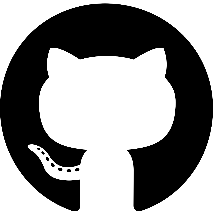
*Gestion de versions d’application avec Git/Github*



|  |  |
| --- | --- |
| Propriétés | Description |
| **Intitulé long** | *Gestion et suivi des modifications* |
| **Formation concernée** | BTS Services informatiques aux organisations |
| **Bloc** | Bloc 2 SLAM – Conception et développement d’applications |
| **Présentation** | Cette séquence permet de se familiariser avec les commandes de base de l’outil de gestion de versions Git qui pourraient être utilisées dans le cadre d’un projet pour réaliser le suivi des modifications effectuées au cours du développement d’une application logicielle : sélection des fichiers à suivre, validation des modifications, gestion de branches, ….  La séquence est un prérequis pour les laboratoires traitant de l’intégration et du déploiement continu d’applications avec Jenkins.  La séquence est réalisée dans un environnement Linux. La machine virtuelle Ubuntu pour la réalisation de la séquence est fournie dans le package. |
| **Compétences** | Bloc 2 SLAM : intégrer en continu les versions d’une solution applicative |
| **Transversalité** |  |
| **Prérequis** | Connaissance des commandes de base Linux  Concepts théoriques liés à Git (fiche savoir) |
| **Outils** |  |
| **Mots-clés** | Gestion de versions, Git, Github |
| **Durée** | 3h |
| **Niveau de difficulté** | Intermédiaire |
| **Auteur(es)** | Zakari BERREMILI avec la relecture de Amal HECKER et Maelle TAURAND |
| **Version** | V7.0 |
| **Date de publication** | 25/09/2023 |
| **Contenu du package** | Machine virtuelle Lubuntu  Ce labo fait partie de l’atelier CI/CD avec Jenkins |

Utilisation de Git et GitHub

Objectifs :

Cette séquence a pour objectif de se familiariser avec les commandes Git qui permettent d’effectuer le suivi des modifications apportées dans les fichiers d’une application et l’utilisation de la plateforme d’hébergement Github pour le partage de ces versions avec d’autres développeurs dans le cadre d’un projet logiciel collaboratif.

Dans cette séquence, vous allez :

* créer un dépôt local et vous initier aux commandes principales de Git pour suivre les modifications apportées à un fichier ;
* créer des branches d’une même application et gérer les conflits éventuels lors de fusion de branches ;
* travailler avec un dépôt distant et synchroniser les modifications apportées en local avec le dépôt distant ;

Les concepts théoriques liés à la séquence sont étudiés dans la fiche savoir X.

Scénario

**Etape 1 : Configuration et initialisation de Git**

**Etape 2 : Initiation à Git**

**Etape 3 : Gestion d branches**

**Etape 4 : Intégration de Git avec GitHub**

Environnement technique

Dans cette séquence, vous utiliserez une machine Ubuntu ou comme nous une machine Lubuntu demandant peu de ressources. (ID : rootsio MDP : Sio1234\*)

Etape 1 : Configuration de base et initialisation de Git

Dans cette partie, vous allez configurer et initialiser un dépôt local Git.

Tâche n°1 : Ouvrir un terminal

Tâche n°2 : Installer Git

Tâche n°3 : Configurer votre identité pour exploiter Git

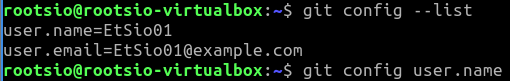
* ~$ git config --global user.name "Exemple d’utilisateur"
* ~$ git config --global user.email test@exemple.com

**Remarque** : ces paramètres peuvent être tout ce que vous voulez à ce stade. Toutefois, lorsque vous réinitialisez ces valeurs globales dans l’étape 4, vous utiliserez le nom d'utilisateur pour votre compte GitHub. Si vous le souhaitez, vous pouvez utiliser votre nom d'utilisateur GitHub dès maintenant.



