



"ePIC Software Components: Building with the Community"

Dmitry Kalinkin (University of Kentucky)
for the ePIC Software and Computing Team
ePIC Computing & Software Review
September 26-27 2024

Review charge

- Is there a comprehensive and cost-effective short and long-term plan for the software and computing of the experiment?
 - The pre detector technical design report (TDR) is scheduled to be delivered in 2025. Are the resources for software and computing sufficient to deliver the TDR?
 - Is the design of the ePIC computing model and resource needs assessment adequate for this stage of the project?
 - Is the ePIC computing model flexible? Can it evolve and integrate new technologies in software and computing?
- Are the plans for software and computing integrated with the HEP/NP community developments, especially given data taking in ten years?
- Are the ECSJI plans to integrate into the software and computing plans of the experiment sufficient?
- Are the plans for integrating international partners' contributions adequate at this stage of the project?

Our philosophy

- We focus on **modern scientific software & computing practices** to ensure the **long-term success of the EIC scientific program** throughout all CD milestones
 - Strong emphasis on modular, orthogonal tools
 - Integration with HTC, CI workflows, and enable use of standard data science toolkits
- We **leverage cutting edge sustainable community software** where appropriate, **avoiding the “not invented here” syndrome**
 - Can build our software on top of a mature, well-supported, and actively developed software stack by using modern community tools, e.g. from CERN, the HPC community, and the data science community
 - Actively collaborate with external software projects, while externalizing some support burden to external projects
- We embrace these practices today to avoid starting our journey to EIC with technical debt
- **We are writing software for the future, not the lowest common denominator of the past!**

EIC Software: 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 HTC 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.

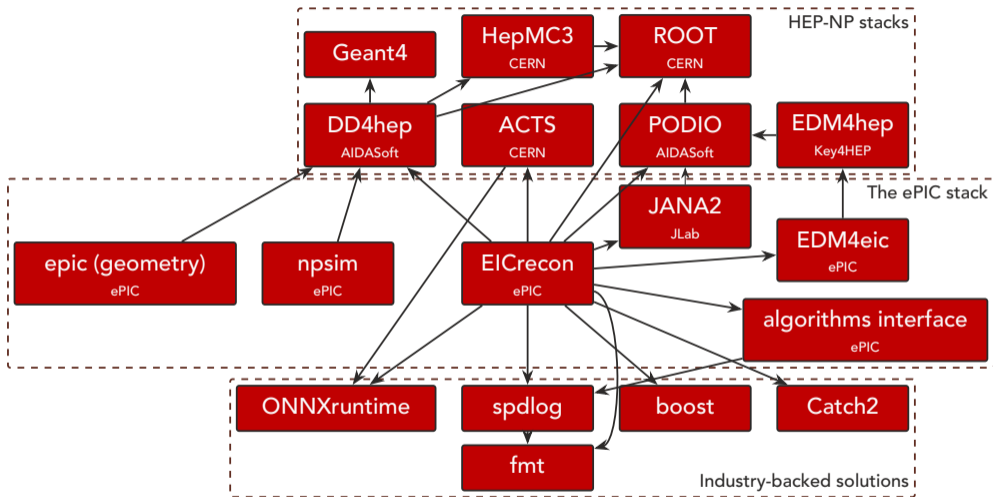
The "Statement of Principles" represent guiding principles for EIC Software. They have been endorsed by the international EIC community. For a full of underlying use cases.

Advertised at

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

- agile development
- production-ready software stack
- meeting near-term needs of ePIC
- timeline-based prioritization
- user-centered design

ePIC Software Stack



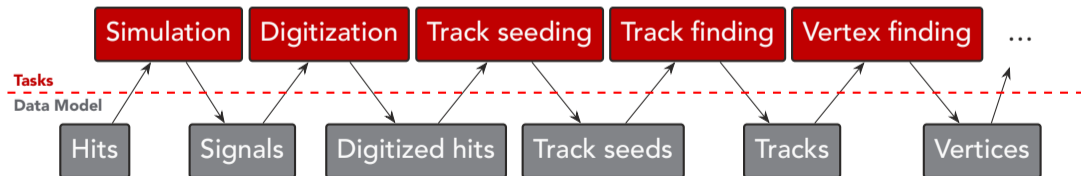
- Modular Simulation, Reconstruction and Analysis Software Stack powered by the NP-HEP community tools

