

GIT et GITHUB



2021

1

Sommaire

- **Chapitre 1 :** Les objectifs d'un gestionnaire de sources
- **Chapitre 2 :** Les fonctionnalités
- **Chapitre 3 :** Les différents gestionnaires de sources
- **Chapitre 4 :** Les problématiques d'intégration des changements
- **Labs**

2

CHAPITRE: LES OBJECTIFS D'UN GESTIONNAIRE DE SOURCES

3

Les objectifs de la gestion de sources

- Gérer les activités de lecture, écriture et fusion sur les sources du projet.
- Conserver un historique des modifications apportées aux sources:
 - Par qui, quand, quoi?
- Permettre de revenir en arrière en cas de besoin.
- Permettre de travailler en équipe sur les mêmes sources d'un projet.
- Gérer les accès concurrents.
- Gérer les conflits.
- Permettre de travailler simultanément sur plusieurs versions d'un logiciel

4

Les objectifs de la gestion de sources

- Permet de retrouver un fichier source supprimé
- Fonctionne avec tout type de fichier txt, php, java, jpg, exe,
- Garantir la sécurité de la configuration du logiciel
 - Autorisation
 - Confidentialité
 - Intégrité
 - Disponibilité

5

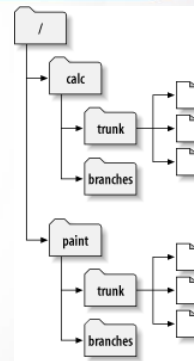
CHAPITRE: LES FONCTIONNALITÉS D'UN GESTIONNAIRE DE SOURCES

6

Les fonctionnalités d'un gestionnaire de sources

Repository

- Un dépôt (local ou distant) qui contient toutes les versions des sources du projet
- Les données des changements apportés à la configuration du logiciel
- Le dépôt a une structure arborescente

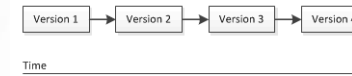


7

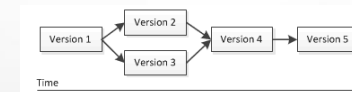
Les fonctionnalités d'un gestionnaire de sources

Version

- Une version ou révision est une instance d'un fichier source qui est strictement distincte des autres instances
- Les versions peuvent se succéder de manière linéaire dans le temps



- En cas de modifications concurrentes d'une version des splits et des merges peuvent avoir lieu

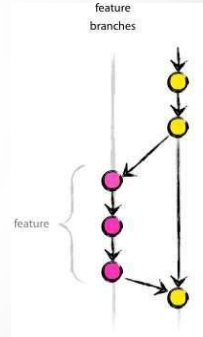


8

Les fonctionnalités d'un gestionnaire de sources

Branche

- Permet de développer en parallèle plusieurs versions du logiciel
- Permet de corriger un problème sur une ancienne version du logiciel
- Permet de fusionner après une divergence

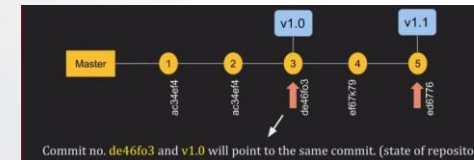
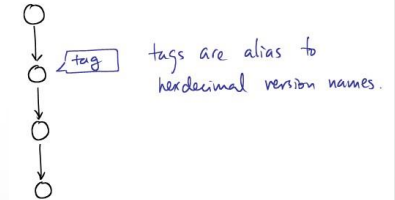


9

Les fonctionnalités d'un gestionnaire de sources

Tag

- Donner un nom explicite à une version du logiciel pour pouvoir y accéder facilement
- Permet de définir les versions du projet
- Permet de nommer des branches



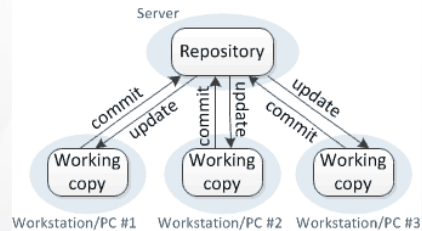
10

Les différents gestionnaires de sources

Modèle centralisé

Centralized Version Control System CVCS

Centralized version control



11

Les différents gestionnaires de sources

Modèle centralisé

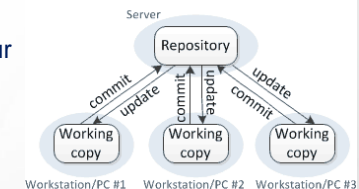
Le dépôt est stocké dans un endroit partagé

