Continuous Integration for the FairRoot Software Stack

Dennis Klein, **Christian Tacke**, Florian Uhlig, Mohammad Al-Turany
GSI Helmholtz Centre for Heavy Ion Research GmbH, Darmstadt, Germany

ACAT 2022 Conference, Bari, Italy

Abstract

In this work we present recent improvements of the Continuous Integration (CI, see beside) for the FairRoot software stack (see beside). We discuss relevant development workflows and how they were improved through automation. Furthermore, we present our infrastructure detailing its hardware and software design choices. The entire toolchain is composed of free and open source software. Finally, this work concludes with lessons learned from an operational as well as a user perspective and outlines ideas for future improvements.

Continuous Integration

Continuous Integration (CI) is a modern software engineering method to efficiently assure software quality. The underlying idea is to continuously rebase (pending) feature branches onto the latest version of the software repository and re-validate them by a set of checks in an automated fashion. This approach allows to naturally and efficiently resolve the merging of concurrent development activities (by one or multiple authors).

The (partial) automation of the software release process (e.g. generation of release assets, documentation, extended testing) including the automated deployment is sometimes described by the term Continuous Deployment (CD). However, the term CI is often understood to subsume CD as well. In this work we adopt this more general definition of CI.

The FairRoot Software Stack

The FairRoot[1] software stack is a toolset for the simulation, reconstruction, and analysis of high energy particle physics experiments, currently used i.e. at FAIR/GSI, and CERN.

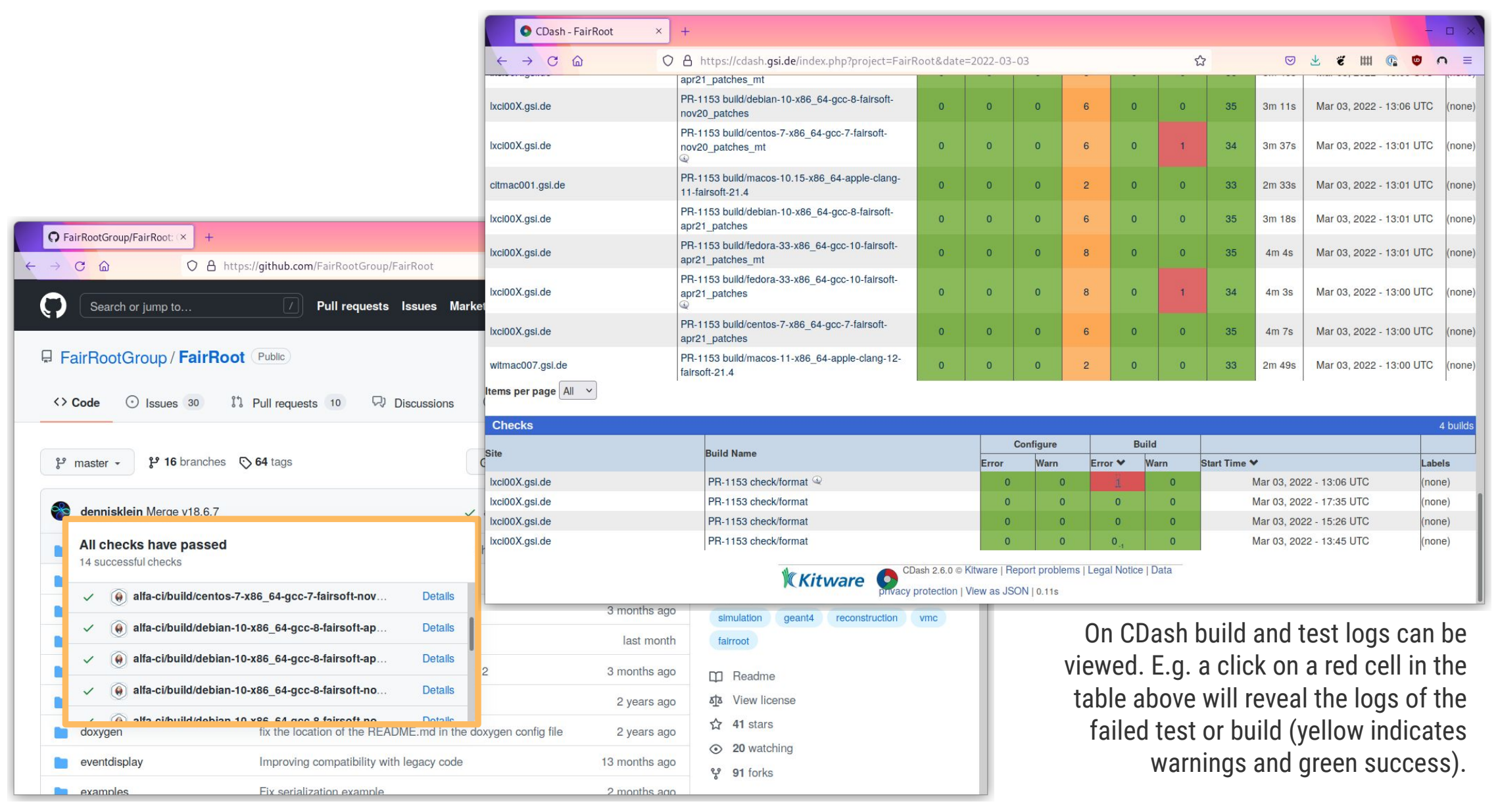
CbmRoot https://fair-center.eu/for-users/experiments/cbm.html
PandaRoot https://panda.gsi.de/
R3BRoot https://www.gsi.de/r3b
AliceO2 http://alice-o2.web.cern.ch/
FairSHiP http://ship.web.cern.ch/ship/

