



GitLab

# Git et Gitlab

- Par CFD-Innovation



# Versioning de code

- Par CFD-Innovation

# Versionner son code

Pour gérer les versions du code au fur et à mesure des modifications

- **Pourquoi ?**

- Pour garder un historique des modifications
  - L'auteur
  - La date et l'heure
  - L'explication
  - Les fichiers modifiés et leurs modifications
- Pour revenir en arrière
- Pour travailler en équipe (mais pas forcément ! Nous y reviendrons)

- **les commits**

- Modifications de fichiers (modifications de lignes/créations/suppressions) = un commit
- Un commit correspond donc à une version du projet à un instant T
- Somme des commits = historique d'un projet

# Les solutions de versioning

- Deux modèles existent :

- Le modèle **centralisé** :

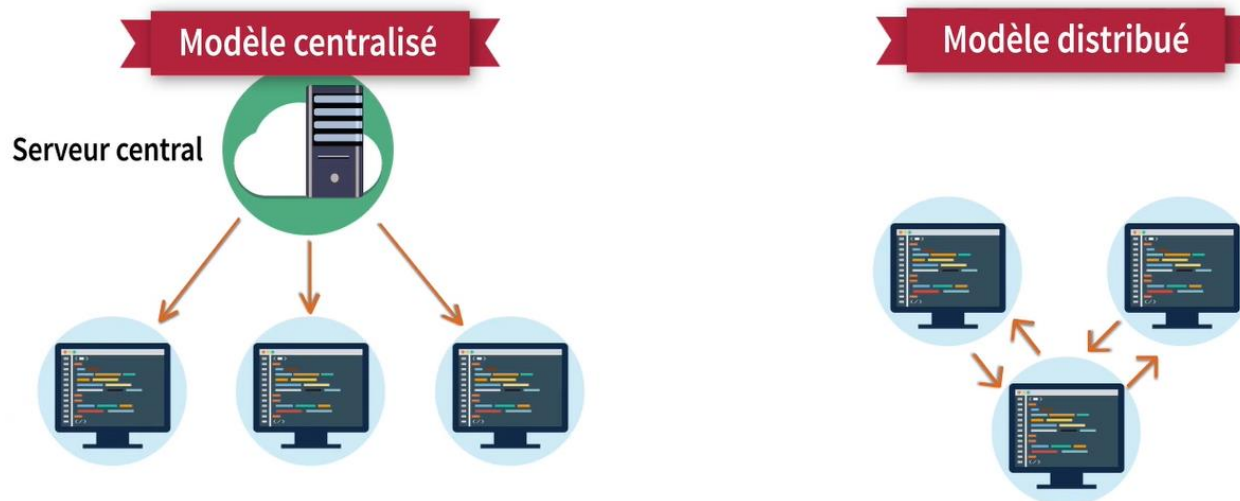
Un serveur central contient l'historique du code et des modifications.

*Exemple : SVN (subversion)*

- Le modèle **distribué** :

Chaque « dépôt » contient l'historique du code et des modifications. Quelques avantages de ce modèle : Moins de risque de perdre le code et son historique, on peut travailler en local sans être connecté au réseau de l'entreprise ou à internet

*Exemple : Git*

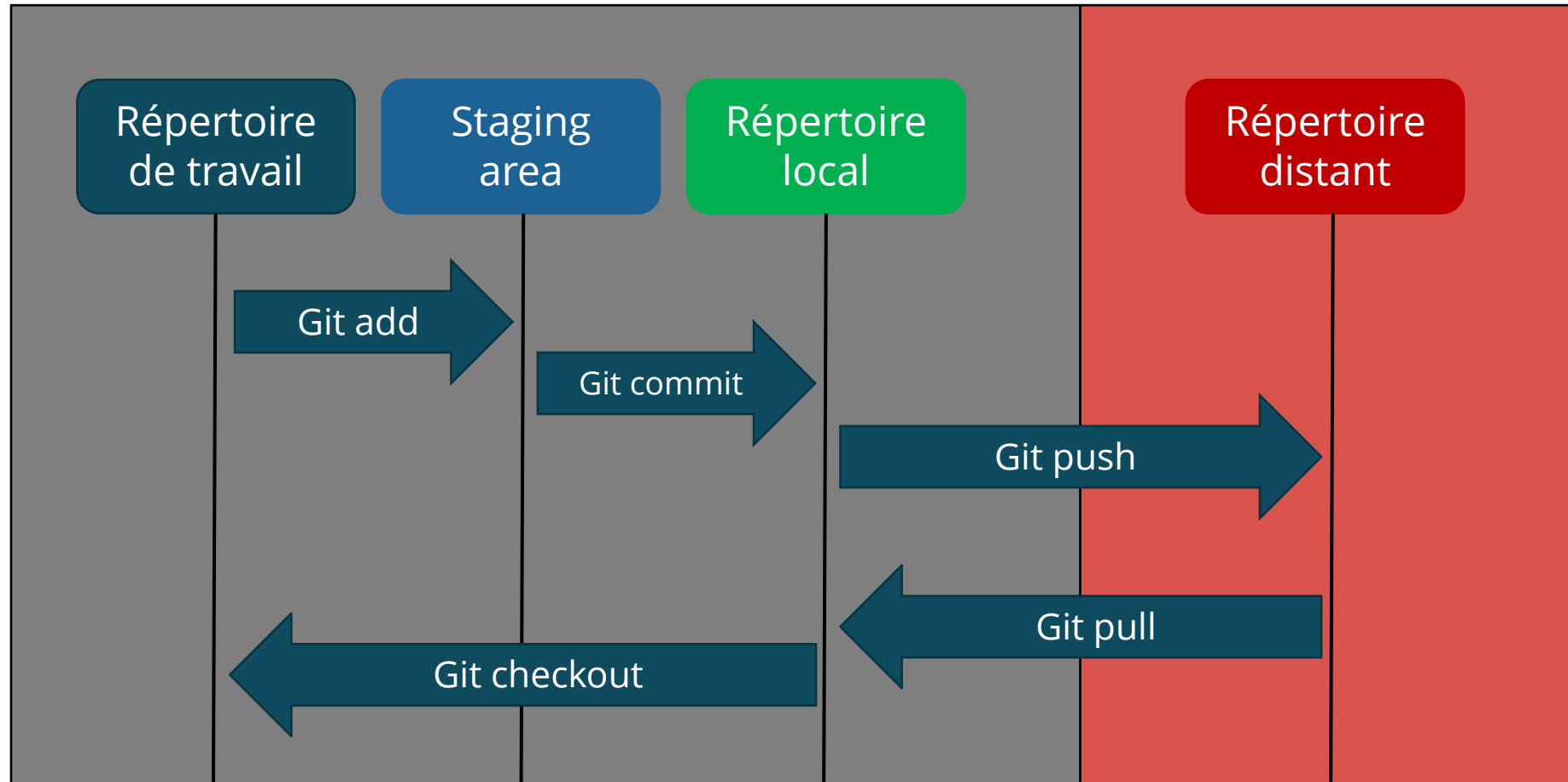




# Utilisation de GIT – Les premiers pas

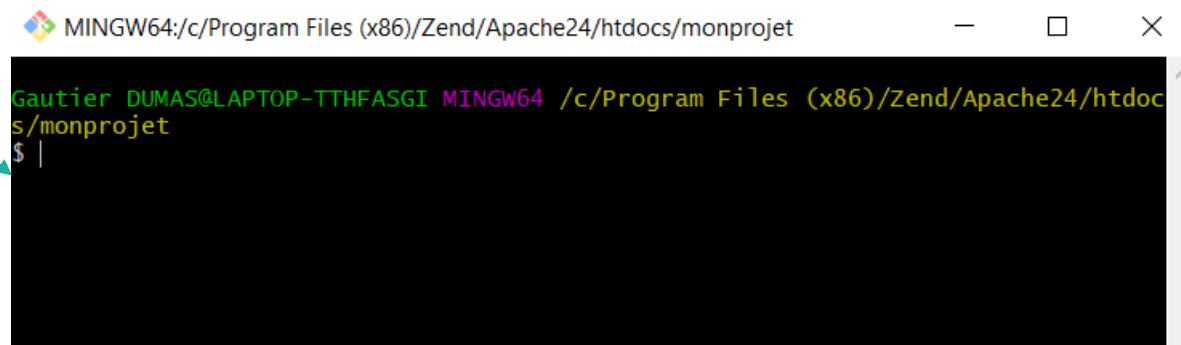
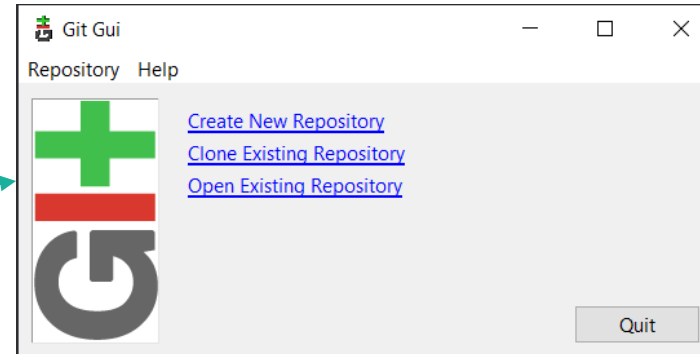
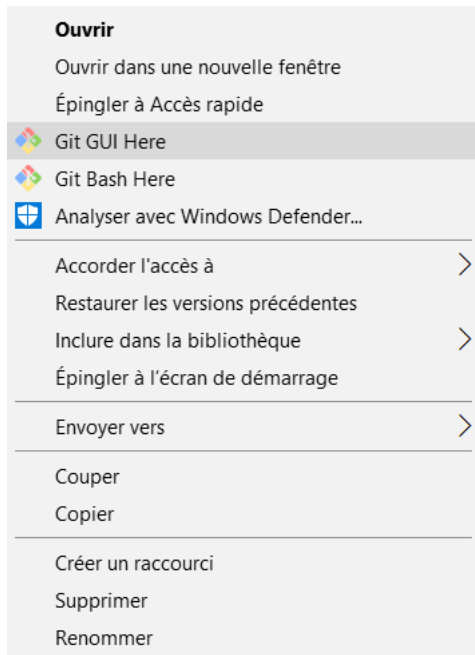
- Par CFD-Innovation

# GIT - Fonctionnement



# Installation GIT

- Téléchargement sur : <https://git-scm.com/download/win>
- Installation à effectuer sur les postes des développeurs
- En ligne de commande et petit GUI



# Configuration du poste

- Une fois git installé, il est presque prêt à être utilisé
- Configuration des informations concernant le développeur

```
git config --global user.name "Gautier DUMAS"  
git config --global user.email "gdumas@cf-d-innovation.fr"
```

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet  
$ git config --global user.name "Gautier DUMAS"  
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet  
$ git config --global user.email "gdumas@cf-d-innovation.fr"
```



# Premiers pas – git init et git add

- Pour démarrer le versioning d'un dossier, positionnez vous sur le dossier à versionner et faire une initialisation avec

```
git init
```

Cela va générer un index contenant les fichiers à suivre (vide par défaut) dans un dossier « .git »

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet
$ git init
Initialized empty Git repository in C:/Program Files (x86)/Zend/Apache24/htdocs/monprojet/.git/
```

- Puis ajoutez dans l'index git les fichiers à suivre à l'aide d'une instruction git add

Pour un fichier : `git add fichier_a_suivre.txt`

Pour tous les fichiers d'un répertoire : `git add .`

# Git status

- Pour connaître l'état à un instant T d'un repository git

```
git status
```

- Permet de savoir s'il y a des fichiers non indexés, des fichiers modifiés ou créés non « commités » etc...

