Container Technology for Phenomenology Tools

Isabel Campos, Alvaro López and Pablo Orviz - CSIC (Spain)
Jorge Gomes – LIP (Portugal)

Sven Heinemeyer  CSIC (Spain)
Emanuele Bagnaschi DESY (Germany)
Outline

• Motivation: Containers and Phenomenology codes
• A pilot case: Mastercode
• Building a container for Mastercode
• Automation tools for building and maintaining containers
• Running Mastercode everywhere: udocker
  • On your desktop
  • On a Linux Cluster
  • On the Grid
Containers - definition

“Linux Containers” is a technology provided by the Linux kernel, to “contain” a group of processes in an independent execution environment: this is called a “container”.

- The “group of processes” can be for example a software package
- Containers are the future of software provisioning both in industry and academia:
  eg. see https://access.redhat.com/articles/1353593

Advantages of Containers versus classical Virtual Machines

- Containers are much more light-weighted than a Virtual Machine: they start faster, run faster, and many more fit on the same hardware host machine
- They provide an enormous simplification of the software deployment processes in any Linux platform with a kernel advanced enough to support container technology (kernel version > 3.8.x)

We have available nowadays two types of virtualization:

- Virtual Machines with Hypervisors: (Xen, KVM) i.e. another OS instance by Virtualization
- Containers: Lightweight Process level handled by the linux kernel
Containers - docker

• To build containers, we use the system software provided by “Docker”
• Docker is an Open Source Linux container engine.
  • [https://www.docker.io/](https://www.docker.io/)
  • git repository at: [https://github.com/dotcloud/docker.git](https://github.com/dotcloud/docker.git)

• Docker is optimized for the deployment of applications as containers
  • Other options like the available lxc are more oriented to deployment of system software/sysadmin

• One can generate a completely tailored “docker image” of any Linux Operating System, with all
the required libraries, compilers, source codes,…

  “docker image” ≠ “VM image”

  • It is a Snapshot of the OS which has been modified to run as a container
  • In particular, when comparing with an image for a hypervisor (VM), it has no Kernel.
  • Still they are large, minimum 200MB (essentially libraries).
  • Those “images” will not run when launched with a hypervisor!
**Collaborative effort** between Experimentals and Theorist in High Energy Particle Physics

The code takes as input experimental data coming from Particle Accelerators and Astrophysics observations, in order to build a consistent model of Nature explaining the experimental data.

- It focuses on the search for Supersymmetric models

It is supported by an ERC Advanced Grant: **“Exploring the Terauniverse with the LHC, Astrophysics and Cosmology”** (J. Ellis)

See [http://johne.web.cern.ch/johne/](http://johne.web.cern.ch/johne/)

---

**Members of the MasterCode collaboration**


**Codes**

- RGE running: SoftSUSY
- Higgs, \((g-2)_{\mu}\): FeynHiggs
- Higgs observables: HiggsBounds and HiggsSignal
- B-physics: SuFla
- B-physics: SuperIso
- EWPO: FeynWZ
- Dark matter observables: Micromegas
- Dark matter observables: DarkSUSY
- Recast of LHC searches: Atom
- Recast of LHC searches: Scorpion
- SUSY decay modes: SDECAY

---

SUSY 2016, Melbourne.
Monday, 3.40pm:

*New SUSY Fits with MasterCode*
(Sven Heinemeyer)

Tuesday, 4.30pm:

*SUSY Predictions for Dark Matter Searches from MasterCode*
(Matt Dolan)
A pilot case: Mastercode

- Mastercode is an “über-code” written in C++ which connects all the different codes
- It does **parametric runs**: scanning through large parameter spaces, in **single core** mode

- Those original codes are treated as subroutines, or sub-codes
- The sub-codes are written in C++ or Fortran. Many different authors, often legacy code is there....
- **The “über-code” is not public**...

