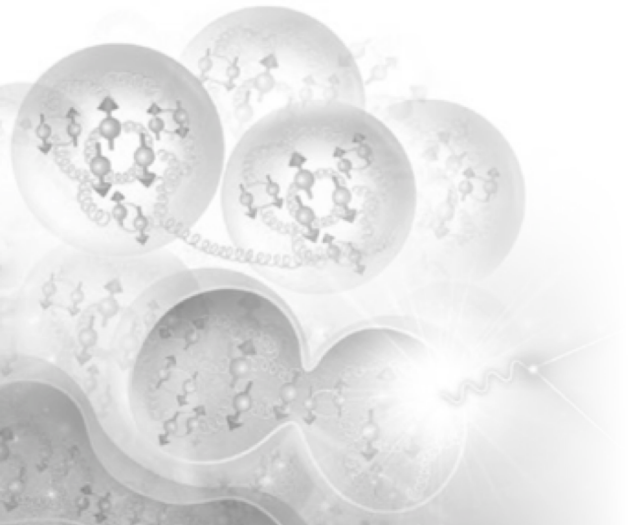
Start the docker

docker pull electronioncollider/epic-gui

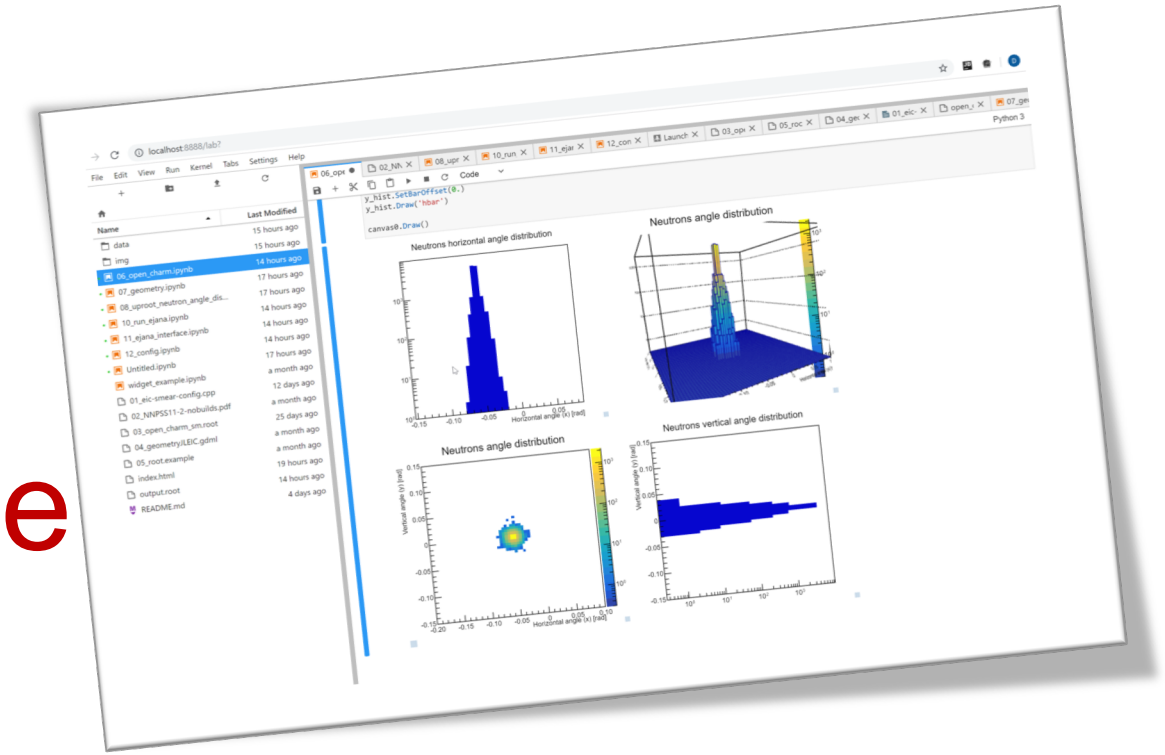
How to install docker:

eic.gitlab.io / documents / quickstart



Entry point for EIC software

Interactive tutorial
Full simulations g4e

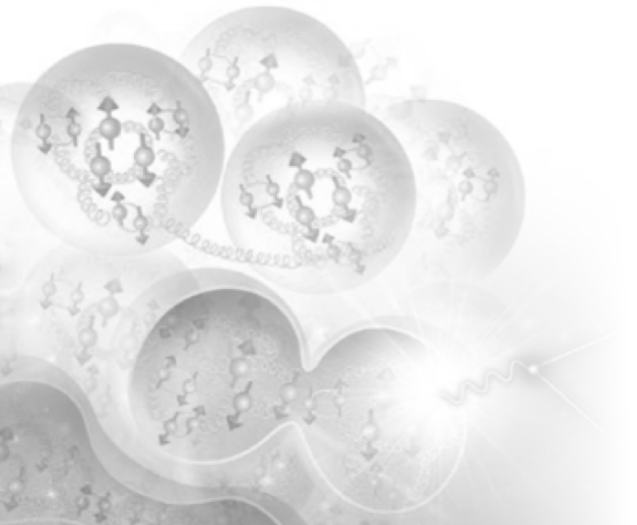


Dmitry Romanov

Update the docker

docker pull electronioncollider/epic-gui

eic.gitlab.io / documents / quickstart /

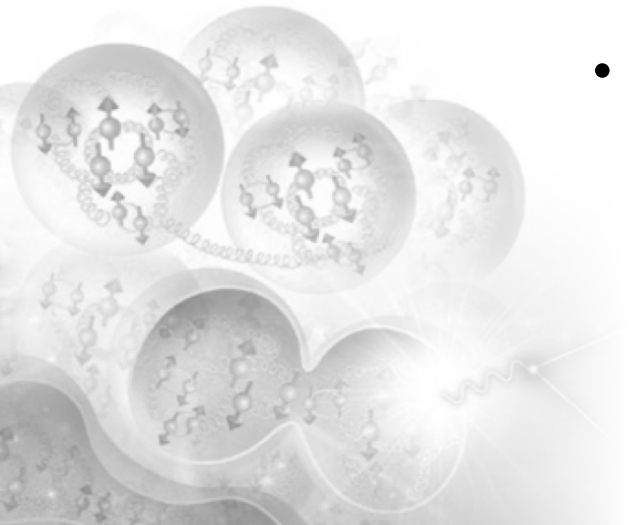


Start the docker

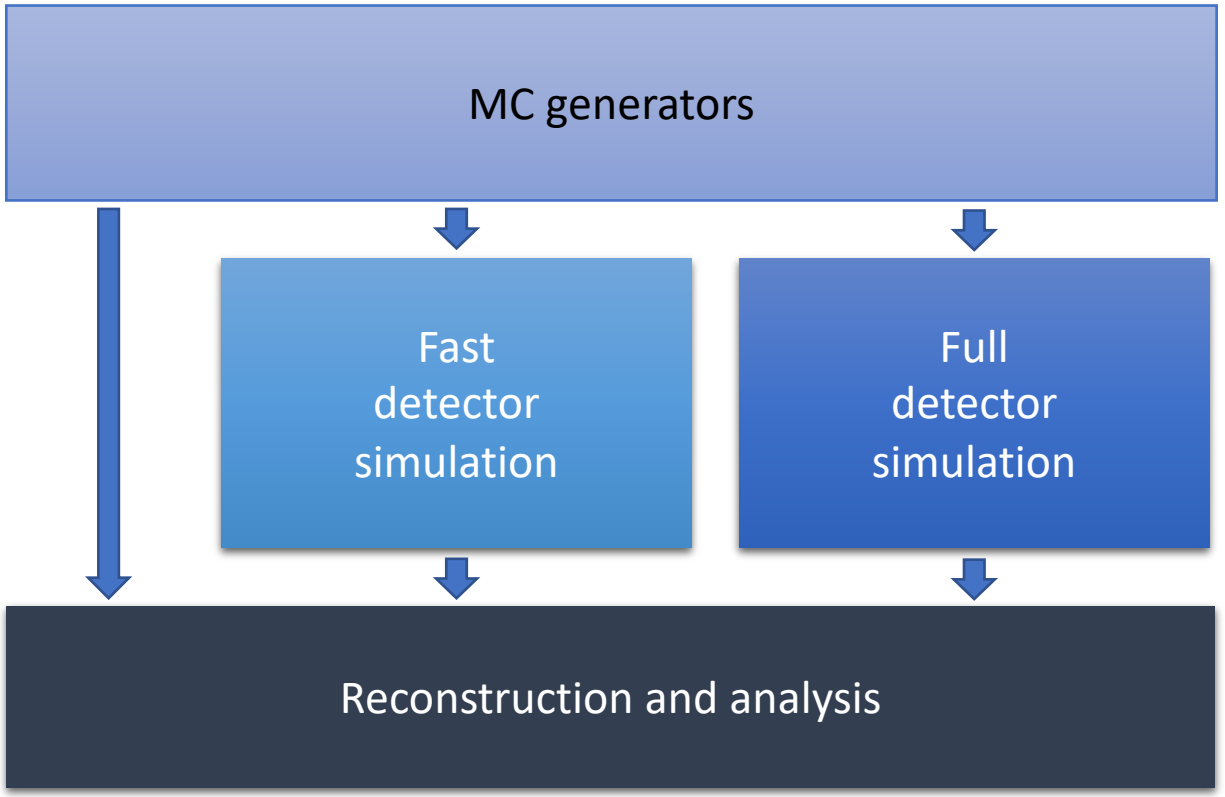
docker pull electronioncollider/epic-gui

Plan:

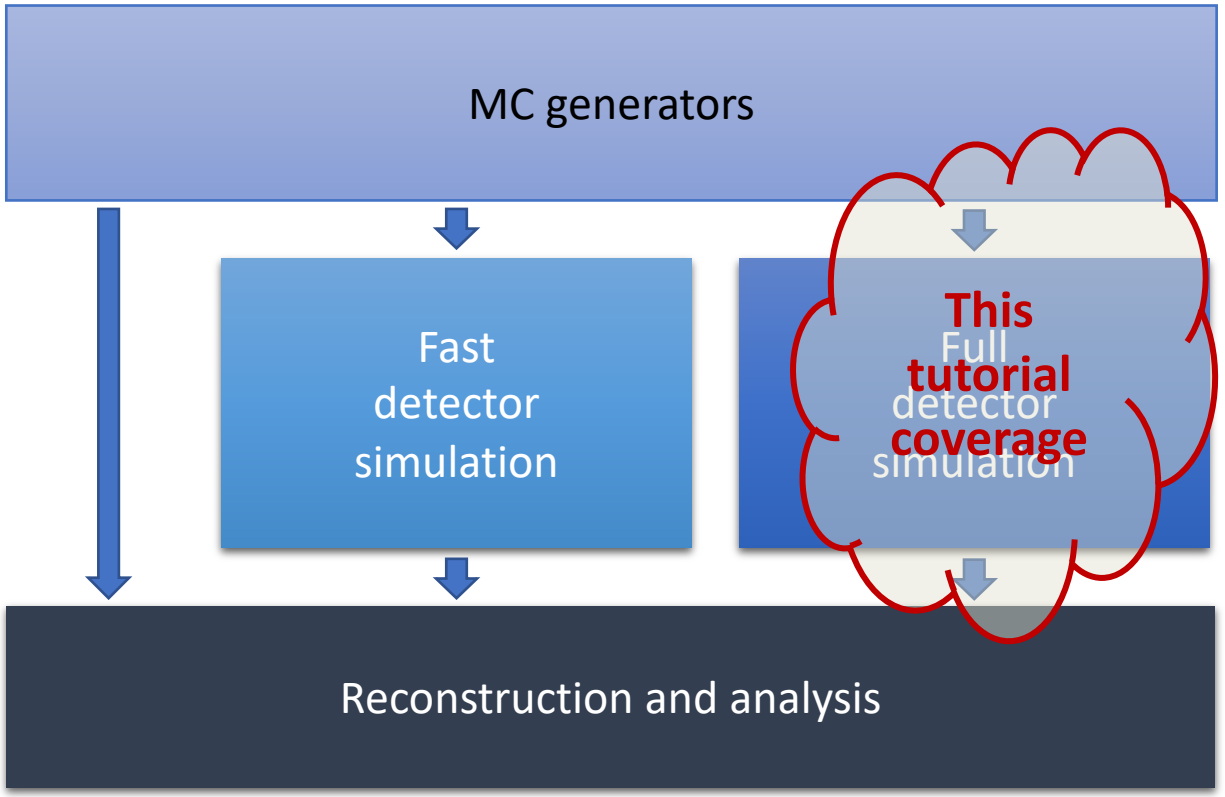
- Short introduction to our software
- **1. Hands on tutorial – running a simulation**
- **2. Hands on tutorial – adding a detector**
- Answering most common questions (presentation)
- Adjourn!



