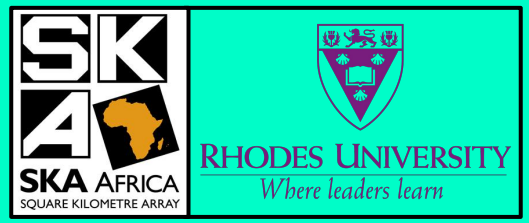
# Towards System Agnostic Reduction/Synthesis Pipelines

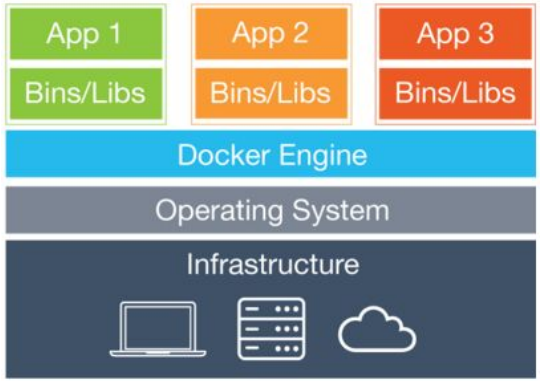



Sphesihle Makhathini, PHISCC 2016, SAAO Auditorium, Cape Town, South Africa

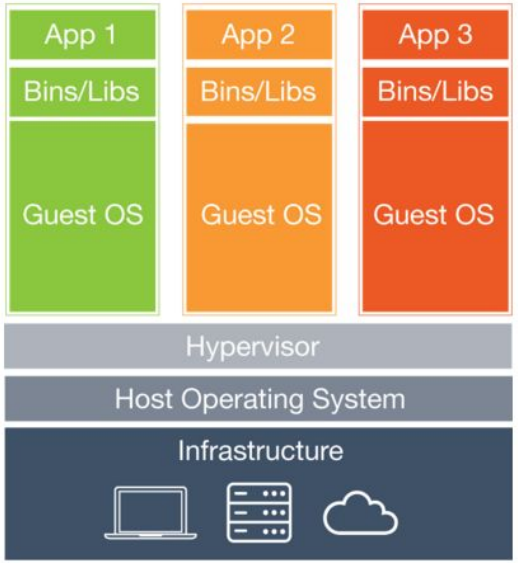
# Container Technology

- Wrap applications in a complete and isolated filesystem
  - Along with libraries and system tool
- Runs on any system with a linux kernel
  - Cloud, Cluster, Laptop, VM
- **Lighter** than a VM
- Requires root privileges
  - Potential security risks
  - Ongoing efforts to remedy this

**Container**




**Virtual Machine**



# Docker

Build Ship Run

- Uses containers to provide isolated environments for developing, building, and distributing applications
- Docker Images (Build Component):
  - Container templates. Containers created from these
  - Hosted freely on an online registry (**docker hub**)

- ❖ Overcome bad coding practices of astronomers
- ❖ Robust (runs on all systems with linux kernel)
  - ➢ Cloud, cluster, laptop
  - ➢ Repeatable, scalable
- ❖ Seamlessly ship pipelines across platforms

