Système de gestion de version

GIT

CESI Nancy

Yoann Calamai - 2021

cesi ÉCOLE SUPÉRIEURE DE L'ALTERNANCE

HESAM UNIVERSITÉ

Sommaire

Sommaire

Exemples

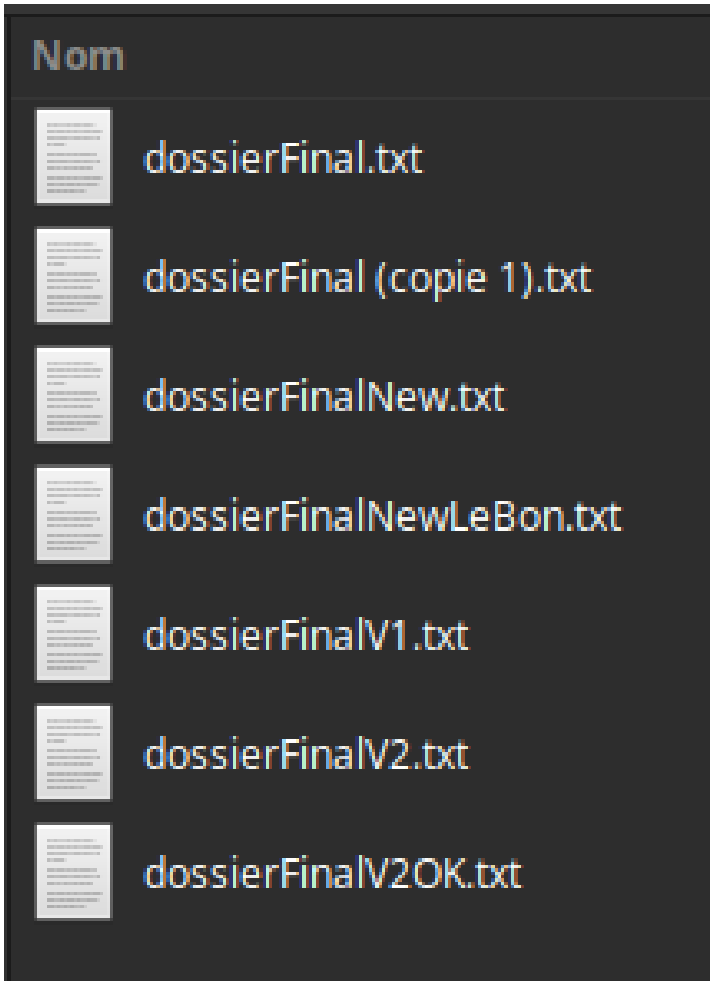
Nous avons travaillé avec un collègue sur le fichier client. Nous avons eu plusieurs versions de travail nommées *Client-1.1.doc*, *Client-1.2.doc* et ainsi de suite.
Il a nommé la version finale *Client-1.0.doc*.

Exemples

Le 15 mars, un collègue m'a envoyé un fichier nommé *2020-03-17-contrat.docx*. Je l'ai modifié le 16 mars.
Quel nom lui donner ?

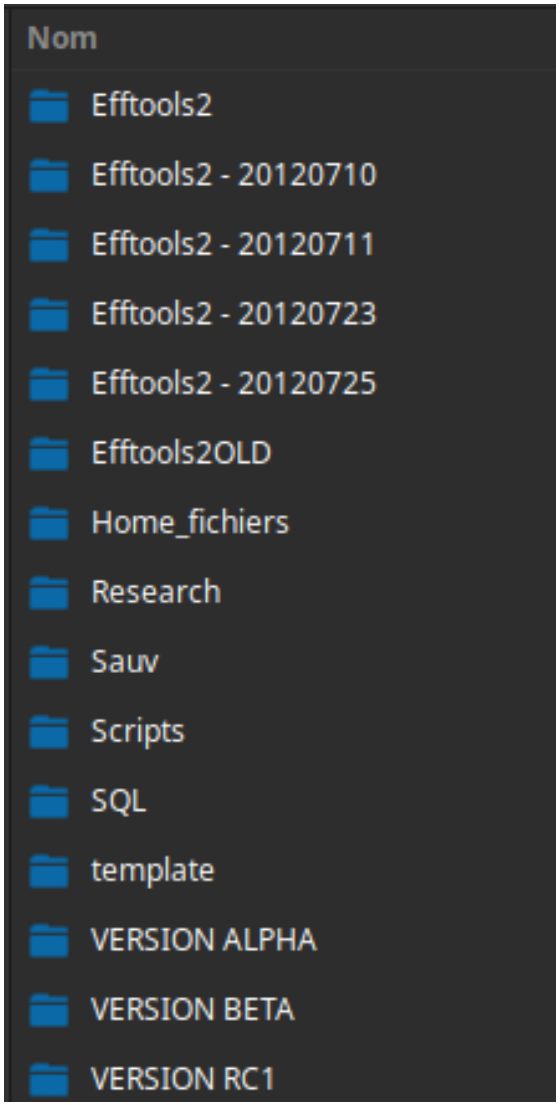
Exemples

Quel est le fichier contenant les dernières modifications ?



Exemples

Quel répertoire contient les sources courantes du projet ?



Exemple de suivi de modification manuel

Document Control

Filename	IDD_[redacted]_FRA_KE5-GLS_Absence&WorkedHoursImport_v5_20151001.docx
Original Author(s)	Nick
Revision Author(s)	Nick

Change Record

Version	Date	Author (s)	Revision Notes
1.0	18/08/2015	Nick	Initial draft for review.
2.0	25/08/2015	Nick	Updates following initial review
3.0	10/09/2015	Nick	Updates / Remove Job
4.0	24/09/2015	Nick	Include Channels lookup table
5.0	01/10/2015	Nick	Updates for import periods / last run check

Approvals Distribution List

Copy No	Approver Name	Title	Approval Date
1	Jeremy	Project Manager	
2	Stan	Project Manager	
3	Marcus	Practice Manager	
4	Laurent	Practice Manager	

Exemples

Exemple de suivi de modification manuel

- Suivi entièrement manuel
- Peut contenir des erreurs/oublis
- Détail des changements non visible

Sommaire

Version Control System (VCS)

La gestion des versions est un travail fastidieux et méthodique.

Les humains ne sont pas doués pour les travaux fastidieux et méthodiques.

Laissons cela à l'ordinateur et concentrons-nous sur la partie du travail où nous sommes meilleurs que l'ordinateur.

