Packaging software in portable manner with Singularity

Olivier Mattelaer
UCLouvain
CP3 & CISM

What do I want to cover

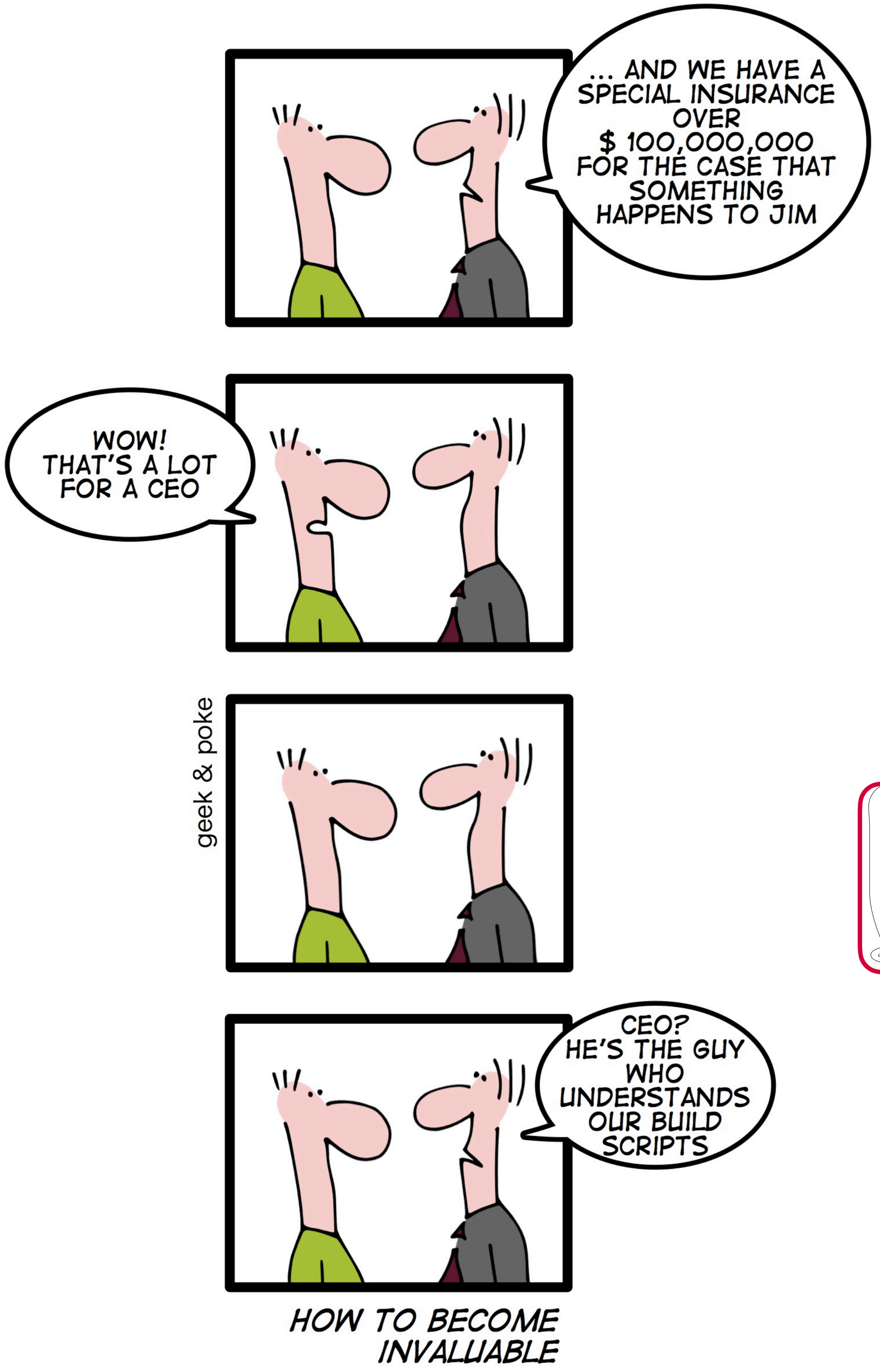
- What is a container
  - ➡ Why it can be interesting for you?

- Singularity: Container for HPC
  - ➡ Features
  - ➡ Limitations

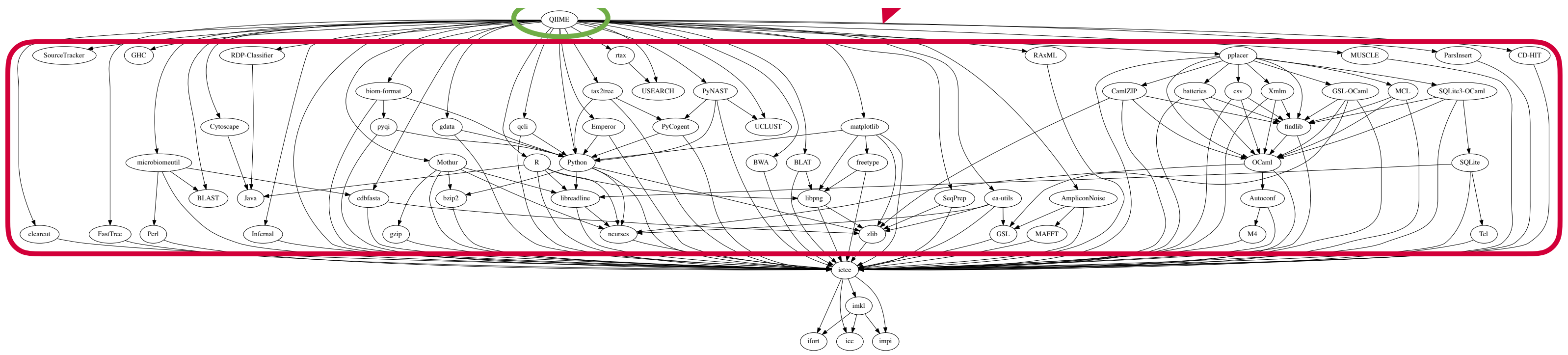
- Tutorial
  - ➡ Show that this is easy to do

- HPC
  - ➡ Details on how to use our setup

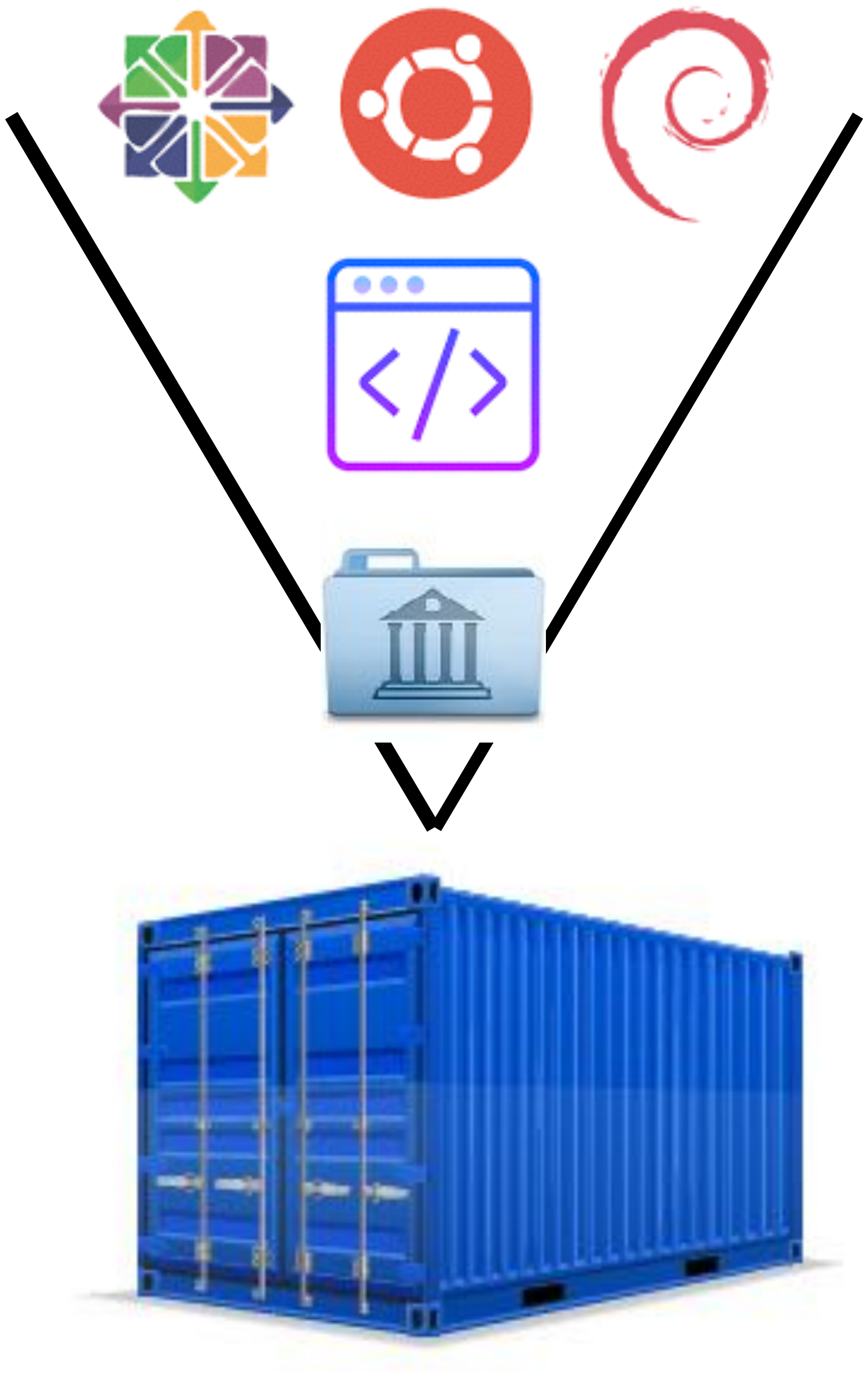
Installing Software


... AND WE HAVE A SPECIAL INSURANCE OVER $ 100,000,000 FOR THE CASE THAT SOMETHING HAPPENS TO JIM
WOW! THAT'S A LOT FOR A CEO
geek & poke
CEO? HE'S THE GUY WHO UNDERSTANDS OUR BUILD SCRIPTS
HOW TO BECOME INVALUABLE

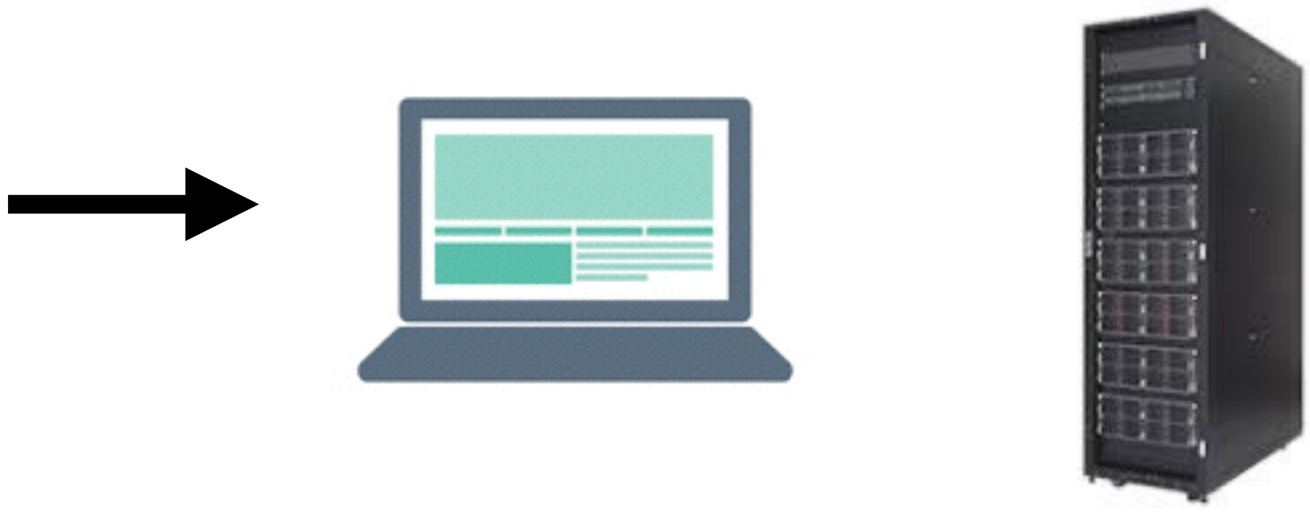
- Tedious/complicated
    - ➡ For user
    - ➡ For sys-admin
- Dependencies Hell



Container Solution



- machine agnostic code
  - ➡ A (small) OS
  - ➡ Your code (executable)
  - ➡ All the dependencies (libraries)
- That can run "everywhere"



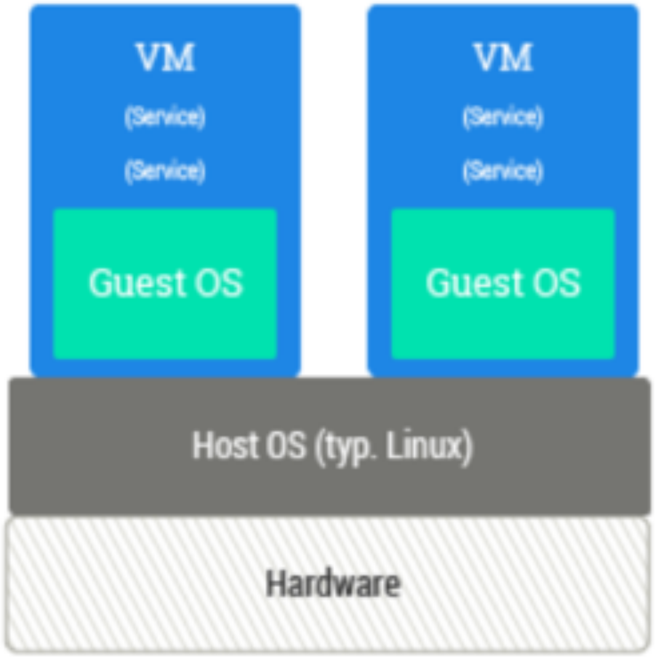
What for?

- ➡ **reproducibility** on any (unix) machine
  - ✦ Nice to send to a collaborator !
- ➡ **deployment** (cloud/laptop/hpc/…)
  - ✦ Nice to distribute the workload
- ➡ With a **paper**
  - ✦ Nice for being able to reproduce results
  - ✦ Nice for other scientists

VM versus container

VM

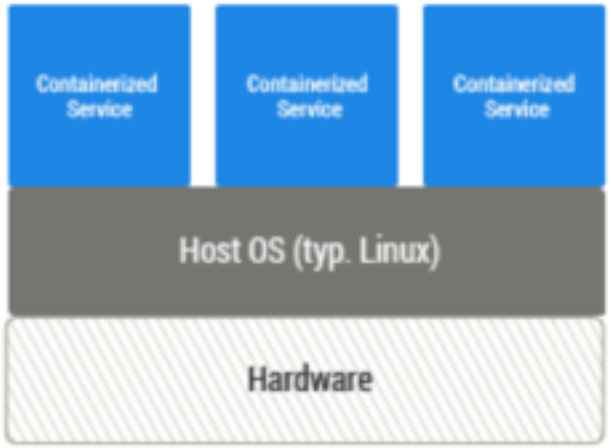
- virtualize the kernel
  - ➡ Hardware virtualisation



- ➡ Flexible
- ➡ slow/resource hungry

container

- Reuse the kernel
  - ➡ Software virtualisation



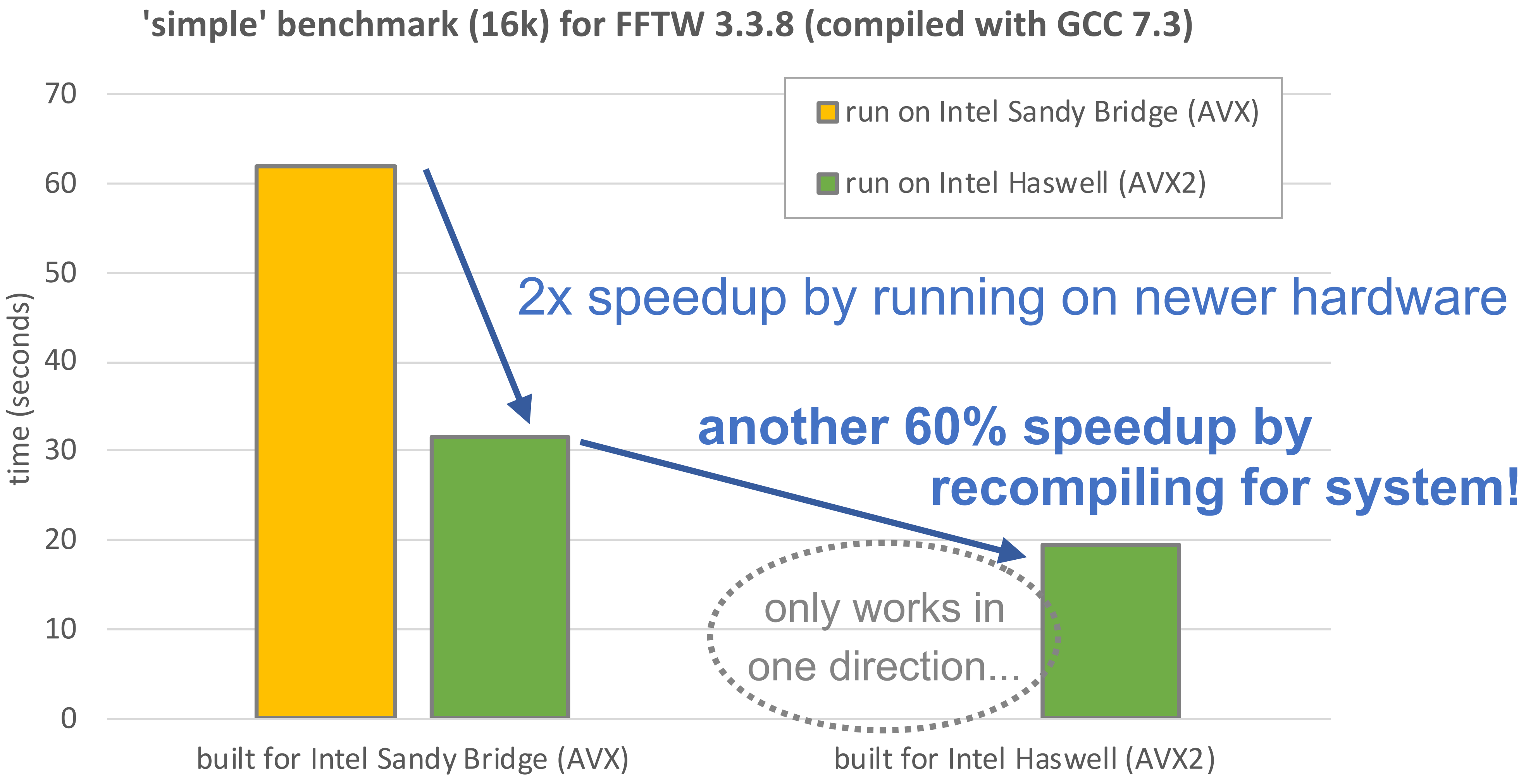
- ➡ Not multi os
- ➡ fast/light
  - ➡ OK for single app
  - ➡ Good for HPC

Containers History

- Are an old idea
  - ➡ Chroot (1979), FreeBSD jails (2000), Solaris containers (2004), LXC (2008)
- Docker (2013)
  - ➡ For/with cloud computing
- Buzz for HPC containers starts ~ 2015
  - ➡ Docker tries to convince HPC structure and failed
- Singularity (2016)
  - ➡ HPC focus

Performance

- They claim “native” performance
  - ➡ understand “small” overheard (couple of percent)
  - ➡ No cpu optimisation


'simple' benchmark (16k) for FFTW 3.3.8 (compiled with GCC 7.3)

2x speedup by running on newer hardware

another 60% speedup by recompiling for system!

only works in one direction...