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/jsonValidator (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
```

## Exemples :

- tout est ok, la branche est « up to date »
- Le fichier my\_new\_file.php n'est pas indexé (untracked files)

# Premiers pas – git commit

- Une fois les fichiers « indexés », vous pouvez effectuer le premier commit

```
git commit -m "Mon premier commit ! Contient la première version"
```

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet (master)
$ git commit -m "Mon premier commit ! Contient la première version"
[master (root-commit) 2ef5f4f] Mon premier commit ! Contient la première version
3 files changed, 3 insertions(+)
create mode 100644 css/style.css
create mode 100644 index.php
create mode 100644 js/traitement.js
```

- N'oubliez pas l'option « m » qui permet de préciser un message dans le commit. Ce message sera très important pour la consultation et la compréhension de l'historique, ainsi que dans la coordination des équipes de développeurs
- L'option « a » (--all) permet d'effectuer directement le commit sur tous les fichiers suivis et ayant subi une modification

```
git commit -am "Commit de mes modifications"
```

# Git – git diff

- Pour connaître la différence entre l'état actuel de votre projet et un commit donné

```
$ git diff ba08cf3
diff --git a/my_new_file.php b/my_new_file.php
index e69de29..f9a2fdf 100644
--- a/my_new_file.php
+++ b/my_new_file.php
@@ -0,0 +1 @@
+modif
```

```
git diff ba08cf3
```

- Pour comparer avec une autre branche

```
$ git diff master
diff --git a/my_new_file.php b/my_new_file.php
index f9a2fdf..ecda087 100644
--- a/my_new_file.php
+++ b/my_new_file.php
@@ -1,1 +1,2 @@
-modif
+modif 2
```

```
git diff master
```

# Premiers pas – Consulter l'historique

- Le git commit permet de pousser les modifications dans une version, pour les consulter, nous utiliserons git log

```
git log
```

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet (master)
$ git log
commit 31f55668e407e22c0a24279021f38a16f8352821 (HEAD -> master)
Author: Gautier DUMAS <gdumas@cf-d-innovation.fr>
Date:   Fri Jan 17 11:47:29 2020 +0100

    Deuxième commit

commit 2ef5f4f96db3ef0e320ad462586cf2e693f83e64
Author: Gautier DUMAS <gdumas@cf-d-innovation.fr>
Date:   Fri Jan 17 11:44:02 2020 +0100

    Mon premier commit ! Contient la première version
```

Commit 2

Commit 1

# Premiers pas – git checkout

- Permet de se positionner sur une version ou sur une branche (concept des branches exploré plus tard)

```
git checkout ID_COMMIT  
Ou  
git checkout master  
Ou  
git checkout ma_branche
```

- Afin de revenir à une version antérieure du code (pour comparer, tester, vérifier, rollback ...)

# Fichier .gitignore

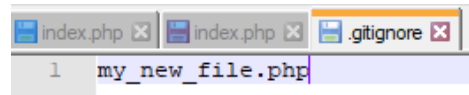
- Utilisation d'un fichier .gitignore qui va « blacklister » les fichiers à ignorer (simple fichier .gitignore à positionner à la racine du projet et contenant la liste des fichiers à ignorer sur chaque ligne)
- Pour des raisons de sécurité et de clarté, nous ignorons régulièrement
  - Les fichiers de configurations (pouvant contenir des passwd ou des clés privés ne devant pas être partagés)  
On préférera « commiter » un template/exemple du fichier de configuration  
*Ex: config.inc.sample.php*
  - Les fichiers et les dossiers temporaires ou les fichiers contenant les « datas »
  - Les fichiers propres aux environnements de développement souvent créés par l'IDE ou l'OS (.project, .workspace ...)
- Le fichier .gitignore est lui-même un fichier à suivre et à versionner

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/ht
jet (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
```

