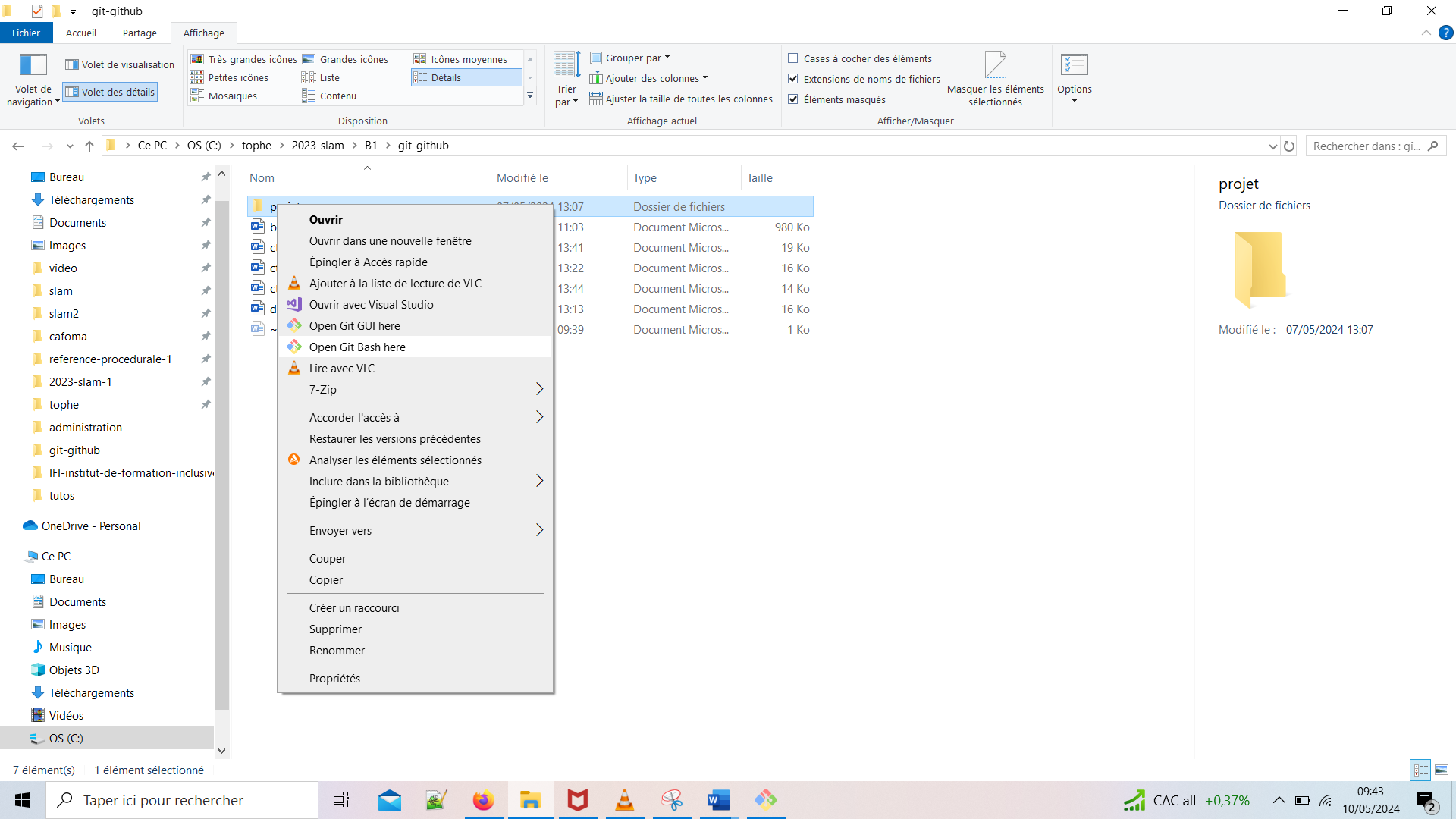
**Le dépôt distant**

Le **dépôt distant** est un peu différent. Il permet de stocker les différentes versions de votre code afin de garder un **historique délocalisé**, c'est-à-dire un historique hébergé sur Internet ou sur un réseau. Il permet de centraliser le travail des développeurs dans un projet collectif.

