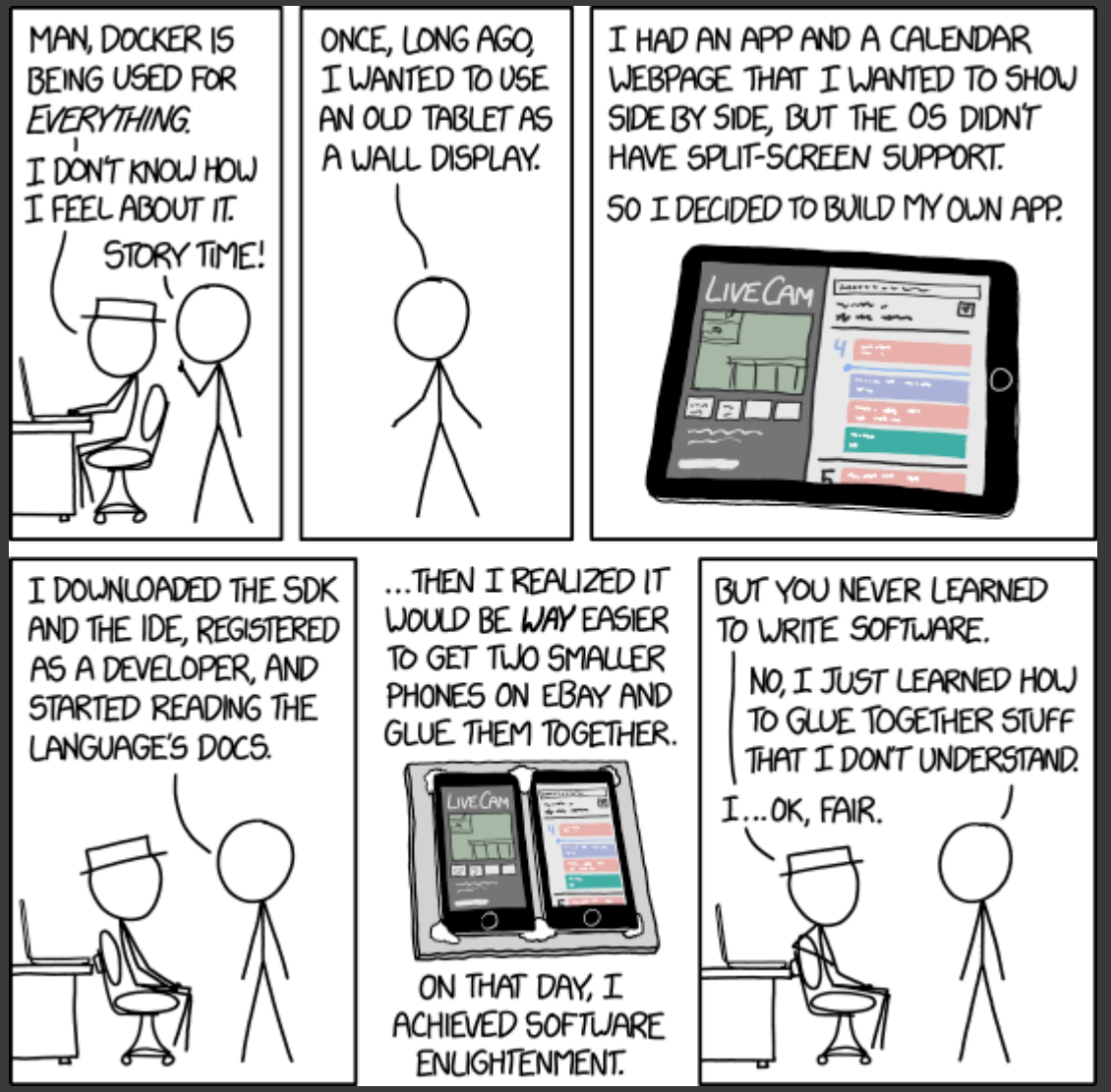
The Good, the Bad and the Ugly

A biased overview of the LOFAR interferometric software and how to use it

Frits Sweijen

Durham University

LOFAR Data School 2024



XKCD 1988

Contents

What this talk is about

LOFAR has three "observing modes"