Giving back to the community

- Contributed fixes for
 - **ACTS**: improvements in tracking and infrastructure, with plans for more development
 - **DD4hep**: general bugs, HepMC3 ROOT and gzip support, new readout segmentation types, ...
 - **EDM4hep**: build infrastructure
 - **Geant4**: addressed bugs UI, performance optimizations
 - **HepMC3**: addressed bugs in ROOT/XRootD IO, build infrastructure
 - **PODIO**: API improvements, build infrastructure, memory access semantics, **streaming over network**
 - **ROOT**: addressed bugs in UI, regressions in ROOT file handling
 - **Snakemake**: addressed general bugs
 - **Spack**: maintenance work on the package set
 - **UpROOT**: S3 object storage support (for legacy test setup at BNL/SDCC)
- EICrecon provides an ever expanding set of reusable LGPL3-licensed algorithms
- For HEP-NP projects, we encourage our users to submit their questions and problems to ePIC Helpdesk first

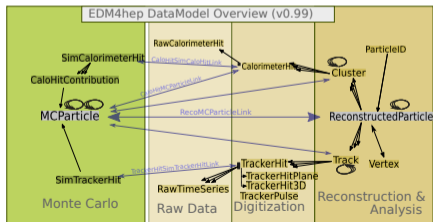
Data model in data-processing framework

- “If you’ve chosen the right data structures and organized things well, the algorithms will almost always be self-evident.” – Rob Pike, 1989
- We spend a lot of time on educating our collaborators on understanding and using of our Event Data Model (e.g. [talk by W. Deconinck](#))
- Event model additions and modifications have to be discussed in weekly ePIC Software & Computing meeting and reviewed closely on GitHub.



PODIO

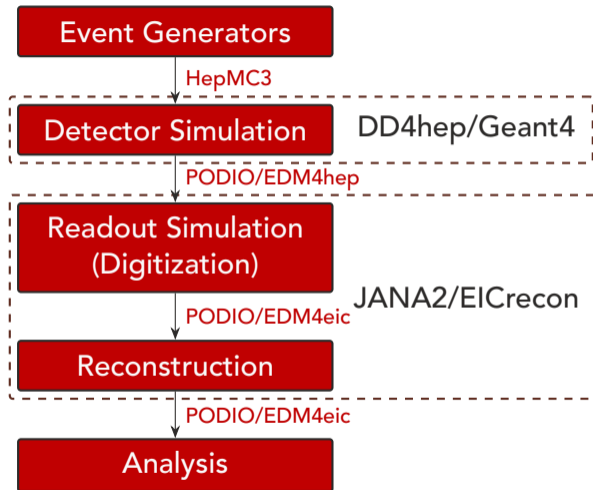
- ... is a YAML-based format for defining data structures
- ... has a C++/Python API
- ... restricts nesting of data structures (hence, Plain Old Data) in favor of the relational model
- ... does not impose row-wise/columnar ordering for in-memory data
- ... implements on-disk representation in ROOT (TTree or RNTuple) and SIO formats, but also allows other implementations



EDM4hep / edm4hep.yaml

```
394 edm4hep::CalorimeterHit:
395   Description: "Calorimeter hit"
396   Author: "EDM4hep authors"
397   Members:
398     - uint64_t cellID // detector specific (geometrical) cell id
399     - float energy [GeV] // energy of the hit
400     - float energyError [GeV] // error of the hit energy
401     - float time [ns] // time of the hit
402     - edm4hep::Vector3f position [mm] // position of the hit in world coordinates
403     - int32_t type // type of hit
```

Simulation framework



- Common simulation and reconstruction geometry is defined with DD4hep
- DDG4 component: interface to Geant4 with first-class PODIO/EDM4hep support
- Embedding of backgrounds (beam-gas interaction, synchrotron radiation) is available as a HepMC preprocessor
- EICrecon implements ePIC detector-specific response and digitization simulation steps

