How to create your own container

Ben Brüers

DESY Zeuthen

28.03.2025

FH Sustainability Forum and DESY IT

Topical Computing Workshop “Containers”

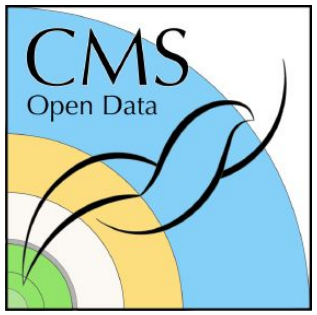
Why would I want to build a container?

- Software in HEP often has VERY specific requirements, e.g.
  - operating system → Alma Linux
  - compiler version → gcc 11
  - an ancient piece of code → Delphes v.2 (>10 years old)
  - three recent pieces of code with a specific version → HepMC3, Fastjet, Rivet
  - some environment variables that have to be defined → LD_LIBRARY_PATH

- Imagine you want to run this software on your PC (Ubuntu), then on the NAF (Alma Linux)
  - you will need quite some time to get it running on both systems

- Why not just use a container?
  - has all requirements, running the software is as easy as running the container!

Building an analysis container



- Will build an analysis container today
  - based on KiSelector framework
  - process CMS opendata, aim to “measure” Z-boson mass

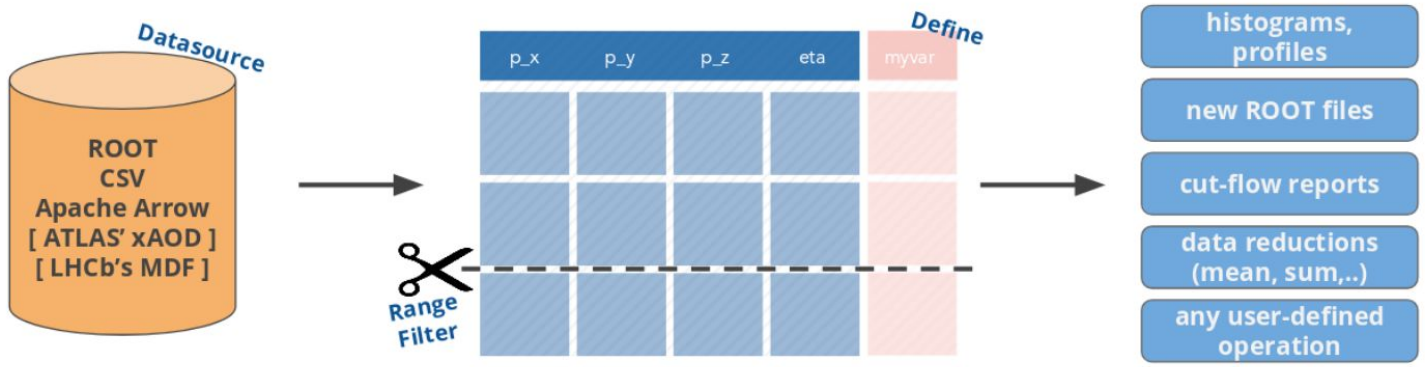
- Admittedly, the requirements of KiSelector are small (ROOT only)
  - at least on the NAF, where ROOT is installed
  - but what if you ran on Amazon’s AWS?
  - → **USE A CONTAINER**

- Building a KiSelector container as a relatively “easy” example of building a container
  - convenient: can be seamlessly used on your computer, the NAF, condor, the grid, everywhere!

KiSelector – a short introduction

- TTree processing & plotting framework, which can
  - run over TTrees, produces histograms
  - apply selections
  - generate cutflows
  - calculate / define new variables
  - plot histograms
  - output TTree

- Employs ROOT's powerful RDataFrame class



http://cds.cern.ch/record/2699587/files/10.1051_epjconf_201921406029.pdf

KiSelector – FileList.py – input files

```
FileList = {
    'data2016': # data2016 is a tag
    {
        'basedir': '/home/ben/data',
        'files': [
            'A4CE5E5C164E.root',
           ],
        'tTree': 'Events',
        'isData': True,  # processing data
    },

    'zjets': # zjets is also a tag
    {
        ...
    }
}
```

