

# Démystifions Git, Github, Gitlab

Arnaud Legrand

Grenoble Alpes University and CNRS

`arnaud.legrand@imag.fr`

9 décembre 2019

# Outline

Contexte

L'historique quand on est tout seul

Travailler à plusieurs

L'écosystème Git

# Où en sommes nous ?

## Contexte

L'historique quand on est tout seul

Travailler à plusieurs

L'écosystème Git

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté!

Solution ?

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté!
- Ce document était bien mieux hier!

## Solution ?

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté!
- Ce document était bien mieux hier!
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner?

## Solution ?

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté !
- Ce document était bien mieux hier !
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner ?
- Mais qui a écrit ça ? Et pourquoi ?

## Solution ?

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté !
- Ce document était bien mieux hier !
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner ?
- Mais qui a écrit ça ? Et pourquoi ?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial.pptx »



# Aaah... l'informatique

## Problème

- Mon ordinateur a planté !
- Ce document était bien mieux hier !
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner ?
- Mais qui a écrit ça ? Et pourquoi ?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial\_FINAL.pptx »

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté !
- Ce document était bien mieux hier !
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner ?
- Mais qui a écrit ça ? Et pourquoi ?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial\_FINAL\_FINAL.pptx »

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté!
- Ce document était bien mieux hier!
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner?
- Mais qui a écrit ça? Et pourquoi?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial\_FINAL\_FINAL\_0925.pptx »

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté!
- Ce document était bien mieux hier!
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner?
- Mais qui a écrit ça? Et pourquoi?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial\_FINAL\_FINAL\_0925.pptx »
- Travailler tout seul!

# Aaah... l'informatique

## Problème

- Mon ordinateur a planté !
- Ce document était bien mieux hier !
- Vous n'avez pas travaillé sur la bonne version du document. Comment fusionner ?
- Mais qui a écrit ça ? Et pourquoi ?

## Solution ?

- Sauvegarde régulière avec une notion d'historique  
« SE15\_PPT\_16x9\_Template\_Arial\_FINAL\_FINAL\_0925.pptx »
- Travailler tout seul !
- Passer son temps à (s')envoyer des mails !

# Gestionnaire de versions

Les gestionnaires de version datent du début des années 70

## Fonctionnalités essentielles

- Conserve un historique des modifications sans dupliquer les dossiers de sauvegardes
- Sauvegarde les données (avec l'historique) sur des ordinateurs distants
- Fusion "automatique" des fichiers édités par plusieurs personnes



- 2005 : créé pour le développement du noyau Linux par Linus Torvalds
  - > 20 millions de lignes de code
  - $\approx$  14 000 développeurs



# Objectif de cet exposé

- Apprendre à utiliser `git`

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



# Objectif de cet exposé

- Apprendre à utiliser ~~git~~

git est "explicite" (à la différence de dropbox et autres)

Heureusement, il y a des interfaces

- Notions essentielles pour utiliser les interfaces à git
  - Historique distribué
  - Branche
  - Conflit et fusion
  - L'écosystème  git

**GitHub**  **GitLab** 

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.





# Où en sommes nous ?

Contexte

L'historique quand on est tout seul

Travailler à plusieurs

L'écosystème Git

# Son propre projet

Temps



Accès à un certain nombre de fichiers.

- Temps continu

# Son propre projet

Temps



Accès à un certain nombre de fichiers.

- Temps continu
- Checkpoint/snapshot : conserver les moments clés du projet pour pouvoir y revenir

# Son propre projet

Temps



Accès à un certain nombre de fichiers.

- Temps continu
- Checkpoint/snapshot : conserver les moments clés du projet pour pouvoir y revenir
  - `git add` pour indiquer quelles modifications on souhaite conserver

# Son propre projet

Temps



Accès à un certain nombre de fichiers.

