How to setup continuous integration (CI) in your Gitlab projects, another step towards software reproducibility

Franck Pérignon

Workshop on Reproducible Software Environments for Research and High-Performance Computing - November 8-10 2023

LABORATOIRE
JEAN KUNTZMANN
MATHÉMATIQUES APPLIQUÉES - INFORMATIQUE

What are we going to talk about today?

- Focus on GitLab (but things work for other similar tools)
  👉 I assume that everybody in this room is quite familiar with Gitlab (basics) and that Git is your everydays'friend.

  Right 🤨 ?
- Continuous Integration (CI) and Continuous Delivery (CD) 🤔

What are we going to talk about today?

- Focus on GitLab (but things work for other similar tools)
  👉 I assume that everybody in this room is quite familiar with Gitlab (basics) and that Git is your everydays'friend.

  Right 🤨 ?
- Continuous Integration (CI) and Continuous Delivery (CD) 🤔

What are we going to talk about today?

- Focus on GitLab (but things work for other similar tools)
  👉 I assume that everybody in this room is quite familiar with Gitlab (basics) and that Git is your everydays'friend.

  Right 🤨 ?
- Continuous Integration (CI) and Continuous Delivery (CD) 🤔
- Take the opportunity to speak about some other "advanced" features of Gitlab (merge-requests, releases, API, ...) 👉 ask if you want some details!

Specifically in the context of reproducibility, software development and guix obviously

How ?

Round trips between ...

- This presentation
  - An introduction about CI/CD concepts and vocabulary
  - Focuses on some specific topics (e.g. Registries, workflow ...)

How ?

Round trips between ...

- This presentation
  - An introduction about CI/CD concepts and vocabulary
  - Focuses on some specific topics (e.g. Registries, workflow ...)
- Some real demos inside a Gitlab project

Materials

Your entry point, a Gitlab group:
https://gricad-gitlab.univ-grenoble-alpes.fr/repro4research
or this page:
https://repro4research.gricad-pages.univ-grenoble-alpes.fr/demos/start

Materials

Your entry point, a Gitlab group:
https://gricad-gitlab.univ-grenoble-alpes.fr/repro4research
or this page:
https://repro4research.gricad-pages.univ-grenoble-alpes.fr/demos/start

- These slides
- A demo project: repro4research/demos/ci-montpellier (empty for now) ...
- A place to experiment: 👩‍🔬 repro4research/sandbox group
  👉 Feel free to do anything in this group, except removing someone else's work !

Materials

Your entry point, a Gitlab group:
https://gricad-gitlab.univ-grenoble-alpes.fr/repro4research
or this page:
https://repro4research.gricad-pages.univ-grenoble-alpes.fr/demos/start

- These slides
- A demo project: repro4research/demos/ci-montpellier (empty for now) ...
- A place to experiment: 👩‍🔬 repro4research/sandbox group
  👉 Feel free to do anything in this group, except removing someone else's work !

And if you're lost, too tired 🥱 or run out of time ...
🧳 Homework: all the projects in the group repro4research/Materials and Demos, tutorials including everything we're going to work on today, detailed and explained

Let's get started ...

Once again, I assume that

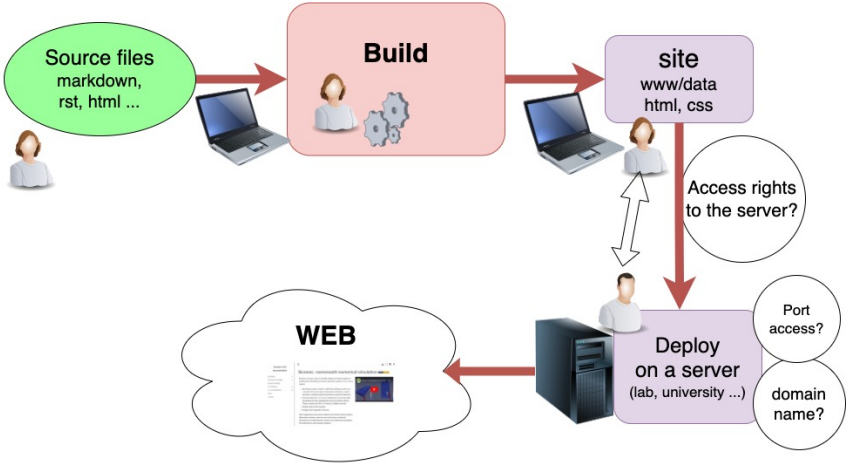
- everybody in this room is quite familiar with Gitlab (basics)
- Git is your everydays'friend
- you are comfortable with CLI in a Terminal
- Docker 😧 ?
- You are registered on gricad-gitlab and have access to the forementioned group repro4research

