Automated testing and validation of MPS applications using Hardware-in-the-Loop

Spring EPICS collaboration Meeting 2026

Marius Braets (Intern), Rémi Nicole & Victor Nadot

Overview

1. Introduction and motivations
2. Solution
3. Results
4. Conclusion

1. Introduction and motivations

Context

- Developing critical Machine Protection System applications that are FPGA-based (Must stop the beam in < 12 µs)

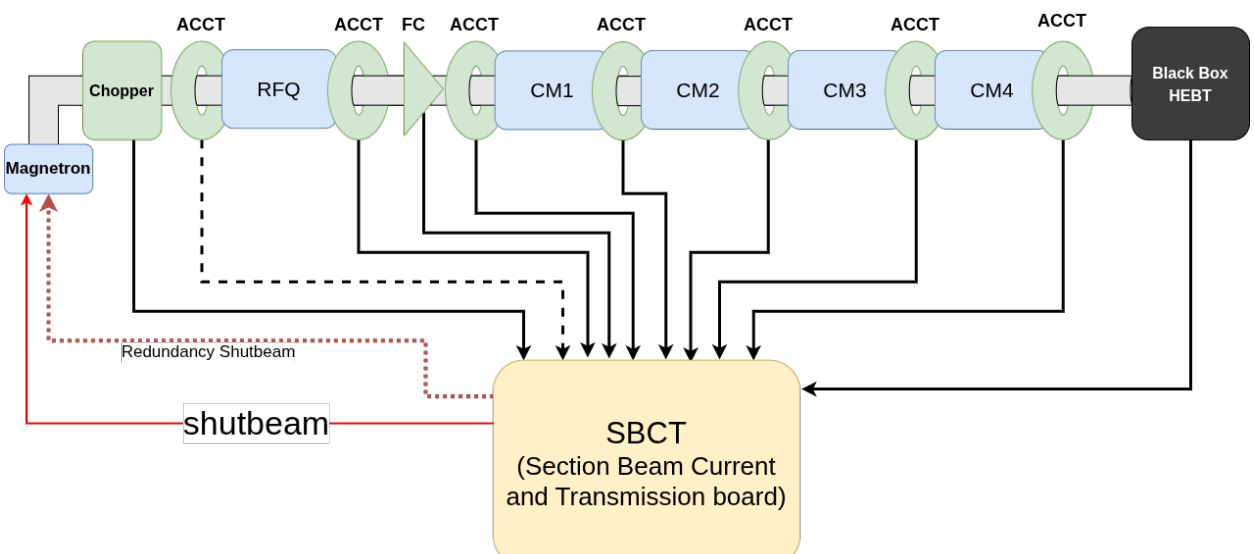
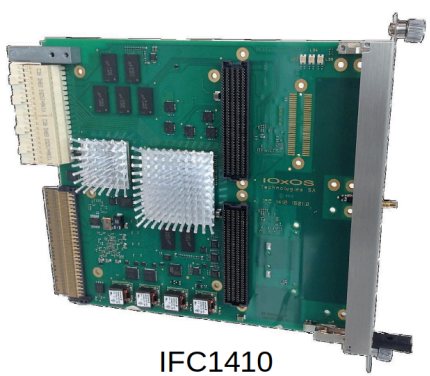
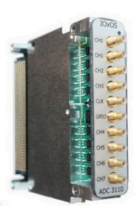


Fig. 2. – SBCT: one of the MPS applications

- 4 firmwares that are MPS related:
  - 2 are generic and used for different accelerators
    - Different EPICS environments (IEE, EPNix)
  - 2 are specific to an accelerator project
  - Different FMC board configurations for the projects (IOxOS: adc3111 or adc3117)





Current tests

- The firmware logic is well tested (VHDL testbenches)
- Other tests exist but are manual:
  - Deployment on board (bitstream, Linux, and EPICS IOC)
  - For EPICS tests, I have several WeTest scripts

Motivations

- Manual testing is time-consuming and prone to human error
- Ensuring safe deployment of new versions in production