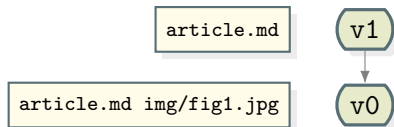
- Temps continu
- Checkpoint/snapshot : conserver les moments clés du projet pour pouvoir y revenir
  - `git add` pour indiquer quelles modifications on souhaite conserver
  - `git commit` pour faire le checkpoint (date, commentaire)

# Une histoire simple

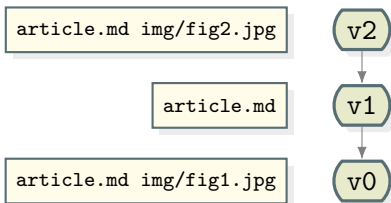
article.md img/fig1.jpg

v0

# Une histoire simple

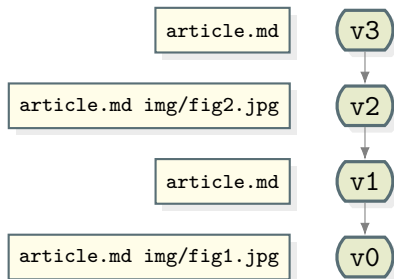


# Une histoire simple

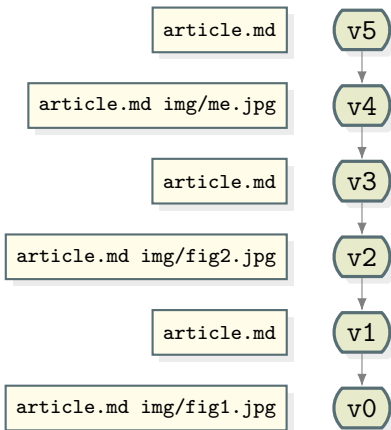




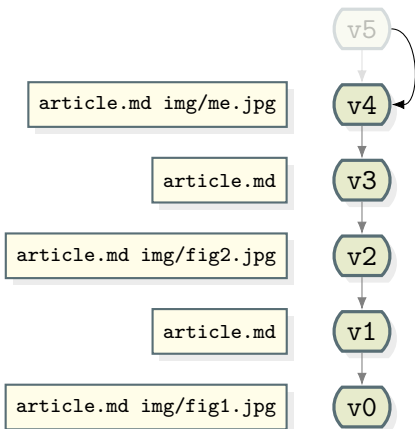
