ESCalate Documentation

Release 0.00.01

Yulia Furletova, Dmitry Romanov

Mar 09, 2021

GETTING STARTED:

CHAPTER

ONE

CONTAINERS

We use Docker as the default containerization technology. Singularity is also can be easily used utilizing docker images.

The primary goal of the images is to provide an easy means for scientists to start running EIC software. There 4 main images:

- electronioncollider/eic-science
- electronioncollider/eic-mceg
- electronioncollider/escalate

1.1 electronioncollider/escalate v 1.2.0

(Changed packet versions and new packets are bold)

Core tools		HENP		MCEG		EIC	
Packet	**Version**	**Packet**	**Version**	**Packet**	**Version**	**Packet**	**Version**
gcc	**9.3.0**	hepmc	2.6.9	LHAPDF6	6.2.3	ejpm	**0.3.40**
CMake	3.17.0	hepmc3	3.2.1	pythia8	8.244	eic-smear	**1.1.2**
python	**3.7.5**	vgm	4.5	DIRE	2.004	jana	**2.0.4**
eigen3	3.3.7	genfit	**2.0.0**	Cernlib	2006-12-20	ejana	**1.3.0**
clhep	2.3.2.2	acts	**1.2.1**	lhapdf5	5.9.1-6	g4e	**1.4.0**
ROOT	**6-22-06**	delphes	3.4.2	PYTHIA6	RAD-CORR	smear	0.1.6
Geant4	**10.6.2**	fastjet	3.3.3				

[#] - See CHANGELOG.rst for other versions

EIC software quick start tutorial

1.2 Installing Docker

The Docker virtualization software is available for Linux, macOS, and Windows. Please follow the instructions to install, configure, and test Docker on your system:

- **Linux**: Docker Engine (free in the Community Edition)
- **macOS**: Docker Desktop (free)
- **Windows**: Docker Desktop (free)

1.3 Obtaining the EIC Software image

The EIC Software images are deployed using the electronioncollider swarm on Docker Hub. The latest versions can be obtained via:

```
docker pull electronioncollider/escalate:v1.1.0
```

This requires Docker to be running on the local system. Please see the *Troubleshoot* section if you receive the error message of `` `no space left on device` ``.

1.4 Running the EIC Software image

The EIC Software images provide an interactive environment which can be started via:

```
docker run -it --rm -p8888:8888 electronioncollider/escalate:1.2.0
```

and can be accessed via the host system's native web browser.

```
http://127.0.0.1:8888/
```

This requires Docker to be running on the local system. The `` `--rm` `` flag is used to automatically clean up the container and remove the file system (and all modified and created files with it) when the container exits. The flag is included for the sake of tutorials but not needed when working with the JupyterLab environment where retaining all data in the container (including all changes and modifications) might be preferred. By default (without `` `--rm` `` flag), a container's file system persists even after the container exits.

You can bind any directory on your system to docker image by using **-v** flag:

```
-v <your/directory>:<docker/directory>
```

Convenient place inside docker image is

```
/home/eicuser/workspace/share
```

So the full command is:

```
docker run -it --rm -p8888:8888  -v <your/directory>:/home/eicuser/workspace/share␣
↪electronioncollider/escalate:1.2.0
```

1.5 Troubleshoot

If docker gives an error like this: > Error starting userland proxy: listen tcp 0.0.0.0:8888: bind: address already in use.

It usually means, that the port 8888 is used by another application. To fix that try to change *-p 8888:8888* flag to *-p <something>:8888* e.g. *-p 9999:8888*. Put the same port in your browser:

```
127.0.0.1:9999/lab
```

Occasionally, the error message of `no space left on device` has been reported when pulling large Docker images. In most cases, this can be prevented by removing all unused containers, images and more via `docker system prune -a`. On macOS, it might be also required to increase the `disk image size`. You find the option in the Docker Desktop application when selecting `Preferences` and `Resources`.

1.6 X11 - Working with GUI

There are several ways of dealing with native GUI applications for escalate and escalate-gui images. E.g. showing standard root browser or Geant4 event viewer.