MeqTrees Docker Image

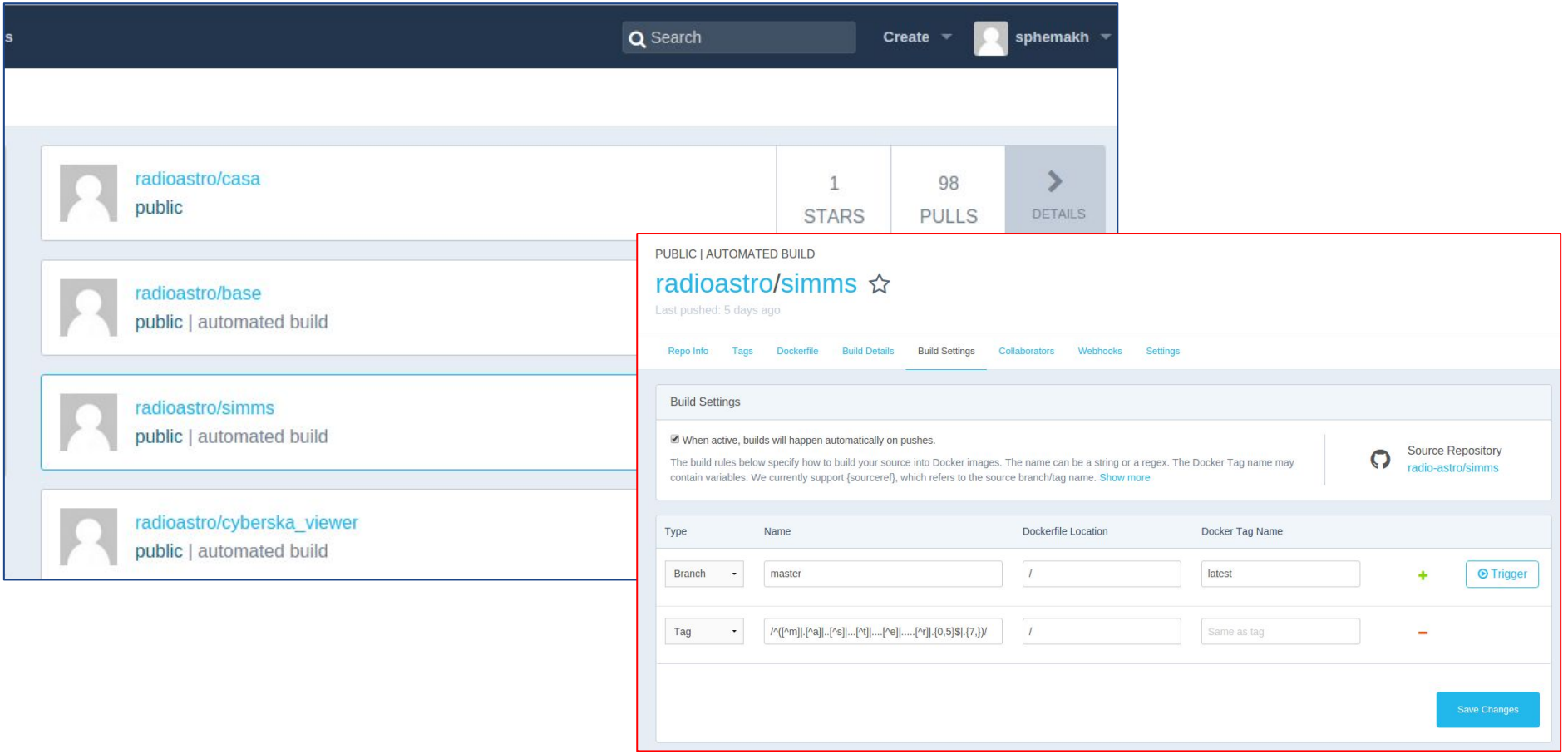
```
FROM radioastro/base:0.2

MAINTAINER gijsmolenaar@gmail.com

RUN apt-get update && \
    apt-get install -y \
        meqtrees=1.3.3-3trusty \
        time \
        && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

CMD /usr/bin/meqtree-pipeliner.py
```

# DOCKER HUB


Search
Create
sphemakh
radioastro/casa
public
1
STARS
98
PULLS
DETAILS
radioastro/base
public | automated build
radioastro/simms
public | automated build
radioastro/cyberska_viewer
public | automated build
PUBLIC | AUTOMATED BUILD
radioastro/simms
Last pushed: 5 days ago
Repo Info
Tags
Dockerfile
Build Details
Build Settings
Collaborators
Webhooks
Settings
Build Settings
When active, builds will happen automatically on pushes.
The build rules below specify how to build your source into Docker images. The name can be a string or a regex. The Docker Tag name may contain variables. We currently support {sourceref}, which refers to the source branch/tag name. Show more
Source Repository
radio-astro/simms
Type
Name
Dockerfile Location
Docker Tag Name
Branch
master
/
latest
Trigger
Tag
/^([^m]|.[^a]|..[^s]|...[^t]|....[^e]|.....[^r]|.{0,5}$|.{7,})/
/
Same as tag
Save Changes

