Chapitre 8

Introduction
À
GitLab CI/CD

Définitions

CI vs CD vs CD

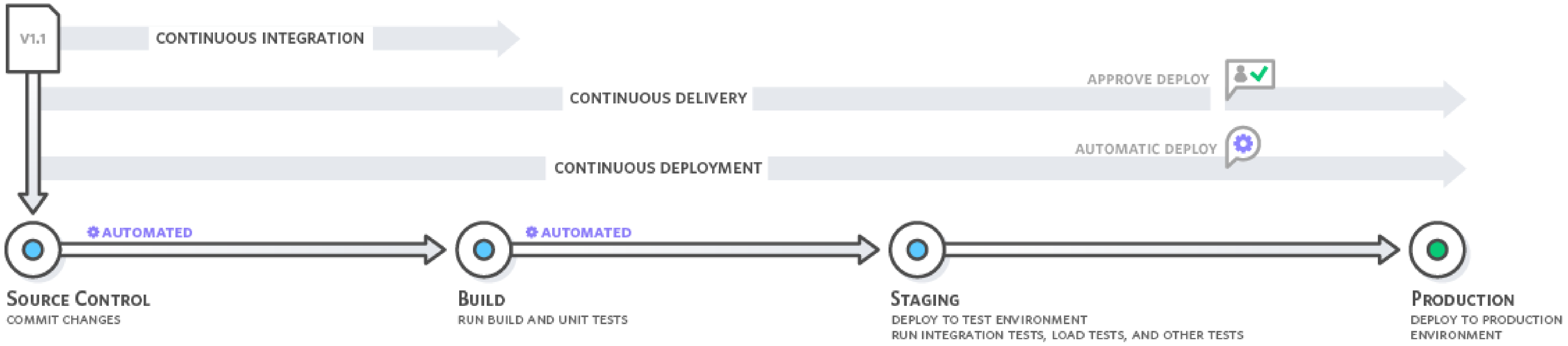


Image : aws.amazon.com

Plan

1. Qu'est-ce que l'intégration ?
2. Quels sont les risques ?
3. La solution : intégration continue
4. Utilisation de Gitlab CI/CD

Qu'est-ce que l'intégration ?

Fusionner **plusieurs sous-systèmes** en un seul.

Chaque sous-parties peut :

- être développée séparément
- utiliser des langages de programmation différents
- utiliser des architectures différentes

Projet en C++ : quelles étapes ?

Lister les étapes depuis le développement jusqu'au déploiement.

Projet en C++ : quelles étapes ?

LOCAL

Code, compile, link, manual tests, unit tests, integration tests, functional tests, commit, rebase-merge, push

REMOTE

Pull, recompile, all tests, installers, sign binaries, install on VMs, all tests, upload installers.

Quels sont les risques ?

En fonction des technologies utilisées ?

En fonction du temps entre chaque intégration ?

➔ **Integration Hell**

La solution : intégration continue

Intégrer "en continue" ➔ plusieurs fois par jour

CI/CD Permet de :

trouver les bugs le plus tôt possible

s'assurer que le produit reste livrable

vérifier la portabilité (Windows / Mac / Linux)

...

La solution : intégration continue

WORKFLOW

1. Faire des modifications **en local**
2. Faire passer **TOUS** les tests **en local**
3. Fusionner les modifications sur le serveur (git push)
4. Exécuter **TOUS** les tests sur le serveur (CI/CD)
5. Déployer le produit final et retester (CI/CD)

CI/CD est automatique (à chaque push)

