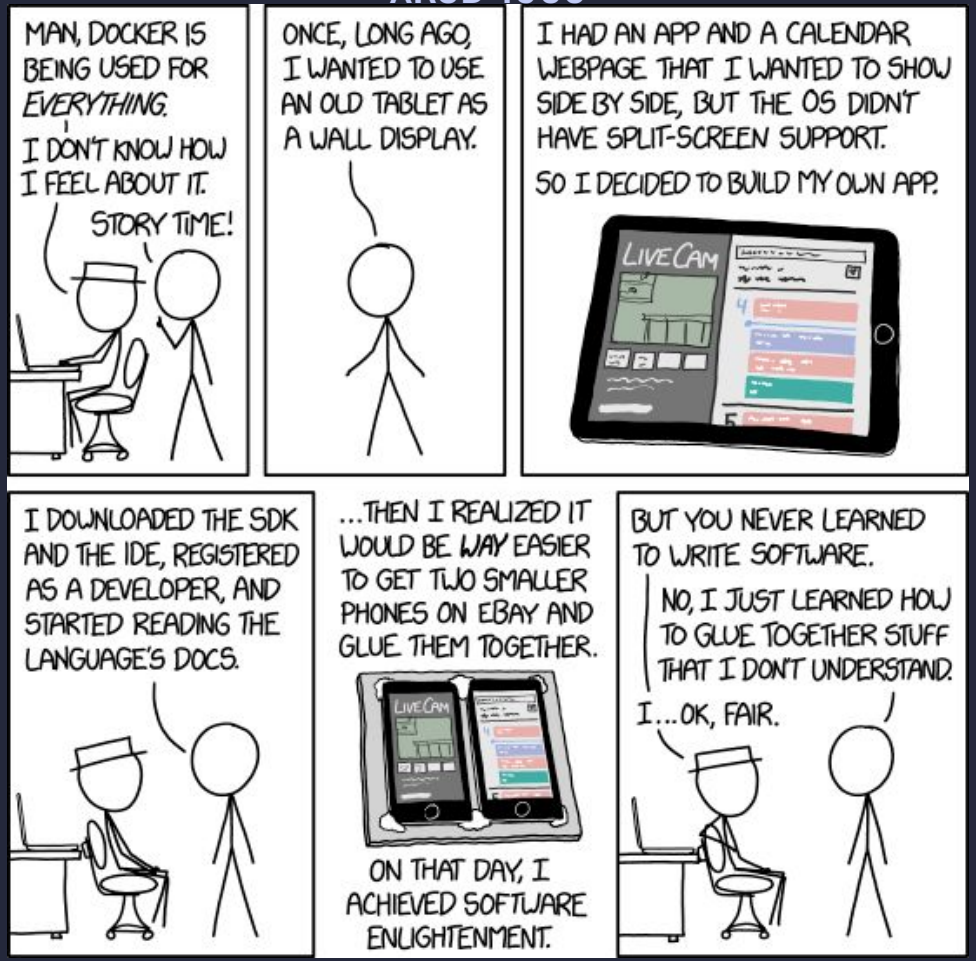
The LOFAR software landscape

A biased overview of LOFAR interferometric software

XKCD 1988



Frits Sweijen, LOFAR Data School 2026, ASTRON

Contents

General landscape

Software distribution and containerisation

Using containers this week

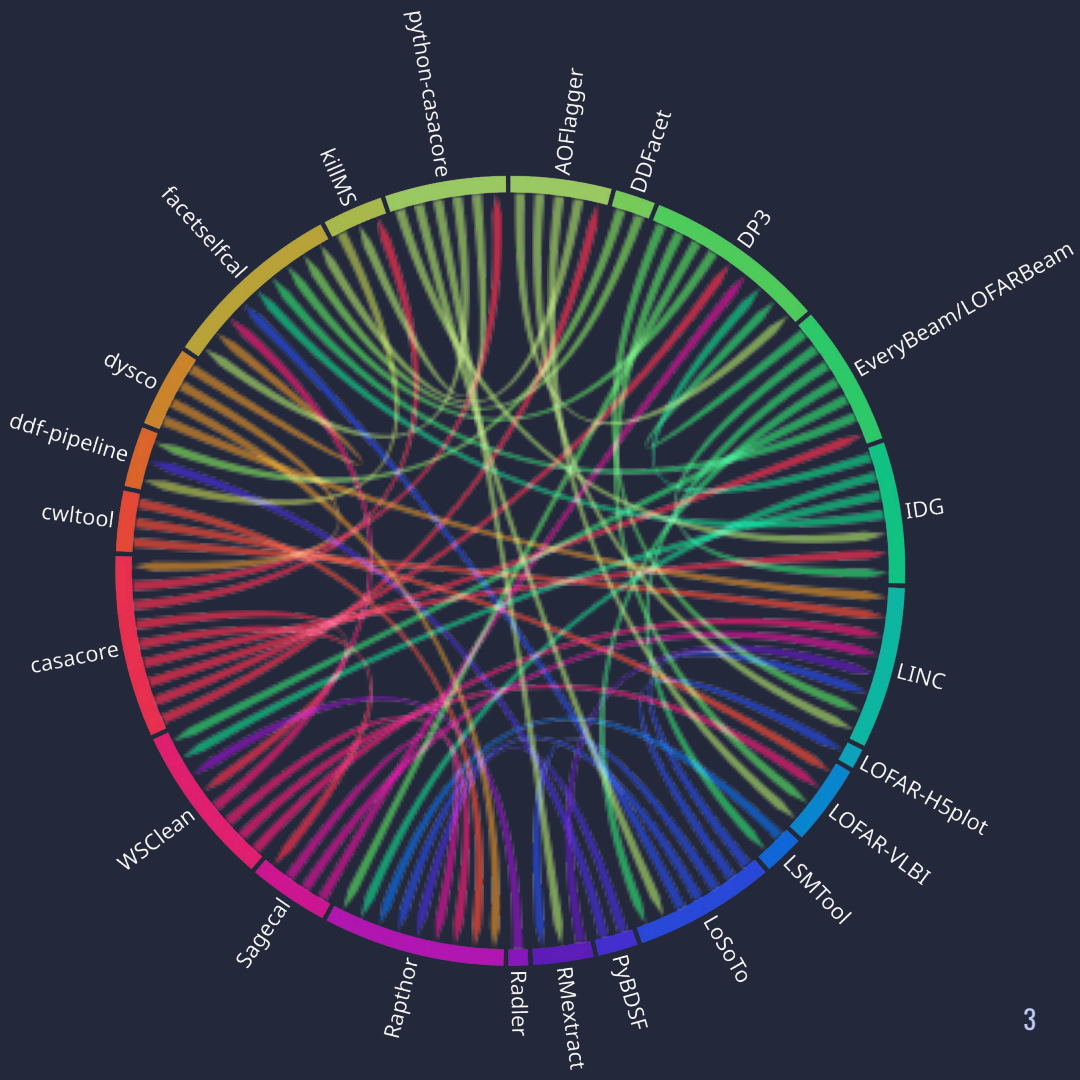
The LOFAR software

Wide array of libraries, tools and pipelines

C(++) for core components

Python for tooling and glue

Workflow languages for pipelines



The LOFAR software

Pipelines are complicated

Dozens of steps that talk to each other

Dozens to hundreds of input MSes

Many outputs: data, calibration tables, plots, images



The LOFAR software

Calibration: correct visibilities for systematics, ionospheric and other corruptions

Imaging: transform data from the visibility domain to sky image domain

Utility: tools for inspecting or manipulating data, calibration solutions etc.

The LOFAR software

Calibration

DP3 (Dijkema+2023)

killMS (Tasse+2014)

LoSoTo (de Gasperin)

Spinifex (Mevius+2025)

Imaging

DDFacet (Tasse+2018)

IDG (vd Tol+2018)

Radler

WSClean (Offringa+2014)

Utility

lofar_helpers

lofar-h5plot

PyBDSF

Rfigui

ShadeMS

etc.

Some LOFAR imaging pipelines

LINC (de Gasperin+2019) https://git.astron.nl/RD/LINC

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

Rapthor https://git.astron.nl/RD

PILOT (Morabito+2021) https://github.com/LOFAR-VLBI/pilot

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

Facetselfcal (van Weeren+2019) https://github.com/rvweeren/lofar_facet_selfcal

TraP (Swinbank+2015) https://git.astron.nl/RD/trap

Software distribution and containerisation



Containerisation

Installing large software stacks is painful

Many dependencies

Complex libraries

Debugging unknown systems is hard

What OS, hardware, versions etc.?

Sometimes you just want to do science, not fight software

Containerisation



Focuses on isolation

Daemon

Requires root access

LINC, Rapthor



Focuses on integration rather than isolation

Single file

No root access required

Can convert from Docker into Apptainer

ddf-pipeline, flocs

Containerisation



Focuses on isolation

Daemon

Requires root access

LINC, Rapthor



Focuses on integration rather than isolation

Single file

No root access required

Can convert from Docker into Apptainer

ddf-pipeline, flocs

My machine becomes your machine

With some caveats

CPU architecture still matters

Don’t mix your environment with the container’s

The container can’t access paths that you don’t allow (works recursively)

Frits' LOFAR containers (flocs)