# Installer git

<https://git-scm.com/download/>

=> lancer « git bash »

  Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

Créez un répertoire (votre projet)

Dans le répertoire, via le menu contextuel, ouvrir gitbash

**Ligne de commande linux**

# Quelques commandes

| **Commande** | **Utilisée pour** |
| --- | --- |
| cd | Changer de répertoire |
| ls | Commande de liste utilisée pour afficher le contenu d'un répertoire. |
| rm | Commande permettant de supprimer un fichier ou un répertoire |
| mv | Permet de déplacer des fichiers ou des dossiers vers un autre emplacement |
| touch | Permet de créer un nouveau fichier vide ou de mettre à jour l'horodatage d'un fichier |
| cp | Permet de faire une copie d'un fichier ou d'un dossier |
| mkdir | Permet de créer un nouveau répertoire |
| pwd | Impression du répertoire de travail, indique l'emplacement actuel dans l'interpréteur de commandes |
| cat | Permet la lecture ou la concaténation d'un fichier |
| less | Affiche le contenu d'un fichier une page à la fois. |
| grep | Expression régulière globale permettant de rechercher le contenu d'un fichier ou d'un dossier |

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Créer un répertoire de travail

Editeur vim =>

i : mode insertion

Escape : sortir mode insertion

:wq! : enregistrer et fermer le fichier

Créer et ouvrir un fichier

Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Saisir dans testshell.sh =>

echo "Bonjour Lina"

Une image contenant texte, capture d’écran, Police

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement Une image contenant texte, capture d’écran, Police

Description générée automatiquement

cat = afficher le fichier

wc -w = compter le nombre de mots

Une image contenant texte, Police, capture d’écran

Description générée automatiquement Une image contenant texte, Police, capture d’écran, typographie

Description générée automatiquement

# Tuyau (pipe)

= un outil de codage qui permet d'utiliser la sortie d'une commande comme entrée pour une autre commande.

Une image contenant texte, Police, écriture manuscrite, capture d’écran

Description générée automatiquement Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Modifier test1.txt, copier =>

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum

Modifier test2.txt, copier =>

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Redirection

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

La redirection permet de modifier l’entrée et/ou la sortie standard.

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement3 types de redirections :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquementRemarque : ctrl/d pour mettre fin à la saisie.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Filtrage (grep)

= Global Regular Expression Print

=> faire des recherches dans des fichiers, des dossiers, ainsi que dans le contenu des fichiers.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Récupérer le fichier prénoms.txt**

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquementUne image contenant texte, Police, capture d’écran, ligne

Description générée automatiquement

Mot exacte =>

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

**Installer et configurer github**

# Créez votre compte GitHub

**Étape 1 :** Allez sur [Github](https://github.com/) dans votre navigateur web (https://github.com/)

**Étape 2**: Cliquez sur le bouton Sign-up en haut à droite de l'écran.

**Étape 3 :** Saisissez votre adresse électronique

**Étape 4**: Choisissez un mot de passe fort

**Étape 5**: Entrez un nom d'utilisateur

**Étape** **6**: Répondez à la question de sécurité

**Étape 7**: CliquezCreate Account

**Étape 8 :** Vous recevrez ensuite un courriel de confirmation pour confirmer votre adresse.

Cet e-mail contiendra un code. Saisissez ce code sur l'écran de confirmation.

## Connexion à GitHub via HTTPS

Lorsque vous utilisez Github, il est nécessaire de s'authentifier à l'aide d'un jeton d'accès personnel via HTTPS.

L’authentification par jeton est une **forme d’authentification** qui permet à un utilisateur d’accéder à un service en ligne, une application, ou un site web sans qu’il n’ait à ressaisir ses identifiants.

Lorsque vous avez fini d'utiliser le jeton, vous pouvez le révoquer pour qu'il ne puisse plus être utilisé.

Il est également possible de fixer un délai d'expiration pour le jeton.

=> Cela permet de sécuriser votre compte.