⚠ Avertissement

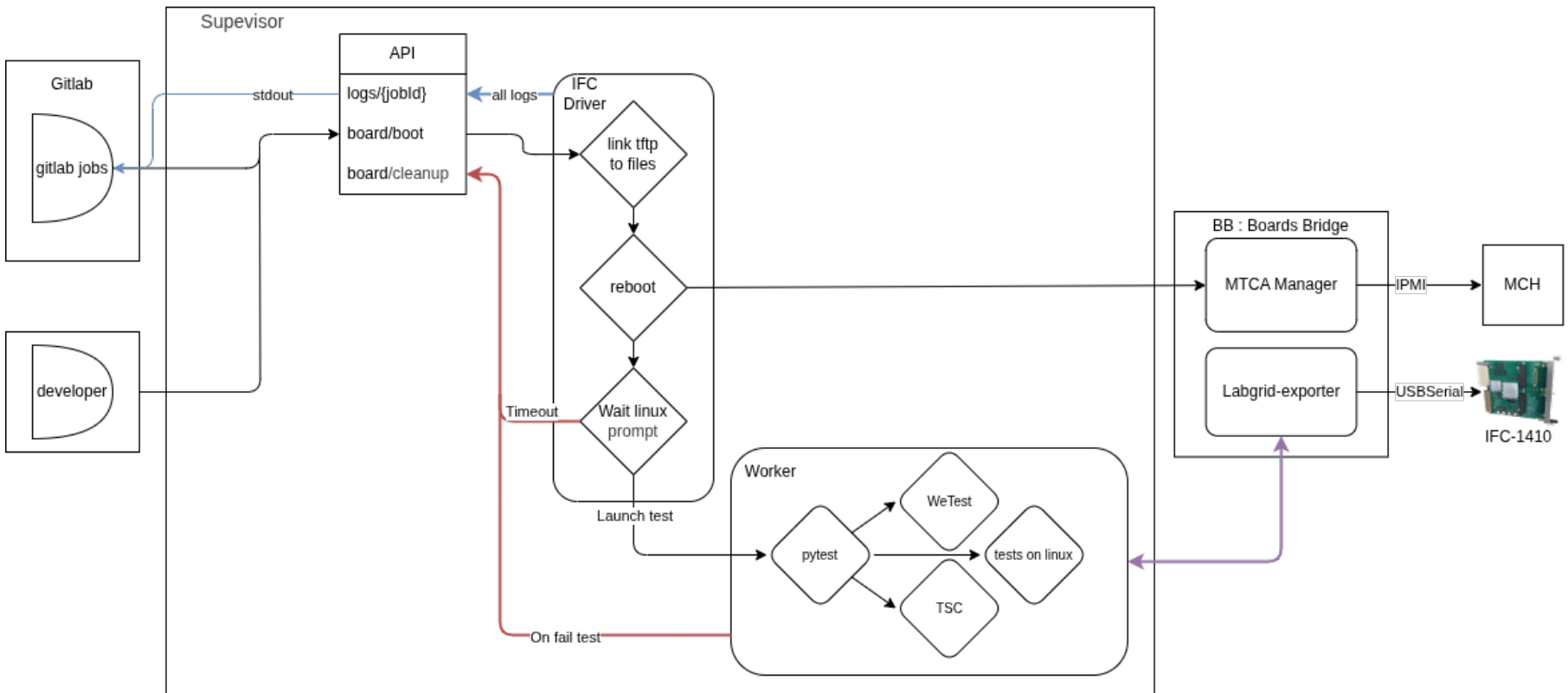
We cannot fail: deploying a bad version could lead to machine damage

? Question

How do we perform hardware-in-the-loop tests?

2. Solution

Architecture



Setup in the Lab

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD



Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD




Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD





Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python development
- Nix packaging
- GitLab CI/CD

Software stack

- Labgrid framework
- Python developmen
- Nix packaging
- GitLab CI/CD

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi



Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi




Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi





Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and validate updates of the FPGA application
  - Rémi: to validate upgrades of IFC1410 NixOS embedded images
- Marius:
  - Developer
- Maintainer:
  - Rémi

Project Scope

- Users:
  - Victor: to test and [illegible] PGA application
  - Rémi: to validate upgrad[illegible] IFC1410 Ni[illegible]em[illegible]ed images
- Marius:
  - Developer
- Maintainer:
  - Rémi

3. Results

How to use the tool?

