

Formation Git

VIII.b - Gitlab



Arnaud MERCIER
arnaud.mercier@hexotech.fr



Nous allons découvrir dans ce chapitre la plateforme gitlab qui permet d'héberger des dépôts git et qui apporte des outils complémentaires autour de git pour la gestion de votre projet comme par exemple:

- gestion de tickets
- CI/CD
- Documentation
- Gestion des droits

Versions de GitLab



<https://gitlab.com>

Version hébergée

Version installée

Gitlab est un service en ligne qui permet entre autre d'héberger des dépôts Git. Il est totalement gratuit pour des projets ouverts au public mais il propose également des formules payantes pour les projets que l'on souhaite rendre privés. Vous pouvez l'utiliser soit via un hébergement sur la plateforme ou sur vos serveurs

Gitlab propose également de nombreux autres services très intéressants comme par exemple:

- Partager du code source avec d'autres développeurs.
- Signaler et gérer les problèmes ou bugs de votre code source via les issues.
- Mise en place d'une CI/CD
- Wiki
- Et bien plus encore

Le Wiki

Markdown example - Edit Page Delete page

Title:

Q Tip: You can move this page by adding the path to the beginning of the title. [More information](#).

Format:

Content:

Write Preview

Main title

Title level 2

Title level 3

Title level 4

Simple content

Text

Simple text

> Quote text

--Scratch text--

This "text" is *italic*.

Création d'une page de Wiki

Last edited by **Arnaud MERCIER** 6 months ago Page history New page

Markdown example

Main title

Title level 2

Title level 3

Title level 4

Simple content

Text

Simple text

Quote text

Scratch text

This word is **bold**.

This word is *italic*.

Get link Markdown documentation

Local file link setup py

- [AT001](#)
- [AT002](#)
- [AT003](#)
- [AT004](#)
- [AT005](#)
- [AT006](#)
- [AT007](#)
- [AT008](#)
- [AT009](#)
- [AT010](#)
- [AT011](#)
- [AT012](#)
- [AT013](#)
- [AT014](#)
- [AT015](#)
- [AT016](#)
- [AT017](#)
- [AT018](#)
- [AT019](#)
- [AT020](#)
- [AT021](#)
- [AT022](#)
- [AT023](#)
- [AT024](#)
- [AT025](#)
- [AT026](#)
- [AT027](#)
- [AT028](#)
- [AT029](#)
- [AT030](#)
- [AT031](#)
- [AT032](#)
- [AT033](#)
- [AT034](#)
- [AT035](#)
- [AT036](#)
- [AT037](#)
- [AT038](#)
- [AT039](#)
- [AT040](#)
- [AT041](#)
- [AT042](#)
- [AT043](#)
- [AT044](#)
- [AT045](#)
- [AT046](#)
- [AT047](#)
- [AT048](#)
- [AT049](#)
- [AT050](#)
- [AT051](#)
- [AT052](#)
- [AT053](#)
- [AT054](#)
- [AT055](#)
- [AT056](#)
- [AT057](#)
- [AT058](#)
- [AT059](#)
- [AT060](#)
- [AT061](#)
- [AT062](#)
- [AT063](#)
- [AT064](#)
- [AT065](#)
- [AT066](#)
- [AT067](#)
- [AT068](#)
- [AT069](#)
- [AT070](#)
- [AT071](#)
- [AT072](#)
- [AT073](#)
- [AT074](#)
- [AT075](#)
- [AT076](#)
- [AT077](#)
- [AT078](#)
- [AT079](#)
- [AT080](#)
- [AT081](#)
- [AT082](#)
- [AT083](#)
- [AT084](#)
- [AT085](#)
- [AT086](#)
- [AT087](#)
- [AT088](#)
- [AT089](#)
- [AT090](#)
- [AT091](#)
- [AT092](#)
- [AT093](#)
- [AT094](#)
- [AT095](#)
- [AT096](#)
- [AT097](#)
- [AT098](#)
- [AT099](#)
- [AT100](#)

MD example

- Direct page link
- Direct file link

other pages

- Link to a future page
- Logbook

Documentation

- Links in markdown

Visualisation du wiki d'un projet

Le wiki vous permet de créer de la documentation pour chaque projets.
Il est possible de rédiger les wiki dans plusieurs langages comme par exemple le markdown
Il y a un suivi de version des modifications (gestion sous git)

Les issues

The 'New Issue' form is a web interface for creating a new task. It includes a title field with the placeholder 'Button save generate an internal error', a type dropdown set to 'Issue', and a description field with a rich text editor. The description contains a detailed bug report with steps to reproduce the error. Below the description, there are checkboxes for 'This issue is confidential' and 'Attach a file'. At the bottom, there are dropdowns for 'Assignee' (Arnaud MERCIER), 'Milestone' (V01R01), and 'Due date' (2022-02-28). A green 'Submit issue' button and a grey 'Cancel' button are at the bottom right.

