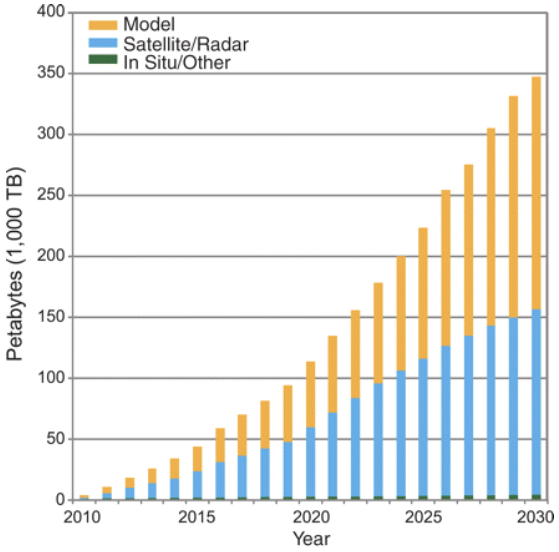
Data near processing support for climate data analysis

Stephan Kindermann, Carsten Ehbrecht
Deutsches Klimarechenzentrum (DKRZ)

Overview

- Background / Motivation
  - Climate community data infrastructure
  - Data processing near data centers needed
- A component system for processing services
- A specific service example
  - Code packaging and deployment
  - Deployment at Data Center / HPC Center / Home Institute / Cloud Infrastructure
- Summary and Outlook

Background: Climate Model Data Processing



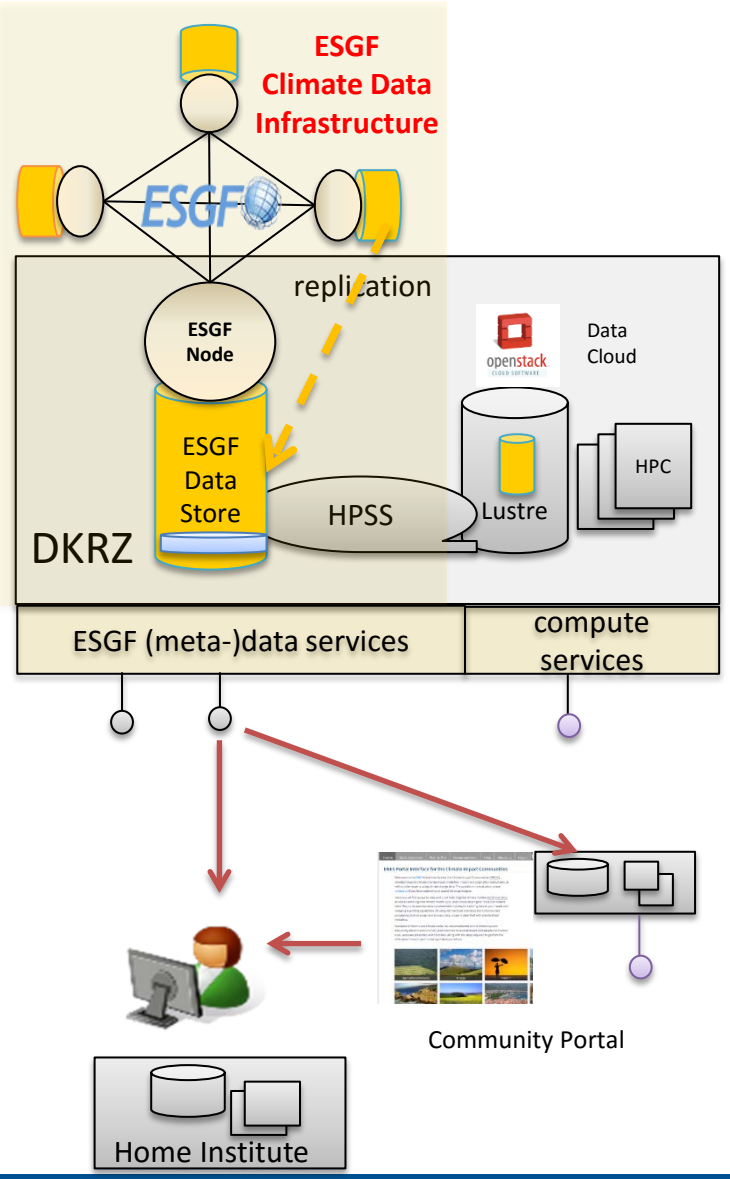
Climate Data Challenges in the 21st Century, Jonathan T. Overpeck, et al. Science 331, 700 (2011); DOI: 10.1126/science.1197869

Main driver for climate data infrastructure development:
Intercomparison Projects

Climate Model Intercomparison Projects (CMIPs):

- CMIP3: ~ 35 TB
- CMIP5: ~ 3 PB = 100x CMIP3
- CMIP6:~ xx PB (> 10x CMIP5)

