

# Computing and software model for T2K and HK

Mathieu Guigue for the T2K and HK collaboration  
IRN Neutrino, November 2021, LPNHE

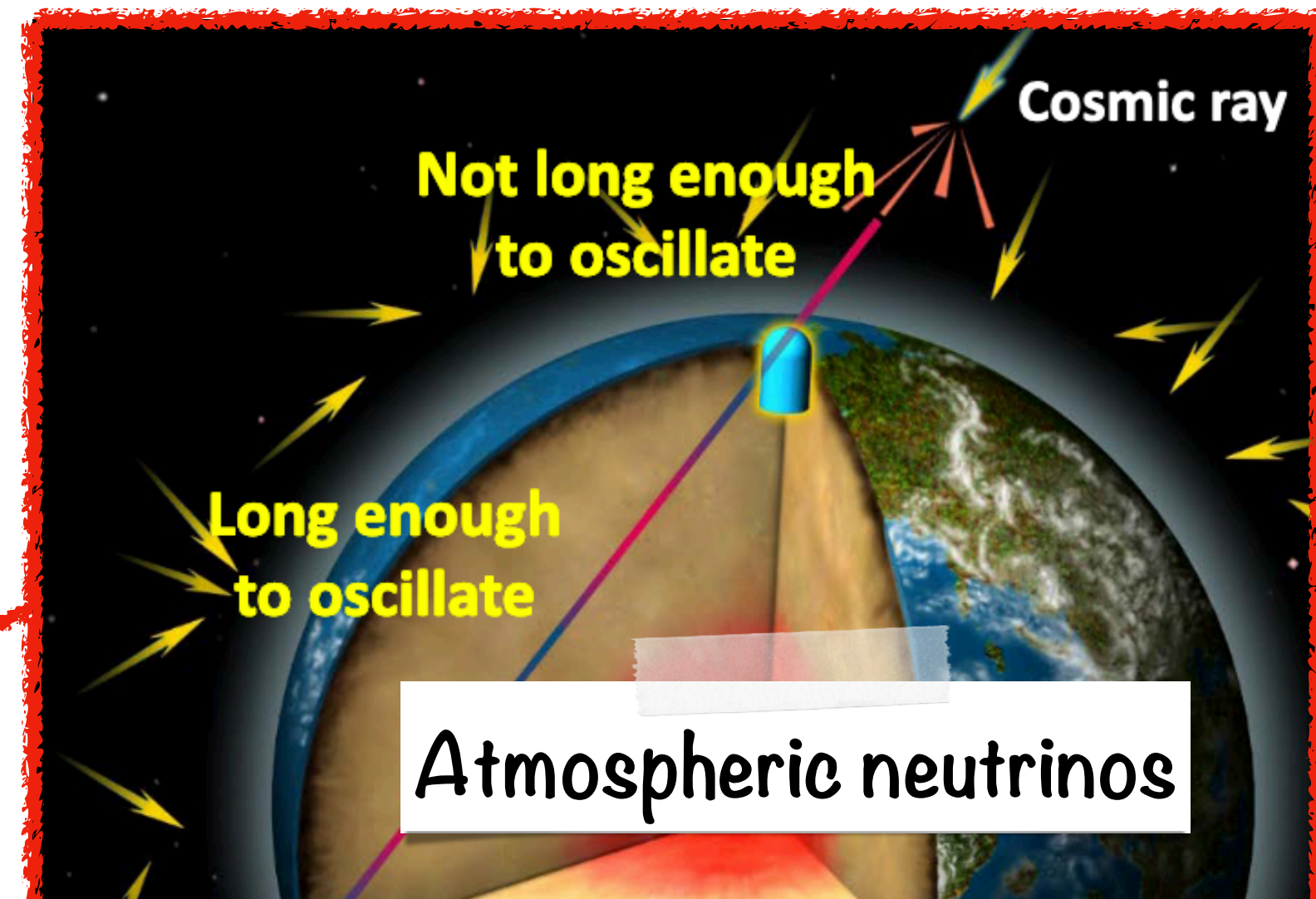
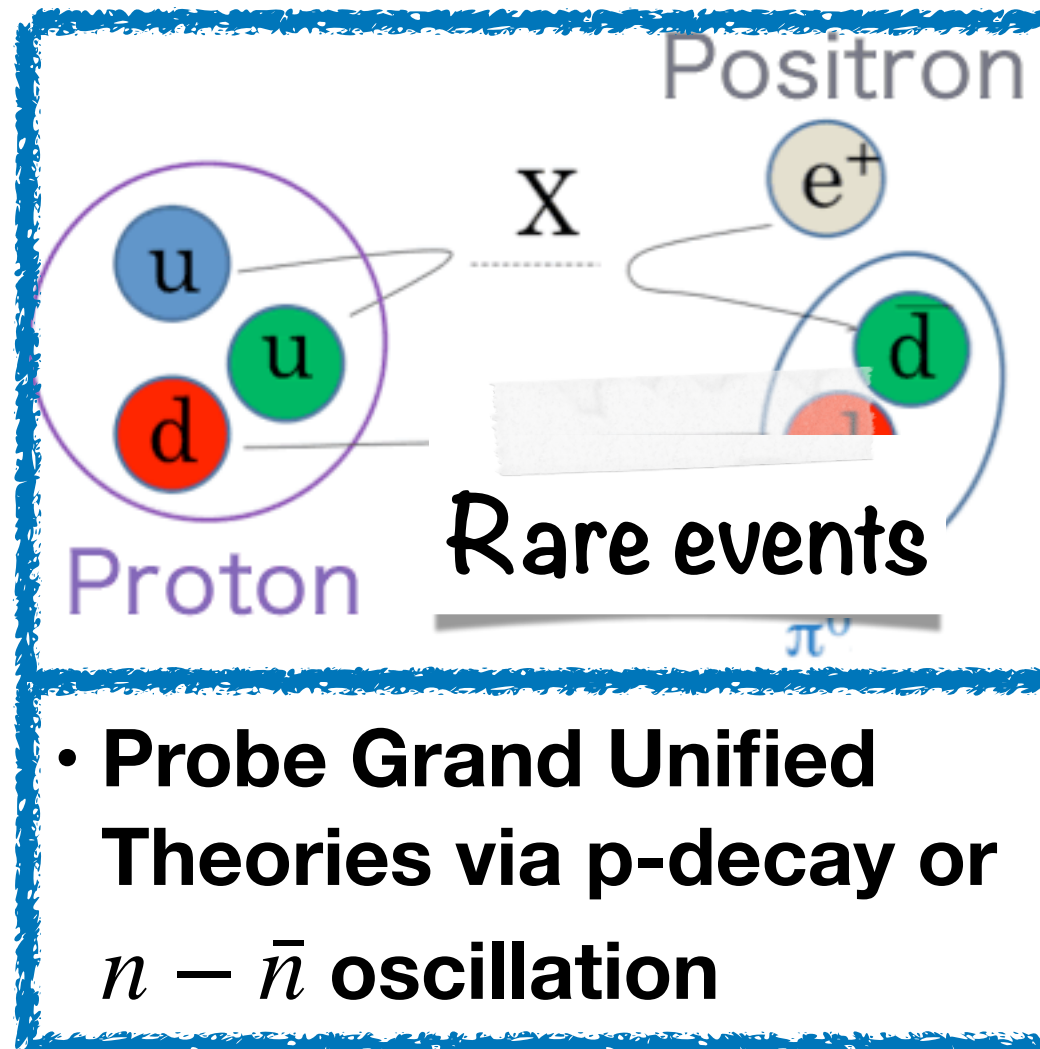


## Solar neutrinos

- MSW effect
- Non-standard interactions

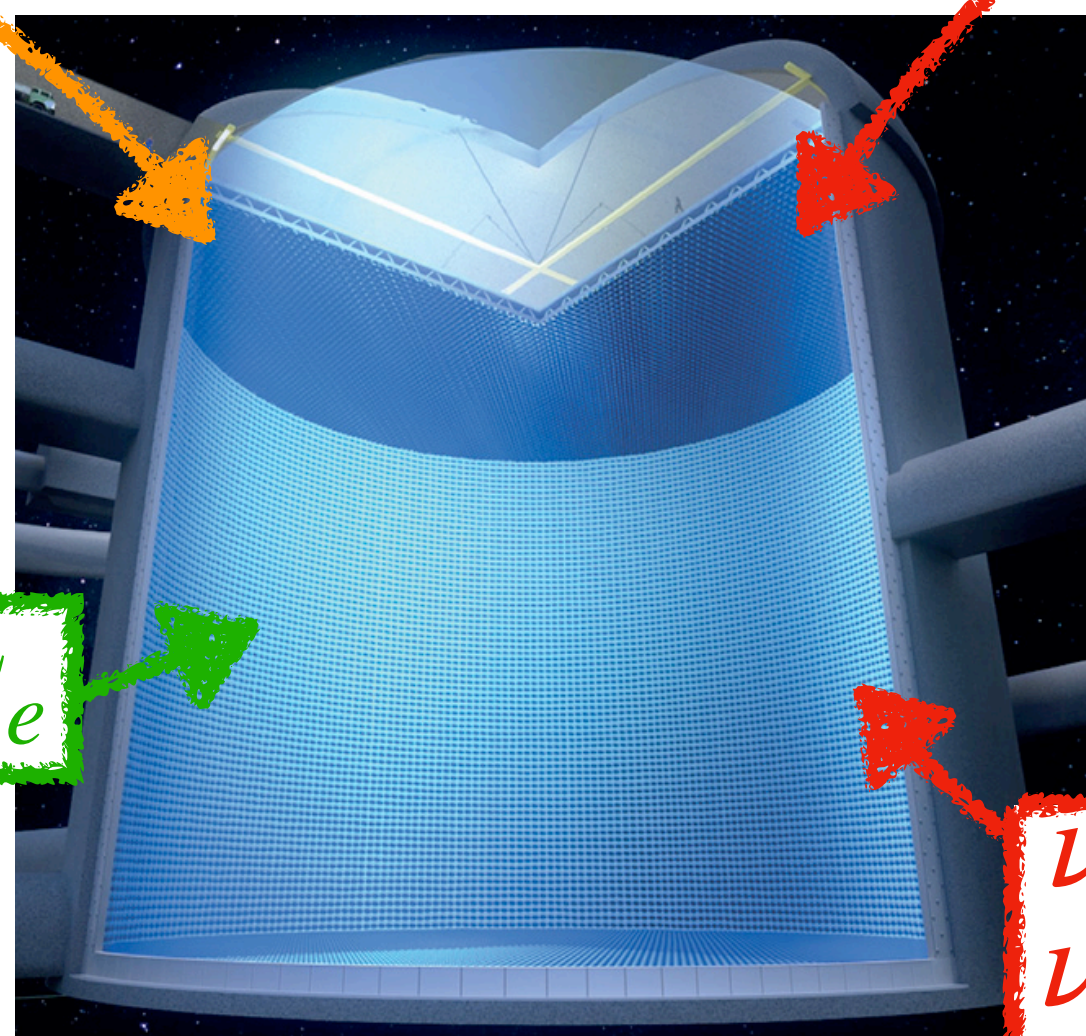
## Supernovae neutrinos

- Transient SN  $\nu$ : constrain SN profile models
- Relic SN  $\nu$ : constrain cosmic star formation



- Observe CP violation for leptons at  $5\sigma$
- Precise measurement of  $\delta_{CP}$
- High sensitivity to  $\nu$  mass ordering

