NIHR | Guy's and St Thomas' Biomedical Research Centre

KING'S College LONDON

Easy-Build Singularity Containers

Jörg Saßmannshausen
Guy's and St Thomas' NHS Foundation Trust and King's College London, UK

Biography:

1983-1986: apprenticeship as laboratory assistant of Chemistry (Germany)
1993-1994: MSc in Chemistry, University of East Anglia (UK)
1995-1997: PhD in Chemistry, University of Leeds (UK)
1998-2010: various *post-doctoral* positions in UK, Germany and Austria
2010: *venia legendi* in Inorganic Chemistry, Technical University of Graz (Austria)
2010: Change from daytime chemist to HPC, *gentleman scientist* now
2010-2019: Various positions at University College London, Francis Crick Institute (UK)
2019-current: Guy's and St. Thomas' Biomedical Research Centre in London (UK)

What is science?

In science, we are doing experiments that are reproducible.

What is science?

In science, we are doing experiments that are reproducible.

In the laboratory, we repeat an analysis three times to iron out any random errors.

What is science?

In science, we are doing experiments that are reproducible.

In the laboratory, we repeat an analysis three times to iron out any random errors.

On the HPC cluster, we only run one calculation, often for a long time, but we do not repeat this run.

Why?

What is science?

In science, we are doing experiments that are reproducible.

In the laboratory, we repeat an analysis three times to iron out any random errors.

On the HPC cluster, we only run one calculation, often for a long time, but we do not repeat this run.

Why?

Part of the reason is that the resource HPC is still expensive, part of it is that the underlying maths is an exact science.
However: that requires the installed program is exact as well!
This implies the installation must be done in a reliable and reproducible way and must be benchmarked!

Part of the reason is that that the resource HPC is still expensive, part of it is that the underlying maths is an exact science.
However: that requires the installed program is exact as well!
This implies the installation must be done in a reliable and reproducible way and must be benchmarked!

The solution for this is not to install software manually but using well defined installation scripts which install and test the software. Thus, an installation on the HPC cluster in London can be reproduced with ease at a different cluster elsewhere. As the same tests are run at both places, we can have some confidence the production runs will produce the same results, subject to well defined error margines.

The solution for this is not to install software manually but using well defined installation scripts which install and test the software. Thus, an installation on the HPC cluster in London can be reproduced with ease at a different cluster elsewhere. As the same tests are run at both places, we can have some confidence the production runs will produce the same results, subject to well defined error margines.

Problem: This requires expert knowledge and time. The scientist I am working with are for example clinicians who, especially right now, do not have the time to install software on HPC clusters or their desktop machine. They are currently busy doing research in the sequencing of Sars-CoV-2, the virus which is causing the Covid-19 illness. This sequencing requires a number of programs to run in sequence and here the used program and the order are important. Thus, in order to repeat what has been done here in London, the same programs in the same sequence need to be run elsewhere.

Solution: Instead of installing the same programs all over again and make sure the order of execution stays the same, we are writing scripts which are executing the programs in a well defined order. Furthermore, we pack the whole environment into a *container*, which can be shipped elsewhere.

Solution: Instead of installing the same programs all over again and make sure the order of execution stays the same, we are writing scripts which are executing the programs in a well defined order. Furthermore, we pack the whole environment into a *container*, which can be shipped elsewhere.

There are two commonly used container systems:
Docker and **Singularity**.
Both have their advantages and disadvantages.
Docker requires a daemon process to run with root-privileges in the background. This might be problematic in a HCP environment.
Singularity does not have this requirement. The singularity container can be executed in the user-space environment. Also, with Singularity containers you can for example use tools like autofs mounts. Furthermore, you can convert an existing Docker image into a Singularity container without much of a problem.
As Nextflow, a popular program to write so called pipelines, is working with both, Docker and Singularity container, it makes it easier to use and mainly depends on the availability of the engine running on the HPC cluster.

