



Gitlab et Gitlab CI

Sommaire

I - De Git à Gitlab : la mise en place d'un workflow

- 0 Rappels sur git
- 1 Serveur gitlab et installation
- 2 Gestion des dépôts git
- 3 Gestion des droits d'accès
- 4 Workflows et organisation
- 5 Les Merge Requests
- 6 Labels, issues board, snippets



Sommaire

II - Intégration continue et déploiement continu avec Gitlab

- 1 Infrastructure Gitlab
- 2 Définition d'un pipeline, des stages et des jobs
- 3 Définition du runner, éléments de configuration, spécificités liées à Docker
- 4 Structure du fichier .gitlab-ci.yml : éléments clés, bonnes pratiques et sécurité
- 5 Déploiement continu : gestion des environnements



DE GIT A GITLAB :

LA MISE EN PLACE D'UN WORKFLOW



0 Rappels sur git

VCS décentralisé : git



disponibilité

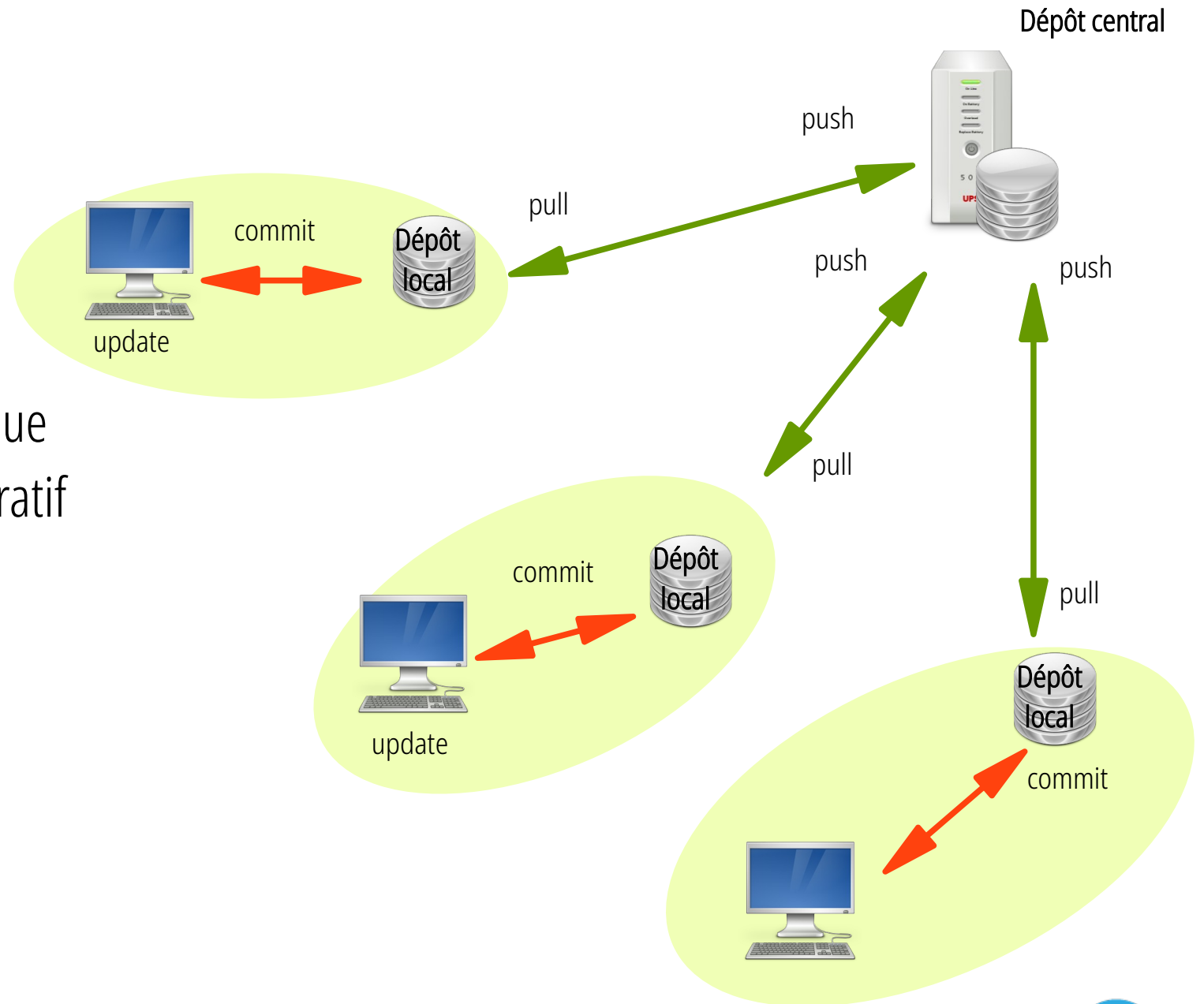
redondance

gestion de l'espace disque

développement collaboratif



complexité



Enregistrement des modifications

◆ Utilisation d'objets : propriétés communes

référéncés par une somme de contrôle (SHA1) : garantir l'unicité et l'intégrité du contenu

immuables

pas de suppression d'objets



Objets git

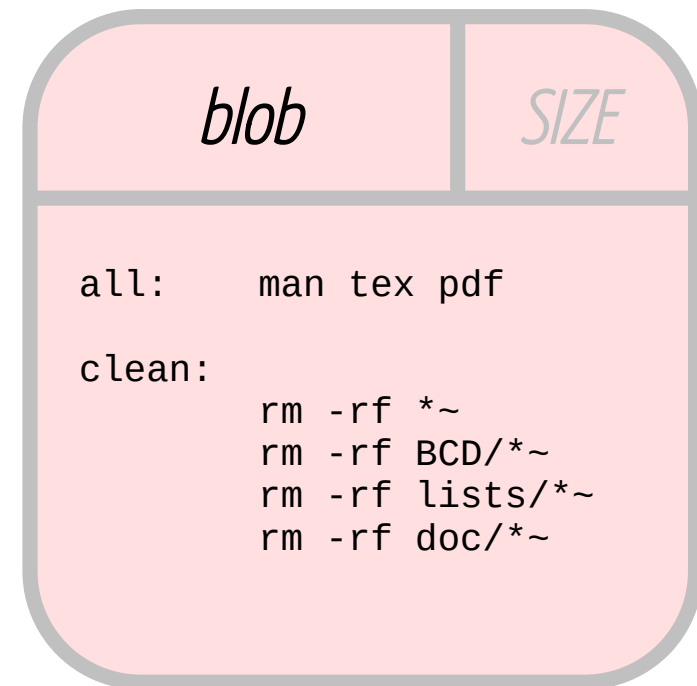
Blobs

Stocke le contenu d'un fichier

Référencé par un SHA1

2 fichiers avec le même contenu dans un dépôt pointent sur le même blob

5e78f...



Objets git

Tree

Liste des pointeurs vers les blobs (fichiers) et/ou les trees (répertoires), leurs noms, leurs ID, leurs permissions

Référencé par un SHA1, recréé lors du moindre changement d'un des composants de l'arbre

12af7...

<i>tree</i>		<i>SIZE</i>
<i>blob</i>	5e78f	Makefile
<i>blob</i>	2a89b	README
<i>tree</i>	cd20c	src
<i>blob</i>	36ca9	config



Objets git

◆ Commits

Référence de l'état du dépôt à un moment donné

Commit parent (état précédent)

Message décrivant la modification apportée

Horodatage

Identification

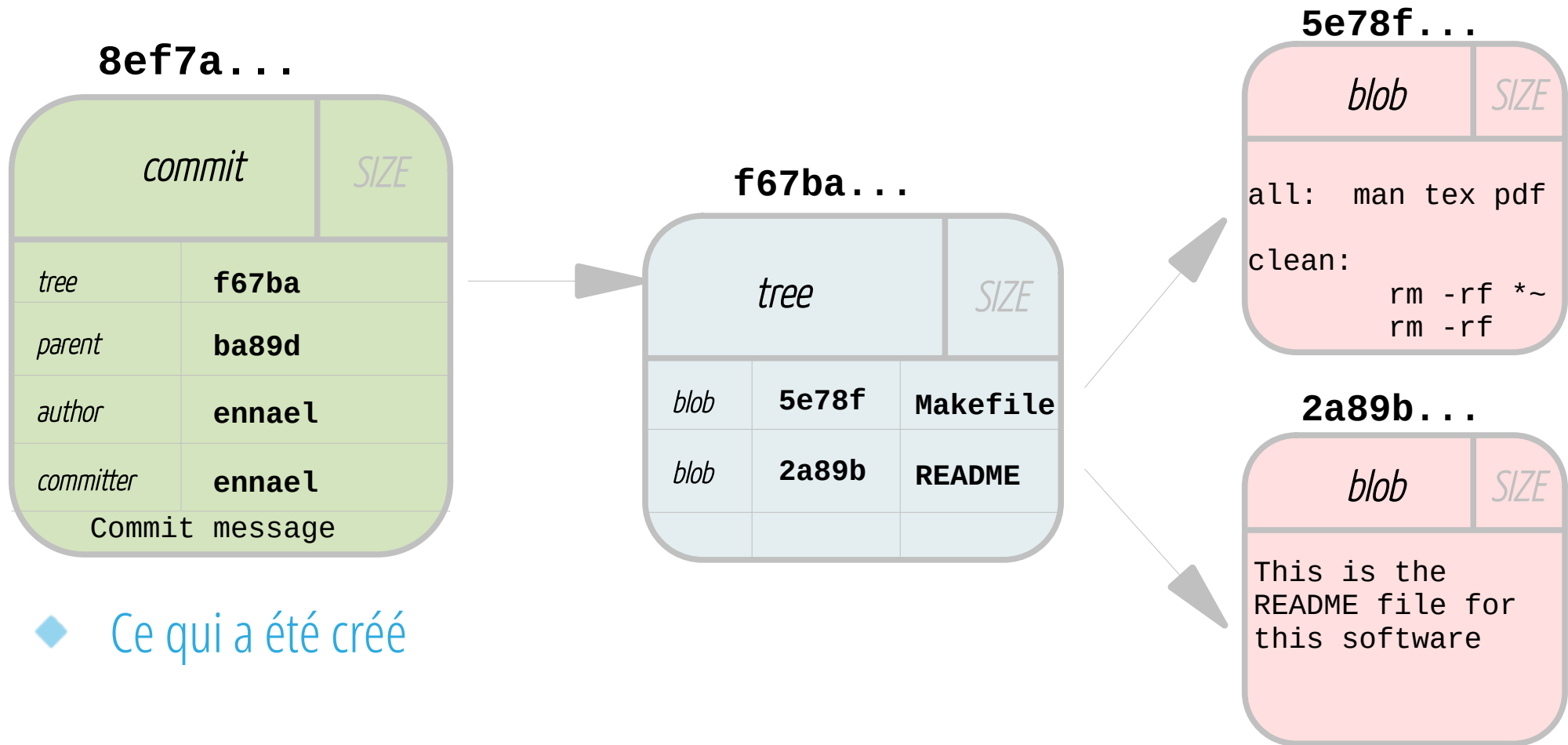
Référencé par un SHA1

ef667...

<i>commit</i>		<i>SIZE</i>
<i>tree</i>	g67ha	
<i>parent</i>	ab89d	
<i>author</i>	ennae1	
<i>committer</i>	ennae1	
The content of my commit message		



Construction des commits



◆ Ce qui a été créé

Un objet tree pour chaque répertoire

Un objet blob pour chaque fichier modifié ou ajouté

Un objet commit pour pointer le nouveau tree racine du dépôt



Espaces de travail

Les zones locales

◆ Dépôt local

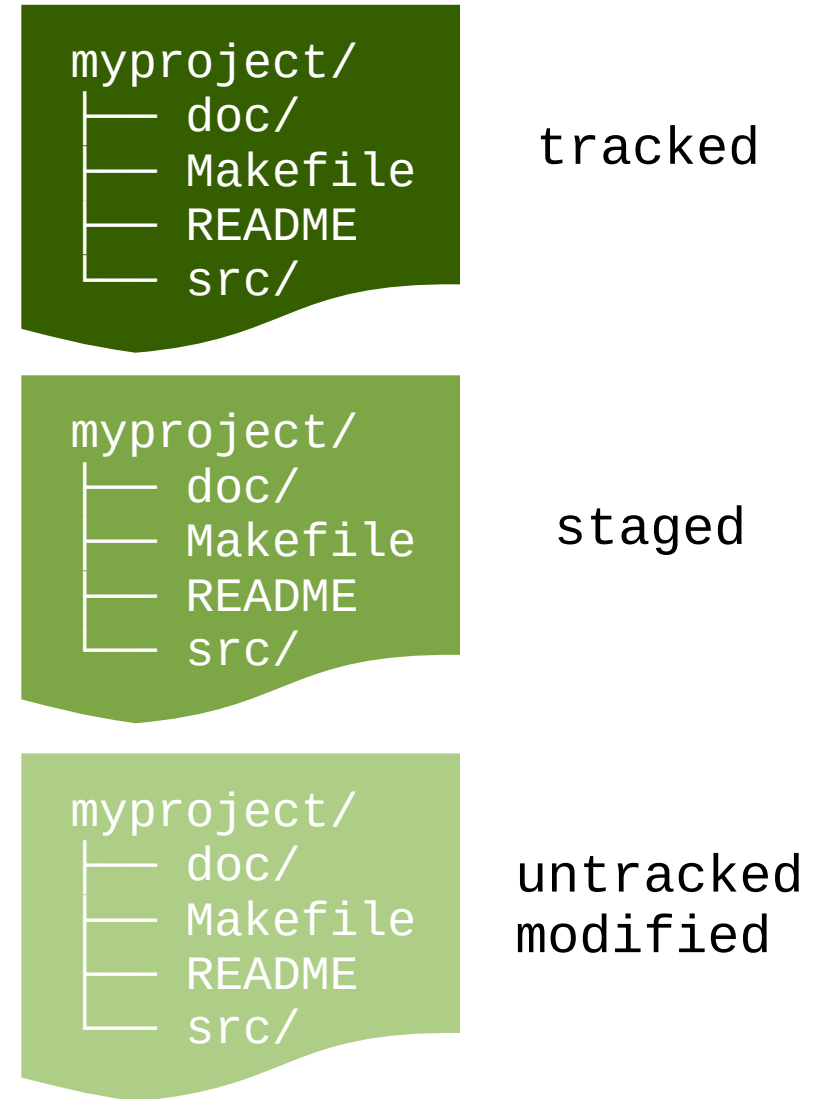
Appelé aussi base de données ou arbre git
Stocke les data et metadata jusqu'au dernier commit

◆ Zone de cache

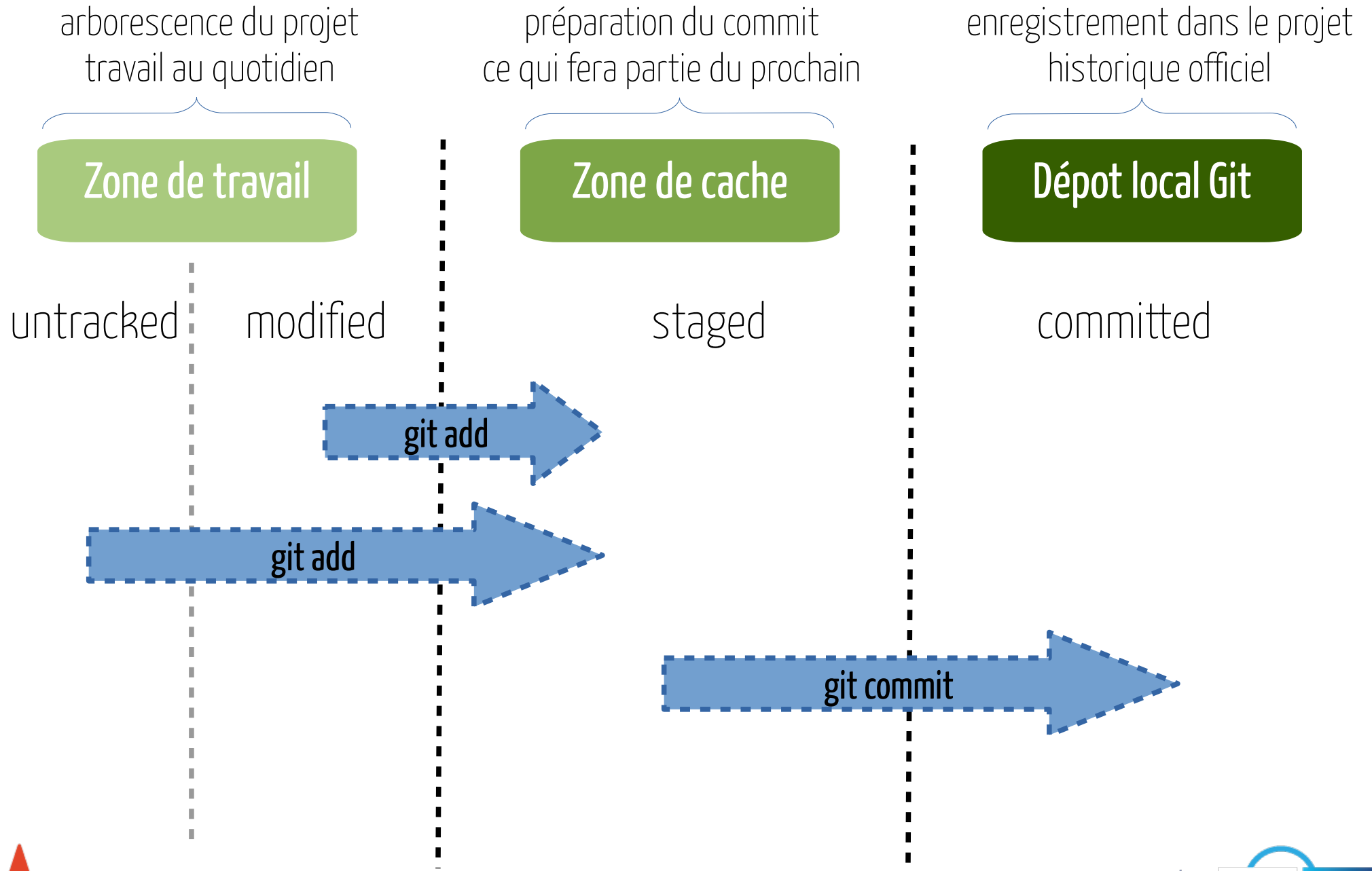
Appelée aussi index
Stocke un snapshot des modifications

◆ Répertoire de travail

Ensemble des fichiers et répertoires du projet



Le processus de commit



Ajouter à la zone de cache

◆ Indexation des modifications

```
git add <fic1>...<ficn>  
git add <répertoire>
```

ajouter de nouveaux fichiers non encore trackés

ajouter des fichiers trackés et modifiés

supprimer des fichiers trackés

Attention : Impossible d'ajouter un répertoire vide
créer un fichier vide, par convention .gitkeep

◆ Bonnes pratiques : à éviter d'utiliser systématiquement

```
git add *  
git add .
```



Finaliser le commit

◆ Commit des modifications

```
git commit [-m <message>]
```

utilise le contenu de la zone de cache pour réaliser le commit
message multilignes en ligne de commande ou en utilisant l'éditeur par défaut

◆ Bonnes pratiques : à éviter d'utiliser systématiquement

```
git commit -a
```



Optimiser son environnement

- ◆ Ignorer des fichiers globalement ou pour un dépôt

1 ou n fichiers à la racine du dépôt et/ou n'importe où dans le dépôt

```
[ennael@localhost test (master)]$ cat <depot>/.gitignore
*.[oa]      # ignore tous les fichiers du dépôt finissant par o ou a
/rep1/      # ignore le répertoire rep1 à la racine du dépôt
rep2/       # ignore tous les répertoires rep2 du dépôt
!rep3/      # n'ignore pas les répertoires rep3 du dépôt
**/rep4     # ignore tous les répertoires rep' du dépôt
```

- ◆ Templates de fichiers en fonction du type de projet

<https://github.com/github/gitignore>



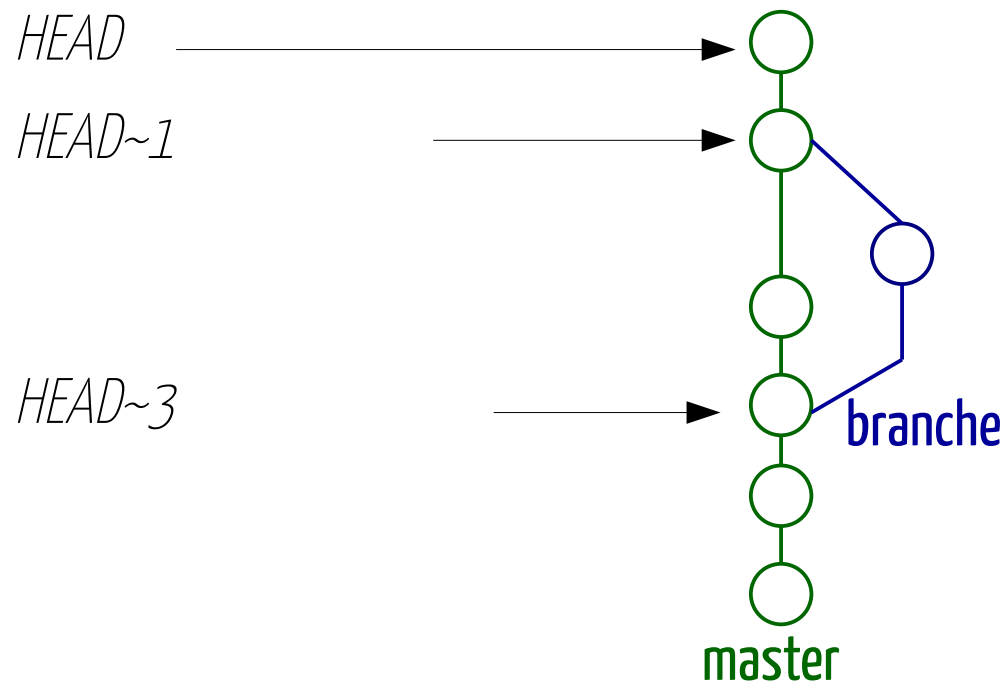
Les révisions git

◆ HEAD

référence vers le dernier commit de la branche courante

◆ HEAD~x ou SHA1~x

désigne le xième commit avant HEAD, sans tenir compte des opérations de merge



Etat général du dépôt

◆ Connaître son status

```
git status [-s] [-b branche]
```

état de la zone de travail

état du cache

conflit

solutions

```
# Modifications qui seront validées :  
#   (utilisez "git reset HEAD <fichier>..." pour désindexer)  
#  
#      modifié :   fic1  
# Fichiers non suivis:  
#   (utilisez "git add <fichier>..." pour inclure dans ce qui  
# sera validé)  
#  
#      essai
```