## J-PARC accelerator neutrinos

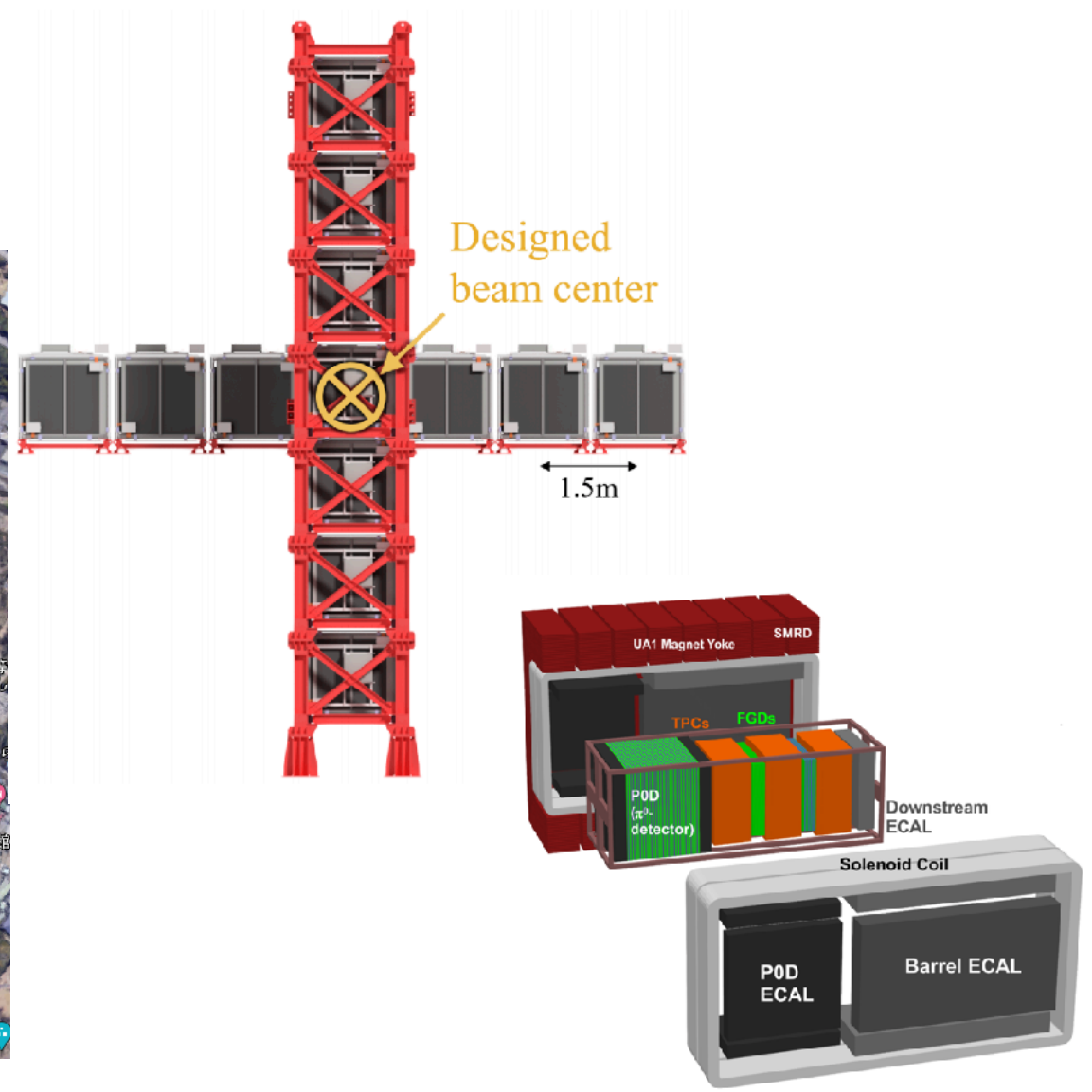


$\bar{\nu}_e$

$\nu_e$   $\bar{\nu}_e$   
 $\nu_\mu$   $\bar{\nu}_\mu$

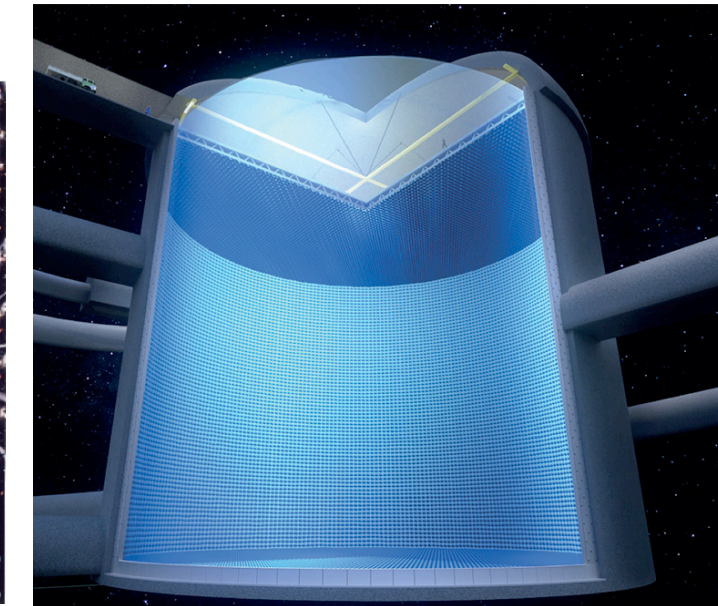
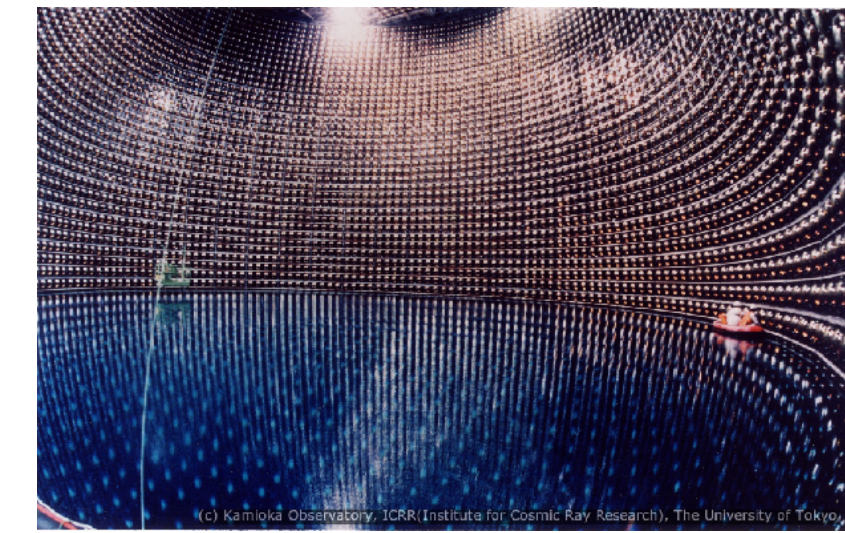
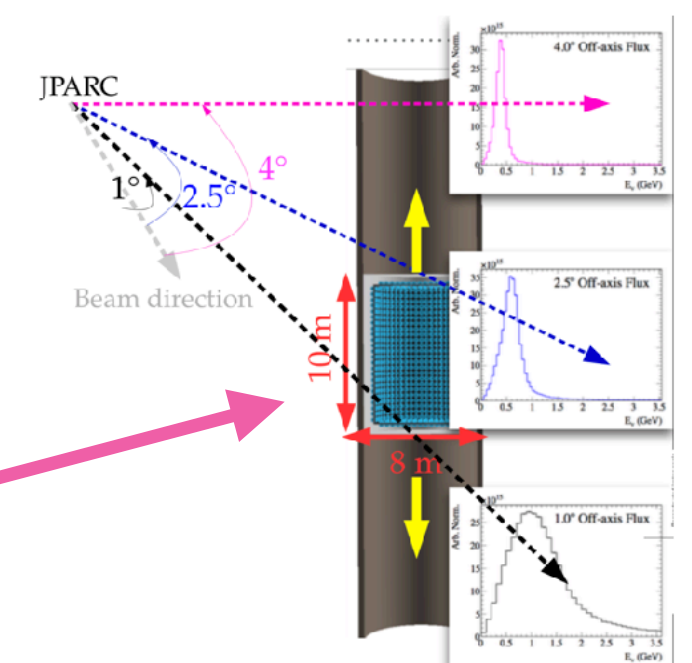
$\nu_e$   $\bar{\nu}_e$   
 $\nu_\mu$   $\bar{\nu}_\mu$





**T2K**

**Hyper-Kamiokande**

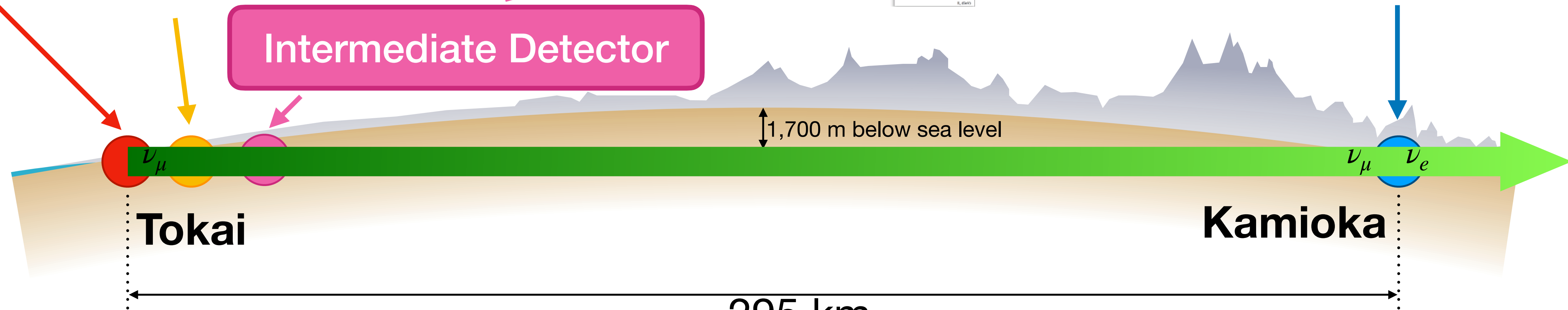


**J-PARC**

**Near Detectors**

**Intermediate Detector**

**Super/Hyper-Kamiokande**





Estimation of required resources  
based on T2K and SK experience

Two main contributors to storage:  
ND280 (mostly MC)  
Far detector (mostly raw data)

Overall needs (now → 2037):  
ND280: ~8.6 PB  
Far detector: ~18.5 PB  
~880 MCPU.h

(minimal with one copy of each file)

Name	Distance along beam	Angle wrt. beam	Expected data rate
INGRID	280 m	0°	78 GB/day
ND280	280 m	2.5°	214 GB/day
IWCD	2 km	0° – 4°	170 GB/day
Far detector	295 km	2.5°	5 TB/day

During construction phase → 2027

Detector	MC (HS06 CPU.h)	MC Storage (TB)
INGRID	0.13M	7
ND280	19.2M	2,250
IWCD	97M	52
Far detector	20M	500
Total	136.33M	2,824

During data taking phase 2027 → 2037

Detector	Data Storage (TB)	MC (HS06 CPU.h)	MC Storage (TB)
INGRID	226	0.51M	26
ND280	669	42.2M	4,950
IWCD	620	684M	367
Far detector	18,440	25M	500
Total	19,955	751.71M	5,858



# Take-away messages

Various computing resources for T2K/HK: RAL+CC-IN2P3+others

- Tiered computing model
- Integration via DIRAC

Utilize cloud resources available at “smaller” institutions

- Demonstrator being built in the context of Jennifer-II consortium
- Integration of LPNHE, LAL and Grif cloud resources

Utilize containers technology for consistent work environment across cluster and clouds