New problem: as said before, I am working with clinicians who want to do science and not writing code. Some are really good at writing code, no doubt, but others are better, or prefer to be in the lab doing experiments and only need the programs as a tool to do things. It is like driving a car: not every car-driver is interested in the thermodynamic processes happening in a combustion engine. That does not mean they are bad drivers though!

New problem: as said before, I am working with clinicians who want to do science and not writing code. Some are really good at writing code, no doubt, but others are better, or prefer to be in the lab doing experiments and only need the programs as a tool to do things. It is like driving a car: not every car-driver is interested in the thermodynamic processes happening in a combustion engine. That does not mean they are bad drivers though!

Solution: Give these clinicians an easy to use tool, which allows them to create a Singularity container without much in-depth knowledge of how to do that so they can concentrate on what they are good at, rather then get frustrated with something they just require for their research. Realistically, these days hardly anybody is making their own glassware for the lab, it just will be ordered from a catalogue. So if we apply a similar principle here, give researchers a kind of 'catalogue' which allows them to create the tool they need, here a Singularity container, they can spent more time doing the research and less time on sometimes frustrating software installation.

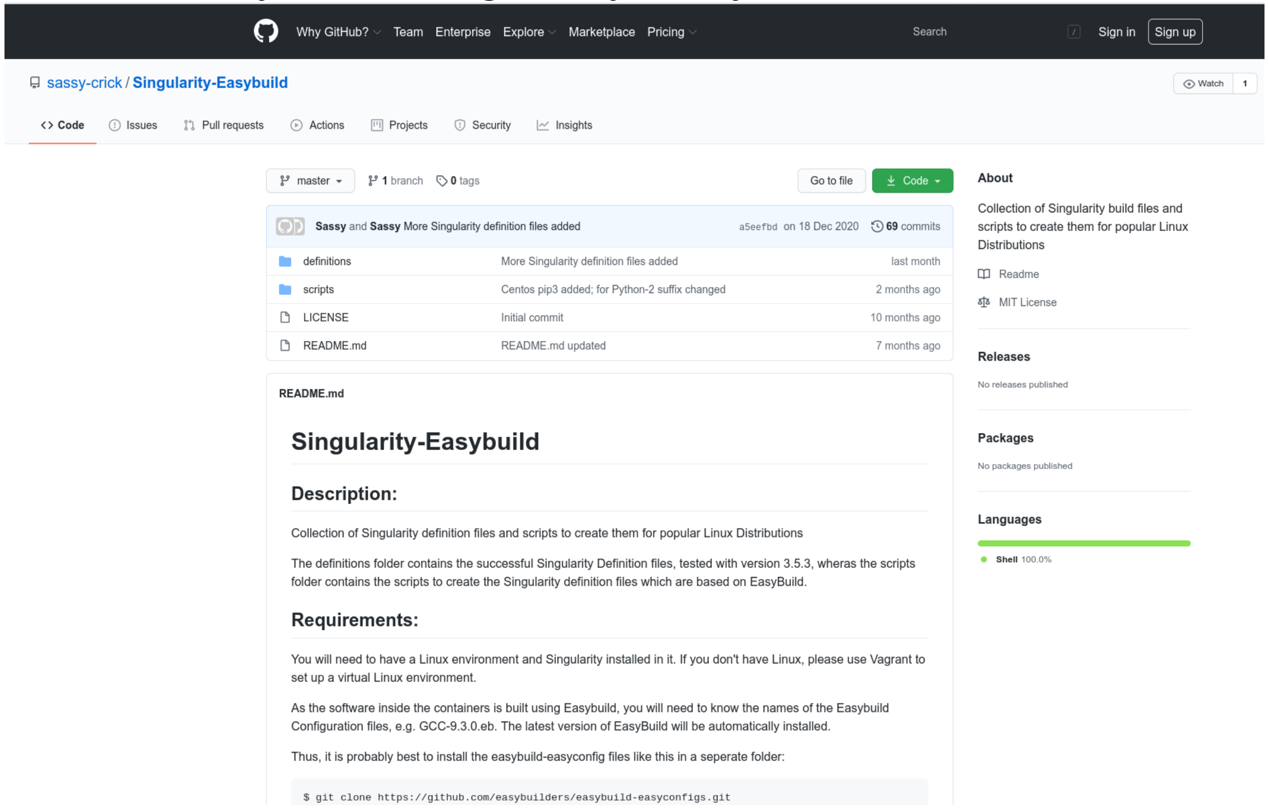
Solution: Give these clinicians an easy to use tool, which allows them to create a Singularity container without much in-depth knowledge of how to do that so they can concentrate on what they are good at, rather then get frustrated with something they just require for their research. Realistically, these days hardly anybody is making their own glassware for the lab, it just will be ordered from a catalogue. So if we apply a similar principle here, give researchers a kind of 'catalogue' which allows them to create the tool they need, here a Singularity container, they can spent more time doing the research and less time on sometimes frustrating software installation.

