A Beginner's Guide To CMS-OpenData-Pipeline

Experimental High Energy Physics Group
Indian Institute of Science Education and Research
Pune, India

A Note To The Reader

This is a guide to install and setup the CMS-OpenData-pipeline and also some of the requirements of the pipeline. This guide will help you navigate through the trenches of CMS-OpenData and also will help you in understanding some of the nitty-gritties of CMS-OpenData. Since, all of the CMS-OpenData is diffcult to cover, this guide only discusses the Run 1 open data. This guide shall be of great help if you are not well very well equipped with using Git, ROOT, `CMSSW` or even the terminal. Consider this as your friend trying to instruct you to drive a car you've never driven before. It is going to take some time, but rest assured, your friend's got your back. Some of the things in this guide can look very complicated(or "tech savvy"), but don't panic, you'll be using these terms in no time once you've mastered them (which you will!). The last section of the guide discusses all the common errors, whether it be installation or compilation of code (believe me, we all make tons of errors). There are illustrative examples of error messages for your reference. If you encounter any new errors which may not have been discussed, feel free to email at **raj.handique519@deccansociety.org** or visit the GitHub page and raise an issue.

And if you're sitting in the Experimental Particle Physics Laboratory at IISER-Pune, sit back, relax, wear a hoodie and ask around if you get stuck. *The cold lab will eventually lead to warm friendships.*

Regards,

Raj Handique
Developer - CMS-OpenData-Pipeline

Contents

List of Figures

Requirements for the pipeline setup

The CMS-OpenData-Pipeline [1] has certain requirements:

1. Docker should be up and running in the local machine. Alternatively, CMS-OpenData-Virtual Machine works as well. (It is recommended not to use the VM as it is painfully slow)
2. CMS Offline Software Framework - `CMSSW` should be set up in the docker container.
3. MiniConda or AnaConda should be installed in your local system.
4. ROOT(6.20 or higher) must be installed in the local machine.
5. Linux-based systems are preferred, Windows systems are not recommended as docker has compatibility issues with Windows.

Refer to the GitHub page for further information on the requirements for the pipeline.

ROOT Installation Cheatsheet

The installation of ROOT [2] can be very tricky and can crash, here is a cheat sheet for that:

1. Installing the ROOT dependencies is a must before building ROOT from source. For, Debian-based distributions a one-liner command for all the dependencies is as follows:

   ```
   sudo apt-get install binutils cmake dpkg-dev g++ gcc libssl-dev git libx11-dev
   libxext-dev libxft-dev libxpm-dev python3 libtbb-dev
   ```

   Additiional optional packages are also listed below:

   ```
   sudo apt-get install gfortran libpcre3-dev \
   libglu1-mesa-dev libglew-dev libftgl-dev \
   libfftw3-dev libcfitsio-dev libgraphviz-dev \
   libavahi-compat-libdnssd-dev libldap2-dev \
   python3-dev python3-numpy libxml2-dev libkrb5-dev \
   libgsl-dev qtwebengine5-dev nlohmann-json3-dev libmysqlclient-dev
   ```

   For dependencies of Fedora, Scientific Linux, CentOS, OpenSuse, MacOS, Arch and Windows, refer to the official dependencies of the respective operating systems.

2. Building ROOT from source. After the dependencies are setup, root can be compiled using the following commands:

   ```
   git clone --branch latest-stable --depth=1 \
   https://github.com/root-project/root.git root_src
   mkdir root_build root_install  cd root_build
   cmake -DCMAKE_INSTALL_PREFIX=../root_install ../root_src
   cmake --build .  -- install -j4
   source ../root_install/bin/thisroot.sh
   ```

   Note: The ROOT installation requires all the dependencies to be met properly, a stable internet connection and a lot of patience. Also, considering the amount of time it takes, the installation is preferred to be done before one plans to go for lunch or decides to take a nap! For detailed information on the installation, please refer to the ROOT website.

Chapter 1

Docker Installation & setting up the local machine

Docker [3] is a software tool that provides a virtual environment for a specific CMSSW environment to run on a local machine. The Docker container containing the `CMSSW_5_3_32` [4] needs to installed.

1.1 Installing Docker

This guide will discuss how to install docker, for people who find reading long websites boring, in a shorter way. This guide is based on `Ubuntu 23.10`. The steps to install Docker are as follows:

1. Set up docker's `apt` repository and run these following commands in the terminal:

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker .com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2. Install the docker packages

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

That's it. Copying and pasting these commands into the terminal should do the job. The longer guide to installing Docker is given on the official Docker installation guide (only if one feel adventurous).