# T2K/HK Computing model

Tier model similar to CERN's

Data stored on T0 and copied on  $\geq 2$  T1

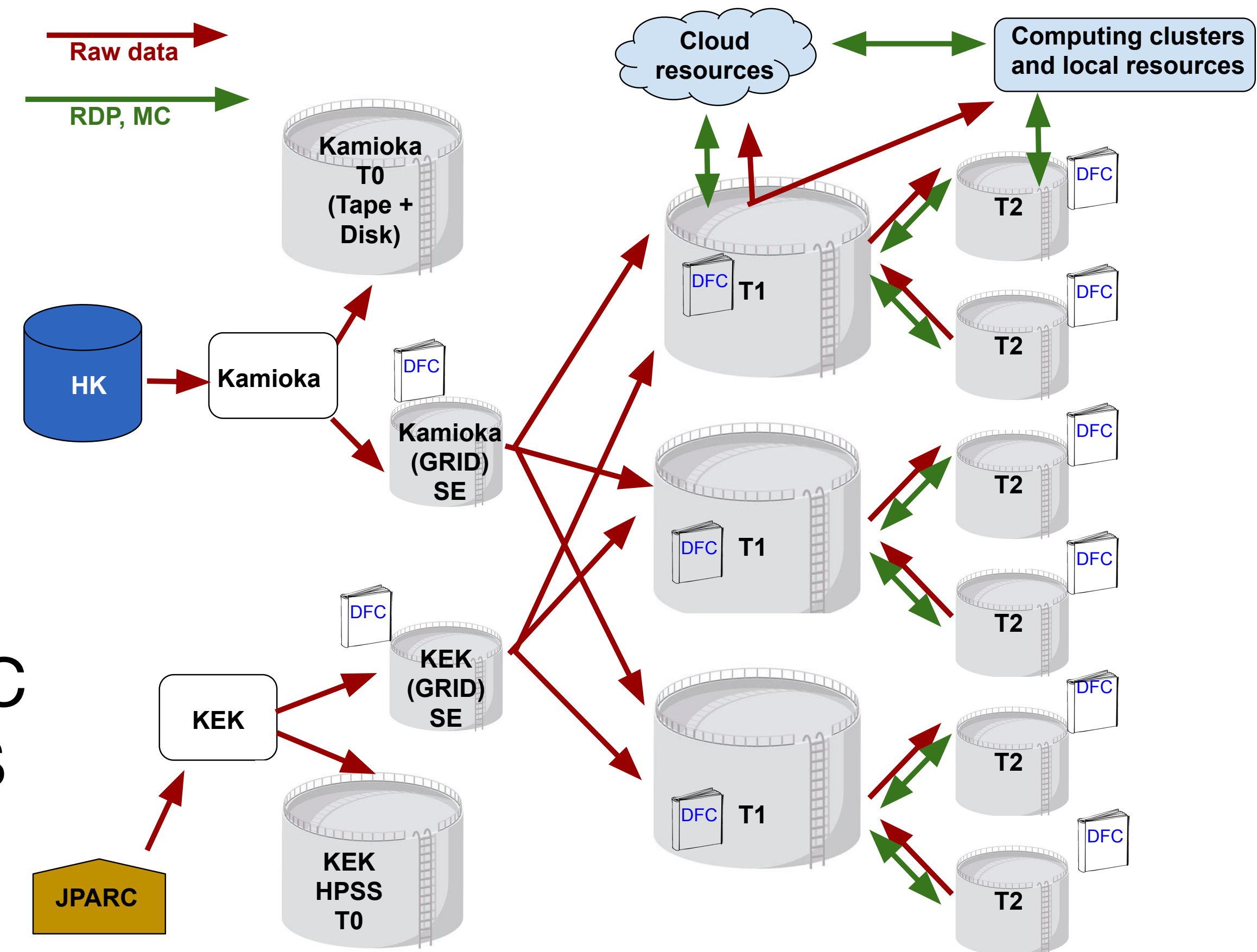
MC productions run on T1 sites

MC stored on  $\geq 1$  T1 and several T2

T2 sites used by users for local analyses

Resources and data management using DIRAC

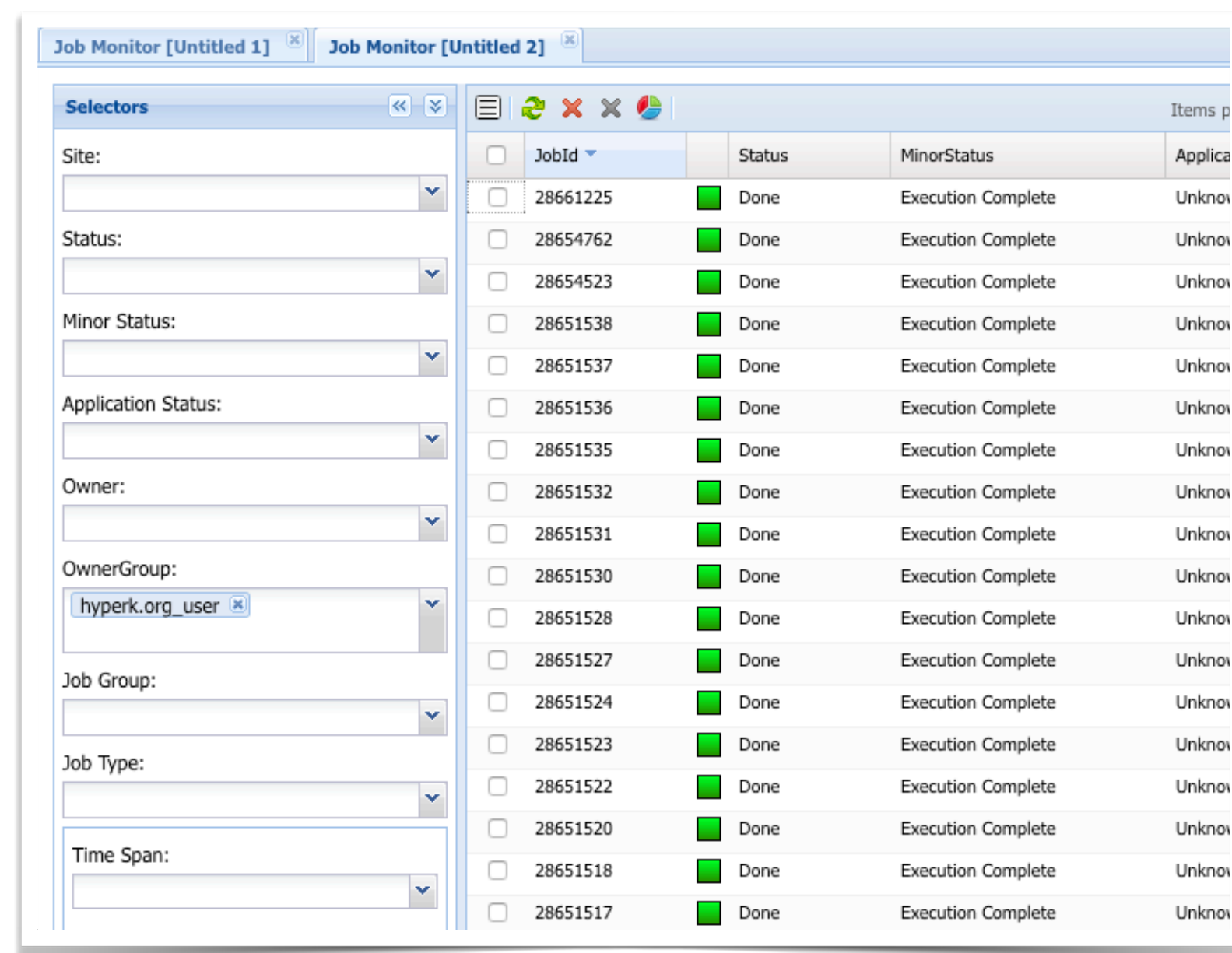
Software containerized and shared via CVMFS





DIRAC = Distributed Infrastructure with Remote Agent Control  
Interface to grant access to resources from heterogeneous sites with  
storage (SE) and/or computing elements (CE)  
Catalog with informations and location of files on SE  
“Unique queue” for submission and job config. via JDL file/Python API

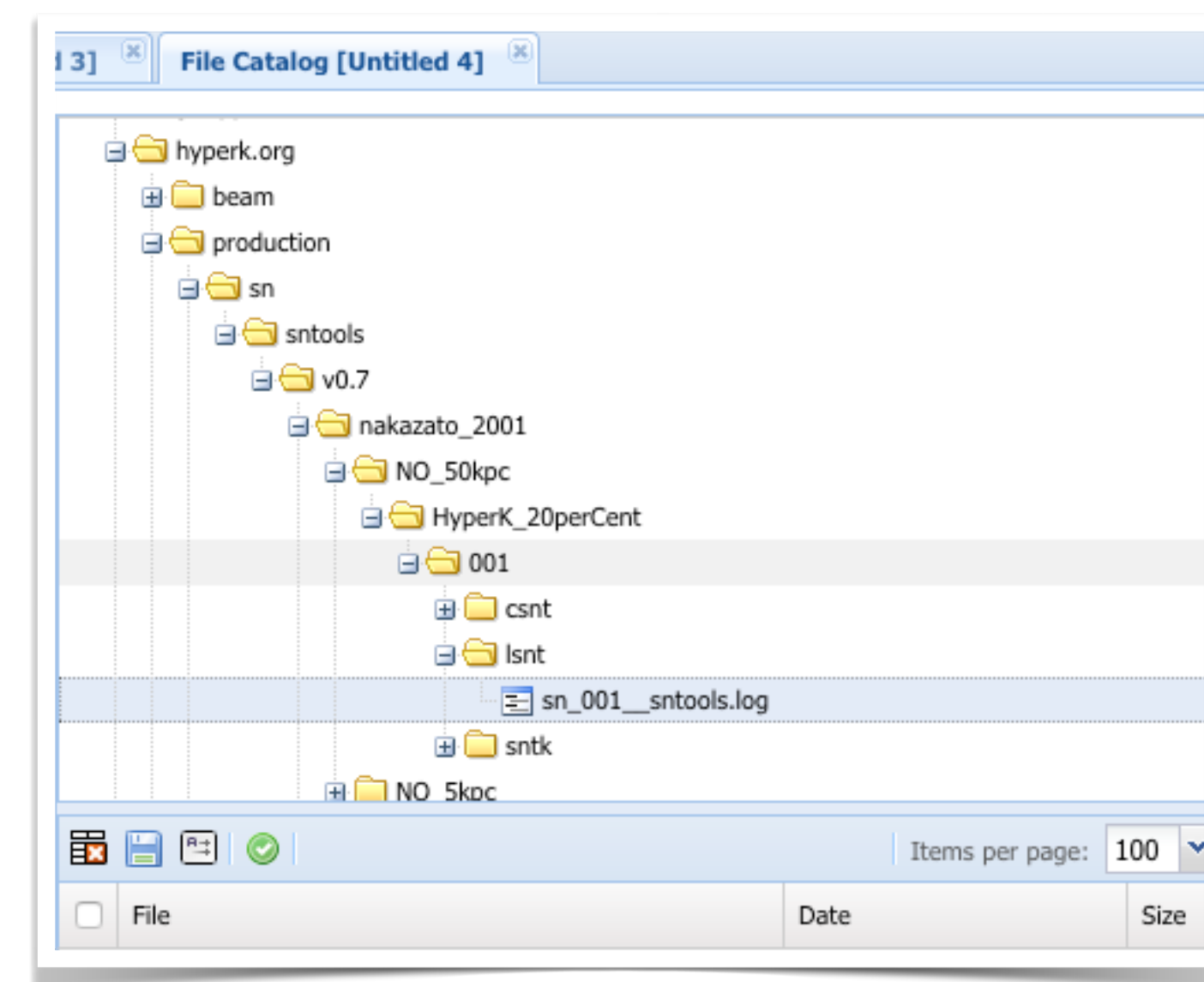
### Job monitoring



The Job Monitor interface displays a table of job statuses. The table has columns for JobId, Status, MinorStatus, and Applica. The status for all jobs shown is 'Done'.