→ **ESGF / IS-ENES Infrastructure**

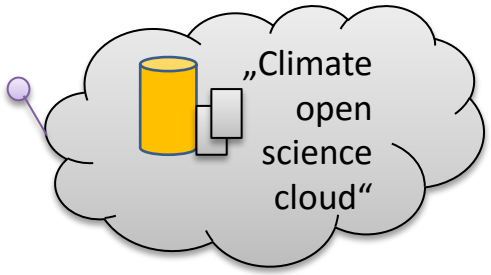


Data Processing:

- „download and process at home“ no longer feasible

→ Data near processing

→ Flexible approach
(… science clouds
are coming …)



Motivation

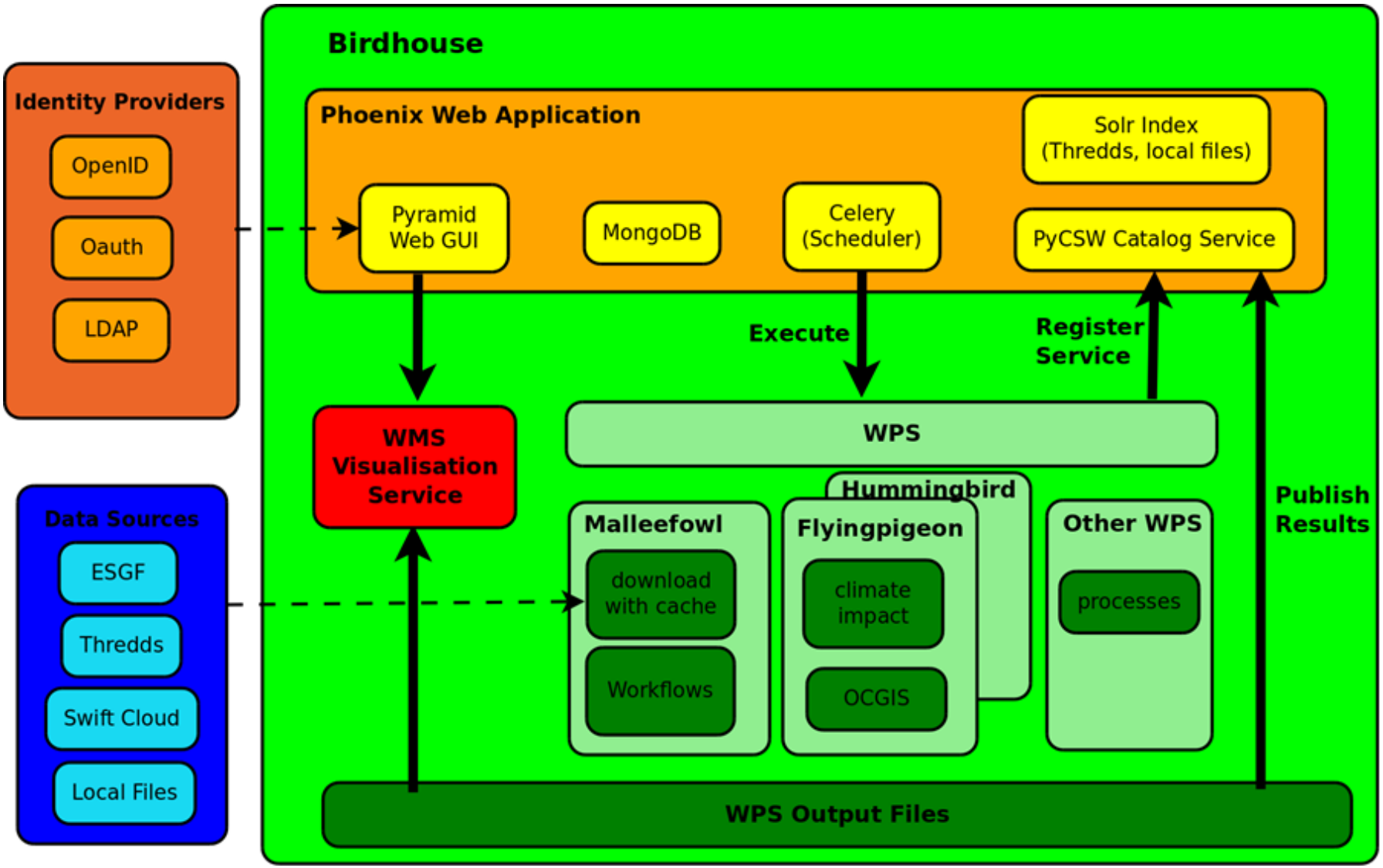
Wanted: A modular climate data processing solution

- Open interfaces
- No re-invention of the wheal: Build on stable open source approaches
- Modular, flexible installation, configuration and deployment system

Approach: An integration solution (*birdhouse*) with an extensible set of processing and data management services (*birds*)

- Based on OGC WPS services (+ other OGC service components)
- Flexible installation and deployment (conda, docker)
- re-usable data management components (ESGF, cloud, thredds data sources)

The Birdhouse approach



Processing Approach – Example bird in birdhouse

Uniform set of packaging recipes

- Maintained on github
  https://github.com/bird-house/conda-recipes
- Available on binstar
  https://anaconda.org/birdhouse/packages

README.rst

conda-recipes for Birdhouse

Additional or customized conda recipes used by Birdhouse components.

