CHALMERS
UNIVERSITY OF TECHNOLOGY

Containers

Stephen Bourke
Onsala Space Observatory

What is a container?

- It’s a file system image (eg. tar archive) that gets used instead of the local file system
- An application to set things up (eg. singularity, docker)

  When you run an application via a container that application is isolated from the rest of the OS

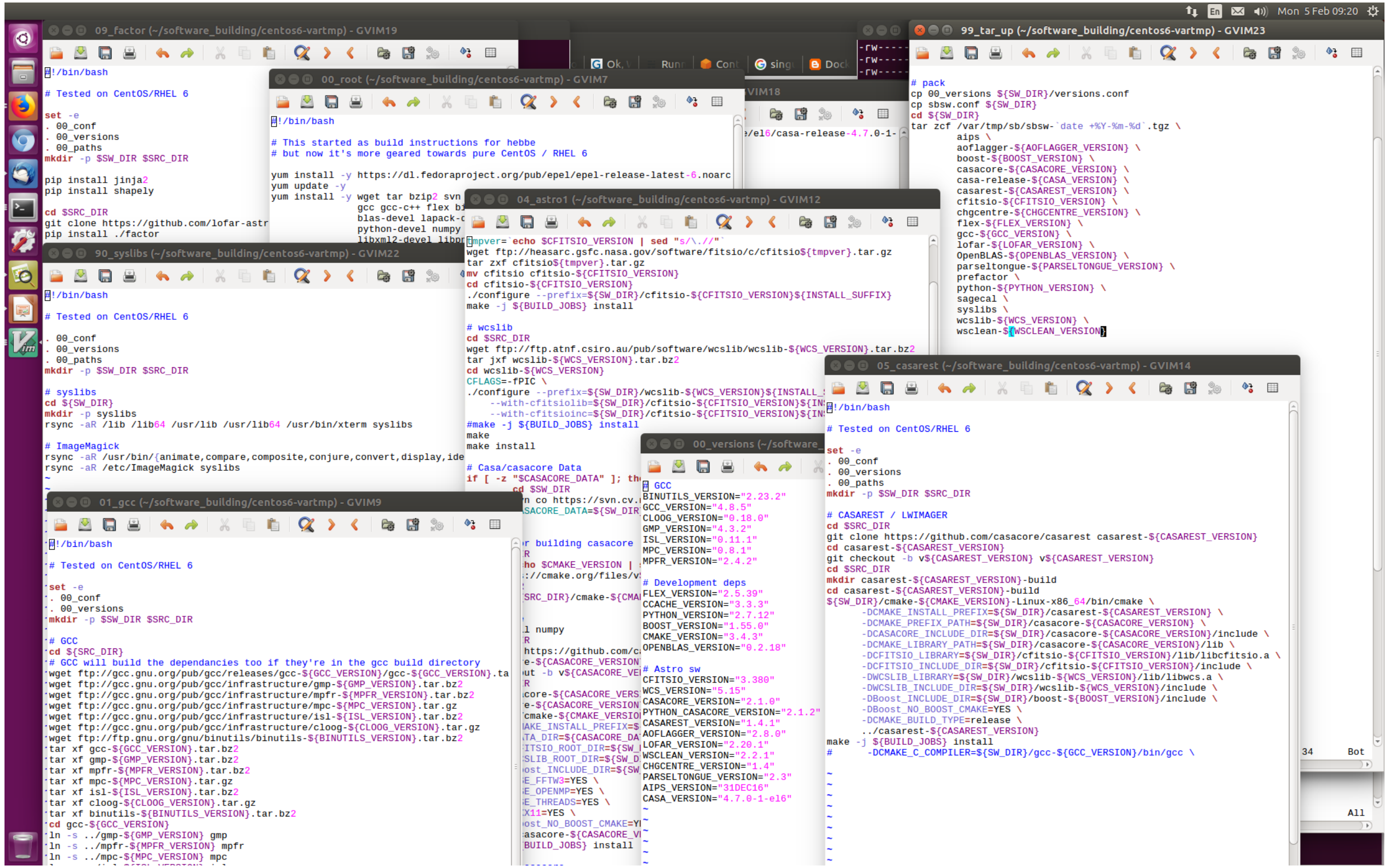
  Possible by Linux kernel features

What can containers do

- Separate application from infrastructure
- Ease / Speed of deployment
- Archive / Repeatability
- Application isolation

