

```
..._mod.use_z = False  
..._operation == "MIRROR_Y":  
...mirror_mod.use_x = False  
...mirror_mod.use_y = True  
...mirror_mod.use_z = False  
..._operation == "MIRROR_Z":  
...mirror_mod.use_x = False  
...mirror_mod.use_y = False  
...mirror_mod.use_z = True
```

```
...selection at the end -add  
...mirror_ob.select= 1  
...modifier_ob.select=1  
...context.scene.objects.active  
...("Selected" + str(modifier_ob.name))  
...mirror_ob.select = 0  
...= bpy.context.selected_objects  
...data.objects[one.name].select  
...print("please select exactly one mirror")
```

--- OPERATOR CLASSES ---

DAQ DEVELOPMENT : PROCESS, INFRASTRUCTURE & TOOLS

A L E S S A N D R O T H E A



Requirements

System design

Team



Development model

Planning

Cycle



Code and tools

DAQ software stack

Code management

Build system & distribution

Firmware



Outlook

OUTLINE

SYSTEM COMPOSITION

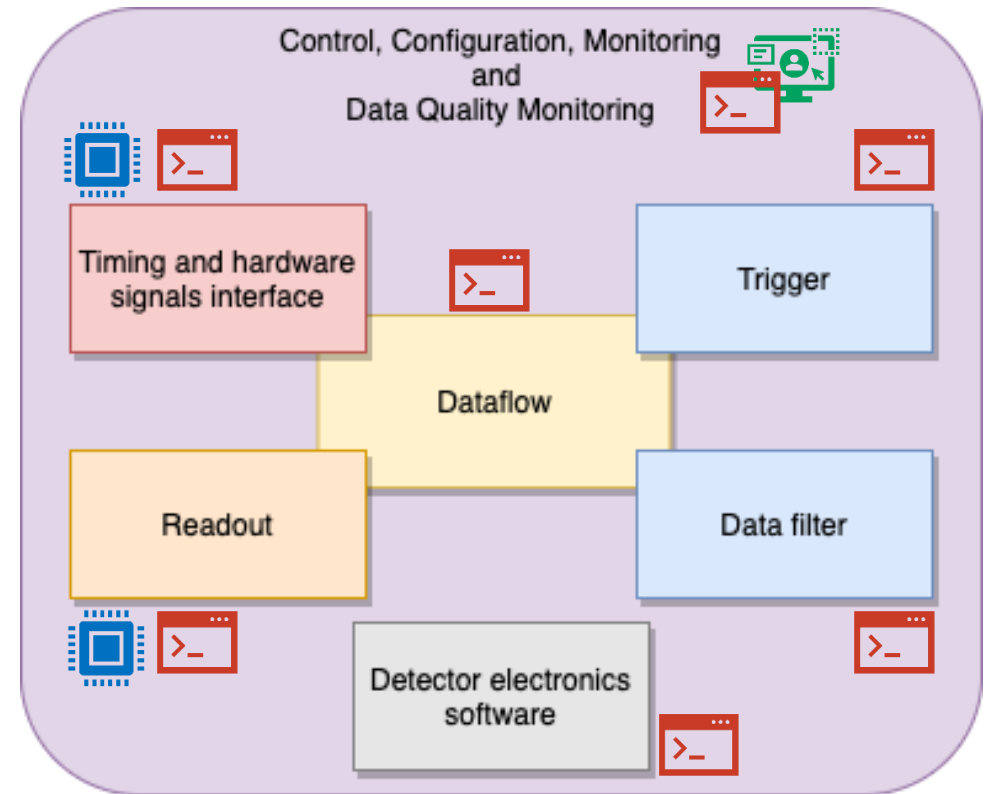
- Software-based system with some firmware elements

- **Firmware**

- Timing and readout

- **Software**

- Custom DAQ framework for data readout/processing/storage
 - + detector electronics control interface
 - Mix of custom & off-the-shelf solutions in CCM, DQM