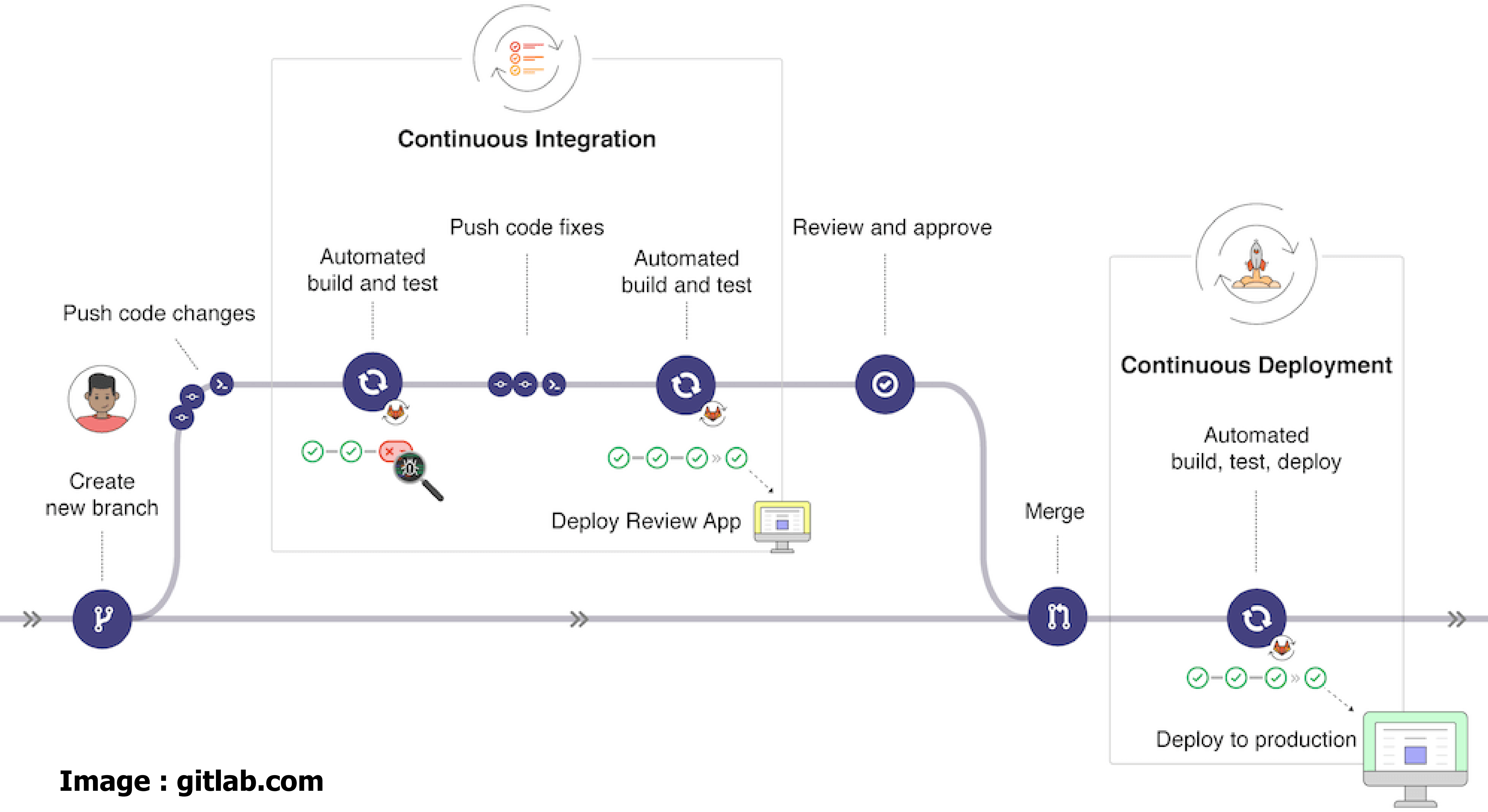
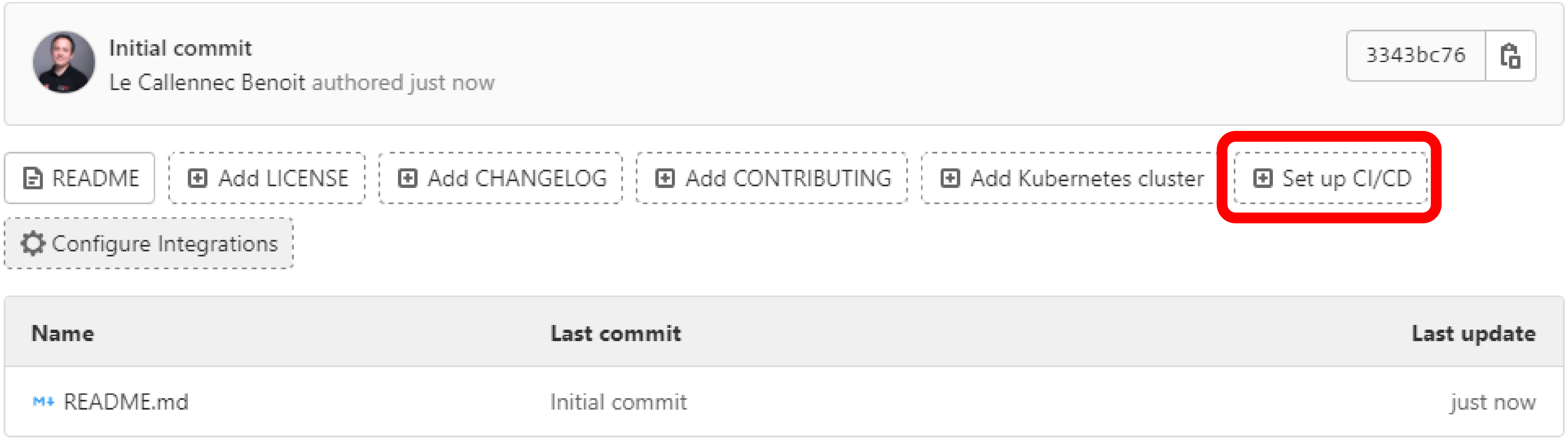


