Testing, also for extensions

DIRAC Users' Workshop

June 20th, 2024

https://indico.cern.ch/e/duw10

Federico Stagni
Technical coordinator
On behalf of the DIRAC consortium

The usual flow of tests

UNIT

Developers' business.

Plus:

- fast to run (most of them)
- Measure-able
- Developer's friendly

Cons:

- (lots of) mocking involved
- Can't test everything

INTEGRATION

DIRAC and Pilot integration tests

- Subject of this presentation

SYSTEM + ACCEPTANCE

This is what we do using the Dirac certification machine (which, as you know, it is changing)

Really using the "Grid"

Why this pres?

- Will be useful for the hackathon part later today
- Very useful for developing and testing, yet I see this not always used
  - Maybe it was never comprehensively explained
- There are improvements added in the last few months

- Q: out of those who have a DIRAC extension, who of you setup the integration tests for it? (and run it in the CI?)
  - If you do not, why?
    - Because you should!

DIRAC integration tests

NB: this is documented

DIRAC integration tests


```
❯ ./integration_tests.py --help

Usage: integration_tests.py [OPTIONS] COMMAND [ARGS]...

Run the DIRAC integration tests.
A local DIRAC setup can be created and tested by running:
  ./integration_tests.py create

This is equivalent to running:
  ./integration_tests.py prepare-environment
  ./integration_tests.py install-server
  ./integration_tests.py install-client
  ./integration_tests.py install-pilot
  ./integration_tests.py test-server
  ./integration_tests.py test-client
  ./integration_tests.py test-pilot
```

Requirement: you need docker

Using docker-compose, this effectively creates a fully autonomous server, client, and pilot

NB: this runs on your local code (pip install –e)

```
Commands
create                Start a local instance of the integration tests
destroy               Destroy a local instance of the integration tests
prepare-environment   Prepare the local environment for installing DIRAC.
install-server        Install DIRAC in the server container.
install-client        Install DIRAC in the client container.
install-pilot         Run a pilot in a container.
test-server           Run the server integration tests.
test-client           Run the client integration tests.
test-pilot            Run the pilot integration tests.
exec-server           Start an interactive session in the server container.
exec-client           Start an interactive session in the client container.
exec-pilot            Start an interactive session in the pilot container.
exec-mysql            Start an interactive session in the server container.
list-services         List the services which have been running.
runsvctrl             Execute runsvctrl inside the server container.
logs                  Show DIRAC's logs from the service container.
```

DIRAC integration tests


```
./integration_tests.py install-server
./integration_tests.py install-client
./integration_tests.py install-pilot
./integration_tests.py test-server
./integration_tests.py test-client
./integration_tests.py test-pilot
```

Switches:
- Operating System
- MySQL
- IAM
- Elastic/OpenSearch

Submits job(s) to the server

Run the pilot, which will picks up the job(s) previously submitted

Runs workflows locally (in the environment created by the Pilot)

Recently, pilot was added, making this a **Grid in a box**

DIRAC integration tests -- for extensions

Extensions

Integration tests can be ran for extensions to DIRAC by specifying the module name and path such as:

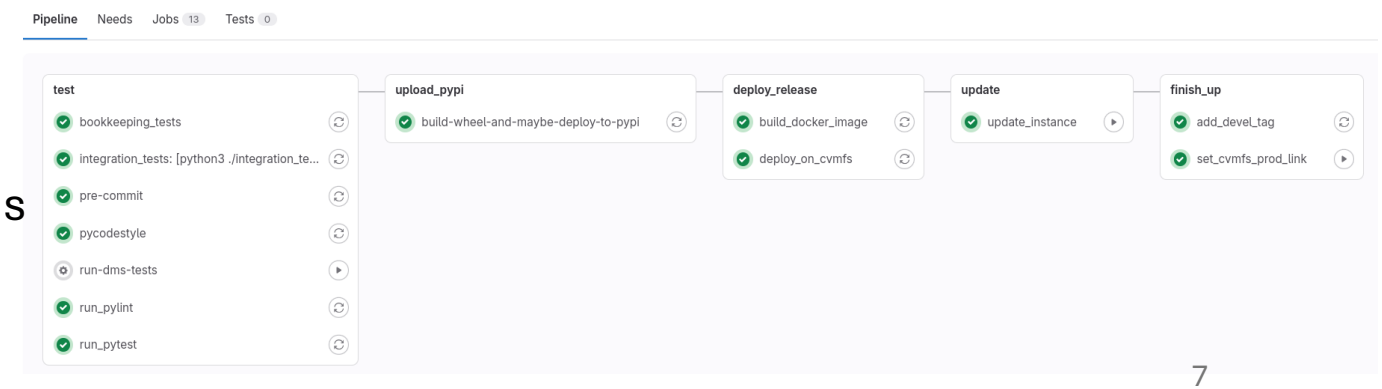
```
./integration_tests.py create --extra-module MyDIRAC=/path/to/MyDIRAC
```

