Platform-independent integration of DAQ software using docker containers

7th Beam Telescopes and Test Beams Workshop
CERN
January 15th-18th 2019

Jordi Duarte-Campderrós

EXCELENCIA MARÍA DE MAEZTU

IFCA Instituto de Física de Cantabria

CERN

CSIC Consejo Superior de Investigaciones Científicas

UC Universidad de Cantabria

Outline

- Why containers? … What's a container?
- Using dockers with DAQ software
- A real case example: EUDAQ+BDAQ53
  - The last year's characterization suite for the next pixel modules at the IT of CMS Phase-II test beams

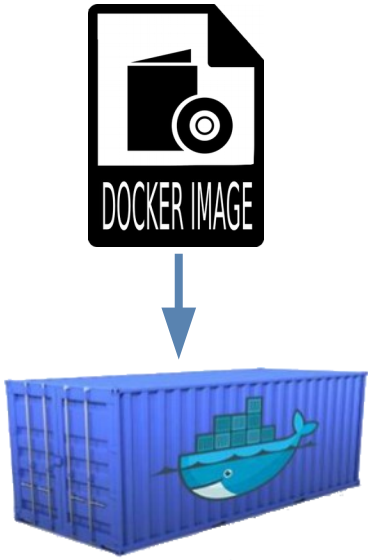
What's a container?

From docker web page: https://www.docker.com/resources/what-container

What is a Container

A standardized unit of software

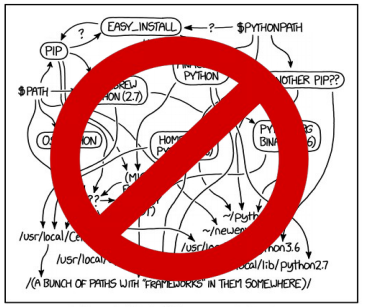
- Packages up code and **ALL** dependencies → **NO PLATFORM** specific



- Standalone and executable package of software
  - Includes everything needed to run an application: code, runtime, system tools, system libraries and settings
  - Small sizes → easy to ship, store and distribute

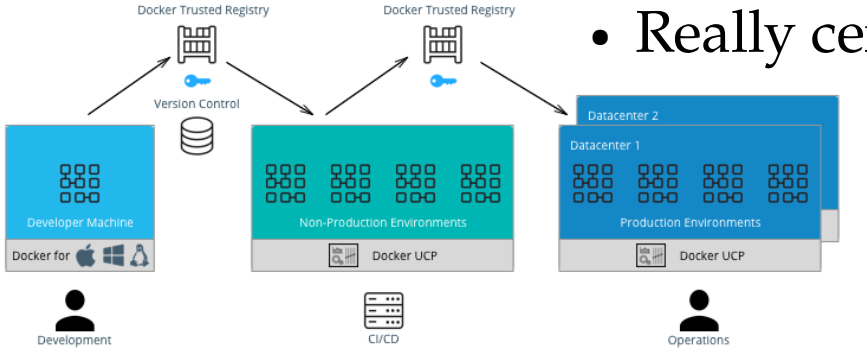
- Become a **container** at runtime (in docker, through a docker-engine)
  - Software isolated from environment (don't care **WHERE** is running)
  - Quick deployment, easy to go from DEV → PROD environments

Why containers?



- Minimized software installation
  - Free from dependency hell

- Multi-platform development and maintenance not needed
- Reproducibility → Same environment and code is used always: DEV ↔ PROD



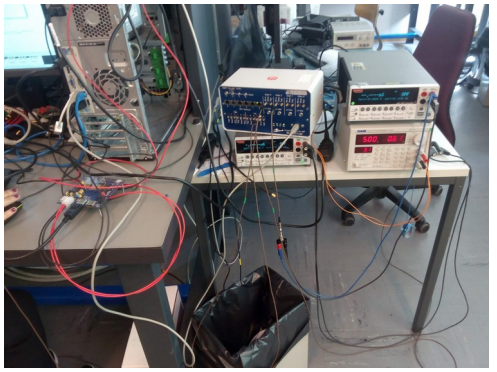
- Really centralized maintenance

Why containers (II)?

From docker "Why docker": https://www.docker.com/why-docker

Why dockers for DAQ software?

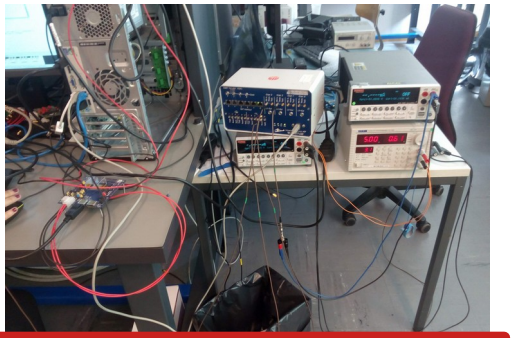
- DAQ point of view: two typical environments while developing new sensors for HEP experiments:
  - The **LAB** environment, where parts or components of the modules (readout chips, sensors,...) are being tested or developed (new) functionalities



  - The **TEST BEAM** environment, where all these new functionalities are tested under a beam of particles, the working conditions the sensors are suppose to be, in order to characterize them

Why dockers for DAQ software?

- DAQ point of view: two typical environments while developing new sensors for HEP experiments:
  - The **LAB** environment, where parts or components of the modules (readout chips, sensors,...) are being tested or developed (new) functionalities
    → the **DEV.** env.



CONTINUOUSLY CHANGING

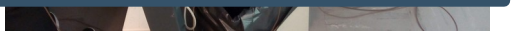
Why dockers for DAQ software?

- DAQ point of view: two typical environments while developing new sensors for HEP experiments:

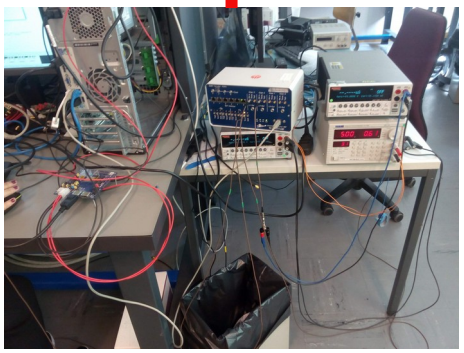
MODERATED CHANGES/STABLE

  - The **TEST BEAM** environment, where all these new functionalities are tested under a beam of particles, the working conditions the sensors are suppose to be, in order to characterize them

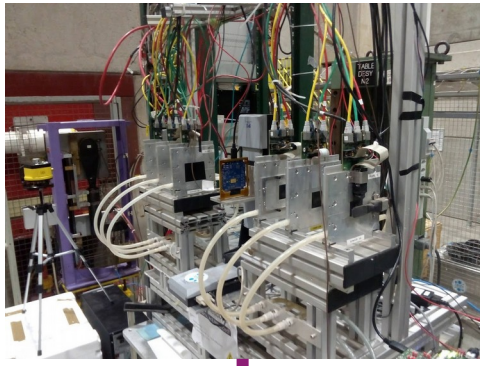
    → the **PROD.** env.



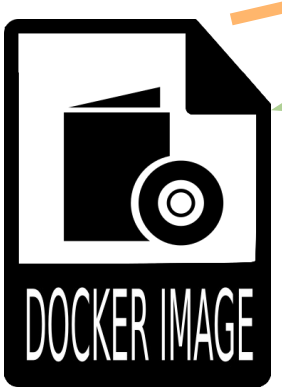
Why dockers for DAQ software?



LAB



TB

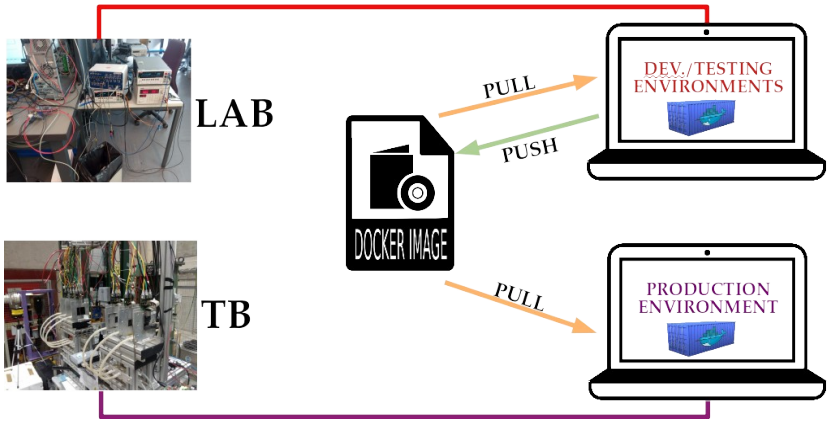


PULL

PUSH

PULL

Why dockers for DAQ software?