radio-astro / **simms**

Unwatch 3 | Star 0 | Fork 1

Code | Issues 1 | Pull requests 1 | Wiki | Pulse | Graphs | Settings

Creates empty measurement sets using the the CASA simulate tool. — Edit

**92** commits | **2** branches | **3** releases | **2** contributors

Branch: **master** | New pull request | New file | Find file | SSH | git@github.com:radio-astr | Download ZIP

**SpheMakh** Build simms image with latest casa

| | |
|---|---|
| simms | Update version number. Prepare for new release |
| tests | fix test |
| .gitignore | project cleanup, prepare for 0.6.0 |
| .travis.yml | project cleanup, prepare for 0.6.0 |
| CHANGES.md | Create CHANGES.md |
| Dockerfile | Build simms image with latest casa |
| LICENSE | add licence |
| MANIFEST.in | project cleanup, prepare for 0.6.0 |
| Makefile | add docker files |

18 lines (12 sloc) | 329 Bytes

```
1  FROM radioastro/casa
2
3  MAINTAINER gijsmolenaar@gmail.com
4
5  RUN apt-get update && \
6      apt-get install -y \
7          python-pip \
8          python-casacore \
9          python-numpy \
10         && \
11     rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
12
13 ADD . /tmp/simms
14
15 RUN cd /tmp/simms && python setup.py install
16
17 CMD /usr/local/bin/simms
```


- Uses Docker to provide a system independent* package for synthesizing radio interferometry data.
- Designed to handle large image cubes
- Easy to install/deploy

*May have I/O issues with Mac OS

# Data Synthesis

$$\mathbf{V}_{pq} = \mathbf{G}_p \left( \sum_s \mathbf{E}_{sp} \mathbf{X}_{spq} \mathbf{E}_{sq}^H \right) \mathbf{G}_q^H$$

# Deployment

```
$ git clone https://github.com/SpheMakh/HI-Inator.git

$ cd HI-Inator

$ make build # Download Docker images
```

# Repository

- **src** : Source code (simulation and imaging scripts)
- **input** : Place input sky model here
- **output** : All output will be dumped here