- **Perfect candidate to be “containerized”**: problems to run the code in current computing centers because it is difficult to install (library dependences and compatibility issues with local software, etc...)  
- One of the main problems they have: finding a computing center where it works!
Building a container for Mastercode

Description of the automated process via github + dockerhub:

1. We have created a project in github: [https://github.com/indigo-dc/docker-mastercode](https://github.com/indigo-dc/docker-mastercode)
2. The project contains just a Dockerfile to build a fedora-based container that includes all the libraries and system software needed to run Mastercode
3. We then created a hub in docker for the container to be built automatically by docker [https://hub.docker.com/r/indigodatacloud/docker-mastercode](https://hub.docker.com/r/indigodatacloud/docker-mastercode)
4. Finally, we connected github-docker to automatically build the container
Github project: Dockerfile

Docker Image adapted to the usage of Phenomenology Tools
We created a project in dockerhub, which reads as source repository our github Mastercode repository.

In particular dockerhub aware of the changes we may commit to the Dockerfile.

Dockerhub provides a service to build Containers automatically on their cloud Facilities.

We use this service to maintain up-to-date The container for Mastercode.
Connect dockerhub ↔ github

A change in the dockerfile, triggers automatically a new build of the container
Using the Mastercode container with docker

On your local Linux box, with root user privileges:

1. Install docker with your favourite package manager (eg. "# yum install docker")
2. Start the docker daemon: # systemctl start docker
3. Download the container: # docker pull indigodatacloud/docker-mastercode
4. Run it

```
# docker run -t -i -v $HOME \
-w $HOME/mcpp-master \ 
indigodatacloud/docker-mastercode \ 
/bin/bash
```

Notice: the container does not contain the code, only the environment necessary to run it:

- Developpers keep doing their own modifications to Mastercode in their private copy.
- In MacOS, download the Docker Tool Box for Mac, and start from (3).
- Yes, you can run Mastercode on your Mac!
On MacOS running boot2docker:

Using default tag: latest
^C

Using default tag: latest

latest: Pulling from indigodatacloud/docker-mastercode
a3ed95caeb02: Pull complete
236608c7b546: Pull complete
75c50b6518be: Pull complete
e9058dcbc910: Pull complete
7265dc1bc8b1: Pull complete
0912b41ddd90: Pull complete
977aca495a74: Pull complete
ebfcb6c1d599: Pull complete
fa48904f716: Pull complete
f2d0727ff9b49: Pull complete
92d8d3220efd: Pull complete
Digest: sha256:fb0dc2d7998159d95132535cc13d9ba3f290cb5b00d96c7ad7840bccc8f18
Status: Downloaded newer image for indigodatacloud/docker-mastercode:latest

Using default tag: latest

latest: Pulling from indigodatacloud/docker-mastercode
a3ed95caeb02: Pull complete
236608c7b546: Pull complete
75c50b6518be: Pull complete
e9058dcbc910: Pull complete
7265dc1bc8b1: Pull complete
0912b41ddd90: Pull complete
977aca495a74: Pull complete
ebfcb6c1d599: Pull complete
fa48904f716: Pull complete
f2d0727ff9b49: Pull complete
92d8d3220efd: Pull complete
Digest: sha256:fb0dc2d7998159d95132535cc13d9ba3f290cb5b00d96c7ad7840bccc8f18
Status: Downloaded newer image for indigodatacloud/docker-mastercode:latest

Boot2docker uses virtualization (VirtualBox) to run the Tiny Linux system where the containers are executed. It does NOT provide native execution.
On a Linux Cluster / HPC system

- Adoption of docker is being very slow in HPC centers
- Thus the tipical situation is that docker is not installed and one cannot run containers without some support from the system software.
- In general Docker adoption will be slow in any computing farm or interactive linux system shared by many users.