all files within one tag are used to produce a histogram

file path

name of TTree

KiSelector – VarList.py – what to histogram

variable taken directly from branch entry per event

binning / labeling

variable derived from branch of class vector per event

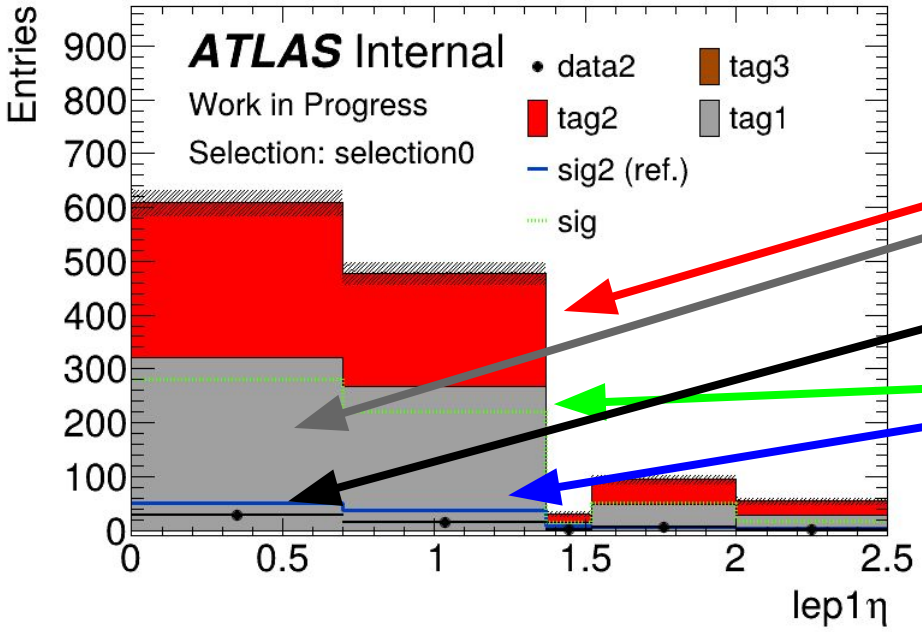
```
VariableList = {
    'nMuon': { # branch in TTree
        'numberOfBins': 8, 'xMin': 0, 'xMax': 8,
        'xLabel': 'number of muons',
    },

    'muonPt_1': { # leading muon pt
        'numberOfBins': 240, 'xMin': 0, 'xMax': 120,
        'unit': 'GeV/c',
        'xLabel': 'muon p_{T,1}',
        # content comes from branch Muon_pt → vector
        'newVariableDefinition':
'getVectorEntrySafe(Muon_pt, 0, -1)',
    },
    ...
}
```

KiSelector – SelList.py – Selections

```
SelectionList = {
        'diMuonMass': { # name of the selection
            'cuts': [
                    {'nMuon':              '>= 2' },
                    {'muonPt_1':           '> 50' }, # using new variable
                        ...
            ],
        }
}
```

KiPlotter – PlottingList.py – what to plot




```
PlottingList = {
    'background': [
    ],
    'data': [
        {'tag': 'data2016'},
    ],
    'signal': [
    ],
}
```

tag from FileList!

How to create our container?

- Many container frameworks out there…
  - Docker, Apptainer
- Will build Apptainer container today
  - well suited for high performance computing applications
- What do we need to do?
  - log in to the NAF
  - write container definition file
  - execute `apptainer build <container-name>.sif <definition-file>.def`

The Apptainer definition file

https://apptainer.org/docs/user/1.0/definition_files.html

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Base container on docker container for Ubuntu, found at https://hub.docker.com/.

→ use ROOT's docker container **"rootproject/root:latest"**

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

Do something, before building container (e.g. download file)

https://apptainer.org/docs/user/1.0/definition_files.html

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Files (not directories) to copy to container (i.e. this is “cp a b” not “cp -r a b”)

Syntax: path-builder path-container

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

Environment variables to define

https://apptainer.org/docs/user/1.0/definition_files.html

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Bash-script what to do when opening container with

apptainer run

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Bash-script what to do when opening container with

apptainer instance
(continuous running in bkg.)

The Apptainer definition file

https://apptainer.org/docs/user/1.0/definition_files.html

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

Test container after building

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Labels (e.g. version) & help message

The Apptainer definition file

```
Bootstrap: docker
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${APPTAINER_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $APPTAINER_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT

%test
    grep -q NAME=\"Ubuntu\" /etc/os-release
    if [ $? -eq 0 ]; then
        echo "Container base is Ubuntu as expected."
    else
        echo "Container base is not Ubuntu."
        exit 1
    fi

%labels
    Author alice
    Version v0.0.1

%help
    This is a demo container used to illustrate a def file that uses all
    supported sections.
```

https://apptainer.org/docs/user/1.0/definition_files.html

Important for us today

It’s your go!!

- Log-in to the NAF (if wanted also connect via vsCode)
  ```
  ssh <username>@naf
  ```
- Go to dust (will need ~1 GB disk space, danger of quota limit on AFS)
  ```
  cd /data/dust/user/<username>/
  ```
- Create a new directory (and move to it)
  ```
  mkdir containerTutorial && cd containerTutorial
  ```
- Clone the tutorial’s git repository (contains files for KiSelector, README)
  ```
  git clone --recursive
  https://gitlab.desy.de/fh-sustainability-forum/sustainable-cod
  ing-tutorial/container-make-your-own.git
  ```
- Change directory
  ```
  cd container-make-your-own
  ```

Container design

- For training purposes, want following behaviour:
  - automatic processing of an input file with KiSelector if using “run”, i.e.

```
apptainer run container.sif <input-file> <tree-name> <tag>
<output-file>
```

→ KiSelector should process <tree-name> in <input-file> as <tag> and produce <output-file>

- Could code this into the %runscript section
  - Difficult to test
  - → create bash-script

```
%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"
```

Container design

- For training purposes, want following behaviour:
  - automatic processing of an input file with KiSelector if using “run”, i.e.

```
apptainer run container.sif <input-file> <tree-name> <tag>
<output-file>
```

→ KiSelector should process <tree-name> in <input-file> as <tag> and produce <output-file>

want this command order!

