Standard Imaging Pipeline

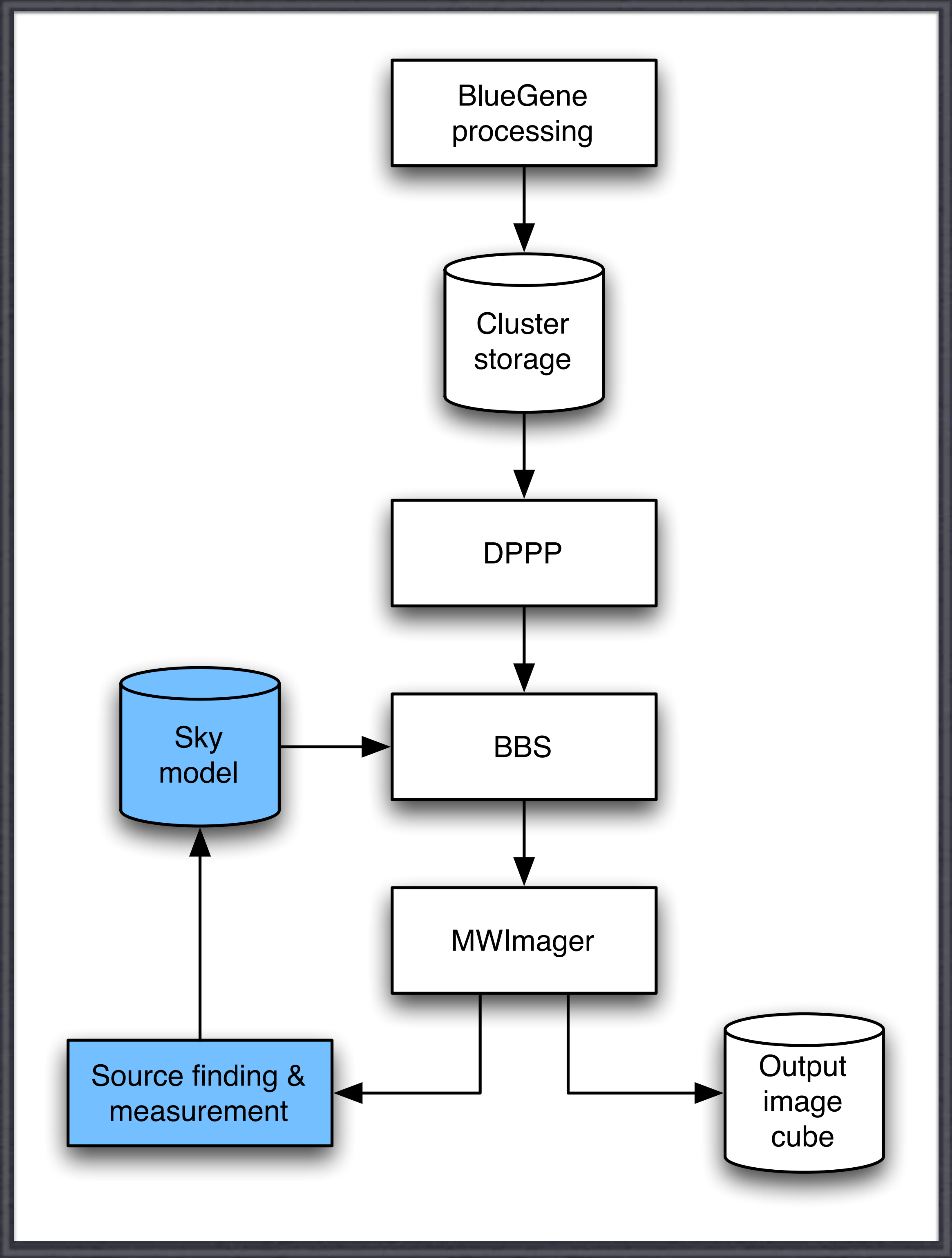
John Swinbank, University of Amsterdam
swinbank@transientskp.org

With thanks to...

- Ronald Nijboer, Ger van Diepen, Adriaan Renting, Marcel Loose, Joris van Zwieten & Evert Rol
  - For building & testing the pipeline components.
- George Heald & the Imaging Busy Week Team
  - For figuring out how to use the pipeline components.
- Teun Grit, Arno Schoenmakers, Harm Paas & Chris Broekema
  - For cluster support.
- Michael Wise & Andre Gunst
  - Motivation!

Standard Imaging Pipeline

+ assorted "housekeeping", additional flagging, etc etc.