Version Control System (VCS)

Les avantages des VCS

- Sauvegarde
- Conservation de l'historique des fichiers (qui a fait quoi quand ?)
- Possibilité de retour en arrière
- Fusion des modifications lors du travail collaboratif
- Visualiser les changements au cours du temps

Version Control System (VCS)

Evolution des VCS

- Systèmes centralisés
  - CVS (Concurrent Versioning System, le papy)
  - SVN (Subversion, très populaire)
- Systèmes décentralisés
  - GIT
  - Mercurial (Hg)
  - Bazaar (bzr)

Sommaire

Fonctionnement global

Les notions

- Dépôts (ou repository)
- Révision (ou commit)
- Copie de travail (ou working copy)
- Index

Fonctionnement global

Les dépôts (ou repository)

- Le répertoire caché .git
- Il contient toutes les données dont GIT a besoin pour gérer l'historique

Fonctionnement global

Les révisions (ou commit)

- L'historique d'un projet est une séquence de commit
- Un commit est caractérisé par
  - une date
  - un auteur
  - une description textuelle
  - un lien vers le commit précédent
  - les différences par rapport au commit précédent

Fonctionnement global

La copie de travail (ou working copy)

- Ce sont les fichiers effectivement présents dans le répertoire géré par GIT
- Leur état peut être différent du dernier commit de l'historique

Fonctionnement global

Les indexes

- L'index est un espace temporaire contenant les modifications prêtes à être commitées
- Ces modifications peuvent être :
  - création de fichier
  - modification de fichier
  - suppression de fichier

Sommaire

Les commandes de base

Configurer l'environnement

- Définir le nom associé à tous les commit

```
git config --global user.name "Yoann Calamai"
```

- Définir l'email associé à tous les commit

```
git config --global user.email "Yoann.Calamai@viacesi.fr"
```

- Active la colorisation de la sortie en ligne de commande

```
git config --global color.ui auto
```

- Où sont situés les fichiers de configuration

```
git config --list --show-origin
```

Les commandes de base

Créer un dépôt

- Créer un dépôt local à partir du nom spécifié

```
git init monprojet
```

- Télécharger un projet et tout son historique

```
git clone https://github.com/github/training-kit
```

Les commandes de base

Consulter l'historique et l'état d'un dépôt

- Lister les commits

```
git log
```

- Voir l'état du répertoire de travail

```
git status
```

Les commandes de base

Effectuer une opération de commit

- Ajoute une modification à l'index

```
git add README.md
```

- Retire une modification de l'index

```
git checkout README.md
```

- Créer un commit à partir de l'index

```
git commit -m "Initial commit"
```

Corriger une erreur

- Annuler tous les commits après un commit, en conservant les modifications localement

```
git reset 223f8a7a6dc1251d6c1de7238691f8f86694a3ac
```

- Supprimer tout l'historique et les modifications effectuées après le commit spécifié

```
git reset --hard 223f8a7a6dc1251d6c1de7238691f8f86694a3ac
```

Travailler avec dépôt distant

- Récupérer tout l'historique du dépôt distant

```
git fetch
```

- Récupérer tout l'historique du dépôt distant et incorporer les modifications

```
git pull
```

- Ajouter une branche distante

```
git remote add origin https://github.com/github/training-kit
```

- Envoyer tous les commits de la branche locale vers une branche distante

```
git push -u origin master
```

Les commandes de base

Travailler avec des branches

- Liste toutes les branches

```
git branch
```

- Créer une branche

```
git branch nouvellefonctionalite
```

- Basculer sur une branche

```
git checkout nouvellefonctionalite
```

- Fusionner une branche avec la branche courante

```
git merge nouvellefonctionalite
```

Sommaire

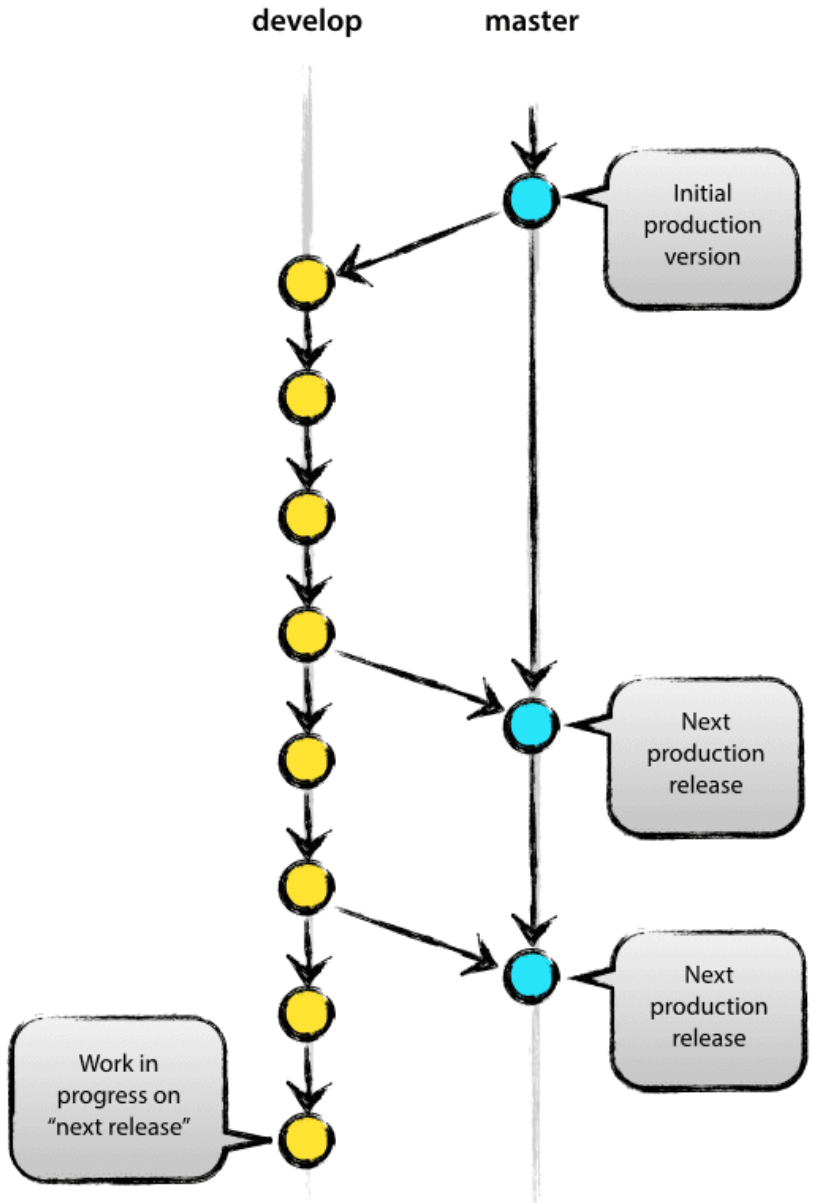
Travailler avec les branches

Pourquoi ?

