DES-Brazil Science Portal

A framework for processing large astronomical datasets and sharing scientific results

Angelo Fausti Neto
angelofausti@linea.gov.br

and LIneA IT team

Science with the LSST: A Brazilian/US Joint Workshop
April 1-4, 2012

Science Portal in perspective

- **Phase I: proof of concept (2007-2010)**
  - Design & prototype (workflows, data model)
  - Development of simple pipelines
  - Single machine / single user environment
- **Phase II: Adapting it to the DES collaboration (2010-2011)**
  - Multi-user environment
  - Orchestration layer & parallelization
  - More complex pipelines (Cluster Finder, Photo-z, Precam, QA)
- **Phase III: Production (2012 -)**
  - Stable environment and release procedure
  - Data server tools
  - Installation wizard to easily integrate pipelines into the Portal
  - Increase computing and storage resources
  - Operation plan for the Data Center

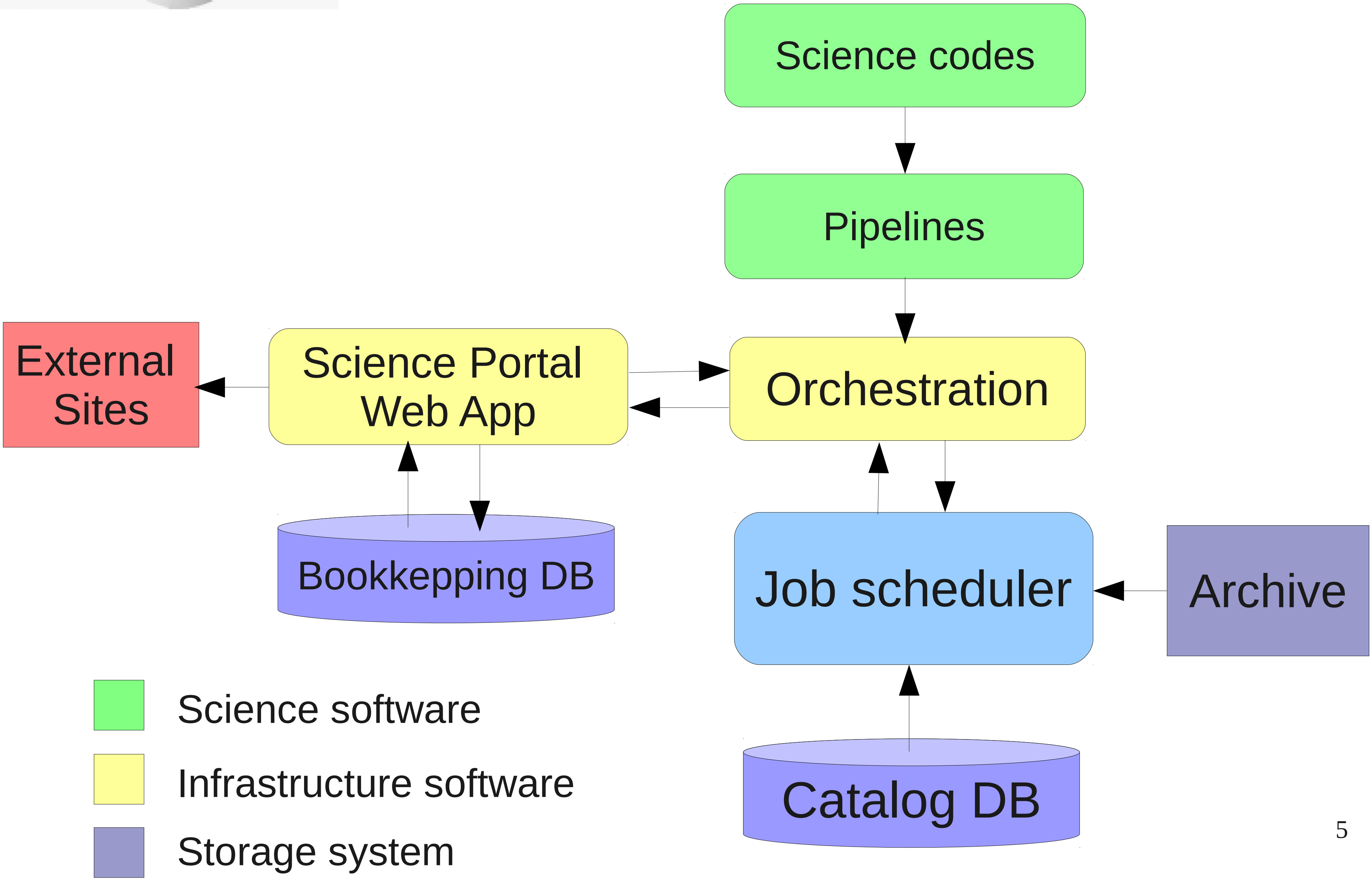
The Science Portal as an analysis framework

- **Large datasets**
  - “Bring the analysis to the data”
  - DES will produce ~500 TB/year and a total of ~2.5 PB by 2017
  - Calibrated images and source catalogs represent about ~350TB (~1.5B objects)
  - Need efficient data transfer, storage and processing
- **Large collaboration**
  - Keep the provenance of data and science codes
  - Keep the history of the analysis
  - Generating value added catalogs
  - Compare the results of different algorithms
  - Sharing of results of the analysis within the collaboration

The Science Portal in a nutshell

- A facility which interfaces a storage system and a computer cluster for the analysis of large datasets
- It is a multi user and collaborative analysis framework
- An open source (Python, PostgreSQL, SQLAlchemy) and extensible solution
- Allow easy integration of the analysis codes prepared by the collaboration
- Inherit analysis codes
- Data acess and data distribution