Chaque développeur a sa copie avec des branches et des tags privés

Utilise le mode de verrouillage pour gérer les développements concurrents sur un même fichier source

Il est nécessaire d'avoir la connexion au dépôt pour *commiter* ses modifications ou mettre à jour sa copie de travail

Centralized version control



12

Les différents gestionnaires de sources

Modèle centralisé

Avantages

- Technologie éprouvée
- Structure simple
- Gestion et utilisation simples à mettre en oeuvre
- Largement disponible (Forges)

Faiblesses

- Gestion et utilisation plus compliquées
- Risque de divergence, peut devenir très complexe structurellement
- Très sensible aux pannes, impossible de travailler hors connexion
- Inadapté aux très grands projets, le temps de mise à jour du dépôt est trop long

13

Les différents gestionnaires de sources

Modèle centralisé – exemple Subversion

- Créé en 2000 pour remplacer CVS
- Logiciel libre
- Multi plateforme (Windows, MacOS,..)
- Documentations très riches, forums actifs
- Implémente des protocoles réseaux sécurisés
- Interfaces graphiques
- Intégré dans plusieurs IDE tel que Eclipse

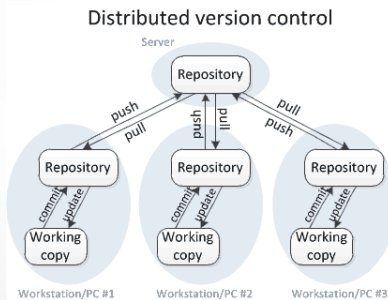


14

Les différents gestionnaires de sources

Modèle distribué

- Distributed Version Control System – DVCS

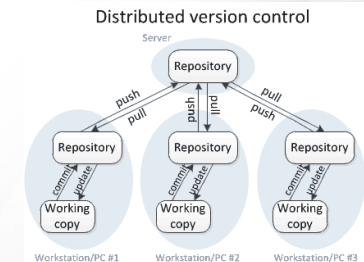


15

Les différents gestionnaires de sources

Modèle distribué

- Chaque développeur a sa copie avec des branches et des tags privés
- Utilise le mode de fusion pour gérer les développements concurrents sur un même fichier source



16

Les différents gestionnaires de sources

Modèle distribué

Avantages

- Travail déconnecté, accès aux dépôts en local sur le poste de travail du développeur
- Moins sensible aux pannes
- Rapidité, le réseau n'est utilisé que pour les opérations de synchronisation
- Branches privées
- Modèle de développement autorisé très souple

Faiblesses

- Gestion et utilisation plus compliquées
- Risque de divergence, peut devenir très complexe structurellement

17

Les différents gestionnaires de sources

Modèle distribué – exemple GIT

- Créé en 2005 par Linus Torvalds pour la gestion des sources du noyau Linux
- Logiciel libre Multi plateforme (Linux, Windows, MacOS,
- Documentations très riches, forums actifs
- Implémente des protocoles réseaux sécurisés (HTTPS)
- Interfaces graphiques
- Intégré dans plusieurs IDE tel que Eclipse
- Ne stocke que le delta entre deux versions d'un objet



18

Problématiques de fusion des changements

- Plus la quantité de code publiée est grosse plus le risque de conflit augmente
- La publication régulière du code source
 - Réduit le risque de conflit
 - Facilite l'identification de la modification à l'origine du problème
- Pour pouvoir publier régulièrement le code source il faut:
 - Publier les modifications apportées à la fin de chaque tâche de développement
 - Faire des petits changements et ne pas apporter des changements à plusieurs composants dans une seule tâche

19

Problématiques de fusion des changements

Bonnes pratiques

- Essayez de travailler le plus possible avec des fichiers texte et le moins possible avec des fichiers binaires
- Faites des updates le plus souvent et régulièrement possible, et obligatoirement avant de commencer à travailler sur un fichier
- Faites des *commit* réguliers, dès que vous avez fini de travailler sur un fichier, et si la tâche prend du temps, un *commit* intermédiaire permettra de sauvegarde votre travail
- Créez un planning pour savoir qui peut travailler et quand sur les fichiers binaires notamment

20

Quiz

- 1. La gestion de code sources est une pratique
 - a. Accessoire pour un projet informatique
 - b. Indispensable pour la réussite d'un projet de développement logiciel
- 2. L'utilisation des branches permet de
 - a. Développer en parallèle plusieurs versions du logiciel
 - b. Corriger un problème sur une ancienne version du logiciel
 - c. Localiser facilement les modification apportées à un ou plusieurs fichiers
- 3. Un dépôt local Git doit obligatoirement avoir la même structure que le dépôt central (branche)
 - a. Vrai
 - b. Faux