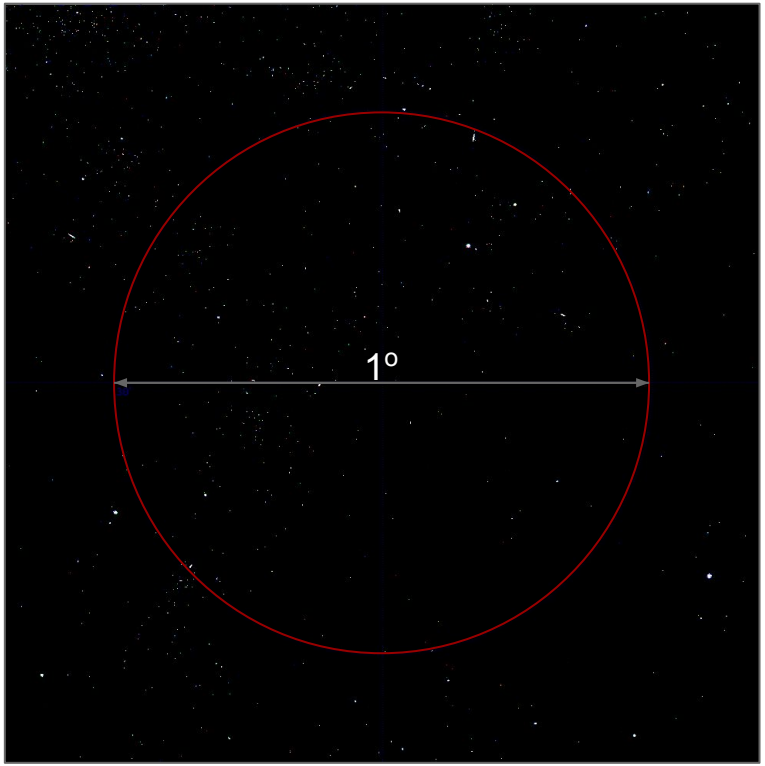
# RUN

- Configure simulation parameters (json file)

```
"sim_id" :"example_sim",
"sky_model": "example.fits",
"component_model": null,
"observatory" :"meerkat",
"direction":"J2000,0deg,-30deg",
"scanlength": 4,
"synthesis": 30,
"dtime": 10,
"npix": 1024,
```

  - Run
    - $ make run config=<config file>
  - Output products
    - Visibility datasets (CASA tables)
    - FITS images