# Générer un jeton d'accès personnel

**Étape 1 :** Connectez-vous à Github

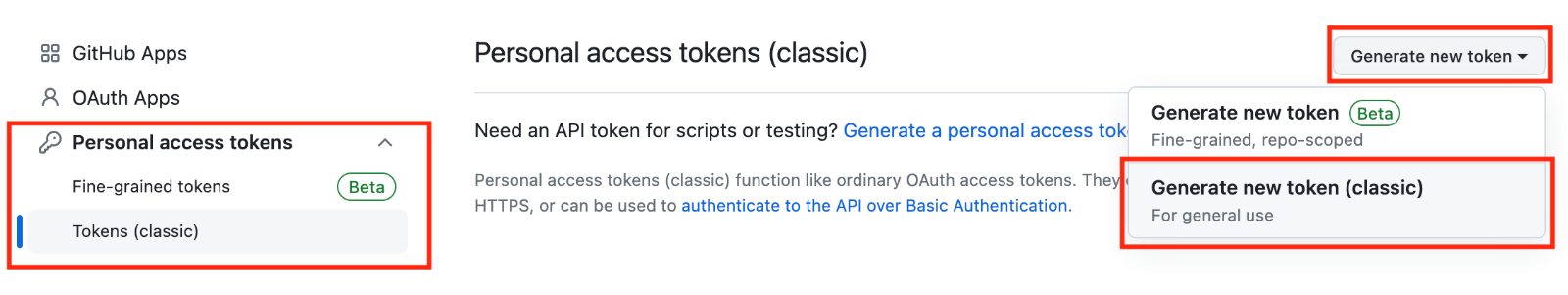
**Étape 2 :** Cliquez sur l'icône de profil en haut à droite de l'écran et sélectionnez Settings.

**Étape 3**: Dans l'écran Paramètres, cliquez sur Paramètres du développeur dans la partie gauche de l'écran.

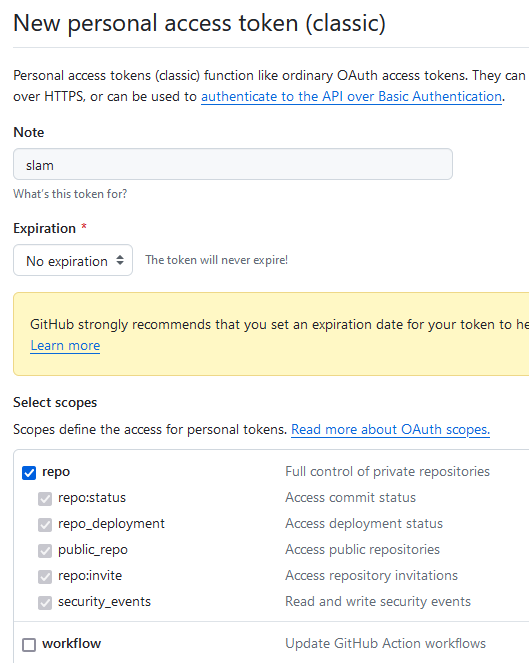
Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Étape 4 :** Sur l'écran Paramètres du développeur, cliquez sur **Jetons d'accès personnels**. Cliquez ensuite sur **Tokens (classic**). Cliquez ensuite sur le bouton **Générer un nouveau jeton** et sélectionnez **Générer un nouveau jeton (classique**).