- Provide a new version (firmware coupled to software via git tag)
- Configuration:
  - Add CI access permissions
  - Add dedicated job to your CI

```
- |
    export REQUIREMENTS="$(cat config/hardware.json)"
    VERSION="$(cat config/version.txt)"
    . cicd/startup/iocs/envfile
    export CONFIG='{ "Sec": "'$Sec'", "bootType": "iee", "topName": "topSbct", "version": "'$VERSION'", "git_hash": "'$CI_COMMIT_SHA'", "token'
    export TESTS_FOLDER="tests"
    export DEBUG=true
    python3 cicd/hil_client.py
```

How to use the tool?

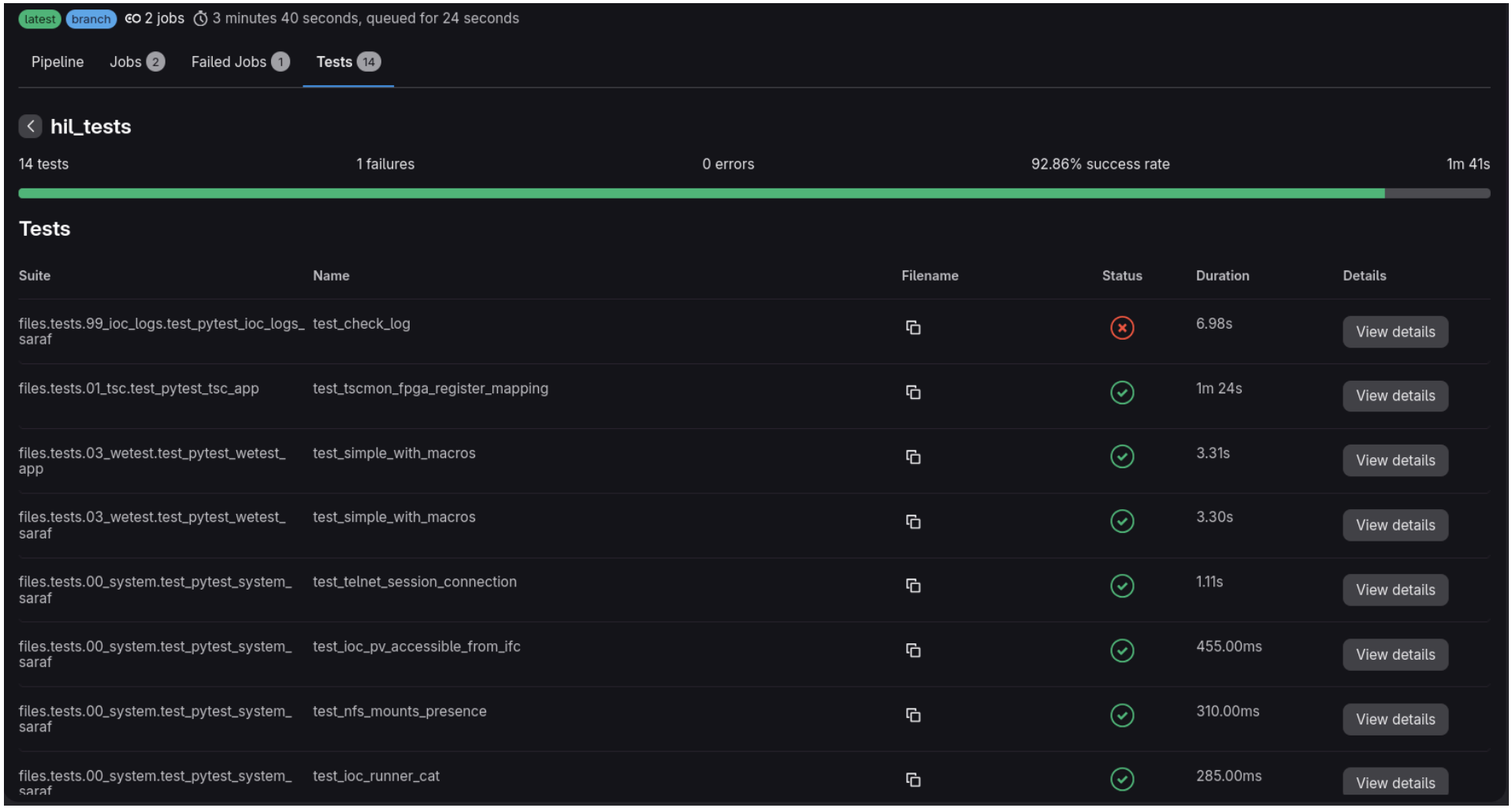
- Pytest: create and call your tests