In the FairSoft[2] source distribution the most important software dependencies needed to run a FairRoot-based experiment software framework are released for Linux and macOS while further "system dependencies" are taken from the Linux distribution and the Homebrew project[3] on macOS. Notable packages in a FairSoft release are

- ROOT (https://root.cern/),
- Boost (https://www.boost.org/),
- Geant3 (https://github.com/FairRootGroup/geant3),
- Geant4 (https://geant4.web.cern.ch/),
- VMC (https://github.com/vmc-project/vmc),
- HEPMC (http://hepmc.web.cern.ch/),
- Pythia6 (https://github.com/alisw/pythia6),
- Pythia8 (https://pythia.org/),
- Vc (https://github.com/VcDevel/Vc),
- VGM (https://github.com/vmc-project/vgm),
- CLHEP (http://proj-clhep.web.cern.ch/),
- DDS (http://dds.gsi.de/), and
- FairMQ (https://github.com/FairRootGroup/FairMQ).

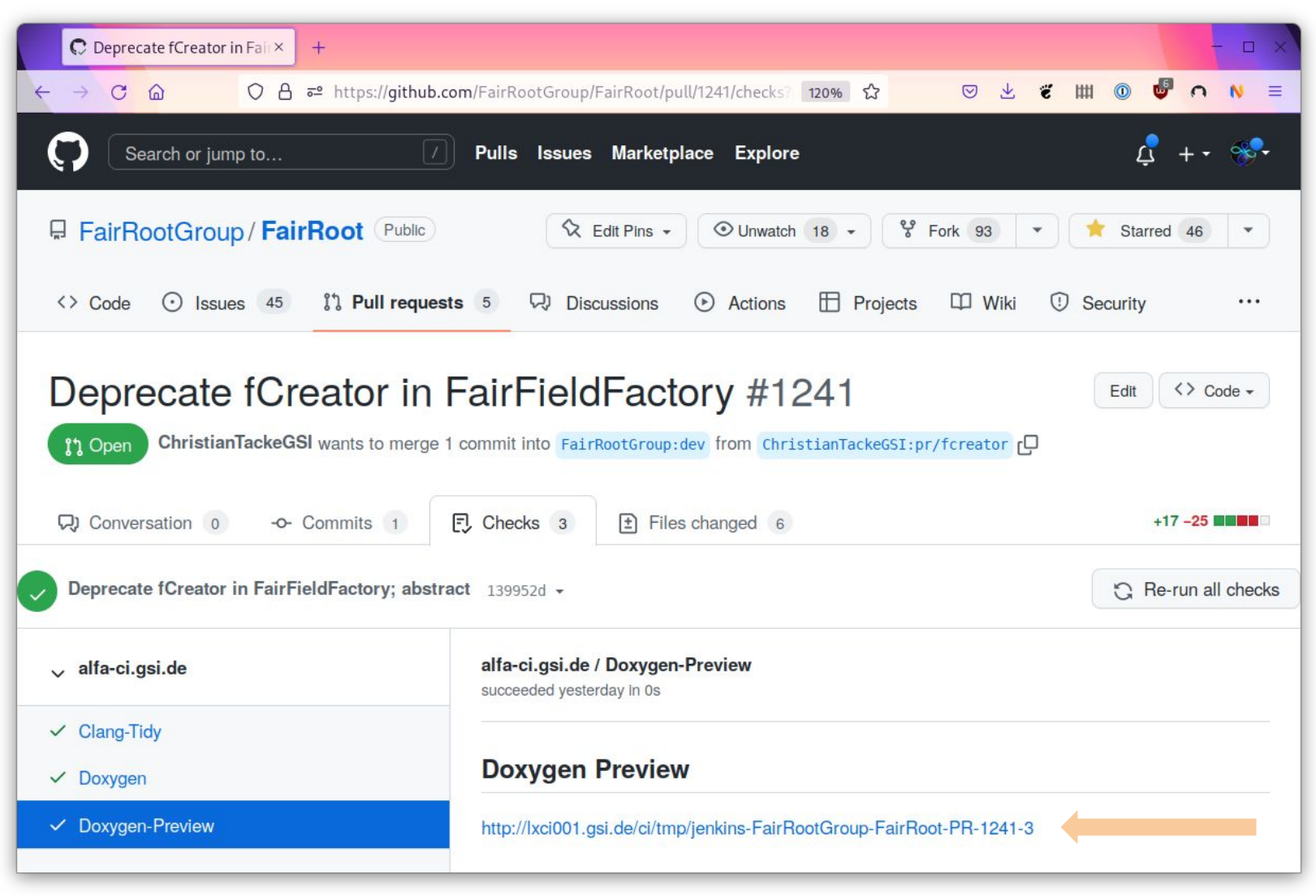
Saving Energy (Ongoing work)

