

Git



Git



- Utile pour :
 - Gérer son code en conservant un historique des différentes versions :
 - Utilise git en local
 - Gérer le travail à plusieurs en partageant le code et en gardant une trace des différentes versions :
 - Utilise git en local ET sur un serveur distant

Git en S1 pour la SAE



- Créer un dépôt sur github en mode privé
- Utiliser le README pour décrire le projet et son avancée
- Faire un dossier par membre du groupe : dans chacun mettre **régulièrement** la description du travail réalisé dans un README.
- Cloner le projet sur les ordinateurs de travail.
- Mettre régulièrement à jour (push / pull)

GIT en S1 pour la SAE



- Nous inviter avec l'adresse universitaire :
 - Philippe.Cam@univ-cotedazur.fr
 - Nina.Singlan@univ-cotedazur.fr
 - Michel.Gautero@univ-cotedazur.fr

Git local (0)



- Installation :
 - Pour linux (ubuntu/debian) :
 - «sudo apt install git-all »
 - Pour macos :
 - Installer Xcode (AppleStore)
 - Ou « brew install git »
 - Après avoir installé « homebrew » (« <https://brew.sh/> »)

Git local (0-1)



- Installation :
 - Pour Windows :
 - <https://git-scm.com/downloads/win>
 - Ou de préférence utiliser « WSL » (Linux dans Windows)
 - Ou une machine virtuelle linux

Git local (1)



- Se faire connaître de git :
 - « git config --global user.email "Vous@exemple.com" »
 - « git config --global user.name "Votre Nom" »

```
mgautero@mgautero-Latitude-5510:~$ cat .gitconfig
[user]
    email = mgautero@gautero.fr
    name = Gautero Michel
mgautero@mgautero-Latitude-5510:~$
```

Git local (2)



- Pour débiter :
 - Créer le dossier du projet, nommé « X » ici :
 - « mkdir X »
 - Se déplacer dans le dossier « X » :
 - « cd X »
 - Initialiser le dépôt git :
 - « git init »

Git local (3)



```
mgautero@mgautero-Latitude-5510: ~/gittest
mgautero@mgautero-Latitude-5510:~/gittest$ git init
astuce: Utilisation de 'master' comme nom de la branche initiale. Le nom de la branche
astuce: par défaut peut changer. Pour configurer le nom de la branche initiale
astuce: pour tous les nouveaux dépôts, et supprimer cet avertissement, lancez :
astuce:
astuce:      git config --global init.defaultBranch <nom>
astuce:
astuce: Les noms les plus utilisés à la place de 'master' sont 'main', 'trunk' et
astuce: 'development'. La branche nouvellement créée peut être renommée avec :
astuce:
astuce:      git branch -m <nom>
Dépôt Git vide initialisé dans /home/mgautero/gittest/.git/
mgautero@mgautero-Latitude-5510:~/gittest$
```

Git local (3-1)



- Création d'un fichier « .gitignore »
 - Dans le dossier du projet
 - Qui contiendra dossiers et fichiers à ignorer
 - <https://www.atlassian.com/git/tutorials/saving-changes/gitignore>

Git local (4)



- Pour ajouter des fichiers au projet :
 - Travailler sur le fichier en développement nommé ici « code.php » dans « X »
 - L'ajouter aux fichiers suivis : « git add code.php »
 - L'ajouter à l'historique :
 - « git commit -m "Message" code.php »
 - Le message devrait faire au plus 72 caractères

Git local (5)



- Pour ajouter des fichiers au projet :
 - Pour les modifications suivantes :
 - « `git commit -m "Texte descriptif" -a` »
 - Fait le commit sur tous les fichiers déjà ajoutés et qui ont été modifiés.
 - Pour modifier le dernier « commit » :
 - « `git commit --amend -m 'blabla'` »

Git local (6)



- Pour voir la différence entre la version actuelle et la dernière version « gittée » :
 - « git diff » pour tous les fichiers
 - « git diff code.php » pour uniquement le fichier « code.php »

Git local (7)



- Pour voir l'état du dépôt :
 - « git status »

```
mgautero@mgautero-Latitude-5510:~/Documents/TestVsCode$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

rien à valider, la copie de travail est propre
```

- Après modification d'un fichier

Git local (8)



