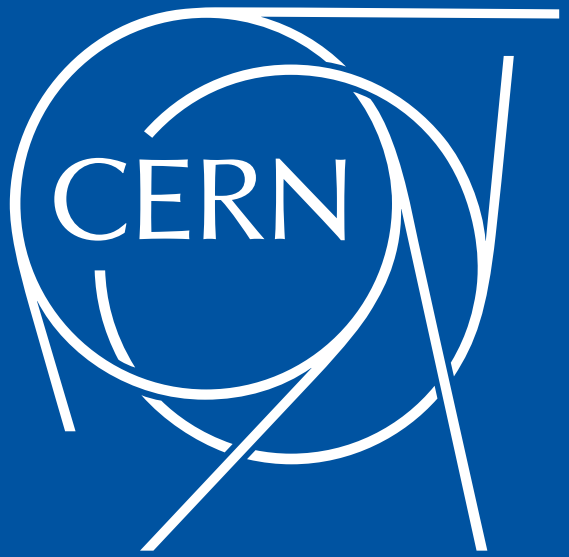

CERN

CI/CD for PLC-based Control Systems

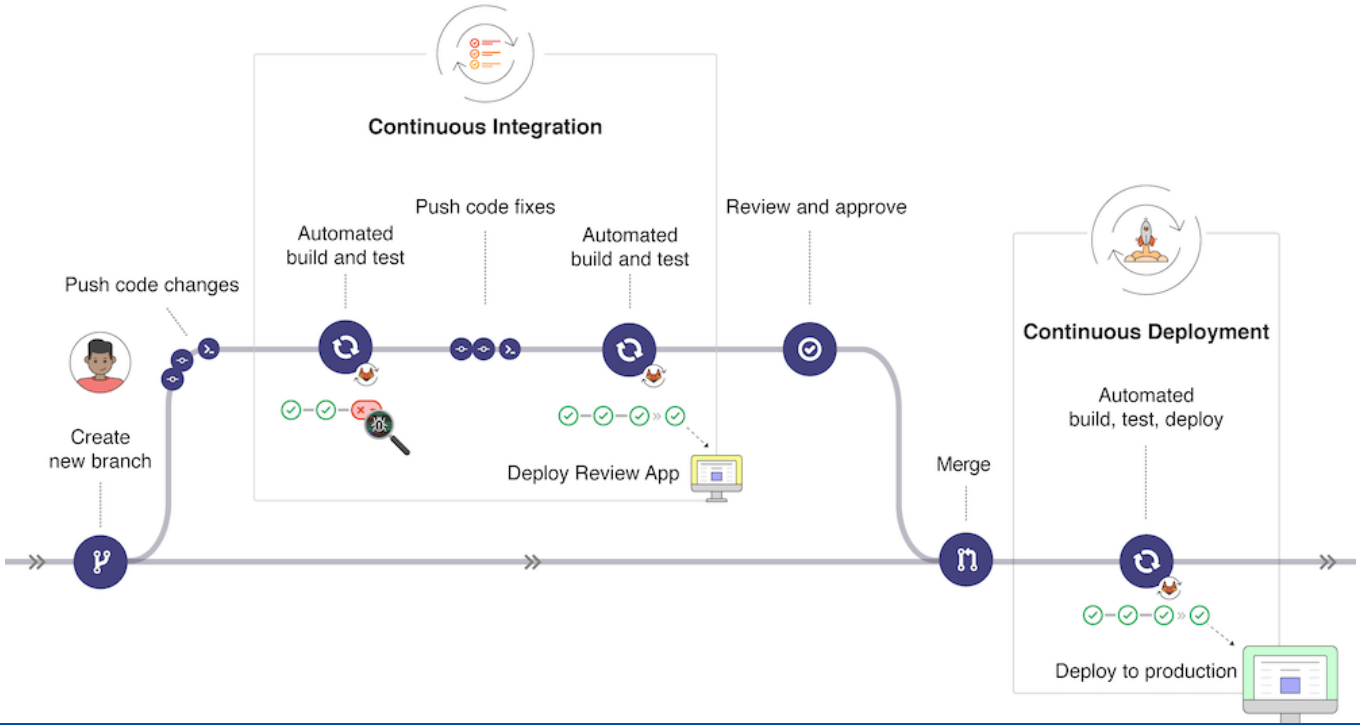
Improving quality assurance, traceability and developer workflows

Brad Schofield, Joao Borrego

Introduction and Motivation

What is CI/CD?