(FFTW 3.3.8 installed in Singularity container)

Plot taken from Kenneth Hoste

Hardware Optimisation

CPU

Need generic compilation

GPU

Special handling to handle GPU
Specific library at run time

MPI

No special handling
But actually needed

No portability here!

https://sylabs.io/guides/3.6/user-guide/mpi.html?highlight=mpi

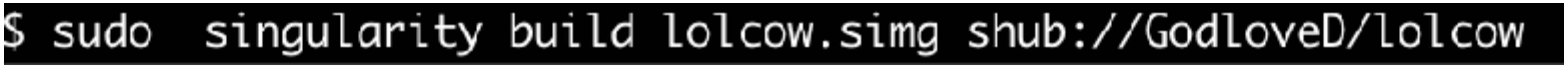
Install Singularity

- On linux:
  - ➡ https://sylabs.io/guides/3.9/user-guide/
- On Windows or Mac (VM):
  - ➡ https://sylabs.io/guides/3.9/admin-guide/installation.html#installation-on-windows-or-mac
- On cluster
  - ➡ Use remote build
  - ➡ You are not allowed to be root on the cluster
  - ➡ (Rootless mode is not activated in CECI cluster)

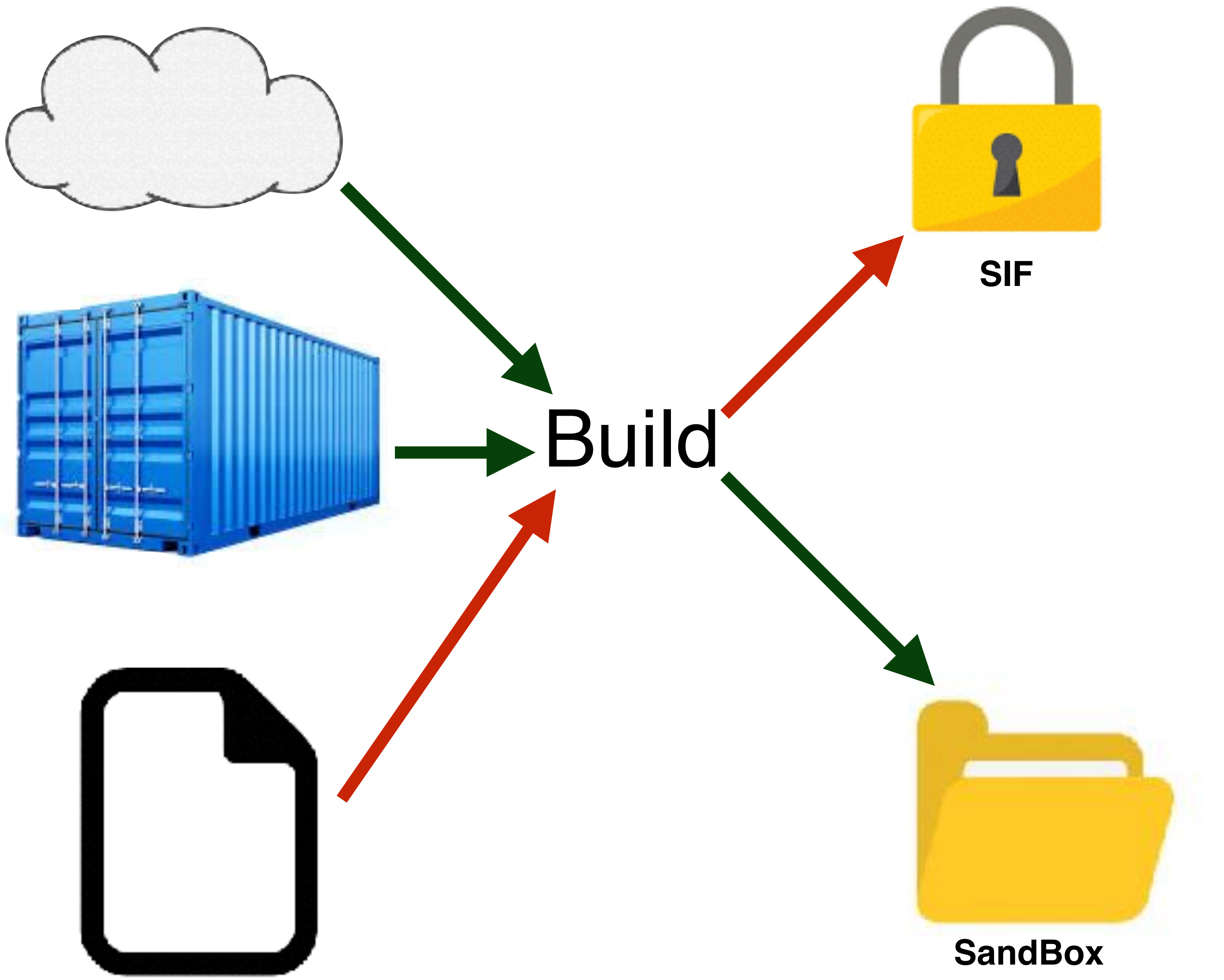
Workflow

- Build
- Test
- Share
- Run

Building an image



```
$ sudo  singularity build lolcow.simg shub://GodloveD/lolcow
```



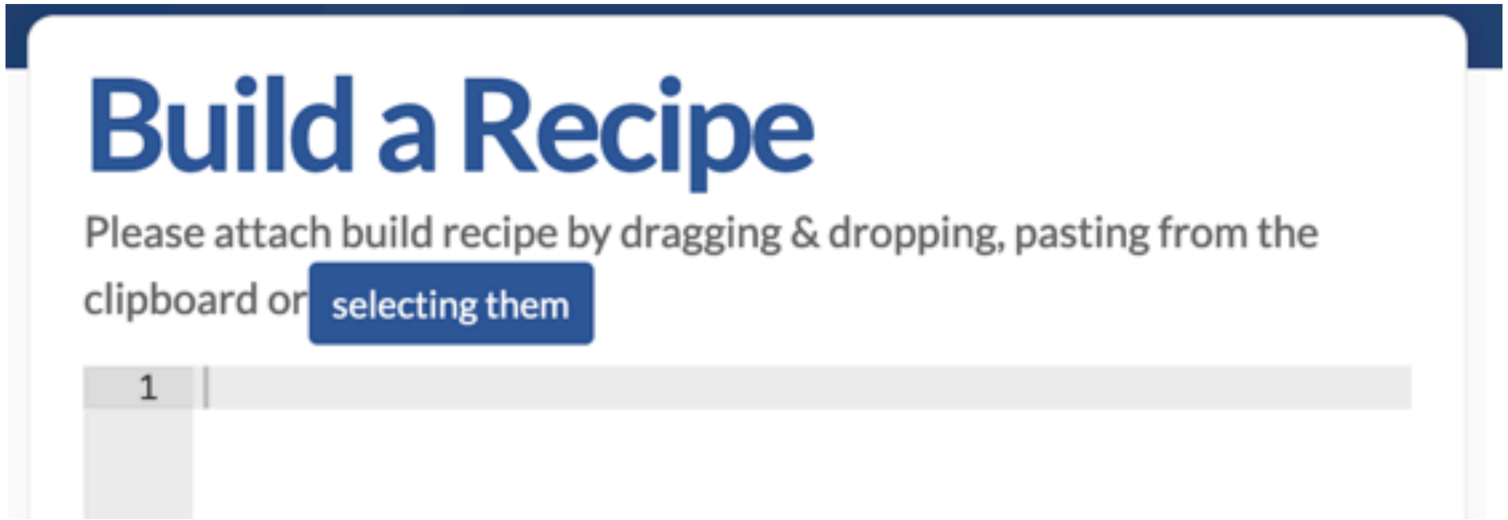
- Singularity Integrity File
  - ➡ Read-only (signed)
  - ➡ default
- Sandbox --sandbox
  - ➡ Full directory
  - ➡ Writable
  - ➡ Can break reproducibility

→ Root privileges is always required

Remote build

- https://cloud.sylabs.io/home
  - ➡ Allow remote build (No need to be root on your machine)
  - ➡ You can do everything from the CECI clusters
    - ✦ No file transfer

Online

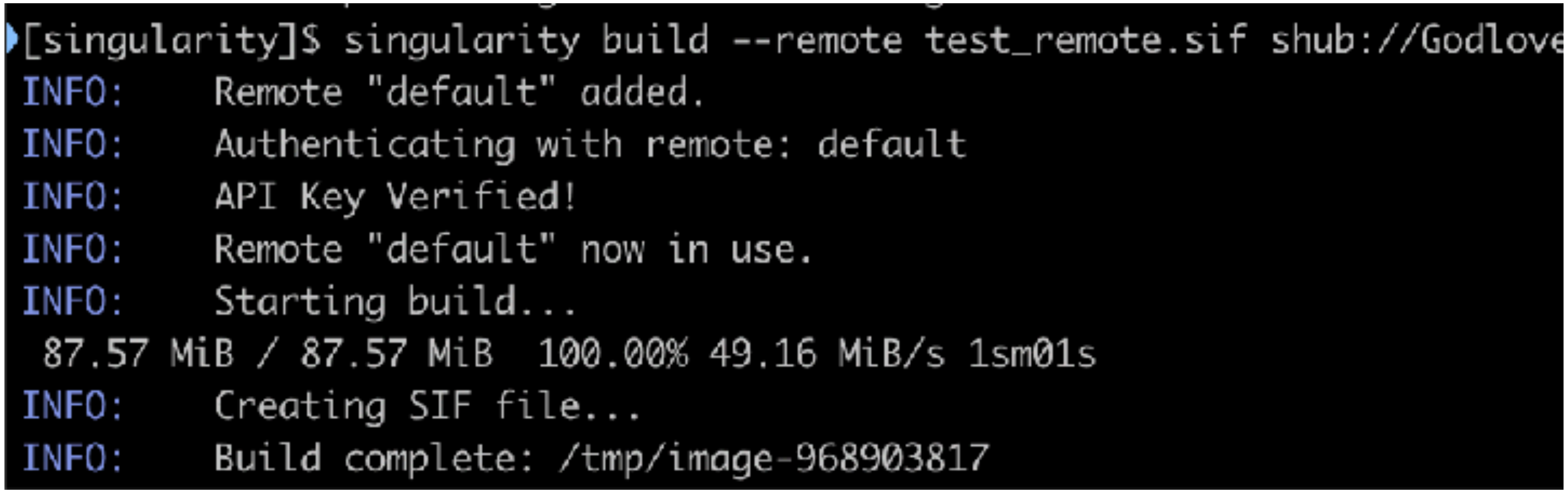


Build a Recipe

Please attach build recipe by dragging & dropping, pasting from the clipboard or

```
1 |
```

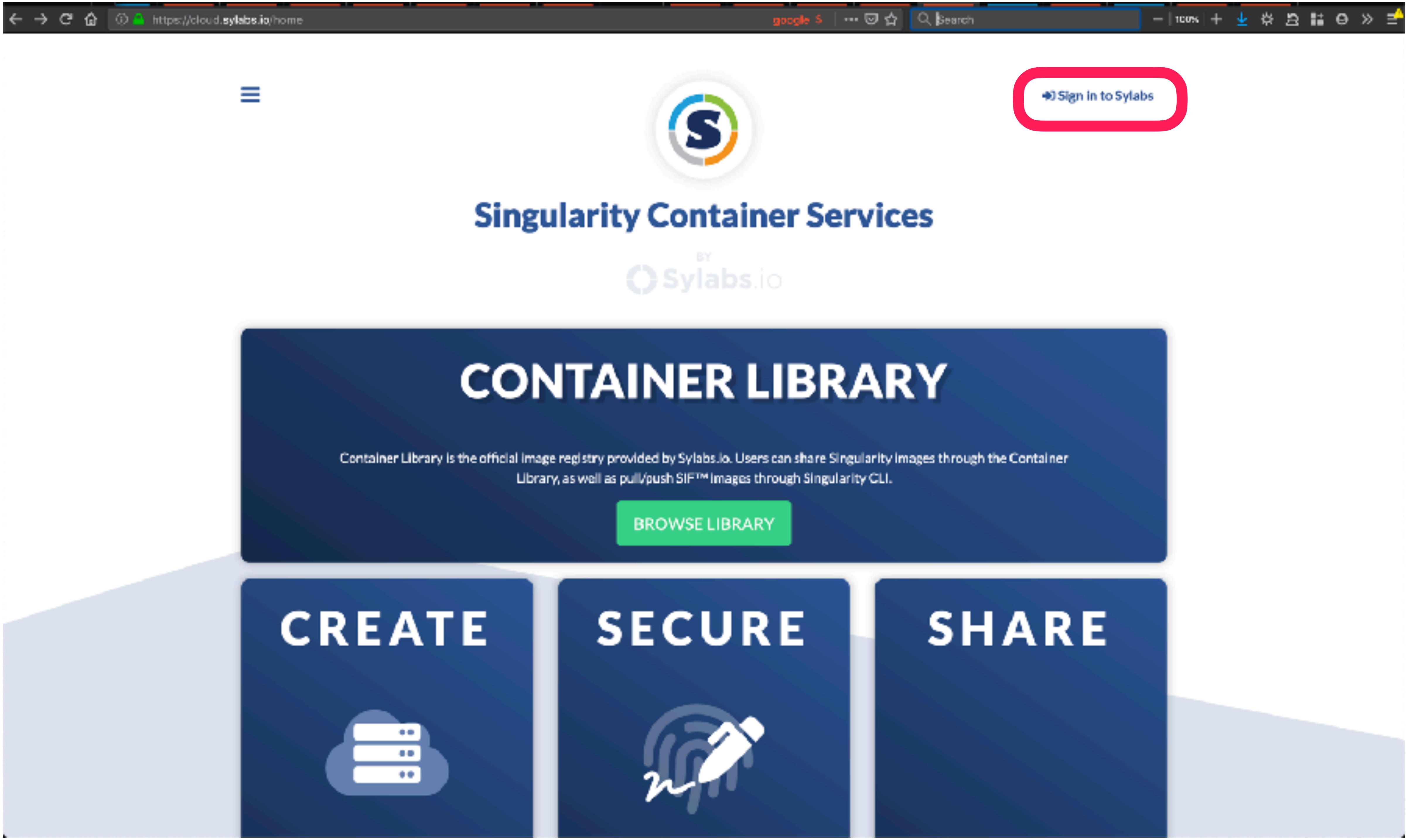
From laptop/cluster



```
[singularity]$ singularity build --remote test_remote.sif shub://Godlove
INFO:    Remote "default" added.
INFO:    Authenticating with remote: default
INFO:    API Key Verified!
INFO:    Remote "default" now in use.
INFO:    Starting build...
 87.57 MiB / 87.57 MiB  100.00% 49.16 MiB/s 1sm01s
INFO:    Creating SIF file...
INFO:    Build complete: /tmp/image-968903817
```

Remote build : Setup

- https://cloud.sylabs.io/home
  - ➡ Create an account



https://cloud.sylabs.io/home

Sign in to Sylabs

Singularity Container Services

BY Sylabs.io

CONTAINER LIBRARY

Container Library is the official image registry provided by Sylabs.io. Users can share Singularity images through the Container Library, as well as pull/push SIF™ images through Singularity CLI.