- Après modification d'un fichier

```
mgautero@mgautero-Latitude-5510:~/Documents/TestVsCode$ git status
```

```
Sur la branche main
```

```
Votre branche est à jour avec 'origin/main'.
```

```
Modifications qui ne seront pas validées :
```

```
(utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
```

```
(utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
```

```
    modifié :          python/README.md
```

```
aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
```

Git local (9)



- Après modification d'un fichier :
 - « git log » : historique des commits

```
mgautero@mgautero-Latitude-5510:~/Documents/TestVsCode$ git log
commit 1f2630c80a7896b4a9c6a0f255588cdedf4886e5 (HEAD -> main, origin/main)
Author: Gautero Michel <mgautero@gautero.fr>
Date:   Wed Jan 31 12:08:41 2024 +0100
```

Création

Git local (10)



- Pour revenir à une version précédente :
 - « git restore fichier1 fichier2 ... »
 - Efface les modifications faites en local pour les remplacer par la dernière version « commitée »
 - «git restore --source numeroDeCommit fichier1 fichier2 ... »
 - Permet de revenir à la version représentée par le commit du numéro donné

Git : les branches (1)

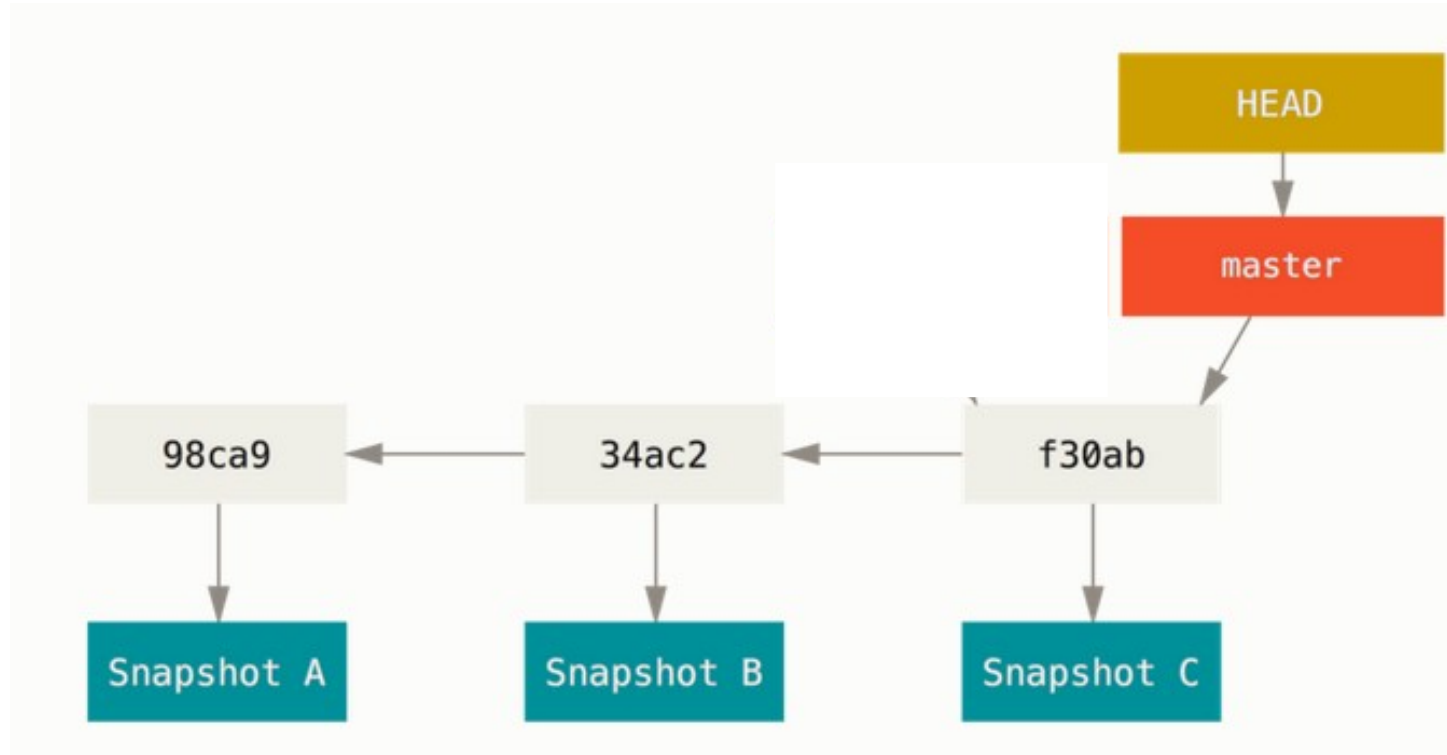


- Une branche montre :
 - L'état d'un projet
 - À un moment donné
- Par défaut, une seule branche :
 - Mais on peut en créer de nouvelles
 - Pour tester diverses solutions
 - Ou pour développer en parallèle