- Gérer plusieurs versions dans un même dépôt
- Plusieurs standards ont émergé : git-flow, github-flow, gitlab-flow

Git-Flow : les branches long terme

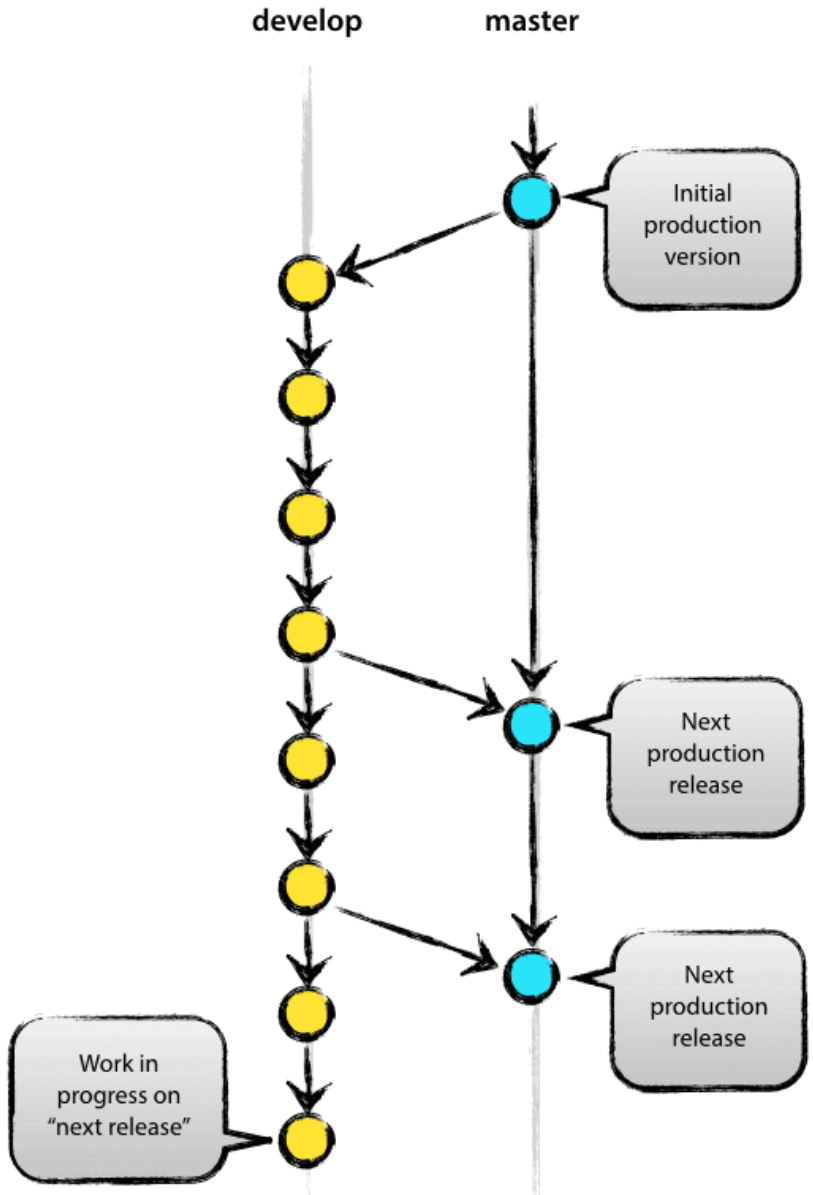
La branche *master* contient les releases de production



Git-Flow : les branches long terme

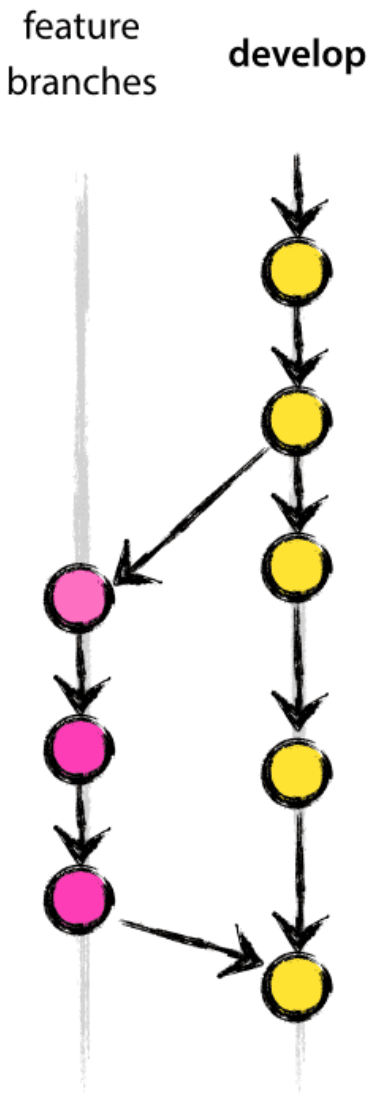
La branche *develop* contient les fonctionnalités stables pour la prochaine release