Git status avant .gitignore



Création du .gitignore

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/ht
jet (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)
```

Git status après .gitignore

# TP 1 – Git bash

1. Assurez vous d'avoir git bash installé dans votre environnement. Sinon installez-le.
2. Dans un dossier local nommé « formationWeb », initialisez le versioning avec git
3. Copiez les supports de formation PDF dans ce dossier et versionnez les à l'aide d'un commit.
4. Ajoutez un fichier « .gitignore » dans lequel vous indiquerez le dossier « notes\_perso » comme à ignorer et versionnez le.
5. Créez un sous-dossier notes\_perso et ajoutez un fichier à l'intérieur.
6. Observez les résultats de la commande « git status » et essayez d'ajouter le dossier au versioning. Que se passe t il ?
7. Ajoutez un nouveau document Word à la racine du dossier formationWeb « notes\_formation.docx ». Ajoutez au contrôle de version ce document vierge. Puis ajoutez quelques lignes à l'intérieur et commitez les modifications.
8. Listez l'historique des modifications du dossier formationWeb



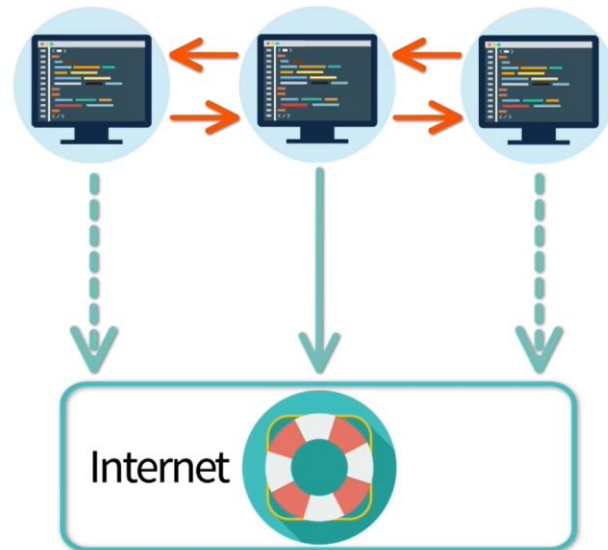


# Remote Git : Introduction

- Par CFD-Innovation

# Remote git

- Inconvénients du modèle distribué :
  - Tout l'historique du code étant en local, si le PC tombe (vol, crash, accident ...), le code et l'historique de celui ci est perdu
  - Le poste local n'est pas forcément accessible aux autres développeurs
- L'utilisation d'un remote git va principalement permettre de
  - Réaliser des sauvegardes externes à l'environnement de développement
  - Faciliter le travail en équipe (mise à disposition du code dans un endroit partagé)



# Github - Gitlab - BitBucket

- **Github, Gitlab et BitBucket** sont des solutions de remote git, proposant :
  - des solutions SaaS dans le Cloud
  - hébergeant des dépôts GIT (privés et/ou publiques selon les solutions)
  - des interfaces web pour consulter les évolutions du projet et gérer les « à côtés »
- Dans le cadre de cette formation, nous aurons un aperçu de GitHub (en tant que consommateur) et c'est la solution GitLab qui sera explorée notamment pour :
  - ses fonctionnalités de versioning
  - ses fonctionnalités de gestion de projets (Issues, CI/CD, Wiki ...)
  - la possibilité d'héberger en on-premise un serveur GITLAB

# GitHub – Introduction

- Accessible en mode SaaS sur <https://github.com/>
- Véritable CV des développeurs en ligne
- Un standard pour les projets Open Source
  - des millions de projets (85 M à l'écriture du document), qui font de GitHub la plateforme d'hébergement de code la plus grande du monde
- Très largement utilisé par les développeurs qui souhaitent utiliser des codes ou librairies Open Source (tout langage confondu)

# GitHub – Page projet

The screenshot shows the GitHub repository page for 'ThePrez / ServiceCommander-IBMi'. The page is divided into several sections:

- Header:** Search bar, navigation links (Pull requests, Issues, Marketplace, Explore), and user profile.
- Repository Info:** Repository name 'ThePrez / ServiceCommander-IBMi' and public status.
- Navigation Tabs:** Code, Issues (7), Pull requests (1), Actions, Projects, Wiki, Security, Insights.
- Statistics:** Unwatch (11), Fork (10), Star (25).
- Main Content:** File tree and commit history.
- Right Sidebar:** About section with project details.

File	Commit Message	Time Ago
.github	fix IBMi CI failure	last month
conf	fixup default /QOpenSys/etc/sc/conf/scrc	3 months ago
eclipse	add code formatter and cleanup template for eclipse	16 months ago
man	update man pages	2 months ago
native	add <code>scbash</code> executable to tuck away env vars	3 months ago
quickstart	Update HANDS_ON.md	13 months ago
samples	Increase startup wait time for Db2 WebQuery	3 months ago
scripts	auto-find UTF8 locale for non-EN_US systems	10 days ago
src/main/java/jesseg/ibmi/opensource	Fix error when temp dir for logs already exists	last month
strtcpsvr	Ignore system group on STRTCPSVR/ENDTCPSVR instance *ALL (#175)	10 days ago

**About**

Service Commander for IBM i

service batch ibm ibmi

Readme

Apache-2.0 license

Code of conduct

25 stars

11 watching

10 forks

**Releases** 27

v1.4.3 **Latest** on 25 Apr

+ 26 releases

Menu de navigation dans le projet :

- Code
- Issues
- Pull Requests
- ...

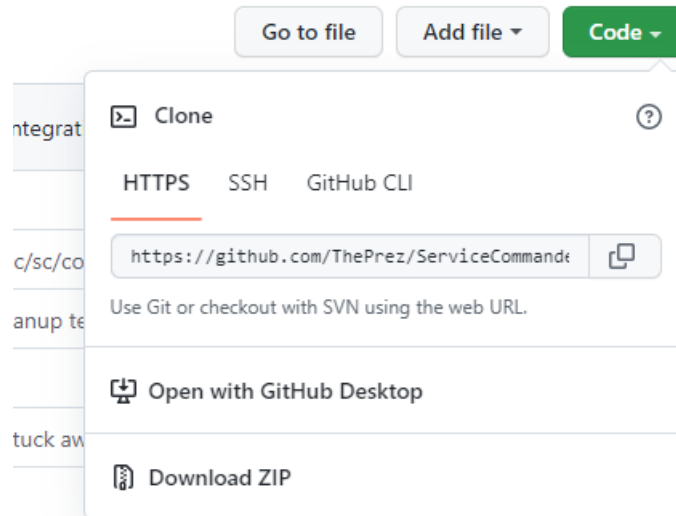
Statistiques d'utilisation du projet. Possible de s'abonner à un projet pour être notifié en cas d'activité sur celui ci

Le code et son arborescence

Statistiques d'utilisation du projet. A regarder attentivement avant intégration dans son application

# GitHub – Clone du projet

- Pour récupérer dans son environnement local les sources d'un projet Open Source sur GitHub :
  - Bouton « Code »
  - Propose les différentes possibilités de récupération du code :
    - HTTPS
    - SSH
    - Download en ZIP
    - GitHub CLI
    - ...



# Suite avec GitLab

---

- Dans la suite du support, nous allons explorer la solution GitLab
- La plupart des fonctionnalités et concepts exprimés sont très similaires d'un environnement à l'autre

# Ouverture d'un compte GitLab.com

- Création d'un compte pour chaque collaborateur / développeur sur [https://gitlab.com/users/sign\\_in#register-pane](https://gitlab.com/users/sign_in#register-pane)

## GitLab.com

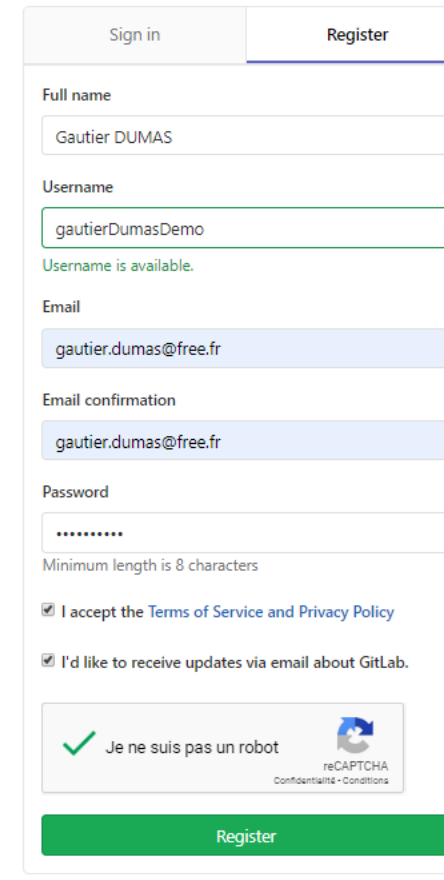
GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab.com Support Forum](#)
- [GitLab Homepage](#)

By signing up for and by signing in to this service you accept our:

- [Privacy policy](#)
- [GitLab.com Terms](#).

- Un nombre illimité de projets (repositories) privés
- Un nombre illimité de collaborateurs  
*Est récemment devenu payant à partir de 5 développeurs dans une équipe*
- Support de la communauté par le forum

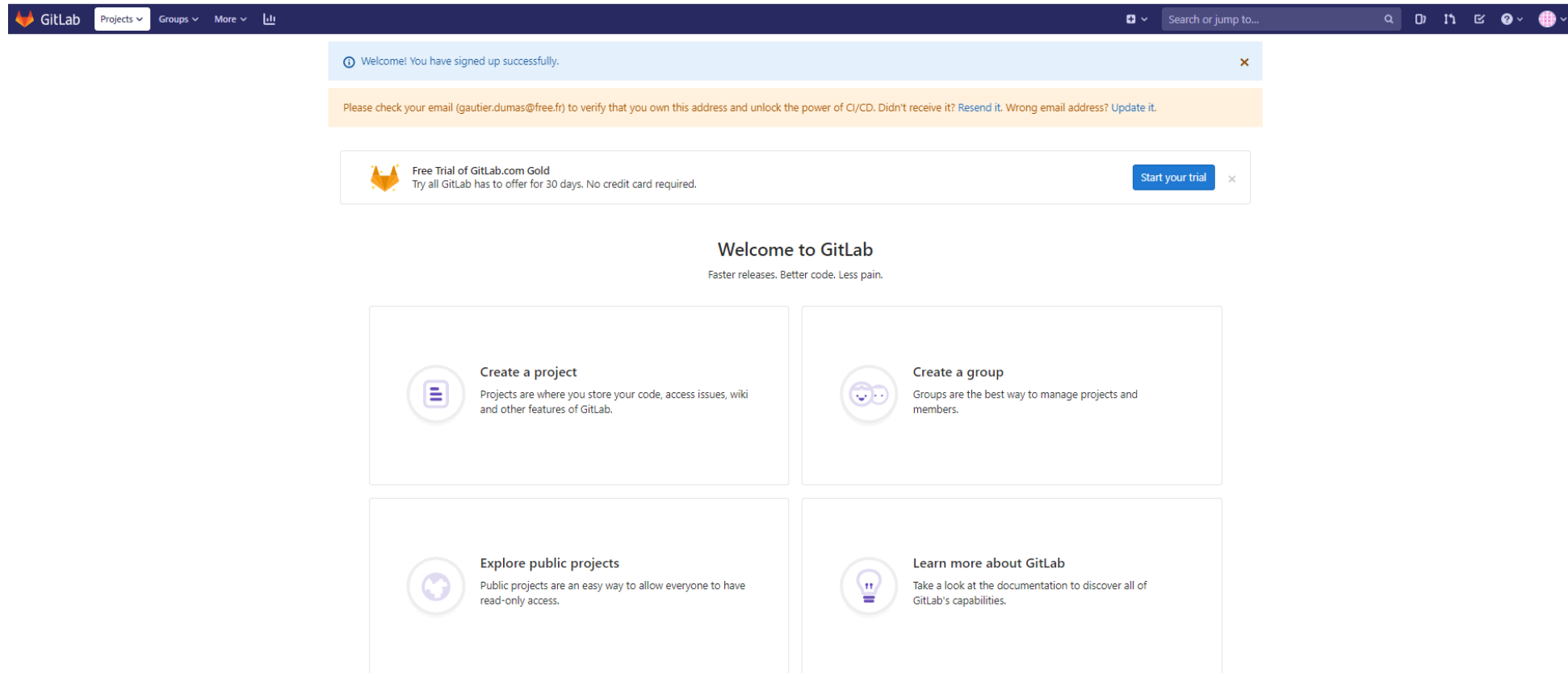


The screenshot shows the GitLab.com registration interface. At the top, there are two tabs: 'Sign in' and 'Register', with 'Register' being the active tab. The form contains the following fields and elements:

- Full name:** A text input field containing 'Gautier DUMAS'.
- Username:** A text input field containing 'gautierDumasDemo'. Below the field, a green message states 'Username is available.'
- Email:** A text input field containing 'gautier.dumas@free.fr'.
- Email confirmation:** A text input field containing 'gautier.dumas@free.fr'.
- Password:** A password input field with masked characters '.....'. Below the field, a note states 'Minimum length is 8 characters'.
- Terms and Conditions:** Two checkboxes are present. The first is checked and labeled 'I accept the Terms of Service and Privacy Policy'. The second is checked and labeled 'I'd like to receive updates via email about GitLab.'.
- reCAPTCHA:** A checkbox labeled 'Je ne suis pas un robot' is checked. To its right is the reCAPTCHA logo and the text 'Confidentialité - Conditions'.
- Register Button:** A large green button labeled 'Register' is at the bottom of the form.



# Welcome page



The screenshot shows the GitLab welcome page. At the top is a dark blue navigation bar with the GitLab logo, links for Projects, Groups, and More, a search bar, and user profile icons. Below the navigation bar are three horizontal banners: a blue success message, an orange email verification notice, and a white banner for a free trial of GitLab.com Gold. The main content area is titled 'Welcome to GitLab' with the tagline 'Faster releases. Better code. Less pain.' Below this are four white cards with rounded corners, each featuring a circular icon and a title. The cards are: 'Create a project' (document icon), 'Create a group' (people icon), 'Explore public projects' (globe icon), and 'Learn more about GitLab' (lightbulb icon).

GitLab Projects Groups More


Welcome! You have signed up successfully.


Please check your email (gautier.dumas@free.fr) to verify that you own this address and unlock the power of CI/CD. Didn't receive it? [Resend it](#). Wrong email address? [Update it](#).


**Free Trial of GitLab.com Gold**  
Try all GitLab has to offer for 30 days. No credit card required. [Start your trial](#)


## Welcome to GitLab

Faster releases. Better code. Less pain.

**Create a project**  
Projects are where you store your code, access issues, wiki and other features of GitLab.

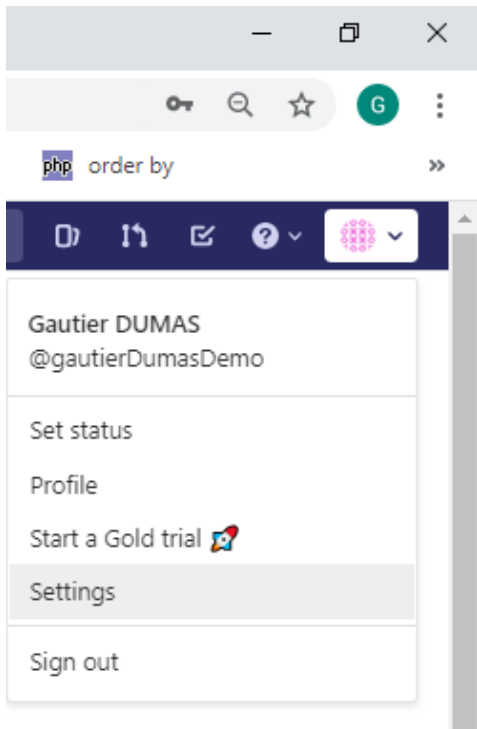
**Create a group**  
Groups are the best way to manage projects and members.

**Explore public projects**  
Public projects are an easy way to allow everyone to have read-only access.

**Learn more about GitLab**  
Take a look at the documentation to discover all of GitLab's capabilities.

# Paramètres initiaux – Profil et Préférences

- Compléter le profil (avatar, profil, email ...)
- Préférences (Thème, langage, page d'accueil par défaut)



## Navigation theme

Customize the appearance of the application header and navigation sidebar.



## Syntax highlighting theme

This setting allows you to customize the appearance of the syntax. [Learn more.](#)



## Localization

Customize language and region related settings. [Learn more.](#)

### Language

Français

This feature is experimental and translations are not complete yet

### First day of the week

Monday

# Paramètres initiaux - Notifications

- Choisir le mode de notification global (configurable par la suite au niveau projet)

## 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

gautier.dumas@free.fr

#### Global notification level

🔔 Participer ▼

☐ Receive notifications about your own activity

Groups (0)

Projects (0)

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

### Notification setting level

🔔 Participer ▼

#### Surveiller

Vous recevrez des notifications pour n'importe quelles activités

#### ✓ Participer

Vous ne recevrez de notification que pour les sujets auxquels vous avez participé

#### En cas de citation

Vous ne recevrez de notifications que pour les commentaires où vous êtes @mentionné

#### Désactivé

Vous ne recevrez aucune notification par courriel

#### Custom

Vous ne recevrez de notification que pour les événements que vous aurez choisis

# Premier groupe

- Les groupes permettent
  - De gérer plusieurs projets et d'y collaborer à plusieurs
  - Les membres d'un groupe ont accès à tous ses projets
- Les groupes peuvent également être imbriqués en créant des sous-groupes
- Les projets appartenant à un groupe sont préfixés avec l'espace de noms du groupe.
- Les projets existants peuvent être déplacés dans un groupe

Nom du groupe

DemoGroup

URL du groupe

https://gitlab.com/ demogroup3

Description du groupe (optional)

Un groupe qui va contenir tous les projets Démo et tous les collaborateurs devant accéder aux sources de la démo

Avatar de groupe

Choose file...

Aucun fichier sélectionné

La taille maximale autorisée pour un fichier est de 200 Kio.

Niveau de visibilité

Qui sera en mesure de voir ce groupe? [Afficher la documentation](#)

☒ Privé

The group and its projects can only be viewed by members.

☐ Public

The group and any public projects can be viewed without any authentication.

Créer un groupe

# Premier projet

- Création à la racine, dans un groupe ou un sous-groupe



- Le projet est l'endroit où nous allons
  - Héberger les fichiers (dépôt)
  - Planifier le travail (tickets)
  - Documentez le projet (wiki)

# Premier projet

Blank project	Create from template	Import project	Intégration et livraison continues pour dépôt externe
---------------	----------------------	----------------	---

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)

Visibility Level

☒ Privé  
L'accès au projet doit être explicitement accordé à chaque utilisateur.

☒ **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 projectAnnuler

P **ProjetDemo**

**Project overview**

Détails

Activité

Releases

Analyse de cycle

Dépôt

Tickets 0

Demandes de fusion 0

Intégration et livraison continue

Opérations

Paquets

Wiki

Extraits de code

Paramètres

# Membres d'un groupe

- Inviter les acteurs du projet dans un groupe
- Détails des rôles et permissions sur : <https://gitlab.com/help/user/permissions>

DemoGroup > Group members

## Group members

Invite member

GitLab member or Email address

Gautier DUMAS

Choose a role permission

Guest

[En savoir plus sur les droits des rôles](#)

Date d'expiration de l'accès

Expiration date

Inviter

GitLab member or Email address

Gautier DUMAS

Choose a role permission

Guest

Guest

Reporter

Developer

Maintainer

Owner

Inviter

# Gestion du dépôt

- Consultation du projet dans « Fichiers »
- Historique des commits
- Historique des branches et des étiquettes
- Statistiques de contributions par développeurs
- Représentation graphique des modifications (git graphique)
- Outil pour comparer les branches

DemoGroup > ProjetDemo > Dépôt

master


projetdemo /

+

Historique


Rechercher un fichier


EDI Web



Initial commit  
Gautier DUMAS a créé il y a 2 minutes

b49345db

Nom	Dernier commit	Dernière mise à jour
 README.md	Initial commit	il y a 2 minutes

 README.md

## ProjetDemo

Mon premier projet de démo

 Dépôt

Fichiers

Commits

Branches

Étiquettes

Contributeurs

Graphique

Comparer

Statistiques



# Gestion des tickets

- Les tickets GitLab peuvent être des bogues, des tâches ou des sujets de discussion.
- Ils sont consultables et filtrables

DemoGroup > ProjetDemo > Tickets > Nouveau

## Nouveau ticket

Titre Demande d'évolution - Pour la démo

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

Description

**Write** Aperçu

**B I ” </> @ ::**

La description détaillée pour indiquer ce qu'il faut faire ou l'objet de la demande.

La syntaxe **\*\*Markdown\*\*** peut être utilisée pour mettre en forme la description.

Des fichiers peuvent être joints.

[Markdown](#) and [quick actions](#) are supported

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

Assignee

Gautier DUMAS

[Assign to me](#)

Due date

Select due date

Milestone

Milestone

Labels

Labels

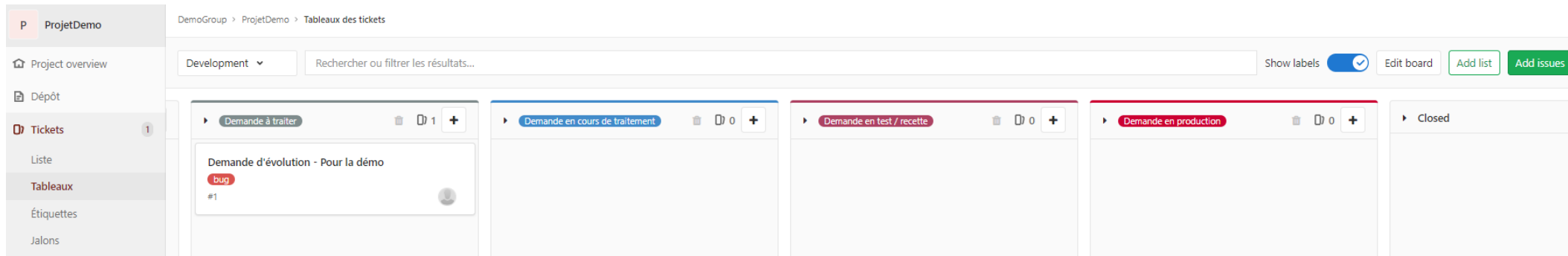
Submit ticket

- Ouverture d'un ticket avec
  - Titre
  - Description
  - Développeur associé (un seul dans la version free)
  - Milestone (groupe de tickets pour des demandes plus importantes)
  - Etiquettes pour classifier les tickets
  - Date limite de résolution

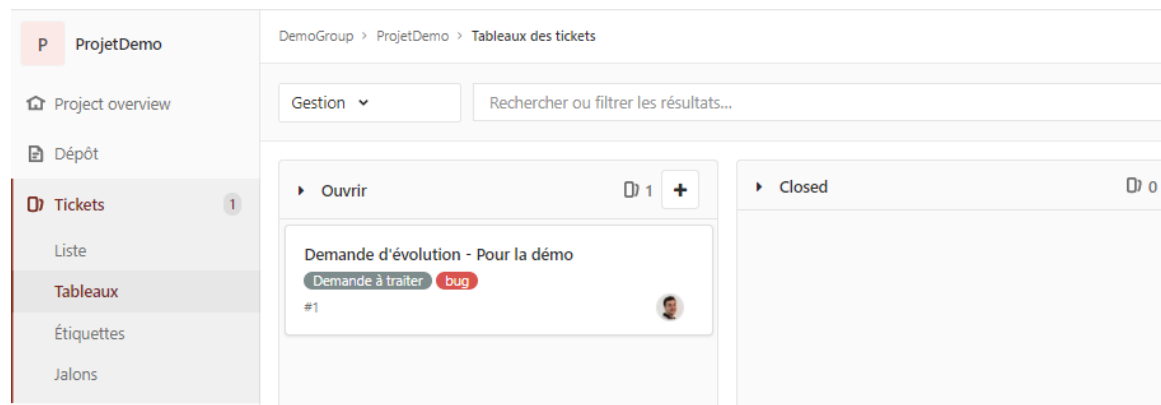
p. 33

# Gestion des tickets

- Affichage des tickets par liste
- Affichage par tableaux (type KanBan), personnalisation des listes et des étiquettes



- Plusieurs tableaux possibles



# Wiki d'un projet

- Gitlab propose au niveau projet, les fonctionnalités d'un wiki
- Cela permet notamment d'écrire la documentation du projet
- C'est un endroit où nous pouvons stocker tous les détails concernant un projet :
  - Sa raison d'être
  - Ses principes
  - Comment utiliser le projet
  - ...
- Un wiki est composé d'une ou plusieurs pages pouvant être hiérarchisées
- L'écriture des pages supporte le format Markdown, Rdoc, AsciiDoc, Org
- Les pages sont elles mêmes versionnées automatiquement et l'historique est consultable

# Wiki – Première page

DemoGroup > ProjetDemo > Wiki > Nouveau

## Create New Page

Titre

Accueil wiki

💡 Astuce : vous pouvez saisir le chemin d'accès complet du nouveau fichier. Nous créerons automatiquement les répertoires manquants.

Format

Markdown

Content

**Write** Aperçu

Voici la page d'accueil du wiki, qui peut référencer des liens, des informations, des images

ou des documents attachés

et qui peut faire le récapitulatif de tout ce qu'il y a à savoir sur le projet |

Markdown is supported

📎 Attach a file

Pour créer un lien vers une (nouvelle) page, il suffit de saisir `[Link Title](page-slug)`. D'autres exemples se trouvent dans la documentation.

Message de commit

Create Accueil wiki

Créer la page

Annuler

DemoGroup > ProjetDemo > Wiki > Accueil wiki

Wiki was successfully updated.

## Accueil wiki

Dernière modification par **Gautier DUMAS** à l'instant

Nouvelle page

Historique de la page

Éditer

Voici la page d'accueil du wiki, qui peut référencer des liens, des informations, des images

ou des documents attachés

et qui peut faire le récapitulatif de tout ce qu'il y a à savoir sur le projet

## Accueil wiki

Plus de Pages

p. 36

# Le fichier .readme

- Apparaît sur la page d'accueil du repository sur Gitlab ou GitHub
- Langage Markdown

Name	Last commit	Last update
CFD_DB_ADAPTER	MAJ getLastinsertid()	1 month ago
Exceptions	maj nom toolkit->cfd-db-adapter	3 months ago
.gitignore	Initial commit	4 months ago
LICENSE	license	4 months ago
README.md	Ajout support pdo dblib	1 month ago
composer.json	maj nom toolkit->cfd-db-adapter	3 months ago

**README.md**

## cfd-db-adapter

Boîte à outils CFD-Innovation

### CFD\_DB\_ADAPTER

Classe permettant de switch entre une base de données DB2, MySQL ou tout autre connecteur supporté par PDO

###Prérequis

- DB2 : ext-ibm\_db2
- MySQL : ext-mysqli
- PDO : ext-pdo

###Exemples :

# Le fichier .readme - Markdown

- Mise en forme de titre, italique, gras, code