- Direct storage: direct dump of TBB voltages for e.g. cosmic ray air showers
- Beam-formed: station-level beam-formed data before correlation for e.g. puslar studies (see Vlad's talk)
- Interferometric: data correlated between stations for e.g. imaging

I will discuss most commonly used sofware

relevant to processing interferometric data

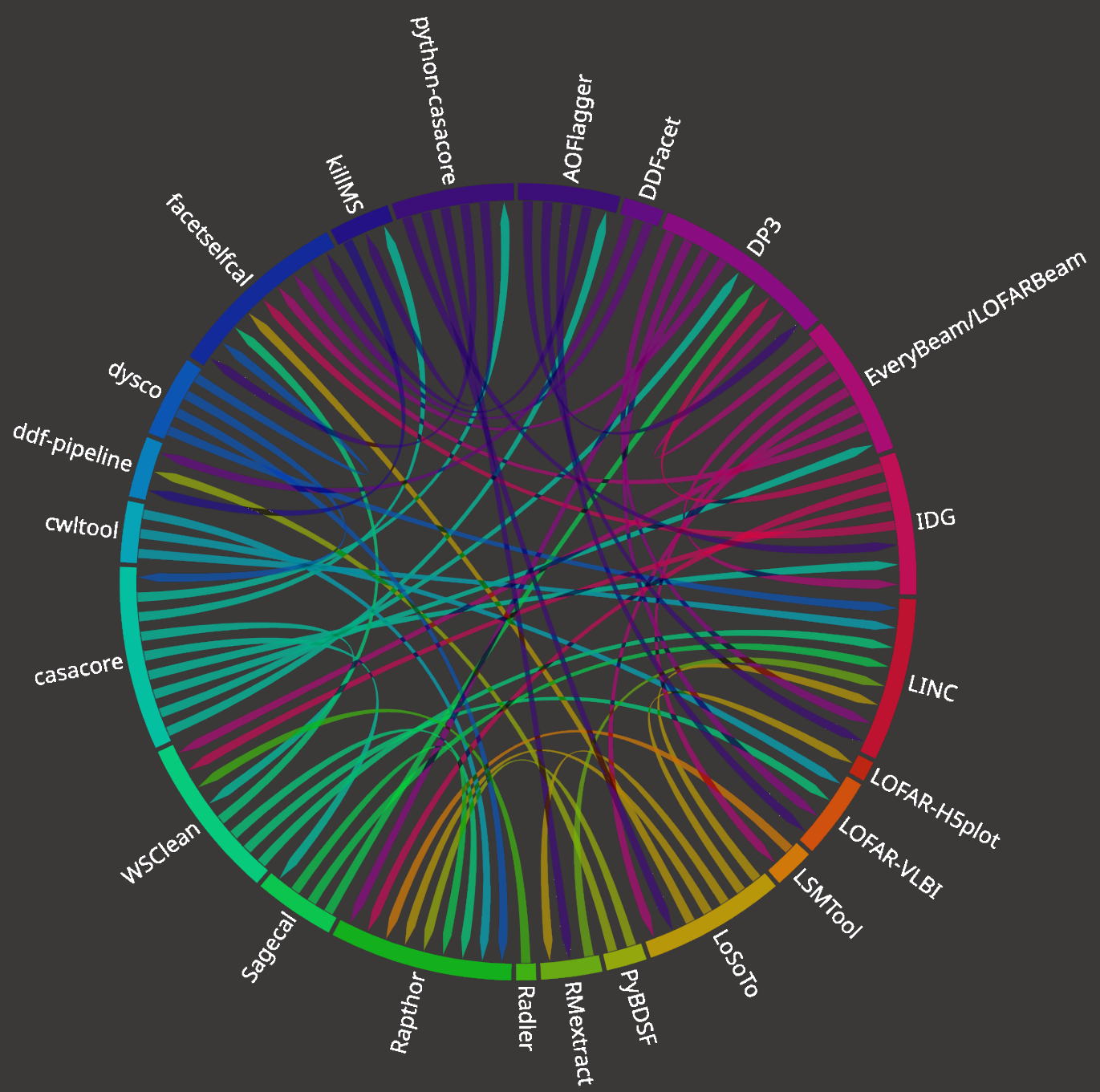
LOFAR software landscape

Many different libraries, programs, packages and pipelines

- C++ for core components
- Python for utilities/pipelines and glue
- Workflow languages for pipelines