What will you take away today?

- How to setup CI in a gitlab project.
- Some tips and advices (hopefully good?) to write your CI scripts.
- Some examples, ready to use and easy to reproduce when back at home (simple and more advanced).
- An overview and some examples of interesting and useful Gitlab features.
- A short introduction on containers, Docker and friends. Not a full understanding but the keys to understand and to use it.
- Headache?

Let's start with some use cases

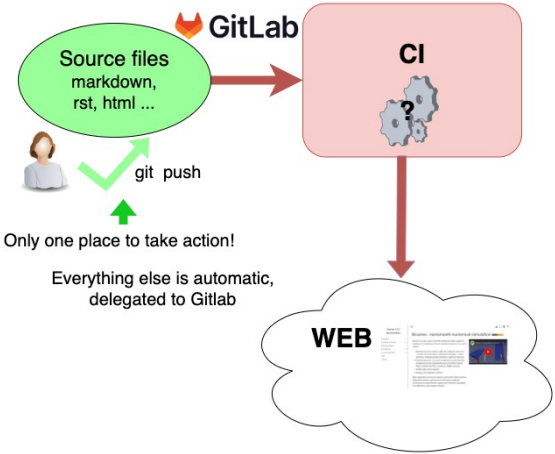
Objective: build and publish a website



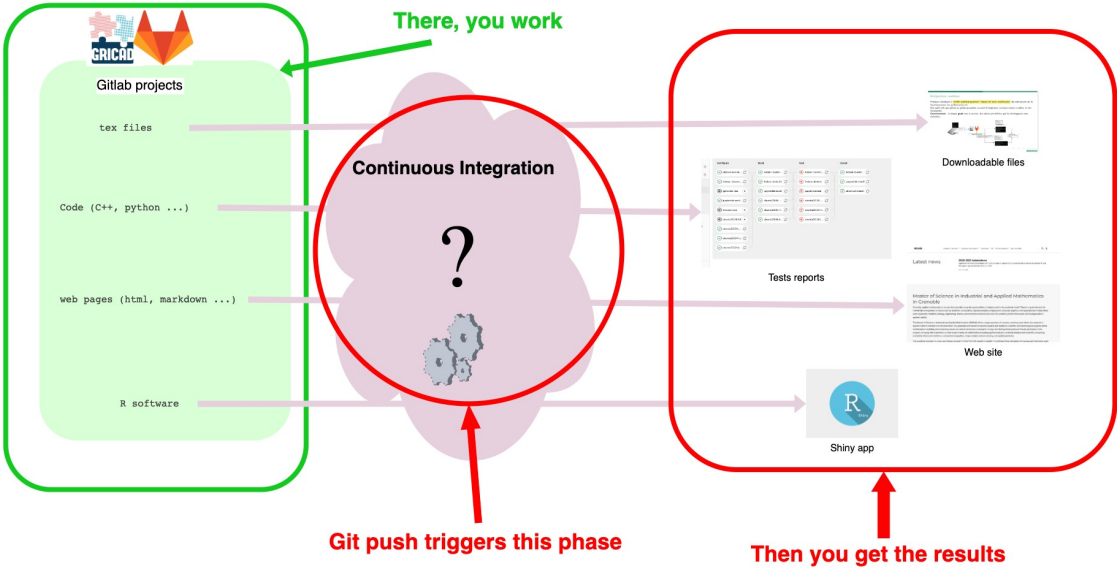
No CI, no Gitlab 😦

Let's start with some use cases

Objective: build and publish a website



Gitlab and CI 😀


There, you work
Gitlab projects
tex files
Code (C++, python ...)
web pages (html, markdown ...)
R software
Continuous Integration
?
Downloadable files
Tests reports
Web site
Shiny app
Git push triggers this phase
Then you get the results