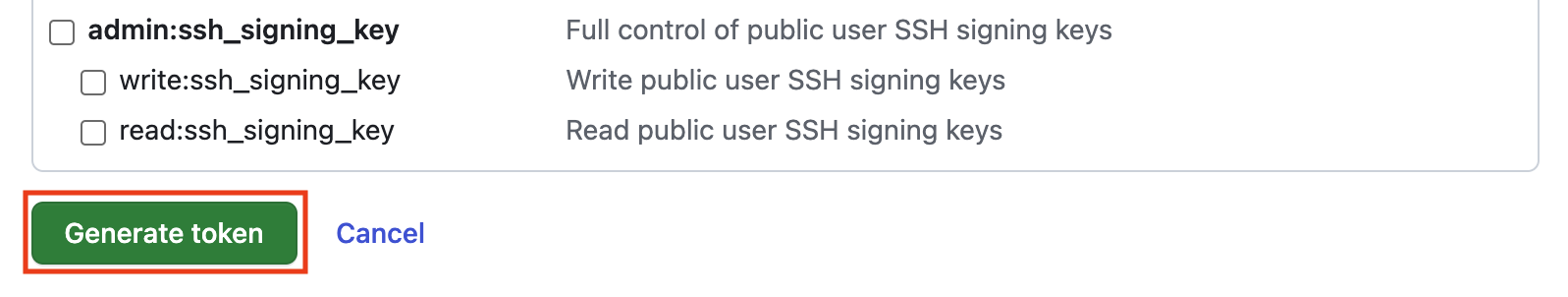


**Étape 5 :** Sur la page Nouveau jeton d'accès personnel, entrez un nom de jeton et un délai d'expiration. Si vous souhaitez révoquer manuellement le jeton, définissez le délai d'expiration surNo Expiration.



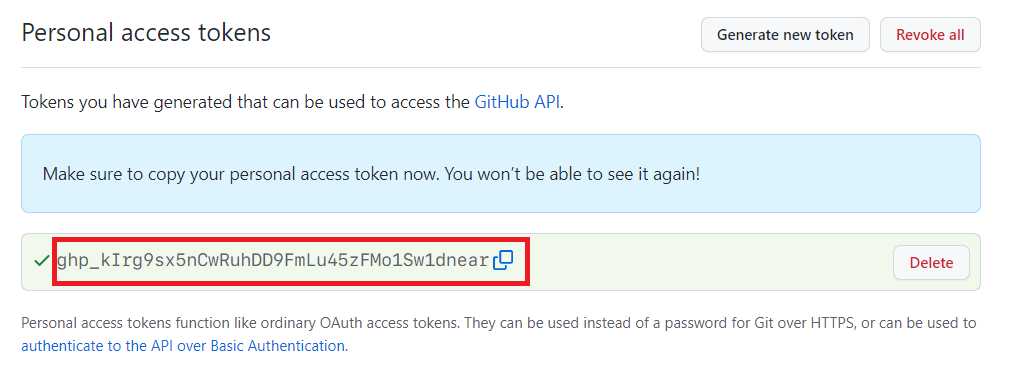
**Étape 6 :** Sous scopes, sélectionnez *repo*.

**Étape 7 :** Faites défiler la page jusqu'à la fin et cliquez sur Generate token.



**Étape 8 :** Le jeton est maintenant généré. Veillez à le copier et à le noter, car il sera masqué lorsque vous quitterez la page.

Ce jeton peut maintenant être utilisé pour se connecter à un référentiel via HTTPS.



**Remarque** : Si vous perdez le jeton, vous pouvez supprimer l'ancien jeton et en créer un nouveau.

**Travailler avec git et github**

# Créer un référentiel

=> <https://github.com/new>

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

# Cloner un référentiel

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

**Utilisez le protocole https pour cloner le projet**

=> copiez l’adresse https

Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

# ****Appréhendez le fonctionnement de Git****

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

Le Working directory = dossier du projet sur votre ordinateur. Répertoire de travail

Le Stage ou index = tous les fichiers modifiés qui serons dans la prochaine version de code. Zone de préparation.

Repository = nouvelles versions d’un projet

# Flux de travail git

Une image contenant texte, Police, capture d’écran, nombre

Description générée automatiquement

Staged area = zone planifiée => suivre les modifications des fichiers placé dans cette zone.

Commit : la validation d’un fichier s’apparente à un point de sauvegarde (nouvelle version du fichier).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

.git = dossier caché, utilisé pour suivre toutes les modifications.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

# Ajouter et valider des fichiers

Vérifier si des modifications ont été effectuées.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

=> branche principale

=> les fichiers sont exactement les même que sur github.

=> branche principale

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Créer un fichier test.txt



git add <fichier> => permettre à git de suivre les modification du fichier (git add .).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Validation des évolutions

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Branches

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Git checkout => créer la nouvelle branche + déplace l’utilisateur sur celle-ci

Créer un fichier test2.txt et ajouter au dépôt local :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Flux de travail github

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

Git push => transfert les modifications sur le « repository » distant.

