GAIA
APE

Versioning *moderne* avec Git

Sylvain Théry

thery@unistra.fr

ICube

Versioning : Version Control System

Sylvain THERY
24 mars 2023

Versioning ?

- conserver toutes les modifications et versions
- permet de récupérer facilement
  - une version spécifique d'un fichier (date, tag, …)
  - l'état de votre code à instant t ou une version v
- uniquement pour les sources (pas de binaire)

Comment ?

- *copies, archives*: fastidieux, beaucoup de place, difficile de retrouver un fichier spécifique.
- mieux : utiliser un **V**ersion **C**ontrol **S**ystem
- encore mieux : un **D**istributed **V**ersion **C**ontrol **S**ystem -> **Git**

GAIA

iCUBE

Sylvain THERY
24 mars 2023

Scénarios

Développeur isolé:

- retrouver le code avant une série de modifications ?
- retrouver la date d'une modification ?

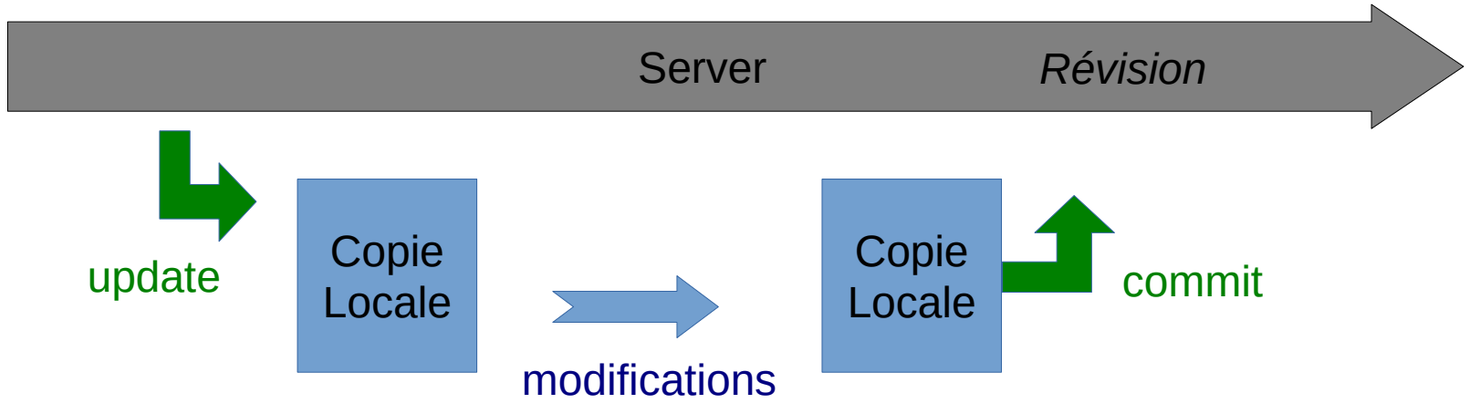
Développeur en collaboration sur projet

- savoir qui a codé quoi ?
- à quel moment ?
- simplifier la gestion des conflits, les éviter ?