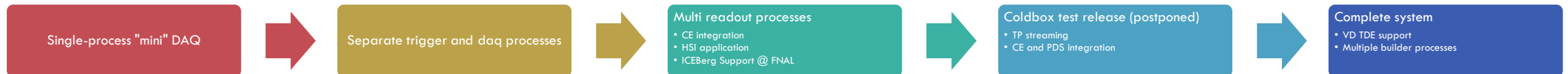
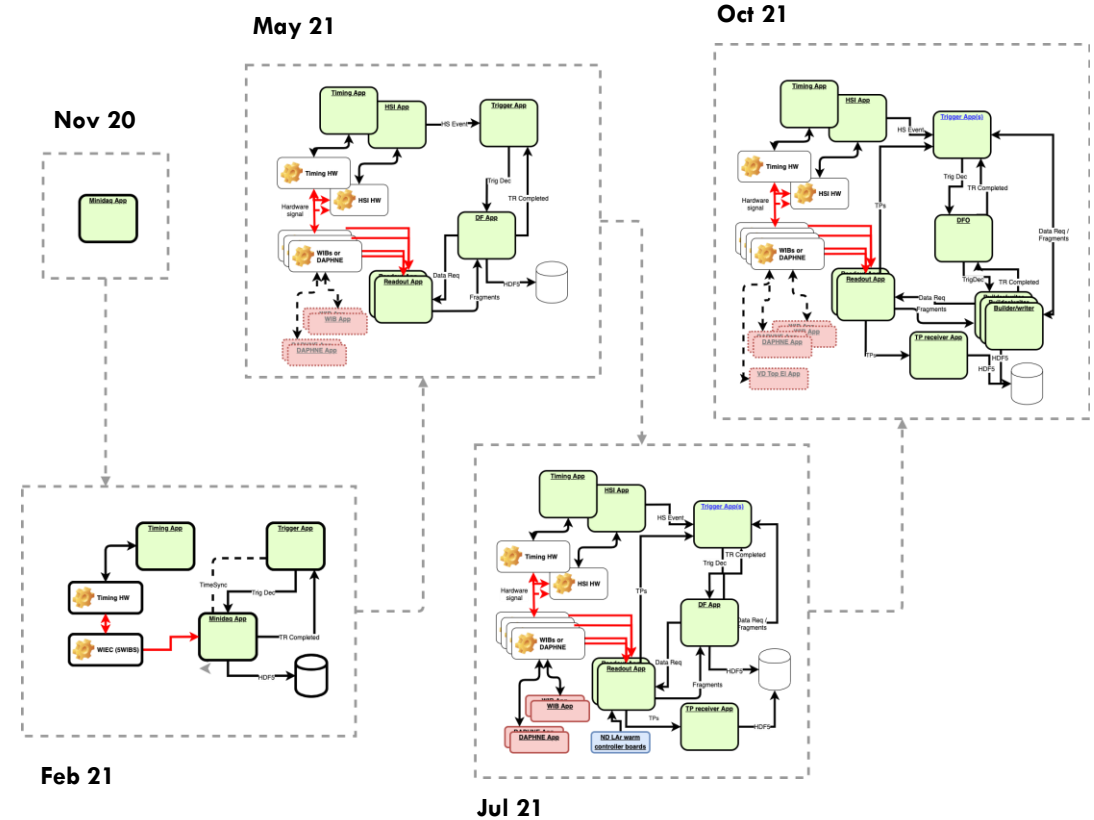
DAQ TEAM

- International
 - Spread over several TZ
 - Mixture of different expertise
 - No-chances for in-person interactions since February 2020
- Small
 - Developments are prioritized and carried out somewhat sequentially