Science Portal components

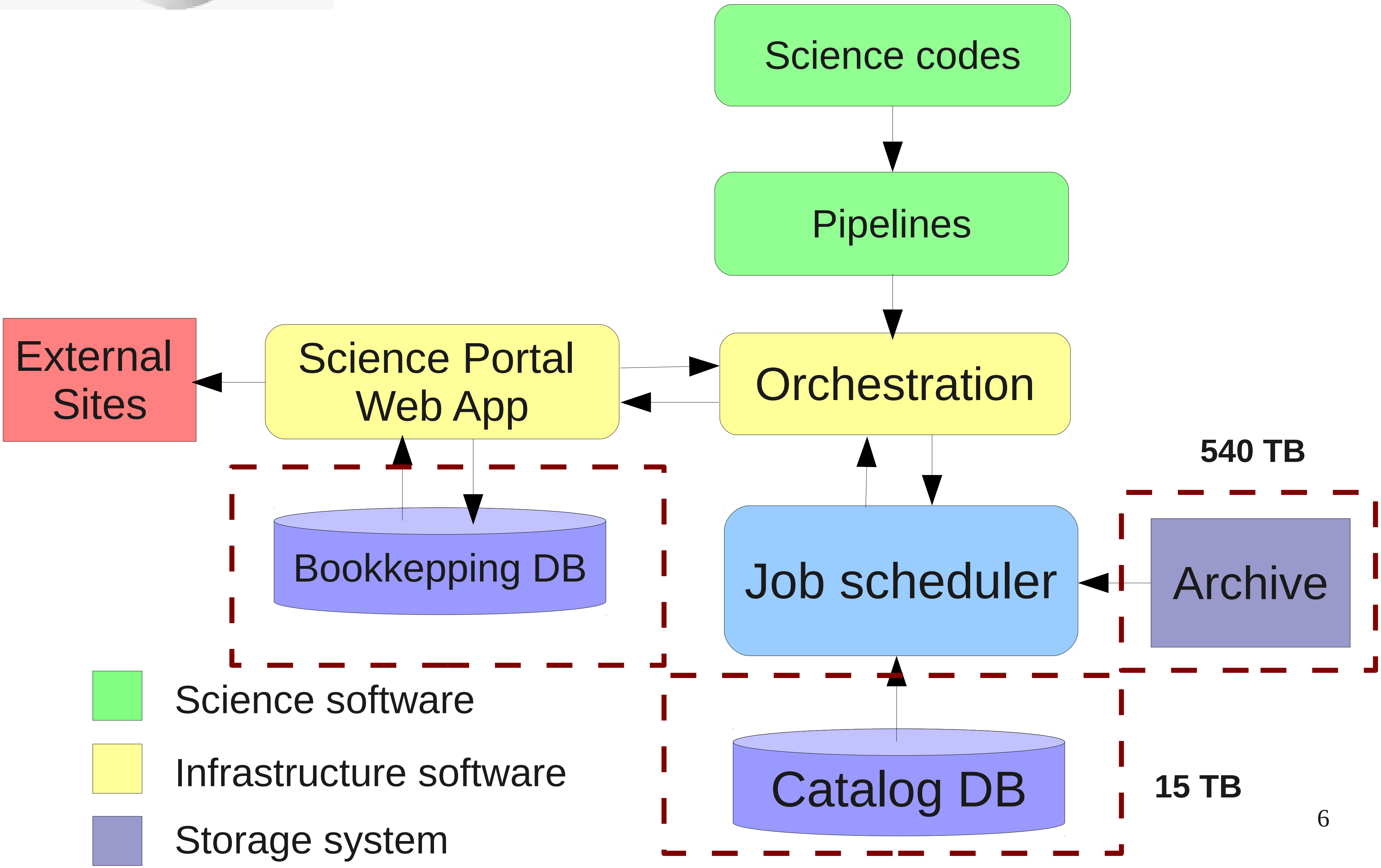


🟩 Science software

🟨 Infrastructure software

🟪 Storage system

Science Portal components



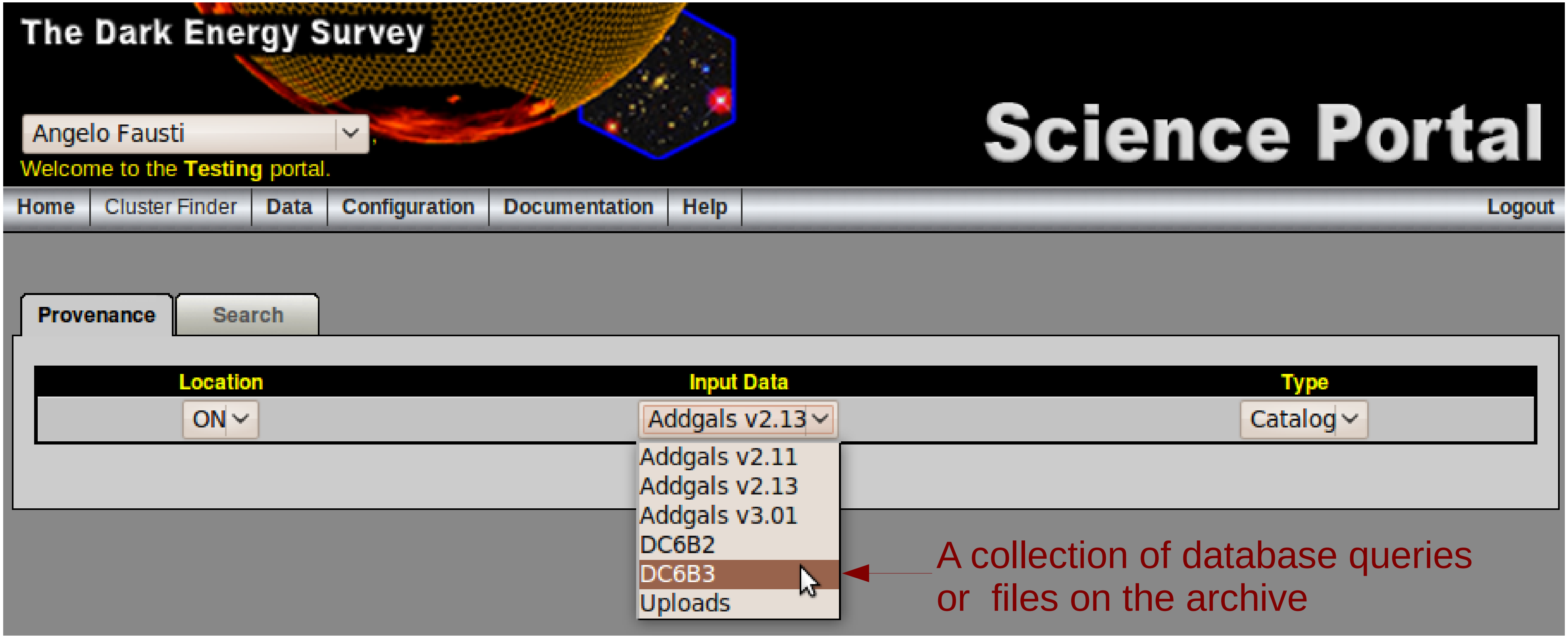
🟩 Science software

🟨 Infrastructure software

🟪 Storage system

Data selection

- Users have access to the **datasets** available in the Portal through the Search Engine



Angelo Fausti

Welcome to the **Testing** portal.

Location	Input Data	Type
ON	Addgals v2.13	Catalog

A collection of database queries or files on the archive

Dataset concept

- **A collection of database queries** defined in XML. Pipelines know only the dataset name and its class. E.g. DC6B dataset:

Database table	Class	Query
Coadd objects	science	SELECT * FROM dr011_coadd_objets_tag
Galaxy truth table	simulation/galaxies	SELECT * FROM dc6_truth_gal
Star truth table	simulation/stars	SELECT * FROM dc6_truth_star
Halo truth table	simulation/halos	SELECT * FROM dc6_truth_halos
DC6 match with UCAC3	match/ucac3	SELECT * FROM dr011_coadd_objects_tag_ucac3
Healpix density map	pixmap	_

DC6 and estimates for DES survey

200 sq deg - objects table has 30M rows x 1000 attributes

- Database load takes 3h at 2787 lines/s

5000 sq deg - objects table with ~1.5B rows x 1000 attributes

- Database load will take ~150h
- Cannot read this table!

New data model for the DES objects catalog

- The problem with large and wide tables
  - Full scan on the database table to select few attributes
  - 2 data pages to read a single row (16 KB) !!!
- Strategies to solve this problem:
  - Tune PostgreSQL configuration ➡
  - Indexing for fast spatial search using QC3 (similar to HTM) ✓
  - Use Q3C index for clustering the data on the disk – improve read access ✓
  - Materialized view (reduce to 130 attributes) ✓
  - Vertical partitioning ➡
  - Horizontal (spatial) partitioning (LSST strategy)

A data model driven by the analysis pipelines

_tag table

Columns used by Science Pipelines, q3c index

130 columns
ingestion rate 2787 lines/s

coadd_objects_photo

77 columns
ingestion rate 4697 lines/s

coadd_objects_astrometry

16 columns
ingestion rate 16113 lines/s

coadd_objects_morph

10 columns
ingestion rate 28559 lines/s

coadd_objects_photoz

8 columns
ingestion rate 25805 lines/s

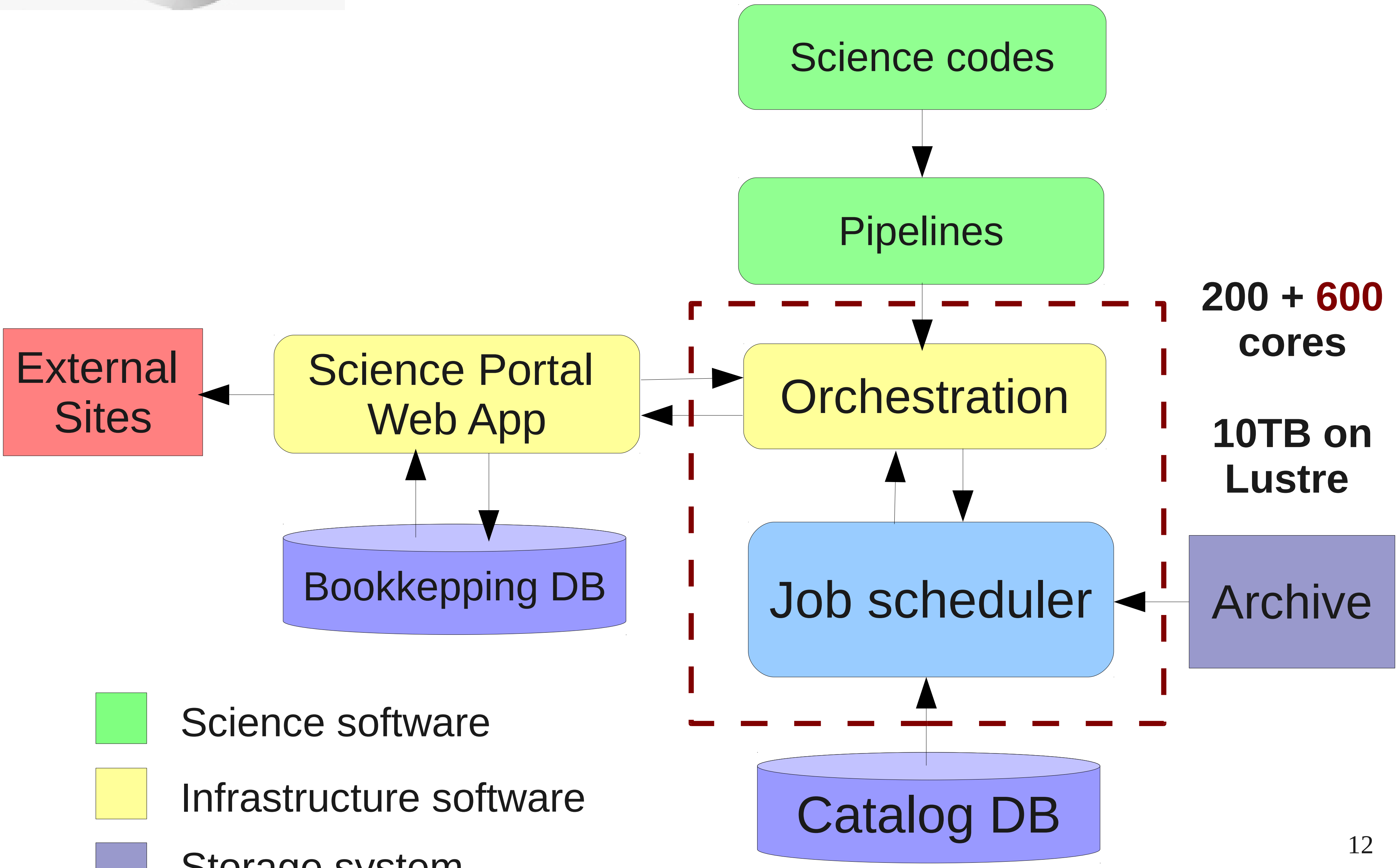
coadd_objects_truth

26 columns
ingestion rate 12012 lines/s

...