JobId	Status	MinorStatus	Applica
28661225	Done	Execution Complete	Unkno
28654762	Done	Execution Complete	Unkno
28654523	Done	Execution Complete	Unkno
28651538	Done	Execution Complete	Unkno
28651537	Done	Execution Complete	Unkno
28651536	Done	Execution Complete	Unkno
28651535	Done	Execution Complete	Unkno
28651532	Done	Execution Complete	Unkno
28651531	Done	Execution Complete	Unkno
28651530	Done	Execution Complete	Unkno
28651528	Done	Execution Complete	Unkno
28651527	Done	Execution Complete	Unkno
28651524	Done	Execution Complete	Unkno
28651523	Done	Execution Complete	Unkno
28651522	Done	Execution Complete	Unkno
28651520	Done	Execution Complete	Unkno
28651518	Done	Execution Complete	Unkno
28651517	Done	Execution Complete	Unkno

### File catalog





Agents: perform periodical actions (data transfer, check DB...)

Workflow: defines a set of executables to be run one after the other

→ useful workflow: calibration.exe -> reconstruction.exe -> analysis.exe

→ use the output of an app as input of the next one

Transformation system: triggers a series of actions on each file that has a given set of metadata

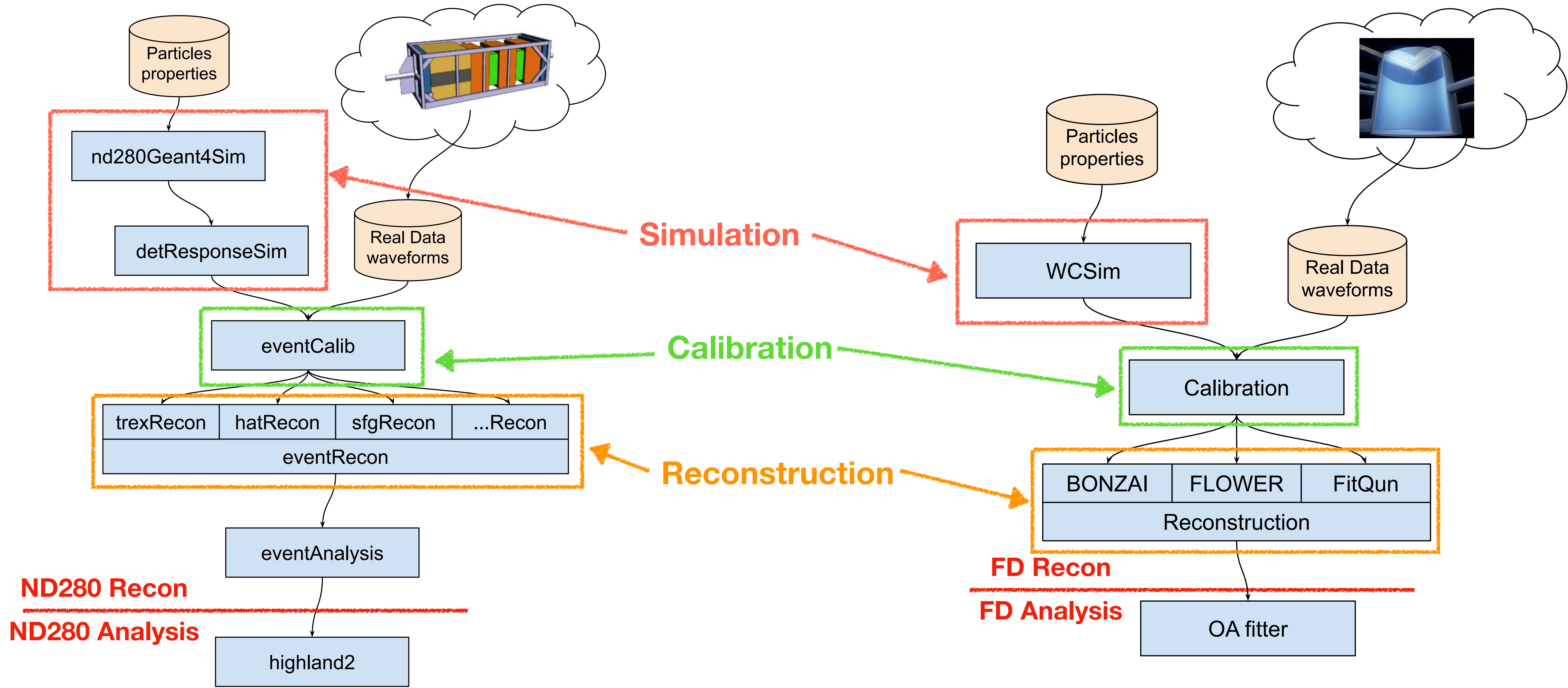
→ runs one or several workflows within jobs

→ perfect for defining productions on (sub) datasets !

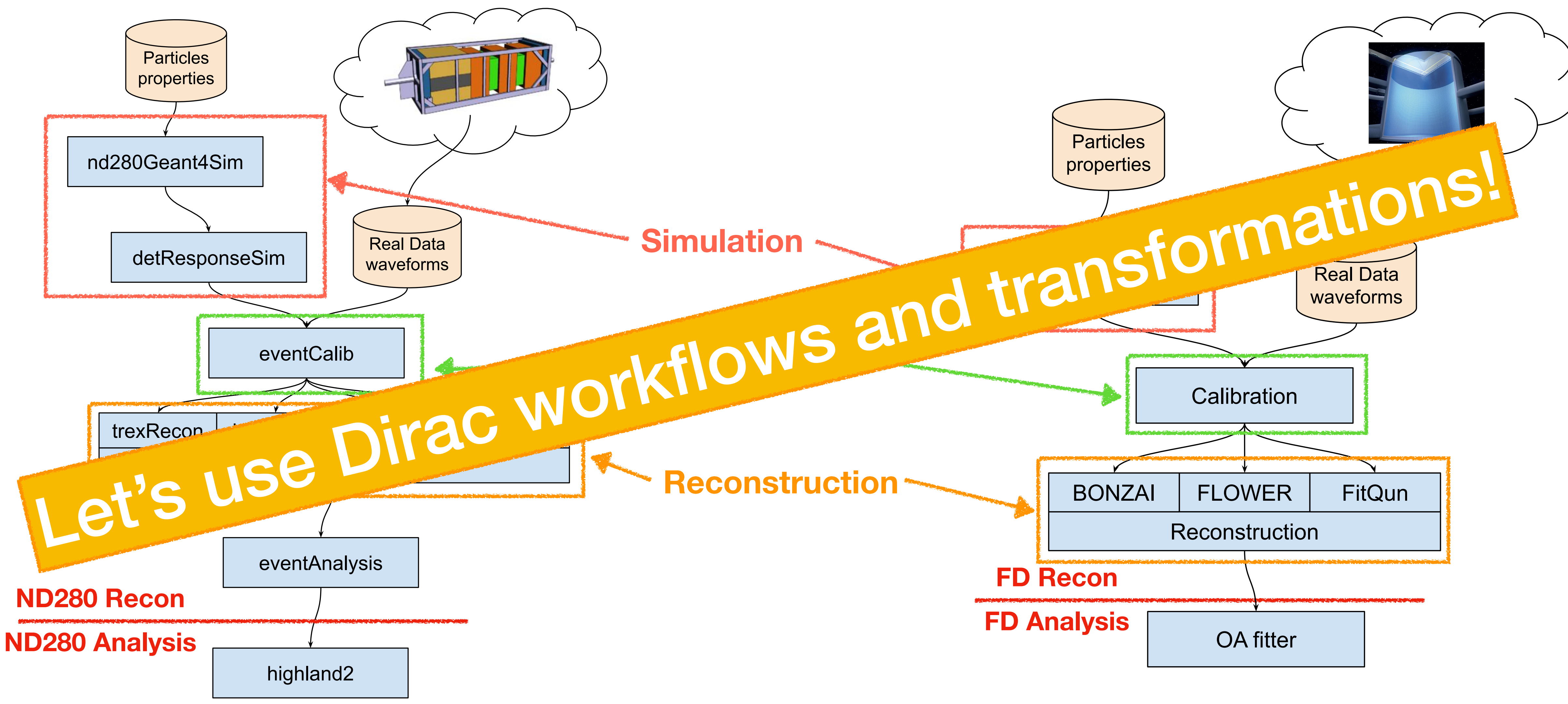
→ interface with the DIRAC data management system (metadata etc)

...











Base tools

Calibration

Utilities

Reconstruction

Simulation

Dependencies (ROOT, G4...)

Many packages being developed

“Old” dependencies (Fortran, ROOT5...)

Strong interdependency of the packages

Manually run tests during development

Code quality checks?

Need an “automated” way to build, test and deploy each package and the software stack in a controlled environment !