Utilisation basique d’un VCS

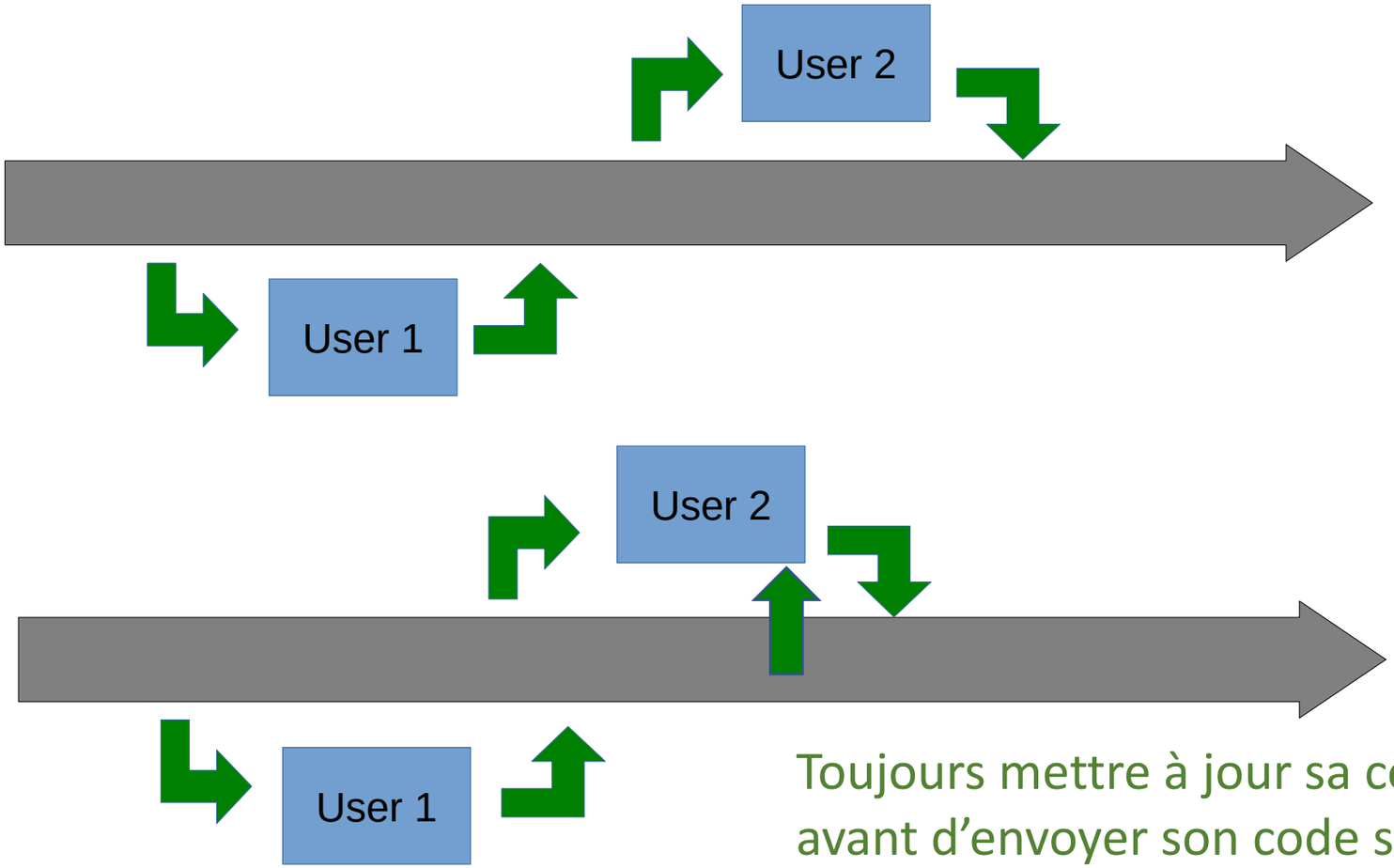
Sylvain THERY
24 mars 2023

1. Creation/mise à jour de la copie locale depuis le serveur
2. modificatons
3. Envoit des modifications (commit) au serveur



Multi-utilisateur

Sylvain THERY
24 mars 2023



Toujours mettre à jour sa copie avant d'envoyer son code sur le serveur.
Il faut régler les conflits avant d'envoyer les modifications

Historique

Sylvain THERY
24 mars 2023

Local Version Control System
SCCS (1972)
RCS (1982)

Centralized Version Control System
CVS (1990)
Subversion / svn (2000) *avantages: plus sur, historique stocké sur un serveur*

Distributed Version Control System
***Git**, Mercurial, … (2005)* *avantages:*
sauvegarde sur plusieurs serveurs
utilisation réseau local /global (+ fichiers + HDD + clé USB)
possibilité de travailler en local temporairairement

Lancé par Linus Torval pour le développement du noyau Linux

GAIA

iCUBE

Sylvain THERY
24 mars 2023

Premier Conseil

Utiliser Git en ligne de commande

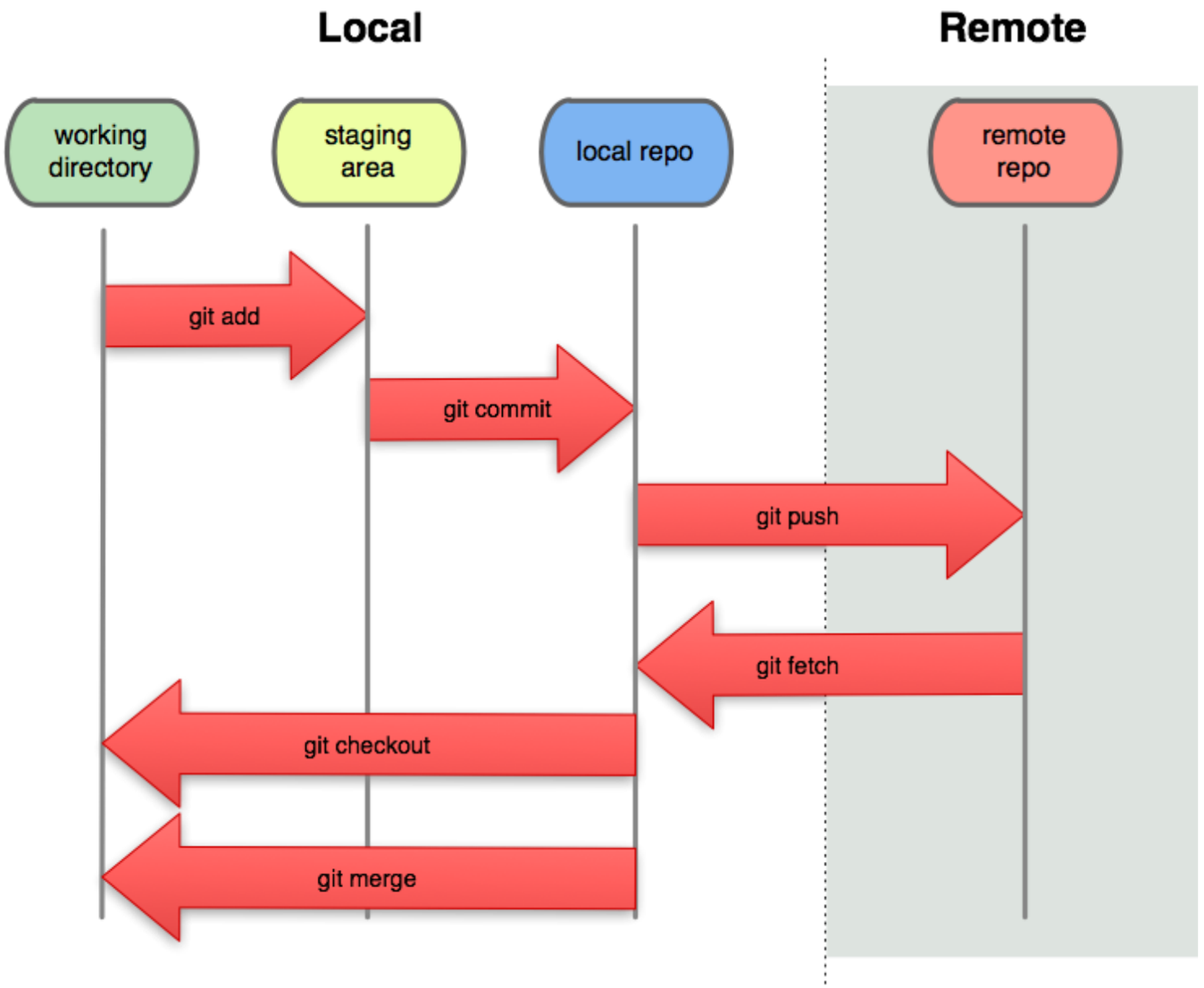
Permet d'utiliser de nombreuses options très utiles,
Affiche des messages très pertinents.

git help commande affiche l'aide de chaque sous commande
git help git pour l'aide globale

Les interfaces poussent à une utilisation type *subversion* perdant au passage les nombreux avantages de git