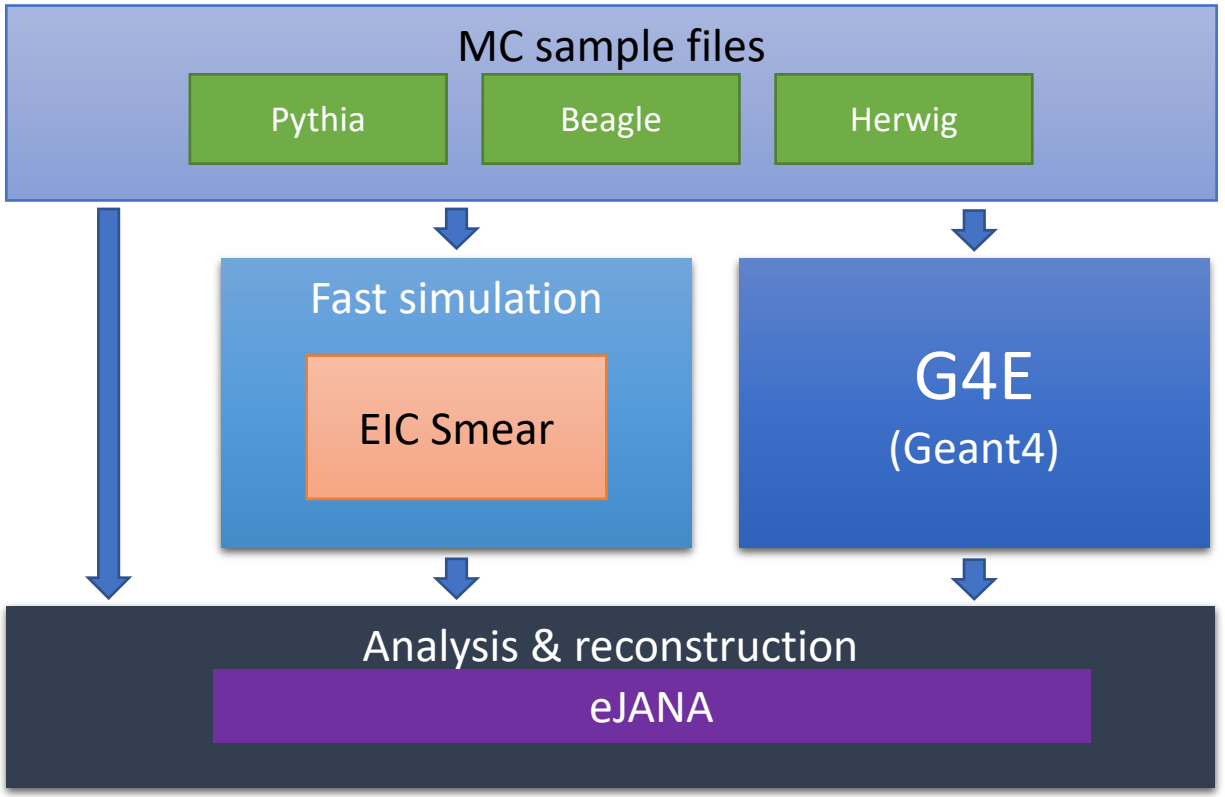
Software chain



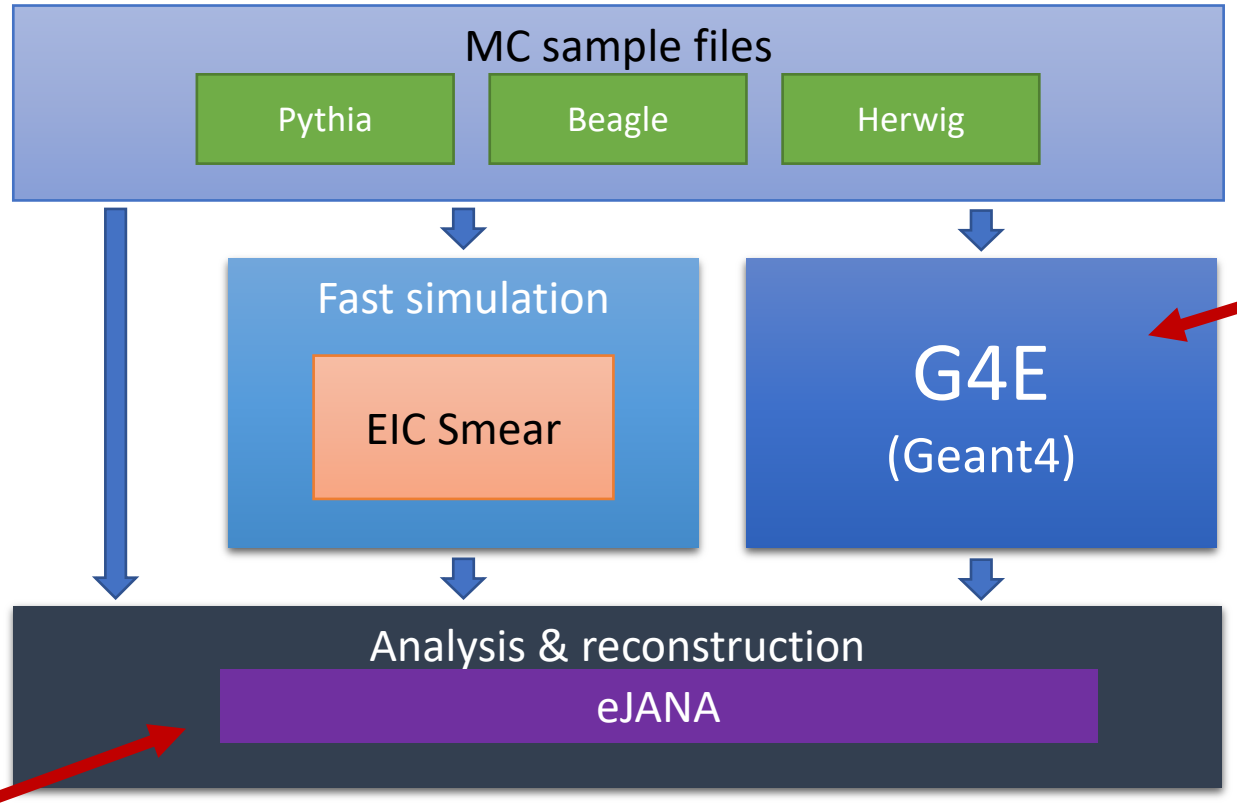
Software chain



What software is in the docker container



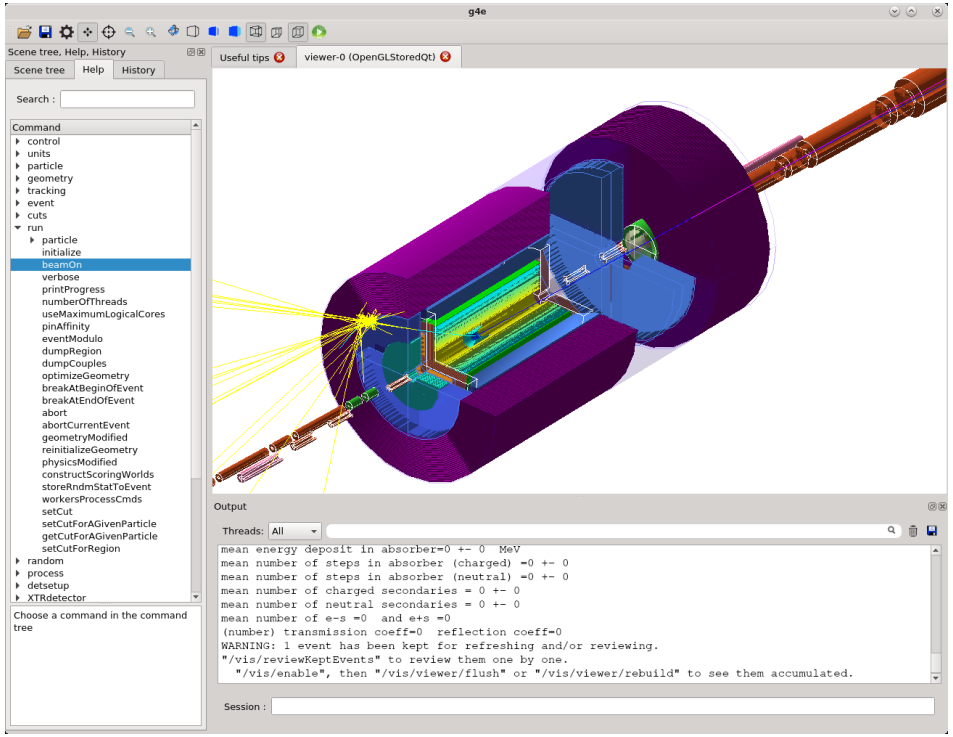
What software is in the docker container



Geant4Eic
Pure Geant4 implementation

ejana = EIC Jana
Community reference reconstruction

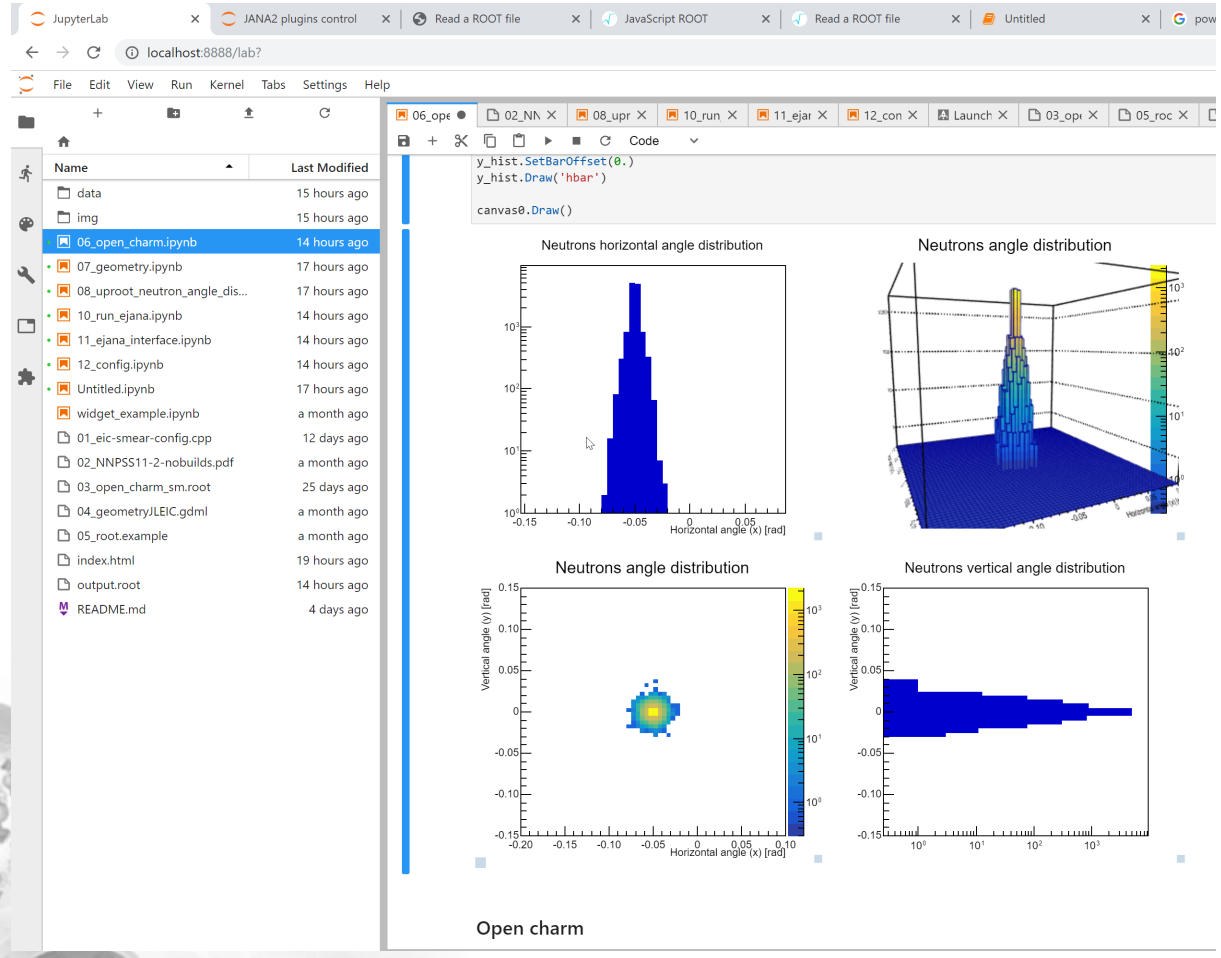
G4E – Geant 4 EIC



https://gitlab.com/jlab-eic/g4e

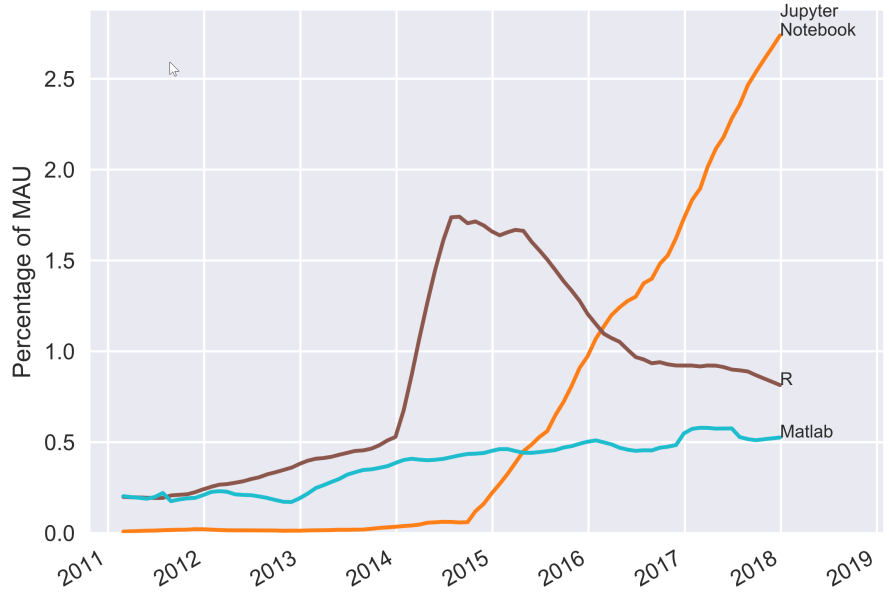
- Standalone C++ Multithreaded Geant4 application
- Various EIC MC file formats: Beagle, Pythia6, HEPMC - Pythia8, Herwig and others
- Integration with accelerator elements
- Infrastructure to import Geant4 detector geometry and simulation code

Jupyterlab new interface to Jupyter



Scientific Languages

There was one other fast-growing 'language' included in the results that I purposefully left out:



Do we hide complexity?