# Example




Moment zero map of model.
Courtesy of Ed Elson

**Telescope Simulation**

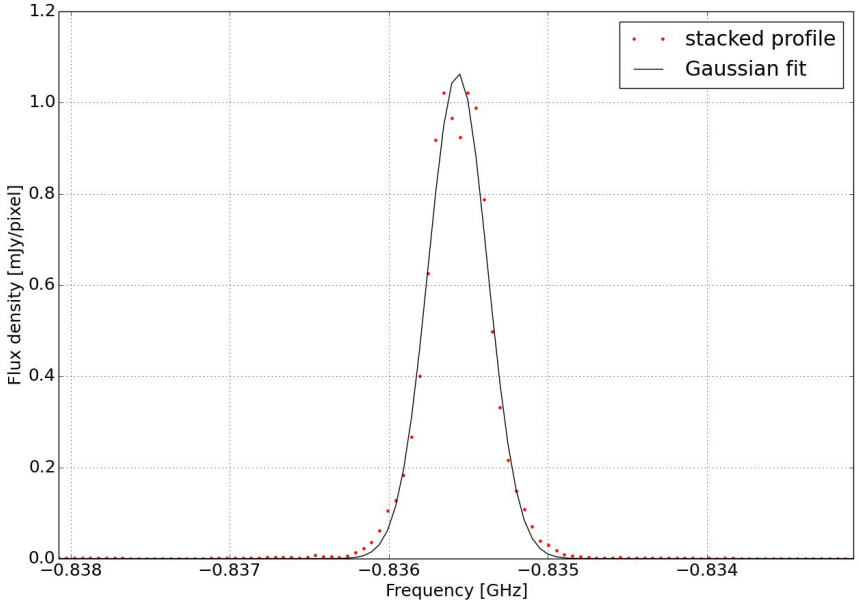
Telescope: MeerKAT

Synthesis: 4000 Hrs

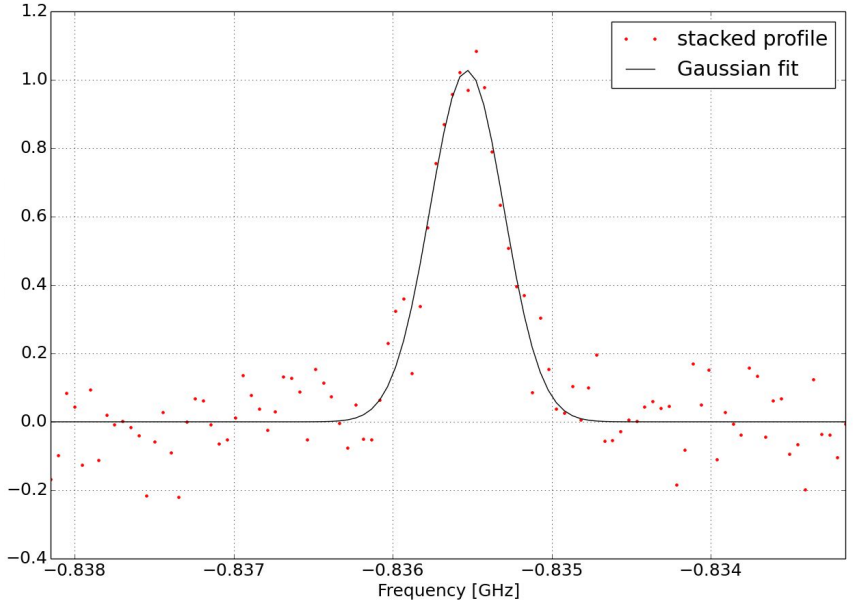
Freq: @ 840MHz, 50kHz BW

# Stacked signal from simulation

Sky model




MeerKAT Simulation



# The bigger picture: Stimela

- Framework for System agnostic data reduction and synthesis
- Python interface to legacy and novel software
  - Don’t worry about installing/building them
- High level of reproducibility
  - Remembers the docker images used for given recipe
  - Re-run recipe from log file using logged docker images.
- Easy to identify/fix broken recipe steps
  - Each software package runs in its own environment
- **FLEXIBLE**

# MODULES (CABS)

simms: Create empty MS

simulator: Simulate visibilities (parametric model)

predict: Simulate visibilities (FITS image)

autoflagger: Automatic flagger

flagms: Manual flagger

calibrator: Self-Cal visibilities (DI + DD)

imager: Image MS (casa, lwimager, wsclean)

sourcery: Source finding and characterization (continuum)

+++

Output

configuration file

Input

```
"msname" : null,
"label" : null,
"tel" : "meerkat",
"pos" : "MeerKAT64_ANTENNAS",
"pos_type" : "casa",
"ra" : "0h0m0s",
"dec" : "-30d0m0s",
"synthesis" : 4,
```

# Sample script: Data synthesis

```
from stimela import Recipe

INPUT = "input"
OUTPUT = "output"
MSDIR = "msdir"

MS = "meerkat_simulation_example.ms"
LSM = "nvss1deg.lsm.html"


# start oterera instance
pipeline = Recipe("Simulation Example", ms_dir=MSDIR)

## 1: Make empty MS
simms_dict = {}
simms_dict["msname"] = MS
simms_dict["telescope"] = "meerkat"
simms_dict["synthesis"] = 1
simms_dict["direction"] = "J2000,90deg,-45deg"
simms_dict["dtime"] = 10
simms_dict["freq0"] = "750MHz"
simms_dict["dfreq"] = "1MHz"
simms_dict["nchan"] = 10
pipeline.add("cab/simms", "simms_example", simms_dict, input=INPUT, output=OUTPUT,
             label="Creating MS")
```

```
## 2: Simulate visibilities into it
simulator_dict = {}
simulator_dict["msname"] = MS
simulator_dict["addnoise"] = True
simulator_dict["sefd"] = 831
simulator_dict["skymodel"] = LSM
pipeline.add("cab/simulator", "simulator_example", simulator_dict, input=INPUT, output=OUTPUT,
             label="Simulating visibilities")


## 3: Image
# Make things a bit interesting by imaging with different weights
imager_dict = {}
imager_dict["weight"] = "briggs"
imager_dict["clean_iterations"] = 1000
briggs_robust = 2, 0, -2
prefix = "stimela-example"

for i, robust in enumerate(briggs_robust):
    imager_dict["msname"] = MS
    imager_dict["robust"] = robust
    imager_dict["imageprefix"] = "%s_robust-%d"%(prefix, i)
    pipeline.add("cab/lwimager", "imager_example_%d"%i, imager_dict, input=INPUT, output=OUTPUT,
                 label="Imaging MS, robust=%f"%robust)

pipeline.run()
```

# Summary

- Docker is very powerful tech
- Ensures that your application runs on the same environment it was developed and tested in
- Stimela offers a system agnostic scripting framework
  - Seamlessly combine various packages
  - Easy to plug in custom tasks (docker images)
- Combine with cloud computing => World domination

# KEEP CALM AND DOCKERIZE