```
# Titre 1
##### Titre 5
Ou
**gras**
*italic*
Ou
```extrait de code```
```

# Le fichier .readme - Markdown

- Citations

```
> Ceci est une citation
```

- Listes

```
* une élément  
* un autre  
  * un sous élément  
  * un autre sous élément  
* un dernier élément
```

- Checkbox

```
- [ ] Checkbox  
- [x] Checkbox cochée
```

# Exemple complet - markdown

```
1  # Titre 1
2
3  ##### Titre 5
4
5  Ou
6
7  **gras**, *italic*, ***gras & italic***
8
9  Ou
10
11  ```extrait de code```
12
13  ou
14
15  - [ ] Checkbox
16  - [x] Checkbox cochée
17
18  ![Markdown](markdown.png)
19
```

## Titre 1

### Titre 5

Ou

**gras**, *italic*, ***gras & italic***

Ou

extrait de code

ou

☐ Checkbox

☒ Checkbox cochée






# Demandes de fusion et Intégration continue


---

- La solution GitLab possède les modules pour
  - Faire des demandes de fusions (Merge requests)
  - Configurer de l'intégration continue (livraison automatique, suivi des livraisons etc...)
- Les notions de CI/CD ne sont pas explorées dans cette formation.

# TP2 - GitLab

1. Créez vous un compte sur [https://gitlab.com/users/sign\\_up](https://gitlab.com/users/sign_up) avec votre adresse professionnelle et personnalisez votre profil
2. Créez un premier projet privé (**supports**) dans votre espace par défaut (doit correspondre à votre nom de profil GitLab)
3. Créez et versionnez un fichier README.md avec la mise en page ci-contre (vous pouvez utiliser le Web IDE disponible dans GitLab et faire un commit directement dans la branche main)
4. Initialisez une page du wiki du projet avec le texte « Bienvenue sur le wiki du projet »
5. Communiquez votre profil GitLab au formateur qui vous ajoutera dans un projet partagé

Name	Last commit
 README.md	Update README.md

 README.md

## Mon premier projet GitLab

Ce projet va contenir les différents supports de formation de la pépinière IBM i :

- support Introduction Open Source
- Le versioning avec Git
- Développement web
  - HTML/CSS/JS
  - PHP
  - NodeJS
  - Node-red

## Documentation

Pour récupérer le projet, reportez vous aux différentes méthodes de récupération (clonage) du projet.

Reportez vous au wiki pour plus d'informations.



# Le nouveau workload

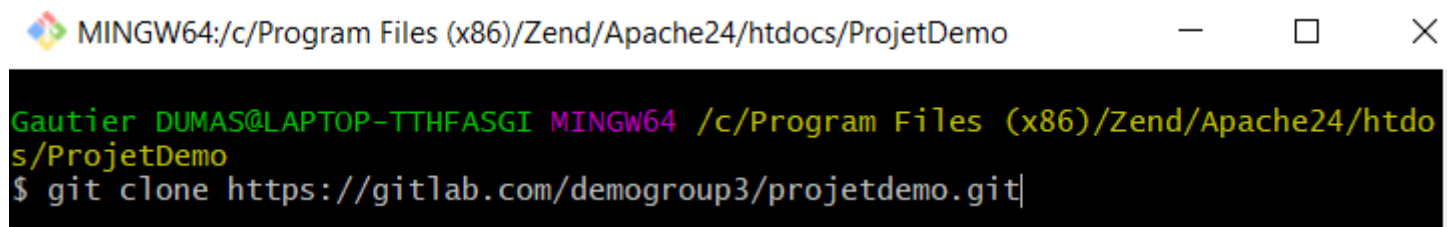
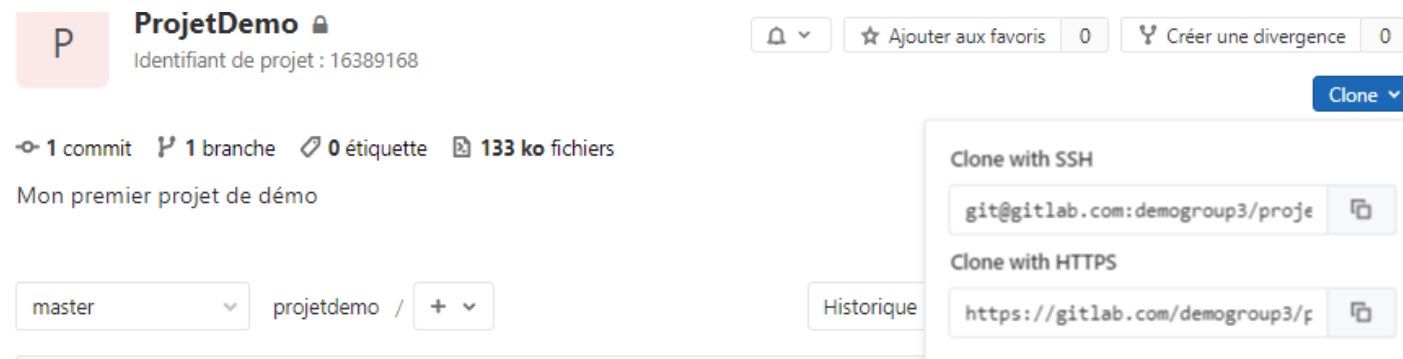
- Par CFD-Innovation

# Récupérer un projet sur le poste

- Pour commencer à travailler sur un projet ou simplement récupérer des sources/documents, il faut commencer par le cloner sur le poste du développeur

```
git clone https://adresse_du_projet
```

- L'adresse est à récupérer dans le projet GitLab (2 protocoles au choix : SSH ou HTTPS)



# Récupérer un projet sur le poste

```
MINGW64:/c:/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c:/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo
$ git clone https://gitlab.com/demogroup3/projetdemo.git
Cloning into 'projetdemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c:/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo
$
```

« Zend » Apache24 » htdocs » ProjetDemo » projetdemo		
Nom	Modifié le	Type
.git	18/01/2020 15:14	Dossier de fichiers
README.md	18/01/2020 15:14	Fichier MD

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c:/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git log
commit b49345db2bb4e4b69460ac0387405cc1b0fb69a6 (HEAD -> master, origin/master, origin/HEAD)
Author: Gautier DUMAS <gaugier.dumas@free.fr>
Date: Sat Jan 18 13:26:58 2020 +0000

Initial commit
```

# Pousser du code sur le remote

- Une fois le code récupéré, nous allons procéder à plusieurs commits locaux (du développement !)
- Ajout de « index.php » dans le répertoire

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

nothing added to commit but untracked files present (use "git add" to track)

Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git add .

Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.php

Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git commit -am "Ajout de index.php - premier commit git local"
[master 60eb93b] Ajout de index.php - premier commit git local
1 file changed, 5 insertions(+)
create mode 100644 index.php
```

# Pousser du code sur le remote

- Modification de « index.php » (deuxième commit pour l'exemple)

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.php

no changes added to commit (use "git add" and/or "git commit -a")

Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git commit -am "Modification de index.php - deuxième commit git local"
[master aa138b3] Modification de index.php - deuxième commit git local
1 file changed, 3 insertions(+), 1 deletion(-)
```

# Pousser du code sur le remote

- Enfin, voici le moment de pousser le code sur GitLab

```
git push origin master
```

- Cette commande demande à Git : « Envoie mes modifications dans la branche master de mon remote origin »
- **origin** est le nom par défaut donné au remote
- **master** représente la branche qui contient le code courant

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/ProjetDemo/projetdemo (master)
$ git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 695 bytes | 695.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0)
To https://gitlab.com/demogroup3/projetdemo.git
b49345d..aa138b3 master -> master
```

DemoGroup > ProjetDemo > Commits

master ▼ projetdemo

18 jan., 2020 3 commits



Modification de index.php - deuxième commit git local  
Gautier DUMAS a créé il y a 7 minutes



Ajout de index.php - premier commit git local  
Gautier DUMAS a créé il y a 8 minutes



Initial commit  
Gautier DUMAS a créé il y a une heure



# Récupérer le travail des autres

- De GitLab au repo local

```
git pull origin master
```

- A faire régulièrement et absolument avant chaque push
- Permet de récupérer les développements / modifications des autres contributeurs avant de pousser ses propres modifications afin d'être synchro.

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/Projet
Demo/projetdemo (master)
$ git pull origin master
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://gitlab.com/demogroup3/projetdemo
* branch          master      -> FETCH_HEAD
   aa138b3..e4b7594 master     -> origin/master
Updating aa138b3..e4b7594
Fast-forward
 developer1.txt | 3 +++
 index.php      | 4 +++-
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 developer1.txt
```

# Fiche récapitulative du workload

//récupérer le projet en local

- git clone https://adresse\_projet\_remote

//Développer

- git add .
- git commit -am "message"
- git commit -am "message"
- ...
- git commit -am "message"
- git commit -am "message"

//Envoyer son code sur le remote en récupérant avant les modifications des collaborateurs

- git pull origin master
- git push origin master

# Importer son projet git local dans remote

- Nous avons notre repository git local, contenant l'ensemble du code et son historique sur le poste (ou l'IFS) du développeur
- La phase d'initialisation consiste à remonter le dépôt dans un projet GitLab remote
  - Pour les sauvegardes (duplication du code et de l'historique)
  - Pour le travail collaboratif

```
cd existing_repo
git remote rename origin old-origin
git remote add origin https://gitlab.com/demogroup3/importdemo.git
git push -u origin --all
git push -u origin --tags
```

# Importer son projet local dans remote

- Pour importer son projet local non encore git (simple répertoire non versionné)

```
cd existing_repo  
git init  
git remote add origin https://gitlab.com/demogroup3/importdemo2.git  
git add .  
git commit -m "Initial commit"  
git push -u origin master
```

# TP3 - GitLab

1. Depuis l'interface web GitLab, dans le dossier « espace personnel », créez un sous-groupe « formation ».
2. A l'intérieur de ce sous-groupe, créez un projet à votre nom (première lettre du prénom et votre nom de famille).  
Exemple : Gautier DUMAS => gdumas

## **Du Local -> GitLab :**

3. Pousser votre dépôt local du TP1 dans le projet formation->gdumas créé au point précédent
4. Ajoutez des fichiers dans votre dossier et poussez les sur le remote pour pouvoir les retrouver depuis l'interface GitLab

## **De GitLab -> Local :**

5. Récupérez le contenu du projet GitLab espace\_perso->formation->gdumas dans un nouveau dossier local « supportsWebEtOpenSource » à créer au même niveau que « formationWeb » créé dans le TP1

# TP3 - GitLab

Pour les 3 prochaines questions, nous allons travailler avec le premier projet collaboratif : <https://gitlab.com/pepiniereibmi2022/formation>

6. Vérifiez que vous y avez bien accès et cloner le sur votre poste local (la VM)
7. Ajoutez un fichier à votre nom dans le dossier local cloné  
*Exemple : gdumas.txt*
8. Versionnez le et poussez le dans le remote pour le partager avec les autres membres de l'équipe
9. Récupérez les fichiers des différents membres de l'équipe à l'aide de l'instruction `git pull`



# Pour aller plus loin

- Par CFD-Innovation

# Création de branches

- Permettent de travailler sur des versions de code qui divergent de la branche principale contenant le code courant
  - Pour développer une nouvelle fonctionnalité en test, n'étant pas sûr de l'intégrer un jour
  - Pour développer une nouvelle fonctionnalité longue sans impacter le master/main (pour appliquer des correctifs par exemple)

- Pour consulter les branches d'un dépôt (par défaut, branche master uniquement)