=> s’authentifier (https)

# Pull request

**Sous github**

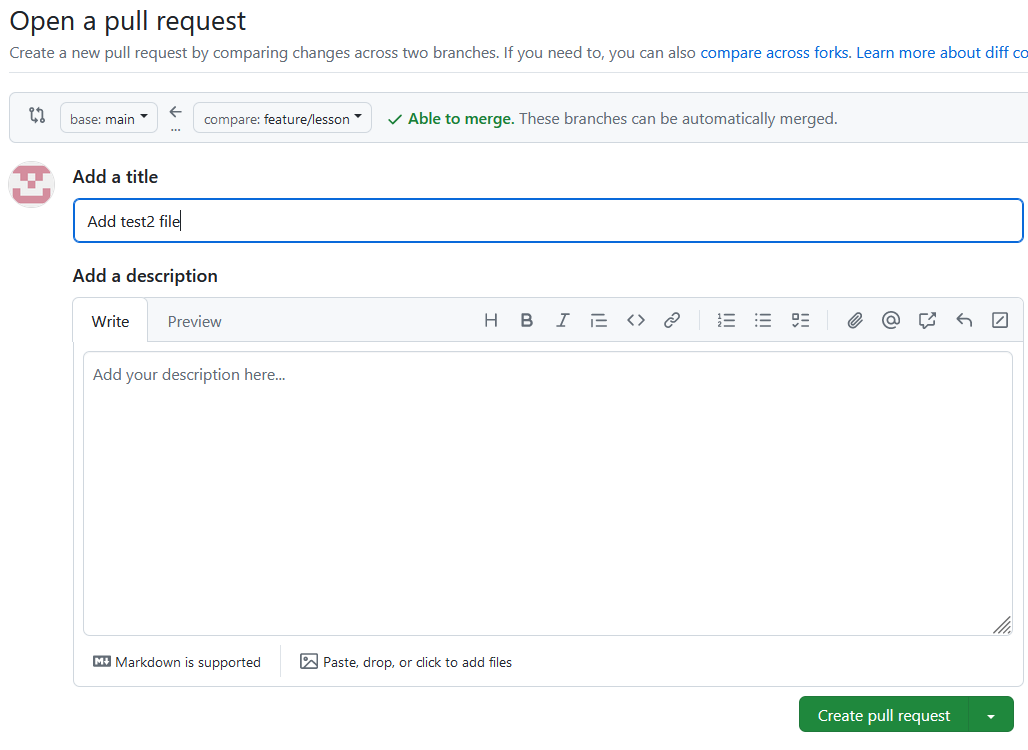
Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Pull request = permet d’obtenir une demande d’examen par les pairs des modifications apportées sur la branche

=> valider les modifications avant de les insérer dans la branche principale.

=> sélectionner « Compare & pull request »



Sélectionner « create pull request »

# Fusionner sur la branche principale

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

L’équipe devra valider les modifications ou les refuser, et le cas échéant les fusionner sur la branche principale.

Lorsque vous créez une demande de validation, vous demandez aux autres développeurs de revoir votre travail et de l'approuver pour qu'il soit fusionné avec le dépôt.

Comme nous n’avons pas de responsable « fusion » nous validons nous même les modifications =>

Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement

Une fois fusionnée, je peux supprimer la branche ou pas => dépend du processus définit dans le projet.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

# Une image contenant texte, capture d’écran, Police Description générée automatiquementMettre à jour le répertoire local

git checkout main => consulter ma branche principale

git pull => reçoit toutes les modifications qui ont été effectuées sur le répertoire distant.

Une image contenant texte, Police, capture d’écran, Graphique

Description générée automatiquement

test2.txt a été chargé.

# ICI => ?????????????

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

git init : initialisez votre répertoire courant en local.