Git : les branches (2)



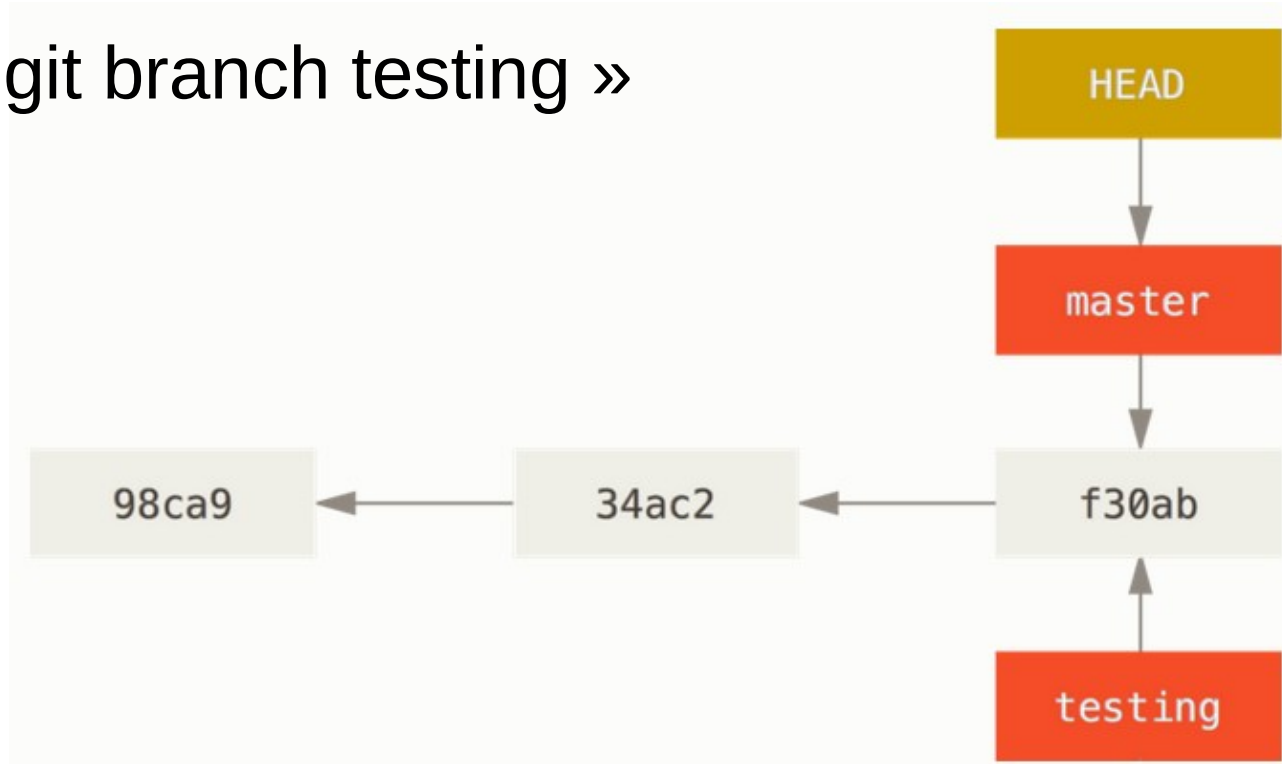
- Avec la branche principale



Git : les branches (3)