1. SSH -X
2. X11 directly

What is the best option:

1.6.1 1. SSH -X

eicuser password is eicuser

```
docker run --rm -it -p127.0.0.1:2222:22 electronioncollider/escalate:latest runssh
```

connect with SSH:

```
ssh -X eicuser@127.0.0.1 -p 2222
```

1.6.2 2. X11

The most convenient is using X11 directly. It require x11 client apps on Macs and Windows and may have some issues with user id's and permissions on Posix (max & linux). It might sound complex, but actiually it is simple and works most of the times. Still we don't use this way for the tutorials, but it is available in the documentation.

Requirements: X11 cliens (windows and mac), additional docker flags (see of each OS)

You can use X11 natively (as natively as possible) with this docker image in your system:

Linux

To use graphics, make sure you are in an X11 session and run the following command:

```
docker run -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix --rm -it --user $(id -
↪u) -p8888:8888 electronioncollider/escalate
```

There might be issues with user id on systems like JLab farms.

Windows

To enable graphics, you must have VcXsrv installed. Make sure VcXsrv is whitelisted in the Windows firewall when prompted.

Start VcXsrv with ‘allow from any origin’ flag

```
docker run --rm -it -p 8888:8888 -e LIBGL_ALWAIS_INDIRECT=1 -e DISPLAY=10.0.75.1:0 ␣
↪electronioncollider/escalate bash
```

OSX

To use graphics on OSX, make sure XQuarz is installed. After installing, open XQuartz, and go to XQuartz, Preferences, select the Security tab, and tick the box “Allow connections from network clients”. Then exit XQuarz.

Afterwards, open a terminal and run the following commands:

```
ip=$(ifconfig en0 | grep inet | awk '$1=="inet" {print $2}')

echo $ip    # To make sure it was successfull
            # If nothing is displayed, replace en0 with en1 and so on

xhost + $ip  # start XQuartz and whitelist your local IP address
```

This will start XQuartz and whitelist your local IP address.

Finally, you can start up docker with the following command:

```
docker run --rm -it -v /tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$ip:0 -p8888:8888␣
↪electronioncollider/escalate
```

Credits:

The EIC Container project is coordinated by David Lawrence and Dmitry Romanov.

CHAPTER

TWO

LOCAL ISNTALLATION

There are several ways how one can install packages locally:

1. Spack package manager
2. EJPM package manager
3. Combined Conda+EJPM isntallation
4. Manual build

What packet manager to use? **The recommendation is**: to use EJPM for personal laptops with Linux OS for personal use and development. Use Spack on farms or if you are familiar with Spack or use Spack for other projects.

EJPM was designed with user laptops/workstations and devlopment in mind. The intention of EJPM is to be a user friendly CLI alternative to "build scritps" that compiles chains of packages with right flags. Spack is a full featured multi-platform package manager designed for clusters and super computers. It is possible to install software on laptops with Spack but at this point it is more prone to compilation errors.

There is also a hybrid variant of using Conda+EJPM where Conda is used to download binaries of compilers, big packages like CERN ROOT and Geant4 and then EJPM is used to build EIC applications on top of it. This type of installation also guarantee a consistency with compillers, binary libraries, etc.

Here is the short comparison table between Spack and EJPM:

spack-ejpm-comparison.

orphan

2.1 EJPM

Detailed EJPM documentation

2.1.1 Installation

ejpm is EIC software centric package/build manager. It is the designed to be the default tool to build G4E on users machine, as it helps with:

- building dependent packages (with "right" compilation flags)
- setup environment variables to run everything
- what packages to install by system packet manager
- rebuild/update/remove/clean existing packages

Still, ejpm is not a requirement.

First, install ejpm itself:

```
pip install --user ejpm
```

(If you have certificate problems (JLab issue), don't have pip, or have other problems, here is the detailed documentation)

Install g4e and possible other EIC packets (ejana, eic-smear, etc.)

```
# 1. System prerequesties
ejpm req centos g4e      # get list of required OS packets. Use `ubuntu` on debian
sudo yum install ...     # install watever 'ejpm req' tells you

# 2. Where to install
ejpm --top-dir=<where-to>    # Directory where packets will be installed

# 3. Install
ejpm install g4e             # install 'Geant 4 EIC' and dependencies (like vgm, hepmc)

# 4.  Source environment
source ~/.local/share/ejpm/env.sh  # Use *.csh file for tcsh
```