Tâche n°4 : Vérifier la bonne configuration

* ~$ git config --list



**Question** : Est-il possible de n’afficher que l’user.name ou l’user.email avec cette commande ? Par quelle syntaxe ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tâche n°5 : Pour initialiser un répertoire en tant que dépôt Git, il faut lancer la commande git init dans le répertoire en question. Si la commande s’exécute correctement, un répertoire caché du nom de .git est créé dans le répertoire en question :

1. Créer un sous-répertoire git-intro dans le répertoire.

2. Se déplacer dans le répertoire créé.

3. Initialiser le répertoire git-intro en tant que dépôt git.

4. Quel est le message affiché par la commande ?

5. Vérifier la création du répertoire .git dans ~/git-intro

Le message affiché indique que vous avez créé un référentiel local dans votre projet contenu dans le répertoire masqué .git. C'est là que se trouve l'historique de vos modifications. Vous pouvez le voir avec la commande ls -a.



Tâche n°7 : Afficher les fichiers modifiés dans le répertoire de travail :

1. Lancer la commande git status dans le répertoire ~/git-intro.

**Question** : Quel est le message affiché ?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ce message vous indique :

* + Que vous êtes sur la branche master.
  + Aucune mise en scène (modification) à mettre à jour.

Le statut du dépôt Git change une fois que les fichiers sont ajoutés et modifiés.

Etape 2 : Suivi des modifications dans un dépôt Git

Créer un fichier texte dans le répertoire ~/git-intro que vous nommerez SIO.txt avec le contenu suivant : « Grâce à mon travail, je suis en bonne voie pour réussir mes examens du BTS SIO. »

Vérifier le contenu du fichier en illustrant le résultat à l’aide d’une capture écran

Afficher l’état du dépôt. Réaliser une capture écran du message. (git status) Que pouvez-vous en déduire ?

Activer le suivi des modifications du fichier SIO.txt (git add).

Afficher à nouveau l’état du dépôt. Réaliser une capture écran du message. Quels sont les changements ?

Pour valider les changements utiliser la commande git avec commit en utilisant l’option -m qui vous permet de l’accompagner d’un message (« Commit de SIO.txt pour commencer à suivre les modifications »)

Une combinaison de 7 caractères est présente dans le résultat affiché. C’est l'ID de validation. Chaque commit est identifié par un hachage SHA1 unique. L'ID de validation est composé des 7 premiers caractères du hachage de validation complet. Utiliser la commande git log. Réaliser une capture d’écran. Peut-on retrouver l’ID de validation ?

Effectuez une modification à fichier SIO.txt en ajoutant en fin de fichier, « Je commence à comprendre Git. » (Indice : Il est possible d’utiliser echo à l’aide de >>). Réaliser une capture d’écran montrant la modification.

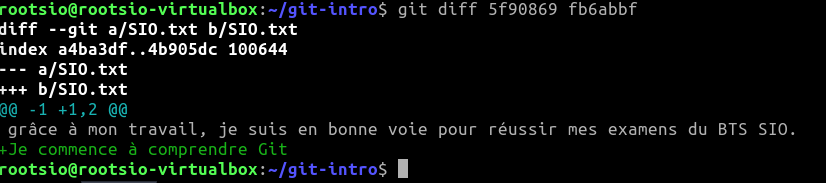
Afficher l’état du dépôt. Réaliser une capture écran du message. Que pouvez-vous en déduire ?

Réactiver le suivi des modifications du fichier SIO.txt. Cela sera nécessaire avant la prochaine validation (commit).

Utiliser le message « ajout d’une ligne supplémentaire au fichier » lors de la validation.

Réaliser une capture d’écran de la commande git log. Quel est son intérêt ? Quel est le commit référencé en en-tête ? Combien de commits ont été réalisés ? Quel est l’intérêt des messages de commit ?

Utiliser la commande git diff pour comparer les deux fichiers. Réaliser une capture d’écran. Que peut-on constater ?



Etape 4 : Branches et fusion

Lorsqu'un dépôt (référentiel) est créé, les fichiers sont automatiquement placés dans une branche appelée « master ». Une branche est une « copie » du projet et plus précisément un pointeur vers un commit représentant un instantané du projet. Dans la mesure du possible, il est recommandé d'utiliser des branches plutôt que de mettre directement à jour la branche « master ». La création d’une nouvelle branche du projet permet d’apporter des modifications dans une autre zone sans affecter la branche principale. Ceci est fait pour aider à prévenir les mises à jour accidentelles qui pourraient remplacer le code existant.

Les branches sont souvent utilisées lors de l'implémentation de nouvelles fonctionnalités ou correctifs. Ils peuvent ainsi être proposés pour examen par les membres de l'équipe puis, une fois vérifiés, être ensuite intégrés dans la branche principale « master ».

