

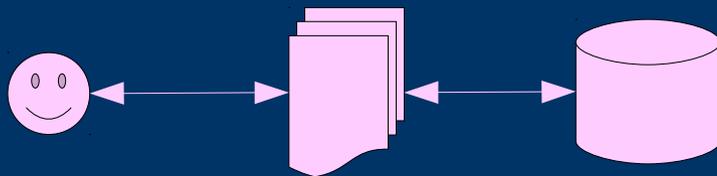
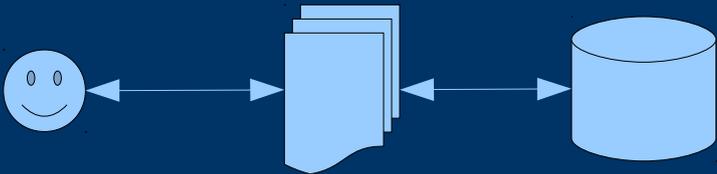
Travailler à plusieurs avec GitLab ou GitHub

Bruno Mermet
Université du Havre
2018

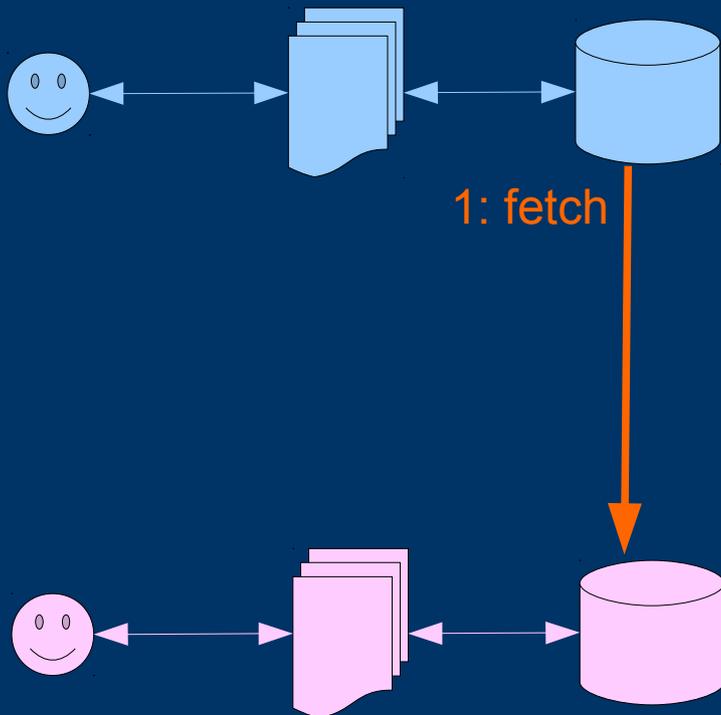
Plan

- Présentation générale du principe
- Démarche avec gitlab

Travailler à 2 avec git

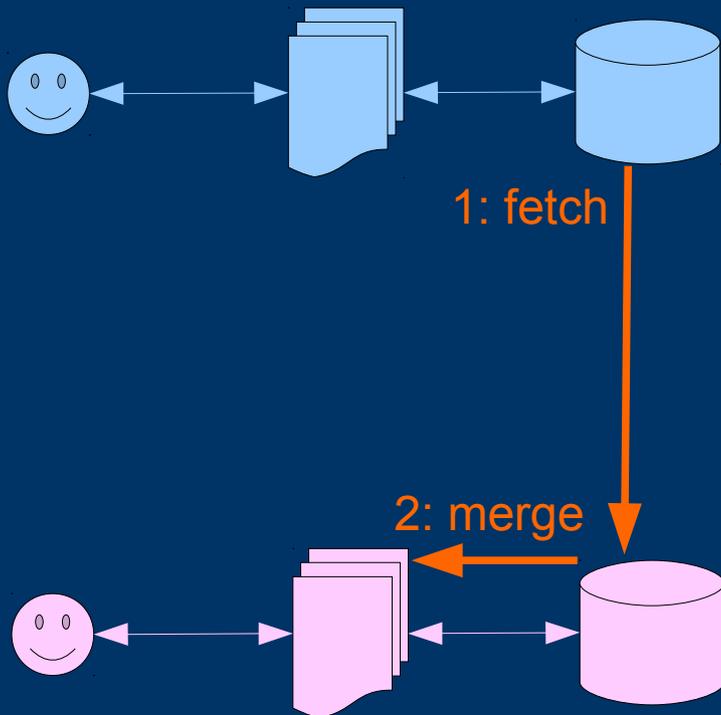


Travailler à 2 avec git



Fetch :
Les branches distantes
sont importées dans le
dépôt local avec un nom de
la forme
dépôtDistant/nomBranche

Travailler à 2 avec git

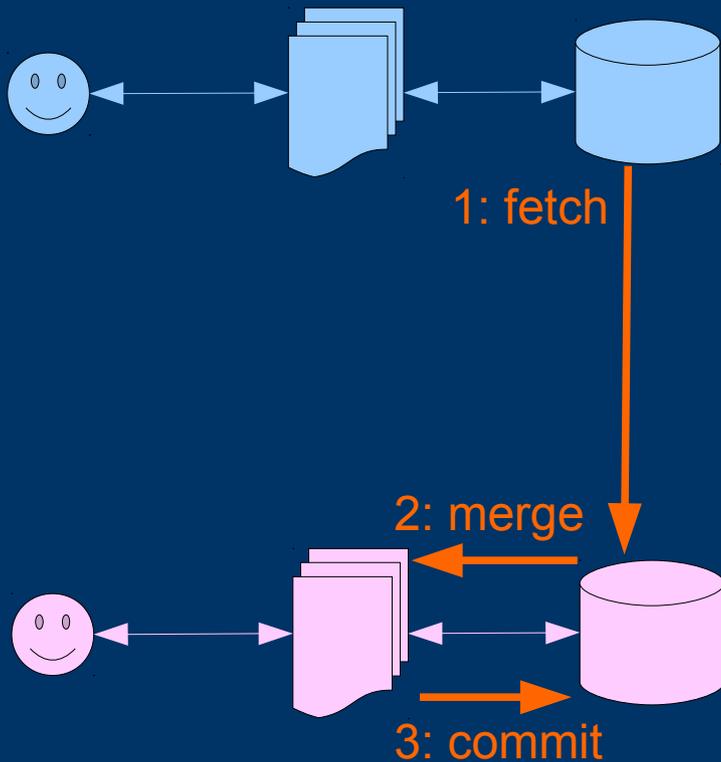


Fetch :

Merge :

On fusionne dans la
branche courante la
branche distante de même
nom

Travailler à 2 avec git



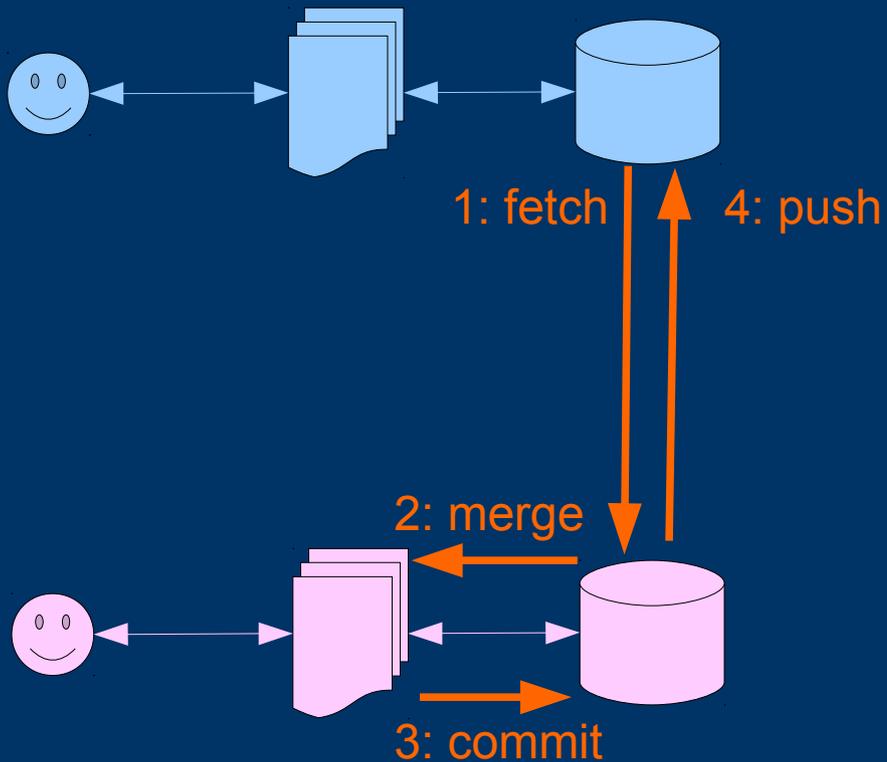
Fetch :

Merge :

Commit :