- Typically, the CI build farm has a low overall utilization with long idle periods and short peak loads
- Deployed an energy monitor
- Implemented the Slurm Power Save Suspend and Resume hooks to shut off idle worker nodes and restart them on-demand (via BMC signals)
- No long term data available yet, evaluation in the proceedings paper

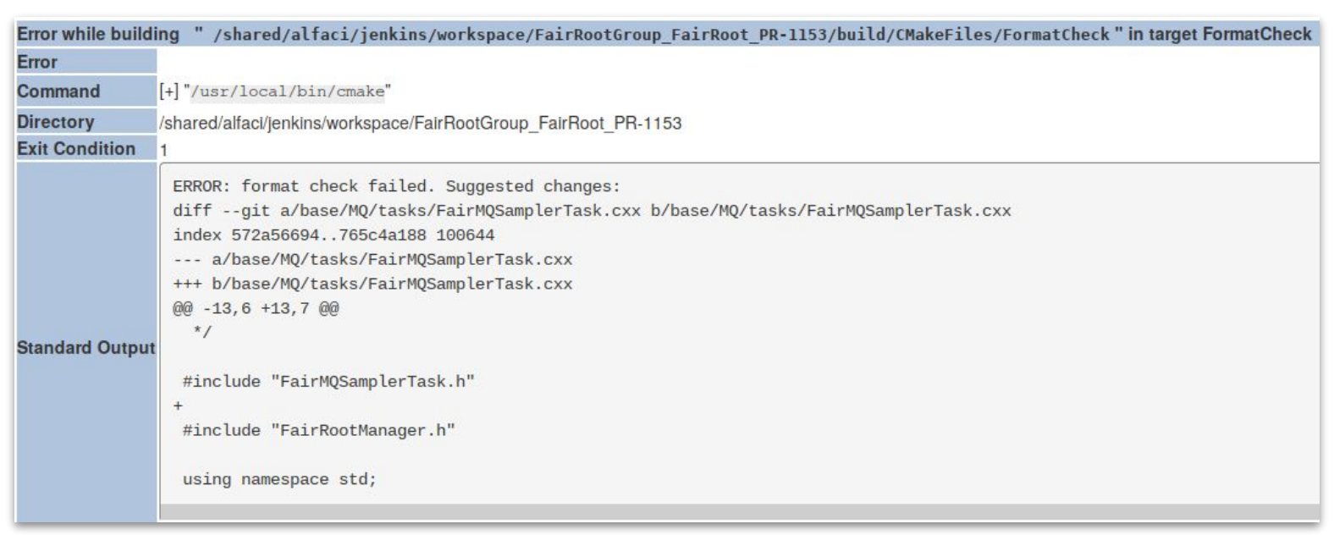


Developers can discover performed checks directly integrated in the Repository User Interface (here github.com)

On CDash build and test logs can be viewed. E.g. a click on a red cell in the table above will reveal the logs of the failed test or build (yellow indicates warnings and green success).