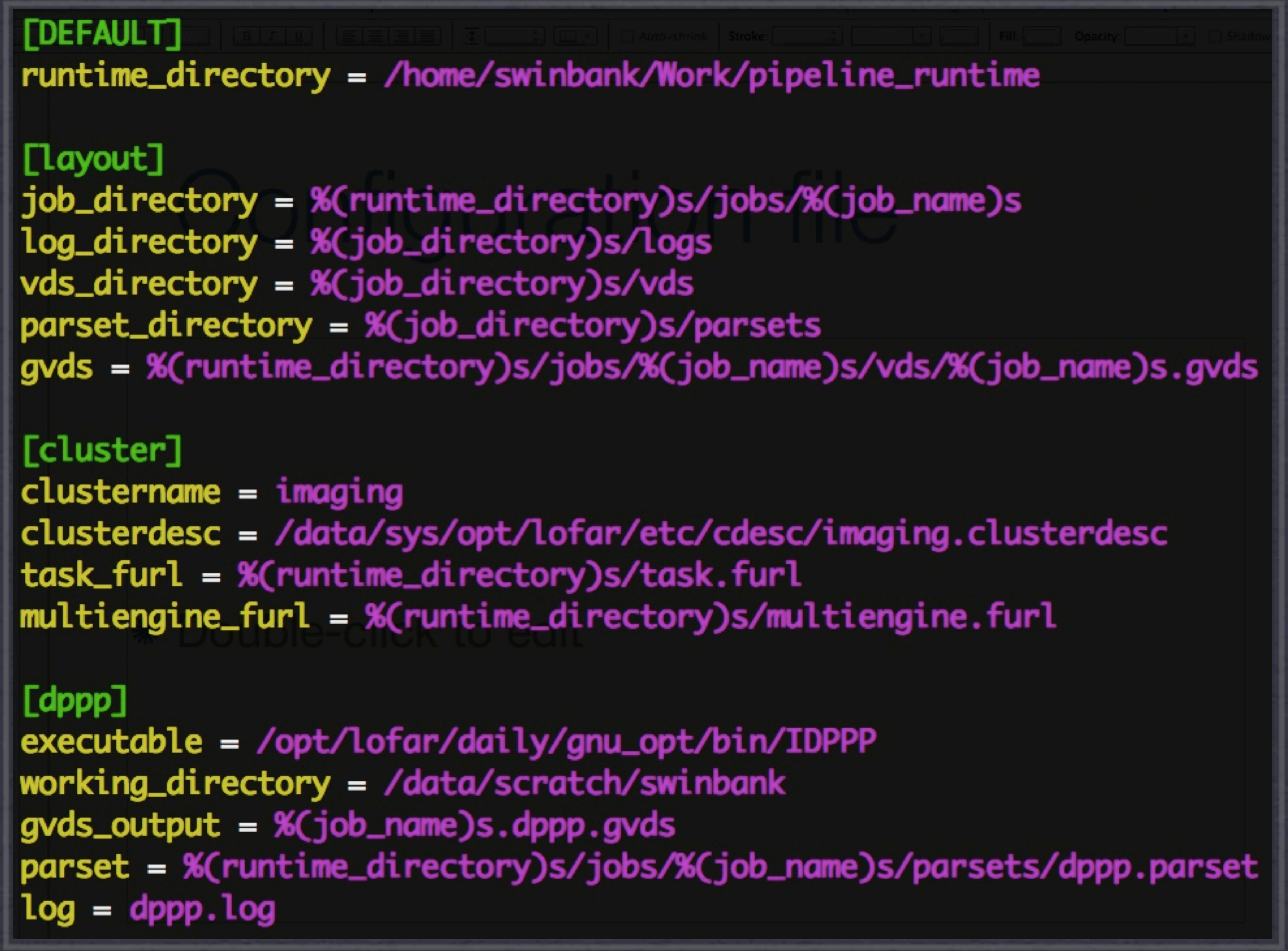


Basic tools

- Python
- Cuisine
- IPython
- Fabric
- Sphinx

Layout

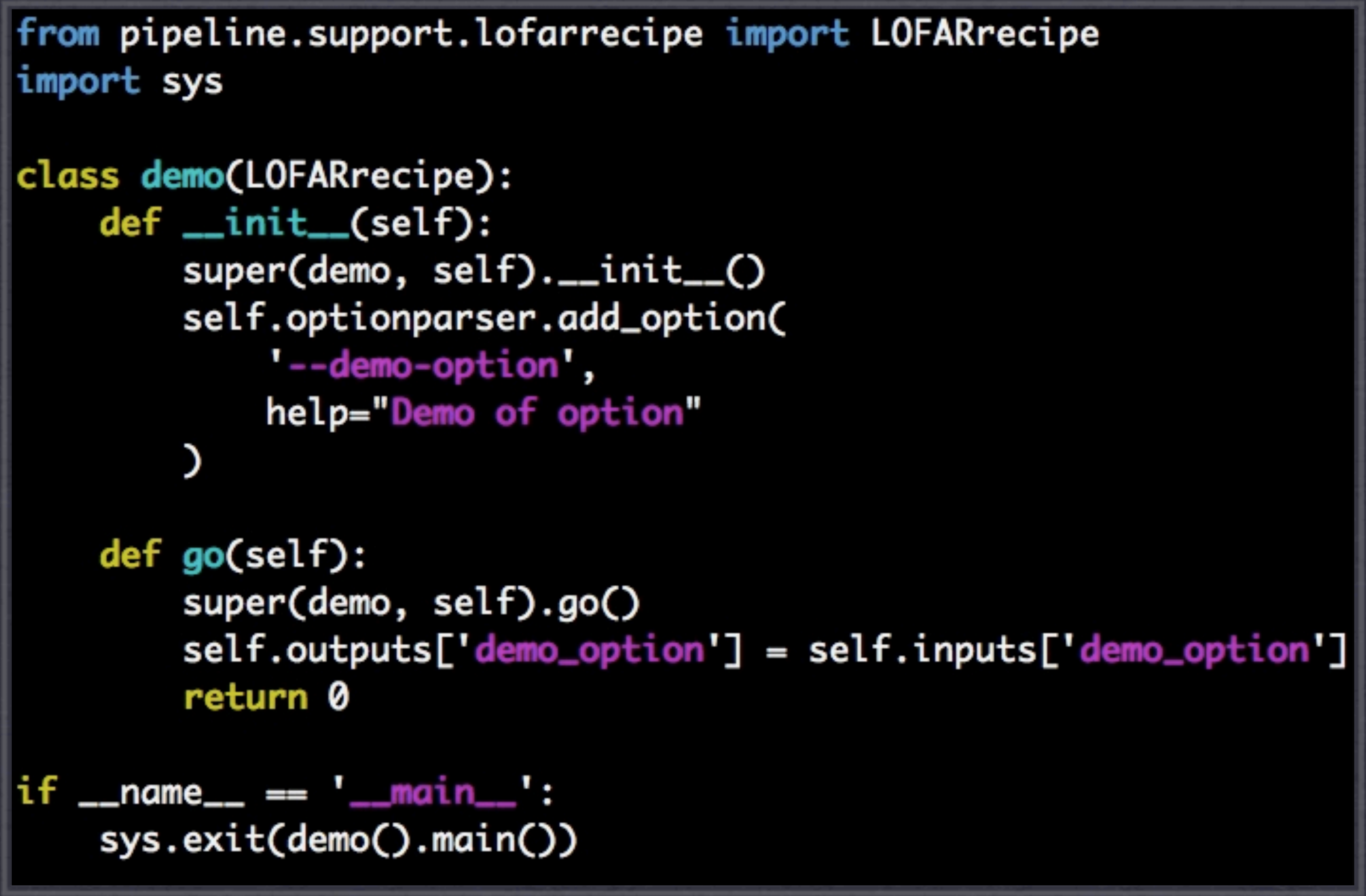
- Standard directory structure: dump configuration files on disk, hit go, come back to find the
    - vds
    - parsets
    - logs
    - results



```
[DEFAULT]
runtime_directory = /home/swinbank/Work/pipeline_runtime

[layout]
job_directory = %(runtime_directory)s/jobs/%(job_name)s
log_directory = %(job_directory)s/logs
vds_directory = %(job_directory)s/vds
parset_directory = %(job_directory)s/parsets
gvds = %(runtime_directory)s/jobs/%(job_name)s/vds/%(job_name)s.gvds

[cluster]
clustername = imaging
clusterdesc = /data/sys/opt/lofar/etc/cdesc/imaging.clusterdesc
task_furl = %(runtime_directory)s/task.furl
multiengine_furl = %(runtime_directory)s/multiengine.furl

[dppp]
executable = /opt/lofar/daily/gnu_opt/bin/IDPPP
working_directory = /data/scratch/swinbank
gvds_output = %(job_name)s.dppp.gvds
parset = %(runtime_directory)s/jobs/%(job_name)s/parsets/dppp.parset
log = dppp.log
```