Créer une nouvelle branche appelée nouvelleBranche à l'aide de la commande git branch.

Utiliser la commande git branch sans nom en argument pour afficher toutes les branches. Réaliser une capture d’écran, que peut-on constater ?

Basculer vers la nouvelle branche à l’aide de git checkout. Quelle commande peut vous permettre de vérifier que la bascule a bien eu lieu, et que l’on se situe sur la bonne branche de travail, active ou courante ?

Ajouter la nouvelle ligne de texte qui suit au fichier SIO.txt, « Ce texte a été ajouté dans la branche nouvelleBranch ». Réaliser une capture d’écran montrant la modification.

Activer le suivi des modifications dans la branche active (nouvelleBranche)

Afficher l’état du dépôt. (git status) Réaliser une capture écran du message. Que pouvez-vous en déduire ?

Utiliser le message « Ajouter une troisième ligne au fichier dans nouvelleBranche fichier » lors de la validation. (git commit -m)

Afficher toutes les validations et réaliser une capture d’écran. Que peut constater ? Peut-on différencier les branches ?

Repasser sur la branche master et vérifier qu’elle est bien la branche active.

Fusionner le contenu du fichier de « nouvelleBranche » vers la branche « master ». (git merge) Réaliser une capture d’écran. Quelles indications sont fournies dans le résultat de la commande ?

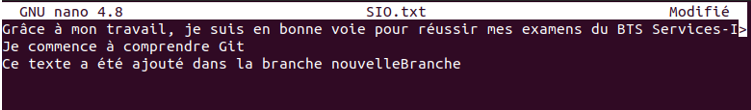
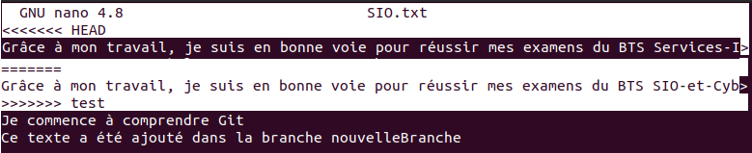
Vérifier que le contenu a bien été ajouté au fichier SIO.txt dans la branche « master ». Réaliser une capture d'écran.

Vérifier que la branche « nouvelleBranche » est toujours disponible ». Réaliser une capture d'écran.

Supprimer la nouvelle branche (git branch -d). Vérifier sa suppression. Réaliser une capture d’écran.

Parfois, vous pouvez rencontrer un conflit lors d’une fusion de branches. En règle générale cela est dû à des modifications multiples sur une même partie d’un fichier, Git ne peut donc plus faire de choix évident pour fusionner automatiquement les modifications. Dans cette partie, vous allez créer une branche de test créer un conflit en modifiant différemment une même partie du fichier dans les deux branches et voir comment localiser et résoudre les conflits pour fusionner les modifications apportées.

1. Créer une nouvelle branche nommée « test » puis basculer vers elle.
2. Vérifier qu’elle est bien devenue la branche active.
3. Utiliser la commande git branch sans nom en argument pour afficher toutes les branches. Réaliser une capture d’écran, que peut-on constater ?
4. A l’aide d’une commande remplacer SIO par SIO-et-Cybersecurite dans le fichier SIO.txt. Réaliser une capture d’écran montrant que la modification est effective.
5. Effectuer une nouvelle validation en utilisant la commande ci-dessous :  
    ~$git commit -a -m « Modif de SIO à SIO-et-Cyb…. »
6. A l’aide d’une recherche sur Internet, expliquer l’utilité de l’option -a.
7. Basculer vers la branche master. Puis vérifier la bascule.
8. A l’aide d’une commande remplacer SIO par Services-Informatiques-aux-Organisations dans le fichier SIO.txt. Réaliser une capture d’écran montrant que la modification est effective
9. Effectuer la validation en utilisant la commande ci-dessous :  
    ~$git commit -a -m « Modif de SIO à Services-Inf…. »
10. Fusionner les deux branches « test » et « master ». Effectuer une capture d’écran. Expliquer pourquoi la fusion des deux branches n’a pas abouti.
11. Afficher l’historique des validations. Peut-on vérifier que toutes les tâches ont été réalisées ?
12. Utiliser la commande cat pour afficher le contenu du fichier SIO.txt. Le fichier contient maintenant des informations pour vous aider à trouver le conflit. Elle vous indique la ligne où la version HEAD (branche master) contenant le mot " Services-Informatiques-aux-Organisations" est en conflit avec la version de la branche « test » contenant le mot " SIO-et-Cybersecurite".
13. Modifier manuellement le fichier SIO.txt pour supprimer le texte en conflit.



