Containerization for scientific reproducibility

WOSSL Workshop

27 July 2020

Stefano Alberto Russo

stefano.russo@inaf.it

What are containers?

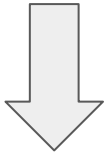
“Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another.”

cio.com

What are containers?

“Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another.”

cio.com



A.K.A. the dependency hell problem

The “dependency hell” problem (take A)

Mike wants to use a new software.

Mike cannot find a precompiled version that works with his OS and/or libraries.

Mike asks/Google for help and gets some basic instructions - like “compile it”.

Mike starts downloading all the development environment, and soon realizes that he needs to upgrade (or downgrade!) some parts of his main Operating Systems.

During this process, something goes wrong.

Mikes spends an afternoon fixing his own OS, and all the next day in trying to compile the software. Which at the end turns out not to do what he wanted.

The “dependency hell” problem (take B)

Mike wants to use a new software.

Mike finds a precompiled versions, he download and install it.

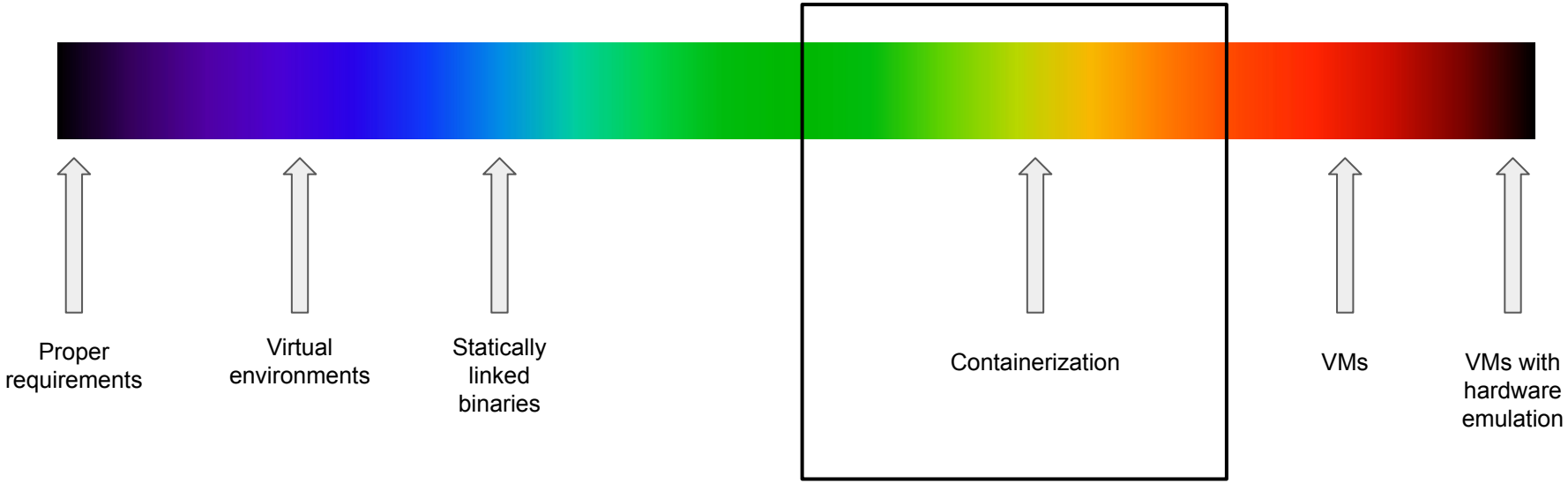
Mike runs the software and get the result “43”.

After a year Mike runs again that software and get the result “42”.

