Accelerated Computing for Physics: accelerators & containers

Andrea Sartirana

the acp project…

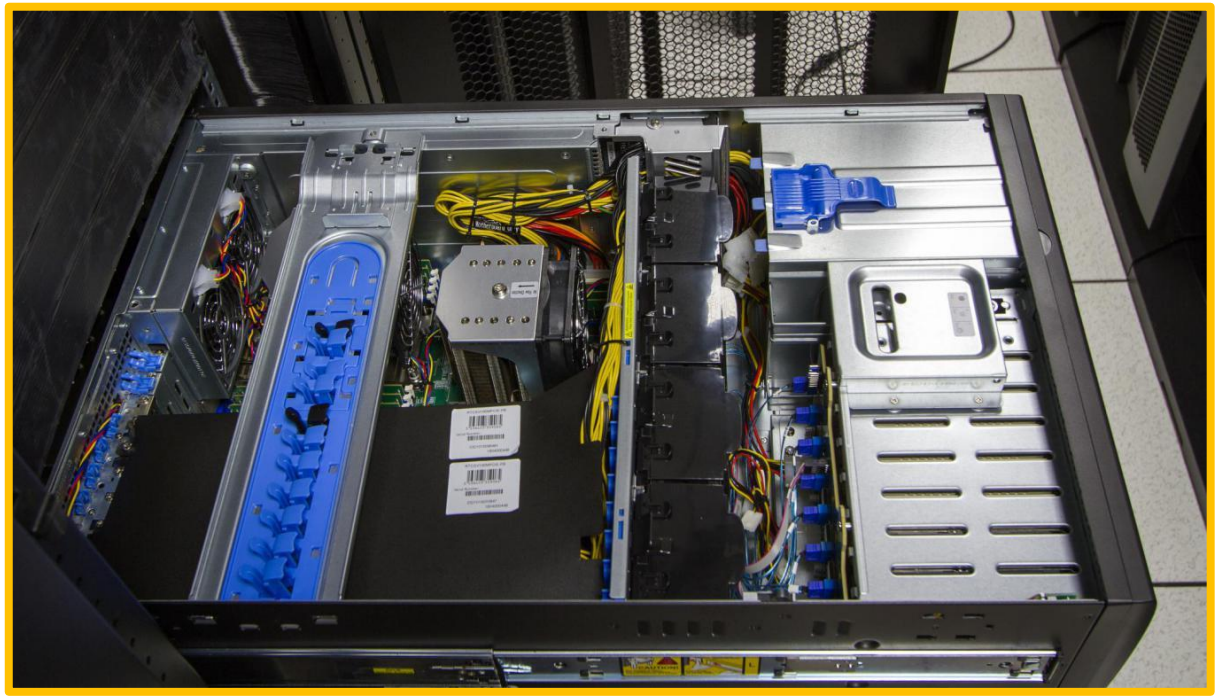
- **A**ccelerated **C**omputing for **P**hysics is a 3 years project approved in 2017 and **funded by P2IO** consortium
- **R&D** project about **computing on accelerated hardware** (GPUs, manycore, FPGA) for HEP and astrophysics
  - **porting** of application
    - directives, etc… ;
  - **adaptability** of applications to different hardware
    - code generation, … ;
  - **deep learning** methods
    - e.g. galaxy morphology classification;
  - customizable/portable environments via **containers**.

…the acp project.

- natural interaction with the **Decalog** master-project
  - **more local** to (the late) Paris-Saclay… but **beyond IN2P3**: IAS, IPNO, IRFU, LAL, LLR;
  - focused on **accelerators**. Deep learning;
- "**containerization**" part: a natural partner for **ComputeOps**
  - exchange of **results and experiences**
    - images, solutions, scripts, … ;
  - provide (accelerated) **pilot applications**
    - see next slides;
  - provides a **test accelerated HW platform**
    - ACP funding: purchase of a GPU and FPGA development platform.

the dev platforms.