Créer une nouvelle issue

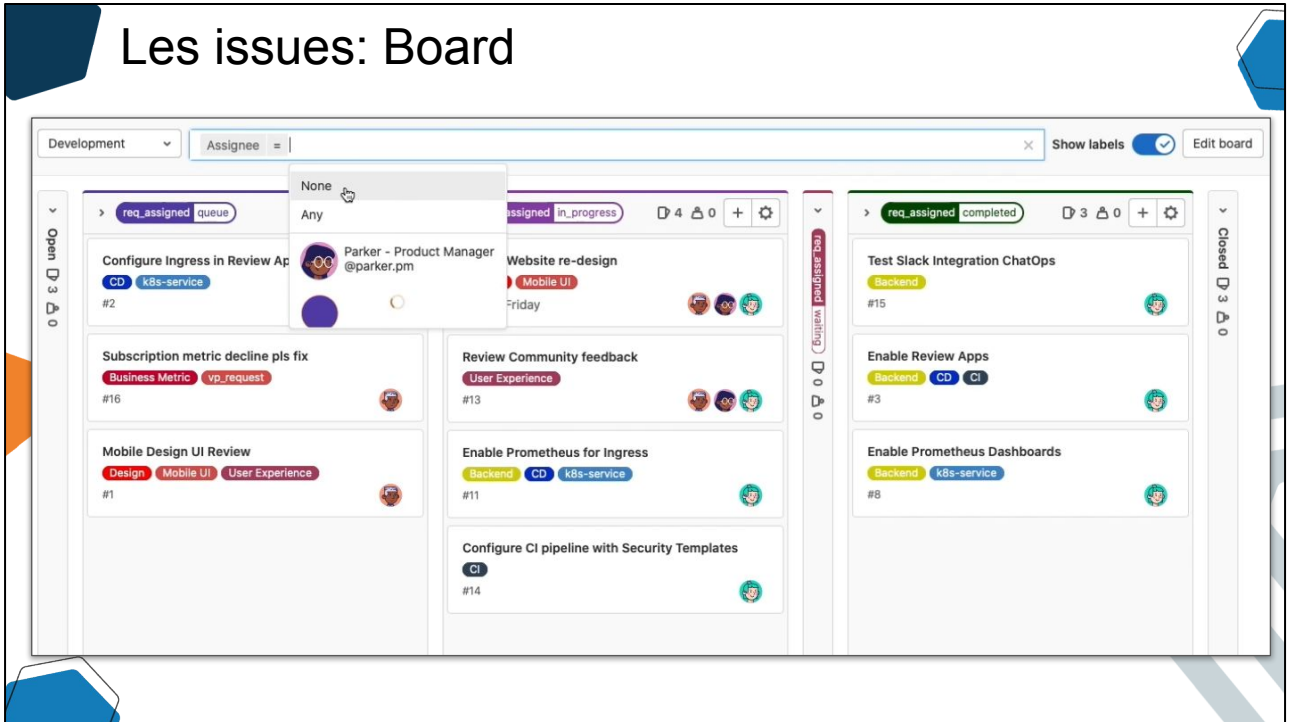
The 'Visualiser/éditer une issue' view shows the details of the issue created in the previous screenshot. It includes a title, a description, and a list of steps to reproduce the error. On the right side, there is a sidebar with various metadata fields: Assignee (Arnaud MERCIER), Epic (This feature is locked, Upgrade plan), Milestone (V01R01), Time tracking (No estimate or time spent), Due date (Feb 28, 2022), Labels (None), Weight, Confidentiality (This issue is confidential), Lock issue (Unlocked), 1 participant, Notifications (checked), and Reference develop/ops_template... A 'Move issue' button is at the bottom right of the sidebar.

Visualiser/éditer une issue

Les issues sont gérés dans une base de donnée. l'objectif est de lister les tâches (bug, dev, doc, ...) et d'y attacher un ensemble de metadatas:

- Type
- Status
- dates de création
- Jalon cible
- Assigné à
- ...

Les issues: Board



Il est possible de visualiser les issues sous forme de board et de les gérer via un glissé déplacé (comme jira)

Les dépôts

The screenshot shows the GitLab interface for a repository named 'cpp_template'. At the top, it displays 'Project ID: 7', 'Template', and 'C++'. Below this, it indicates '3 Commits', '1 Branch', '0 Tags', '410 KB Files', and '410 KB Storage'. A note states 'This repository is a c/cpp project template.' The main section shows a table of files with columns 'Name', 'Last commit', and 'Last update'. The files listed are: doc, example, include/template, src, test, .gitignore, .gitlab-ci.yml, CMakeLists.txt, README.md, licences.txt, logo.png, and template.pc.in. All files were last updated '11 months ago' with the commit 'add template files for a cpp project (lib)'. A blue button labeled 'Dépôt sous Gitlab' is positioned at the bottom of the file list.