Pipeline configuration

Python standard ConfigParser system

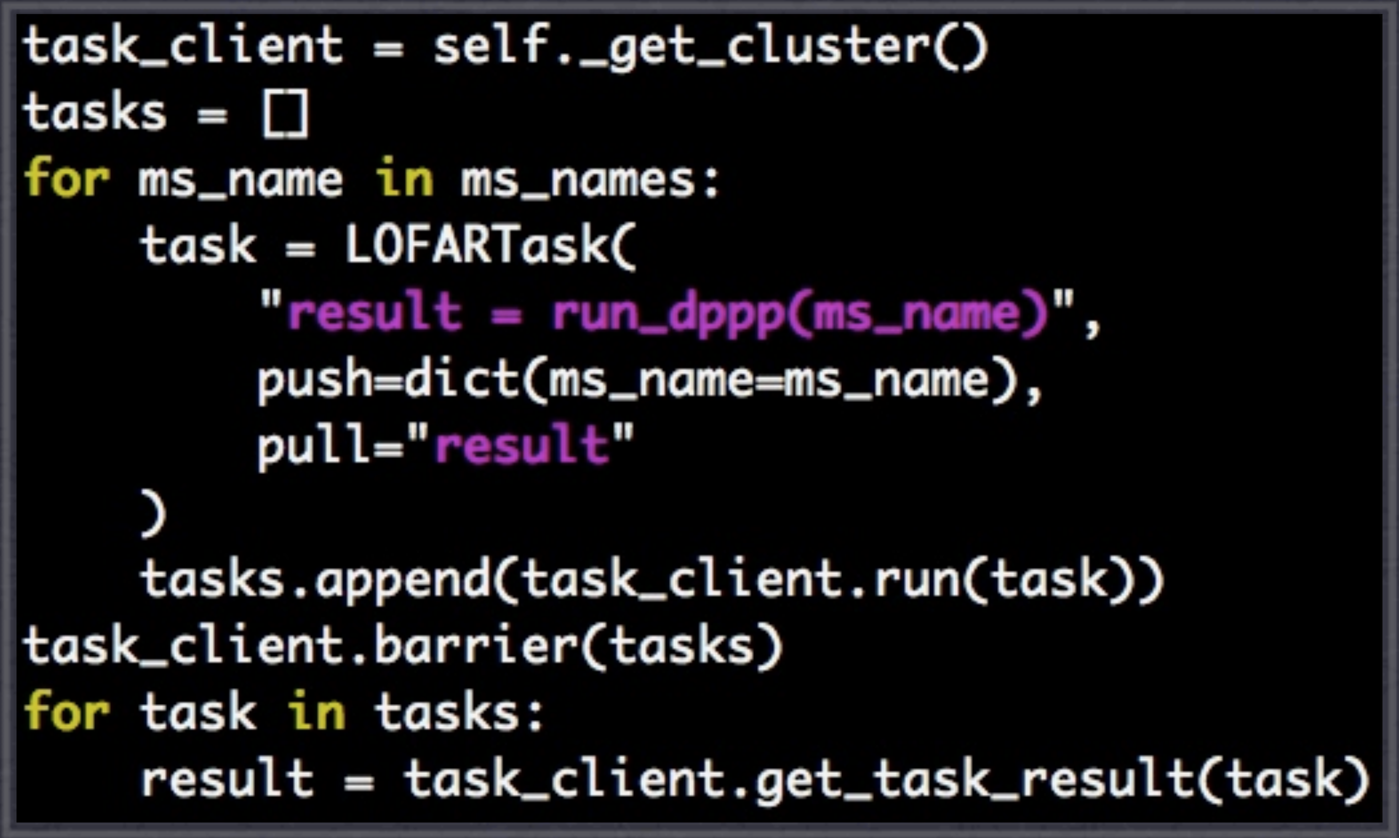


```python
from pipeline.support.lofarrecipe import LOFARrecipe
import sys

class demo(LOFARrecipe):
    def __init__(self):
        super(demo, self).__init__()
        self.optionparser.add_option(
            '--demo-option',
            help="Demo of option"
        )

    def go(self):
        super(demo, self).go()
        self.outputs['demo_option'] = self.inputs['demo_option']
        return 0

if __name__ == '__main__':
    sys.exit(demo().main())
```

Pipeline ‘recipes’

Well defined inputs & outputs

Standard option parsing, logging, etc



```
task_client = self._get_cluster()
tasks = []
for ms_name in ms_names:
    task = LOFARTask(
        "result = run_dppp(ms_name)",
        push=dict(ms_name=ms_name),
        pull="result"
    )
    tasks.append(task_client.run(task))
task_client.barrier(tasks)
for task in tasks:
    result = task_client.get_task_result(task)
```