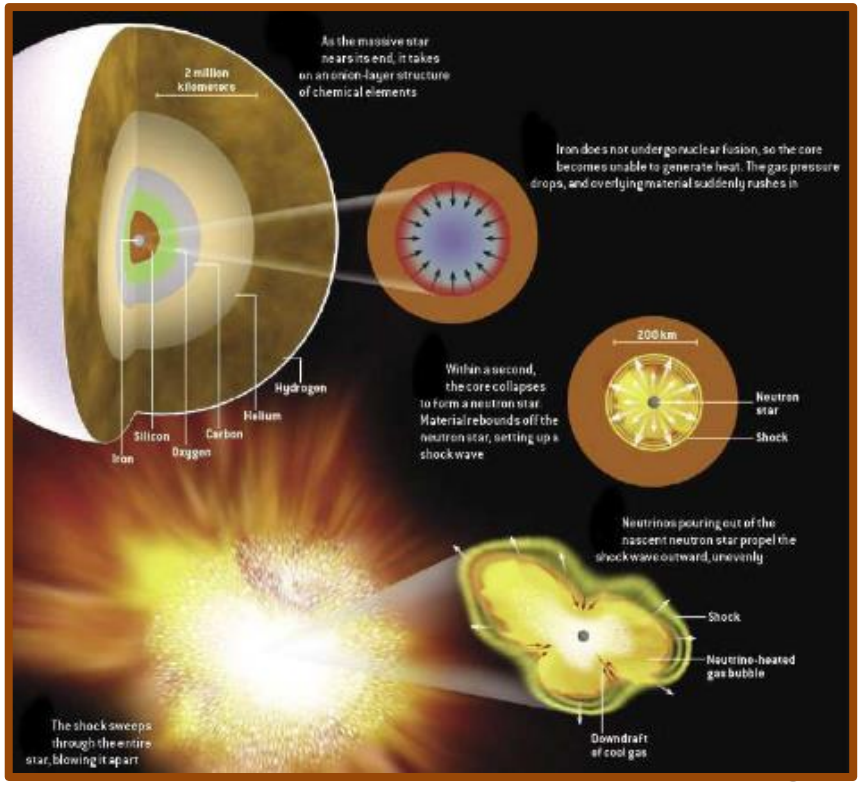
- The **old gridcl platform**
  - gridcl: (2012-2015) GPGPU/manycore R&D with OpenCL;
  - **heterogeneous** (and **outdated**) HW
    - 2 nodes with 2 nvidia K20, 1 node with 5 nvidia Titan, 2 nodes with 2 Xeon Phy, 1 node with 2 AMD FirePro (W9100);
- new **GPGPU platform**
  - 2 **Nvidia Tesla V100**, 16GB;
  - 2 Xeon Gold 6138, 20 cores, 192GB RAM;
- **FPGA** platform
  - **FPGA** + **AMD Epyc** CPU;
  - still *to be purchased*.



pilot applications…

Stellar electron capture rates

- finite-T Hartree-Fock and RPA using Skyrme interactions;
- *V. Lafage* and *F. Bastier* (IPNO);
- (very) old code "cleaned" and **GPU-ized**[*]
  - F90 with **OpenACC directives;**
- **tested on the new platform**
  - x8.7 speedup wrt 2011 on a M2090 GPU;
  - more comfortable with 16GB;
- currently running (& being developed) in **PGI-OpenACC singularity containers**
  - using debian-9 and sl6.



[*]https://indico.in2p3.fr/event/17206/contributions/64142/attachments/50038/63808/JII_GPU_2018.pdf

…pilot applications…

HPC Algorithms for high Resolution Detectors

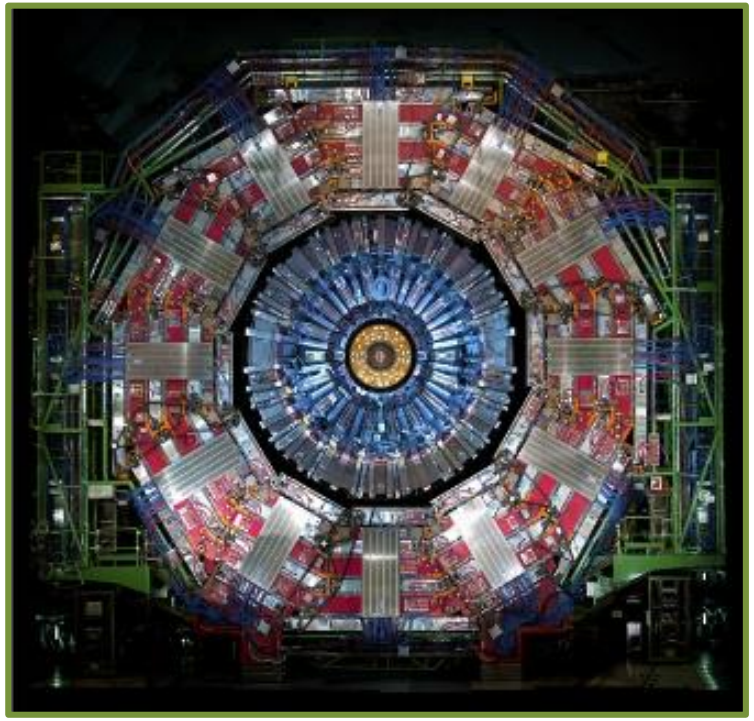
- dev. at **LLR** During **2018 Google Summer of Code** [*]
  - *Abhinav Kumar* (Mentors: *G. Grasseau et al*);
  - development still ongoing;
- **image reconstruction** in the **CMS HGCAL** (HLLHC upgrade)
  - **machine-learning** based on **Tensorflow** and ROOT;
  - optimized for running on extensible processor and GPU platforms;
- preparing a **containerized version**
  - provide everything with a Singularity image (not yet ready. *Need to clean and validate*);