Continuous Integration (CI)

👉 A devops tool

👉 Mostly dedicated to people working on softwares

Concept : *practice of systematically and automatically checking the impact of any modification to the sources on operation, performance and so on*

- A Gitlab project
- A bunch of scripts
- Any push leads to the build, install, test ... of the project
- A report is generated and published

Continuous Integration (CI)

👉 A devops tool

👉 Mostly dedicated to people working on softwares but (quite) easy to use, fits with many usages

Concept : *practice of associating with each modification to the sources a series of operations that will be carried out automatically*

- Build and publish website
- Build (pdf) documents (markdown, latex ...)
- Deploy: Shiny (R) or Voila (Python)
- ...

Continuous Delivery (CD)

Concept : *practice to automate the infrastructure provisioning and application release process.*

Next step after CI

- Make the software ready for production
- Deploy the code to production environment

Why should you use CI/CD?

- Makes collaboration between developers easier
- Identify and fix errors and issues more easily and more rapidly
- Ensures that changes in the code or new features do not lead to regression
- Cleaner, more stable, more portable code
- Anticipate, plan and test different environments (debug/release, different OS, differents parameters ...)
- Delivery and deployment: ready-to-use Docker-like images (Docker, Singularity ...)
- Frees up time for developers and reduce time-to-release or time-to-new-feature

Why should you use CI/CD?

- Makes collaboration between developers easier
- Identify and fix errors and issues more easily and more rapidly
- Ensures that changes in the code or new features do not lead to regression
- Cleaner, more stable, more portable code
- Anticipate, plan and test different environments (debug/release, different OS, differents parameters ...)
- Delivery and deployment: ready-to-use Docker-like images (Docker, Singularity ...)
- Frees up time for developers and reduce time-to-release or time-to-new-feature

👍 Happier developers and users !

👍 A fundamental tool for software quality and reproducibility

CI - What and How?

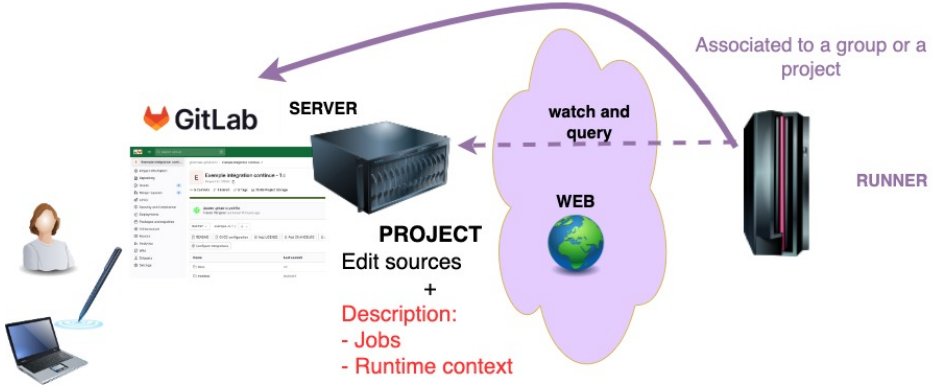
Job : a sequence of operations to be executed (configure, build ...)

- Each job is run independently of the others
- Each job has its own 'context' of execution

CI - What and How?

The Runner : a host in charge of the execution of the jobs (through an executor , e.g. Docker or shell)

CI - What and How?



The runner keeps contact with the Gitlab server and detect every action in the project (git push)

Step by step CI setup

On GitLab, a powerful tool: gitlab-ci
https://docs.gitlab.com/ee/ci/

Setup?

1. Create a .gitlab-ci.yml at the root of the Gitlab project repository
   - CI is on!
   - This file lists all tasks that must be executed (the jobs!): what, where, in which context ...