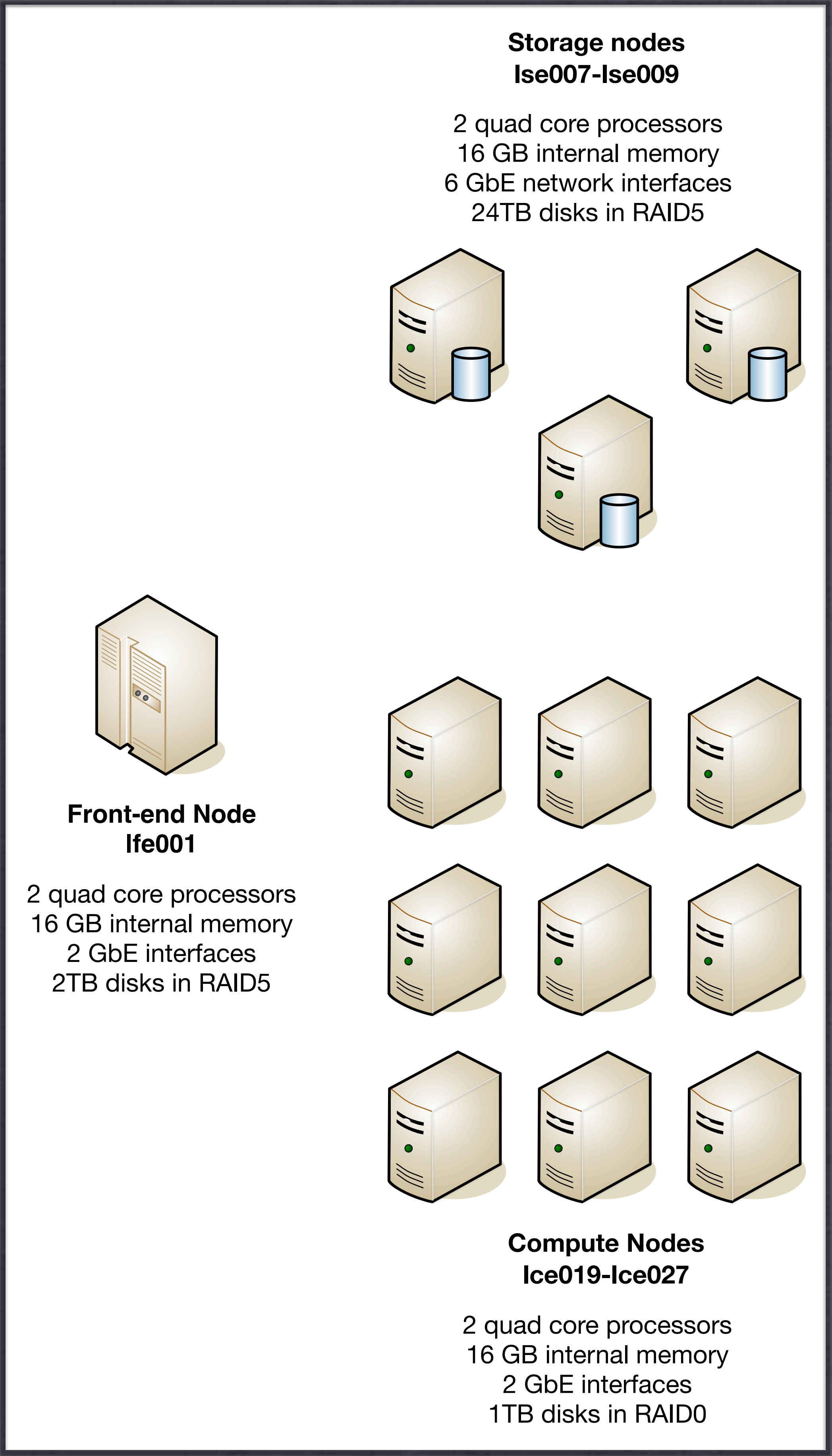
IPython Tasks

Quick & easy parallelism; push & pull functions to nodes

Dependencies, priorities, etc, available

Cluster layout

Imaging cluster “sub3”



Storage nodes
lse007-lse009

2 quad core processors
16 GB internal memory
6 GbE network interfaces
24TB disks in RAID5