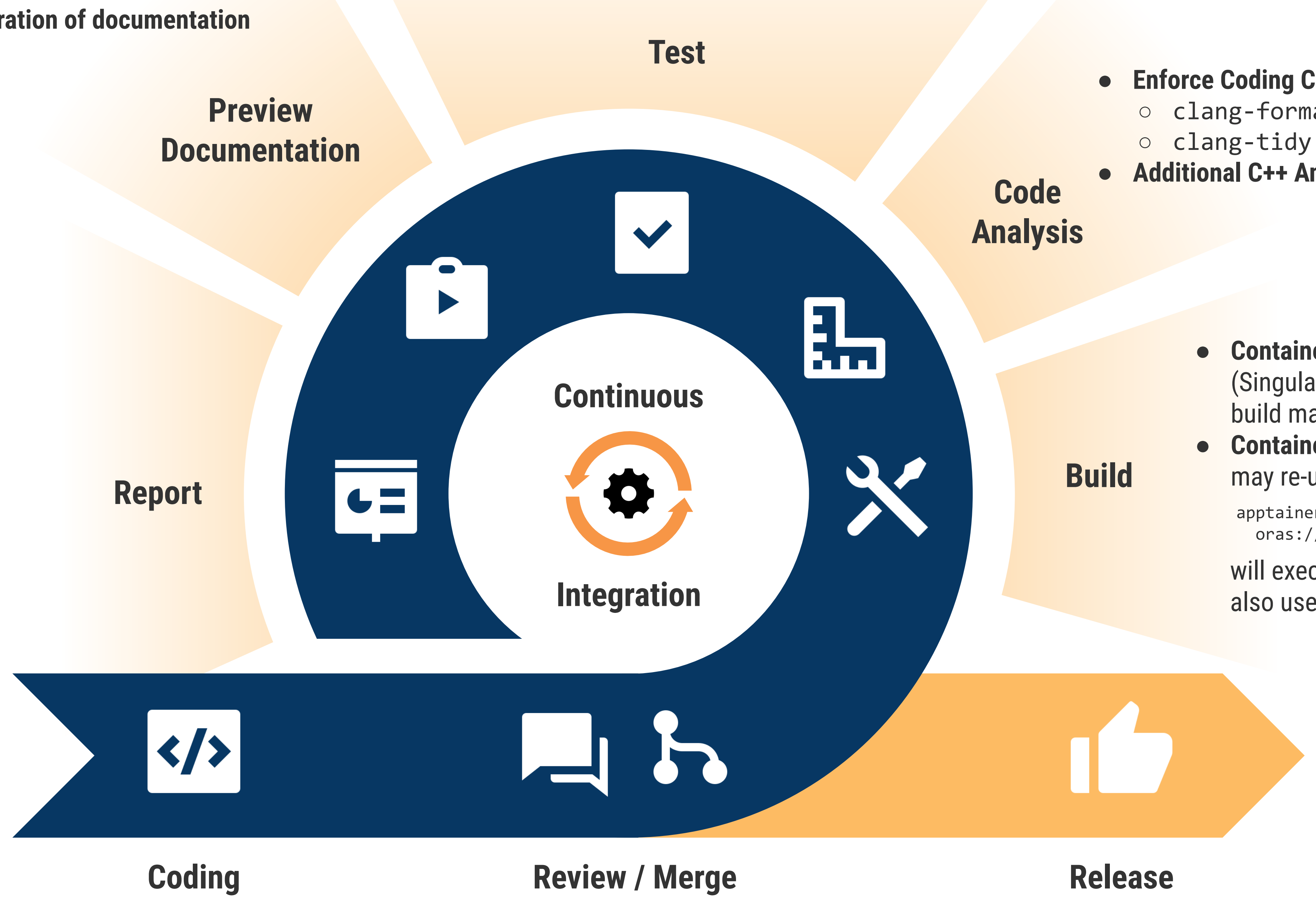
This example shows how a link to the doxygen based documentation preview in a FairRoot pull request is provided to the reviewers/developers.