- Could code this into the %runscript section
  - Difficult to test
  - → create bash-script

```
%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"
```

Task 1: entry bash script

⏱ 10 mins

- **Write a bash-script that:**
  - Checks if 4 arguments were given
  - Goes to the KiSelector installation path defined by var. `$PATH_TO_KISELECTOR`
  - Runs `python KiSelector.py` with arguments

```
-s $KISEL_CONFIGS/SelectionList.py -var  $KISEL_CONFIGS/VariableList.py --fileIsData
--fileName <input-file> --fileTree <tree> --fileTag <tag> -o <output-file>
```

Task 1: entry bash script

⏱ 10 mins

- **Write a bash-script that:**
  - Checks if 4 arguments were given
  - Goes to the KiSelector installation path defined by var. `$PATH_TO_KISELECTOR`
  - Runs `python KiSelector.py` with arguments

```
-s $KISEL_CONFIGS/SelectionList.py -var $KISEL_CONFIGS/VariableList.py --fileIsData
--fileName <input-file> --fileTree <tree> --fileTag <tag> -o <output-file>
```

- **Hints:**
  - To use command line arguments.: variables names are `$1` to `$4`
  - Check #args.: `if [ "$#" -ne 4 ]; then <do-smth-if-number-is-wrong-and-exit>; fi`

Task 1: entry bash script

🕛 10 mins

- **Write a bash-script that:**
  - Checks if 4 arguments were given
  - Goes to the KiSelector installation path defined by var. `$PATH_TO_KISELECTOR`
  - Runs `python KiSelector.py` with arguments

```
-s $KISEL_CONFIGS/SelectionList.py -var $KISEL_CONFIGS/VariableList.py --fileIsData
--fileName <input-file> --fileTree <tree> --fileTag <tag> -o <output-file>
```

- **Hints:**
  - To use command line arguments.: variables names are `$1` to `$4`
  - Check #args.: `if [ "$#" -ne 4 ]; then <do-smth-if-number-is-wrong-and-exit>; fi`

- **To test:**
  - set environment variables:

```
export PATH_TO_KISELECTOR=$(pwd)/kiselector
export KISEL_CONFIGS=$(pwd)/kiSelConfigs
```

  - use inputs from README (or from mini text here)

```
- input file
`/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2
_NanoAODv9-v1/130000/0A4230E2-0C75-604D-890F-A4CE5E5C164E.root`
- TTree `Events`
- tag `data2016`
- some output file name you came up with
```

Task 1: solution

5 mins

- Example solution for bash script
- Pitfall: output file is in kiselector directory, because we cd there before running KiSelector (so the output is made relative to the kiselector dir)
  - Fix by using $(pwd) upfront the output file directory (i.e. using absolute path)

```
if [ "$#" -ne 4 ]; then echo "wrong number of args, should be 4 "; exit; fi

cd $PATH_TO_KISELECTOR

python KiSelector.py \
      -s $KISEL_CONFIGS/SelectionList.py -var $KISEL_CONFIGS/VariableList.py --fileIsData \
      --fileName $1 --fileTree $2 --fileTag $3 -o $4
```

Task 2: build container

⏱ 10 mins

- Using the just created script, we will now build the container
- Open a new file `kiSelAna.def`
- Define an apptainer container that:
  - is based on CERN ROOT's docker container **rootproject/root:latest**
  - **contains the files** in the `kiSelConfigs` directory (KiSelector defs.)
  - **clones** the KiSelector repo `https://gitlab.desy.de/ben.brueers/kiselector.git`
  - when opened with "run", executes the bash-script from previous task (hint: pass cmd. line args. to the script by `$*`) (hint: ensure script is executable `chmod u+x <script-name>`)
  - defines the env. variables `PATH_TO_KISELECTOR` and `KISEL_CONFIGS`

Task 2: build container

10 mins

- Using the just created script, we will now build the container
- Open a new file `kiSelAna.def`
- Define an apptainer container that:
  - is based on CERN ROOT's docker container **rootproject/root:latest**
  - **contains the files** in the `kiSelConfigs` directory (KiSelector defs.)
  - **clones** the KiSelector repo `https://gitlab.desy.de/ben.brueers/kiselector.git`
  - when opened with "run", executes the bash-script from previous task (hint: pass cmd. line args. to the script by `$*`) (hint: ensure script is executable `chmod u+x <script-name>`)
  - defines the env. variables `PATH_TO_KISELECTOR` and `KISEL_CONFIGS`
- To build the container `apptainer build kiSelAna.sif kiSelAna.def`
- To test the container `apptainer run kiSelAna.sif`
- For interactive access (debugging…) `apptainer shell kiSelAna.sif`

Task 2: build container

10 mins

- Using the just created script, we will now build

- Open a new file `kiSelAna.def`

- Define an apptainer container that:
  - is based on CERN ROOT's docker container **rootproject/root:latest**
  - **contains the files** in the `kiSelConfigs` directory (KiSelector defs.)
  - **clones** the KiSelector repo `https://gitlab.desy.de/ben.brueers/kiselector.git`
  - when opened with "run", executes the bash-script from previous task (hint: pass cmd. line args. to the script by `$*`) (hint: ensure script is executable `chmod u+x <script-name>`)
  - defines the env. variables `PATH_TO_KISELECTOR` and `KISEL_CONFIGS`

- To build the container `apptainer build kiSelAna.sif kiSelAna.def`
- To test the container `apptainer run kiSelAna.sif`
- For interactive access (debugging…) `apptainer shell kiSelAna.sif`

Hint: by default, all operations in all sections are executed in the root directory **"/"**.
Keep that in mind!!