1.2 Installing Docker-Desktop (Optional)

Installation of Docker-Dektop requires one to download the latest package. This guide will discuss the installation on a Debian-based system (`Ubuntu 23.10`). The steps are as follows:

1. Download the latest .DEB package from the docker website.
2. Install the package using the following command:

   ```
   sudo apt-get install ./docker-desktop-<version>-<arch>.deb
   ```

3. To enable and start Docker-Desktop, use the following command:

   ```
   systemctl --user start docker-desktop
   ```

That's it, Docker-Desktop is now set up. Now, one can go ahead to setup the docker container for CMS Offline Software Framework. The longer guide to the installation of Docker-Desktop can be found here.

1.3 Setting up the Docker container with CMSSW_5_3_32

The docker container with `CMSSW_5_3_32` is available on Dockerhub or on GitLab. For Run 1 Datasets, `CMSSW_5_3_32` is preferred. The installation process for the software environment is discussed below:

1. To enable the docker daemon one is preferred to run the following commands:

   ```
   sudo su
   systemctl start docker
   systemctl enable docker
   systemctl restart docker
   ```

2. Fetch the docker container on the local machine by running the following command on the terminal:

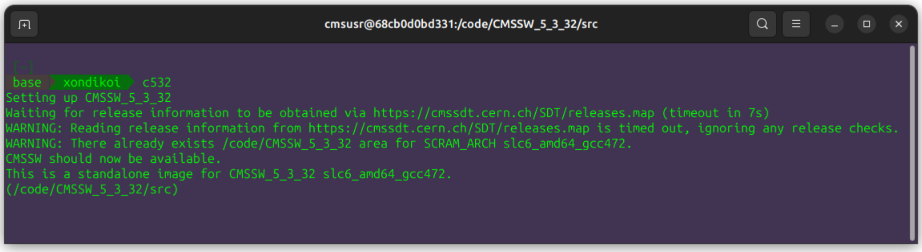
   ```
   docker pull cmsopendata/cmssw_5_3_32-slc6_amd64_gcc472
   ```

   Then to run the docker use the following command:

   ```
   docker run --name CMSSW_5_3_32 -P -p 5902:5902 -p 6081:6081 -it cmsopendata/
   cmssw_5_3_32-slc6_amd64_gcc472 /bin/bash
   ```

   The name given to the container is `CMSSW_5_3_32`. After the fetching is done one shall see these messages on the terminal:

   This will start the docker container in the terminal and one be able to access the CMS offline software environment from here.



```
[~]
 base  xondikoi  c532
Setting up CMSSW_5_3_32
Waiting for release information to be obtained via https://cmssdt.cern.ch/SDT/releases.map (timeout in 7s)
WARNING: Reading release information from https://cmssdt.cern.ch/SDT/releases.map is timed out, ignoring any release checks.
WARNING: There already exists /code/CMSSW_5_3_32 area for SCRAM_ARCH slc6_amd64_gcc472.
CMSSW should now be available.
This is a standalone image for CMSSW_5_3_32 slc6_amd64_gcc472.
(/code/CMSSW_5_3_32/src)
```

Figure 1.1: The figure shows docker container starting with CMSSW_5_3_32

Note: Once one uses the exit command or closes the terminal mistakenly (which we all do a lot of times), the docker container needs to be started again.

3. To start the Docker container again, the following terminal command should be used:

```
docker start -i CMSSW_5_3_32
```

Note: To start the docker container, one is compelled to type this command in their terminal over and over again. So, it is better if one makes a terminal shortcut command or an alias in the .bashrc (or whatever terminal one uses) file. Locate the .bashrc file in the /home/ directory and add the following lines to the file:

```
alias c532="docker start -i CMSSW_5_3_32"
```

Now, one just has to use the command “c532” to start the docker container.

Note: Emacs [5] installation inside the docker container, in case one wants to edit the source code inside the container (which is very handy BTW !), is discussed below. This doesn't mean you can't install any other text editor, but Emacs is much better to use inside the container (also much better than Vim !). Since, the docker container comes with CentOS, the following command shall do the job:

```
sudo yum install emacs
```

To open any file, use the following command:

```
emacs testfile.cc
```

Note: Emacs shortcuts are definitely very different than the regular keyboard shortcuts. So, a guide to all the Emacs shortcuts can be found here!

Now that the docker container is set up and ready to use, one can now start setting up the pipeline in the container.