Pour les *night builds*



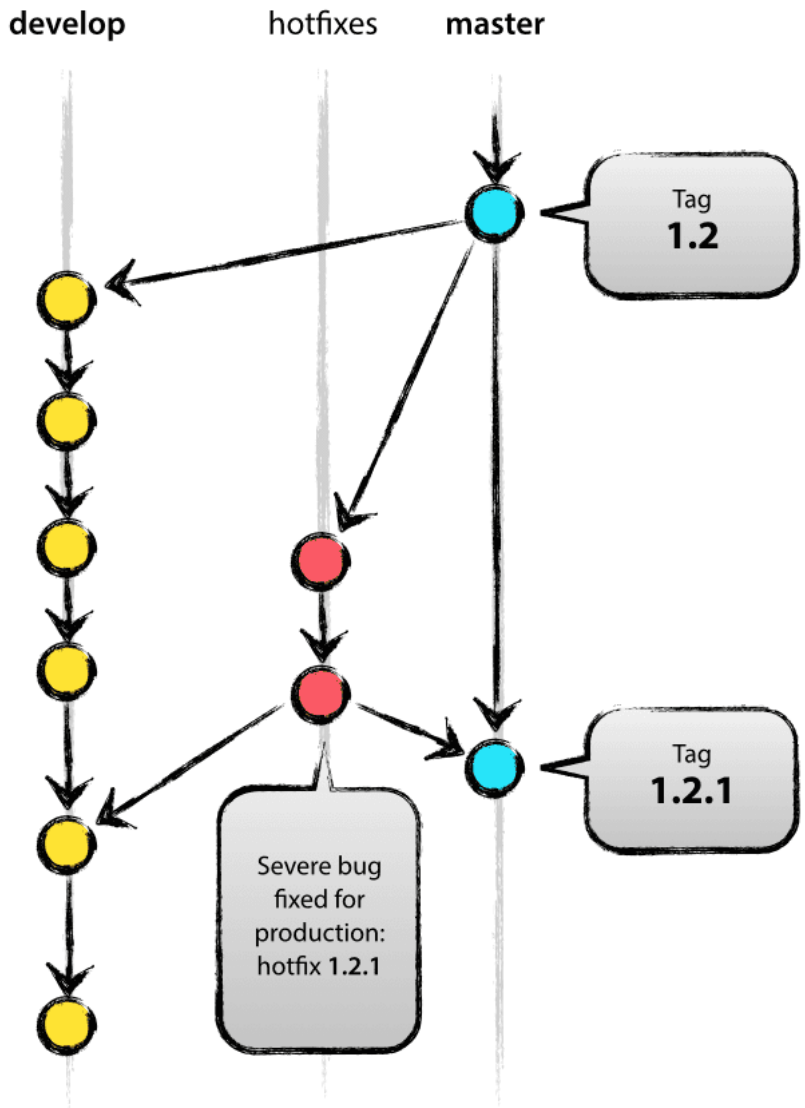
Git-Flow : les branches éphémères

Une branche est créée pour chaque fonctionnalité.



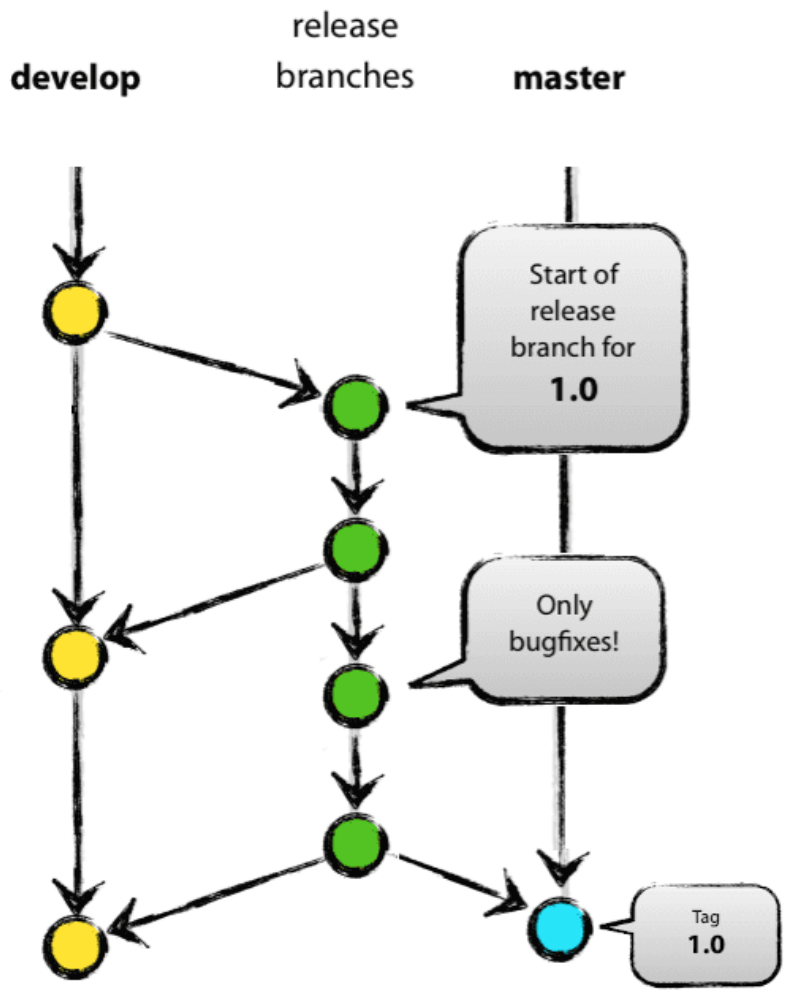
Git-Flow : les branches éphémères

Une branche est créée pour chaque hotfixes pour être fusionnée avec *master* et *develop*