2.1.2 User packages

If you have ROOT and Geant4 and don't want EJPM to build them, here is how to use your packages with ejpm:

```
# Before running 'ejpm install g4e'
ejpm set root `$ROOTSYS`     # Path to ROOT installation
ejpm set geant <path>        # Path to Geant4 installation
```

Hint (!). Run ejpm to overview all installed packets, environment and status by 'ejpm' command

Here is the sample output:

```
> ejpm

EJPM v0.01.19
top dir :
    /eic
state db :
    ~/.local/share/ejpm/db.json  (users are encouraged to inspect/edit it)
env files :
    ~/.local/share/ejpm/env.sh
    ~/.local/share/ejpm/env.csh

INSTALLED PACKETS: (*-active):
 vgm:
    * /eic/vgm/vgm-v4-5 (owned)
 root:
    * /eic/root/root-v6-16-00 (owned)
 geant:
      /eic/geant/geant-v10.5.0 (owned)
    * /eic/geant4-10.6-betta
 hepmc:
    * /eic/hepmc/hepmc-HEPMC_02_06_09 (owned)
 g4e:
    * /eic/g4e/g4e-dev (owned)
```

2.1.3 Switch software versions

```
# ejpm config <package> branch <version>
ejpm config g4e branch v1.3.8
# or
ejpm config g4e branch master
# now install a new version
ejpm install -f g4e
```

To check packet configuration:

```
ejpm config g4e
```

To check or change global configuration (for all packages)

```
ejpm config global      # prints version
ejpm config global cxx_version=17
```

Warning: if you install a new version of package EJPM doesn't rebuild/reinstall downstream packages.

This means that if you install additional version of g4e you don't have to do anything, as there is no packages depending on g4e. G4E is on top. But if you install a different version of ROOT or ACTS you have to rebuild packages, that depend on it. That is EJPM weakness.

orphan

2.2 Spack

2.2.1 Install spack

Spack is a package management tool designed to support multiple versions and configurations of software on a wide variety of platforms and environments. Spack allows to automatically build target packages with all needed dependencies. Sapck documentation

The installation consist of 3 steps then: 1. Install spack itself 2. Install EIC repository (with EIC packages) 3. Run spack command to install g4e (or other packages)

To install spack and EIC repository:

```
git clone https://github.com/spack/spack.git

#Source environment

# For bash/zsh users
$ . spack/share/spack/setup-env.sh

# For tcsh/csh users
$ source spack/share/spack/setup-env.csh
```

You should be able now to use spack:

```
spack info root

(!) By default, all packages will be downloaded, built and installed
in this spack directory

`More documentation on spack
installation <https://spack.readthedocs.io/en/latest/getting_started.html
↪#installation>`__
```

Clone and add eic-spack repository:

```
# Adding the EIC Spack Repository
git clone https://github.com/eic/eic-spack.git

# Add this repository to your Spack configuration
spack repo add eic-spack
```

2.2.2 Example install and use g4e with spack

To install g4e with spack (this will install the latest stable version)

```
spack install g4e
```

To install concrete version of spack:

```
spack info g4e            # to see the available versions
spack install g4e@1.3.7   #
```

To see what is going to be installed/built

```
spack speck -I g4e
```

To use G4E with spack:

```
spack load g4e

# or with exact version
spack load g4e@1.3.7
```

More documentation of spack usage

CHAPTER

THREE

FARMS

3.1 Spack CVMFS central installation

ESCalate framework is installed on CVMFS and can be used directly on farms with spack

1. Source spack environemnt

Tcsh/csh users (default for ifarm)

```
source /cvmfs/eic.opensciencegrid.org/packages/setup-env.csh
```

bash/zsh

```
source /cvmfs/eic.opensciencegrid.org/packages/setup-env.sh
```

2. Load escalate module

```
spack load escalate@1.1.0
```

It should be ready to work

3. Test run

```
which g4e
ejana
```

3.2 JLab IFarm

3.2.1 Singularity installation

There is no docker on IFarm, but one can use Singularity

```
[user@ifarm1801 ~]$ module load singularity
[user@ifarm1801 ~]$ singularity shell --cleanenv /group/eic/escalate/singularity-
↪latest
Singularity> source /container/app/userenv.sh
```

And it is ready to go

3.2.2 Examples

1. Simple way to smear a file

```
smear /path/to/file.txt
```

Here are 2 more advanced examples files for fast and full simulation. Please run them in your work directory.