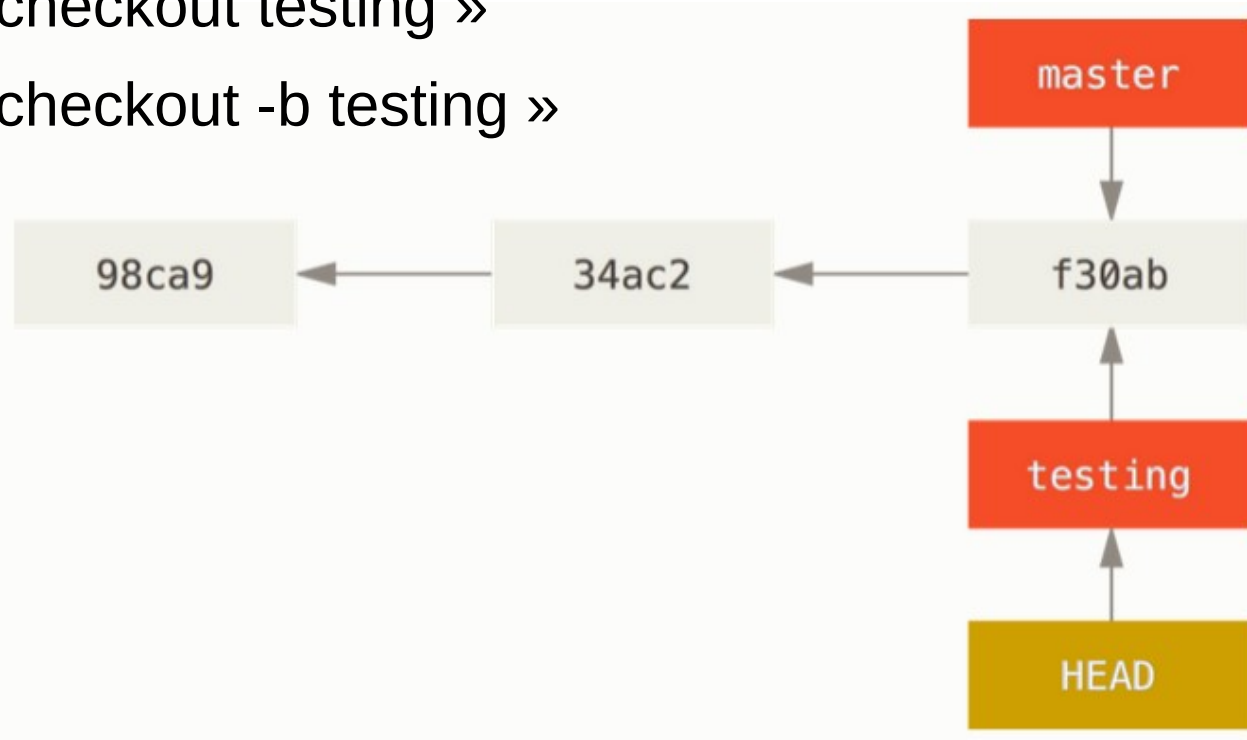
- Création d'une nouvelle branche « testing »
 - « git branch testing »



Git : les branches (4)