```
git branch
```

- Pour créer une nouvelle branche

```
git branch ma_nouvelle_branche
```

- Pour se placer dans une branche (une nouvelle ou dans la courante)

```
git checkout ma_nouvelle_branche  
ou  
git checkout main
```



# Fusion de branches

- Une fois le développement terminé (dev, test, recette...) dans la branche de développement, il faut la fusionner avec la branche principale

1. Se positionner sur la branche main

```
git checkout main
```

2. Fusionner avec la branche souhaitée

```
git merge ma_branche
```

- L'inverse est aussi possible pour récupérer dans une branche les modifications réalisées dans le main (pour ne pas trop s'éloigner du code courant)

1. Se positionner sur la branche souhaitée

```
git checkout ma_branche
```

2. Fusionner avec la branche main

```
git merge main
```

# Résoudre des conflits

- Dans le déroulement courant d'un projet de développement, la fusion de branches pourra provoquer des conflits
  - Lorsque plusieurs personnes travaillent en même temps sur un même fichier
  - Lorsqu'un même fichier est modifié différemment dans différentes branches
- Ces conflits sont alors indiqués dans la console suite au « git merge xxx »

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet (master)
$ git merge newbranch
Auto-merging index.php
CONFLICT (content): Merge conflict in index.php
Automatic merge failed; fix conflicts and then commit the result.
```

- Vous pouvez éditer ce conflit en ouvrant directement le(s) fichier(s) en conflit avec votre éditeur ou IDE habituel



```
php x .gitignore x index.php x
<?php
<<<<<< HEAD
echo "Mon premier projet versionné - en français";
=====
echo "My first versionned project";
>>>>>> newbranch
echo "La deuxième version de mon projet";
```



```
php x .gitignore x index.php x
<?php
echo "My first versionned project";
echo "La deuxième version de mon projet";
```

```
Gautier DUMAS@LAPTOP-TTHFASGI MINGW64 /c/Program Files (x86)/Zend/Apache24/htdocs/monprojet (master|MERGING)
$ git commit -a
[master 0d0965e] Merge branch 'newbranch'
```

# Gestion des conflits

- L'un des problèmes les plus récurrents qui peut arriver lorsque plusieurs développeurs travaillent sur une même branche est l'apparition de conflits.
- Prenons comme exemple un développeur, **Dev1**, travaillant sur un projet versionné.
- Un second développeur, **Dev2**, travaille également sur ce même projet.
- **Dev1** et **Dev2** travaillent chacun sur une partie différente du projet, cependant, des fichiers sont en communs dans ces deux parties.
- **Dev1** push son travail, tout se passe normalement.
- Vient maintenant le tour de **Dev2** de push :

```
$ git push
warning: redirecting to https://gitlab.com/pbec-cfd/formation-git.git/
To https://gitlab.com/pbec-cfd/formation-git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://gitlab.com/pbec-cfd/formation-git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

# Gestion des conflits

- En effet, lorsque **Dev2** tente de push, cela échoue car il tente de push sur une version du répertoire « périmé » qu'il a en local. Tout cela étant dû au fait que **Dev1** et **Dev2** aient travaillé en même temps et que **Dev2** ai push en premier.
- Pour résoudre ce problème, il faut mettre à jour le répertoire local afin de pouvoir re-push. De la, deux possibilités :
  - Soit le « git pull » se passe bien et git arrive à mélanger les modifications seul et dans ce cas il suffit de re-push derrière.
  - Soit le « git pull » échoue car git n'arrive pas à mélanger les modifications seul et il s'agit alors d'un conflit :

```
$ git pull
warning: redirecting to https://gitlab.com/pbec-cfd/formation-git.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 273 bytes | 1024 bytes/s, done.
From https://gitlab.com/pbec-cfd/formation-git
   d9f0caa..16b5ee2  main       -> origin/main
Auto-merging fichier.txt
CONFLICT (content): Merge conflict in fichier.txt
Automatic merge failed; fix conflicts and then commit the result.
```

# Gestion des conflits

- Une fois que le conflit est constaté, les fichiers listés et mis en cause sont modifiés et, aux endroits posant problèmes, ressembleront à ca :

```
C'est un fichier
<<<<<< HEAD
C'est un conflit !
=====
C'est un premier édit !
>>>>>> 16b5ee2c42904b1ee60d8877fd4c89770b379307
```

- Avec, suivant le « HEAD » les changements de **Dev2** qui n'ont pas pu être push
- Et en second les modifications d'ores et déjà push sur le répertoire distant et push par **Dev1**

# Gestion des conflits

- Il faut alors modifier les fichiers afin de corriger les conflits que Git n'a pas su corriger seul :

```
C'est un fichier | Bec, 1  
C'est un premier édit !  
C'est un conflit !
```

- Maintenant que les modifications sont faites, on peut re-tenter de push :

```
Pierre BEC@LAPTOP-BMF8B71J MINGW64 /d/Pierre BEC/OneDrive - CFD-INNOVATION/Forma  
tions/Formation Inter/Git/formation-git (main|MERGING)  
$ git add fichier.txt  
  
Pierre BEC@LAPTOP-BMF8B71J MINGW64 /d/Pierre BEC/OneDrive - CFD-INNOVATION/Forma  
tions/Formation Inter/Git/formation-git (main|MERGING)  
$ git commit -m "résolution de conflit"  
[main b9ef5a5] r|@solution de conflit  
  
Pierre BEC@LAPTOP-BMF8B71J MINGW64 /d/Pierre BEC/OneDrive - CFD-INNOVATION/Forma  
tions/Formation Inter/Git/formation-git (main)  
$ git push  
warning: redirecting to https://gitlab.com/pbec-cfd/formation-git.git/  
Enumerating objects: 10, done.  
Counting objects: 100% (10/10), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (6/6), 629 bytes | 314.00 KiB/s, done.  
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0  
To https://gitlab.com/pbec-cfd/formation-git  
16b5ee2..b9ef5a5 main -> main
```

# git stash

- Si vous devez revenir en urgence (pour un correctif par exemple) sur le master alors que vous n'avez pas terminé votre développement dans une branche, plutôt que de faire un commit « à moitié fait », vous pouvez mettre de côté le code modifié avec la commande git stash.

```
git stash
```

- Pour récupérer les modifications en cours et une fois revenue sur la branche de développement, utilisez la commande git stash pop.

```
git stash pop
```

- N'oubliez jamais de faire un « git commit » ou « git stash » avant de changer de branche sous peine de perdre vos modifications

# Retrouvez qui a fait une modification

- Avec la commande git blame, vous pouvez consulter l'historique ligne à ligne d'un fichier. Nous retrouvons pour chaque ligne l'identifiant du dernier commit