# Une histoire simple



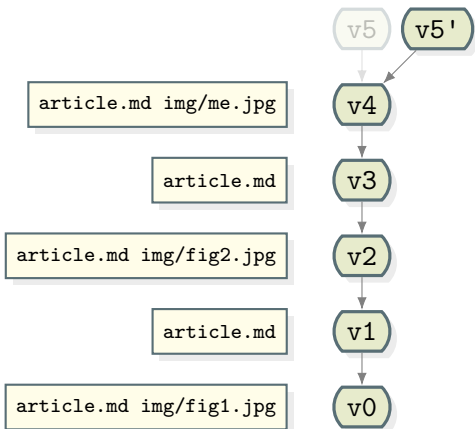
# Une histoire simple



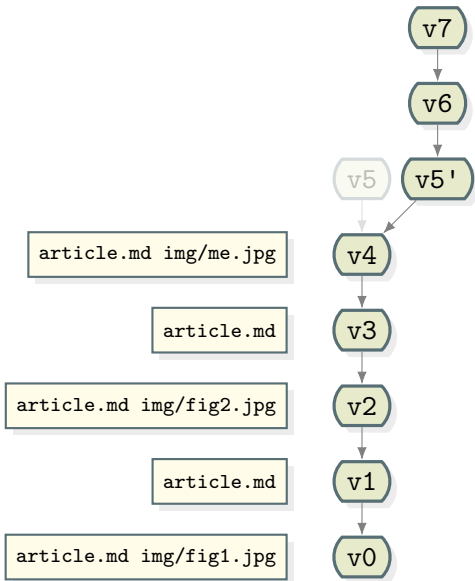
# Une histoire simple



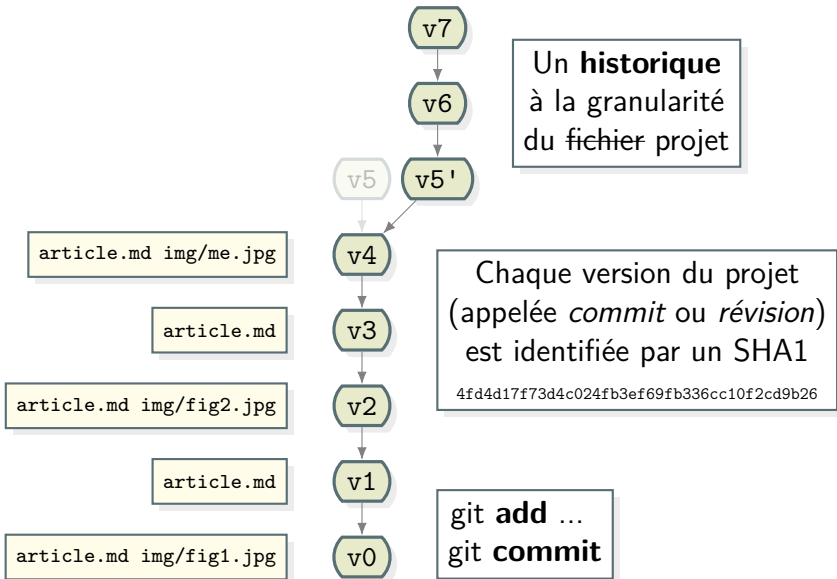
# Une histoire simple



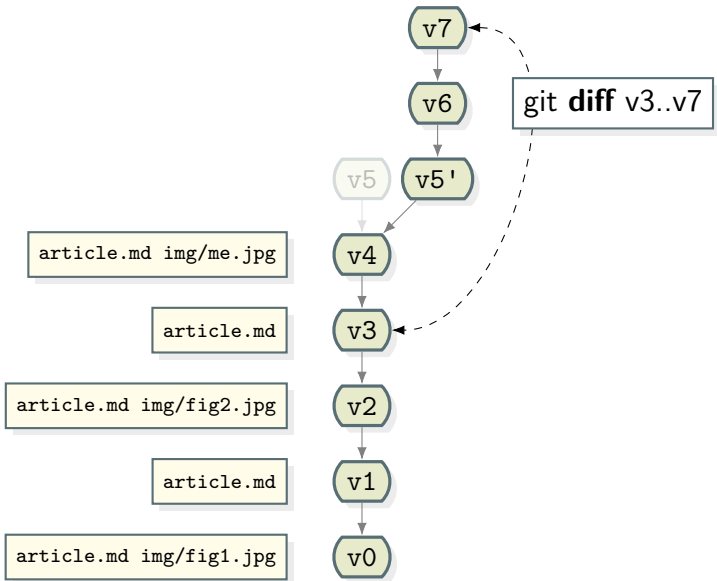
# Une histoire simple



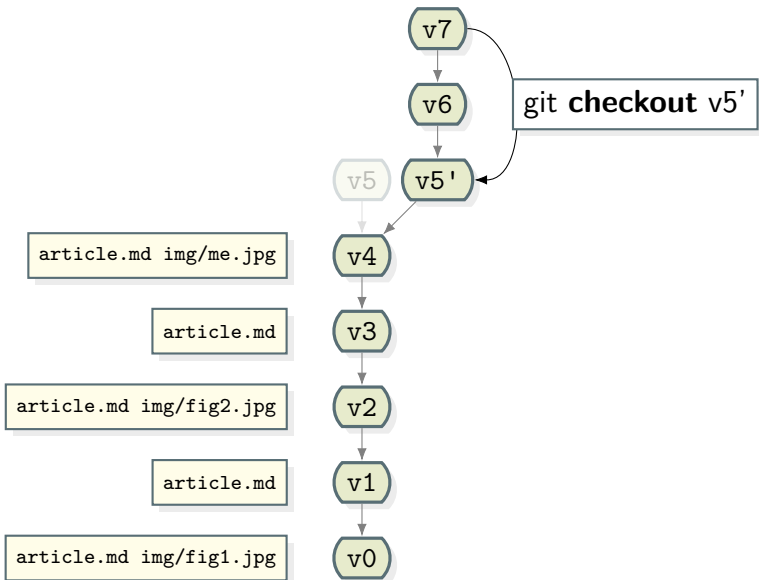
# Une histoire simple



# Une histoire simple

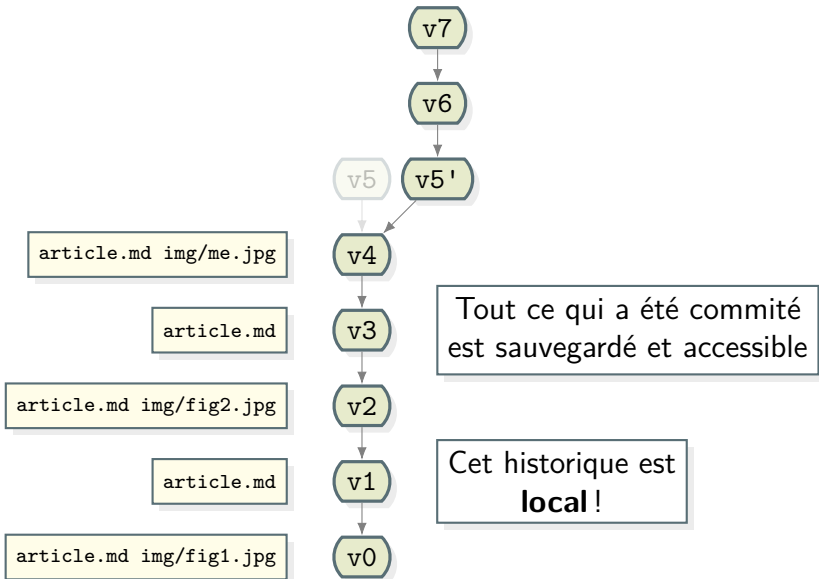


# Une histoire simple

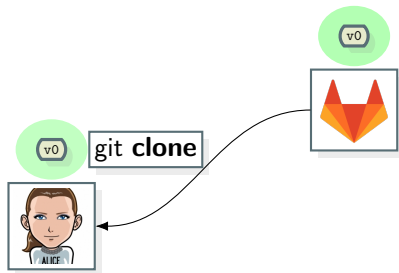




# Une histoire simple



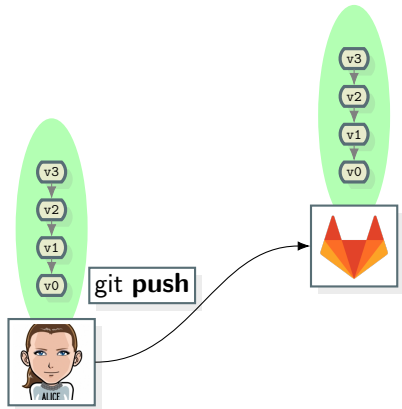