BROWSE LIBRARY

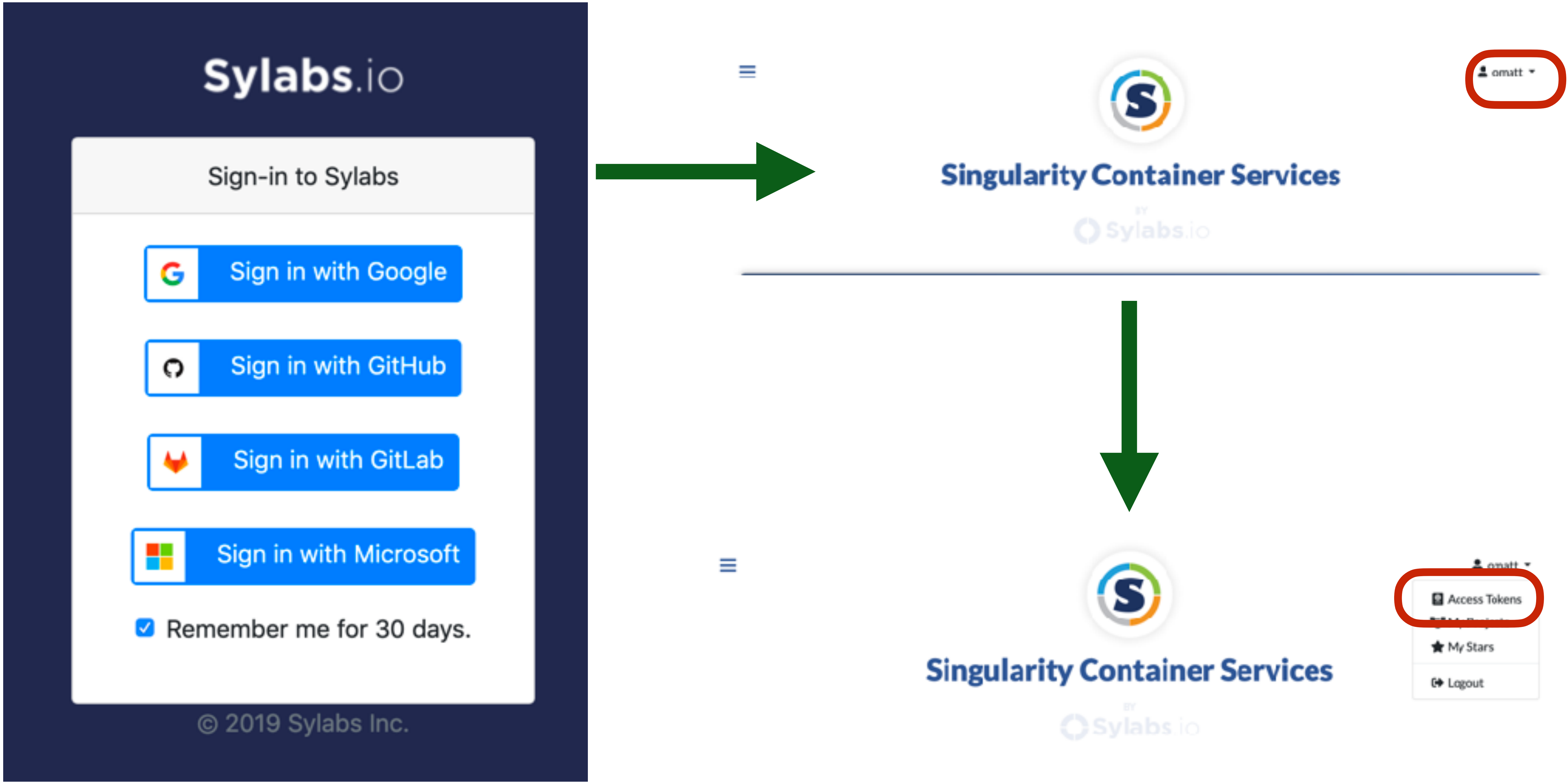
CREATE

SECURE

SHARE

Remote build : Setup

- https://cloud.sylabs.io/home
  - ➡ Create an account



Sylabs.io

Sign-in to Sylabs

Sign in with Google

Sign in with GitHub

Sign in with GitLab

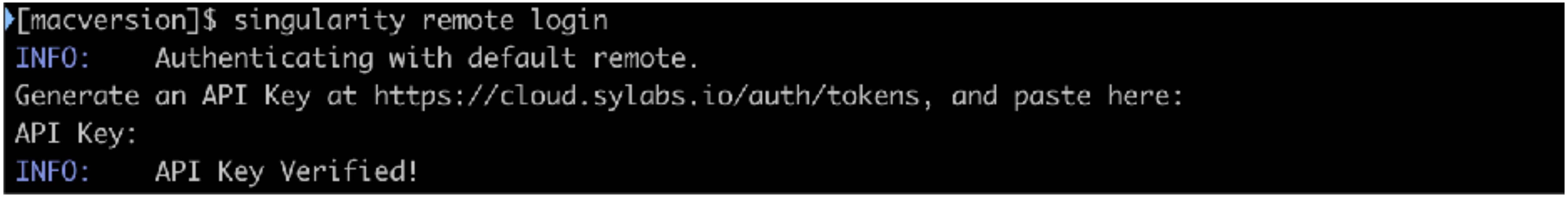
Sign in with Microsoft

☑ Remember me for 30 days.

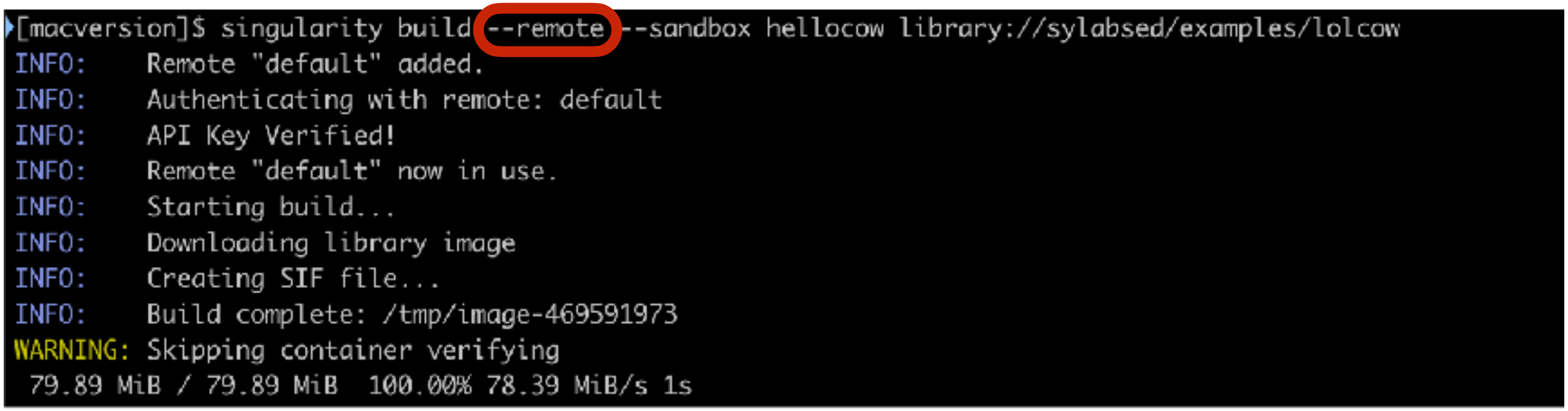
© 2019 Sylabs Inc.

Remote build : Setup

- https://cloud.sylabs.io/home
  - ➡ Create an account
  - ➡ Run on the cluster/your machine:
    - ✦ singularity remote login



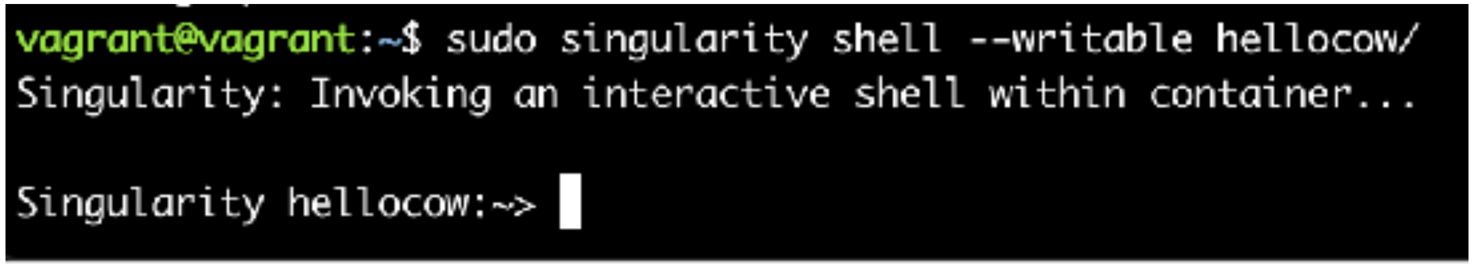
```
[macversion]$ singularity remote login
INFO:    Authenticating with default remote.
Generate an API Key at https://cloud.sylabs.io/auth/tokens, and paste here:
API Key:
INFO:    API Key Verified!
```



```
[macversion]$ singularity build --remote --sandbox hellocow library://sylabsed/examples/lolcow
INFO:    Remote "default" added.
INFO:    Authenticating with remote: default
INFO:    API Key Verified!
INFO:    Remote "default" now in use.
INFO:    Starting build...
INFO:    Downloading library image
INFO:    Creating SIF file...
INFO:    Build complete: /tmp/image-469591973
WARNING: Skipping container verifying
 79.89 MiB / 79.89 MiB  100.00% 78.39 MiB/s 1s
```

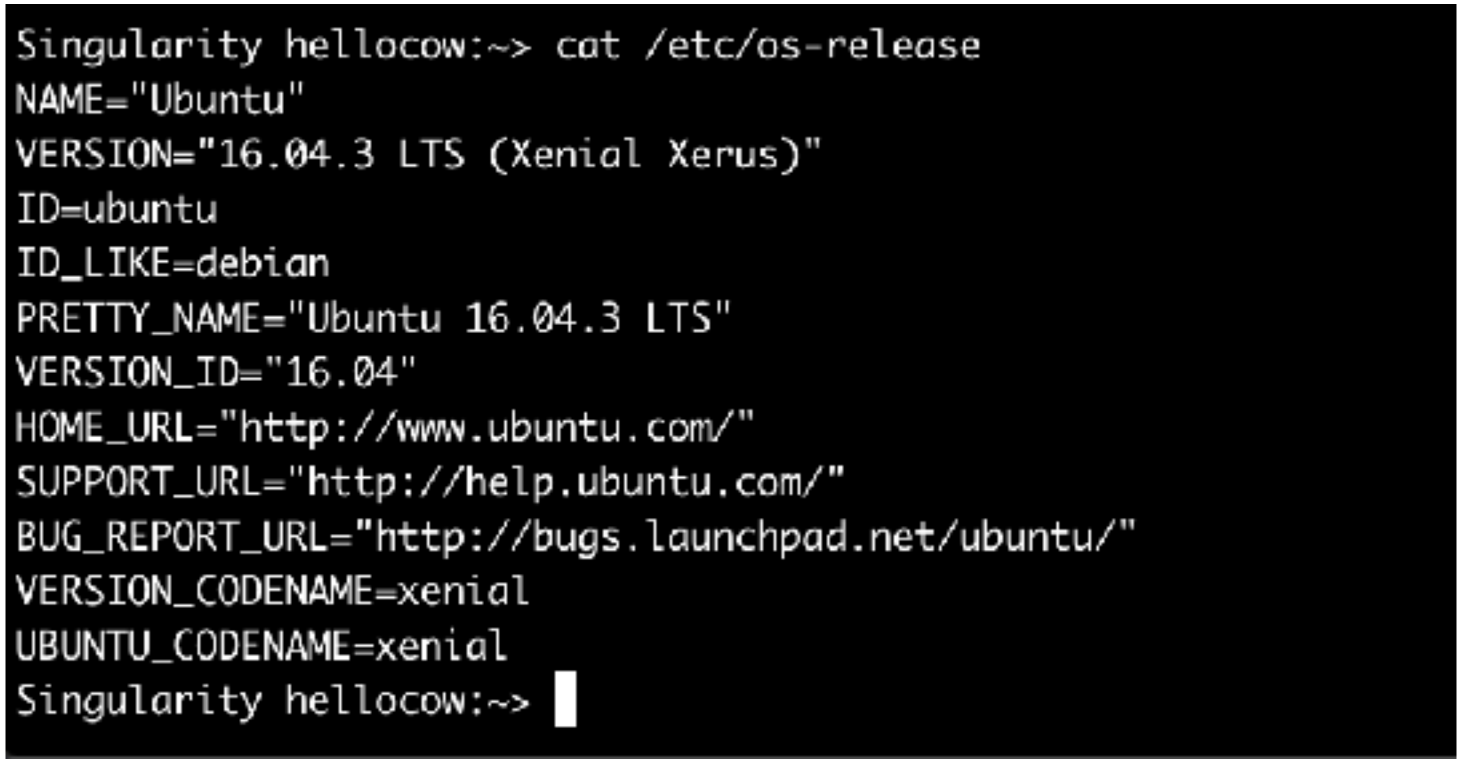
➡ export SINGULARITY_REMOTE=True

Testing and Modifying image (sandbox)



```
vagrant@vagrant:~$ sudo singularity shell --writable hellocow/
Singularity: Invoking an interactive shell within container...

Singularity hellocow:~>
```

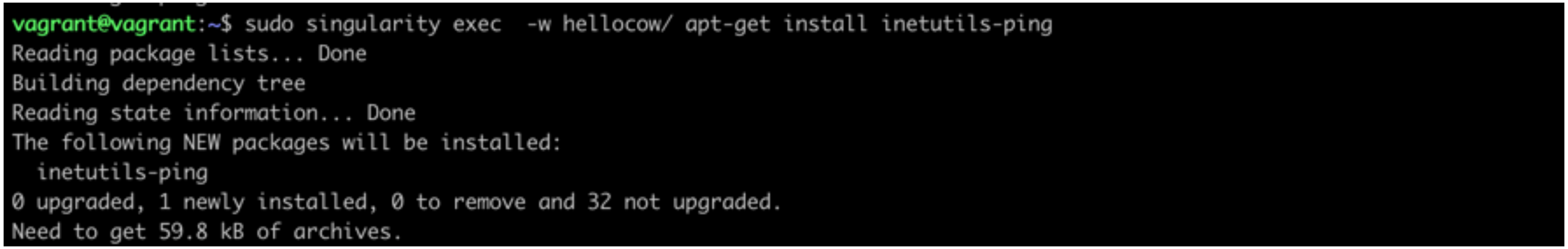
- You can check that it has is own os:



```
Singularity hellocow:~> cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04.3 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.3 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial
Singularity hellocow:~>
```

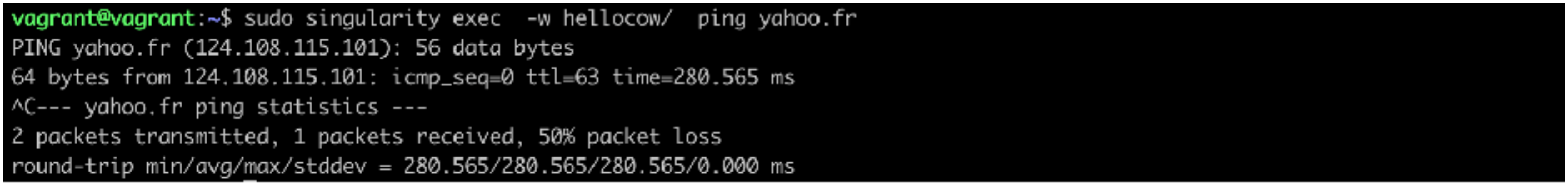
- If running without sudo
  - ➡ Can not become root in the image
- Running shell breaks reproducibility

Testing and Modifying image (II)



```
vagrant@vagrant:~$ sudo singularity exec  -w hellocow/ apt-get install inetutils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  inetutils-ping
0 upgraded, 1 newly installed, 0 to remove and 32 not upgraded.
Need to get 59.8 kB of archives.
```

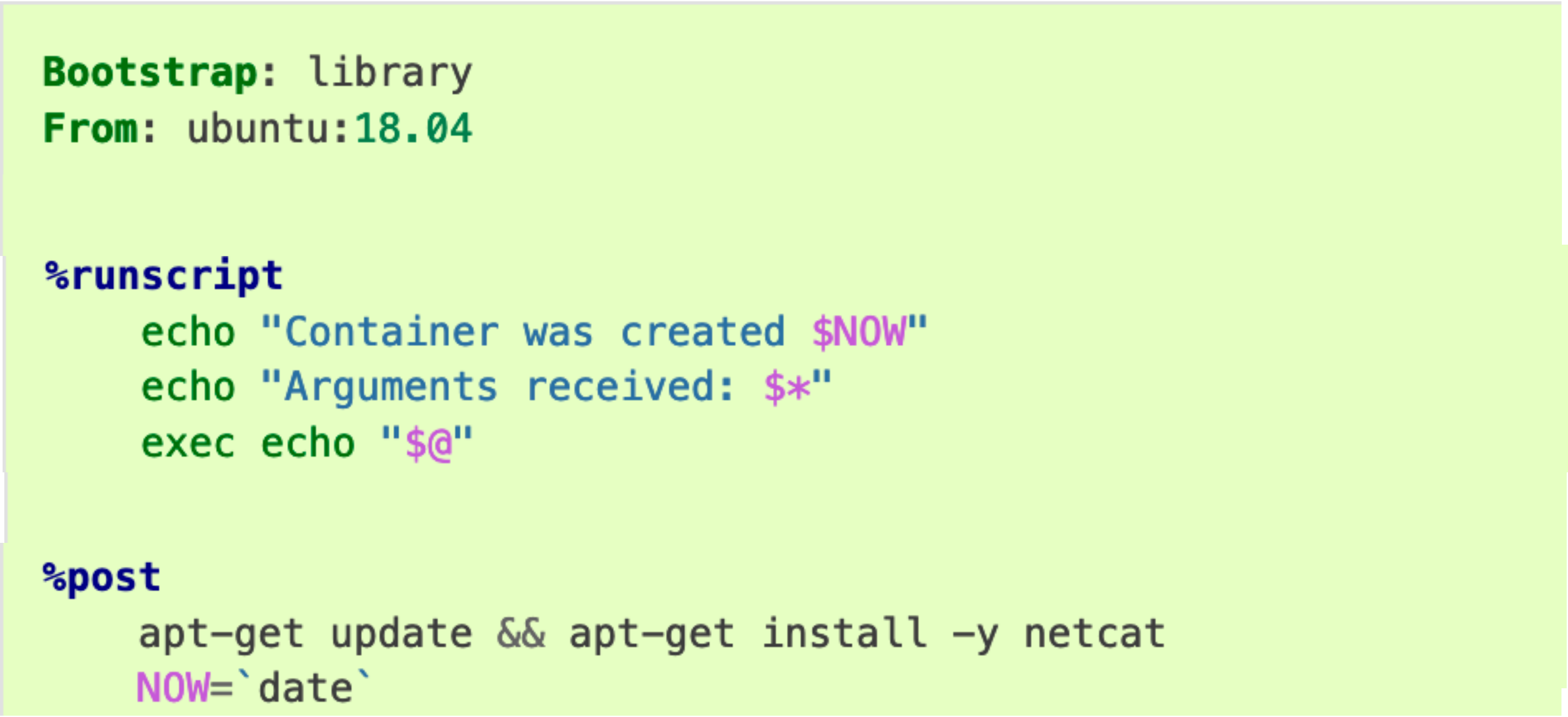
- Test:



```
vagrant@vagrant:~$ sudo singularity exec  -w hellocow/  ping yahoo.fr
PING yahoo.fr (124.108.115.101): 56 data bytes
64 bytes from 124.108.115.101: icmp_seq=0 ttl=63 time=280.565 ms
^C--- yahoo.fr ping statistics ---
2 packets transmitted, 1 packets received, 50% packet loss
round-trip min/avg/max/stddev = 280.565/280.565/280.565/0.000 ms
```