[*] https://github.com/grasseau/HAhRD

…pilot applications.

CMS Matrix Element Method

- this code uses the Matrix Element Method for the **analysis of** the Higgs boson production with two top quarks (**ttH channel**) in the **CMS experiment**
  - dev. by *G. Grasseau*. GridCL proj.;
  - used in prod at CCIN2P3[*];
- based on **OpenCL and cuda**
  - adapted for **MPI…**;
  - … and **multi-GPU** (>30GPUs);
- to-do: see if we can create an **image** which is self contained (except CMSSW) and can **run on all platforms.**
  - containerization of openCL apps;



[*]https://indico.cern.ch/event/587955/contributions/2937584/

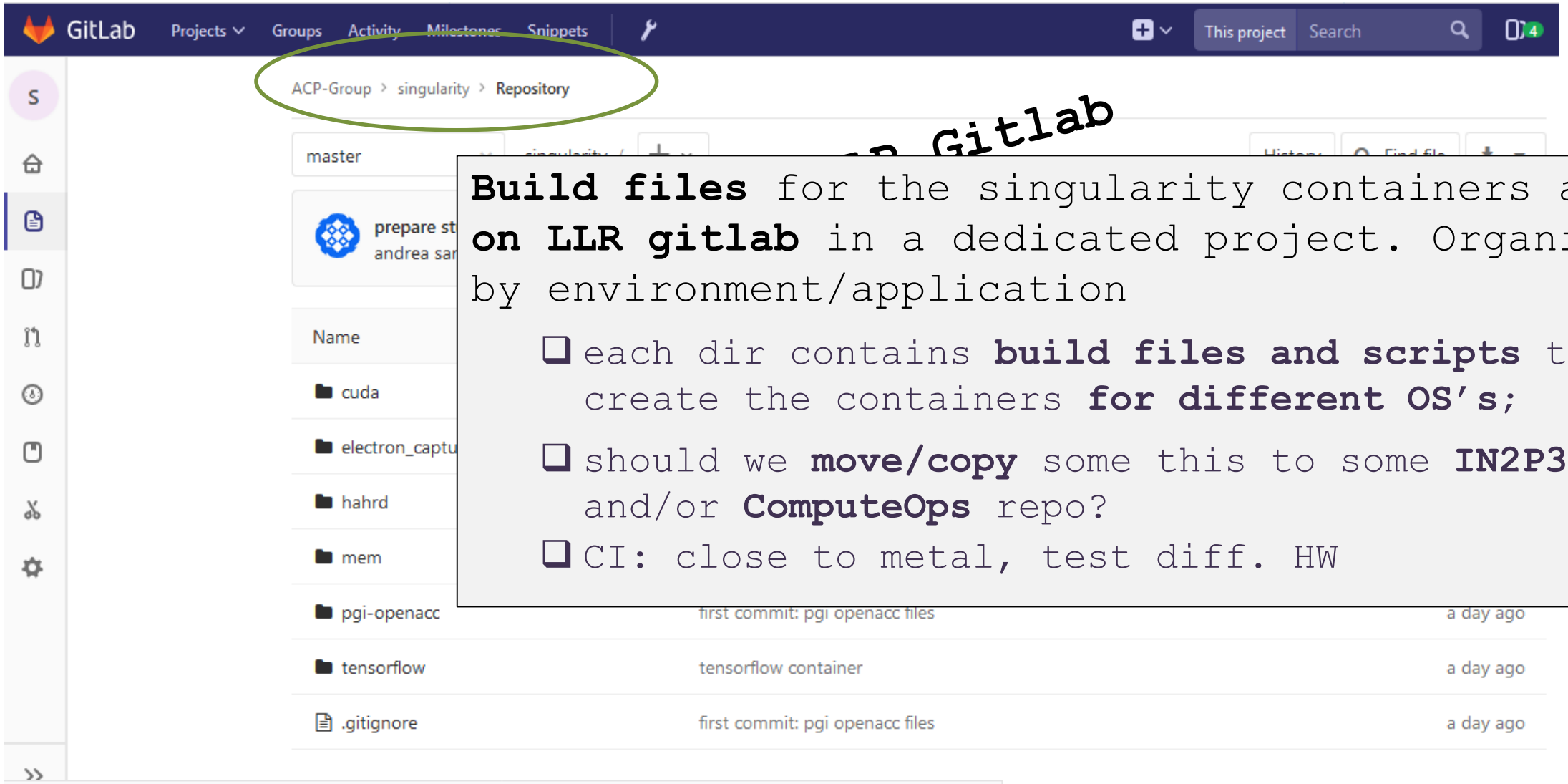
pilot applications++.