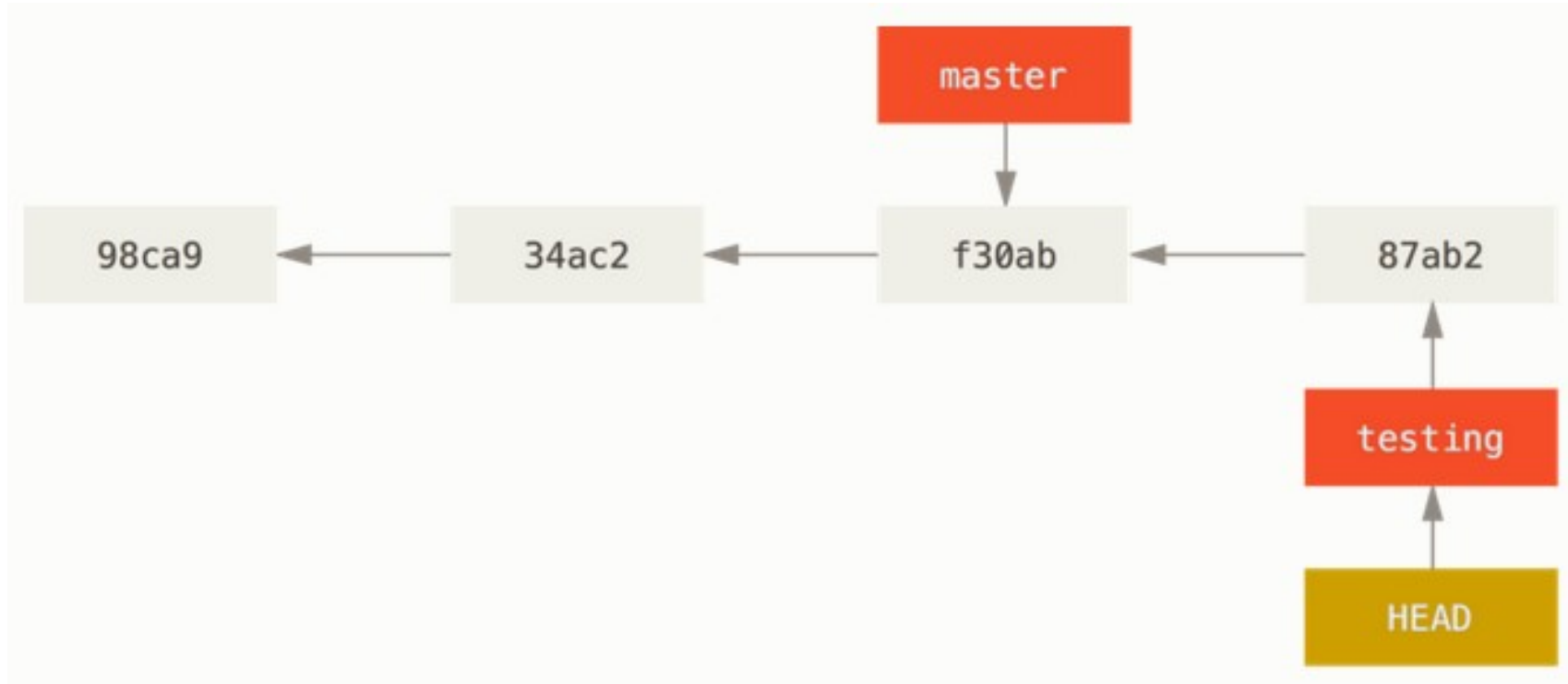
- En ayant basculé sur la branche « testing »
 - « git checkout testing »
 - « git checkout -b testing »



Git : les branches (5)



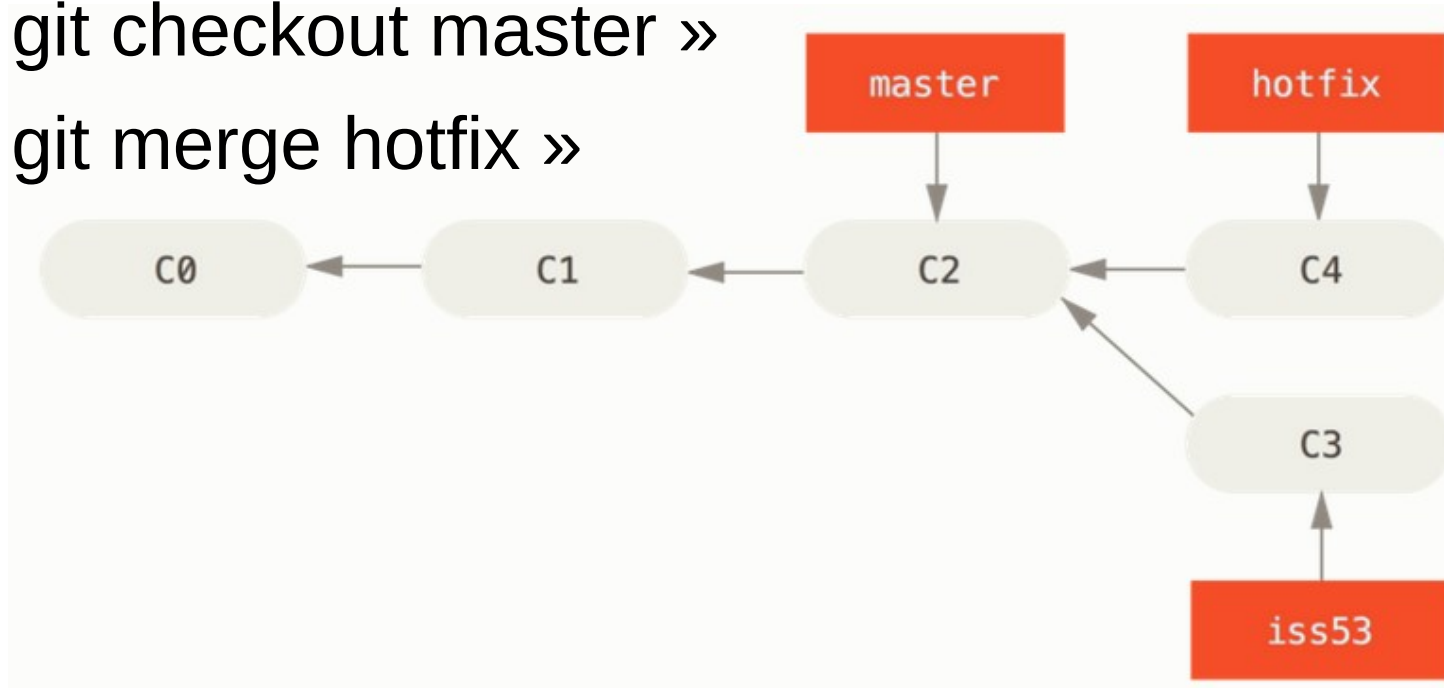
- Après un nouveau « commit »