Reconstruction framework

Data-processing algorithms organized

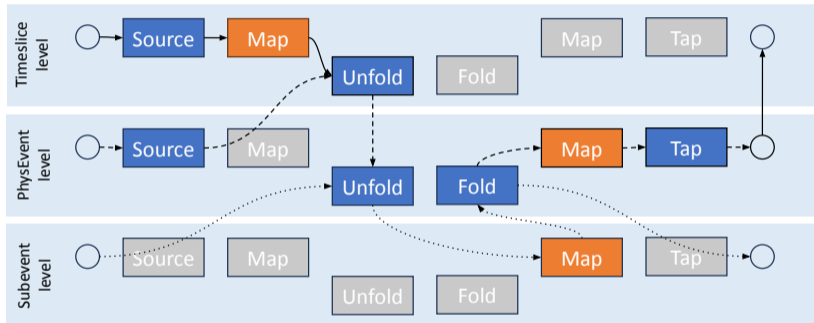
- JANA2 framework is a rewrite of JANA multithreaded framework with focus on Streaming DAQ and heterogeneous hardware support
- EDM4eic defines PODIO structures for data that can be passed from algorithm to algorithm
- EICrecon uses JANA2 as backbone for passing immutable data within processed events and timeframes
- Modular algorithms for tracking, vertexing, calorimetry, jet reconstruction, PID

Near-term goals

- External algorithm wiring configurability for JANA2
- Metadata handling (extend support in JANA2)
- Consistent conditions DB interface (evaluate `nopayloaddb` – HSF reference implementation)

Frame-based reconstruction

Advanced event processing topology implemented in JANA2



- Source
 - JEventSource::GetEvent()
- Map
 - JOmniFactory::Process()
 - JEventProcessor::Preprocess()
 - JEventSource::Preprocess()
 - JEventUnfolder::Preprocess()
 - JEventFolder::Preprocess()
- Tap
 - JEventProcessor::Process()
- Unfold
 - JEventUnfolder::Unfold()
- Fold
 - JEventFolder::Fold()

→ Timeslice
- - - - - Event
· · · · · Subevent

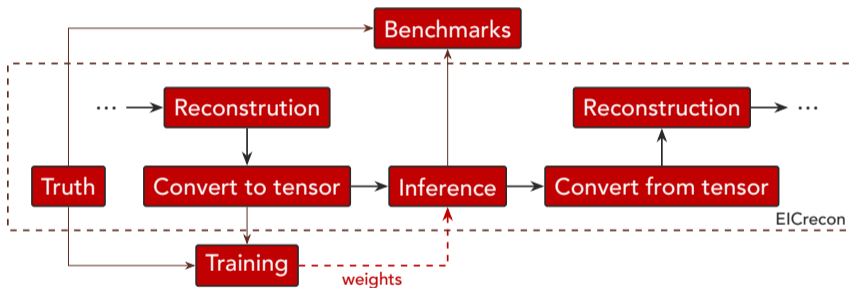
Parallel Sequential

(Diagram by Nathan Brei)

Artificial Intelligence in ePIC

Success stories:

- Momentum calibration for low- Q^2 tagger implemented using TMVA
- Centralized integration to ONNX inference models available

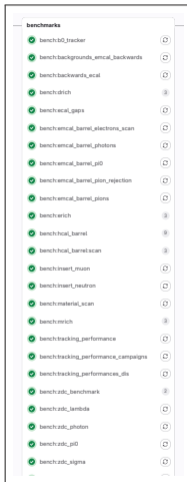
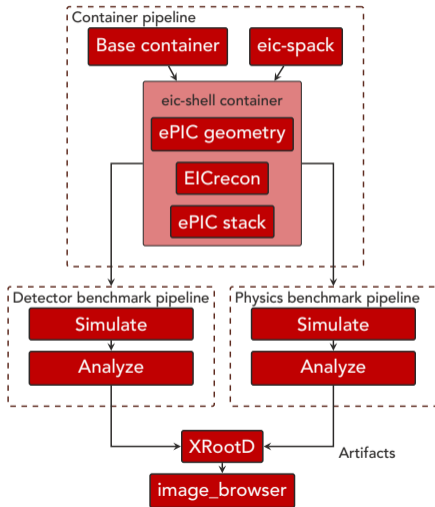


- Many successful applications of AI/ML outside of the reconstruction
Powered by ease of integration: PODIO + Uproot → Tensorflow/Torch
- We are investigating integration of AI/ML workflows into our productions