# Backup sur un server distant



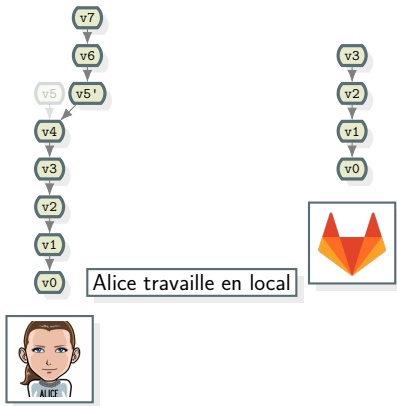
# Backup sur un server distant



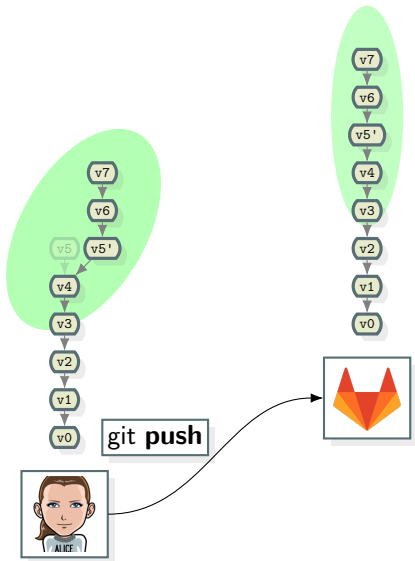
# Backup sur un server distant



# Backup sur un server distant



# Backup sur un server distant



# Backup sur un server distant

On synchronise des **branches**  
en échangeant  
des morceaux d'historiques



# Où en sommes nous ?

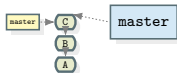
Contexte

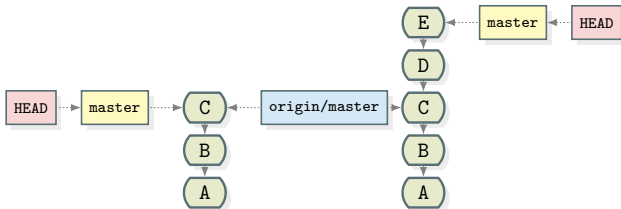
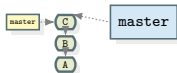
L'historique quand on est tout seul