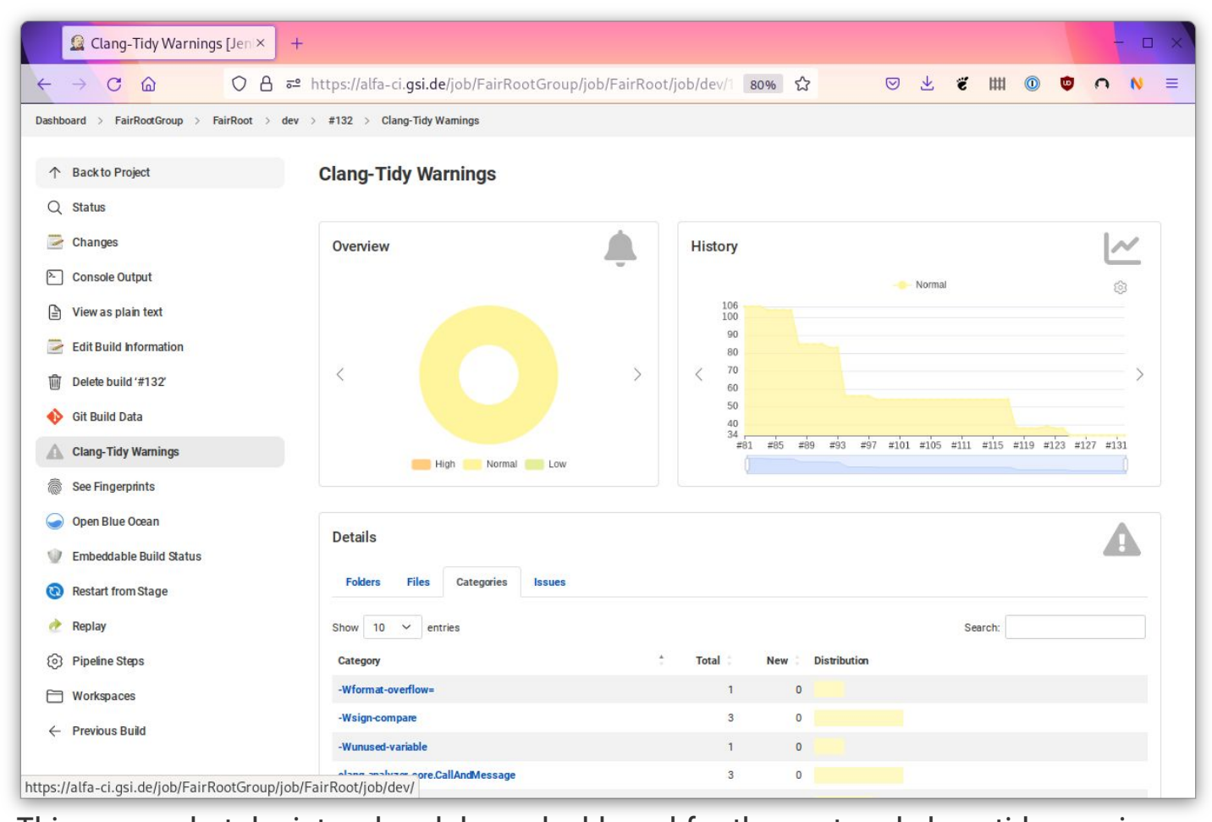


```
Error while building " /shared/alfaci/jenkins/workspace/FairRootGroup_FairRoot_PR-1153/build/CMakeFiles/FormatCheck " in target FormatCheck
Error
Command          [+] "/usr/local/bin/cmake"
Directory        /shared/alfaci/jenkins/workspace/FairRootGroup_FairRoot_PR-1153
Exit Condition   1
Standard Output  ERROR: format check failed. Suggested changes:
                 diff --git a/base/MQ/tasks/FairMQSamplerTask.cxx b/base/MQ/tasks/FairMQSamplerTask.cxx
                 index 572a56694..765c4a188 100644
                 --- a/base/MQ/tasks/FairMQSamplerTask.cxx
                 +++ b/base/MQ/tasks/FairMQSamplerTask.cxx
                 @@ -13,6 +13,7 @@
                   */

                  #include "FairMQSamplerTask.h"
                 +
                  #include "FairRootManager.h"

                  using namespace std;
```

The screenshot shows a failed clang-format check viewed via CDash.

- **Run Extensive Test Suites** as part of the central CI cycle
- **Automated generation of documentation**
- **Advanced build log analysis** with CDash[5] and the Warnings Next Generation Jenkins[6] Plugin
- **Enforce Coding Conventions** via
  - `clang-format`
  - `clang-tidy`
- **Additional C++ Analysis** via Address, Thread, UB, Leak Sanitizers