Les différentes *zones* de Git

Sylvain THERY
24 mars 2023

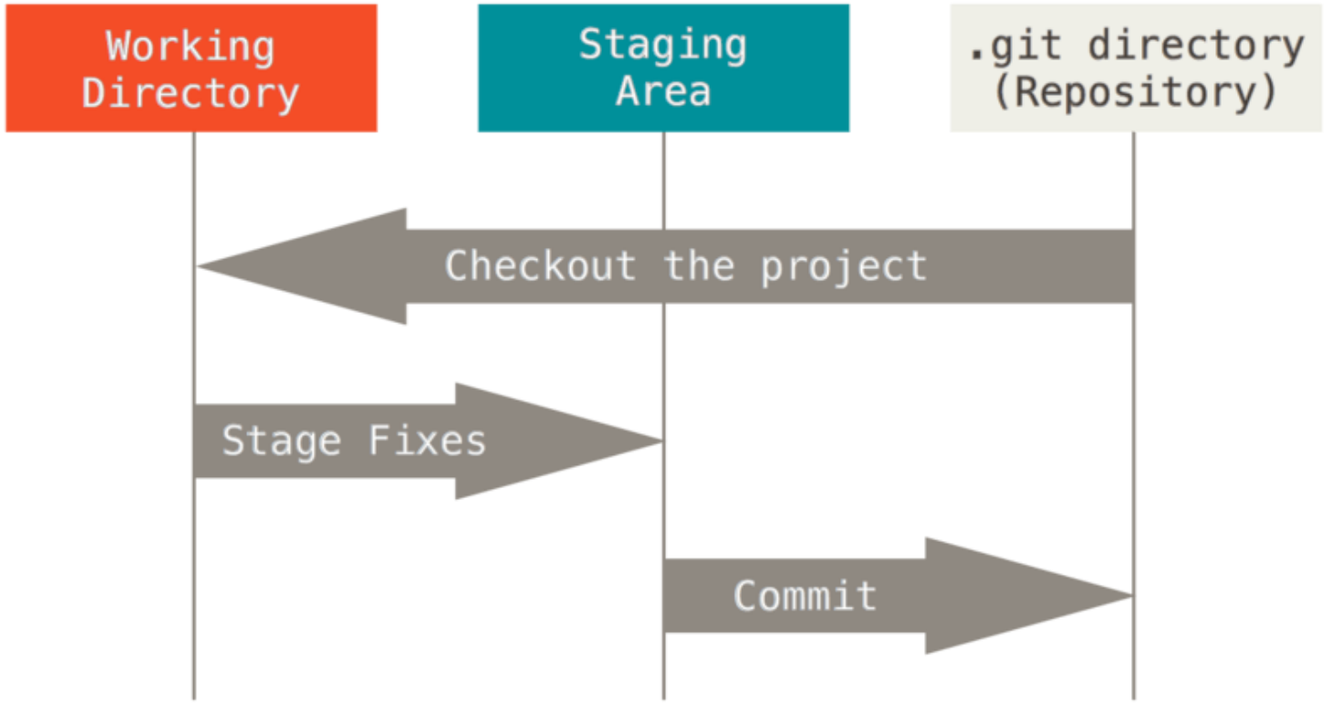


Zone d’index de Git

Sylvain THERY
24 mars 2023

La zone d’index (staging Area) est une zone tampon entre le depot local et la copie locale de travail

Elle permet d’envoyer dans le dépôt local**, une partie** des modifications faites dans la copie de travail (add & commit)



Sylvain THERY
24 mars 2023

Pour commencer

Deux possibilités:

- utiliser un projet existant (depuis github ou gitlab)
- faire son propre dépôt distant localement

Pour utiliser un dépôt existantr: **`git clone url_depot`**

- crée un répertoire contenant la copie de travail
- le dépôt local et l'index sont cachés dans le rep *.git/*
- toute commande git exécutée dans la copie de travail s'applique au dépôt.

Alternatives:

- commencer en créant un dépôt vide avec git init
- Pour créer son propre dépôt distant local :

  `git init --bare`

  l'option `--bare` permet de créer un depot local sans répertoire de travail

GAIA

iCUBE

git add

Sylvain THERY
24 mars 2023

`git add` a deux fonctions:

- demander à git d'utiliser un/des fichiers
- demander à git d'ajouter les modifications d'un fichier dans l'index.

Remarques: les fichiers doivent toujours être ajoutés explicitement

Erreur Classique: oublie d'ajouter un fichier avant de faire le commit

Solution: toujours faire `git status` avant le commit

GAIA

iCUBE

Sylvain THERY
24 mars 2023

git commit

`git commit` :

- demande à git d'ajouter au dépôt local les modifications référencées dans l'index

- `-a` ajoute tous fichiers modifiés de la copie de travail dans l'index avant de faire le commit

- `-m` permet mettre un message "online", sinon il ouvre un editeur de texte pour saisir le message (obligatoire)

- `--amend -m` permet de changer/corriger le dernier commit

GAIA

iCUBE

Conseil commit

Sylvain THERY
24 mars 2023

Utiliser **git commit** sans l'option **–a**
en conjunction avec **git add**

Permet de faire des commit plus homogènes, qui ont un sens, contenant des modifications inter-dépendantes

Mettez des commentaires pertinents

permets de faire des recherches dans l'historique.