CHALMERS
UNIVERSITY OF TECHNOLOGY

Before:



Mon 5 Feb 09:20
09_factor (~/software_building/centos6-vartmp) - GVIM19
00_root (~/software_building/centos6-vartmp) - GVIM7
04_astro1 (~/software_building/centos6-vartmp) - GVIM12
99_tar_up (~/software_building/centos6-vartmp) - GVIM23
90_syslibs (~/software_building/centos6-vartmp) - GVIM22
01_gcc (~/software_building/centos6-vartmp) - GVIM9
00_versions (~/software_
05_casarest (~/software_building/centos6-vartmp) - GVIM14

```
#!/bin/bash

# Tested on CentOS/RHEL 6

set -e
. 00_conf
. 00_versions
. 00_paths
mkdir -p $SW_DIR $SRC_DIR

pip install jinja2
pip install shapely

cd $SRC_DIR
git clone https://github.com/lofar-astr
pip install ./factor
```

```
#!/bin/bash

# This started as build instructions for hebbe
# but now it's more geared towards pure CentOS / RHEL 6

yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarc
yum update -y
yum install -y wget tar bzip2 svn
               gcc gcc-c++ flex bi
               blas-devel lapack-c
               python-devel numpy
               libxml2-devel libpr
```

```
e/el6/casa-release-4.7.0-1-
```

```
tmpver=`echo $CFITSIO_VERSION | sed "s/\.//"`
wget ftp://heasarc.gsfc.nasa.gov/software/fitsio/c/cfitsio${tmpver}.tar.gz
tar zxf cfitsio${tmpver}.tar.gz
mv cfitsio cfitsio-${CFITSIO_VERSION}
cd cfitsio-${CFITSIO_VERSION}
./configure --prefix=${SW_DIR}/cfitsio-${CFITSIO_VERSION}${INSTALL_SUFFIX}
make -j ${BUILD_JOBS} install

# wcslib
cd $SRC_DIR
wget ftp://ftp.atnf.csiro.au/pub/software/wcslib/wcslib-${WCS_VERSION}.tar.bz2
tar jxf wcslib-${WCS_VERSION}.tar.bz2
cd wcslib-${WCS_VERSION}
CFLAGS=-fPIC \
./configure --prefix=${SW_DIR}/wcslib-${WCS_VERSION}${INSTALL_
    --with-cfitsiolib=${SW_DIR}/cfitsio-${CFITSIO_VERSION}${IN
    --with-cfitsioinc=${SW_DIR}/cfitsio-${CFITSIO_VERSION}${IN
#make -j ${BUILD_JOBS} install
make
make install

# Casa/casacore Data
if [ -z "$CASACORE_DATA" ]; the
        cd $SW_DIR
        n co https://svn.cv.
        SACORE_DATA=${SW_DIR

        r building casacore
        R
        ho $CMAKE_VERSION |
        ://cmake.org/files/v

        SRC_DIR}/cmake-${CMAK

        l numpy
        R
        https://github.com/c
        e-${CASACORE_VERSION
        ut -b v${CASACORE_VE
        R
        core-${CASACORE_VERS
        e-${CASACORE_VERSION
        cmake-${CMAKE_VERSIO
        AKE_INSTALL_PREFIX=$
        TA_DIR=${CASACORE_DA
        ITSIO_ROOT_DIR=${SW_
        SLIB_ROOT_DIR=${SW_D
        ost_INCLUDE_DIR=${SW
        E_FFTW3=YES \
        E_OPENMP=YES \
        E_THREADS=YES \
        X11=YES \
        ost_NO_BOOST_CMAKE=Y
        asacore-${CASACORE_V
        BUILD_JOBS} install
```

```
# pack
cp 00_versions ${SW_DIR}/versions.conf
cp sbsw.conf ${SW_DIR}
cd ${SW_DIR}