```
git blame nom_du_fichier.php
```

- Pour plus de détails sur le commit en question, git show

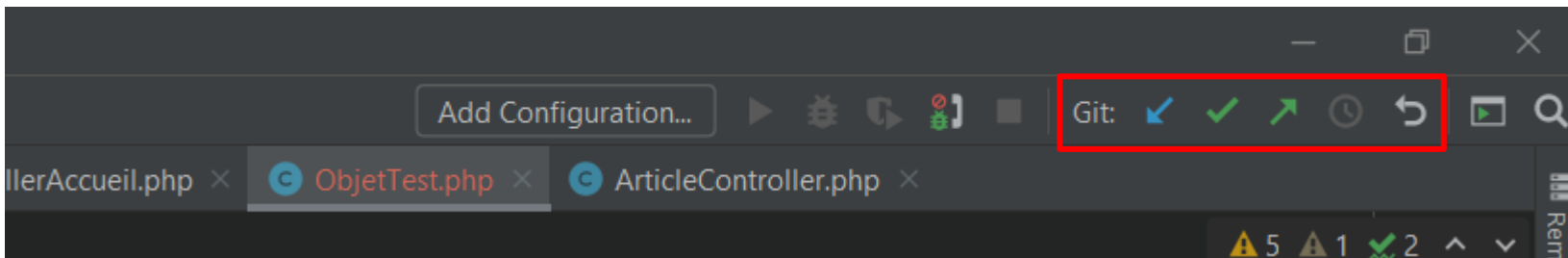
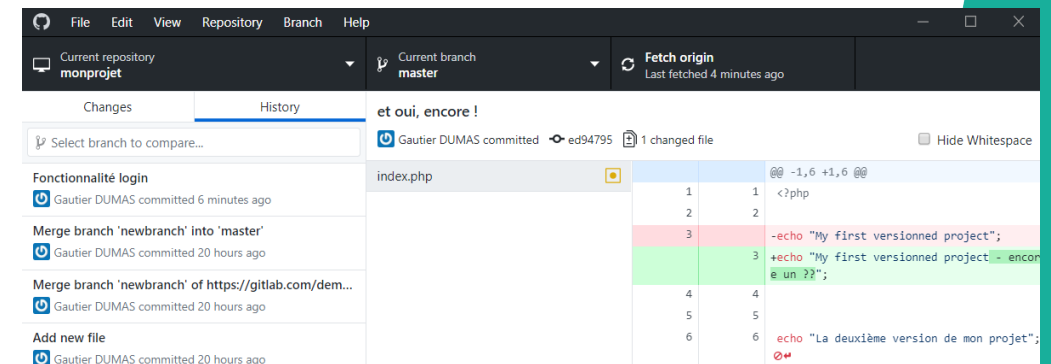
```
git show ID_DU_COMMIT
```

- Vous avez donc une explication (dans le message du commit), l'auteur et la date de la modification.



# Utilisation d'un client GIT graphique

- Il en existe beaucoup !
- Desktop GitHub :
  - Solution Open Source et gratuite
  - propose une interface graphique simple et intuitive
  - peut se connecter au dépôt GitLab
  - Téléchargement sur <https://desktop.github.com/>
- Intégration dans PHPStorm
- Intégration dans VSCode (*va être détaillé dans la suite de la formation*)



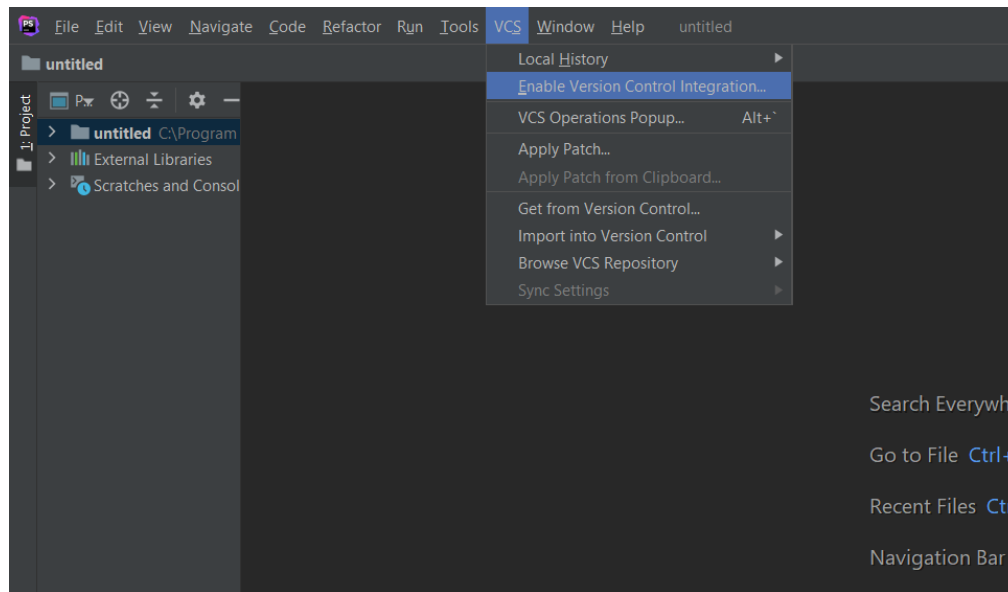


# Intégration git PHPStorm

- Par CFD-Innovation

# Utilisation de Git via PHP Storm (1/4)

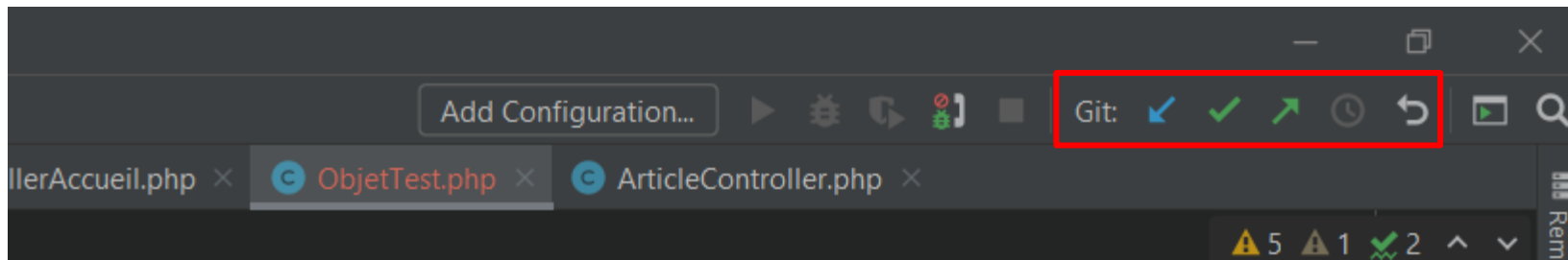
- Si vous voulez versionner un projet déjà existant il faudra activer le versioning depuis le menu



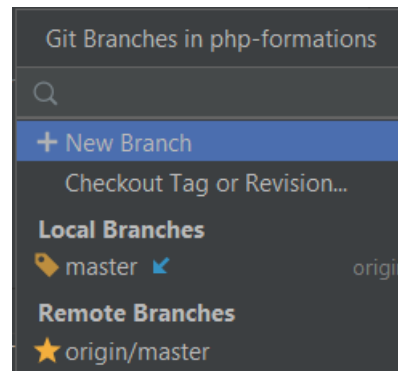
- Si vous démarrez un projet depuis un dépôt Git existant alors le versioning sera activé de base sur le projet

# Utilisation de Git via PHP Storm (2/4)

- Afin de faciliter l'utilisation de git, un plugin vous permet de gérer votre VCS via une interface

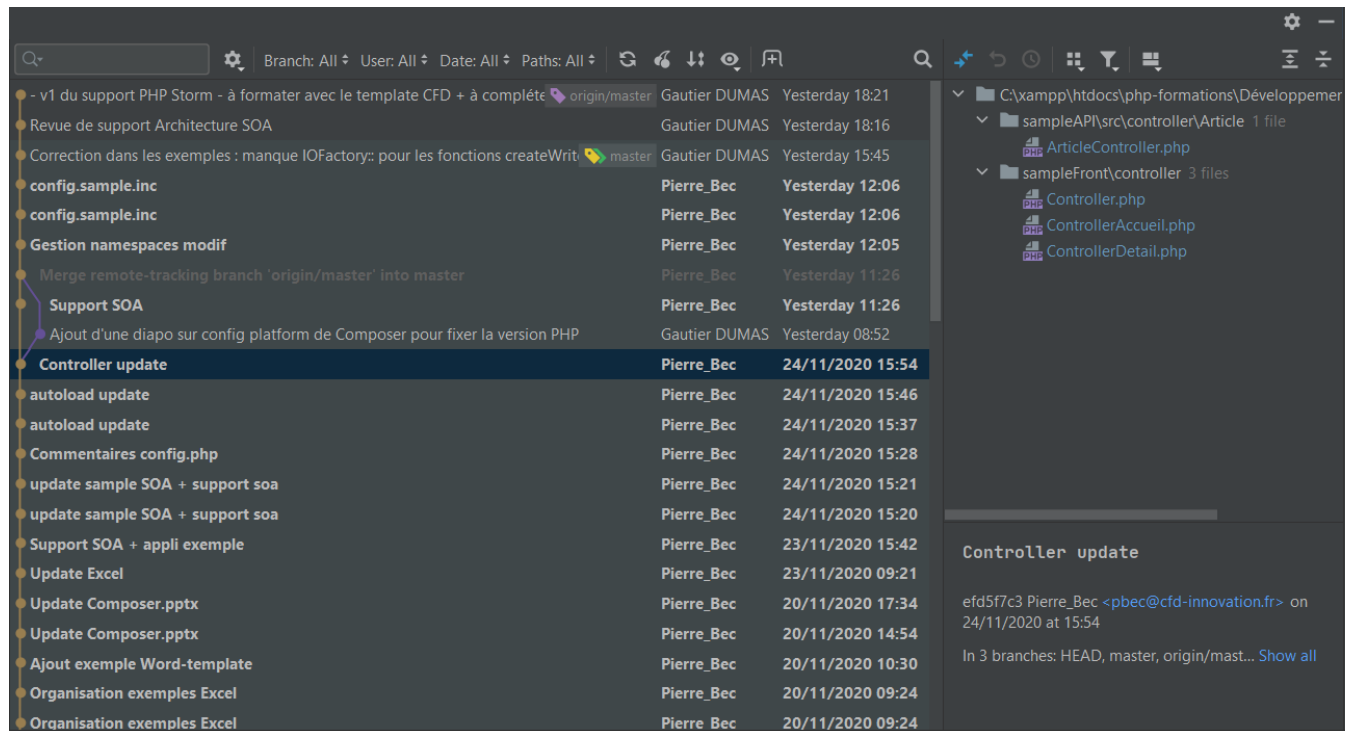


- Ce plugin fournit également un menu de gestion des branches en bas à droite



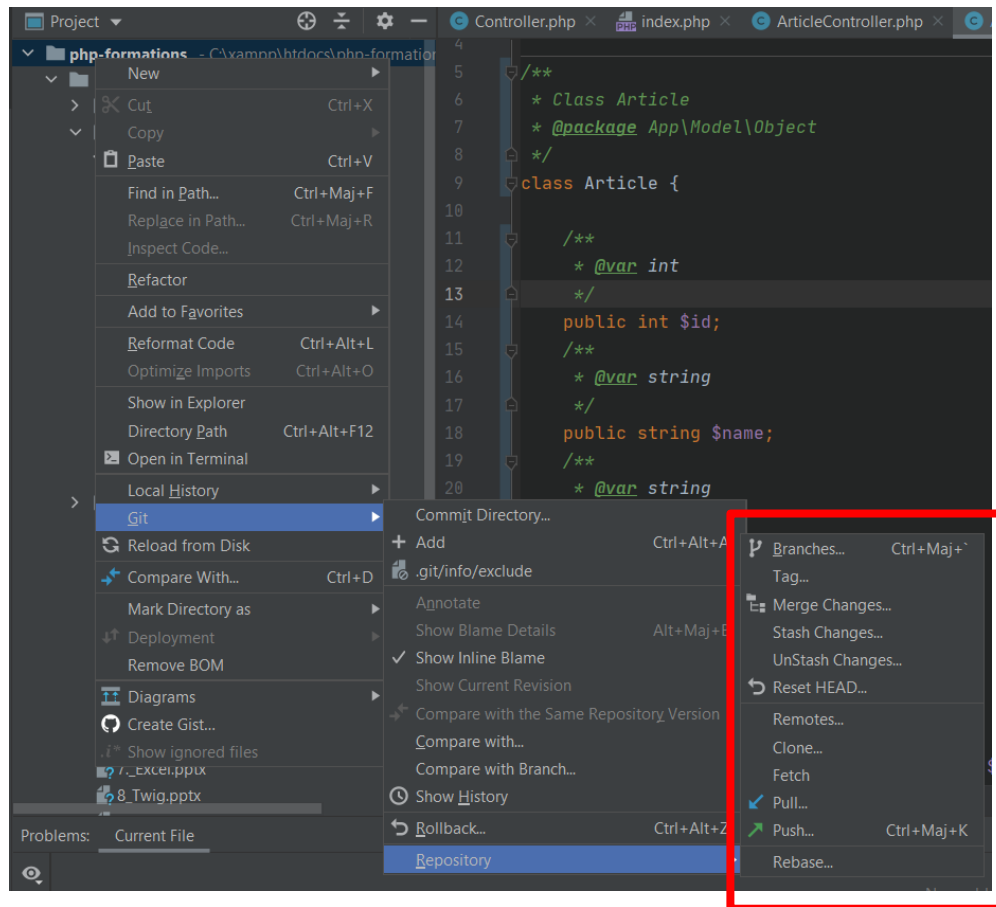
# Utilisation de Git via PHP Storm (3/4)

- Il vous est aussi mis à disposition un menu très complet vous permettant de visualiser l'évolution des branches avec un grand nombre d'actions possibles.
- Pour plus d'informations se référer à la documentation :  
<https://www.jetbrains.com/help/phpstorm/using-git-integration.html>



# Utilisation de Git via PHP Storm (4/4)

- Attention : L'interface Git en haut à droite agit sur l'ensemble du projet PHPStorm
- Si vous avez plusieurs dépôts différents dans votre projet :
  - Utilisez le menu contextuel (clique droit sur le dossier)





Visual Studio Code

# Intégration git VSCoode

- Par CFD-Innovation

# Démarrer un projet

- Deux solutions pour démarrer un projet :
  - Ouvrir un dossier existant et versionné
  - Cloner un dépôt Git

Pour utiliser des fonctionnalités git, vous pouvez ouvrir un dossier contenant un dépôt git ou le cloner à partir d'une URL.

Ouvrir un dossier


Cloner le dépôt

Pour en savoir plus sur la façon d'utiliser git et le contrôle de code source dans VS Code lisez [nos documents](#).

- Afin de cloner un dépôt, il est nécessaire de
  - Renseigner l'url du dépôt
  - Sélectionner l'emplacement de travail local où sauvegarder le dépôt cloné

`https://gitlab.com/cfd-innovation/skeleton/apiws`

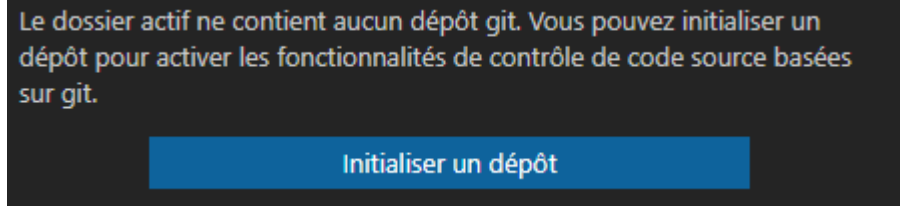
URL de dépôt `https://gitlab.com/cfd-innovation/skeleton/apiws`

 Cloner à partir de GitHub



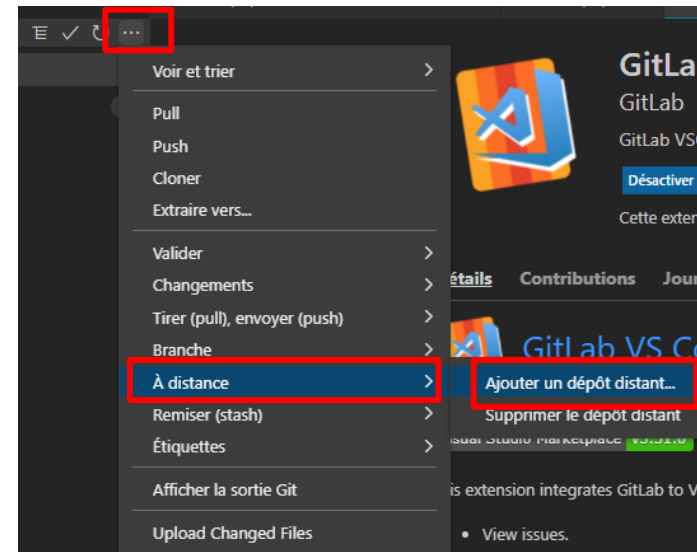
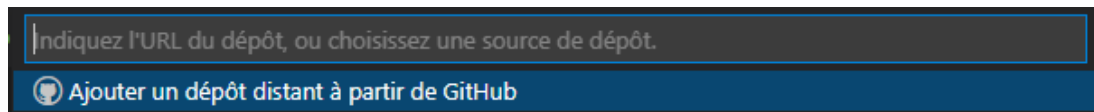
# Démarrer un projet

- Afin de lier un nouveau projet à un dépôt distant, il faut commencer par initialiser le versioning du projet :



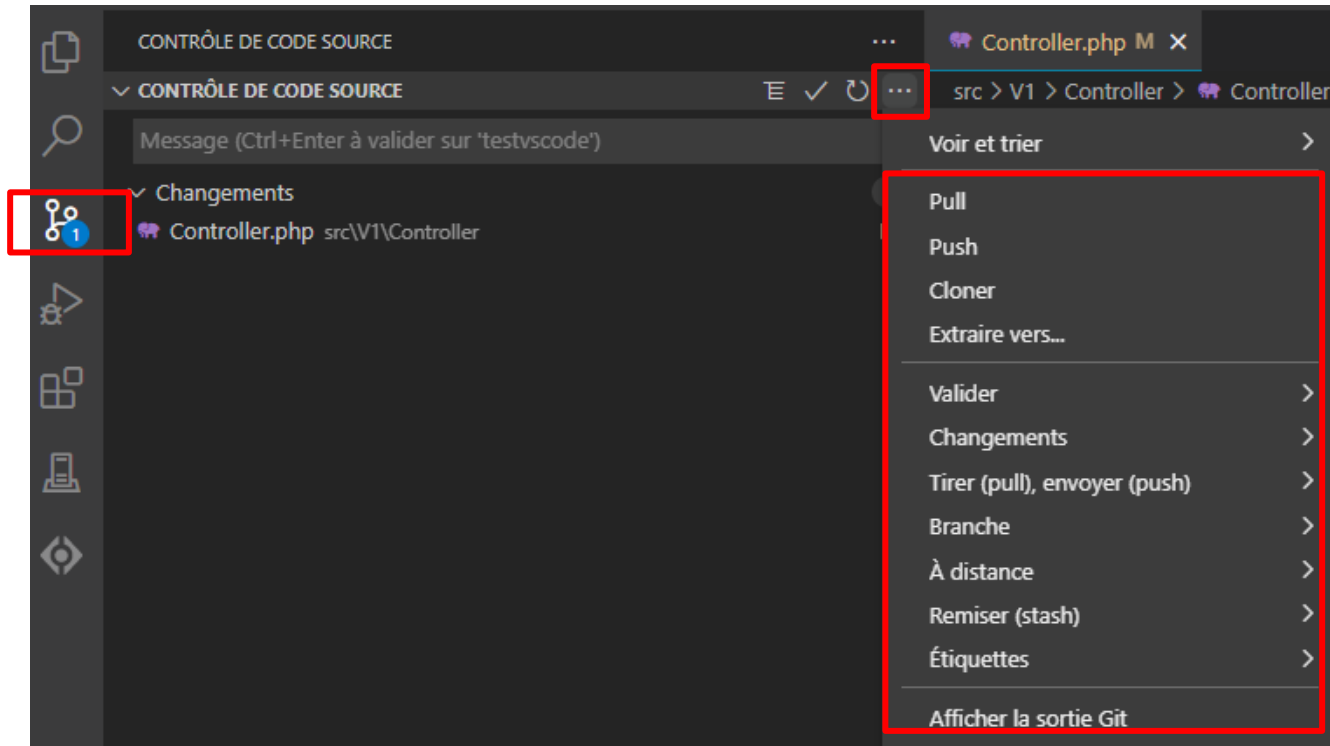
- Une fois le dépôt local initialisé, il faut le lier à un dépôt distant via le menu général de versioning :

- Il faut alors renseigner l'URL du dépôt distant et valider via la touche entrée



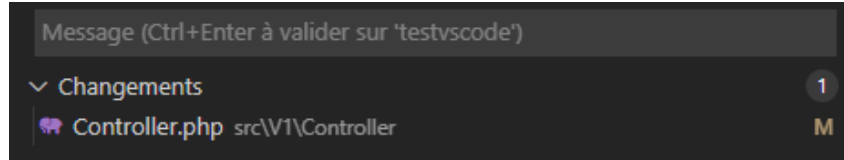
# Fonctionnalités de versioning

- Depuis le menu de versioning, il est possible d'avoir accès aux différentes fonctionnalités :



# Git commit et push

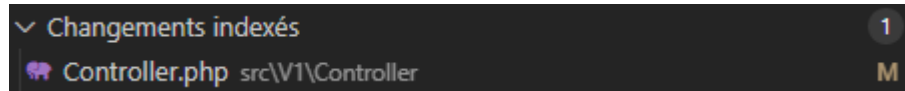
- Lorsque vous modifiez un ou plusieurs fichiers, il apparaissent dans la liste « changements » du menu de versioning.



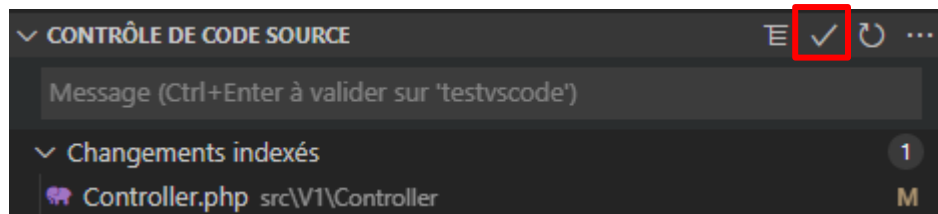
- Afin de « valider » les modifications, il faut les « mettre en attente » via le bouton « + »



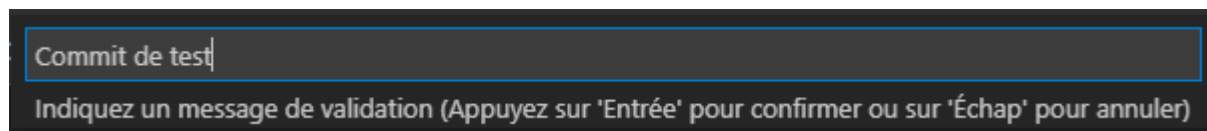
- Ces mêmes modifications apparaissent ensuite dans les changements indexés :