Chapter 2

CMS-OpenData-Pipeline - Setup & Usage

The CMS-OpenData-Pipeline is developed by the Experimental High Energy Physics group at IISER-Pune. The pipeline is capable of making custom Ntuples consisting of high-level physics objects (for e.g: electrons, muons, jets) from Run 1 AODs (Analysis Object Datasets) [6].

2.1 Setting up and the CMS-OpenData-Pipeline in the Docker Container

The steps to set up the CMS-OpenData-Pipeline in the docker container are as follows:

1. The CMS-OpenData-Pipeline sits on the GitHub repository, so one needs to clone the repository onto the docker container.
   The repository should be cloned into the `/code/CMSSW_5_3_32/src/` directory in the docker container. The following command shall clone the repository:

   ```
   git clone https://github.com/phazarik/CMS-OpenData-analysis.git
   ```

2. This will give the following output in the terminal:

   ```
     Cloning into 'CMS-OpenData-analysis'...
     remote: Enumerating objects: 336, done.
     remote: Counting objects: 100\% (199/199), done.
     remote: Compressing objects: 100\% (124/124), done.
     remote: Total 336 (delta 114), reused 124 (delta 73), pack-reused 137
     Receiving objects: 100\% (336/336), 38.07 MiB | 2.46 MiB/s, done.
     Resolving deltas: 100\% (172/172), done.
   ```

   The repository is successfully cloned to the docker container.

2.2 Using the CMS-OpenData-Pipeline

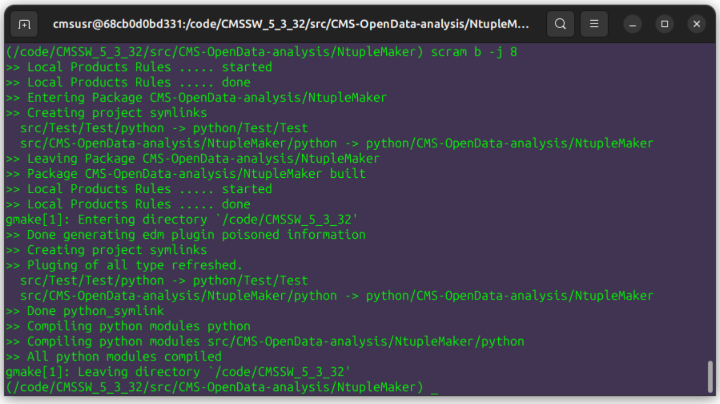
After setting up the pipeline in the docker container, one shall be able to use it. This section of the manual shall guide to make a simple Ntuple and also solve a test problem.

2.2.1 Creating a test custom Ntuple

Let us see, how can we create a simple Ntuple from a sample input file:

1. Navigate to the CMS-OpenData-Pipeline's Ntuple maker directory. The directory path is given as: `/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker/Test/`.
2. In this directory you'll find a python file named testntuple.py, which we will be using for our test problem. There shall also be a C++ source file named testntuplemaker.cc that will be used by the python file in the `../src/` directory. You can access these files and make changes later.
3. Now, let us produce a custom Ntuple that will contain high-level muons. For that we'll have to compile the source code before we can exeute it. To compile the files, one has to run the following command:

```
scram b -j 8
```


cmsusr@68cb0d0bd331:/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleM...

```
(/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker) scram b -j 8
>> Local Products Rules ..... started
>> Local Products Rules ..... done
>> Entering Package CMS-OpenData-analysis/NtupleMaker
>> Creating project symlinks
  src/Test/Test/python -> python/Test/Test
  src/CMS-OpenData-analysis/NtupleMaker/python -> python/CMS-OpenData-analysis/NtupleMaker
>> Leaving Package CMS-OpenData-analysis/NtupleMaker
>> Package CMS-OpenData-analysis/NtupleMaker built
>> Local Products Rules ..... started
>> Local Products Rules ..... done
gmake[1]: Entering directory `/code/CMSSW_5_3_32'
>> Done generating edm plugin poisoned information
>> Creating project symlinks
>> Pluging of all type refreshed.
  src/Test/Test/python -> python/Test/Test
  src/CMS-OpenData-analysis/NtupleMaker/python -> python/CMS-OpenData-analysis/NtupleMaker
>> Done python_symlink
>> Compiling python modules python
>> Compiling python modules src/CMS-OpenData-analysis/NtupleMaker/python
>> All python modules compiled
gmake[1]: Leaving directory `/code/CMSSW_5_3_32'
(/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker) _
```