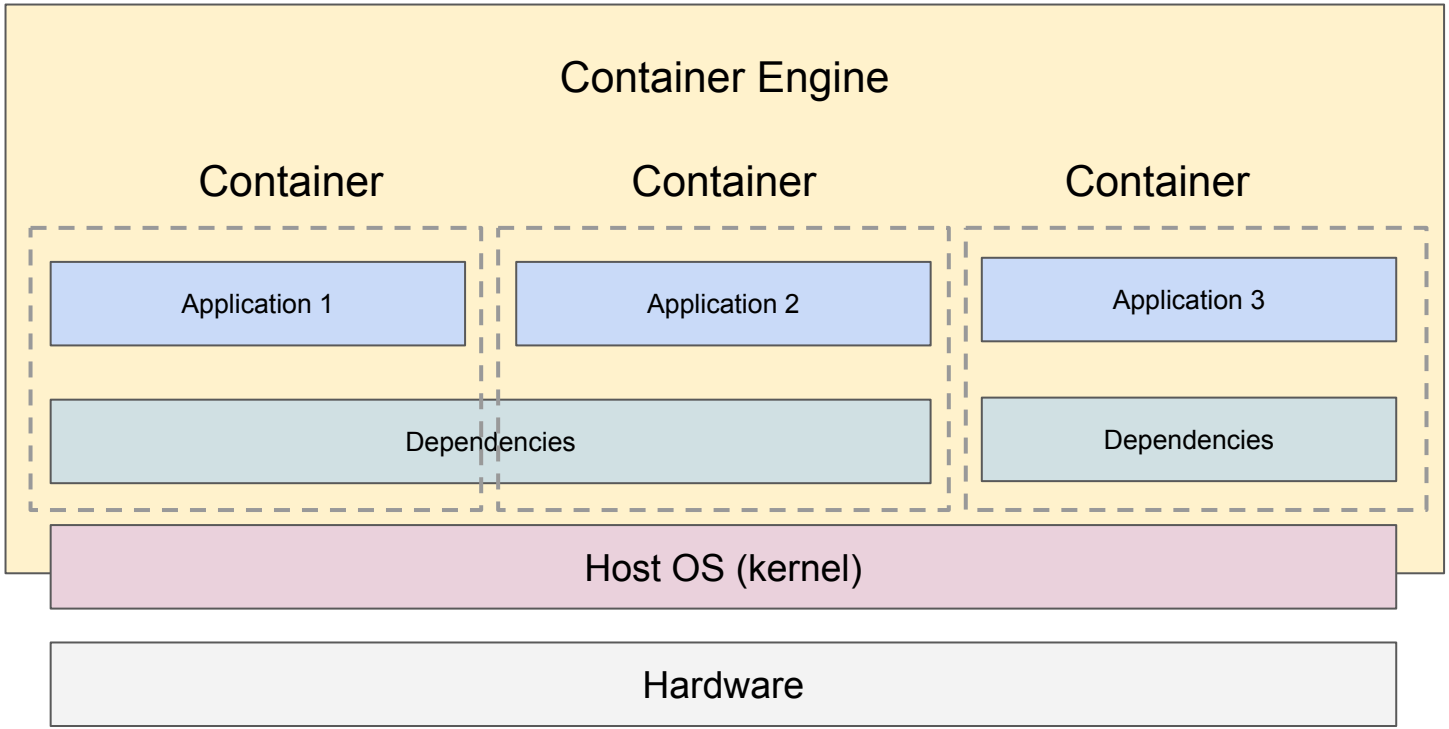
Mike takes a deep dive into the problem and finds out that a library used by the software was called in the wrong way due to an API change in the version he had, meaning that the “43” was wrong.