Hint: use
- %files
- %environment
- %post
- %runscript

Task 2: solution

5 mins

- Discuss example solution for def file

```
Bootstrap: docker
From: rootproject/root:latest

%files
 kiSelConfigs/* /kiSelConfigs/
 entryScript.sh /entryScript.sh

%environment
 export PATH_TO_KISELECTOR=/kiselector
 export KISEL_CONFIGS=/kiSelConfigs/

%post
 chmod u+x /entryScript.sh
 git clone
https://gitlab.desy.de/ben.brueers/kiselector.git

%runscript
 /entryScript.sh $*
```

Important! Always relative to root directory “/”

Task 3: run container locally

5 mins

- Want to analyse CMS open data and “measure” Z-boson mass
- Downloaded CMS open data for you already at
  ```
  /pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2_NanoAODv9-v1/
  ```

- Try to run your container with one of the downloaded files, i.e.
  ```
  apptainer run kiSelAna.sif <the-file-you-chose> Events data2016 <output-file>
  ```

!! You most must put a $(pwd) in front of your output file or use an absolute path, because otherwise the container will try to write the file into the container, rather than out !!

Task 3: run container locally

⏱ 5 mins

- Want to analyse CMS open data and “measure” Z-boson mass
- Downloaded CMS open data for you already at
  `/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2_NanoAODv9-v1/`

- Try to run your container with one of the downloaded files, i.e.
  `apptainer run kiSelAna.sif <the-file-you-chose> Events data2016 <output-file>`

```
OSError: The path '/pnfs/desy.de/<...>.root" does not exist. Please correct the path to your root file(s) of TTrees.
```

Task 3: run container locally

🕛 5 mins

- Want to analyse CMS open data and “measure” Z-boson mass
- Downloaded CMS open data for you already at
  `/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2_NanoAODv9-v1/`

- Try to run your container with one of the downloaded files, i.e.
  `apptainer run kiSelAna.sif <the-file-you-chose> Events data2016 <output-file>`

```
OSError: The path '/pnfs/desy.de/<...>.root" does not exist. Please correct the path to your root file(s) of TTrees.
```

Apptainer does not bind /pnfs! Use
`apptainer run --bind /pnfs/desy.de kiSelAna.sif ...`

Task 4: use container on HTCondor

10 mins

- Now ready to use container on HTCondor
- Create directory structure for output
  `mkdir -p condorRun/output condorRun/joblog`

- Create condor job file `condorRun/KiSelector.condor`
- Test with `condor_submit condorRun/KiSelector.condor`

```
+MySingularityImage = "kiSelAna.sif"

# has to be an absolute path
executable = <absolute-path-in-the-container-to-the-bash-script-you-wrote>

# $Ff(path) turns a relative path into an absolute path
arguments = "$(aFile) $(tree) $(tag) $Ff(condorRun/output/)$(Cluster)_$(Process).root"

# do not transfer exectuable, as it is in the container
transfer_executable = false
# apptainer is hungry and the container big (Ubuntu-based)
request_memory = 4G

log    = condorRun/joblog/KiSelector_$(Cluster)_$(Process).log
output = condorRun/joblog/KiSelector_$(Cluster)_$(Process).out
error  = condorRun/joblog/KiSelector_$(Cluster)_$(Process).err
queue aFile, tree, tag from (
/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2_NanoAODv9
-v1/130000/0A4230E2-0C75-604D-890F-A4CE5E5C164E.root, 'Events', data2016
)
```

Task 4: use container on HTCondor

10 mins

- Now ready to use container on HTCondor
- Create directory structure for output
  `mkdir -p condorRun/output condorRun/joblog`

- Create condor job file `condorRun/KiSelector.condor`
- Test with `condor_submit condorRun/KiSelector.condor`

Replace this!!

```
+MySingularityImage = "kiSelAna.sif"

# has to be an absolute path
executable = <absolute-path-in-the-container-to-the-bash-script-you-wrote>

# $Ff(path) turns a relative path into an absolute path
arguments = "$(aFile) $(tree) $(tag) $Ff(condorRun/output/)$(Cluster)_$(Process).root"

# do not transfer exectuable, as it is in the container
transfer_executable = false
# apptainer is hungry and the container big (Ubuntu-based)
request_memory = 4G

log    = condorRun/joblog/KiSelector_$(Cluster)_$(Process).log
output = condorRun/joblog/KiSelector_$(Cluster)_$(Process).out
error  = condorRun/joblog/KiSelector_$(Cluster)_$(Process).err
queue aFile, tree, tag from (
/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAODv2_NanoAODv9
-v1/130000/0A4230E2-0C75-604D-890F-A4CE5E5C164E.root, 'Events', data2016
)
```

important!! (both)