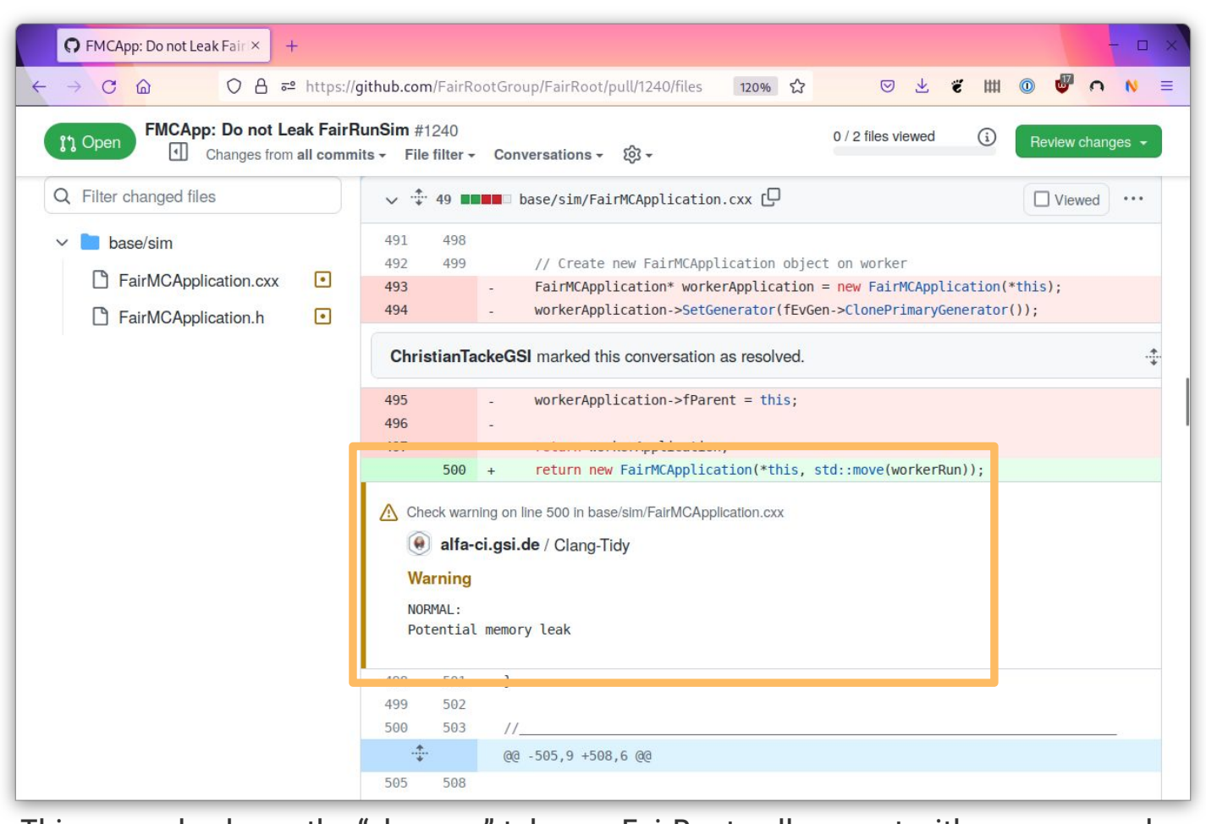


This screenshot depicts a breakdown dashboard for the captured clang-tidy warnings for the FairRoot dev branch realized with the Warnings Next Generation Jenkins Plugin

- **Containerized Builds** (on Linux) with Apptainer[9] (Singularity) to enable heterogeneous and large build matrices
- **Container Definitions public** so the developer may re-use the build environment locally, e.g.

```
apptainer exec \
  oras://ghcr.io/fairrootgroup/fairmq-dev/fedora-35-sif:latest <cmd>
```

will execute the given `<cmd>` in the same environment that is also used by FairMQ's CI jobs based on the Fedora 35 image.



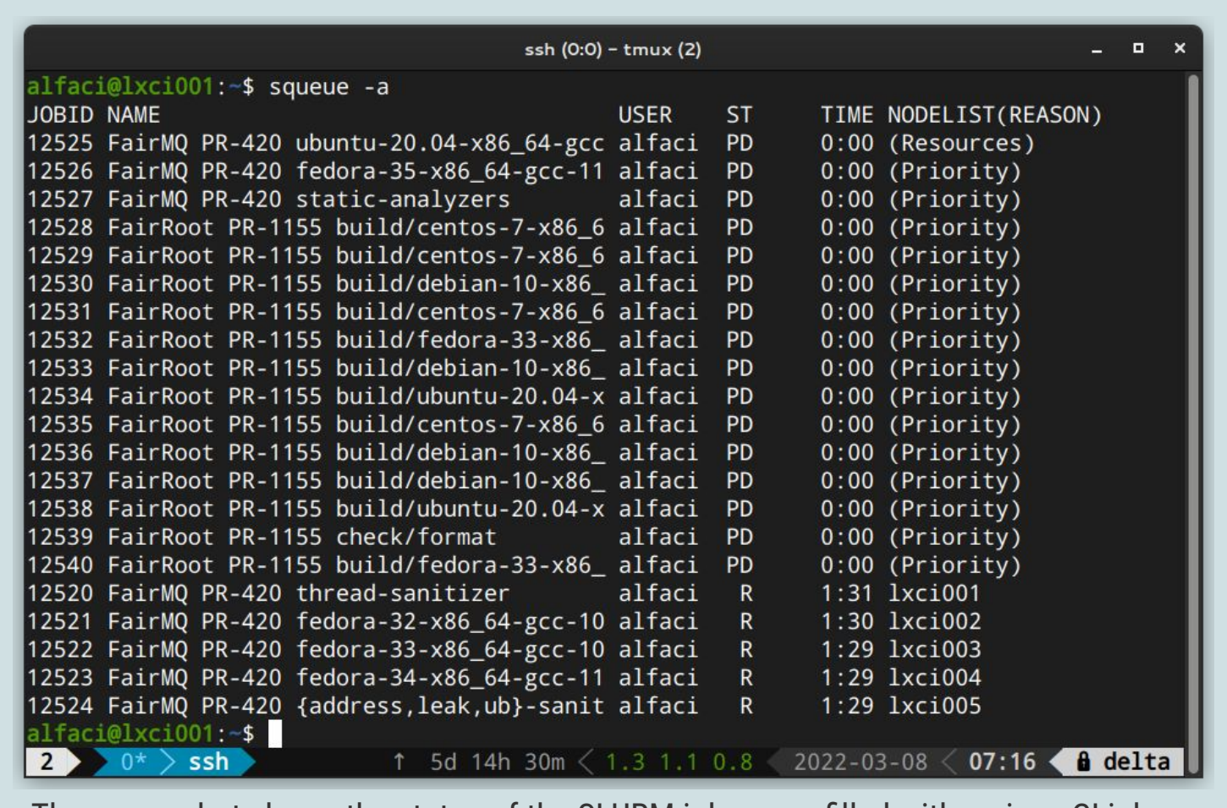
This example shows the "changes" tab on a FairRoot pull request with a source code annotation about a clang-tidy warning directly in the github.com user interface (realized with the Warnings Next Generation Jenkins Plugin).