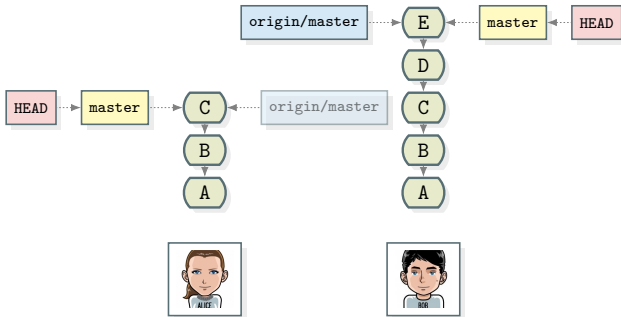
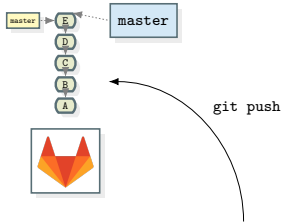
Travailler à plusieurs

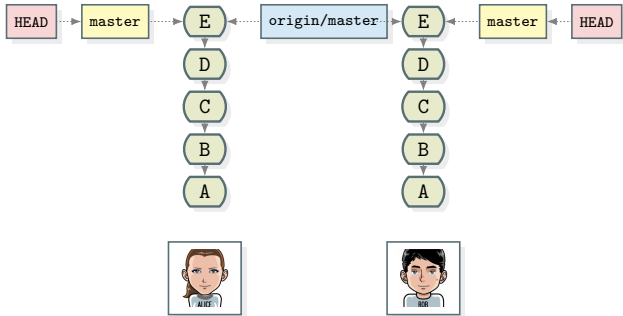
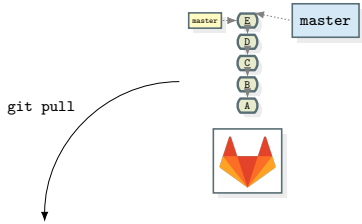
L'écosystème Git

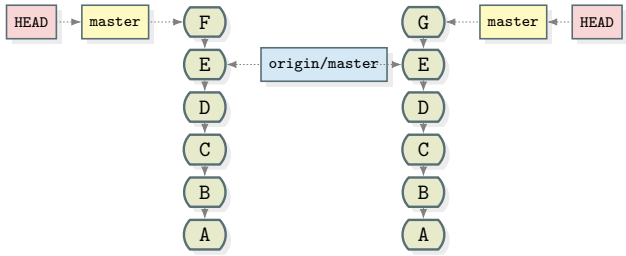
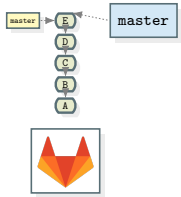


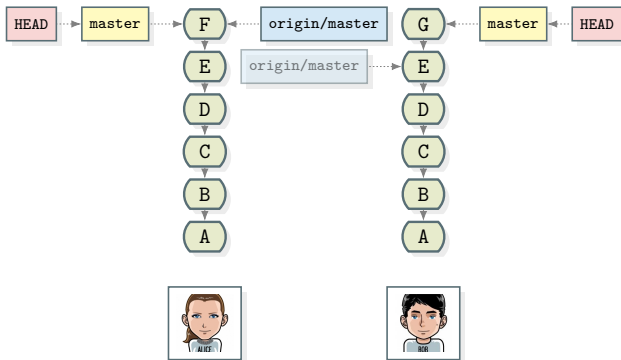
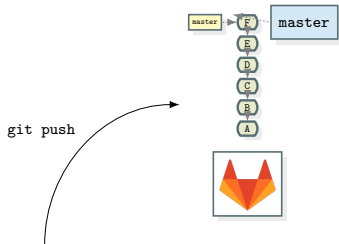


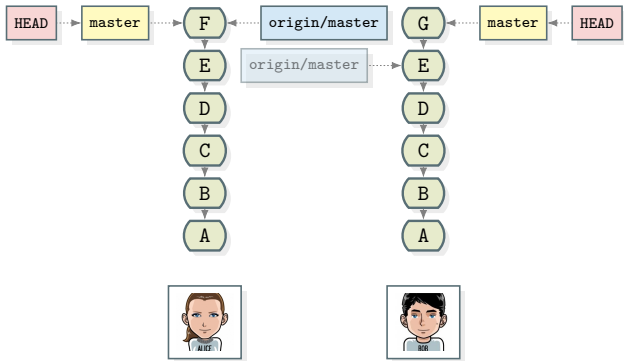
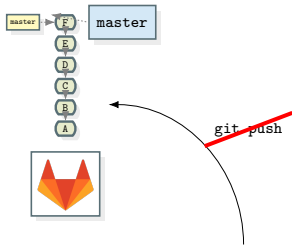