Pre-built monolithic container with a large software stack
All core components such as AOFlagger, DDFacet, DP3, killMS, LoSoTo, Spinifex, WSClean etc.

Utilities such as h5plot, lsmtool, pybdsf, shadems, rfigui, facetselfcal etc.

Many Python packages based on community requests

Support for main calibration/imaging pipelines LINC, ddf-pipeline, facetselfcal, PILOT, Raptor.

x86 only; sorry Apple users, pre-built mac/ARM containers aren't possible (no base containers)
Try to procure a Linux cluster, group up or see if Rosetta or similar can potentially be used.

Frits' LOFAR containers (flocs)

Side note: flocs has evolved from container to "ecosystem" over the years

https://github.com/FLOCSoft

Flocs: the main container

Flocs-lta: CLI to easily search the LTA (HBA long baseline focussed atm)

Flocs-runners: simple way to run most HBA interferometric pipelines

Flocs-processing: hands-off processing via Airflow

Apptainer quickstart for later

Apptainer quickstart for later

Downloading a Docker container

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

Downloading flocs

```
wget
‘https://public.spider.surfsara.nl/project/lofarvwf/fsweijen/containers/flocs_v6.3.0_haswell_haswell.sif’
```

Apptainer quickstart for later

Interactive use (starts a shell inside the container)

```
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 quickstart for later

Interactive use (starts a shell inside the container)

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

Start a shell

Non-interactive use

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

Apptainer quickstart for later

Interactive use (starts a shell inside the container)

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

Start a shell

Make these directories available

Non-interactive use

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

Apptainer quickstart for later

Interactive use (starts a shell inside the container)

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

Start a shell — Make these directories available — Use this container

Non-interactive use

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

Apptainer quickstart for later

Interactive use (starts a shell inside the container)

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

Start a shell — Make these directories available — Use this container

Non-interactive use

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

Execute a command directly

Wrap up points

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

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