GAIA

iCUBE

Conseil: git status

Sylvain THERY
24 mars 2023

Avant toutes autres commandes faire un ***`git status`***
pour verifier l'état de votre dépôt / index / copie locale

Donne les infos sur l'état de

- la copie locale
- la zone d'index
- le dépôt local (par rapport à son origine)

GAIA

iCUBE

git status

Sylvain THERY
24 mars 2023

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 7 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 7 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   example.cpp
        modified:   first_src.cpp

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        data.txt
```

GAIA

iCUBE

git diff

Sylvain THERY
24 mars 2023

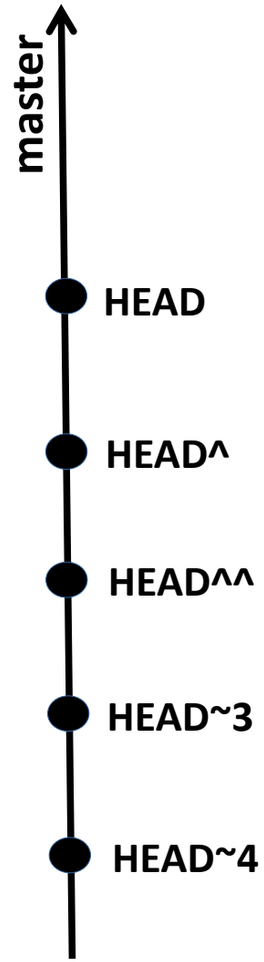
`git diff` affiche les differences entre:
la copie de travail et le dépôt local

`git diff` ***`commit_hash`*** les differences entre:
la copie locale et le depot local après ce commit

`git diff` ***`hash1 hash2`*** les differences entre:
le depot local après hash1 et après hash2

On peut utiliser des raccourcis: HEAD dernier commit
HEAD^ le precedent, HEAD^^ et HEAD~n …

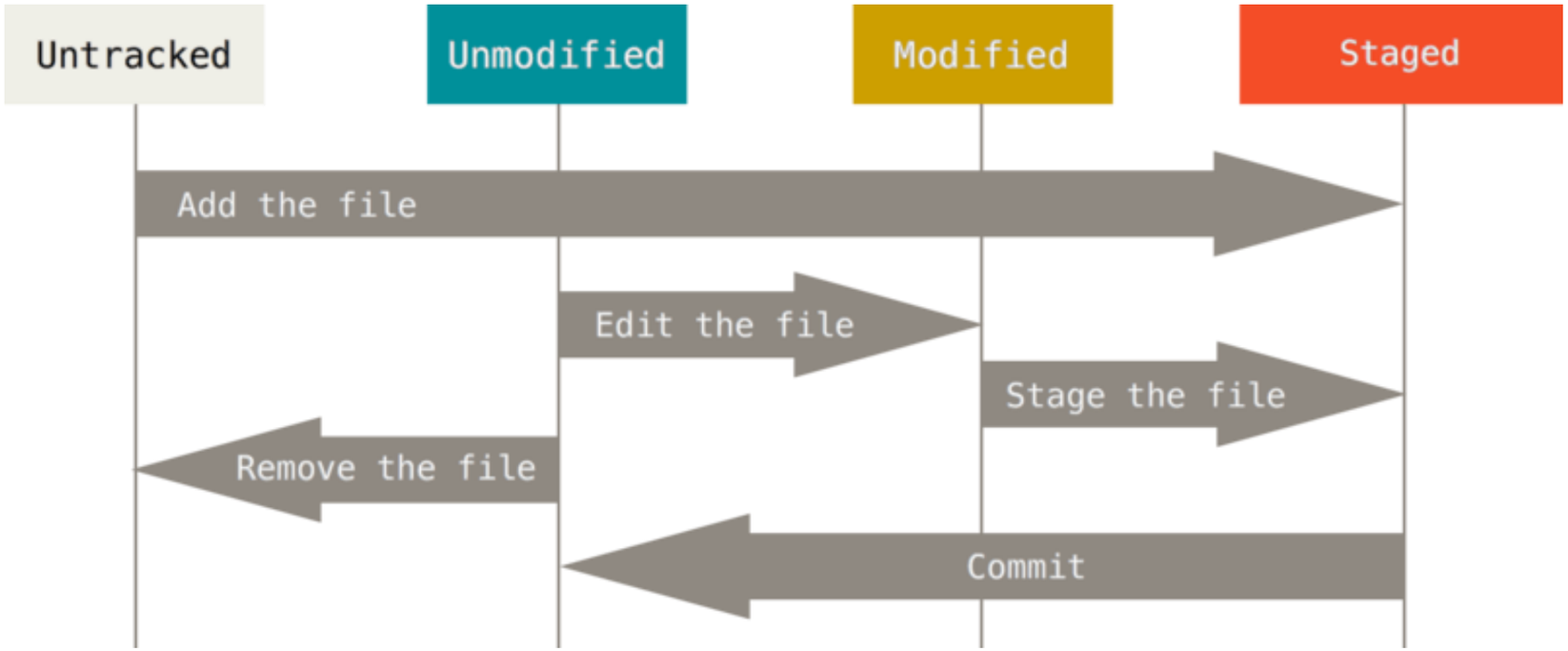
Pas très lisible, mieux vaut utiliser une interface externe



Etats d’un fichier

Sylvain THERY
24 mars 2023

Tout fichier (de la copie de travail) peut avoir 4 états distincts



.gitignore

Eviter de laisser trainer des fichiers inutiles / indésirables dans la copie locale:

- polluent les messages de la commande **`status`**
- peuvent être ajoutés au dépôt par erreur

Essayer de ne pas polluer la copie de travail:

- compilation dans un repertoire séparé (CMake)

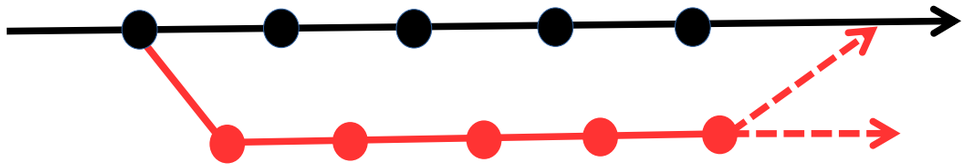
Mais parfois c'est inévitable
Meilleure façon de résoudre le problème:
ajouter un fichier ***.gitignore*** dans le dépôt

Il contient la liste des fichiers (et repertoires) qui seront ignorés par git

Branches

Branche

- séparer le développement d'une fonctionalité
- séparer le travail d'un programmeur
- une branche est repéré par sa tête (dernier commit)



Tag

- Indique version spécifique (1.3)

Conseil : faites des branches

Sylvain THERY
24 mars 2023

Dans les “vieux” VCS la creation de branches étaient couteuse en temps et espace disque.

Avec git c’est gratuit

Conseil:

Ne travaillez jamais dans votre branche principale

Créez une branche pour chaque nouveau sujet

- Permet de travailler en // sur plusieurs sujets
- Permet de basculer rapidement vers un développement urgent

GAIA

iCUBE

Sylvain THERY
24 mars 2023

commandes *branches*

git branch –a liste les branches (-a pour les branches des serveurs)

git branch *brch* crée une branche nommé *brch*

git branch –d *brch* supprime une branche nommé *brch*
(aucun commit n'est supprimé, juste la reference à la branche)

git checkout *brch* change le pointeur HEAD vers le dernier commit de la branche *brch*
-> la copie de travail locale est mise à jour

git merge *brch* Fusionne une branche dans la branche courante

git rebase *brch* déplace les commits a la fin de la branche *brch*

Liste exhaustive sur https://git-scm.com/docs

GAIA

iCUBE

git log

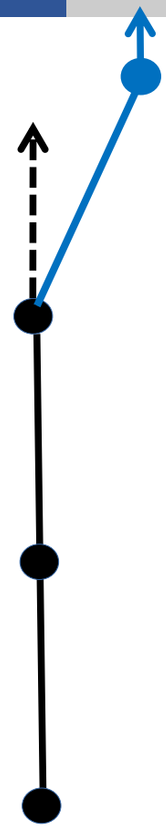