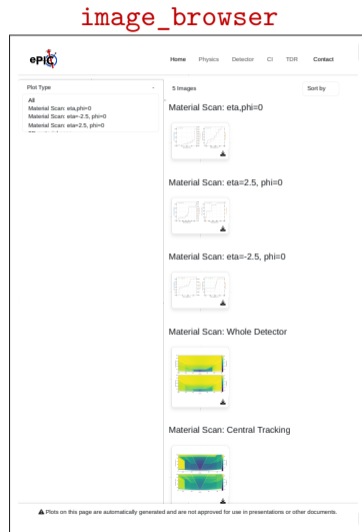
Deployment and Workflows

- Software-defined infrastructure
 - Publically developed workflows: GitHub Actions, GitLab CI, Snakemake, Bash script workflows for running tests, container building, simulations, data analysis
 - Arifacts are publically accessible (Downloadable Artifacts and Web Pages, XRootD)
- Continuous Integration
 - Automatic coding suggestions (formatting, static and dynamic analyses)
 - Automatic comparisons fo simulation/reconstruction outputs to the reference
- Continuous Delivery
 - Build of Docker/Singularity container with current versions of software
 - Export of geometry as image renders and to CAD (.step) format
 - Export of algorithm wiring graphs (Graphviz .dot)
 - Material maps, Calibrations, Machine Learning weights
 - Sub-detector performance plots
 - ePIC Detector Physics performance plots (Analysis-as-a-service)

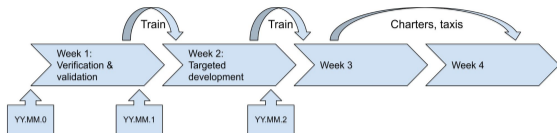
Validation workflow



Running in a dedicated
GitLab CI
(AMD EPYC 7H12 - 256
threads, 512 GB)



Simulation Campaigns



- Regular schedule for software releases and accompanying simulation campaigns
- Delivered **29** campaigns since October 2022
- Intuitive path design provides ease of access

Path: /EPIC/RECO/24.09.0/epic_craterlake/DIS/NC/10x100/minQ2=1000/

File name: `pythia8NCDIS_10x100_minQ2=1000_beamEffects_xAngle=-0.025_hiDiv_2.1409.eicrecon.tree.edm4eic.root`

- Container version
 - Geometry configuration
 - Physics process
 - Event generator
 - Electron and proton beam energies
 - Minimal Q^2
- HepMC3 files provided and validated by Physics WGs + setups available with single particles for detector studies

Software and Simulation Readiness for TDR

- All hands effort was organized at a **Collaboration meeting in January 2024:**
- 4 groups were given identical **"Charge" with 6 questions** like:
 - "...Are you aware of any MCEGs that are required for detector and physics studies for the TDR and are missing in the charter productions for ePIC?"
 - "...What features are missing in the detector simulations for ePIC that are required for detector and physics studies for the TDR? ..."
 - "...What validation of the detector simulations is required for the TDR other than the consistency check with engineering design via the Detector Geometry Matrix?"
 - "...How would you describe the minimum viable product for the reconstruction for the TDR?"
- Individual responses of each group were **discussed together** after and summarized at a **plenary close-out session**
- An **internal document** was formed based on out outcome of the discussions
 - ⇒ drill into details in follow-up meetings
 - ⇒ initiate/delegate, inquire, track progress
 - ⇒ **a success!**

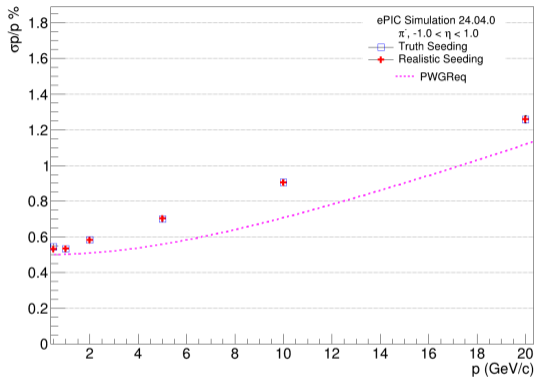
Task forces on Reconstruction Priorities, jointly with ACs

Task Forces are temporary in nature, different from (Standing) ePIC WGs, deal with high-priority tasks

