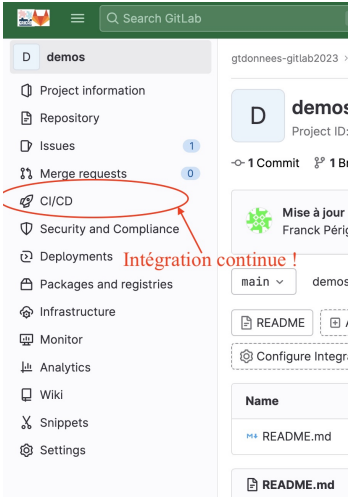
Principes et exemples de mise en place de l'intégration continue sur Gitlab

Franck Pérignon

Journée Gitlab, 29 juin 2023

Intégration continue ?

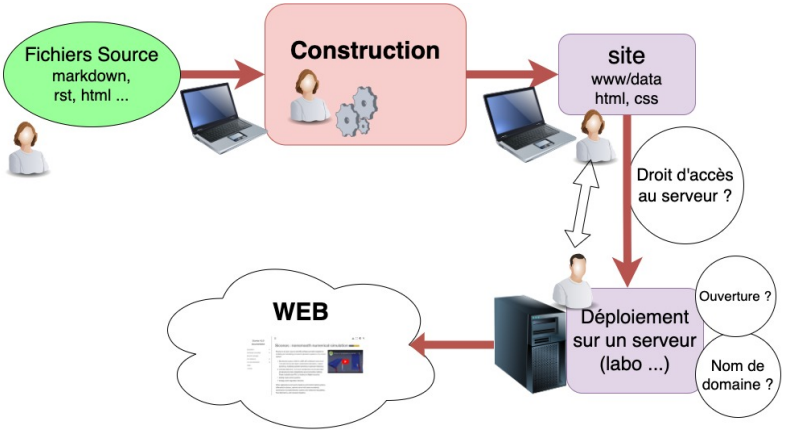


Une fonctionnalité essentielle de Gitlab ...

... évoquée et entraperçue ce matin.

Un exemple pour commencer

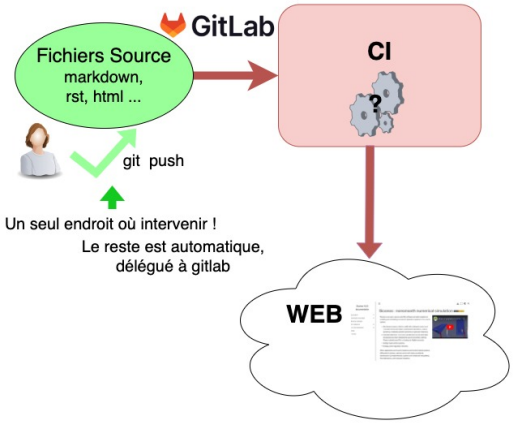
Objectif : créer un site web et le publier.



Sans gitlab et l'intégration continu 😦

Un exemple pour commencer

Objectif : créer un site web et le publier.



Avec gitlab et l'intégration continue 😀

Intégration continue (CI)

👉 À l'origine, un outil pour les développeurs et la production de logiciels

Concept : *une pratique consistant à vérifier systématiquement et automatiquement l'impact de toute modification des sources sur le fonctionnement, les performances ...*

- Un projet Gitlab
- Une série de scripts associés au projet
- À chaque "push" (modification des sources), le code est construit, installé, testé ...
- Un rapport est publié sur la page du projet

CI - De nombreux avantages

- Facilite les développements au quotidien.
- Permet de produire un code stable, robuste et portable.
- Permet de s'assurer que le résultat de nouvelles modifications n'introduit pas de régression du code
- Permet d'anticiper différents types d'utilisation du code.
- Déployer le code, fournir des "images" prêtes à l'emploi (Docker, Singularity ...)
- ...