Task 4: solution

5 mins

- Check that condorRun/output/…root exists and is non-empty
- Check that condorRun/joblog/…out looks like this

```
Saving temporary files to /tmp/tmp_1bq0u_3
Reading list of files and TTrees from: /tmp/tmp_1bq0u_3/FileList_tmp.txt.
Making sure all provided files and TTrees exist. Further checking for the normalisation histogram
and entries.
Reading list of selections from: /kiSelConfigs//SelectionList.py.
Reading list of variables from: /kiSelConfigs//VariableList.py.
Saving resulting histograms to:
/data/dust/user/brueers/sustainable-coding-workshop/containers/runthrough/container-make-your-own
/condorRun/output/534288_0.root
Enabling ROOT's internal multi-threading
Used number of threads (ROOT's multi-threading): 1
               [1/1] EXECUTING LOOP of the events and filling the histograms. This can take a
very long time. On TTree "Events" and file
"/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleM
uon/NANOAOD/UL2016_MiniAODv2_NanoAODv9-v1/130000/0A4230E2-0C75-604D-890F-A4CE5E5C164E.root"
               [1/1] DONE looping over the events and filling the histograms for TTree "Events"
and file
/pnfs/desy.de/FH-Sustainability-Computing-Workshop/cms/Run2016G/SingleMuon/NANOAOD/UL2016_MiniAOD
v2_NanoAODv9-v1/130000/0A4230E2-0C75-604D-890F-A4CE5E5C164E.root"
```

Task 5: Work with more powerful container

⏱ 10 mins

- The solutions container has extra features
  - process file with
  - condor-file creation with
  - merging of KiSelector files with
  - easy plotting with

```
apptainer run process
apptainer run condorfile
apptainer run merge
apptainer run plot
```

Task 5: Work with more powerful container

⏱ 10 mins

- The solutions container has extra features
  - process file with `apptainer run process`
  - condor-file creation with `apptainer run condorfile`
  - merging of KiSelector files with `apptainer run merge`
  - easy plotting with `apptainer run plot`

!! Don't forget $(pwd)!!

- Now:
  - build solutions container

    ```
    apptainer build kiSelAna.sif .solutions/kiSelAna.def
    ```

  - create condor docker file

    ```
    apptainer run --bind /pnfs/desy.de kiSelAna.sif condorfile
    <your-condor-submission-directory> <path-to-apptainer-image>
    <path-to-file-list>
    ```

    e.g. `apptainer run --bind /pnfs/desy.de kiSelAna.sif condorfile $(pwd)/condorAll $(pwd)/kiSelAna.sif $(pwd)/kiSelConfigs/FileList.py`

  - submit jobs

    ```
    condor_submit <your-condor-submission-directory>/KiSelector.condor
    ```

  - monitor progress `watch condor_q`

Task 6: Merge results and plot

10 mins

- Have processed all the data!!
- Can use container to merge resulting histograms and plot data

- Merge:

```
apptainer run kiSelAna.sif merge <output-file> <input-files>
```

- Plot:

```
apptainer run kiSelAna.sif plot <output-from-merging> <directory-to-store-plots>
```

!! Don't forget $(pwd)!!

Task 6: Merge results and plot

10 mins

- Have processed all the data!!
- Can use container to merge resulting histograms and plot data

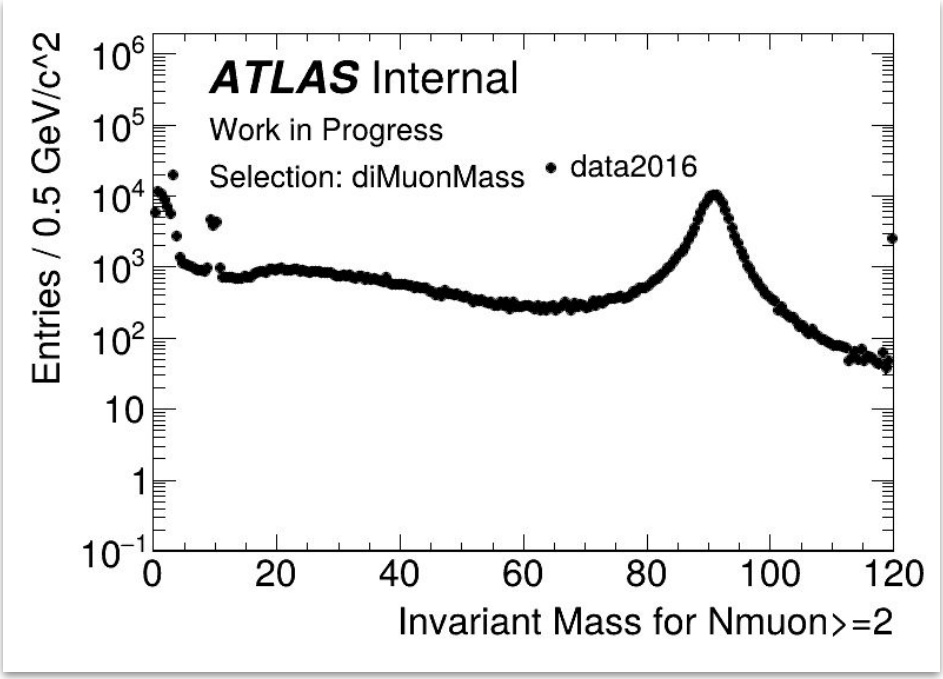
- Merge:

```
apptainer run kiSelAna.sif merge <output-file> <input-files>
```

