2° Forum della Ricerca Sperimentale e Tecnologica

# A small-scale support infrastructure for GitLab CI/CD

C. Urban, D. Tavagnacco, M. Sponza, C. Knapic and M. Landoni

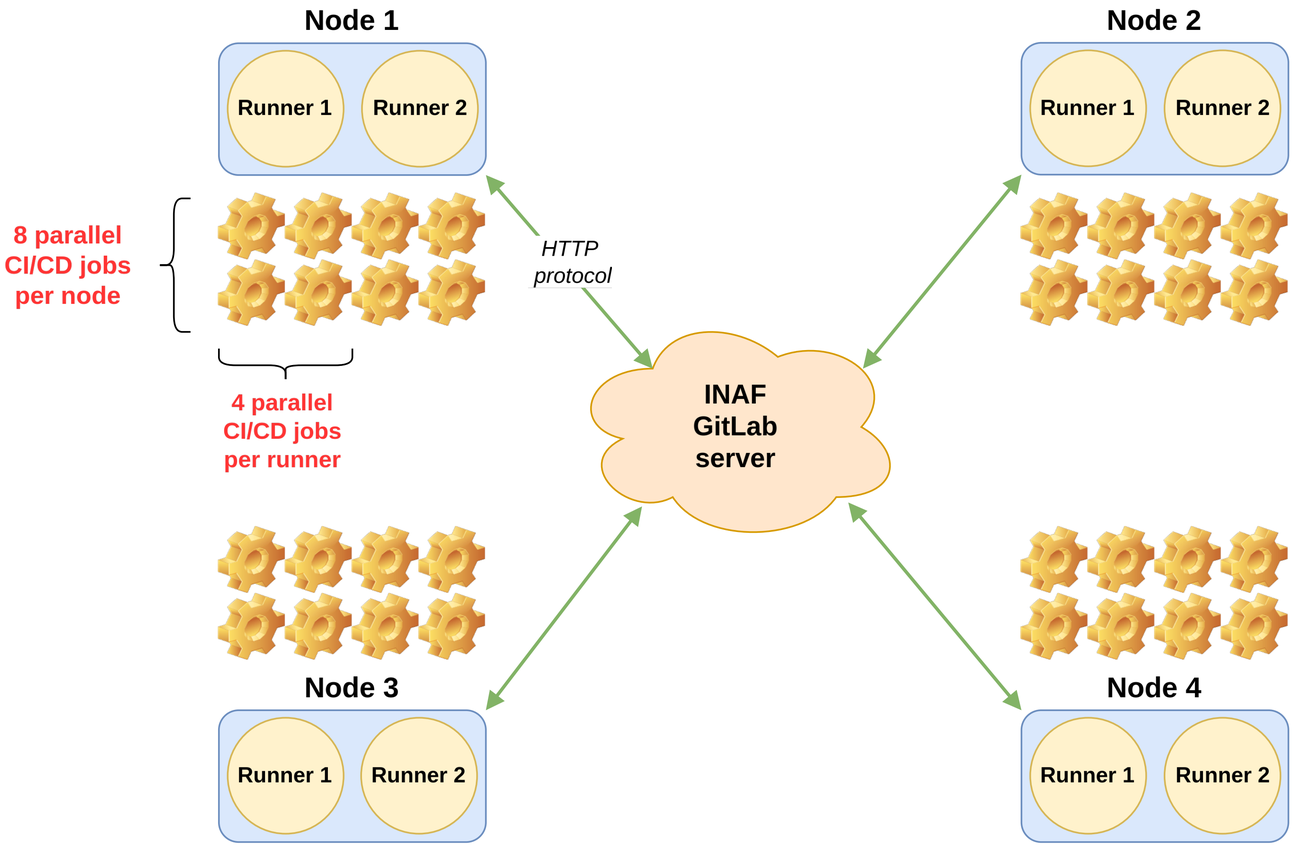
INAF - Astronomical Observatory of Trieste

## Introduction

Continuous integration and Continuous Delivery/Deployment (CI/CD) constitute a pillar of the present-day software development. Whether you need to compile, test, distribute your code, or simply automatically generate documentation, there is always a pipeline involved in one way or another. GitLab allows us to define a CI/CD pipeline in a relatively simple way, by filling in a file called `.gitlab-ci.yml` and saving it in the root of our repository. Runners are the applications that are responsible for executing CI/CD jobs. More precisely, jobs are submitted to runners and each runner is associated to a well-defined executor type (i.e. Shell, Docker, etc...). The IA2 group, in collaboration with other working groups at INAF, has started to build up a simple test infrastructure whose purpose, once in production, would be to guarantee an adequate processing power in order to support the execution of several simultaneous CI/CD jobs. This kind of approach could help a lot our developers and researchers.