Name	Last commit	Last update
doc	add template files for a cpp project (lib)	11 months ago
example	add template files for a cpp project (lib)	11 months ago
include/template	add template files for a cpp project (lib)	11 months ago
src	add template files for a cpp project (lib)	11 months ago
test	add template files for a cpp project (lib)	11 months ago
.gitignore	add template files for a cpp project (lib)	11 months ago
.gitlab-ci.yml	add template files for a cpp project (lib)	11 months ago
CMakeLists.txt	add template files for a cpp project (lib)	11 months ago
README.md	add template files for a cpp project (lib)	11 months ago
licences.txt	Add licence file	11 months ago
logo.png	add template files for a cpp project (lib)	11 months ago
template.pc.in	add template files for a cpp project (lib)	11 months ago

The screenshot shows the content of the 'README.md' file. It starts with the text 'this folder is a template for a C/C++ project'. Below this, there are sections for 'Build', 'Install', and 'Tests'. The 'Build' section contains the command '\$ make build'. The 'Install' section contains the command '\$ make install'. The 'Tests' section contains the commands '\$ make test' and '\$ cpptest --project=compile_commands.json'. A blue button labeled 'Readme associé' is positioned at the bottom right of the README content.

```
this folder is a template for a C/C++ project

Build

$ make build

Install

$ make install

You can change the install destination folder

$ make install

Tests

$ make test
$ cpptest --project=compile_commands.json
```

La fonction majeure de gitlab, reste la gestion de dépôt git.
Il est possible de visualiser l'historique et le contenu du code.

La page d'accueil permet également de visualiser le fichier readme.md du projet

Les dépôts: Commits

master python_template Author Search by message

12 Dec, 2021 2 commits

[RELEASE] V1.8.0 *** b0f1e3d9

Arnaud MERCIER authored 2 months ago

[DOC] Improve release scheme & documentation *** 086152c5

Arnaud MERCIER authored 2 months ago

09 Dec, 2021 2 commits

[VERSION] 1.8.0 bfdeacd7

Arnaud MERCIER authored 2 months ago

[PACKAGES] Replace exact package versions by >= *** 23937c1f

Arnaud MERCIER authored 2 months ago

Visualiser les commits

Commit b0f1e3d9 authored 2 months ago by Arnaud MERCIER Browse files Options

[RELEASE] V1.8.0

- [DOC] Describe how to install custom package
- [DOC] Improve release scheme & documentation
- [PACKAGES] Replace exact package versions by >=

parents ada737fc 086152c5

No related merge requests found

Pipeline #2488 passed with stage in 1 minute and 21 seconds

Changes 4 Pipelines 1

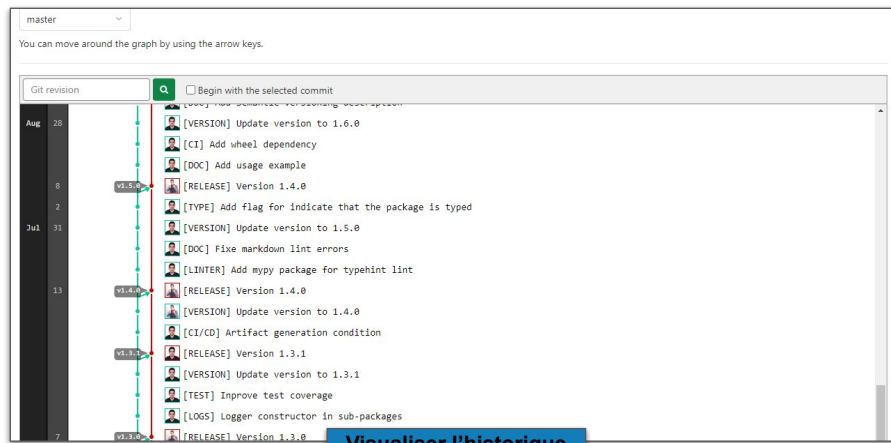
Showing 4 changed files with 114 additions and 24 deletions Hide whitespace changes Inline Side-by-side

▼ README.md View file @b0f1e3d9

```
1 1 # Python Template #
2 2
3 3 ![Pasquil](logo.png)
4 4
5 5 This folder is a template for a python project
6 6
7 7 ## Mind Set ##
8 8
9 9 ... @ -144,9 +142,10 @@ if __name__ == "__main__":
10 10     git remote update
11 11     ...
12 12
13 13 2. Create branch and checkout it
14 14 2. Create branch from develop and checkout it
15 15
16 16 '''bash
17 17 git checkout develop
18 18 git checkout -b {PROJECT_NAME}
19 19
20 20
```