Smilei

- Particle-In-Cell code for plasma simulation
  - designed for high performances on super-computers;
  - applied to a wide range of physics studies;
- code hybrid MPI/OpenMP. Not (yet) running on GPU
  - optimized to explit vectorization;
- not, strictly speaking, ACP. Interesting pilot app. for ComputeOps
  - containerization: work in progress (to be tested on LLR HPC clusters).

[*]https://indico.in2p3.fr/event/17206/contributions/64186/attachments/50681/64788/JI.pdf

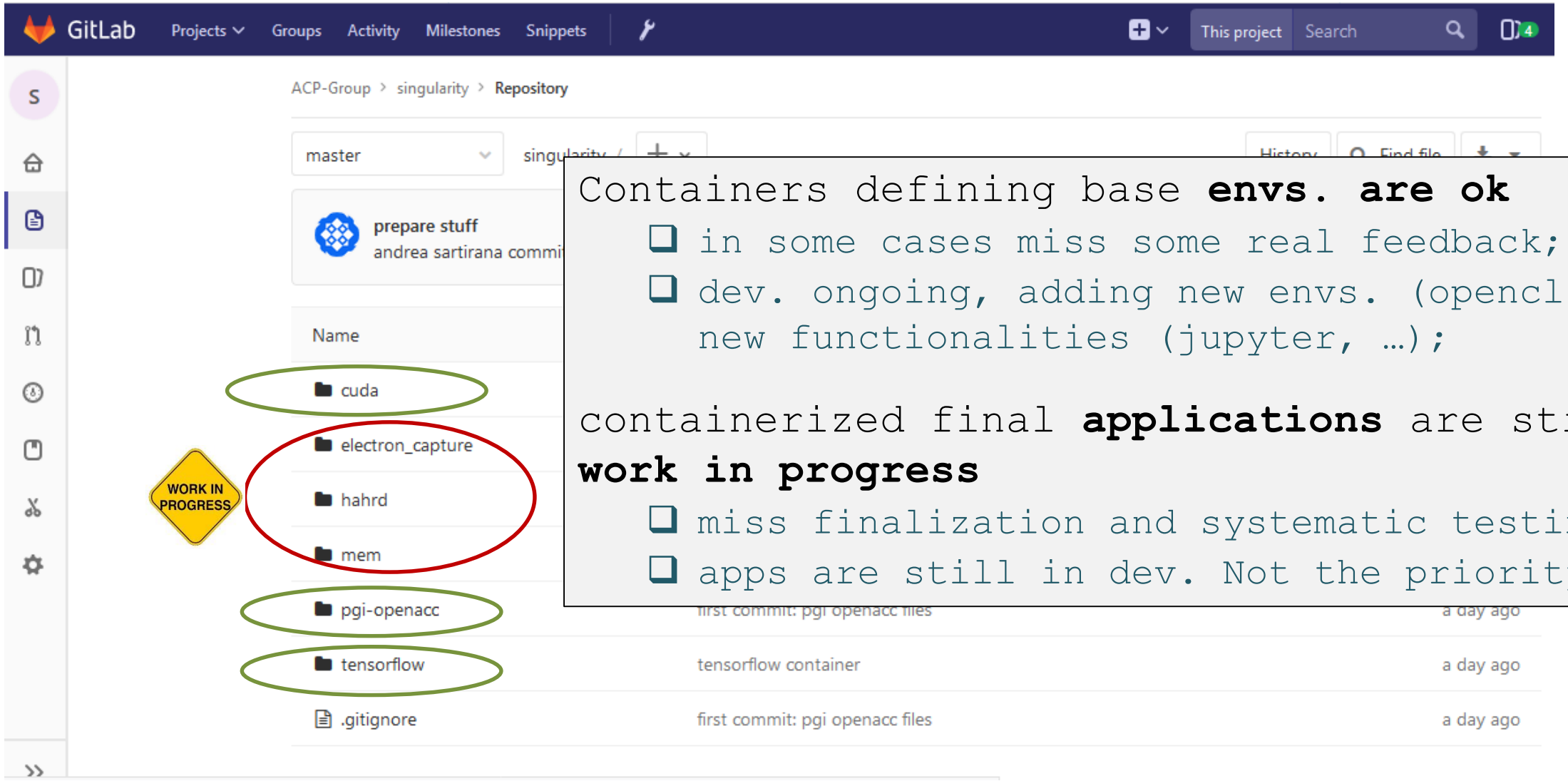
singularity containers…



Build files for the singularity containers are **on LLR gitlab** in a dedicated project. Organized by environment/application