Many inter-dependencies

Partly community-maintained, partly ASTRON-maintained



LOFAR software landscape

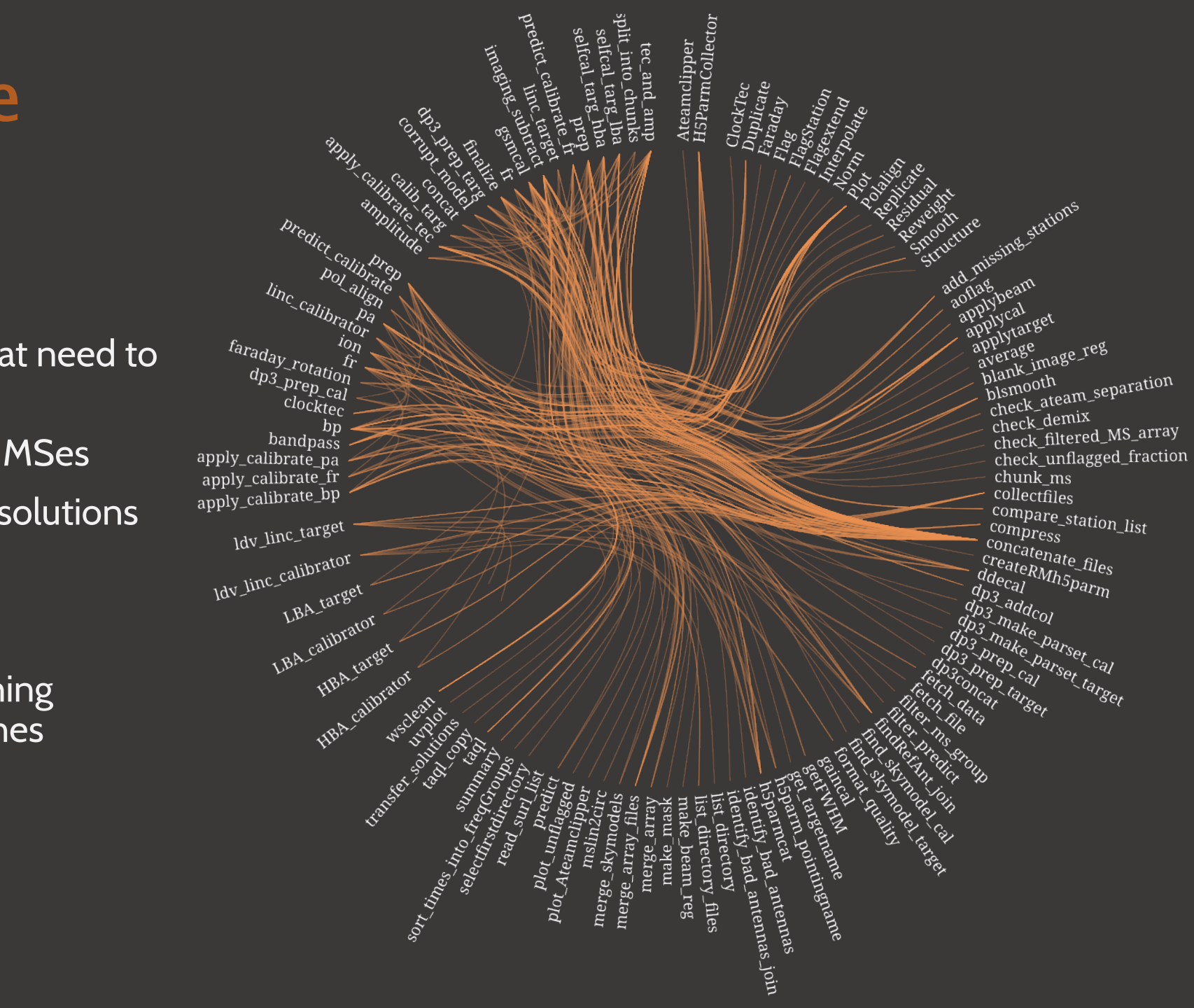
Pipelines are complicated

- Dozens of individual steps that need to talk to each other
- Dozens to hundreds of input MSes
- Many output datasets, plots, solutions