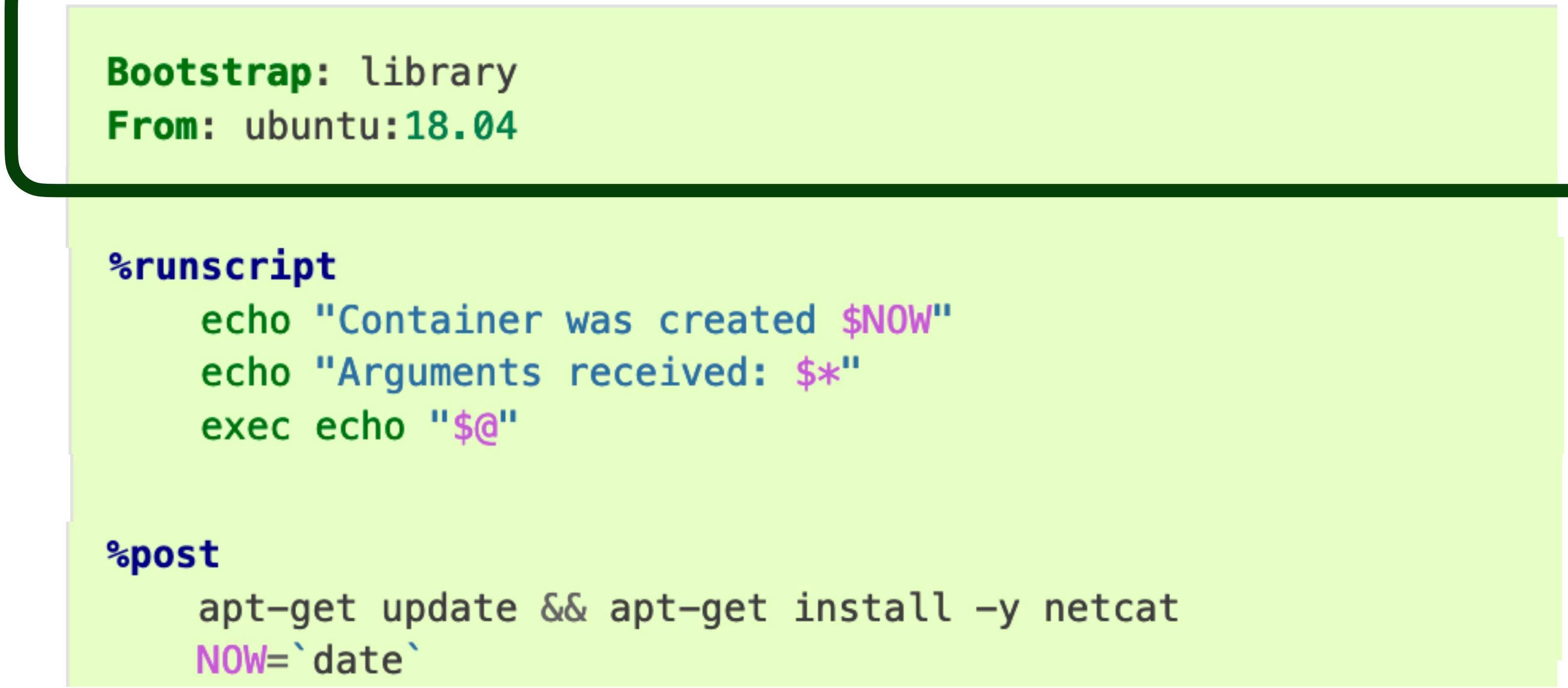
- Allow to create an image step by step and keep a script with all modification
  - ➡ What if we want something more powerful

Recipe file



```
Bootstrap: library
From: ubuntu:18.04


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


%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
```

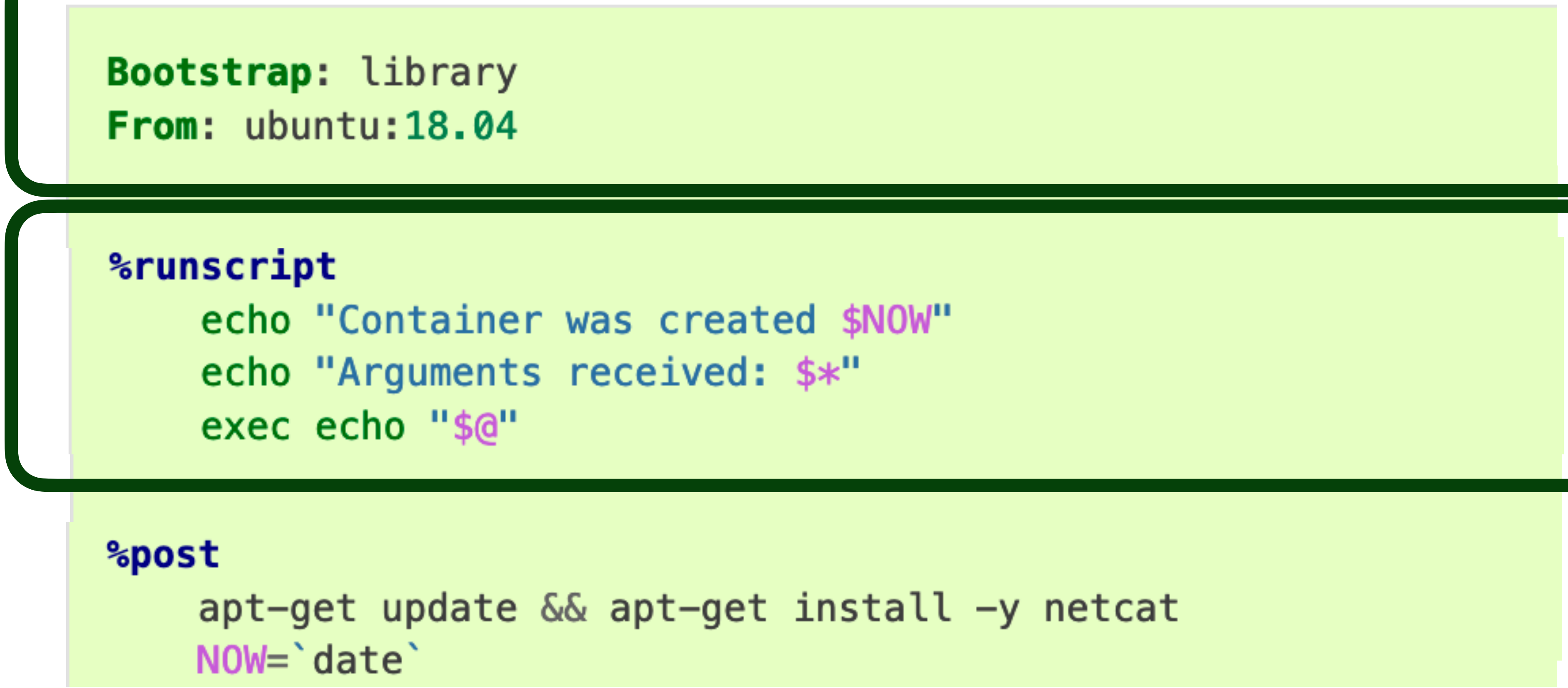
Recipe file



```
Bootstrap: library
From: ubuntu:18.04

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

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
```

Based on

Recipe file

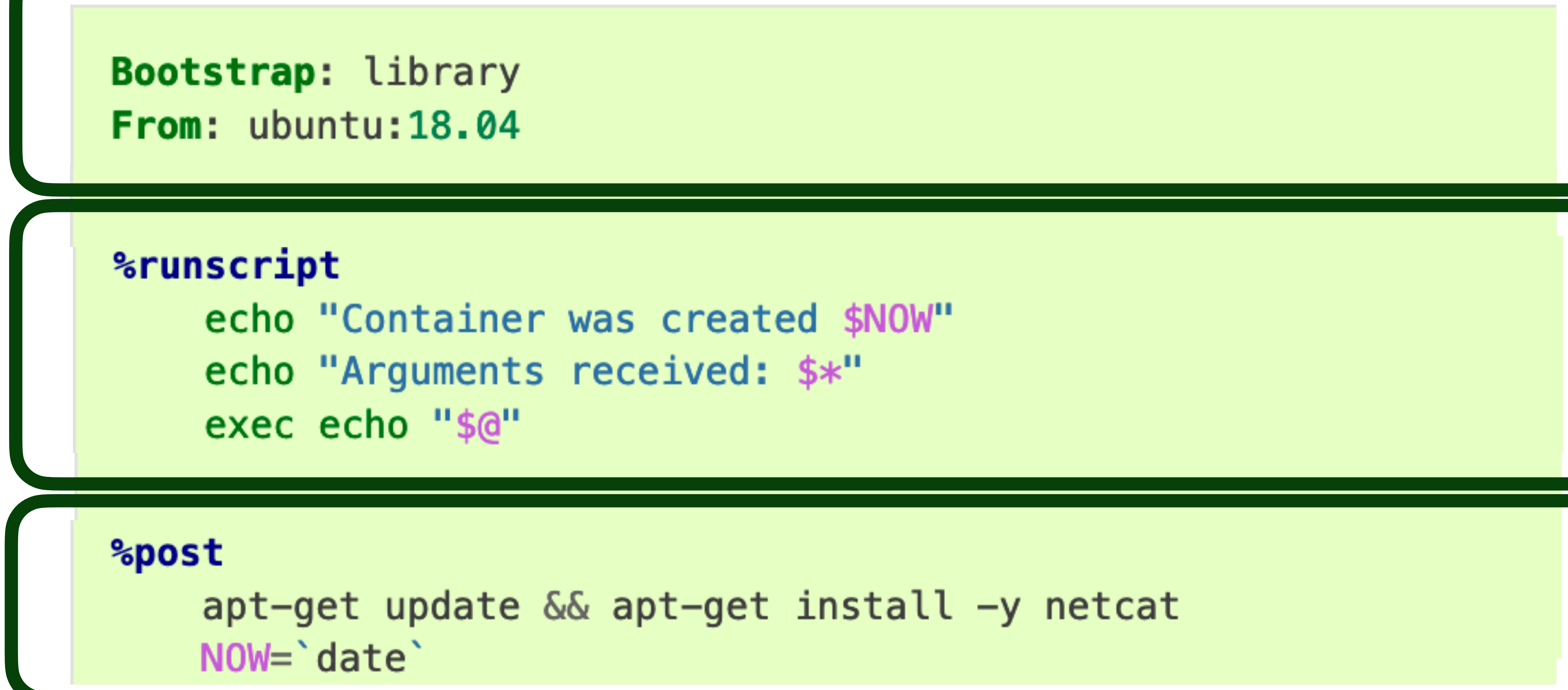


```
Bootstrap: library
From: ubuntu:18.04

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

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
```

Based on

What to do

Recipe file



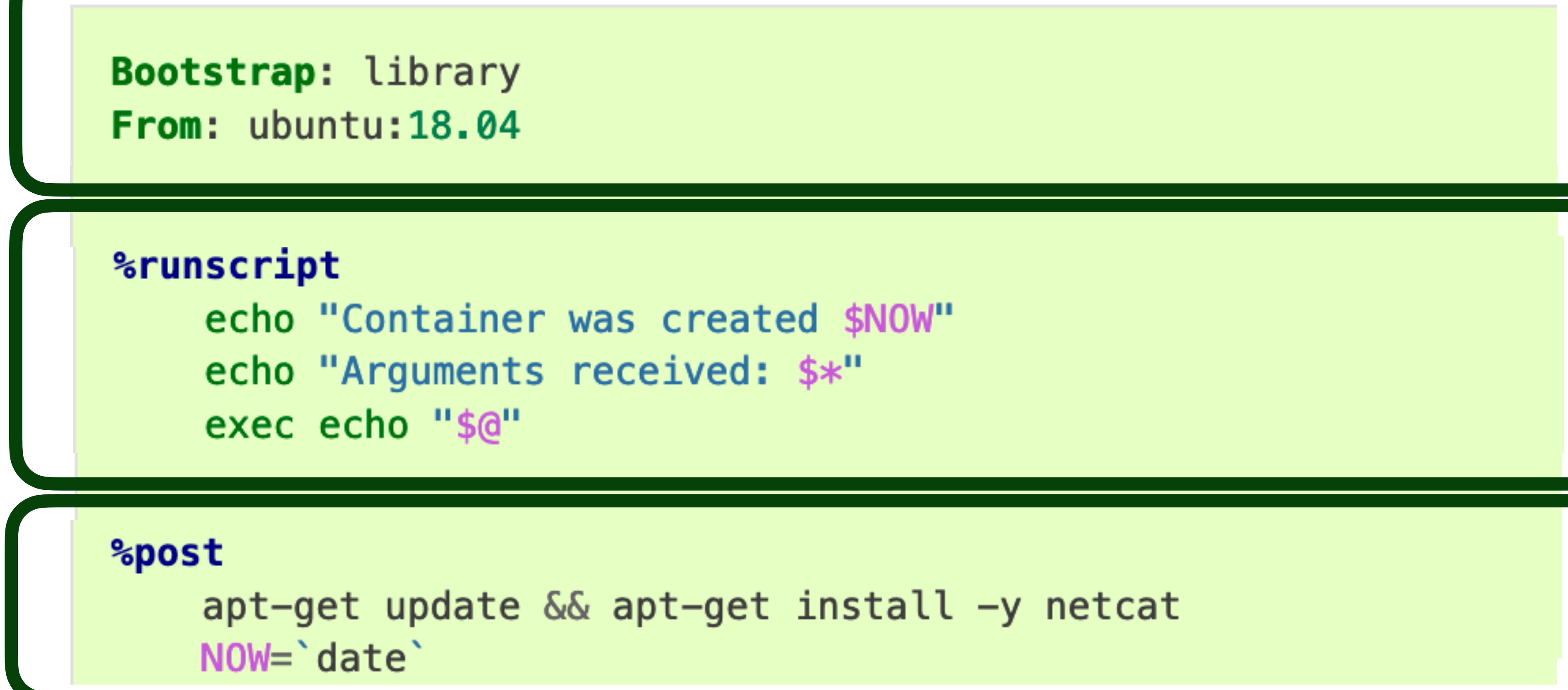
```
Bootstrap: library
From: ubuntu:18.04

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

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
```

Based on

What to do

How to install

Recipe file



```
Bootstrap: library
From: ubuntu:18.04
```

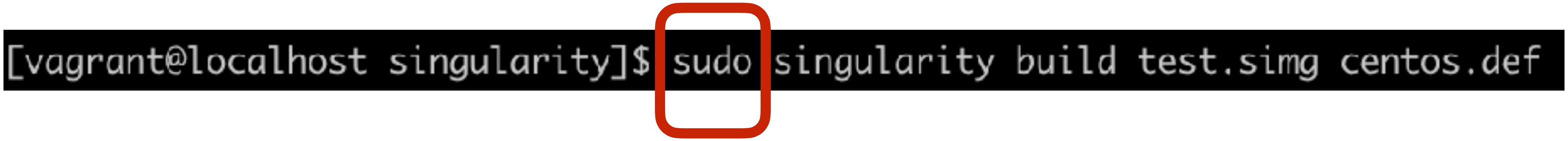
Based on

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

What to do

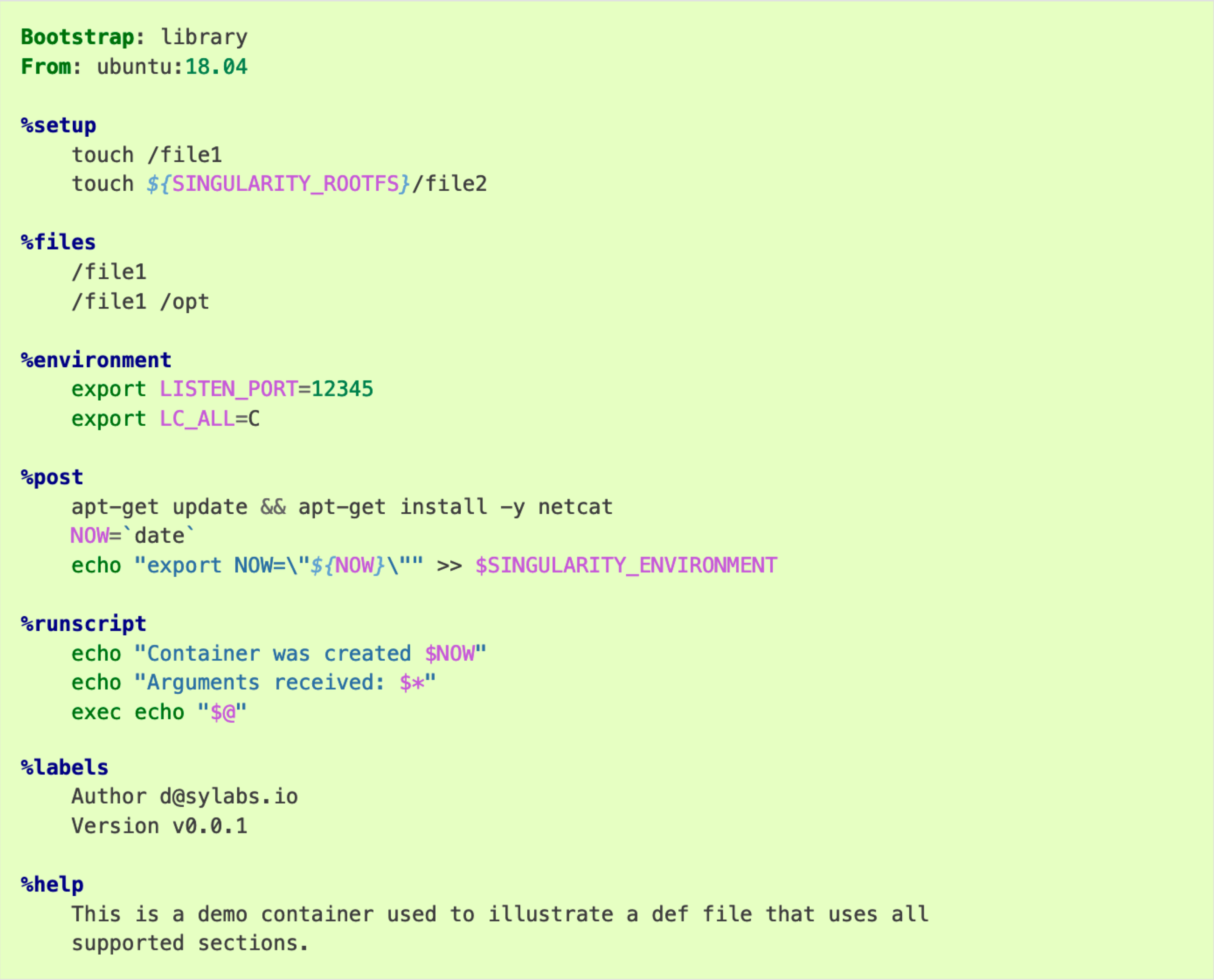
```
%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
```