21

Quiz

- 4. La commande qui permet d'initialiser un dépôt Git est
 - a. Git add
 - b. Git init
 - c. Git commit
- 5. En utilisant Git le développeur a besoin de communiquer avec le serveur
 - a. Pour récupérer ou publier des modifications du dépôt central
 - b. Pour réserver des fichiers à modifier
 - c. Il n'est pas utile de communiquer avec le serveur

22

Quiz

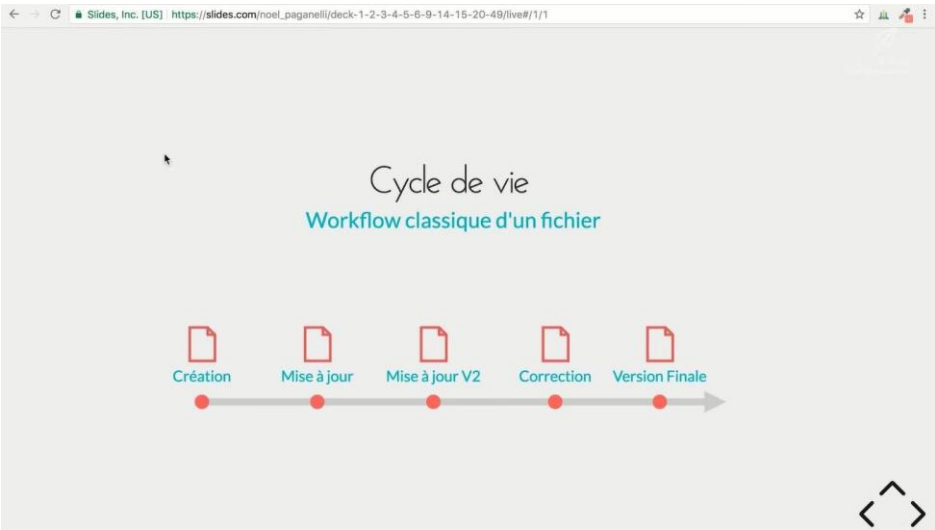
- 6. L'utilisation des tags est possible avec
 - a. Une gestion du type Centralized Version Control System CVCS
 - b. Une gestion du type Distributed Version Control System DVCS
 - c. N'est plus utilisée avec les système moderne tel que Git

23

Utilisation de Git - Lab



24



25

Un outil qui s'occupe de tout !

Installation: <http://git-scm.com/>

26

git --distributed-even-if-your-workflow-isnt

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.

Learn Git in your browser for free with **Try Git**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.14.2
Release Notes (2017-09-22)
Downloads for Mac

27

Un outil en ligne de commande
Ouvrir un terminal

cmd

Terminal

28

Check install
Vérifier la version de GIT

 `git --version`



29

1er Paramétrage
Définir l'identité de l'utilisateur



```
git config --global user.name "Noel Paganelli"  
git config --global user.email "noel@lacapsule.academy"
```