Recueillir de l'information

◆ La commande git log

affichage synthétique

```
git log --oneline
```

filtrer l'historique des modifications concernant un fichier

```
git log -oneline -- <fichier>
```

afficher les fichiers modifiés par commit

```
git log --oneline --name-status
```

filtrer l'historique des commits en fonction de l'action réalisée

```
git log --oneline --diff-filter=<filter>
```

filtrer des commits en fonction du message de commit

```
git log --oneline --grep="<chaine>"
```

afficher l'historique des modifications sur une fonction/méthode donnée

```
git log --oneline -L :<fonction>:<fichier_contenant_la_fonction>
```



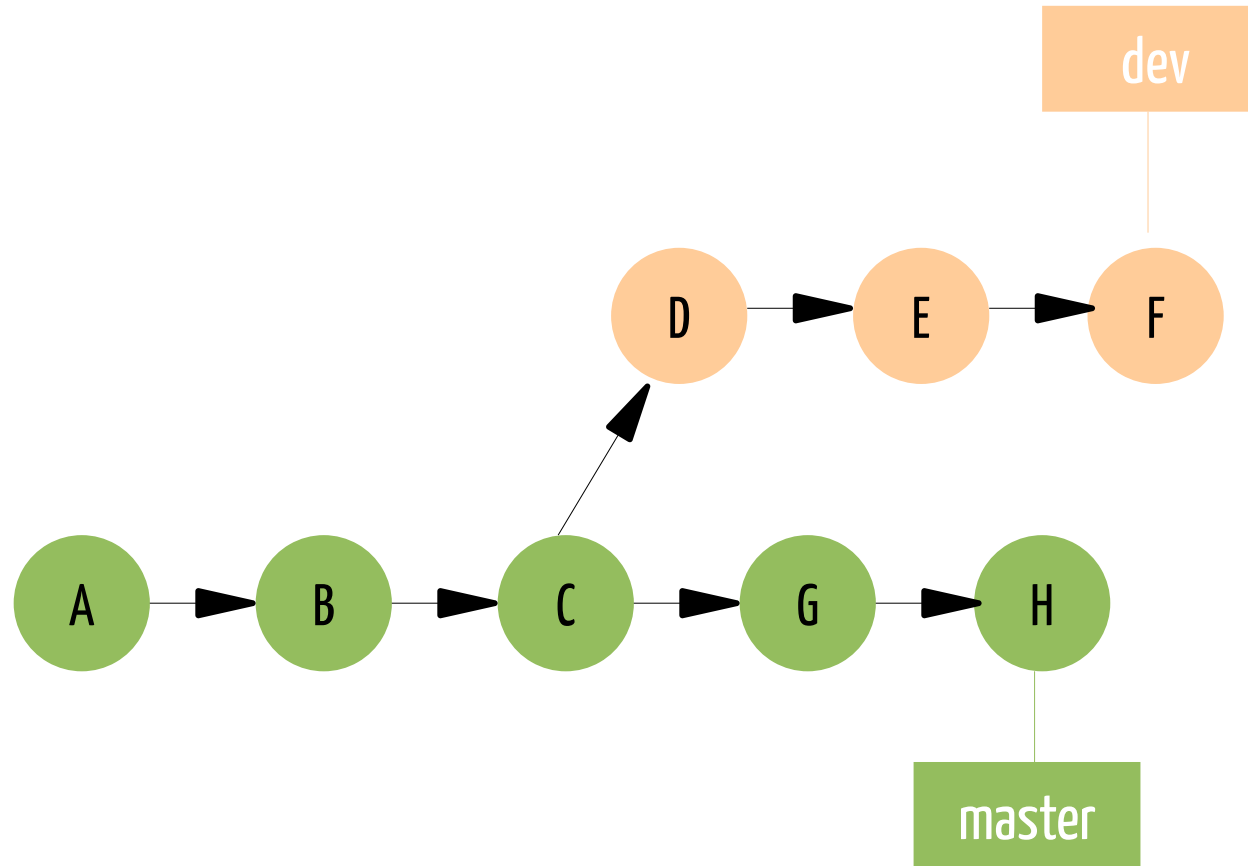
Recueillir de l'information

- ◆ La commande `git log`
affichage des branches

```
[ennael@localhost BCD (master)]$ git log --graph --all --oneline
*   cdd63ee Merge branch 'test'
|  \
| * a1a981d fix typo
| * | 9d0e767 fix path
| * | 215095c create a new variable for tests
| * | 83173a7 Merge branch 'test'
|  \  \
|   |  \
|   |   \
| * c0d17bc add new variable for test
| * | 6e20954 add new variable for new tests
|  /
| /
* fc34af2 remove unneeded list
* ac20415 add comment on last modification (user test)
```



Gestion des branches



Une branche est une référence au commit HEAD, contenu dans un fichier
`.git/refs/heads/master`

HEAD est une référence au dernier commit réalisé



Collecter de l'information

- ◆ Lister les branches locales

```
git branch [-v]
```

- ◆ Afficher l'historique d'une branche

```
git log --oneline <branche>
```

- ◆ Comparer des branches

Lister les commits spécifiques à brancheB comparée à brancheA

```
git log --oneline <brancheA>..<brancheB>
```



Commandes de base : gestion branches

◆ Créer une branche locale

```
git branch <branche> [SHA1]
```

◆ Changer de branche

```
git checkout <branche> [SHA1]
```

◆ Lister les branches locales

```
git branch [-v]
```

◆ Fusionner des branches

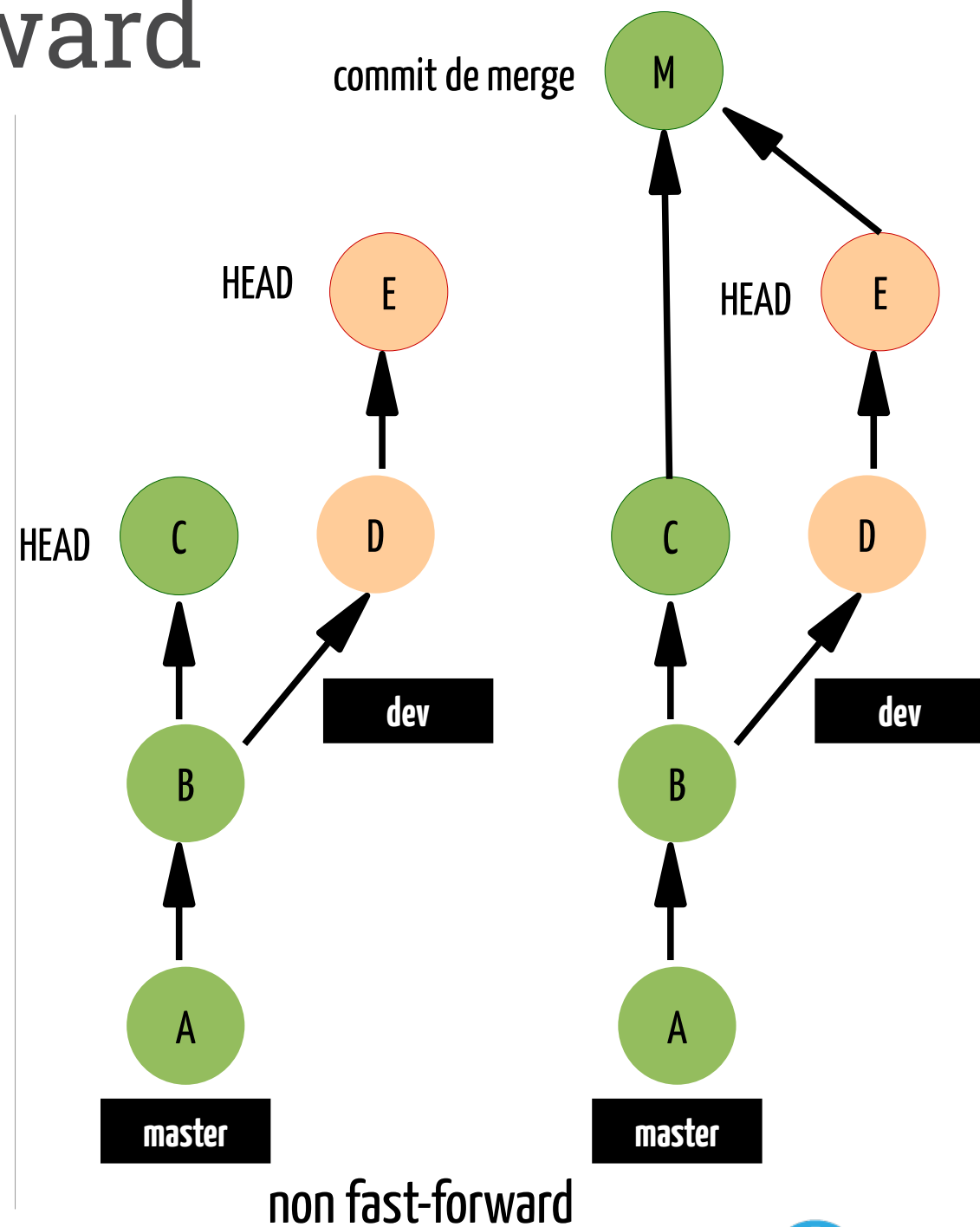
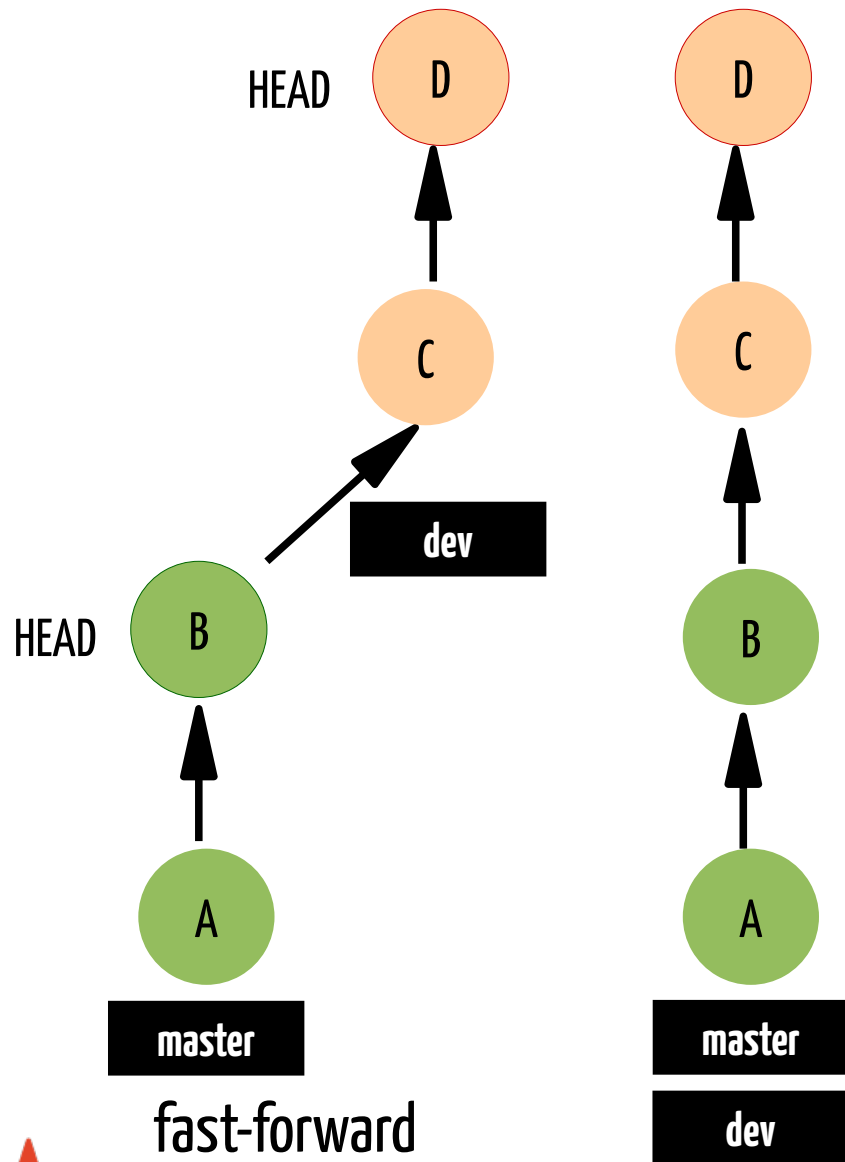
```
git merge <branche>
```

◆ Supprimer une branche locale

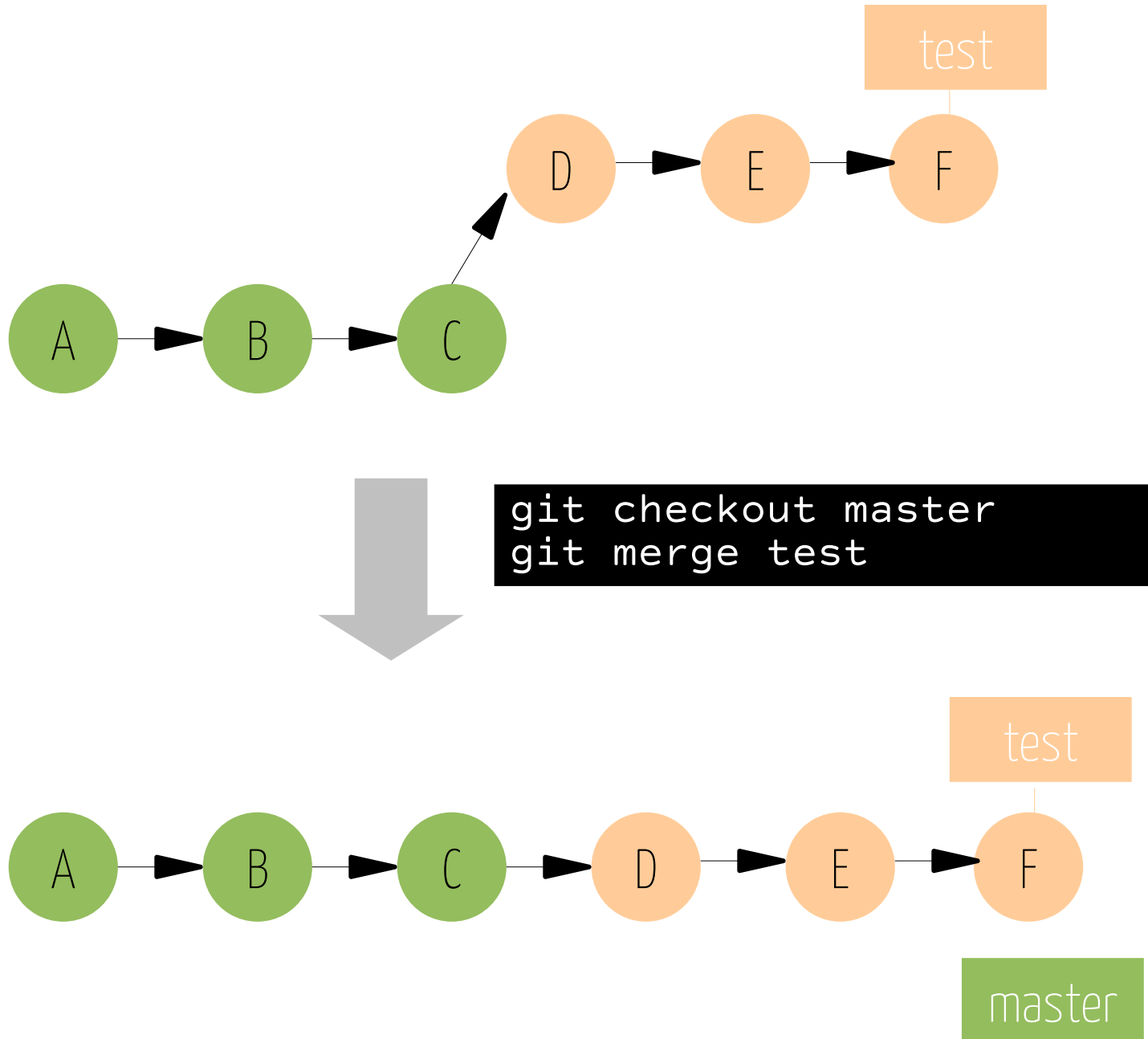
```
git branch -d <branche>
```



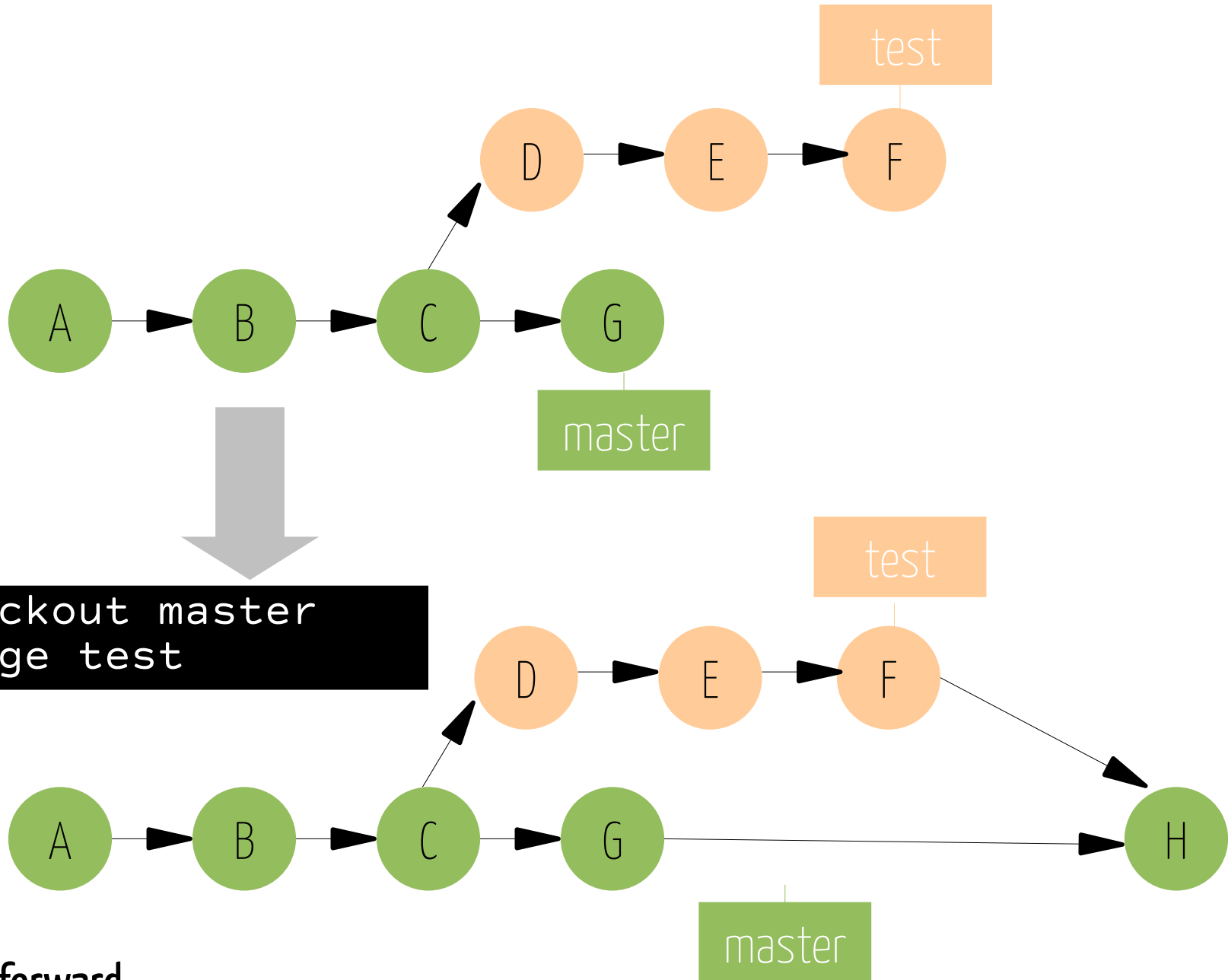
Branches fast-forward



Fusion de branches fast-forward



Fusion de branches non fast-forward



fast-forward



Résolution manuelle d'un conflit

- ◆ Résoudre le conflit et partager

 - modifier le fichier pour résoudre le conflit

 - ajouter le fichier à l'index

 - commiter le merge

- ◆ Annuler un merge en cours

```
git merge --abort
```



Fusion et bonnes pratiques

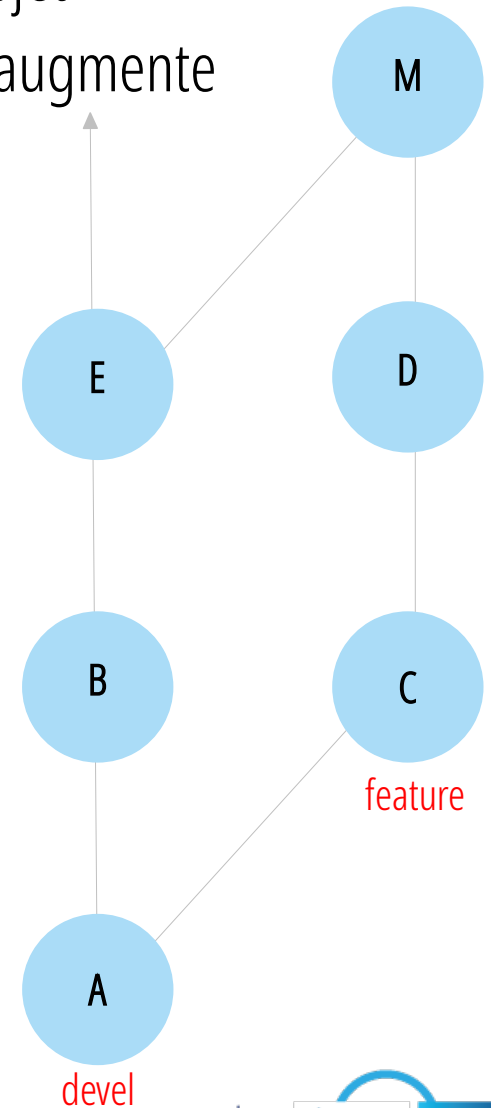
- ◆ Une branche de travail...