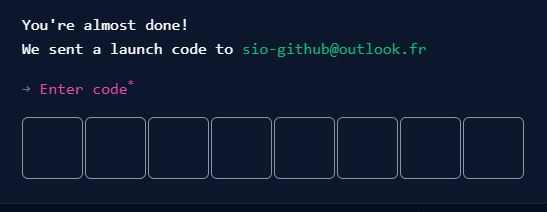
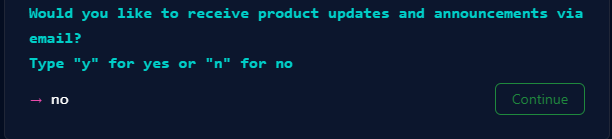
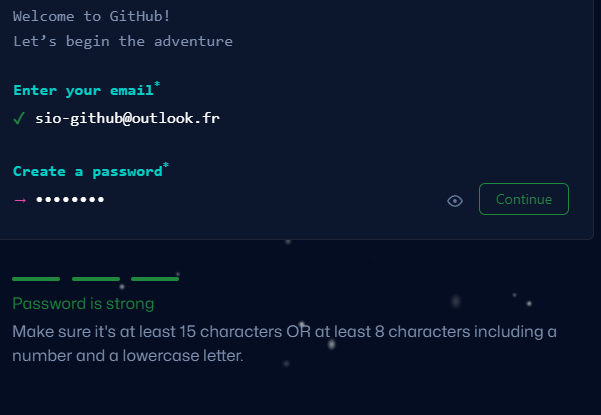
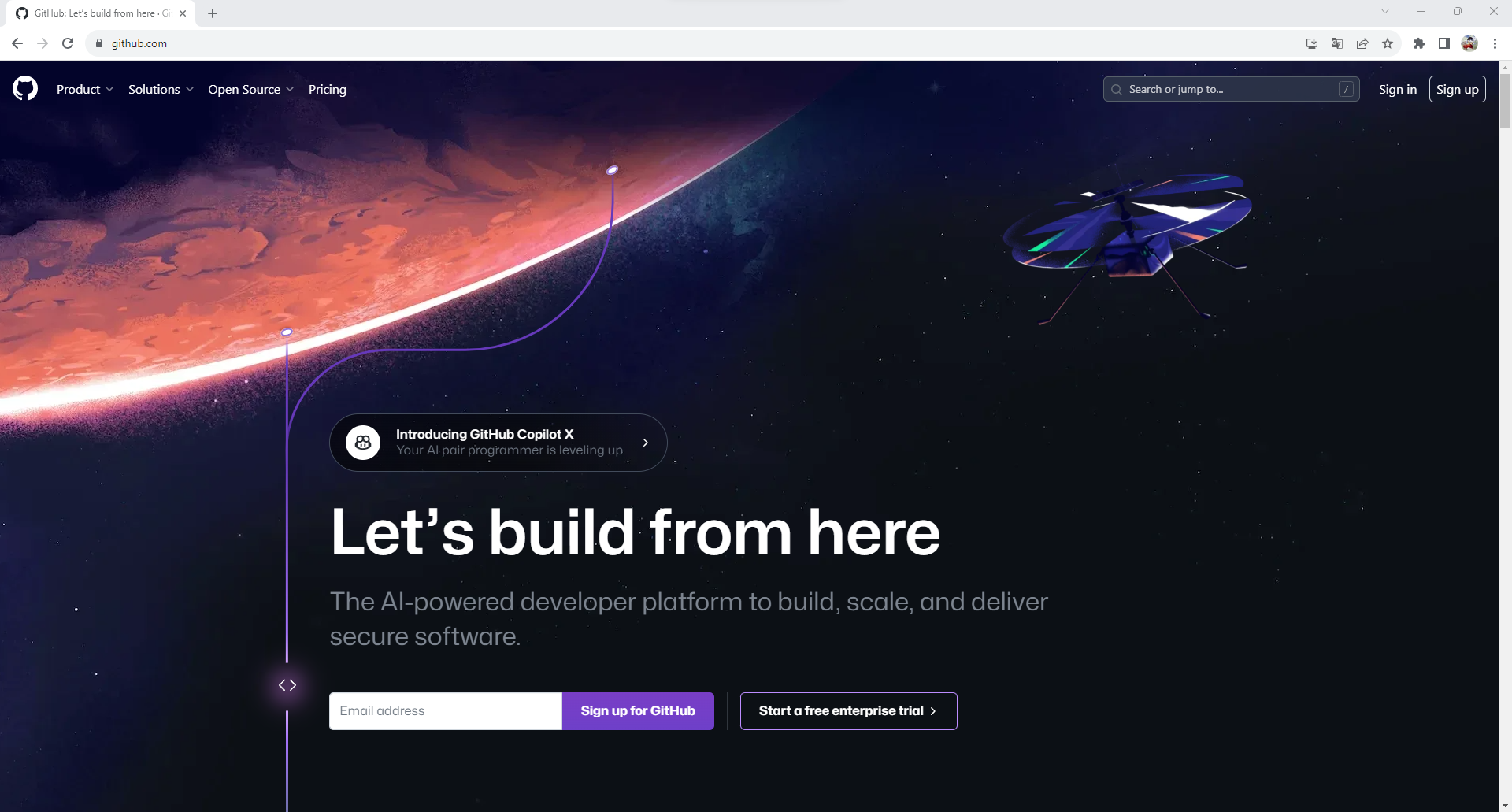
1. Enregistrer les modifications
2. Vérifier que vous avez bien modifié à l'aide de la commande cat.
3. Activer le suivi et validez les modifications de SIO.txt. (git add SIO.txt/git commit -a -m « fusion manuelle depuis la branche test »)
4. Utiliser la commande git log pour vérifier la validation. Si nécessaire, vous pouvez utiliser q pour quitter l'affichage git log.