Git : les branches (7)



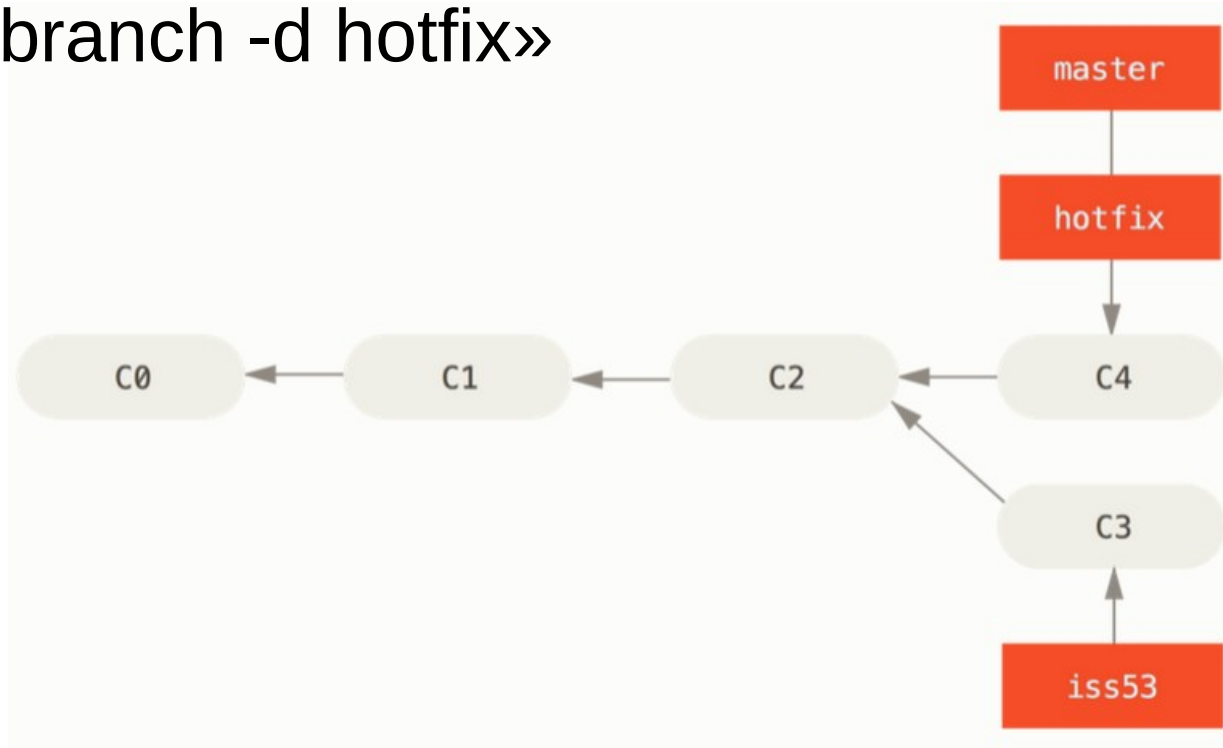
- Fusion de « master » et de « hotfix »
 - « git checkout master »
 - « git merge hotfix »



Git: les branches (8)