... ne doit pas représenter un espace de travail désynchronisé du projet

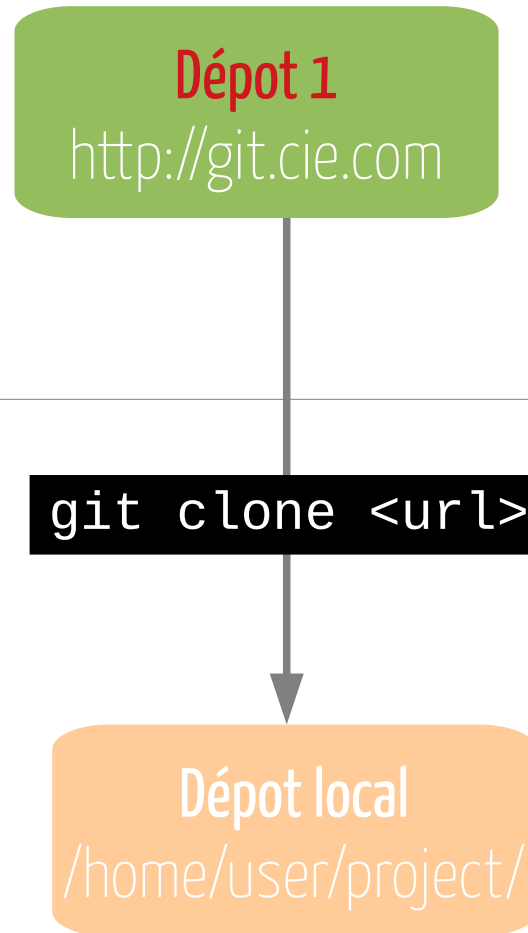
Plus l'écart entre 2 branches se creuse, plus la probabilité de conflit augmente

- ◆ Une bonne pratique pour faciliter le merge final de la branche

Maintenir une branche de travail synchronisée



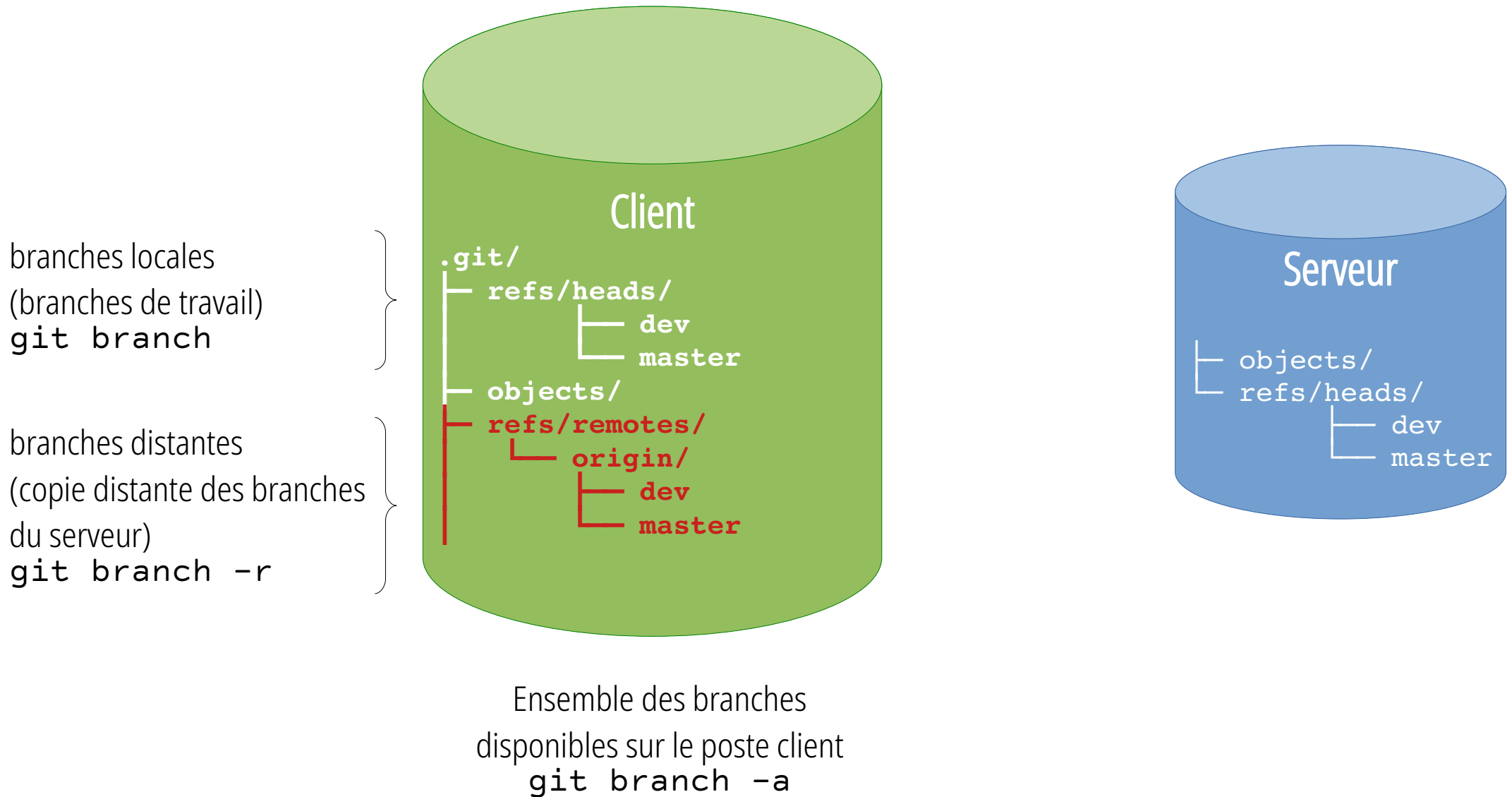
Gérer le dépôt git distant



Le premier dépôt est toujours appelé « **origin** »



Branches locales et distantes



Partager ses branches

- ◆ Récupérer une branche distante créée sur le serveur

```
git fetch  
git checkout <branche>
```

- ◆ Partager une branche créée localement

```
git branch <branche>  
git push -u origin <branche>
```

-u : met en place le tracking de branches

établit un lien entre la branche locale et la branche du serveur

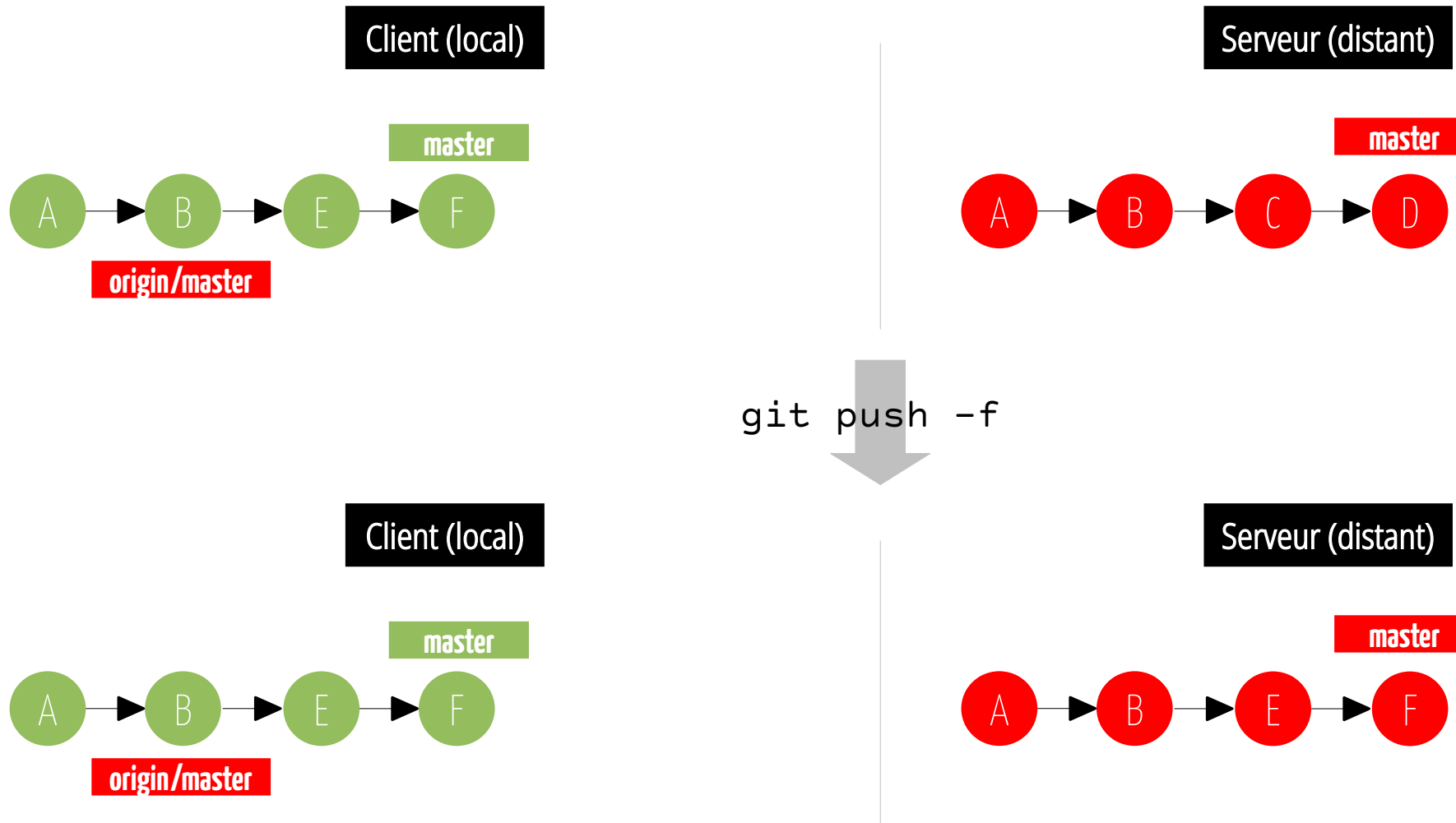
alimente automatiquement le status de la branche (avance, retard, synchro)

permet un push de la branche sans argument à préciser



Prérequis pour l'envoi de commits

- ◆ Les branches doivent être fast-forward : le push n'est pas un véritable merge
- ◆ Push et branches non fast-forward



Mise à jour

- ◆ `git pull` = `git fetch` + `git merge`
- ◆ Mettre à jour le suivi des branches distantes sans rien modifier dans les branches locales ni la zone de travail locale
- ◆ Passer en revue les modifications disponibles sur le serveur distant

```
[ennael@localhost sandgit (master)]$ git fetch
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From ssh://git.hupstream.com/git/users/sandgit
* [new branch]      test1      -> origin/test1
```



Utilisation des tags (références)

◆ Pourquoi utiliser des tags

Identifier un commit spécifique en utilisant une étiquette au lieu d'un hash

Permet de mettre en évidence un commit important dans l'historique – souvent utilisé pour mettre en avant une release

Une étiquette qui pointe un commit

◆ Le tag est une référence

Peut être supprimé sans impact sur l'historique des commits

Peut être créé quel que soit le moment sur n'importe quel commit

◆ Génération d'un tag

2 types de tags: lightweight (fichier plat), annotated (objet)



Tagger un commit

◆ Lister les tags

```
git tag
```

◆ Créer un tag annoté

```
git tag -a <tag> [<SHA1>] -m "message"
```

◆ Supprimer un tag local

```
git tag -d <tag>
```

◆ Trouver de l'information sur un tag

```
git show v1.0
tag v1.0
Tagger: Anne Nicolas <anicolas@hupstream.com>
Date:   Sun Jan 5 21:53:32 2014 +0100

my message to explain this tag

commit 9e0b67c59a1a20c1193d4eec1acd1a4e9b8506c1
Author: Anne Nicolas <anicolas@hupstream.com>
Date:   Sun Jan 5 01:23:57 2014 +0100
```

```
update plan
```



Utiliser les tags : bonnes pratiques

- ◆ Utiliser des chaînes compréhensibles et normalisées
- ◆ Gestion du versioning : Convention utilisée

Tags réservés aux responsables de projet

```
v[major].[minor].[patch]
```

Possibilité de créer des tags (hors tags de version)



Commandes complémentaires

- ◆ Rejouer un ou plusieurs commits existant

```
git cherry-pick [SHA1...]
```

- ◆ Annuler un ou plusieurs commits

```
git revert [SHA1...]
```



Quand committer ?

◆ commit often

commit atomique

simplifier les revues de code, la recherche de bug, l'utilisation du diff, le revert

◆ perfect later

outils de réécriture d'historique : reset, rebase, rebase interactif

clarifier, simplifier, rendre un historique plus cohérent avant partage

◆ publish once

éviter de réécrire un historique déjà partagé

rendre public un historique



Le message de commit

- ◆ Représentation du commit dans l'historique : 1 commit = 1 ligne
la forme la plus couramment utilisée quelque soit l'outil utilisé

```
19eb739a20 ARM: bitmain: Enable saving variables to SD card
060fc5c8a0 ARM: bitmain: Enable legacy u-boot format
9dcf2e5c0d ARM: bitmain: Enable nand and smcc drivers
d999a7b7b6 spi: xilinx_spi: Trivial fixes in axi qspi driver
ce90654d1c xilinx: Sync DTs with Linux kernel
dd4c642757 clk: zynqmp: Fix clk dump values
```

SHA1
abbrégé

première ligne du message de commit
(header)

i2c: busses: i2c-stm32f4: Remove incorrectly placed ' ' from function...



lag-linaro authored and Wolfram Sang committed 5 days ago



a00cb25

i2c: busses: i2c-st: Fix copy/paste function misnaming issues



lag-linaro authored and Wolfram Sang committed 5 days ago



721a6fe

i2c: busses: i2c-pnx: Provide descriptions for 'alg_data' data structure



lag-linaro authored and Wolfram Sang committed 5 days ago



3e0f867

i2c: busses: i2c-ocores: Place the expected function names into the d...



lag-linaro authored and Wolfram Sang committed 5 days ago



d4c73d4



Le message de commit

`clk: zynqmp: Fix clk dump values`

With "clk dump" command, few clocks are showing up incorrect values and some clocks are displayed as "unknown".

Add missing clocks to zynqmp clock driver to display proper clocks rates.

Implement a simple way to get clock source, instead of calling functions. Change existing functions to this simple mechanism.

Fix gem clock name "gem_rx" to "gem_tx" which was incorrect. Change dbf_fpd & dbf_lpd clk names to dbg_fpd & dbg_lpd.

◆ Structure du message de commit : le header

header : 50 caractères - synthèse en une ligne

évite les messages tronqués, quel que soit l'outil utilisé



Here is my very long long long long subject commit that will be cut off using... ...

Anne authored just now



b18d62d9



Le message de commit

◆ Structure du message de commit : les paragraphes

laisser une ligne entre le header et le reste du message
délimiter le header du message, plus de lisibilité dans le log complet
limiter les lignes à 72 caractères (cf éditeurs intelligents)

◆ Formattage du header

format utilisé dans tous les messages proposés par défaut par l'outil (merge, revert...)
commence par une majuscule
pas de point final
emploi de l'impératif
utilisation de paragraphes pour structurer le message



Le message de commit

◆ Structure du message de commit : le contenu du message

expliquer quoi, pourquoi mais pas comment

éviter de surcharger le message avec des informations récupérables autrement

```
arm64: zynqmp: Update device tree properties for nand flash
```

```
Update the following device tree properties for nand flash
```

- Set software ecc mode.
- Set bch as ecc algo.
- Set read block to 0.

◆ Structure du message de commit : commits spécifiques

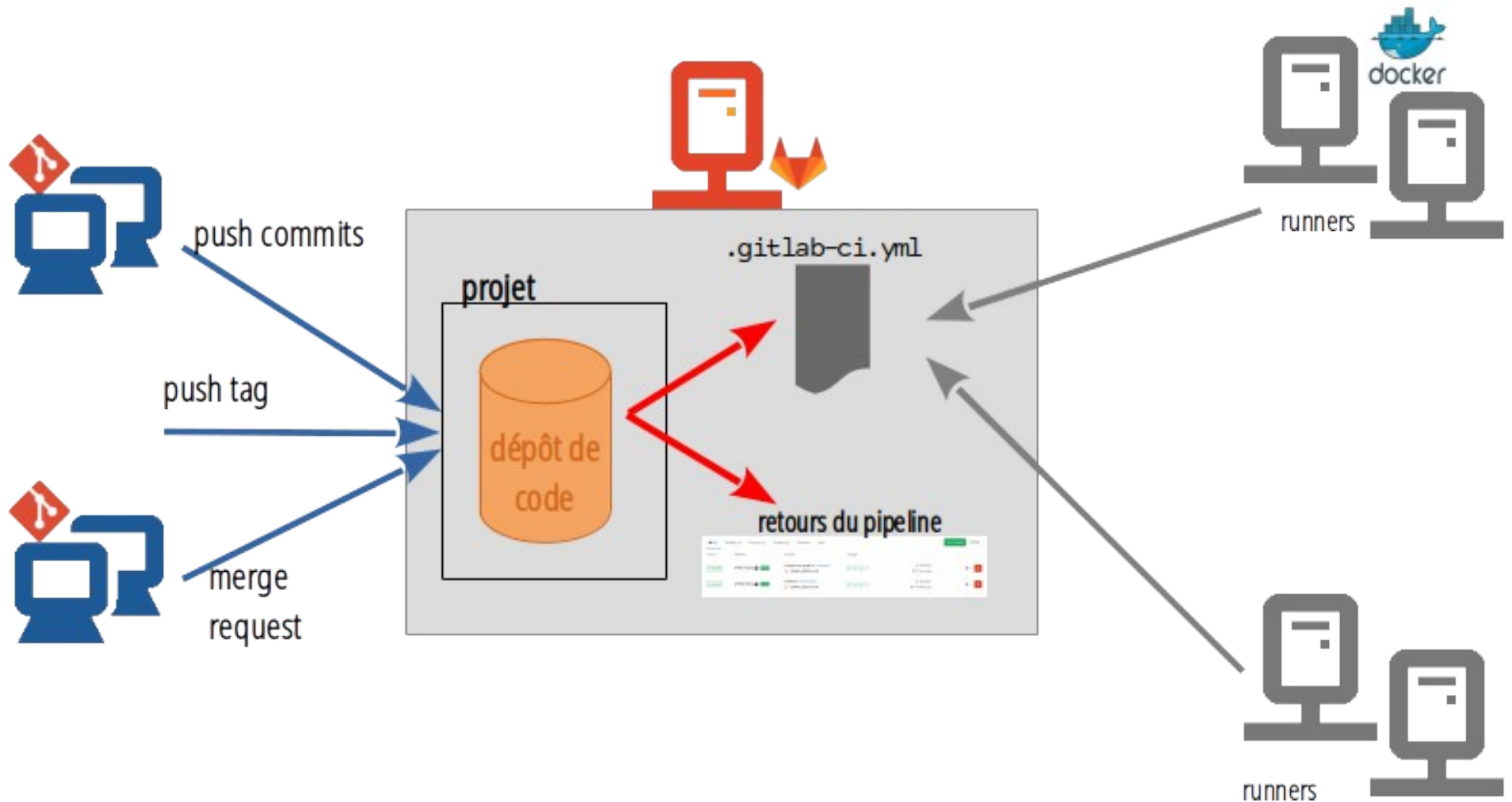
commits de merge, revert : compléter les messages par défaut en ajoutant de l'information sur la résolution de conflit, le moment du merge, la cause du revert...