```
commit 45df4893c76a863afc4596ebd48d139f81771c80 (HEAD -> br_test)
Author: Sylvain Thery <thery@unistra.fr>
Date:   Sun Jan 8 15:53:51 2023 +0100

    add some tests

commit 71186514c90b93388f80c4317488063491l5b5df (master, tag: v0.9)
Author: Sylvain Thery <thery@unistra.fr>
Date:   Sun Jan 8 15:49:26 2023 +0100

    add example

commit da9e5f80b476fb5f1270644b296321b91cea8c9f
Author: Sylvain Thery <thery@unistra.fr>
Date:   Sun Jan 8 15:47:25 2023 +0100

    a first source file

commit eff988def4edd7287d817618a59213286aa61d0b (serveur1/master)
Author: Sylvain Thery <thery@unistra.fr>
Date:   Sun Jan 8 15:44:57 2023 +0100

    first commit
```

Tête de la branche courante

Tag info

point de divergence

Origine de la branche courante
nom du serveur / branche



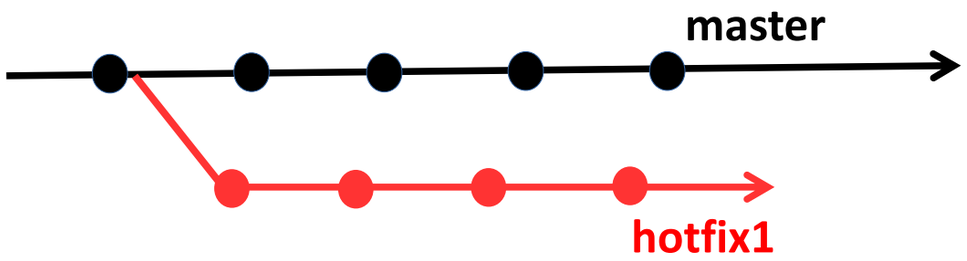
Nombreuses possibilités et options:

```
git log v1.2.. compute.cpp
```

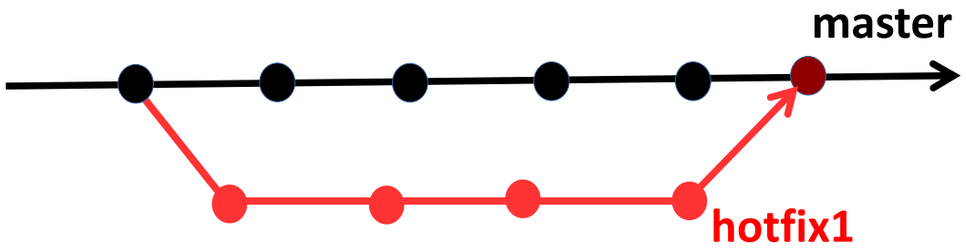
affiche tous les commits qui ont modifiés le fichier compute.cpp depuis la version taggée v1.2

Fusion de branches

Sylvain THERY
24 mars 2023



Lors que le développement de la fonctionnalité est terminée, il doit être intégré à la branche d'origine.



Attention: si les développements effectués en parallèle concernent les même fichiers, il peut y avoir des conflits.

GAIA

iCUBE

Conflit lors d'une fusion

Sylvain THERY
24 mars 2023

- Si des modifications des deux branches ne sont pas compatibles, il y a conflit(s) (message)

- git instrumente le source avec une séries de

  <<<<<<< HEAD

  …

  =======

  …

  >>>>>>> BRANCH-NAME

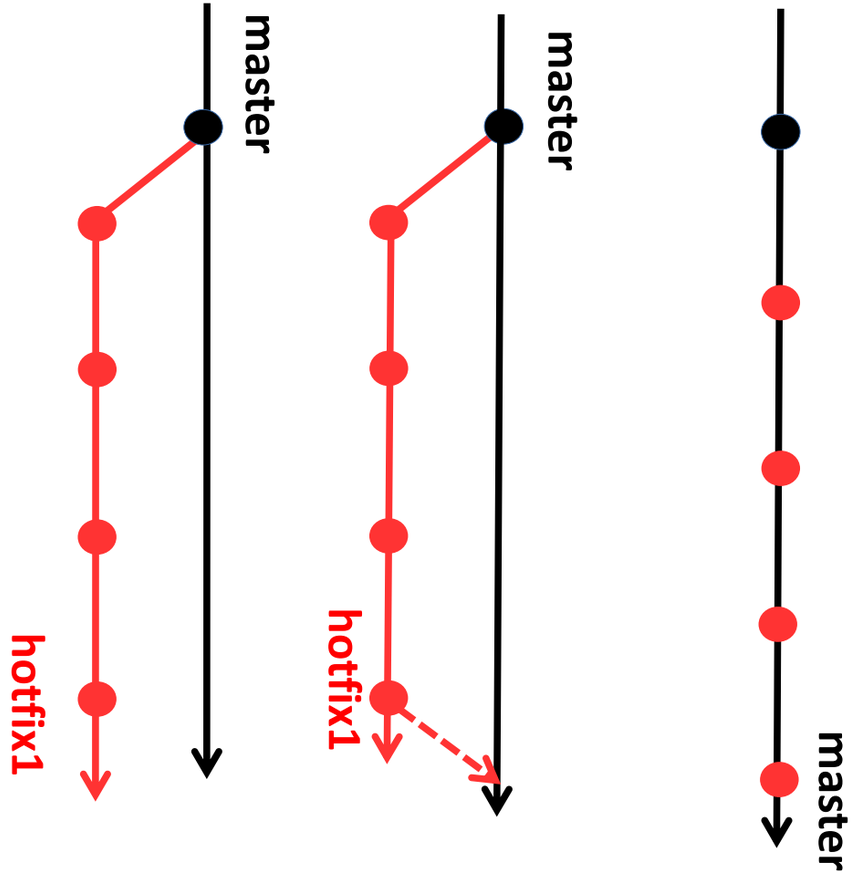
- git vous laisse choisir

- une fois les modifications finalisées: *git add & git commit*

GAIA

iCUBE

Fast-forward ?



Ici il n'y a eu aucun commit sur la branche master pedant la *vie* de la branche hotfix1

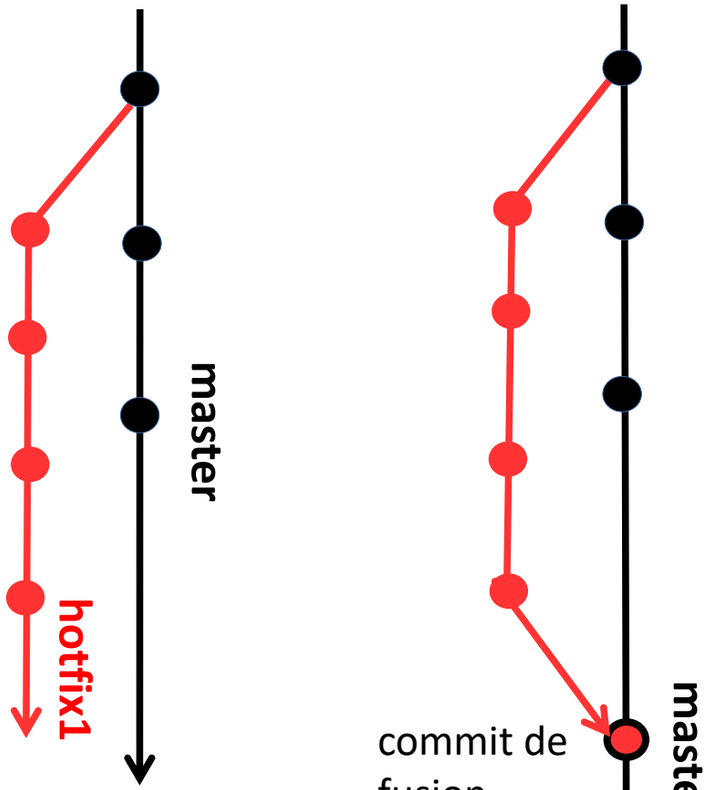
Il est ici inutile de garder la divergence des deux branches dans l'historique.

git vous informe qu'il va faire un **fast-forward**

Il déplace les commits de la branche hotfix1 vers la branche master

No fast-forward

Sylvain THERY
24 mars 2023



Ici la branche master a été modifiée en parallèle des modifications de la branche hotfix1

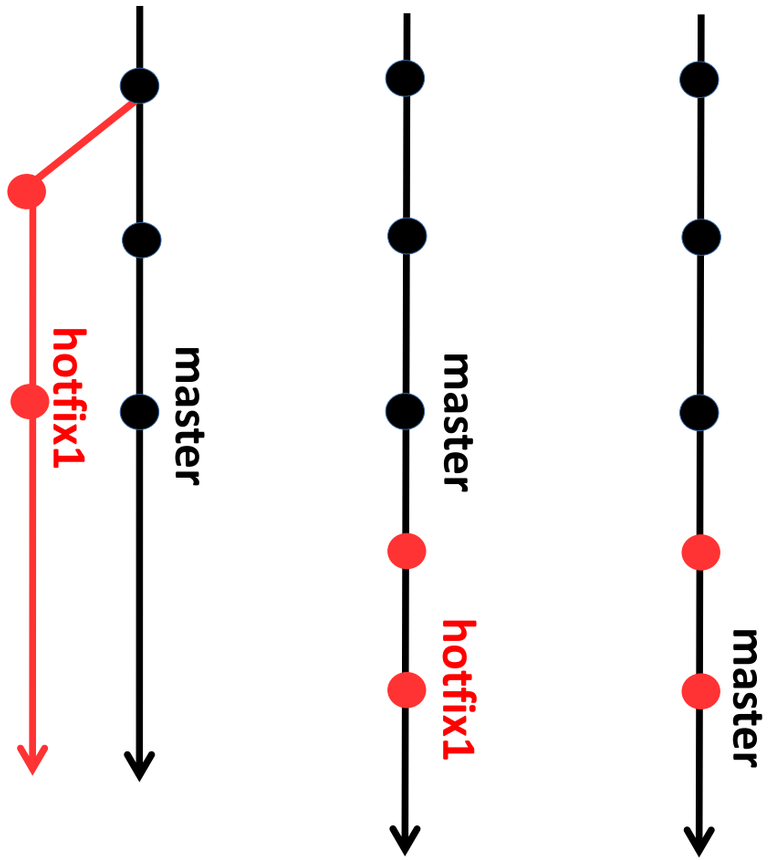
Un *vrai* merge doit être fait.

Un commit spécifique de fusion apparaitra dans l'historique.

On peut forcer ce comportement même si il n'y pas de commit dans master en utilisant l'option **no-ff** de **merge**

Rebasing

Sylvain THERY
24 mars 2023



A l'opposée de l'option no-ff, il est possible de forcer git à générer un historique simple et linéaire même si les deux branches ont des modifications.

Il faut déplacer les commits de hotfix1 pour les appliquer derrière ceux de master

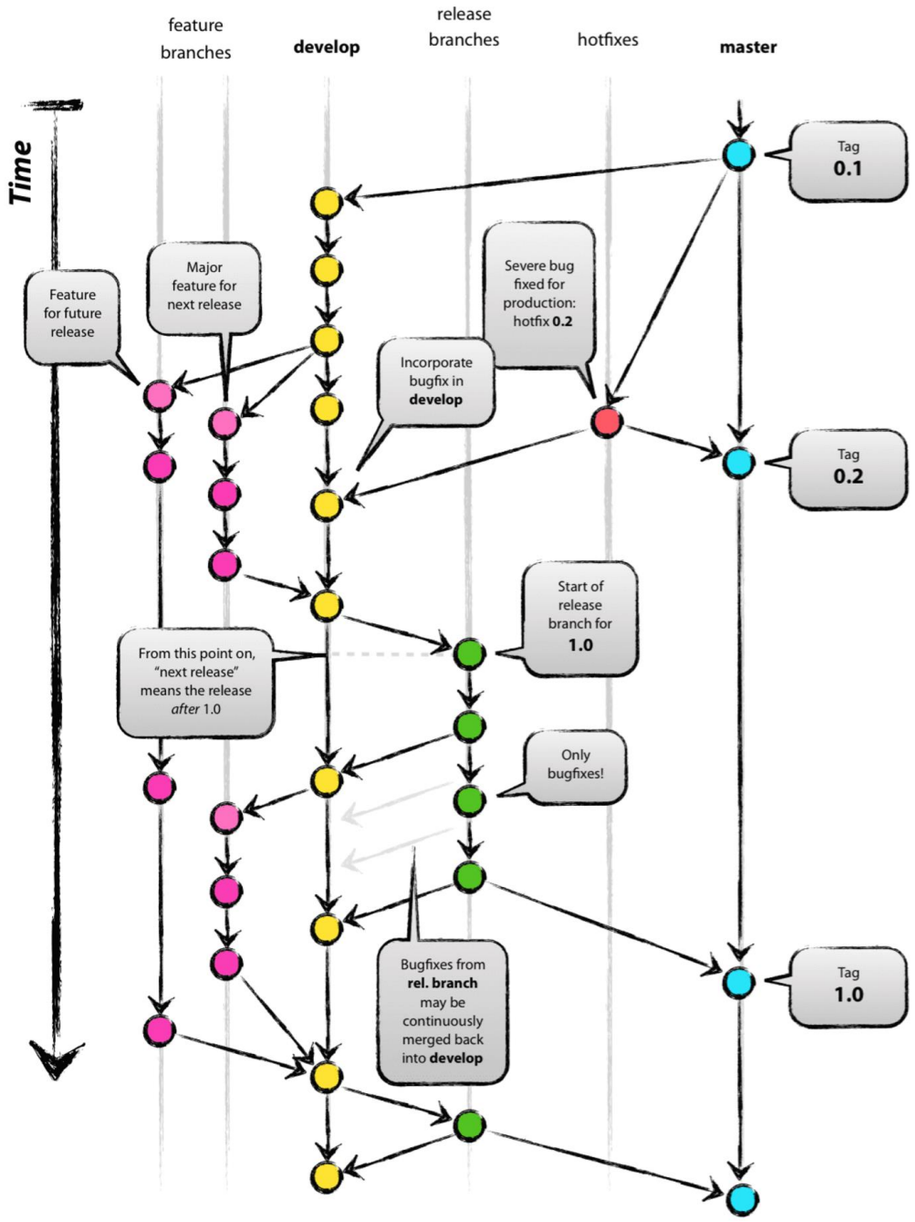
Attention : les conflits possibles sont gérés pendant le rebase