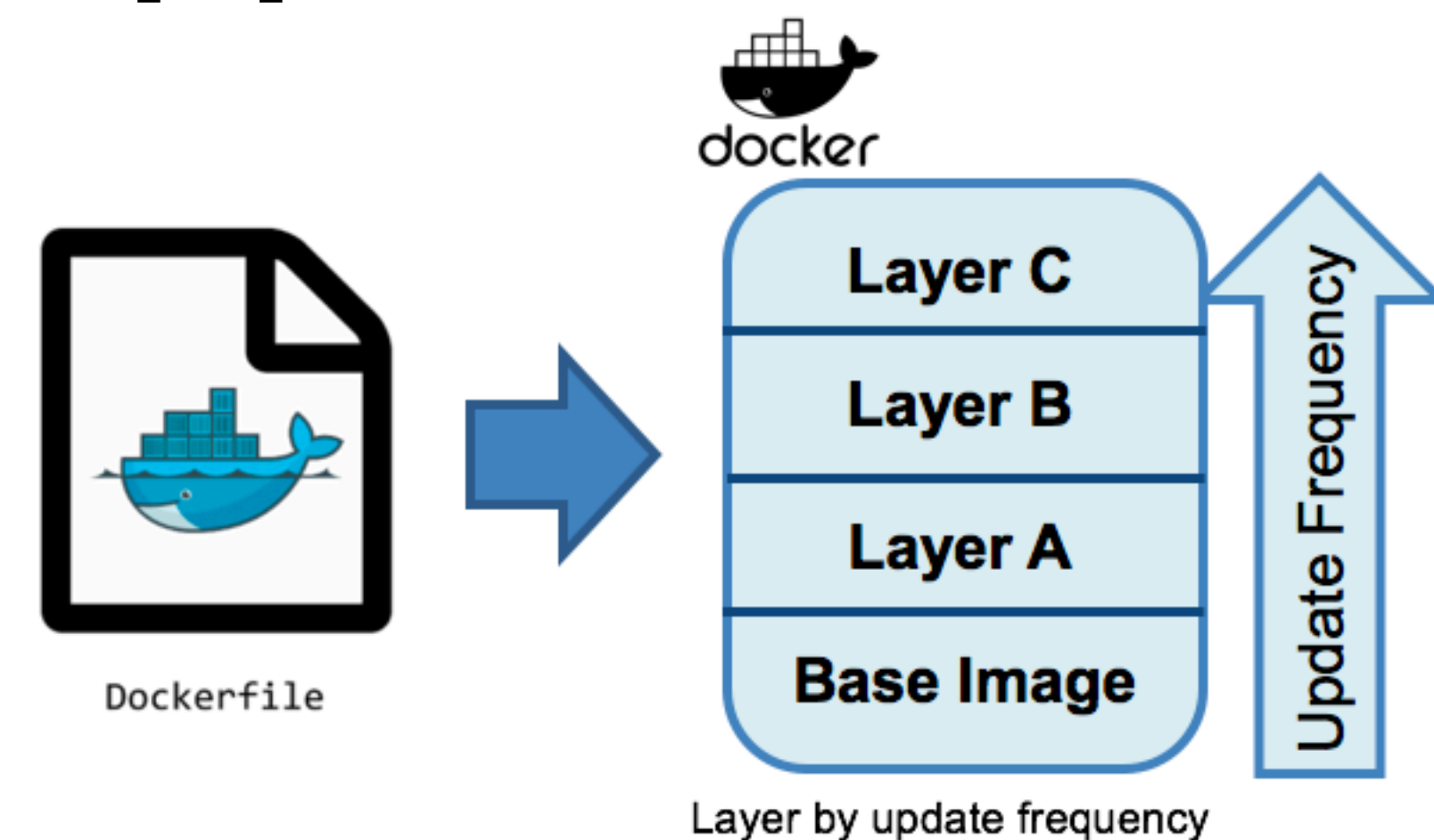


# Containers: “Layer cake” approach

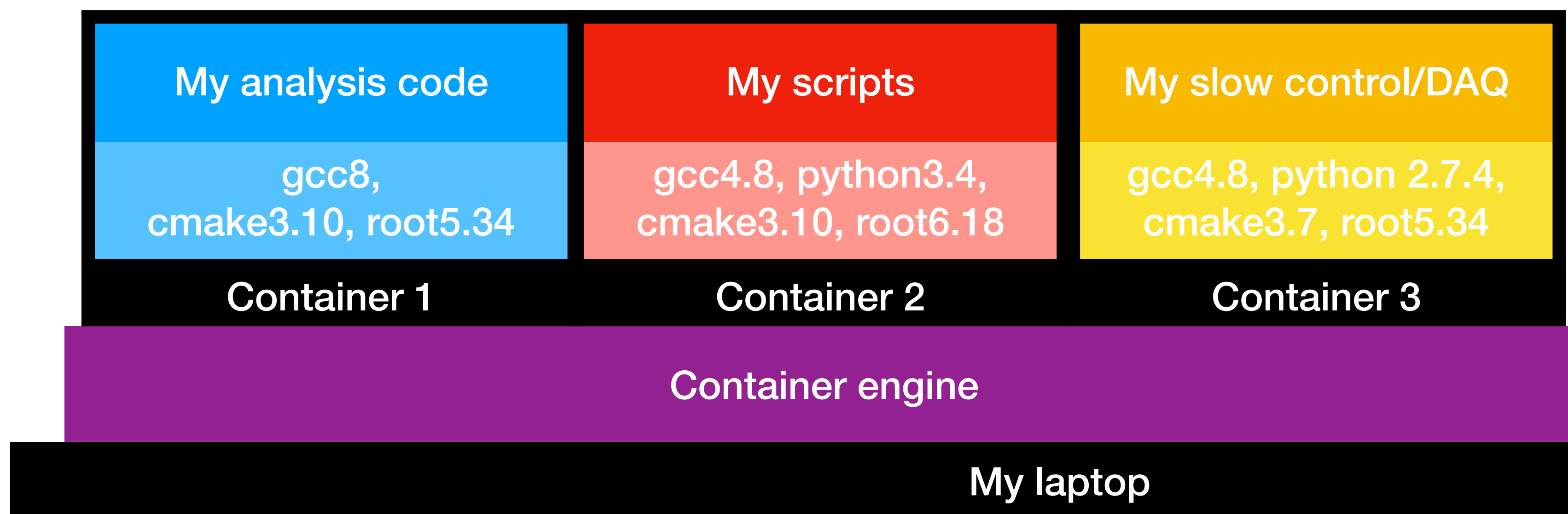
“Containers are a method of operating system virtualization that allow you to run an application and its dependencies in resource-isolated processes.”

<https://aws.amazon.com/containers/>

→ Containers are not virtual machines!



<https://openliberty.io/>





# Why using containerized code?

Standard container “methods”: Singularity & Docker

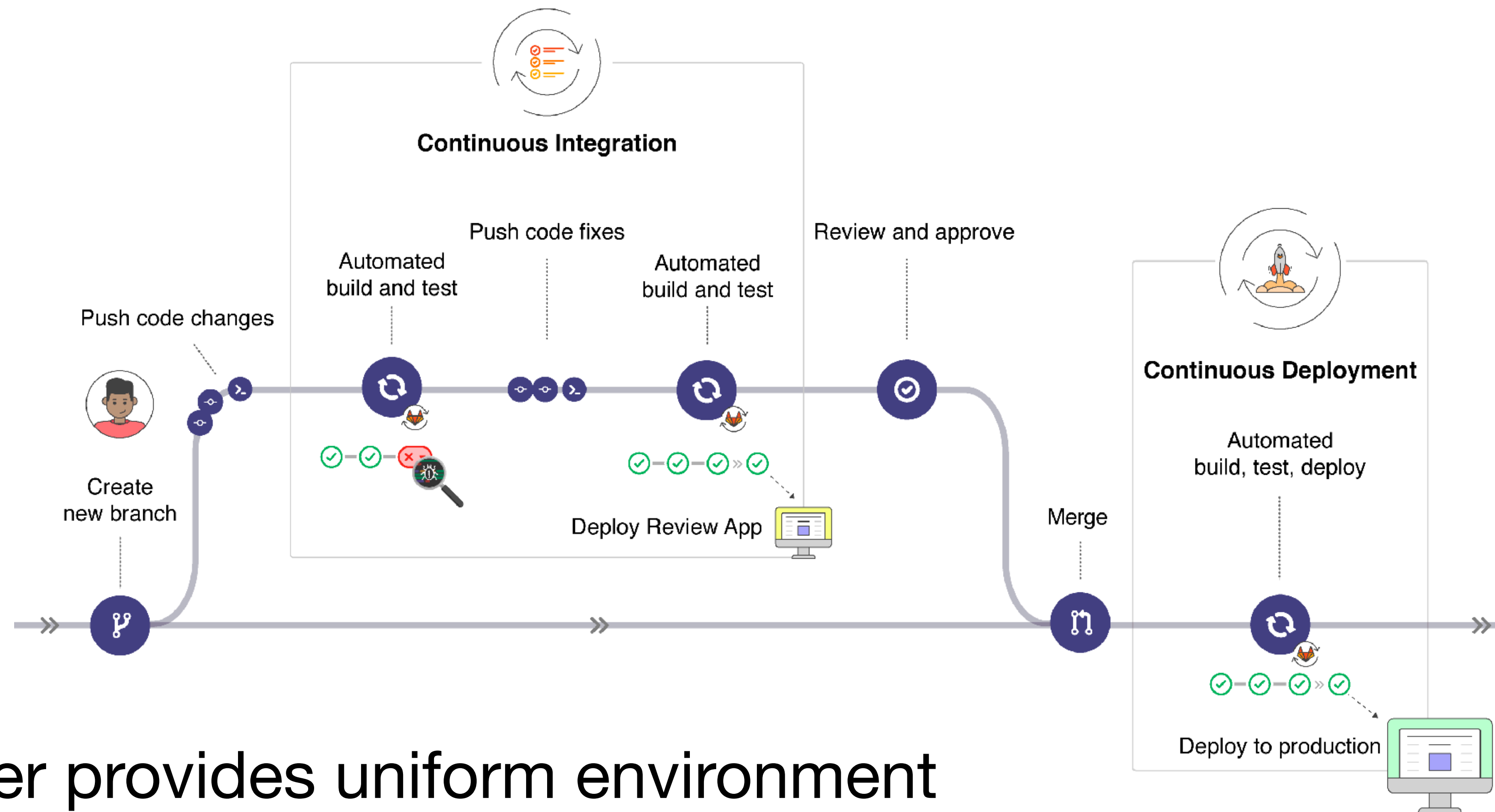
Provides:

- **uniform** development and production environment
- **reproducibility** of installation and code testing
- portability
- processes isolation
- networking **control** capabilities
- dependencies management while isolating package code
- facilitate software packaging and **sharing** (using images)
- better control over **resource usage** and dynamic resource allocation

Great for **Continuous Integration** and **job submission** on grids!



Well-suited for versioning systems like Git



→ Docker provides uniform environment

*Intensive usage in software industry and fundamental physics experiments*



**Basically every experiment** e.g. ATLAS, LHCb, T2K, Project 8

Integrated in most of Git frameworks e.g. Gitlab, Github ([gitlab.in2p3.fr](https://gitlab.in2p3.fr))

“Runners” setup on hosting server or any distant machine

T2K: 1 at t2k-ci-p@UWarsaw and 1 ccosvms0006@CC-IN2P3

Gitlab stores produced Docker images

Implementation of complex logics

Possibility of nightly (or “on-event”) builds of packages and stacks

Build code within several containers (CENTOS, Debian...)

Quick test of built code and/or run extended validations depending on conditions



Base tools

Calibration

Utilities

Reconstruction

Simulation

Dependencies (ROOT, G4...)

## Problems:

Many packages being developed  
“Old” dependencies (Fortran, ROOT5...)  
Manually run tests during development  
Code quality checks

## Solution:

For each package:

- one Dockerfile
- one CI configuration file (identical)

Automated builds for merge requests and releases  
Test and validation during merge requests and releases  
Building and storing images for  
Testing: individual packages  
Production: entire software stack



Base tools

Calibration

Utilities

Reconstruction

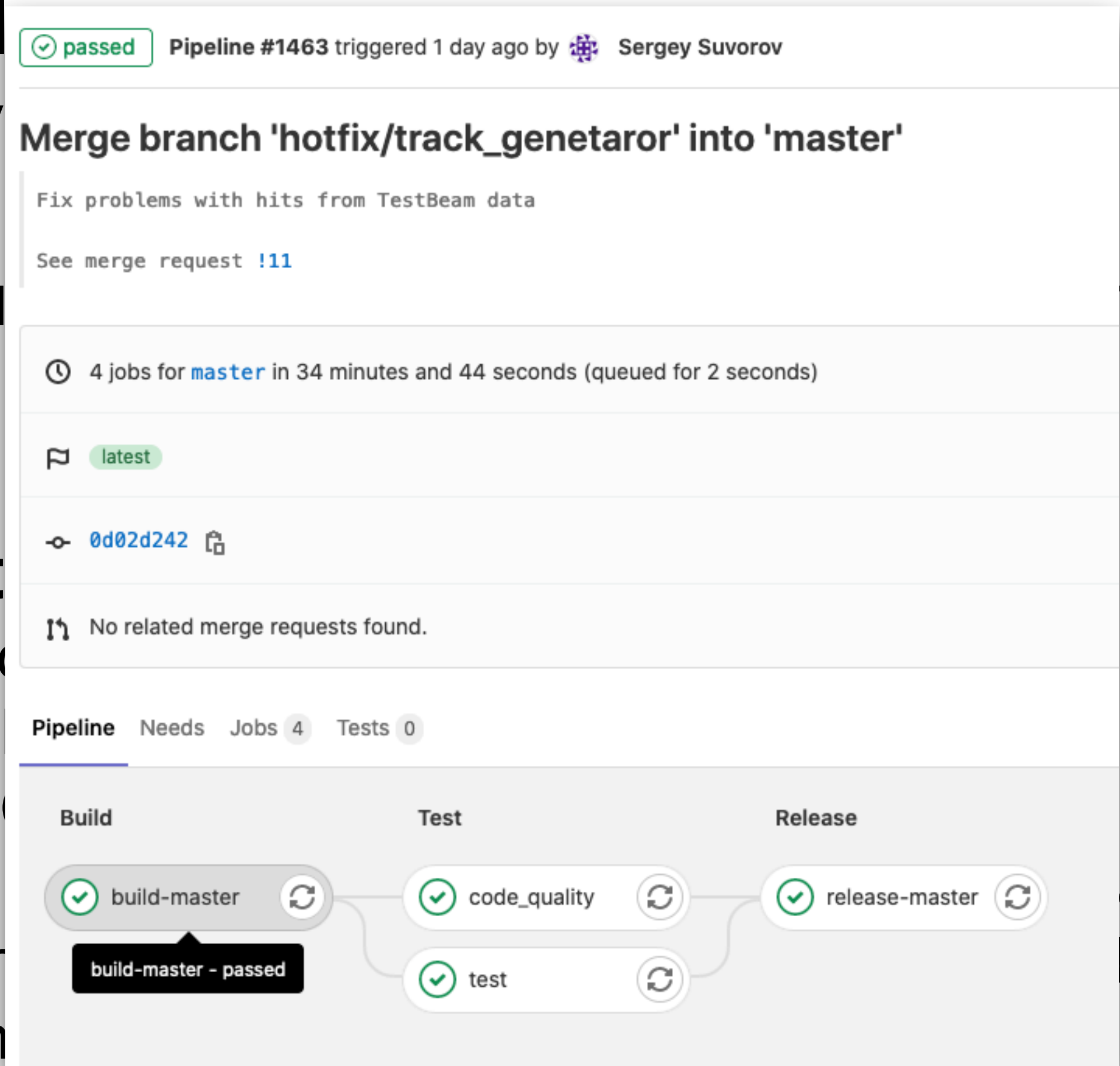
Simulation

Dependencies (ROOT, G4...)