Etape 4 : Intégration de Git avec GitHub

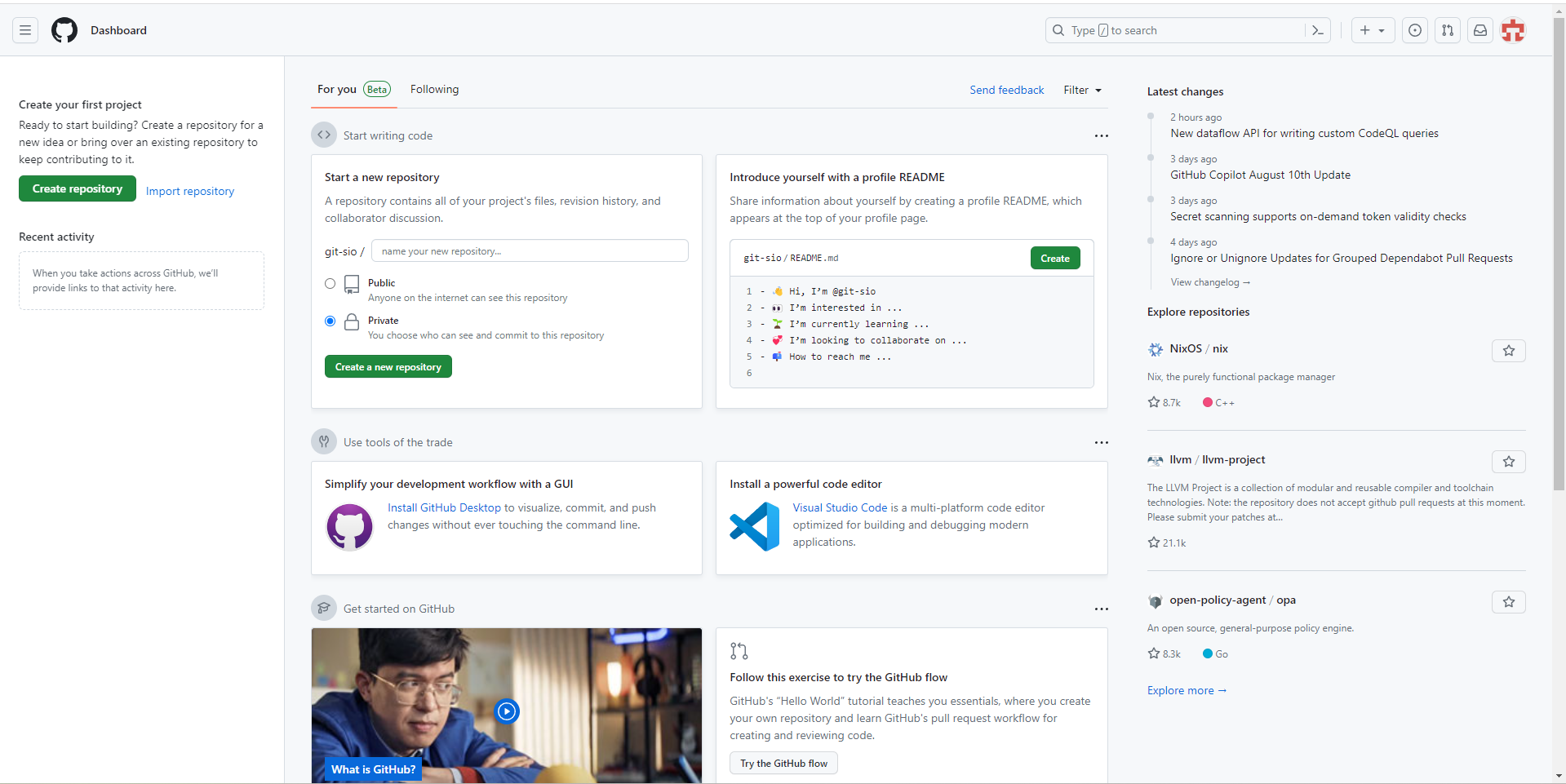
Jusqu'à présent, toutes les modifications apportées au fichier SIO.txt ont été stockées sur votre machine locale. Git s'exécute localement et ne nécessite aucun serveur de fichiers central ou service d'hébergement basé sur le cloud. Git permet à un utilisateur de stocker et de gérer localement des fichiers. Bien que Git soit utile pour un seul utilisateur, l'intégration d’un dépôt Git local avec un serveur hébergé dans le cloud comme GitHub est utile lorsqu’on travaille en mode collaboratif sur un projet. Chaque membre de l'équipe projet conserve une copie du repository (référentiel) sur sa machine locale et met à jour le repository (référentiel) commun distant hébergé en ligne pour partager les modifications éventuelles.