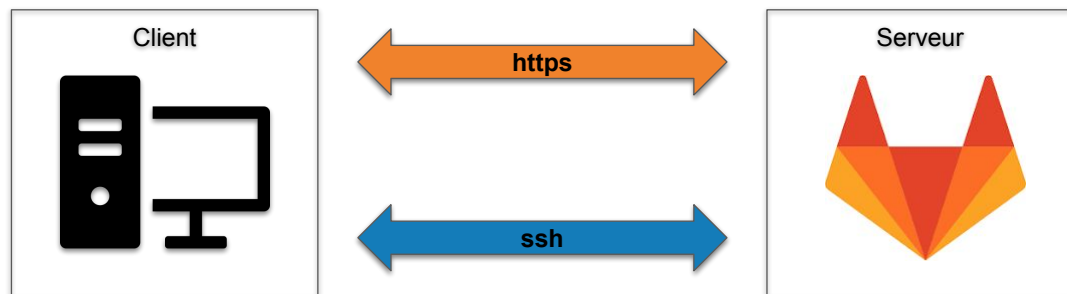
Visualiser le contenu d'un commit

Les dépôts: Historique



Visualiser l'historique

Communication avec Gitlab



Pour travailler avec les dépôts git, il existe deux modes de communication possible:

- https (login + token)
- ssh (clée publique et privée)

https et token

User Settings

- Profile
- Account
- Applications
- Chat
- Access Tokens**
- Emails
- Password
- Notifications
- SSH Keys
- GPG Keys
- Preferences
- Active Sessions
- Authentication log
- Usage Quotas

User Settings > Access Tokens

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Name

repo

Expires at

YYYY-MM-DD

Scopes

- ☐ **api**
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
- ☐ **read_user**
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
- ☐ **read_api**
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
- ☒ **read_repository**
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
- ☒ **write_repository**
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
- ☐ **sudo**
Grants permission to perform API actions as any user in the system, when authenticated as an admin user.

Create personal access token

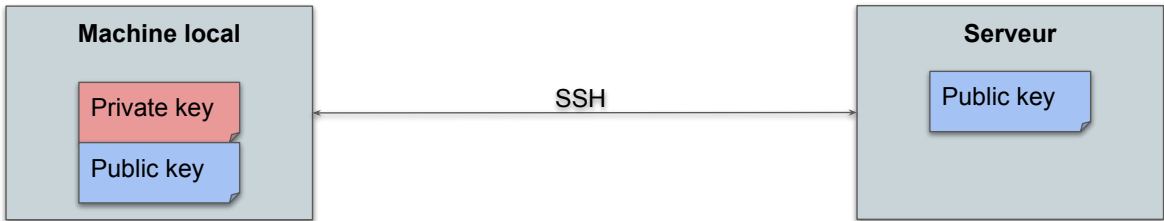
Création d'un token pour les accès au dépôt

Pour la connexion en https, il faut s'identifier via son login gitlab et via un token personnel en guise de mdp.

Pour créer un token, il faut se rendre dans les settings de votre compte->access tokens

Il faut alors donner un nom et sélectionner les 'scopes'. Pour travailler avec un dépôt git, cochez read/write repository

SSH



```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/schacon/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/schacon/.ssh/id_rsa.
Your public key has been saved in /Users/schacon/.ssh/id_rsa.pub.
```



les clés sont dans le dossier <home>/ssh

Le système ssh repose sur deux clés, une publique et une privée. La privée reste chez le générateur et la publique est envoyée aux autres systèmes souhaitant communiquer de manière sécurisée avec le générateur de la clé ssh.

Les messages sont alors cryptés avec une des deux parties de la clé (publique ou privée) et seule l'autre partie de la clé peut décoder le message. Cette méthode est très robuste et évite d'avoir un système de login/mot de passe.

Pour générer la clé sur son PC, il faut utiliser la commande "ssh-keygen". Une fois la génération terminée, copier la partie publique et la coller dans l'interface web de Gerrit dans l'onglet settings->ssh.

Conseil: si vous avez plusieurs PCs, inutile de générer plusieurs fois des clés ssh, utilisez la même clé privée sur vos différents PCs.

SSH

User Settings

- Profile
- Account
- Applications
- Chat
- Access Tokens
- Emails
- Password
- Notifications
- SSH Keys**
- GPG Keys
- Preferences
- Active Sessions
- Authentication log
- Usage Quotas

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to generate one or use an existing key.