Front-end Node
lfe001

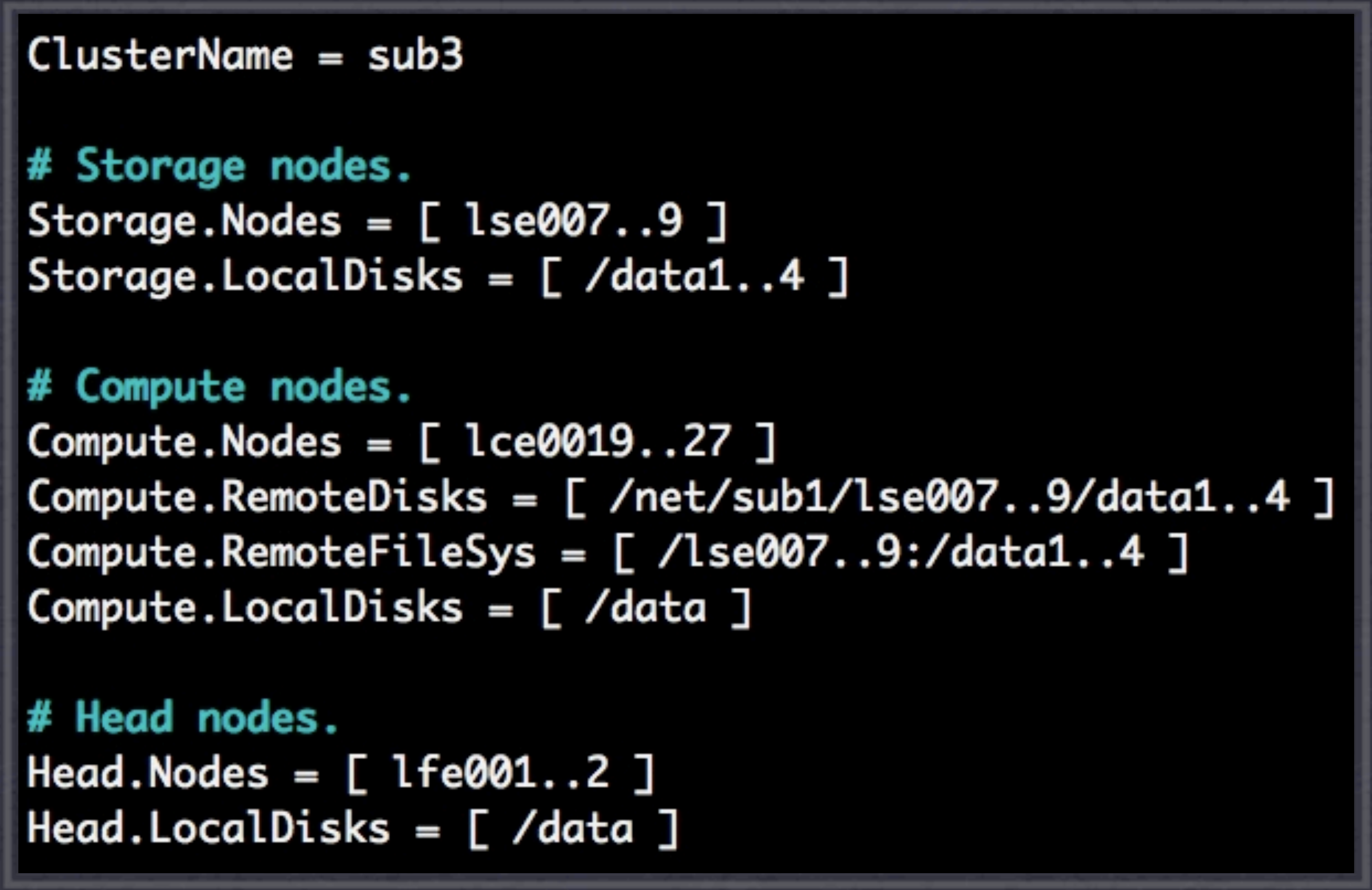
2 quad core processors
16 GB internal memory
2 GbE interfaces
2TB disks in RAID5

Compute Nodes
lce019-lce027

2 quad core processors
16 GB internal memory
2 GbE interfaces
1TB disks in RAID0

Cluster layout

Imaging cluster “sub3”