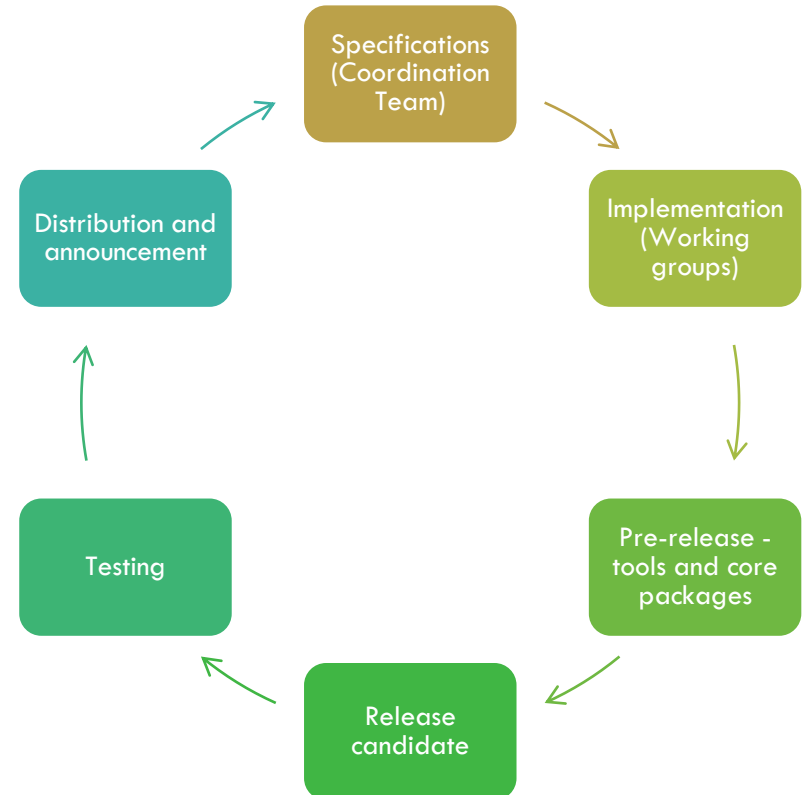
DEVELOPMENT PLAN

- Release-based, driven by
 - **Short-term DAQ & DUNE milestones:** complete system for PD-HD-II by Q1 2022; Support for coldbox run(s) in 2021
 - **Team distribution:** keep engagement high with short release cycles (by-monthly) & release goals aligned with project milestones.
- Each release characterized by a reference system layout
- Hybrid incremental/iterative development model
 - new elements added and existing interfaces updated at each release
- Plan aligned with Vertical Drift milestones



DEVELOPMENT CYCLE

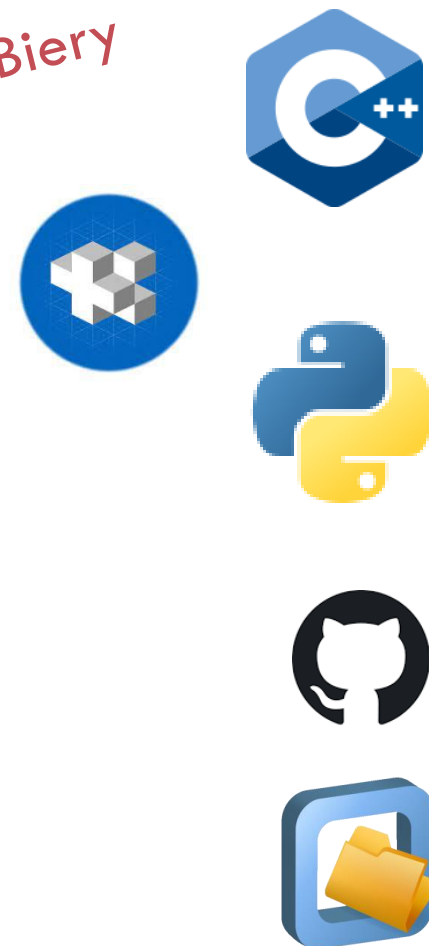
- Development cycles are aligned with releases
- Release goals are discussed/confirmed in the DAQ coordination team, and specifications agreed upon
- Implementation carried out within the consortium WGs
- 2 weeks ahead of the release date: pre-release distribution – semi-final version of tools, external dependencies and core packages
- On release date: final build & 24h “cooldown” period (11th hour bugs and patches)
- Distribution and announcement to the consortium



DAQ APPLICATION SOFTWARE

See talk by K.Biery

- Code organised in packages
 - “package” basic versionable entity
- Programming languages:
 - C++
 - jsonnet for data-structure definition
 - Python for configuration and scripting
- Code hosted on GitHub
- Deployed on CVMFS as UPS products
- 25 DAQ packages in the latest release (**dunedaq-2.6**)



DAQ SOFTWARE STACK

OS: Centos7/Scientific Linux 7

- Old. Dictated by CERN/FNAL support, for compatibility with expected computing environment at the labs

Basic tools: gcc, python, cmake...