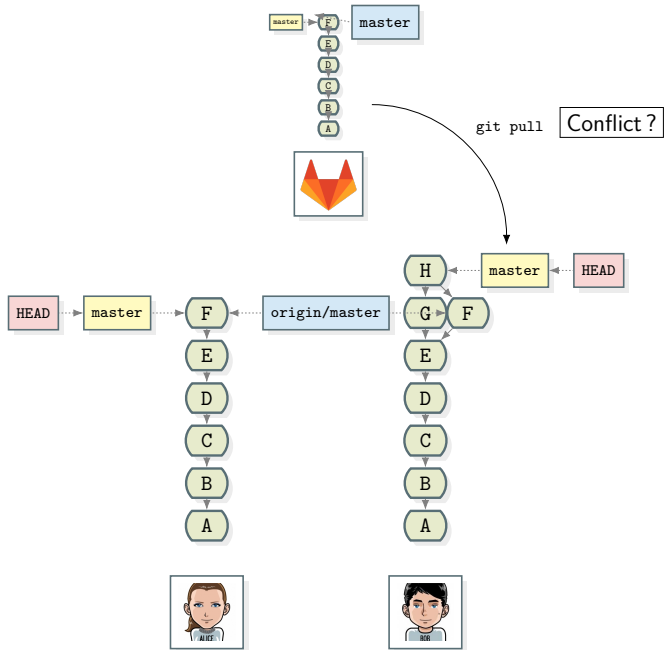




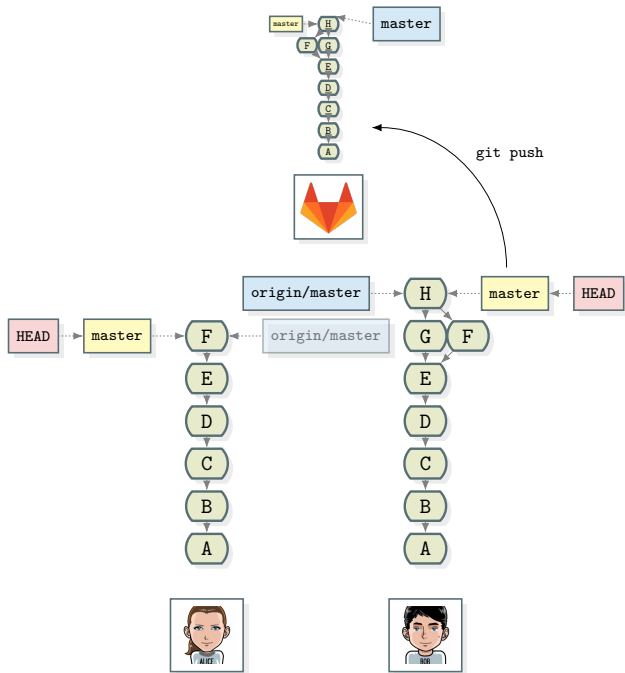


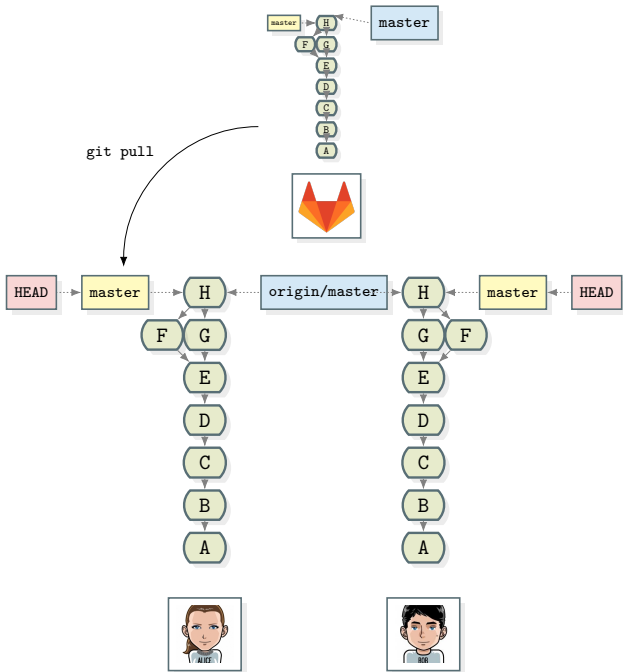












## À retenir

- Collaborer = gérer un historique distribué
- Le serveur ne sert à rien : Alice et Bob pourraient échanger directement leurs historiques
- Chacun a tout l'historique  $\rightsquigarrow$  très résilient