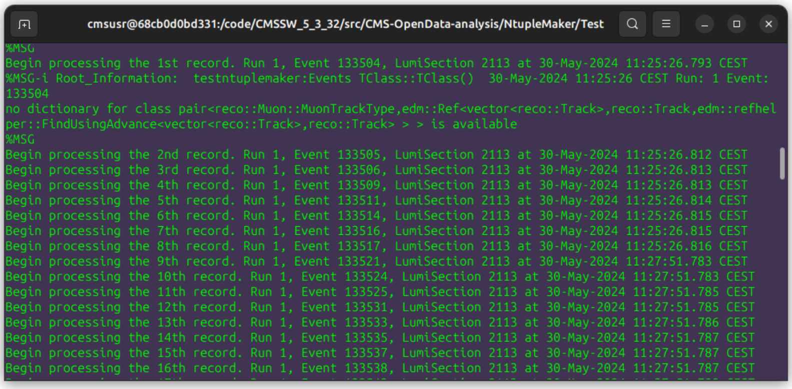
Figure 2.1: The `scram` command compiles the source code and also looks out for errors in the source code.

4. After successful compilation, the custom Ntuple can be made using the following command:

```
cmsRun testntuple.py nEvents=5000 isData=False
```

This will run the python file and produce an custom NTuple named `testoutput.root`. This file can be accessed through TBrowser in ROOT [2].

Note: To run over all events, set `nEvents=-1`.



cmsusr@68cb0d0bd331:/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker/Test

```
%MSG
Begin processing the 1st record. Run 1, Event 133504, LumiSection 2113 at 30-May-2024 11:25:26.793 CEST
%MSG-i Root_Information:  testntuplemaker:Events TClass::TClass()  30-May-2024 11:25:26 CEST Run: 1 Event:
133504
no dictionary for class pair<reco::Muon::MuonTrackType,edm::Ref<vector<reco::Track>,reco::Track,edm::refhel
per::FindUsingAdvance<vector<reco::Track>,reco::Track> > > is available
%MSG
Begin processing the 2nd record. Run 1, Event 133505, LumiSection 2113 at 30-May-2024 11:25:26.812 CEST
Begin processing the 3rd record. Run 1, Event 133506, LumiSection 2113 at 30-May-2024 11:25:26.813 CEST
Begin processing the 4th record. Run 1, Event 133509, LumiSection 2113 at 30-May-2024 11:25:26.813 CEST
Begin processing the 5th record. Run 1, Event 133511, LumiSection 2113 at 30-May-2024 11:25:26.814 CEST
Begin processing the 6th record. Run 1, Event 133514, LumiSection 2113 at 30-May-2024 11:25:26.815 CEST
Begin processing the 7th record. Run 1, Event 133516, LumiSection 2113 at 30-May-2024 11:25:26.815 CEST
Begin processing the 8th record. Run 1, Event 133517, LumiSection 2113 at 30-May-2024 11:25:26.816 CEST
Begin processing the 9th record. Run 1, Event 133521, LumiSection 2113 at 30-May-2024 11:27:51.783 CEST
Begin processing the 10th record. Run 1, Event 133524, LumiSection 2113 at 30-May-2024 11:27:51.783 CEST
Begin processing the 11th record. Run 1, Event 133525, LumiSection 2113 at 30-May-2024 11:27:51.785 CEST
Begin processing the 12th record. Run 1, Event 133531, LumiSection 2113 at 30-May-2024 11:27:51.785 CEST
Begin processing the 13th record. Run 1, Event 133533, LumiSection 2113 at 30-May-2024 11:27:51.786 CEST
Begin processing the 14th record. Run 1, Event 133535, LumiSection 2113 at 30-May-2024 11:27:51.787 CEST
Begin processing the 15th record. Run 1, Event 133537, LumiSection 2113 at 30-May-2024 11:27:51.787 CEST
Begin processing the 16th record. Run 1, Event 133538, LumiSection 2113 at 30-May-2024 11:27:51.787 CEST
```

Figure 2.2: Running the cmsRun command should give out this output in the terminal. The process might take sometime to establish the connection with the CERN server, so don't panic if it gets stuck, just sit back relax and let it brew slowly.

Congratulations, we've now successfully created the first Ntuple containing muon information of 5000 events from a **DYJetsToLL** [7] simulated sample. Now, let's take this custom Ntuple out of the docker container to the local machine. (ROOT doesn't really work in the docker container, so it's better to have in the local machine).