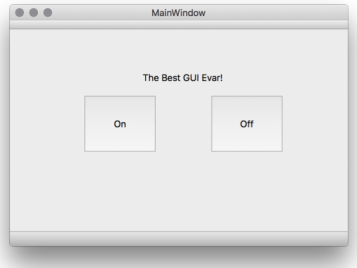
Jupyter lab, GUI,

Python, scripts, analysis

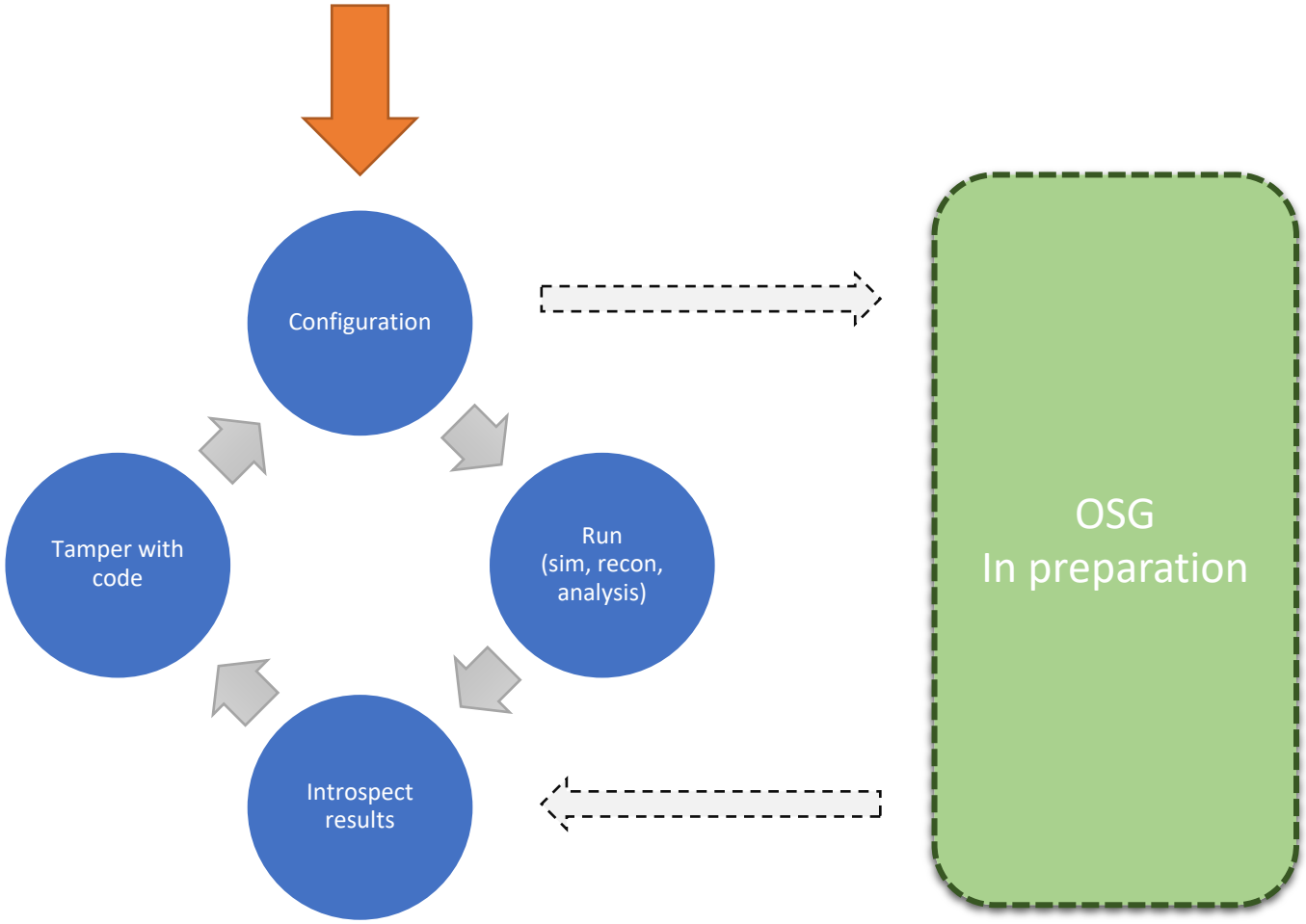
C++, eJANA, plugins

JANA, eic-smear, ROOT, Geant4





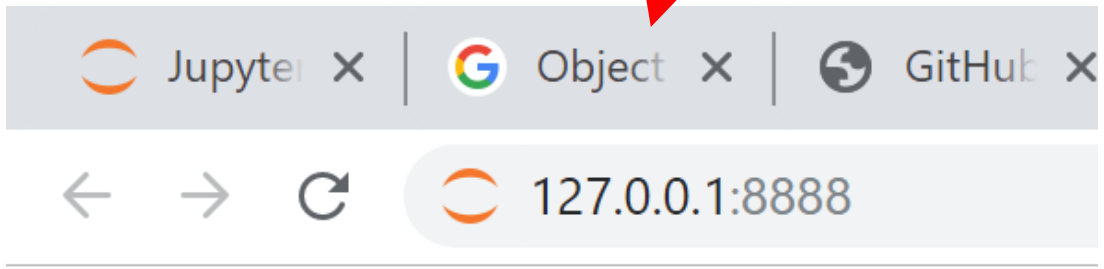
Run on Open Science Grid



Start the docker

```
docker run --rm -it -p 8888:8888 -p 6080:6080
electronioncollider/epic-gui
```

```
...
[I 20:42:19.975 LabApp] The Jupyter Notebook is running at:
[I 20:42:19.975 LabApp] http://127.0.0.1:8888/
[I 20:42:19.975 LabApp] Use Control-C to stop this server
```



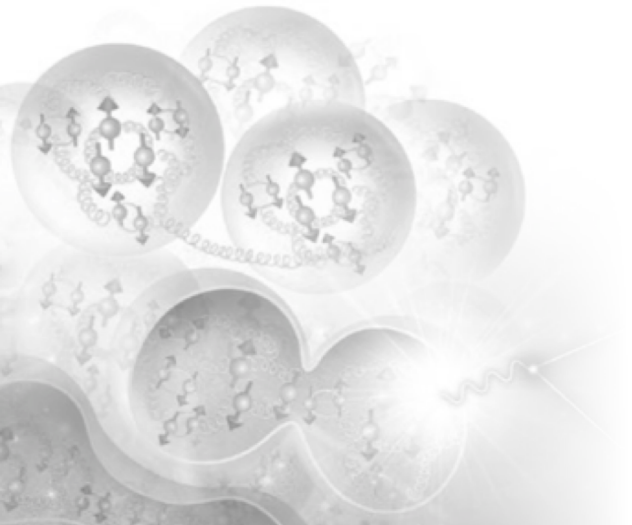
Ways to interact G4E with the docker

1. JupyterLab (in browser)
2. noVNC (in browser)
3. Any VNC viewer
4. X11 (directly or through ssh)
5. Remote debugging

… or just install everything on your machine

https://gitlab.com/eic/ejpm

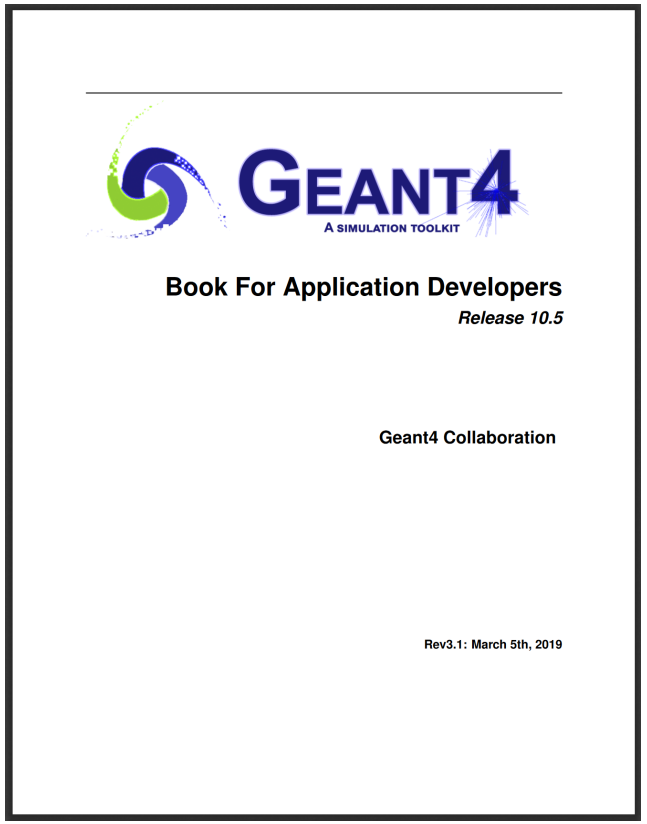
Part 2 – Importing a ready detector to G4E



Geant4Eic (g4e)

https://gitlab.com/jlab-eic/g4e

- Keep close to raw Geant 4 (10.6)
- Small code base and fast compilation is good - KISS
- Users coding in “Geant4” paradigms
- Coding is ~~OK~~ GOOD
- Few interfaces are well defined and documented. E.g.
  - How to move in a detector
  - What is the output root file structure
  - etc.
- It is up to users what to do, but there are Recommendations to commit code back



Main detectors – sub detectors

Beamlines

ERHIC | JLEIC | Future updates?

Master detectors

BEAST | JLEIC | ePhenix | Etc…



Subdetectors

Now 24 subdetectors

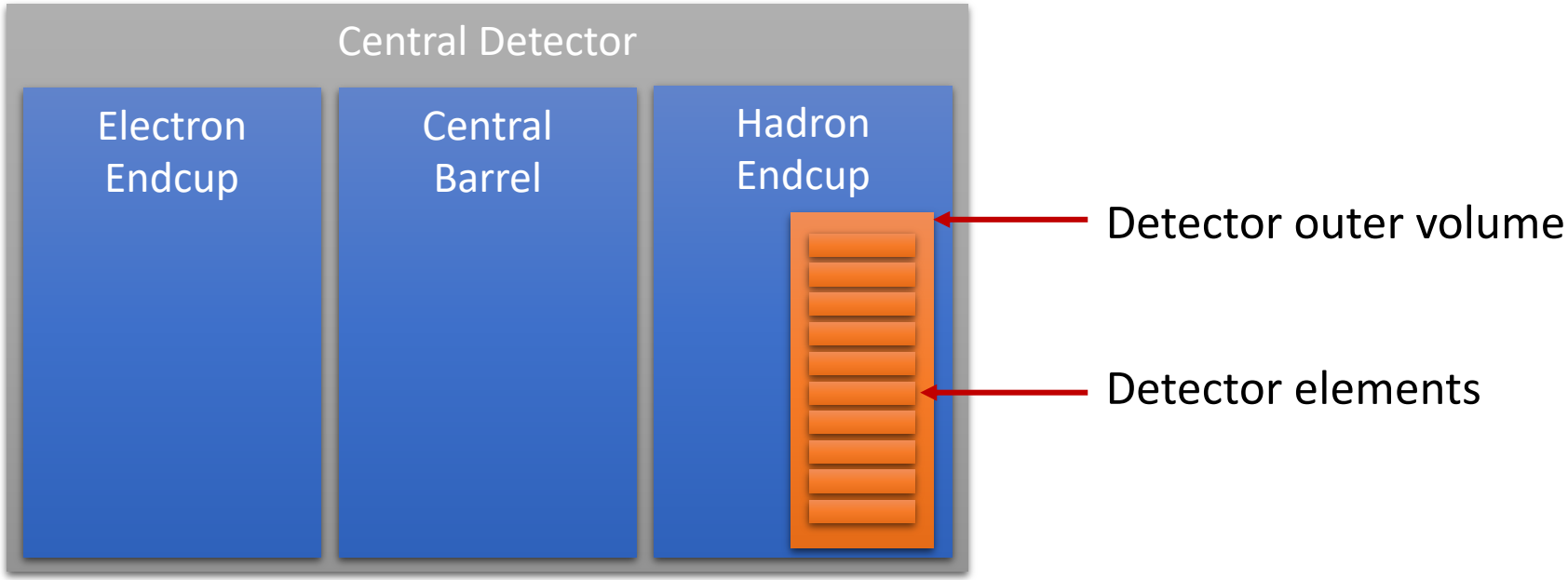
How one can add subdetecors to G4E

1. Create detector in place (**in this tutorial**)
2. Import detector from a library (**this tutorial**)
3. Import detector from ROOT Geometry (example is upcoming)
4. STEP file import (example is upcoming)
5. Define geometry with Geant4py (official G4 python library).
   (with C++ running speed)