- each dir contains **build files and scripts** to create the containers **for different OS's;**
- should we **move/copy** some this to some **IN2P3** and/or **ComputeOps** repo?
- CI: close to metal, test diff. HW

…singularity containers.



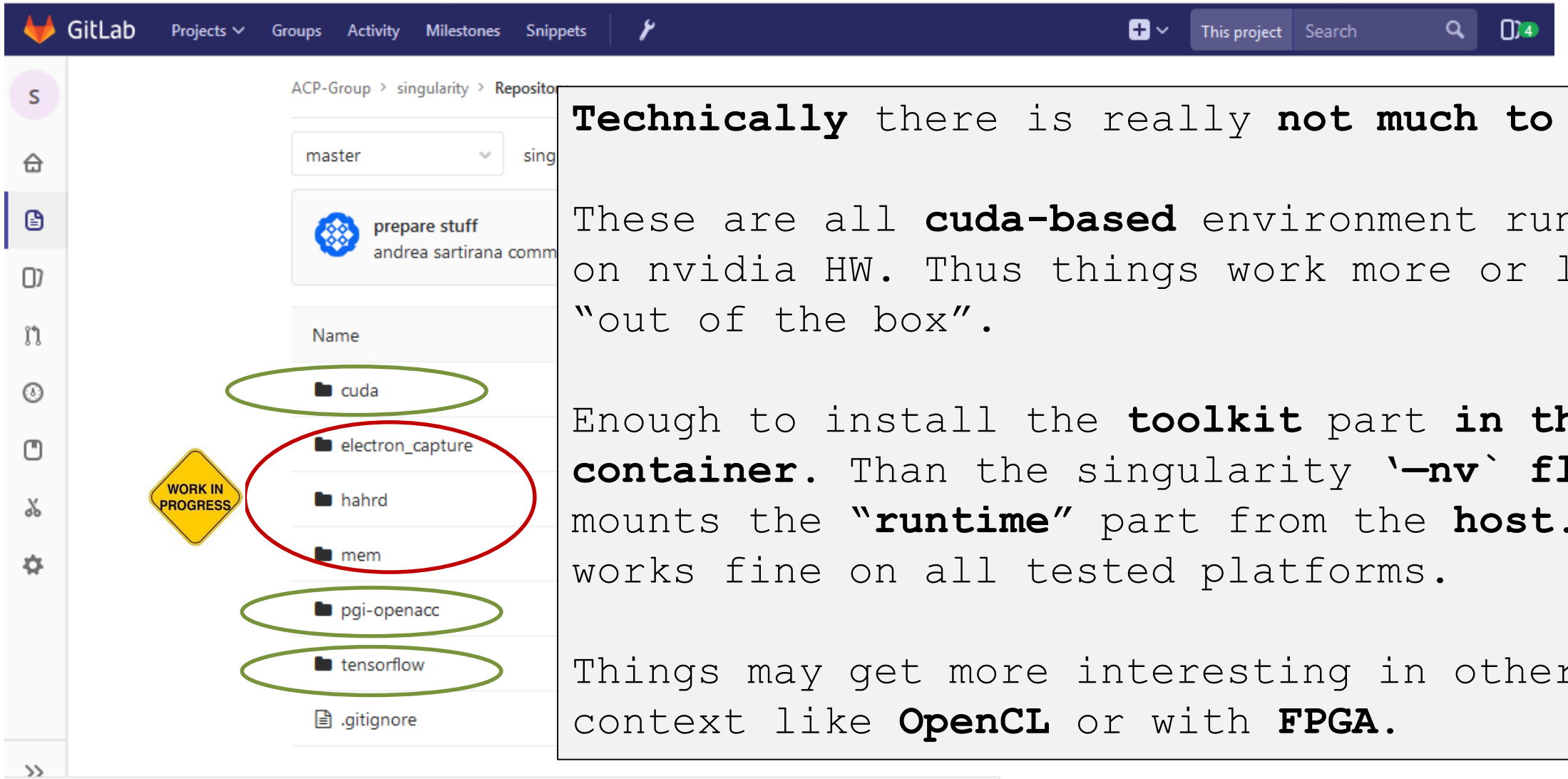
Containers defining base **envs. are ok**

- ❑ in some cases miss some real feedback;
- ❑ dev. ongoing, adding new envs. (opencl) or new functionalities (jupyter, …);

containerized final **applications** are still **work in progress**

- ❑ miss finalization and systematic testing;
- ❑ apps are still in dev. Not the priority.