Il existe de nombreux services de gestion de dépôt de codes sources distribués basés sur GIT : GitHub mais aussi Stash d'Atlassian, GitLab, etc.. Ici nous utiliserons GitHub.

1. Si nécessaire, se rendre sur github.com et créez un compte GitHub. (votre adresse mail sera différente et doit être une adresse valide)



Et enfin arriver à votre tableau de bord GitHub



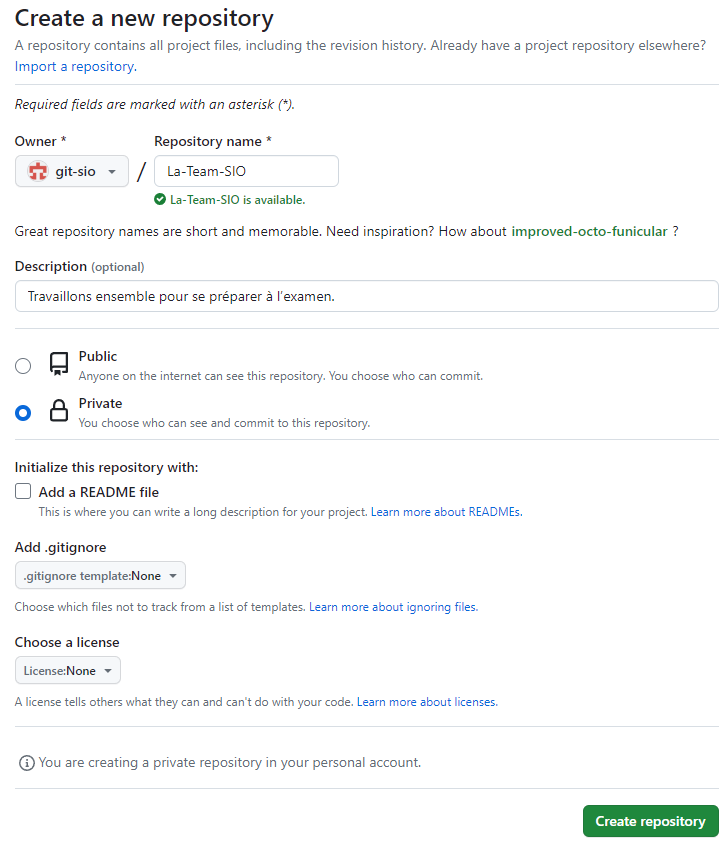
1. Créer un dépôt (repo). Sélectionner le bouton «Create repository » ou « New repository » s’il en existe déjà d’autres. Vous pouvez aussi cliquer sur l'icône "+" dans le coin supérieur droit et sélectionnez "New repository".
2. Créer un référentiel à l'aide des informations suivantes :

Nom du référentiel : La-Team-SIO

Description : Travaillons ensemble pour se préparer à l’examen.