tar zcf /var/tmp/sb/sbsw-`date +%Y-%m-%d`.tgz \
        aips \
        aoflagger-${AOFLAGGER_VERSION} \
        boost-${BOOST_VERSION} \
        casacore-${CASACORE_VERSION} \
        casa-release-${CASA_VERSION} \
        casarest-${CASAREST_VERSION} \
        cfitsio-${CFITSIO_VERSION} \
        chgcentre-${CHGCENTRE_VERSION} \
        flex-${FLEX_VERSION} \
        gcc-${GCC_VERSION} \
        lofar-${LOFAR_VERSION} \
        OpenBLAS-${OPENBLAS_VERSION} \
        parseltongue-${PARSELTONGUE_VERSION} \
        prefactor \
        python-${PYTHON_VERSION} \
        sagecal \
        syslibs \
        wcslib-${WCS_VERSION} \
        wsclean-${WSCLEAN_VERSION}
```

```
#!/bin/bash

# Tested on CentOS/RHEL 6

. 00_conf
. 00_versions
. 00_paths
mkdir -p $SW_DIR $SRC_DIR

# syslibs
cd ${SW_DIR}
mkdir -p syslibs
rsync -aR /lib /lib64 /usr/lib /usr/lib64 /usr/bin/xterm syslibs

# ImageMagick
rsync -aR /usr/bin/{animate,compare,composite,conjure,convert,display,ide
rsync -aR /etc/ImageMagick syslibs
~
~
```

```
#!/bin/bash

# Tested on CentOS/RHEL 6

set -e
. 00_conf
. 00_versions
mkdir -p $SW_DIR $SRC_DIR

# GCC
cd ${SRC_DIR}
# GCC will build the dependancies too if they're in the gcc build directory
wget ftp://gcc.gnu.org/pub/gcc/releases/gcc-${GCC_VERSION}/gcc-${GCC_VERSION}.ta
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/gmp-${GMP_VERSION}.tar.bz2
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/mpfr-${MPFR_VERSION}.tar.bz2
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/mpc-${MPC_VERSION}.tar.gz
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/isl-${ISL_VERSION}.tar.bz2
wget ftp://gcc.gnu.org/pub/gcc/infrastructure/cloog-${CLOOG_VERSION}.tar.gz
wget ftp://ftp.gnu.org/gnu/binutils/binutils-${BINUTILS_VERSION}.tar.bz2
tar xf gcc-${GCC_VERSION}.tar.bz2
tar xf gmp-${GMP_VERSION}.tar.bz2
tar xf mpfr-${MPFR_VERSION}.tar.bz2
tar xf mpc-${MPC_VERSION}.tar.gz
tar xf isl-${ISL_VERSION}.tar.bz2
tar xf cloog-${CLOOG_VERSION}.tar.gz
tar xf binutils-${BINUTILS_VERSION}.tar.bz2
cd gcc-${GCC_VERSION}
ln -s ../gmp-${GMP_VERSION} gmp
ln -s ../mpfr-${MPFR_VERSION} mpfr
ln -s ../mpc-${MPC_VERSION} mpc
```

```
# GCC
BINUTILS_VERSION="2.23.2"
GCC_VERSION="4.8.5"
CLOOG_VERSION="0.18.0"
GMP_VERSION="4.3.2"
ISL_VERSION="0.11.1"
MPC_VERSION="0.8.1"
MPFR_VERSION="2.4.2"

# Development deps
FLEX_VERSION="2.5.39"
CCACHE_VERSION="3.3.3"
PYTHON_VERSION="2.7.12"
BOOST_VERSION="1.55.0"
CMAKE_VERSION="3.4.3"
OPENBLAS_VERSION="0.2.18"

# Astro sw
CFITSIO_VERSION="3.380"
WCS_VERSION="5.15"
CASACORE_VERSION="2.1.0"
PYTHON_CASACORE_VERSION="2.1.2"
CASAREST_VERSION="1.4.1"
AOFLAGGER_VERSION="2.8.0"
LOFAR_VERSION="2.20.1"
WSCLEAN_VERSION="2.2.1"
CHGCENTRE_VERSION="1.4"
PARSELTONGUE_VERSION="2.3"
AIPS_VERSION="31DEC16"
CASA_VERSION="4.7.0-1-el6"
~
```