## Le conflit

Au moment du pull, il se peut que le merge ne soit pas possible et qu'un conflit apparaisse. Pas de panique !

**Démonstration !**

# Le conflit

Au moment du pull, il se peut que le merge ne soit pas possible et qu'un conflit apparaisse. Pas de panique !

## Démonstration !

- Git travaille à la granularité du fichier et de la ligne  
⇒ Modularisez autant que possible

# Le conflit

Au moment du pull, il se peut que le merge ne soit pas possible et qu'un conflit apparaisse. Pas de panique !

## Démonstration !

- Git travaille à la granularité du fichier et de la ligne
  - ⇒ Modularisez autant que possible
- Il y aura conflit dans un fichier si la même ligne a été modifiée de chaque côté (même un espace)
  - ⇒ Ne réindentez pas votre texte pour rien !
  - ⇒ Faites des commits différents pour les modifications de fonds, de forme (git **status** ; git **diff** ; git **add**)
  - ⇒ Faites de **petits** commits **logiques** (~~git commit -a~~)

# Le conflit

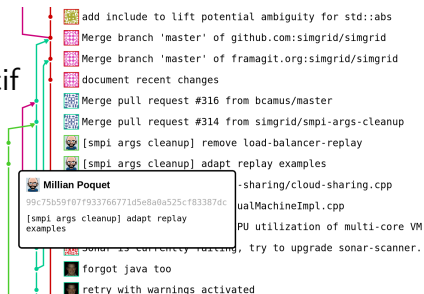
Au moment du pull, il se peut que le merge ne soit pas possible et qu'un conflit apparaisse. Pas de panique !

## Démonstration !

- Git travaille à la granularité du fichier et de la ligne
  - ⇒ Modularisez autant que possible
- Il y aura conflit dans un fichier si la même ligne a été modifiée de chaque côté (même un espace)
  - ⇒ Ne réindentez pas votre texte pour rien !
  - ⇒ Faites des commits différents pour les modifications de fonds, de forme (git **status** ; git **diff** ; git **add**)
  - ⇒ Faites de **petits** commits **logiques** (~~git commit -a~~)
- Les fichiers binaires n'ont pas de lignes... le diff et la fusion sont souvent impossibles
  - ⇒ Préférer le format texte

# En conclusion, git :

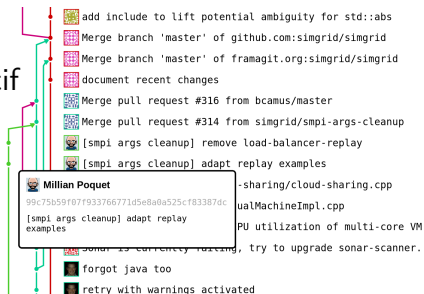
- Ça versionne ! ( « hist » )
- Ça facilite le travail collaboratif ( « merge » ) !
- Ça permet de stocker ses modifications localement ( « commit » ) !
- Ça permet de partager ses modifications quand on le souhaite ( « pull/push » ) !





# En conclusion, git :

- Ça versionne ! ( « hist » )
- Ça facilite le travail collaboratif ( « merge » ) !
- Ça permet de stocker ses modifications localement ( « commit » ) !
- Ça permet de partager ses modifications quand on le souhaite ( « pull/push » ) !



Mais comment mettre ça en place ?

- Comment avoir un serveur de backup ?
- Comment gérer qui peut lire/écrire ?