How to install



```
[vagrant@localhost singularity]$ sudo singularity build test.simg centos.def
```

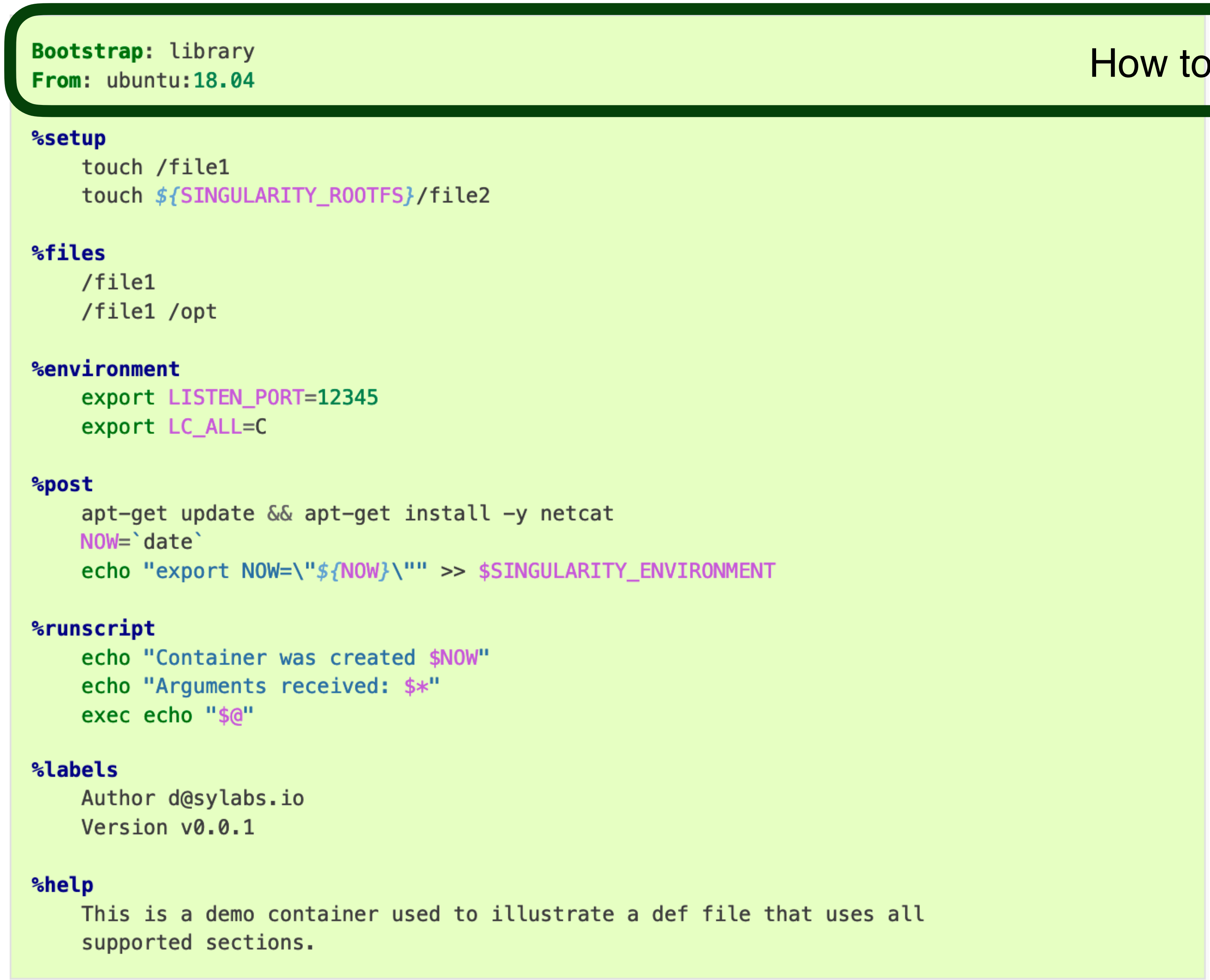
Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

Also %test %startscript + support for app

Recipe file

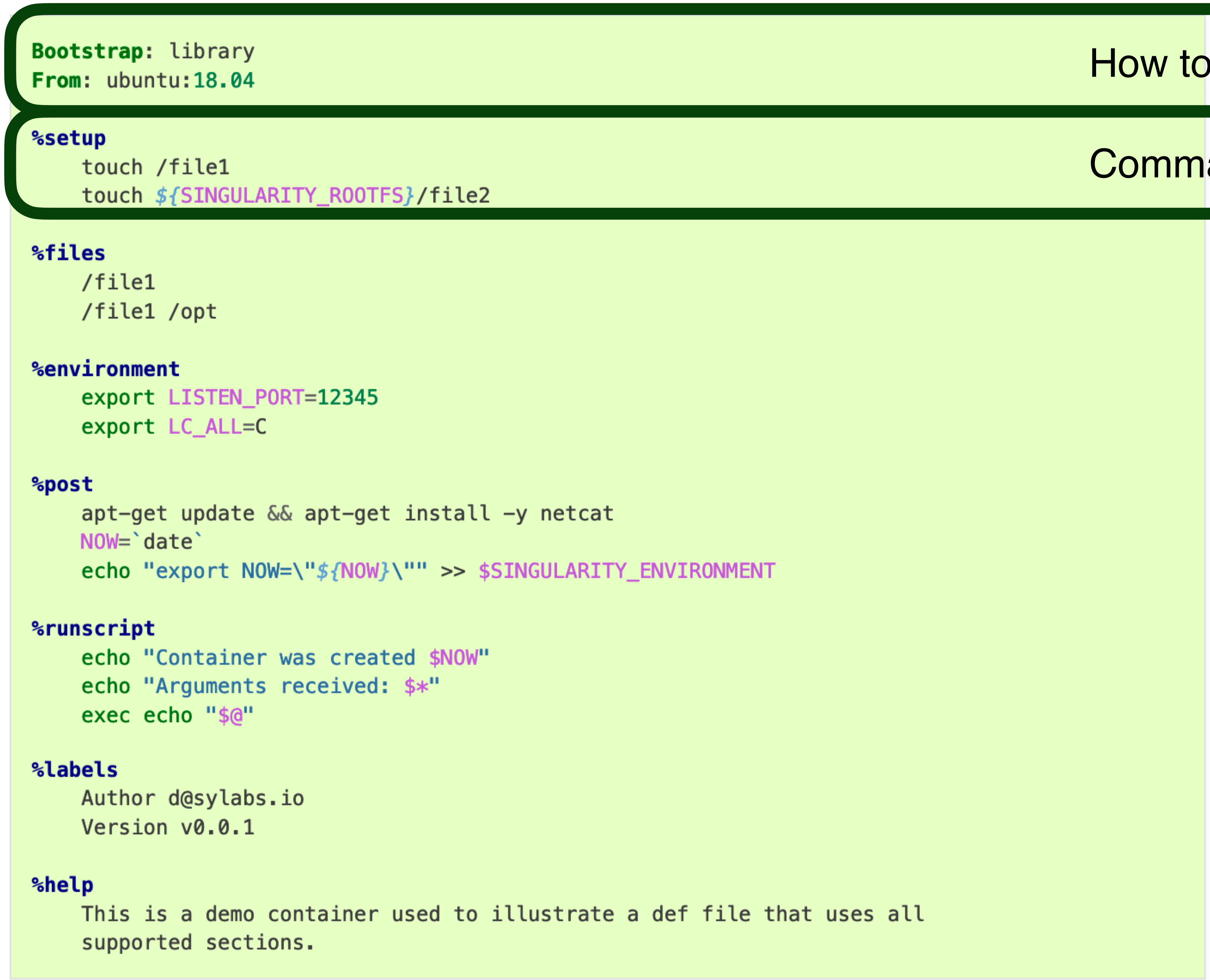


```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Also %test %startscript + support for app

Recipe file



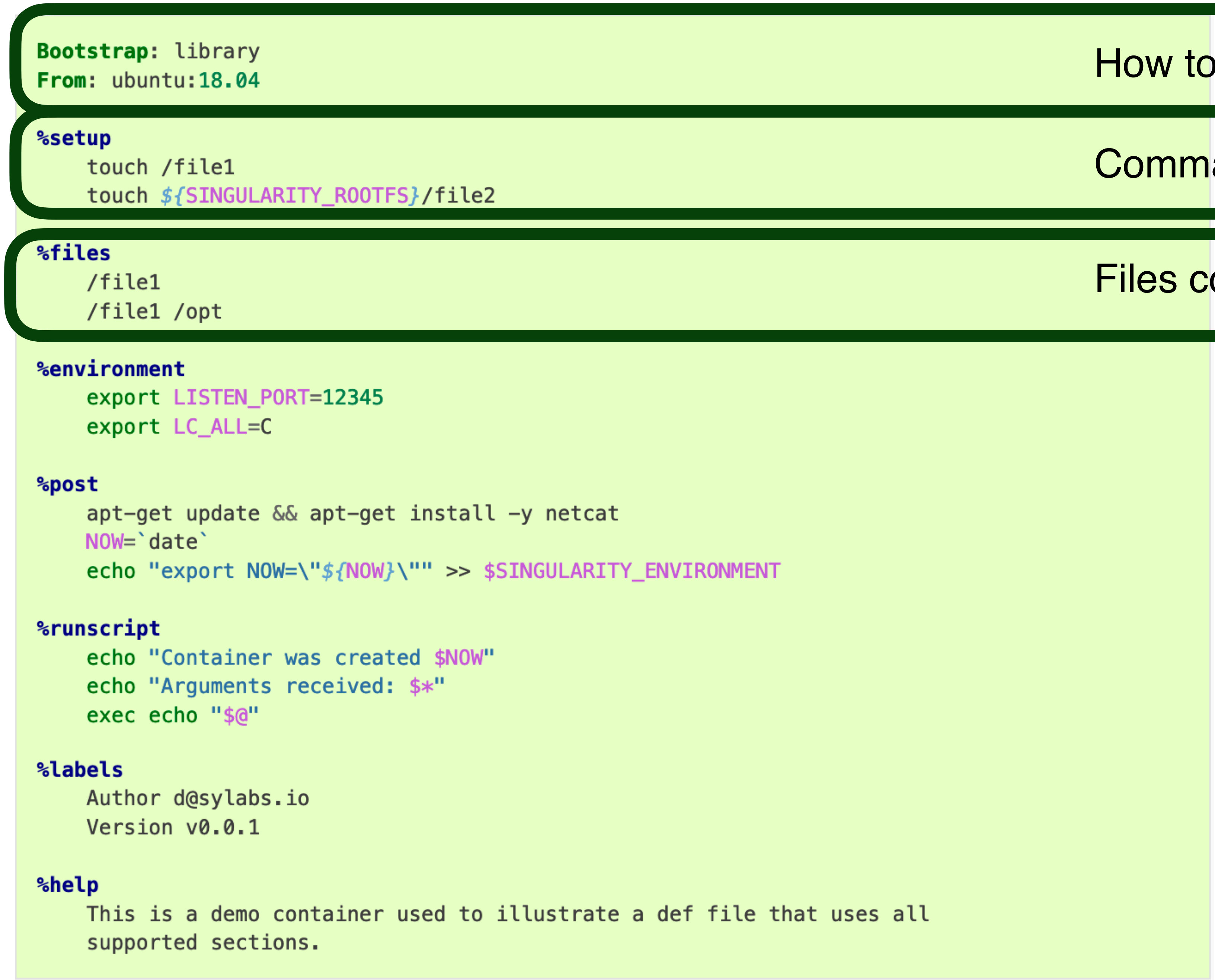
```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Command run on the host

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

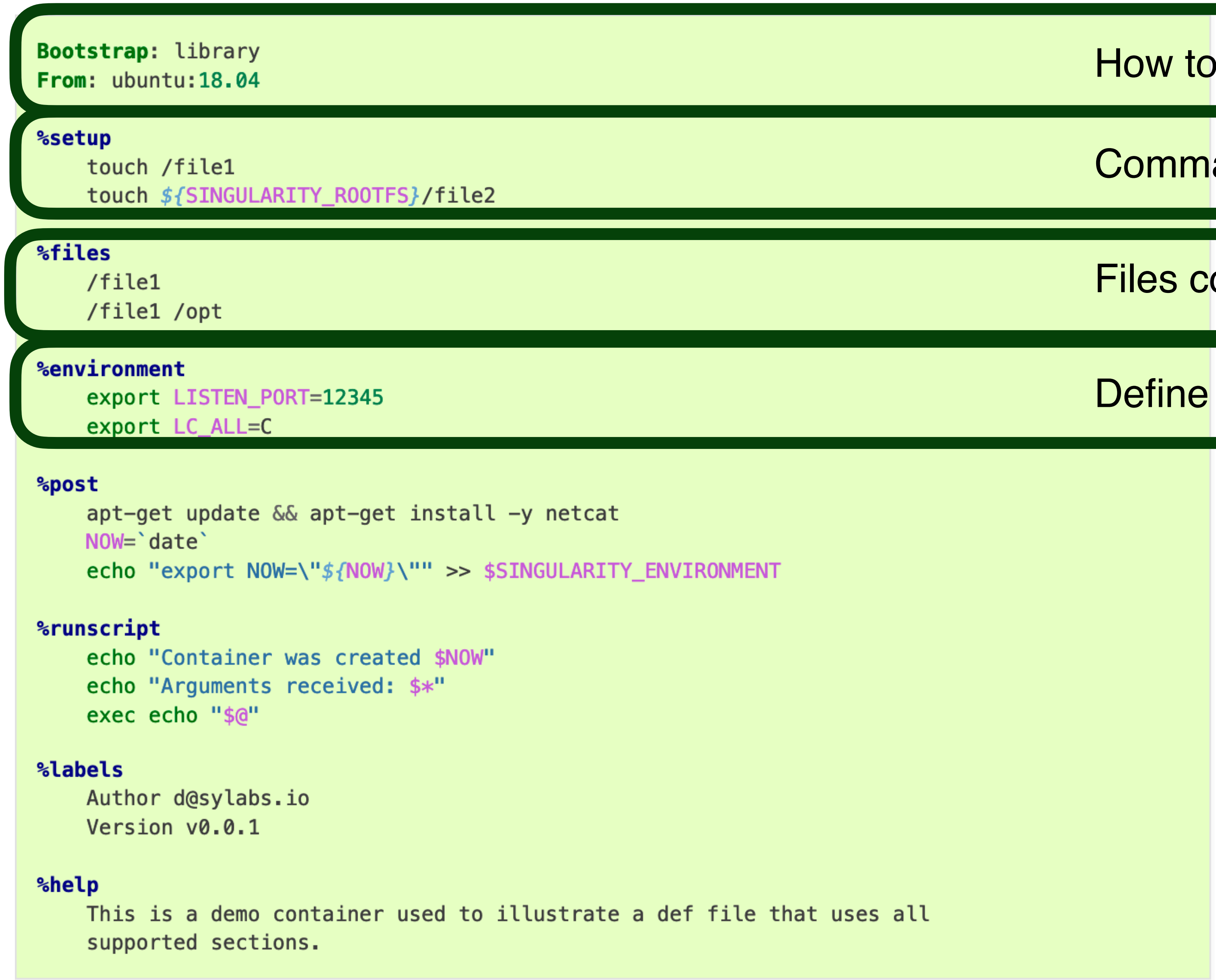
How to start (previous container/…)