2.2.2 Copying the custom Ntuple to the local machine

To copy the file to the local machine, let us make a directory where one will perform further analysis. In the /home/username/Desktop/ directory, create a new directory using the following command:

```
mkdir cmsopdAna
```

Now navigate to the /home/username/Desktop/cmsopdAna/ directory. For performing copy and paste actions between the docker container and the local machine, the following commands shall be used:

1. To copy a file from the docker container to the local machine, the following command shall be used:

   ```
   docker cp CMSSW:/code/CMSSW/path/file.root /home/localmachine/destination/path/
   ```

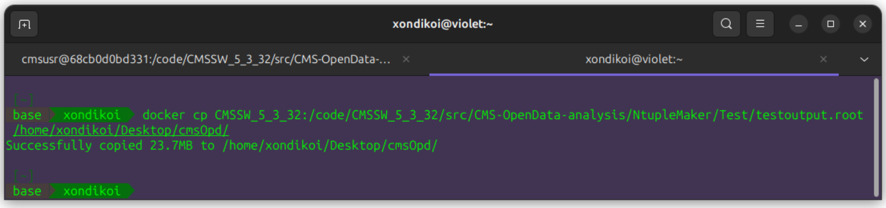
2. To copy a file from the local machine to the docker container, the following command should be used:

   ```
   docker cp /home/localmachine/path/file.root CMSSW:/code/CMSSW/destination/path/
   ```

   This is an example directory, the user's directory will vary accoridingly.

3. In the test problem, the test Ntuple can be copied by using the following command:

```
docker cp CMSSW_5_3_32:/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker/
Test/testoutput.root /home/username/Desktop/cmsopdAna/
```



```
[~]
base  xondikoi  docker cp CMSSW_5_3_32:/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker/Test/testoutput.root
 /home/xondikoi/Desktop/cmsOpd/
Successfully copied 23.7MB to /home/xondikoi/Desktop/cmsOpd/

[~]
base  xondikoi
```

Figure 2.3: Copying the `testouput.root` file to the local machine from the docker container using the `docker cp` command

Here, the first directory `CMSSW_5_3_32:/code/CMSSW_5_3_32/src/CMS-OpenData-analysis/NtupleMaker/Test/testoutput.root` is the directory where the Ntuple is stored in the docker container. The second directory `/home/username/Desktop/cmsopdAna/` is the directory in the local machine where we want to copy our Ntuple. The directory paths can be modified as per the requirement of the user.

2.2.3 Plotting observables - Di-Muon invarient mass

Now that the output file is copied out to the local machine, one can now perform analyses on the output file. In our test problem, we've the following properties - transverse momentum pT, polar angle (ϕ), pseudorapidity (η).

Let us try to plot the dimuon invarient mass of the Drell-Yan sample. A sample code is provided in the following GitHub repository.

Clone the repository to the `/home/username/Desktop/cmsopdAna/`. Now, to get some simple plots, one has to compile and execute these files in ROOT.

```
git clone https://github.com/xondikoi03/CMS-OpenData-AnalysisCodes.git
```

The `/Test-example-dimuon/` branch from the Git repository contains the analysis codes for the test problem named `cmsAna.C`. The Git branch needs to be changed from `main` by doing `git switch Test-example-dimuon`. One can access them with any text editor (preferebly Emacs), and design their own analyses. To run the analyseis codes using ROOT, these are the following steps:

1. Initialize ROOT using the following command in the terminal:

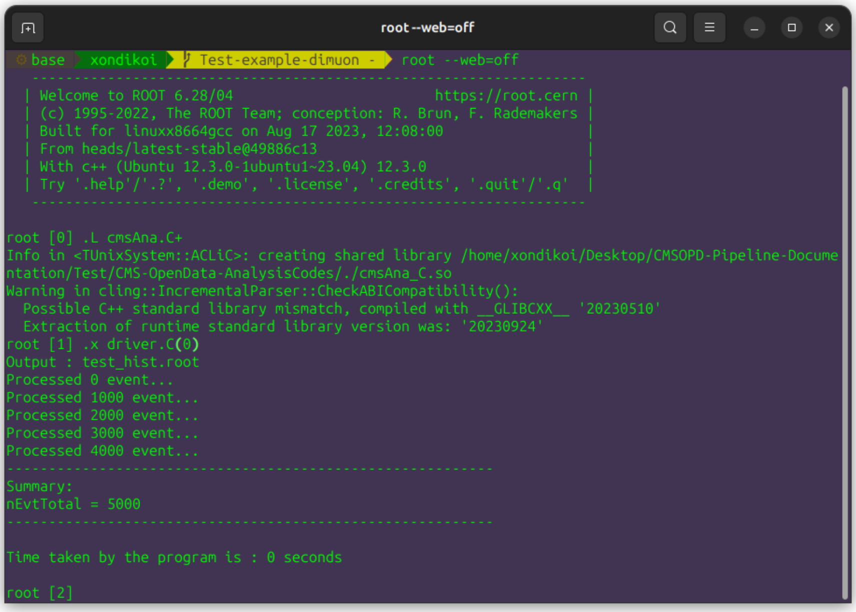
```
root --web=off
```

2. Compile the code using the following command:

```
.L cmsAna.C+
```

3. Execute the driver script to run the analysis using the following command:

```
.x driver.C
```



```
 base  xondikoi  Test-example-dimuon -  root --web=off
   ------------------------------------------------------------------
  | Welcome to ROOT 6.28/04                        https://root.cern |
  | (c) 1995-2022, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Aug 17 2023, 12:08:00                 |
  | From heads/latest-stable@49886c13                                |
  | With c++ (Ubuntu 12.3.0-1ubuntu1~23.04) 12.3.0                   |
  | Try '.help'/'.?', '.demo', '.license', '.credits', '.quit'/'.q'  |
   ------------------------------------------------------------------

root [0] .L cmsAna.C+
Info in <TUnixSystem::ACLiC>: creating shared library /home/xondikoi/Desktop/CMSOPD-Pipeline-Docume
ntation/Test/CMS-OpenData-AnalysisCodes/./cmsAna_C.so
Warning in cling::IncrementalParser::CheckABICompatibility():
  Possible C++ standard library mismatch, compiled with __GLIBCXX__ '20230510'
  Extraction of runtime standard library version was: '20230924'
root [1] .x driver.C(0)
Output : test_hist.root
Processed 0 event...
Processed 1000 event...
Processed 2000 event...
Processed 3000 event...
Processed 4000 event...
----------------------------------------------------------
Summary:
nEvtTotal = 5000
----------------------------------------------------------

Time taken by the program is : 0 seconds

root [2]
```

Figure 2.4: This terminal snippet shows a successful execution of the analysis codes producing histograms from the custom Ntuple.

4. Now one can use the TBrowser to access the output histogram files:

```
TBrowser a;
```

5. Navigate through the filesystem to open the histogram file as shown below:

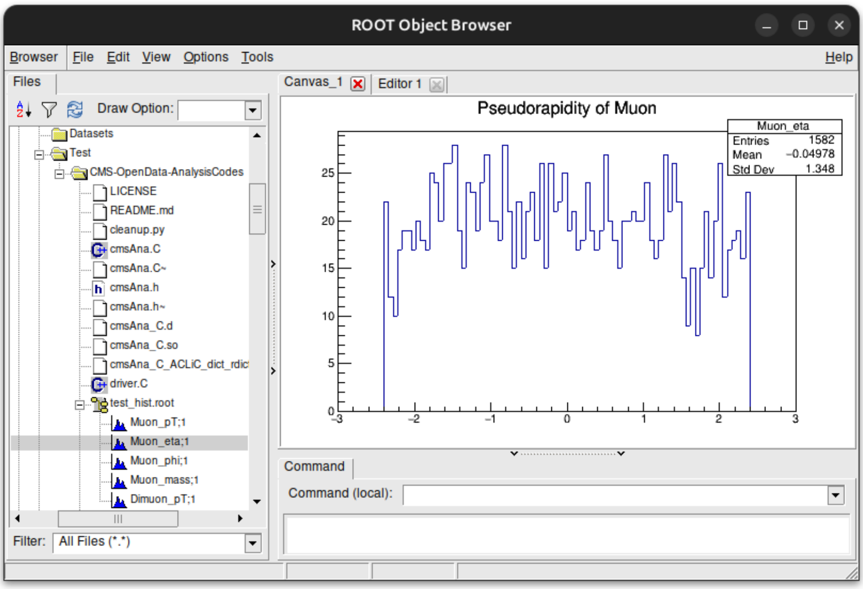


Figure 2.5: TBrowser filesystem showing the test_hist.root file.