Public/Private : Private

1. Cliquer sur « create repository »



1. Créer un nouveau répertoire La-Team-SIO dans le répertoire git-intro. Indiquer la commande vous permettant de le faire.
2. Se déplacer dans ce nouveau répertoire.
3. Copier le fichier SIO.txt situé dans le répertoire parent dans ce répertoire.
4. Afficher le contenu du répertoire et vérifier qu’il contient le fichier SIO.txt
5. Lancer la commande git permettant d’initialiser le répertoire courant.
6. S’assurer que vos variables git globales correspondent à votre compte GitHub.

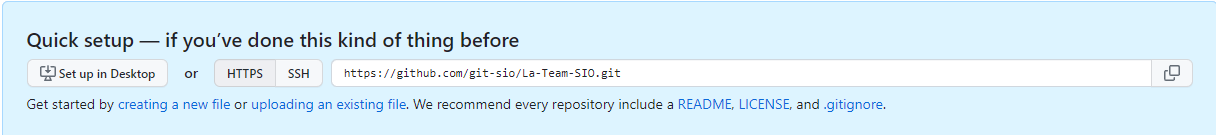


Les tâches suivantes vont permettre la synchronisation du dépôt local et du dépôt github distant

1. Envoi vers le dépôt GitHub. Utiliser la commande git remote add pour ajouter une URL Git comme alias distant.

git remote add origine https://github.com/git-sio/La-Team-SIO.git

La valeur "origine" pointe vers le dépôt nouvellement créé sur GitHub. L’url peut être copiée sur GitHub afin d’éviter des erreurs de saisie.



1. Vérifier que cela a bien été pris en compte

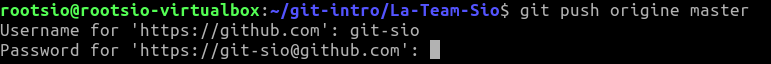
git remote –verbose



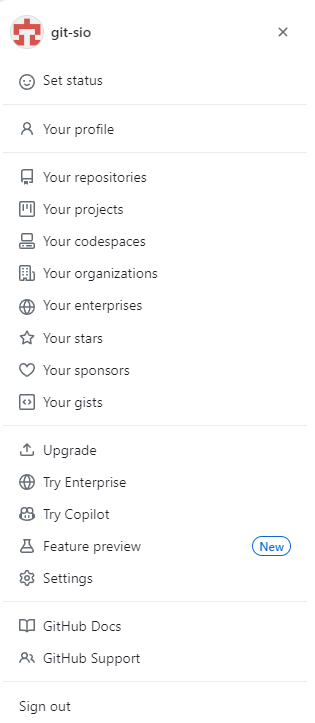
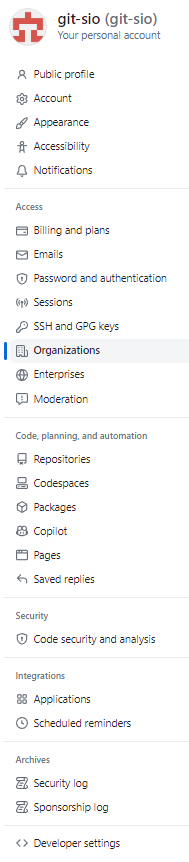
1. Afficher l’historique des modifications effectuées.



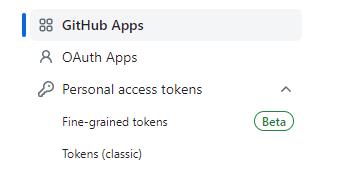
1. Activer le suivi des modifications du fichier SIO.txt.
2. Utiliser le message « ajout de SIO.txt, fichier du répertoire La-Team-Sio» lors de la validation.
3. Afficher l’historique pour vérifier la validation.
4. Afficher l’état du dépôt. Réaliser une capture écran du message. Que pouvez-vous en déduire ?
5. Utilisez la commande git push origin master pour envoyer (pousser) le fichier vers votre dépôt GitHub. Vous serez invité à entrer un nom d'utilisateur et un mot de passe.