On archive le résultat de la fusion

Travailler à 2 avec git



Fetch :

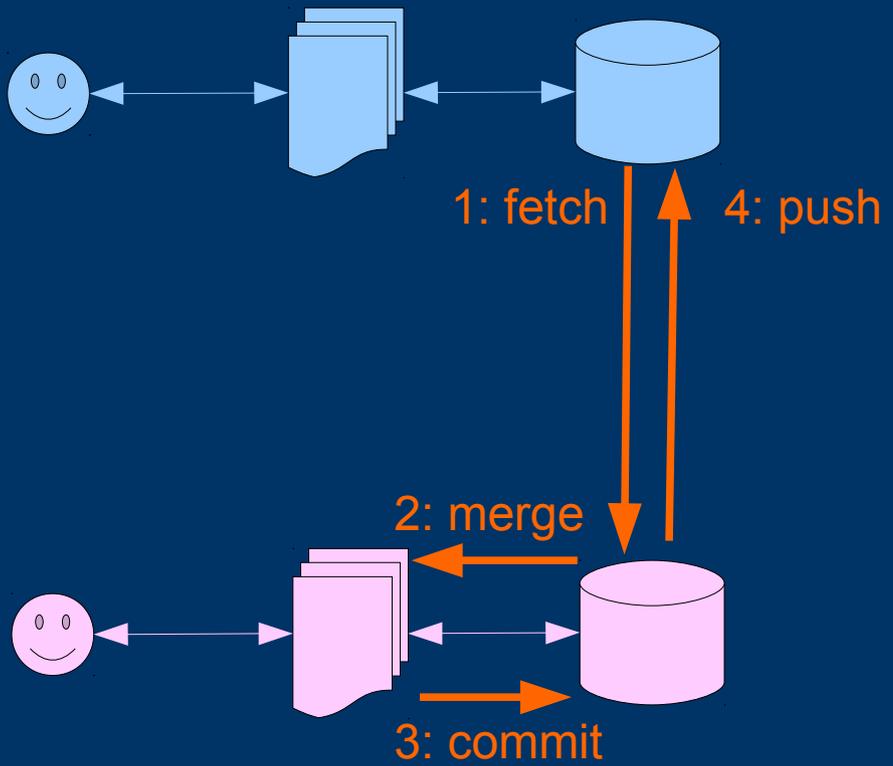
Merge :

Commit :

Push :

On transfère sur le dépôt distant l'historique de la branche courante

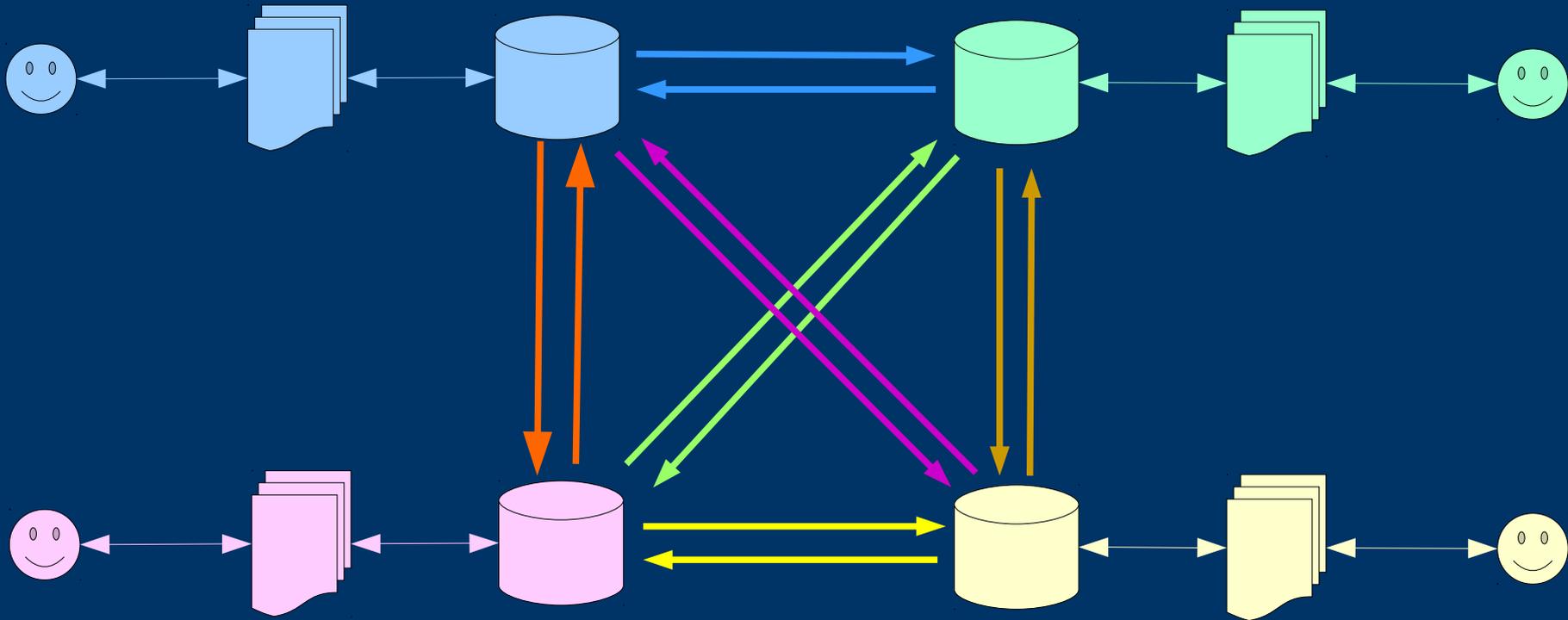
Travailler à 2 avec git



Fetch :	Pull
Merge :	
Commit :	
Push :	
On transfère sur le dépôt distant l'historique de la branche courante	