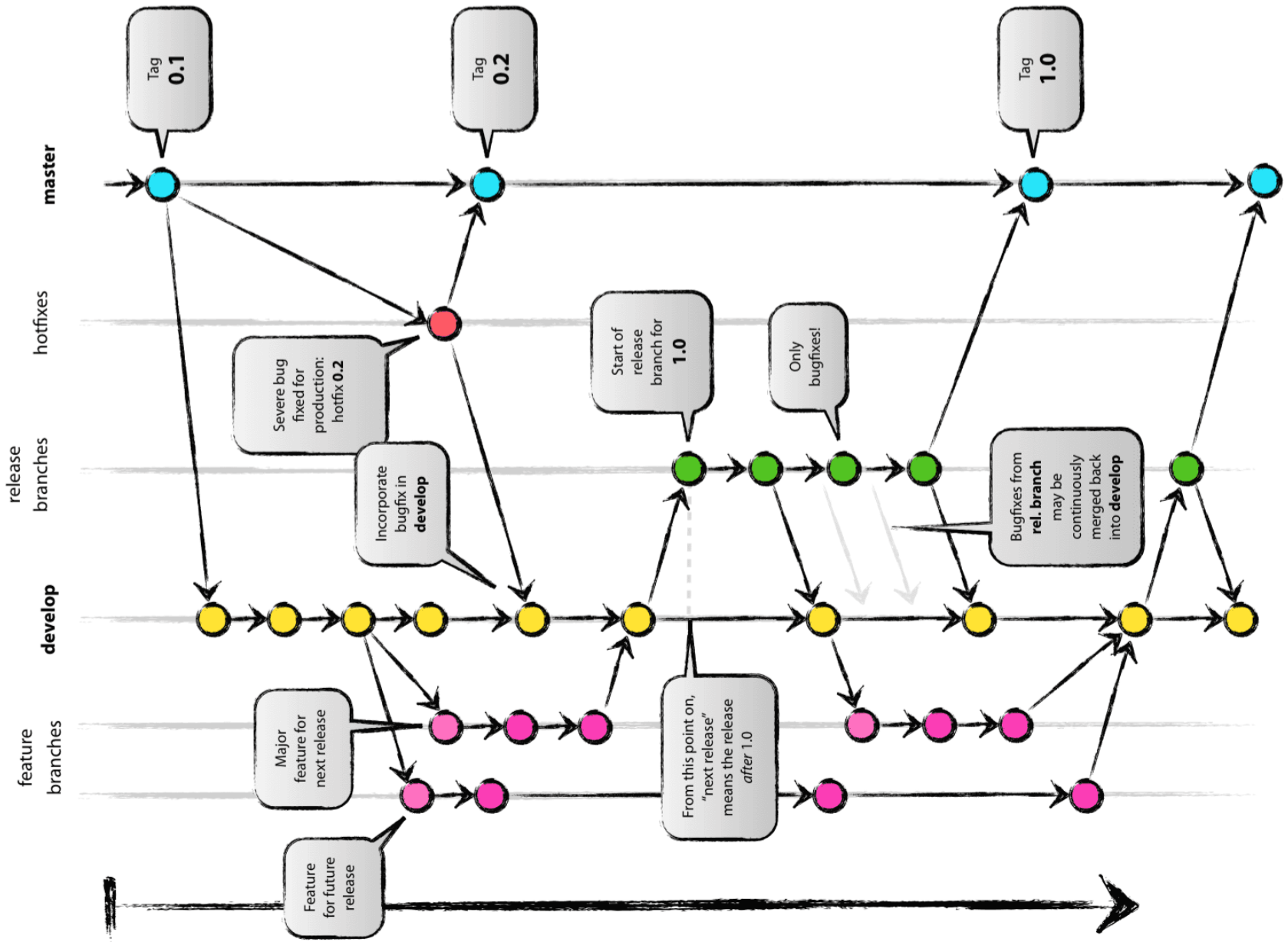


Git-Flow : les branches éphémères

Une branche est créée pour chaque release candidate pour être fusionnée avec *master* et *develop*



Git-Flow : Vue globale



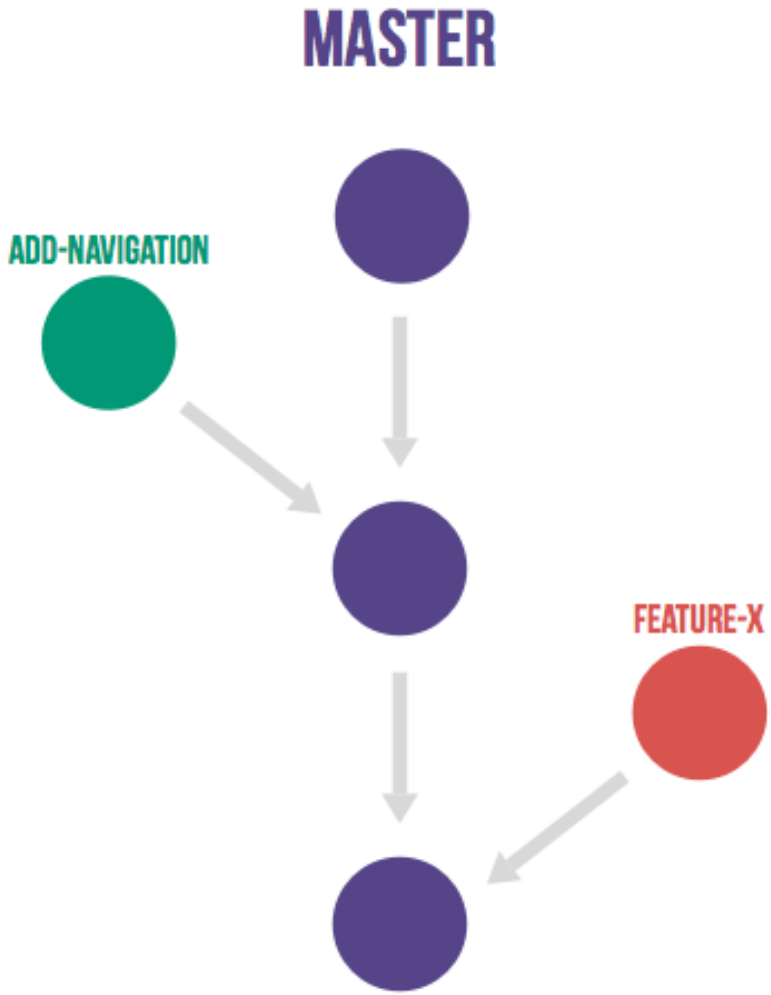
Git-Flow : La puissance complexe

Inconvenients

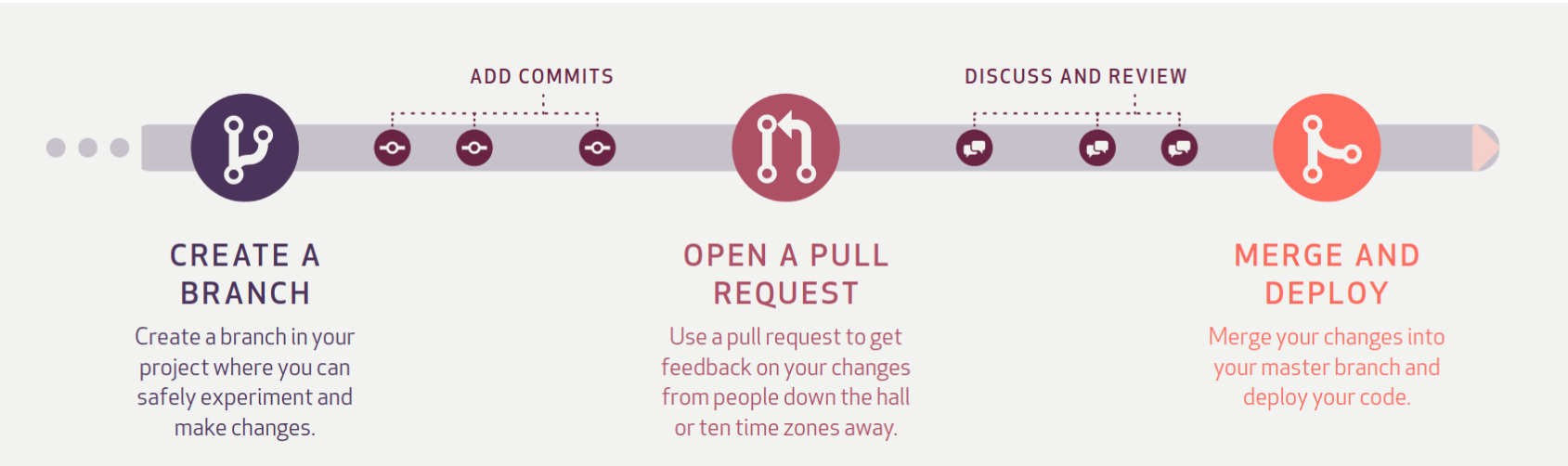
- La branche par défaut *main* est utilisée pour la production
- Complexité (hotfixes et release) == source d'erreur
- Beaucoup d'équipes ont un fonctionnement plus simple