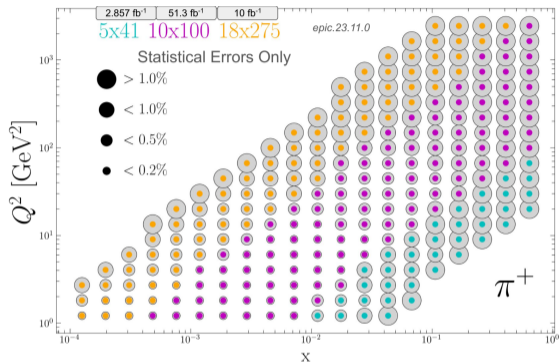
- ✓ **Electron Finder:** Develop an efficient and accurate algorithm for identifying electrons and identifying the scattered electron of the DIS process
- ✓ **Vertexing and PID:** Enhance the vertexing capabilities and particle identification techniques to study heavy flavor physics
 - ✓ Implement PID LUT tables
 - **Secondary Vertexing:** Identify a suitable secondary vertexing tool for ePIC and integrate into the ePIC software stack
- ✓ **Low-Q²:** Integrate the low-Q² tagger into the reconstruction framework for precise measurements of photo production and vector mesons
 - **Particle Flow:** Improve the jet reconstruction using particle flow information
 - ↑ (Priority beyond TDR)
 - **Backgrounds:** Introduce and evaluate realistic backgrounds in the ePIC simulations

ePIC Software supporting studies for (Pre-)TDR

Select studies

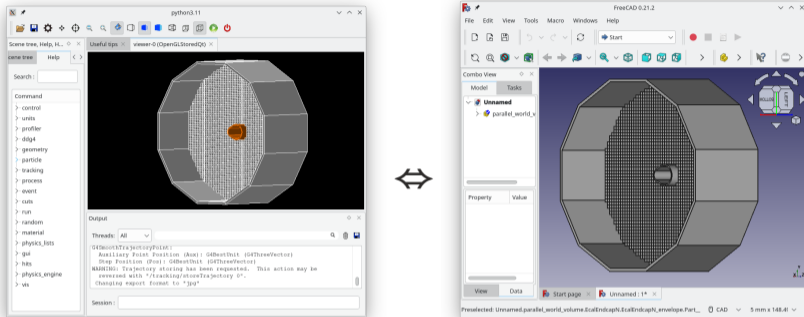


Detector benchmark, candidate for a plot in TDR: **Momentum resolution for the tracker**
plot by Shyam Kumar



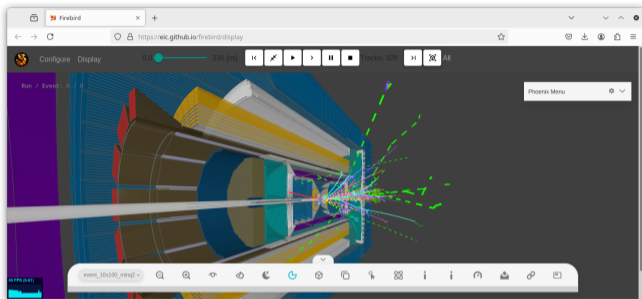
Plot for **physics** section of the TDR:
Expected statistical uncertainty of Unpolarized TMD PDFs
plot by Gregory Matousek

Detector model validation



- Export from CAD to STL for inclusion in ePIC geometry
- Export from ePIC geometry to STEP
 - Tool for geometry development
 - Validation from the project and detector group engineers

ePIC Event Display



- “Firebird” is available live at <https://eic.github.io/firebird/display>
- Based on HSF’s Phoenix display. Lead developer - Dmitry Romanov (JLab)
- Geometry exported from DD4hep with optimizations
- Events exported from EDM4hep to JSON (ROOT to be done)
- Other plans include:
 - ACTS integration, better the UX to support reconstruction development

Growing workforce across the timezones

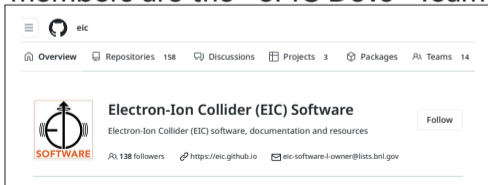
Weekly software news
(slides also posted on [Indico](#))

WG News

Physics and Detector Simulations WG

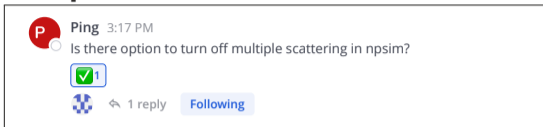
- Start evaluation of simulation vs. engineering designs
 - Creating CAD models converted from ePIC DD4hep geometry.
 - Started with tracking this week (and then PID, calorimetry).
 - Converted models (and views) will be distributed to WGs.
- Progress in CAD to DD4Hep conversion ([see slides](#) from Sam Henry and Tuna Tasali)

<https://github.com/eic> – 200+ members are the “ePIC Devs” Team



The screenshot shows the GitHub repository page for "Electron-Ion Collider (EIC) Software". The repository has 158 repositories, 3 projects, and 14 teams. It has 138 followers and a link to <https://eic.github.io>. The repository description is "Electron-Ion Collider (EIC) software, documentation and resources".

