The background of the slide is a photograph of York Minster, a large Gothic cathedral, seen from a low angle. In the foreground, there are red brick buildings with white-framed windows and lush green trees and bushes. The sky is blue with scattered white clouds.

Introduction to ePIC Software

Stephen JD Kay
University of York

HSF-India/ePIC Workshop
14/05/25

Statement of Principles

- EIC software governed by a Statement of Principles -

Statement of Principles

- EIC software governed by a Statement of Principles -

EIC SOFTWARE: Statement of Principles

- 1 We aim to develop a diverse workforce, while also cultivating an environment of equity and inclusivity as well as a culture of belonging.**
- 2 We will have an unprecedented compute-detector integration:**
 - We will have a common software stack for online and offline software, including the processing of streamed data and its time-ordered structure.
 - We aim for autonomous alignment and calibration.
 - We aim for a rapid, near-real-time turnaround of the raw data to online and offline productions.
- 3 We will leverage heterogeneous computing:**
 - We will enable distributed workflows on the computing resources of the worldwide EIC community, leveraging not only HPC but also HPC systems.
 - EIC software should be able to run on as many systems as possible, while supporting specific system characteristics, e.g., accelerators such as GPUs, where beneficial.
 - We will have a modular software design with structures robust against changes in the computing environment so that changes in underlying code can be handled without an entire overhaul of the structure.
- 4 We will aim for user-centered design:**
 - We will enable scientists of all levels worldwide to actively participate in the science program of the EIC, keeping the barriers low for smaller teams.
 - EIC software will run on the systems used by the community, easily.
 - We aim for a modular development paradigm for algorithms and tools without the need for users to interface with the entire software environment.

- 5 Our data formats are open, simple and self-descriptive:**
 - We will favor simple flat data structures and formats to encourage collaboration with computer, data, and other scientists outside of NP and HEP.
 - We aim for access to the EIC data to be simple and straightforward.
- 6 We will have reproducible software:**
 - Data and analysis preservation will be an integral part of EIC software and the workflows of the community.
 - We aim for fully reproducible analyses that are based on reusable software and are amenable to adjustments and new interpretations.
- 7 We will embrace our community:**
 - EIC software will be open source with attribution to its contributors.
 - We will use publicly available productivity tools.
 - EIC software will be accessible by the whole community.
 - We will ensure that mission critical software components are not dependent on the expertise of a single developer, but managed and maintained by a core group.
 - We will not reinvent the wheel but rather aim to build on and extend existing efforts in the wider scientific community.
 - We will support the community with active training and support sessions where experienced software developers and users interact with new users.
 - We will support the careers of scientists who dedicate their time and effort towards software development.
- 8 We will provide a production-ready software stack throughout the development:**
 - We will not separate software development from software use and support.
 - We are committed to providing a software stack for EIC science that continuously evolves and can be used to achieve all EIC milestones.
 - We will deploy metrics to evaluate and improve the quality of our software.
 - We aim to continuously evaluate, adapt/develop, validate, and integrate new software, workflow, and computing practices.

EIC Software is:

- 1. Diverse
- 2. Intergrative
- 3. Heterogeneous
- 4. User-centred
- 5. Accessible
- 6. Reproducible
- 7. Collaborative
- 8. Agile

The "Statement of Principles" represents guiding principles for EIC Software. They have been submitted by the international EIC community. For a list of authors, see <https://aic.github.io/eic-software/statement-of-principles/>



Statement of Principles

- EIC software governed by a Statement of Principles -

EIC SOFTWARE: Statement of Principles



- 1 We aim to develop a diverse workforce, while also cultivating an environment of equity and inclusivity as well as a culture of belonging.**
- 2 We will have an unprecedented compute-detector integration:**
 - We will have a common software stack for online and offline software, including the processing of streamed data and its time-ordered structure.
 - We aim for autonomous alignment and calibration.
 - We aim for a rapid, near-real-time turnaround of the raw data to online and offline productions.
- 3 We will leverage heterogeneous computing:**
 - We will enable distributed workflows on the computing resources of the worldwide EIC community, leveraging not only HPC but also HPC systems.
 - EIC software should be able to run on as many systems as possible, while supporting specific system characteristics, e.g., accelerators such as GPUs, where beneficial.
 - We will have a modular software design with structures robust against changes in the computing environment so that changes in underlying code can be handled without an entire overhaul of the structure.
- 4 We will aim for user-centered design:**
 - We will enable scientists of all levels worldwide to actively participate in the science program of the EIC, keeping the barriers low for smaller teams.
 - EIC software will run on the systems used by the community, easily.
 - We aim for a modular development paradigm for algorithms and tools without the need for users to interface with the entire software environment.



- 5 Our data formats are open, simple and self-descriptive:**
 - We will favor simple flat data structures and formats to encourage collaboration with computer, data, and other scientists outside of NP and HEP.
 - We aim for access to the EIC data to be simple and straightforward.
- 6 We will have reproducible software:**
 - Data and analysis preservation will be an integral part of EIC software and the workflows of the community.
 - We aim for fully reproducible analyses that are based on reusable software and are amenable to adjustments and new interpretations.
- 7 We will embrace our community:**
 - EIC software will be open source with attribution to its contributors.
 - We will use publicly available productivity tools.
 - EIC software will be accessible by the whole community.
 - We will ensure that mission critical software components are not dependent on the expertise of a single developer, but managed and maintained by a core group.
 - We will not reinvent the wheel but rather aim to build on and extend existing efforts in the wider scientific community.
 - We will support the community with active training and support sessions where experienced software developers and users interact with new users.
 - We will support the careers of scientists who dedicate their time and effort towards software development.
- 8 We will provide a production-ready software stack throughout the development:**
 - We will not separate software development from software use and support.
 - We are committed to providing a software stack for EIC science that continuously evolves and can be used to achieve all EIC milestones.
 - We will deploy metrics to evaluate and improve the quality of our software.
 - We aim to continuously evaluate, adapt/develop, validate, and integrate new software, workflow, and computing practices.