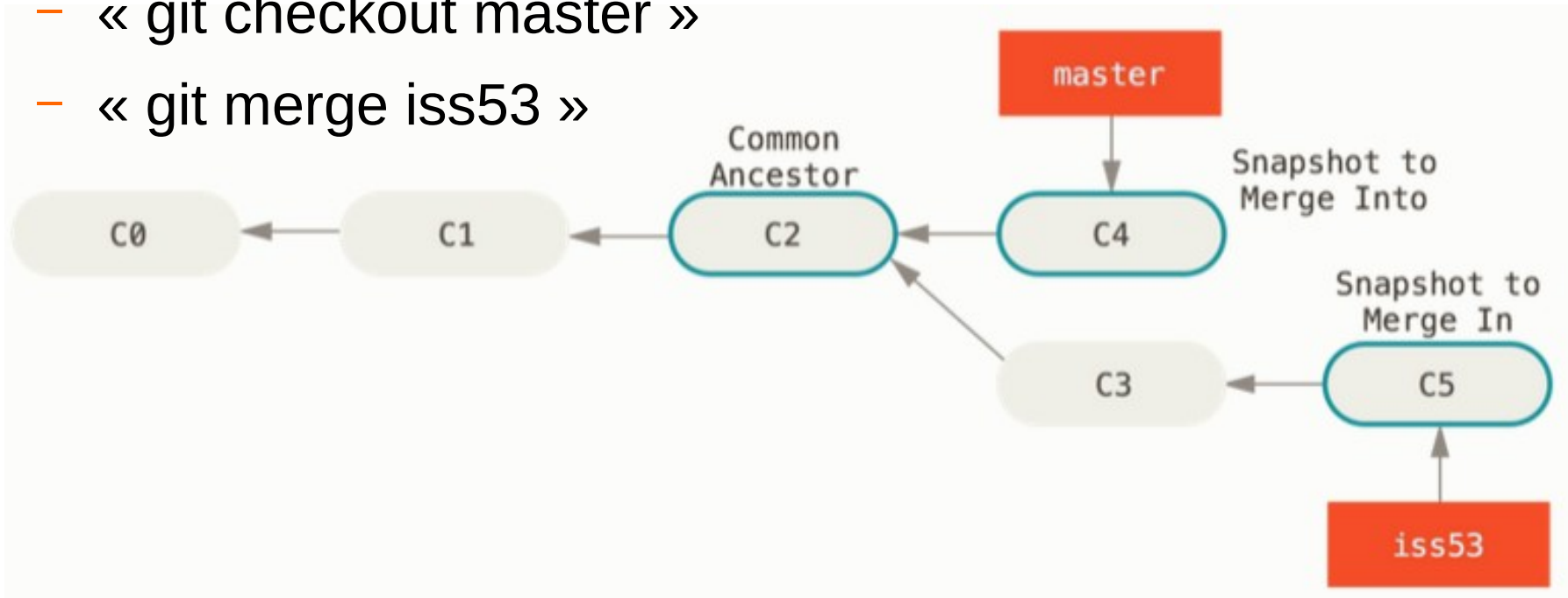
- Effacement d'une branche :
 - « git branch -d hotfix »



Git: les branches (9)



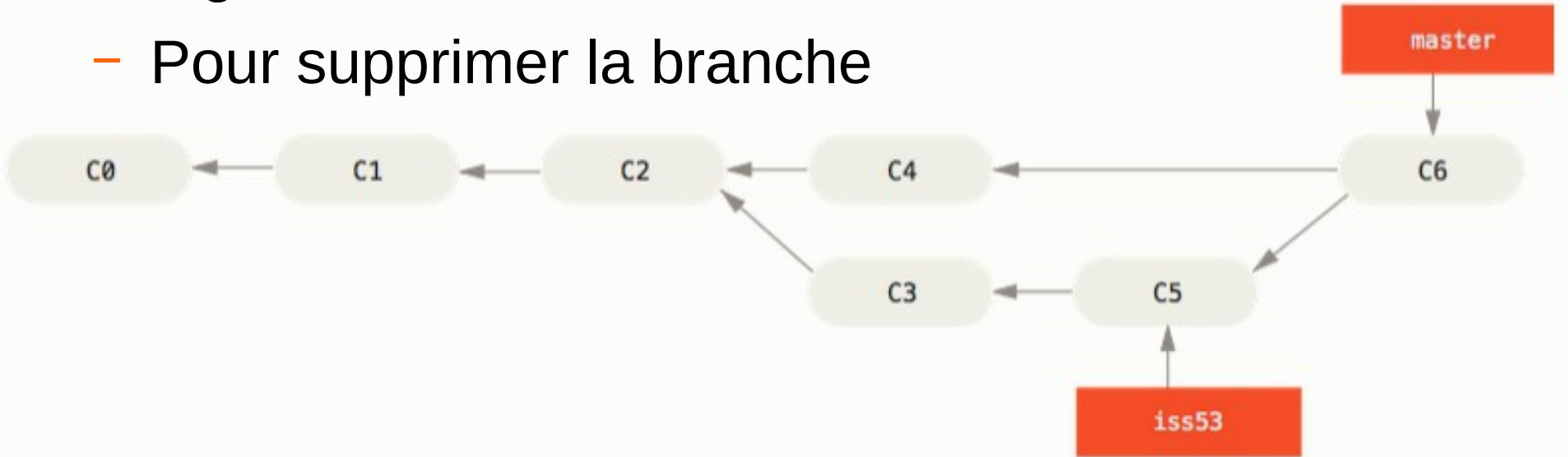
- Fusion de « master » et « iss53 »
 - « git checkout master »
 - « git merge iss53 »