Image : gitlab.com

CI/CD et GitLab

Set up CI/CD

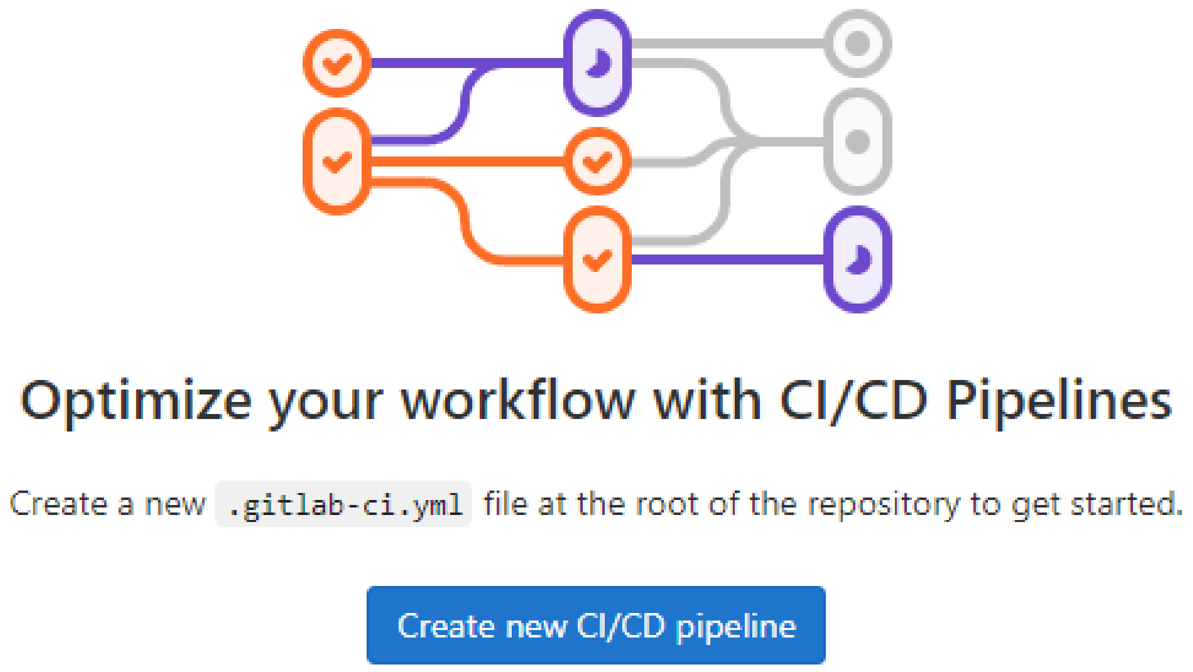

Initial commit
Le Callennec Benoit authored just now

3343bc76

Name	Last commit	Last update
README.md	Initial commit	just now

CI/CD et GitLab

Set up CI/CD

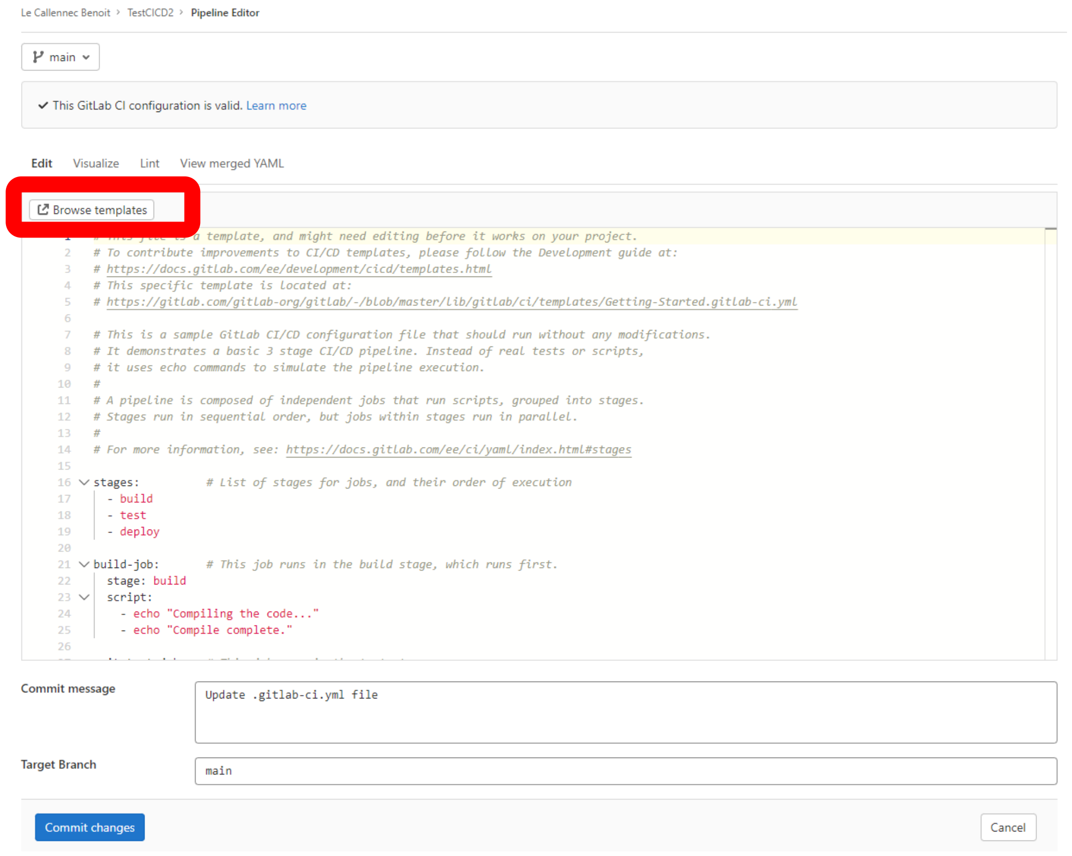


Optimize your workflow with CI/CD Pipelines

Create a new `.gitlab-ci.yml` file at the root of the repository to get started.

CI/CD et GitLab

Set up CI/CD


Le Callennec Benoit › TestCICD2 › Pipeline Editor

main

✓ This GitLab CI configuration is valid. Learn more

Edit Visualize Lint View merged YAML

Browse templates

```
# This file is a template, and might need editing before it works on your project.
# To contribute improvements to CI/CD templates, please follow the Development guide at:
# https://docs.gitlab.com/ee/development/cicd/templates.html
# This specific template is located at:
# https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Getting-Started.gitlab-ci.yml

# This is a sample GitLab CI/CD configuration file that should run without any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or scripts,
# it uses echo commands to simulate the pipeline execution.
#
# A pipeline is composed of independent jobs that run scripts, grouped into stages.
# Stages run in sequential order, but jobs within stages run in parallel.
#
# For more information, see: https://docs.gitlab.com/ee/ci/yaml/index.html#stages

stages:          # List of stages for jobs, and their order of execution
  - build
  - test
  - deploy

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Compiling the code..."
    - echo "Compile complete."