~Helpdesk channel on Mattermost

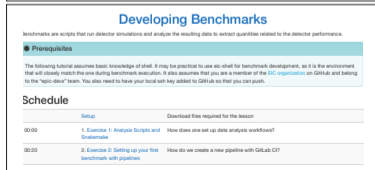


The screenshot shows a Mattermost message in a helpdesk channel. The message is from a user with a red profile picture and asks "Is there option to turn off multiple scattering in npsim?". The message has a green checkmark icon and a "1" next to it, indicating a reply. Below the message, there is a "1 reply" button and a "Following" button.

Online documentation



The screenshot shows the ePIC Landing Page. It features a navigation menu with buttons for "Get started", "ePIC Tutorials", "HEP Software Training Center", and "FAQ". Below the navigation menu, there is a "Welcome to the ePIC Landing Page!" message and a link to the mailing list: eic-project-compse-l@lists.bnl.gov. There is also a link to subscribe: <https://lists.bnl.gov/mailing-lists/eic-project-compse-l>.



The screenshot shows the "Developing Benchmarks" page. It features a "Prerequisites" section with a list of requirements and a "Schedule" section with a table of lessons.

Setup	Download files required for the lesson
00:00	1. Exercise 1: Analyze Scripts and Snapshots How does one set up data analysis workflows?
00:20	2. Exercise 2: Setting up your first benchmark with pipelines How do we create a new pipeline with GitLab CI?

Building our ePIC Software & Computing community

Regular in-person meetings



Organizing internal efforts on all fronts: Development, Simulation, Streaming/DAQ, Validation and User Learning.



Strengthen collaboration with HEP, specifically ACTS, CERN EP-SFT, HSF, Key4HEP, Rucio.

Summary and Outlook

- ePIC Software effort embraces open development model with aim at sustainability
- Building on top of NP-HEP community's past experience, we are working together with it on improving common set of state-of-art tools
- Today, ePIC Software is well prepared to deliver crucial results needed for finalizing detector design and validating its fitness for the purposes of the EIC science program