Subdetectors are developed **under its own repos**
It would be good at this point not to disturb it

Volumes hierarchy

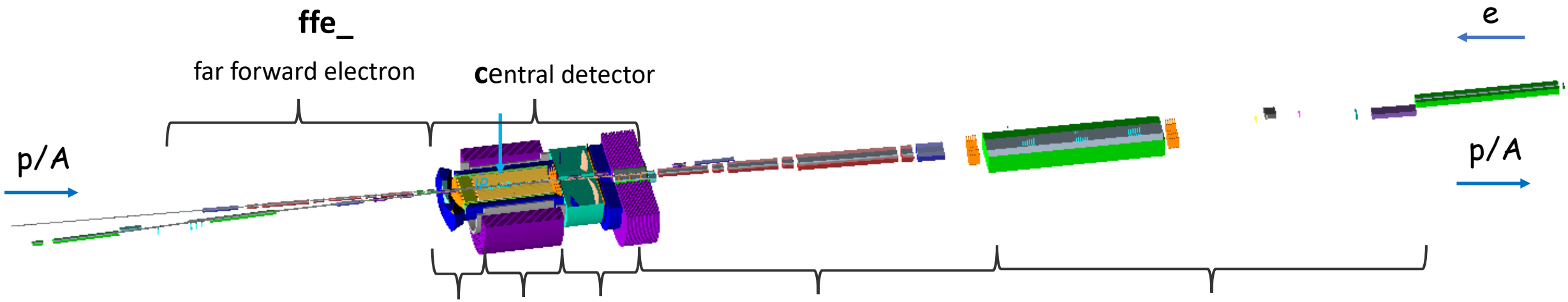
- Each detector is placed in an outer placeholder volume



Naming convention

Names - <region>_<DETECTOR>_sub…

Examples:
cb_VTX = central barrel Vertex detector
ffe_LUMI = far forward electron Luminosity monitor



Naming sum up

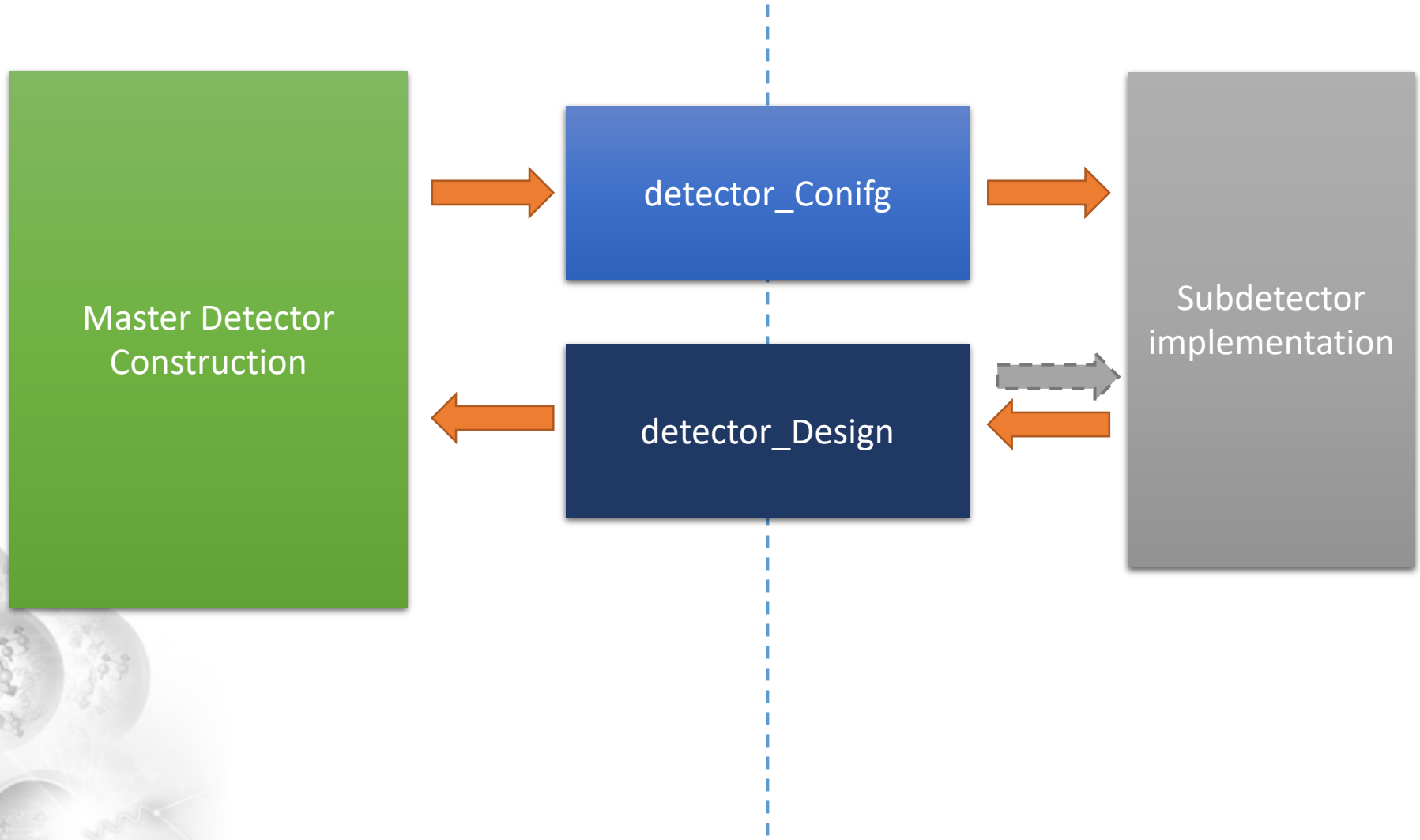
1. Central Detector (c):
   - Barrel (cb) == Central Barrel
     - *Solenoid (cb_Solenoid)*
   - Electron endcap (ce) == Central Electron endcap
     - *GEM tracking (ce_GEM)*
   - Ion endcap (ci) == Central detector Ion endcap
     - GEM tracking (ci_GEM)

2. Forward ion (fi) direction area near D1 magnet:
   - Tracker detector1 (fi_TRKD1)

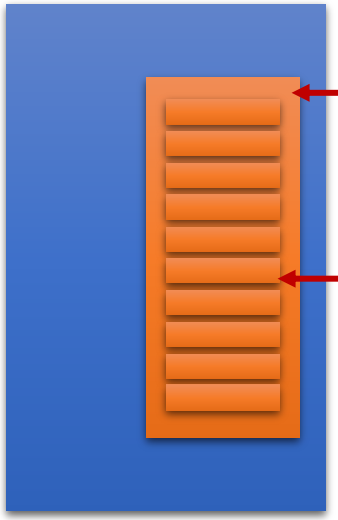
3. FarForward ion (ffi) direction area (near D2, D3 magnets)
   - ZeroDegree Calorimeter (ffi_ZDC)
   - Roman Pots (ffi_RPOTS)

4. Far forward electron (ffe) direction area
   - Low*Q2 tagger (ffe_LQ2)
   - Electron Polarimeter (ffe_CPOL)
   - Luminosity monitor (**ffe_LUMI**)