How to achieve this?

I am providing a set of very easy to use bash commands, which allow the user to automatically create the Singularity definition file. This file not only allows the user to create a Singularity container, per default it also means the container can be rebuild at a different platform. Thus, one of the requirements of software installation, namely reproducibility, is already achieved. This definition file can also be added in a supplementary information of any publication, allowing other researchers to reproduce the results.

GitHub repository

I have set up a GitHub repository for the collection of these bash-scripts which can be accessed here:
https://github.com/sassy-crick/Singularity-Easybuild



sassy-crick / **Singularity-Easybuild**

master · 1 branch · 0 tags

Sassy and **Sassy** More Singularity definition files added — a5eefbd on 18 Dec 2020 — 69 commits

definitions	More Singularity definition files added	last month
scripts	Centos pip3 added; for Python-2 suffix changed	2 months ago
LICENSE	Initial commit	10 months ago
README.md	README.md updated	7 months ago

README.md

Singularity-Easybuild

Description:

Collection of Singularity definition files and scripts to create them for popular Linux Distributions

The definitions folder contains the successful Singularity Definition files, tested with version 3.5.3, wheras the scripts folder contains the scripts to create the Singularity definition files which are based on EasyBuild.

Requirements:

You will need to have a Linux environment and Singularity installed in it. If you don't have Linux, please use Vagrant to set up a virtual Linux environment.

As the software inside the containers is built using Easybuild, you will need to know the names of the Easybuild Configuration files, e.g. GCC-9.3.0.eb. The latest version of EasyBuild will be automatically installed.

Thus, it is probably best to install the easybuild-easyconfig files like this in a seperate folder:

```
$ git clone https://github.com/easybuilders/easybuild-easyconfigs.git
```

About

Collection of Singularity build files and scripts to create them for popular Linux Distributions

Readme

MIT License

Releases

No releases published

Packages

No packages published

Languages

● Shell 100.0%

Two im portant folders: scripts and definitions.
definitions contains previously created and used Singularity definition files
scripts containds the bash-scripts to create the Singularity definition files

The name of the bash script is build like this:

```
$ container-build-centos7-envmodules-python2.sh
```

In general:

container-build-[Linux-distro][version]-[module-version]-{python2}.sh
So we got:

Linux-distro:	Centos	/	Debian
Version:	7	/	9 or 10
Module:	envmodules	/	envmodules
	lmod	/	lmod
Python2	(y)	/	(y)

The Python2 is the Python version for the operating system, not the version installed by EasyBuild! If not mentioned, the operating system is using Python3.

Requirements:
Singularity needs to be installed.
For Debian: *debootstrap* needs to be installed as well.

Requirements:
Singularity needs to be installed.
For Debian: *debootstrap* needs to be installed as well.

↓

This can be put inside a virtual machine line VirtualBox and then vagrant can be used to run. This does work on Linux for cross-compiling, Mac and Windows.
So we can use a virtual environment to build a Singularity container. Please refer to the vagrant website on how to install and configure vagrant. They also provide a ready-made Singularity container!