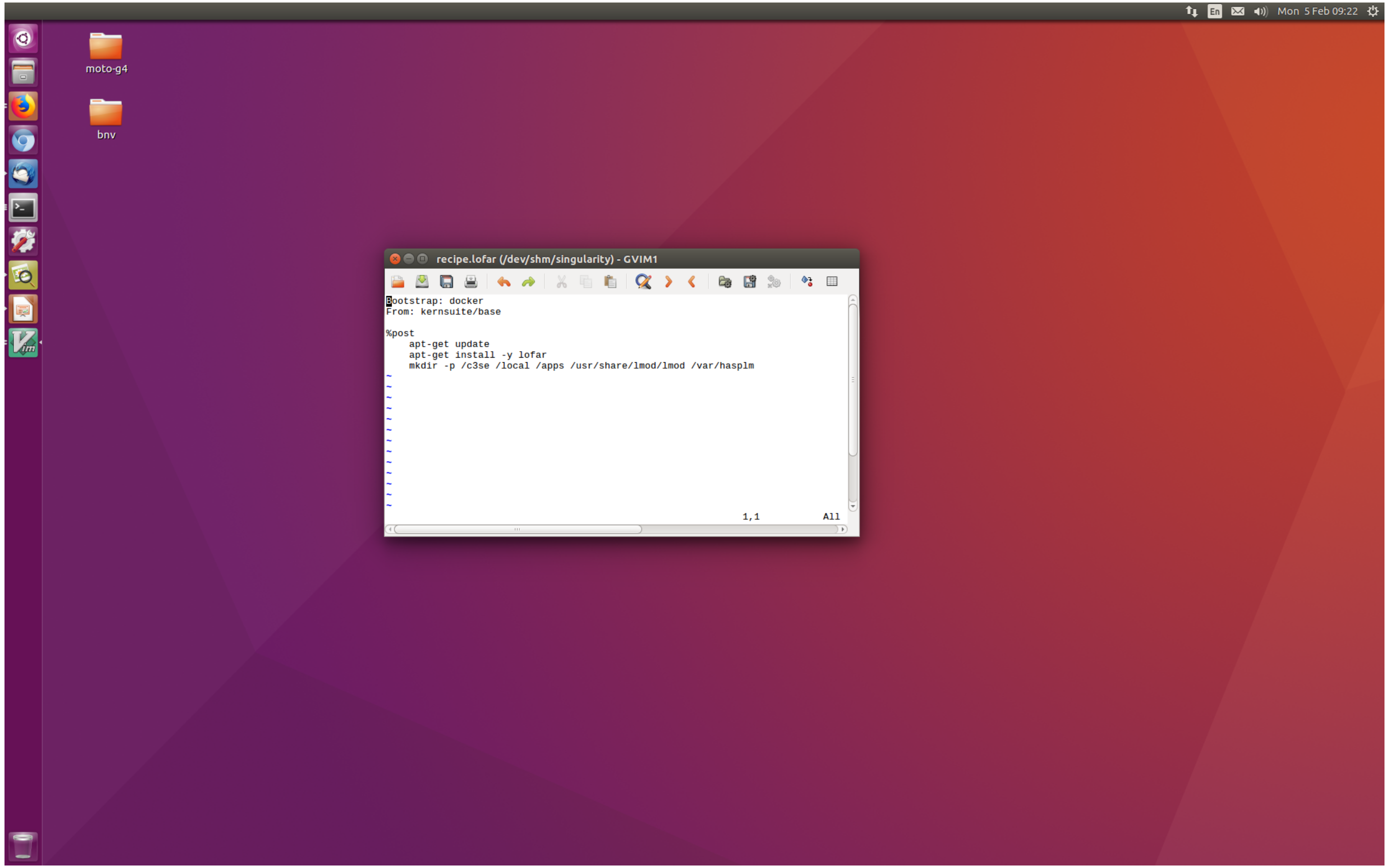
```
#!/bin/bash

# Tested on CentOS/RHEL 6

set -e
. 00_conf
. 00_versions
. 00_paths
mkdir -p $SW_DIR $SRC_DIR

# CASAREST / LWIMAGER
cd $SRC_DIR
git clone https://github.com/casacore/casarest casarest-${CASAREST_VERSION}
cd casarest-${CASAREST_VERSION}
git checkout -b v${CASAREST_VERSION} v${CASAREST_VERSION}
cd $SRC_DIR
mkdir casarest-${CASAREST_VERSION}-build
cd casarest-${CASAREST_VERSION}-build
${SW_DIR}/cmake-${CMAKE_VERSION}-Linux-x86_64/bin/cmake \
      -DCMAKE_INSTALL_PREFIX=${SW_DIR}/casarest-${CASAREST_VERSION} \
      -DCMAKE_PREFIX_PATH=${SW_DIR}/casacore-${CASACORE_VERSION} \
      -DCASACORE_INCLUDE_DIR=${SW_DIR}/casacore-${CASACORE_VERSION}/include \
      -DCMAKE_LIBRARY_PATH=${SW_DIR}/casacore-${CASACORE_VERSION}/lib \
      -DCFITSIO_LIBRARY=${SW_DIR}/cfitsio-${CFITSIO_VERSION}/lib/libcfitsio.a \
      -DCFITSIO_INCLUDE_DIR=${SW_DIR}/cfitsio-${CFITSIO_VERSION}/include \
      -DWCSLIB_LIBRARY=${SW_DIR}/wcslib-${WCS_VERSION}/lib/libwcs.a \
      -DWCSLIB_INCLUDE_DIR=${SW_DIR}/wcslib-${WCS_VERSION}/include \
      -DBoost_INCLUDE_DIR=${SW_DIR}/boost-${BOOST_VERSION}/include \
      -DBoost_NO_BOOST_CMAKE=YES \
      -DCMAKE_BUILD_TYPE=release \
      ../casarest-${CASAREST_VERSION}
make -j ${BUILD_JOBS} install
#      -DCMAKE_C_COMPILER=${SW_DIR}/gcc-${GCC_VERSION}/bin/gcc \
~
```