GitHub-Flow : simple et efficace

GitHub a réduit le modèle précédent à une branche principale *main* et une branche par fonctionnalité.



GitHub-Flow : simple et efficace



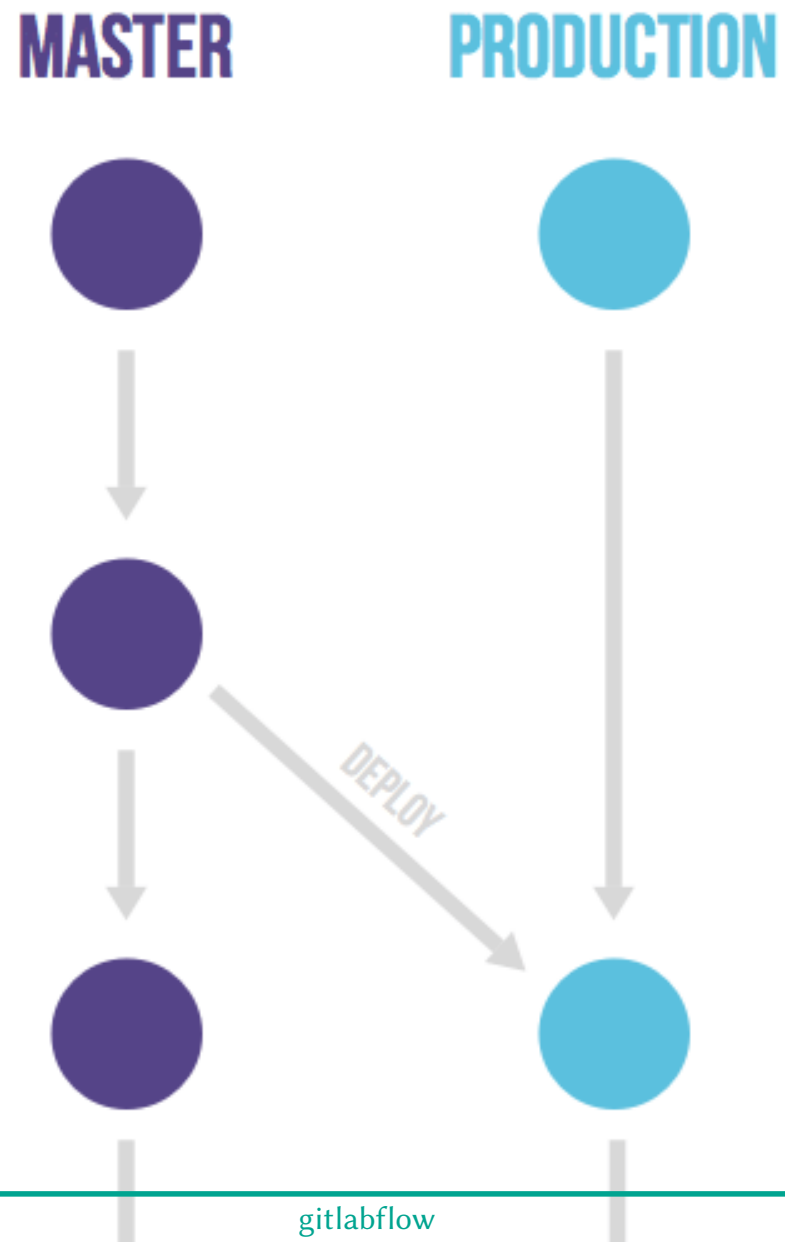
GitLab-Flow : workflow à choix multiple

Plusieurs modèles

- Production
- Environnement
- Release

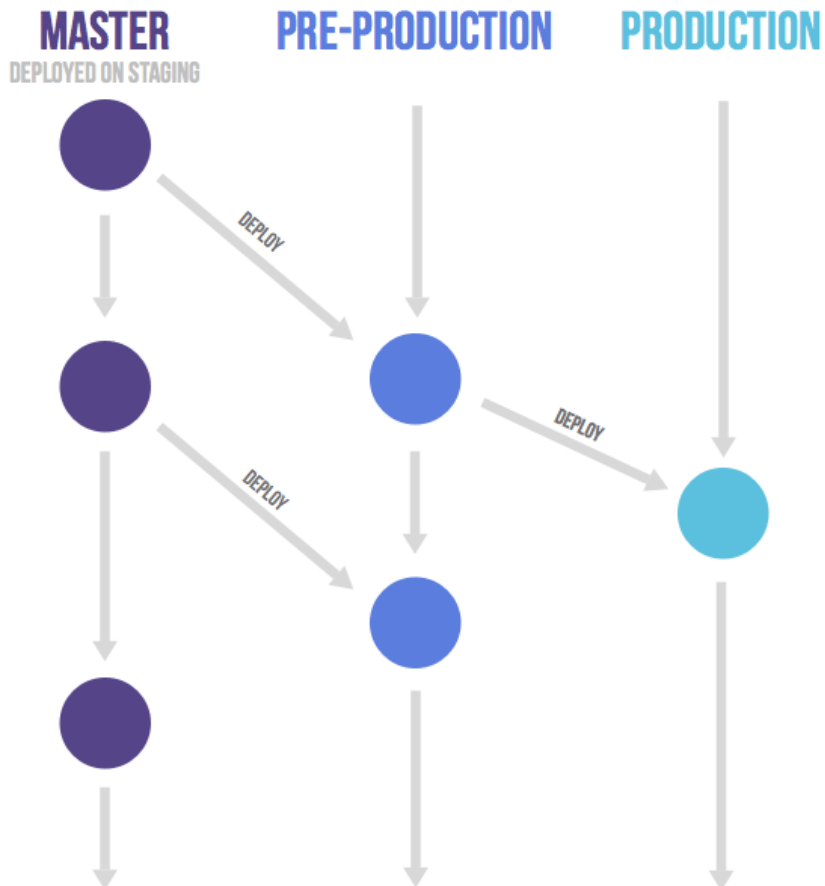
GitLab-Flow : Branche production

La branche *master* est utilisée pour le développement courant. La branche *production* contient les releases de production.