Step by step CI setup

On GitLab, a powerful tool: gitlab-ci
https://docs.gitlab.com/ee/ci/

Setup?

1. Create a .gitlab-ci.yml at the root of the Gitlab project repository
   - CI is on!
   - This file lists all tasks that must be executed (the jobs!): what, where, in which context ...
2. Define and register runners: hosts for the jobs

Step by step CI setup

On GitLab, a powerful tool: gitlab-ci
https://docs.gitlab.com/ee/ci/

Setup?

1. Create a .gitlab-ci.yml at the root of the Gitlab project repository
   - CI is on!
   - This file lists all tasks that must be executed (the jobs!): what, where, in which context ...
2. Define and register runners: hosts for the jobs
3. 😦 Describe (yaml) the jobs in the file .gitlab-ci.yml

Advice: many templates can be found. Get inspired, copy and paste are your friends.

Vocabulary

- job : a sequence of actions to be executed in some pre-defined context, on a runner
  - Each job is run independently of the others
  - Everything is removed when the job finishes but the artifacts
- artifacts : some directories and files to be kept and transfered between jobs. Intermediate build results
- runner : some machine, hosting and executing a job
- executor : used to run your job on the runner (shell, Docker, ...)
- pipeline : a sequence of (possibly dependant) jobs
  Each pipeline corresponds to a single commit
- stage : of the pipeline, may contains several jobs, parallel execution

Let's start the demo!

- The project: *https://gricad-gitlab.univ-grenoble-alpes.fr/repro4research/demos/ci-montpellier*
- First step: grant access to all attendees of the tutorial → Gitlab API

A quick word about Gitlab API

A tool to interact with the platform, to automate some operations.

A possible way to "talk" to the API: python-gitlab package

- Python scripts to control (from your laptop) and automate actions inside your projects
- Pre-requisite: have a personnal token
  ❓ Gitlab doc - Personnal token

🧳 Demo Project Demos/Gitlab API, you'll learn to

1. create a personnal access token,
2. use a script to register everybody into the group repro4research/Demos

A first simple job

```yaml
my_job:
  script:
  - ls
  - whoami
  - uname -a
  - source scripts/make_something.sh
```

- A (new) language, yaml
- A name: my_job, could be anything
- Some instructions to be executed: keyword **script** as you would execute them in your Terminal.

👉 **Demos**, let's see

- The pipeline editor and the Web IDE
- The content of the Build Menu
- What's happening when a job is launched

One step further ...

👉 **Demos**: build and make available a pdf file. Let's see

- Artifacts
- Image keyword and Docker executor

A few words about docker

https://www.docker.com/

An opensource platform to create, deploy and manage virtualised application containers on an operating system.

🤔

A few words about docker

https://www.docker.com/

An opensource platform to create, deploy and manage virtualised application containers on an operating system.

🤔

Image: a "package" which contains everything needed to run our application

Container: lightweight execution environment, alternative to virtual machines

The container is built/started from the image

In practice:

- run applications/services (python, g++, R, latex ...)
- in an environment of your choice (within certain limits) on the machine of your choice.

For instance, on my Mac laptop, I can launch a 'linux ubuntu' session with the command

```
docker run -ti ubuntu
# or debian
docker run -ti debian:latest
# or ...
```

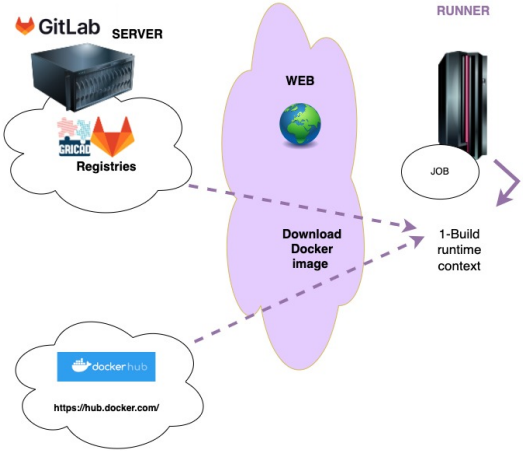
A container (an "instance" of the image) from an image (the model, ubuntu or debian in our case) on which I can run linux commands, compile code and so on.