- Il est maintenant possible de faire un commit via le bouton « V » tout en haut de l'interface de versioning.

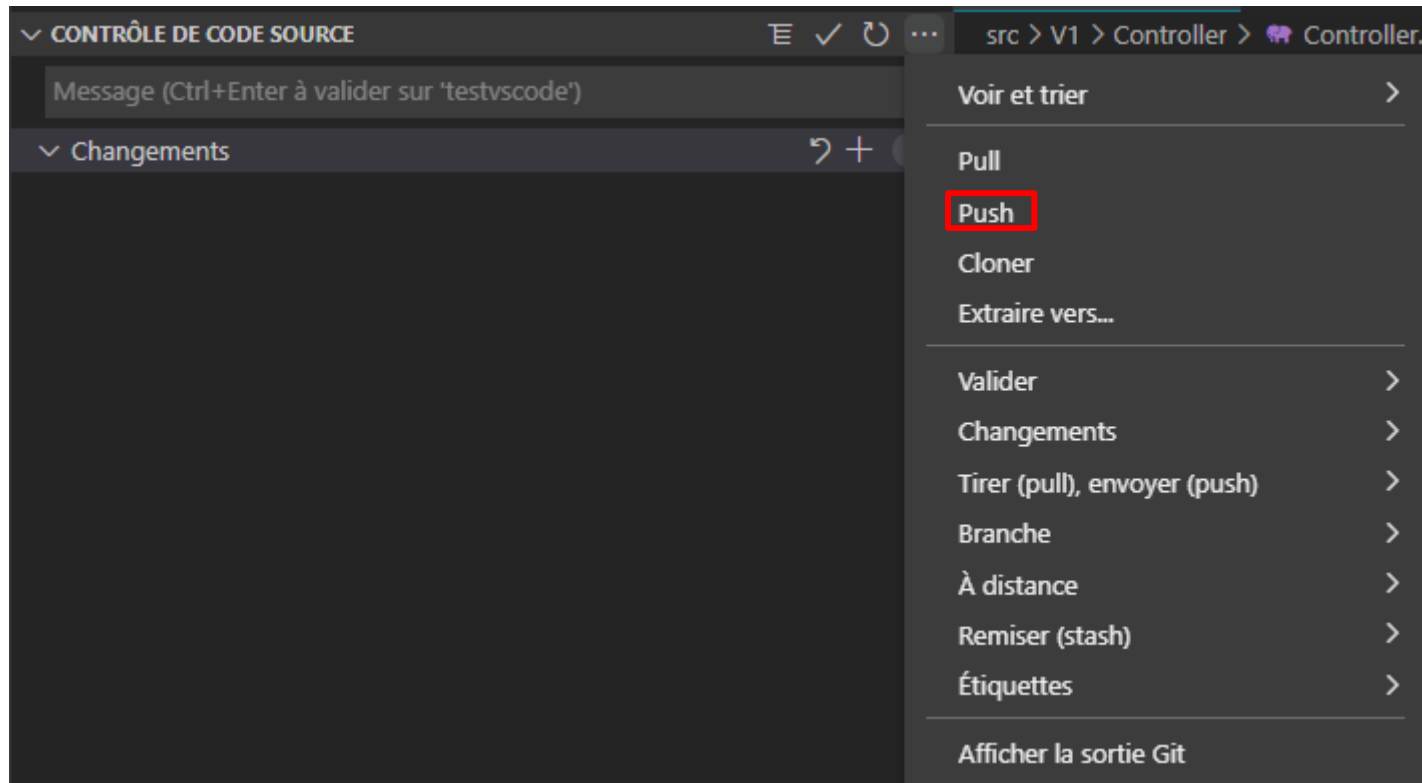


- Il suffit alors d'entrer le message de commit et d'appuyer sur entrée



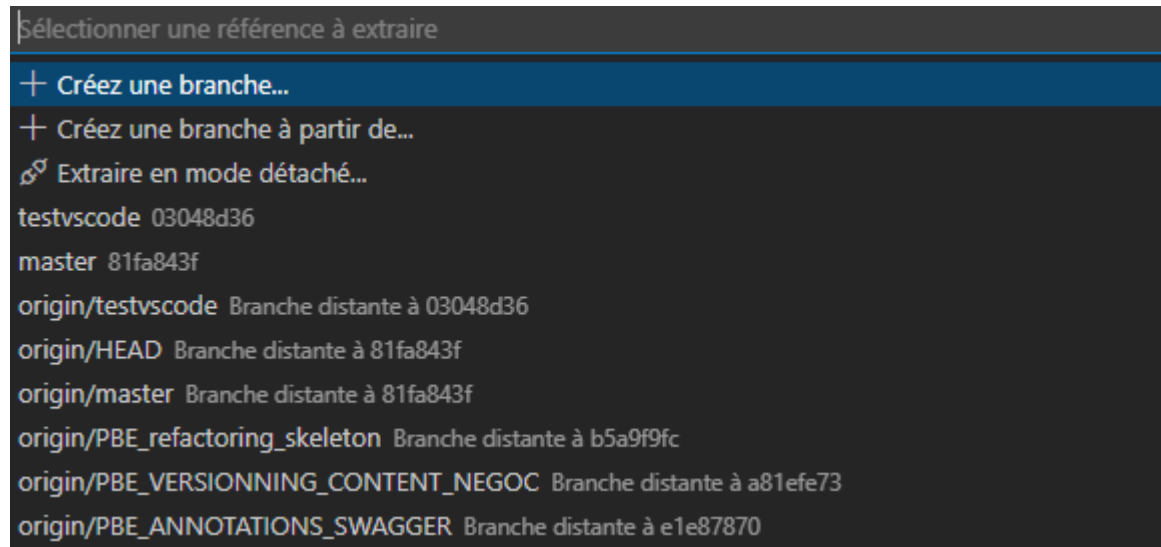
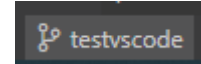
# Git commit et push

- Une fois les changements commités, il faut push via le menu général de versioning pour que les changements soient envoyés sur le dépôt distant



# Gestion des branches

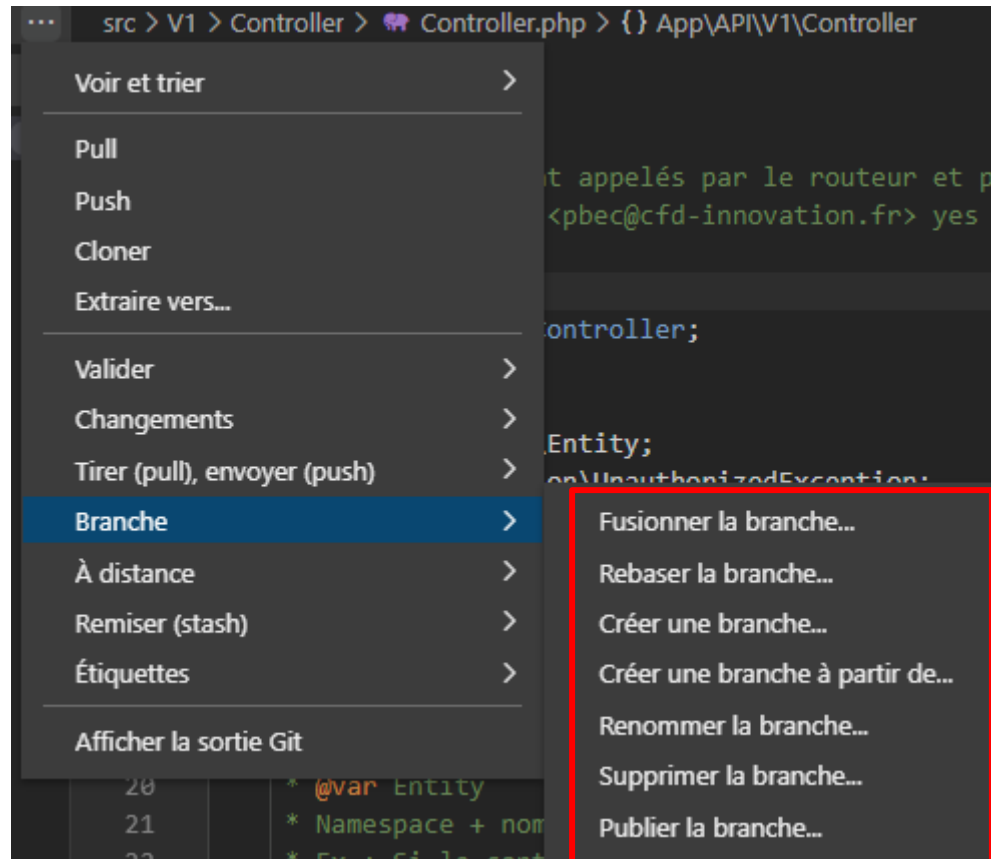
- Tout en bas à gauche de VSCode, le nom de la branche courante est affiché :
- Lors du clique sur ce bouton, l'ensemble des fonctionnalités liées aux branches apparaissent :



- Il est alors possible de créer une nouvelle branche ou alors de switcher sur les branches en local et sur le dépôt distant.

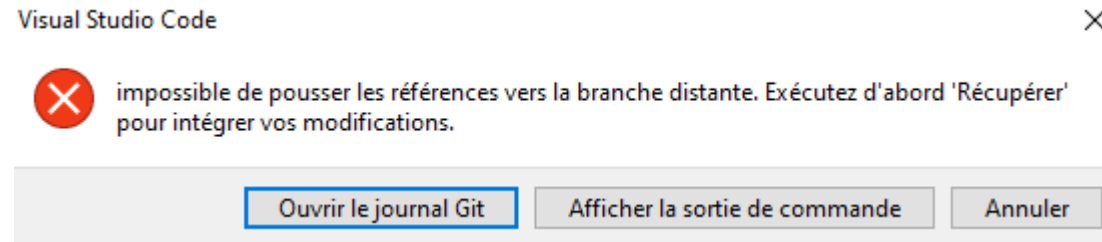
# Gestion des branches

- Il est aussi possible d'avoir accès à plus de fonctionnalités sur la branche actuelle via le menu général de versioning :



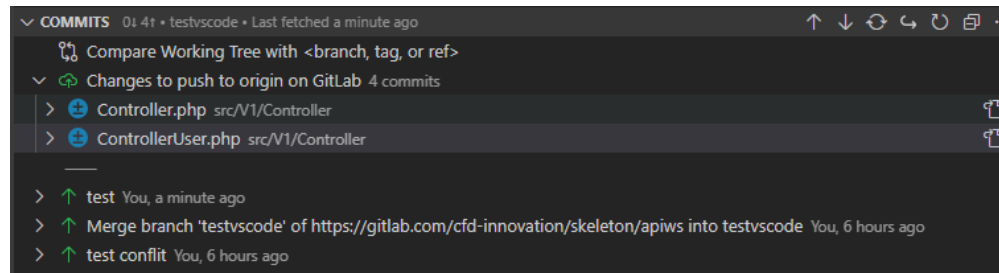
# Gestion des conflits

- Lors d'une tentative de push sur une branche non pull, message d'erreur invitant à pull avant de push.

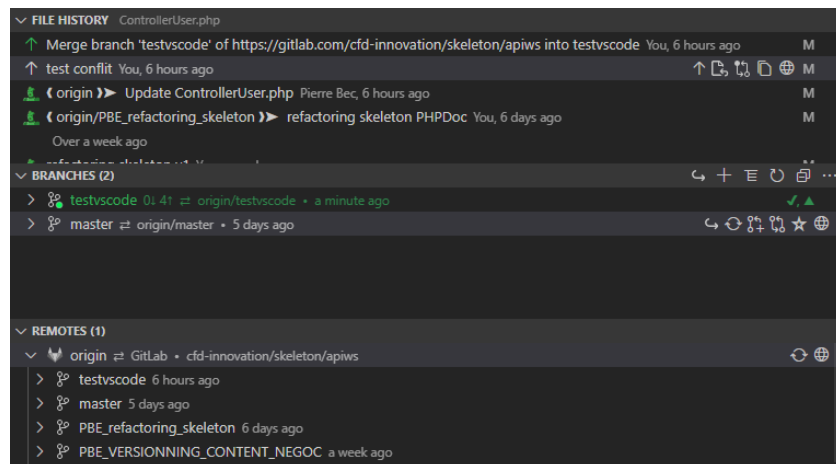


# Git Lens

- Afin d'accéder à plus d'informations comme l'historique des modifications par fichiers, historique des commits, etc... Une extension « Git Lens » existe.
- Elle permet notamment de visualiser l'historique des commits ainsi que ceux qui ne sont pas push.



- D'autres informations tel que le file history, les branches locales et les remotes :





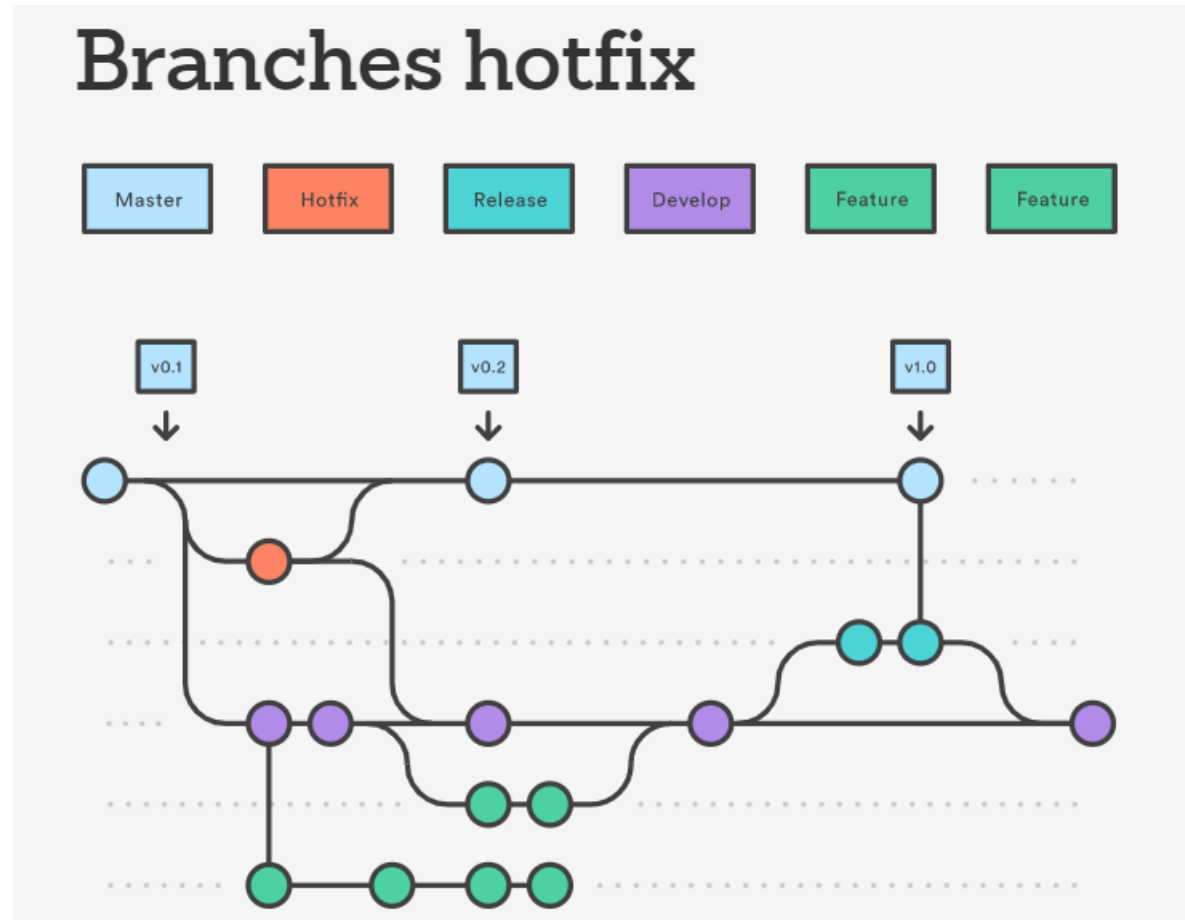


# Organisation - GitFlow

- Par CFD-Innovation

# Git Flow – exemple complet

- <https://www.atlassian.com/fr/git/tutorials/comparing-workflows/gitflow-workflow>



# Les principes par branches features

- La branche **main** (ou anciennement master) représentera les différentes versions de production
- La branche **develop** représentera la version fusionnée des différents développements
- A chaque nouvelle fonctionnalité à développer : une nouvelle branche **feature** permettra d'isoler le développement en cours
- Avant chaque mise en production, passage par une branche **release** (recette)
- Si Hotfix nécessaire sur la prod : branche de **hotfix** par correctifs
- Le fonctionnement par branche permet :
  - De cloisonner les différentes versions du code
  - D'organiser les développements et en connaître leurs états au fur et à mesure
  - De garder l'historique en fonction de chaque fonctionnalité / hotfix
  - De sécuriser en interdisant les modifications directement dans les environnements de production ou sur des branches partagées telles que develop

# Merge Request GitLab

- Création merge request de la branche feature vers la branche develop

The screenshot illustrates the GitLab interface for creating a merge request. It is divided into three main sections: a top navigation bar, a left sidebar, and a main content area.

**Top Navigation Bar:** Displays the GitLab logo, a menu icon, and the project name "bonjour".

**Left Sidebar:** Contains navigation links for "Project information", "Repository", "Issues", and "Merge requests". The "Merge requests" link is highlighted with a blue bar.

**Main Content Area:**


- Project Information:** Shows the project name "bonjour" and a notification: "You pushed to `salut-create` 6 minutes ago". A blue button labeled "Create merge request" is visible.
- New merge request form:** This section is divided into two columns.
  - Source branch:** A dropdown menu showing "gdumas/bonjour" and a text input field containing "salut-create".
  - Target branch:** A dropdown menu showing "gdumas/bonjour" and a text input field containing "develop".
  - Commit information:** Below the source branch, it shows a commit titled "Ajout de salut.php" by "Gautier DUMAS" authored 8 minutes ago, with a commit hash "5a4f234c".