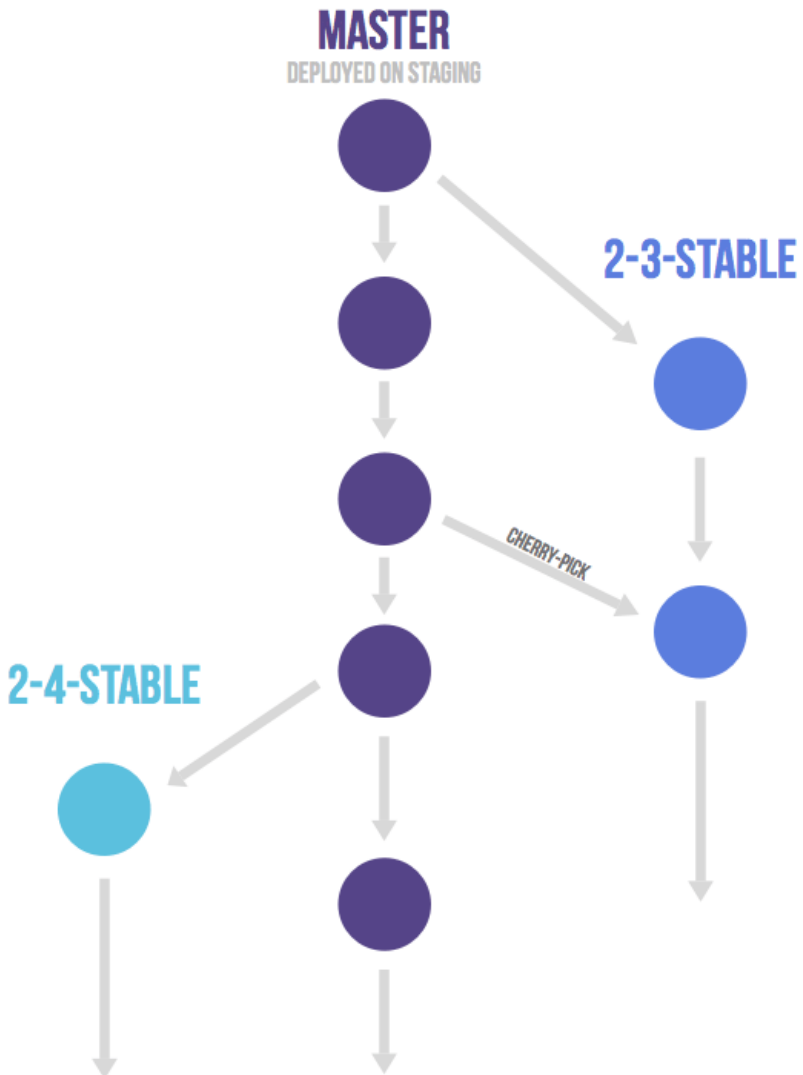
GitLab-Flow : Branche environnement

La branche *master* est utilisée pour le développement courant. Les branches *preproduction* et *production* sont utilisées pour déployer les versions automatiquement dans un environnement particulier.



GitLab-Flow : Branche release

La branche *master* est utilisée pour le développement courant. Une branche est créée par release pour maintenir différentes versions en parallèle.



Sommaire

Les alias

Le système d'alias permet de définir des raccourcis

Pour créer des alias il suffit d'éditer la section `[alias]` du fichier `~\.gitconfig`

Ou via la commande `git config`

- `git config --global alias.co checkout`
- `git config --global alias.br branch`
- `git config --global alias.ci commit`

Ignorer des fichiers

Pour ignorer des fichiers ou des répertoires automatiquement, git a un système interne basé sur le fichier `.gitignore`

```
1  #Ceci est un commentaire
2
3  #Ignorer un répertoire
4  logs
5  #Ignorer un ou plusieurs fichiers
6  *.log
7  file-debug.log*
8  #Ignorer tout sauf /src/frontapp
9  /*
10 !/src
11 /src/*
12 !/src/frontapp
```

Aide à la création de fichiers `.gitignore`

`https://www.toptal.com/developers/gitignore`

Executer des actions

Git permet d'exécuter des scripts avant ou après des commandes, grâce aux crochets (hooks).

Utilisation

- Validation de commit (`pre-commit`, `prepare-commit-msg`, ...)
- Mettre en place l'environnement (`post-checkout`, `post-merge`,...)
- Gérer un workflow (`post-checkout`, `post-merge`, `post-push`...)
- ...