Problem  
Many  
“Old”  
Manual  
Code

Solution  
For each  
- one  
- one  
Automated  
Test and  
Building

Testing: individual packages  
Production: entire software stack





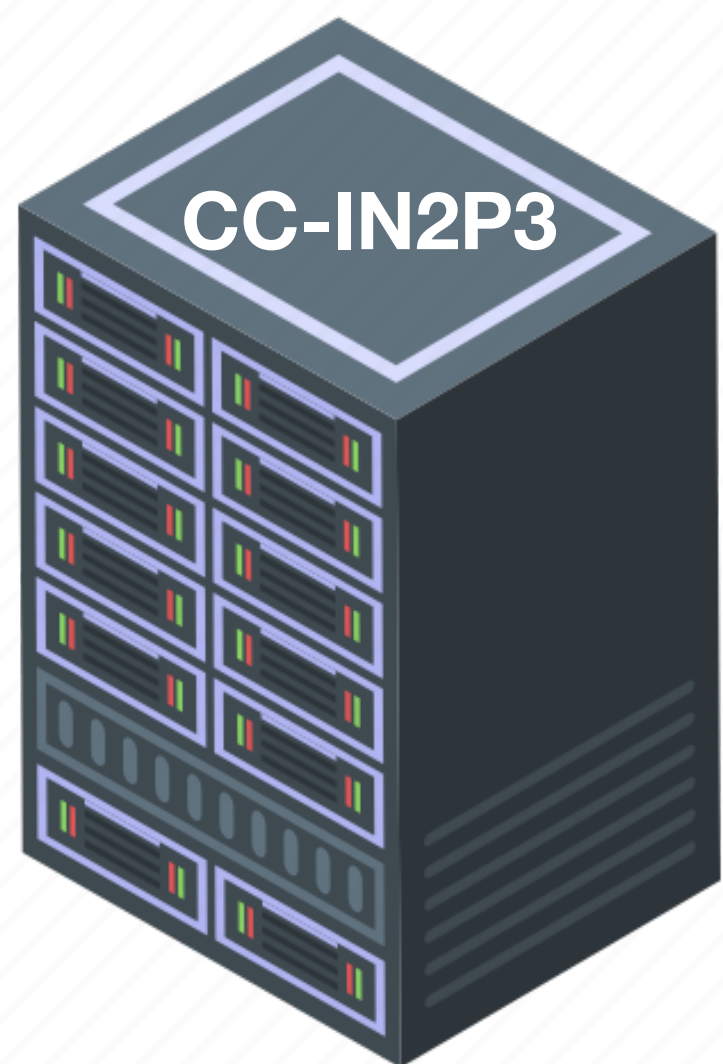
Useful tools for developer:

- don't need to install all dependencies
- don't care about the OS flavor (or if using MacOS)

**Example:** entire ND280 software stack in 3 commands and 10 seconds

```
~ > docker login git.t2k.org:8088
Authenticating with existing credentials...
Login Succeeded
~ > docker pull git.t2k.org:8088/nd280/framework/nd280softwaremaster:14.0
14.0: Pulling from nd280/framework/nd280softwaremaster
Digest: sha256:0ef00a64d9e783604c3c686913468f0c022ff80d6c3495707e5254c5ed022428
Status: Image is up to date for git.t2k.org:8088/nd280/framework/nd280softwaremaster:14.0
git.t2k.org:8088/nd280/framework/nd280softwaremaster:14.0
~ > docker run --rm -it git.t2k.org:8088/nd280/framework/nd280softwaremaster:14.0
[root@beee8e979a6e nd280SoftwareMaster_14.0]#
```

Use docker-compose for volume mounting and special configuration

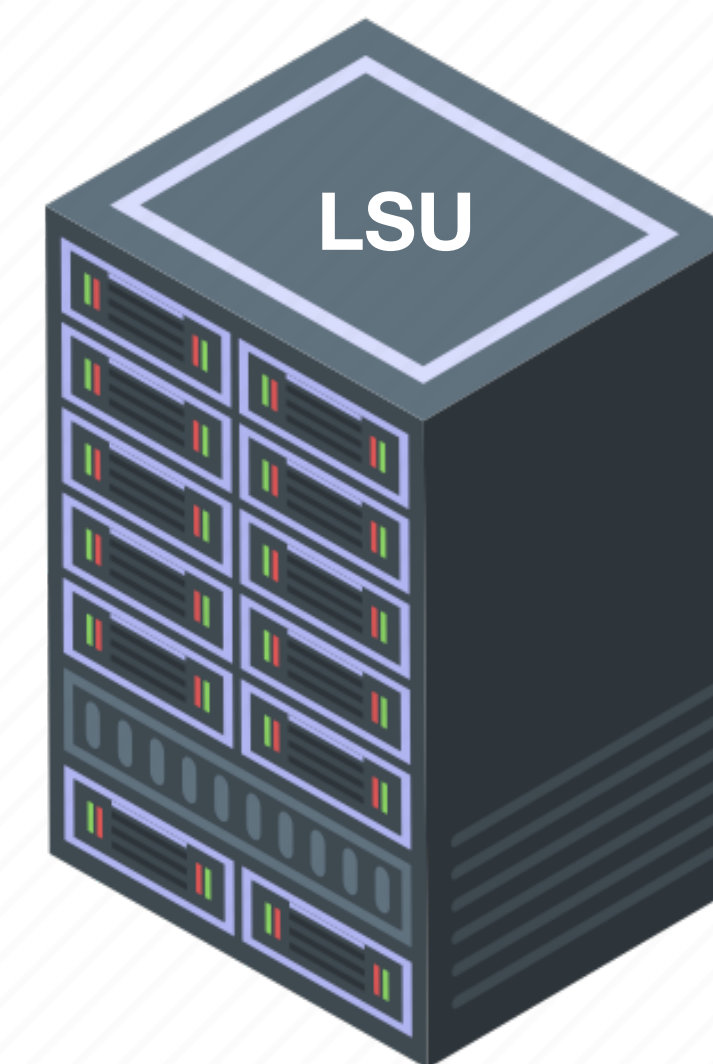


People want most recent OS so I will use CENTOS-7



People want something stable, so we should use CENTOS-6

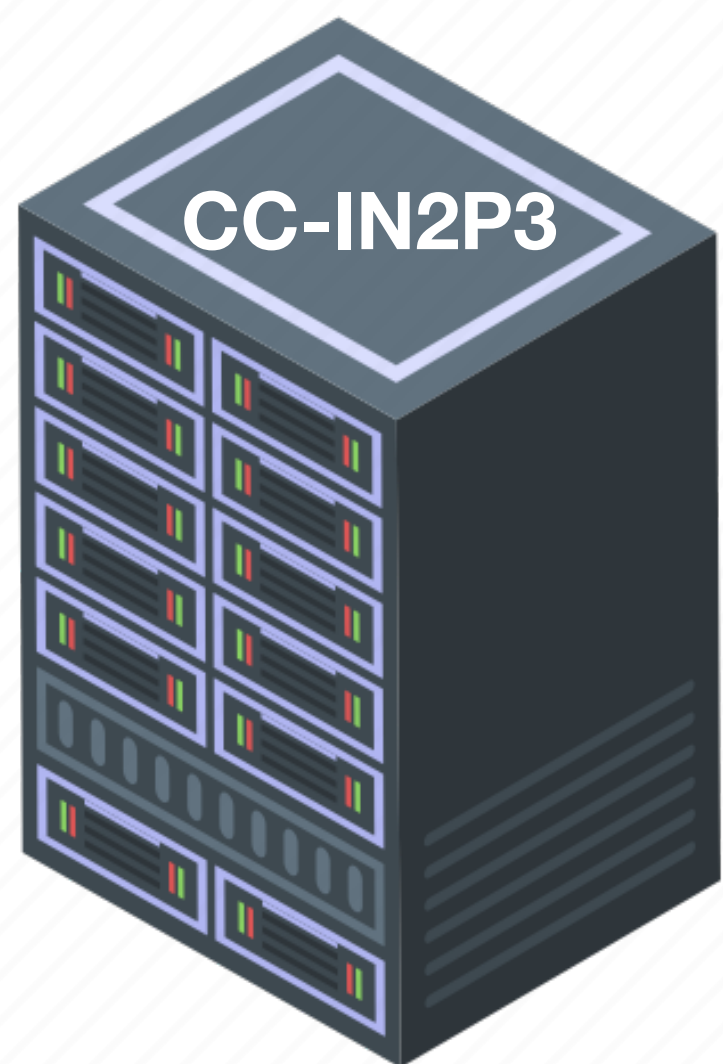
Meanwhile, at a cluster far far away...



I have some machines with Ubuntu, you should use them...

Oh and also some on Manjaro, because I believe this is the best OS ever...



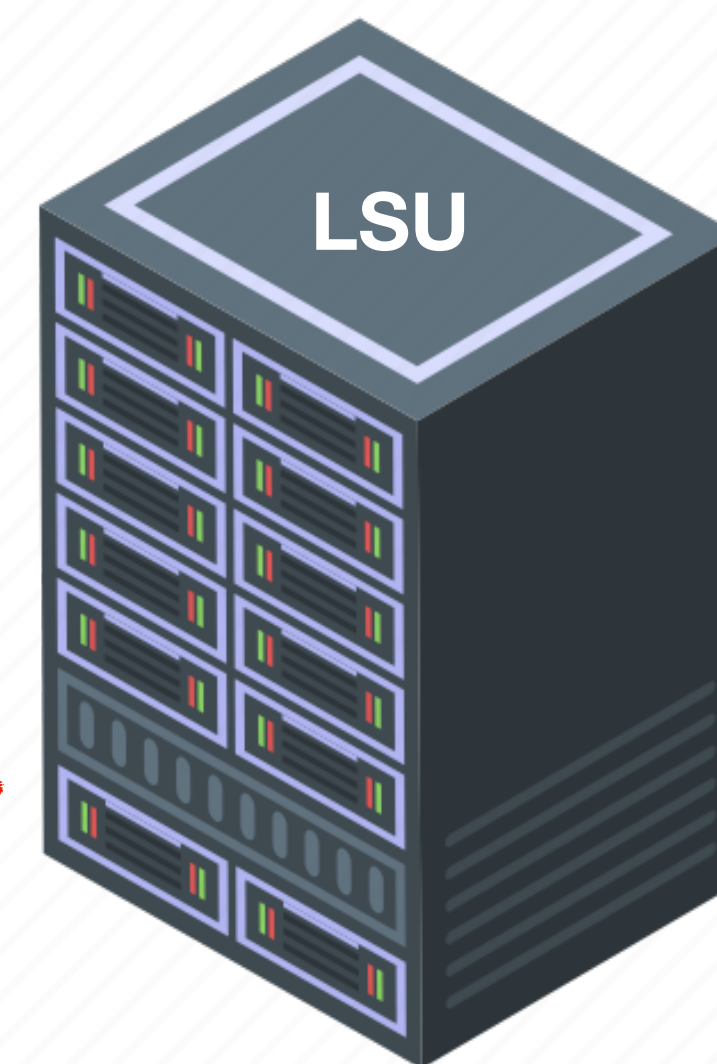


People want most recent OS so I will use CENTOS-7



People want something stable, so we should use CENTOS-6

Meanwhile, at a cluster far far away...