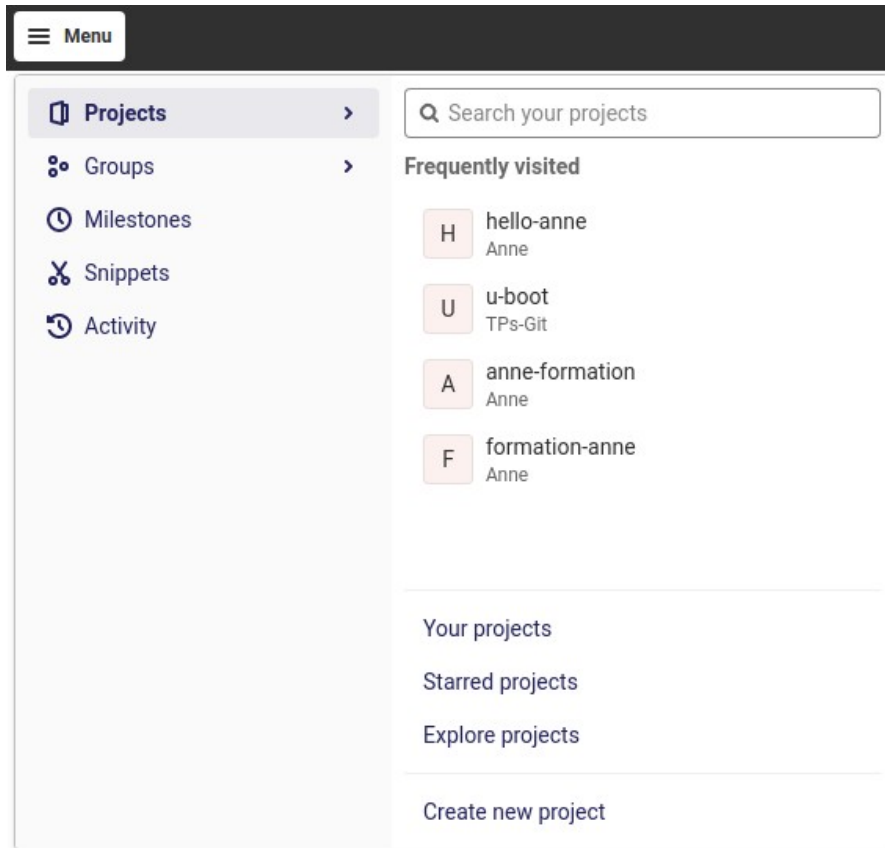
1 Gestion des dépôts git



Présentation



Interface de l'utilisateur



◆ Projets

ouverture d'un projet existant, création d'un projet

◆ Groupes

ouverture d'un groupe existant, création d'un groupe

◆ Milestones

liste des milestones existants (jalons), création

◆ Snippets

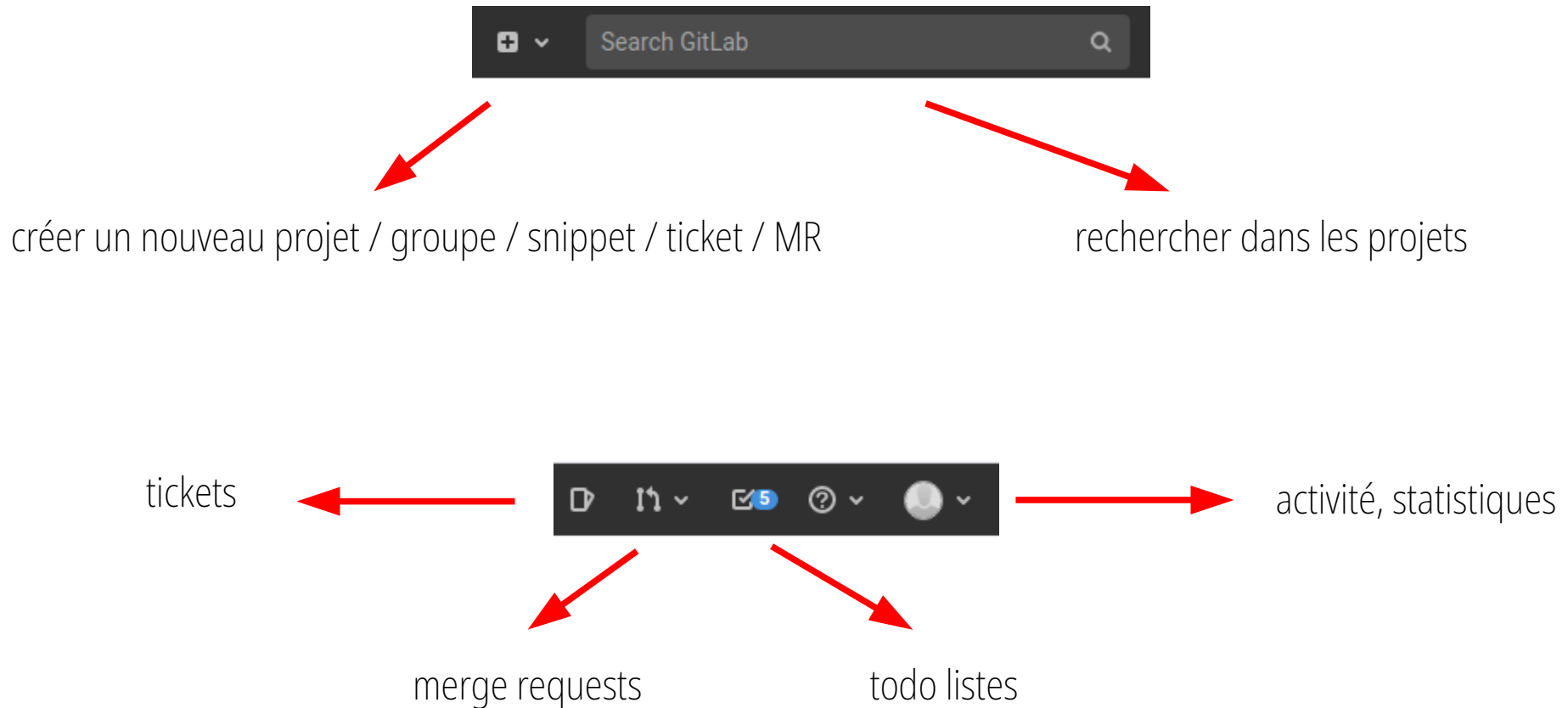
partage de code

◆ Activité

dashboard : push de commit, merge requests, issues (tickets),
commentaires, wiki



Interface de l'utilisateur



Gérer les notifications

◆ Trop de notifications tue les notifications !

Définir le niveau acceptable et utile des notifications reçues

User Settings > Notifications

Notifications

You can specify notification level per group or per project.

By default, all projects and groups will use the global notifications setting.

Global notification settings

Notification email

anicolas@hupstream.com

Global notification level

🔔 Participate ▼

☐ Receive notifications about your own activity

Groups (1)

🔊 dev_admin

🔔 Global ▼

Global notification email ▼

Projects (2)

To specify the notification level per project of a group you belong to, you need to visit project page and change notification level there.

🔊 Anne / u-bootadmin

🔔 Global ▼



Créer un nouveau projet

◆ Définition d'un projet

Dépôt git et ensemble des outils associés pour la gestion de son fonctionnement : issue tracker, CI et/ou CD associés, wiki, ...

Visibilité

Membres



Créer un nouveau projet

Utiliser un template de dépôt
pour sa création



Importer un dépôt depuis
Github, Gitlab, un serveur
git...



Blank project**Create from template****Import project**

Nom du projet

URL du projet

Identifiant « slug » du projet

Want to house several dependent projects under the same namespace? [Create a group.](#)

Project description (optional)

Description format

Visibility Level ?

- ☒ **Privé**
L'accès au projet doit être explicitement accordé à chaque utilisateur.
- ☐ **Interne**
Votre projet est accessible à tous les utilisateur et utilisatrices authentifiés.
- ☐ **Public**
Votre projet est accessible sans aucune authentification.

☐ **Initialize repository with a README**
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project**Annuler**

affiché
automatiquement sur
la page du projet



Visibilité : définit l'accès
au projet en fonction
des utilisateurs



Le fichier README

- ◆ Essentiel dans les bonnes pratiques d'un projet

contenu affiché par défaut sur la page d'accueil du projet
format asciidoc ou markdown

- ◆ Contenu du fichier

description générale du projet

statut du projet : modifications prévues et objectif du développement ou projet terminé

exigences concernant l'environnement de développement en vue de son intégration

instruction pour l'installation et l'utilisation

liste des technologies utilisées

bugs connus et corrections éventuelles apportées

section FAQ

droits d'auteurs et informations sur la licence



Gérer le dépôt du projet

espace de nom

configurer les notifications

forker le dépôt

visibilité

cloner le dépôt

commits,
branches, tags,
fichiers

IDE web

télécharger une
archive du projet

branche courante

ajout des fichiers de
base git / gitlab

fichiers du dépôt

ajouter un fichier, une
branche, un tag

accéder à l'historique
des commits

dev_admin > u-bootadmin > Détails



u-bootadmin

Identifiant de projet : 4



Ajouter aux favoris

0

Créer une divergence

1

Clone

62 620 commits 9 branches 369 étiquettes 150,8 Mo fichiers

master

u-bootadmin

+

Historique

Rechercher un fichier

EDI Web



Merge branch 'feature1' into 'master'
Anne a créé il y a une semaine



c94063bd



README

Configuration de l'intégration et de la livraison continues

Add LICENSE

Ajouter un CHANGELOG

Ajouter un CONTRIBUTING

Ajouter une grappe de serveurs Kubernetes

Nom	Dernier commit	Dernière mise à jour
.github	Add a github template telling people to not use pull re...	il y a un an
.gitlab/issue_templates	ajout des templates d'issues	il y a une semaine



Gestion des branches

◆ Liste des branches partagées sur le serveur





Overview Active Stale All

Filter by branch name

Delete merged branches New branch

Protected branches can be managed in [project settings](#).

Active branches

Y master default protected	
2fa1cdc6 · Mise à jour de la licence · 1 week ago	 
Y dev	
4d85b2b6 · Mise à jour de fic2 · 1 week ago	3 3 Merge request Compare  

nom de la branche

HEAD de la branche

Status de la branche : default, protected

stale branch : branche " archivée " car sans activité depuis au moins 3 mois - suppression ?



Création d'une branche

◆ Créer une nouvelle branche

New Branch

Branch name

Create from

Existing branch name, tag, or commit SHA

Create branch

nom de la branche à créer

HEAD de la branche (branche, commit, tag)



Gestion des tags

◆ Liste des tags partagées sur le serveur

Tags give the ability to mark specific points in history as being important

Filter by tag name

Last updated

New tag



v1.2

a18b805f · Update hello.c · 1 week ago



v1.1

6fbe0439 · Merge branch 'master' of https://gitlab-training.hupstream.com/ennael/ci-hello-anne · 1 week ago



v1.0

49007b8a · Update .gitlab-ci.yml file · 1 week ago



nom du tag

SHA1 du commit pointé par le tag

téléchargement de la version correspondante



Création d'un tag

New Tag

Tag name

Create from

Existing branch name, tag, or commit SHA

Message

Optionally, add a message to the tag. Leaving this blank creates a [lightweight tag](#).

Release notes

Optionally, create a public Release of your project, based on this tag. Release notes are displayed on the [Releases](#) page. [More information](#)

Write Preview

B *I* ” </> 🔗 ☰ ☷ ☹ ☰ ↗

Anne NICOLAS (7):

Fix du titre et compléments de doc
ajout de la licence isc
Démarre une todo liste
Mise à jour de la documentation
Ajoute la toto liste
Complète la todo list
Ajout du fichier Maintainers

Markdown is supported

📎 Attach a file

Create tag

Cancel



Comparer 2 révisions

◆ Outil de comparaison





comparer 2 SHA1, 2 branches, 2 tags

liste des commits

liste des différences

Source ... Target

Commits (2)

	Ajout de l'environnement production Anne authored 1 week ago	<input type="text" value="f28772a9"/> 
	Merge branch 'master' of https://gitlab-training.hupstream.com/ennael/ci-hello-anne Anne authored 1 week ago	<input checked="" type="checkbox"/> <input type="text" value="6fbe0439"/> 

Showing 1 changed file ▾ with 2 additions and 0 deletions

▾  .gitlab-ci.yml 



```
...      @@ -38,6 +38,8 @@ hello_deploy:
38      38      - mkdir bin
39      39      script:
40      40      - mv hello bin/
41      41      + environment:
42      42      +   name: production
43      43      rules:
44      44      - if: '$CI_COMMIT_TAG != null'
45      45      - when: never
```



3 Gestion des droits d'accès



Groupes



◆ Définition

Ensemble de projets, accompagnés des informations de droits d'accès à ces projets. Chaque groupe a un espace de nom

Groups

Your groups Explore public groups

Filter by name... Last created ▾ New group

  D devel Owner

0 0 1 0 0
2 5 4 3 0

- 1 - groupes auxquels l'utilisateur appartient
- 2 - sous-groupes
- 3 - nombre de projets contenus
- 4 - nombre de membres
- 5 - visibilité
- 6 - configuration (si permissions suffisantes)



Groupe

◆ Cas d'utilisation

Moyen simple d'organiser un ensemble de projets sous un même espace de nom, ajouter des membres et donner accès à l'ensemble de ces projets

Créer un groupe, ajouter des membres et le mentionner avec @<groupe> dans les issues et merge requests

Exemple : entreprise = 1 groupe et 1 sous-groupe / équipe

- démarrage d'un nouveau développement (issue)
- commentaire : "@entreprise, let's do it! @entreprise/backend you're good to go!"
- l'équipe backend a besoin de l'aide de l'équipe frontend, ajoute un commentaire : "@entreprise/frontend could you help us here please?"



Espaces de noms

◆ Les différents types

`http://gitlab.example.com/username`

`http://gitlab.example.com/groupname`

`http://gitlab.example.com/groupname/subgroup_name`

◆ Mention des utilisateurs et des groupes

`@username`

`@groupname`

`@groupname/subgroup_name`

Attention noms réservés : https://docs.gitlab.com/ee/user/reserved_names.html



Créer un groupe : les composantes

- ◆ Chemin

espace de nom qui contiendra les projets

- ◆ Nom du groupe

- ◆ Description

- ◆ Avatar

- ◆ Niveau de visibilité

private seuls les membres listés ont accès au projet

internal seuls les détenteurs d'un compte gitlab ont accès au projet

public le projet est accessible pour un utilisateur non connecté



Gérer les membres d'un groupe

◆ Ajout

Nom, rôle

devel > **Members**

Members

Add new member to **devel**

Administrator ✕

Search for members by name, username, or email, or invite new ones using their email address.

Developer ▼

[Read more](#) about role permissions

Expiration date

On this date, the member(s) will automatically lose access to this group and all of its projects.

Add to group



Les rôles : définition

◆ Guests

contributeurs non actifs dans un projet privé - lecture et commentaires sur les issues

◆ Reporters

contributeurs en lecture seule - pas de droits en écriture sur le dépôt mais possibilité de créer une nouvelle issue

◆ Developers

contributeurs directs - accès à tout sauf mention contraire (ex : protection de branche)

◆ Maintainers

super développeurs - push sur master, déploiement en production

◆ Owners

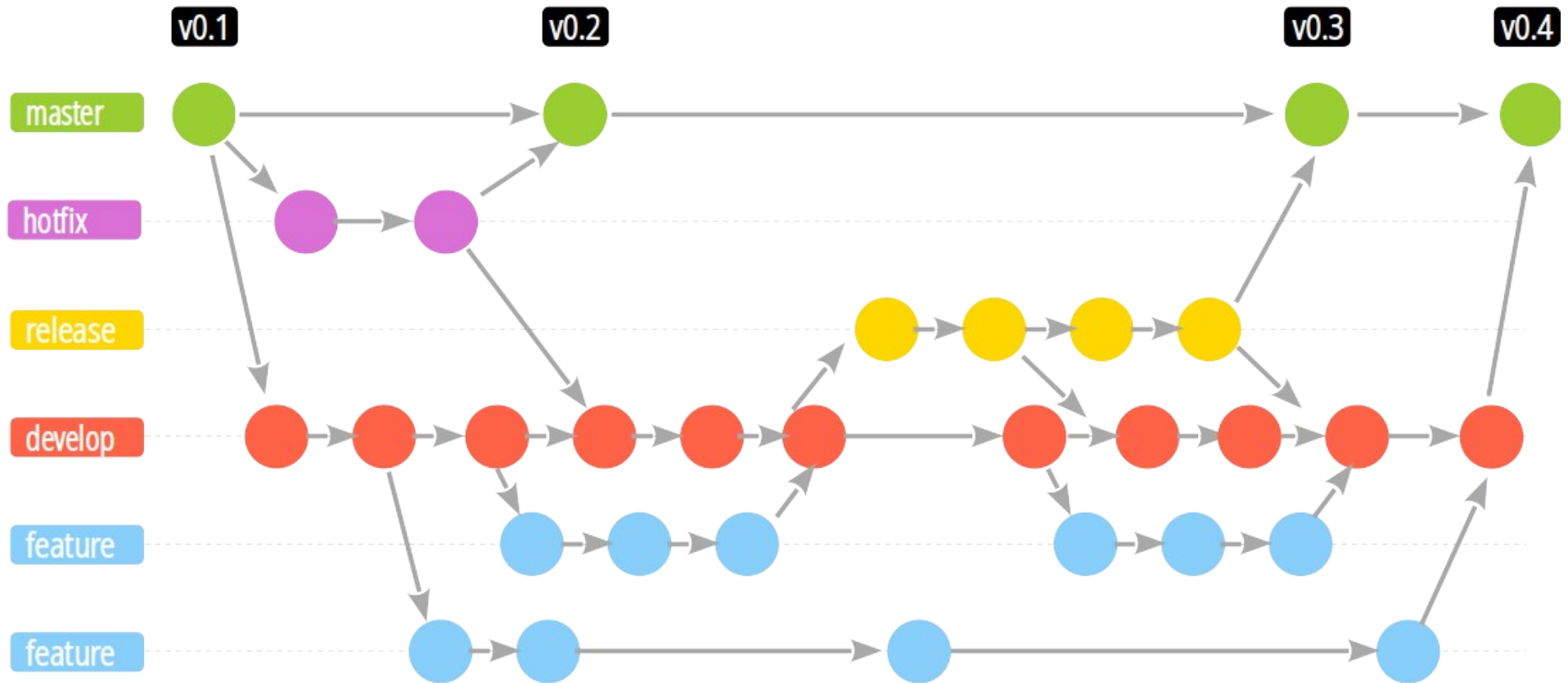
administrateurs - attribution des accès aux groupes et suppressions



4 Workflows et organisations - Merge Requests



gitflow : les nuances



◆ master
production

◆ develop
développement

◆ feature branches
développement d'une feature

◆ release branches
bug fix avant mise en production

◆ hotfix
branche de correction de bug sur la production

modèle jugé complexe et inadapté dans un certain nombre de cas



github flow : simplification

- ◆ master

production

- ◆ feature branches

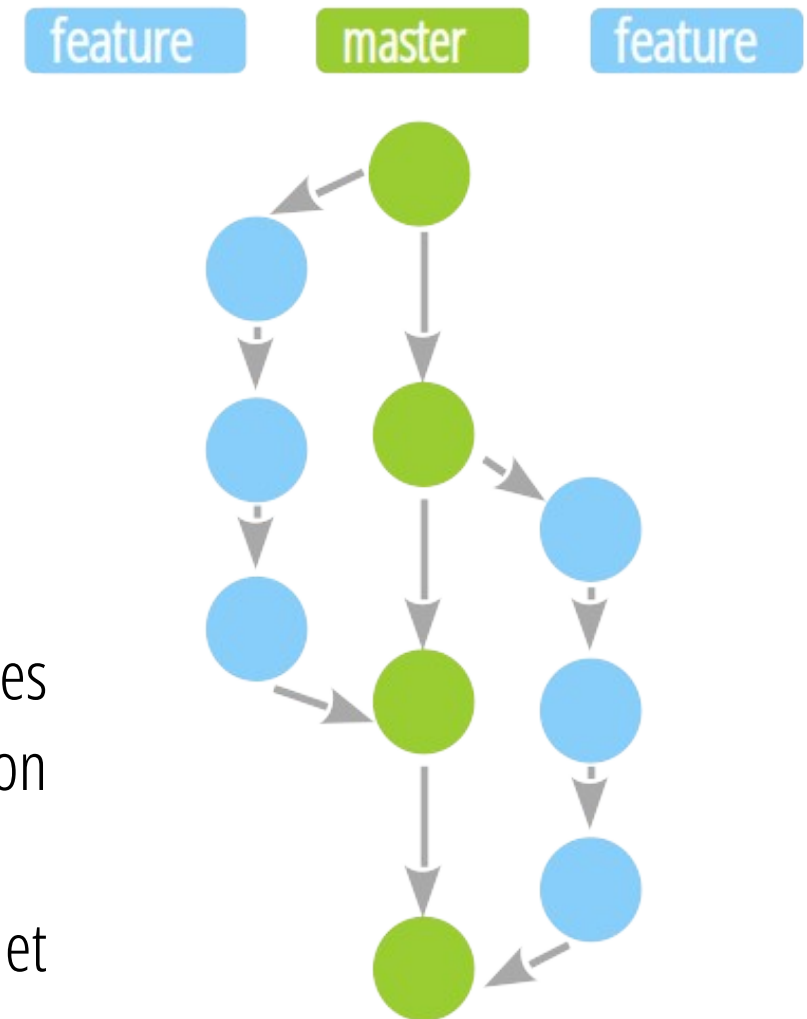
développement d'une fonctionnalité ou d'un bug fix