Continuous Integration/Continuous Deployment is a development methodology emphasising frequent, small commits to version control, supported by automated build and test steps



Introduction and Motivation

Why is CI/CD useful?

- Simplifies the work of the developer; only need to concentrate on code changes; build and test are taken care of
- Facilitates cooperation; several developers can work in parallel
- Improves traceability, from bug report/feature request to deployed product

Introduction and Motivation

ICALEPCS 2021

Why is CI/CD difficult to use for PLC development?

- Automation of the build and deployment phases is not trivial
- Implementation of automatic testing is not straightforward

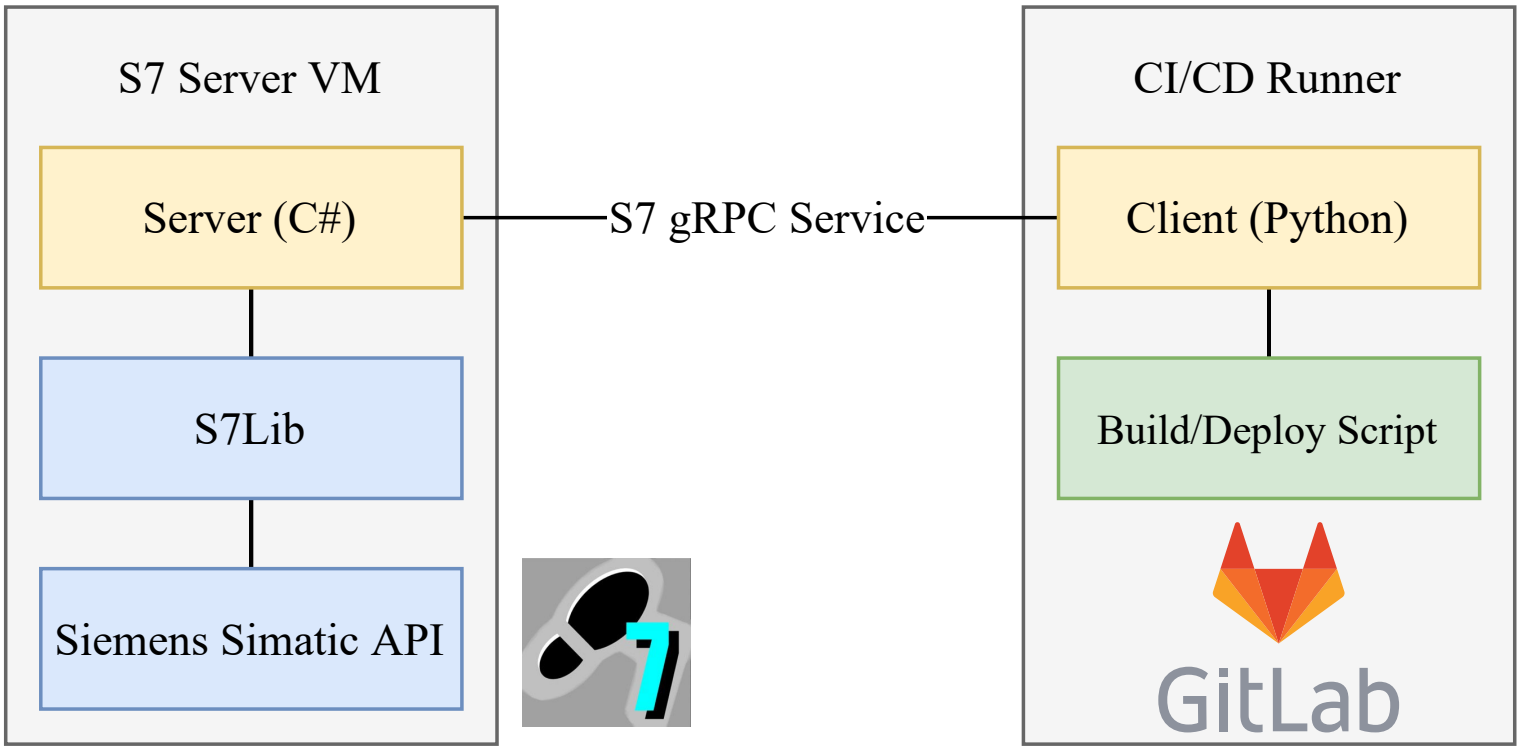
Enabling CI/CD for PLC development

- Creation of tools to automate the build and deploy stages
- Creation of test interfaces which allow suites of fully automated functional tests to be run against physical (or simulated) PLCs

Automated Build Tools

Wrapping API in gRPC Service

- Common approach for all PLC vendors: wrap API (Simatic, TIA or Unity) with a high-level gRPC service

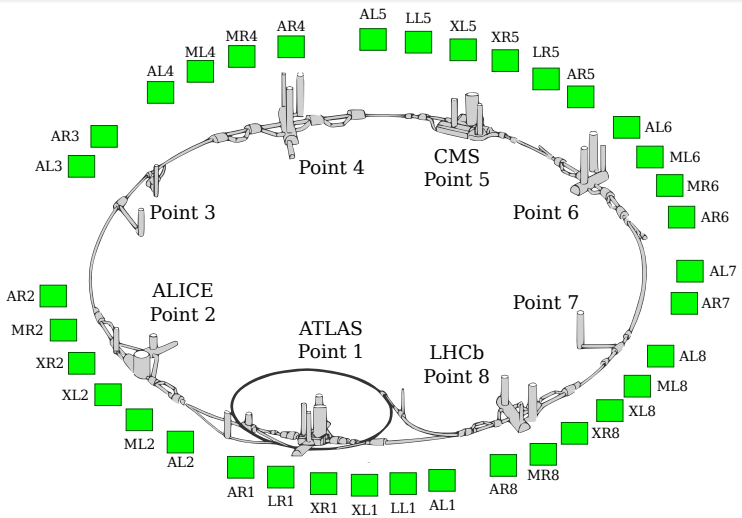


Use Case: LHC Powering Interlocks

ICALEPCS 2021

Description

Interlock system for superconducting magnets of the LHC. 36 PLCs, sharing common code but each configured differently.



Details

- Siemens S7-319 PLCs each with different HW config
- Common source code (SCL, IL) versioned in `git`
- Specific config handled by code unique to each
- Lab setup with single PLC available for testing
- Want to ensure that code tested in lab is correctly deployed to all production projects

Demo Part I: Build Automation

Automated Testing Tools

Approach

- Use OPC UA used to interface PLC with tests
- Use IO simulators used to 'mock' field input
- `Siemens Simulation Unit` provides C API; wrapped in Python package for ease of use
- Use `pytest` to implement the tests. Fixtures manage connections to PLC and IO simulator

```
def test_quench_abort_status_rb(plc_client,
↪   simba):
    """Main Dipole Quench Status test"""
    plc = plc_client.get_root()
    status_abort =
    ↪   plc.get_child("7:A_A_1_ST_ABORT")
    quench_status =
    ↪   plc.get_child(['7:1_Instance1_Circuit1',
    ↪   '7:Quench_Status'])

    simba.write_io_bin('S000I4.1', 1)
    wait()
    assert status_abort.get_value() is True
    assert quench_status.get_value() is
    ↪   False
```

Demo Part II: Test Automation


CERN