Value added catalogs

Science Portal components



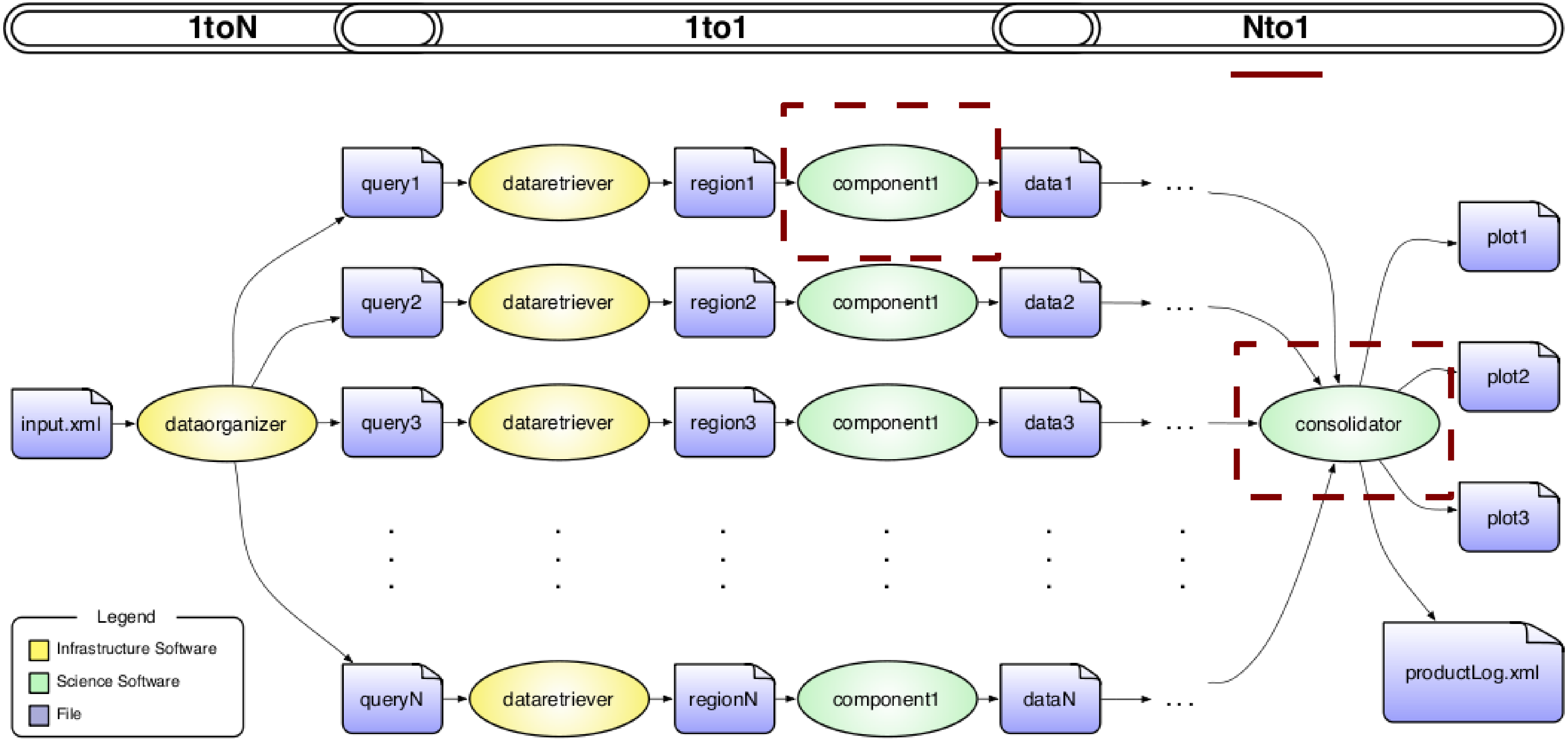
🟩 Science software

🟨 Infrastructure software

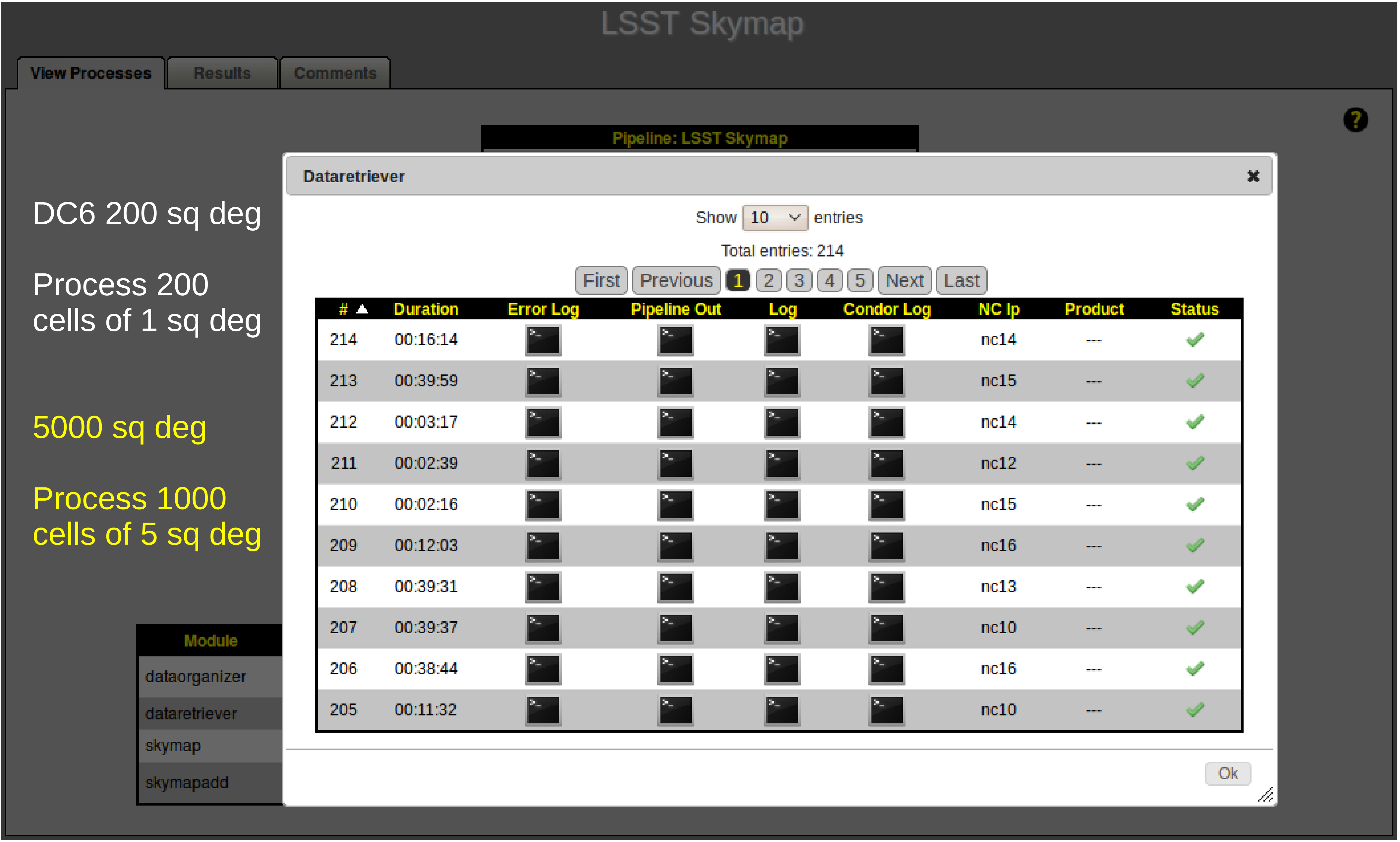
🟪 Storage system

Orchestration layer and parallel processing "map reduce" approach

It breaks a catalog query into N rectangular sub-queries, then process each region in parallel and consolidate the results



Data retriever bottleneck



DC6 200 sq deg

Process 200 cells of 1 sq deg

5000 sq deg

Process 1000 cells of 5 sq deg

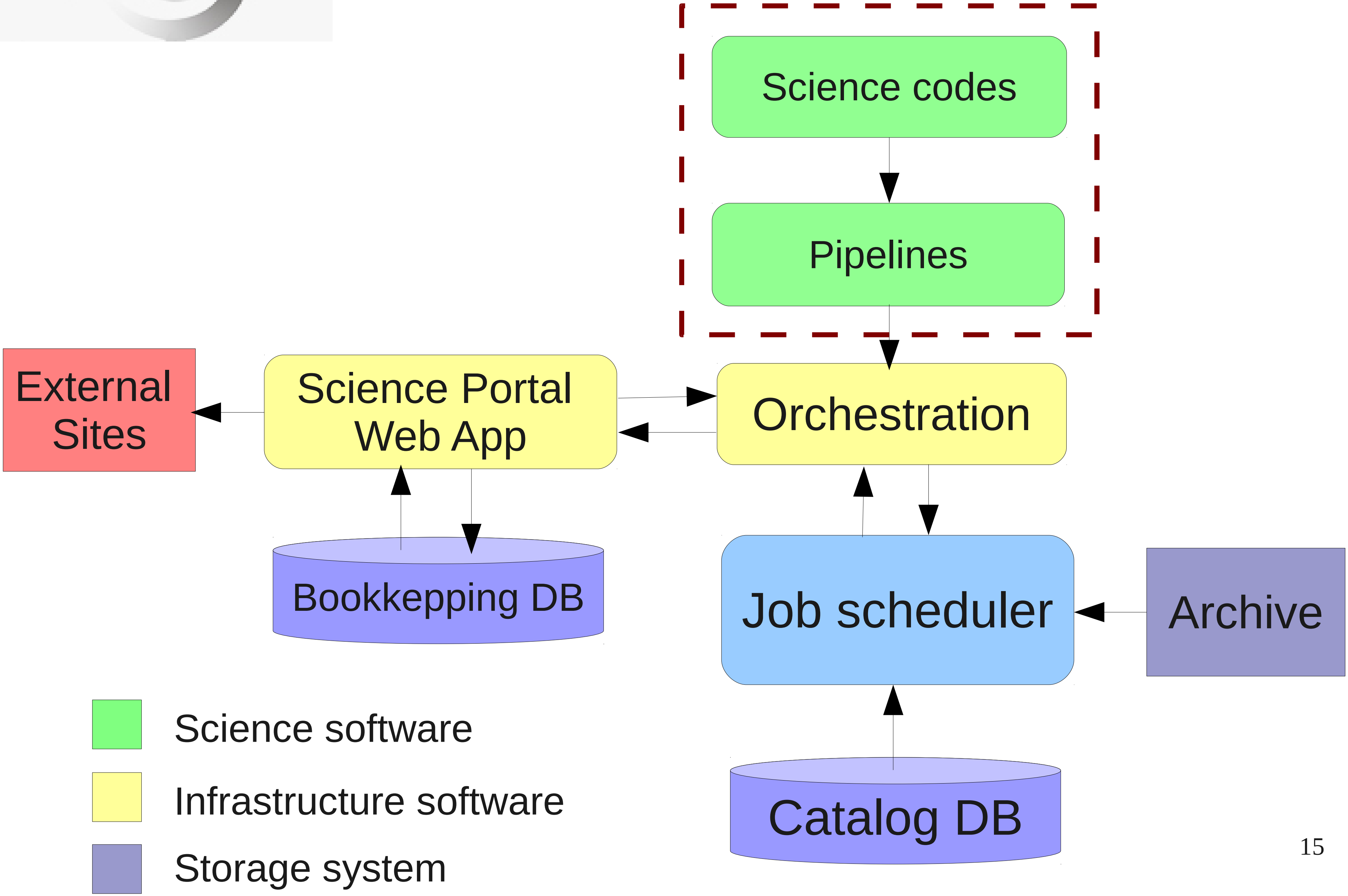
Show 10 entries

Total entries: 214

# ▲	Duration	Error Log	Pipeline Out	Log	Condor Log	NC Ip	Product	Status
214	00:16:14	▪	▪	▪	▪	nc14	---	✓
213	00:39:59	▪	▪	▪	▪	nc15	---	✓
212	00:03:17	▪	▪	▪	▪	nc14	---	✓
211	00:02:39	▪	▪	▪	▪	nc12	---	✓
210	00:02:16	▪	▪	▪	▪	nc15	---	✓
209	00:12:03	▪	▪	▪	▪	nc16	---	✓
208	00:39:31	▪	▪	▪	▪	nc13	---	✓
207	00:39:37	▪	▪	▪	▪	nc10	---	✓
206	00:38:44	▪	▪	▪	▪	nc16	---	✓
205	00:11:32	▪	▪	▪	▪	nc10	---	✓