…*singularity containers.*



Technically there is really **not much to say.**

These are all **cuda-based** environment running on nvidia HW. Thus things work more or less "out of the box".

Enough to install the **toolkit** part **in the container.** Than the singularity **`—nv` flag** mounts the **"runtime"** part from the **host.** This works fine on all tested platforms.

Things may get more interesting in other context like **OpenCL** or with **FPGA.**

images.

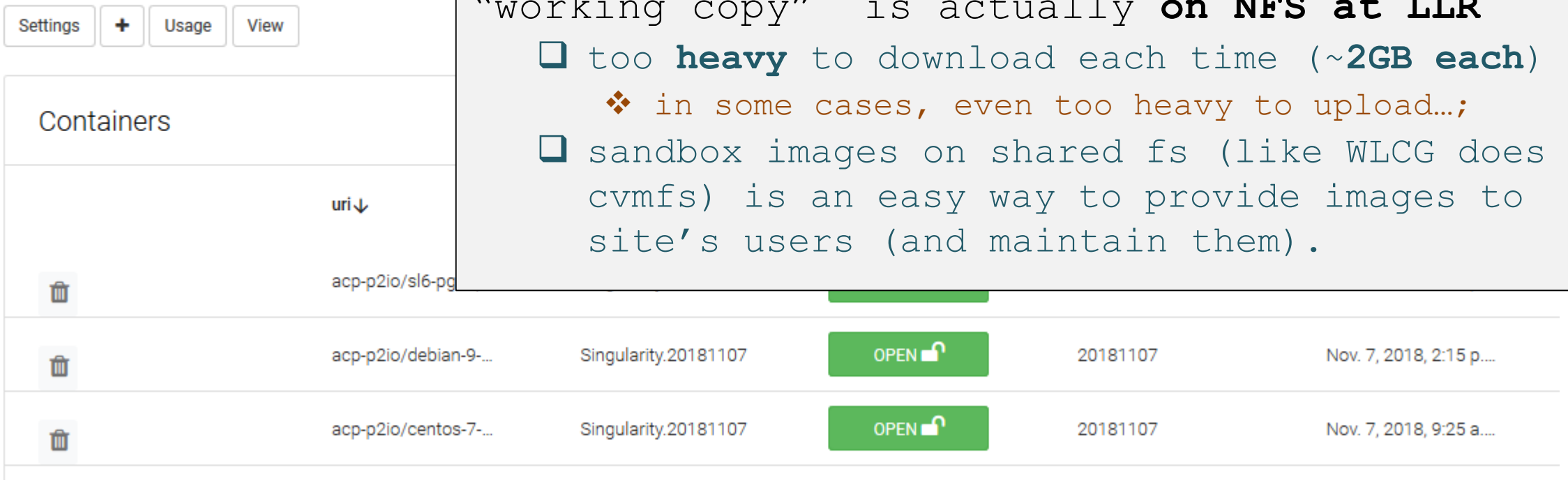
Created a dedicated collection on the **IN2P3 sregistry** to "publish" images

- sregistry does **not** have way to define **metadata** beyond name and tag;