30


Initialisation
Initialise GIT dans le projet


```
git init
```



31

Etat des lieux
Voir à tout moment le statut des fichiers

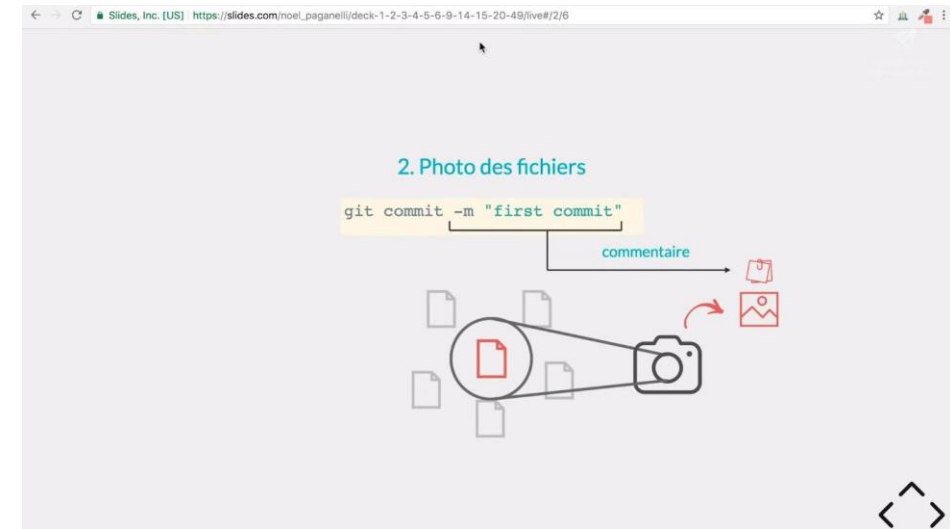
 `git status`



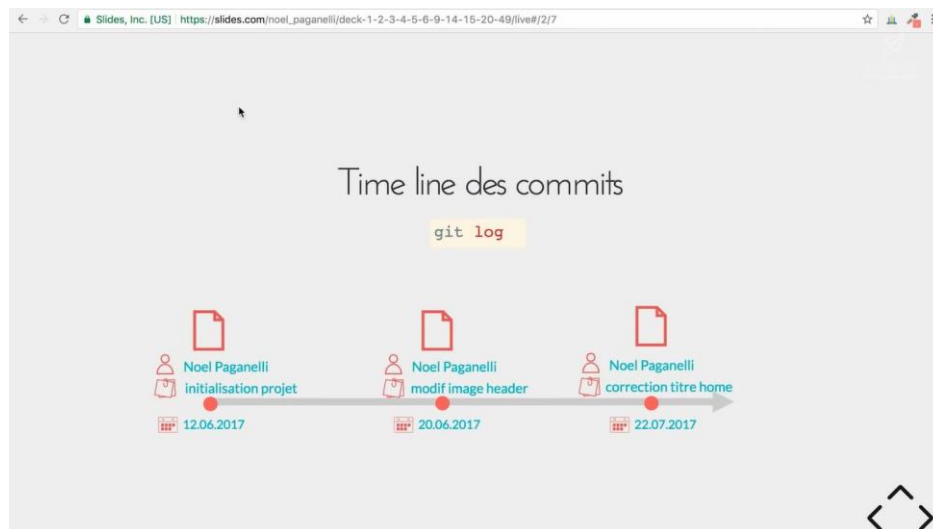
32



33



34



35



36

Slides, Inc. [US] https://slides.com/noel_paganelli/deck-1-2-3-4-5-6-9-14-15-20-49/live#/3/3

Lister les branches

```
git branch
```

37

Slides, Inc. [US] https://slides.com/noel_paganelli/deck-1-2-3-4-5-6-9-14-15-20-49/live#/3/5

Branches indépendantes

Le commit s'applique uniquement sur la branche active

38

Slides, Inc. [US] https://slides.com/noel_paganelli/deck-1-2-3-4-5-6-9-14-15-20-49/live#/3/7

1. Sélectionner la branche principale

```
git checkout master
```

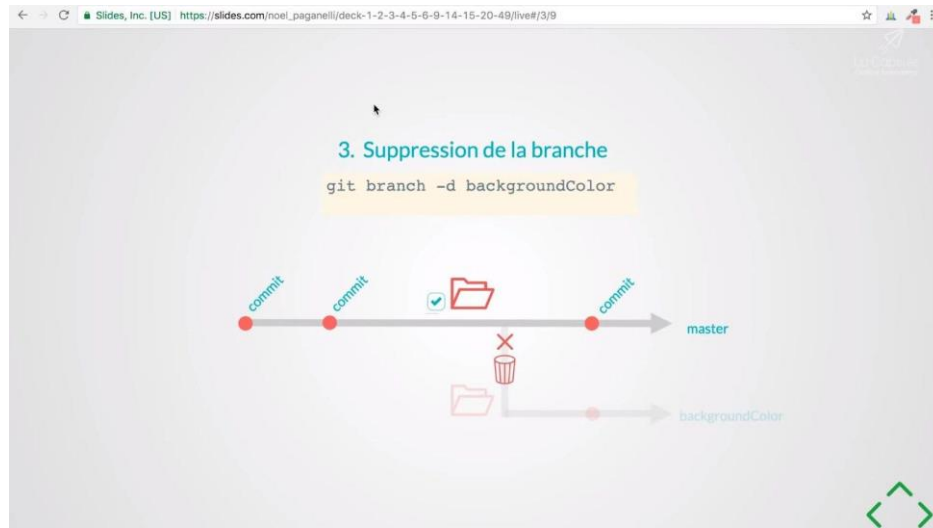
39

Slides, Inc. [US] https://slides.com/noel_paganelli/deck-1-2-3-4-5-6-9-14-15-20-49/live#/3/8

2. Fusionner les commits

```
git merge colorBackground
```

40



41

Ce qu'il manque jusque là

Capacité de sauvegarde distance : dois-je faire une copie dans ma dropbox ?