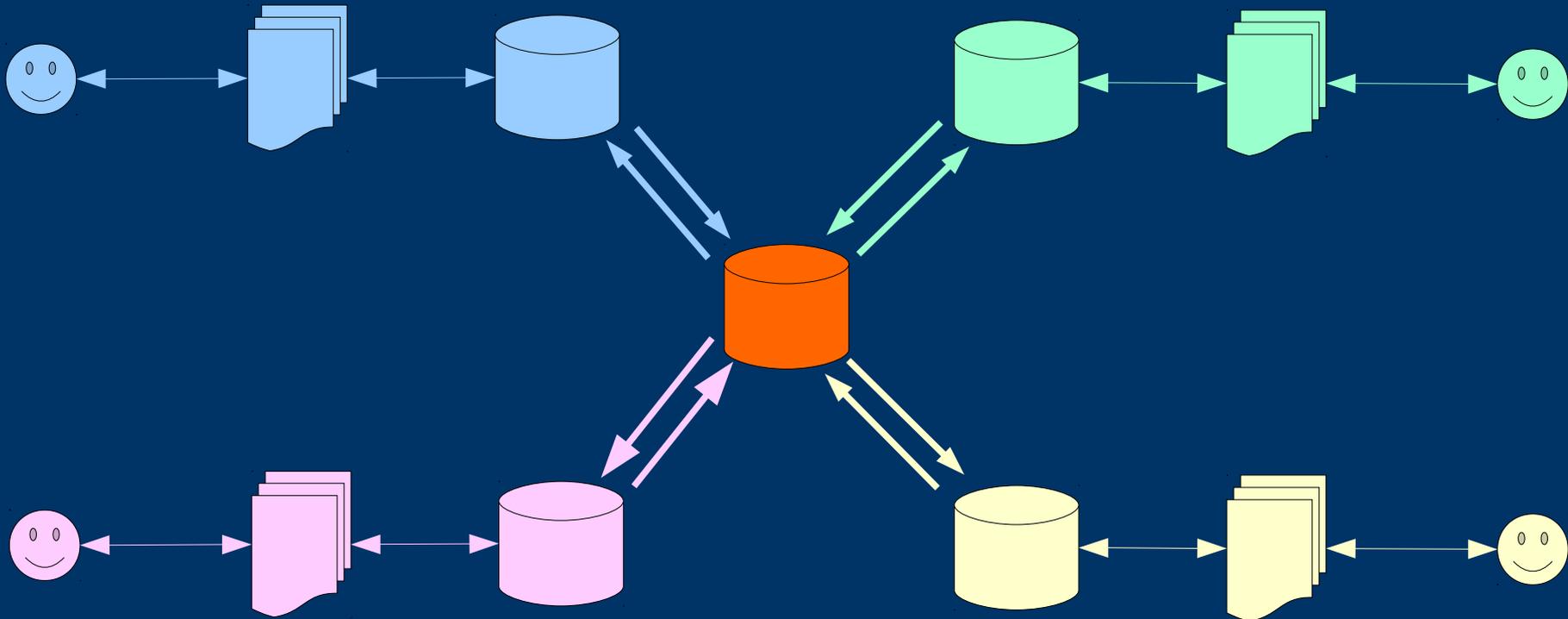
Travailler à plusieurs avec git

1. Version complètement décentralisée



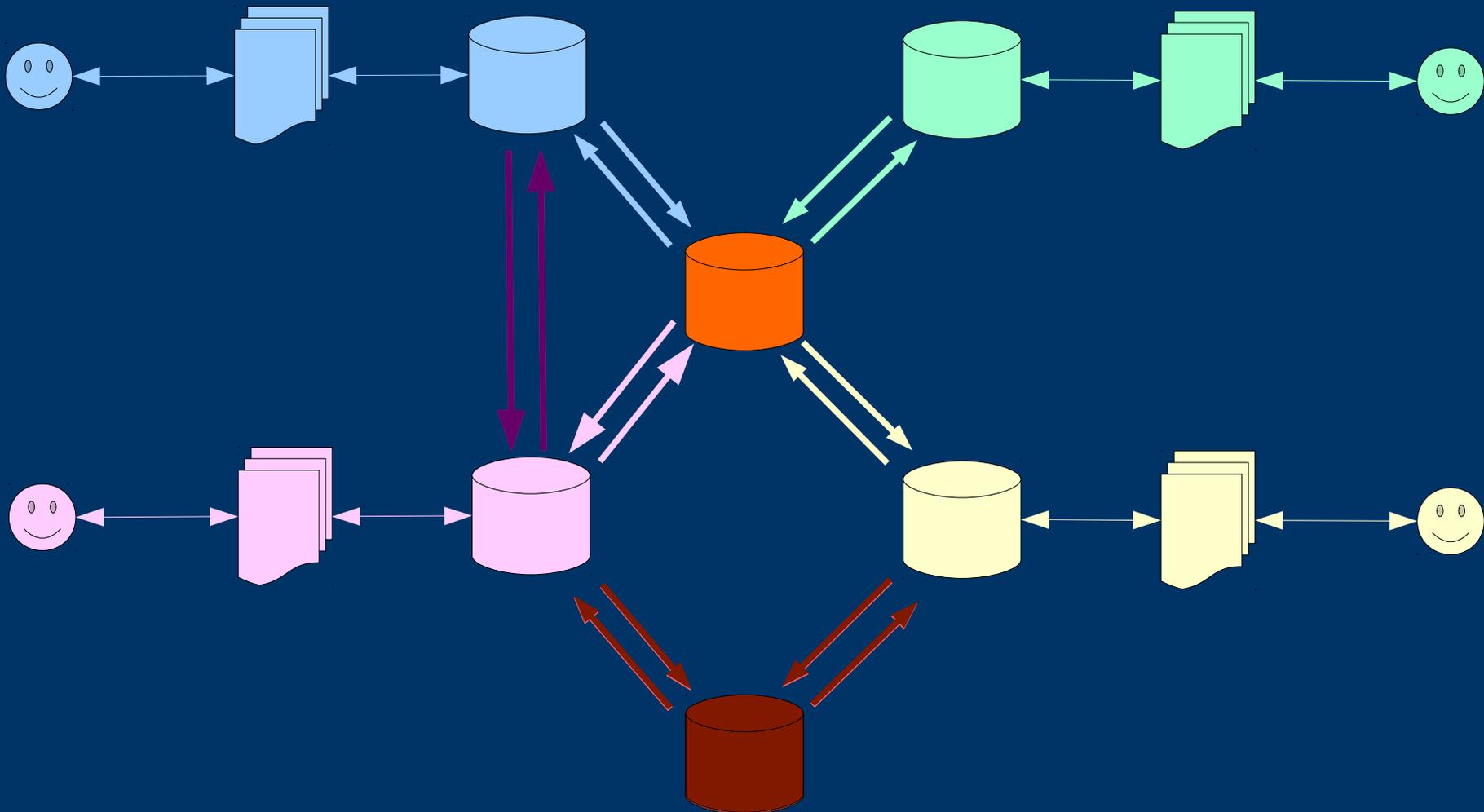
Travailler à plusieurs avec git

2. Version complètement centralisée



Travailler à plusieurs avec git

2. Version complètement centralisée



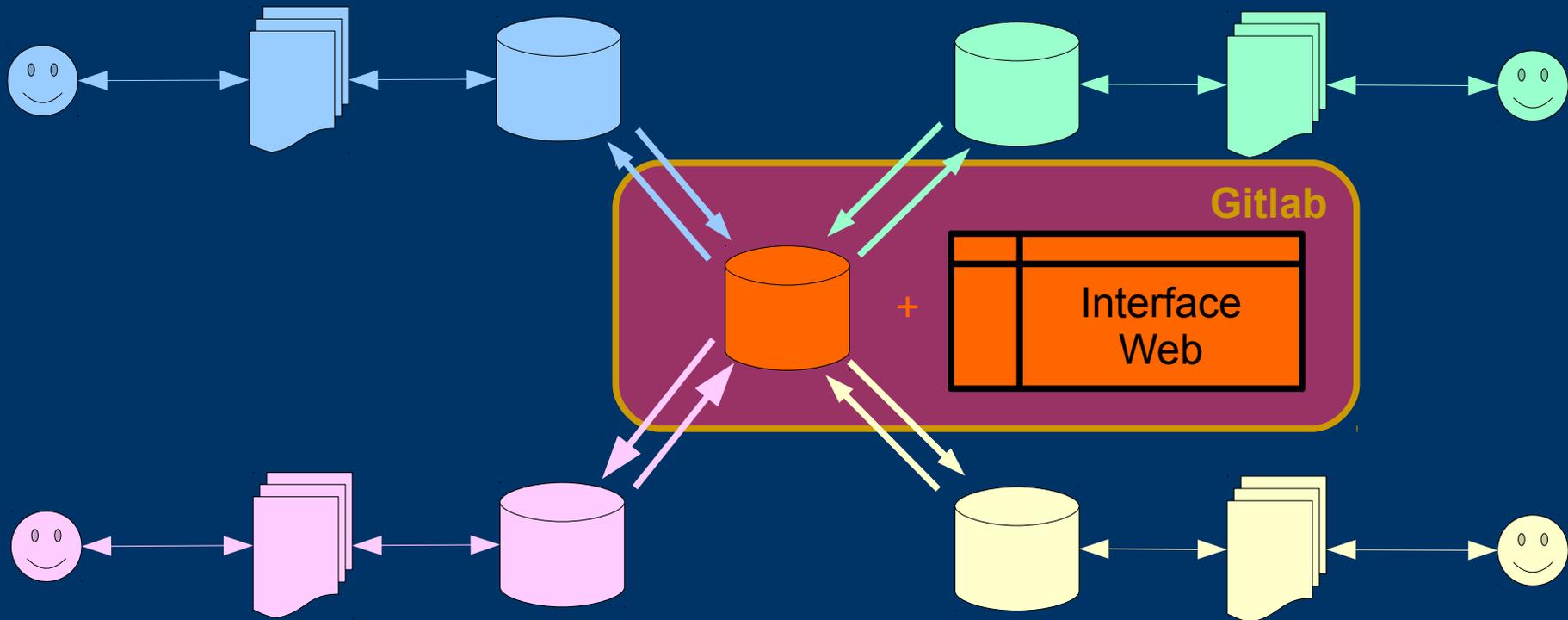
Travailler à plusieurs avec git

3. En résumé...

- On travaille toujours avec 2 dépôts
 - Le dépôt local
 - Un dépôt « distant » (*remote*) auquel on associe un nom
- Les 4 principales commandes git permettant d'interagir avec un dépôt distant sont :
 - **git fetch** : on récupère en local le contenu du dépôt distant
 - **git pull** : équivalent d'un `git fetch` suivi d'un `git merge`
 - **git push** : on transfère l'état actuel de la branche courante (et son historique) vers le dépôt distant
 - **git clone** : on crée un dépôt local à partir d'un dépôt distant