Full dependencies list

abseil-cpp acts acts-dd4hep actsvg afterburner aida algorithms assimp at-spi2-core berkeley-db boost bzip2 cairo catch2 cernlib clhep cli11 cmake cnpq compositeproto cppcoro cppgsl cppzmq cpuinfo curl davix dawn dawncut dbus dd4hep dlpack double-conversion dpmjet east edm4eic edm4hep eic-smear eicrecon eigen elfutils emacs epic expat fastjet fcgi fftw findutils fixesproto fjcontrib flex fmt font-util fontconfig fp16 freetype fridibi ftgl fxdiv g4abla g4emlow g4ensdfstate g4incl g4ndl g4particlexs g4photonevaporation g4pii g4radioactivedecay g4realsurface g4saiddata gaudi gcc-runtime gdb gdbm geant4 geant4-data gettext git git-lfs gl2ps glew glib glibc gloo glx gmake gmp gnutils gobject-introspection googletest graphviz gsl harfbuzz hepmc hep3 heppdt hwloc icu4c imagemagick inputproto intel-oneapi-tbb intel-tbb irt isa-l iwyu jana2 jq json-c json-glib juggler k4actstracking k4fwcore kbproto krb5 lcov lhpdf libbsd libdrm libedit libevent libffi libgcrypt libgpg-error libice libiconv libidn2 libjpeg-turbo libmd libnsl libpciaccess libpng libpthread-stubs libsm libsodium libtiff libtirpc libtool libunistring libx11 libxau libxaw libxcb libxcomposite libxcrypt libxdmcp libxext libxfixes libxft libxi libxkbcommon libxml2 libxmu libxpm libxrender libxscrnsaver libxslt libxt libxtst libyaml libzmq llvm lz4 madx mesa-glu motif nano ncurses nettle nhttp2 nlohmann-json nopayloadclient npsim numactl oniguruma onnx openblas opencascade openssl openmpi openssl osg-ca-certs pango pcre pcre2 perl perl-b-hooks-endofscope perl-capture-tiny perl-class-data-inheritable perl-class-inspector perl-class-singleton perl-datetime perl-datetime-locale perl-datetime-timezone perl-devel-cover perl-devel-stacktrace perl-digest-md5 perl-dist-checkconflicts perl-eval-closure perl-exception-class perl-exporter-tiny perl-file-sharedir perl-file-sharedir-install perl-file-spec perl-json perl-memory-process perl-module-implementation perl-module-runtime perl-mro-compatible perl-namespace-autoclean perl-namespace-clean perl-package-stash perl-params-validationcompiler perl-role-tiny perl-scalar-list-utils perl-specio perl-sub-exporter-progressive perl-sub-identify perl-sub-quote perl-test-fatal perl-test-requires perl-time-hires perl-try-tiny phonebook-cli pigz pixman pkgconf pmix podio prmon protobuf psimd pthreadpool py-anyio py-appdirs py-argon2-cffi py-argon2-cffi-bindings py-asteval py-asttokens py-astunparse py-attrs py-awkward py-awkward-cpp py-babel py-backcall py-bcrypt py-beautifulsoup4 py-bleach py-bokeh py-boost-histogram py-boto3 py-botocore py-bottleneck py-cherberus py-certifi py-cffi py-cfgv py-charset-normalizer py-click py-cloudpickle py-coloredlogs py-comm py-configargparse py-connectionpool py-contourpy py-cryptography py-cycler py-dask py-datrie py-debugpy py-decorator py-deepdiff py-defusedxml py-deprecated py-distlib py-distributed py-docutils py-dpath py-entrypoints py-epic-capybara py-executing py-fastjsonschema py-filelock py-flatbuffers py-fonttools py-fsspec py-future py-gevent py-gitdb py-gitpython py-graphviz py-greenlet py-gssapi py-hepunits py-hist py-histoprint py-htgettoken py-humanfriendly py-identify py-idna py-importlib-metadata py-ipykernel py-ipython py-ipython-genutils py-jedi py-jinja2 py-jinja2-cli py-jmespath py-json5 py-jsonschema py-jupyter-client py-jupyter-console py-jupyter-core py-jupyter-server py-jupyterlab py-jupyterlab-pygments py-jupyterlab-server py-kiwisolver py-lmfit py-locket py-xml py-markupsafe py-matplotlib py-matplotlib-inline py-mistune py-mplhep py-mplhep-data py-mpmath py-msgpack py-nbclassic py-nbclient py-nbconvert py-nbformat py-nest-asyncio py-networkx py-nodeenv py-notebook py-notebook-shim py-numexpr py-numpy py-onnx py-onnxruntime py-ordered-set py-packaging py-pandas py-pandocfilters py-paramiko py-parso py-partd py-particle py-pexpect py-pickleshare py-pillow py-pip py-pip py-plac py-platformdirs py-pre-commit py-prometheus-client py-prompt-toolkit py-protobuf py-psutil py-ptyprocess py-pulp py-pure-eval py-pybind11 py-pycairo py-pyparser py-pygithub py-pygments py-pyjwt py-pyopenssl py-pyparsing py-pyrsistent py-python-dateutil py-pytz py-pyyaml py-pyzmq py-requests py-reretry py-s3transfer py-scipy py-seaborn py-send2trash py-setuptools py-six py-smart-open py-smmmap py-sniffio py-sortedcontainers py-soupsieve py-stack-data py-stopit py-sympy py-tabulate py-tblib py-terminado py-throttler py-tinycss2 py-toml py-tomli py-toolz py-toposort py-torch py-tornado

Full dependencies list

py-tqdm py-traitlets py-typing-extensions py-uhi py-uncertainties **py-uproot** py-urllib3 py-vector py-virtualenv py-wcwidth py-webencodings
py-websocket-client py-wheel py-wrapt py-wurlitzer py-xyzservices py-yapf py-yte py-zict py-zipp py-zope-event py-zope-interface pythia8 python
python-venv qhull qt-base range-v3 rapidjson re2 readline recordproto renderproto rngstreams **root** rpcsvc-proto scitokens-cpp scrnsaverproto simsipm sleep
snakemake spdlog sqlite stow tar tcl tk unuran util-linux-uuid util-macros valgrind vc vdt xbitmaps xcb-proto xcb-util xcb-util-cursor xcb-util-image
xcb-util-keysyms xcb-util-renderutil xcb-util-wm xerces-c xextproto xeyes xkbdata xproto xrootd xtrans xxhash xz yaml-cpp zlib zlib-ng zstd

Marked are dependencies with contributions from ePIC collaboration or developed by it