  - It will take time for sysadmins to overcome the concerns of their security teams.
  - It is yet another service to maintain...
  - .... you name it.
We have developed “udocker”

- tool to execute content of docker containers in user space when docker is not available
  - enables download of docker containers from dockerhub
  - enables execution of docker containers by non-privileged users
- can be used to execute the content of docker containers in Linux batch systems and interactive clusters managed by others
- wrapper around other tools to mimic docker capabilities
  - current version uses proot to provide a chroot like environment without privileges (runs on CentOS 6, CentOS 7, Fedora, Ubuntu)

- More info and downloads at:
  - https://indigo-dc.gitbooks.io/udocker/content/doc/user_manual.html
We have developed “udocker”

• Examples:

  # download, but could also import or load a container exported/save by docker
  $ udocker.py pull fedora:latest
  $ udocker.py create --name=myfed fedora:latest

  # install Firefox, and run it accessing /dev, for audio (feature not supported in docker!)
  $ udocker.py run myfed /bin/bash yum install –y firefox
  $ udocker.py run --bindhome --hostauth --hostenv -v /sys -v /proc \n    -v /var/run -v /dev --user=myusername --dri myfed firefox

  udocker is NOT an alternative to docker: we need the container image built by docker.

  Is a tool to handle and run containers with regular user privileges and/or when docker is not
  available for some reason: it is very convenient to access clusters and Grid resources

  SUSY 2016, Melbourne.
We have developed “udocker”

Basic description:

- Everything is stored in the $HOME or some other directory belonging to the user (tunable parameter).
- Container layers are download to the above specified directory
- Directory trees can be created/extracted from these container layers
- `proot` uses the debugger ptrace mechanism to change pathnames and execute transparently inside a directory tree
- No impact on read/write or execution, only impact on system calls using pathnames (ex. open, chdir, etc)
- **Does not require installation of software** in the host system:
  - `udocker` is a python script
  - `proot` is statically compiled
Running Mastercode with udocker: Linux Cluster

1. Download udocker from: https://github.com/indigo-dc/udocker

2. Download the container: 
   ```
   $ ./udocker.py pull indigodatacloud/docker-mastercode
   ```

3. Make sure you have enough space to uncompress it by pointing to a proper directory (default is $HOME): 
   ```
   export UDOCKER_DIR=/MY_LARGE_FILESYSTEM/userabc/.udocker
   ```

4. Create the container directory tree on your user space:
   ```
   $ ./udocker.py create indigodatacloud/docker-mastercode
   bb889c79-2872-37f3-adad-cd9e937dd6f0
   ```

5. You probably want to give it a nicer name:
   ```
   $./udocker.py name bb889c79-2872-37f3-adad-cd9e937dd mymastercode
   ```
Running Mastercode with udocker: Linux Cluster: batch script

export MASTERDIR=/gpfs/csic_users/userabc/mastercode
export UDOCKER_DIR=$MASTERDIR/.udocker

../udocker-master/udocker.py run --hostauth \
-v /home/csic/cdi/ica/mcpp-master \
-v /home/csic/cdi/ica \
-user=userabc \
-w /home/csic/cdi/ica/mcpp-master mastercode \
'/bin/bash -c "pwd; ./udocker-mastercode.sh"'

-hostauth : to use the /etc/passwd of the host machine

-v makes directories available inside the container

-user: your userid

-w working directory from where the commands will be issued

Where **udocker-mastercode.sh** is the (usual) command line sequency to execute Mastercode

./mc_point.py --run-mode mc-cmssm --predictors all --inputs 500 600 0 10 --print-mc-spectrum > output.txt
Conclusions

• We have shown how it is now possible to automatize container building for a complex case of a Phenomenology code with the help of docker

• We have developed a tool to run Mastercode without docker, and without worrying about modifying the system software
  • On a desktop (Linux / MacOS)
  • On a Linux Cluster
  • On the Grid

• We are very confident that this will be a huge step towards the facilitation of the usage of Mastercode.
Questions?

“Using udocker we can run the container as an unprivileged user
Without needing any additional system software”