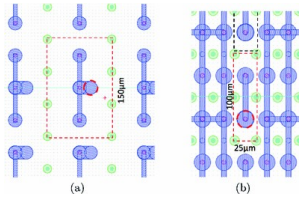
- Minimize software installation (dependencies, libraries, ..)
- Exactly same software in the lab than in the Test beam areas (reproducibility)

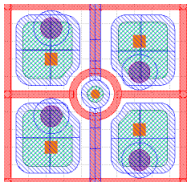
- Simplified maintenance (no need for multi-platform support)
- Very quick deployment from LAB to TB areas
  - Changes, fixes, etc.. are very quickly propagated

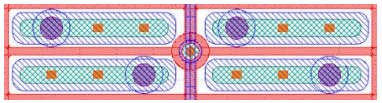
A real-case example

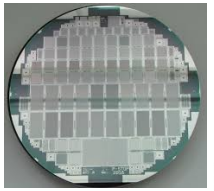
CMS Inner Tracker Phase-II: pixel sensors characterization campaign at 2018

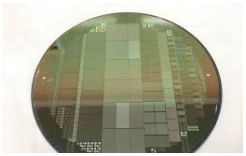
- sensor technology (3D, planars), geometries (several thickness), geometry layout (25x100/50x50 um^2 pitch), radiation tolerance, ...

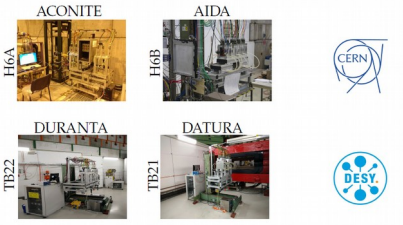




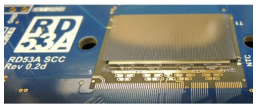




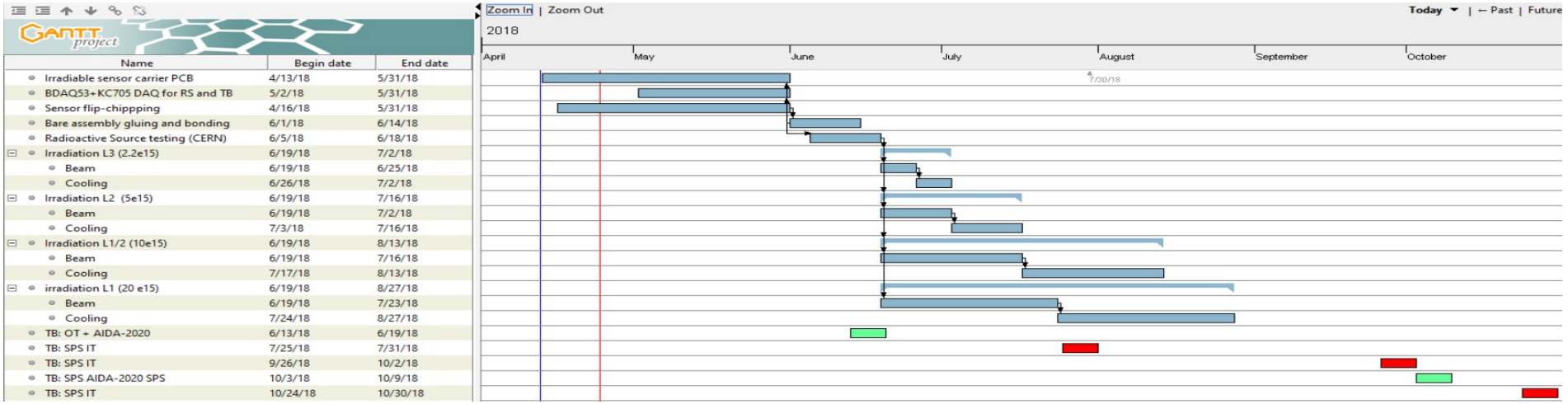




- Test beam characterization @ CERN and DESY
  - **EUDET/AIDA-**like telescopes
- New readout chip**: RD53A**
- Several DAQ systems (YARR/BDAQ53)
- Using a commercial FPGA board (Xilinx KC705)
- Several configurations/approaches for the chip carriers boards and FMC adapter cards



A real-case example



Name	Begin date	End date
Irradiable sensor carrier PCB	4/13/18	5/31/18
BDAQ53+KC705 DAQ for RS and TB	5/2/18	5/31/18
Sensor flip-chipping	4/16/18	5/31/18
Bare assembly gluing and bonding	6/1/18	6/14/18
Radioactive Source testing (CERN)	6/5/18	6/18/18
Irradiation L3 (2.2e15)	6/19/18	7/2/18
Beam	6/19/18	6/25/18
Cooling	6/26/18	7/2/18
Irradiation L2 (5e15)	6/19/18	7/16/18
Beam	6/19/18	7/2/18
Cooling	7/3/18	7/16/18
Irradiation L1/2 (10e15)	6/19/18	8/13/18
Beam	6/19/18	7/16/18
Cooling	7/17/18	8/13/18
irradiation L1 (20 e15)	6/19/18	8/27/18
Beam	6/19/18	7/23/18
Cooling	7/24/18	8/27/18
TB: OT + AIDA-2020	6/13/18	6/19/18
TB: SPS IT	7/25/18	7/31/18
TB: SPS IT	9/26/18	10/2/18
TB: SPS AIDA-2020 SPS	10/3/18	10/9/18
TB: SPS IT	10/24/18	10/30/18

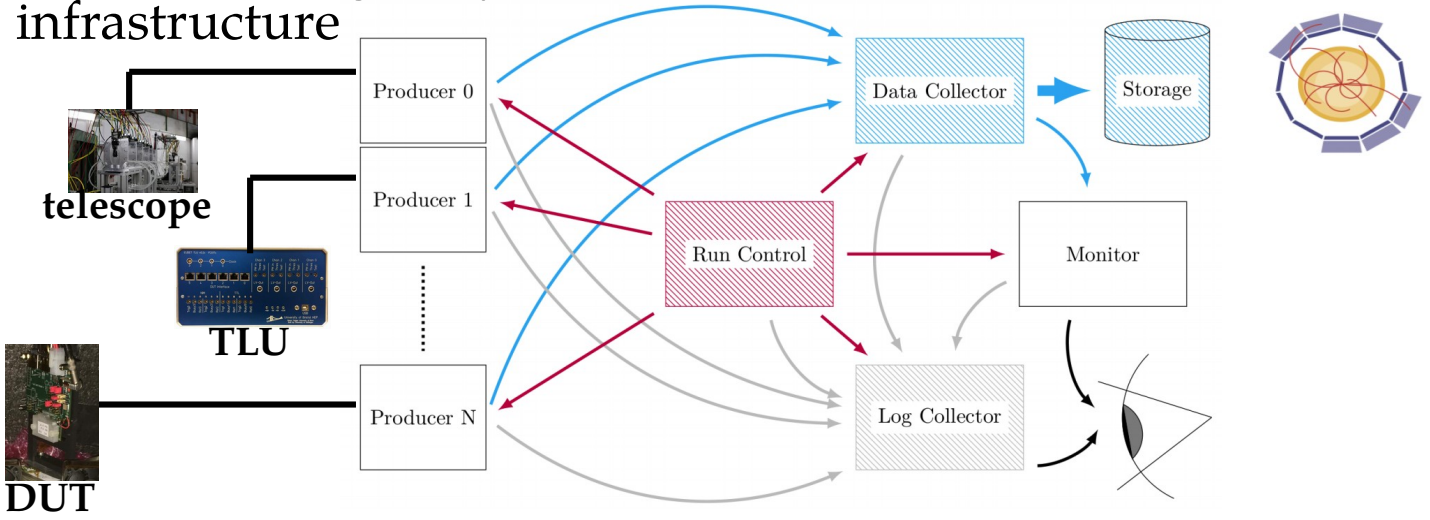
- Very tight agenda, with scheduled test beams while
  - Sensor modules (with RD53A assembled) not even delivered
  - BDAQ53 in continuous development (fixes, new functionalities, …)
  - BDAQ53 integration into EUDAQ
  - Several open choices for the test beam setups (Mezzanines cards, adapter cards, …)