```
git checkout hotfix1
git rebase master
git checkout master
git merge hotfix1
git branch -d hotfix1
```

Branches & Stratégie

Sylvain THERY
24 mars 2023

Exemple de stratégie classique pour un *gros* projet



Git checkout

git checkout peut être utilisé pour

1. changer de branche
   `git checkout branch_name`

2. se déplacer vers n'importe quel commit (place HEAD sur le commit)
   donné par son code sha : ***`git checkout sha`***
   relativement à la tête de la branche *brch*: ***`git checkout brch~N`***

3. mettre à jour un fichier ou un repertoire de la copie de travail locale depuis sa version du dépôt local.
   depuis la branche courante: ***`git checkout -- file_dir`***
   depuis la branche *br1* : ***`git checkout br1 file_dir`***

Attention cette commande peut effacer les modifications de la copie locale de travail (non annulable)

GAIA

iCUBE

Sylvain THERY
24 mars 2023

Fréquence taille des commits

Comment choisir quand faire un commit ?

- Souvent (petits commits) : Historique complet mais lourd
- Rarement (gros commits): Historique simple mais incomplet

=> A chacun de trouver son compromis

Remarque:
tant que les commits sont uniquement dans le repo local on peut éditer l'historique des *n* dernier commits avec `git rebase -i HEAD~n`

GAIA

iCUBE

Git stash

Sylvain THERY
24 mars 2023

Comment rapidement sauvegarder toutes les modifications sans faire de commit ? Une autre solution existe la commande **stash**

`git stash` permet de stocker des modifications dans une zone privée et de les réappliquer quand vous le voulez. Lors du stockage il annnule toutes vos modifications de la copie locale, qui est alors à jour avec le dépôt.

Scénario: j'ai oublié de faire une branche

