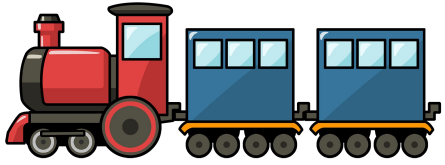
Stimela2

Scalable and fully reproducible data reduction workflows, from on-premises to HPC to cloud

Oleg Smirnov (Rhodes University & SARAO)
+**Sphesihle Makhathini** (Wits U. & Rhodes), **Simon Perkins** (SARAO),
Jonathan Kenyon (Rhodes), **Landman Bester** (SARAO & Rhodes),

https://ratt.center
RATT
Centre For Radio Astronomy Techniques & Technologies • Rhodes University



NRF
National Research Foundation

SARAO
South African Radio Astronomy Observatory

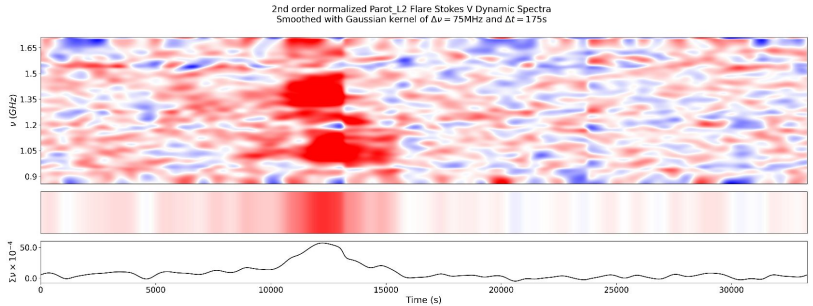
Rationale / Squaring The Circle

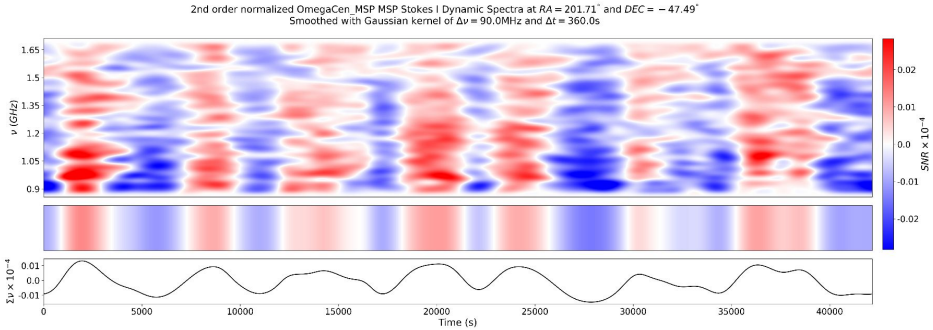
- I want to **simply** script my workflows, but I want them to **scale up** and **deploy** on funky clusters
- I want to try **dodgy** stuff quickly, but I want **portability** and **reproducibility**
- I want to use containers, but I don't want to be rebuilding images all the time (and oh wait I *really* want to try this wsclean bugfix right now...)
- I want for-loops and conditionals
- I want documentation but I don't like writing it
- I want my errors caught up front but I don't want to be straightjacketed
- I want to generate tons of data products, but I don't want to think too hard about naming them
- I want to log everything, but I don't want to be buried in logfiles

https://stimela.readthedocs.io

Case Study: TRON (& Breifast)

- Pipeline to mine archival MeerKAT data for image-plane transients & variables
- TRON starts with a (calibated) MeerKAT MS, makes snapshot images at raw time cadence, convolves to different timescales, analyses & detects (breifast), extracts lightcurves and dynamic spectra, generates catalogs, etc. etc.
- MSs and image cubes are in ~Tb regime
  - unfolding the time and frequency axis quickly explodes data product sizes
  - data “anti-reduction”





Cab: defines an atomic task

- a binary package (e.g. wsclean)
- a CASA task
- a Python function or code snippet
- backed by a versioned container image
  ...but can be natively installed, too
- has inputs and outputs defined by a YaML **schema**

Recipe: defines a sequence of *steps*

- has overall inputs and outputs defined by a YaML schema
- (can be defined as a loop/scatter construct)

Step: a cab or a recipe invocation

- using a set of supplied parameters which are mapped to inputs and outputs according to the schema
- (can be (conditionally) skipped)

Package: a collection of cabs and/or recipes

- just a bunch of YaML documents
  - backed by an image registry
- we use pip/PyPI to manage packages
- **cult-cargo:** our collection of standard cabs & associated images (on quay.io)
  https://github.com/caracal-pipeline/cult-cargo

`$ stimela doc file.yml [file2.yml...] what`

- loads specified YaML and prints documentation
- a CASA task
- a Python function or code snippet
- backed by a versioned container image
  ...but can be natively installed, too
- has inputs and outputs defined by a YaML **schema**

`$ stimela run file.yml [file2.yml...] recipe foo=bar`

- loads specified YaML and executes recipe (or runs a standalone cab)
- different execution backends supported
  - **native** (local software installs)
  - **singularity** (Singularity/Apptainer), with images automatically downloaded and built (on-demand or in advance, using the `stimela build` command)
  - **kube**: each step is scheduled as a k8s pod
  - **slurm**: each step is scheduled via srun
- Conditional skips can make for Makefile-like behaviour
- Rudimentary profiling available

Cab definitions: how to put a wildcat in a box

Cabs can wrap existing (command-line) tools:

- Registry/image/version information
- Command to run (natively or in in container)
- Formal inputs and outputs (the **schema**)
  - using Python typing syntax

```
ms: List[MS]
weight: Union[str, Tuple[str, float]]
```

- Policies describing how these map onto CLI

Cabs can wrap Python functions (CASA tasks):

- Registry/image/version information
- Module and function
- Schema

Cabs can be inline Python code.

Existing third-party tools (e.g. wsclean, DDFacet) need a full cab definition matching their CLI. (Tedious...)

New Python apps can get a CLI “for free”.

- Standard YaML has very limited structural features
- Stimela provides an extra layer of functionality on top of YaML that promotes composability and modularity
- Everything is one global nested namespace

```
cabs:
  wsclean:
    ...
opts:
  backend:
    ...
recipe:
  ...
lib:
  ...
```

- Successive YaML documents are **merged** in
- An **_include** directive merges in other YaML (which can then be further augmented)

```
_include: (cultcargo)wsclean.yml
```

- A **_use** directive inserts previously defined content (which can then be further augmented)

```
_use: vars.tron.bands.L
```

- Step parameters support a formula language and {}-substitutions

```
cds: '{recipe.dirs.cubes}/cube-{recipe.htc.cadence}.zarr'
out-image: =STRIPEXT(current.cds) + '.mean.fits'
```

The use/include/merge concept has many powerful uses.

- Separating logical recipe from configs
- Layered configs
- Surgical tweaks
  - use a different version image for cab X
  - use a local install for cab Y
- Runtime configuration
  - Examples: slurm, k8s
  - Performance tuning

Stimela Using K8s On AWS