- Moved away from ancient system versions
- Selected versions available pre-compiled via FNAL-SciSoft

External libraries and tools: boost, highfive, felix,...

- Either pre-compiled by SciSoft or compiled and packaged by DAQ
 - Patches only allowed for compilation purposes
- External build scripts versioned and tagged as part of every release
- 33 externals in the last release (centrally provided)

DAQ – core components

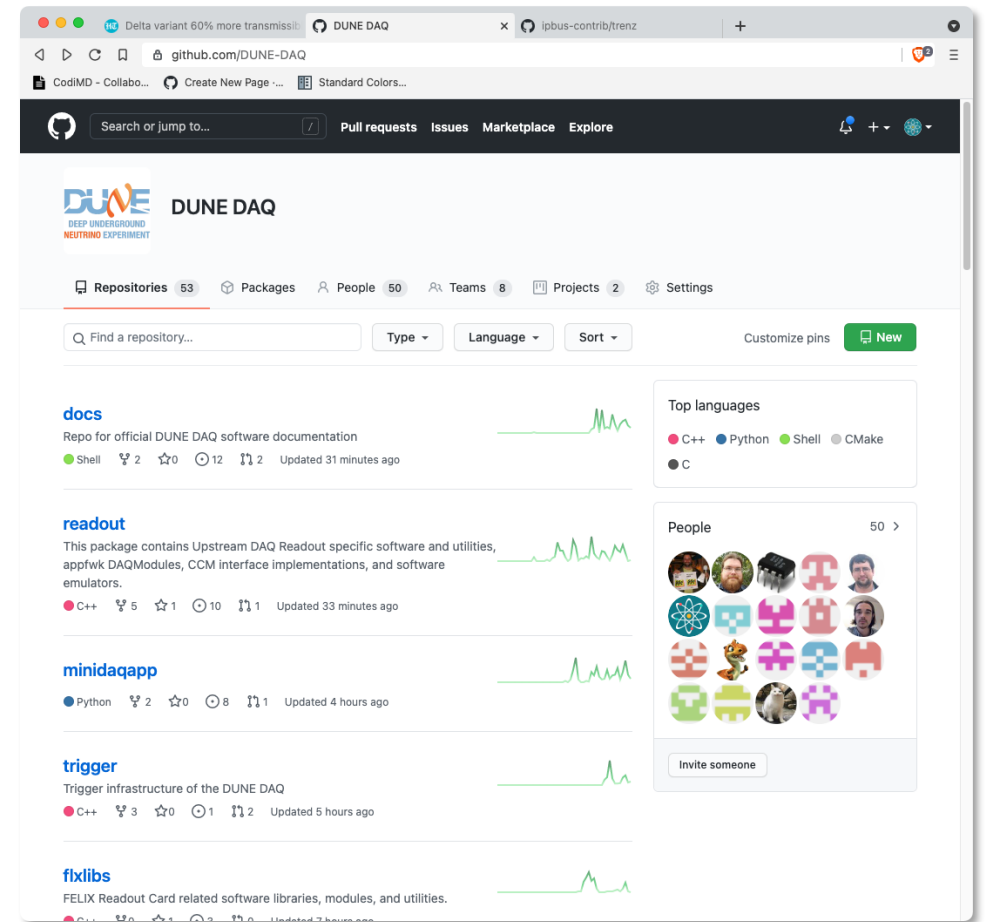
- i.e. packages required to build modules and applications
- Logging, error reporting, application framework

DAQ / Detector – application components

- i.e. packages implementing modules and applications
- Felix readout interface, trigger record builders, data writers, ...