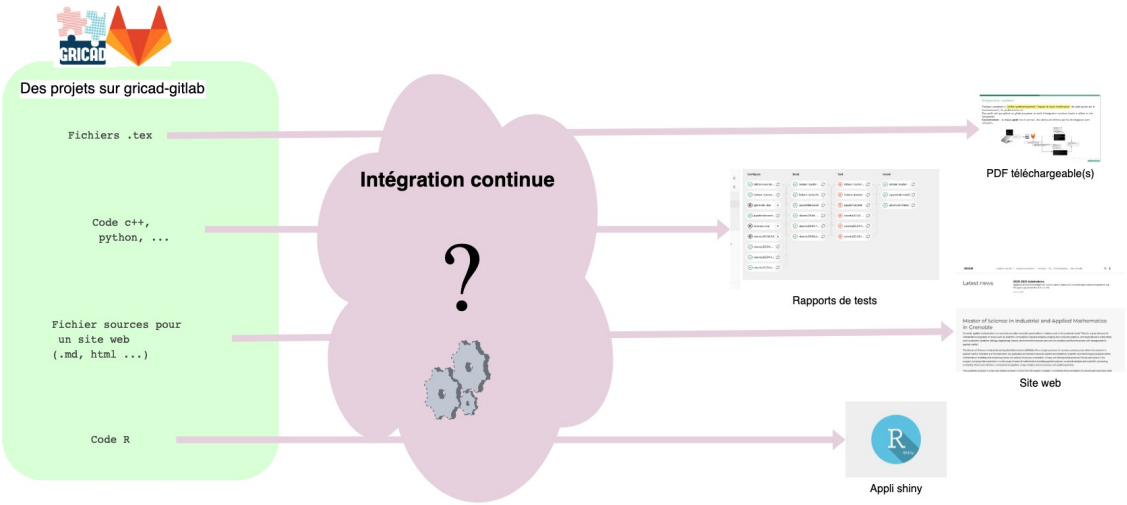
👉 un outil fondamental pour la qualité des logiciels, la reproductibilité

Intégration continue (CI)

👉 À l'origine, un outil pour les développeurs et la production de logiciels
mais exploitable SIMPLEMENT pour d'autres usages

Concept : *une pratique consistant à associer à chaque modification des sources une série d'opérations qui seront exécutées automatiquement*

- Génération et publication de site web
- Création de documents (markdown, latex ...)
- Mise en ligne Shiny (R), Voila (Python)
- ...


Des projets sur gricad-gitlab
Fichiers .tex
Code c++,
python, ...
Fichier sources pour
un site web
(.md, html ...)
Code R
Intégration continue
?
PDF téléchargeable(s)
Rapports de tests
Latest news
Site web
Appli shiny

CI - Principe et fonctionnement

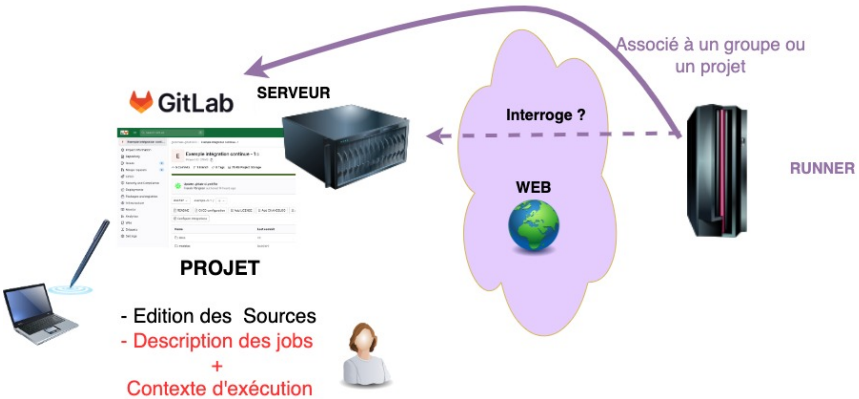
Job : une suite d'actions à exécuter

- Chaque job est indépendant
- Un contexte d'exécution est associé à chaque job

CI - Principe et fonctionnement

Runner : une machine hôte qui va exécuter votre job (possiblement via Docker)

CI - Principe et fonctionnement



Le runner interroge Gitlab pour détecter toute activité dans le projet

Les runners

Runner : une machine, un hôte sur lequel vont être exécutées les tâches d'intégration continue.