Build Farm

- 5 Linux hosts (20-28 CPU cores, 128-256GB RAM each) + 3 Mac minis (2x `x86_64`, 1x `aarch64`) for **public, untrusted** CI jobs
- 3 Linux hosts (20-28 CPU cores, 128-256GB RAM each) for **private, trusted** CI (mainly CD) jobs
- All hosts have direct attached SSD storage suitable for **cross job build caches**
- Job run times range from a few minutes to multiple hours < 6h depending on repo and change.

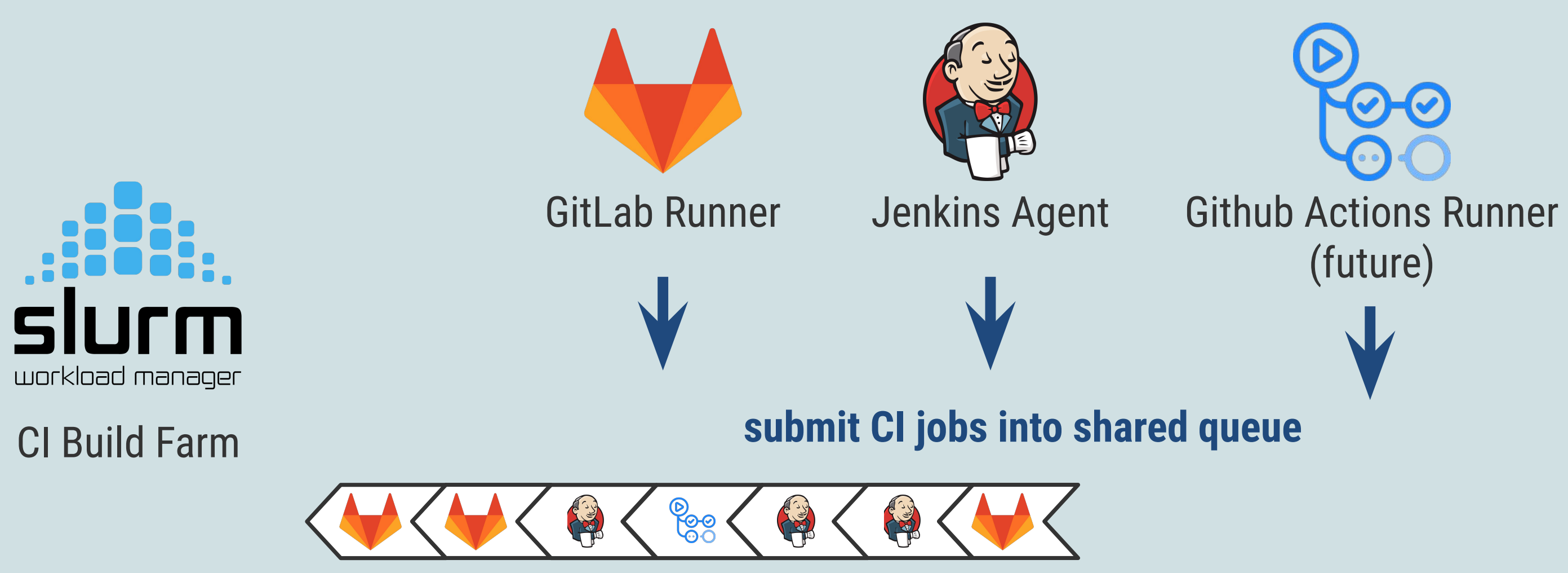
Infrastructure

The relevant source code repositories are hosted in a heterogeneous environment, some are on github.com, some on various GitLab[6] instances. GitLab natively comes with a CI service. As of ~2021 only, github.com started to offer a CI service. Because we started to use the CI method before, we operate a Jenkins CI service as well. In anticipation of the still quickly changing CI service landscape, we aim to keep a **weak** dependency on those CI services only. In addition, we want to support multiple instances in parallel and still share our build farm. To achieve this, CI jobs are scheduled onto the build farm via a shared queue - realized with the Slurm[4] resource manager.



```
ssh (0:0) - tmux (2)
alfaci@lxci001:~$ squeue -a
JOBID NAME                                USER   ST     TIME NODELIST(REASON)
12525 FairMQ PR-420 ubuntu-20.04-x86_64-gcc alfaci PD     0:00 (Resources)
12526 FairMQ PR-420 fedora-35-x86_64-gcc-11 alfaci PD     0:00 (Priority)
12527 FairMQ PR-420 static-analyzers        alfaci PD     0:00 (Priority)
12528 FairRoot PR-1155 build/centos-7-x86_6 alfaci PD     0:00 (Priority)
12529 FairRoot PR-1155 build/centos-7-x86_6 alfaci PD     0:00 (Priority)
12530 FairRoot PR-1155 build/debian-10-x86_ alfaci PD     0:00 (Priority)
12531 FairRoot PR-1155 build/centos-7-x86_6 alfaci PD     0:00 (Priority)
12532 FairRoot PR-1155 build/fedora-33-x86_ alfaci PD     0:00 (Priority)
12533 FairRoot PR-1155 build/debian-10-x86_ alfaci PD     0:00 (Priority)
12534 FairRoot PR-1155 build/ubuntu-20.04-x alfaci PD     0:00 (Priority)