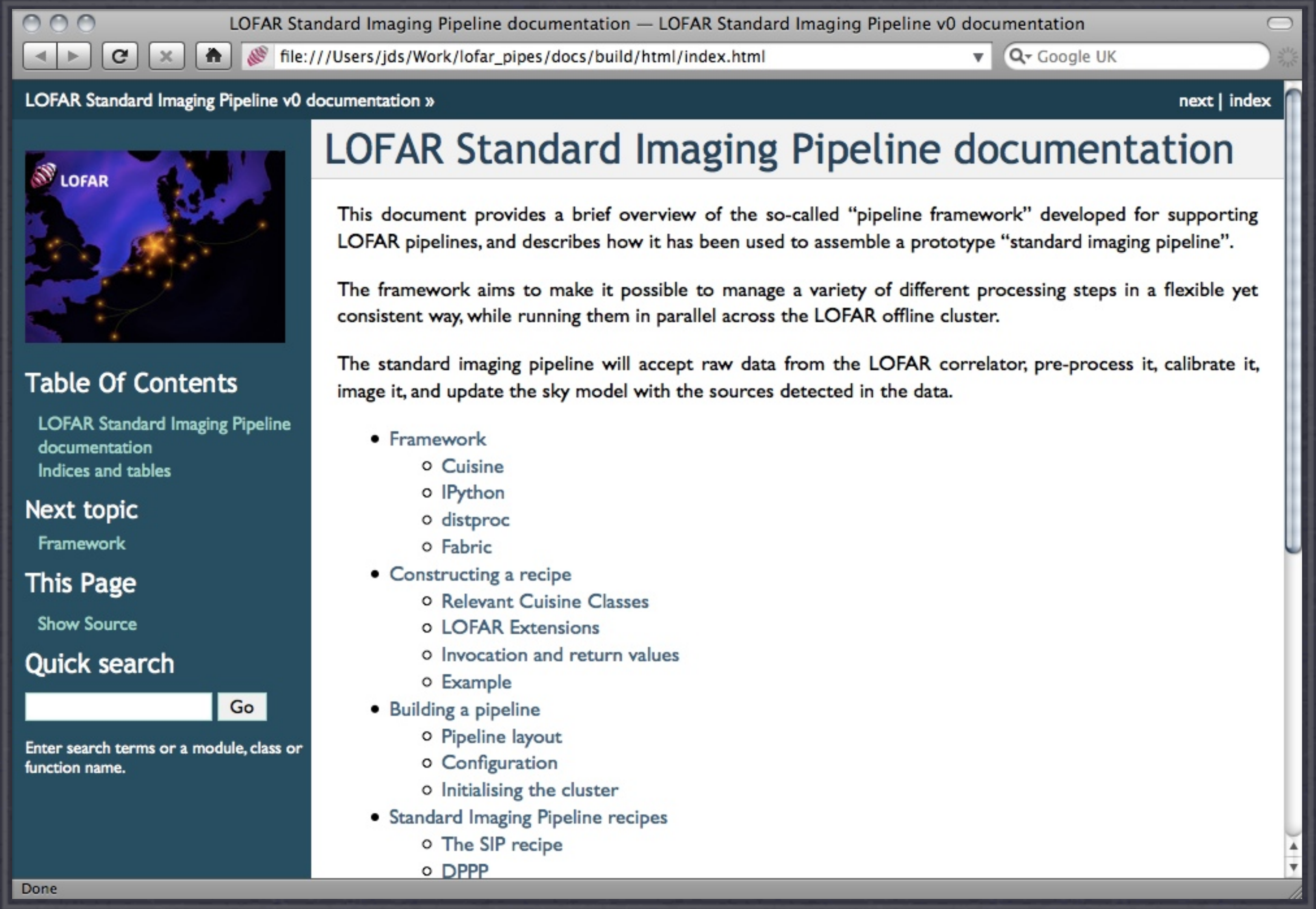


```
ClusterName = sub3

# Storage nodes.
Storage.Nodes = [ lse007..9 ]
Storage.LocalDisks = [ /data1..4 ]

# Compute nodes.
Compute.Nodes = [ lce0019..27 ]
Compute.RemoteDisks = [ /net/sub1/lse007..9/data1..4 ]
Compute.RemoteFileSys = [ /lse007..9:/data1..4 ]
Compute.LocalDisks = [ /data ]

# Head nodes.
Head.Nodes = [ lfe001..2 ]
Head.LocalDisks = [ /data ]
```

Deploying to the cluster

- Fabric reads clusterdesc file, starts IPython engines.

```
$ fab head_node start_controller

[lfe001] run: bash /home/swinbank/Work/lofar_pipes/deploy/
ipcontroller.sh /home/swinbank/Work/pipeline_runtime start

$ fab compute_nodes start_engine

[lce019] run: bash /home/swinbank/Work/lofar_pipes/deploy/
ipengine.sh /home/swinbank/Work/pipeline_runtime start

[lce020] run: bash /home/swinbank/Work/lofar_pipes/deploy/
ipengine.sh /home/swinbank/Work/pipeline_runtime start

[...]
```


LOFAR Standard Imaging Pipeline documentation — LOFAR Standard Imaging Pipeline v0 documentation
file:///Users/jds/Work/lofar_pipes/docs/build/html/index.html
Google UK
LOFAR Standard Imaging Pipeline v0 documentation »
next | index
LOFAR
Table Of Contents
LOFAR Standard Imaging Pipeline documentation
Indices and tables
Next topic
Framework
This Page
Show Source
Quick search
Go
Enter search terms or a module, class or function name.
Done

LOFAR Standard Imaging Pipeline documentation

This document provides a brief overview of the so-called "pipeline framework" developed for supporting LOFAR pipelines, and describes how it has been used to assemble a prototype "standard imaging pipeline".

The framework aims to make it possible to manage a variety of different processing steps in a flexible yet consistent way, while running them in parallel across the LOFAR offline cluster.

The standard imaging pipeline will accept raw data from the LOFAR correlator, pre-process it, calibrate it, image it, and update the sky model with the sources detected in the data.