Key

Paste your public SSH key, which is usually contained in the file `~/.ssh/id_ed25519.pub` or `~/.ssh/id_rsa.pub` and begins with `'ssh-ed25519'` or `'ssh-rsa'`. Do not paste your private SSH key, as that can compromise your identity.

Typically starts with `"ssh-ed25519 ..."` or `"ssh-rsa ..."`

Title **Expires at**

e.g. My MacBook key

Give your individual key a title.

Your SSH keys (2)

amercier@LAPTOP-5652PBFD
d7e713dd625303a556add4b386663fc9
Created 2 weeks ago

Ajout d'une clé ssh dans les paramètres utilisateur

Une fois la génération terminée, copier la partie publique et la coller dans l'interface web de gitlab dans l'onglet user settings -> ssh keys

Conseil: si vous avez plusieurs PCs, inutile de générer plusieurs fois des clés ssh, utilisez la même clé privée sur vos différents PCs.

Merge Request

New Merge Request

Source branch

devops/python_template develop

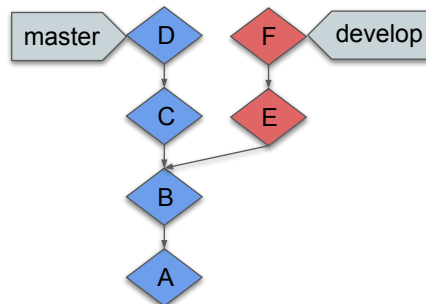
Update commit message rules
Arnaud MERCIER authored 2 weeks ago

Compare branches and continue

Target branch

devops/python_template master

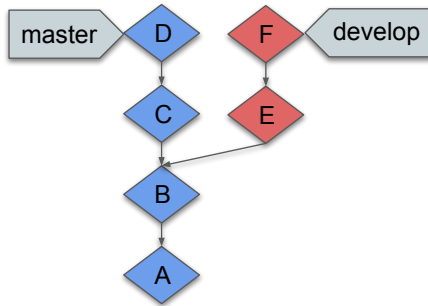
[RELEASE] V1.8.0
Arnaud MERCIER authored 2 months ago



L'objectif du merge request, est de faire une demande de merge d'une branche dans une autre. L'intérêt de passer par ce système, est qu'il permet de partager le merge avec d'autres personnes. Ils peuvent alors relire et valider le merge. Ce système de relecture augmente grandement la qualité du code.

De plus, cela permet de maîtriser le moment où sera merge cette branche dans la branche cible.

Merge Request: Création



New Merge Request

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

Title: Create the revision 2.0.0

Description: **Write** [Preview](#) **B I u `</code>`**

- Add `makefile`
- Update documentation
- Fixe `TestU`

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

Assignee: [Arnaud MERCIER](#)

Reviewer: [Administrator](#)

Milestone: [Milestone](#)

Labels: [Labels](#)

Merge options: ☒ Squash commits when merge request is accepted. [?](#)

Lors de la création du MR, il est possible de conner:

- Un titre: préfixé par WIP pour indiquer que le travail est toujours en cours
- Une description qui explique ce qu'apporte le MR au projet
- Assigné: Personne responsable du MR (qui validera)
- Reviewer: Personnes qui reliront le code
- Delete source branch: supprime automatiquement la branche de travail après le merge
- Squash commit: fusionne les commits de la branche de travail en un seul commit lors du merge

Merge Request: Revue

The screenshot displays a Merge Request (MR) review interface. On the left, a file explorer shows the project structure with files like `example.py`, `docs`, `conf.py`, `logs`, `config.yaml`, `src`, `sub_package`, and `_init_.py`. The main area shows a diff for `demos/example.py`. The diff highlights changes in the `import` statements. A comment by Arnaud MERCIER is visible, discussing the use of `src.hello` or `template.hello`. A callout box labeled '1' points to the code diff, indicating 'Relecture du code et commentaires'. Another callout box labeled '2' points to the comment, indicating 'Prise en compte de la relecture et échanges'.