Module
dataorganizer
dataretriever
skymap
skymapadd

Science Portal components



Science codes

Pipelines

External Sites

Science Portal Web App

Orchestration

Bookkepping DB

Job scheduler

Archive

Catalog DB

🟩 Science software

🟨 Infrastructure software

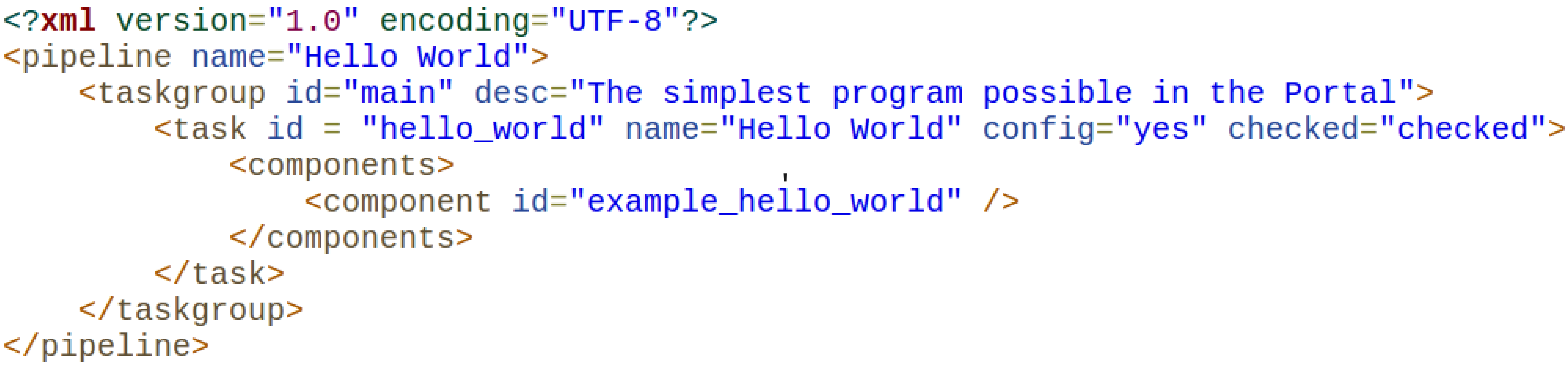
🟪 Storage system

Science Pipelines

- Pipeline
  - Ordered sequence of pipeline components
  - Dependencies defined in XML
- Pipeline component
  - Perform a discrete analysis step - modularity
  - A Python wrapper to an external executable (Fortran, C/C++, IDL or even Python) - flexibility
  - Input and configuration (optional) also defined in XML)

“Hello World!” Pipeline

```
cat des/pipelines/example_hello_world/pipeline.xml
```

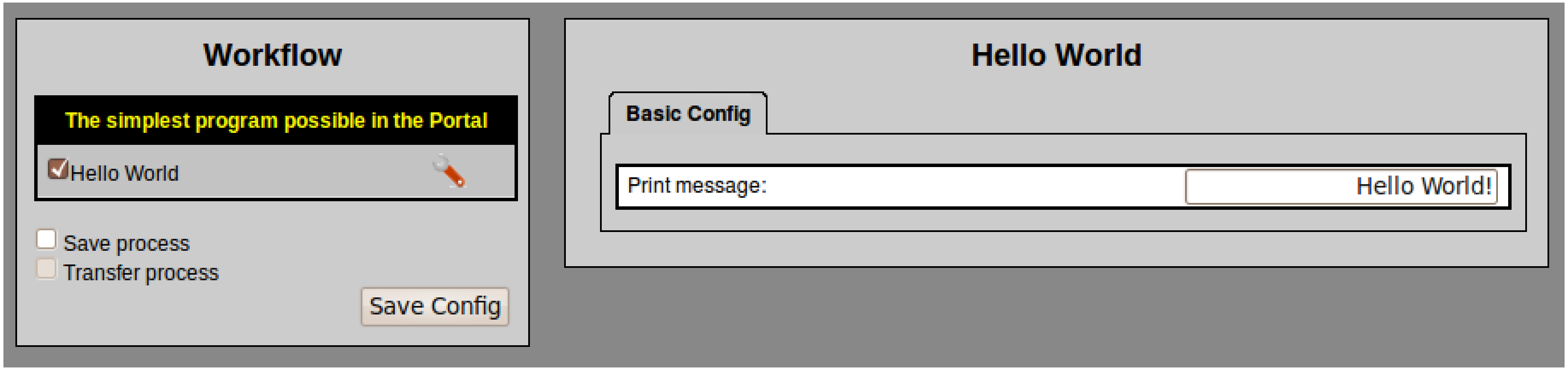


```
<?xml version="1.0" encoding="UTF-8"?>
<pipeline name="Hello World">
    <taskgroup id="main" desc="The simplest program possible in the Portal">
        <task id = "hello_world" name="Hello World" config="yes" checked="checked">
            <components>
                <component id="example_hello_world" />
            </components>
        </task>
    </taskgroup>
</pipeline>
```

“Hello World!” Configuration

```
cat des/components/example_hello_world/config/component_config.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet name="" type="text/xsl" href="/static/xml/mkform.xsl"?>
<config module="example_hello_world" version="0.1">
    <section id="Basic Config">
        <subsection>
            <scalar id="message" name="Print message:" default="Hello World!" type="string">
                <values>
                    <value>Hello World!</value>
                </values>
            </scalar>
        </subsection>
    </section>
</config>
```



Workflow

The simplest program possible in the Portal

☑ Hello World

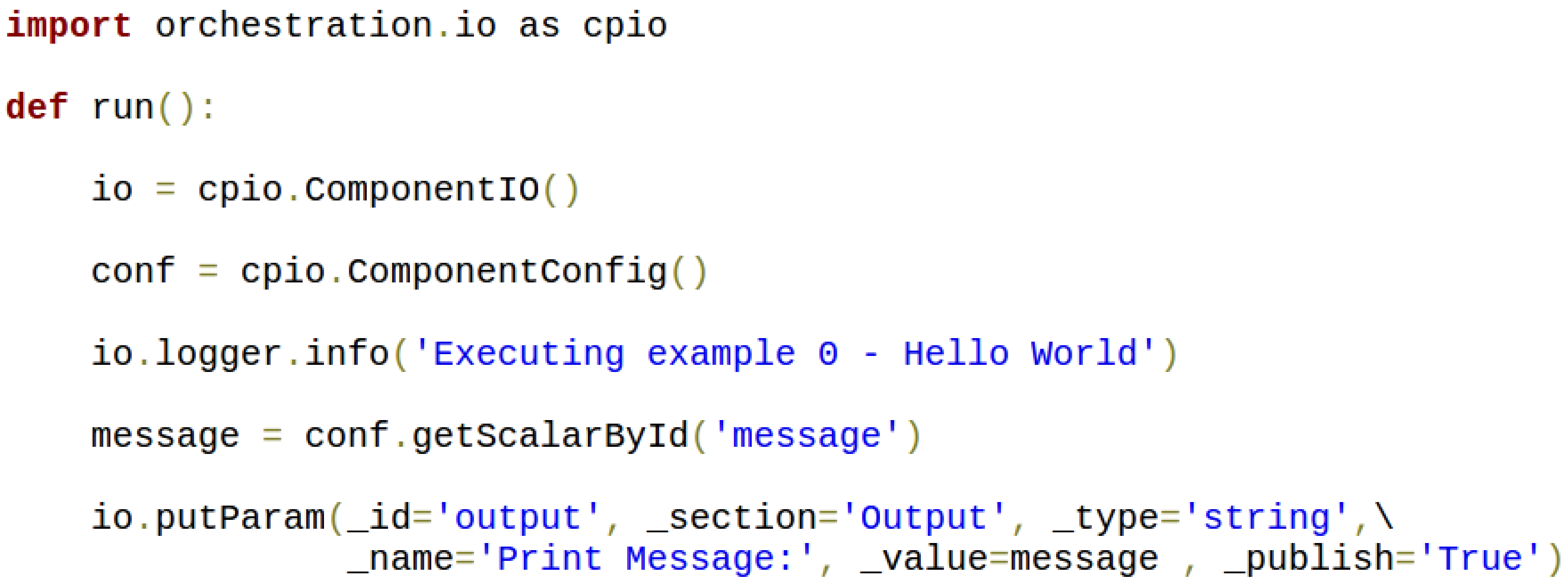
☐ Save process
☐ Transfer process

Hello World

Print message: Hello World!

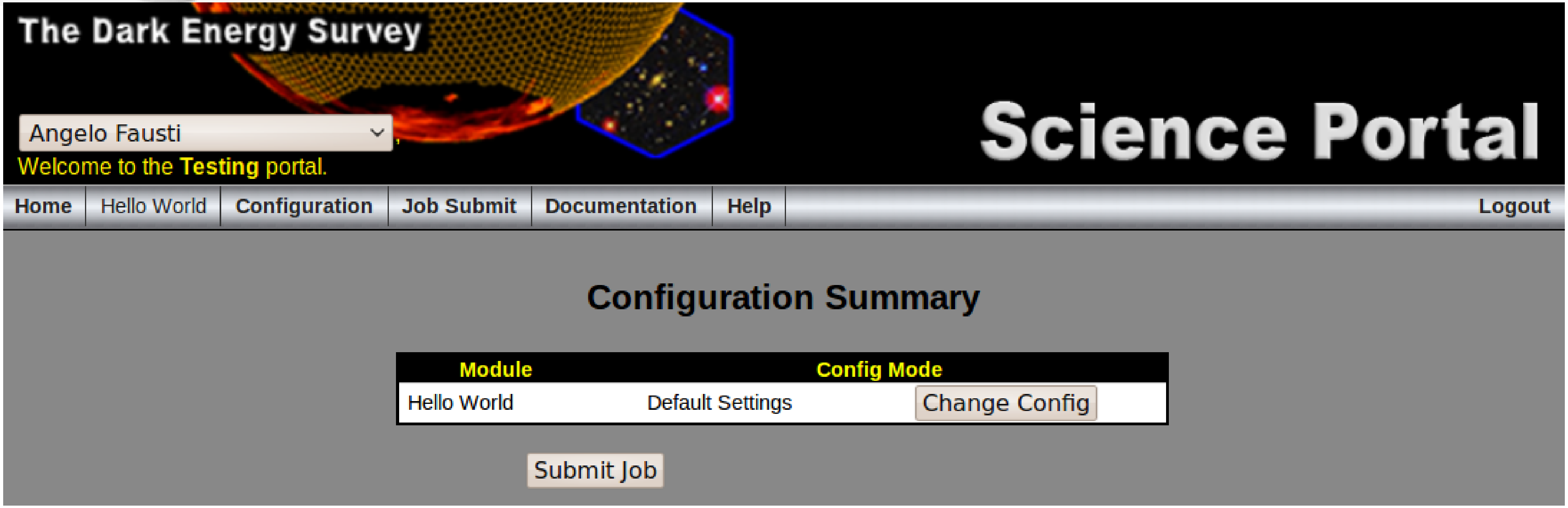
“Hello World!” Portal wrapper

```
cat des/components/example_hello_world/__init__.py
```



```
import orchestration.io as cpio

def run():

    io = cpio.ComponentIO()

    conf = cpio.ComponentConfig()

    io.logger.info('Executing example 0 - Hello World')

    message = conf.getScalarById('message')

    io.putParam(_id='output', _section='Output', _type='string',\
                _name='Print Message:', _value=message , _publish='True')
```