- ◆ déploiements fréquents

- ◆ Les bémols

manque de guide sur la façon de déployer, les environnements de release et l'intégration de la gestion des issues

présuppose que le merge d'une feature est fonctionnel et déployable



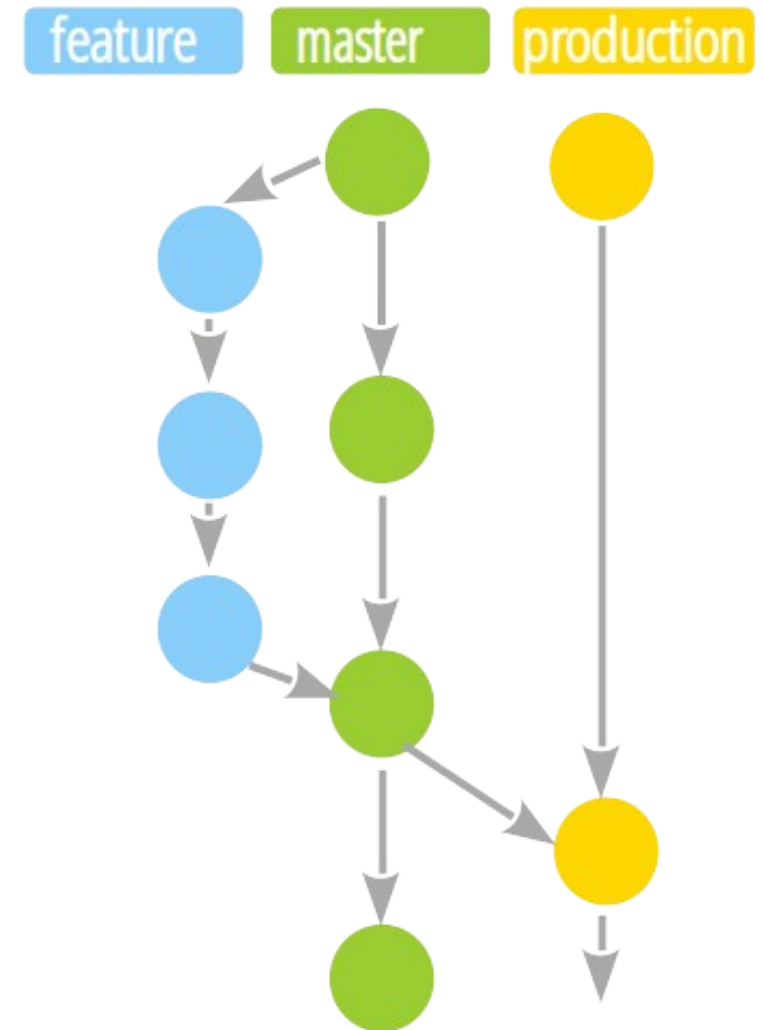
Gitlab flow : branche de production

◆ Définition

reflète le code déployé

commit de merge = déploiement

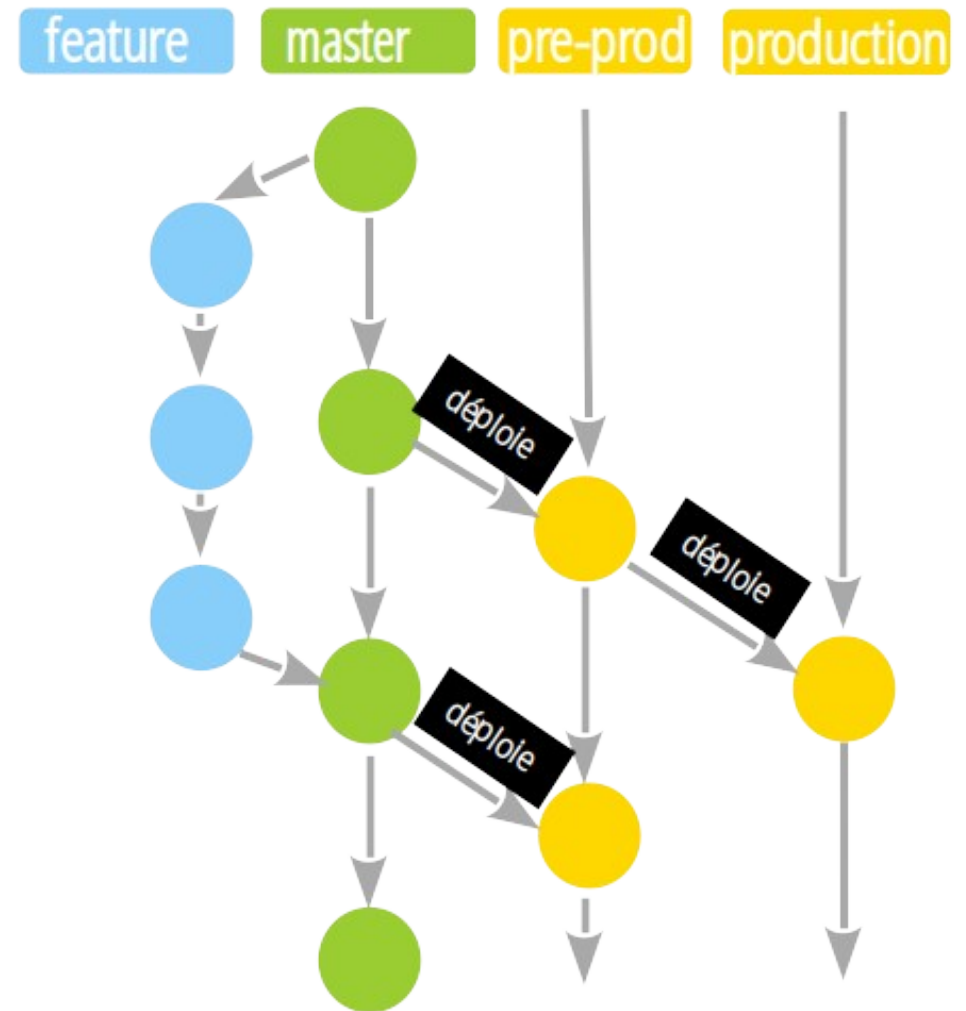
- utilisation de tag pour acter le déploiement (plus de précisions sur le moment du déploiement)



Gitlab flow : branche d'environnement

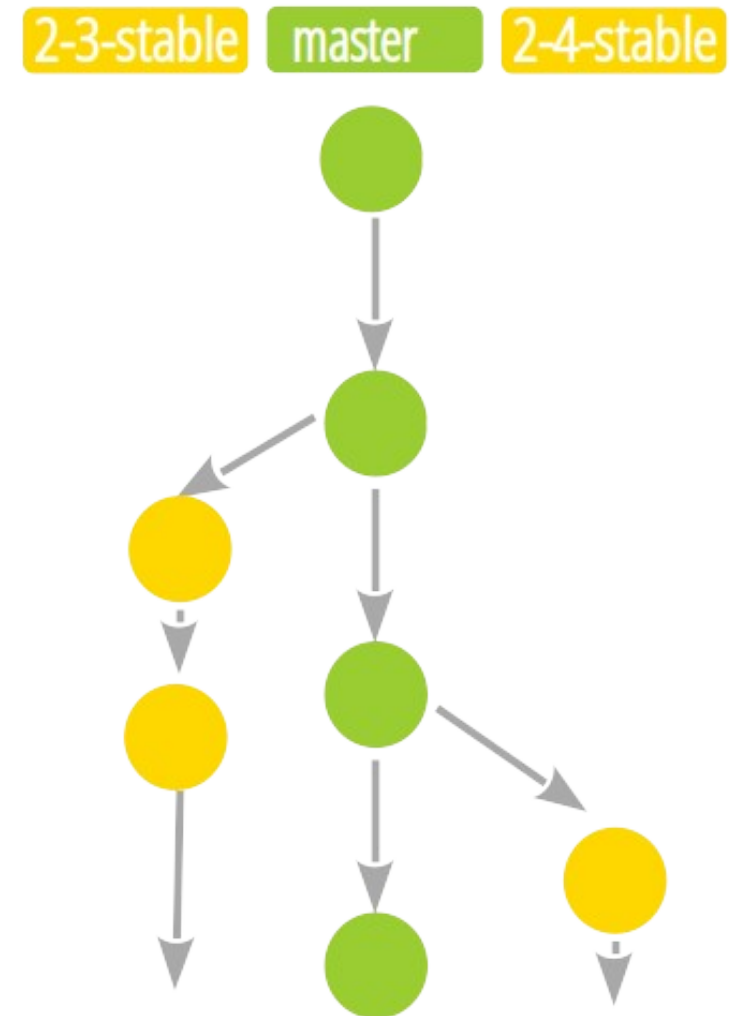
◆ Définition

définit des branches par environnement de déploiement



Gitlab flow : branche de release

- ◆ contient les versions mineures
- ◆ ne sont versés que des bugs fixes sérieux :
merge dans master puis cherry-pick dans la
branche de release («upstream first »)
- ◆ 1 bug fix => incrémente numéro PATCH
- ◆ utilisation de tags :
MAJOR.MINOR.PATCH



Workflow à base de fork

- ◆ développeur

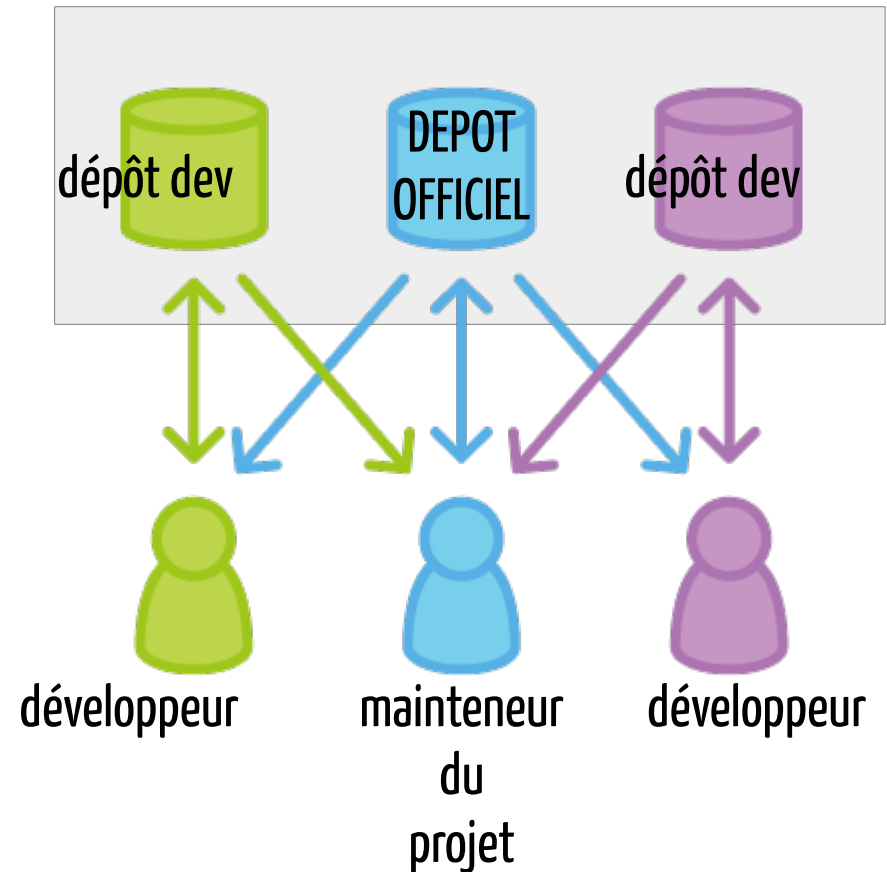
= 1 dépôt privé local + 1 dépôt distant

- ◆ mainteneur

= dépôt officiel du projet

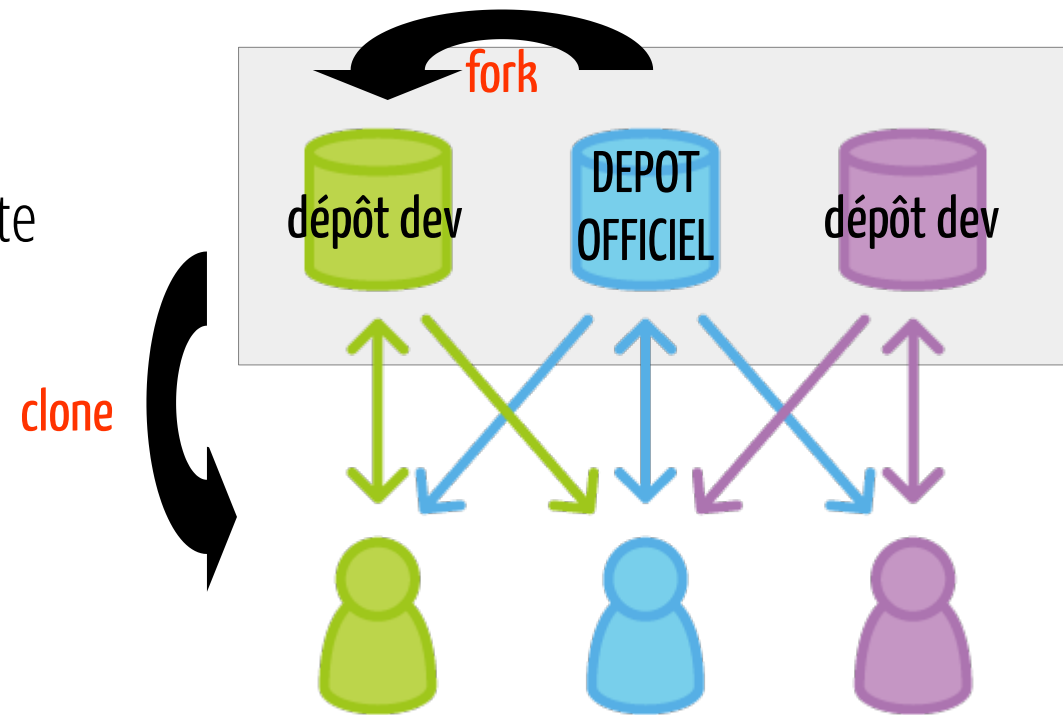
- ◆ cas d'usage

permet de gérer des intervenants extérieurs sans donner des droits d'accès au dépôt officiel



Workflow à base de fork

- ◆ dépôt public dev est un fork du dépôt officiel
un clone côté serveur (git clone --bare)
- ◆ utilisation de 2 remotes
origin (dev) et upstream (officiel) avec git remote add
- ◆ dev peut mettre à jour depuis le dépôt officiel
(git pull upstream master)
- ◆ dev pousse ses modifications sur le dépôt distant dev



Bonnes pratiques et workflow efficace

- ◆ Pas de commit direct dans master mais dans des branches de feature
- ◆ Branches de feature : pousser régulièrement le contenu de la branche pour éviter qu'un autre développeur ne produise un doublon
- ◆ Messages de commit
- ◆ Tester avant de merger
 - CI sur la branche de feature
 - développements plus longs : merge de master dans la branche de feature
- ◆ Déploiements basés sur des branches et des tags
- ◆ Tags posés manuellement



Bonnes pratiques et workflow efficace

- ◆ Ne jamais rebaser des commits déjà poussés
- ◆ Tous les développeurs ciblent master et démarrent une branche depuis master
- ◆ Fixer les bugs depuis master puis dans les branches de release



Organiser le développement



Gérer les milestones

◆ Rôle

versionner du développement ou suivi des sprints en méthode Agile
visualiser les tickets, les merge requests ouvertes, finalisées
utiliser le time tracking
visualiser l'activité
lister les participants

Anne > hello-ci > Milestones > **v2.0 rc1**

Open Milestone May 8, 2021–May 17, 2021

Edit

Close milestone

Delete

v2.0 rc1

Release Candidate 1 - Version 2.0

Issues 2 Merge requests 1 Participants 1 Labels 2

Unstarted Issues (open and unassigned)

0

Ongoing Issues (open and assigned)

2

Completed Issues (closed)

0

Finalisation des tests unitaires associés

#2

Correction typos documentation

#1

documentation

enhancement

0% complete



Start date
May 8, 2021

Edit

Due date
May 17, 2021 (5 days remaining)

Edit

Issues 2 New issue
Open: 2 Closed: 0

Time tracking
Estimated: 7h



Merge requests 1
Open: 1 Closed: 0 Merged: 0

Releases
None

Reference: ennael/hello-ci%v2.0 rc



Gérer les milestones

◆ Création

nom, description, date de début et de fin (facultatif)

New Milestone

Title	<input type="text" value="v2.0 rc1"/>	Start Date	<input type="text" value="Select start date"/> Clear start date
Description	<div><div>Write Preview</div><div>B I ” </> 🔗 ☰ ☰ ☰ ☰ 📎</div><div>Release Candidate 1 - Version 2.0</div><div>Markdown is supported Attach a file</div></div>	Due Date	<input type="text" value="Select due date"/> Clear due date
Create milestone		Cancel	



Gérer les releases

◆ Contenu

Version 2.0 RC1

0% complete

Milestone **v2.0 rc1** Issues **2**
Open: 2 • Closed: 0

Assets 5

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Other

hupstream

Evidence collection

v2.0-rc1-evidences-15.json bccd0fd5

Collected just now

00e36af3 v2.0-rc1 Created just now by

} assets : archive du contenu du projet pour le milestone, lien associé

} evidence + SHA : fichier décrivant l'ensemble des metadata du milestone attestant son contenu dans la chaîne de production logicielle



Gérer les releases

◆ Contenu

associées à un tag existant
permettent la création d'un tag et d'une release
créées automatiquement lors de l'ajout d'un tag
possibilité d'associer un fichier de release notes

Project overview > Releases

v4.0

Assets 4

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Evidence collection

- v4.0-evidences-16.json f02d12aa
- Collected 3 minutes ago
- RELEASE-NOTES.en.txt

5e47a0d9 v4.0 Created 3 minutes ago by

Version 2.0 RC1

0% complete

Milestone v2.0 rc1 Issues 2
Open: 2 • Closed: 0

Assets 5

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)



Wiki et issue board

◆ Wiki et issues : niveau de visibilité

Disabled	pour tout le monde
Only team members	seuls les membres de l'équipe même si le projet est public ou internal
Everyone with access	tout le monde en fonction du niveau de visibilité

◆ Issue board

utilisation des fonctionnalités : developers et plus
création/suppression de listes et déplacement d'issues

◆ Confidential issues

un tag précis ou utilisation de wildcards
protection par rapport à la création, modification, suppression de tags donnés



Permissions et CI

◆ Liste des rôles

admin

maintainer

developer

guest/reporter

◆ Descriptif complet des permissions

<https://docs.Gitlab.com/ee/user/permissions.html#Gitlab-ci-cd-permissions>



5 Merge request



Branche protégée

◆ Définition

ne peut pas être créée si elle n'existe pas sauf par les mainteneurs

interdit le push sauf pour les mainteneurs

interdit le push --force

empêche la suppression de branche

appliqué par défaut à la branche master, configurable par les mainteneurs

◆ Configuration

Dans le dépôt : Settings > Repository > Protected branches

Allowed to merge, Allowed to push - Mainteneurs par défaut

Wildcards pour les noms des branches à protéger

◆ Suppression d'une branche protégée

nécessite les droits suffisants

uniquement via l'interface web



Tag protégé

◆ Fonctionnement et utilisation

Settings > Repository > Protected tag

un tag précis ou utilisation de wildcards

protection par rapport à la création, modification, suppression de tags donnés



Que faire avec une merge request

- comparer les changements entre branches (diff branches)
- revue et discussion des modifications proposées
- builder, tester, déployer le code pour une branche donnée avec la CI
- WIP merge request
- visualiser le process de déploiement avec les graphs pipelines
- assigner à un utilisateur et modifier cette assignation autant de fois que nécessaire
- assigner à un milestone
- organiser les issues et merge requests avec les labels
- time tracking
- résoudre les conflits depuis l'interface web
- définir le type de merge autorisé
- créer des merge requests par mail
- squash des commits



Merge request : déroulé gitlab flow

- ◆ création d'une branche de feature
- ◆ ouverture d'une merge request non assignée (WIP)
partager le travail en cours
- ◆ utilisation de raccourcis pour désigner les gens potentiellement intéressés ex : /cc @rtp
- ◆ commentaires et discussions sur le développement en cours
- ◆ demande de modifications
commits par n'importe quel membre de l'équipe mais de préférence l'initiateur de la merge request
- ◆ finalisation la merge request
assigner à 1 participant
- ◆ titre indique merge ou non : [WIP]....
- ◆ merge non fast-forward systématique
- ◆ suppression de la branche de feature

Attention aux branches protégées : assigner la MR à un utilisateur ayant les droits de modification




Créer une merge request

dev_admin > u-bootadmin > Merge Requests > **New**

New Merge Request

Source branch

dev_admin/u-bootadmin next

 Merge branch 'next' ...
Tom Rini a créé il y a 3 semaines

5a8fa095


Compare branches and continue

branche à fusionner

HEAD de la branche

Target branch

dev_admin/u-bootadmin master

 Merge branch 'feature1' into 'master' ...
Anne a créé il y a une semaine

c94063bd

branche dans laquelle exécuter la fusion

HEAD de la branche



Créer une merge request

dev_admin > u-bootadmin > Merge Requests > New

New Merge Request

From **next** into **master** [Change branches](#)

branches à merger
avec la MR

Titre

WIP: Next

Remove the **WIP:** prefix from the title to allow this **Work In Progress** merge request to be merged when it's ready.
Ajouter [description templates](#) pour aider vos contributeurs à communiquer efficacement !

Description

Write Aperçu

Describe the goal of the changes and what reviewers should be aware of.

Markdown and quick actions are supported

[Attach a file](#)

Assignee

Unassigned

[Assign to me](#)

assigner la MR

Milestone

Milestone

associer la MR à un milestone

Labels

Labels

associer la MR à des labels

Merge options

☐ Squash commits when merge request is accepted. ?

Submit demande de fusion

Cancel

fusionner les commits entrants lors du merge



Gérer une merge request

dev_admin > u-bootadmin > Merge Requests > 13

Ouvrir Opened dans 7 heures by Anne

Edit

Close demande de fusion

WIP: Add documentation

Vue d'ensemble 0 Commits 0 Changes

Create documentation tree to allow future contribution



Demande de fusion de next dans master

La branche source est à 450 commits de retard de la branche cible

Ouvrir dans l'EDI Web

Récupérer la branche



Merge requests are a place to propose changes you have made to a project and discuss those changes with others.

Interested parties can even contribute by pushing commits if they want to.

Currently there are no changes in this merge request's source branch. Please push new commits or use a different branch.



Créer un fichier



0



0



Afficher toute l'activité



Anne @ennael changed milestone to %1.0 in 7 hours



Anne @ennael added To Do label in 7 hours



Write Aperçu

B I ” </> @ :: ≡ ≡ ≡ ≡ ↗

Write a comment or drag your files here...

To Do

Mark as done



Assignee

Éditer



Anne
@ennael

Jalon

Éditer

1.0

Suivi du temps



Aucune estimation ou temps passé

Étiquettes

Éditer

To Do

Verrouiller merge request

Éditer

Déverrouillé

1 participant-e



Notifications



Référence : dev_admin/u-bootad...