6. Navigate to the Dimuon_inv_mass plot of the file, the plot should look as shown below:

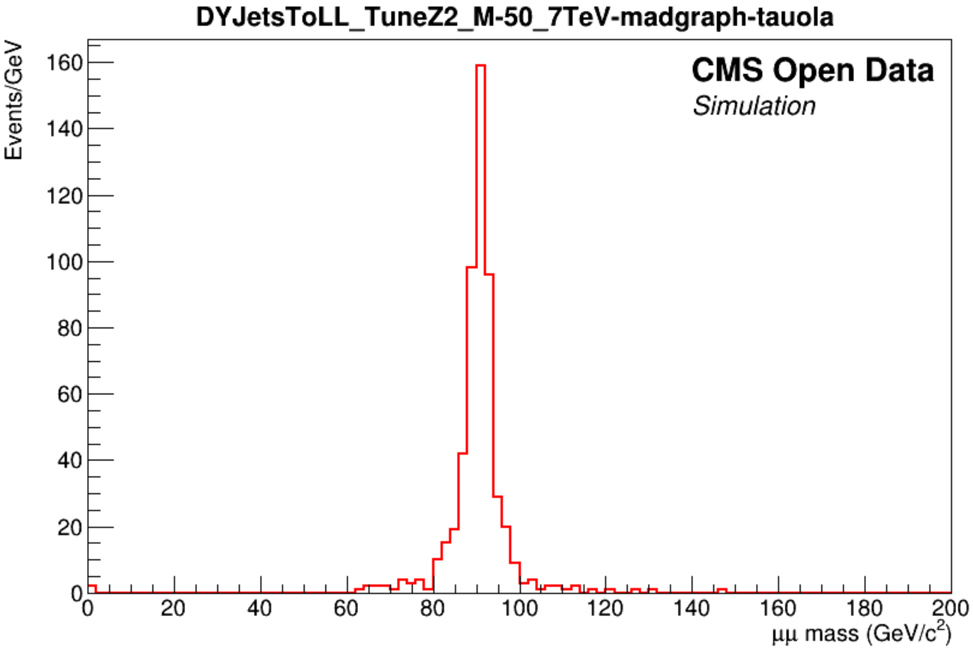


Figure 2.6: The plot shows dimuon invarient mass, which in a Drell-Yan process should peak at around 90 GeV as it is a "$Z- > ll$" process.

That's it, we've now plotted the Dimuon mass using the CMS-OpenData-Pipeline. Fun, isn't it? One can thus do even more complex analyses using the CMS-OpenData-Pipeline,

Chapter 3

Common errors & how to solve them?

Every piece of code that any human have ever written has at some point of time given out a runtime error. That's why compilers with error statements became an industry standard. There were a plenty of issues while building this pipeline, and there can a lot more while setting this up in one's system. This section of the guide discusses some of the few common errors during the installation and setup of the CMS-OpenData-Pipeline.

3.1 Errors in docker container - `CMSSW_5_3_32`

There can be a few errors while setting up the docker container, some of the few errors are:

1. The docker container installation may not be successful, if the user's local machine doesn't support `KVM support`.

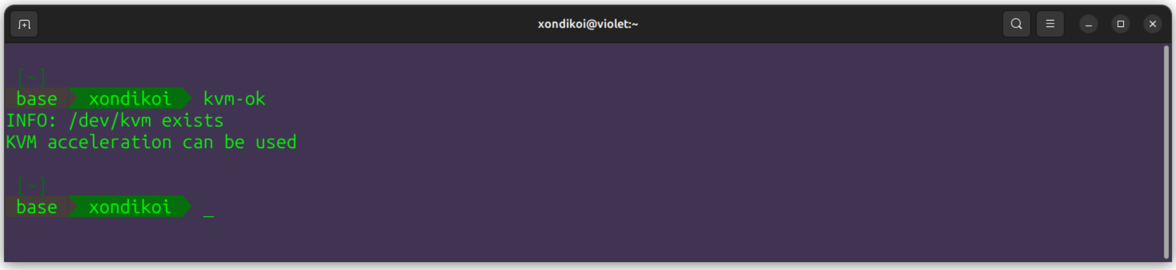
   One can initialise KVM by using the following commands:

   For Intel processors:
   `modprobe kvm_intel`

   For AMD processors:
   `modprobe kvm_amd`

   To check if KVM is running in the system, use the following command:

   `kvm-ok`



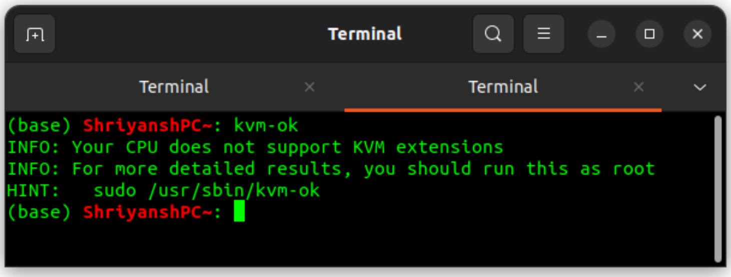
xondikoi@violet:~

```
[~]
 base  xondikoi  kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used

[~]
 base  xondikoi  _
```

Figure 3.1: This message should pop up to indicate KVM is enabled in the system.