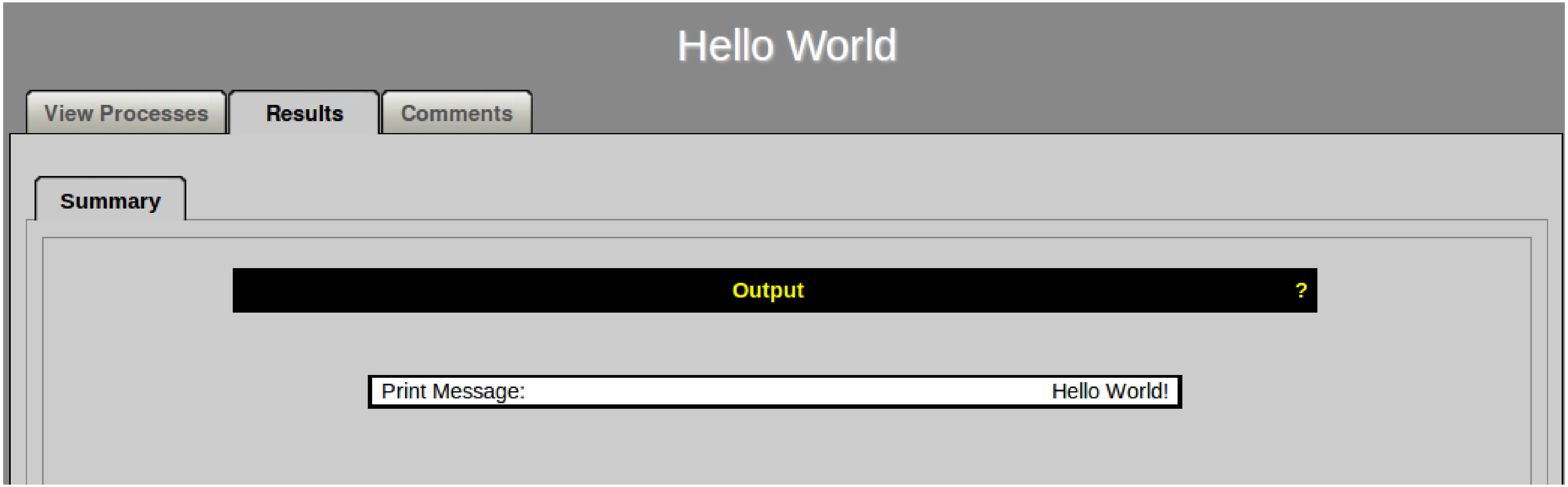
Process submission and results



Welcome to the **Testing** portal.

Configuration Summary

Module	Config Mode	
Hello World	Default Settings	Change Config



Hello World

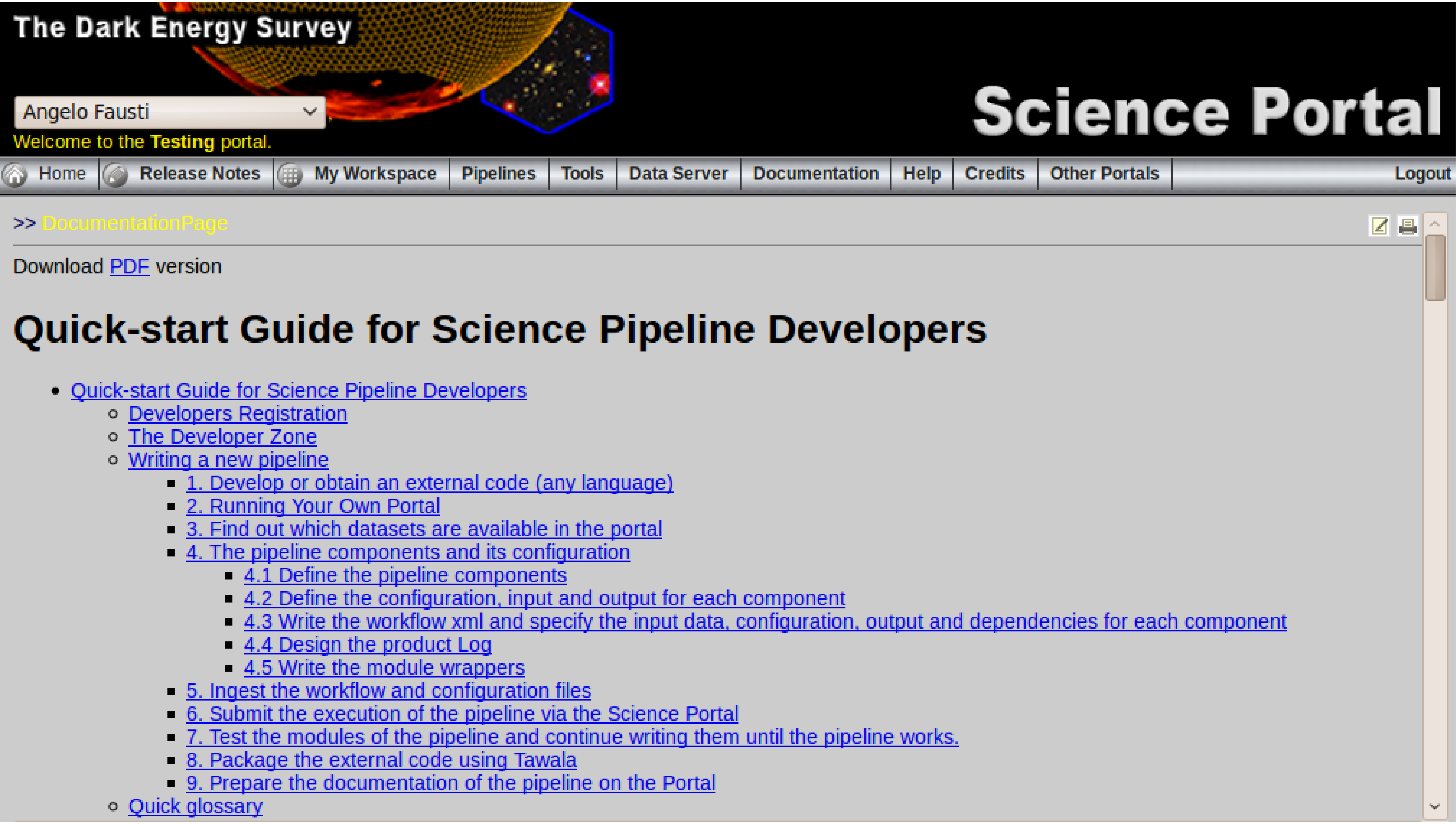
Output

Print Message: Hello World!

Pipeline integration

- Science codes are installed in the OS by a package manager – tawala
- API with standard methods for calling external codes **`io.run_command()`**, logging and produce the outputs (publish files, plots and tables)
- Automatically build the web interfaces for pipeline configuration, submission, process monitor and to display the results (Product Log)
- No HTML coding necessary to integrate a pipeline on the portal web application!
- **It requires 9 steps - needs to be automated**

Pipeline Installation Wizard

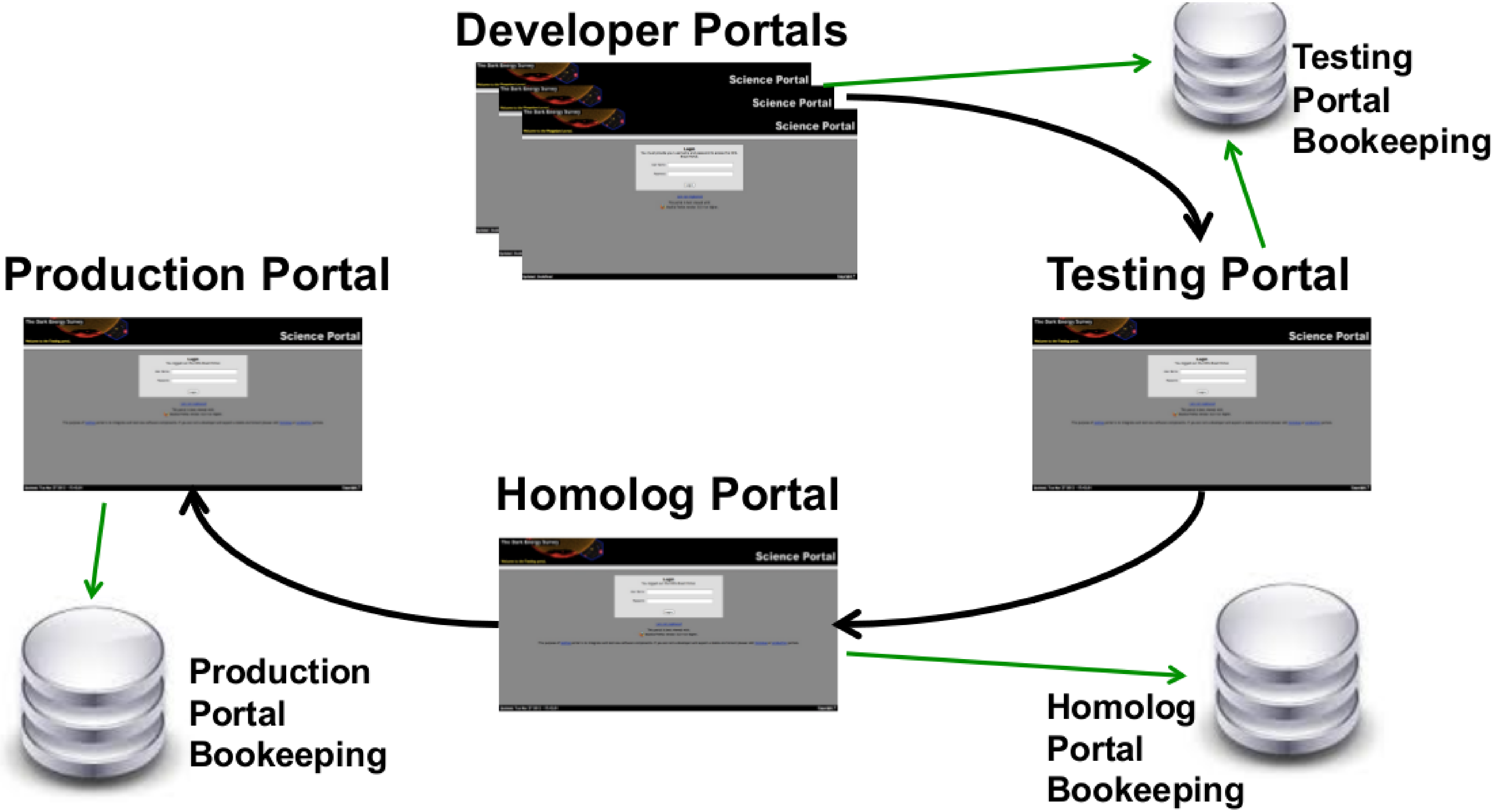


Download PDF version

Quick-start Guide for Science Pipeline Developers

- Quick-start Guide for Science Pipeline Developers
  - Developers Registration
  - The Developer Zone
  - Writing a new pipeline
    - 1. Develop or obtain an external code (any language)
    - 2. Running Your Own Portal
    - 3. Find out which datasets are available in the portal
    - 4. The pipeline components and its configuration
      - 4.1 Define the pipeline components
      - 4.2 Define the configuration, input and output for each component
      - 4.3 Write the workflow xml and specify the input data, configuration, output and dependencies for each component
      - 4.4 Design the product Log
      - 4.5 Write the module wrappers
    - 5. Ingest the workflow and configuration files
    - 6. Submit the execution of the pipeline via the Science Portal
    - 7. Test the modules of the pipeline and continue writing them until the pipeline works.
    - 8. Package the external code using Tawala
    - 9. Prepare the documentation of the pipeline on the Portal
  - Quick glossary

Towards stability



FUTURE PROJECTS

Data server tools

- Catalog Server
  - Sample queries, Advanced queries
- Image Server
  - Cutout Service
- Data request
- Datasets
  - Viewer, Dataset definition, Pipeline data ➡
- Archive viewer
- Import
  - Upload data ➡
  - My DB
- Export
  - Data Release

Use open source solutions!