I have some machines with Ubuntu, you should use them...

Oh and also some on Manjaro, because I believe this is the best OS ever...



T2K/HK/xxx computing people

## Job environment

Starting point: Docker image of software stack (see previous slides)

Conversion of software stack image to Singularity image\*

Upload image on CVMFS (for caching)

Dirac finds an available node “somewhere”

Job starts a container and run analysis/simulation/... → **focus on science!**

## Job submission via DIRAC

Image with DIRAC client\*\* installed, additional configuration files and scripts

Job submission and data retrieval from File Catalog

Sharing files with host via docker-compose → **plug-and-play!**

\* Conversion from docker to singularity maintained by Singularity people: <https://github.com/singularityhub/docker2singularity>

\*\* An example: <https://github.com/mariojmdavid/docker-dirac>





CC-IN2P3 Tier1 for LHC (WLCG)

→ infrastructure, expertise available

Low-rate data stored on tapes

Disk and CPU for productions

Database management

Two scenarii considered:

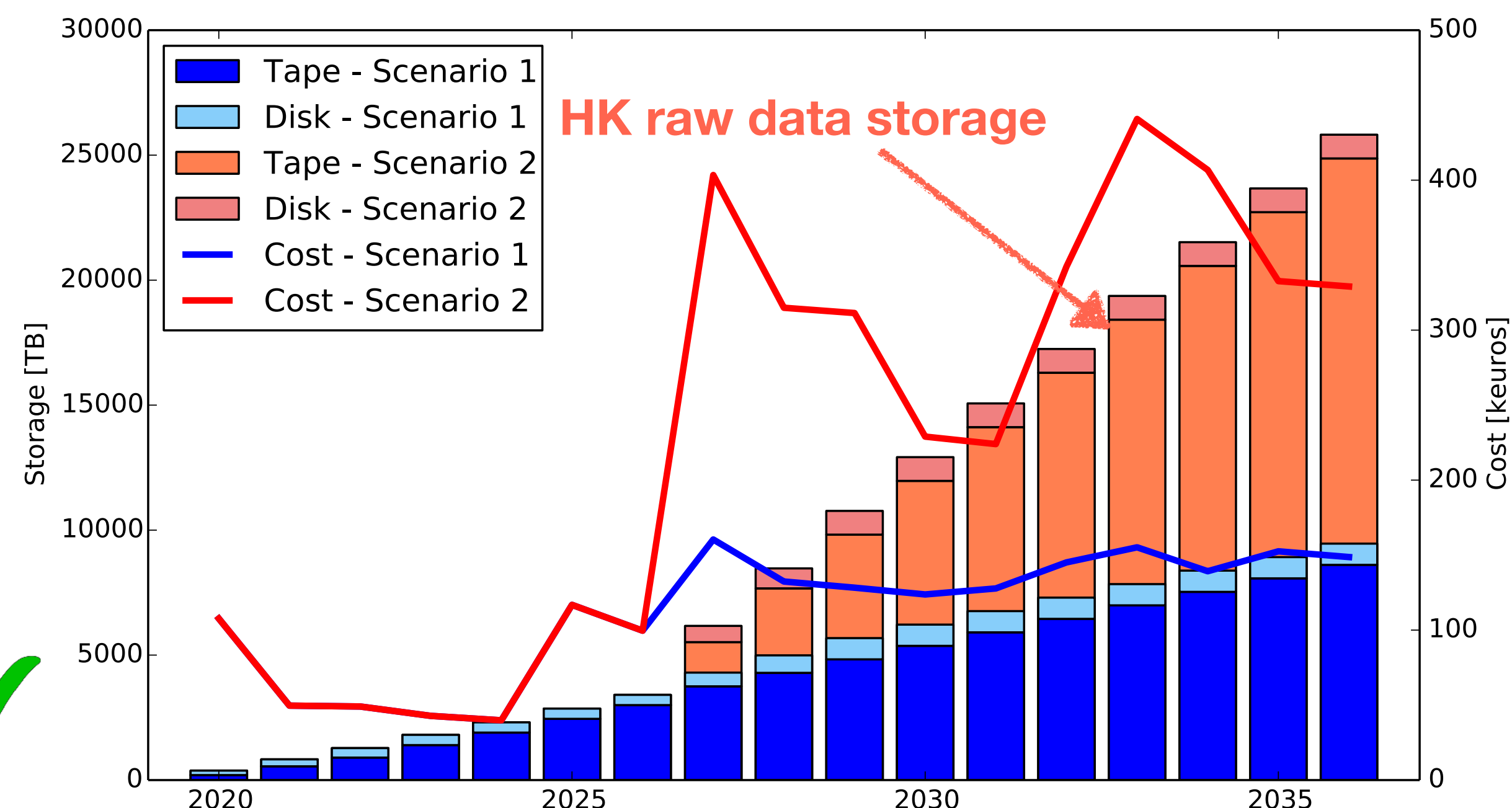
1. ND280 data storage

2. Near and Far detectors data storage

First step: CC-IN2P3 as T1 site for T2K

→ integration of CC as grid site into GridPP ✓

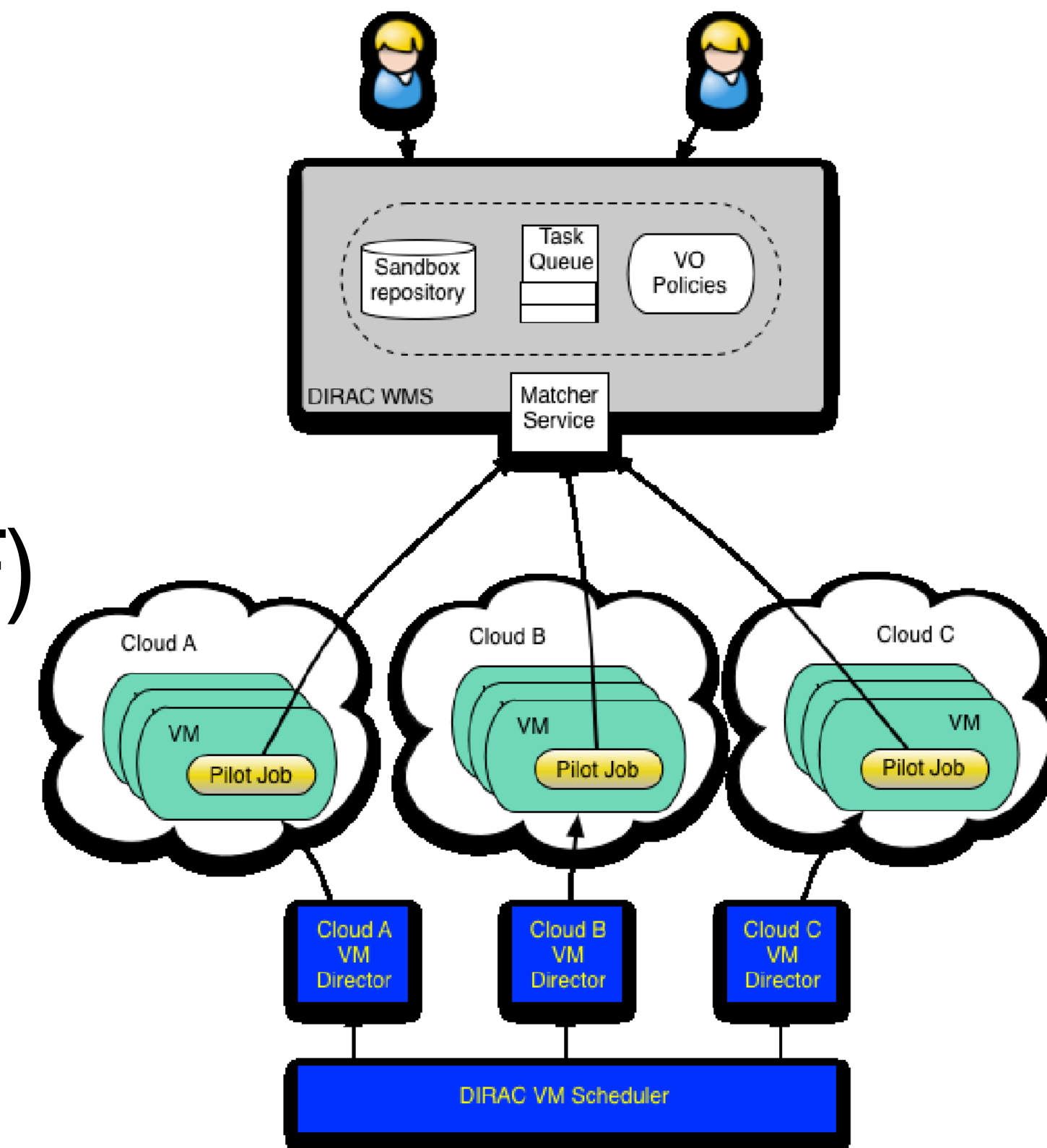
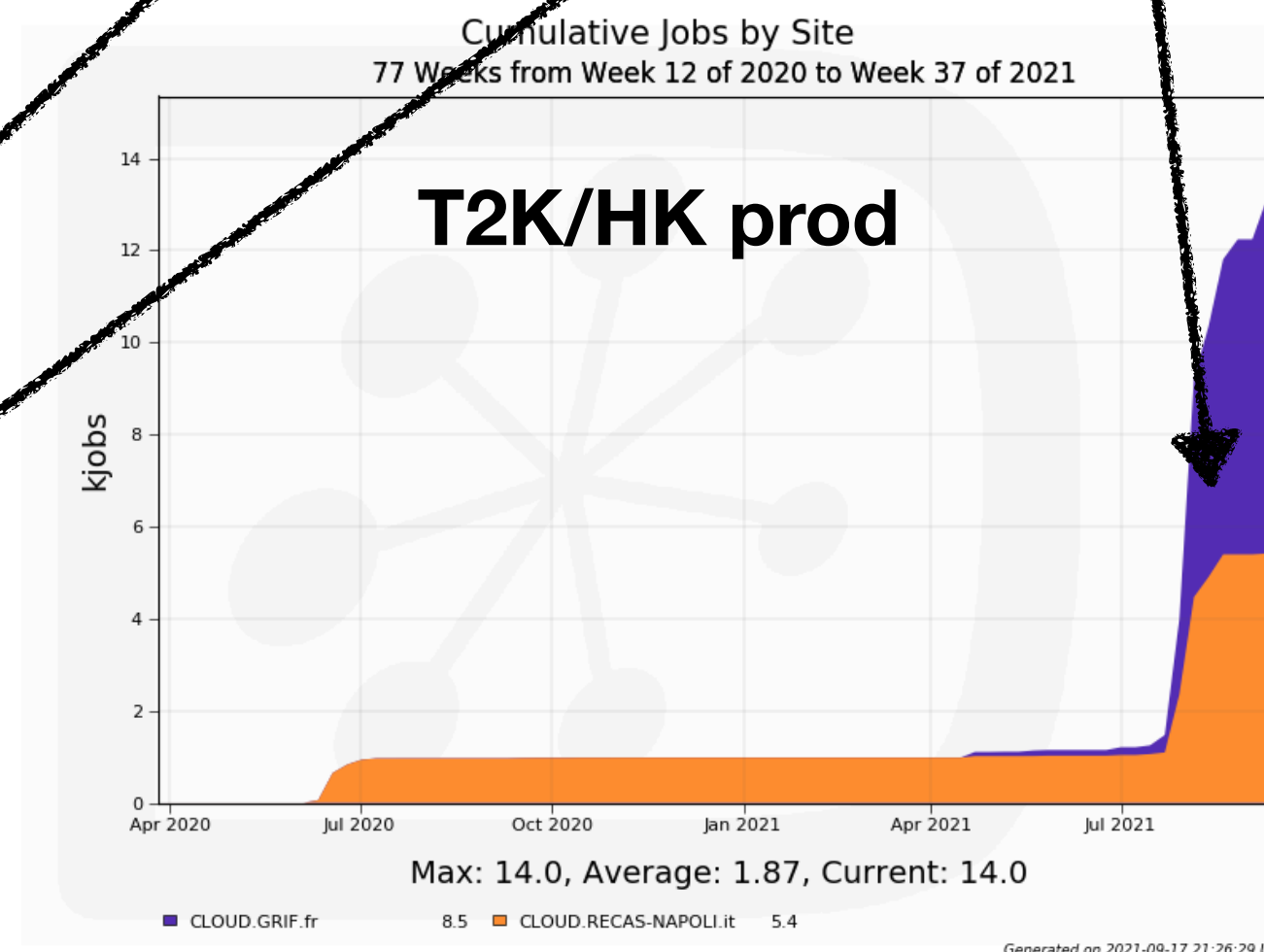
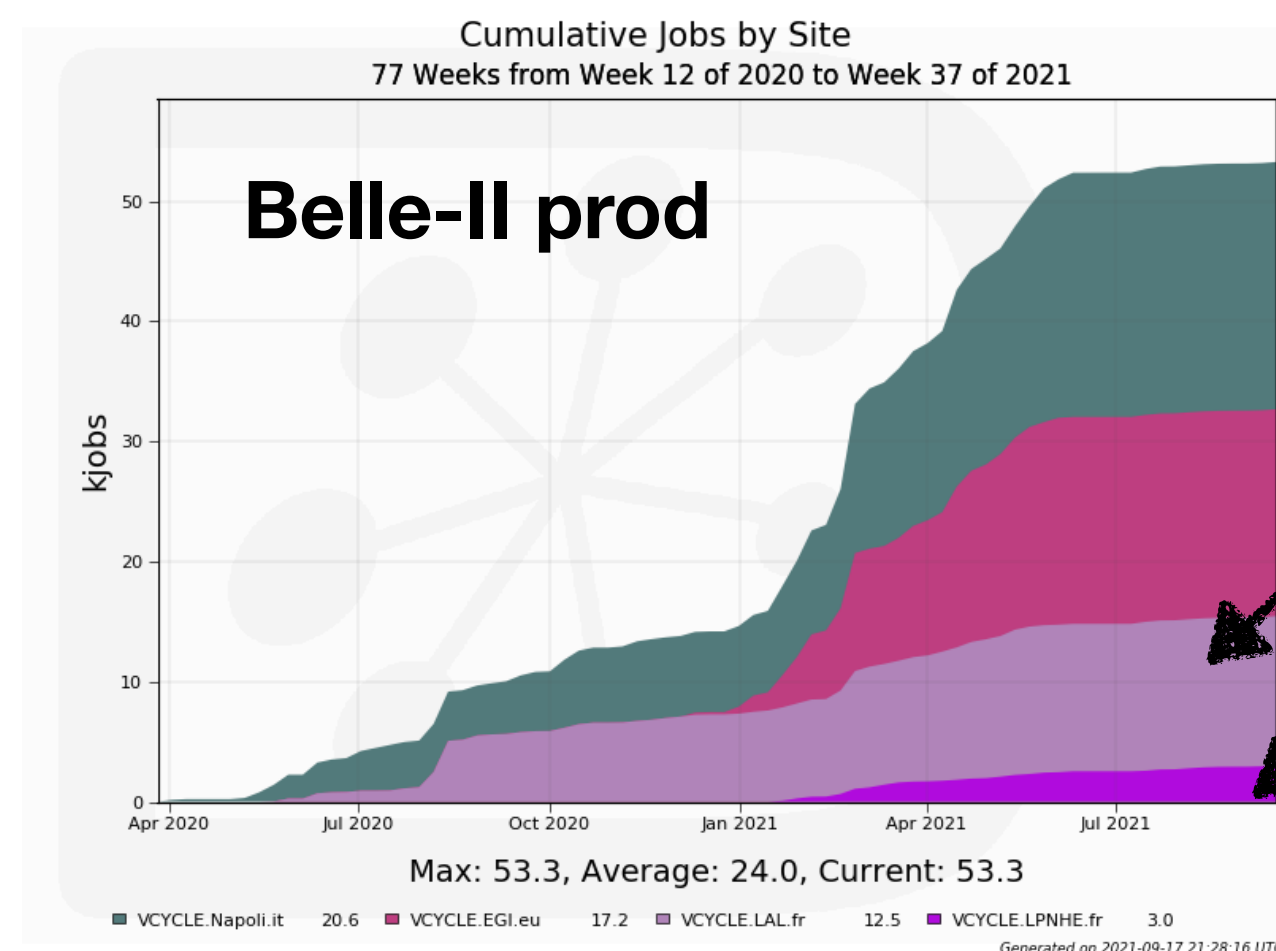
→ disk allocation and data transfer