→ EXCELENT USE-CASE FOR DAQ-SOFTWARE DOCKERIZATION

EUDET telescopes

https://telescopes.desy.de/Main_Page

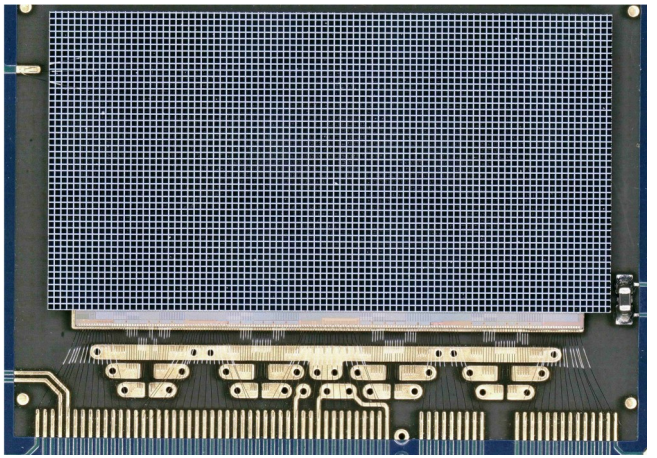
- **EUDET/AIDA-**like telescopes
  - High resolution telescopes,
  - available around the world (CERN, DESY, SLAC, Bonn, …);
  - stable (>10 years usage)
- **EUDAQ:** DAQ framework
  - Allows to integrate your Device Under Test (DUT) into the DAQ infrastructure



DUT Readout: RD53A

https://cds.cern.ch/record/2287593/files/%20RD53A_Manual_V3-42.pdf
https://gitlab.cern.ch/silab/bdaq53

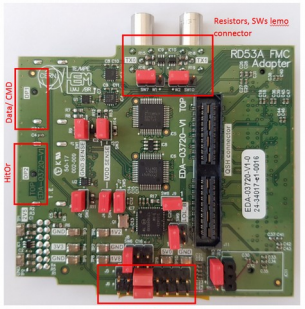
- **RD53A**: a demonstrator readout chip for HL-LHC upgrade of ATLAS and CMS
  - 65 um CMOS technology
  - Not a production chip
    - Several implemented choices (as Analog Front-Ends, ..)



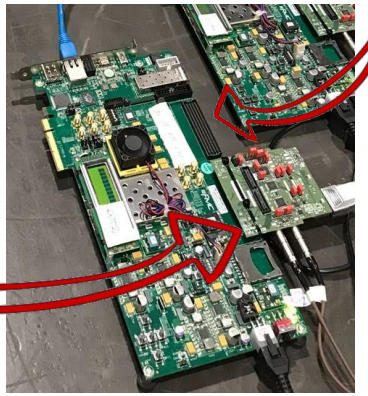
- **BDAQ53**: Bonn DAQ for the RD53A chip

bdaq53

- Supports several DAQ hardware platforms, in particular the **KC705** (a commercial Kintex 7 FPGA type)

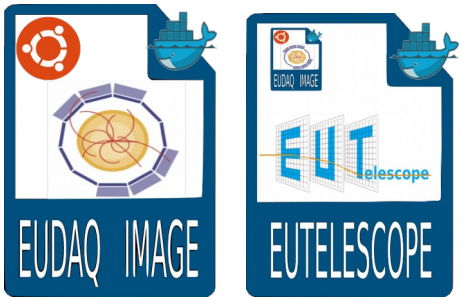


- Firmware modifications for the CMS IT setup made by **Esther Silva** (IFCA) in order to use the KC705 with the **CERN FMC** card



EUDAQ dockerization

https://github.com/duartej/dockerfiles-eudaqv1
https://github.com/duartej/dockerfiles-eutelescope



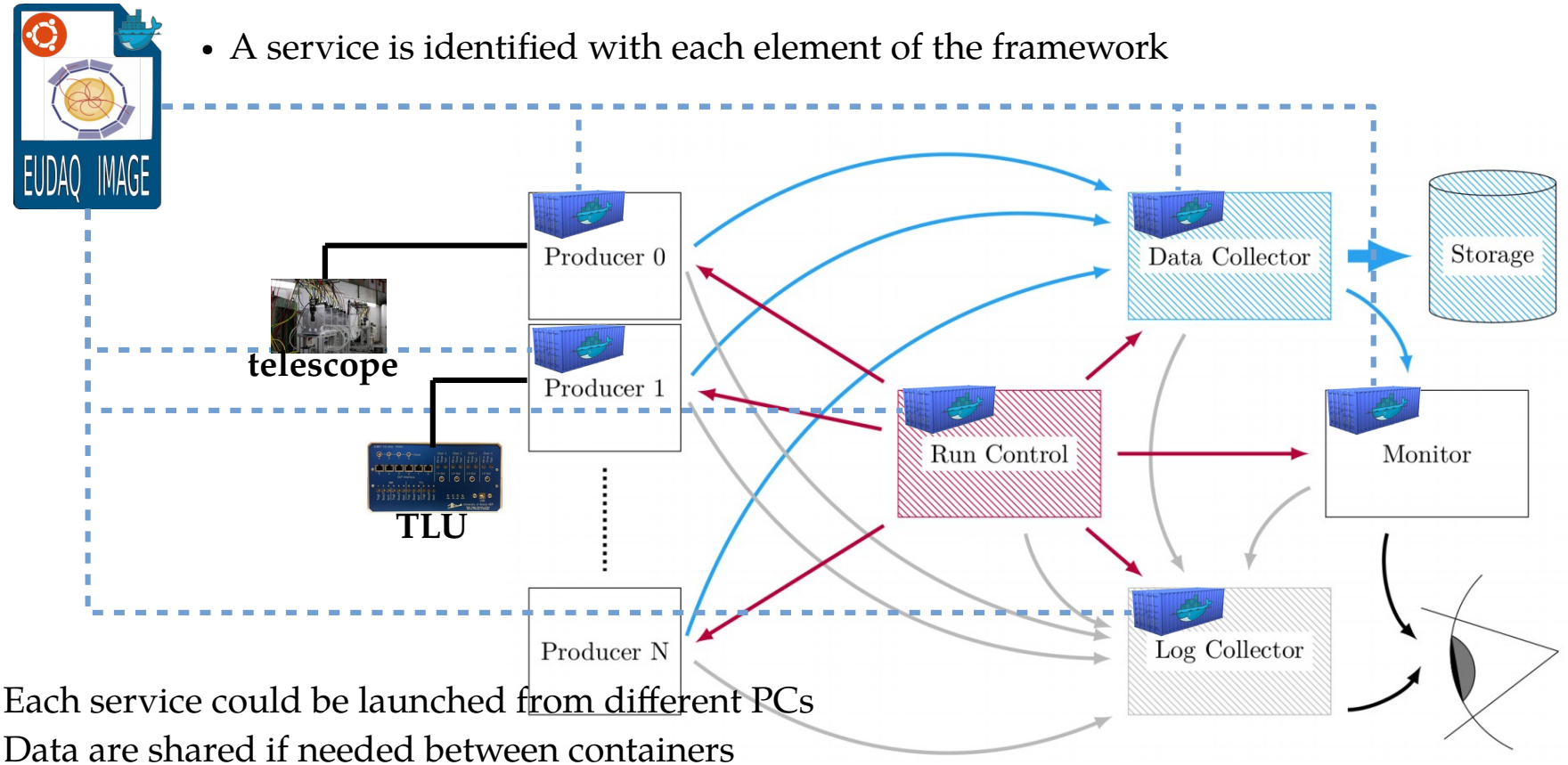
- Built over an Ubuntu 16
- EUDAQ v1.x-dev + all its dependencies
  - GCC-14, ROOT, QT5, ...
  - TLU and CMS Phase-I pixel support included
  - using eudaq fork: https://github.com/duartej/eudaq
    - Every change in this repo will trigger a new building in the dockerhub image repository
  - EUTelescope image: EUDAQ image + LCIO+EUTelescope
- Image creation and setup: https://github.com/duartej/dockerfiles-eudaqv1

```
$ git clone -b eutelescope https://github.com/duartej/dockerfiles-eudaqv1
$ cd dockerfiles-eudaqv1
$ source setup.sh
$ docker pull duartej/eudaqv1-ubuntu:latest
$ docker-compose -f docker-compose.yml -f production.yml up
```