The "Statement of Principles" represents guiding principles for EIC Software. They have been submitted by the international EIC community. For a list of activities, see <https://eic.github.io/activities/principles.html>

EIC Software is:

- 1. Diverse
- 2. Intergrative
- 3. Heterogeneous
- 4. User-centred
- 5. Accessible
- 6. Reproducible
- 7. Collaborative
- 8. Agile

- <https://eic.github.io/activities/principles.html>

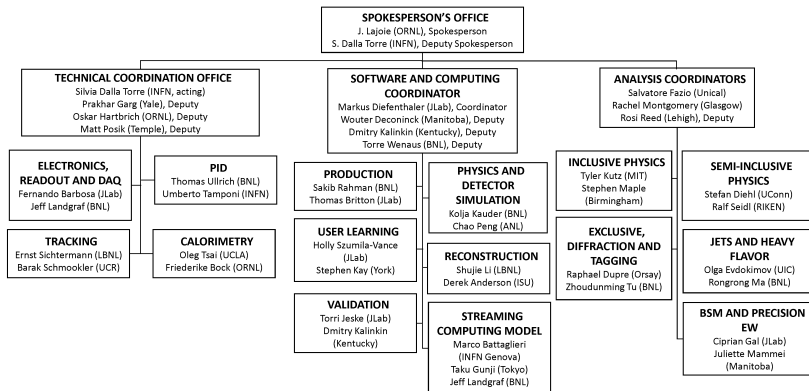
ePIC Software and Computing

- ePIC collaboration is structured with numerous working groups
 - <https://wiki.bnl.gov/EPIC/index.php?title=Collaboration>

ePIC Software and Computing

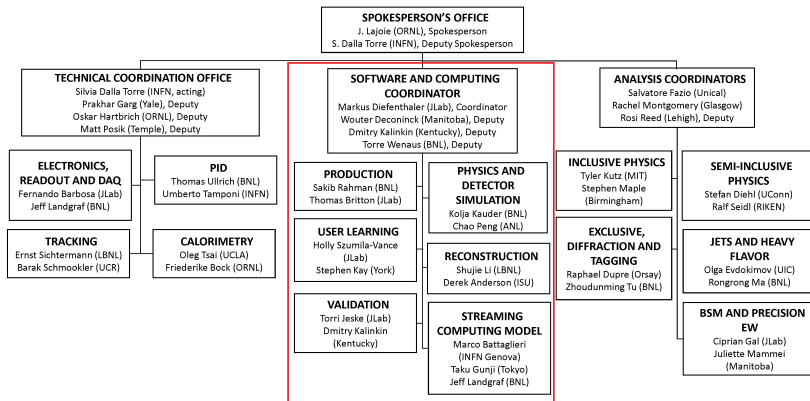
- ePIC collaboration is structured with numerous working groups

- <https://wiki.bnl.gov/EPIC/index.php?title=Collaboration>



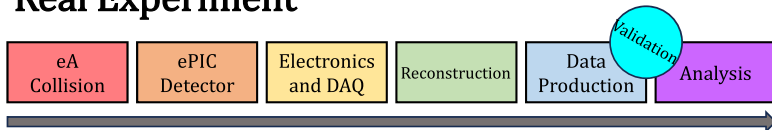
ePIC Software and Computing

- ePIC collaboration is structured with numerous working groups
 - <https://wiki.bnl.gov/EPIC/index.php?title=Collaboration>
- Software and Computing is a key pillar though!



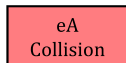
What is our Pipeline?

Real Experiment



What is our Pipeline?

Real Experiment



Electron and Ion beams collide.
Produce some particles.

What is our Pipeline?

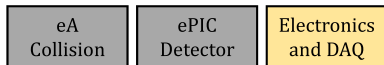
Real Experiment



Produced particles interact with the ePIC detector.

What is our Pipeline?

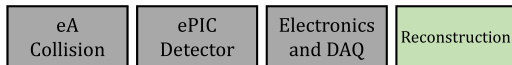
Real Experiment



Electronics and Data Acquisition systems produce and interpret electrical signals from detectors.

What is our Pipeline?

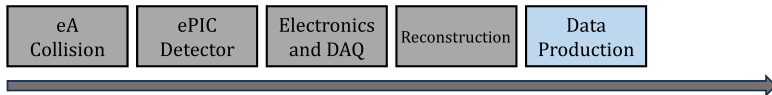
Real Experiment



Reconstruct signals from DAQ system.

What is our Pipeline?

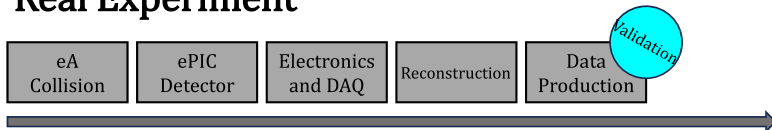
Real Experiment



Process reconstructed events
into data structure. Apply
calibrations, corrections.
Interpret events at a low level.

What is our Pipeline?

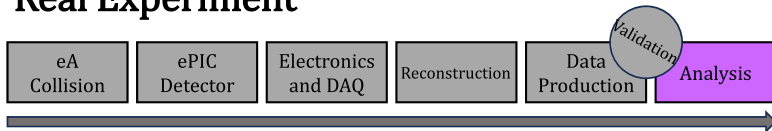
Real Experiment



Validation checks of data before analysis. QA of detector subsystems.

What is our Pipeline?

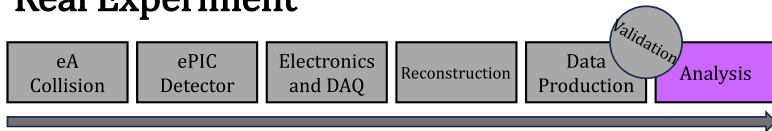
Real Experiment



Analyse the output data! Pick your physics channel of interest, try to find it!

What is our Pipeline?

Real Experiment

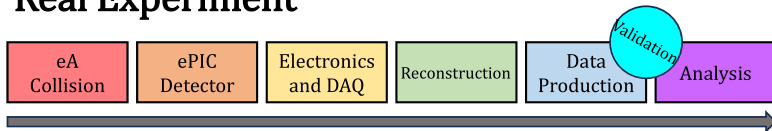


Analyse the output data! Pick your physics channel of interest, try to find it!

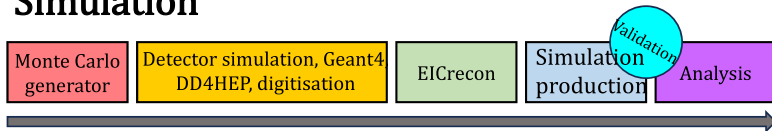
- So what does that look like for a **simulation**?

What is our **Simulation** Pipeline?

Real Experiment




Simulation



What is our **Simulation** Pipeline?

Simulation

Monte Carlo
generator




Input events from MC event
generator, particle guns.
Optionally, add physics
backgrounds

What is our **Simulation** Pipeline?

Simulation

Monte Carlo
generator

Detector simulation, Geant4
DD4HEP, digitisation

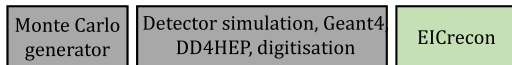


Geant4 simulations driven by
DD4HEP, output data in the EIC
data model (EDM4hep +
EDM4eic, described in PODIO)

Algorithms to transform the
Geant4 hits to mimic real detector

What is our **Simulation** Pipeline?

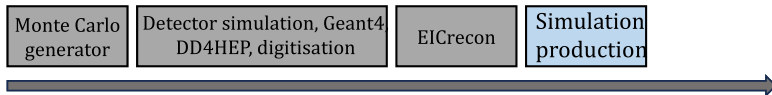
Simulation



Realistic reconstruction algorithms starting from raw detector output. Digitisation included to “look” like real data.