Rôle des labels

◆ Définition

Permet de catégoriser les merge requests avec des étiquettes (texte + couleur)
Visualisables sur tous les outils représentant merge requests

◆ 2 types de labels

Project labels : assignés à des issues et/ou merge requests pour 1 projet donné

Group labels assignés à des issues et/ou merge requests pour les projets d'un groupe ou d'un sous-groupe



Branche par défaut

◆ Définition

branche protégée

branche proposée par défaut pour les merge requests

branche récupérée en tant que branche locale au moment du clone

configurable pour le projet, le groupe

Settings > Repository > Default branch

Default branch

Set the default branch for this project. All merge requests and commits are made against this branch unless you specify a different one.

Default branch

☒ Auto-close referenced issues on default branch

When merge requests and commits in the default branch close, any issues they reference also close. [?](#)



Merge Request et workflow

◆ Configurer le fonctionnement des merge requests sur un projet

Settings > General > Merge requests

◆ Merge method

Merge requests

Choose your merge method, merge options, merge checks, and merge suggestions.

Merge method

Determine what happens to the commit history when you merge a merge request.

☒ Merge commit

Every merge creates a merge commit.

☐ Merge commit with semi-linear history

Every merge creates a merge commit.

Fast-forward merges only.

When there is a merge conflict, the user is given the option to rebase.

☐ Fast-forward merge

No merge commits are created.

Fast-forward merges only.

When there is a merge conflict, the user is given the option to rebase.

- Pas de restriction sur le type de merge
- Création d'un commit de merge quelle que soit la situation
- Merge fast-forward obligatoire
- Branche de travail à jour (rebase)
- Commit de merge
- Merge fast-forward obligatoire
- Pas de commit de merge



Merge Request et workflow

◆ Merge options

- Pas de restriction sur le type de merge
- Création d'un commit de merge quelle que soit la situation

Merge options

Additional settings that influence how and when merges are done.

- ☐ Automatically resolve merge request diff discussions when they become outdated
- ☒ Show link to create or view a merge request when pushing from the command line
- ☒ Enable "Delete source branch" option by default

Existing merge requests and protected branches are not affected.

- Message console locale lors du push de branches ou de commits

```
remote:
remote: To create a merge request for feature1, visit:
remote:   https://gitlab-training.hupstream.com/ennael/hello-
ci/-/merge_requests/new?merge_request%5Bsource_branch%5D=feature1
remote:
```

- Supprimer la branche une fois la merge request exécutée



Merge Request et workflow

◆ Squash commits when merging

Génère un commit unique dans la branche recevant la merge request

Ne conserve pas l'historique de la branche à merger

Simplifie l'historique

- pas de squash de commits au moment d'une merge request

Squash commits when merging

Set the default behavior of this option in merge requests. Changes to this are also applied to existing merge requests. [What is squashing?](#)

☐ **Do not allow**

Squashing is never performed and the checkbox is hidden.

☒ **Allow**

Checkbox is visible and unselected by default.

☐ **Encourage**

Checkbox is visible and selected by default.

☐ **Require**

Squashing is always performed. Checkbox is visible and selected, and users cannot change it.

- Propose le squash des commits

- Forcer le squash des commits sur les merge requests



Merge Request et workflow

◆ Merge checks

Lier l'intégration continue et les discussions à l'exécution d'une merge request

Merge checks

These checks must pass before merge requests can be merged.

☐ Pipelines must succeed

To enable this feature, configure pipelines. [How to configure pipelines for merge requests?](#)

☐ Skipped pipelines are considered successful

Introduces the risk of merging changes that do not pass the pipeline.

☐ All discussions must be resolved

- La merge request n'est possible que si le pipeline s'exécute correctement

- Vérifie le status des discussions en cours liées à la merge request



Revue de code en ligne



Review du code en ligne

◆ Proposer une modification en ligne : suggestions

Commentaires associés au code proposé dans le ou les fichiers concernés

Soumission d'une review et de modifications

Une ou plusieurs lignes

Commit de la suggestion dans la branche proposant la merge request

1 - Consulter les modifications proposées par la merge request

Anne > hello-ci > Merge requests > 15

Open Created just now by Anne Maintainer

Edit Mark as draft

Améliore la documentation du code

Overview 0 Commits 1 Pipelines 2 Changes 1

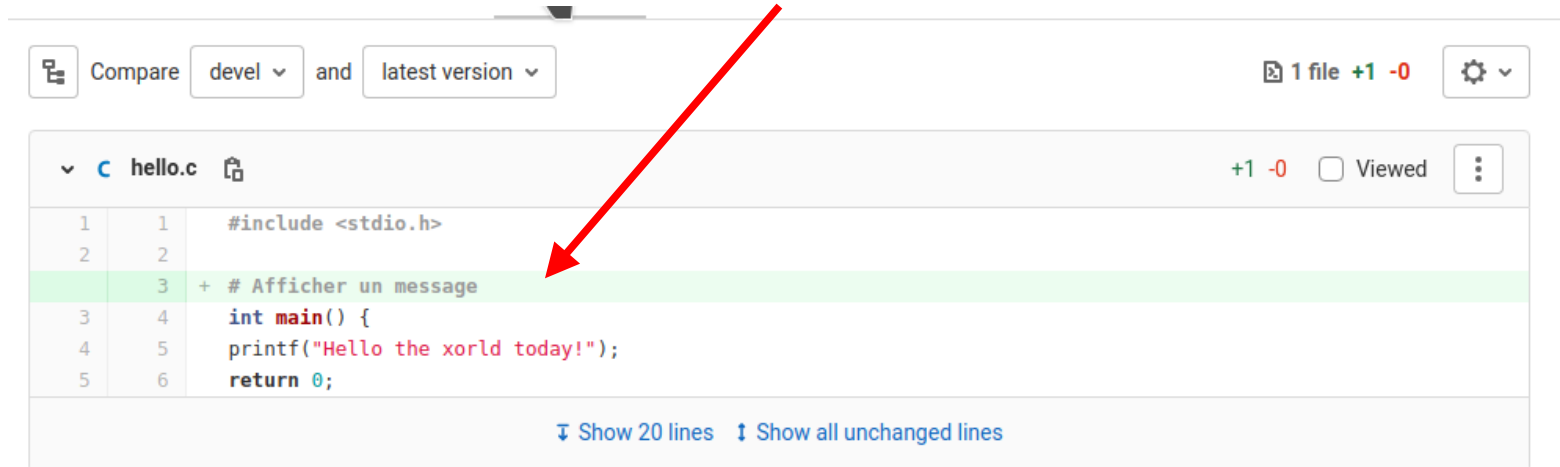
Request to merge documentation/code into devel

Open in Web IDE Check out branch



Review du code en ligne

2 - Localiser la modification dans l'ensemble proposé



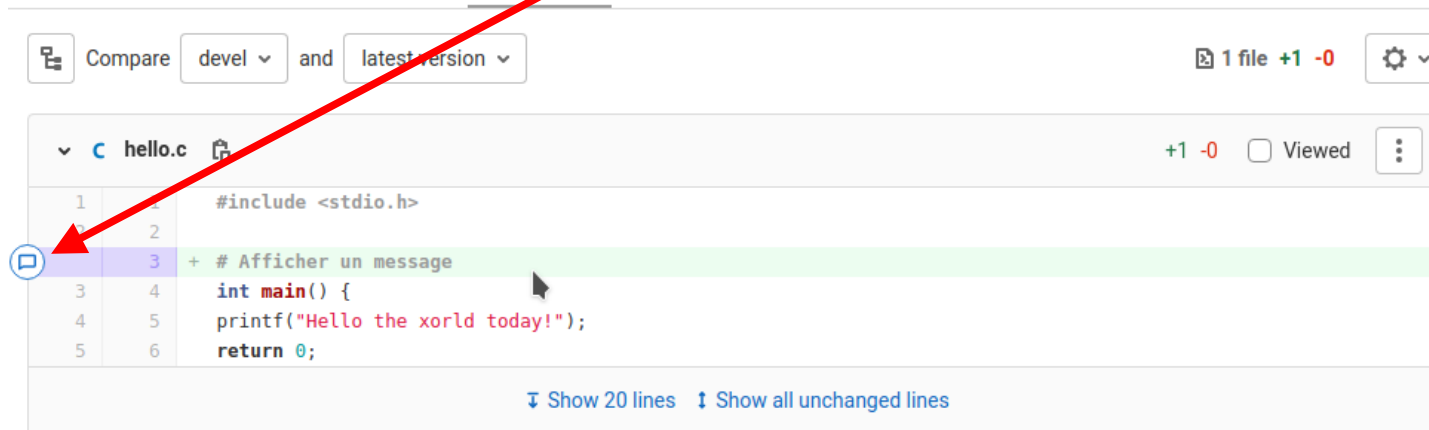
Compare devel and latest version 1 file +1 -0

hello.c +1 -0 Viewed

1	1	#include <stdio.h>
2	2	
3	+	# Afficher un message
3	4	int main() {
4	5	printf("Hello the xorld today!");
5	6	return 0;

Show 20 lines Show all unchanged lines

3 - Survoler en laissant apparaître l'icône de commentaire et cliquer



Compare devel and latest version 1 file +1 -0

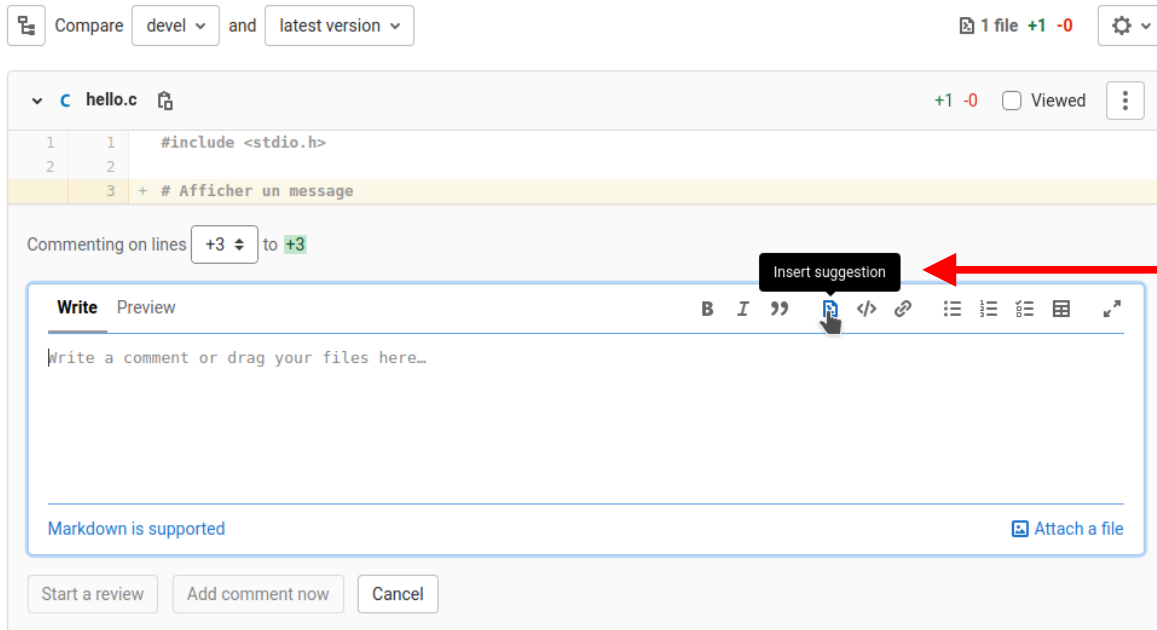
hello.c +1 -0 Viewed

1	1	#include <stdio.h>
2	2	
3	+	# Afficher un message
3	4	int main() {
4	5	printf("Hello the xorld today!");
5	6	return 0;

Show 20 lines Show all unchanged lines

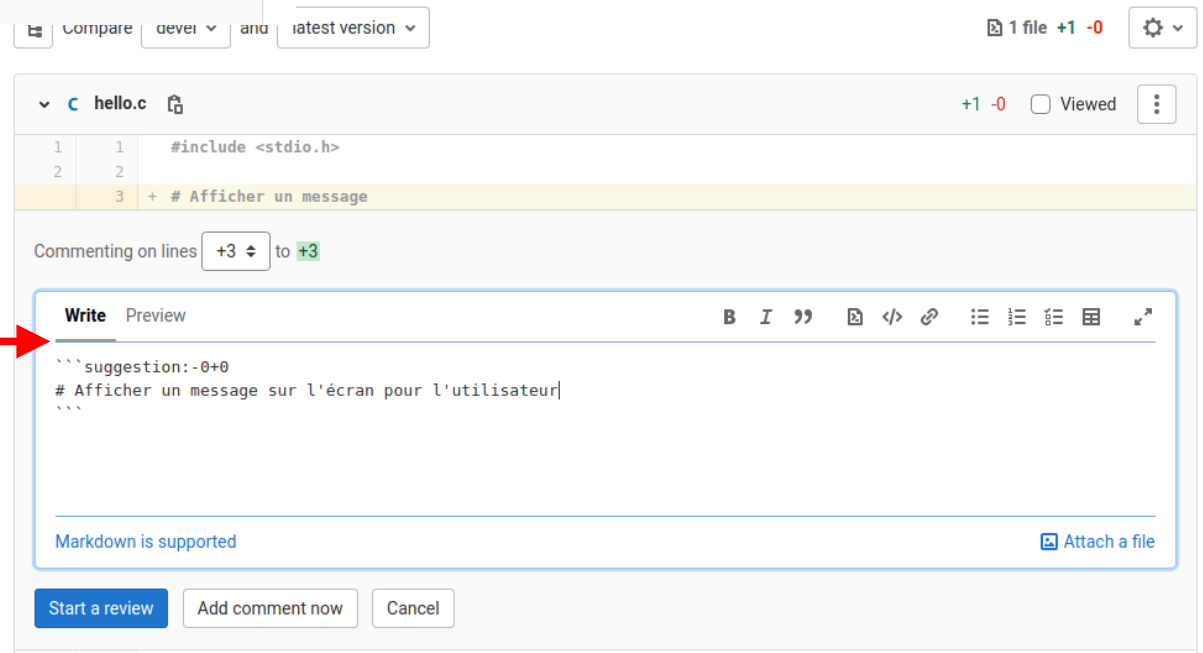


Review du code en ligne



4 - Ajouter une suggestion - cliquer sur l'icône

5 - Apporter les modifications à proposer



Review du code en ligne

Compare devel and latest version 1 file +1 -0

hello.c +1 -0 Viewed

```
1 1 #include <stdio.h>
2 2
3 + # Afficher un message
```

Pending **Anne** @ennael

Afficher un message sur l'écran pour l'utilisateur

Submit review 1 Add comment now

```
3 4 int main() {
4 5     printf("Hello the world today!");
5 6     return 0;
```

[Show 20 lines](#) [Show all unchanged lines](#)

6 - Envoyer la suggestion

7 - Committer la suggestion

Pending comments Submit review 1

hello.c +1 -0 Viewed

```
1 1 #include <stdio.h>
2 2
3 + # Afficher un message
```

Anne @ennael · just now

Suggested change

```
3 - # Afficher un message
3 + # Afficher un message sur l'écran
```

Commit message

Commentaire du code pour le développeur

Apply suggestion Apply

[Start a new discussion...](#)

```
3 4 int main() {
4 5     printf("Hello the world today!");
5 6     return 0;
```

[Show 20 lines](#) [Show all unchanged lines](#)



Review du code en ligne

◆ Historique de la branche à merger

documentation/c... hello-ci Author View open merge request Search by message

12 May, 2021 2 commits

Commentaire du code pour le développeur
Anne authored just now 1b6ed02f

Améliore la documentation du code
Anne authored 7 minutes ago 5f9a9936

HEAD antérieur à la suggestion

Commit ajouté par la suggestion



6 Gestion des Issues



Objectif des issues

◆ Créer du lien entre les développeurs

Un changement significatif du code, une correction de bug démarrent par la création d'une issue (ticket)

Création d'une branche depuis master contenant n°issue

Développement finalisé ou discussion nécessaire => une ou plusieurs merge requests



Créer une issue

Nouveau ticket

tickets avec un titre similaire

type de ticket :
issue
incident

Similar issues ?

Titre

test|

utiliser des templates pour pré-remplir le formulaire

Ajouter [description templates](#) pour aider vos contributeurs à communiquer efficacement !

test

#1 • 51 minutes ago by • updated 5 minutes ago

Type

Issue

Description

Write

Aperçu

Write a comment or drag your files here...

Markdown and quick actions are supported

Attach a file

assigner le ticket

☐ This issue is confidential and should only be visible to team members with at least Reporter access.

issue confidentielle

associer le ticket à un milestone

Assignee

Unassigned

Assign to me

Due date

Select due date

Milestone

Milestone

Labels

Étiquettes

associer le ticket à un label

deadline
(time tracking)



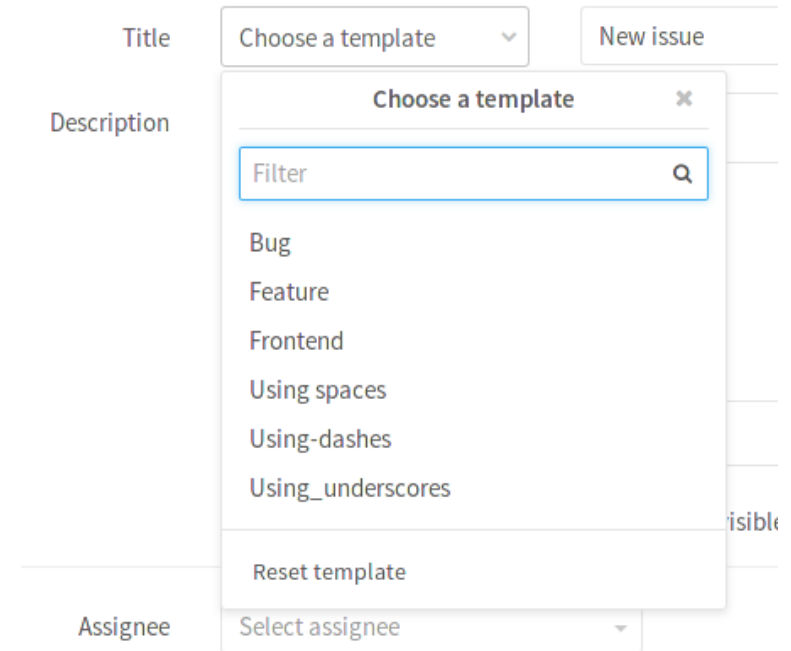
Lier les issues et les merge requests

- ◆ Mention dans le message de commit ou dans la description de la merge request
ex : fixes #14, closes #67...
Ajout de commentaire dans l'issue et lien vers la merge request
- ◆ Fermeture de l'issue après merge du code dans la branche par défaut
- ◆ Création d'une référence à une issue sans la fermer
ex : "Duck typing is preferred. #12"



Créer des templates

- ◆ utilisé pour les issues et merge requests
- ◆ format yaml
- ◆ dans un sous-répertoire du projet



The screenshot shows the 'New issue' form in GitLab. The 'Title' field has a dropdown menu open with the title 'Choose a template'. The dropdown menu lists several template categories: Bug, Feature, Frontend, Using spaces, Using dashes, and Using underscores. There is a search bar at the top of the dropdown with the placeholder text 'Filter'. Below the list of categories, there are two buttons: 'Reset template' and 'Select assignee'.

```
<dépôt_git>/ .gitlab/issue_templates/<nom_template>.md  
<dépôt_git>/ .gitlab/merge_request_templates/<nom_template>.md
```

- ◆ un template à utiliser pour un projet spécifique
instructions, guide, informations demandées
- ◆ des templates dédiés pour les différentes étapes du workflow
proposition de feature, amélioration, bug report...



Créer des templates

◆ Exemple : template de type bug report

Summary

(Summarize the bug encountered concisely)

Steps to reproduce

(How one can reproduce the issue - this is very important)

Example Project

(If possible, please create an example project here on GitLab.com that exhibits the problematic behaviour, and link to it here in the bug report)

(If you are using an older version of GitLab, this will also determine whether the bug has been fixed in a more recent version)

What is the current bug behavior?

(What actually happens)

What is the expected correct behavior?



7 Labels, issues boards, snippets



Créer des labels

◆ Prérequis

minimum développeur

◆ Procédure

dans le projet ou le groupe, issues > Labels > New label

liste par défaut proposée à l'ajout

possibilité de diffuser les labels d'un projet à un groupe

possibilité de prioriser les labels pour l'affichage



Utiliser des labels

- ◆ Outils de filtre d'affichage

 - project board

 - group issue boards

- ◆ Suivi de l'activité

 - notifications pour un label donné : alerte lors de l'assignation d'une issue ou un merge request à ce label



Utiliser les snippets

- ◆ Stocker et partager un extrait de code, personnel ou pour un projet

Nouvel extrait de code

Titre

Description

Write Aperçu




B *I* ” ” </> 🔗 ☰ ☷ ☰ ☷ ☰ ☷

Write a comment or drag your files here...

Markdown and quick actions are supported

 Attach a file

Niveau de visibilité ?

- ☒  Privé
The snippet is visible only to project members.
- ☐  Interne
The snippet is visible to any logged in user.
- ☐  Public
The snippet can be accessed without any authentication.

File

Optionally name this file to add code highlighting, e.g. example.rb for Ruby.