- docker-compose configurations are available to run properly each element of the framework (run control, data collector, ...)
  - Development & production configurations

EUDAQ dockerization

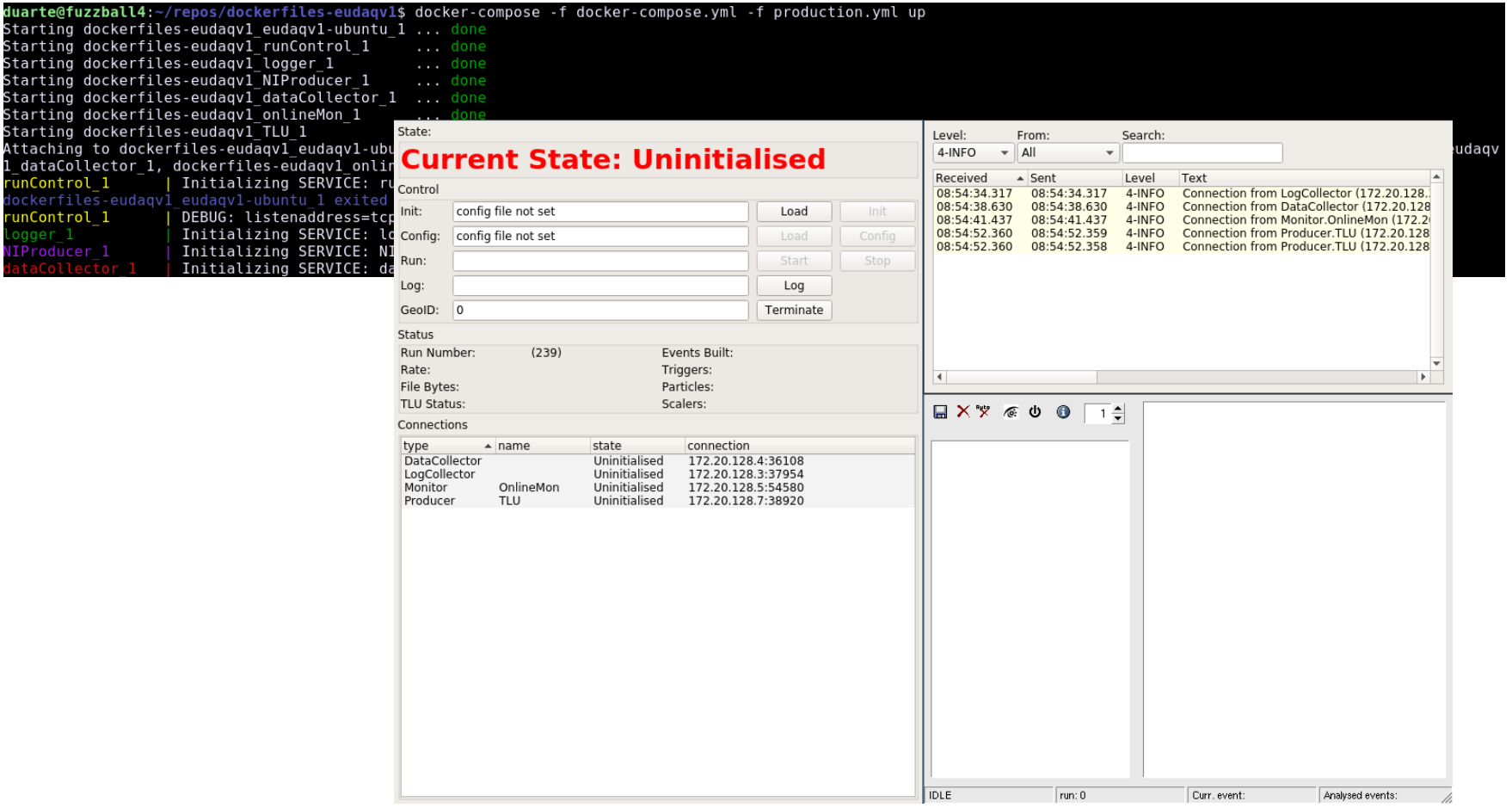
- A service is identified with each element of the framework



- Each service could be launched from different PCs
- Data are shared if needed between containers
- Hardware communicate perfectly with the containerized services
- Only need to be exposed the IP address of RunControl service (all other elements are being attached to it)

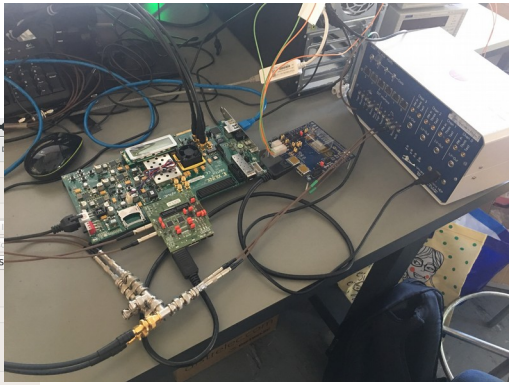
EUDAQ dockerization

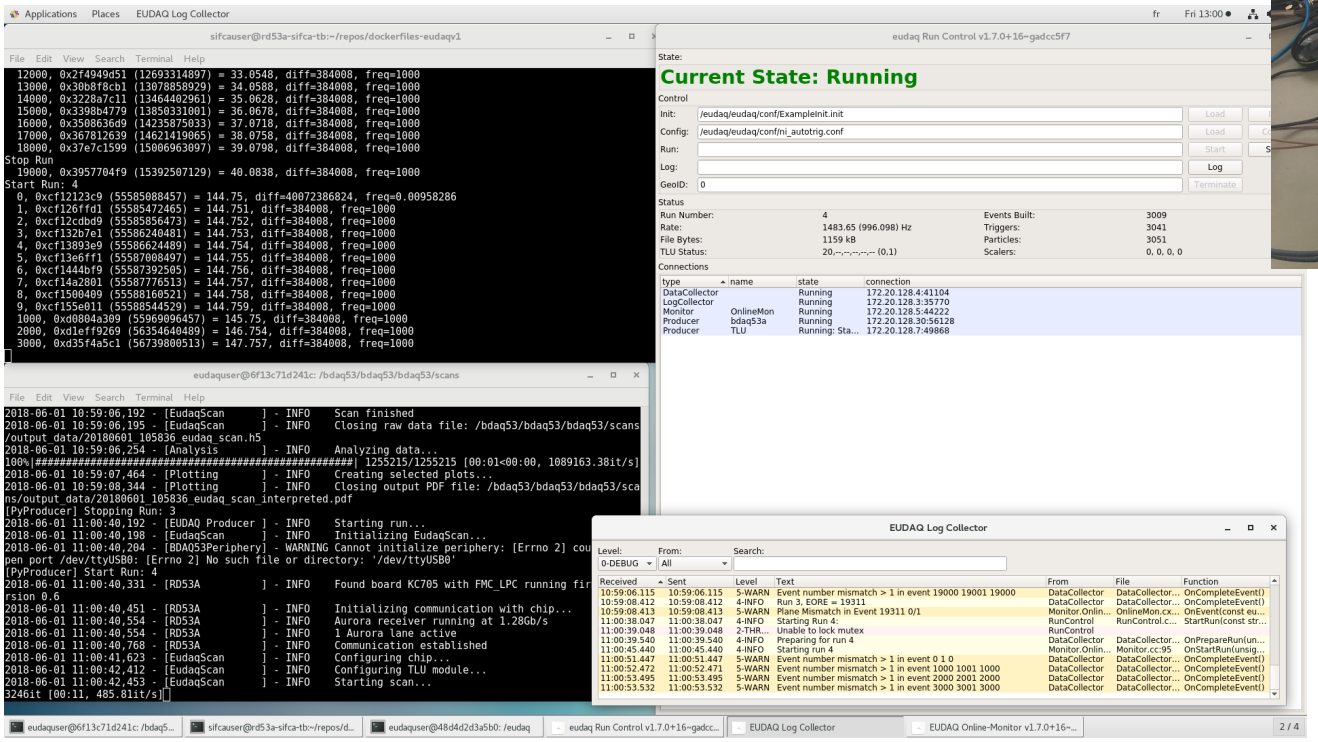
Example launching available services (default config.) in prod. :



EUDAQ dockerization

TLU and a RD53A with dockerized EUDAQ in a LAB setup:





BDAQ53 dockerization

https://github.com/duartej/dockerfiles-bdaq53a

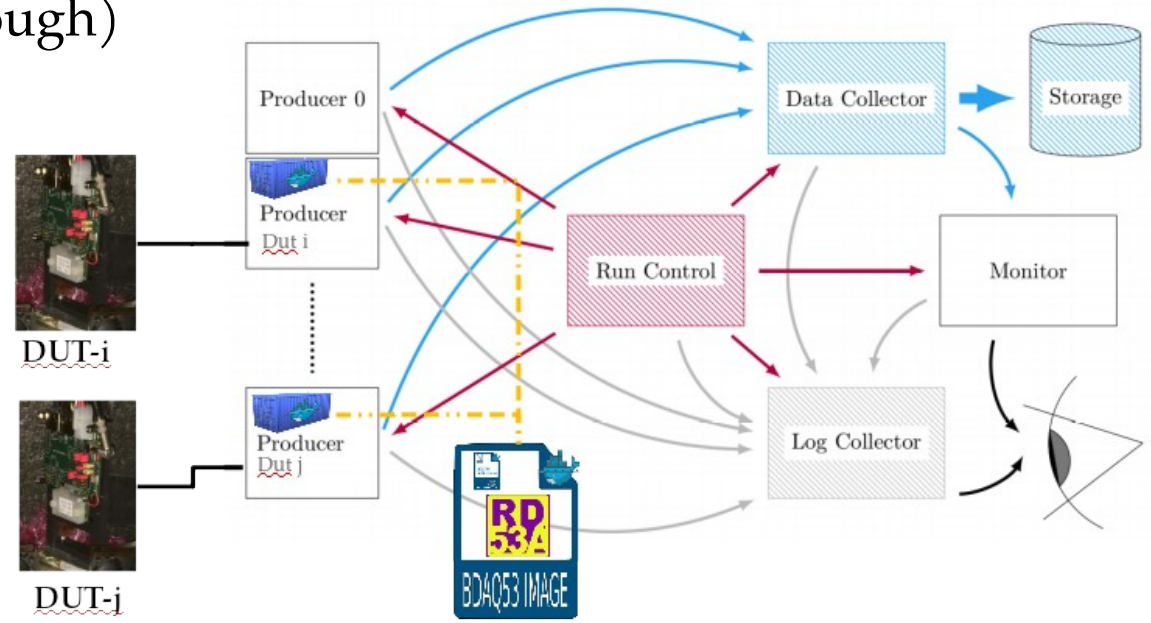
- Built over the EUDAQ image
- BDAQ53 dependencies
  - python mostly (numpy, scipy, matplotlib,...)
  - using bdaq53 fork: https://gitlab.cern.ch/sifca/bdaq53
    - Every change in this repo will trigger a new building in the dockerhub image repository
  - No master branch**:**
    - **eutelescope** branch: uses EUDAQ image as base
    - **Plain** branch: uses EUTELESCOPE image as base

- Image creation and setup: https://github.com/duartej/dockerfiles-bdaq53a

```
$ EDAQDOCKER=<path to dockerfiles-eudaqv1 local folder>
$ git clone -b eutelescope https://github.com/duartej/dockerfiles-bdaq53a.git
$ cd dockerfiles-bdaq53a
$ source setup.sh ${EDAQDOCKER}
$ docker pull duartej/bdaq53:eutelescope
```

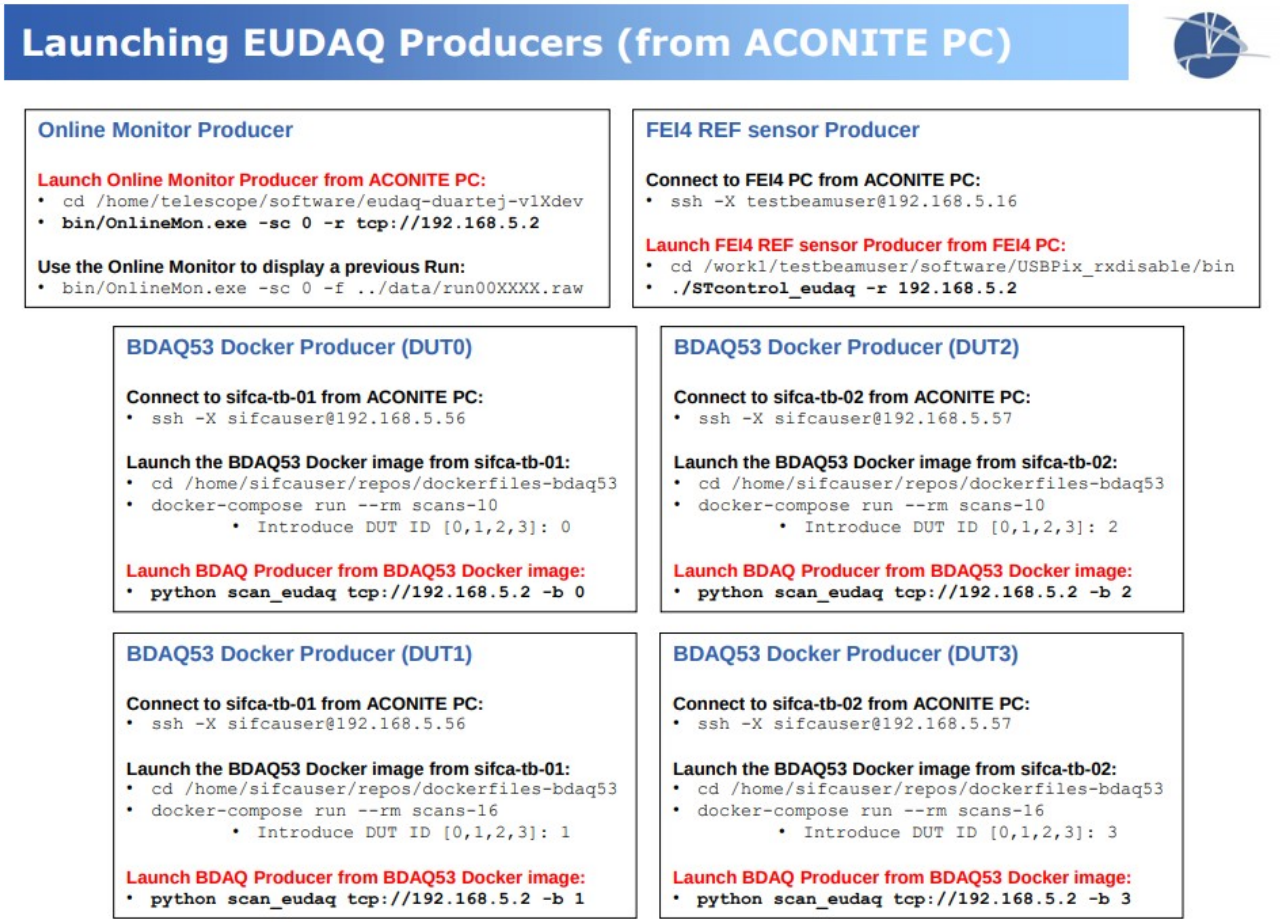
BDAQ53 dockerization

- The container is used to launch the BDAQ53 producer (chip command control and data sending controlled by the EUDAQ run control)
  - BDAQ53 is integrated into EUDAQ
- Each container is associated to a unique DUT
  - DUT isolated from the host computer (board IP address should be unique in the host PC, though)
  - Potentially the same PCs could control any number of DUTs, launching any number of containers (we tested up to 2 DUTS per PC)



BDAQ53 dockerization

- Extracted from the oct.-2018 TB's shifter instructions slides:



Launching EUDAQ Producers (from ACONITE PC)

Online Monitor Producer

Launch Online Monitor Producer from ACONITE PC:
- cd /home/telescope/software/eudaq-duartej-v1Xdev
- **bin/OnlineMon.exe -sc 0 -r tcp://192.168.5.2**

Use the Online Monitor to display a previous Run:
- bin/OnlineMon.exe -sc 0 -f ../data/run00XXXX.raw

FEI4 REF sensor Producer

Connect to FEI4 PC from ACONITE PC:
- ssh -X testbeamuser@192.168.5.16

Launch FEI4 REF sensor Producer from FEI4 PC:
- cd /work1/testbeamuser/software/USBPix_rxdisable/bin
- **./STcontrol_eudaq -r 192.168.5.2**