What is our **Simulation** Pipeline?

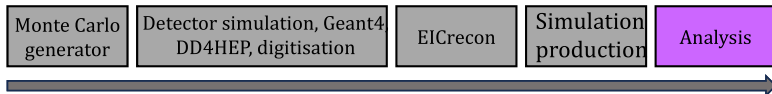
Simulation



Processing of the data using
calibration information,
corrections etc.

What is our **Simulation** Pipeline?

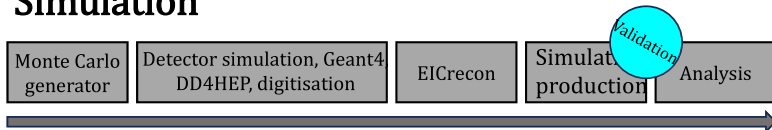
Simulation



User analyses in C++/ROOT or python/uproot. Facilitated by using flat, PODIO data model.

What is our **Simulation** Pipeline?

Simulation



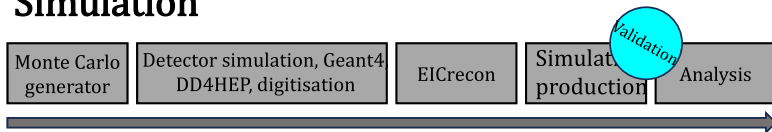
QA test, benchmarking etc.

Benchmarks for detectors and physics are included –

<https://eic.github.io/tutorial-developing-benchmarks/>

What is our **Simulation** Pipeline?

Simulation



QA test, benchmarking etc.

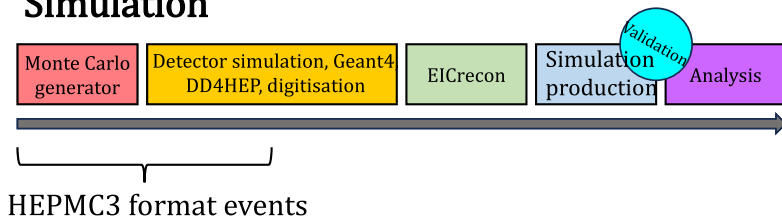
Benchmarks for detectors and physics are included –

<https://eic.github.io/tutorial-developing-benchmarks/>

- What do our inputs and outputs look like?

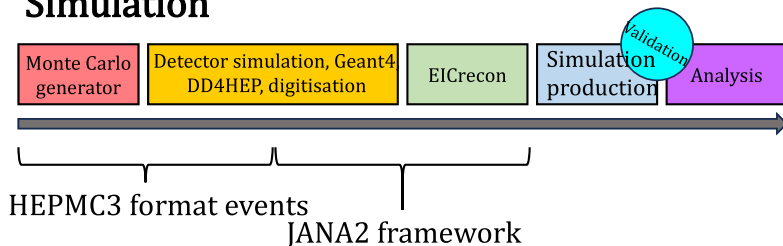
Mapping the Simulation Pipeline Inputs

Simulation



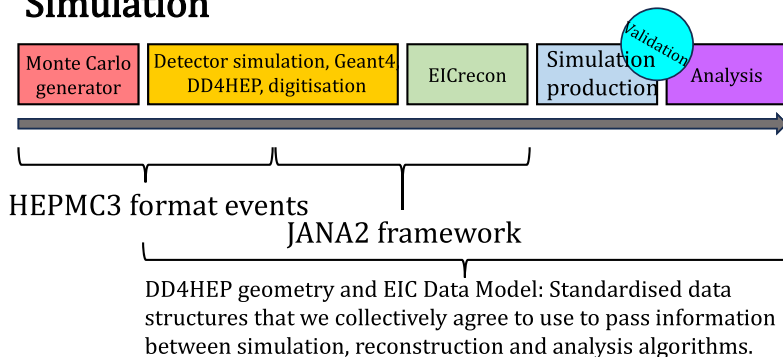
Mapping the Simulation Pipeline Inputs

Simulation



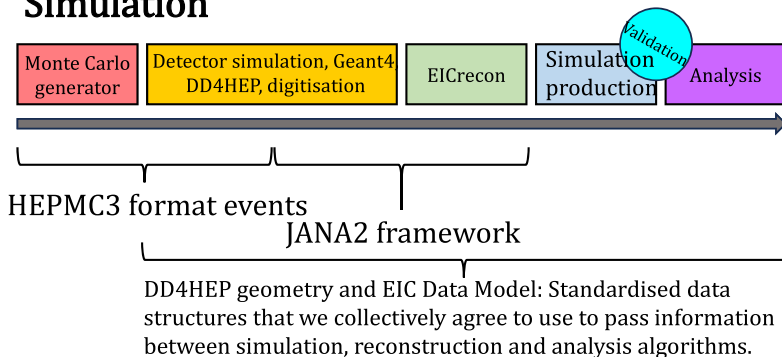
Mapping the Simulation Pipeline Inputs

Simulation



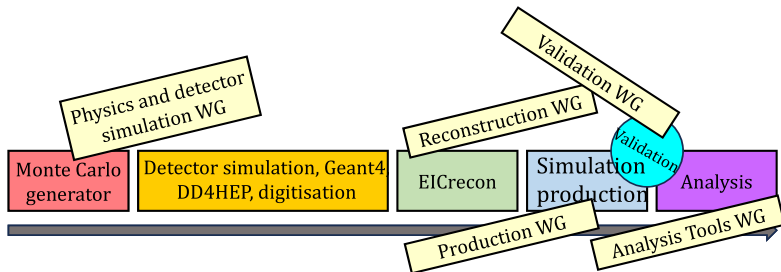
Mapping the Simulation Pipeline Inputs

Simulation



- How do the working groups fit in?

Interfacing with Working Groups



Simulation Campaigns

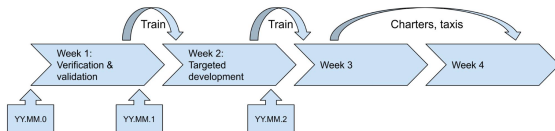
- With that reproducibility objective in mind, run regular simulation campaigns
- Objectives:

Simulation Campaigns

- With that reproducibility objective in mind, run regular simulation campaigns
- Objectives:
 - 1. Continuous deployment of software for detector/physics simulations
 - 2. Regular updates of simulation productions for detector/physics studies and geometry/algorithm development
 - 3. Validation and QA for simulation productions on datasets that require substantial time/resources

Simulation Campaigns

- With that reproducibility objective in mind, run regular simulation campaigns
- Objectives:
 - 1. Continuous deployment of software for detector/physics simulations
 - 2. Regular updates of simulation productions for detector/physics studies and geometry/algorithm development
 - 3. Validation and QA for simulation productions on datasets that require substantial time/resources



- **Train:** Major central campaign, fixed schedule
- **Charter:** Special interest runs
- **Taxis:** Bespoke runs

Latest Simulation Campaign

- The latest simulation campaigns are -

Latest Simulation Campaign

- The latest simulation campaigns are -
 - [25.04.1](#)
 - [25.03.1](#)
 - [25.02.0](#)

Latest Simulation Campaign

- The latest simulation campaigns are -
 - 25.04.1
 - 25.03.1
 - 25.02.0
- We will discuss the naming convention and the file structure of the campaigns in the next tutorial!
- Details and info on the simulation campaigns are available on the [Production Working Group Website](#)

Latest Simulation Campaign

- The latest simulation campaigns are -
 - [25.04.1](#)
 - [25.03.1](#)
 - [25.02.0](#)
- We will discuss the naming convention and the file structure of the campaigns in the next tutorial!
- Details and info on the simulation campaigns are available on the [Production Working Group Website](#)
- You can also check the [firehose channel](#) on Mattermost to see the latest files being produced

What can we study?