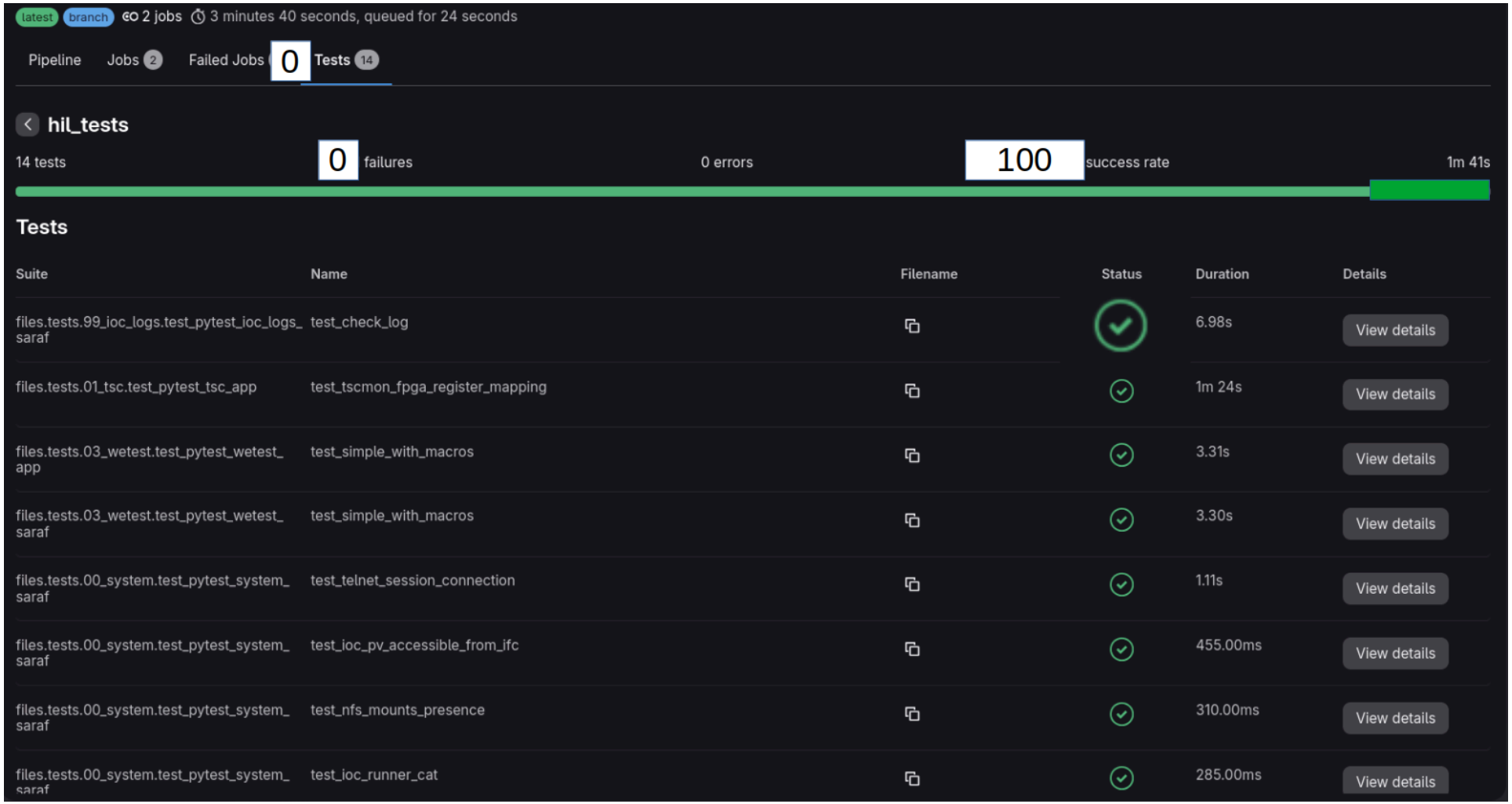
```python
def test_ioc_runner_started(shell_cmd: ShellDriver):
    result, _, exitcode = shell_cmd.run("systemctl is-active ioc-runner@*", timeout=10)
    assert ['active'] == result
    assert exitcode == 0

def test_simple_with_macros(wetest):
  Sec=wetest.Sec
  returncode, stdout, stderr = wetest.run("03_wetest/misc/basic_test_with_macros.yml", extra_args=['--no-pdf-output', '-m', f'Sec={Sec}'])
  if returncode != 0:
      pytest.fail(f'Command failed with returncode: {returncode}\nstdout:\n{stdout}\nstderr:\n{stderr}')
  assert 'FAILED' not in stderr
```