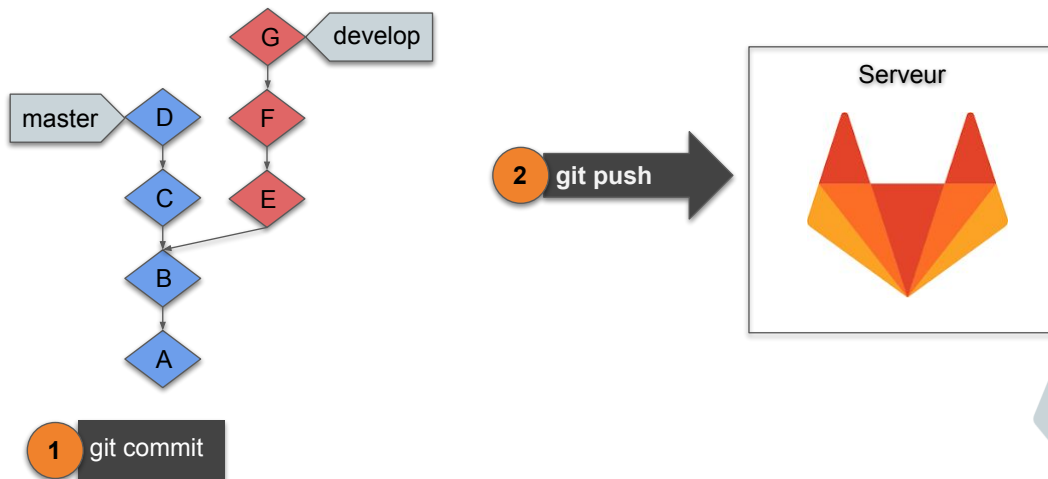
1 Relecture du code et commentaires

2 Prise en compte de la relecture et échanges

Lors de la relecture d'un MR, les relecteurs peuvent voir les modifications apportées par les commits de la branche de travail. Il peuvent également ajouter des commentaires sur le code et ainsi échanger avec le développeur.

Lorsque le développeur a pris en compte une remarque, il peut la clôturer via le bouton "resolved". La MR est intégrable lorsque l'ensemble des remarques sont clôturées.

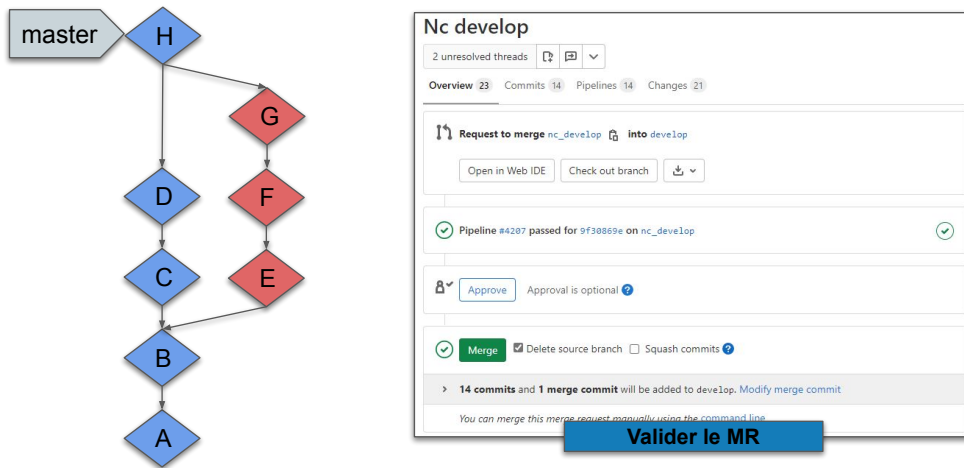
Merge Request: Correction



Si la remarque nécessite une modification de code, le développeur fait alors la modification sur son dépôt local dans la branche de travail. Il ajoute alors un nouveau commit pour prendre en compte les remarques. Enfin il push simplement la branche de travail pour ajouter automatiquement les nouveaux commits au MR

Les relecteurs sont alors notifiés de la maj du code et peuvent ainsi faire une nouvelle relecture

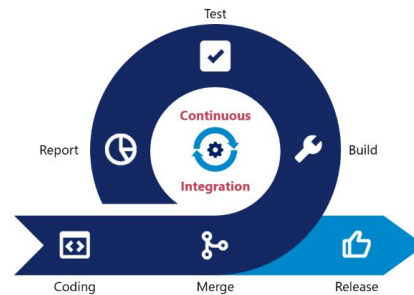
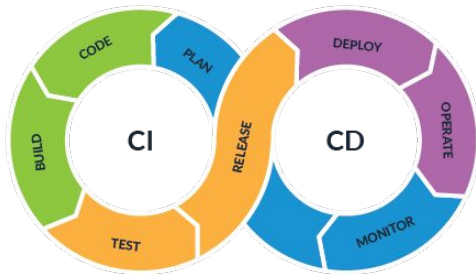
Merge Request: Validation



Quand l'ensemble des remarques sont clôturées et que le préfixe WIP est retiré du titre, alors le bouton "merge" est disponible:

- Delete source branch: supprime automatiquement la branche de travail après le merge
- Squash commits: fusionne les commits de la branche de travail en un seul commit lors du merge

CI/CD: Présentation



Gitlab propose des outils de CI/CD

En génie logiciel, CI/CD (parfois écrit CICD) est la combinaison des pratiques d'intégration continue et de livraison continue ou de déploiement continu¹.