- We can use the detector simulation and our software stack to study many things

What can we study?

- We can use the detector simulation and our software stack to study many things
- **Broadly**, we are generally interested in learning something about **physics** or about our **detector**

Detector

Physics

What can we study?

- We can use the detector simulation and our software stack to study many things
- **Broadly**, we are generally interested in learning something about **physics** or about our **detector**

Detector

- Resolution
- Efficiency
- Digitisation
- Optimisation
- Geometry
- ...

Physics

What can we study?

- We can use the detector simulation and our software stack to study many things
- **Broadly**, we are generally interested in learning something about **physics** or about our **detector**

Detector

- Resolution
- Efficiency
- Digitisation
- Optimisation
- Geometry
- ...

Physics

- Kinematics
- Phase space
- Acceptance
- Algorithm performance
- ...

Use Cases

- There are numerous use cases we could envisage

Use Cases

- There are numerous use cases we could envisage

Detector Study

Use Cases

- There are numerous use cases we could envisage

Detector Study

Physics Study

Use Cases

- There are numerous use cases we could envisage

Detector Study

Physics Study

Validation

Use Cases

- There are numerous use cases we could envisage

Detector Study

Physics Study

Validation

**Implement
Detector**

Use Cases

- There are numerous use cases we could envisage

Detector Study

Physics Study

Validation

**Implement
Detector**

**Implement
Physics**

Use Cases

- There are numerous use cases we could envisage

Detector Study

Physics Study

Validation

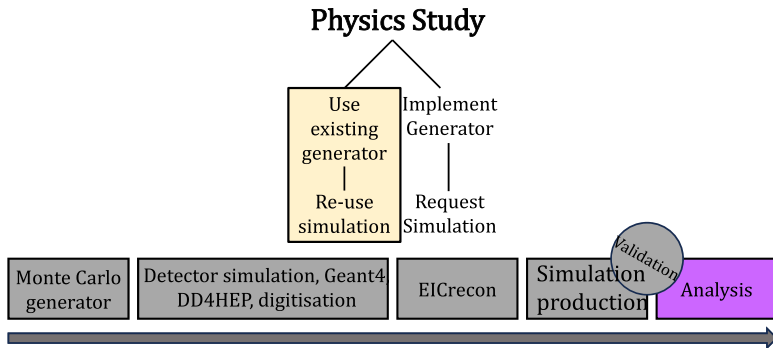
**Implement
Detector**

**Implement
Physics**

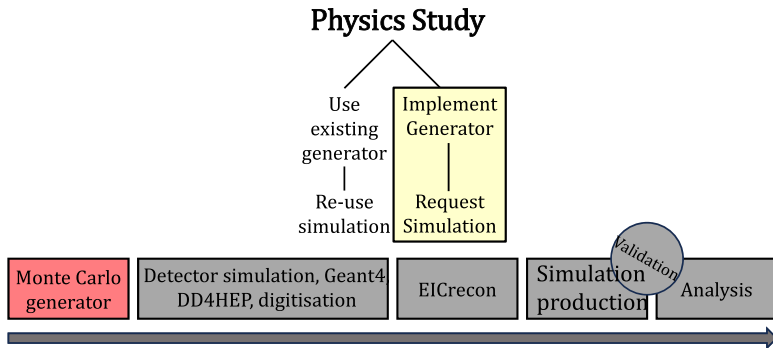
- Don't necessarily need full simulation chain in each case!

Use Case: Physics Study

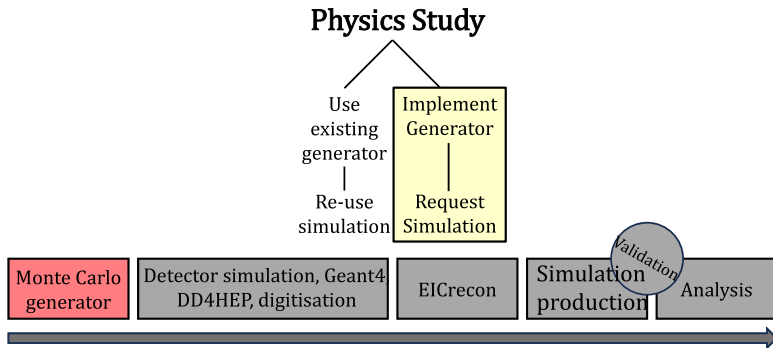
Use Case: Physics Study



Use Case: Physics Study



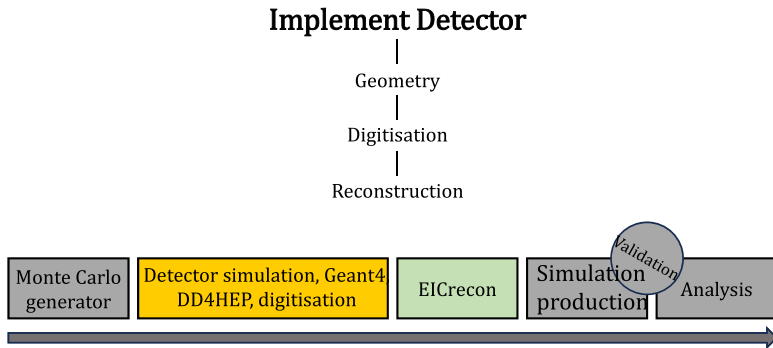
Use Case: Physics Study



- **Important** - For submission to simulation campaigns, new generators/studies should follow [input preprocessing guidelines](#) from the production WG