See Project_Scope_Statement-DataManager-v1.doc

View Target

Examine by eye small sample of objects in the database catalog and image cutouts

- FITS web viewer - FITSJS
- Interactive plots
- List the properties of these targets
- Add annotations
- Link these objects to external databases information

ASCOT

See Project_Scope_Statement-view-target_v1.doc

Remote processing

- Extend the portal to support pipeline processing on **remote resources**.
- Although centrally located, the Portal should be able to submit pipeline executing to a remote data center where the input data located.

Conclusions

- We presented the Sicience Portal as an analysis framework being applied to DES
- The infrastructure software is in advanced development stage – working pipelines (Cluster Finder, Photo-z, Stellar catalog and QSO simulations) and sicence applications (QA, Precam and Quick Reduce)
- DES (5000 sq deg) and LSST (18000 sq deg) face similar issues in DM
- LIneA 2012 - transition from development environment to a production environment

EXTRA SLIDES

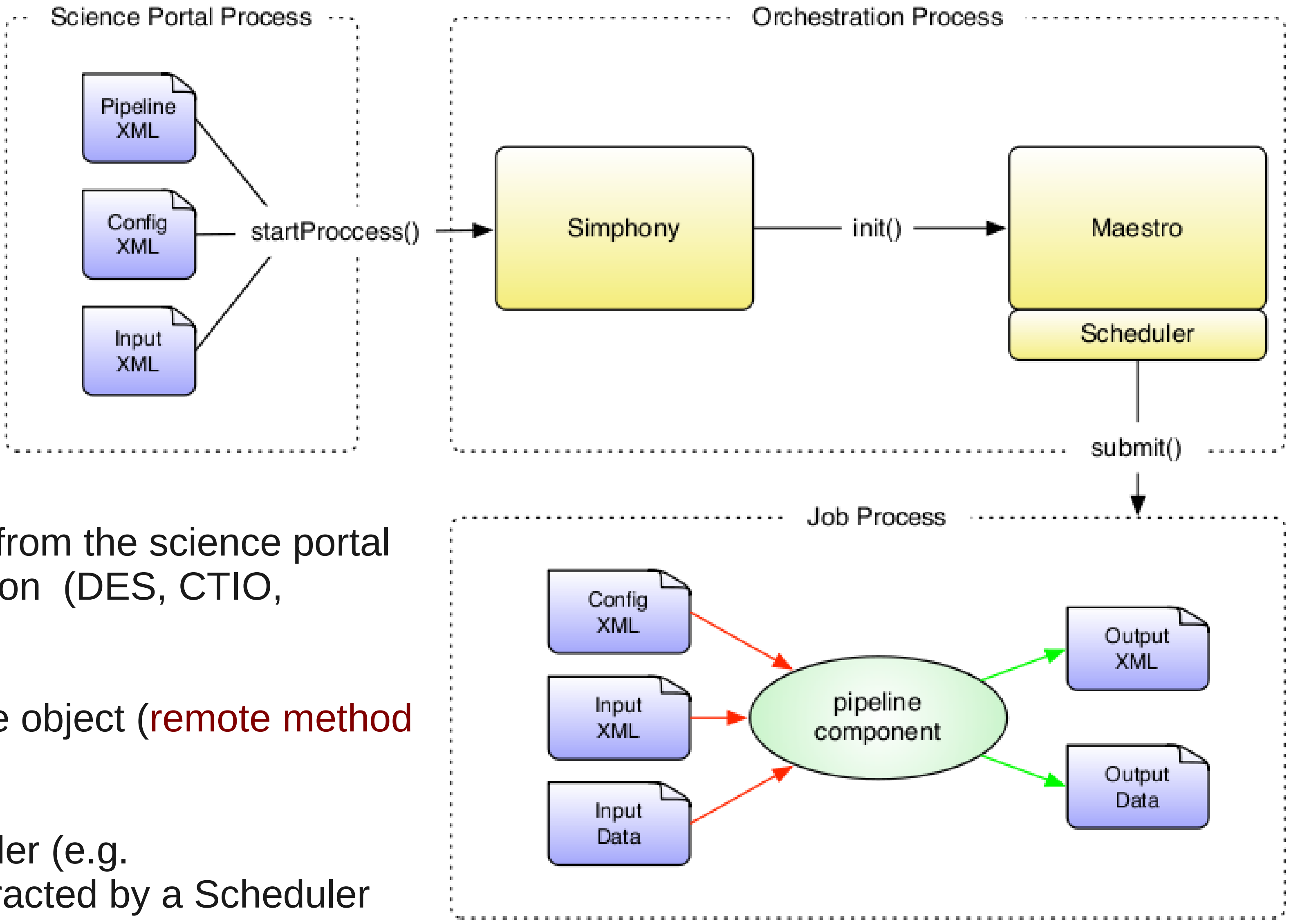
Ingestion procedure

- Well defined but some steps still need to be automated
- Selective ingestion
  - Creates a pg table from FITS header information and a list of columns
  - Format the FITS data for postgres COPY command
- For each table...
  - First order the data on the disk based on q3c_ang2ipix(ra, dec) (clustering)
  - Create the Q3C index
  - ANALYSE table
- Create auxiliary tables
  - Key (ID, RA, DEC)
  - Create the helpix map
  - Match with reference catalogs using Q3C
- Dataset definition and pipeline association

Fast spatial search using Q3C on postgreSQL

- Dataorganizer produces N sub-queries, where N is the number of cells
  - SELECT * FROM dr011_coadd_objects_tag WHERE ra BETWEEN 330 AND 332 AND dec between -44 AND -42;
  - Installed Q3C on Postgres (equivalent to HTM)
  - SELECT * FROM mytable WHERE q3c_poly_query(ra, dec, '{330, -44, 332, -44, 332, -42, 330, -42}');
- Q3C also provides functions for:
  - Radial search
  - Eliptical search
  - Positional macthing (matched DC6B with UCAC3 catalog in 2.3 seconds) ✓

Orchestration



- Decoupled from the science portal web application (DES, CTIO, SDSS3, etc.)

- Pyro remote object (remote method invocation)

- Job scheduler (e.g. condor) abstracted by a Scheduler class

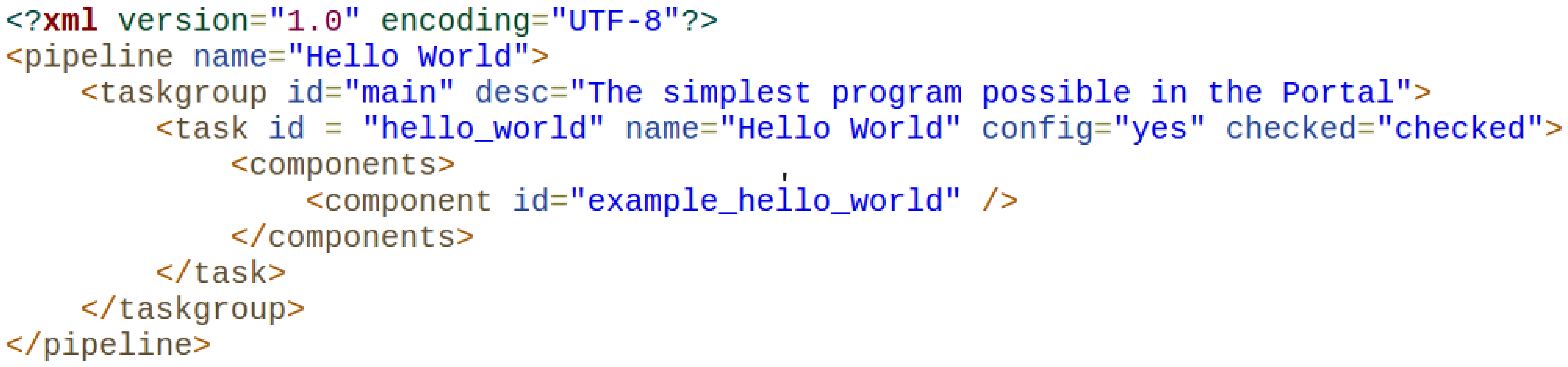
Each task defined in the pipeline.xml is a job

Process products

- Pipeline output
  - Published data files, plots and tables
  - Descriptive XML files
  - Saved to archive after removing intermediary products
  - Scales by the output size x N processes - large
- Bookkeeping DB
  - Description of the input data
  - Pipeline configuration
  - Process execution history
  - Scales with the N processes x M XML files – not large

“Hello World!” Pipeline

```
cat des/pipelines/example_hello_world/pipeline.xml
```