Fast simulation (eic-smear):

```
from pyjano.jana import Jana

Jana().plugin('beagle_reader')\
      .plugin('open_charm')\
      .plugin('eic_smear', detector='jleic')\
      .plugin('jana', nevents=20000, output='hepmc_sm.root')\
      .source('/group/eic/mc/BEAGLE/eD_5x50_Q2_1_10_y_0.01_0.95_tau_7_noquench_
↪kt=ptfrag=0.32_Shd1_ShdFac=1.32_Jpsidifflept_test40k_fixpf_crang.txt')\
      .run()
```

Full simulation:

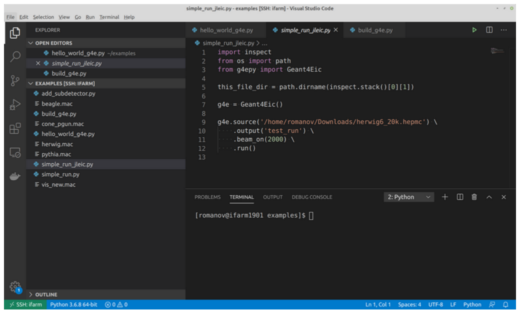
```
from g4epy import Geant4Eic
g4e = Geant4Eic()\
      .source('/group/eic/mc/BEAGLE/eD_5x50_Q2_1_10_y_0.01_0.95_tau_7_noquench_
↪kt=ptfrag=0.32_Shd1_ShdFac=1.32_Jpsidifflept_test40k_fixpf_crang.txt')\
      .output('hello')\
      .beam_on(200)\
      .run()
```

Look at the docker or tutorials repo for more examples

3.2.3 Convenient work on IFarm from home

Since many of us experience COVID-19 related teleworking from home, here is a suggestion of a very convenient workflow - vscode + remote ssh.

It allows to work with remote ssh scripts and code (and have unlimited consoles) like they are on your home computer. Here is how it looks like:

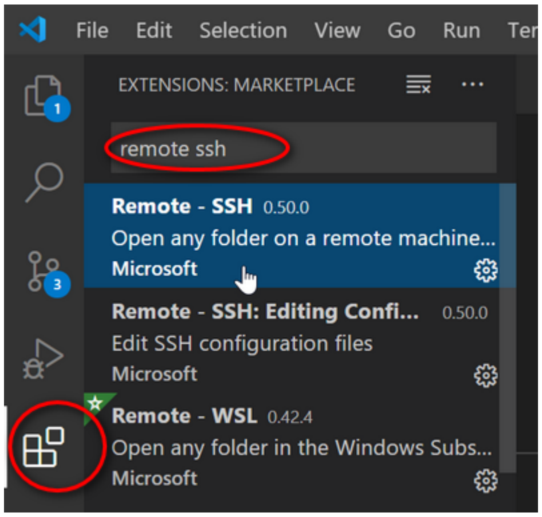


```
import inspect
from os import path
from g4epy import Geant4Eic

this_file_dir = path.dirname(inspect.stack()[0][1])

g4e = Geant4Eic()

g4e.source('/home/romanov/Downloads/herwig6_20k.hepmc') \
    .output('test_run') \
    .beam_on(2000) \
    .run()
```

```
[romanov@ifarm1901 examples]$
```

How to set it up:

- install vscode https://code.visualstudio.com/
- **open it and install remote ssh plugin. Here is where it is:**


File
Edit
Selection
View
Go
Run
Ter
EXTENSIONS: MARKETPLACE
remote ssh
Remote - SSH 0.50.0
Open any folder on a remote machine...
Microsoft
Remote - SSH: Editing Confi... 0.50.0
Edit SSH configuration files
Microsoft
Remote - WSL 0.42.4
Open any folder in the Windows Subs...
Microsoft

- then here is a short and good tutorial, how to use it: https://code.visualstudio.com/remote-tutorials/ssh/getting-started
- edit config file (see below)
- after you connect, open folder (or whatever), you'll see, you are on ifarm

The trick with ifarm is that you first have to do proxy jump through `login.jlab.org`. To solve this automaticaly edit ssh settings which usually are located in `~/.ssh/config` folder

(To do this vscode press F1 and type in `remote ssh config`)

```
Host jlogin
    User romanov
    HostName login.jlab.org
    Port 22

Host ifarm
    User romanov
    HostName ifarm
    Port 22
    ProxyJump jlogin
```

If on windows you have it is because windows installs old ssh. Fix is to add one more line to the config

```
ProxyCommand ssh -W %h:%p ifarm
```

Now you can also connect to ifarm like `ssh ifarm` and use it in vscode.