DAQ CODE HOSTING

- GitHub: good balance between features and flexibility
 - Offers access control, task management, wikis and continuous integration
 - No need for FNAL / CERN accounts for new users
- DUNE-DAQ Organisation
 - 53 repositories, 50 users and growing



DAQ PACKAGE

- Application software elementary block
 - Versioned, each in a dedicated git repository
 - with a defined set of dependencies
 - Corresponds to a cmake package
- Enforces standard package structure
- Supported by cmake extension (**daq-cmake**).
 - Easily add components (library, plugins, etc..)
 - Generate code from schema files
 - Standardize build and installation targets



Example: flxlibs

BUILD SYSTEM

- Toolset to streamline the development of DUNE DAQ packages (**daq-buildtools**).
- Main functions
 - Create lightweight, self-contained C++ / python work-areas based on a DAQ release
 - Where one or more DAQ packages can be easily checked out and compiled
 - Guide the developer through the stages of the build (cmake project generation, build, installation, ...)
 - Run unit-tests
 - Perform code linting/formatting
 - Provide a runtime environment for testing newly compiled code



PACKAGING

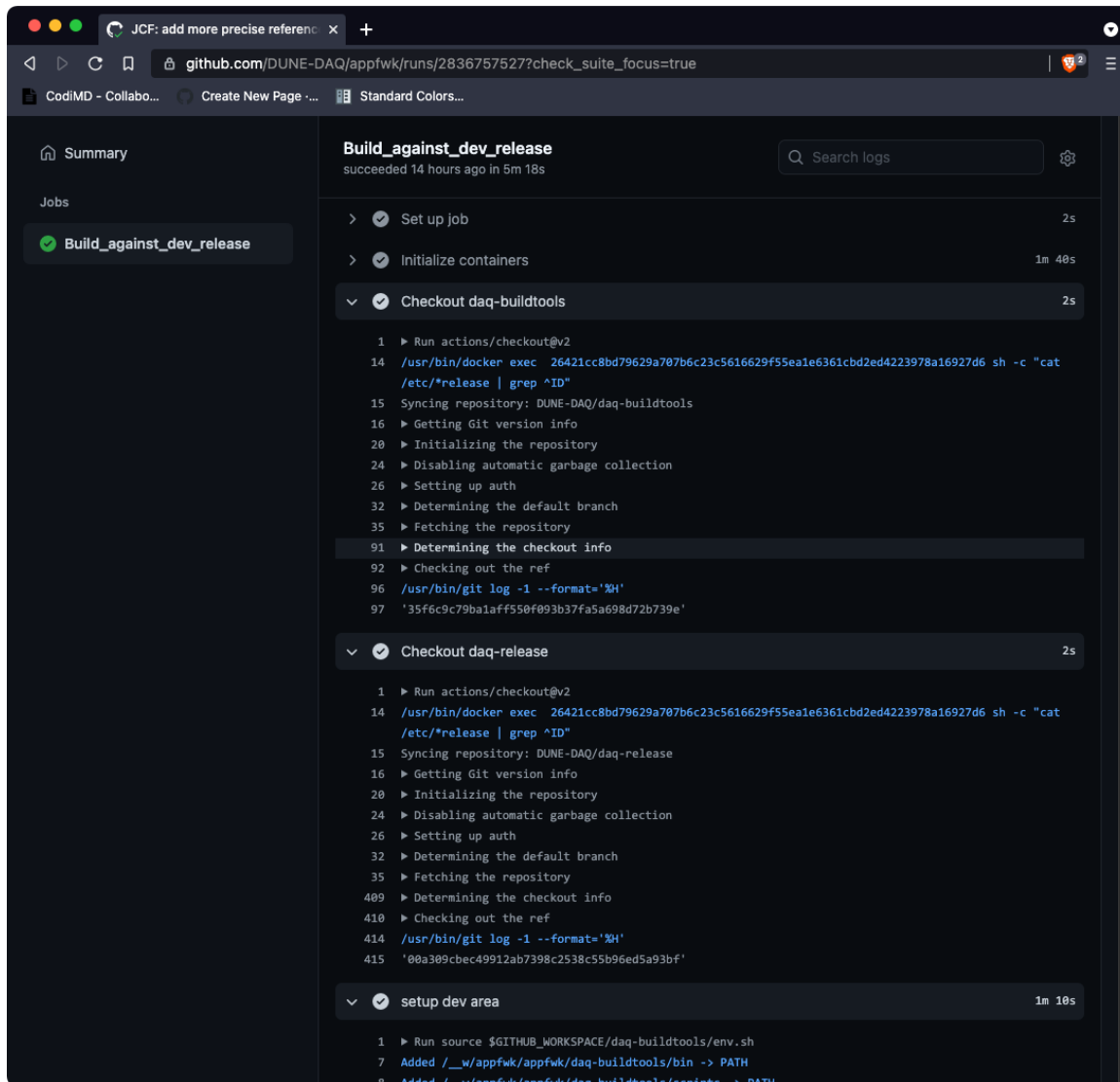
- DAQ Software distributed as UPS (Unix Product Support) products
 - Easily configurable to distribute additional elements together with header and libraries (e.g. schema files, python modules)
 - And to make them available via environment manipulation
 - Native support for concurrent installation of multiple versions of each product on the same machine
 - Large pool of packages and tools in UPS format made available by FNAL SciSoft pre-compiled
- Ongoing: evaluating SPACK (<https://spack.io/>) as a potential, more modern alternative