- Des shared runners disponibles (⚠️ variable selon la plateforme) pour tous les projets
- Des runners privés, associés à un groupe ou un projet

  👉 n'importe quelle machine à votre disposition : votre poste, un serveur, une machine virtuelle ...

  👉 pas d'ouverture vers l'extérieur nécessaire.

Comment installer un runner ?

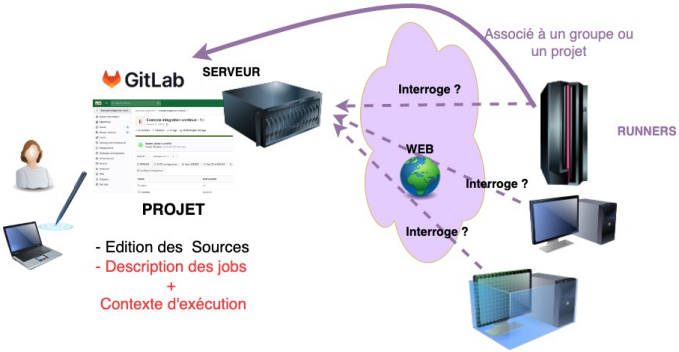
Deux étapes :

- Disposer d'une machine avec Docker et gitlab-runner
  https://docs.gitlab.com/runner/install/index.html
  Installation standard et sans difficulté.
  Exemple, gitlab runner pour ubuntu/debian :
  https://docs.gitlab.com/runner/install/linux-repository.html
- Enregistrer le runner auprès du projet/groupe

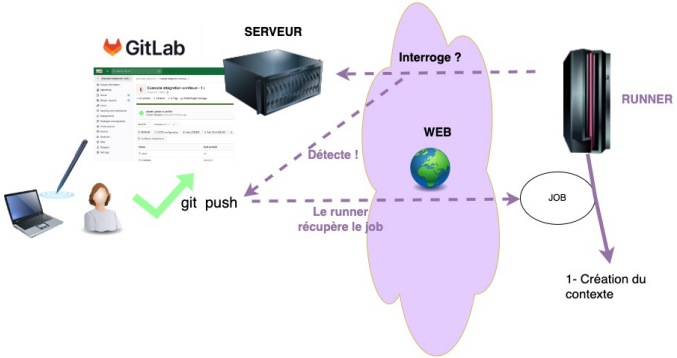
```
gitlab-runner register
```

Informations à fournir : voir Settings du projet → CI/CD → Runner

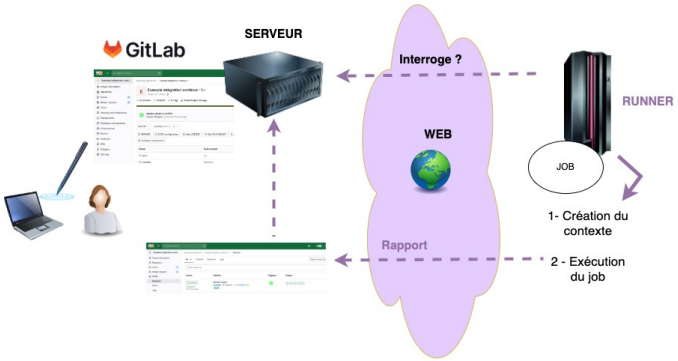
Hors contexte pour aujourd'hui ...