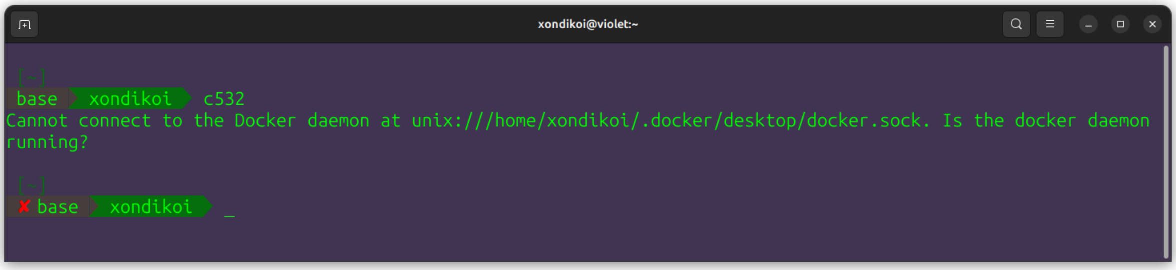
Unfortunately, if the following message shows up, that indicates, the user's machine will not be able to run docker or a virtual machine. That is death sentence, and the user will have to change systems (yikes !)



```
(base) ShriyanshPC~: kvm-ok
INFO: Your CPU does not support KVM extensions
INFO: For more detailed results, you should run this as root
HINT:   sudo /usr/sbin/kvm-ok
(base) ShriyanshPC~: █
```

Figure 3.2: This message indicates the system doesn't support KVM.

2. The docker engine starts automatically as the local machine starts, but sometimes, the engine decides to take a break unexpectedly and shuts down, thus killing your docker container. You may encounter the following error:



```
 [~]
 base  xondikoi  c532
Cannot connect to the Docker daemon at unix:///home/xondikoi/.docker/desktop/docker.sock. Is the docker daemon
running?

 [~]
 ✘ base  xondikoi  _
```

Figure 3.3: This message indicates the docker engine is disabled.

To enable the docker engine and the docker-desktop the following commands shall do the magic:

```
sudo systemctl start docker

systemctl --user start docker-desktop
```

This shall restart the engine and docker will be usable again.

3. You may also encounter compiler errors while compiling the `C++` source codes in the `../NtupleMaker/src/` directory. But that will be user's analyses specific errors, which can be solved by a little debugging. We'll leave that to the user to find the missing `semi-colon(;)` in the thousand lines of code.

3.2 Errors in ROOT installation

ROOT installation can be tricky and crash, you shall refer to the ROOT Installation Cheatsheet. But the installation process is quite tricky and gets stuck precisely at 71 percent. That's another death sentence, this means that the user will have to force quit the process and start all over again to build ROOT from source. So while building ROOT, it is recommended to use 75 % of the processor's resources and not use up all the resources entirely for the installation.

For example, if a machine has 8 cores, it is recommended to use 6 cores for the ROOT installation.

```
cmake --build . -- install -j6
```

That shall solve most of the problems for ROOT installation. All the dependencies should be met and the system libraries should be upto date.

3.3 Errors in the CMS-OpenData-Pipeline

The CMS-OpenData-Pipeline is very robust. If the user follows the set of instructions properly, that should result in no runtime errors. However, the CMS-OpenData-Pipeline currently works only on `CMSSW_5_3_32` and may run into errors in different versions of `CMSSW`.

Remarks from the developer

The CMS-OpenData-Pipeline is a novel idea to facilitate scientists and researchers outside the CMS Collaboration to be able to use the CMS-OpenData for their own analyses and search for new physics. We aim to make scientific knowledge open and accessible to everyone, but this novel effort has a long way to go. So, if any issues are found in the CMS-OpenData-pipeline, a pull request or raising an issue in the GitHub repository is appreciated.

With that said, let's make experimental particle physics fun for all !

Bibliography

[1] "Construction of a pipeline for facilatating easier & efficient analysis of CMS-OpenData" - Raj Handique - B.Sc Project (2024)

[2] https://root.cern/

[3] https://docs.docker.com/get-docker/

[4] https://github.com/cms-sw/cmssw/tree/CMSSW_5_3_X

[5] https://www.gnu.org/software/emacs/

[6] https://opendata.cern.ch/docs/cms-getting-started-aod

[7] https://opendata.cern.ch/record/1395