- Plot:

```
apptainer run kiSelAna.sif plot <output-from-merging> <directory-to-store-plots>
```

- Isn't it beautiful? 😍



Ta… plot

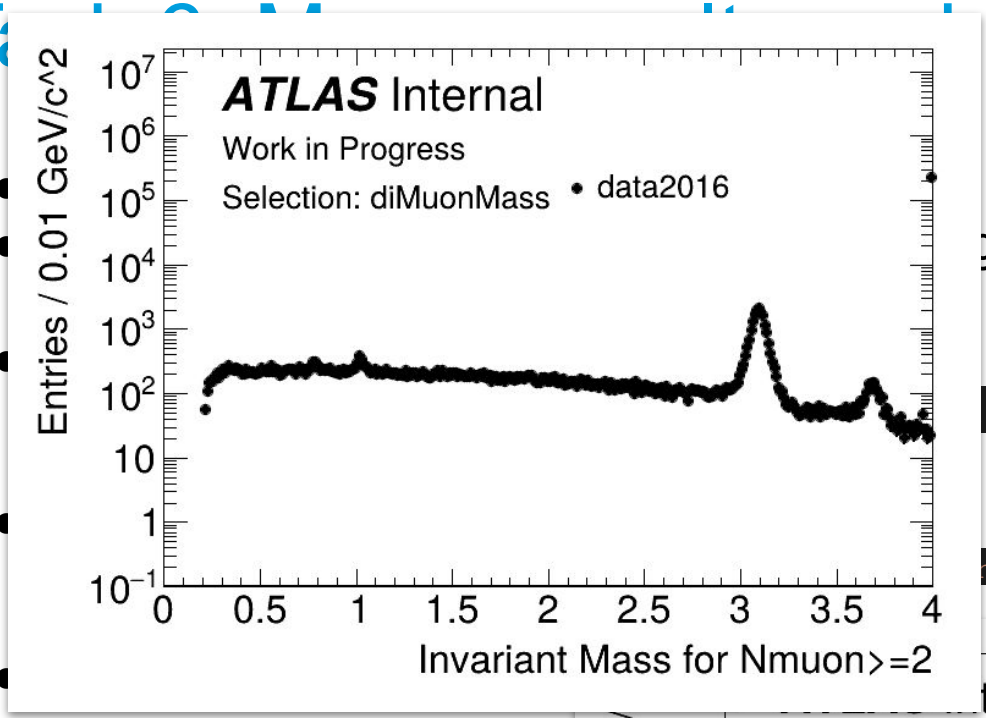
10 mins

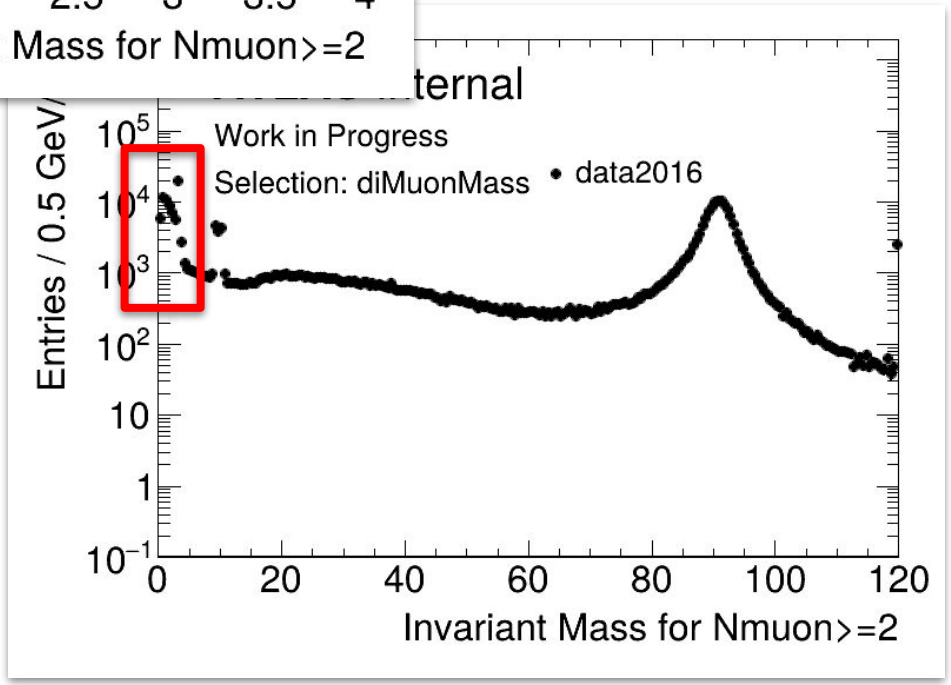
- …
- …
- …
- …
- …

…grams and plot data

```
…-file> <input-files>
```

```
…ng> <directory-to-store-plots>
```