Possibilité d'associer plusieurs runners à un groupe/projet

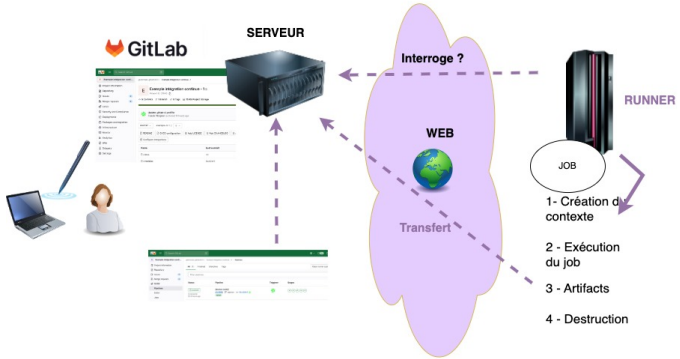


Activité dans le projet ⇒ un runner "prend" un job



Sur le runner

- Création d'un contexte
- Exécution du job dans ce contexte
- Rapport de fin de job envoyé vers le projet Gitlab



artifacts : des répertoires ou des fichiers produits par le job, à conserver

- sauvegardés sur le serveur Gitlab
  ⚠ attention à l'espace occupé
  possibilité de fixer une "date de péremption"
- peuvent être transmis entre les jobs

Comment mettre en place la CI ?

Sur 

, un outil : gitlab-ci

https://docs.gitlab.com/ee/ci/

Mise en place ?

1. Créer un fichier nommé .gitlab-ci.yml à la racine du dépôt
   - Activation de la CI
   - Contient la liste des tâches à effectuer : quelles actions, dans quel contexte etc.

Comment mettre en place la CI ?

Sur 

, un outil : gitlab-ci

https://docs.gitlab.com/ee/ci/

Mise en place ?

1. Créer un fichier nommé .gitlab-ci.yml à la racine du dépôt
   - Activation de la CI
   - Contient la liste des tâches à effectuer : quelles actions, dans quel contexte etc.
2. Définir des runners : où vont tourner les jobs ?

Comment mettre en place la CI ?

Sur GitLab, un outil : gitlab-ci
https://docs.gitlab.com/ee/ci/

Mise en place ?

1. Créer un fichier nommé .gitlab-ci.yml à la racine du dépôt
   - Activation de la CI
   - Contient la liste des tâches à effectuer : quelles actions, dans quel contexte etc.
2. Définir des runners : où vont tourner les jobs ?
3. 😦 Décrire (en yaml) les jobs dans le fichier .gitlab-ci.yml
   Un petit conseil : beaucoup de modèles, inspirez vous, copiez-collez !

Premier exemple de job

```yaml
make_pdf:
  stage: build
  image: aergus/latex:latest
  script:
    - ls
    - pdflatex article.tex
  artifacts:
    paths:
      - article.pdf
    expire_in: 1 week
```

- un nom : make_pdf
- des instructions : le mot-clé **script**
- un contexte d'exécution : le mot-clé **image**
- ce qu'il faut conserver : le mot-clé **articfact**

👉 **Démos** dans le projet exemple-ci-1 du groupe gtdonnees-gitlab2023/

Vocabulaire

- job : une suite d'actions exécutées sur un runner.
  - Chaque job est indépendant.
  - Rien n'est conservé à la fin (sauf demande explicite via les artifacts).
- artifacts : des répertoires ou des fichiers à conserver et à transmettre d'un job à l'autre.
- runner : une machine hôte qui va exécuter votre job (possiblement via Docker)
- pipeline : une série de jobs exécutés sur un ou plusieurs runners. Chaque pipeline correspond à un commit
- stage : un "étage" du pipeline, peut contenir plusieurs jobs exécutés en parallèle

Génération de pages web - Gitlab-pages

Un outil pour publier un site web statique associé à votre projet

- Pages privées ou publiques
- Publication et hébergement délégués à Gitlab
- Doc : *https://about.gitlab.com/features/pages/*

Principe

1. Choisir un outil capable de générer un site web
   Exemples : Pelican, Sphinx, Mkdocs ...
   Sources en markdown -> html
2. Sauvegarder les sources dans un projet Gitlab
3. Créer un job CI nommé "pages"

Génération de pages web - Gitlab-pages - Exemple

```
pages:
  stage: test
  image: python:3.8.5-slim-buster
  script:
  - python -m pip install --upgrade pip mkdocs-bootswatch
  - mkdocs build
  - mv site public
  artifacts:
    paths:
    - public
```

Résultat : vos pages sur un serveur nommé
https:// nom_du_groupe .gricad-pages.univ-grenoble-alpes.fr/ nom_du_projet .