Une image contenant Police, texte, capture d’écran, Graphique

Description générée automatiquement

=> retour vide = le repository(réfrentiel) local n’est pas connecté à un repository distant.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

my-first-repo = répertoire créé auparavant.

git remotre – v => vérifie que le réperoire local est connecté à un répertoire distant.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

git remote add oringin => ajoute une connexion à un dépôt distant

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

git pull => récupère toute les modifications du référentiel

Attention, à cette étape le répertoire local est vide, je n’ai pas encore configurer la branche principale en locale.

git checkout main => configurer une branche principale comme suivant la branche principale distante.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

# Commande push

* Push/pull seules commandes pour communiquer avec le dépôt distant.

Modifier test2.txt

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Toujours vérifier que vous êtes à jour / au repository distant (éviter les conflits).

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Git push => transfert les modifications locales vers le référentiel distant.

Une image contenant texte, Police, logo, Graphique

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Lorsque vous utilisez « git push », Git compare un instantané de votre dépôt local avec le dépôt distant et ne remplace que les fichiers qui ont été modifiés.

Remarques :

* s’il y a des conflits la fusion sera refusée
* avant de lancer un « push », il est recommander de faire un « pull » pour obtenir la dernière version afin de réduire les risques de conflit

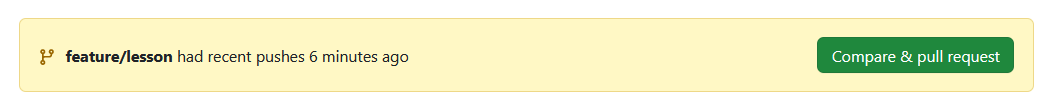
# Commande pull

Modifier sur le site github le fichier test.txt.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

* « commit change »



Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

=> valider les modifications sur github.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement



Git pull => fusionne les modifications distantes avec le code en locale (si pas de conflit).

# Exemple de flux de travail

Une image contenant ligne, cercle, capture d’écran

Description générée automatiquement

Une image contenant diagramme, ligne, capture d’écran, cercle

Description générée automatiquement

Après validation des évolutions et modifications les branches principale et secondaires peuvent être fusionnées.

Une image contenant ligne, cercle, capture d’écran, diagramme

Description générée automatiquement

**Workflow dans git et github**

=> avant de créer une branche, toujours s’assurer que votre code est à jour (pull) = git pull

Créer une nouvelle branche :

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Ajouter du nouveau contenu à cette branche, par exemple :

Une image contenant Police, texte, Graphique, capture d’écran

Description générée automatiquement

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

La branche que vous avez créée n'est visible que pour vous. Vous devez valider vos modifications et pousser les modifications acceptées vers le serveur principal avant que les membres de votre équipe ne puissent les voir.



# Pointeur « HEAD »

Dossier .git : est responsable de tracer toutes les modifications du projet.

Pointeur HEAD = pointe sur la branche courante, est un fichier du répertoire .git

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement.

=> id haché qui identifie le commit actuel.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Changeons de branche :

Revenons dans la branche principale.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement



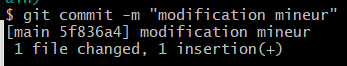
=> Modifier le fichier test.txt



Mettre à jour les commit =>

Une image contenant texte, Police, capture d’écran, horloge

Description générée automatiquement





Chaque fois qu'une modification est apportée à un commit, l'identifiant unique haché est mis à jour et devient le dernier commit pour ce répertoire de travail.

# Commandes de diffusion

= comparer les différentes versions des fichiers, branches et commit.

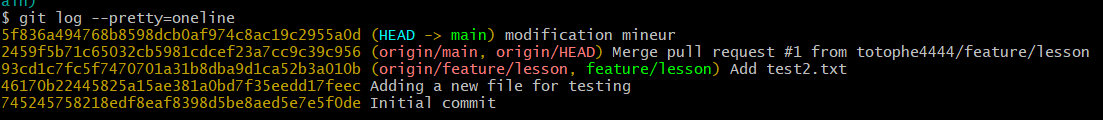


Modifier une nouvelle fois le fichier test.txt :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

=> en rouge le texte d’origine



Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

Comparer des branches =>

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Les fichiers individuels, les branches et les commits peuvent tous être comparés avec Git diff.