DISTRIBUTION

- DAQ Software primary distribution channel: CVMFS (CernVM-File System) service
 - Tarballs and docker images also available
- Repositories hosted at FNAL
 - dunedaq.opensciencegrid.org, production:
DAQ releases, ups products pools, build tools
 - dunedaq-develop.opensciencegrid.org, development:
nightly builds, experimental ups products
- Updates available world-wide in $O(1h)$ to any CVMFS client
 - no user intervention required



The screenshot shows a GitHub Actions workflow run for the job 'Build_against_dev_release'. The workflow is successful and consists of several steps:

- Set up job** (2s)
- Initialize containers** (1m 40s)
- Checkout daq-buildtools** (2s)
 - Run actions/checkout@v2
 - 14 /usr/bin/docker exec 26421cc8bd79629a707b6c23c5616629f55ea1e6361cbd2ed4223978a16927d6 sh -c "cat /etc/*release | grep ^ID"
 - 15 Syncing repository: DUNE-DAQ/daq-buildtools
 - 16 ▶ Getting Git version info
 - 20 ▶ Initializing the repository
 - 24 ▶ Disabling automatic garbage collection
 - 26 ▶ Setting up auth
 - 32 ▶ Determining the default branch
 - 35 ▶ Fetching the repository
 - 91 ▶ Determining the checkout info
 - 92 ▶ Checking out the ref
 - 96 /usr/bin/git log -1 --format='%H'
 - 97 '35f6c9c79ba1aff550f093b37fa5a698d72b739e'
- Checkout daq-release** (2s)
 - 1 ▶ Run actions/checkout@v2
 - 14 /usr/bin/docker exec 26421cc8bd79629a707b6c23c5616629f55ea1e6361cbd2ed4223978a16927d6 sh -c "cat /etc/*release | grep ^ID"
 - 15 Syncing repository: DUNE-DAQ/daq-release
 - 16 ▶ Getting Git version info
 - 20 ▶ Initializing the repository
 - 24 ▶ Disabling automatic garbage collection
 - 26 ▶ Setting up auth
 - 32 ▶ Determining the default branch
 - 35 ▶ Fetching the repository
 - 409 ▶ Determining the checkout info
 - 410 ▶ Checking out the ref
 - 414 /usr/bin/git log -1 --format='%H'
 - 415 '00a309c49912ab7398c2538c55b96ed5a93bf'
- setup dev area** (1m 10s)
 - 1 ▶ Run source \$GITHUB_WORKSPACE/daq-buildtools/env.sh
 - 7 Added /__w/appfwk/appfwk/daq-buildtools/bin -> PATH
 - 8 Added /__w/appfwk/appfwk/daq-buildtools/scripts -> PATH

CONTINUOUS INTEGRATION AND NIGHTLY BUILDS

- Two tier CI setup, based GitHub actions
 - Full build of all packages (develop branch) every night
 - Single repo: build against the last successful nightly at every commit

DOCUMENTATION

- Online documentation hosted on [readthedocs.io](https://dune-daq-sw.readthedocs.io)
- Generated from the content of docs/ in each DAQ package
- Documentation sources in Markdown
- Native support for multiple concurrent versions

The screenshot shows the DUNE DAQ Software Documentation Home page. The navigation menu includes: [readout - Readout software and utilities](#), [Building](#), [Examples](#), [Enabling the fake TP source](#), [A Deeper Look Into Readout: Functional Elements](#), and [Looking into the directories](#).