Use Case: Implement Detector

Use Case: Implement Detector



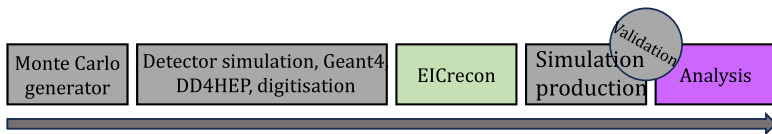
Use Case: Implement Physics

Use Case: Implement Physics

Implement Physics

Prototype analysis
on existing
reconstruction

Develop enhanced
reconstruction

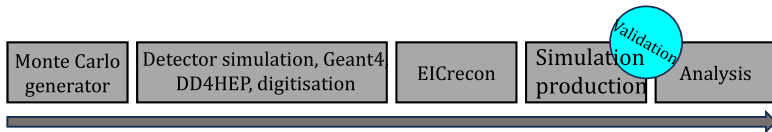


Use Case: Validation

Use Case: Validation

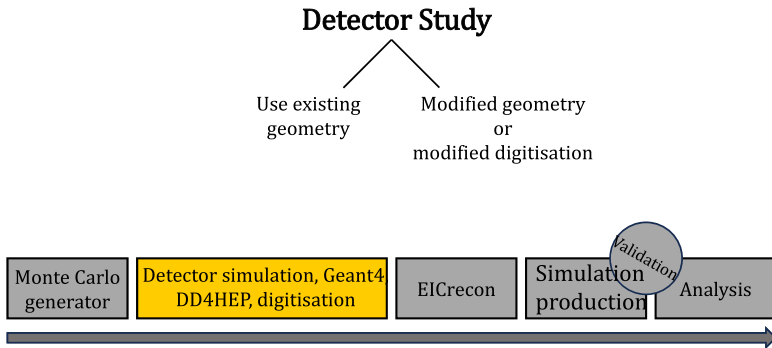
Validation

Compare the output
of physics and
detector studies



Use Case: Detector Study

Use Case: Detector Study



Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**

Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected

Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected

Contributing users



Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected

Contributing users

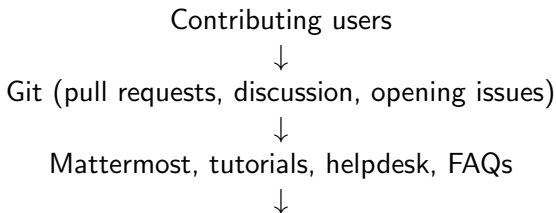


Git (pull requests, discussion, opening issues)



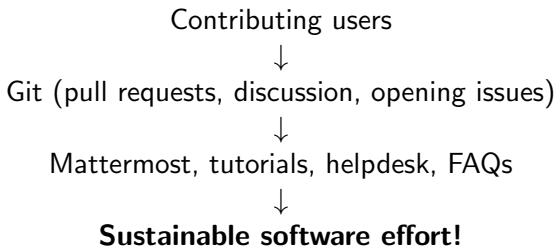
Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected



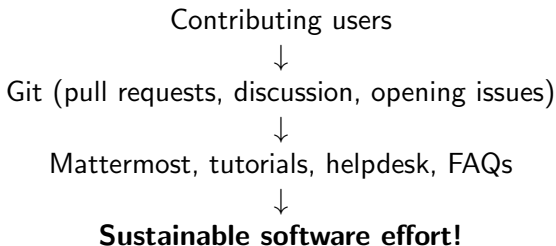
Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected



Collaborative Software

- Principles outlined earlier aren't standalone
- **Interconnected with each other**
 - Sustainability, reproducibility and collaborative all connected



- Where can we access some of the collaborative tools and material available to us?

eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components

eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components
- It contains a common set of EIC software pre-installed

eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components
 - It contains a common set of EIC software pre-installed
- ddsim/npsim: The ePIC detector geometry and simulation

[illegible]

eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components
 - It contains a common set of EIC software pre-installed
- EICrecon: Our reconstruction toolkit

```
jay@jay:~$ singularity exec --writable --bind /tmp/.singularity /eic-shell eicrecon -h
Usage:
  eicrecon [options] source1 source2 ...

Description:
  Command-line interface for running JANA plugins. This can be used to
  read in events and process them. Command-line flags control configuration
  while additional arguments denote input files, which are to be loaded and
  processed by the appropriate EventSource plugin.

Options:
  -h --help                Display this message
  -v --version             Display version information
  -j --javaname             Display JANA version information
  -c --config              Display configuration parameters
  -l --loadconfig <file>  Load configuration parameters from file
  -d --dumpconfig <file>  Dump configuration parameters to file
  -b --benchmark           Run in benchmark mode
  -L --list-factories      List all the factories without running
  -p --plugin <name>       Specify a configuration parameter
  -P --pluginparam <name> Specify a parameter value for a plugin

  --list-default-plugins  List all the default plugins
  --list-available-plugins List plugins at $JANA_PLUGIN_PATH and $EICRECON_PATH
  --plugins=<values>      Comma-separated list of extra plugins to load
  --plugins-to-ignore=<values> Comma-separated list of plugins to ignore

Examples:
  eicrecon -Pplugin=plugin.plugin1.plugin2 -bthreads=4 infile.root
  eicrecon -Pplugin=plugin1.plugin2 -bthreads=4 infile.root

jag_dev@jay:~$ singularity exec --writable --bind /tmp/.singularity /eic-shell
```

cic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components
- It contains a common set of EIC software pre-installed

Afterburner: A way to quickly apply beam effects to simulated events

```
jag_dsw > java -jar jupyter-stephon-zohary-ibpwc-20ac-7dc6-dadeflange-28af5-bfddbf-vr165 ibpwcsh -h
EIT afterthoughts.

Process heap(2)C and other compatible files adding crossing angle and base effects.

Usage: ibpwc [OPTS] [-input file...]
```

Diagnostics:

```
-input File TEXT ...      Input file
```

Options:

```
-h,-help                  Print this help message and exit
-o,-output TEXT           Base name for Output files (||| no extension)
-p,-preset TEXT           IPR 1DPS high divergence/default); 1 IPRs high acceptance; 2 IPRs out; 3 IPRs right divergence.... (H = see below)
-i,-invert TEXT           Input format auto {default}, hepm2, hepm3, ipw, def, gfi, tfermost, root
-o,-out-format TEXT       Output format: hepm3 {default}, tfermost, root, hepm2, GSI, rowe [I/O events file is saved]
-s,-start INT             Start event number [all previous are skipped]
-e,-event INT            End event limit (end processing after this event)
-l,-limit LIMIT           Limit number of events to process. (Shutdown after this number of parsed events).
-a,-afterburner TEXT     Afterburner is applied
-m,-plot-off              Don't produce validation plots
-x,-exit-ca               Check existing crossing angle and exit if Coloured Data
-V,-version              Display program version information and exit
```

Preset flag, values [1..12] set config and auto determine energy source file:

```
0 IPRs, high divergence, auto read energy {default},
1 IPRs, high acceptance, auto read energy
2 IPRs, auto, auto read energy
3 IPRs, high divergence, auto read energy {default},
4 IPRs, high acceptance, auto read energy
5 IPRs, auto, auto read energy
```