Command run on the host

Files copy into the container

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

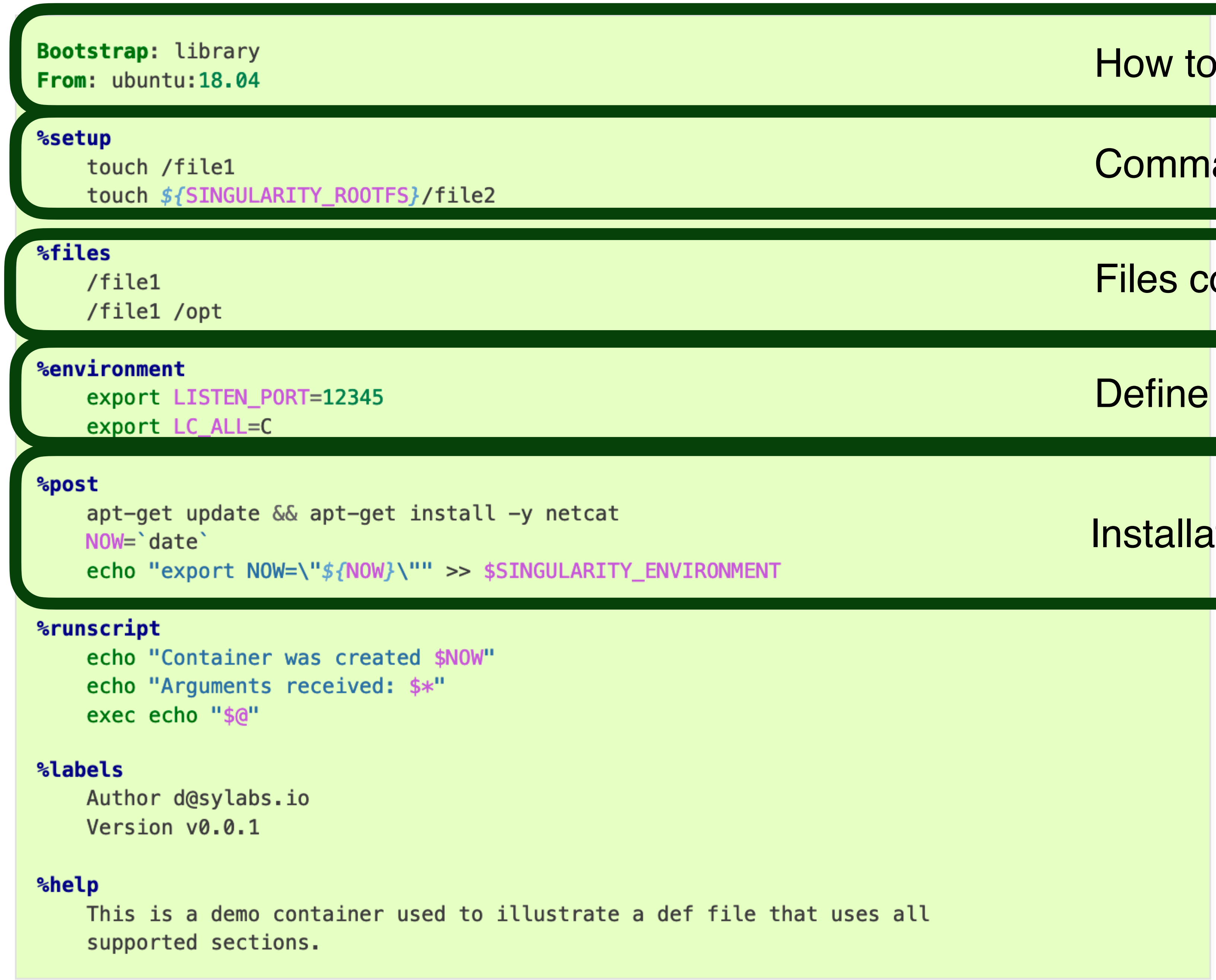
Command run on the host

Files copy into the container

Define environment variables

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Command run on the host

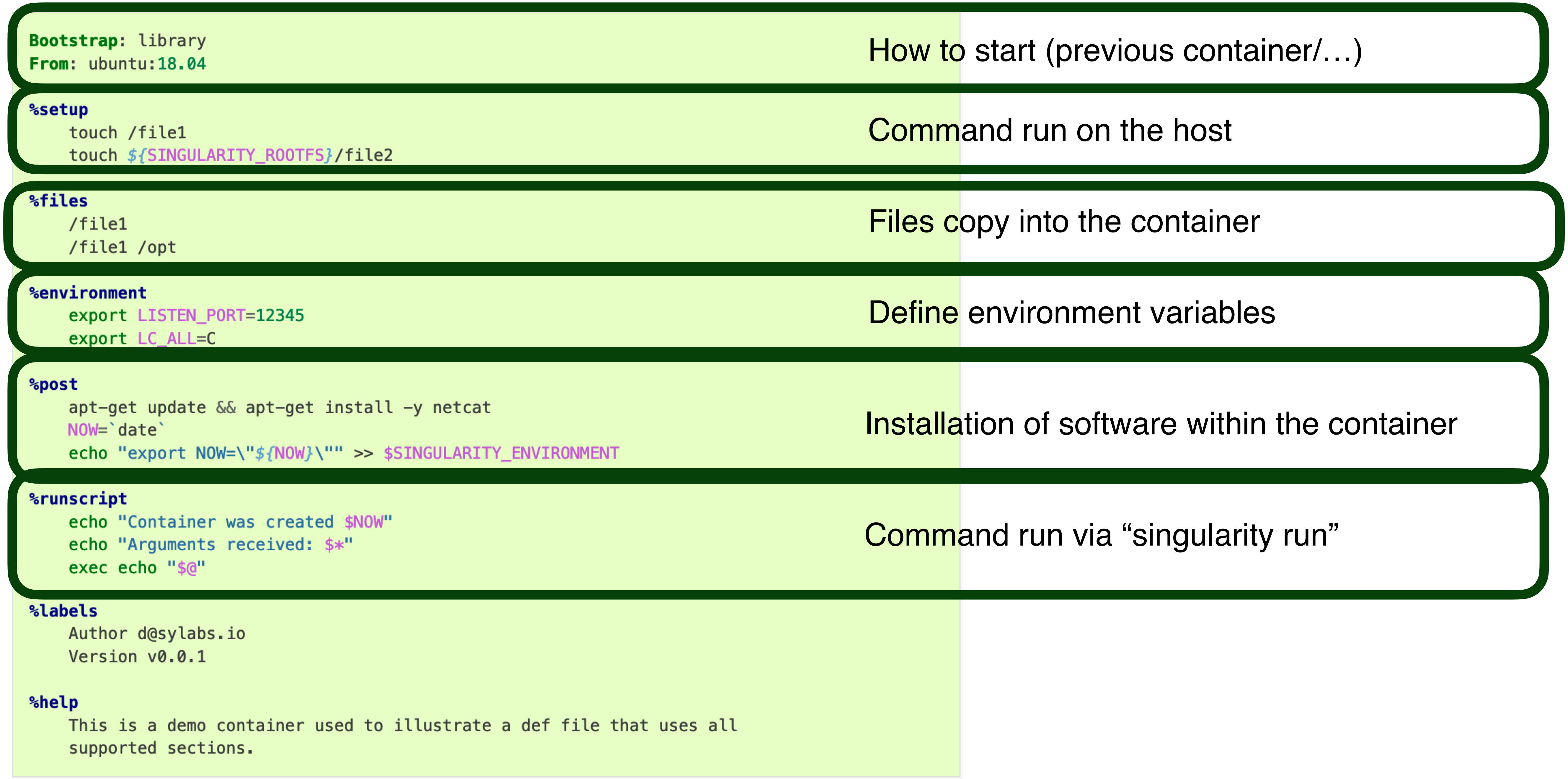
Files copy into the container

Define environment variables

Installation of software within the container

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Command run on the host

Files copy into the container

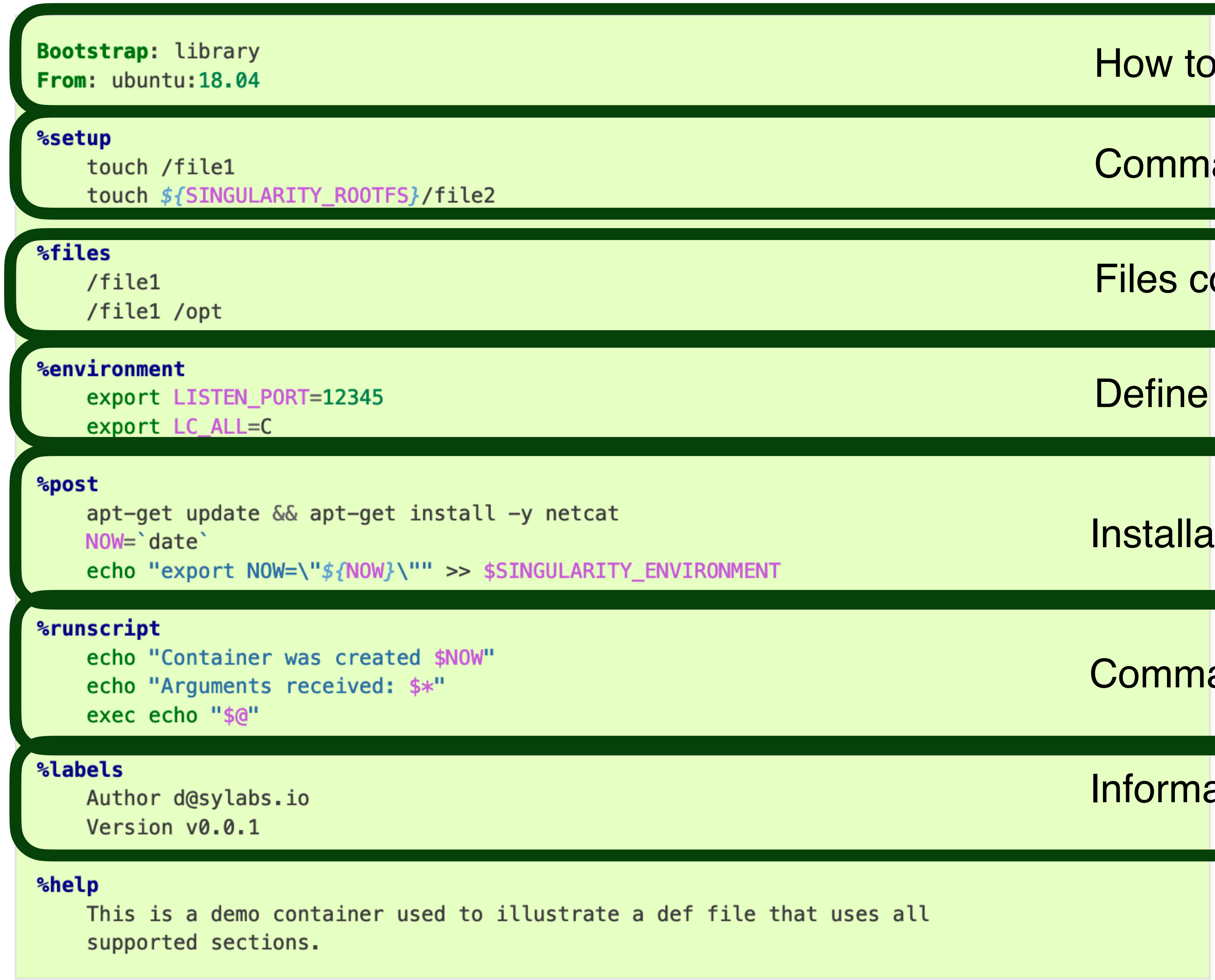
Define environment variables

Installation of software within the container

Command run via “singularity run”

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Command run on the host

Files copy into the container

Define environment variables

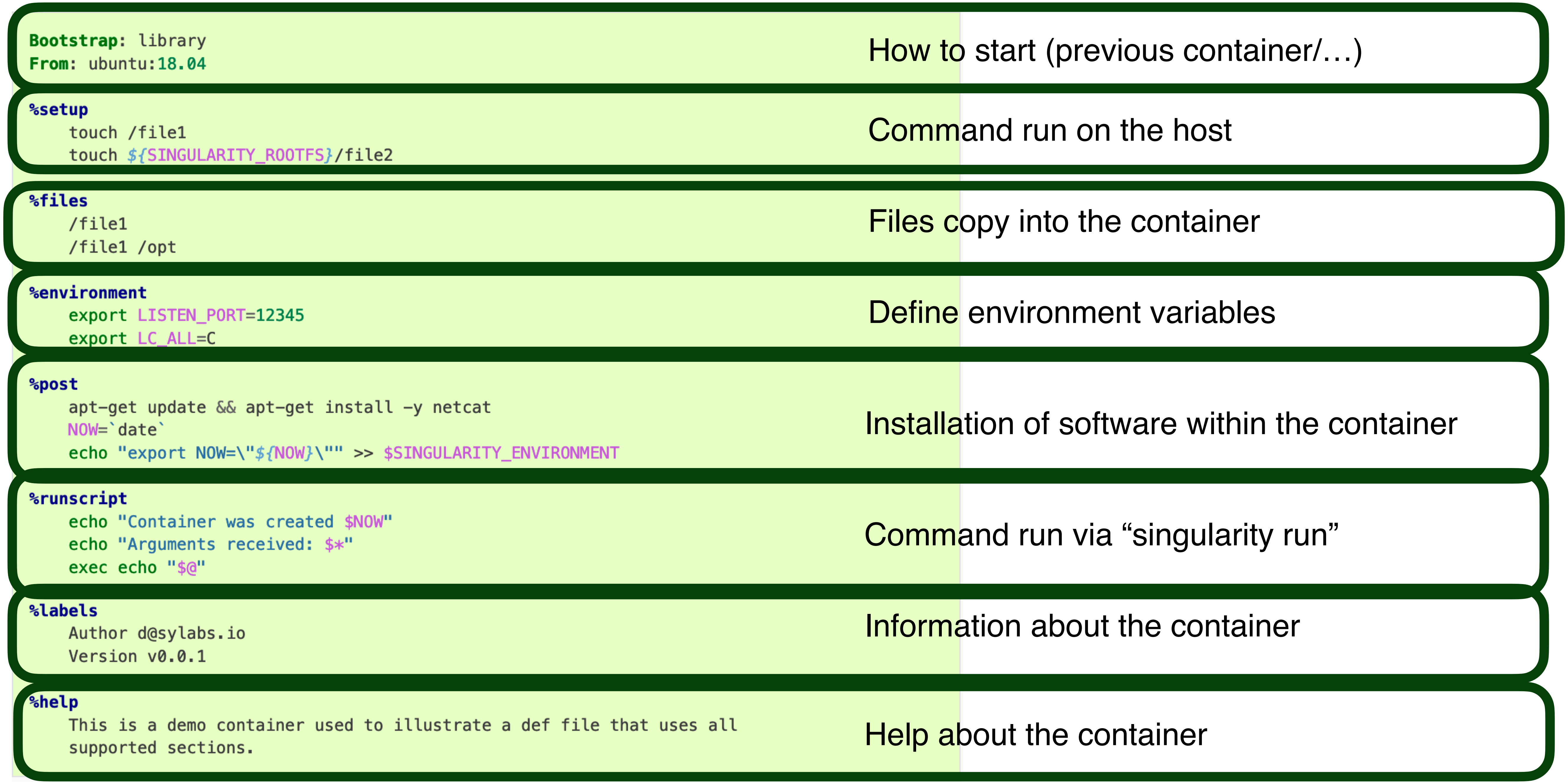
Installation of software within the container

Command run via “singularity run”

Information about the container

Also %test %startscript + support for app

Recipe file



```
Bootstrap: library
From: ubuntu:18.04

%setup
    touch /file1
    touch ${SINGULARITY_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}\"" >> $SINGULARITY_ENVIRONMENT

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

%labels
    Author d@sylabs.io
    Version v0.0.1

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

How to start (previous container/…)

Command run on the host

Files copy into the container

Define environment variables

Installation of software within the container

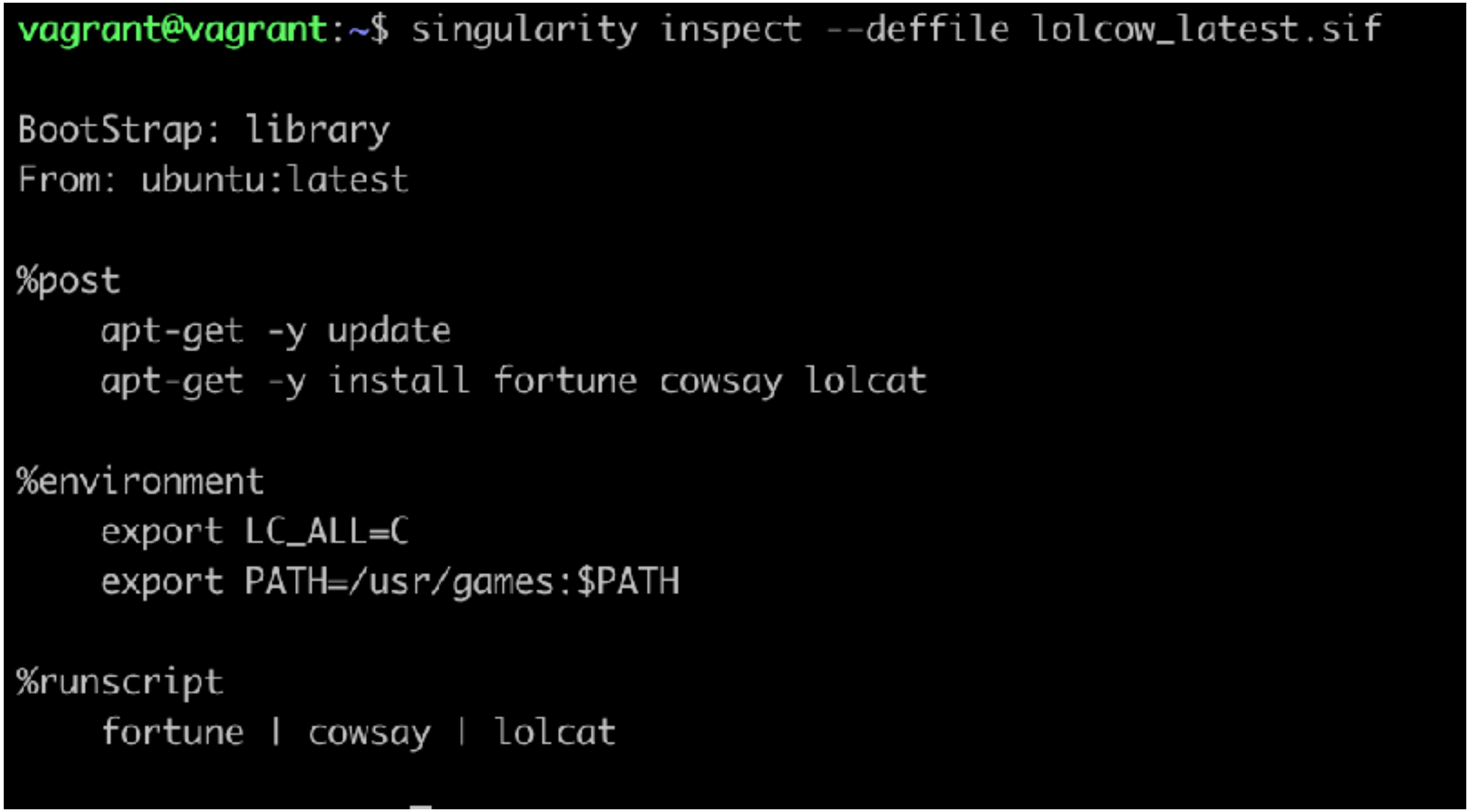
Command run via “singularity run”

Information about the container

Help about the container

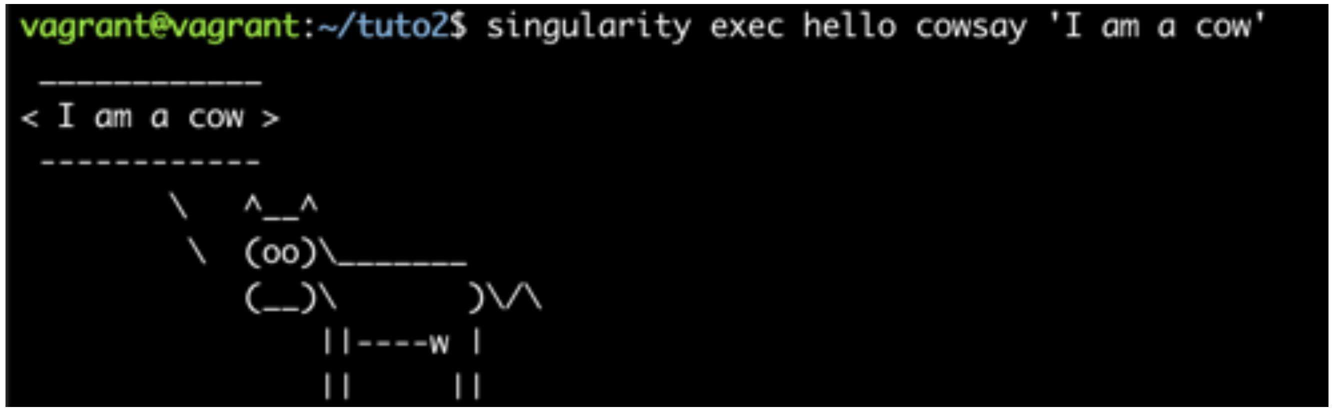
Also %test %startscript + support for app

Recover recipe file



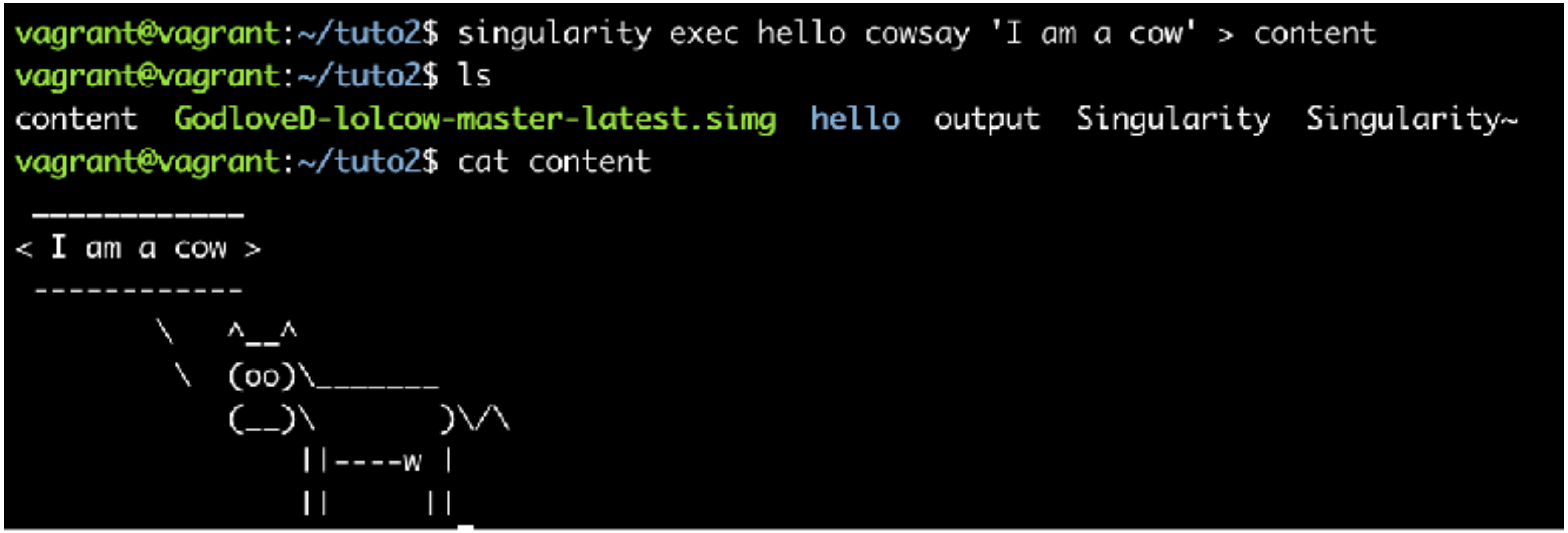
```
vagrant@vagrant:~$ singularity inspect --deffile lolcow_latest.sif