BDAQ53 Docker Producer (DUT0)

Connect to sifca-tb-01 from ACONITE PC:
- ssh -X sifcauser@192.168.5.56

Launch the BDAQ53 Docker image from sifca-tb-01:
- cd /home/sifcauser/repos/dockerfiles-bdaq53
- docker-compose run --rm scans-10
  - Introduce DUT ID [0,1,2,3]: 0

Launch BDAQ Producer from BDAQ53 Docker image:
- **python scan_eudaq tcp://192.168.5.2 -b 0**

BDAQ53 Docker Producer (DUT2)

Connect to sifca-tb-02 from ACONITE PC:
- ssh -X sifcauser@192.168.5.57

Launch the BDAQ53 Docker image from sifca-tb-02:
- cd /home/sifcauser/repos/dockerfiles-bdaq53
- docker-compose run --rm scans-10
  - Introduce DUT ID [0,1,2,3]: 2

Launch BDAQ Producer from BDAQ53 Docker image:
- **python scan_eudaq tcp://192.168.5.2 -b 2**

BDAQ53 Docker Producer (DUT1)

Connect to sifca-tb-01 from ACONITE PC:
- ssh -X sifcauser@192.168.5.56

Launch the BDAQ53 Docker image from sifca-tb-01:
- cd /home/sifcauser/repos/dockerfiles-bdaq53
- docker-compose run --rm scans-16
  - Introduce DUT ID [0,1,2,3]: 1

Launch BDAQ Producer from BDAQ53 Docker image:
- **python scan_eudaq tcp://192.168.5.2 -b 1**

BDAQ53 Docker Producer (DUT3)

Connect to sifca-tb-02 from ACONITE PC:
- ssh -X sifcauser@192.168.5.57

Launch the BDAQ53 Docker image from sifca-tb-02:
- cd /home/sifcauser/repos/dockerfiles-bdaq53
- docker-compose run --rm scans-16
  - Introduce DUT ID [0,1,2,3]: 3

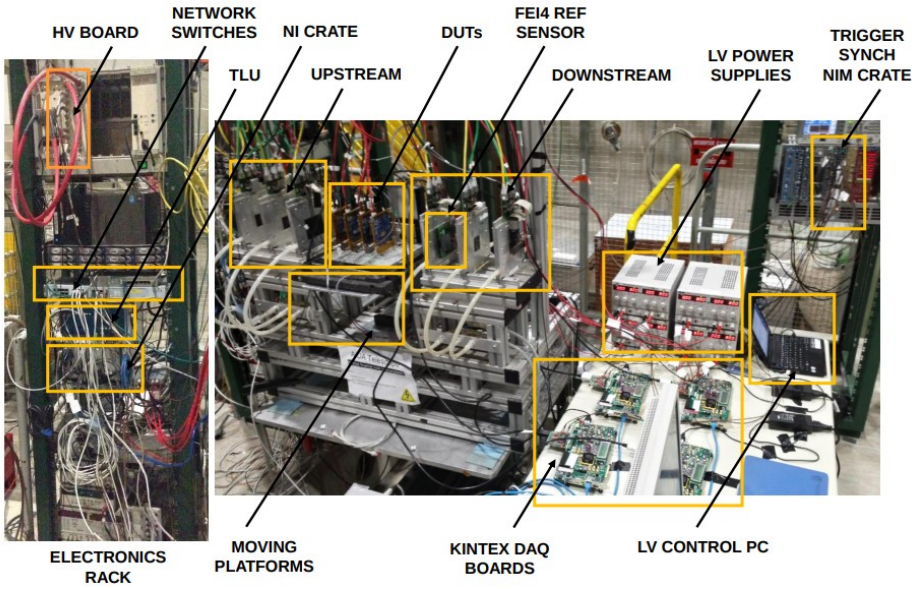
Launch BDAQ Producer from BDAQ53 Docker image:
- **python scan_eudaq tcp://192.168.5.2 -b 3**

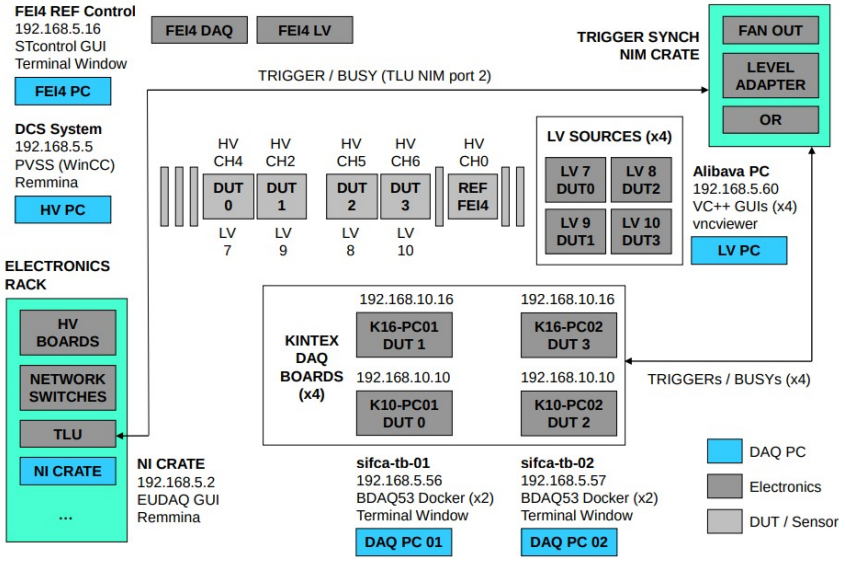
9

Slide from **Fco. Javier Gonzalez Sanchez**

Deployment: CMS IT Phase-II TB campaign 2018

- EUDAQ containers routinely used at LAB
  - Never deployed at TB areas due to other high priority tasks, but ready to be tested
- BDAQ53 containers routinely deployed and used at test beams at CERN
  - Proof of concept and tests at first TB (June 2018), used on all posterior TBs, with increasing complexity (more DUTs)
  - Different areas (H6A, H6B, H2)
  - Up to 4 DUTs at the same setup: 2 DAQ PCs with 2 containers each





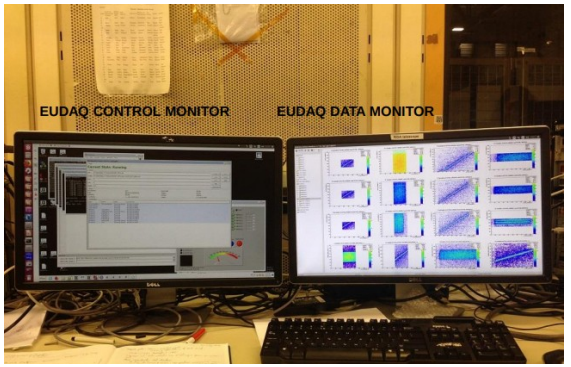
Conclusions

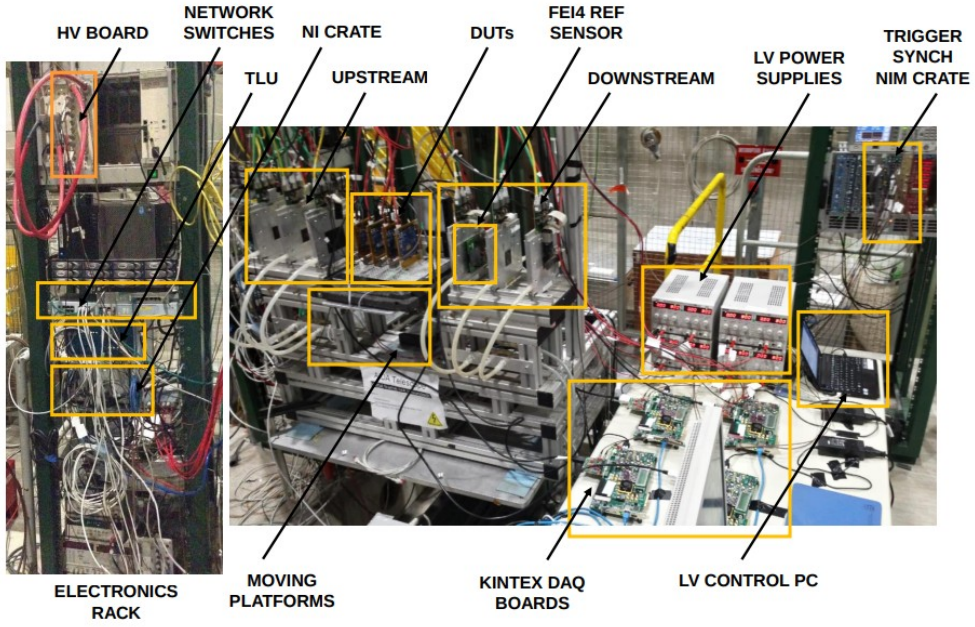
- DAQ software for EUDET telescopes and RD53A chip has been dockerized
  - No issues found (hardware communication, services communication, CPU and storage speed, ...)