What's the point?

- Access to a wide range of systems and tools potentially unavailable on your OS
- Easy to reproduce users environments
- Docker is available as an executor for gitlab-ci
- You can use the CI to create your own images and save them in a Gitlab project (gitlab registries, later, be patient).

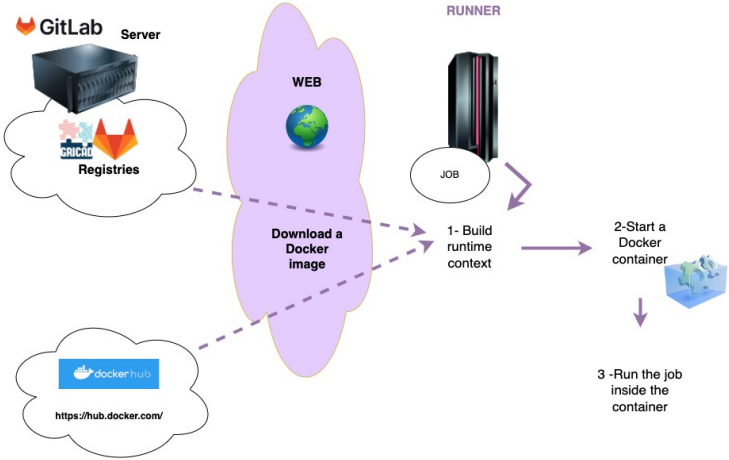
👉 An essential tool to ensure (more or less) reproducible environments

Docker executor for Gitlab CI



Registries : a set of downloadable images

Docker executor for Gitlab CI



Container: an isolated execution context for each job

Build and publish web pages - Gitlab-pages

A tool to automatically publish a static web site associated with your project

- Private or public pages
- Publication and hosting delegated to Gitlab
- Doc : *https://about.gitlab.com/features/pages/*

How?

1. Choose a modern site generators (e.g. Pelican, Sphinx, Mkdocs ..., more here)
   A tool able to generate html files
2. Save sources in a gitlab project
3. Write a job named "pages"

Gitlab Pages demo

👉 **Demos**, let's see

- Stage
- Pipeline
- Pages
- Jobs dependencies

🧳 A complete project with pdf builder and gitlab-pages

Control the CI: rules, conditions and tags

CI: easy to setup. You should pay attention to the ecological footprint and resource consumption ☹️

👉 Best practice: think, and add rules to run only what is really necessary

Control the CI: rules, conditions and tags

CI: easy to setup. You should pay attention to the ecological footprint and resource consumption ☹️

👉 Best practice: think, and add rules to run only what is really necessary

Control CI is quite easy, with either:

- Commit messages
  e.g.: do not execute CI for every commit → add [skip CI] in the commit message
- Rules in yaml
  e.g.: a reduced pipeline for devel. branches (and the whole stuff for main or releases), manual control of jobs ...

👉 **Demos**: let's see "when" and "rules" keywords

Back to the runners

Runner: a host (computer, virtual, whatever ...) to collect and run CI tasks.

On GitLab, either:

- shared runners available (⚠️ depends on the platform) for all projects

Back to the runners

Runner: a host (computer, virtual, whatever ...) to collect and run CI tasks.

On GitLab, either:

- shared runners available (⚠️ depends on the platform) for all projects
- self-managed, private runners, linked to a single project or group

  👉 *could be any machine at your disposal (laptop, server, virtual machine ...)*
  👉 *could be isolated in a private network. It just needs to be able to ping (http) the gitlab server and to clone a project.*

Back to the runners

Runner: a host (computer, virtual, whatever ...) to collect and run CI tasks.