The other options auto energy settings manually, not checking the source file:

```
ipw_hdic_41x5, ipw_ibdic_180x5, ipw_hdic_180x10, ipw_hdic_275x9, ipw_hdic_275x18
ipw_hdic_41x5, ipw_ibdic_180x5, ipw_hdic_180x10, ipw_hdic_275x9, ipw_hdic_275x18
ipw_hdic_41x5, ipw_ibdic_180x5, ipw_hdic_180x10, ipw_hdic_275x9, ipw_hdic_275x18
ipw_hdic_41x5, ipw_ibdic_180x5, ipw_hdic_180x10, ipw_hdic_275x9, ipw_hdic_275x18
ipw_hdic_41x5, ipw_ibdic_180x5, ipw_hdic_180x10, ipw_hdic_275x9, ipw_hdic_275x18
Example of manual configuration setting:
ibpwc -v ipw_hdic_180x5 source_file.hepmc
```

jag_dsw > java -jar jupyter-stephon-zohary-ibpwc-20ac-7dc6-dadeflange-28af5-bfddbf-vr165

eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components

- It contains a common set of EIC software pre-installed

ROOT: A way to open and analyse simulated output

```
jagdev@jaggyr-steph-jdkay:~$ singularity exec --writable --nv --net --hostname root -t root -h

usage: root [-h m] [-w W] [-e E] [-n N] [-t T] [-q Q] [-l L] [-v V]
            [--config CONFIG] [-h help] [--version VERSION]
            [--notebook NOTEBOOK] [--web WEB] [--web-type WEB=TYPE]
            [--suboff SUBOFF]
            [dir] [data.root...data.root] [file.C...file.C]
            [file.C.so...file.C.so]

ROOT is Object-Oriented Technologies.

root is an interactive interpreter of C++ code. It uses the ROOT framework. For more information on ROOT, please refer to
An extensive Users Guide is available from that site (see below).

OPTIONS:
  -h          Run in batch mode without graphics
  -e          Exit on exceptions
  -a          Executes the command passed between single quotes
  -m          Do not execute login and logout macros as specified in .rootrc
  -t          Enable thread-safety and implicit multi-threading (DST)
  -q          Exit after processing command line macro files
  -l          Do not show the ROOT banner
  -a          Show the ROOT splash screen
  -c          print ./configure options
  --config    Show summary of options
  -h, ?, --help Show the ROOT version
  --version   Execute ROOT notebook
  --notebook  Use web-based display for graphics, browser, geometry
  --web       Use the specified web-based display such as chrome, firefox, qjs
  --web-type  For more options see the documentation of TROOT::SetWebDisplay()
  --subtypes  Stable any kind of web-based display
  [dir]       If dir is a valid directory cd to it before executing
  [data.root...data.root] Open the given ROOT file(s); remote protocols (such as http://) are supported
  [file.C...file.C] Executes the ROOT macro file(s).C...file.C
  [file.C.so...file.C.so] Compilation flags as well as no arguments can be passed, see format in https://root.cern/manual/root_macros_and_shared_libraries/
  [file.C.so...file.C.so] They should be already-compiled ROOT macros (shared libraries) or
  [file.C.so...file.C.so] regular user shared libraries e.g. userlib.so with a function userlib(org)

jagdev@jaggyr-steph-jdkay:~$ singularity exec --writable --nv --net --hostname root -t root -h
```


eic-shell: A Common Environment

- eic-shell is a singularity/docker container with a curated selection of software components

- It contains a common set of EIC software pre-installed

Common software stack → Reproducible!

The Full Software Stock

- Of course, there's a lot more software

The Full Software Stock

- Of course, there's a lot more software
- Here's the full spider's web

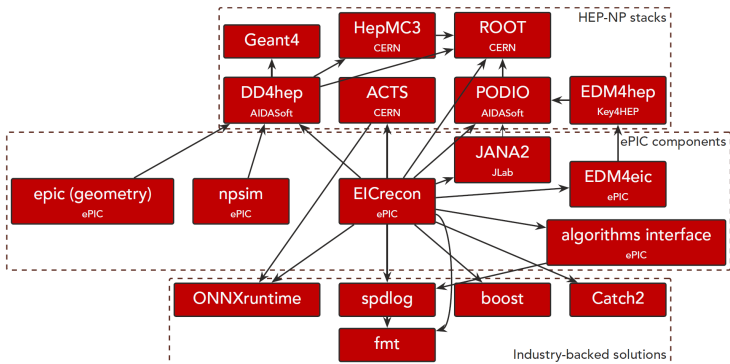


Image - Dmitry Kalinkin, CHEP 2024

Extra Resources

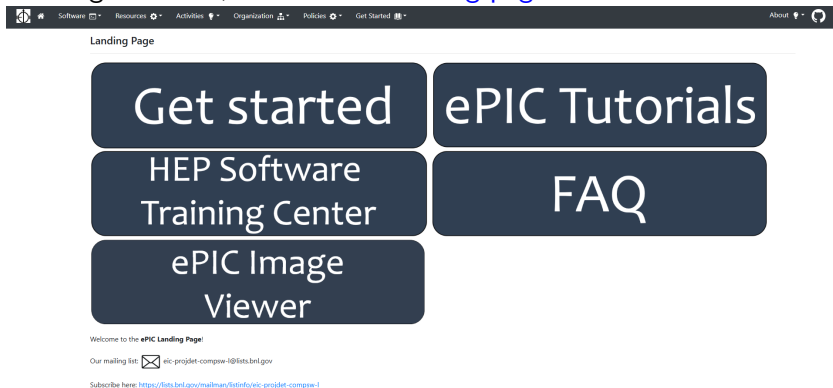
- There are a range of additional resources available

Start from <https://eic.github.io/>

Extra Resources

- There are a range of additional resources available

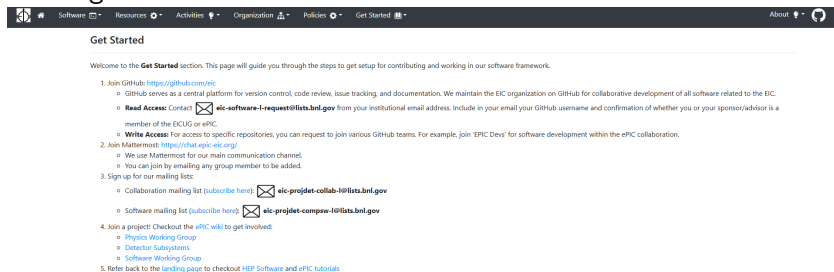
Under get started, we have the [landing page](#)



Extra Resources

- There are a range of additional resources available




Clicking “Get Started”



The screenshot shows the top navigation bar of the EIC website with links for Software, Resources, Activities, Organization, Policies, and Get Started. The 'Get Started' page content includes a welcome message and a numbered list of steps for getting involved, such as joining GitHub, contacting the EIC software team, joining Mattermost, signing up for mailing lists, and checking out project wikis.