How to use the scripts

After downloading the GitHub site, we first create the Singularity definition file like this:

```
$ container-build-centos7-lmod.sh GCC-9.3.0.eb
Do we need a second Easybuild recipe (y/N)?:
```

You will be asked if you want to add another EasyBuild configuration (EC) file.
We say 'No' here for now. This feature is handy if you are writing your own EC files for example, or you modified an existing one. This is more advanced and will be covered in the tutorial on Friday.
After hitting return, we got the Singularity definition file:

```
$ ls
Singularity.GCC-9.3.0-Lmod-centos7
```

Again, the file name contains all the relevant information:
It is a Singularity file, for GCC-9.3.0, using the Lmod modules and the operating system is centos7.
Lets have a look inside this file. This is important if we want to do modifications to it.

The definition file contains 3 different parts:
- The installation of the operating system with all relevant software, including EasyBuild
- The installation of the requested software with EasyBuild
- The post-installation required to run the Singularity container

The installation of the operating system with all relevant software, including EasyBuild

```
Bootstrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/x86_64/
Include: yum
%post
yum --assumeyes update
yum install --quiet --assumeyes epel-release
yum install --quiet --assumeyes python3 setuptools Lmod python3-pip patch make gcc-c++
yum install --quiet --assumeyes bzip2 gzip tar zip unzip xz curl wget file git which perl-Data-Dumper
yum install --quiet --assumeyes perl-Thread-Queue  libibverbs-dev libibverbs-devel rdma-core-devel
yum install --quiet --assumeyes openssl-devel libssl-dev libopenssl-devel openssl
# install EasyBuild using pip3
pip3 install -U pip
pip3 install wheel
pip3 install -U setuptools
pip3 install 'vsc-install<0.11.4' 'vsc-base<2.9.0'
pip3 install easybuild
# create 'easybuild' user (if missing)
id easybuild || useradd easybuild
# create /app software installation prefix + /scratch sandbox directory
if [ ! -d /app ]; then mkdir -p /app; chown easybuild:easybuild -R /app; fi
if [ ! -d /scratch ]; then mkdir -p /scratch; chown easybuild:easybuild -R /scratch; fi
if [ ! -d /home/easybuild ]; then mkdir -p /home/easybuild; chown easybuild:easybuild -R /home/easybuild;fi
# install Lmod RC file
cat > /etc/lmodrc.lua << EOD
scDescriptT = {
  {
    ["dir"]       = "/app/lmodcache",
    ["timestamp"] = "/app/lmodcache/timestamp",
  },
}
EOD
```

The installation of the requested software with EasyBuild

```
# verbose commands, exit on first error
set -ve
set -o noclobber
# We set this so if we need to open the container again, we got the environment set up correctly
cat >> /home/easybuild/.bashrc << 'EOG'
export EASYBUILD_PREFIX=/scratch
export EASYBUILD_TMPDIR=/scratch/tmp
export EASYBUILD_SOURCEPATH=/scratch/sources:/tmp/easybuild/sources
export EASYBUILD_INSTALLPATH=/app
export EASYBUILD_PARALLEL=4
export MODULEPATH=/app/modules/all
alias eb="eb --robot --download-timeout=1000"
EOG
# configure EasyBuild
cat > /home/easybuild/eb-install.sh << 'EOD'
#!/bin/bash
shopt -s expand_aliases
export EASYBUILD_PREFIX=/scratch
export EASYBUILD_TMPDIR=/scratch/tmp
export EASYBUILD_SOURCEPATH=/scratch/sources:/tmp/easybuild/sources
export EASYBUILD_INSTALLPATH=/app
export EASYBUILD_PARALLEL=4
alias eb="eb --robot --download-timeout=1000"
EOD
echo "eb --fetch GCC-9.3.0.eb" >>  /home/easybuild/eb-install.sh
echo "eb GCC-9.3.0.eb" >>  /home/easybuild/eb-install.sh
cat >> /home/easybuild/eb-install.sh << 'EOD'
mkdir -p /app/lmodcache
$LMOD_DIR/update_lmod_system_cache_files -d /app/lmodcache -t /app/lmodcache/timestamp /app/modules/all
EOD
chmod a+x /home/easybuild/eb-install.sh
su -l easybuild -c /home/easybuild/eb-install.sh
# cleanup, everything in /scratch is assumed to be temporary
rm -rf /scratch/*
```