Ta… plot

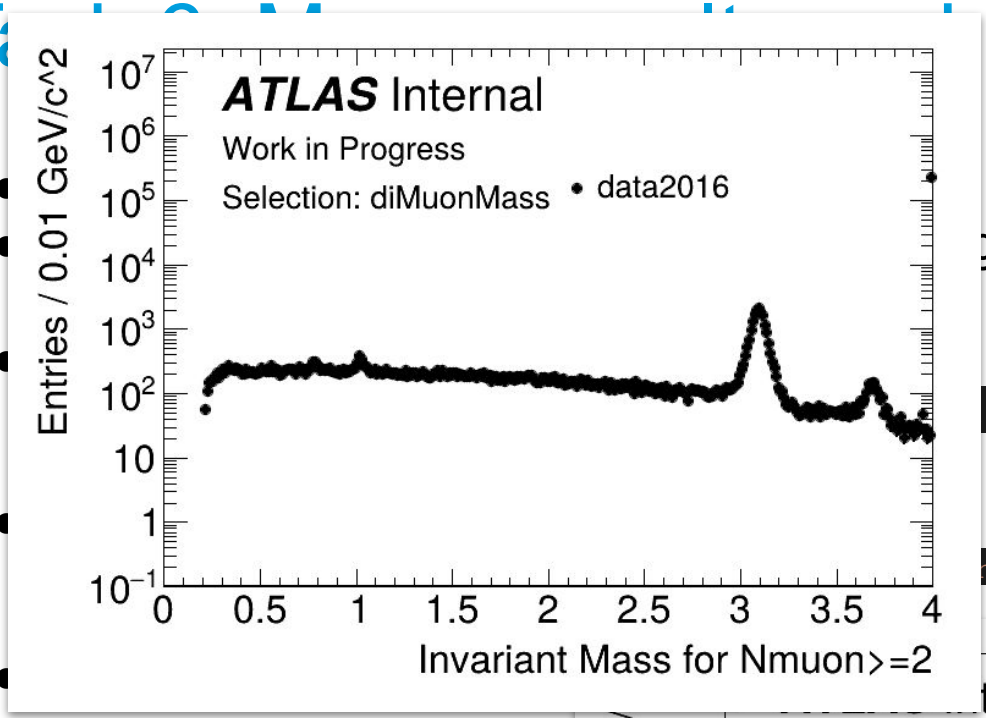
10 mins

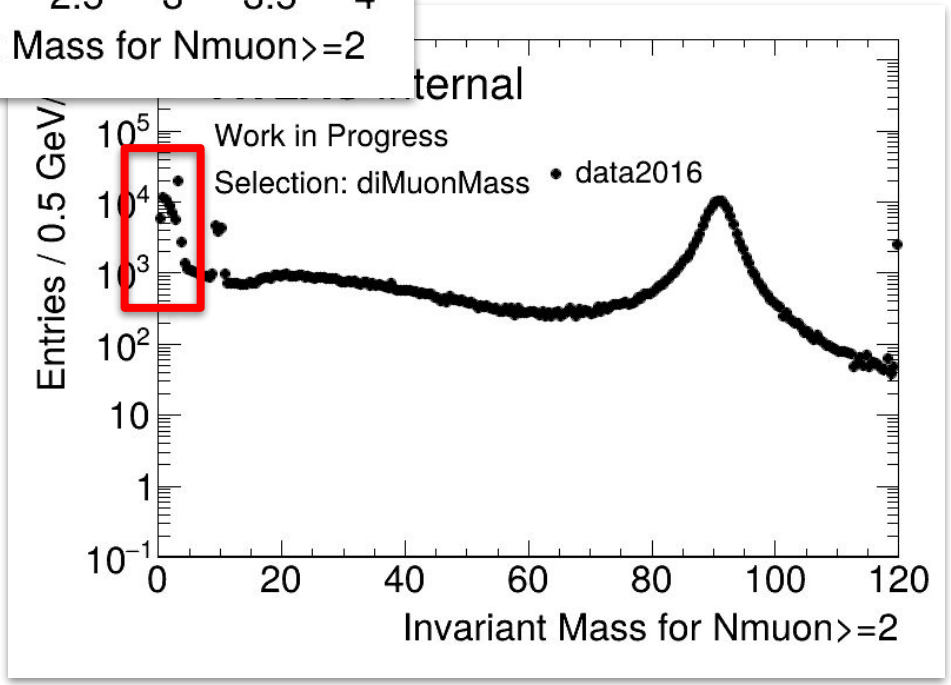
- …
- …
- …
- …
- …

…grams and plot data

```
…-file> <input-files>
```

```
…ng> <directory-to-store-plots>
```