Common Workflow Language

- Declarative language for running programs and creating pipelines

COMMON
WORKFLOW
LANGUAGE



LOFAR user software landscape – pipeline components

Calibration: correct visibilities for systematics, ionospheric distortions and other corruptions

- Derive calibration solutions and correct datasets
- AOFlagger, DP3, facetselfcal, killMS, LoSoTo, RMextract

Imaging: transform data from visibilities into an image of the sky

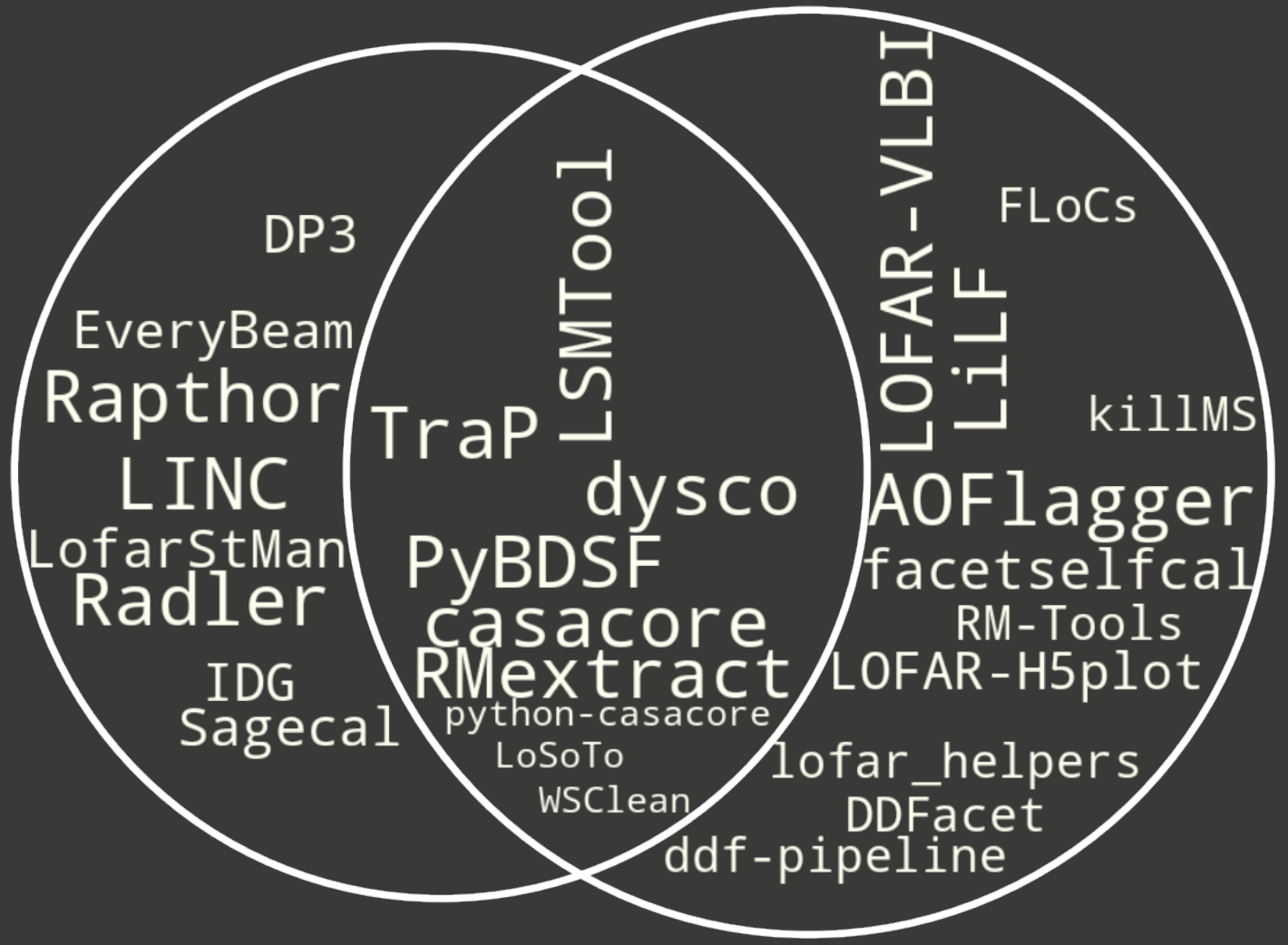
- Deconvolution of the image
- DDFacet (SSD), WSClean (IDG, wgridder)