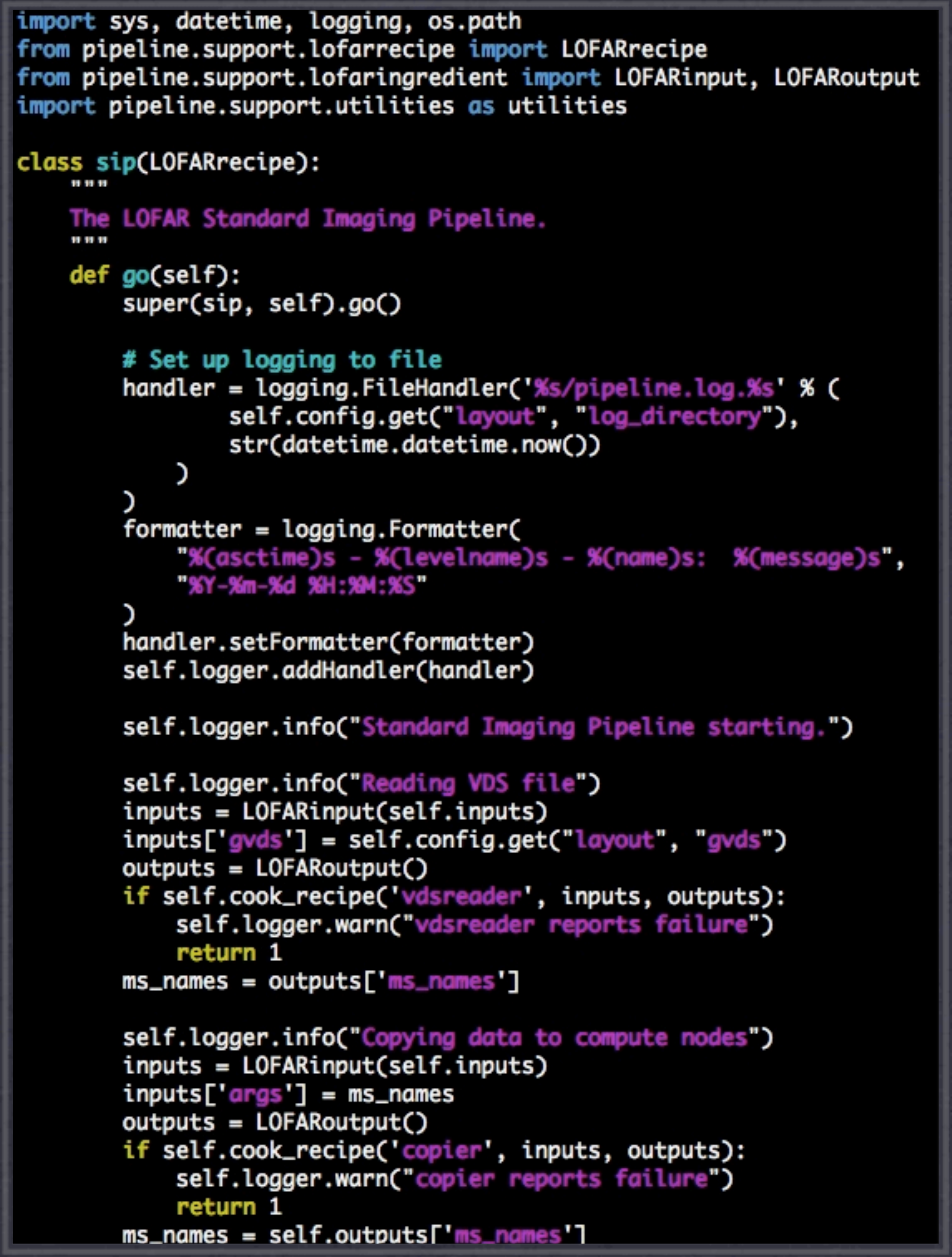
Documentation in progress

Using Sphinx

Wednesday, 23 September 2009

The (proto) SIP

Still a work in progress



```
import sys, datetime, logging, os.path
from pipeline.support.lofarrecipe import LOFARrecipe
from pipeline.support.lofaringredient import LOFARinput, LOFARoutput
import pipeline.support.utilities as utilities

class sip(LOFARrecipe):
    """
    The LOFAR Standard Imaging Pipeline.
    """
    def go(self):
        super(sip, self).go()

        # Set up logging to file
        handler = logging.FileHandler('%s/pipeline.log.%s' % (
                self.config.get("layout", "log_directory"),
                str(datetime.datetime.now())
            )
        )
        formatter = logging.Formatter(
            "%(asctime)s - %(levelname)s - %(name)s:  %(message)s",
            "%Y-%m-%d %H:%M:%S"
        )
        handler.setFormatter(formatter)
        self.logger.addHandler(handler)

        self.logger.info("Standard Imaging Pipeline starting.")

        self.logger.info("Reading VDS file")
        inputs = LOFARinput(self.inputs)
        inputs['gvds'] = self.config.get("layout", "gvds")
        outputs = LOFARoutput()
        if self.cook_recipe('vdsreader', inputs, outputs):
            self.logger.warn("vdsreader reports failure")
            return 1
        ms_names = outputs['ms_names']

        self.logger.info("Copying data to compute nodes")
        inputs = LOFARinput(self.inputs)
        inputs['args'] = ms_names
        outputs = LOFARoutput()
        if self.cook_recipe('copier', inputs, outputs):
            self.logger.warn("copier reports failure")
            return 1
        ms_names = self.outputs['ms_names']
```