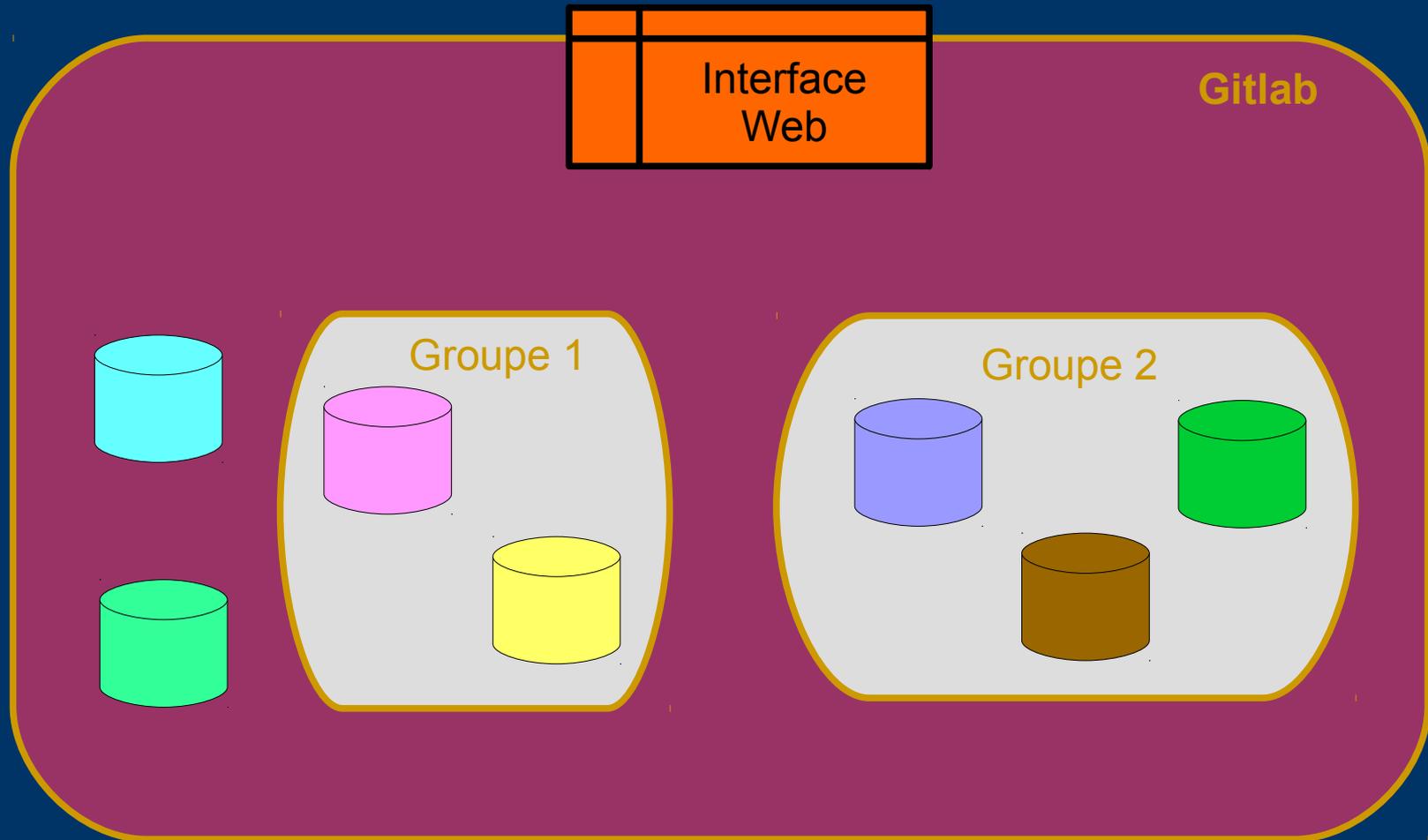
Travailler à plusieurs avec git

4. Faciliter les choses



Gitlab et consorts (Github, Gitbucket)

Structure générale



Gitlab et Github

Résumé

- Fonctionnalités offertes
 - Gestion de multiples dépôts git, permettant de faciliter le travail à plusieurs
 - Interface Web pour interagir avec le dépôt git
 - Gestion de « pull request » ou « merge requests »
 - Approche de l'intégration continue
- Utilisation
 - Soit depuis gitlab.com / github.com
 - Soit via une installation locale (gratuite pour gitlab)

Travailler à plusieurs avec un dépôt commun

- Principe de base
 - Dépôt « en ligne » accessible par tous, au moins occasionnellement
 - Chacun « clone » le dépôt commun sur sa machine
 - Périodiquement :
 - Mise à jour de son dépôt local à partir du dépôt commun
 - Transfert de versions stables de son travail dans le dépôt commun
- Principe « évolué »
 - Faire un « fork » du dépôt commun et travailler avec ce dépôt personnel distant
 - Faire des « pull request » vers le dépôt commun lorsqu'on a atteint une version stable sur son dépôt personnel distant

Gestion de la synchronisation entre dépôts

- Un dépôt est constitué
 - D'un ensemble de « versions », identifiées par un « hash »
 - D'un ensemble de « références » : branches et étiquettes
- Localement, un dépôt distant est identifié par un nom
- Une branche *b* du dépôt distant *d* est suivie localement par une branche de suivi *d/b*
- L'action « fetch » récupère dans le dépôt local les versions disponibles sur un dépôt distant et met à jour ou crée les branches de suivi
- L'action « pull » fait un « fetch » puis un « merge » dans une branche locale *b* à partir de la branche de suivi associée (*b/d* par défaut)
- L'action « push » transfère la branche locale sur le serveur. Si une branche de même nom n'existe pas sur le serveur, elle est créée. Sinon, si un « fast-forward » peut être effectué, la branche sur le serveur est mise à jour. Sinon, le « push » est refusé.

Commencer avec gitlab

- Se rendre sur la page <https://www-apps.univ-lehavre.fr/forge>
- Se connecter avec le C.A.S. de l'université
 - Crée un *groupe* personnel
- Ajouter une clé SSH (menu en haut à droite, option « settings » puis menu à gauche, option « SSH Keys »)
 - éventuellement, sur sa machine
 - ssh-keygen
 - Ssh-add
- Configurer la langue (Settings → Profile → Preferred Language)
- Créer éventuellement un ou plusieurs *groupe(s)* et y ajouter éventuellement des membres (menu « Groups », bouton Nouveau Groupe)
- Pour démarrer un projet, créer un dépôt dans le groupe adéquat

Créer un projet dans Gitlab (1) (Projects → New Project)

Nom du groupe

Optionnel... donc
indispensable !

Nom du dépôt

The screenshot shows the 'New Project' form in GitLab. It has three tabs: 'Blank project' (selected), 'Create from template', and 'Import project'. The 'Project path' field contains 'https://www-apps.univ-lehavre.fr/forge/' and a dropdown menu showing 'mermetb'. The 'Project name' field contains 'my-awesome-project'. Below these fields is a link: 'Want to house several dependent projects under the same namespace? [Create a group](#)'. The 'Project description (optional)' field is empty with a placeholder 'Description format'. The 'Visibility Level' section has three radio buttons: 'Privé' (selected), 'Interne', and 'Public'. The 'Privé' option has a lock icon and the text 'L'accès au projet doit être explicitement accordé à chaque utilisateur.' The 'Interne' option has a shield icon and the text 'Votre projet peut être accédé par n'importe quel utilisateur authentifié-e.' The 'Public' option has a globe icon and the text 'Votre projet peut être accédé sans aucune authentification.' At the bottom, there are two buttons: 'Create project' (green) and 'Cancel' (white).

Créer un projet dans GitLab (2)

Retour de GitLab (1)



URI SSH ou
HTTPS du dépôt

SSH : pas
d'identifiant à saisir
grâce à la clé SSH

HTTPS : connexion
possible depuis
l'extérieur de
l'université

Créer un projet dans GitLab (2)

Retour de GitLab (1)



URI SSH ou
HTTPS du dépôt

Git global setup

```
git config --global user.name "Bruno Mermet"  
git config --global user.email "bruno.mermet@univ-lehavre.fr"
```

Rappels sur la
configuration de git si
ce n'est pas déjà fait

Create a new repository

```
git clone git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git  
cd MonProjet  
touch README.md  
git add README.md  
git commit -m "add README"  
git push -u origin master
```

Exemple d'utilisation si
ce dépôt est là pour
héberger un nouveau
projet

Créer un projet dans GitLab (2)

Retour de GitLab (2)

Existing folder

```
cd existing_folder
git init
git remote add origin git@forgebl.univ-lehavre.fr:mermetb/MonProjet.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Consignes pour utiliser ce dépôt pour héberger un projet qui existe déjà en local

Existing Git repository

```
cd existing_repo
git remote rename origin old-origin
git remote add origin git@forgebl.univ-lehavre.fr:mermetb/MonProjet.git
git push -u origin --all
git push -u origin --tags
```

Consignes pour utiliser ce dépôt pour héberger un projet qui est déjà hébergé sur un autre dépôt

Démarrer un nouveau projet

1. Cloner le dépôt GitLab

- `git clone git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git`
- `tree`

```
├── MonProjet
└── .git/
```

- `cd MonProjet ; git remote -v`

```
origin  git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git (fetch)
origin  git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git (push)
```

- `git branch`

- `git log`

```
fatal: bad default revision 'HEAD'
```

Démarrer un nouveau projet

1. Cloner le dépôt GitLab

- `git clone git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git`
- `tree`

```
├── MonProjet
└── .git/
```

- `cd MonProjet ; git remote -v`

```
origin  git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git (fetch)
origin  git@forgeb1.univ-lehavre.fr:mermetb/MonProjet.git (push)
```

Origin : nom attribué (par défaut) au dépôt GitLab que l'on vient de cloner.