Utility: tools that do not fit in either category or are generally useful

- Manipulation or inspection of images, MSes, calibration solutions etc.
- facetselfcal, LOFAR-H5plot, lofar_helpers, LSMTool, shadems

LOFAR software landscape

ASTRON
developed/maintained



externally
developed/maintained

LOFAR user software landscape – pipelines

LINC (de Gasperin+2019); https://linc.readthedocs.io/en/latest/index.html

ddf-pipeline (Shimwell+2017, Tasse+2018); https://github.com/mhardcastle/ddf-pipeline

VLBI-CWL (Morabito+2021); https://git.astron.nl/RD/VLBI-cwl

– Wiki coming: https://github.com/LOFAR-VLBI/lofar-vlbi-pipeline/wiki

LiLF (de Gasperin+2019,2020); https://github.com/revoltek/LiLF

facetselfcal (van Weeren+2021) https://github.com/rvweeren/lofar_facet_selfcal

TraP (Swinbank+2015); https://github.com/transientskp/tkp

LOFAR user software landscape – pipeline components

Calibration

- DP3 (ASTRON; Dijkema+2023)
- killMS (Tasse+2014)
- LoSoTo (de Gasperin)
- RMextract (Mevius 2018)

Imaging

- DDFacet (Tasse+2018)
- Radler (ASTRON)
- WSClean (Offringa+2014)

Utility/data editing

- lofar_helpers (de Jong)
- LOFAR-H5plot (Sweijen)
- LoSiTo (Rafferty, Edler, de Gasperin)
- PyBDSF (Mohan & Rafferty 2015)
- shadems (RATT-RU)

Pipelines

Pre-processing

- Averaging to 12 kHz, 1 s
- RFI excision
- Dysco compression
- Ran after observing by the observatory

Initial an direction-independent calibration

- LOFAR Initial Calibration Pipeline (LINC)
- Instrumental effects: polarisation alignment, clock drifts, bandpasses
- Direction-independent ionospheric effects

Pipelines

Direction-dependent calibration and imaging

- DDF-pipeline: direction-dependent calibration and imaging with the Dutch array using killMS + DDFacet
- Rapthor: direction-dependent calibration and imaging with the Dutch array using DP3 + WSClean
- LOFAR-VLBI: calibration and imaging with the full international array
- LiLF: direction-dependent calibration and imaging for LBA with the Dutch array using DP3 + DDFacet + WSClean

Time/frequency-domain

- DynSpecMS: time-frequency search for variability "dynamic spectrum"
- Transient Pipeline "TraP": image-plane searches of transients

Solar and space weather

- Solar imaging pipeline

Calibration

DP3 (https://dp3.readthedocs.io/en/latest/; https://git.astron.nl/RD/DP3)

– Find or apply calibration solutions (in H5parm format); can exploit frequency coherency

killMS (https://github.com/saopicc/killMS)

– Find or apply calibration solutions (in npz format); can exploit time coherency

LoSoTo (https://github.com/revoltek/losoto)

– LOFAR Solution Tool for manipulating and plotting H5parms

RMextract (https://github.com/lofar-astron/RMextract)

– extract TEC, vTEC, Earthmagnetic field and Rotation Measures from GPS and WMM data

Imaging

DDFacet (https://github.com/saopicc/DDFacet)

– Facet-based radio imager supporting full-polarisation DD corrections and smearing corrections

WSClean (https://wsclean.readthedocs.io/en/latest/; https://gitlab.com/aroffringa/wsclean)

– Imager supporting various gridders (w-stacking, IDG, wgridder) and multi-scale deconvolution

Image Domain Gridder (IDG; https://git.astron.nl/RD/idg)

– Can apply smooth screens or include the primary beam during gridding

– Use through WSClean

Utilities

LOFAR-H5plot (https://tikk3r.github.io/lofar-h5plot/; https://github.com/tikk3r/lofar-h5plot)

– Interactively explore calibration solutions in H5parm format

lofar_facet_selfcal (https://github.com/rvweeren/lofar_facet_selfcal)

– Advanced calibration tool

lofar_helpers (https://github.com/jurjen93/lofar_helpers)