The “dependency hell” problem: solutions spectrum

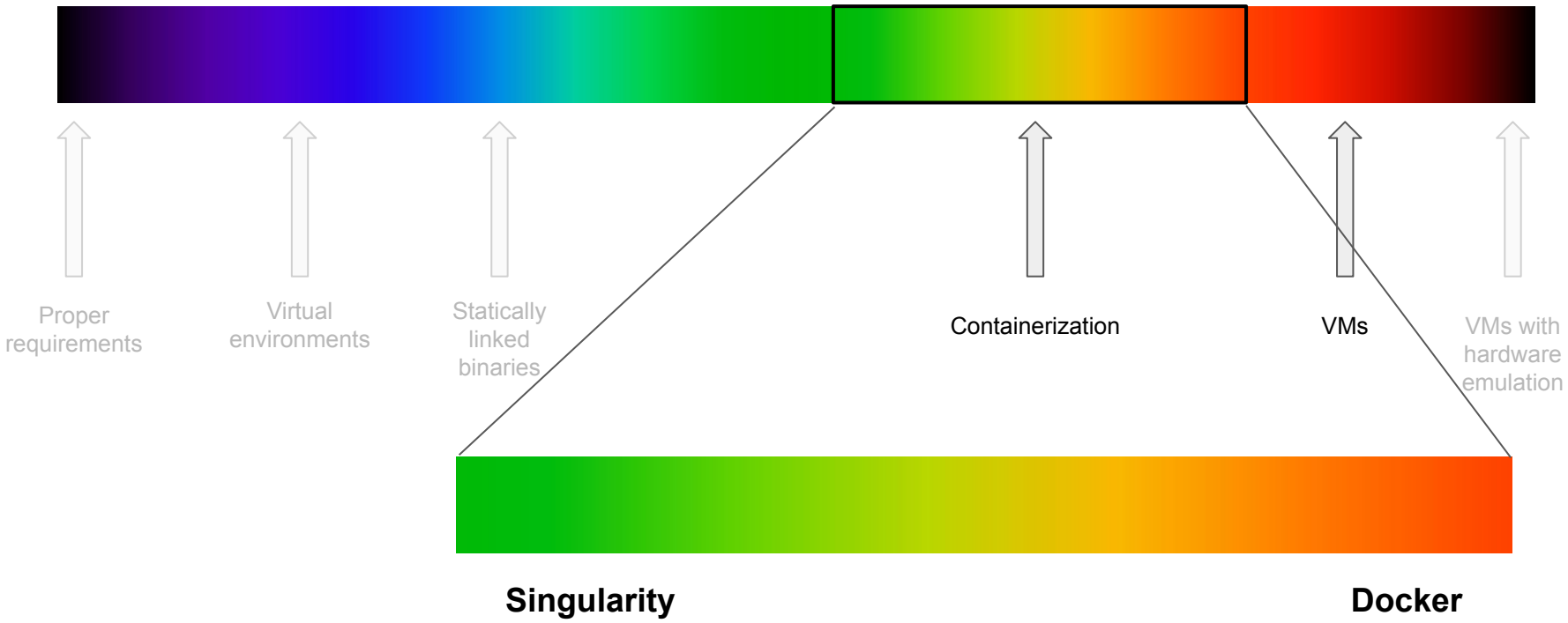
The “dependency hell” problem: solutions spectrum



Container engine



The “dependency hell” problem: solutions spectrum



Singularity vs Docker

Singularity	Docker
Scientific computing	IT industry standard
Running container are seen as processes	Running containers are seen as (micro)services
Build as root, run as user	Need near-root access or proper orchestrators
Limited or no support for networking	Extensive support for networking

Singularity vs Docker

Singularity	Docker
Filesystem: only partially isolated,directories as $HOME, /tmp, /proc, /sys, and /dev are all binded by default.	**Filesystem:** completely isolated by default, volume or folder binds must be explicitly set
Environment: from the host	**Environment:** from scratch
Network: from the host	**Network:** dedicated subnet

More similar to an environment

More similar to a virtual machine

We will focus on Docker in the following

The “GitHub” for Docker Container Images

Gcc on Docker Hub

library/gcc – Docker Hub

https://hub.docker.com/_/gcc/

Search | Explore | Help | Sign up | Sign in

OFFICIAL REPOSITORY

gcc

Last pushed: 9 days ago

Repo Info | Tags

Short Description

The GNU Compiler Collection is a compiling system that supports several languages.

Docker Pull Command

```
docker pull gcc
```

Full Description

Supported tags and respective `Dockerfile` links

- `4.9.4`, `4.9`, `4` (4.9/Dockerfile)
- `5.4.0`, `5.4`, `5` (5/Dockerfile)
- `6.4.0`, `6.4`, `6` (6/Dockerfile)
- `7.2.0`, `7.2`, `7`, `latest` (7/Dockerfile)

Gcc on Docker Hub (pull command)

```
$ docker pull gcc:5.4
```

Gcc on Docker Hub (downloading)

```
$ docker pull gcc:5.4
5.4: Pulling from library/gcc
aa18ad1a0d33: Extracting [=================================>                ] 33.98 MB/52.6 MB
15a33158a136: Download complete
f67323742a64: Download complete
c4b45e832c38: Downloading [===================>                              ] 51.59 MB/134.7 MB
e5d4afe2cf59: Download complete
4c0020714917: Downloading [=======>                                           ] 30.59 MB/200.4 MB
b33e8e4a2db2: Download complete
c8dae0da33c9: Waiting
```

- You are downloading a minimalistic Linux distribution on top which has been installed **gcc** (v5.4).
- Thanks to Docker's incremental file system, another container based on the same Linux minimalistic distribution or **ggc** itself *will not* require to download/store it again.