```
git stash                    sauve les modifications
git checkout –b testing      crée la branche
git stash apply              réapplique les modif.
```

Remarque: Ne s'applique qu'aux modifications

Git revert (annuler un commit)

Sylvain THERY
24 mars 2023

Comment annuler un commit ?

Il est fortement déconseillé d'effacer un commit
afin de préserver l'intégrité du dépôt (distribué)

git revert génère un commit qui annule des modifications:

`git revert` *`commit`* — *annule un commit*

`git revert` *`commitA..commitB`* — *annule une suite de commits*

`git revert` *`HEAD`* — *annule le dernier commit*

`git revert` *`HEAD..HEAD~3`* — *annule les 4 dernier commits*

Sylvain THERY
24 mars 2023

Git remote

git peut se connecter à plusieurs serveurs pour y stocker l'historique des commits

Le dépôt local contient la dépôts distants (url)

La commande `git remote` permet de gérer cette liste

- affichage: git remote [-v]
- ajouter: git remote add repo_name url
- enlever : git remote rm repo_name
- renommer : git remote rename old new

Le nommage d'un depot permet une utilisation plus pratique:
git remote add **central** *git@github.com:organization/projetX*

GAIA

iCUBE

Commandes dépôt local / distant

Sylvain THERY
24 mars 2023

git clone *url* : copie le depot distant dans le dépôt local

git push : envoit les commits du dépôt local vers son dépôt d'origine.

git pull [*url]*: met à jour le dépôt local (et la copie) à partir de son dépôt d'origine.

git fetch *[url]*: met à jour le dépôt local

Liste exhaustive sur https://git-scm.com/docs

Conseil: pas de pull

Sylvain THERY
24 mars 2023

git pull met à jour le dépôt local et la copie de travail : Danger

1 mettre à jour le dépôt local avec *git fetch*

2 vérifier les modifications avec *git diff*

3 fusionner la branche distante rapatriée dans la branche locale avec *git merge*

GitHub.com / GitLab

Sylvain THERY
24 mars 2023

Comment (ou) héberger son dépôt distant:

github.com / gitlab.unistra.fr / icube-forge.unistra.fr

Pourquoi ?

- Backup
- Visualisation du code & branches
- Edition interactive
- **Pull-request / Merge-request**
- Publication /mise à disposition
- Intégration continue

GAIA

iCUBE

Conseil github/gitlab

Sylvain THERY
24 mars 2023

Utiliser la connection par ssh:

ajouter vos clés ssh publiques dans l’interface

utiliser les url de type : git@github.com:compte/projet

Plus besoin d’entrer le mot de passe

Pull Request

Sylvain THERY
24 mars 2023

Comment collaborer à un projet ?

- Faire un fork (clone interne à github)
  => sa version privée du projet

- Travailler sur sa version du projet

- Proposer ses modifications sous la forme d'une branche fusionnable, sans conflit, avec une pull-request

GAIA

iCUBE

Pull Request

Sylvain THERY
24 mars 2023

Vous créez une pull-request à partir d'une de vos branches

La proposition est étudiée par le responsable

Phase de discussion, liée au code:

Le responsable vous demande une modification

Vous la faite et la *pusher* sur la branche de la pull-request

Elle apparait directement dans l'interface

Le responsable valide et merge votre branche

Conclusion

Sylvain THERY
24 mars 2023

Utilisez git aussi souvent que possible

Utilisez les commandes dans un shell

Faites des commits homogènes régulièrement.

Avec des messages informatifs

Faites des branches

Utiliser GitHub / GitLab

References

https://git-scm.com/

https://www.atlassian.com/git/

http://nvie.com/posts/a-successful-git-branching-model/

http://rogerdudler.github.io/git-guide/index.fr.html

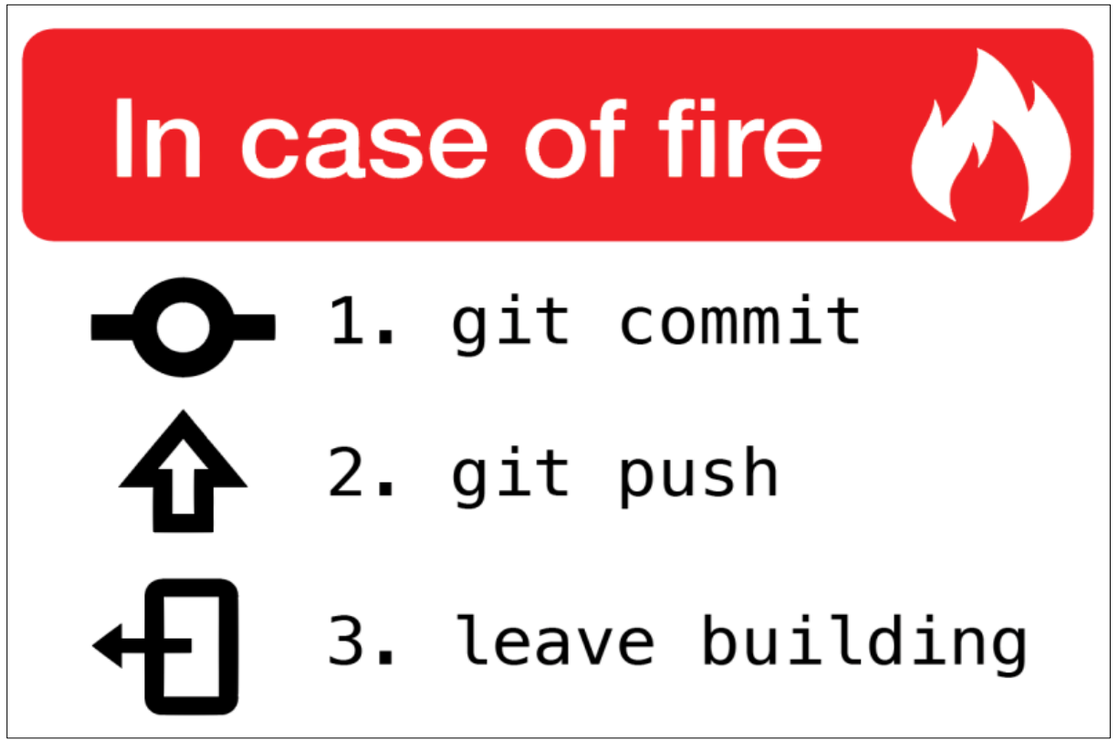
https://marklodato.github.io/visual-git-guide/index-fr.html

http://ndpsoftware.com/git-cheatsheet/previous/git-cheatsheet.html

https://learngitbranching.js.org/

GAIA

iCUBE



For some slide images thanks to:
https://frank.taillandier.me/
https://git-scm.com/book/