👉 **Démos** dans le projet exemple-ci-1 du groupe gtdonnees-gitlab2023/

Règles et CI sous conditions

La CI peut rapidement devenir consommatrice de ressoures 😦
👉 Bonnes pratiques :

- ne pas lancer la CI systématiquement
  [skip CI] dans le message de commit
- ajouter des filtres/règles dans le fichier yml

Règles et CI sous conditions

Exemple :

```yaml
workflow:
  rules:
    - if: $CI_COMMIT_MESSAGE =~ /\[publish\].*/i
      when: always
    - when: never
```

👉 **Démos** dans le projet exemple-ci-1 du groupe gtdonnees-gitlab2023/

https://www.docker.com/

Une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation.

https://www.docker.com/

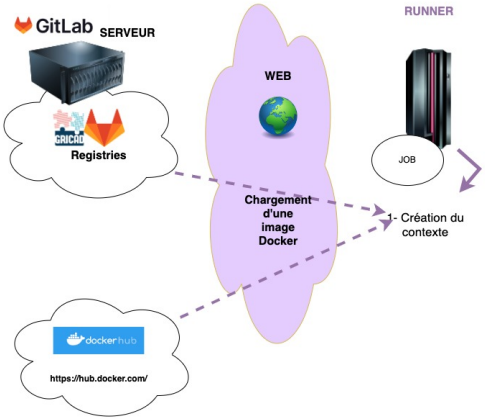
Une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation.

image : un "pack" qui contient tout ce qui est nécessaire à l'exécution de notre application

container : environnement d'exécution léger, alternative aux machines virtuelles

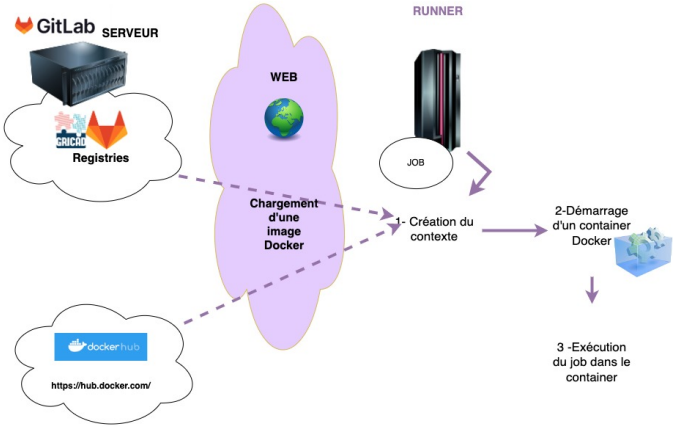
Le container est construit/démarré à partir de l'image

Docker : ce qu'il faut retenir pour la CI Gitlab



Registries : une banque d'images disponibles téléchargeables

Docker : ce qu'il faut retenir pour la CI Gitlab



Container = un contexte d'exécution isolé pour chaque job

Pourquoi Docker ?

- Large choix de systèmes
- Facilite la reproduction des environnements utilisateurs
- Possibilité d'utiliser la CI pour créer ses propres images et les sauvegarder dans un projet gitlab (gitlab registries).

👉 En bonus, **démo** dans le projet exemple-ci-2 du groupe gtdonnees-gitlab2023/

Démo : CI/CD pour un code de calcul

À chaque **push** vers le serveur, des tâches pré-définies par les développeurs sont exécutées.