Gcc on Docker Hub (downloaded)

```
$ docker pull gcc:5.4
5.4: Pulling from library/gcc
aa18ad1a0d33: Pull complete
15a33158a136: Pull complete
f67323742a64: Pull complete
c4b45e832c38: Pull complete
e5d4afe2cf59: Pull complete
4c0020714917: Pull complete
b33e8e4a2db2: Pull complete
c8dae0da33c9: Pull complete
Digest: sha256:e6ef7f0295b9d915f8521de360e30803bf8561cfb9cea8e320aa66761be8ec42
Status: Downloaded newer image for gcc:5.4
```

Terminology warning:

- image: a “file” from which you can run a container
- container: an “entity” run from an image

Run Gcc (5.4) with Docker

```
$ docker run gcc:5.4 gcc -v
```

Run Gcc (5.4) with Docker

```
$ docker run gcc:5.4 gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/local/libexec/gcc/x86_64-linux-gnu/5.4.0/lto-wrapper
Target: x86_64-linux-gnu
Configured with: /usr/src/gcc/configure --build=x86_64-linux-gnu --disable-multilib
--enable-languages=c,c++,fortran,go
Thread model: posix
gcc version 5.4.0 (GCC)
$
```

Entering in the Gcc (5.4) container

Execute a (bash) shell in the container

```
$ docker run -t -i gcc:5.4 bash
root@b9c1414bab3d:/#
```

You are root (and the prompt changes)

List the root directories

```
root@b9c1414bab3d:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt     proc  root  run  sbin  srv
sys  tmp  usr  var
```

Entering in the Gcc (5.4) container

List running processes

```
root@b9c1414bab3d:/# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  1 13:54 pts/0    00:00:00 bash
root         8     1  0 13:54 pts/0    00:00:00 ps -ef
```

Get the container IP address

```
root@b9c1414bab3d:/# ip addr show dev eth0
    [...]
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
    [...]
```

Entering in the Gcc (5.4) container

List running Docker containers (on another shell of your computer)

```
$ docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED        STATUS      PORTS        NAMES
b9c1414bab3d   gcc:5.4  "bash"   3 seconds ago  Up 1 second           friendly_goodall
```

Exit the shell, and therefore the container

```
root@b9c1414bab3d:/# exit
$
```

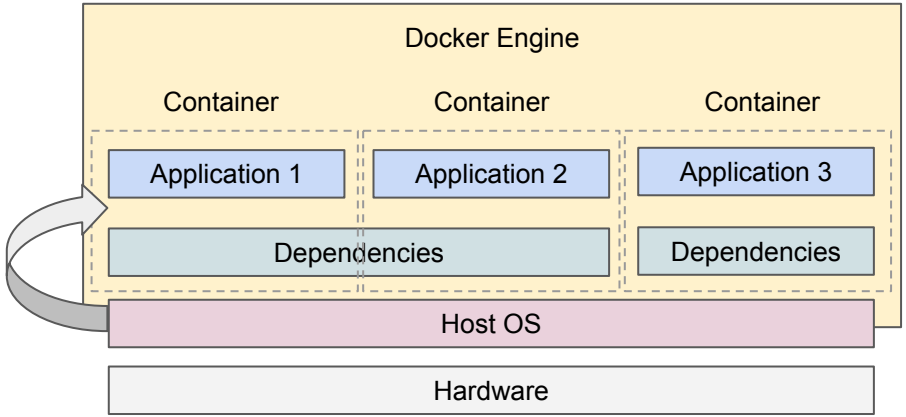
When you exit a container, you lose every change to the container File System

Share files with a Docker container

Docker containers are isolated from your main (host) Operating System

- But you can make some folders visible from the containers as bindings (think about an usb pendrive)
- Just append "`-v your_os_folder:path_inside_the_container`" to docker command

We are creating a bridge



Compile your code with Gcc (5.4)

Our test.c code:

```
#include<stdio.h>

int main()
{
    printf("I run a very complex simulation and the result is 42\n");
}
```

Compile your code with Gcc (5.4)

```
$ docker run -v$PWD:/data gcc:5.4 gcc -o /data/Test/test.bin --verbose /data/Test/test.c
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/local/libexec/gcc/x86_64-linux-gnu/5.4.0/lto-wrapper
Target: x86_64-linux-gnu
Configured with: /usr/src/gcc/configure --build=x86_64-linux-gnu --disable-multilib
--enable-languages=c,c++,fortran,go
Thread model: posix
gcc version 5.4.0 (GCC)
COLLECT_GCC_OPTIONS='-o' '/data/Test/test.bin' '-v' '-mtune=generic' '-march=x86-64
[...]
$
```

Run your code compiled with Gcc (5.4)

On your computer → no!