Detector design interface



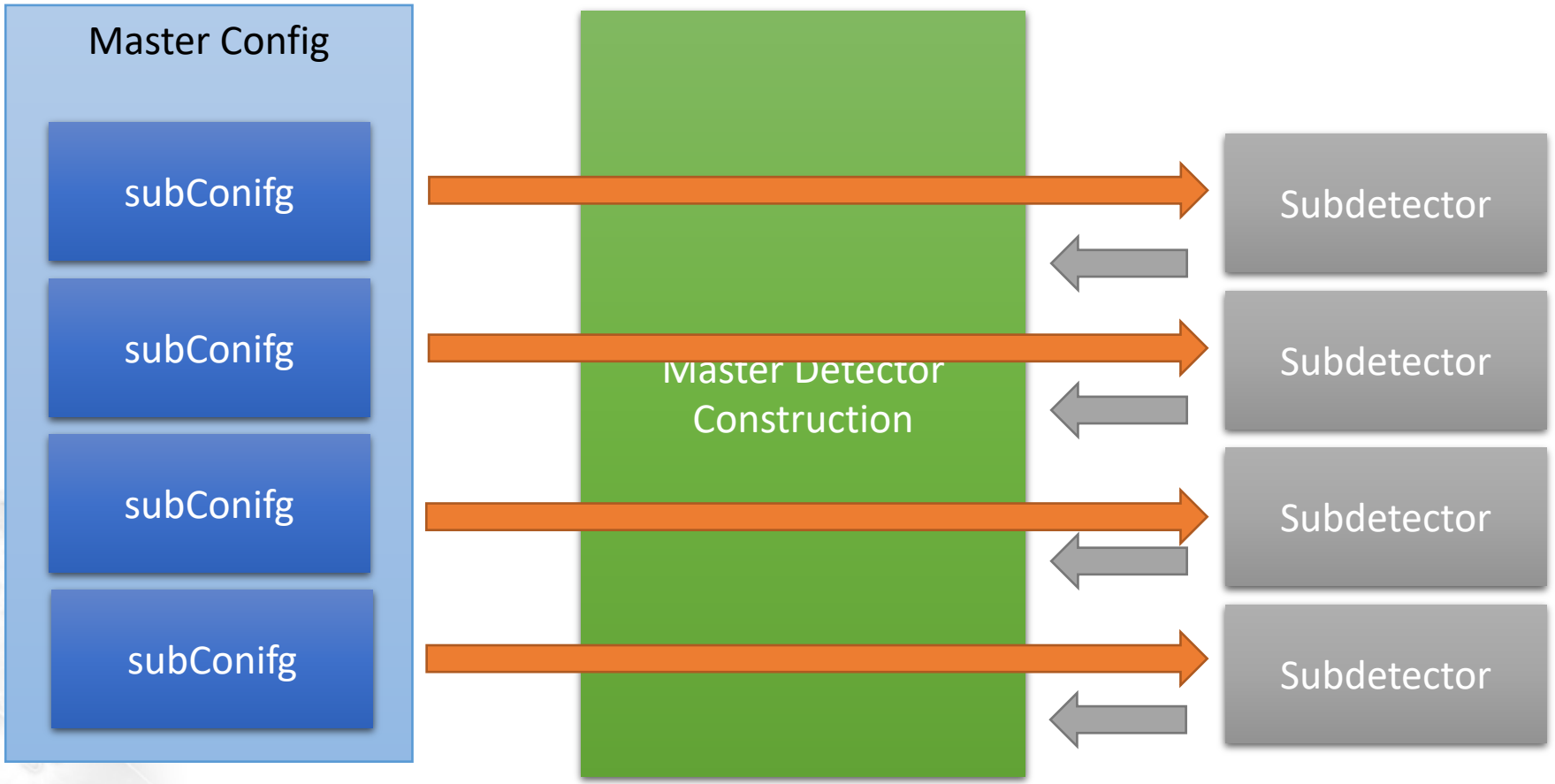
Volumes hierarchy



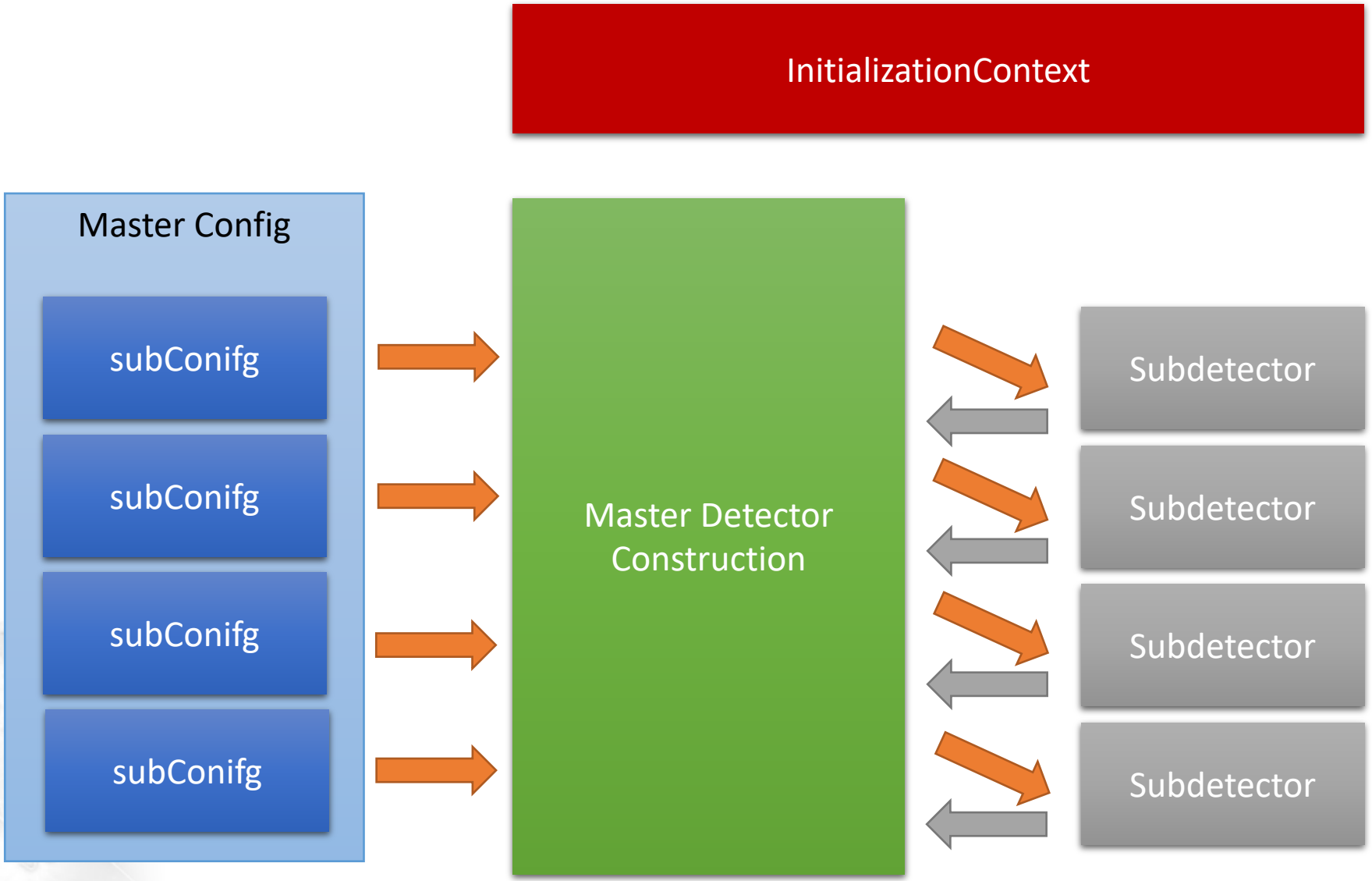
Detector outer volume - **ffe_LUMI.Construct(…)**

Detector elements - **ffe_LUMI.ConstructDetectors()**

Configuration



Configuration



Final improvements

- Mapping User Actions

```
actionInit = initContext.ActionInitialization
actionInit.AddUserActionGenerator([&](){return new lumi::EventAction();});
actionInit.AddUserActionGenerator([&](){return new lumi::RunAction();});
```

- Using global root output

```
initContext.RootManager->GetMainRootFile()
initContext.Arguments->OutputBaseName
```

- Adding ffe_LUMI_Config to master_Config

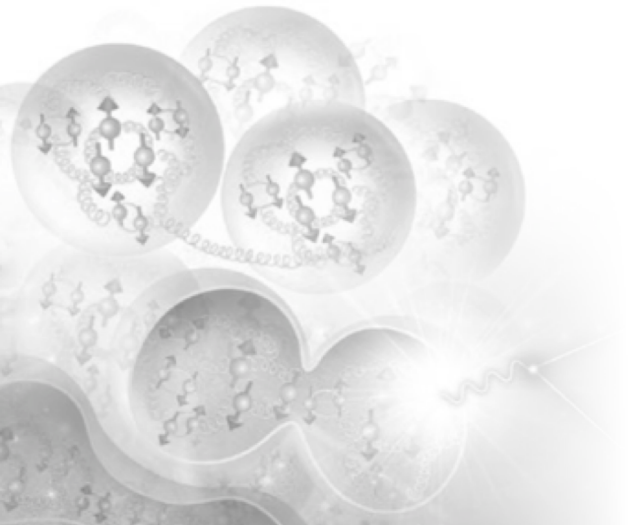
Multithreading



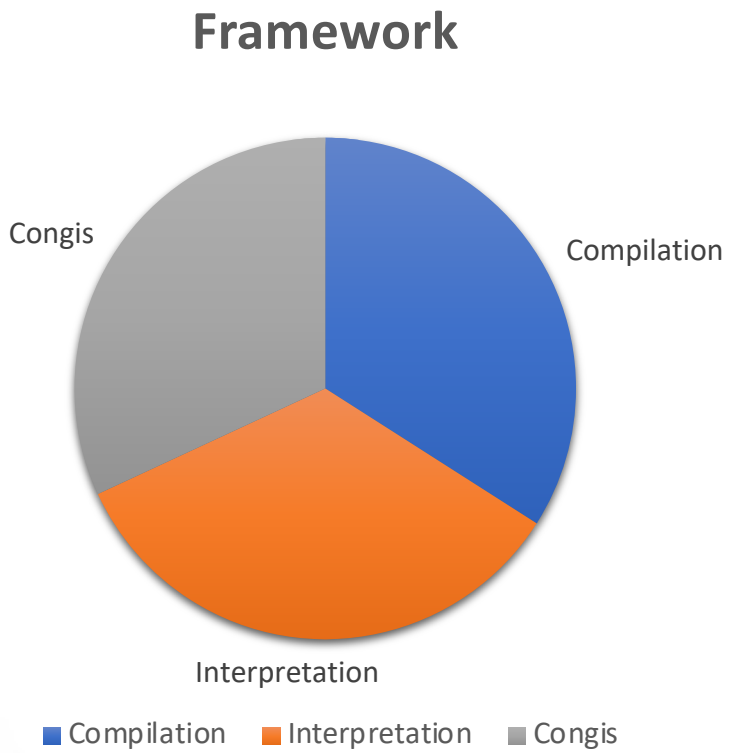
- G4E supports Geant4 multithreading
- But it was turned off for EHRIC beamline implementation
- But this week everything was fixed, and it works back
- But some issues are spotted
- But we will fix them ASAP and now you can run it
  (on your own risk)

Multithreading

- .threads(8) – in python API
- -t or --threads or -j flags on program run
- Please stay tuned, we will fix the final bugs ~1 week



Compilation vs interpretations



How access parameters from Geant 4 macros:

```
messenger = new G4GenericMessenger(this, "/my/params/");
messenger->DeclareProperty("param", fParam);
```