Le CI/CD comble le fossé entre les activités et les équipes de développement et d'exploitation en imposant l'automatisation de la création, des tests et du déploiement des applications. Les pratiques DevOps modernes impliquent le développement continu, le test continu, l'intégration continue, le déploiement continu et la surveillance continue des applications logicielles tout au long de leur cycle de vie. La pratique CI/CD, ou pipeline CI/CD, constitue l'épine dorsale des opérations DevOps modernes.

CI/CD: Création du pipeline

```
15 before_script:
16   - python3 -V
17   - make clean
18   - make venv
19   - source __venv__bin/activate
20   - make init
21
22 lint:
23   script:
24     - make lint
25
26 type_check:
27   script:
28     - make typecheck
29   allow_failure: true
30
31 test:
32   script:
33     - make test
34
35 doc:
36   script:
37     - make doc
38     - rm -rf /home/gitlab-runner/delivery/$CI_PROJECT_NAME/$CI_COMMIT_BRANCH
39     - mkdir -p /home/gitlab-runner/de
```

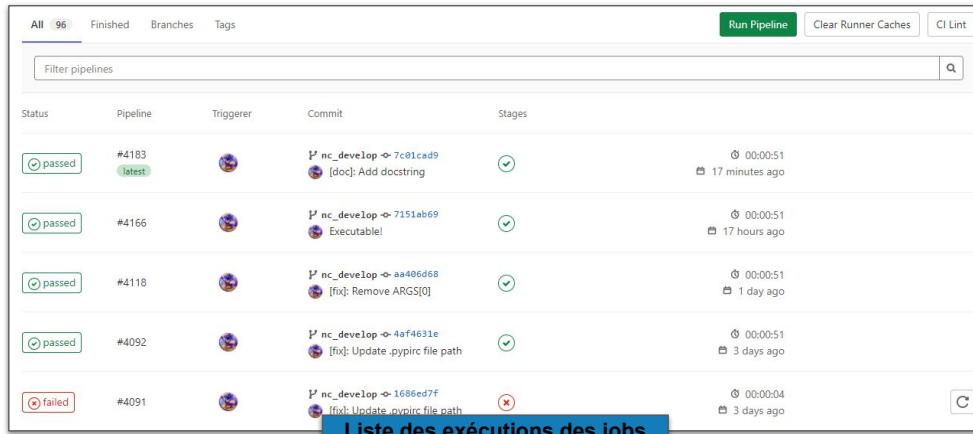
fichier .gitlab-ci.yml

<https://docs.gitlab.com/ee/ci/yaml/index.html>

Pour utiliser la CI/CD sur le dépôt, il faut dans un premier temps décrire les actions à effectuer pour la CI et pour la CD. Pour cela, il faut créer un script du nom de .gitlab-ci.yml à la racine du dépôt et l'ajouter à la gconf

Ce script est alors exécuté automatiquement par gitlab lors de certaines actions sur le dépôt, comme par exemple un commit