HIL test coverage

- System
  - NFS folders mounted
  - TscMon working
  - Register mapping test (TscMon level)
  - Check procServ connection
  - IOC services running
  - Check PV connection
- WeTest
  - Register mapping test
  - Functional tests
- IOC shell parser


latest
branch
2 jobs
3 minutes 40 seconds, queued for 24 seconds
Pipeline
Jobs 2
Failed Jobs 1
Tests 14
hil_tests
14 tests
1 failures
0 errors
92.86% success rate
1m 41s
Tests
Suite
Name
Filename
Status
Duration
Details
files.tests.99_ioc_logs.test_pytest_ioc_logs_saraf
test_check_log
6.98s
View details
files.tests.01_tsc.test_pytest_tsc_app
test_tscmon_fpga_register_mapping
1m 24s
View details
files.tests.03_wetest.test_pytest_wetest_app
test_simple_with_macros
3.31s
View details
files.tests.03_wetest.test_pytest_wetest_saraf
test_simple_with_macros
3.30s
View details
files.tests.00_system.test_pytest_system_saraf
test_telnet_session_connection
1.11s
View details
files.tests.00_system.test_pytest_system_saraf
test_ioc_pv_accessible_from_ifc
455.00ms
View details
files.tests.00_system.test_pytest_system_saraf
test_nfs_mounts_presence
310.00ms
View details
files.tests.00_system.test_pytest_system_saraf
test_ioc_runner_cat
285.00ms
View details


latest
branch
2 jobs
3 minutes 40 seconds, queued for 24 seconds
Pipeline
Jobs 2
Failed Jobs 0
Tests 14
hil_tests
14 tests
0 failures
0 errors
100 success rate
1m 41s
Tests
Suite
Name
Filename
Status
Duration
Details
files.tests.99_ioc_logs.test_pytest_ioc_logs_saraf
test_check_log
6.98s
View details
files.tests.01_tsc.test_pytest_tsc_app
test_tscmon_fpga_register_mapping
1m 24s
View details
files.tests.03_wetest.test_pytest_wetest_app
test_simple_with_macros
3.31s
View details
files.tests.03_wetest.test_pytest_wetest_saraf
test_simple_with_macros
3.30s
View details
files.tests.00_system.test_pytest_system_saraf
test_telnet_session_connection
1.11s
View details
files.tests.00_system.test_pytest_system_saraf
test_ioc_pv_accessible_from_ifc
455.00ms
View details
files.tests.00_system.test_pytest_system_saraf
test_nfs_mounts_presence
310.00ms
View details
files.tests.00_system.test_pytest_system_saraf
test_ioc_runner_cat
285.00ms
View details

4. Conclusion and future

4. Conclusion & Future

- 2 hardware configurations supported
- 1 application CI tested, time to make the tests pass!
- 2+1 additional applications to add to HIL
- The framework is ready: now it is time to use it!
- Special thanks to Marius!

Thank you, any questions?