```
...ult revision 'HEAD'
```

Démarrer un nouveau projet

2. Initialiser la branche master

- touch README.md
- git add .
- git commit -m « création fichier ReadMe »

```
[master (commit racine) 30d9696] création fichier ReadMe
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
```

- git branch -vv

```
* master 30d9696 [origin/master: disparue] création fichier ReadMe
```

La branche « master » actuelle est sensée suivre la branche « master » du dépôt initial, nommé « origin », mais cette branche distante n'existe pas encore

Démarrer un nouveau projet

3. Transférer une première version sur le serveur

- `git push -u origin master`

Nom de la branche devant être transférée sur le dépôt (si non spécifié, il s'agit de la branche courante). Donc inutile ici.

Nom du serveur vers lequel faire le transfert (si non spécifié, transfert fait vers le serveur par défaut ; voir `git remote -v`). Donc inutile ici.

Raccourci de « `--set-upstream` » : permet de fixer la branche distante suivi par la branche locale si ce n'est déjà fait. Donc inutile ici.

Démarrer un nouveau projet

3. Transférer une première version sur le serveur

- git push

```
Counting objects: 3, done.  
Writing objects: 100% (3/3), 220 bytes | 0 bytes/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://www-apps.univ-lehavre.fr/forge/mermetb/MonProjet.git  
* [new branch]    master -> master
```

- git branch -vv

```
* master 30d9696 [origin/master] creation fichier ReadMe
```

Page d'accueil d'un projet (après création d'une première version)

Bruno Mermet > MonProjet > Details



MonProjet ▾

Projet pour illustrer le fonctionnement de GitLab

☆ Mettre en favori 0
🔗 Fourcher 0
SSH git@forgeb1.univ-lehavre.fr:merm
📄
📄
+

🔔 Surveillé ▾

Fichiers (72 ko)
Validation (1)
Branche (1)
Étiquette (0)
LisezMoi
Ajouter un journal des modifications
Ajouter une licence
Ajouter un guide de contribution
Mettre en place l'intégration continue (CI)

master ▾
MonProjet / +
Historique
🔍 Rechercher un fichier
👤 ▾



edition README

Bruno Mermet a validé Il y a 3 minutes

431f8a99 📄

Nom	Dernière validation	Dernière mise à jour
📄 README.md	edition README	Il y a 3 minutes

📄 README.md

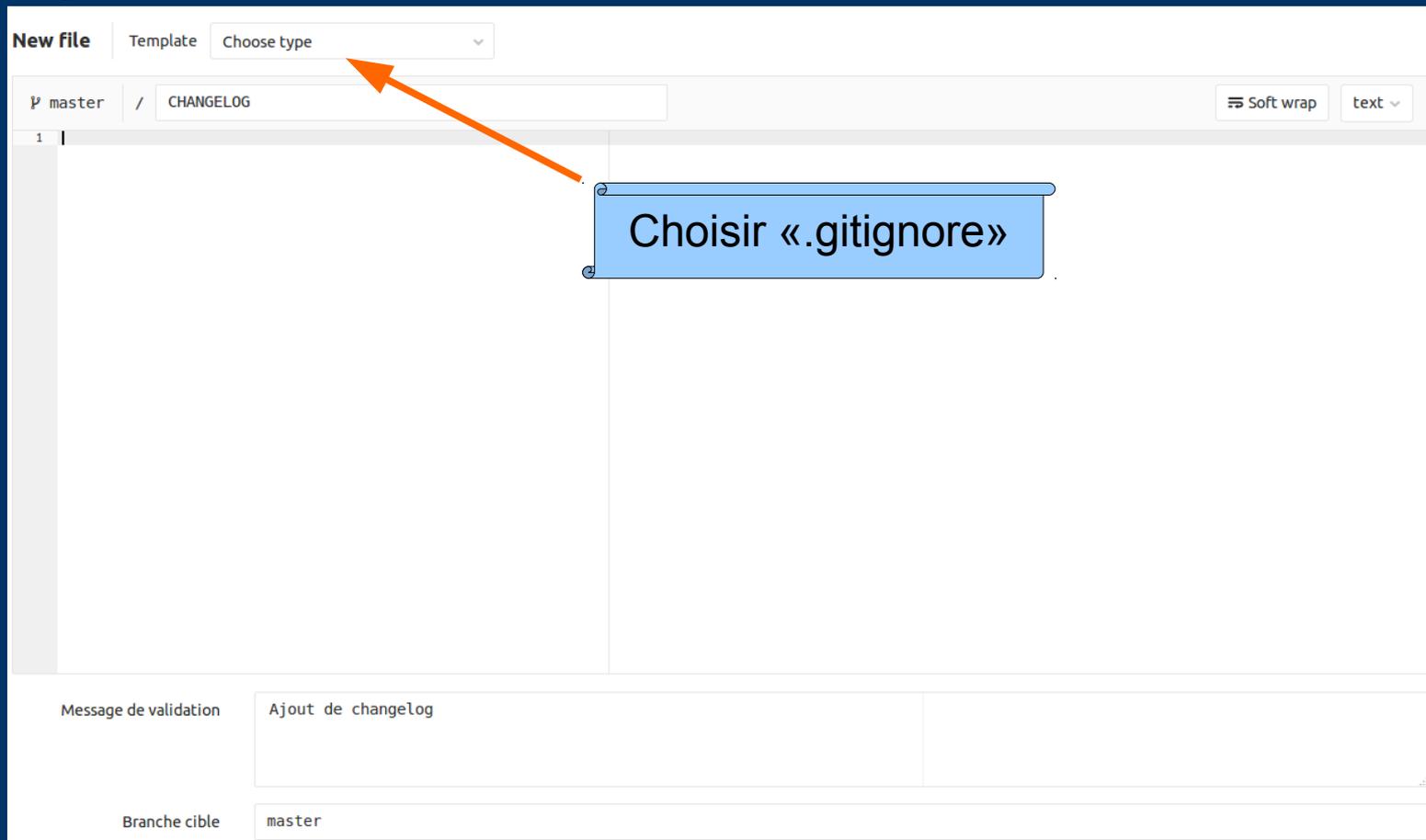
Titre

Création d'un fichier « .gitignore » standard

- Depuis la page d'accueil du projet, cliquer sur « Ajouter un journal des modification »

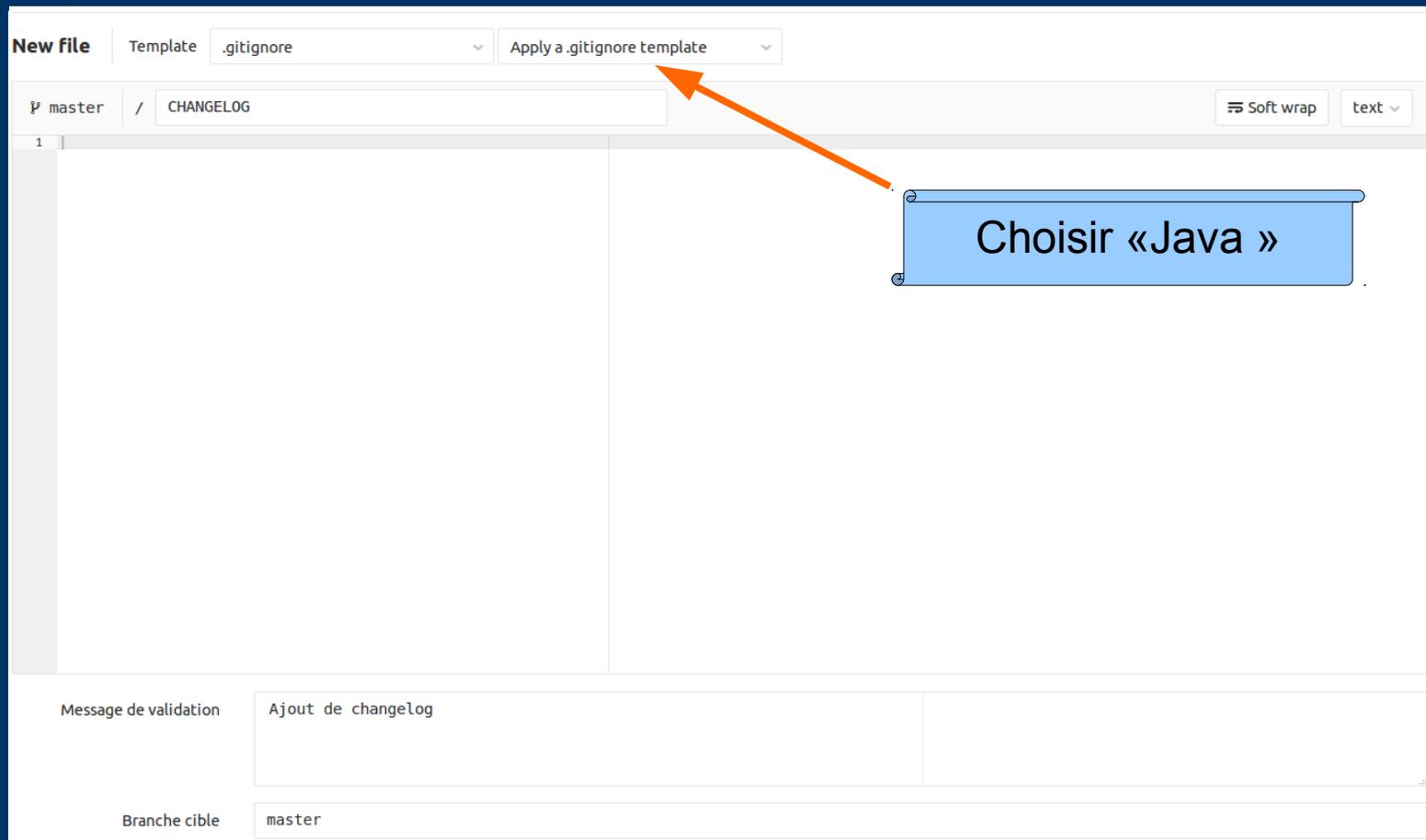
Création d'un fichier « .gitignore » standard

- Depuis la page d'accueil du projet, cliquer sur « Ajouter un journal des modification »



Création d'un fichier « .gitignore » standard

- Depuis la page d'accueil du projet, cliquer sur « Ajouter un journal des modification »



Création d'un fichier « .gitignore » standard

- Depuis la page d'accueil du projet, cliquer sur « Ajouter un journal des modification »

The screenshot shows the 'New file' dialog in an IDE. The 'Template' dropdown is set to '.gitignore' and the language is 'Java'. The dialog displays a standard .gitignore file template with the following content:

```
1 # Compiled class file
2 *.class
3
4 # Log file
5 *.log
6
7 # BlueJ files
8 *.ctxt
9
10 # Mobile Tools for Java (J2ME)
11 .ntj.tmp/
12
13 # Package Files #
14 *.jar
15 *.war
16 *.ear
17 *.zip
18 *.tar.gz
19 *.rar
20
21 # virtual machine crash logs, see http://www.java.com/en/download/help/error_hotspot.xml
22 hs_err_pid*
23
```

At the bottom of the dialog, there is a 'Message de validation' field containing 'Ajout de changelog'. Below this is a 'Branche cible' field set to 'master'. A blue callout box with the text 'Cliquer pour valider' has an orange arrow pointing to the 'Commit changes' button at the bottom left. A 'Cancel' button is located at the bottom right.

Création d'un fichier « .gitignore » standard

The screenshot shows the GitLab interface for a project named 'MonProjet'. At the top, there is a user profile icon with the letter 'M' and the project name 'MonProjet' with a small bell icon. Below this, it says 'Projet pour illustrer le fonctionnement de GitLab'. There are several utility buttons: 'Mettre en favori' (0), 'Fourcher' (0), 'SSH' (dropdown), 'git@forgeb1.univ-lehavre.fr:merm' (file icon), a download icon, and a plus icon. A 'Surveillé' (watched) button is also present.

Below the utility buttons, there are several action buttons: 'Validations (2)', 'Branche (1)', 'Étiquette (0)', 'LisezMoi', 'Ajouter un journal des modifications', 'Ajouter une licence', 'Ajouter un guide de contribution', and 'Mettre e...'. The 'Ajouter un journal des modifications' button is highlighted with a dashed border.

The main content area shows the current branch 'master' and the project name 'MonProjet / +'. There are buttons for 'Historique', 'Rechercher un fichier', and a download icon. A recent commit is shown: 'Ajout de changelog' by Bruno Mermet, validated 8 minutes ago, with commit hash '0ec3ad47' and a file icon.

Below the commit, there is a table listing files in the repository:

Nom	Dernière validation	Dernière mise à jour
.gitignore	Ajout de changelog	Il y a 8 minutes
README.md	Ajout ReadMe	Il y a environ une heure

Travailler sur une branche récupérée depuis le serveur

- emacs Hello.java ; javac Hello.java
- git add Hello.java ; git commit -m "création Hello"

```
4e2cf67 (HEAD, master) création Hello  
8d46431 (origin/master, origin/HEAD) Initial commit
```

Mise à jour d'une branche sur le serveur

Cas où le nom existe des 2 côtés

- git push
- git log --oneline --decorate

```
4e2cf67 (HEAD, origin/master, origin/HEAD, master) création Hello  
8d46431 Initial commit
```

Mise à jour d'une branche sur le serveur

Branche nouvelle en local (1)

- `git checkout -b addition`
- `emacs Operation.java ; emacs Hello.java ; javac Hello.java`
- `git add . ; git commit -m "ajout Addition"`
- `git log --decorate --oneline --graph -all`

```
* 1919fe4 (HEAD, addition) ajout Addition
* 4e2cf67 (origin/master, origin/HEAD, master) création Hello
* 8d46431 Initial commit
```

- `git push`

```
fatal: La branche courante addition n'a pas de branche amont.
Pour pousser la branche courante et définir la distante comme amont,
utilisez
```

```
git push --set-upstream origin addition
```

Mise à jour d'une branche sur le serveur

Branche nouvelle en local (2)

- `git push --set-upstream origin addition`
- `git log --decorate --oneline --graph --all`

```
* 1919fe4 (HEAD, origin/addition, addition) ajout Addition
* 4e2cf67 (origin/master, origin/HEAD, master) création Hello
* 8d46431 Initial commit
```

- `git config -l`

```
...
remote.origin.url=https://github.com/mermet-iut/intro.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
branch.addition.remote=origin
branch.addition.merge=refs/heads/addition
```

Transfert d'une étiquette locale sur le serveur

- `git tag -a "v1.0"`
- `git log --oneline --decorate --graph --all`