12535 FairRoot PR-1155 build/centos-7-x86_6 alfaci PD     0:00 (Priority)
12536 FairRoot PR-1155 build/debian-10-x86_ alfaci PD     0:00 (Priority)
12537 FairRoot PR-1155 build/debian-10-x86_ alfaci PD     0:00 (Priority)
12538 FairRoot PR-1155 build/ubuntu-20.04-x alfaci PD     0:00 (Priority)
12539 FairRoot PR-1155 check/format         alfaci PD     0:00 (Priority)
12540 FairRoot PR-1155 build/fedora-33-x86_ alfaci PD     0:00 (Priority)
12520 FairMQ PR-420 thread-sanitizer        alfaci  R     1:31 lxci001
12521 FairMQ PR-420 fedora-32-x86_64-gcc-10 alfaci  R     1:30 lxci002
12522 FairMQ PR-420 fedora-33-x86_64-gcc-10 alfaci  R     1:29 lxci003
12523 FairMQ PR-420 fedora-34-x86_64-gcc-11 alfaci  R     1:29 lxci004
12524 FairMQ PR-420 {address,leak,ub}-sanit alfaci  R     1:29 lxci005
alfaci@lxci001:~$
 2  0* ssh          ↑ 5d 14h 30m  1.3 1.1 0.8  2022-03-08  07:16  delta
```

The screenshot shows the status of the SLURM job queue filled with various CI jobs (R for Running, PD for Pending)


slurm workload manager
CI Build Farm
GitLab Runner
Jenkins Agent
Github Actions Runner (future)
submit CI jobs into shared queue

Lessons Learned

- Long Running CI jobs are acceptable
- Design CI checks cleanly from the bottom up and avoid unnecessary dependencies on the CI system

Ongoing and Future Work

- Publish pre-built CI container images on a registry and/or CVMFS[8], so the developer can re-create a CI environment for a local interactive debugging session.
- Deploy and Operate a Github Actions Runner and explore if Jenkins can be phased out

References

[1] FairRoot - https://fairroot.gsi.de/
[2] FairSoft - https://github.com/FairRootGroup/FairSoft
[3] Homebrew - https://brew.sh/
[4] Slurm - https://slurm.schedmd.com/
[5] CDash - https://www.cdash.org/
[6] GitLab - https://about.gitlab.com/
[7] Jenkins - https://www.jenkins.io/
[8] CVMFS - https://cernvm.cern.ch/fs/
[9] Apptainer - https://apptainer.org/

All URLs have been last visited on 19th October 2022.