  - Buttons:** A blue button labeled "Compare branches and continue" is located below the commit information.
- Form Fields:**
  - Assignees:** A dropdown menu showing "Gautier DUMAS".
  - Reviewers:** A dropdown menu showing "Unassigned".
  - Milestone:** A dropdown menu showing "Milestone".
  - Labels:** A dropdown menu showing "Labels".
  - Merge request dependencies:** A text input field with a placeholder "Enter merge request URLs or references (e.g. path/to/project/merge\_request\_id)".
  - Merge options:** Two checkboxes: "Delete source branch when merge request is accepted." and "Squash commits when merge request is accepted." (checked).
- Bottom Bar:** Contains a blue button labeled "Create merge request" and a grey button labeled "Cancel".

**Bottom Section:** A section titled "New merge request" showing the "From" and "into" branches. It includes a "Change branches" link and a "Description" field with a "Write" tab selected. The description text reads: "Voici la nouvelle fonctionnalité que je souhaite intégrer dans `develop`."

# Merge Request GitLab - Merge

- Une fois la Merge Request créée, une personne en charge d'une revue de code pourra l'accepter (ou la refuser) en cliquant sur « Merge »


Gautier DUMAS > bonjour > Merge requests > !1


Open Created just now by  **Gautier DUMAS** Maintainer Edit Mark as draft


## Ajout de salut.php




Overview 0 Commits 1 Changes 1


Voici la nouvelle fonctionnalité que je souhaite intégrer dans develop.








 Request to merge **salut-create** into **develop** Open in Web IDE Check out branch

 Approval is optional  
[View eligible approvers](#)

 **Merge** ☐ Delete source branch ☐ Squash commits  
[Adds 1 commit and 1 merge commit to develop. Modify merge commit](#)

 0  0  Oldest first Show all activity

 Gautier DUMAS @gdumas assigned to @gdumas just now

Write Preview **B** *I* **”** **</>**       

Write a comment or drag your files here...

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

Comment Close merge request

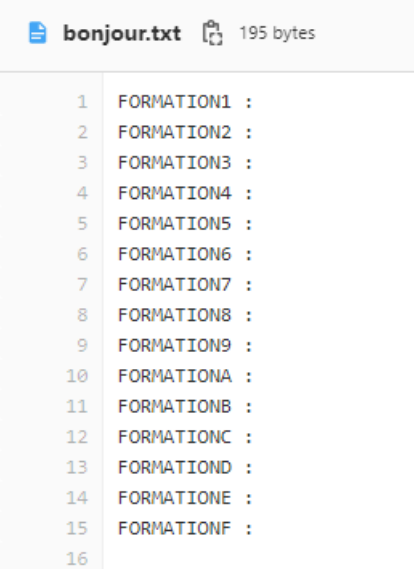
Les développements  
sont maintenant  
intégrés à la branche  
develop.

Le développeur peut  
passer à une autre  
feature

# TP4 – Les branches git

Toujours avec le projet <https://gitlab.com/pepiniereibmi2022/formation> dans votre environnement de développement local (la VM)

1. Lister les branches du projet (git branch -a)
2. Positionnez vous sur la branche develop (git checkout develop)
3. Créez une branche « `bonjour-formationxx` » à partir de develop (où xx est votre numéro de profil) à l'aide de la commande « `git branch bonjour-formationxx` »
4. Positionnez vous dans cette branche (git checkout bonjour-formationxx)
5. Dans le fichier « `bonjour.txt` », ajoutez votre prénom et nom associé à votre profil de formation.  
Pousser le code sur le remote une fois la fonctionnalité terminée (toujours dans votre branche)
6. Effectuez une « Merge Request » de votre branche « `bonjour-formationxx` » vers la branche « **develop** » afin de demander l'intégration de votre fonctionnalité dans la branche commune develop. Assignez le profil du formateur à la merge request pour qu'il valide l'ajout.



The screenshot shows a text editor window titled 'bonjour.txt' with a file icon and '195 bytes' on the right. The editor contains 16 lines of text, each starting with a number from 1 to 16 followed by a colon and the word 'FORMATION' followed by a letter from A to P. The text is as follows:

```
1 FORMATION1 :
2 FORMATION2 :
3 FORMATION3 :
4 FORMATION4 :
5 FORMATION5 :
6 FORMATION6 :
7 FORMATION7 :
8 FORMATION8 :
9 FORMATION9 :
10 FORMATIONA :
11 FORMATIONB :
12 FORMATIONC :
13 FORMATIOND :
14 FORMATIONE :
15 FORMATIONF :
16
```



〈 INNOVATION 〉

*Open your IBM i !*

Gautier DUMAS

gdumas@cf-d-innovation.fr

06 07 52 54 16

@GautierDumas

/in/gautier-dumas

Merci de votre attention

<https://cf-d-innovation.fr>