# Où en sommes nous ?

Contexte

L'historique quand on est tout seul

Travailler à plusieurs

L'écosystème Git

# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

## Un réseau social des développeurs

- Statistiques, émulation, gestion de communauté
- *Issues*, correction de bugs, revue de code

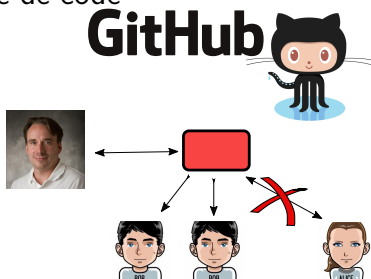
# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

## Un réseau social des développeurs

- Statistiques, émulation, gestion de communauté
- *Issues*, correction de bugs, revue de code
- Le *fork* et la *pull request*



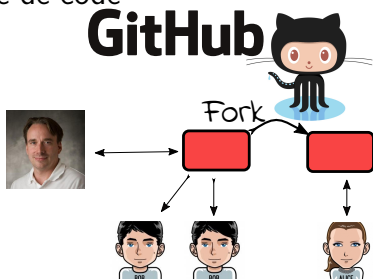
# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

## Un réseau social des développeurs

- Statistiques, émulation, gestion de communauté
- *Issues*, correction de bugs, revue de code
- Le *fork* et la *pull request*



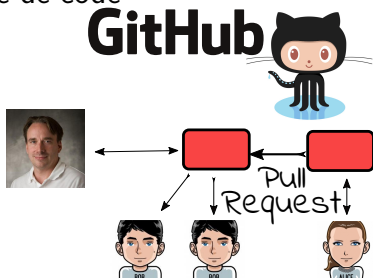
# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

## Un réseau social des développeurs

- Statistiques, émulation, gestion de communauté
- *Issues*, correction de bugs, revue de code
- Le *fork* et la *pull request*



# Les plates-formes de développement collaboratif (1/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics
- Interfaces web (navigation, preview, édition en ligne)
- Gestion des permissions (lecture/écriture, public/privé) et des utilisateurs

## Un réseau social des développeurs

- Statistiques, émulation, gestion de communauté
- *Issues*, correction de bugs, revue de code
- Le *fork* et la *pull request*

## Des tas de fonctionnalités incroyables [→MOOC RR avancé]

- Intégration continue
- Déversement vers des archives (Zenodo, SWH)



## Les plates-formes de développement collaboratif (2/2)

GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics ?!?

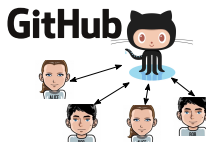
# Les plates-formes de développement collaboratif (2/2)

## GitHub, GitLab, et les autres !

- Hébergement **gratuit** de projets publics ?!?

GitHub Leaders du marché mais logiciel propriétaire

- Projets privés :
  - gratuit pour étudiants/académiques
  - payant pour les autres



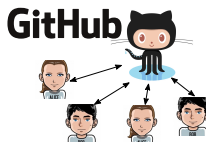
# Les plates-formes de développement collaboratif (2/2)

## GitHub, GitLab, et les autres !

- Hébergement **gratuit** de projets publics?!?

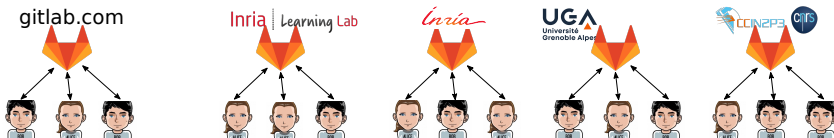
**GitHub** Leaders du marché mais logiciel propriétaire

- Projets privés :
  - gratuit pour étudiants/académiques
  - payant pour les autres



**GitLab** Logiciel libre, plusieurs instances

- Instance `gitlab.com` similaire à GitHub
- Compagnies, institutions : support avancé



# Les plates-formes de développement collaboratif (2/2)

## GitHub, GitLab, et les autres !

- Hébergement gratuit de projets publics ?!?

## GitHub Leaders du marché mais logiciel propriétaire

- Projets privés :
  - gratuit pour étudiants/académiques
  - payant pour les autres

## GitLab Logiciel libre, plusieurs instances

- Instance `gitlab.com` similaire à GitHub
- Compagnies, institutions : support avancé

## Lequel choisir ? (+ FramaGit, BitBucket, SourceForge, ...)

- Visibilité/pérénnité
- Projets privés ou publiques
- Confidentialité des données

# Interfaces graphiques

- JupyterLab-git
- Rstudio
- Emacs Magit
  
- GitHub Desktop
- GitKraken
- SmartGit