BootStrap: library
From: ubuntu:latest

%post
    apt-get -y update
    apt-get -y install fortune cowsay lolcat

%environment
    export LC_ALL=C
    export PATH=/usr/games:$PATH

%runscript
    fortune | cowsay | lolcat
```

Run with image



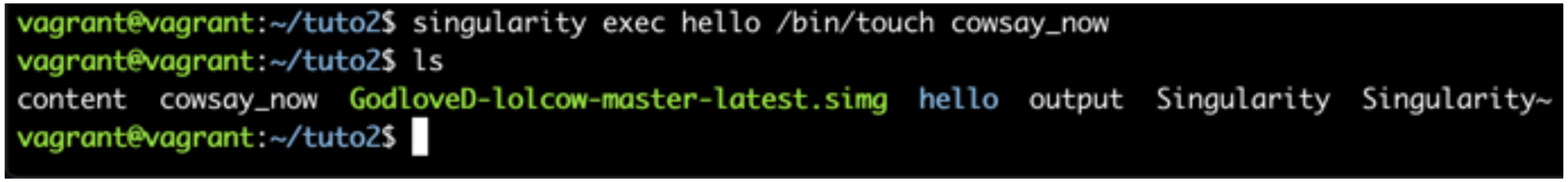
```
vagrant@vagrant:~/tuto2$ singularity exec hello cowsay 'I am a cow'
 ____________
< I am a cow >
 ------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

- Shell/piping works as normal



```
vagrant@vagrant:~/tuto2$ singularity exec hello cowsay 'I am a cow' > content
vagrant@vagrant:~/tuto2$ ls
content  GodloveD-lolcow-master-latest.simg  hello  output  Singularity  Singularity~
vagrant@vagrant:~/tuto2$ cat content
 ____________
< I am a cow >
 ------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

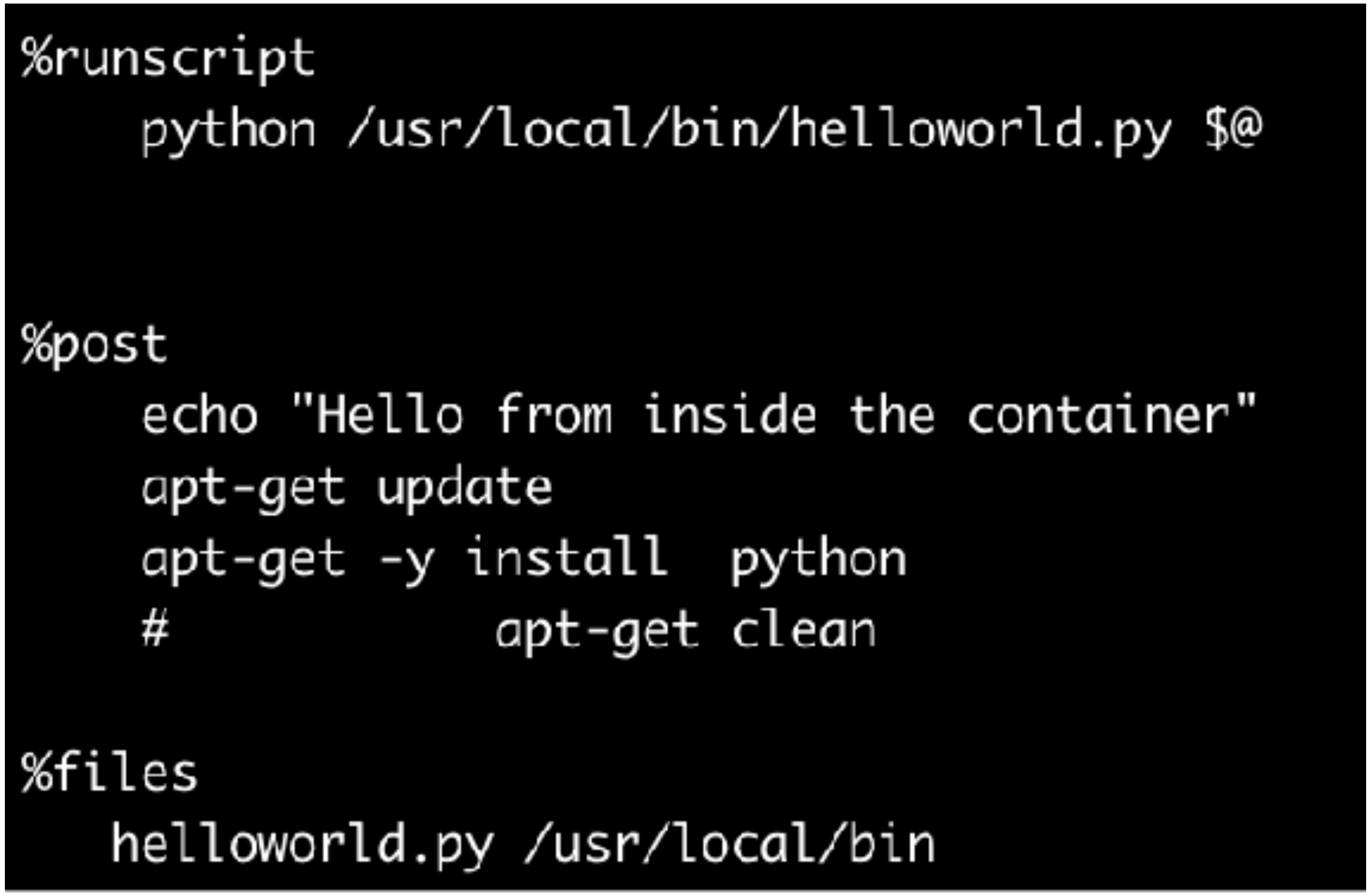
- As said before filesystem is the one of the host



```
vagrant@vagrant:~/tuto2$ singularity exec hello /bin/touch cowsay_now
vagrant@vagrant:~/tuto2$ ls
content  cowsay_now  GodloveD-lolcow-master-latest.simg  hello  output  Singularity  Singularity~
vagrant@vagrant:~/tuto2$ 
```

Run with image

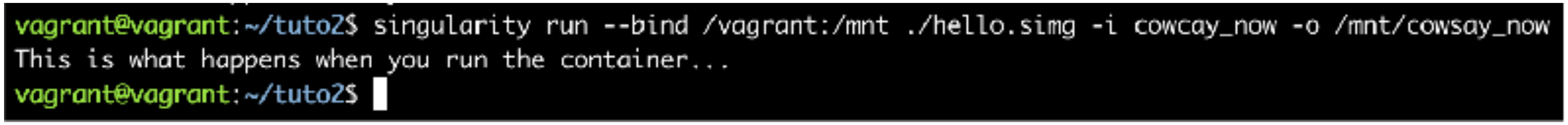
- Image are executable! (not --sandbox)
  - ./lolcow.simg
  - Run the “%runscript” part of the definition file!
    - Behave as an app
      - Think of putting help/…



```
%runscript
    python /usr/local/bin/helloworld.py $@


%post
    echo "Hello from inside the container"
    apt-get update
    apt-get -y install  python
    #           apt-get clean

%files
   helloworld.py /usr/local/bin
```

More on filesystem

- Special directory automatically mounted:
  - ➡ $HOME, /tmp, /proc, /sys, /dev
- You can create different mount point
  - ➡ Allow you to specify the path to data/output (specific to system)



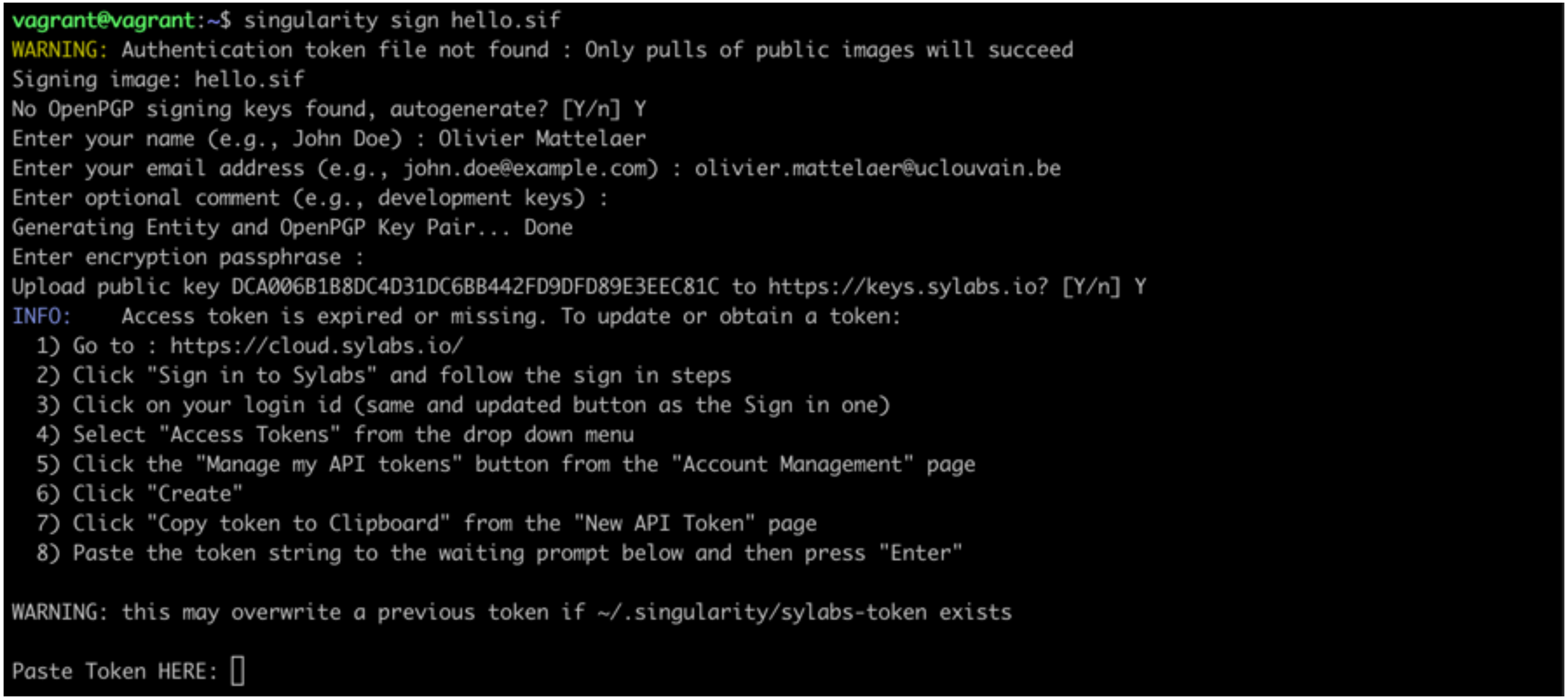
```
vagrant@vagrant:~/tuto2$ singularity run --bind /vagrant:/mnt ./hello.simg -i cowcay_now -o /mnt/cowsay_now
This is what happens when you run the container...
vagrant@vagrant:~/tuto2$ 
```

- ➡ File is now written in /vagrant of the VM

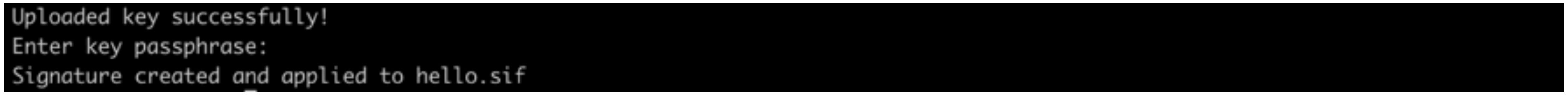
- Also possible via environment variable:
  - ➡ export SINGULARITY_BINDPATH=/vagrant:/mnt

Share

- You can store/distribute your singularity image via the singularity cloud
  - ➡ You can also provide your definition file directly online (easier)
- You need to **sign** your local container first: **Singularity sign container.sif**



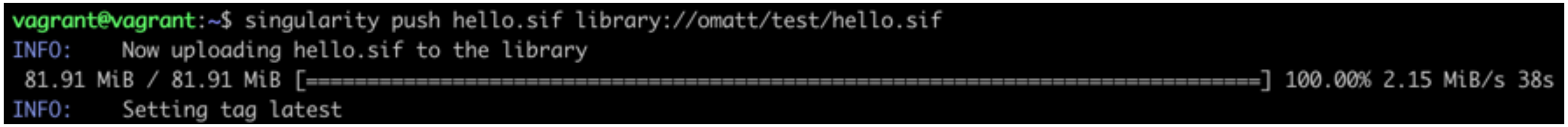
```
vagrant@vagrant:~$ singularity sign hello.sif
WARNING: Authentication token file not found : Only pulls of public images will succeed
Signing image: hello.sif
No OpenPGP signing keys found, autogenerate? [Y/n] Y
Enter your name (e.g., John Doe) : Olivier Mattelaer
Enter your email address (e.g., john.doe@example.com) : olivier.mattelaer@uclouvain.be
Enter optional comment (e.g., development keys) :
Generating Entity and OpenPGP Key Pair... Done
Enter encryption passphrase :
Upload public key DCA006B1B8DC4D31DC6BB442FD9DFD89E3EEC81C to https://keys.sylabs.io? [Y/n] Y
INFO:    Access token is expired or missing. To update or obtain a token:
  1) Go to : https://cloud.sylabs.io/
  2) Click "Sign in to Sylabs" and follow the sign in steps
  3) Click on your login id (same and updated button as the Sign in one)
  4) Select "Access Tokens" from the drop down menu
  5) Click the "Manage my API tokens" button from the "Account Management" page
  6) Click "Create"
  7) Click "Copy token to Clipboard" from the "New API Token" page
  8) Paste the token string to the waiting prompt below and then press "Enter"

WARNING: this may overwrite a previous token if ~/.singularity/sylabs-token exists