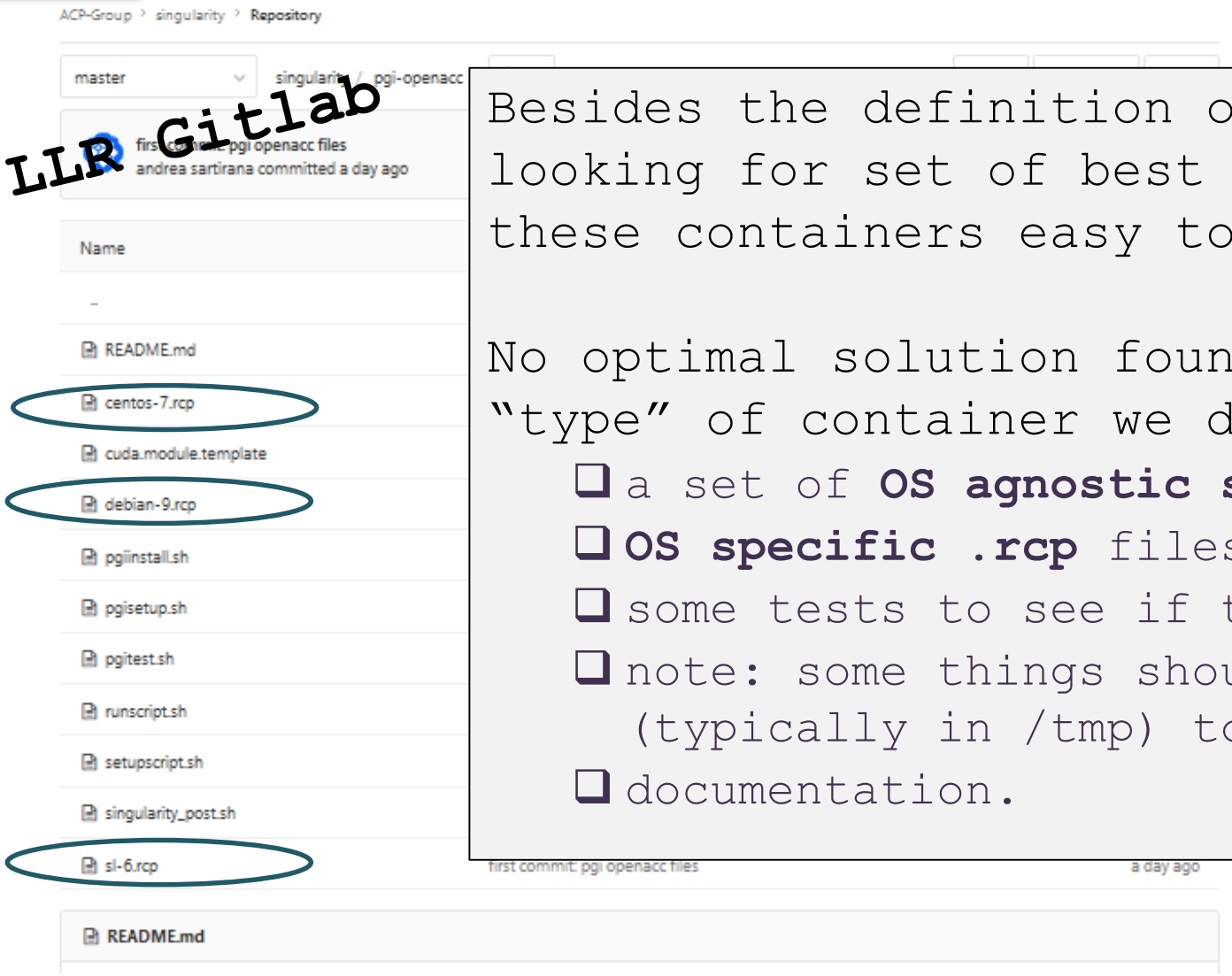
"working copy" is actually **on NFS at LLR**

- too **heavy** to download each time (~**2GB each**)
  - in some cases, even too heavy to upload…;
- sandbox images on shared fs (like WLCG does on cvmfs) is an easy way to provide images to site's users (and maintain them).

acp-p2io collection



best practices.



Besides the definition of useful images we are looking for set of best practices (or rules) to make these containers easy to build, use and maintain.

No optimal solution found so far. Currently, for each "type" of container we define

- a set of **OS agnostic scripts** for setting up stuff;
- **OS specific .rcp** files and conf files;
- some tests to see if the image works correctly
- note: some things should be **present on the build host** (typically in /tmp) to allow build: e.g. *pgi sources*;
- documentation.