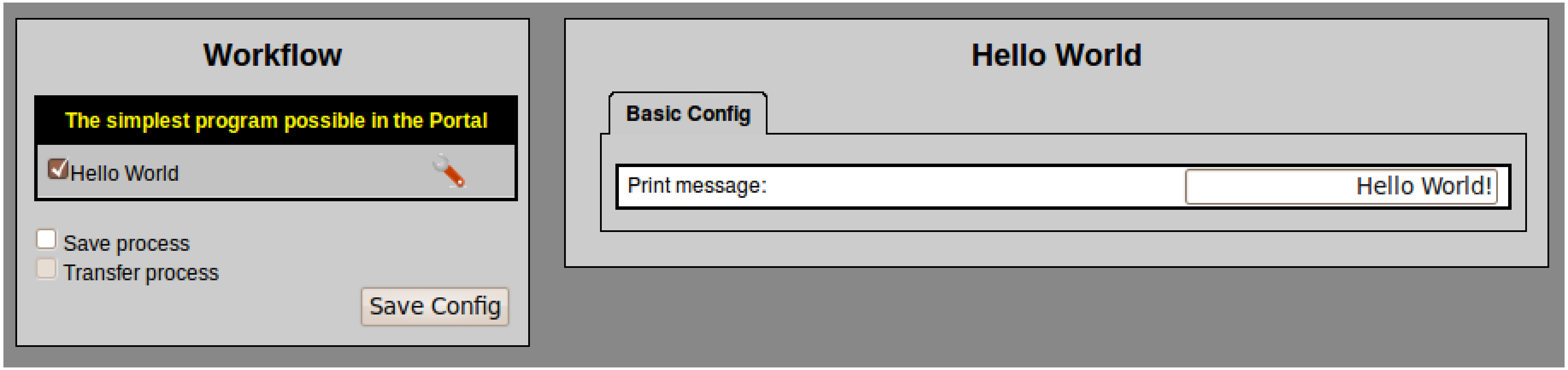


```
<?xml version="1.0" encoding="UTF-8"?>
<pipeline name="Hello World">
    <taskgroup id="main" desc="The simplest program possible in the Portal">
        <task id = "hello_world" name="Hello World" config="yes" checked="checked">
            <components>
                <component id="example_hello_world" />
            </components>
        </task>
    </taskgroup>
</pipeline>
```

“Hello World!” Configuration

```
cat des/components/example_hello_world/config/component_config.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet name="" type="text/xsl" href="/static/xml/mkform.xsl"?>
<config module="example_hello_world" version="0.1">
    <section id="Basic Config">
        <subsection>
            <scalar id="message" name="Print message:" default="Hello World!" type="string">
                <values>
                    <value>Hello World!</value>
                </values>
            </scalar>
        </subsection>
    </section>
</config>
```



Workflow

The simplest program possible in the Portal

☑ Hello World

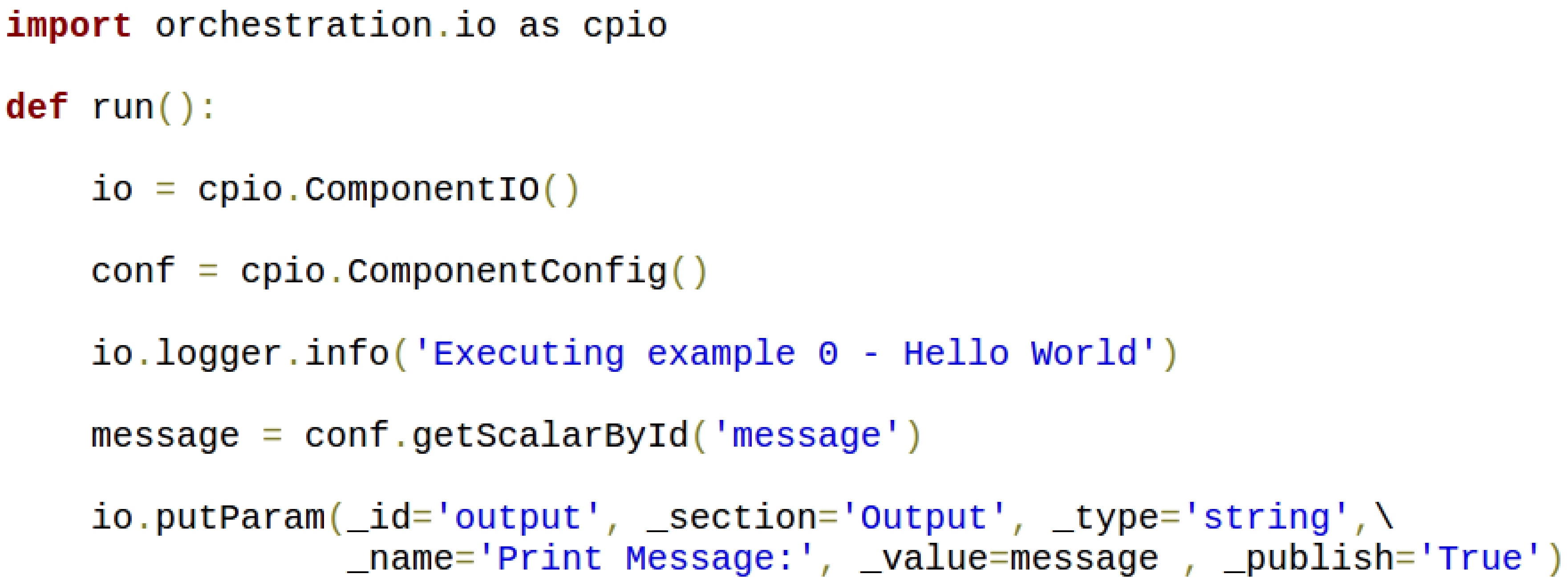
☐ Save process
☐ Transfer process

Hello World

Print message: Hello World!

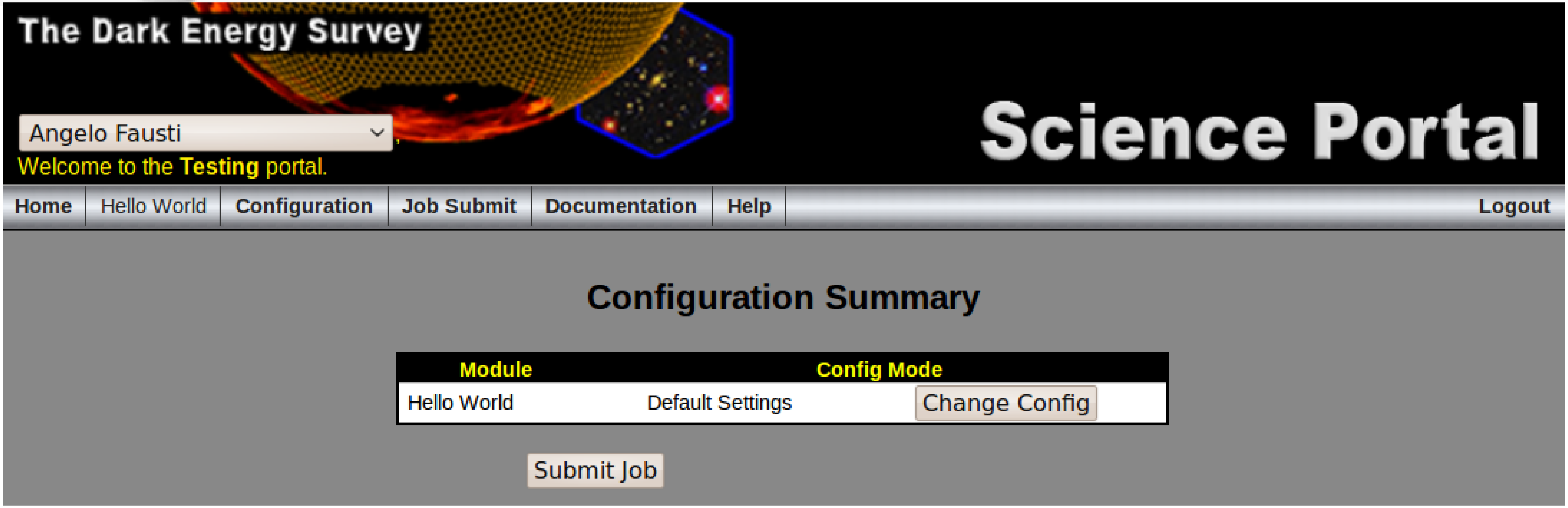
“Hello World!” Portal wrapper

```
cat des/components/example_hello_world/__init__.py
```



```
import orchestration.io as cpio

def run():

    io = cpio.ComponentIO()

    conf = cpio.ComponentConfig()

    io.logger.info('Executing example 0 - Hello World')

    message = conf.getScalarById('message')

    io.putParam(_id='output', _section='Output', _type='string',\
                _name='Print Message:', _value=message , _publish='True')
```

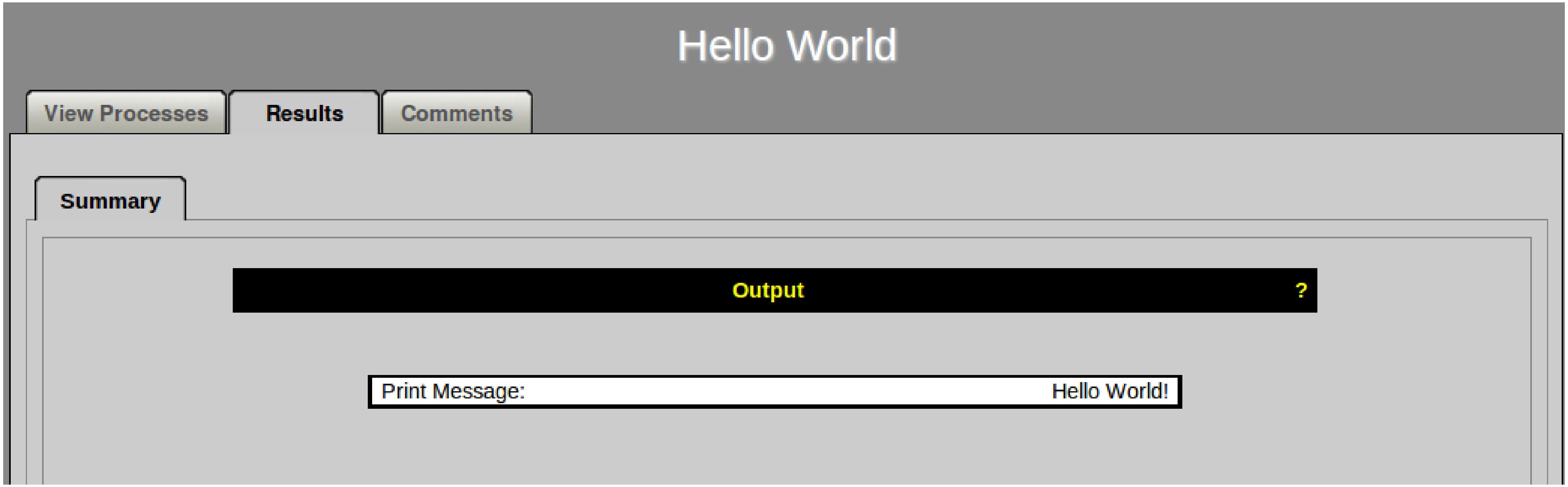
Process submission and results



Welcome to the **Testing** portal.