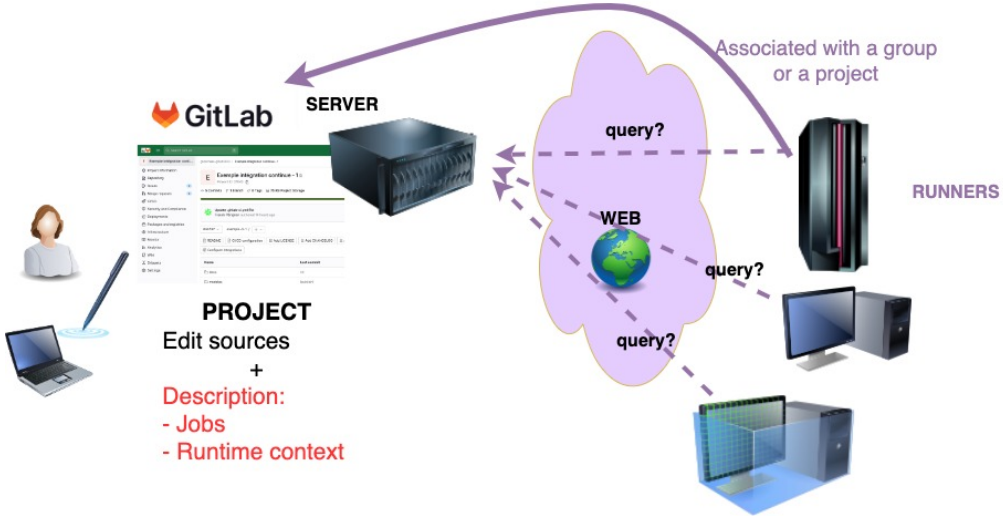
On GitLab, either:

- shared runners available (⚠️ depends on the platform) for all projects
- self-managed, private runners, linked to a single project or group

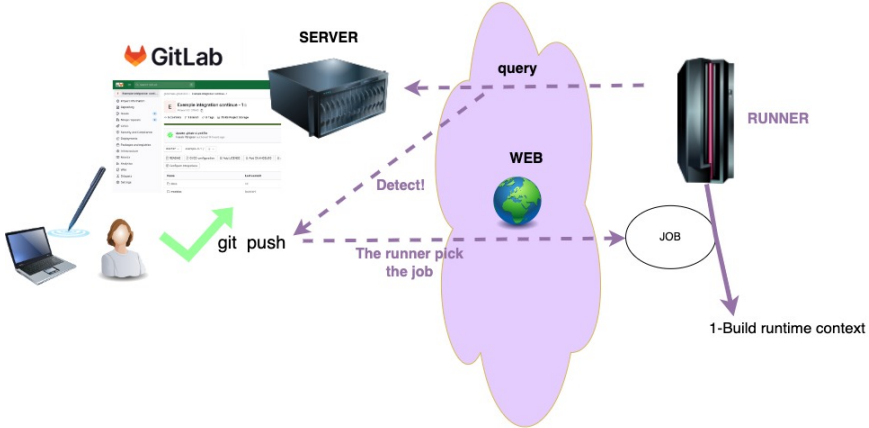
  👉 *could be any machine at your disposal (laptop, server, virtual machine ...)*
  👉 *could be isolated in a private network. It just needs to be able to ping (http) the gitlab server and to clone a project.*

Where to see/find them?

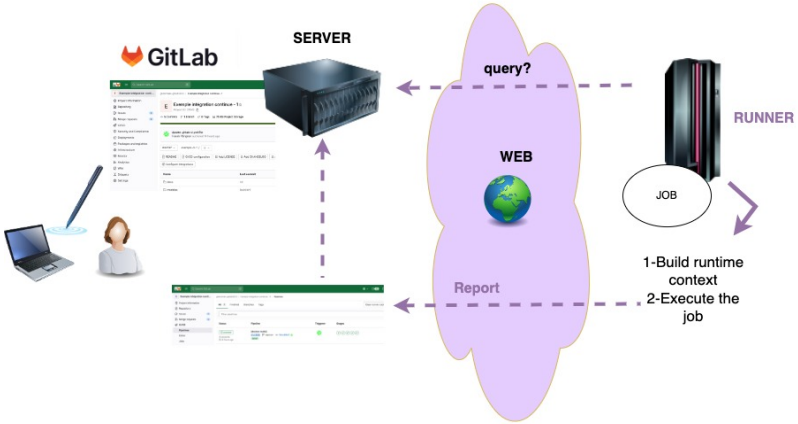
- Build/runners menu (group)
- Settings/CI CD menu (group)