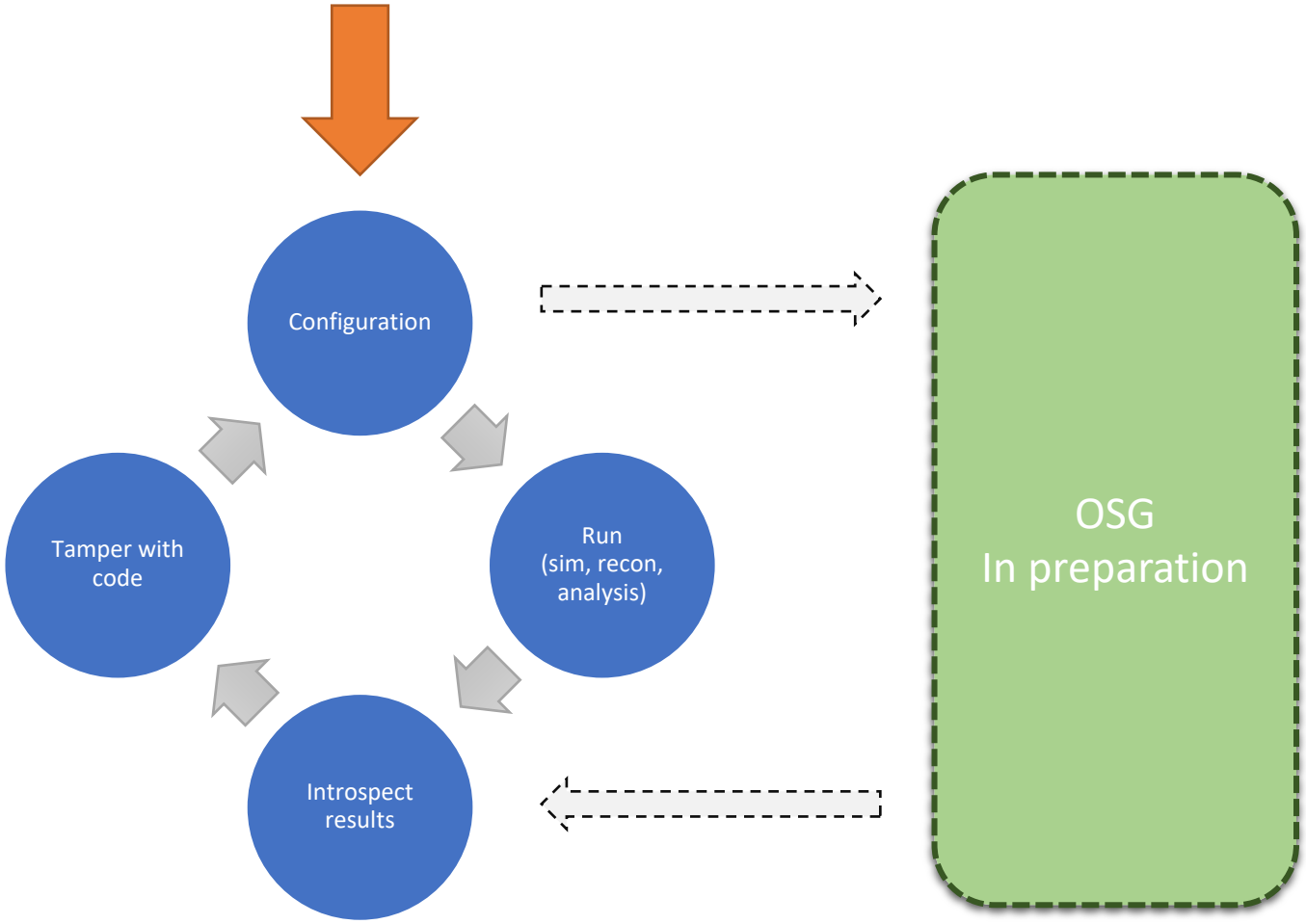
The real question:

how users submit their configuration (in terms of customized files) to various forms of batch/grid jobs???

The answer:

…Is outside of frameworks responsibility

Run on Open Science Grid

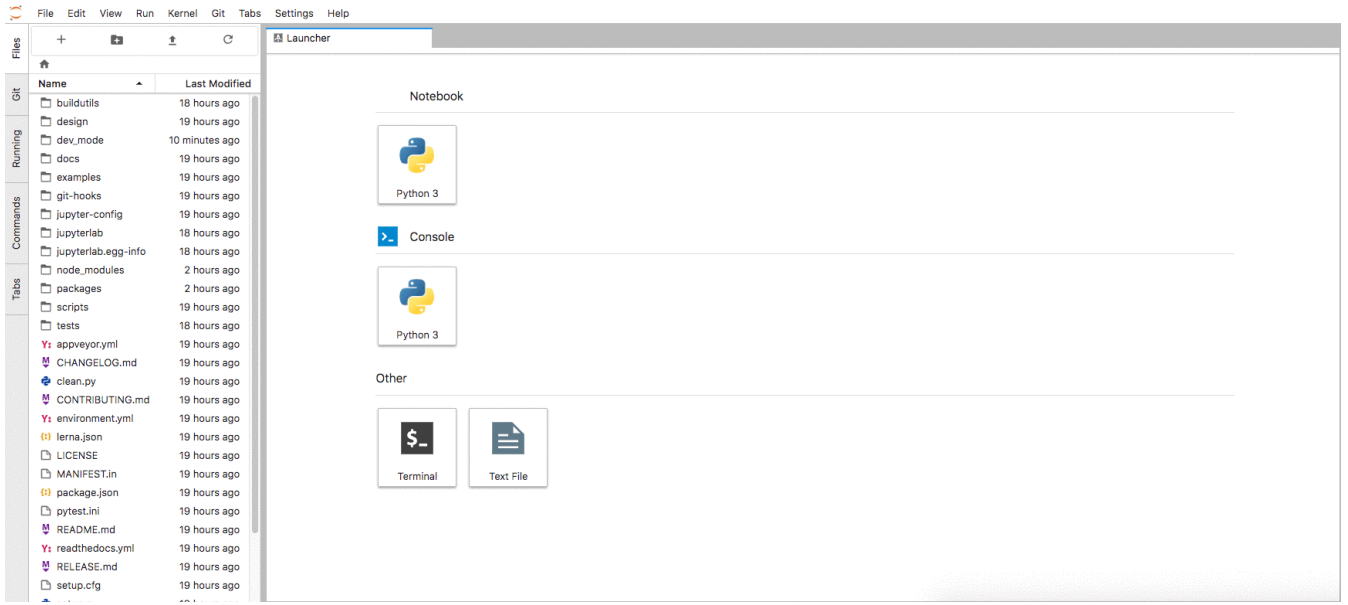


How can I save data?

1. Remove “remove” flag!

   docker run ~~--rm~~ -it -p 8888:8888 electronioncollider/epic-gui

2. The project is a git repository. Jupyterlab-git is installed



How to process your file with this docker?

- docker run -it --rm -p 8888:8888 -p 6080:6080 \
  -v my_folder:/home/eicuser/epw/share \
  electronioncollider/epic-gui

- -v source:destination – flag to bind your and docker directories

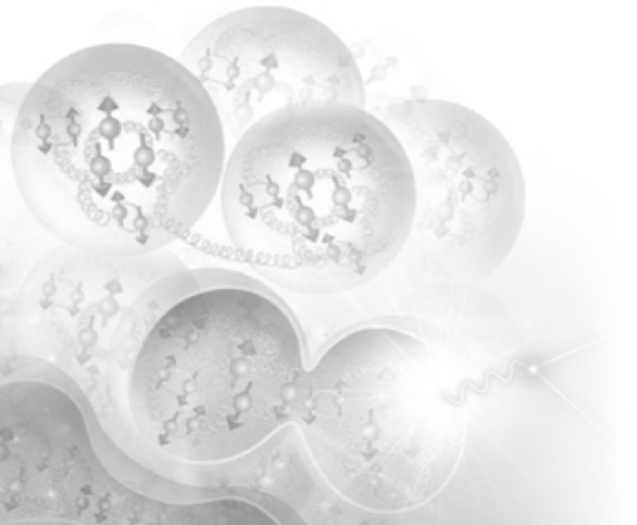
- /home/eicuser/epw/share – to instantly see in the default JupyterLab directory

- Windows – you will need
  - sharing a drive or…
  - use WSL2 or…
  - Virtualbox

Full docker run string

- docker run -it --rm -p8888:8888 -p 6080:6080 -p 5901:5901 --cap-add=SYS_PTRACE --security-opt seccomp=unconfined electronioncollider/epic-gui

  - --cap-add=SYS_PTRACE --security-opt seccomp=unconfined - enables debugging
  - 8888 – JupyterLab, 6080 – noVNC, 5901 – any VNC client



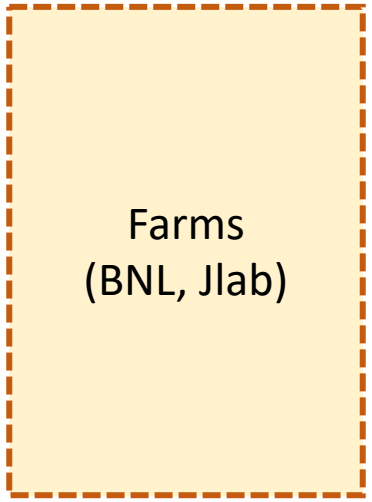
How to install things on my machine?

NO EFFORT AT ALL
Novice

Some effort
Experts

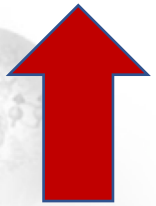
Efforts required axis











This tutorials

Conda, CVMFS

EIC Jana ~~Packet~~ Build Managaer = ejpm

Users input needed!