- DAQ software containers **SUCCESSFULLY** used during the 2018 TB campaign for the CMS Inner Tracker Phase-II group
  - All data show in conf./workshops/papers/... from CMS IT Phase-II obtained at CERN during 2018 was taken with the BDAQ53 DAQ software dockerized
  - Proved to fulfill expectations:
    - Minimize maintenance
    - Reproducibility between LAB and TB areas
    - Robustness
    - Rapid deployment:
      - Download image with changes in the TB DAQ PCs and run

- Just used docker, but other alternatives as *singularity* (used at CERN clusters) are planned to be checked as well

BACKUP SLIDES





Launching EUDAQ Producers (from ACONITE PC)

Online Monitor Producer

Launch Online Monitor Producer from ACONITE PC:
- `cd /home/telescope/software/eudaq-duartej-v1Xdev`
- **`bin/OnlineMon.exe -sc 0 -r tcp://192.168.5.2`**

Use the Online Monitor to display a previous Run:
- `bin/OnlineMon.exe -sc 0 -f ../data/run00XXXX.raw`

FEI4 REF sensor Producer

Connect to FEI4 PC from ACONITE PC:
- `ssh -X testbeamuser@192.168.5.16`

Launch FEI4 REF sensor Producer from FEI4 PC:
- `cd /work1/testbeamuser/software/USBPix_rxdisable/bin`
- **`./STcontrol_eudaq -r 192.168.5.2`**

BDAQ53 Docker Producer (DUT0)

Connect to sifca-tb-01 from ACONITE PC:
- `ssh -X sifcauser@192.168.5.56`

BDAQ53 Docker image from sifca-tb-01:
`me/sifcauser/repos/dockerfiles-bdaq53`
`-compose run --rm scans-10`
- `Introduce DUT ID [0,1,2,3]: 0`

AQ Producer from BDAQ53 Docker image:
`scan_eudaq tcp://192.168.5.2 -b 0`

BDAQ53 Docker Producer (DUT2)

Connect to sifca-tb-02 from ACONITE PC:
- `ssh -X sifcauser@192.168.5.57`

Launch the BDAQ53 Docker image from sifca-tb-02:
- `cd /home/sifcauser/repos/dockerfiles-bdaq53`
- `docker-compose run --rm scans-10`
  - `Introduce DUT ID [0,1,2,3]: 2`

Launch BDAQ Producer from BDAQ53 Docker image:
- **`python scan_eudaq tcp://192.168.5.2 -b 2`**

Docker Producer (DUT1)

sifca-tb-01 from ACONITE PC:
`sifcauser@192.168.5.56`

BDAQ53 Docker image from sifca-tb-01:
`me/sifcauser/repos/dockerfiles-bdaq53`
`-compose run --rm scans-16`
- `Introduce DUT ID [0,1,2,3]: 1`

AQ Producer from BDAQ53 Docker image:
`scan_eudaq tcp://192.168.5.2 -b 1`

BDAQ53 Docker Producer (DUT3)

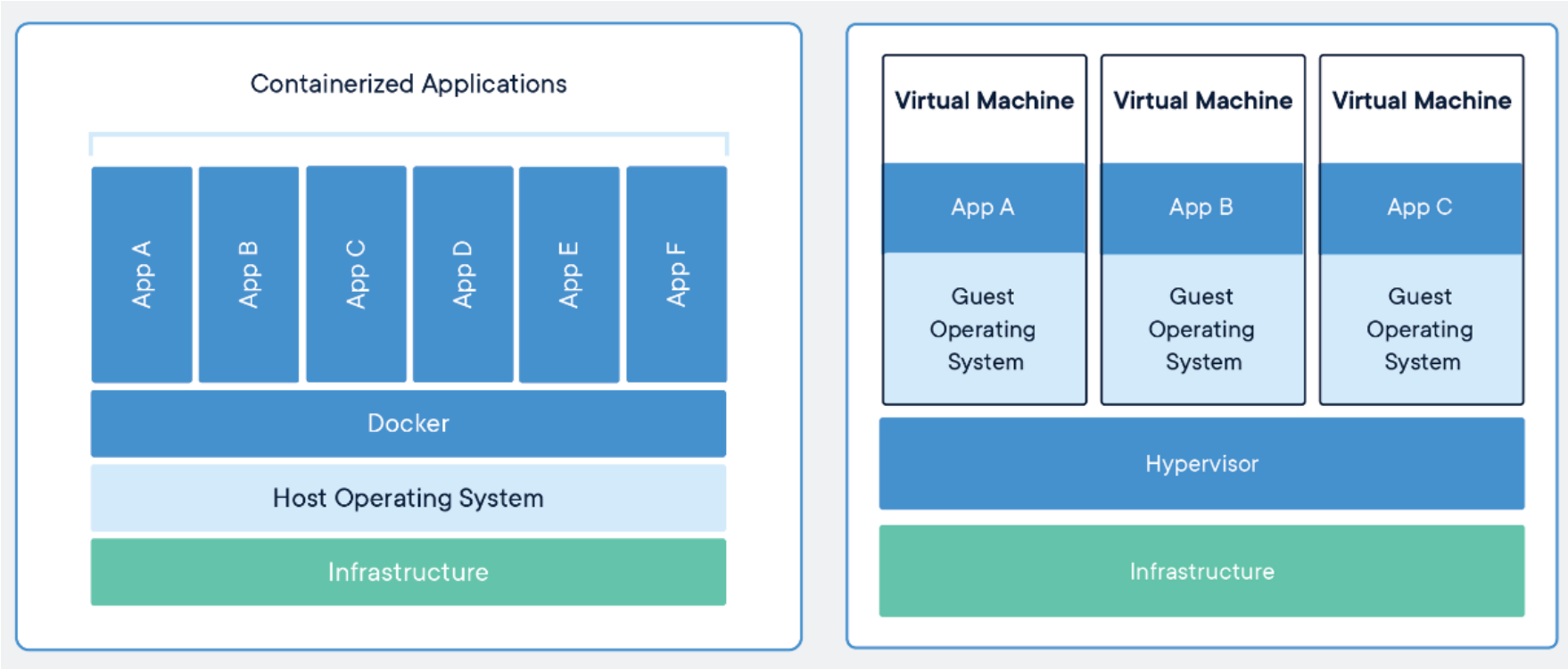
Connect to sifca-tb-02 from ACONITE PC:
- `ssh -X sifcauser@192.168.5.57`

Launch the BDAQ53 Docker image from sifca-tb-02:
- `cd /home/sifcauser/repos/dockerfiles-bdaq53`
- `docker-compose run --rm scans-16`
  - `Introduce DUT ID [0,1,2,3]: 3`