Paste Token HERE: []
```



```
Uploaded key successfully!
Enter key passphrase:
Signature created and applied to hello.sif
```

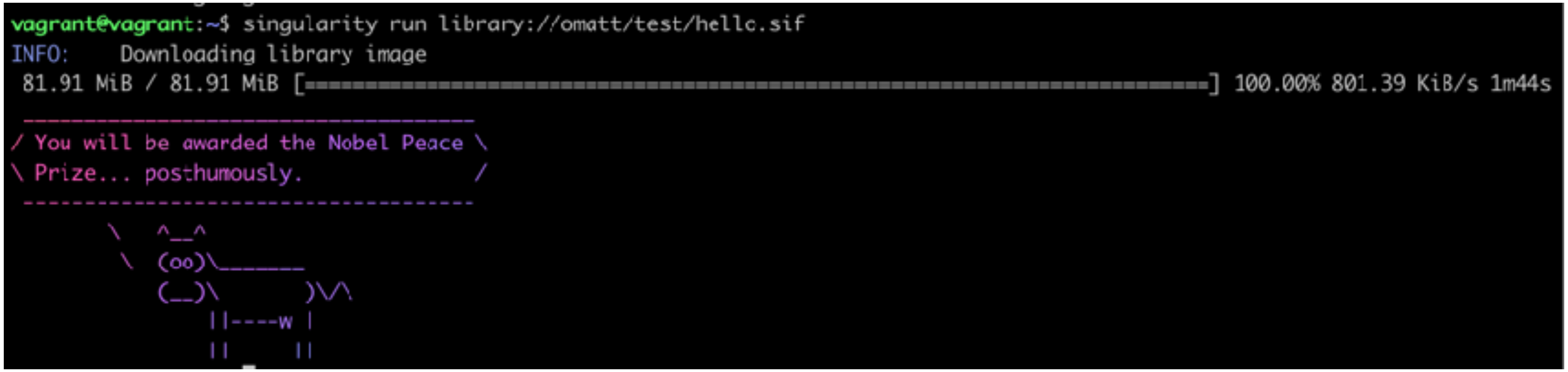
Share

- You can store/distribute your singularity image via the singularity cloud
  - ➡ You can also provide your definition file directly online (easier)
- You need to **sign** your local container first: **Singularity sign container.sif**
- Then you can push it to the cloud: **Singularity push container.sif LOCATION**
  - ➡ **LOCATION** should be **library://LOGIN/COLLECTIONS/FILES**



```
vagrant@vagrant:~$ singularity push hello.sif library://omatt/test/hello.sif
INFO:    Now uploading hello.sif to the library
 81.91 MiB / 81.91 MiB [==============================================================================] 100.00% 2.15 MiB/s 38s
INFO:    Setting tag latest
```

- You can now download/run it:
  - Singularity pull library://omatt/test/hello.sif
  - Singularity run library://omatt/test/hello.sif



```
vagrant@vagrant:~$ singularity run library://omatt/test/hellc.sif
INFO:    Downloading library image
 81.91 MiB / 81.91 MiB [==========================================================================] 100.00% 801.39 KiB/s 1m44s
 _____________________________________
/ You will be awarded the Nobel Peace \
\ Prize... posthumously.              /
 -------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

CÉCI clusters

- Singularity is available on
  - ➡ Lemaitre3
  - ➡ dragon2
  - ➡ Hercules2
  - ➡ Nic5

MPI

https://support.ceci-hpc.be/doc/_contents/UsingSoftwareAndLibraries/Singularity/index.html

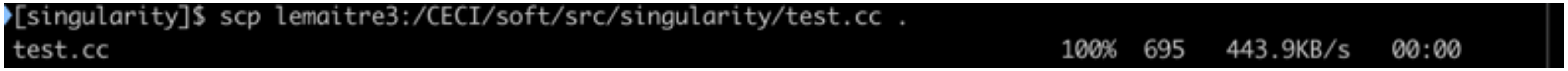
- MPI support requires
  - ➡ That you install the same slurm version as the one on our cluster
  - ➡ That you have the same version of mpi on the machine

- So you need matching pieces
  - ✓ We provide a starting container
    - ➡ Correct version of slurm
    - ➡ For each openmpi version
- You can use such container as base for your work

https://sylabs.io/guides/3.6/user-guide/mpi.html?highlight=mpi

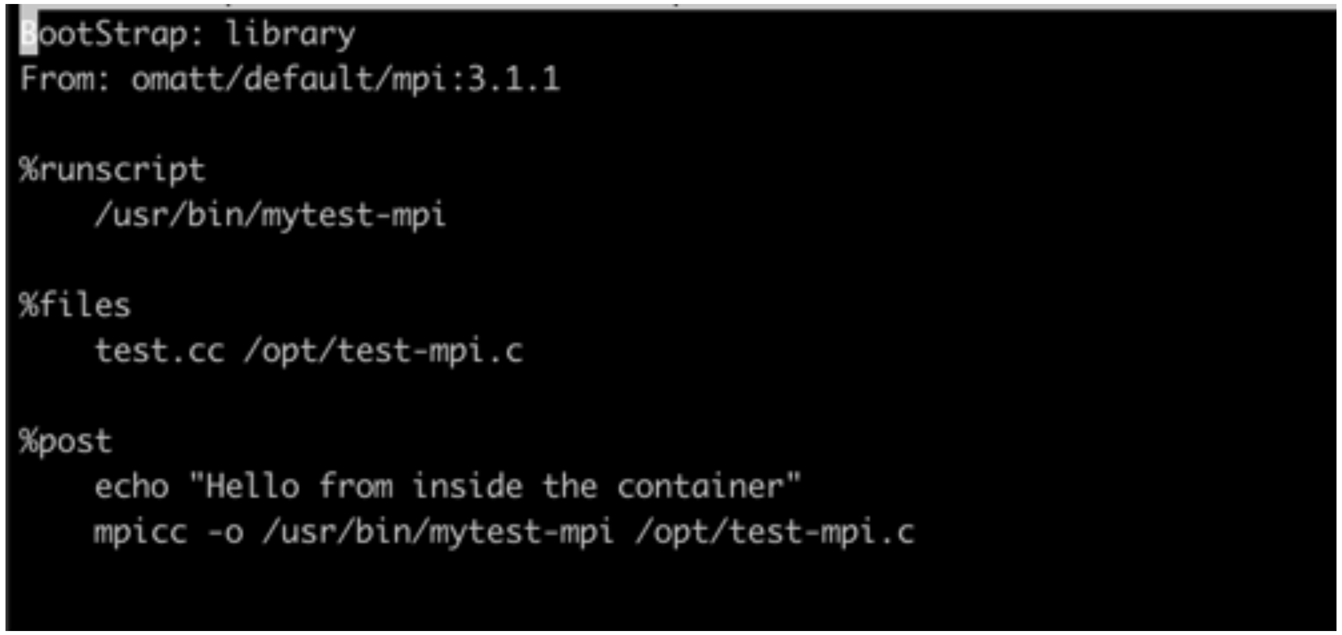
MPI on lemaitre3

- Copy your source code



```
[singularity]$ scp lemaitre3:/CECI/soft/src/singularity/test.cc .
test.cc                                                 100%  695   443.9KB/s   00:00
```

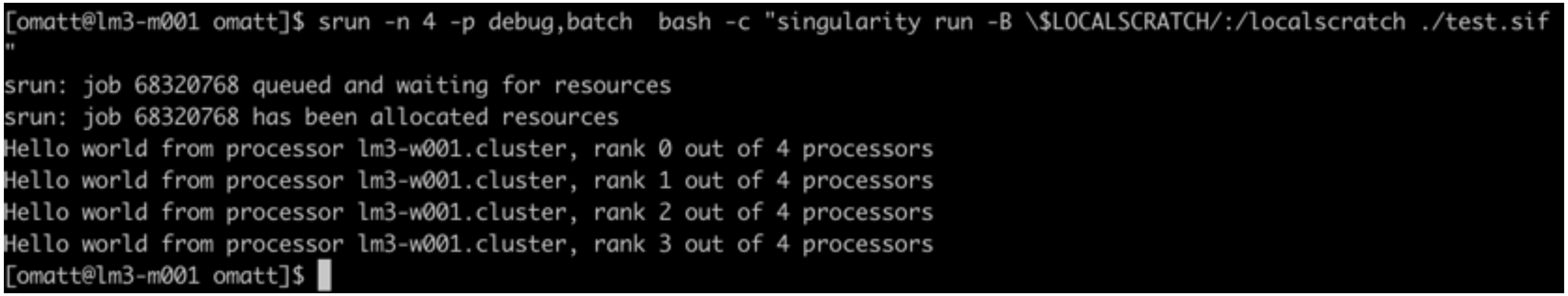
- Create your container (based on the one provided)



```
BootStrap: library
From: omatt/default/mpi:3.1.1

%runscript
    /usr/bin/mytest-mpi

%files
    test.cc /opt/test-mpi.c

%post
    echo "Hello from inside the container"
    mpicc -o /usr/bin/mytest-mpi /opt/test-mpi.c
```

From is the image created for the CECI

- Copy your container on lm3 and run it

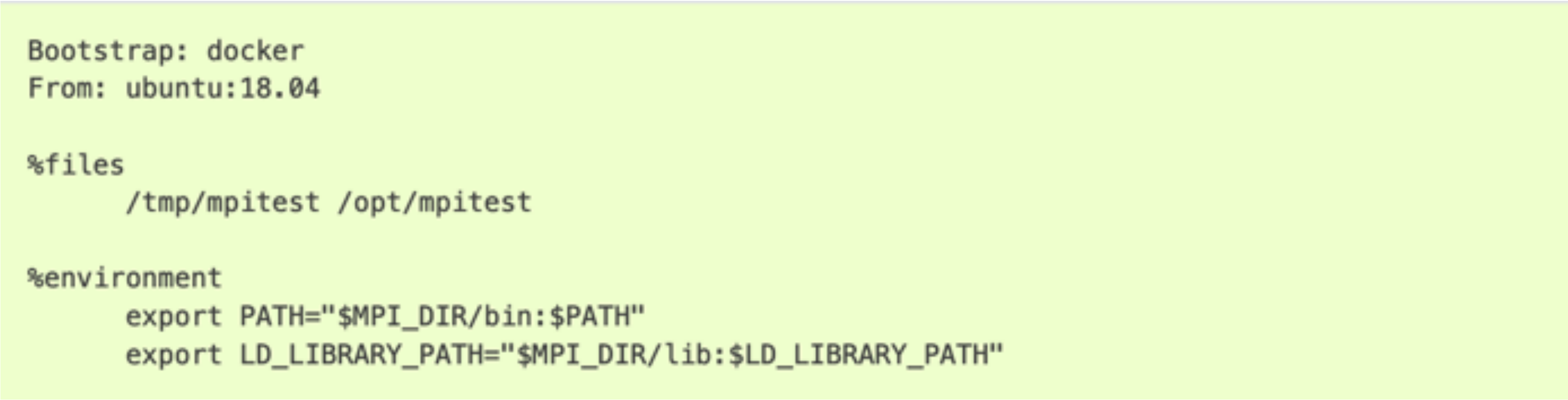


```
[omatt@lm3-m001 omatt]$ srun -n 4 -p debug,batch  bash -c "singularity run -B \$LOCALSCRATCH/:/localscratch ./test.sif
"
srun: job 68320768 queued and waiting for resources
srun: job 68320768 has been allocated resources
Hello world from processor lm3-w001.cluster, rank 0 out of 4 processors
Hello world from processor lm3-w001.cluster, rank 1 out of 4 processors
Hello world from processor lm3-w001.cluster, rank 2 out of 4 processors
Hello world from processor lm3-w001.cluster, rank 3 out of 4 processors
[omatt@lm3-m001 omatt]$
```

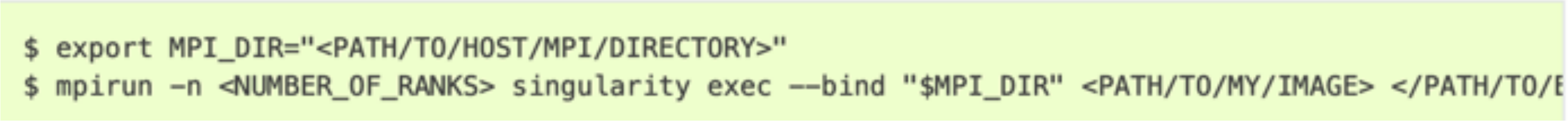
Note the binding path.

MPI bind method

- I failed to have it working on lemaitre3…
- Idea:
  - ➡ Compile your executable on the host
  - ➡ Move the binary within the singularity file
    - ✦ And nothing else
  - ➡ Link the host library on the flight



```
Bootstrap: docker
From: ubuntu:18.04

%files
      /tmp/mpitest /opt/mpitest

%environment
      export PATH="$MPI_DIR/bin:$PATH"
      export LD_LIBRARY_PATH="$MPI_DIR/lib:$LD_LIBRARY_PATH"
```

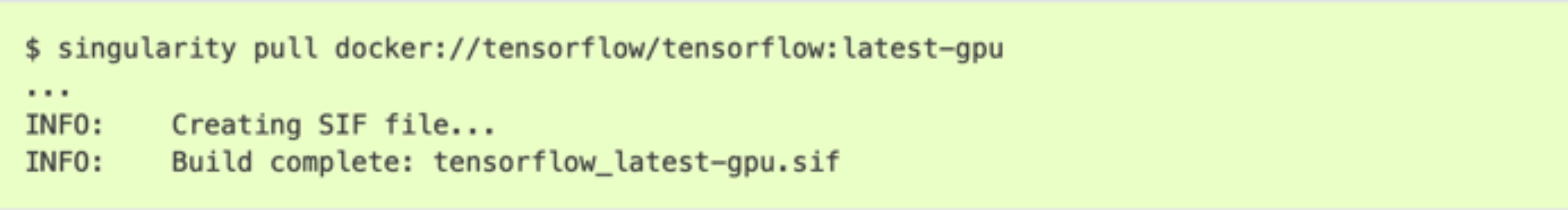


```
$ export MPI_DIR="<PATH/TO/HOST/MPI/DIRECTORY>"
$ mpirun -n <NUMBER_OF_RANKS> singularity exec --bind "$MPI_DIR" <PATH/TO/MY/IMAGE> </PATH/TO/E
```

https://sylabs.io/guides/3.8/user-guide/mpi.html#bind-model

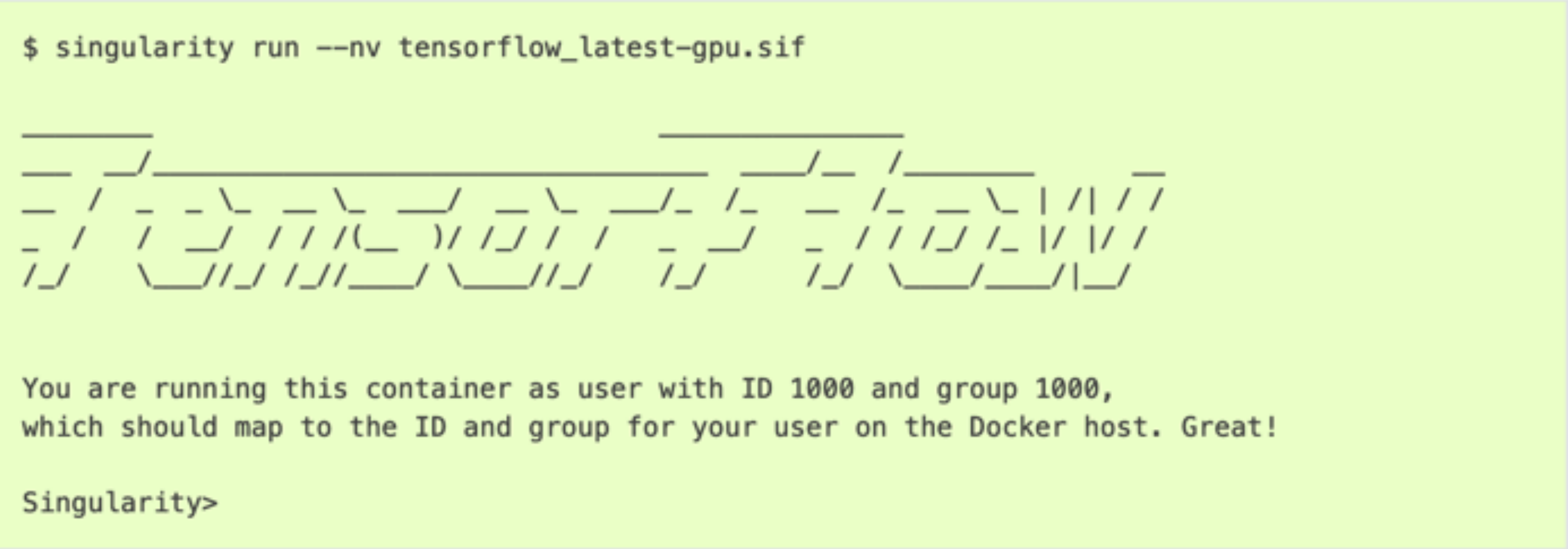
GPU

- Let’s take a image with require some gpu



```
$ singularity pull docker://tensorflow/tensorflow:latest-gpu
...
INFO:    Creating SIF file...
INFO:    Build complete: tensorflow_latest-gpu.sif
```

- To link to the GPU, you need to add —nv



```
$ singularity run --nv tensorflow_latest-gpu.sif

________                               _______________
___  __/__________________________________  ____/__  /________      __
__  /  _  _ \_  __ \_  ___/  __ \_  ___/_  /_   __  /_  __ \_ | /| / /
_  /   /  __/  / / /(__  )/ /_/ /  /   _  __/   _  / / /_/ /_ |/ |/ /
/_/    \___//_/ /_//____/ \____//_/    /_/      /_/  \____/____/|__/


You are running this container as user with ID 1000 and group 1000,
which should map to the ID and group for your user on the Docker host. Great!

Singularity>
```

Hands-on Session

- Follow the tutorial at the following page:
  - ➡ https://github.com/oliviermattelaer/Singularity-Tutorial

Conclusion

- Singularity
  - ➡ Nice way to share code with colleague
  - ➡ Portability and reproducibility
- Few command to learn
  - ➡ But not that complicated!
- Need to be root on machine
  - ➡ Ok that’s annoying…
    - ✦ Virtual machine option quite practical
  - ➡ Remote building exists for recipe files