Get Started

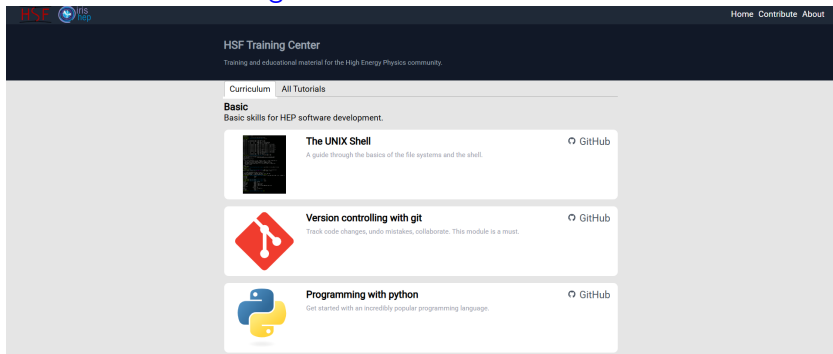
Welcome to the **Get Started** section. This page will guide you through the steps to get setup for contributing and working in our software framework.

1. Join GitHub: <https://github.com/eic>
 - GitHub serves as a central platform for version control, code review, issue tracking, and documentation. We maintain the EIC organization on GitHub for collaborative development of all software related to the EIC.
 - **Read Access:** Contact  eic-software-l-request@lists.bnl.gov from your institutional email address. Include in your email your GitHub username and confirmation of whether you or your sponsor/advisor is a member of the EICUG or ePIC.
 - **Write Access:** For access to specific repositories, you can request to join various GitHub teams. For example, join 'ePIC Devs' for software development within the ePIC collaboration.
2. Join Mattermost: <https://chat.eic-eic.org/>
 - We use Mattermost for our main communication channel.
 - You can join by emailing any group member to be added.
3. Sign up for our mailing lists:
 - Collaboration mailing list ([subscribe here](#)):  eic-projdet-collab-l@lists.bnl.gov
 - Software mailing list ([subscribe here](#)):  eic-projdet-compaw-l@lists.bnl.gov
4. Join a project! Check out the [ePIC wiki](#) to get involved:
 - [Physics Working Group](#)
 - [Detector Subsystems](#)
 - [Software Working Group](#)
5. Refer back to the [landing page](#) to checkout HEP Software and [ePIC tutorials](#)

Extra Resources

- There are a range of additional resources available

HSF Software Training Centre



The screenshot shows the HSF Training Center website. The header includes the HSF and IRIS HEP logos on the left and navigation links (Home, Contribute, About) on the right. The main content area is titled "HSF Training Center" with a subtitle "Training and educational material for the High Energy Physics community." Below this, there are tabs for "Curriculum" and "All Tutorials". Under the "Curriculum" tab, the "Basic" section is highlighted, with the description "Basic skills for HEP software development." Three tutorial cards are listed: "The UNIX Shell" (with a terminal icon), "Version controlling with git" (with a git logo icon), and "Programming with python" (with a python logo icon). Each card includes a brief description and a link to the GitHub repository.

HSF IRIS HEP

Home Contribute About


HSF Training Center


Training and educational material for the High Energy Physics community.


Curriculum All Tutorials

Basic

Basic skills for HEP software development.

**The UNIX Shell**
A guide through the basics of the file systems and the shell.
[GitHub](#)

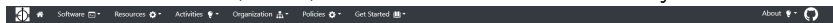
**Version controlling with git**
Track code changes, undo mistakes, collaborate. This module is a must.
[GitHub](#)

**Programming with python**
Get started with an incredibly popular programming language.
[GitHub](#)

Extra Resources

- There are a range of additional resources available

Numerous [tutorials](#), note, different to the next few days!



ePIC Tutorials

Future tutorials schedule:

Please join the [Mattermost Software Tutorials](#) channel for updates/announcements and questions about tutorials.

Note that the tutorials as presented below are not intended to be followed in a strict sequence. New users should start with the "Setting up an environment" tutorial. Beyond this, we encourage users to pick and choose the tutorials on topics they want to explore.

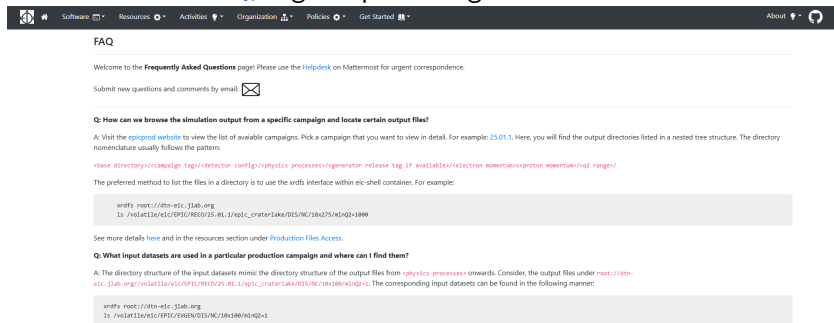
Current tutorials are summarised in the table below -

| Tutorial | Difficulty | Tags | Description | Resources |
|---|-------------------|--------------------------------------|---|---|
| Setting up an environment | Beginner | [Setup] [Environment] | Get started with the ePIC software environment | Video 1 • Video 2 |
| Analysis and simulation output | Beginner/Advanced | [Analysis] [Data] | Learn how to analyze simulation data | Video Videos2 (=8C.B&CI) |
| Simulating detectors | Expert | [Simulation] [Detector] [DD4hep] | Development of detector geometry using DD4hep | Video |
| Modifying geometry and digitization | Advanced | [Geometry] [Digitization] | Learn to customize detector configurations | - |
| Understanding simulation output | Advanced | [Simulation] [Data] | Deep dive into simulation data structure | Video (jn*#sp0*) |
| Getting started with physics analysis | Advanced | [Physics] [Analysis] | Physics analysis introduction | Video (NjNB#8) |
| Inclusive kinematics reconstruction | Advanced | [Reconstruction] [Kinematics] | Learn kinematics reconstruction techniques | Video (209gA) |
| Reconstruction algorithms | Expert | [Algorithms] [Reconstruction] | Study different reconstruction approaches | Video 1 • Video 2 |
| Developing benchmarks | Expert | [Benchmarking] [Performance] | Learn to create performance benchmarks | - |
| Simulations with ddsim and geant4 | Advanced | [Simulation] [Geant4] [ddsim] | Run simulations with different frameworks | Video 1 • Video 2 |
| Reconstruction framework | Expert | [Framework] [JANA2] [Reconstruction] | Working with the JANA2 reconstruction framework | Video 1 • Video 2 |
| Analysis bootcamp | Advanced | [Python] [Analysis] [Bootcamp] | Python-based analysis techniques | - |

Extra Resources

- There are a range of additional resources available


There is also a **FAQ**, a good place to get involved!