```
* 1919fe4 (HEAD, tag: v1.0, origin/addition, addition) ajout Addition
* 4e2cf67 (origin/master, origin/HEAD, master) création Hello
* 8d46431 Initial commit
```

- `git push origin "v1.0"`

Récupérer la dernière version disponible sur le serveur

- Git pull (équivalent de git fetch + merge)
- Permet d'intégrer dans son travail la dernière version disponible sur le serveur, et donc de prendre en compte le travail des autres

Afficher les branches distantes suivies par les branches locales

- `git branch -vv`
- `git status`
- `git status -sb`

Forks et Merge Requests

Fork : Quid ? Cur ?

- Quid ?

Copie d'un dépôt GitLab, sous la forme d'un nouveau dépôt GitLab

- Cur ?

- Ne pas autoriser n'importe qui à modifier le projet principal
- Créer un nouveau projet à partir d'un projet existant
- Avoir un meilleur suivi des modifications

Fork : Quomodo ?

The screenshot shows the top section of a GitLab repository page. At the top center is a profile picture with the letter 'M' and the repository name 'MonProjet'. Below the name is the description 'Projet pour illustrer le fonctionnement de GitLab'. A row of buttons includes 'Mettre en favori', 'Fourcher' (highlighted with a red arrow and the text 'Cliquer ici !'), and 'Surveillé'. Below this are navigation links for 'Validations (2)', 'Branche (1)', 'Étiquette (0)', and 'LisezMoi', followed by buttons for 'Ajouter un journal des modifications', 'Ajouter une licence', and 'Ajouter un guide de contribution'. The bottom part of the screenshot shows a commit history table with columns for 'Nom', 'Dernière validation', and 'Dernière mise à jour'. The first entry is 'Ajout de changelog' by Bruno Mermet, validated 8 minutes ago.

Nom	Dernière validation	Dernière mise à jour
.gitignore	Ajout de changelog	Il y a 8 minutes
README.md	Ajout ReadMe	Il y a environ une heure

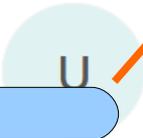
Fork : Quomodo ?

Bruno Mermet > MonProjet > Fork project

Fork project

A fork is a copy of a project.
Forking a repository allows you to make changes without affecting the original project.

Click to fork the project

 Correction	 Temp	 EclairALaVanille	 Equipe 1	 duFond
 Lmodialel	 absents	 Equipe 3	 Genie Logiciel Projet Equipe 6	

Cliquer sur le
groupe dans
lequel « forker »
le projet

Fork : Quomodo ?

MonProjet **a**
Projet pour illustrer le fonctionnement de GitLab

Mettre en favori 0 Fourcher **1** SSH git@forgeb1.univ-lehavre.fr:merm

Surveillé

branche (1) Étiquette (0) LisezMoi Ajouter un journal des modifications Ajouter une licence Ajouter un guide de contribution Mettre

master MonProjet / + Historique Rechercher un fichier

Ajout de changelog
Bruno Mermet a validé Il y a environ une heure **0ec3ad47**

Nom	Dernière validation	Dernière mise à jour
.gitignore	Ajout de changelog	Il y a environ une heure
README.md	Ajout ReadMe	Il y a environ 2 heures

MonProjet **a**
Projet pour illustrer le fonctionnement de GitLab
Fourché depuis Bruno Mermet / MonProjet

Mettre en favori 0 Fourcher 0 SSH git@forgeb1.univ-lehavre.fr:Temp

Global

branche (1) Étiquette (0) LisezMoi Ajouter un journal des modifications Ajouter une licence Ajouter un guide de contribution Mettre

master MonProjet / + Historique Rechercher un fichier

Ajout de changelog
Bruno Mermet a validé Il y a environ une heure **0ec3ad47**

Nom	Dernière validation	Dernière mise à jour
.gitignore	Ajout de changelog	Il y a environ une heure
README.md	Ajout ReadMe	Il y a environ 2 heures

Fork : travail sur le nouveau dépôt

- `git clone git@forgeb1.univ-lehavre.fr:Temp/MonProjet.git`
- `cd intro`
- `emacs Hello.java`
- `git add .`
- `git commit -m "ajout d'un Au Revoir"`
- `git push`

Merge Request (1)

- Merge Request : Quid ?

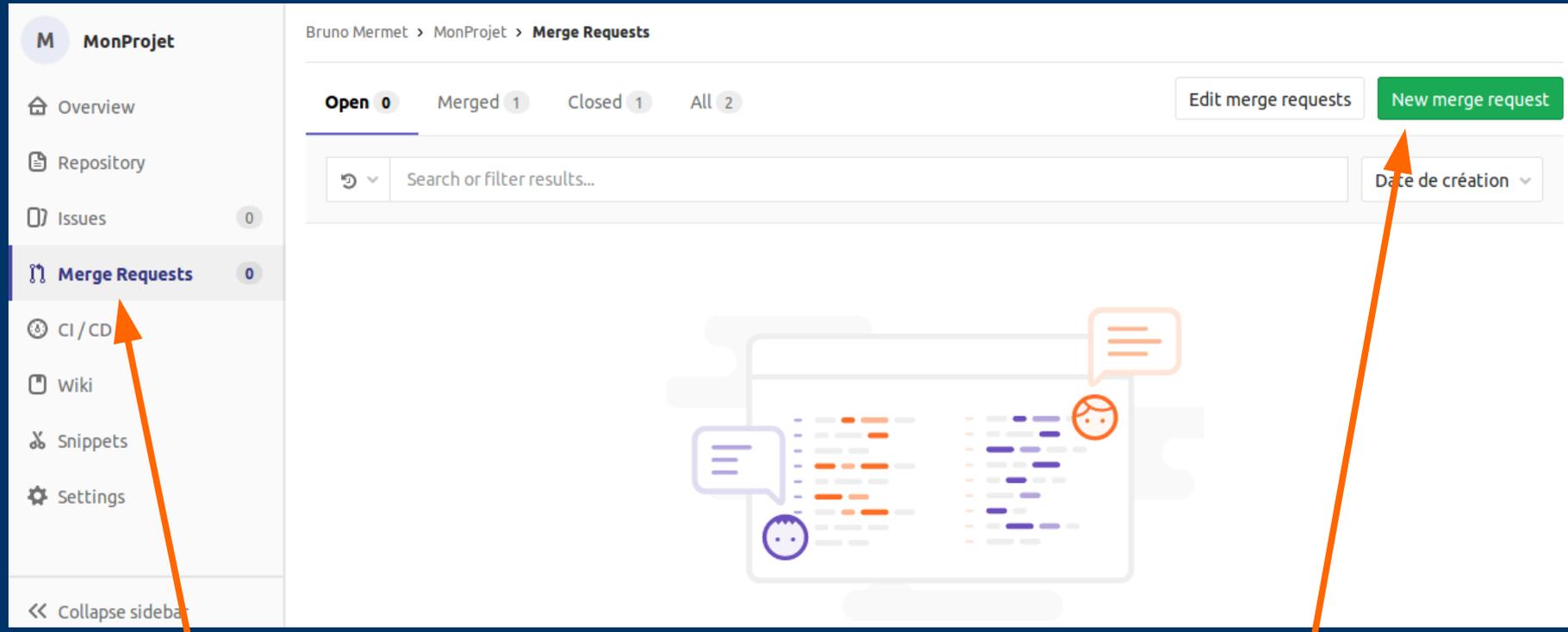
Demande d'intégration du travail fait

- Soit d'un « fork » dans le dépôt principal
- Soit d'une branche vers une autre

- Cur ?

- Prévenir un autre développeur qui validera la demande de fusion
- Prévenir le propriétaire du dépôt d'origine qu'on a fait un travail qu'il est susceptible d'intégrer dans son projet

Merge Request (2) Quomodo ?



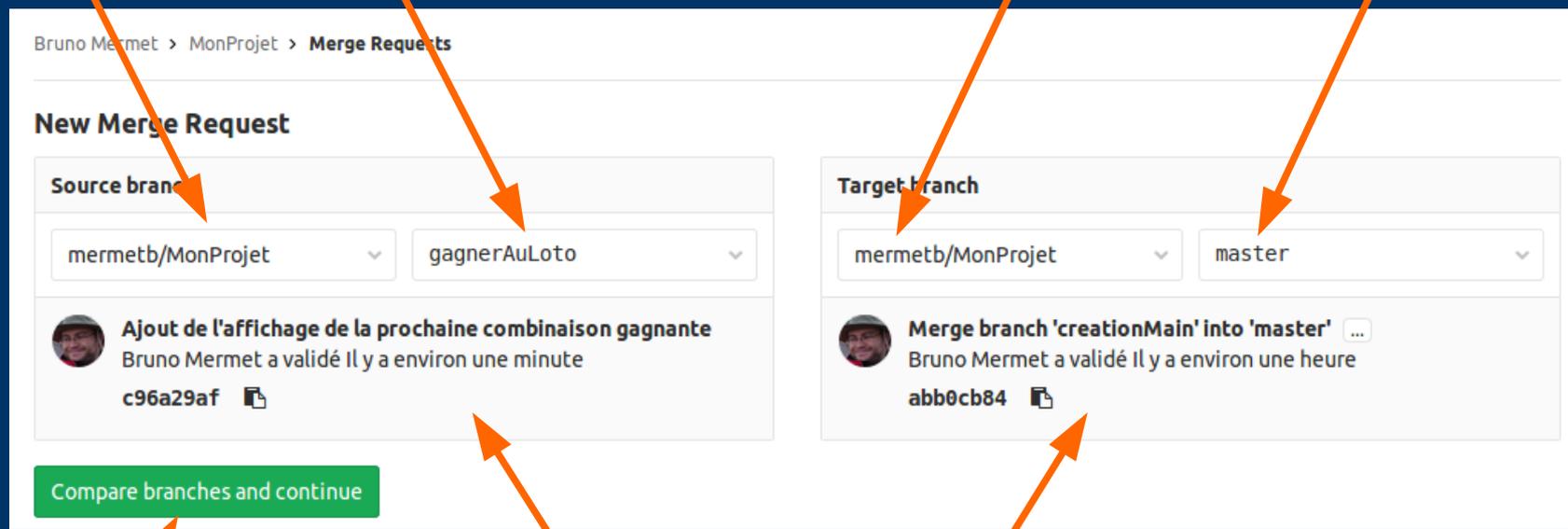
Cliquer là

Puis là !

Merge Request (3)

Dépôt de la branche à intégrer
branche à intégrer

Dépôt de la branche cible
branche cible



Cliquer ici

Messages des derniers commits

Merge Request (4)

Modification éventuellement le titre

Expliquer le but de la modification

Affecter la tâche de relecture éventuellement à un contributeur précis

Pour rendre automatique la suppression de la branche sur le serveur

New Merge Request

From `gagnerAuLoto` into `master` [Change branches](#)

Title

Start the title with `WIP:` to prevent a **Work In Progress** merge request from being merged before it's ready.
Add [description templates](#) to help your contributors communicate effectively!

Description Write Preview B I ↵ <> ☰ ☰ ☑ 🗑️

Markdown and quick actions are supported [Attach a file](#)

Assignee [Assign to me](#)

Milestone

Labels

Source branch

Target branch [Change branches](#)

Remove source branch when merge request is accepted.

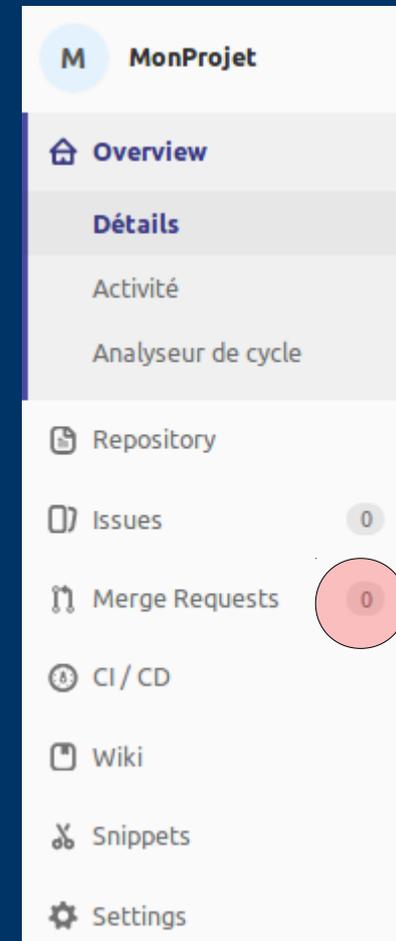
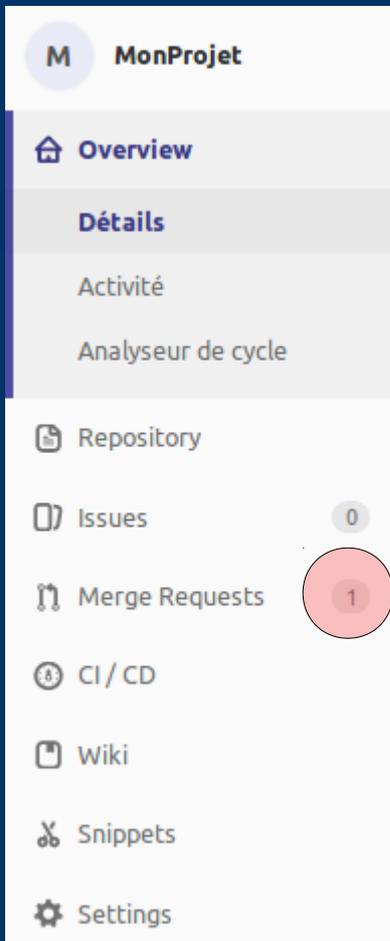
[Submit demande de fusion](#) [Cancel](#)

Voir le cours sur la gestion de projet avec GitLab

Cliquer ici

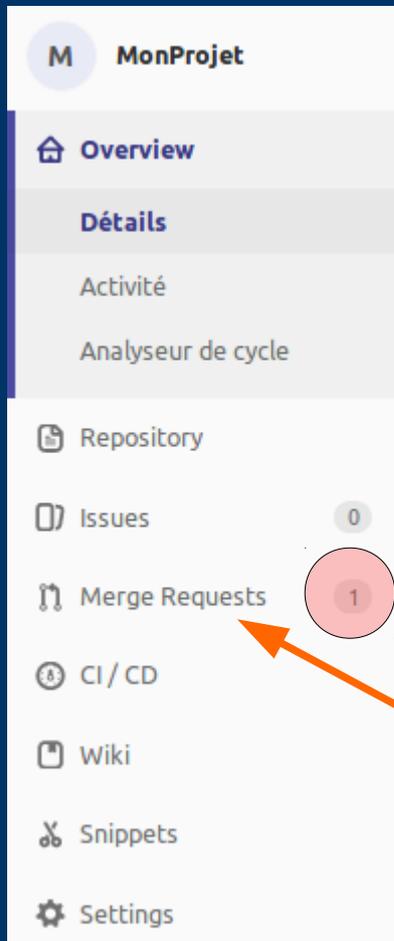
Merge Request

Côté destination (1) vs Côté source



Merge Request

Côté destination (1) vs Côté source



Cliquer ici

Merge Request

Côté destination (1) vs Côté source

M MonProjet

- Overview
- Détails
- Activité
- Analyseur de cycle
- Repository
- Issues 0
- Merge Requests 1
- CI / CD
- Wiki
- Snippets
- Settings

Bruno Mermet > MonProjet > Merge Requests

Open 1 Merged 1 Closed 1 All 3

Edit merge requests New merge request

Search or filter results... Date de création

Ajout de l'affichage de la prochaine combinaison gagnante
13 · opened il y a 2 jours by Bruno Mermet updated il y a 2 jours

Cliquer ici

Fork et Pull Request Côté destination (2)

Cliquer ici pour
refuser
définitivement !

Cliquer ici si on
est d'accord !

Remplir et cliquer si on
veut d'abord
commenter ou
discuter !

The screenshot shows a GitHub Pull Request page. At the top right, there are buttons for 'Edit' and 'Close demande de fusion'. The title of the pull request is 'Ajout de l'affichage de la prochaine combinaison gagnante'. Below the title, there are buttons for 'Merge', 'Remove source branch', and 'Modify commit message'. There is also a 'Check out branch' button and a download icon. Below these buttons, there is a text area for writing a comment, with a 'Write' button and a 'Preview' button. At the bottom, there are buttons for 'Comment' and 'Close merge request'. Annotations with orange arrows point to the 'Merge' button, the 'Write' button, and the 'Close merge request' button.

Se synchroniser avec les évolutions faites sur le dépôt commun

- git pull (équivalent à git fetch + merge)