– h5_merger and many other utilty scripts for dealing with MSes, FITS files, DS9 regions

LoSiTo (https://losito.readthedocs.io/en/latest/; https://github.com/darafferty/losito)

– LOFAR Simulation Tool for simulating various effects and corruptions into a dataset

shadems (https://github.com/ratt-ru/shadeMS)

– Efficiently plot data in MSes, e.g. uv coverage

PyBDSF (https://pybdsf.readthedocs.io/en/latest/; https://github.com/lofar-astron/PyBDSF)

– Source finder through gaussian fitting

Software distribution

Containers and how to use them

Containerisation

Installing large software stacks is painful

- Many dependencies
- Some libraries are hard to install

Helping people with unknown setups is hard

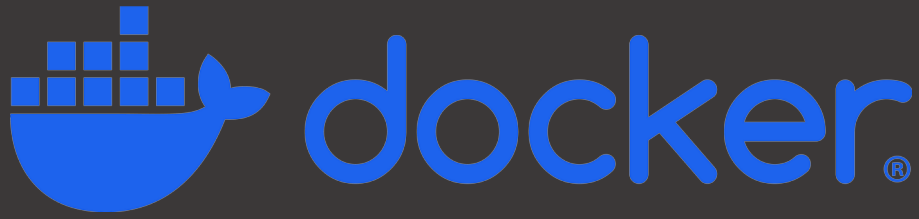
- What hardware, OS, library versions etc.?
- How was it built?

Most people just want to do science, not become sysadmin

Containerisation

Docker

- Industry standard mostly for cloud and VM
- Requires root access



Singularity

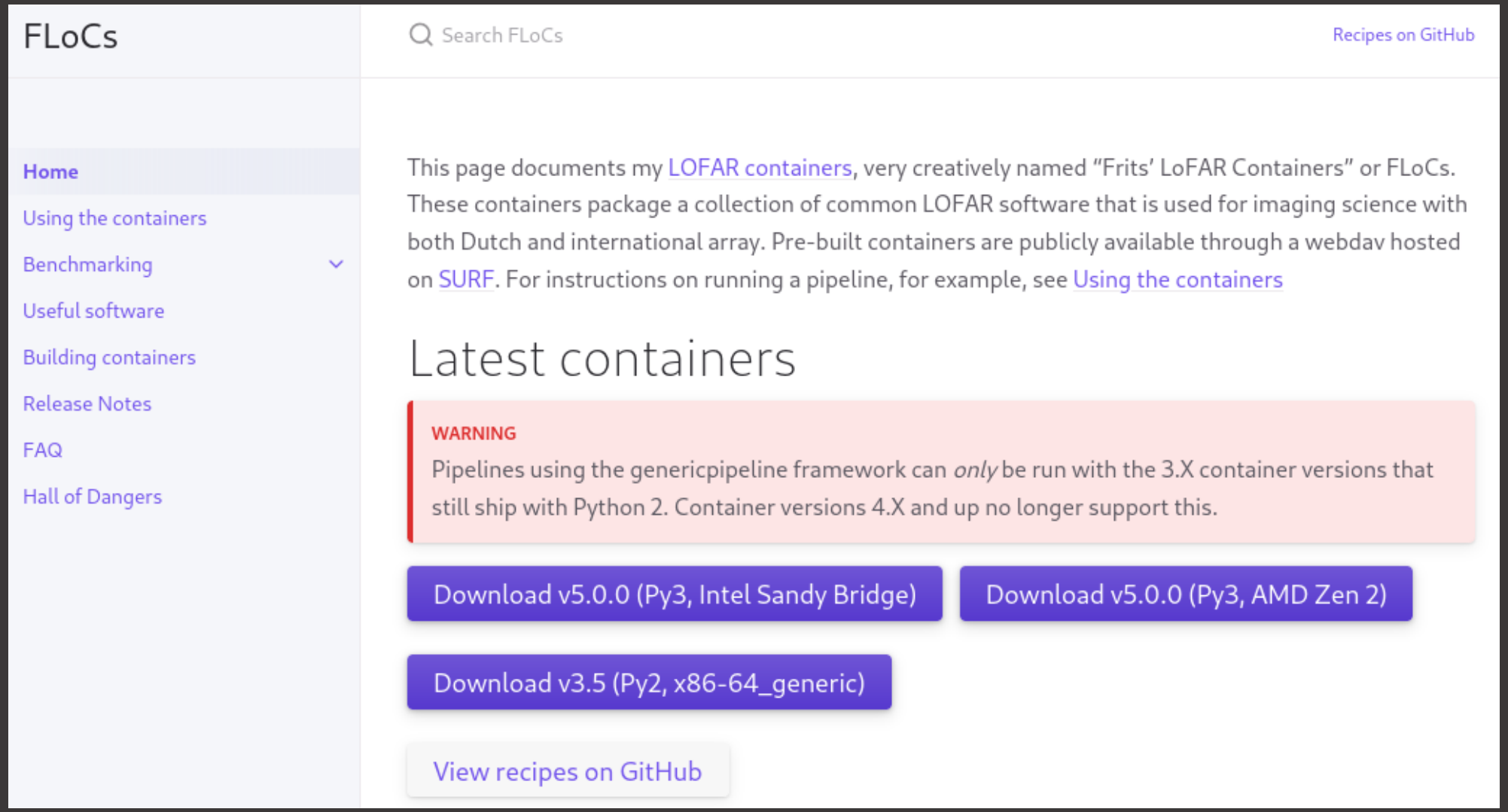
- HPC oriented favouring integration over isolation
- Single file
- No root access required
- Supports Docker