CI/CD: Liste des jobs

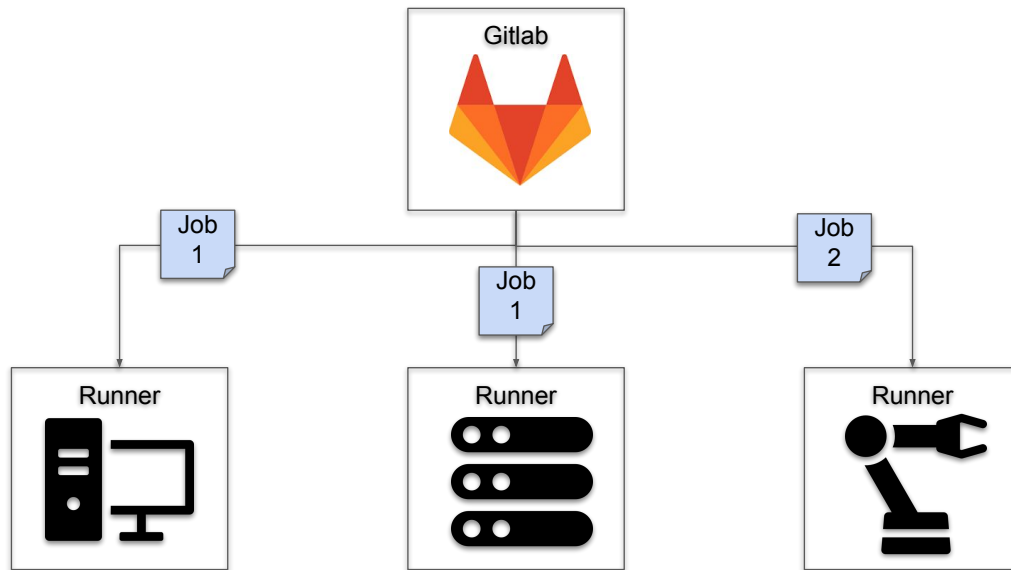


All	96	Finished	Branches	Tags	Run Pipeline	Clear Runner Caches	CI Lint
Filter pipelines							
Status	Pipeline	Triggerer	Commit	Stages			
passed	#4183 latest		nc_develop -> 7c91cad9 [doc]: Add docstring		00:00:51 17 minutes ago		
passed	#4166		nc_develop -> 7151ab69 Executable!		00:00:51 17 hours ago		
passed	#4118		nc_develop -> aa486d68 [fix]: Remove ARGV[0]		00:00:51 1 day ago		
passed	#4092		nc_develop -> 4af4631e [fix]: Update .pyprc file path		00:00:51 3 days ago		
failed	#4091		nc_develop -> 1686ed7f [fix]: Update .pyprc file path		00:00:04 3 days ago		

Liste des exécutions des jobs

Nous pouvons alors voir dans gitlab la liste des jobs (ci ou cd) ainsi que leur état

CI/CD: Runners

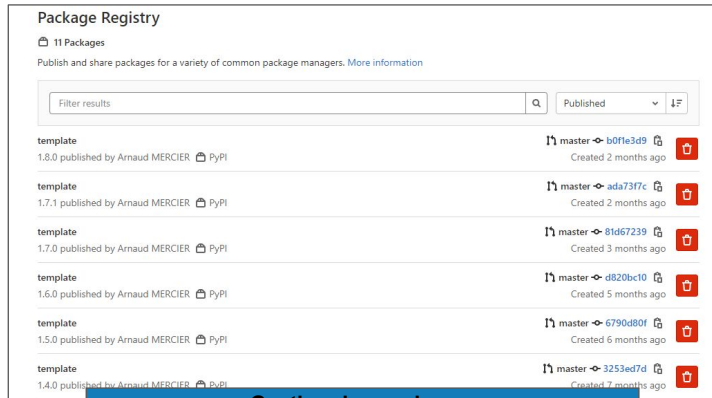


Gitlab n'exécute pas directement les scripts de CI/CD. Pour cela il faut lui attribuer des runners.

Le runner est un outil qui s'installe sur des cibles (banc, PC, ...) et qui communique avec gitlab.

Quand gitlab souhaite exécuter un job, il regarde la liste des runners disponibles et vérifié si il sont capable d'exécuter le script (language, ressources, ...). Une fois le runner identifié, il lui demande l'exécution du job et attend le résultat.

Package registry



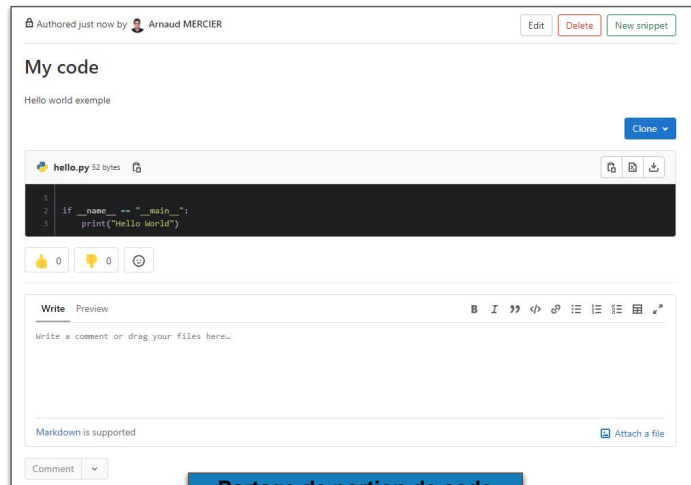
Gestion des packages

- Composer packages
- Conan packages
- Generic packages
- Maven packages
- npm packages
- NuGet packages
- MyPi packages
- ...

Gitlab fournit également un système de gestion de packages (binaires) dans plusieurs formats.

Il est possible de construire les packages via la CD et de les enregistrer directement dans la gestion de package.

Les snippets



Partage de portion de code

Les snippets permettent de partager et échanger sur une portion de code

Alternatives



<https://bitbucket.org/product>

Fonctions:

Dépôt Git
Gestion d'anomalie
Relecture
Intégration continue
...



<https://github.com>

Fonctions:

Dépôt Git
Gestion d'anomalie
Relecture
Intégration continue
...

Plusieurs alternatives existent pour GitLab comme par exemple Bitbucket et GitHub

Bilan



<https://gitlab.com>

Version hébergée

Version installée

Nous avons vu dans ce chapitre les fonctions de base de gitlab