Inside the container → yes!

```
$ docker run -v$PWD:/data gcc:5.4 /data/Test/test.bin
I just ran a very complex simulation and the result is 42
```

The Dockerfile

library/gcc – Docker Hub

https://hub.docker.com/_/gcc/

Search | Explore | Help | Sign up | Sign in

OFFICIAL REPOSITORY

gcc

Last pushed: 9 days ago

Repo Info | Tags

Short Description

The GNU Compiler Collection is a compiling system that supports several languages.

Docker Pull Command

```
docker pull gcc
```

Full Description

Supported tags and respective `Dockerfile` links

- `4.9.4`, `4.9`, `4` (4.9/Dockerfile)
- `5.4.0`, `5.4`, `5` (5/Dockerfile)
- `6.4.0`, `6.4`, `6` (6/Dockerfile)
- `7.2.0`, `7.2`, `7`, `latest` (7/Dockerfile)

The Dockerfile

- The *Dockerfile* is what defines a Docker Container. Think about it as its source code.
- When you build it, it generates a *Docker Image*. When you run a Docker Image, this “becomes” a *Docker Container*, as mentioned before.

```
FROM <base image>

RUN <a setup command>

COPY <source file/folder on your OS> <dest file/folder in the container>

RUN <another setup command>
```

A container for our code

Let’s see how to include and compile your test code directly from a Dockerfile

```
FROM gcc:5.4

# Add the test code
COPY test.c /opt

# Compile the test code
RUN gcc -v -o /opt/test.bin /opt/test.c
```

A container for our code

Let’s now build it with“test.c” and the Dockerfile files in a folder named “Test”:

```
$ docker build Test -t testcontainer
Sending build context to Docker daemon  10.24kB
Step 1/3 : FROM gcc:5.4
 ---> b87db7824271
Step 2/3 : COPY test.c /opt
 ---> f5478f7830ee
Step 3/3 : RUN gcc -v -o /opt/test.bin /opt/test.c
 ---> Running in c839379f1fbe
Using built-in specs.
COLLECT_GCC=gcc
[...]
Removing intermediate container c839379f1fbe
 ---> 2f0c6f89fdc0
Successfully built 2f0c6f89fdc0
Successfully tagged testcontainer:latest
```

A container for our code

..and we can now run it:

```
$ docker run testcontainer /opt/test.bin
I just ran a very complex simulation and the result is 42
```

More about how to build your own containers in the breakout demo session

The last bit: an hard truth

Containers alone do not guarantee reproducibility

...they are just a tool!

Main dos and don'ts (1)

Do not build a container without explicitly freezing all dependencies

..because it will cause to get always the “latest” version if the container has to be rebuilt, and this includes Git repositories as well:

```
RUN git clone https://github.com/myuser/myrepo.git
```

```
RUN git clone https://github.com/myuser/myrepo.git && git checkout 653e7g2
```

```
RUN pip install pandas
```