Several runners may be connected to a single group or project

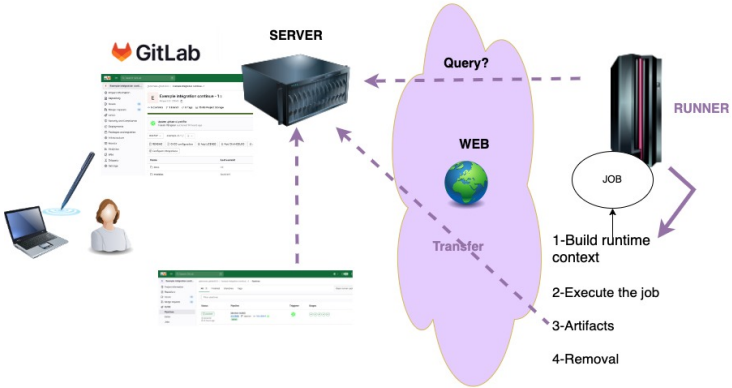


Git push ⇒ each job is "taken" by a runner (👍 tip: use tags to control runner scope)



On the runner

- Creation of a 'context' (executor ...)
- Clone the project into the 'context' and execute the required operations (job)
- Send a report to the project owning the job



Artifacts : files or directories, results of the job, that must be kept (for some time)

- Saved on the Gitlab server. ⚠️ Pay attention to the disk memory footprint
  - Set 'expiry date'
  - Save only what is required
- Possibly transfered between jobs

How to declare/install a runner?

1. Find a host and install gitlab-runner (and Docker)
   Standard and easy to install - Example: gitlab runner for ubuntu/debian

How to declare/install a runner?

1. Find a host and install gitlab-runner (and Docker)
   Standard and easy to install - Example: gitlab runner for ubuntu/debian
2. Ask for a new runner in the group or the project (Gitlab web page)
   - Project: Settings →CI/CD →New project runner
   - Group: Build →Runner →New group runner

How to declare/install a runner?

1. Find a host and install gitlab-runner (and Docker)
   Standard and easy to install - Example: gitlab runner for ubuntu/debian
2. Ask for a new runner in the group or the project (Gitlab web page)
   - Project: Settings →CI/CD →New project runner
   - Group: Build →Runner →New group runner
3. Register the runner with the project or group (Command line, on the runner)
   👉 setup communication between the runner and the gitlab server

```
gitlab-runner register \
--url https://gricad-gitlab.univ-grenoble-alpes.fr \
--token <SOME-TOKEN>
```

👉 **Demos**, let's see

- Runners registration
- Runner tags
- Executors

What have we achieved so far?

- ✅ A project with a .gitlab-ci.yml file. CI is on.
- ✅ One or more runners available and ready
- ✅ Basic keywords known and understood (image, artifacts, script, ...)

What have we achieved so far?

✅ A project with a .gitlab-ci.yml file. CI is on.
✅ One or more runners available and ready
✅ Basic keywords known and understood (image, artifacts, script, ...)

👉 **Demos**: let's turn to a "real" software project
Step by step demo ...
🧳 CI in a software project - Complete example

Démo : CI/CD for a "real" software environment

What do we need/want?

- Configure, build and test a software (cmake, make, make test)
- For differents OS (ubuntu, debian ...)
- For different configurations (Debug, release ...)
- Generate documentation and publish the software webpage
- Control and automate releases publications
- Collaborative work (dev and users)
- ...

👉 **Demos**

A first pipeline

Configure, build and test a software for a given context

👉 **Demos**, let's see

- CI variables
- before script
- needs

A word about CI (predefined or not) variables

Some kind of environment variables to control the behavior of your jobs and pipelines, among other things.