LINC and Rapthor offer Dockers

DDF-pipeline and FLoCs are Singularity

My “machine” becomes your “machine”

Frits' LOFAR Containers a.k.a. FLoCs



https://tikk3r.github.io/flocs/

Frits' LOFAR Containers a.k.a. FLoCs

Monolithic container with a large software stack

- All core components such as AOFlagger, DDFacet, DP3, killMS, LoSoTo, RMextract, WSClean
- Additional utilities such as LOFAR-H5plot (interactive inspection of H5parms), LSMTool (skymodel manipulation), PyBDSF (source finding), shadems (plotting of e.g. uv coverage)
- Support for lofar_helpers and facetselfcal
- Many Python packages for LOFAR and general included

Support for running main calibration/imaging pipelines

- LINC not included, but supported, including user-friendly wrappers handling setup for you (see https://github.com/tikk3r/flocs/tree/fedora-py3/runners)
- DDF-pipeline included
- Rapthor not included, but supported
- facetselfcal not included, but supported

Frits' LOFAR Containers a.k.a. FLoCs

Things to keep in mind

- Do not mix host and container environment
- Paths that need to be accessible should be bound explicitely (and works recursively)
- CPU architecture still matters. If you see `Illegal instruction (core dumped)` let us/me know.
- Singularity is now called Apptainer. Depending on your system's installed version the command is either `singularity` or `apptainer`

apptainer container usage

Downloading a Docker container

```
apptainer pull /path/to/container.sif docker://astronrd/linc:latest
```

Downloading FLoCs

```
wget ‘https://lofar-webdav.grid.sara.nl/software/shub_mirror/tikk3r/lofar-grid-hpccloud/intel/flocs_v5.0.0_sandybridge_sandybridge_mkl_cuda.sif’
```

Not as nice, but illustrates that Singularity containers are just big files.

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir /path/to/container.sif
```

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir /path/to/container.sif
```

launch a shell in the container

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir /path/to/container.sif
```

launch a shell in the container

make these directories accessible

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir /path/to/container.sif
```

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir/path/to/container.sif
```

launch a shell in the container

make these directories accessible

use this container

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

execute a command in the container

apptainer container usage

Interactive use

```
apptainer shell -B /my/dir,/other/dir/path/to/container.sif
```

launch a shell in the container

make these directories accessible

use this container

Non-interactive use

```
apptainer exec -B /my/dir,/other/dir /path/to/container.sif <command>
```

execute a command in the container

execute this command

Wrap up points

When calibrating, inspect your solutions carefully!

- Pipelines produce “inspection plots”
- LoSoTo for automated plotting, LOFAR-H5plot for interactive plotting

CPU architecture matters

If you see `illegal instruction`, see `#software` channel for a generic container

https://tikk3r.github.io/flocs/

https://github.com/tikk3r/flocs/

Thank you for your attention!