```
* ae61556 (HEAD, origin/master, origin/HEAD, master) Merge pull request #1 fro
| \
| * a2b0d08 Ajout d'un Au Revoir
| /
| * 1919fe4 (tag: v1.0, origin/addition, addition) ajout Addition
| /
| * 4e2cf67 création Hello
| * 8d46431 Initial commit
```

Fork : synchroniser la copie avec le dépôt initial : initialiser le processus

- Possible uniquement via la ligne de commande !
 - Depuis le clone du « fork »
 - git remote -v

```
origin https://UrlGitLab/depotFork.git (fetch)
origin https://UrlGitLab/depotFork.git (push)
```

- git remote add upstream https://github.com/depotInitial.git
- git remote -v ; git branch -a

```
origin https://UrlGitLab/depotFork.git (fetch)
origin https://UrlGitLab/depotFork.git (push)
upstream https://UrlGitLab/depotInitial.git (fetch)
upstream https://UrlGitLab/depotInitial.git (push)
```

```
creationAffectation
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
```

Fork : synchroniser la copie avec le dépôt initial : effectuer la mise à jour

- git fetch upstream
- git branch -a

```
creationAffectation
* master
remotes/origin/HEAD -> origin/master
remotes/origin/master
remotes/upstream/master
```

- git merge upstream/master

```
Merge made by the 'recursive' strategy.
TestTrain.java | 9 ++++++++
train/Train.java | 20 ++++++++
2 files changed, 27 insertions(+), 2 deletions(-)
```

- git push

Et encore plus...

- Git permet de faire beaucoup d'autres choses, à découvrir avec la documentation indiquée (`git help`, *Pro Git*), notamment :
 - Le retour en arrière (`git reset` notamment)
 - Complétion du commit précédent fait trop vite (`git commit --amend`)
 - Le repositionnement d'une branche sur une version quelconque (`git rebase`)
 - Un stockage temporaire d'un travail non finalisé (`git stash`)
 - Comparer différentes versions (`git diff`)
 - ...
- Certaines opérations de Git peuvent être risquées... L'intérêt du dépôt commun est de pouvoir à tout moment revenir à une version correcte

Pour vos projets

- Le « responsable git »
 - Crée le dépôt Github
 - Choisit la méthode utilisée pour travailler (via des *forks* et des *Pull/Merge Requests*, ou pas...)
- Pour le développement d'une fonctionnalité
 - Créer une branche (à partir du dépôt de base ou d'un *fork*)
 - Développer, tester, documenter la fonctionnalité, en récupérant régulièrement la dernière version disponible sur le dépôt et en la fusionnant dans sa branche
 - Lorsque la fonctionnalité est terminée, la ré-intégrer dans le tronc, supprimer la branche, transférer sur le serveur, prévenir l'équipe

Développement d'une fonctionnalité sans validation par un tiers : démarche

On part de la dernière version du tronc	git checkout master git pull
On crée la nouvelle branche et on travaille dedans	git checkout -d <i>func</i>
On développe la nouvelle fonctionnalité par étapes. On ne fait de « commit » que de versions correctes	while (!terminé) { Coder et documenter Compiler Tester git commit (Éventuellement, git push) }
On peut utiliser le serveur pour sauvegarder des phases intermédiaires	
On revient sur le tronc	git checkout master
On récupère la dernière version	git pull
On intègre dans le tronc la fonctionnalité	git merge <i>func</i>
On supprime la branche	git branch -d <i>func</i>
On transfère la modif. sur le serveur	git push

Développement d'une fonctionnalité avec validation par un tiers : démarche

On part de la dernière version du tronc	git checkout master git pull
On crée la nouvelle branche et on travaille dedans	git checkout -d <i>fonc</i>
On développe la nouvelle fonctionnalité par étapes. On ne fait de « commit » que de versions correctes	while (!terminé) { Coder et documenter Compiler Tester git commit (Éventuellement, git push) }
On peut utiliser le serveur pour sauvegarder des phases intermédiaires	
Depuis GitHub/GitLab, créer une pull/merge request	
Validation ou non par le tiers après discussion éventuelle	
On repasse sur le tronc	git checkout master
On supprimer la branche	git branch -d <i>fonc</i>
On synchronise son tronc	git pull