> CONCERNED with MISCROSOFT word? Use vscodium. `vscodium` relates to `vscode` the same way as `Chromium` browser relates to `Google Chrome`

CHAPTER FOUR

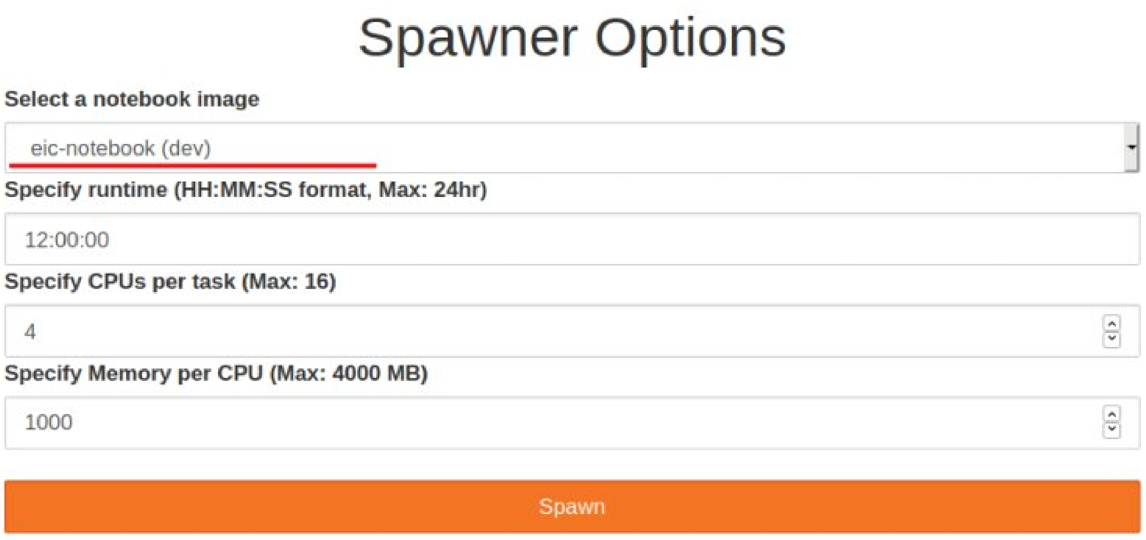
JUPYTER-HUB

```
http://jupyterhub.jlab.org
```

Escalate framework image added to Jefferson Lab Jupyter Hub! It is still in beta stage, several features are not yet working (and we are working on them). Someting works differently compared to when you run the image in docker. This page describe this.

Go to jupyterhub.jlab.org (follow authentication instructions if you are using it for the first time)

In the **Spawner options -> Select a notebook image**, set **eic-notebook (dev)** there



Spawner Options

Select a notebook image: eic-notebook (dev)

Specify runtime (HH:MM:SS format, Max: 24hr): 12:00:00

Specify CPUs per task (Max: 16): 4

Specify Memory per CPU (Max: 4000 MB): 1000

Fig. 1: jlab_jupyterhub_spawner

You should end up in your jefferson lab home directory.

4.1 Differences with running in docker

When you run in docker you start as *eicuser* with user-ID=1000. On JupyterHUB you run in your JLab home directory with your CUE user (and your CUE user-ID). This implies that:

1. Your JLab home dir .bashrc is being called instead of one in docker. If you set custom python version or compiler in your .bashrc it will interfere with what is used in docker (will not work most of the time).
2. Since you start in your CUE home dir, you don't have the examples and tutorials. Just clone them to your JLab home dir: `bash git lfs clone https://gitlab.com/eic/escalate/workspace.git # 'lfs' is to pull data files!`
3. All docker contents are readonly. One can't run `sudo` to elevate privilegies and change something in the container. Which means that you can't change eJana or G4E inside the docker, but you can install them in your home directory and setup ejpm to use them.

4.2 What is not working

1. Inspecting root files by ckicking on them. Solution - use uproot to explore the files.