Le mot de passe attendu est en fait un jeton sécurisé généré par Github. Pour le générer, vous devez suivre les étapes suivantes :



* Pour le jeton, rendez-vous sur   
    
  GitHub.com, sur votre avatar en haut à  
    
  droite et cliquez sur setting.
* Cliquer ensuite sur Developer settings

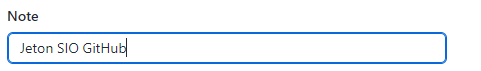


* Cliquer ensuite sur Personnal acces token et token (classic)

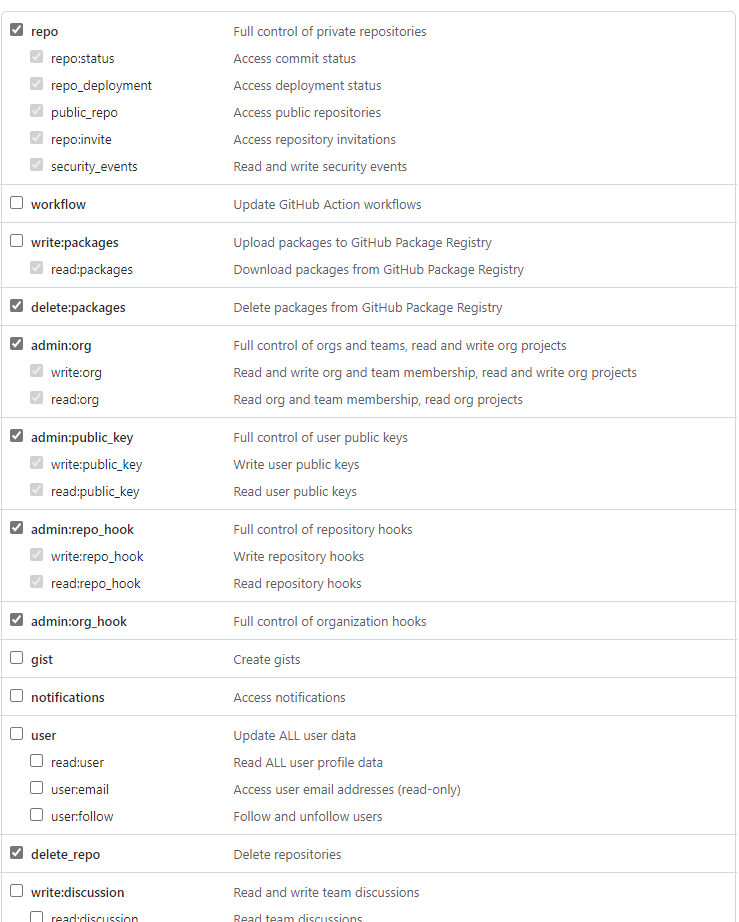
* Generate New Token



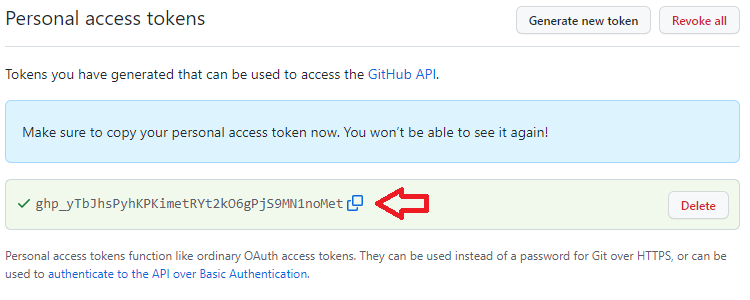
* Remplir la case Note



* Sélectionner les cases ci-dessous :



* Enfin cliquer sur
* Copier le jeton avant d’aller le coller sur le Terminal



Copier le jeton et le coller en lieu et place du mot de passe demandé.

Rappel de la commande :



* Vérifier le fichier sur GitHub. (Pour revenir à la racine du site il faut cliquer sur l’icône en forme de chat tout en haut à gauche)
* Vous devriez voir que le fichier SIO.txt a été ajouté à ce dépôt GitHub. Cliquer sur le fichier pour afficher le contenu.