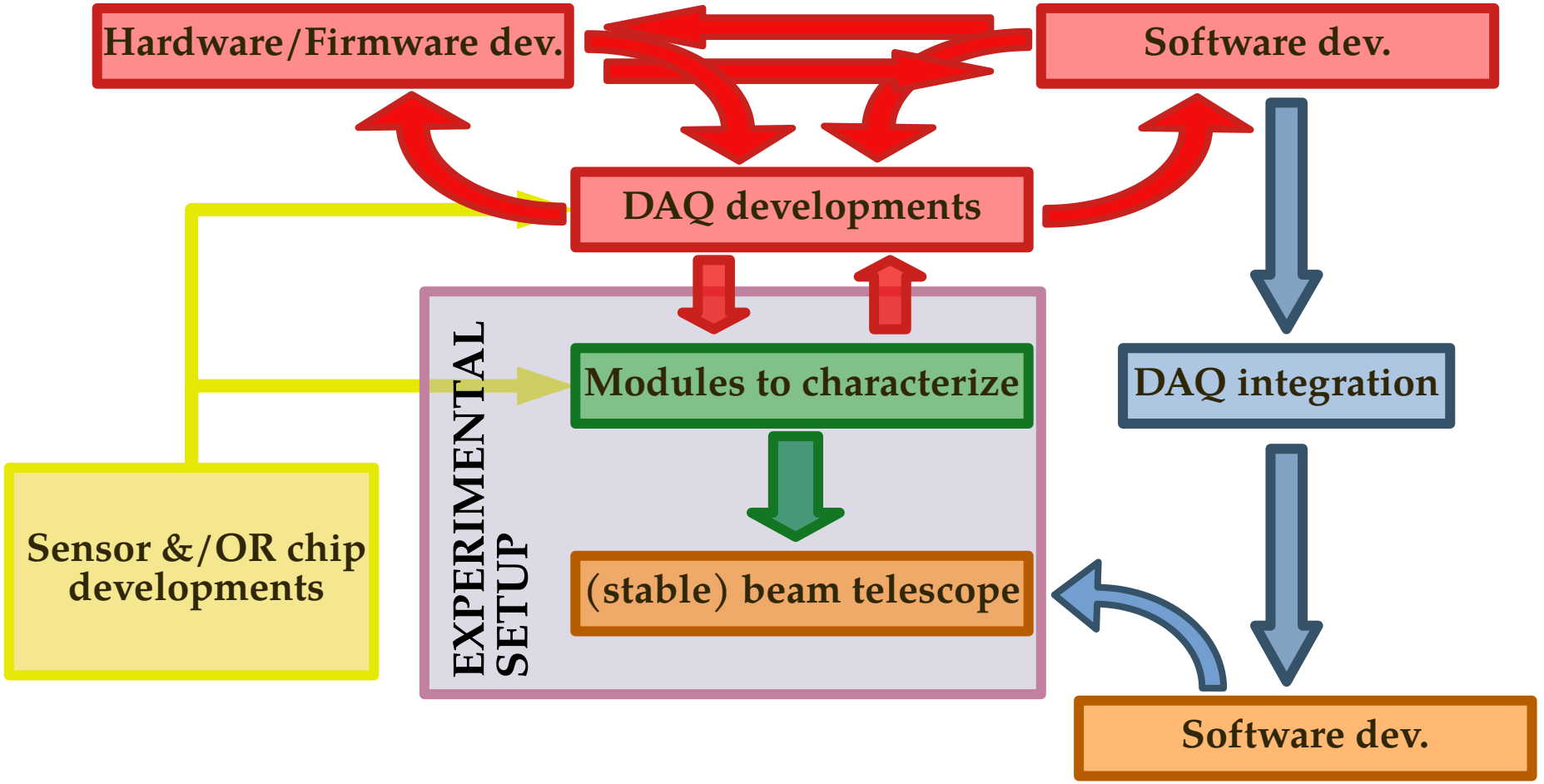
Launch BDAQ Producer from BDAQ53 Docker image:
- **`python scan_eudaq tcp://192.168.5.2 -b 3`**

Container vs. Virtual Machine

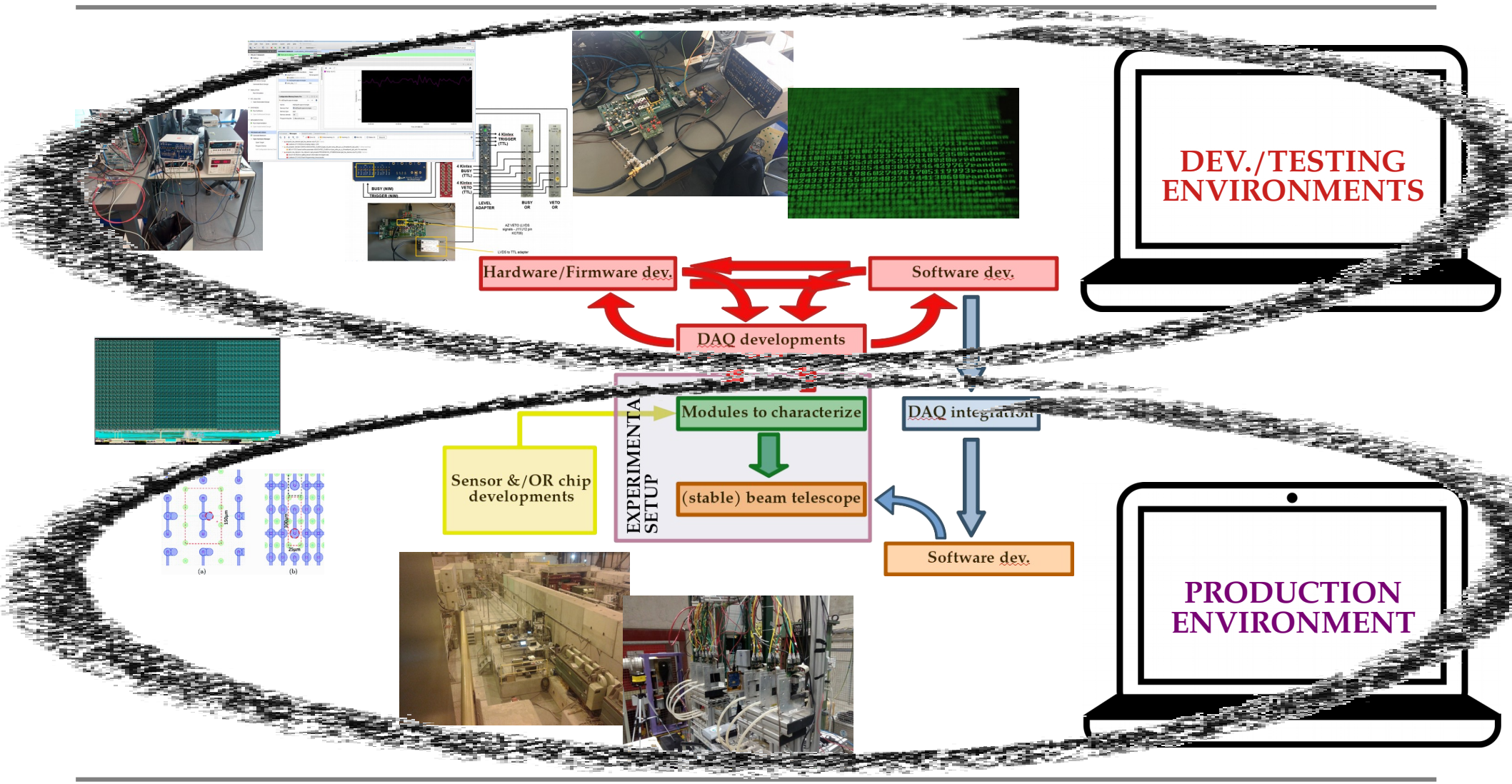
- Plenty of explanations at the Internet, a particular good one:
  https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine



DAQ software at HEP experiments



DAQ software at HEP experiments

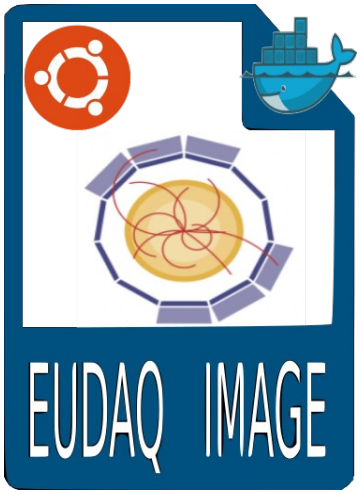


EUDAQ dockerization

https://github.com/duartej/dockerfiles-eudaqv1
https://github.com/duartej/dockerfiles-eutelescope

- Image build and usage instructions:
  https://github.com/duartej/dockerfiles-eudaqv1
- Dockerhub:
  https://cloud.docker.com/u/duartej/repository/docker/duartej/eudaqv1-ubuntu
  https://cloud.docker.com/u/duartej/repository/docker/duartej/eutelescope
- Latest built image from dockerhub:

```
docker pull duartej/eudaqv1-ubuntu
docker pull duartej/eutelescope
```



BDAQ53 dockerization

https://github.com/duartej/dockerfiles-bdaq53a

- Image build and usage instructions:
  https://github.com/duartej/dockerfiles-bdaq53a
  - **eutelescope** branch: uses EUDAQ image as base
  - **Plain** branch: uses EUTELESCOPE image as base
- Latest built image from dockerhub:

```
docker pull duartej/bdaq53:eutelescope
docker pull duartej/bdaq53:plain
```