We've built a base for our software stack.
Users are welcome to steer and drive the further development

Grow with user input

Develop

Support

Workflow environment for EICUG

- **to use** (tools, documentation, support) **and**
- **to grow with user input** (direction, documentation, tools)

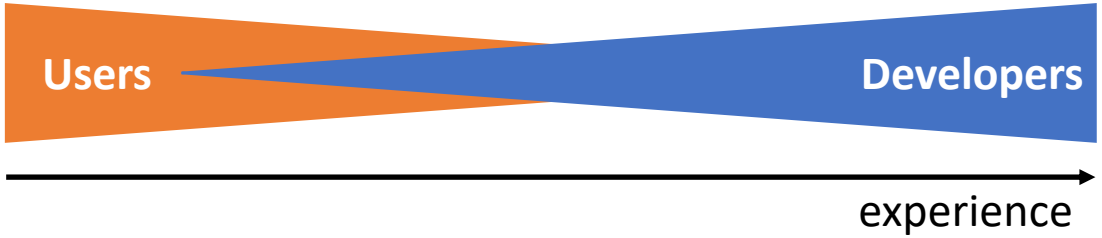


Thank you!

https://gitlab.com/jlab-eic/g4e

https://gitlab.com/eic/ejpm

support team — **weekly shifts**



software-support@eicug.org — **Mailing list** (anyone can contact)
Google forum (for archive of support requests and start of knowledge base)

http://eicug.slack.com/ — **EICUG Slack workspace with software-support channel**

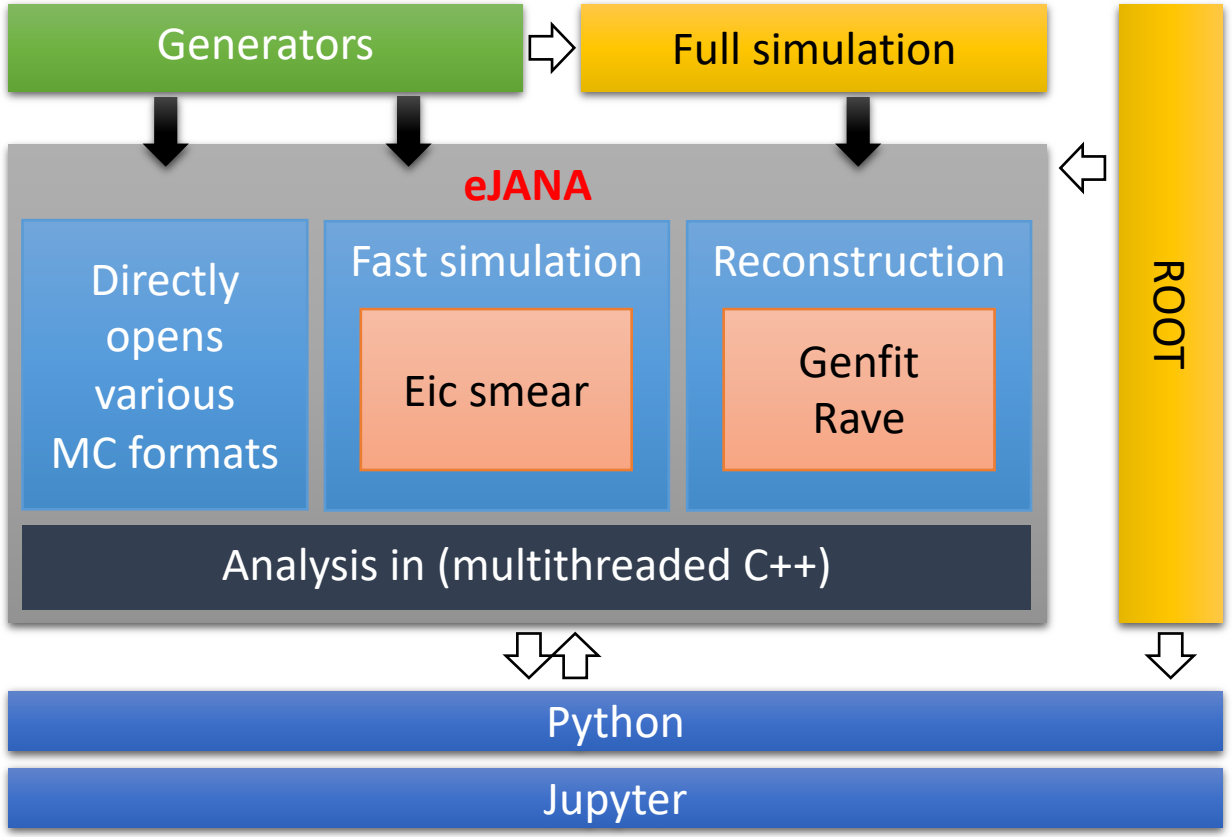
e^JANA

e^JANA **is JANA + plugins** for EIC data reconstruction and analysis

But also:

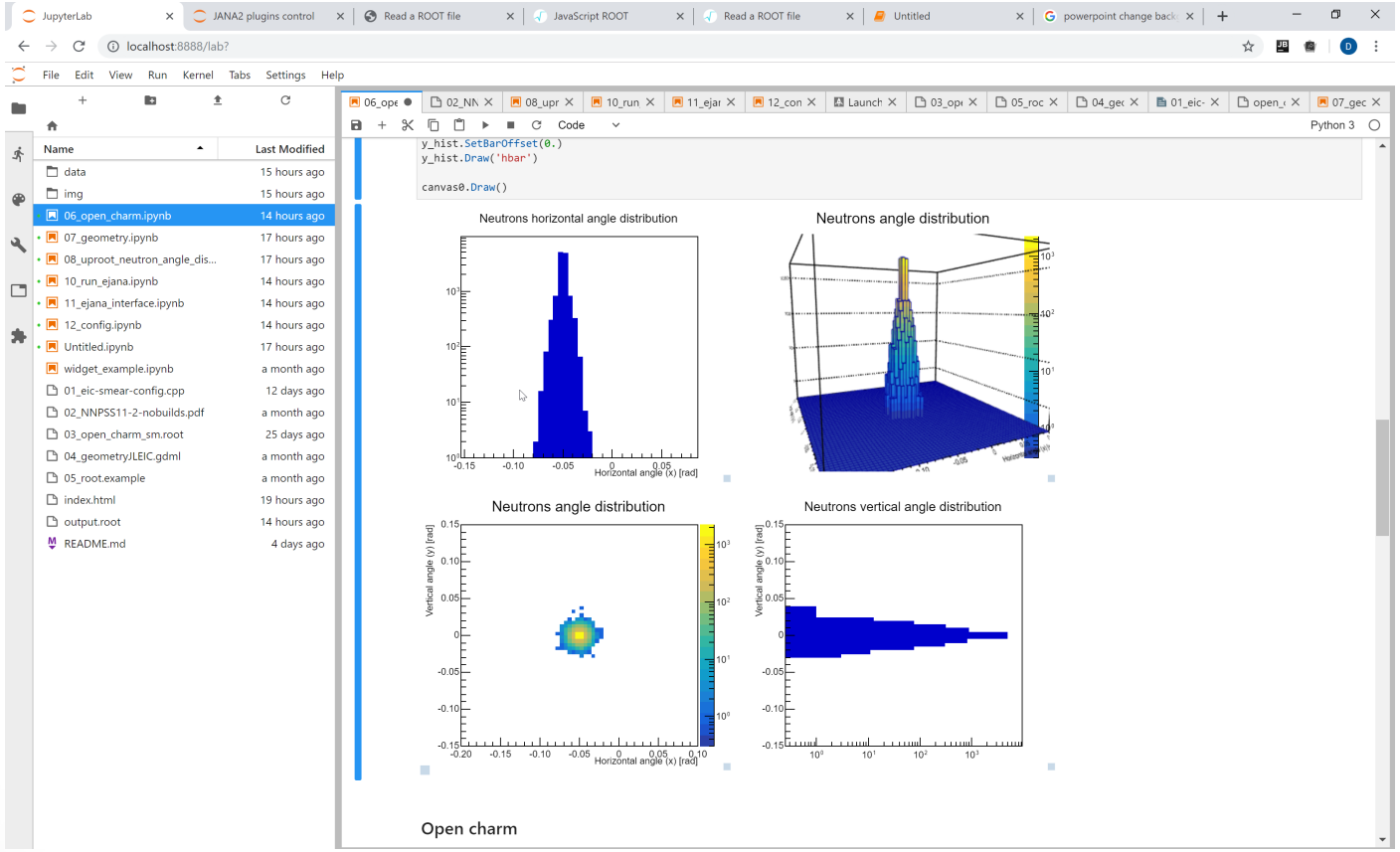
- tools to manage dependencies and run eJANA in different environments
- Integration with python and extensions to Jupyter Lab

  (ejpm, edock, pyjano, and others..)



e^JANA stands for EIC JANA

Jupyter lab, Jupyter notebooks, EPW, epic…



Transparency between layers

- JupyterLab -> Python/ROOT C++. Python -> Command line…

../data/beagle_eD.txt

[3]:
```
jana.run()
```

Total events processed: 10001 (~ 10.0 kevt)

▸ **Full log**

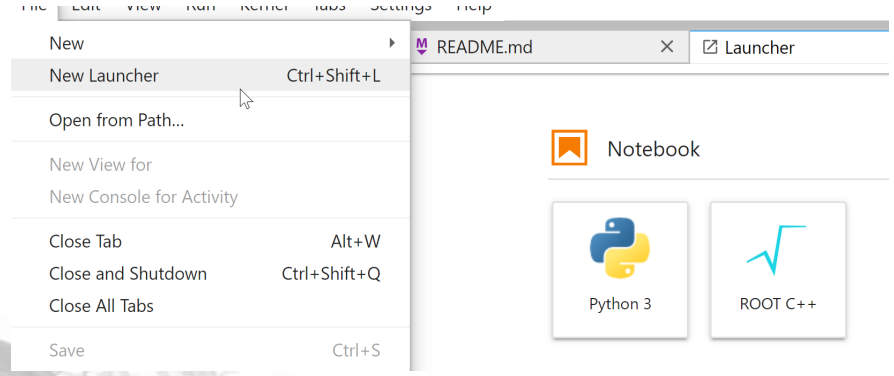
▾ **Run command**

```
ejana
-Pplugins=beagle_reader,vmeson,event_writer
-Pnthreads=1
-Pnevents=10000
-Poutput=beagle.root
../data/beagle_eD.txt
-Pjana:debug_plugin_loading=1
```

Working with ROOT and CLI

- Run docker with bash
  - docker run -it -p 8888:8888 eicdev/epic:latest **bash**
  - To run jupyter lab environment **> jlab**

- **Run ROOT + C++ in notebooks:**




```
[1]: auto file = TFile::Open("beagle.root");

[2]: auto hst = file->Get<TDirectory>("vmeson")->Get<TH1>("h2_XQ2_true_log");

[3]: auto c = new TCanvas("myCanvasName","The Canvas Title",800,600);
     hst->Draw("colz");
     c->Draw();
```