```

Commit message: Update .gitlab-ci.yml file

Target Branch: main

Commit changes Cancel

CI/CD et GitLab

Choisir l'image de base : `image: gcc`

Run first pipeline

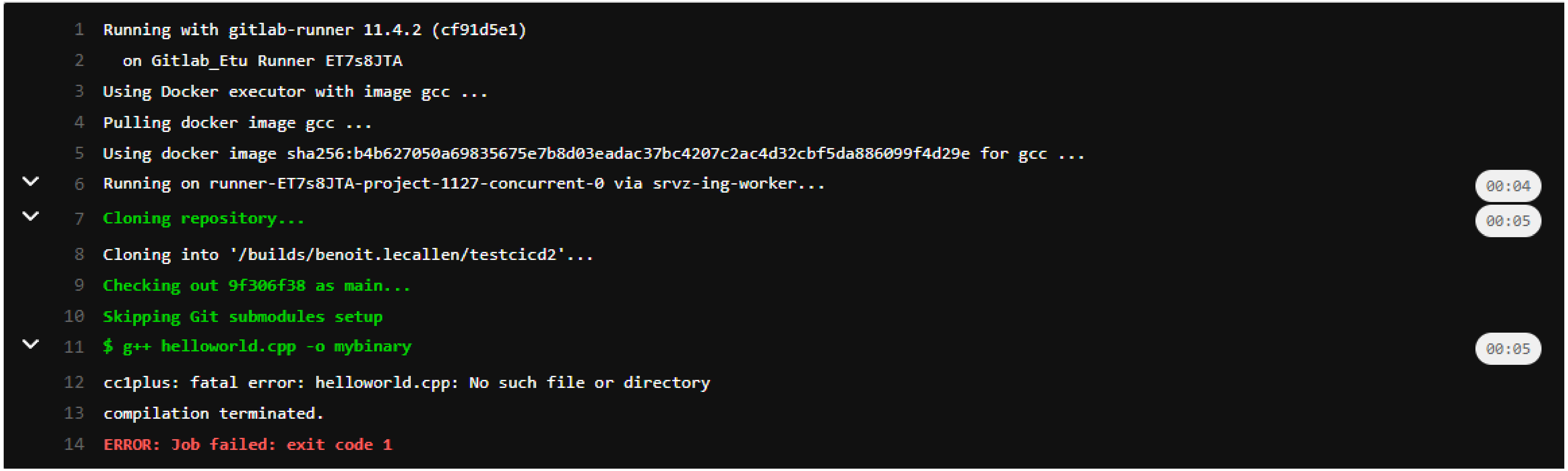
```
Error : missing runner
```

Rajouter un runner

```
tags:
  - alpine-docker
```

CI/CD et GitLab

First pipeline executed



```
Running with gitlab-runner 11.4.2 (cf91d5e1)
  on Gitlab_Etu Runner ET7s8JTA
Using Docker executor with image gcc ...
Pulling docker image gcc ...
Using docker image sha256:b4b627050a69835675e7b8d03eadac37bc4207c2ac4d32cbf5da886099f4d29e for gcc ...
Running on runner-ET7s8JTA-project-1127-concurrent-0 via srvz-ing-worker...                 00:04
Cloning repository...                                                                       00:05
Cloning into '/builds/benoit.lecallen/testcicd2'...
Checking out 9f306f38 as main...
Skipping Git submodules setup
$ g++ helloworld.cpp -o mybinary                                                            00:05
cc1plus: fatal error: helloworld.cpp: No such file or directory
compilation terminated.
ERROR: Job failed: exit code 1
```

CI/CD et GitLab

Définir les `stages` (défault : `.pre`, `build` , `test` , `deploy` ,`.post`)

Les `stages` contrôlent l'ordre d'exécution des `jobs` :