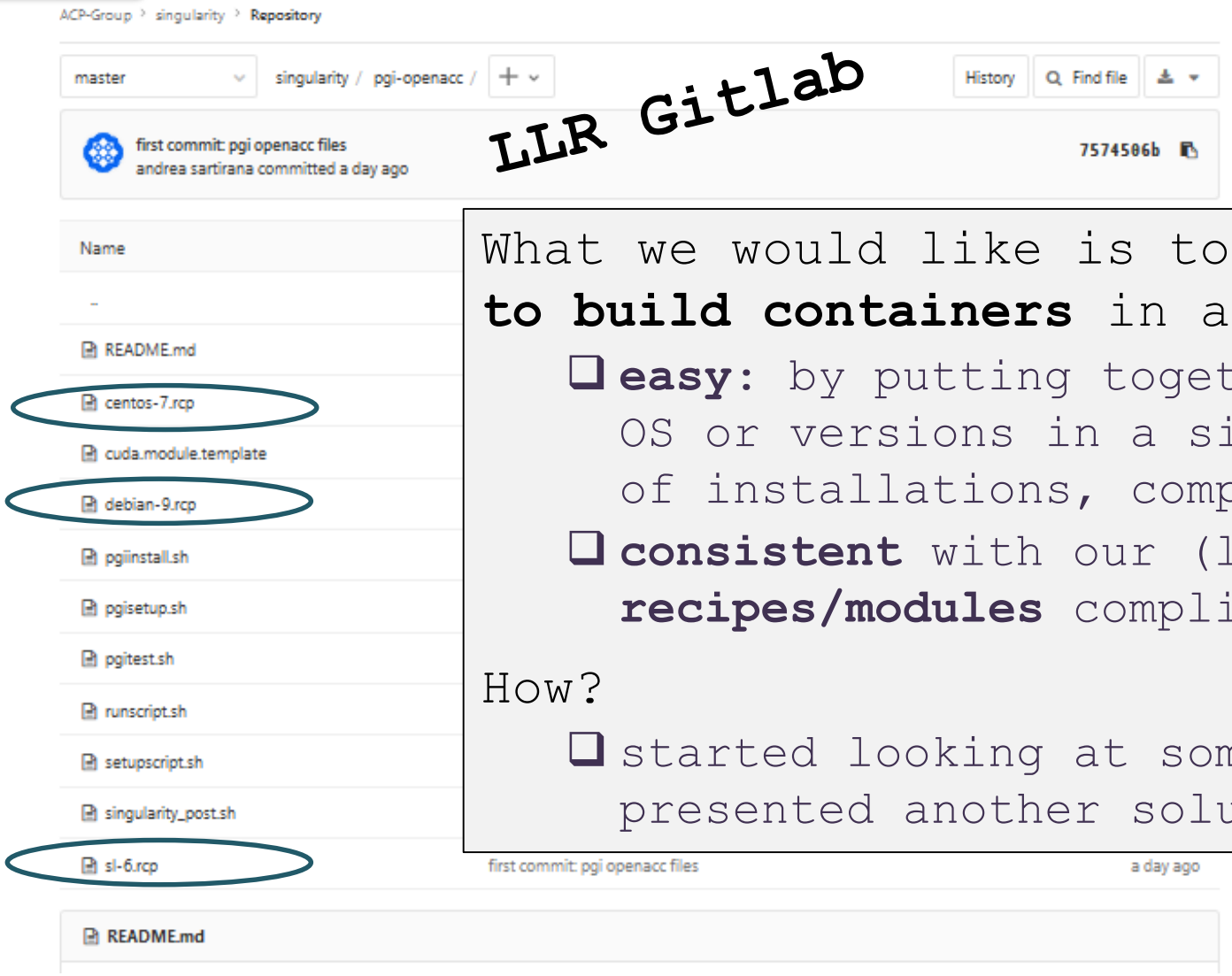
best practices.

debian-9.rcp 1.16 kB

```
Bootstrap: debootstrap

OSVersion: stretch

MirrorURL: http://ftp.fr.debian.org/debian/

%help

This image provide pgi installation with openacc and cuda
To run this image you need to provied the licence file by mounting

...

%label

%environment

%setup

`dirname $SINGULARITY_BUILDDEF`/setupscript.sh

%post

# install packages
#apt-get update
apt-get -y install --allow-unauthenticated wget hostname\
   vim g++ environment-modules emacs\
   openjdk-8-jre-headless libgtk2.0-0

apt-get clean

/usr/sbin/singularity_post.sh

%runscript

/usr/sbin/runscript.sh $@
```

Example

modularity.



What we would like is to **allow** (ourselves and) **users to build containers** in a way that is

- **easy**: by putting together **modular parts** and choosing OS or versions in a simple way, avoiding complexities of installations, compilations, etc…;
- **consistent** with our (local) environment: have **blessed recipes/modules** compliant with a standard HEP env;

How?

- started looking at something with **puppet**. Hadrien presented another solution (**spack**).

summary & outlook

- Been working on **singularity containers** for environments and applications running on our GPU platform
  - applications still have to be finalized;
  - todo: OpenCL (may be useful for FPGA), … ;
- still open question: which is the **best way to organize** the **images** and their **recipes?**
  - more metadata for built images;
  - allow the users to easily and consistently create an images;
- **explore/compare** different products
  - e.g. **nvidia docker** and **DIGITS** ;
- start working on **FPGA** when the new platform is there.