Capacité de travailler en équipe : dois-je faire un git dans ma dropbox et partager le dossier ?

Git et ses services en ligne

Des entreprises et des structures du public fournissent des serveurs centralisés pour git :

[Github](#) qui est le plus à la mode et que nous utiliserons

[Gitlab](#) qui est le concurrent le plus sérieux de github aujourd'hui

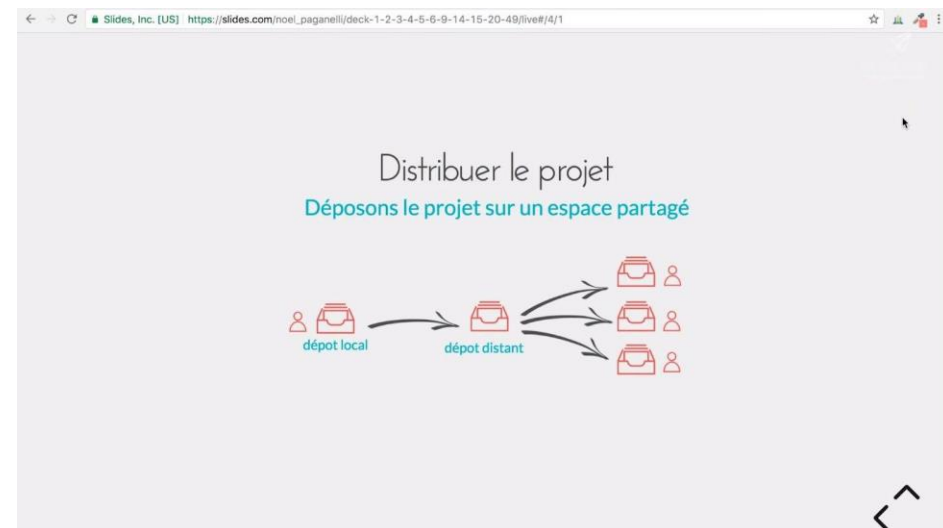
[Bitbucket](#) Créé par Atlassian

[SourceForge](#) ...