The main content area displays instructions for getting "tp_frames.bin" TP data:

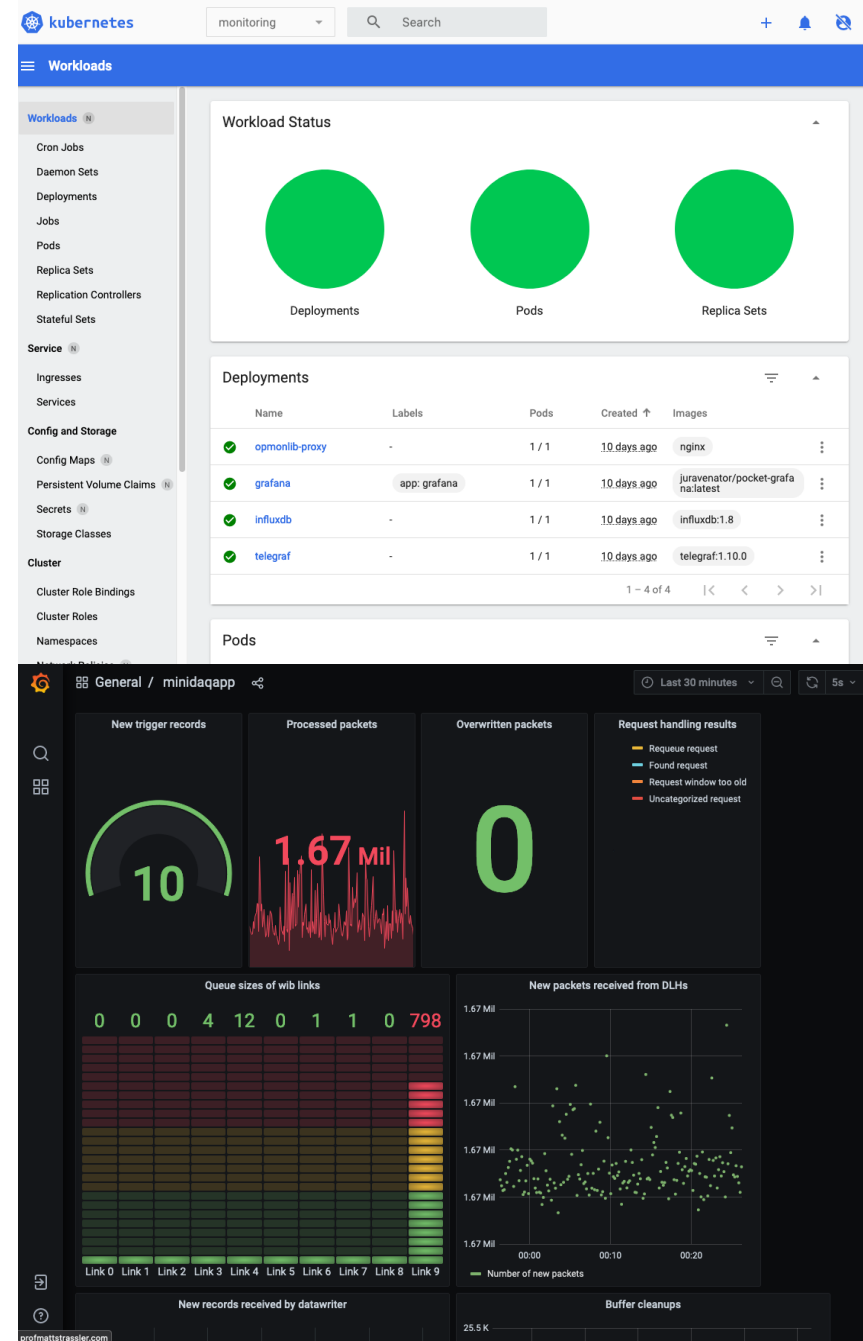
```
curl https://cernbox.cern.ch/index.php/s/686nd0gupT1i2RW/download -o /tmp/tp_frames.bin
```

Below the code is a section titled "A Deeper Look Into Readout: Functional Elements Data-flow Diagram (DFD)". The text states: "The functional elements are seen on the following DFD. Color codes indicate the ownership (or responsibility) of the DAQ subsystems: Dataflow (=blue) and Upstream DAQ (=red)."