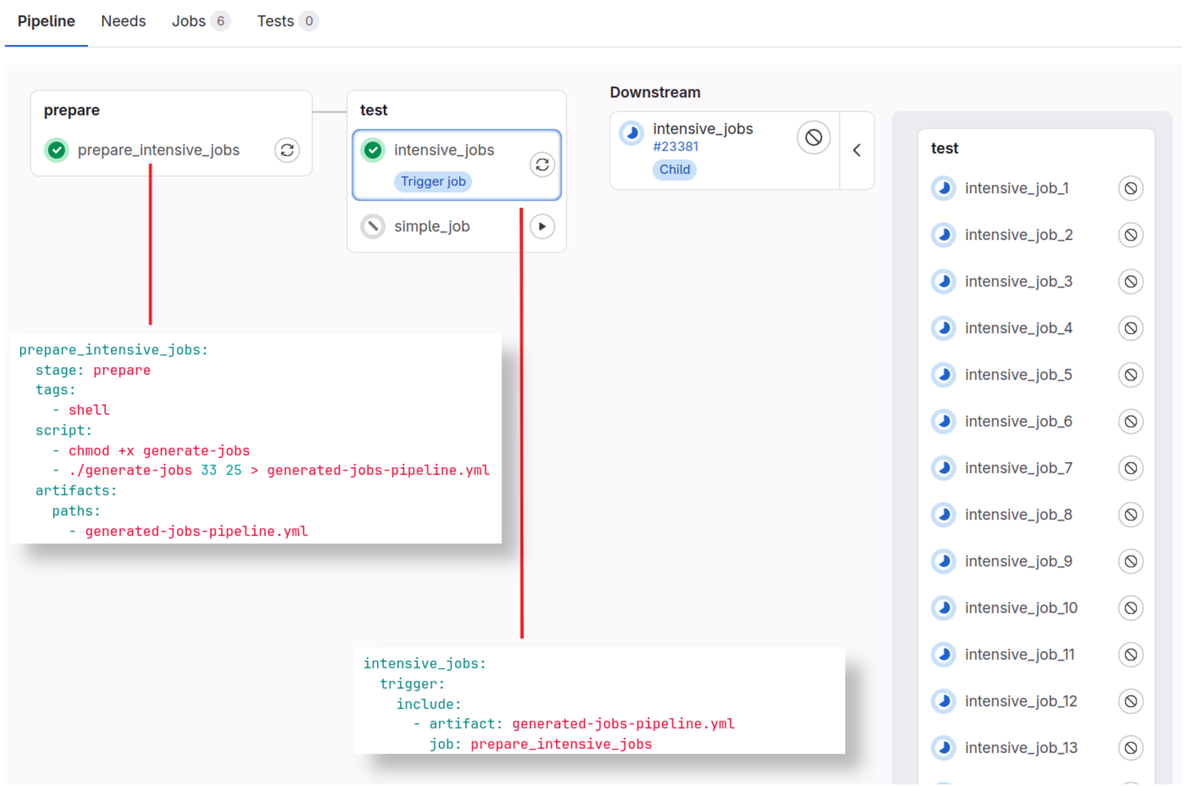
## Test platform

In order to perform our tests, we set up four physical machines with a Intel(R) Xeon(R) CPU E5645 @ 2.40GHz (1 thread/core, 6 cores/socket and 2 sockets) and 96 GB of RAM (12 banks of 8 GB each one). On each of these nodes we configured two runners with a Docker executor. These runners are "Instance runners", namely globally visible from all the repositories present on the INAF GitLab installation. Each runner was configured to be able to handle four parallel jobs. In this way, with four nodes, **we obtained a total amount of 32 parallel CI/CD jobs** (4 jobs/runner, 2 runners/node and 4 nodes). The runners communicate with the INAF GitLab server through HTTP requests.




## Exploiting dynamic child pipelines for repetitive tasks

When we are dealing with the generation of a pipeline containing many identical jobs, it can be particularly useful to take advantage of **dynamic child pipelines**. Roughly speaking, we can state that these pipelines **consist of two main jobs**. The first one produces an artifact representing a pipeline definition: this pipeline contains our jobs configuration and is generated through instructions in any programming language (e.g. a Bash script). The second job builds a trigger to launch our jobs.




## Runner configuration

We can configure a runner in two steps:

1. Create a new runner using the GitLab user interface
2. Install/register the runner using the token obtained at point 1.

We usually want to place runners on a different node than the one hosting our GitLab server. Potentially, we could install a runner even on our laptop or on an embedded device, such as a RaspberryPi.

## Pull policy and concurrent pull of Docker images

The runners with a Docker executor allow us to specify the so-called "pull policy" for the images we want to use. The default policy is to always pull images (i.e. `pull_policy = "always"`). Of course, this is not the best choice in terms of efficiency and, in fact, there is the possibility to choose another policy that allows to **pull an image only if that image is not already stored locally** (i.e. `pull_policy = "if-not-present"`). However, we have proven that in the case of concurrent pull, namely automatically launching all the 32 jobs, the first-time pull is attempted by all the jobs at the same time. It is possible to set more than one policy or repeat the same policy, in the eventuality the pull operation fails.

## Restrictions on Docker images

Giving unrestricted access to all the public available Docker images could pose a potential security risk. Within the runner configuration file we can define a list containing the names of the allowed images (e.g. `allowed_images = ["python:3.11", "postgres:16"]`). Next, if we override the "image" setting in our pipeline configuration file with an image that is not in that list, the pipeline will fail.

## References

GitLab runners: `https://docs.gitlab.com/runner/`
GitLab CI/CD pipelines: `https://docs.gitlab.com/ee/ci/pipelines/`

## Contacts

Cristiano Urban: `cristiano.urban@inaf.it`
Daniele Tavagnacco: `daniele.tavagnacco@inaf.it`
Massimo Sponza: `massimo.sponza@inaf.it`
Cristina Knapic: `cristina.knapic@inaf.it`
Marco Landoni: `marco.landoni@inaf.it`