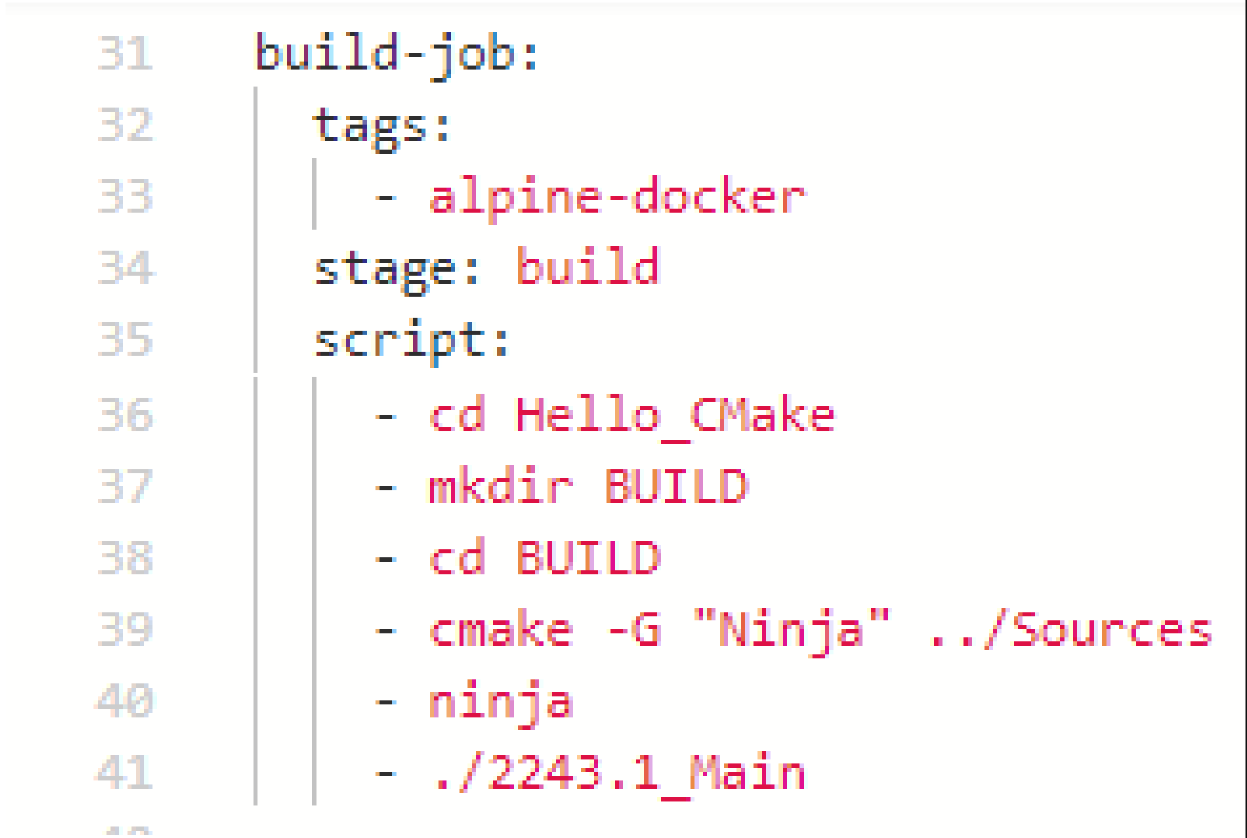
- **Même `stage` → les `jobs` tournent en parallèle**
- **Sinon → les `jobs` attendent les `jobs` du `stage` précédent**