The post-installation required to run the Singularity container

```
%environment
# make sure that 'module' and 'ml' commands are defined
source /etc/profile
# increase threshold time for Lmod to write cache in $HOME (which we don't want to do)
export LMOD_SHORT_TIME=86400
# purge any modules that may be loaded outside container
module --force purge
# avoid picking up modules from outside of container
module unuse $MODULEPATH
# pick up modules installed in /app
module use /app/modules/all
# load module(s) corresponding to installed software
module load GCC/9.3.0

%labels
Author J. Sassmannshausen <jorg.sassmannshausen@kcl.ac.uk>
GCC-9.3.0
```

Building the container

Now that we have the Singularity definition file, we can build the container. That is the only step which requires root or sudo access, and it does take some time, depending on what you are installing and which hardware you are using.
Note: When you are building for a different host, you need to know a bit about the target's host hardware. A container build on the latest release of a given CPU might not work on a say 5 year old CPU.

Building the container is easy:

```
$ sudo singularity build Singularity.GCC-9.3.0-Lmod-centos7.sif Singularity.GCC-9.3.0-Lmod-centos7
```

This will build the container as a Singularity Image File, hence the suffix .sif

It is also possible to build the container as a sandbox. This means, instead of the container format, we basically have a directory structure. The advantage here is that you can make changes inside this directory if required. We will do this in the tutorial on Friday.

Running the container

Once we got the container build, we can use it.
In our example, we only build a container containing GCC version 9.3.0. Although that appears to be bit pointless to build such a container, there is some advantage here. As mentions before, we can open up the container again and thus we can add software to it as required. This allows a user to build customised pipelines for example, which then can be used on other systems, like a cloud base one. Remember the hardware limitations I mentioned before!

So in our case, in order to run the container we would simply do, as user:

```
$ singularity run GCC-9.3.0-Lmod-centos7.sif gcc --version
gcc (GCC) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

So now we got a CentOS7 container running GCC version 9.3.0 on a Debian machine.

Summary

- Singularity containers are easy to build using the provided bash scripts in connection with the name of the EasyBuild configuration file.
- The only requirements are a Linux environment with Singularity installed, and sudo access. This can conveniently achieved by using VirtualBox and vagrant.
- No in-depth knowledge of either Singularity or EasyBuild is required.
- The Singularity definition file can be published, if required the container can be published as well. This allows researcher to make it easier for others to reproduce their work and thus verify it.
- As an outlook, as this build process can be automated, it is possible for a user simply to provide the Singularity definition file to a scheduler like SLURM, get it build on the relevant cluster, and do the required calculations.

Acknowledgement

I am greatful for Kenneth for providing EasyBuild and for helpful discussion. The same goes for the nice people on the EasyBuild Slack channel.

Furthermore, I am greatful for the help I received over the years in the #linuxhelp IRC channel on Chatjunkies. In particularly my friend 'lowkey' there who helped me a lot over the years to improve my bash scripting.

Likewise, I would like to thank my colleagues from the Rosalind team at King's College London, who warmly welcomed me when I joined, for their camradie and their help to settle in, specially as I arrived just before the lockdown.

We are hiring:

https://jobs.kcl.ac.uk/gb/en/job/013465/Full-Stack-Engineer

Tutorial

29. January 2021 14:00 – 14:30 UTC (Virtual)
Please register at:

https://easybuildcontainerstutorial.splashthat.com/

Thanks for Christian Kniep and his team from Amazon for organizing this!

In the tutorial we
- build a simple Singularity container to understand the concept of the scripts
- open up a provided container to install other software in it.

The last bit can be done with EasyBuild, or for more adventorus users, some own software.
There is a Slack-Channel (#eum21-containers-workshop) on the EasyBuild Slack for additional help. For details please see the conference website.