- A lot of **predefined variables**: e.g. *CI_PROJECT_DIR*. More: try "env" in your jobs.
- Defined in your CI script (global or job level)

```
variables:
name: value
```

- Defined with the user interface of your group/project: Settings → CI/CD → variables.
  May be masked or protected.

👉 **Demos**, let's try to add other operating systems and to use CI variables

2 problems arise in the previous demo:

- annoying repetition
- before script not compatible with all OS, costly in time and resources and rather useless: we don't need to test apt or equivalent tools!

2 problems arise in the previous demo:

- annoying repetition
  👍 use templates!
- before script not compatible with all OS, costly in time and resources and rather useless: we don't need to test apt or equivalent tools!
  👍 build your own images!

CI templates

2 problems arise in the previous demo:

- annoying repetition
  👍 use templates!
- before script not compatible with all OS, costly in time and resources and rather useless: we don't need to test apt or equivalent tools!
  👍 build your own images!

👉 **Demos**, let's see

- templates
- variables
- reports in artifacts

Jobs to build Docker images and Gitlab registries

As mentioned before:

- before script not compatible with all OS, costly in time and resources and rather useless: we don't need to test apt or equivalent tools!
  👍 build your own images is the solution!

👉 **Demos**, let's see

- Kaniko and CI to build docker images
- Gitlab registries

A few words about git workflow

Workflow a method to organize your repository management

Why organize?

- Different people, different habits in different contexts
- A complicated even chaotic management, potentialy inefficient.
- Waste of time!

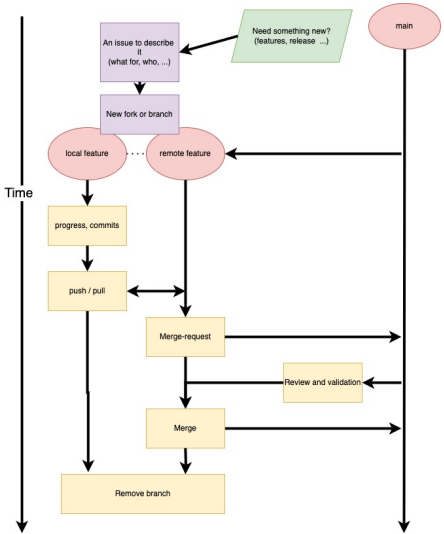
Enforce reproducibility!

Which workflow?

Many possibilities, but no single answer...

- Centralized workflow: everyone works on the same branch 🙁
- Git workflow
- GitHub flow
- OneFlow
- ...

Gitlab, git, workflow, ... A few tips



1. Use issues to declare every problem, new development, etc
2. Create a new branch (or fork) for each new feature, release, bug resolution, etc.
3. Use merge-requests and benefit from the review process.
4. Synchronize your repository regularly (pull/push). The longer a branch lives, the harder it will be to merge ...

Control the CI workflow

Ok, you choose a git workflow, good. But how can you handle it properly with the CI? Same behavior for all branches, for MR, ... ?

No!

- Add rules
- Use **Workflow keyword**: control when pipelines are created (among other things)

👉 **Demos**, combine branch, MR and issues. Use a CI workflow and rules.

Let's switch to Continuous Delivery

✅ CI to configure, build, test our software
✅ Able to switch between different contexts (OS, parameters ...)
✅ Control the workflow (rules, ...)

More?

- Release
- Install the software and make it available
- Produce documentation ...

Make your software available with CI/CD

👉 **Demos**

- CI/CD to deliver "ready-to-use" Docker images, with your software
- How to write into other projects registries

Prerequisite: a deploy token
How?

- Must be owner in a group or maintainer in a project
- Settings → repository → deploy token
- Use the token to feed *CI_DEPLOY_USER* and *CI_DEPLOY_PASSWORD* variables

Release

👉 **Demos**

- CI to produce a release of your software each time a new tag is created

Triggered jobs and cross projects

👉 **Demos**

More?

- **Gitlab, SWH and HAL**
  - Add a codemeta.json into your git repository to prepare the way for HAL (👍 *https://codemeta.github.io/codemeta-generator/*)
  - "Declare" your git repository to SWH
- CI with a guix image?