CI/CD et GitLab

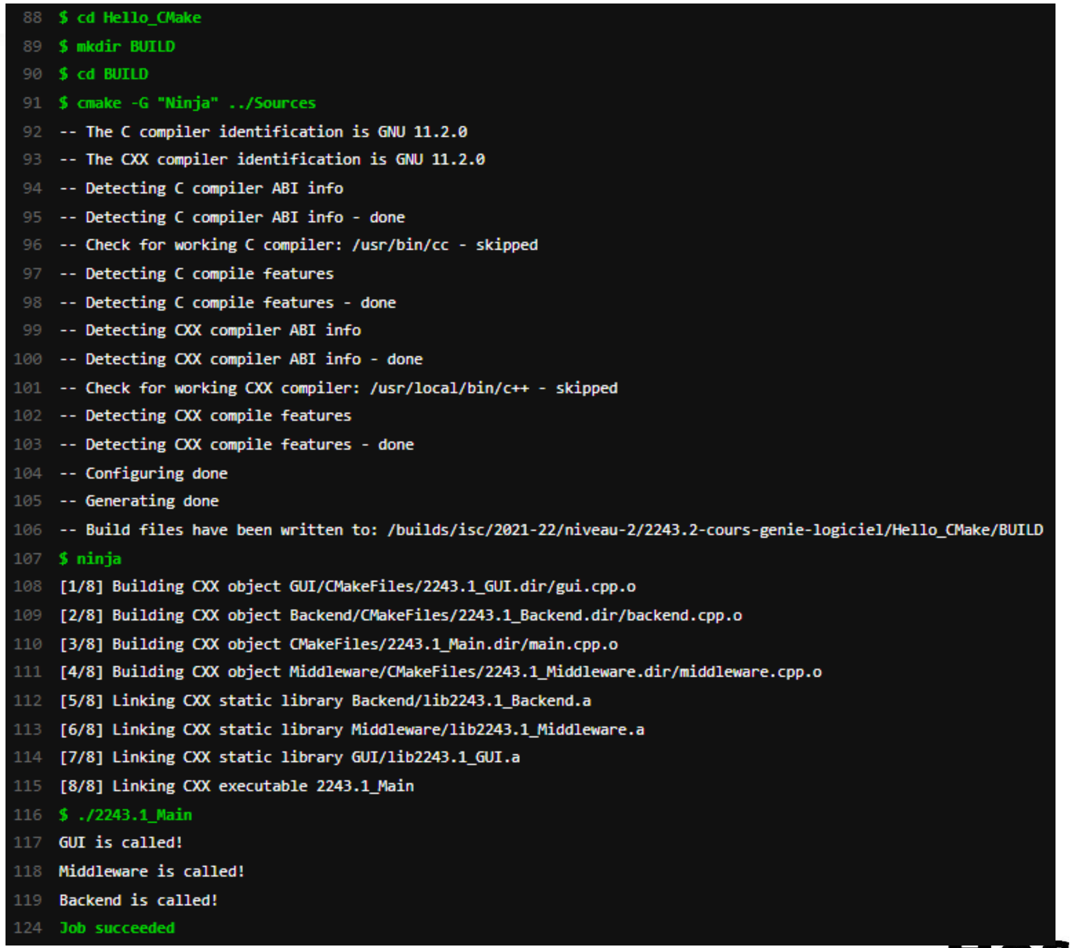
Installer des outils particuliers : `cmake`

```
before_script:
  - apt-get update --yes
  - apt-get install --yes cmake
  - apt-get install --yes ninja-build
```

Compiler le tout



```
build-job:
  tags:
    - alpine-docker
  stage: build
  script:
    - cd Hello_CMake
    - mkdir BUILD
    - cd BUILD
    - cmake -G "Ninja" ../Sources
    - ninja
    - ./2243.1_Main
```



```
$ cd Hello_CMake
$ mkdir BUILD
$ cd BUILD
$ cmake -G "Ninja" ../Sources
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/local/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /builds/isc/2021-22/niveau-2/2243.2-cours-genie-logiciel/Hello_CMake/BUILD
$ ninja
[1/8] Building CXX object GUI/CMakeFiles/2243.1_GUI.dir/gui.cpp.o
[2/8] Building CXX object Backend/CMakeFiles/2243.1_Backend.dir/backend.cpp.o
[3/8] Building CXX object CMakeFiles/2243.1_Main.dir/main.cpp.o
[4/8] Building CXX object Middleware/CMakeFiles/2243.1_Middleware.dir/middleware.cpp.o
[5/8] Linking CXX static library Backend/lib2243.1_Backend.a
[6/8] Linking CXX static library Middleware/lib2243.1_Middleware.a
[7/8] Linking CXX static library GUI/lib2243.1_GUI.a
[8/8] Linking CXX executable 2243.1_Main
$ ./2243.1_Main
GUI is called!
Middleware is called!
Backend is called!
Job succeeded
```

EXERCICE

OBJECTIF

Pouvoir exécuter CMake et Ninja sur le serveur CI/CD, automatiquement à chaque push