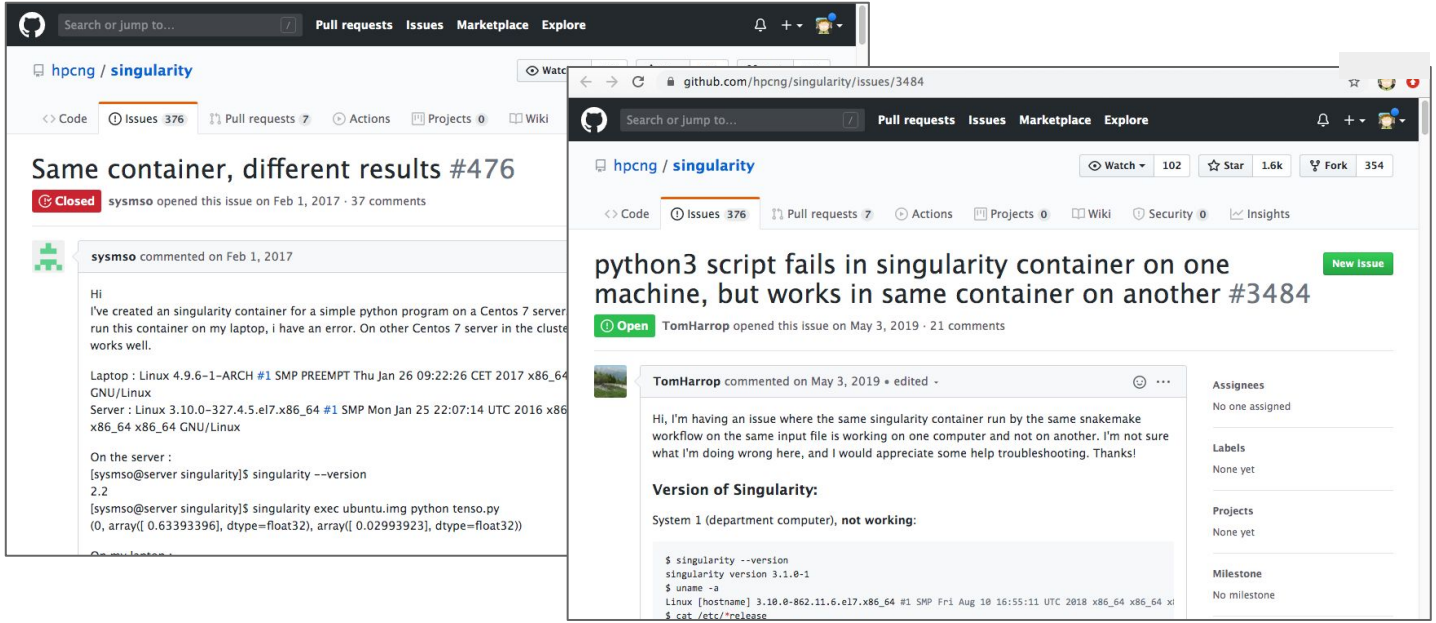
```
RUN pip install pandas==0.21.0
```

*Note: unfortunately, this is the default behaviour in Singularity

Main dos and don'ts (2)

Do not rely on the external environment (variables, files..) at runtime*

..because your containers will be error prone to different setups:



Main dos and don'ts (3)

Do not rely on unofficial or personal online resources

..because the risk of having them taken down is high, and your container will not rebuild.
If you have to, include these resources in your container source folder or on secure mirrors.

Search or jump to... / Pull requests Issues Marketplace Explore

zeek / zeek-docker

Watch 23 Star 48 Fork 21

Code Issues 3 Pull requests Actions Projects Wiki Security Insights

Unable to build 3.0.0 docker image. (404 Not Found) #13

New issue

Closed kahou82 opened this issue on Apr 1 · 0 comments

kahou82 commented on Apr 1 Contributor

Trying to do make build-stamp.3.0.0 but see the following error

```
URL is https://www.bro.org/downloads/zeek-3.0.0.tar.gz
--2020-03-31 22:33:05--  https://www.bro.org/downloads/zeek-3.0.0.tar.gz
Resolving www.bro.org (www.bro.org)... 192.150.187.43
Connecting to www.bro.org (www.bro.org)|192.150.187.43|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.zeek.org/downloads/zeek-3.0.0.tar.gz [following]
--2020-03-31 22:33:05--  https://www.zeek.org/downloads/zeek-3.0.0.tar.gz
Resolving www.zeek.org (www.zeek.org)... 192.0.78.150, 192.0.78.212
Connecting to www.zeek.org (www.zeek.org)|192.0.78.150|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://zeek.org/downloads/zeek-3.0.0.tar.gz [following]
--2020-03-31 22:33:06--  https://zeek.org/downloads/zeek-3.0.0.tar.gz
Resolving zeek.org (zeek.org)... 192.0.78.150, 192.0.78.212
Connecting to zeek.org (zeek.org)|192.0.78.150|:443... connected.
HTTP request sent, awaiting response... 404 Not Found
```

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked pull requests
Successfully merging a pull request

*Note: unfortunately, this is the default behaviour in Singularity

Main dos and don'ts (4)

Avoid use a container without a specific version tag, even when extending it

..because the container might get updated over time!

```
$ docker run gcc
```

```
$ docker run gcc:v5.4
```

```
FROM gcc (in the Dockerfile)
```

```
FROM gcc:v5.4 (in the Dockerfile)
```

Main dos and don'ts (5)

.. in general, always ask yourself:

what could change if I run this container in another environment,
or if I rebuild it in ten years time?

Hope it helps :)

Questions?

Stefano Alberto Russo

stefano.russo@inaf.it