After



```
Bootstrap: docker
From: kernsuite/base

%post
    apt-get update
    apt-get install -y lofar
    mkdir -p /c3se /local /apps /usr/share/lmod/lmod /var/hasplm
```

Introduction

2 ways to think of containers:

- As an application with all of its dependencies
- As an OS running alongside the native OS

A container may be limited in its use of system resources

- It can be restricted to a subset of cores / memory

Implementations

Singularity

- HPC Science & Engineering
- Containers used like any other application
- Native network comms
- Supported by many HPC centres

Docker

- PaaS / Micro Services
- Daemon runs an manages containers
- NAT
- Run on your own [cloud] resources

Singularity

- You need admin right to create containers (mostly)
  - Do this on your own machine
- Copy container image to remote host
- Run container as a normal user

Singularity commands

- build
  - --sandbox
  - Can put build specs in a text file
- exec
  - --writable
- run
  - Shortcut for most useful command in the container (defaults to bash)

Example

CHALMERS
UNIVERSITY OF TECHNOLOGY

If you *really* need gvim on hebbe

```
sudo singularity build gvim recipe.gvim
scp gvim hebbe:
ssh hebbe
./gvim
# Hooray!
```

Real world examples

```
sudo singularity build --sandbox ubuntu/ docker://kernsuite/base

sudo singularity exec --writable ubuntu/ bash

        (do stuff)

sudo singularity build ubuntu.simg ubuntu/
```

Part 2

Review: Containers

- A Container Image:
  - A file system that will be used as /
- A Running Container:
  - An application running with the image as its file system

Review: Container Image

- The Container Image (file system) can be very minimal
  - just the application, its libraries and a few low level files
- More common is that it will contain a base OS distribution
  - This is just because it’s an easy way of making container images
  - Most of that stuff isn’t needed

Containers: Process Isolation

- Linux kernel can isolate the container for rest of the OS
  - This is the default in docker
  - “instance” commands in Singularity
  - Multiple applications can be run in the same container

CHALMERS
UNIVERSITY OF TECHNOLOGY

Inside the Container Image

- A typical linux filesystem
- Optionally some extra bits for convenience
  - “run” script, “exec” script, “environment” file, meta data file (Creator, date, etc)
- Default format is squashfs
  - Can be a plain directory structure
  - Can export an Image as a tar file

- You can also import a tar file as a container
  - But you don’t really have to
  - Can just untar and use the directory as the container image

Archive Container Images

- Upload to singularity-hub or docker-hub
  - Run them on future versions of linux and reproduce results
  - Linux (the kernel) takes backwards compatibility very seriously
  - (If singularity, singularity-hub is still around :-P)

Singularity Summary

- Very straigh forward and versatile
- Any idea I try seems to work
- GPU acceleration is a exception
  - Often possible but a bit messy
  - You’re stuck with the host systems kernel module
  - Nvidia do not maintain backwards compatibility

More examples

AVANCEZ

1829

CHALMERS

UNIVERSITY OF TECHNOLOGY