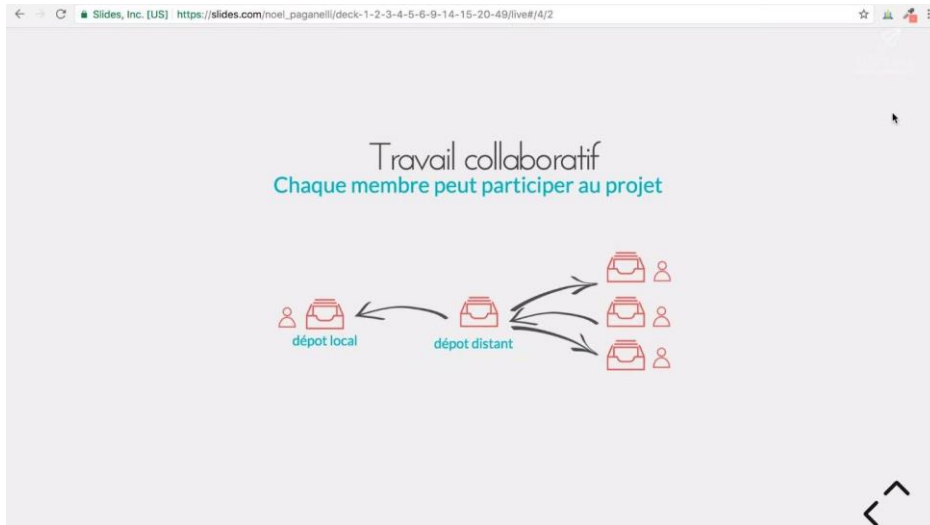
42



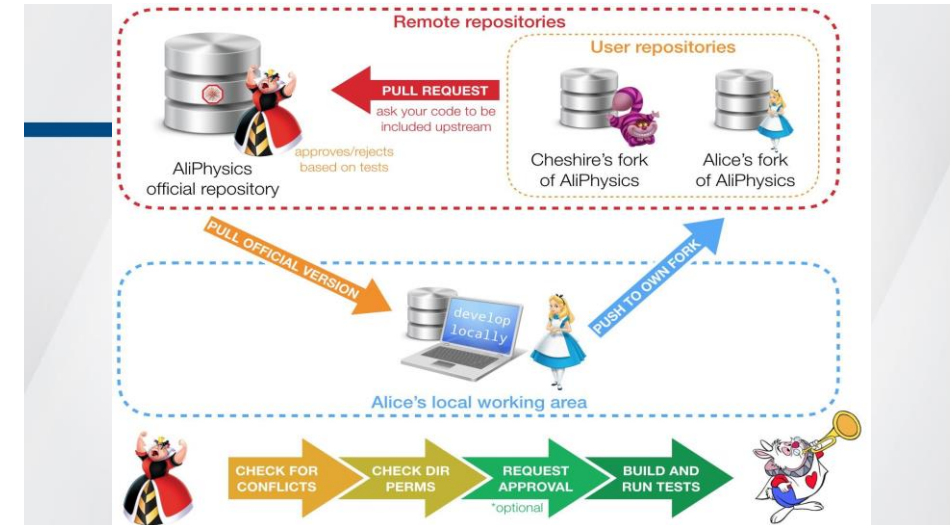
43



44



45



46

Le vocabulaire des serveurs distants pour Git

- **Remote** : Serveur distant permettant la synchronisation manuelle de notre dépôt
- **origin** : Nom du serveur principal, un peu comme master est la branche principale. Pour l'instant, nos serveurs n'en ont pas
- **push** : Envoyez les modifications effectuées sur un serveur donné.
- **pull** : Retrouvez les informations depuis un serveur donné
- **clone** : Copie sur votre PC d'un dépôt trouvé en ligne
- **fork** : dérivé d'un dépôt d'un-e autre développeur-se
- **upstream** : Par convention, nom d'un second serveur, généralement le serveur source du fork

47

Créons un compte Github

Github a des défauts (pas open source par exemple) mais reste l'outil le plus moderne pour de la gestion de code versionné en équipe

Github propose un environnement complet

- Gestion d'équipe
- Connexions à d'autres applications
- Gestion de ticket
- Gestion de fusion de branches
- etc.



48

Création du dépôt distant Github

➤ Créer un repo « projet-abc » (ne pas cocher initialiser le repository)

➤ Ajouter ce serveur à votre dépôt local

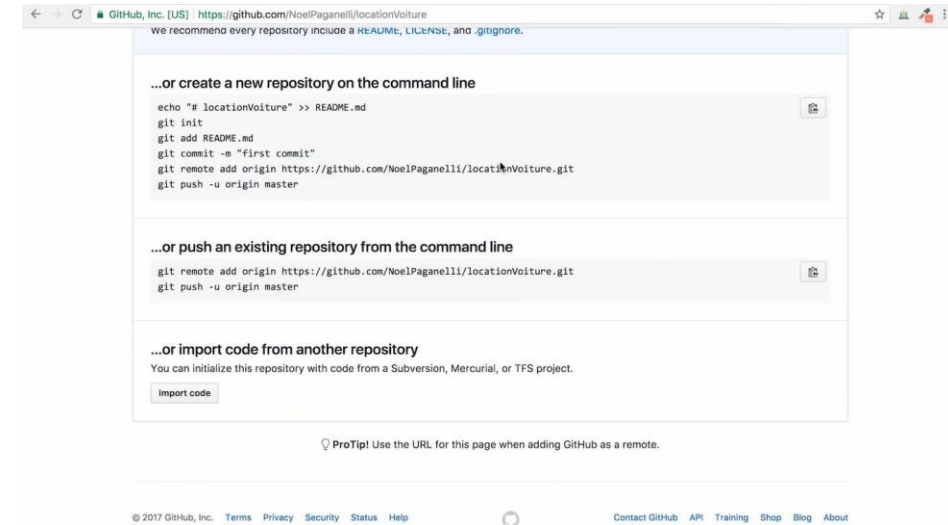
`git remote add origin [adresse du serveur]`