This will modify the setup process based on the contents of `MyDIRAC/tests/.dirac-ci-config.yaml`. See the Vanilla DIRAC file for the available options.

python3 ./integration_tests.py create "LHCBDIRAC_RELEASE=$${lhcbdiracTag}" TEST_HTTPS=No --extra-module LHCbDIRAC=../LHCbDIRAC

LHCbDIRAC example with:

- Extra service (Oracle DB)
- Pilot extension, and pilot specific commands



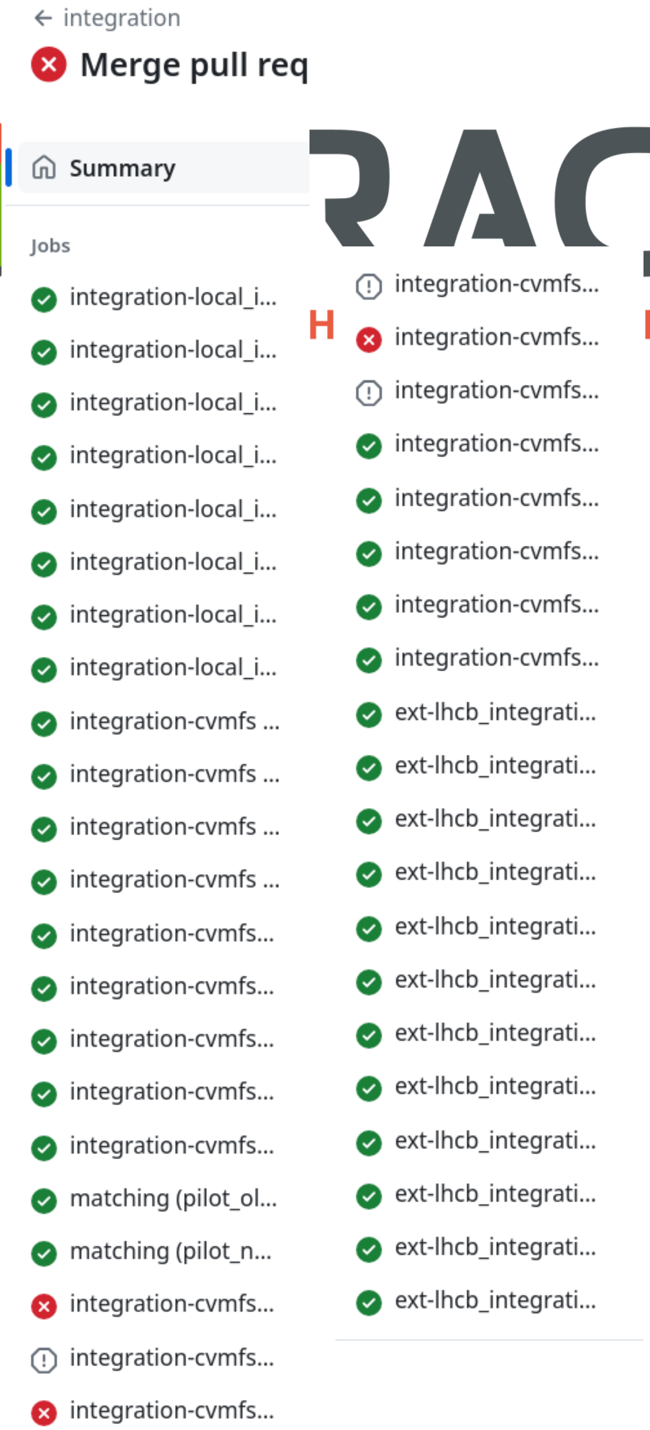
Pilot integration/system tests

NB: this is of interest if you either have a DIRAC extension, and even more if you have a Pilot extension

Integration tests

Goal: running pilots with different configurations

- DIRAC code branches (production and pre-releases)
- VOs
- Computing Element types
  - e.g. Singularity or Pool
- CVMFS or not
- JSON formats
- Match
- Pilots extensions (LHCb and Belle2 are there)
  - So, if you have a Pilot extension, you can add it there (no need to setup this in the CI for your extension)



Security

- The pilots need a CS to where to connect to, so using the DIRAC certification setup (atm lbcertifdirfac70)
- In order for this to work, an accepted certificate is needed
  - Only running on Pilot code forks (i.e. in https://github.com/<your_user>/Pilot/actions/)
  - Secrets need to be added to your GitHub Actions config
    - And for this, you need to ask me

DiracX integration tests

DiracX

- Similar concepts to the DIRAC integration tests, prob a simpler/better implementation
  - At least for the server
- Relies on `run_demo.sh`, i.e. an automated full local installation of a kubernetes cluster running DiracX and its dependencies
- Simply started with `pytest` (see temporary doc)
- Run in the CI, with Coverage collection
- Also run DIRAC integration tests against DiracX services (fails for now, of course)

Questions?