**First production for T2K and HK at CC-IN2P3 completed**

WP5: Development of joint cloud demonstrator between T2K/HK and Belle-II

- “VM director” instead of “Pilot director”: instantiate VMs which starts “pilot job”
- VCYCLE for VM lifetime: restarted when job done
- Authentication via EGI
- Integrated clouds (INFN-Napoli, LAL, LPNHE, GRIF)



→ Talk at **EGI2020**



T2K/HK computing model is standard in particle physics experiments  
Tiers, distributed computing and storage, DIRAC...

Exploration of containers potential for various aspects

Software → uniform and extremely reproducible, storable for later use

Computing → not expensive, adaptable to clusters, job submission

DAQ → heterogeneous hardware, scalability

A lot of expertise and tools already available at CC-IN2P3

**All neutrinos experiments would highly benefit from these!**

# Backup



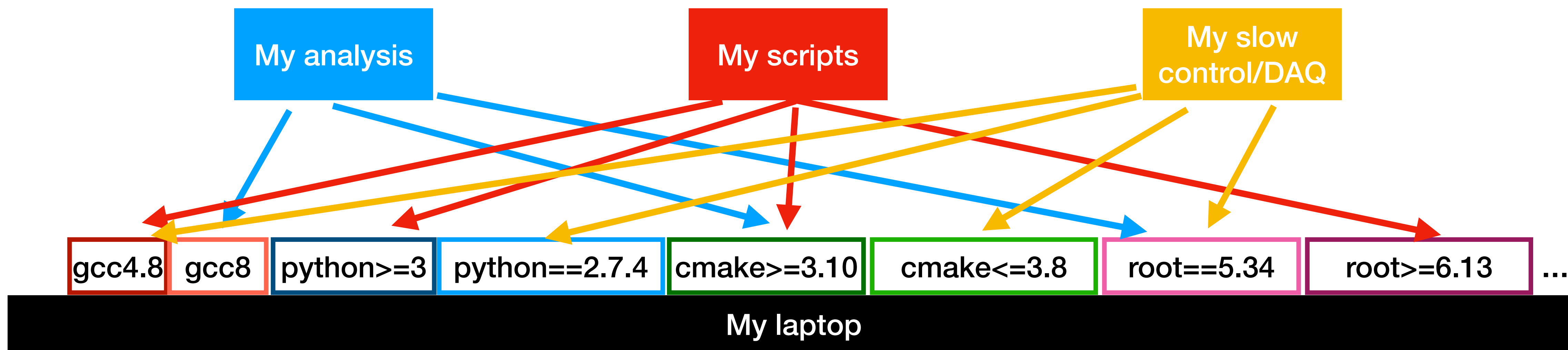
Conflicting dependencies between projects

Manage and deploy software on heterogeneous/networked systems

Deploy many applications (slow control, DAQ, web interface...)

Development on networked hosts

Development by multiple agents/groups



# Docker terminology

**Docker:** open-source project to create, deploy and run applications via containers

**Docker Inc.:**

- provides applications to run on Mac/Linux/Windows
- provides free hosting and automatic builds of images

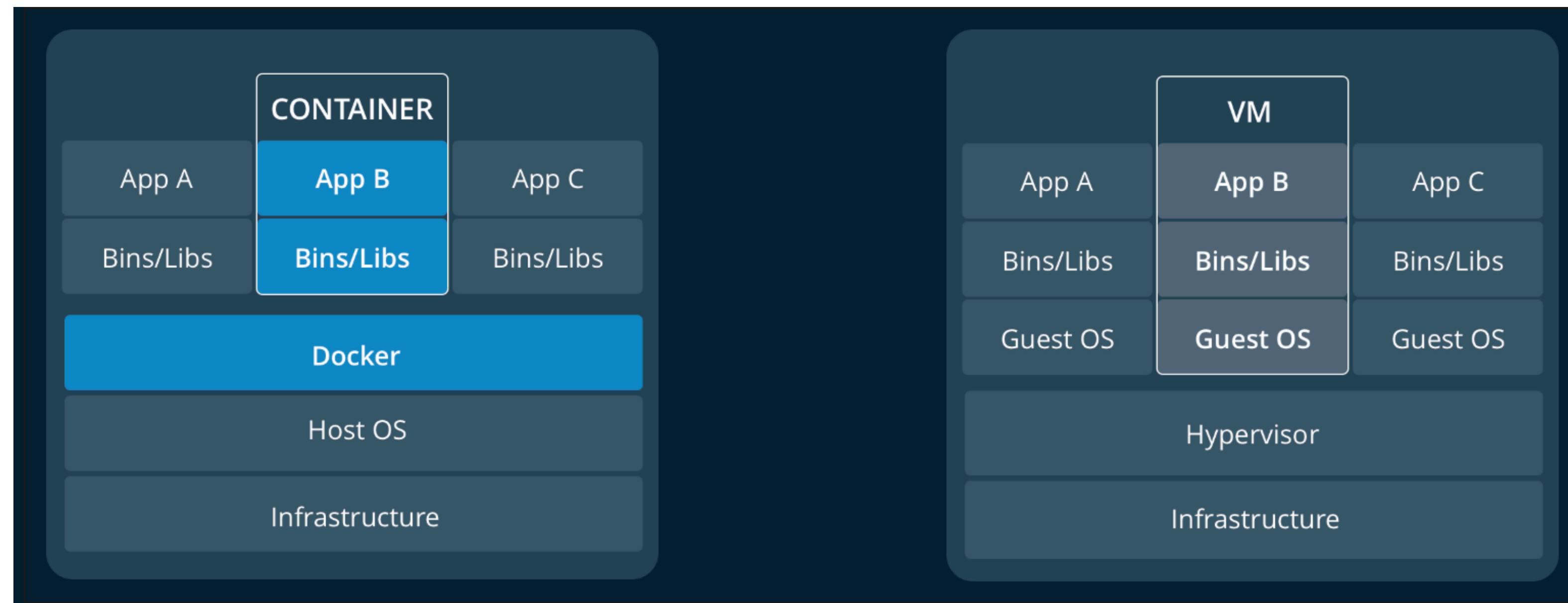
**Container:** self-container application easily deployable in an environment

**Container image:** compressed container used to create functioning containers

**Docker engine:** back-end of Docker software running on computing element (laptop, server...) and managing containers

**Docker client:** interface that communicates with the Docker engine





Container runs on Linux as a **process** and share host machine kernel

→ direct access to host resources

VM runs as “independent” **guest operating system**

→ virtual access to the host resources

→ VM need more resources than containers (CPU, memory, disk space)