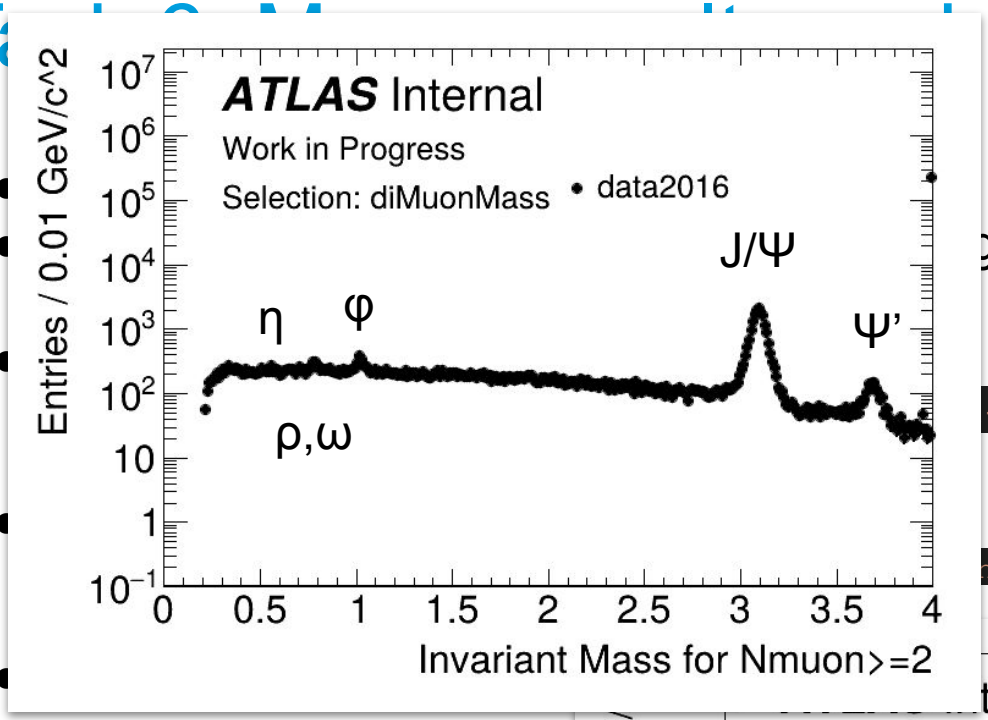
Ta… plot

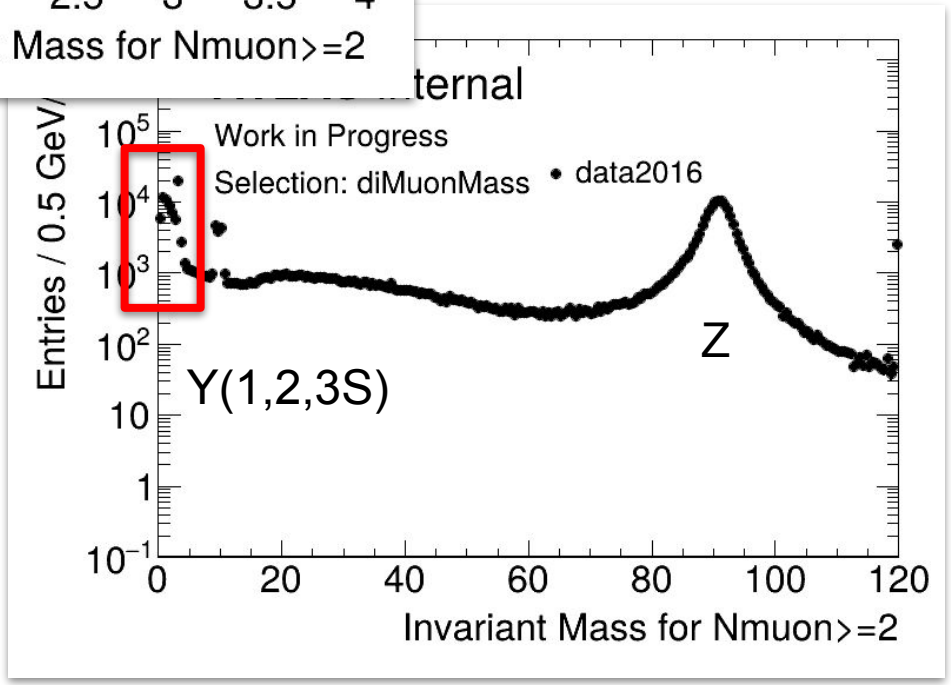
10 mins

- …
- …grams and plot data
- …
- …
- …

```
…-file> <input-files>
```

```
…ng> <directory-to-store-plots>
```





https://opendata.cern.ch/record/12342

Summary

- Learnt how to build a container with Apptainer
  - executes KiSelector to produce histograms from ntuples

- Built custom entry script executed with “apptainer run”
- Built container
- Used container on the NAF
- Used container on HTCondor
- Used container to merge and plot results

Difficulties observed today

- <live-mention-them-here>
-

Thank you for taking part!

Backup slides

KiSelector – inputs for histogramming

```
FileList = {
    'data2016': # data2016 is a tag
    {
        'basedir': '/home/ben/data',
        'files': [
            'A4CE5E5C164E.root',
           ],
        'tTree': 'Events',
        'isData': True,  # processing data
    },

    'zjets': # zjets is also a tag
    {
        ...
    }
}
```

```
VariableList = {
    'nMuon': { # branch in TTree
        'numberOfBins': 8, 'xMin': 0, 'xMax': 8,
        'xLabel': 'number of muons',
    },

    'muonPt_1': { # leading muon pt
        'numberOfBins': 240, 'xMin': 0, 'xMax': 120,
        'unit': 'GeV/c',
        'xLabel': 'muon p_{T,1}',
       # content comes from branch Muon_pt → vector
        'newVariableDefinition':
'getVectorEntrySafe(Muon_pt, 0, -1)',
    },
    ...
}
```

```
SelectionList = {
       'diMuonMass': { # name of the selection
            'cuts': [
                    {'nMuon':            '>= 2' },
                    {'muonPt_1':         '> 50' }, # using new variable
                        ...
            ],
       }
}
```