Configuration Summary

Module	Config Mode
Hello World	Default Settings



Hello World

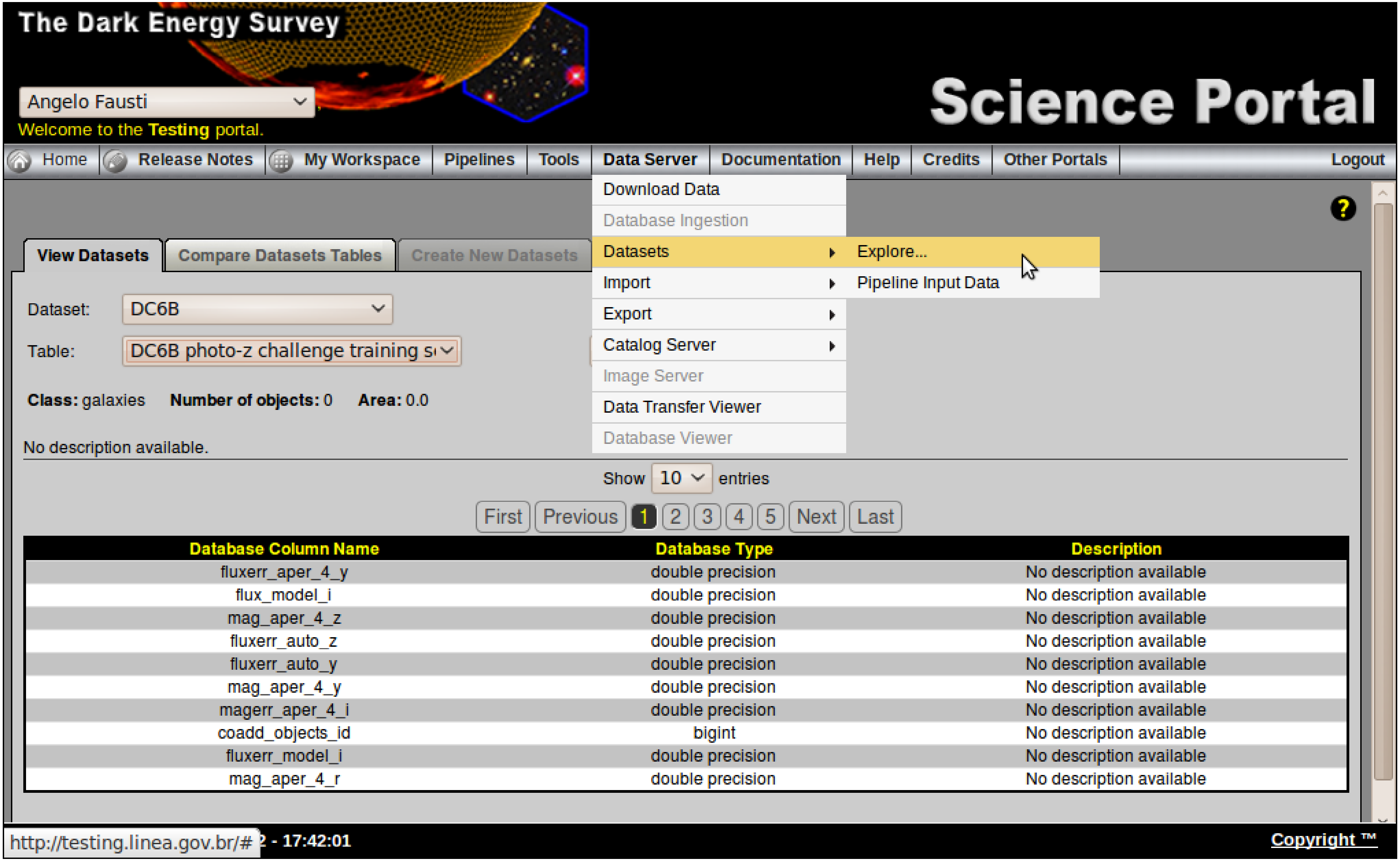
Output

Print Message: Hello World!

Science Portal Web Application

- Pipeline execution
  - Data selection
  - Pipeline configuration
  - Process execution monitoring
- Display, share and add comments on the analysis results
- **“My Workspace” - access process products and configuration**
- **Compare the results of processes with different input data or configuration (e.g Cluster Finder Consolidator)**

Data discovery on the Portal



Dataset: DC6B

Table: DC6B photo-z challenge training s

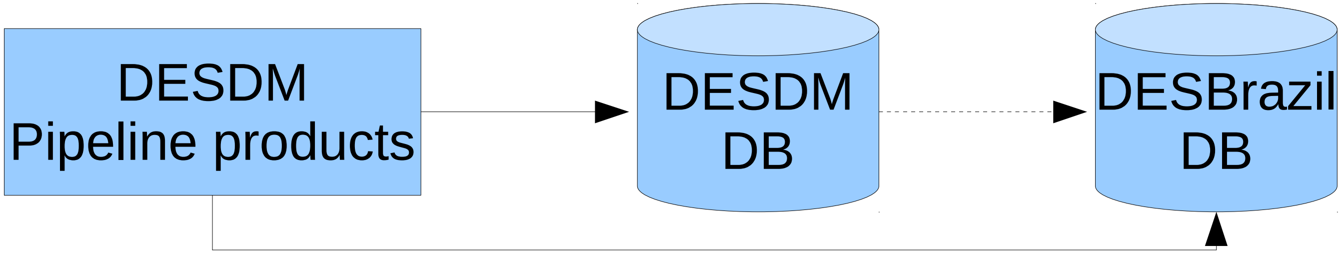
Class: galaxies **Number of objects:** 0 **Area:** 0.0

No description available.

Database Column Name	Database Type	Description
fluxerr_aper_4_y	double precision	No description available
flux_model_i	double precision	No description available
mag_aper_4_z	double precision	No description available
fluxerr_auto_z	double precision	No description available
fluxerr_auto_y	double precision	No description available
mag_aper_4_y	double precision	No description available
magerr_aper_4_i	double precision	No description available
coadd_objects_id	bigint	No description available
fluxerr_model_i	double precision	No description available
mag_aper_4_r	double precision	No description available

Protocol for bulk transfer

- Incremental releases to DES-Brazil based on flat files as soon as they are available/produced by the DESDM pipeline
- Avoid the overhead of querying multi-million objects from DESDM database and (re)produce large FITS files for the full area catalog

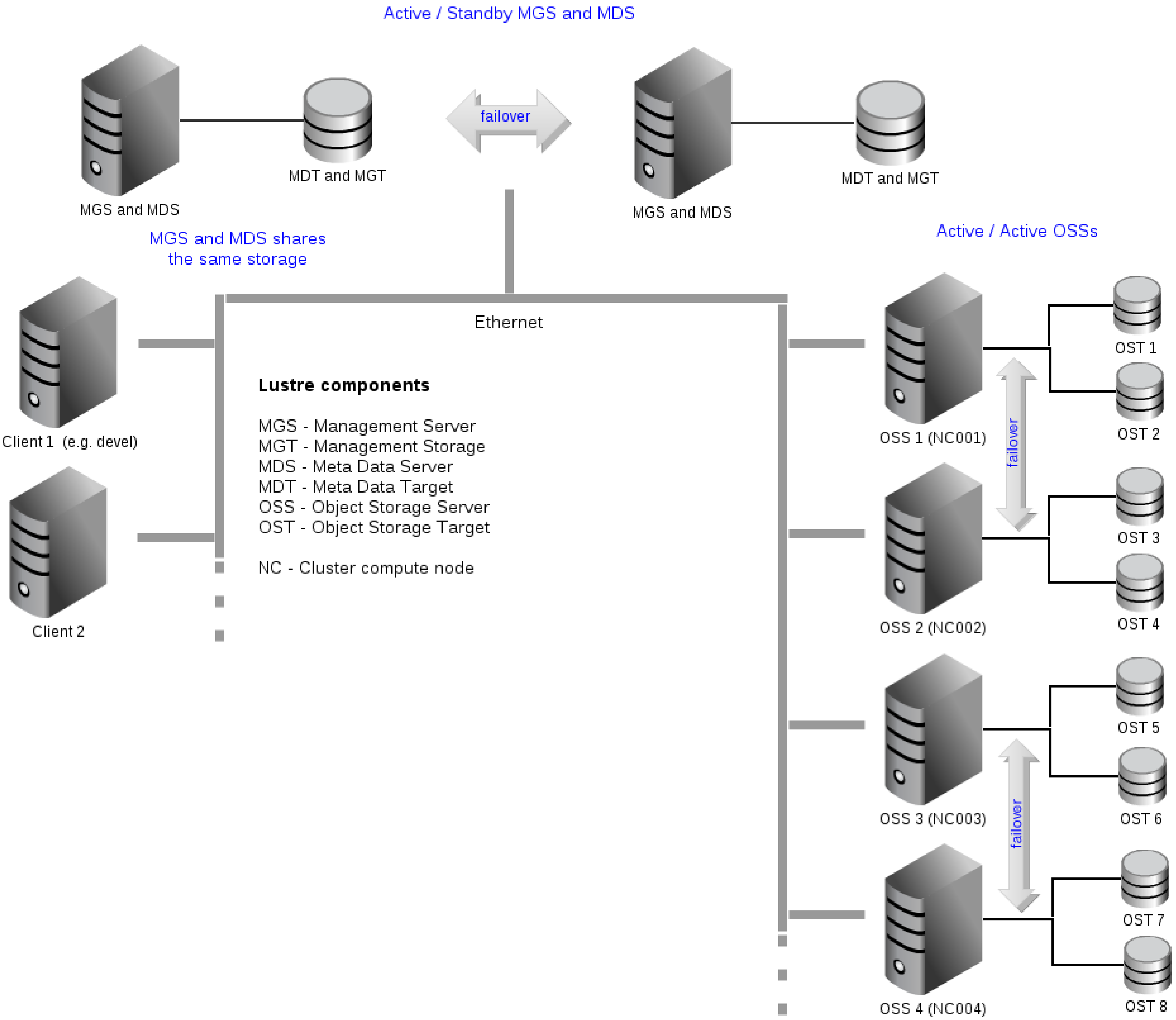


- We don't have to wait the release of the co-added materialized to start to transfer and ingest the data into the portal database
- Increasing our readiness we increase also the interest on the portal for science analysis

BCC Simulation

- 5000 sq deg objects catalog (~528M objects x 100 attributes)
- The data is available in 330 FITS files of about 100 – 200MB (~60 GB total)
- Database load ~ 14h at 10458 lines/s
- Indexing and database clustering using QC3 (similar to HTM) took ~25h
- Auxiliar tables ~ 5h
- Final database size is 161G
- Ingestion time ~55h

LUSTRE topology for /scratch



Lustre components

MGS - Management Server
MGT - Management Storage
MDS - Meta Data Server
MDT - Meta Data Target
OSS - Object Storage Server
OST - Object Storage Target

NC - Cluster compute node