1



Utiliser les snippets

◆ Snippets personnels

Visibilité définie

Indépendant de tout projet, accessible depuis le menu général

◆ Snippets liés à un projet

Accessible depuis le menu snippets du projet

◆ Fonctionnalités

gestion de commentaires, votes, téléchargement, intégration du lien

dev_admin > u-bootadmin > Extraits de code

Tous 2 Private 2 Internal 0 Public 0

Nouvel extrait de code



test3

\$3 · authored dans 7 heures by Anne

0 0

updated dans 7 heures



test2

\$2 · authored dans 7 heures by Anne

1 0

updated dans 7 heures





Gitlab et Gitlab CI

Sommaire

- 1 Runners
- 2 Pipelines et jobs
- 3 Le fichier .gitlab-ci.yml
- 4 Déploiement continu
- 5 Gestion avancée des variables
- 6 Docker



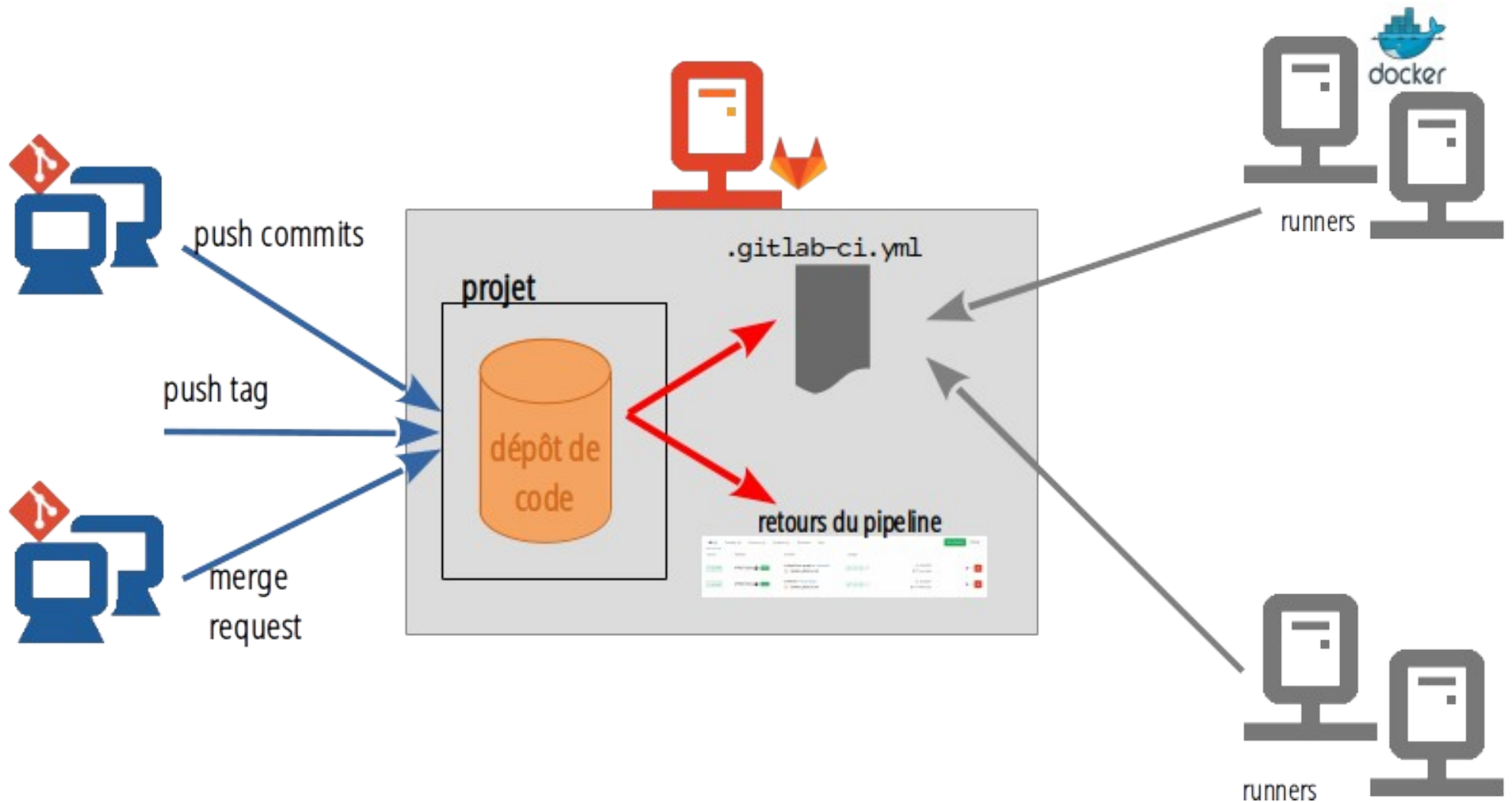
Introduction

◆ Définition

un fichier en yaml `.gitlab-ci.yml` à la racine du projet définissant les tests à réaliser
un ou plusieurs runners pour exécuter les tests
une page par projet relative à l'ensemble des builds

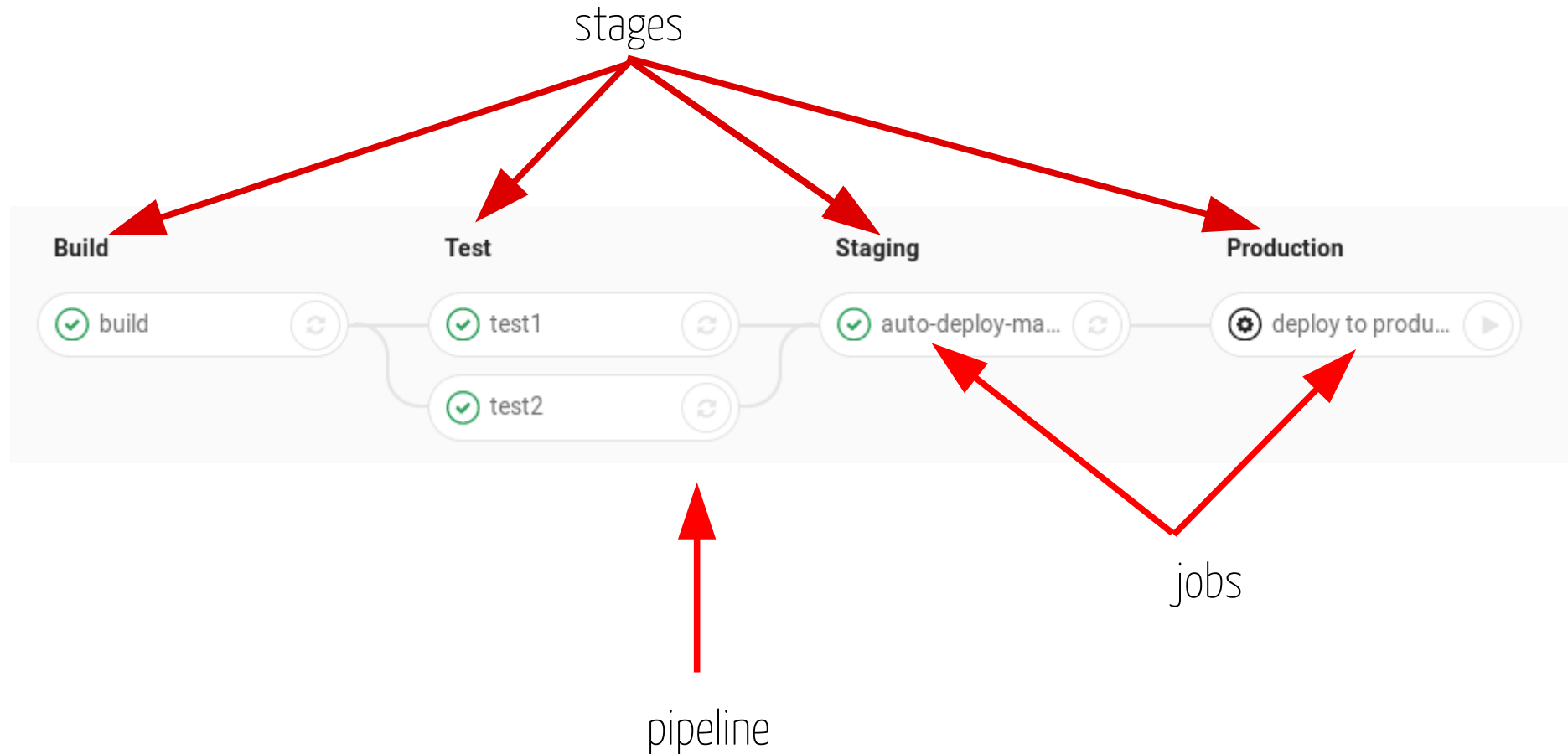


Présentation



Pipelines, jobs, stages

◆ <Project> / CI/CD / Pipelines



Les jobs d'un stage sont exécutés en parallèle, les stages sont exécutés séquentiellement



Vocabulaire

◆ Runner

Exécutable installé sur un serveur physique ou virtuel

Se connecte au serveur Gitlab pour récupérer les jobs à exécuter

Contenu des jobs : build, test, déploiement de code

◆ Job

Ensemble de tâches définies pour être exécutées par un runner

Défini par le fichier .gitlab-ci.yml

◆ Stage

Ensemble de jobs exécutés en parallèle - 3 par défaut : build , test, deploy

◆ Pipeline

Ensemble de stages exécutés consécutivement



1 Runners

Différents types de runners

◆ Shared runner

utilisable par tous les projets

jobs avec des pré-requis similaires entre différents projets, optimiser le nombre de runners

utilise une file de type fair queue

activer / désactiver des runners

◆ Specific runner

à utiliser pour des jobs ayant des pré-requis spécifique, ou des projets pour des demandes spécifiques (ex : authentification)

utilise une file de type FIFO

◆ Group runner

projets multiples appartenant à un groupe et qui ont accès à un ensemble de runners

utilise une file de type FIFO



Enregistrement du runner

◆ Enregistrement en ligne



Admin Area > **Runners**

A 'Runner' is a process which runs a job. You can set up as many Runners as you need. Runners can be placed on separate users, servers, even on your local machine.

Each Runner can be in one of the following states:

- **shared** - Runner runs jobs from all unassigned projects
- **group** - Runner runs jobs from all unassigned projects in its group
- **specific** - Runner runs jobs from assigned projects
- **locked** - Runner cannot be assigned to other projects
- **paused** - Runner will not receive any new jobs

Set up a shared Runner manually

1. [Install GitLab Runner](#)
2. Specify the following URL during the Runner setup: `http://192.168.100.137/` 
3. Use the following registration token during setup: `GmyvnsKTU-x8jYXEqcXk` 

Reset runners registration token

4. Start the Runner!

```
gitlab-runner register
```



Choisir un executor

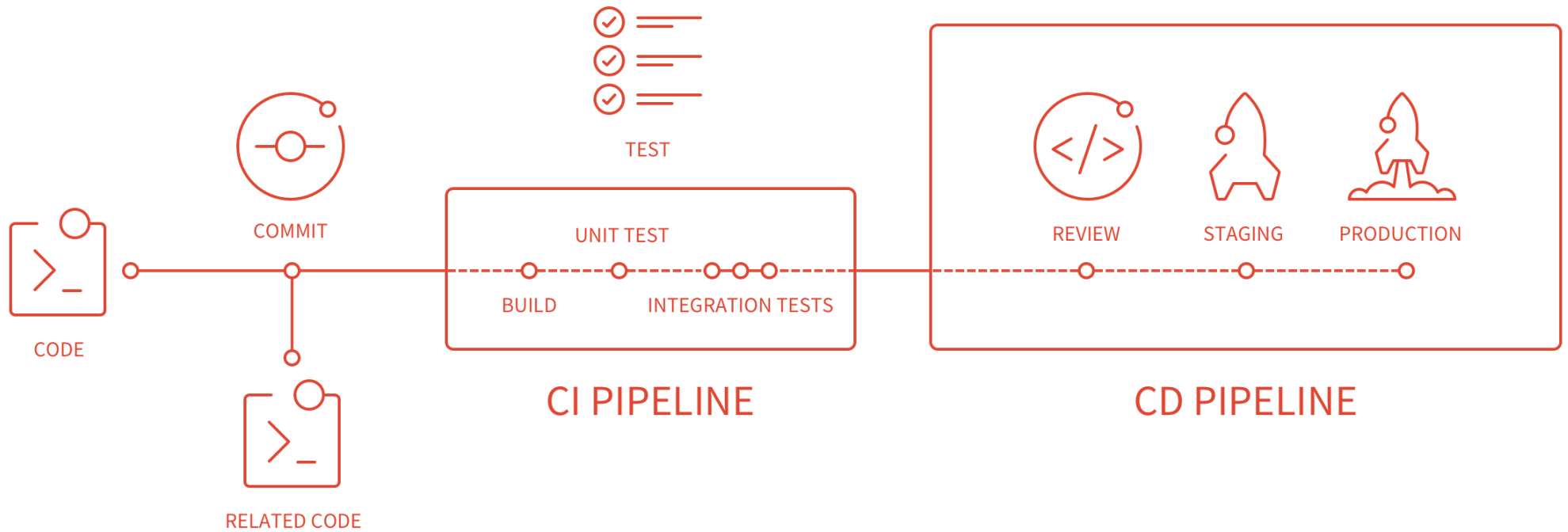
◆ Comparaison

Executor	SSH	VirtualBox	Docker	Kubernetes
Environnement de build nettoyé pour chaque job	✗	✓	✓	✓
Réutilisation d'un clone existant	✓	✗	✓	✗
Migration de la machine runner	✗	partiel	✓	✗
Environnements de build complexes	✗	✓	✓	✓
Debug des erreurs de build	facile	difficile	moyen	moyen



2 Pipelines et jobs

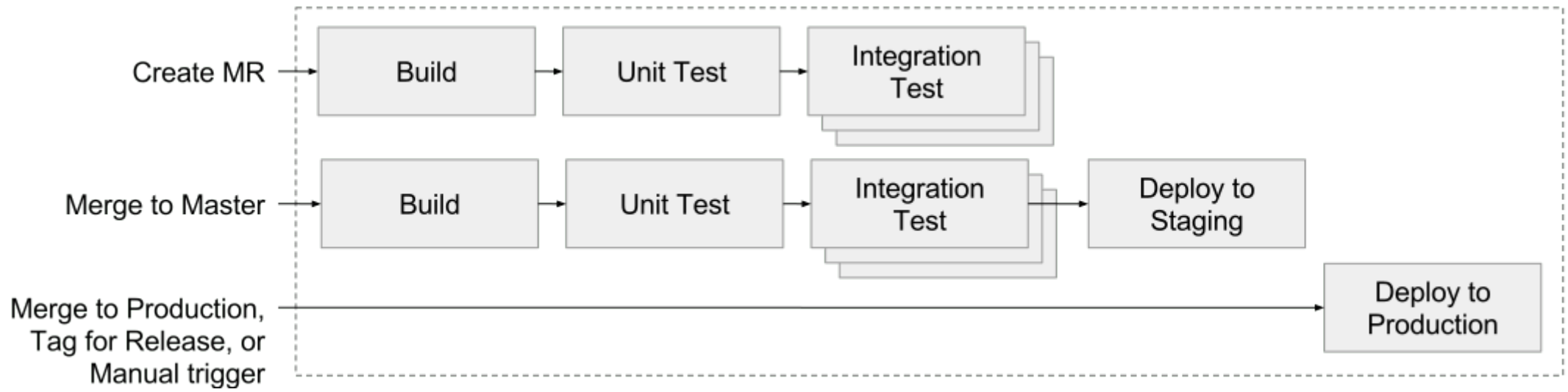
Pipelines



- ◆ intégration continue
- ◆ déploiement continu



Pipelines et workflows



- ◆ Workflow de branche (cf git flow)
- ◆ Workflow gitlab (branches de features et master)
- ◆ Workflow à base de fork



Pipelines et branches protégées

- ◆ Un utilisateur ayant les droits de réaliser une MR ou un push sur la branche peut :

- exécuter manuellement des pipelines
- exécuter de manière programmée des pipelines
- exécuter des pipelines au moyen de triggers
- relancer / annuler un job existant
- utiliser des runners protégés (utilisation de tags)



3 Le fichier .gitlab-ci.yml

Introduction

◆ Rôle du fichier

exécution de jobs, définis pour le projet, indépendamment les uns des autres
écrit en YAML
positionné à la racine du projet



Syntaxe du YAML

◆ Format générique

```
<variable>: « <valeur> »  
# ceci est un commentaire
```

◆ Liste

```
maliste:  
- element1  
- element2
```

indentations : 2 espaces par convention

◆ Dictionnaire / table de hashage

```
personne:  
  nom: doe  
  prenom: john
```

◆ Commentaires

```
# ceci est un commentaire
```



jobs : les bases du fonctionnement

- ◆ basé sur les codes retours

 - 0 succès

 - 1 échec

- ◆ un job non assigné est assigné au stage test par défaut
pré-requis minimum

- ◆ démarrage du job

 - chaque job démarre dans un environnement propre
checkout du dépôt git



jobs : syntaxe de base

```
<nom du job>:  
  script: "commande"
```

```
<nom du job>:  
  script:  
    - commande1  
    - commande2
```

- ◆ nom du job : unique et arbitraire - excepté des mots-clés réservés
(image, services, stages, types, before_script, after_script, variables, cache)
- ◆ script : mot-clé
pré-requis minimum
- ◆ nom du script
exécuter une commande directement ou faire appel à un script dans le dépôt de code.
- ◆ protéger une commande avec des quotes pour éviter toute interprétation YAML



Visualiser et exécuter un pipeline

exécution manuelle du pipeline

vérification de .gitlab-ci.yml

Run Pipeline CI Lint

Status	Pipeline	Commit	Stages	
passed	#7927123 by latest	pipeline-graph d4de4a5c Update .gitlab-ci.yml	✓ ✓ ✓ →	00:00:55 1 hour ago
passed	#7562143 by latest	master 4a2f619e Update .gitlab-ci.yml	✓ ✓ ✓ →	00:00:57 2 weeks ago

status du pipeline

Commit déclencheur

status des stages



Visualiser les jobs

sysadmin > hupstream-common > Jobs

All 34 Pending 0 Running 0 Finished 34

CI lint

Status	Job	Pipeline	Stage	Name	Coverage
passed	#475 ✓ master → f74caed8	#224 by	test	test:jessie	23:23 2 months ago
failed	#471 ✓ master → f74caed8	#224 by	test	test:jessie	02:25 2 months ago

```
- ntp was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting sshd to /builds/sysadmin/hupstream-common/tests/roles/sshd
- sshd was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting fail2ban to /builds/sysadmin/hupstream-common/tests/roles/fail2ban
- fail2ban was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting logcheck to /builds/sysadmin/hupstream-common/tests/roles/logcheck
- logcheck was installed successfully
- dependency etckeeper-module already pending installation.
- extracting rsyslog to /builds/sysadmin/hupstream-common/tests/roles/rsyslog
- rsyslog was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting sudo to /builds/sysadmin/hupstream-common/tests/roles/sudo
- sudo was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting certificates to /builds/sysadmin/hupstream-common/tests/roles/certificates
- certificates was installed successfully
- dependency etckeeper is already installed, skipping.
- extracting postfix to /builds/sysadmin/hupstream-common/tests/roles/postfix
- postfix was installed successfully
- adding dependency: saslauth
- dependency etckeeper-module already pending installation.
- extracting rsnapshot-client to /builds/sysadmin/hupstream-common/tests/roles/rsnapshot-client
- rsnapshot-client was installed successfully
- extracting etckeeper-module to /builds/sysadmin/hupstream-common/tests/roles/etckeeper-module
- etckeeper-module was installed successfully
- extracting saslauth to /builds/sysadmin/hupstream-common/tests/roles/saslauth
- saslauth was installed successfully
- dependency etckeeper is already installed, skipping.
```

test:jessie

Retry

Duration: 2 minutes 25 seconds

Timeout: 1h (from project)

Runner: gitlab-runner (#1)

Commit f74caed8

.gitlab-ci.yml: change directoy rights

✓ Pipeline #224 from master

test

✓ test:jessie

✓ test:stretch-backports

→ ✗ test:jessie



Vérification et correction

◆ pipeline editor

éditer le pipeline

vérifier la syntaxe YAML

visualiser le pipeline obtenu

vérifier le contenu du fichier en cours d'utilisation

Write pipeline configuration Visualize Lint View merged YAML

```
1 stages:
2   - deps
3   - cleaning
4   - lint
5   - build
6   - test
7   - deploy
8
9 hello_deps:
10  stage: deps
11  trigger:
12    project: ennael/projetshell
13    branch: master
14
15 hello_clean:
16  script: rm -rf bin/
17  rules:
18    - exists:
19      - bin/
20
21 hello_lint:
22  stage: lint
23  script:
24    - gcc -fsyntax-only hello.c
25  rules:
26    - if: '$CI_COMMIT_BRANCH =~ /^feature/'
```

Commit message:

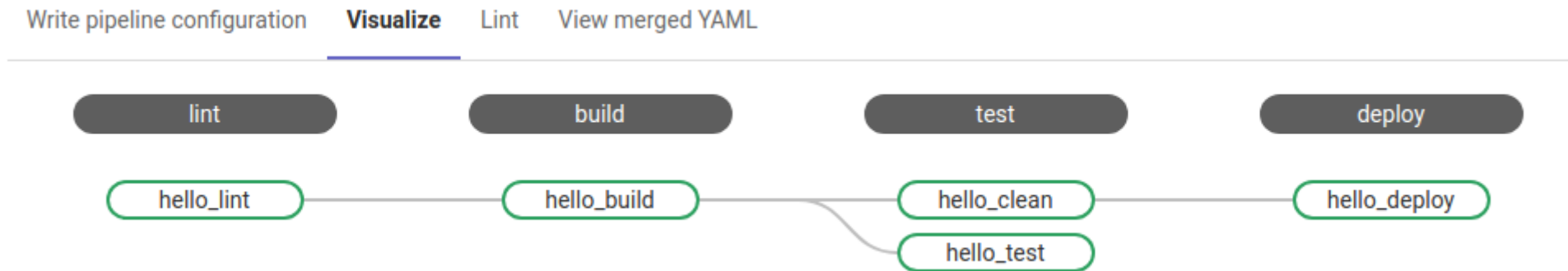
Target Branch:



Vérification et correction

◆ pipeline editor

visualisation du pipeline



visualisation des blocs et valeurs par défaut

Write pipeline configuration Visualize **Lint** View merged YAML

✓ **Status:**
Syntax is correct. CI configuration validated, including all configuration added with the `includes` keyword. [More information](#)