The DFD is divided into four domains: TP Handling Domain, Raw Streaming Domain, Requested Data Domain, and SNB Data Domain. The Front-End Domain is at the bottom. The diagram shows the flow of data (RAW) and requests (Data Requests, Requested Data, SNB Request) between various components like Stream TPs, TP Buffer, Format TPs, Generate TPs, Stream RAW data, RAW debug/calib data streams, Raw Formatting (calib, channel), Raw Processing, Receive from Frontend, Raw-data Buffer, Request Handling, Sending Data, Stream to store, SNB store handling, and SNB Data Store. A legend at the bottom right indicates "v: latest".

SERVICES DISTRIBUTION

- Prototyping the use of Kubernetes + Docker for distributing CCM/DQM services to small installations and testbeds
- Promising solution for delivering complex configurations with small effort
- Prototype under evaluation (**pocket**) includes
 - Monitoring: InfluxDB+Grafana
 - Logging: ELK stack



DAQ FIRMWARE

Code organisation similar to software

- Firmware “package” basic versionable entity
- 1 package 1 git repository

with different granularity

- Just 6 packages between timing and readout
- Of larger complexity – multiple projects per package

Languages

- VHDL
- Tcl for scripting

Toolsets

- Xilinx Vivado & MentorGraphics Questasim

Code hosted on GitLab at CERN

BUILD SYSTEM, CI AND DISTRIBUTION

- **Build system**

- IPBbus Builder – Initially developed in CMS
- Defines standard package structure, streamlines the typical simulation and sythesis workflow

- **Continuous integration**

- Simulation and builds triggered at every commit, full builds for Merge requests

- **Automatic distribution**

- MR and tags build products automatically uploaded to EOS, including address tables and build logs

GitLab Merge branch 'newbold/hsi_dev' into 'master'

Newbold/hsi_dev
See merge request 128

45 jobs for v5.4.0 in 19 minutes and 59 seconds (queued for 3 seconds)

latest

212bbdf0

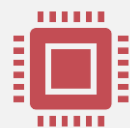
No related merge requests found.

Pipeline Needs Jobs 45 Tests 0

Setup	Checks	Builds	Publish
setup_buill...	check:boreas...	build:boreas_...	publish:tag
	check:boreas...	build:boreas_...	
	check:bore...	build:bore...	
	check:boreas...	build:boreas_...	
	check:chronos...	build:chronos...	
	check:chronos...	build:chronos...	
	check:crt_pc...	build:crt_pc0...	
	check:crt_pc...	build:crt_pc0...	
	check:endpoi...	build:endpoi...	
	check:endpoi...	build:endpoi...	
	check:fanout...	build:fanout_...	
	check:fanout...	build:fanout_...	
	check:fanout...	build:fanout_...	
	check:fanout...	build:fanout_...	
	check:ourobo...	build:ourobor...	
	check:ourobo...	build:ourobor...	
	check:ourobo...	build:ourobor...	
	check:ourobo...	build:ourobor...	
	check:overlor...	build:overlor...	
	check:overlor...	build:overlor...	
	check:overlor...	build:overlor...	
	check:overlor...	build:overlor...	
	check:overlor...	build:overlor...	
	check:overlor...	build:overlor...	
	check:overlor...	tarball_bare_...	
	check:overlor...	tarball_repo	

Timing system firmware CI

SUMMARY



A solid development process has been devised by DAQ consortium in the last year

for both software and firmware



A complete set of tools has been created to support the process from code checkout to products distribution



A release-based development plan is in place

aligned with HD and VD project milestones



R&D continues to improve tool and coverage

e.g. SPACK, Kubernetes, etc...