Git : les branches (10)



- Résultat fusion « master » et « iss53 »
 - « git branch -d iss53 »
 - Pour supprimer la branche



Git : les tags



- Mettre un tag permet de marquer un commit sans créer de nouvelle branche.
- Permet de plus facilement y revenir dessus
- Création : « `git tag -a v0.1 -m "my version 0.1"` »
- Suppression :
 - « `git tag -d v0.1` » pour supprimer en local

Git distant (1)



- On utilisera « github » (<https://github.com>) :
 - Créer son compte
 - Créer un « repository » :
 - « Private », « Add a README file », « Add a .gitignore », « Choose a licence (GPL) »
 - Ajouter une clé ssh :
 - « Settings à partir de l'icone en haut à droite »
 - « ssh and gpg keys », « New ssh key »

Git distant (1-1) / ssh



- Pour faire la clé ssh :
 - Pour Ubuntu/Debian :
 - Installer le client ssh si nécessaire :
 - « sudo apt install openssh-client »
 - Puis « ssh-keygen -t ed25519 » et répondre aux questions posées (fichier destination...)
 - Pour MacOS :
 - « ssh-keygen -t ed25519 »
 - Pour Windows : utiliser « WSL »

Git distant (1-2) / ssh



- Copier le texte généré pour la clé publique
 - Par défaut stocké dans « .ssh/id_ed25519.pub » dans votre dossier de départ
- Le coller dans la zone texte proposée par github

Git distant (1-3) / ssh




mgautero (mgautero)

Your personal account

[Go to your personal profile](#)

 Public profile



 Account

 Appearance

 Accessibility

 Notifications

Access

 Billing and plans 

 Emails

 Password and authentication

 Sessions

 **SSH and GPG keys**

 Organizations


 Enterprises

 Moderation 

Add new SSH Key

Title

Key type

Authentication Key 

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Git et Github



- Pour ne pas avoir à retaper tout le temps, la clé ssh, faire :
 - «git config credential.helper 'cache --timeout=<timeout>' »
 - Le timeout par défaut est de 15 minutes

Git distant (2)



- En local :
 - Créer un dossier pour le projet
- Récupérer un projet déposé sur github : « git clone url »
- Lister les branches distantes : « git branch -a »
- Recréer en local une branche distante :
 - « git branch MissZ origin/MissZ » crée une branche locale MissZ et la lie à la branche distante

Git distant (3)



- Envoyer les derniers « commit » vers github :
 - « git push » en étant dans le dossier du projet
 - Envoie les modifications de la branche locale active

Git distant (4)



- Récupérer la dernière version du projet :
 - « git fetch »
 - Pour récupérer les dernières modifications de la branche en cours mais sans faire de « merge » local.
 - « git pull »
 - Pour faire le « git fetch » et ensuite automatiquement faire les « merge » voulus
 - Le tout en étant dans le dossier du projet

Git distant (5)



- Supprimer un tag distant :
 - « git push origin --delete v0.1 »
- Supprimer une branche distante :
 - « git push origin --delete nomDeBranche »

Git distant (4)



- Inviter des collaborateurs à son projet :

General

Access

Collaborators

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

[Add people](#)