Parameter	Value
Lint Job - hello_lint	<code>gcc -fsyntax-only hello.c</code> When: on_success
Build Job - hello_build	<code>gcc -Wall hello.c -o hello</code> Only policy: branches, tags When: on_success



Stages

```
stages:  
  - build  
  - test  
  - deploy  
  
job 1:  
  stage: build  
  script: make build dependencies  
  
job 2:  
  stage: build  
  script: make build artifacts  
  
job 3:  
  stage: test  
  script: make test  
  
job 4:  
  stage: deploy  
  script: make deploy
```

◆ stages

définis globalement, au début du fichier

◆ stage :

permet de regrouper des jobs qui seront exécutés en parallèle, définit l'ordre proposé par le pipeline

◆ Par défaut : build, test, deploy



Exécution du job suivant : when

```
job
  script :
    - script.sh
  when: <valeur>
```

- ◆ **on_success**

le job sera exécuté uniquement si tous les jobs du stage précédent sont passés

- ◆ **on_failure**

le job sera exécuté uniquement si un job est en échec

- ◆ **always**

le job s'exécutera quoi qu'il se passe (même en cas d'échec)

- ◆ **manual**

le job s'exécutera uniquement par une action manuelle



Exemple

```
stages:  
  - build  
  - test  
  - report  
  - clean  
  
job:build:  
  stage: build  
  script:  
    - make build  
  
job:test:  
  stage: test  
  script:  
    - make test  
  when: on_success # s'exécutera uniquement si le job `job:build` passe  
  
job:report:  
  stage: report  
  script:  
    - make report  
  when: on_failure # s'exécutera si le job `job:build` ou `job:test` ne passe pas  
  
job:clean:  
  stage: clean  
  script:  
    - make clean # s'exécutera quoi qu'il se passe  
  when: always
```



Gestion des artefacts

◆ Objectif

Définit une liste de fichiers et répertoires à attacher à un job après une exécution
Envoyés au serveur Gitlab, téléchargeables via l'interface web
activés par défaut

◆ artifacts: paths

```
job:  
  artifacts:  
    paths:  
      - bin/*  
      - logs/*
```

réalise une archive avec les fichiers des répertoires bin/ et logs/



Gestion des artefacts

- ◆ modifier le nom de l'archive zip envoyée au serveur Gitlab (par défaut artifacts.zip)

```
package:  
  artifacts:  
    name: mon_archive
```

possibilité d'utiliser des variables globales Gitlab

```
package:  
  artifacts:  
    name: "$CI_JOB_NAME-$CI_COMMIT_REF_NAME"
```

- ◆ artifacts: untracked

```
package:  
  artifacts:  
    untracked: yes  
  paths:  
    - binaries/
```

booléen

crée une archive avec tous les fichiers untracked dans les chemins spécifiés



Gestion des artefacts

◆ artifacts: when

```
package:  
  artifacts:  
    when: <paramètre>
```

on_success upload des artefacts si le job est correctement exécuté (default)
on_failure upload des artefacts si le job échoue
always upload des artefacts quelque soit le statut du job

◆ artifacts: expire_in

défaut : artefacts conservés 30 jours
modifier le délai de conservation sur le serveur Gitlab



Gestion des artefacts

◆ Désactiver les artefacts

```
job:  
  stage: build  
  script: make build  
  dependencies: []
```

dépendances vides

◆ Récupérer les artefacts de certains jobs uniquement

```
job3:  
  script: monscript  
  artifacts:  
    paths:  
      - target/*.sh  
  dependencies :  
    - job2
```

Récupérer les artefacts d'un job donné

Seuls les artefacts du job build2 seront récupérés



before_script, after_script

```
before_script:  
  - global before script  
  
job:  
  before_script:  
    - execute this instead of global before script  
  script:  
    - my command  
  after_script:  
    - execute this after my script
```

- ◆ script(s) à exécuter avant ou après exécution du job

- ◆ lecture séquentielle

positionnés en haut du fichier, ils s'appliquent à tous les jobs

- ◆ before_script

s'exécute avant le job y compris les tâches de déploiement mais après restauration des artefacts

- ◆ after_script

s'exécute à la fin de l'exécution du job, y compris les jobs échoués. (ex : cleaning après finalisation des jobs)



Conditions d'exécution : only, except

```
job:  
  only:  
    - <keyword>...  
  except:  
    - <keyword>...
```

- ◆ établir une politique pour la génération des jobs
 - only branches ou tags pour lesquels les jobs seront exécutés
 - except branches ou tags pour lesquels les jobs ne seront pas exécutés
- ◆ utilisation des expressions régulières (ruby) : /<expression>/
- ◆ par défaut si non précisé :

```
only:  
  - branches  
  - tags
```



Conditions d'exécution : mots clés

- ◆ `branches`

quand un un push est effectué sur la branche spécifiée

- ◆ `tags`

quand un tag est créé

- ◆ `schedules`

par rapport à une planification à paramétrer dans l'interface web

- ◆ `merge_requests`

lors de la création ou la mise à jour d'une merge request



Conditions d'exécution

◆ only, except : exemples

```
<job>:
  only:
    - /^issue-.*$/i
  except:
    - branches
```

```
<job>:
  only:
    - branches
  except:
    - master
```

```
<job>:
  only:
    - tags
```



Conditions d'exécution

◆ only: refs, except: refs

```
deploy:  
  only:  
    refs:  
      - master
```

à utiliser si combinaison de références et variables

◆ only: variables, except: variables

```
deploy:  
  script: cap staging deploy  
  only:  
    refs:  
      - branches  
  variables:  
    - $RELEASE == "staging"  
    - $STAGING
```

```
end-to-end:  
  script: rake test:end-to-end  
  except:  
    variables:  
      - $CI_COMMIT_MESSAGE =~ /skip-end-to-end-tests/i
```



Conditions d'exécution

◆ only: changes, except: changes

```
job:  
  script: script.sh  
  only:  
    refs :  
      - merge_requests  
  changes:  
    - fic1  
    - rep1/*
```

Créer un job lorsqu'une modification est effectuée sur une liste de fichiers et poussée sur le serveur.

Ne pas combiner avec des conditions relatives aux tags

Ne fonctionne pas sur les branches et tags nouvellement créés



rules: if

◆ Fonctionnement

si vrai, ajoute le job au pipeline

si when: never, le job n'est pas ajouté au pipeline

si aucun if n'est vrai, le job n'est pas ajouté au pipeline

```
job:
  script: echo "Hello, Rules!"
  rules:
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME =~ /^feature/ && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME != $CI_DEFAULT_BRANCH'
      when: never
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME =~ /^feature/'
      when: manual
      allow_failure: true
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME'
```



rules: if

◆ Variable de clause : CI_PIPELINE_SOURCE

<code>merge_request_event</code>	Création d'un pipeline en cas de nouvelle merge request ou de mise à jour d'une merge request
<code>parent_pipeline</code>	Création d'un pipeline depuis un autre pipeline
<code>push</code>	Création d'un pipeline suite à un push (branches, tags)
<code>schedule</code>	Création d'un pipeline programmé
<code>web</code>	Création d'un pipeline suite à utilisation du bouton sur l'interface Gitlab



rules:if

◆ Principales variables utilisées avec " if "

- if: \$CI_COMMIT_TAG si un tag est poussé sur le serveur
- if: \$CI_COMMIT_BRANCH si un commit est poussé sur le serveur dans une branche
- if: '\$CI_COMMIT_BRANCH == "master"'
 si les commits sont poussés sur master
- if: '\$CI_COMMIT_BRANCH == \$CI_DEFAULT_BRANCH'
 si les commits sont poussés dans la branche par défaut
- if: '\$CI_COMMIT_BRANCH =~ /regex-expression/'
 si le commit est poussé dans la branche correspondante



Conditionner l'exécution de jobs

◆ Exemples

```
mybuild:
  script: build myscript
  rules:
    - if: '$CI_COMMIT_BRANCH == "master"'
      when: delayed
      start_in: '3 hours'
      allow_failure: true
```

```
myjob:
  script: echo "Hello world"
  rules:
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
      when: manual
      allow_failure: true
    - if: '$CI_PIPELINE_SOURCE == "schedule"'
```

```
job:
  script: echo "Hello, Rules!"
  rules:
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME =~ /^feature/ && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == "master"'
      when: always
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME =~ /^feature/'
      when: manual
      allow_failure: true
    - if: '$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME'
```

```
job:
  script: echo "Hello, Rules!"
  rules:
    - if: '$CI_PIPELINE_SOURCE == "schedule"'
      when: manual
      allow_failure: true
    - if: '$CI_PIPELINE_SOURCE == "push"'
```



rules:changes, rules:exist

◆ Exemple d'utilisation

```
docker build:  
  script: docker build -t my-image:$CI_COMMIT_REF_SLUG .  
  rules:  
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'  
      changes:  
        - Dockerfile  
      when: manual  
      allow_failure: true
```

```
job:  
  script: docker build -t my-image:$CI_COMMIT_REF_SLUG .  
  rules:  
    - exists:  
      - Dockerfile
```



Choisir le runner

◆ tags

```
job:  
  tags:  
    - ruby  
    - postgres
```

Assigner des tags à un runner (pendant l'installation ou après)

Déclencher un job sur un runner disposant du ou des tags spécifiés

Dispatcher les jobs selon des runners en fonction de tags



Gestion des caches

◆ Utilisation d'un cache pour les jobs

créer un cache de fichiers partagés pour les jobs pour accélérer l'exécution
peut générer des inconsistences

◆ Nettoyer le cache

interface gitlab : CI/CD > Pipelines > Clear Runner Cache
génère une nouvelle clé après le push

Administrator > hello_hapi > Pipelines

All 3 Pending 0 Running 0 Finished 3 Branches Tags				Run Pipeline	Clear Runner Caches	CI Lint
Status	Pipeline	Commit	Stages			
passed	#3 by latest	master - 56e16ca3 Update package.json		00:00:31 1 day ago		



Gestion des caches

◆ objectif

Liste des fichiers et répertoires qui doivent constituer un cache partagé entre plusieurs jobs
ex: réutiliser une ou plusieurs dépendances pour plusieurs jobs et accélérer le pipeline

◆ Fonctionnement

global ou par job

création du cache et utilisation par les jobs / pipelines suivants (sur le runner)

mise à jour éventuellement dans un job ultérieur



Artefacts vs Cache

◆ Cache

accélérer les exécutions d'un travail donné pour les pipelines suivants, en stockant les dépendances téléchargées afin qu'elles n'aient plus à être récupérées

◆ Artefact

transmettre les résultats d'un stage aux suivants mais pas au job suivant
ensemble de fichiers téléchargeables

◆ Ordre

ordre de restauration : cache puis artefacts

Les artefacts et les caches définissent leurs chemins d'accès par rapport au répertoire du projet et ne peuvent pas créer de lien vers des fichiers en dehors de celui-ci.



Artefacts vs Cache

◆ Cache

Désactivés s'ils ne sont pas définis globalement ou par job

Disponibles pour tous les jobs définis s'ils sont activés globalement

Utilisé dans les pipelines suivants par le job dans lequel il a été créé

Stocké sur le runner

◆ Artefact

Désactivés s'ils ne sont pas définis par job

Ne peut être activé que par tâche, pas globalement

Créés dans un pipeline et utilisables par les jobs suivants du même pipeline

Toujours téléchargés sur GitLab

Durée d'expiration pour contrôler l'utilisation du disque



Gestion des caches : déclaration

- ◆ Lister les répertoires et/ou fichiers constituant le cache

```
monscript:  
  script: test  
  cache:  
    paths:  
      - binaries/*.sh  
      - .config
```

- ◆ key

```
monscript:  
  script: test  
  cache:  
    paths:  
      - binaries/*.sh  
      - .config  
  key: "$CI_COMMIT_REF_SLUG"
```

définir un cache spécifique pour éviter d'en réécrire le contenu

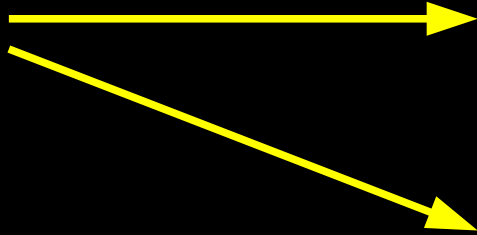
key: "\$CI_COMMIT_REF_SLUG" : associer le cache à une branche identifiée



Désactiver un job

◆ Désactiver un job

```
hidden_job:  
  script:  
    - run test
```



```
#hidden_job:  
#  script:  
#    - run test
```

```
.hidden_job:  
  script:  
    - run test
```

commenter toutes les lignes

ou

faire précéder le nom du job d'un point



Etendre un job


◆ extends

définit l'héritage depuis un autre job

```
.test:
  script: test.sh
  stage: test
  only:
    refs:
      - branches

main:
  extends: .tests
  script: monscript.sh
  only:
    variables:
      - $SCRIPT

main:
  script: monscript.sh
  stage: test
  only:
    variables:
      - $SCRIPT
    refs:
      - branches
```



Anchors

- ◆ Alléger l'écriture du fichier avec des templates

```
.template_test: &job_key # clé cachée définissant une ancre 'job_key'
  image: debian
  services:
    - postgres
    - postfix

test1:
  <<: *job_key           # Ajoute le contenu de 'job_key'
  script:
    - test1

test2:
  <<: *job_key           # Ajoute le contenu de 'job_key'
  script:
    - test2
```



Anchors, before_script et after_script

- ◆ Inclure une liste de commandes dans des jobs spécifiques

```
.defined_before: &defined_before
```

- command_before1
- command_before2

```
.defined_after: &defined_after
```

- command_after1
- command_after2

```
mon_job:
```

```
  before_script:
```

- command_before
- *defined_before

```
  script:
```

- command

```
  after_script:
```

- *defined_after



anchors, script, variables

◆ Définir des commandes à inclure dans " script "

```
.include_jobs: &include_jobs
```

- command1
- command2
- command3

```
mon_job:  
  script:  
    - *include_jobs  
    - command
```

◆ Définir des variables globales pour certains jobs

```
variables: &global_variables  
  VAR1: valeur1  
  VAR2: valeur2
```

```
mon_job  
  stage: clean  
  variables:  
    <<: *global_variables  
    GIT_STRATEGY: none  
  script: echo $VAR1
```



Définir des valeurs par défaut

◆ Le bloc "default "

Utilisable avec : image, services, before_script, after_script, tags, cache, artifacts, retry, timeout, interruptible

Peut être surchargé dans un job

Remplace la déclaration globale de chacune de ces directives

```
default:  
  tags: highmem  
  image: debian
```

```
mon_job:  
  script:  
    - command1  
    - command2
```



Options git push

◆ Options spécifiques à Gitlab

ne pas exécuter de pipeline

```
git push -o ci.skip
```

ou

```
git push --push-option=ci.skip
```



Debug tracing

◆ Mettre en place le debug de jobs

Pré-requis : les jobs ne doivent être visibles que pour les membres de l'équipe. Supprimer les jobs de debug avant de relâcher les permissions.

Activer le debug

```
job_name:  
  variables:  
    CI_DEBUG_TRACE: "true"
```



4 Déploiement continu

Spécificités de .gitlab-ci.yml

◆ En résumé : aucune !

Le déploiement est une commande ou un ensemble de commande comme les autres, exécutables dans Gitlab CI (« script »)

Nécessité de gérer des secrets

Possibilité de lier un déploiement à une branche

Conservation de l'historique de déploiement en cas de déploiement en environnements multiples – possibilité de rollback

Déploiements manuels dans les environnements critiques



Gestion des environnements

◆ Définition

Mise en évidence d'un environnement de déploiement




Accessible dans Opérations > Environnements

Anne > Plop > **Environnements**

Available **1** Stopped **0**

Enable review app

New environment

Environment	Deployment	Job	Commit	Updated	Auto stop in
production	#6 by 	deploy #509	 master  e7559146 Merge branch 'Dev' into 'master'	18 hours ago	 

↑
Environnement
de déploiement
Liste des
déploiements
déjà effectués

↑
Numérotation
du
déploiement

↑
Numéro de
job

↑
Événement
déclencheur
du
déploiement

↑
Relancer le déploiement











↑
Stopper l'environnement



Gestion des environnements

◆ Liste des déploiements

Anne > gitlab-ci-test > Environments > **production**

production							View deployment	Monitoring	Edit	Stop
Status	ID	Triggerer	Commit	Job	Created	Deployed				
✓ success	#2		 v3.0  1cc155a7  Ajout url	deploy_hello (#...	1 day ago	1 day ago				
✓ success	#1		 v2.0  db6adbcb  Ajout des environnements	deploy_hello (#...	1 day ago	1 day ago				

Relancer le déploiement
Rollback d'un déploiement

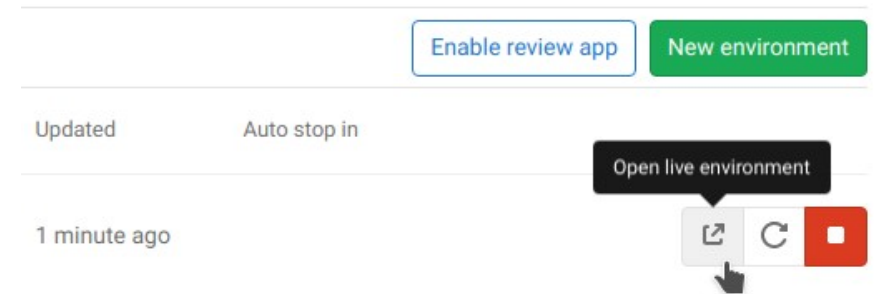


Déclaration d'un environnement

◆ Dans le stage de déploiement

Déclaration du nom de l'environnement

Créé s'il n'existe pas



```
stages:
  - test
  - deploy

test:
  stage: test
  script: echo "Running tests"

deploy_staging:
  stage: deploy
  script:
    - echo "Deploy to production server"
environment:
  name: production
  url: <url>
only:
  - master
```

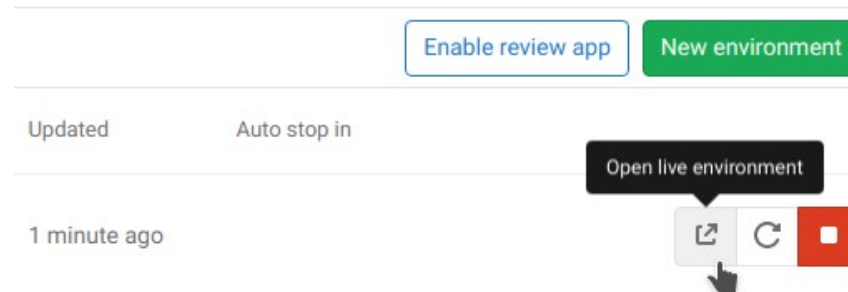


Compléter la définition

◆ url

Crée un bouton pointant vers l'url définie

```
deploy_staging:
  stage: deploy
  script:
    - echo "Deploy to production server"
  environment:
    name: production
    url: <url>
  only:
    - master
```



6 Gestion des variables

Utilisation de variables

variables:

```
POSTGRES_DATABASE: postgres  
POSTGRES_PASSWORD: password  
DB_HOST: postgres  
DB_USERNAME: root
```

- ◆ variables globales ou par job
- ◆ tableau de paires clés/valeurs, chaînes de caractères ou entiers
- ◆ convention : clé en majuscule
- ◆ les variables locales surchargent les variables globales
- ◆ désactiver les variables globales pour un job :

```
<nom_job>:  
  variables: {}
```



Introduction


- ◆ Où positionner des variables ?
 - sur Gitlab : le fichier .gitlab-ci.yml
 - globalement (en-tête de fichier), par job
- ◆ sur le runner : dans le fichier de configuration du runner config.toml
- ◆ au niveau du projet et/ou du groupe
- ◆ sur un pipeline manuel

```
variables:  
  DATABASE_URL: "postgres://postgres@postgres/my_database"
```



Variables de groupe et de projet

◆ Liste de précedence

- 
- 1 - variables trigger ou de pipeline programmé
 - 2 - variables projets ou variables protégées
 - 3 - variables de groupe ou variables protégées
 - 4 - variables définies au niveau du job (YAML)
 - 5 - variables définies au niveau global (YAML)
 - 6 - variables de déploiement
 - 7 - variables pré-définies



Variables de groupe et de projet

◆ Groupe

<groupe> > settings > CI/CD > Variables

◆ Projet

<groupe> > settings > CI/CD > Variables

Variables ?

Collapse

Variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. You can use variables for passwords, secret keys, or whatever you want.

Protected



Save variables

Hide values



Variables de pipeline

◆ Exécution manuelle d'un pipeline

Run Pipeline

Run for

master

Existing branch name or tag

Variables

Variable



Input variable key

Input variable value

Specify variable values to be used in this run. The values specified in [CI/CD settings](#) will be used by default.

Run Pipeline

Cancel



Expressions

◆ Utilisation

avec only / except : limiter les jobs créés dans un pipeline après un push évaluée avant la création du pipeline

```
deploy:  
  script: cap staging deploy  
  only:  
    variables:  
      - $RELEASE == "staging"
```



Expressions

◆ Syntaxe

chaînes égales

```
$VARIABLE == "valeur"
```

présence d'une variable non définie

```
$VARIABLE == null
```

présence d'une variable vide

```
$VARIABLE == ""
```

comparer 2 variables

```
$VARIABLE_1 == $VARIABLE_2
```

matcher un pattern

```
$VARIABLE =~ /^contenu.*\/
```



Variables protégées

◆ Définition

utilisables seulement dans les pipelines exécutés sur des branches ou des tags protégés.
non accessibles aux autres pipelines

◆ Modification

variable de groupe : <groupe> > settings > CI/CD > Variables

variable de projet : <projet> > settings > CI/CD > Variables

Variables ?Collapse

Variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. You can use variables for passwords, secret keys, or whatever you want.

Protected ☒