➤ Pour le premier push de synchronisation

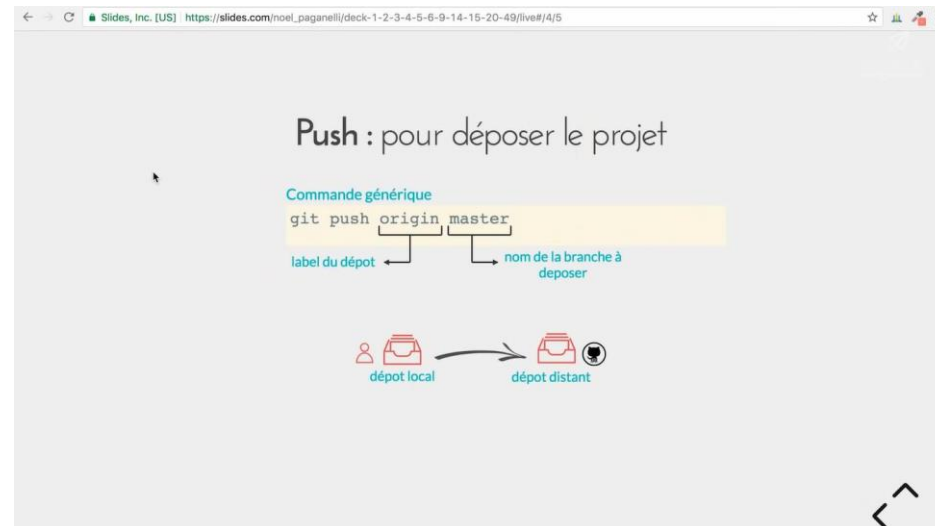
`git push -u origin master`

Git, envoie ma branche actuelle sur la branche master du serveur origin. (-u:) Cette branche correspondra à partir de maintenant à celle sur mon dépôt local

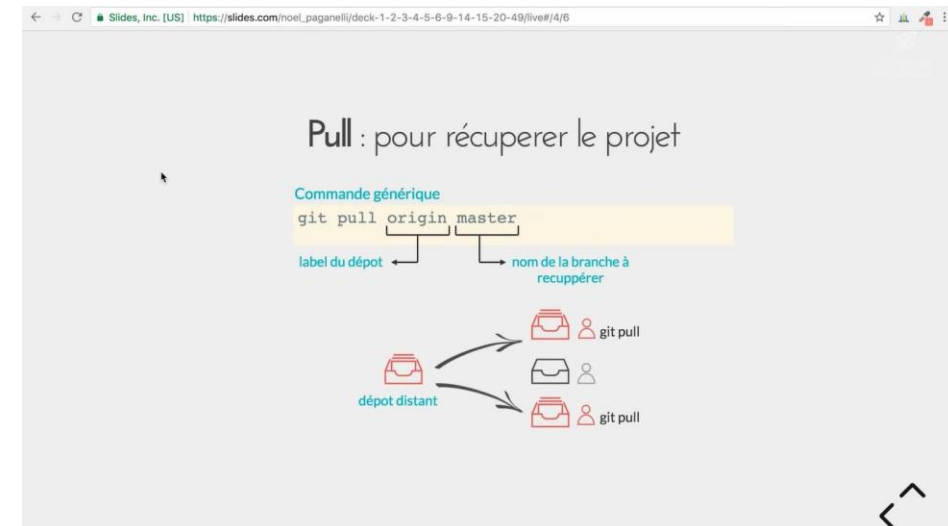
49



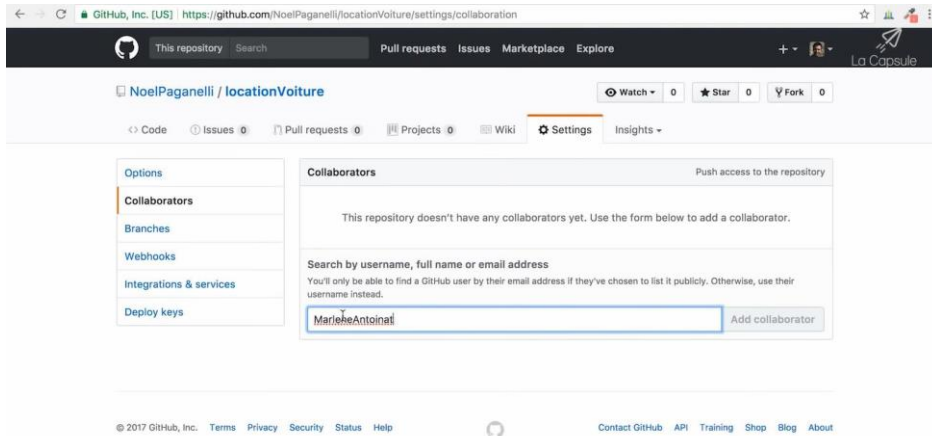
50



51



52



53

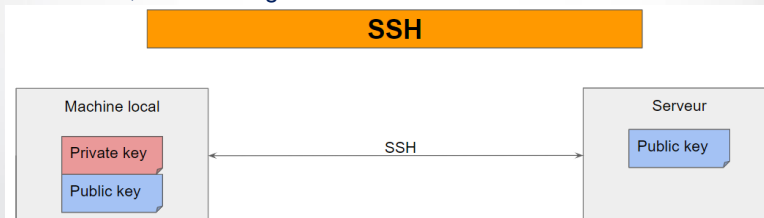
ACCÉDER EN SSH À VOS DÉPÔTS GIT DISTANTS



54

Fonctionnement du protocole SSH git

- Secure Shell (SSH) est à la fois un programme informatique et un protocole de communication sécurisé.
- Le protocole de connexion impose un échange de clés de chiffrement en début de connexion.
- Par la suite, tous les segments TCP sont authentifiés et chiffrés.