git hooks

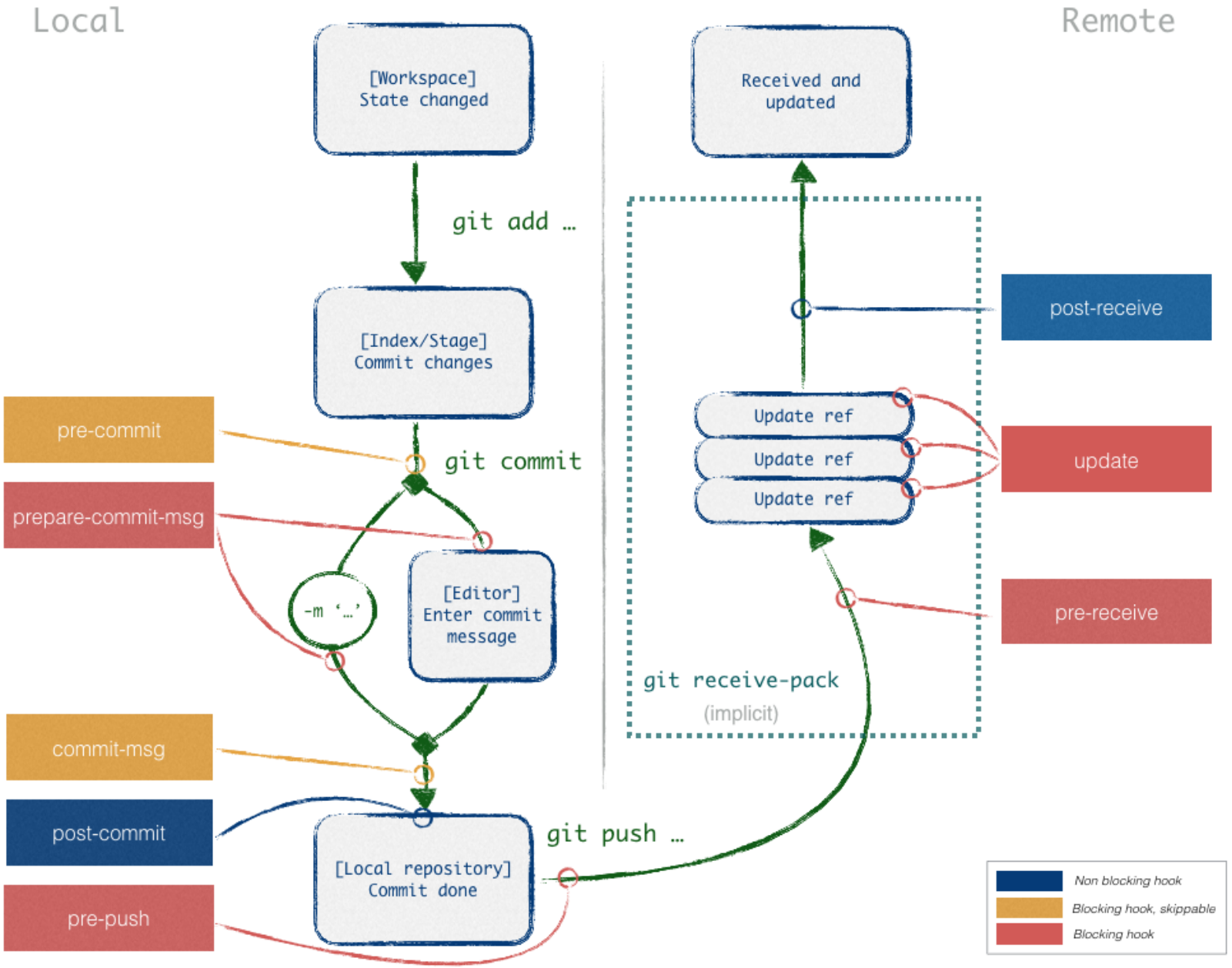
Pour mettre en place des crochets, il faut ajouter des fichiers dans `.git/hooks`.

Les fichiers sont nommés par le nom du crochet. Par exemple, un crochet `pre-commit`sera nommé `pre_commit`.
Dans chaque répertoire `.git/hooks`, il existe des exemples.

Changer l'emplacement des hooks

```
git config core.hookspath crocodile
```

git hooks


Local
Remote
[Workspace]
State changed
git add …
[Index/Stage]
Commit changes
pre-commit
git commit
prepare-commit-msg
-m '…'
[Editor]
Enter commit
message
commit-msg
post-commit
pre-push
[Local repository]
Commit done
git push …
Received and
updated
post-receive
Update ref
Update ref
Update ref
update
pre-receive
git receive-pack
(implicit)
Non blocking hook
Blocking hook, skippable
Blocking hook

Sommaire

Pour aller plus loin

Liens intéressants

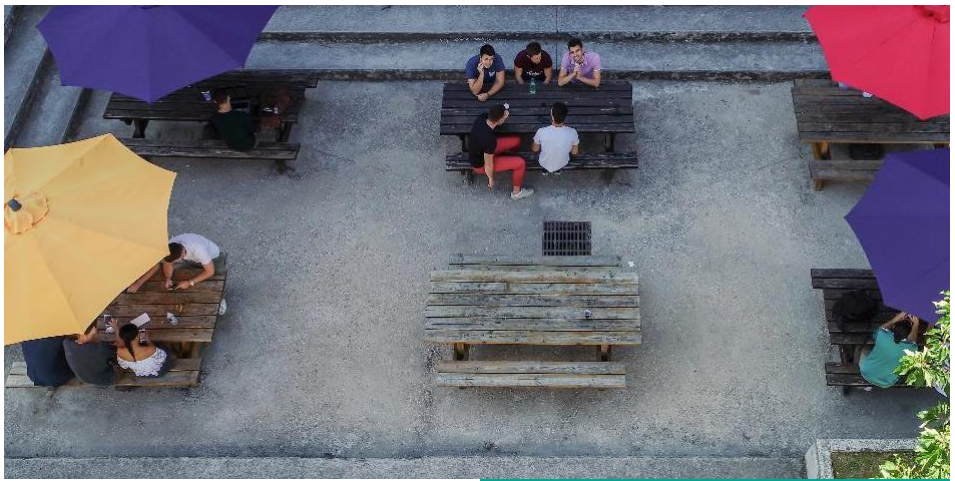
- https://borntocode.fr/git-alias-etre-un-bon-developpeur-faineant/
- https://github.com/GitAlias/gitalias
- http://blog.kfish.org/2010/04/git-lola.html
- https://about.gitlab.com/blog/2020/04/07/15-git-tips-improve-workflow/
- https://www.gitkraken.com/learn/git/best-practices/git-branch-strategy
- https://github.com/nvie/git-toolbelt
- https://github.com/github/training-kit
- https://learngitbranching.js.org/?locale=fr_FR

Pour aller plus loin

Liens intéressants

- https://www.analysisandsolutions.com/code/git-hooks-summary-cheat-sheet.htm
- http://ryanflorence.com/deploying-websites-with-a-tiny-git-hook/
- https://codeinthehole.com/tips/tips-for-using-a-git-pre-commit-hook/
- https://longair.net/blog/2011/04/09/missing-git-hooks-documentation/
- https://delicious-insights.com/fr/articles/git-hooks/
- https://www.mixmum.fr/blog/git-rebase/

HESAM
UNIVERSITÉ



cesi

CAMPUS
D'ENSEIGNEMENT SUPÉRIEUR
ET DE FORMATION PROFESSIONNELLE