The screenshot shows the Mattermost web interface. At the top is a navigation bar with links for Software, Resources, Activities, Organization, Policies, Get Started, and About. Below this is a header for the 'FAQ' section. The main content area contains a welcome message, a link to submit questions, and a specific FAQ entry. The FAQ entry asks how to browse simulation output and provides a detailed answer with a code example for using the xrdfs interface.

FAQ

Welcome to the **Frequently Asked Questions** page! Please use the [Helpdesk](#) on Mattermost for urgent correspondence.

Submit new questions and comments by email: 

Q: How can we browse the simulation output from a specific campaign and locate certain output files?

A: Visit the [epicprod website](#) to view the list of available campaigns. Pick a campaign that you want to view in detail. For example: [25.01.1](#). Here, you will find the output directories listed in a nested tree structure. The directory nomenclature usually follows the pattern:

```
<base directory>/campaign tag/<detector config>/physics processes/<generator release tag if available>/electron momentum<proton momentum>/cq2 range/
```

The preferred method to list the files in a directory is to use the xrdfs interface within eic-shell container. For example:

```
xrdfs root://dn-eic.jlab.org
ls /volatl1e/eic/EPIC/REC0/25-01.1/epic_craterlake/015/MC/18x175/nInQ2=1000
```

See more details [here](#) and in the resources section under [Production Files Access](#).

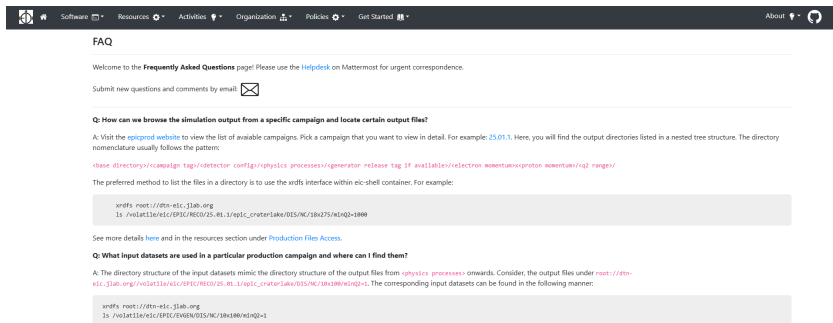
Q: What input datasets are used in a particular production campaign and where can I find them?

A: The directory structure of the input datasets mimic the directory structure of the output files from `<physics processes>` onwards. Consider, the output files under `root://dn-eic.jlab.org/volatl1e/eic/EPIC/REC0/25-01.1/epic_craterlake/015/MC/18x180/nInQ2=1`. The corresponding input datasets can be found in the following manner:

```
xrdfs root://dn-eic.jlab.org
ls /volatl1e/eic/EPIC/GEN0/015/MC/18x180/nInQ2=1
```


Extra Resources

- There are a range of additional resources available



The screenshot shows a Mattermost web interface. The top navigation bar includes links for Software, Resources, Activities, Organization, Policies, Get Started, and About. The main content area is titled 'FAQ' and contains the following text:

Welcome to the **Frequently Asked Questions** page! Please use the [Helpdesk](#) on Mattermost for urgent correspondence.

Submit new questions and comments by email: 

Q: How can we browse the simulation output from a specific campaign and locate certain output files?

A: Visit the [epicprod website](#) to view the list of available campaigns. Pick a campaign that you want to view in detail. For example: [25.01.1](#). Here, you will find the output directories listed in a nested tree structure. The directory nomenclature usually follows the pattern:

```
<base directory>/campaign tag/<detector config>/physics processes/<generator release tag if available>/electron momentum xproton momentum/cq range/
```

The preferred method to list the files in a directory is to use the xrdfs interface within eic-shell container. For example:

```
xrdfs root://dn-eic.jlab.org
ls /volatl1e/eic/EPIC/REC0/25_01.1/epic_crater1aka/015/NC/18x275/n1nQ2=1000
```

See more details [here](#) and in the resources section under [Production Files Access](#).

Q: What input datasets are used in a particular production campaign and where can I find them?

A: The directory structure of the input datasets mimic the directory structure of the output files from `<physics processes>` onwards. Consider, the output files under `root://dn-eic.jlab.org/volatl1e/eic/EPIC/REC0/25_01.1/epic_crater1aka/015/NC/18x100/n1nQ2=1`. The corresponding input datasets can be found in the following manner:

```
xrdfs root://dn-eic.jlab.org
ls /volatl1e/eic/EPIC/ENG0/015/NC/18x100/n1nQ2=1
```

But of course, there's one very important resource to mention...

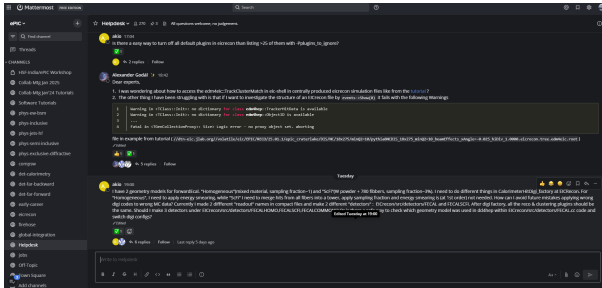
Mattermost

- Mattermost: useful place to get involved and find things out
- Email from me earlier with an invite link - **please join!**

Mattermost

- Mattermost: useful place to get involved and find things out
- Email from me earlier with an invite link - **please join!**

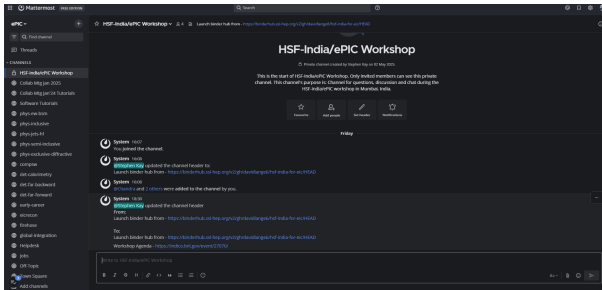
Got a software problem? Ask in the Helpdesk



Mattermost

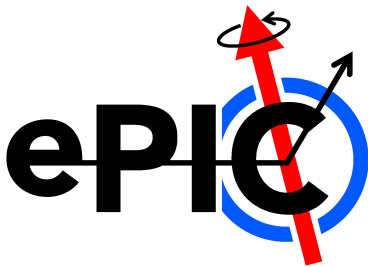
- Mattermost: useful place to get involved and find things out
- **Email from me earlier with an invite link - please join!**

We also have our own channel for this workshop! I've invited everyone who joined Mattermost - I'll do another pass after the session today!



If you did not receive an invite link or you aren't in the HSF-India/ePIC channel, please let me know!

Before we move on, any questions?



stephen.kay@york.ac.uk

Alright then, enough listening to talks for one day.
Time to get to work on some software!