55

Génération des clés SSH git

- Il faut alors générer une paire de clé SSH via la commande:

./ssh-keygen

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/codeur-pro/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/codeur-pro/.ssh/id_rsa.
Your public key has been saved in /Users/codeur-pro/.ssh/id_rsa.pub.
```

A ce moment là, deux clés vont être générées dans votre dossier 'home' par défaut:

- Linux: /home/[user]/.ssh
- Windows: C:\Utilisateurs\[user]\.ssh

Attention: Le dossier .ssh est un dossier caché

56

Génération des clés SSH



On retrouve alors dans le dossier .ssh les fichiers suivants:

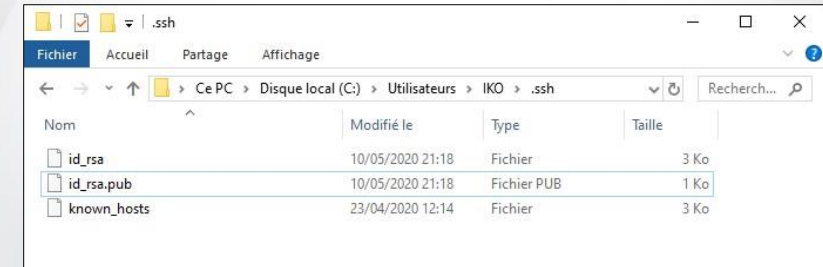
- **id_rsa**: clé **privé** à conserver sur son PC et à ne surtout pas partager
- **id_rsa.pub**: clé **publique** à envoyer sur les machines avec lesquels vous voulez communiquer en SSH
- **Info**: Vous pouvez ne pas indiquer de mot de passe et ainsi avoir une communication SSH qui sera tout de même sécurisée.
- L'intérêt ici, c'est que vous n'aurez pas à écrire ce mot de passe lors d'un git clone ou git pull/push.

57

Mise en place du SSH sur GitHub



- Pour commencer, il faut copier le contenu de la clé SSH publique. Pour cela, il faut se rendre dans le dossier où ssh-keygen a généré les clés. Puis, éditez le fichier **id_rsa.pub** et enfin copier l'intégralité de son contenu.

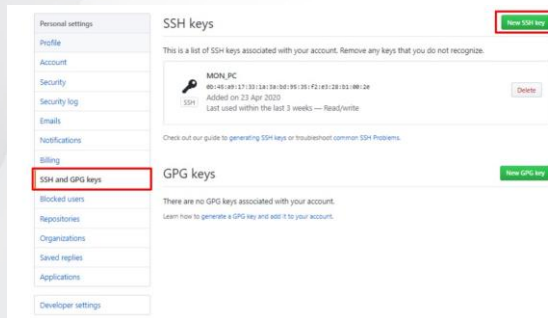


58

Mise en place du SSH sur GitHub



- Vous pouvez alors vous rendre dans **les settings de votre compte** sur le site de **GitHub**



Onget **SSH and GPG keys**: "new SSH key".

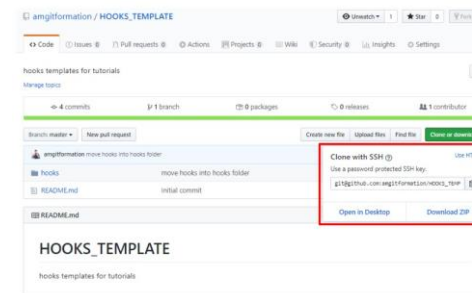
Il faut alors simplement coller le contenu du fichier **id_rsa.pub** que l'on a copié depuis notre PC, puis valider.

59

Cloner le dépôt en SSH



- Maintenant que votre communication SSH est en place, vous pouvez la tester en clonant un dépôt. Dans les options du bouton "clone or download" choisir "use SSH" puis copier l'URL du dépôt.



Il ne vous reste plus qu'à taper, dans le terminal, la commande:

git clone [url_ssh]

Vous pouvez également changer le protocole que vous utilisez sur un dépôt déjà cloné et ainsi passer de https à ssh et inversement:

git remote set-url origin [URL_SSH OU URL_HTTPS]

60