The recipes are available on Binstar (Anaconda Server):

https://anaconda.org/birdhouse

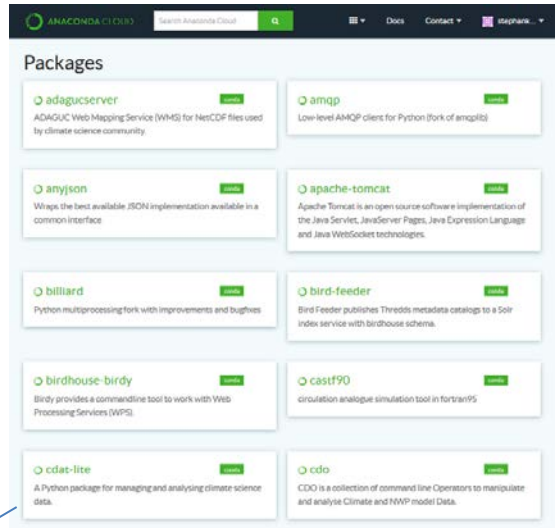
The birdhouse documentation shows how to use conda package and how to create new ones

More docs on building conda packages:

- http://conda.pydata.org/docs/build.html
- http://conda.pydata.org/docs/build_tutorials.html
- http://docs.anaconda.org/draft/examples.html#BinstarBuild
- http://docs.anaconda.org/draft/build-config.html

Other conda channels used:

- http://anaconda.org/ioos (sci-wms, ...)
- http://anaconda.org/nesii (ocgis, icclim)
- http://anaconda.org/scitools (iris)
- http://anaconda.org/r (R packages)
- http://anaconda.org/asmeurer (R packages)



Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz
cdo

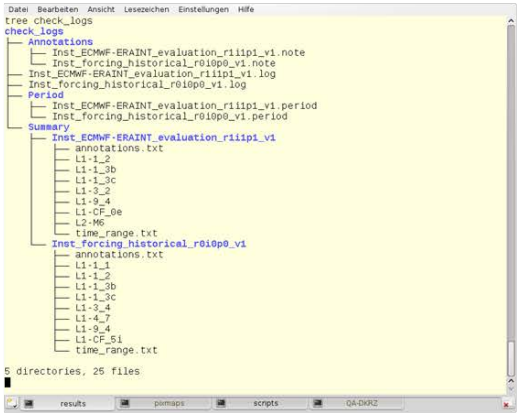
(Executable) QA components

- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

Processing Approach – Example bird in birdhouse

```
Terminal
stephan@stephan-mint1 ~ $ conda install -c birdhouse -c ioos qa-dkrz
Fetching package metadata: ........
Solving package specifications: .........

Package plan for installation in environment /home/stephan/miniconda:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    gmp-5.1.2                  |                2         594 KB
    libuuid-1.0.3              |                2          30 KB
    udunits2-2.2.20            |                0         143 KB
    curl-7.45.0                |                0         528 KB
    hdf5-1.8.15.1              |                2         1.9 MB
    isl-0.12.2                 |                0         1.1 MB
    mpfr-3.1.2                 |                0         407 KB
    cloog-0.18.0               |                0         617 KB
    libnetcdf-4.3.3.1          |                3         892 KB
    mpc-1.0.1                  |                0          61 KB
    gcc-4.8.5                  |                3        65.8 MB
    qa-dkrz-0.5.7              |                6         7.5 MB
    ------------------------------------------------------------
                                                Total:      79.4 MB

The following NEW packages will be INSTALLED:

    cloog:     0.18.0-0
    curl:      7.45.0-0
    gcc:       4.8.5-3
    gmp:       5.1.2-2
    hdf5:      1.8.15.1-2
    isl:       0.12.2-0
    libnetcdf: 4.3.3.1-3
    libuuid:   1.0.3-2
    mpc:       1.0.1-0
    mpfr:      3.1.2-0
    qa-dkrz:   0.5.7-6
    udunits2:  2.2.20-0

Proceed ([y]/n)?
```



Default anconda channel

Birdhouse channel

IOOS (U.S. Integrated Ocean observing System) channel

Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz

cdo

(Executable) QA components

- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

Processing Approach – Example bird in birdhouse

Packaging of components to OGC WPS service

- Recipes again hosted on github
- Include docker target

README.rst

Hummingbird

build passing

Hummingbird (the bird)

Hummingbirds are among the smallest of birds. They hover in mid-air at rapid wing flapping rates, typically arc times per second, but possibly as high as 200 times per second, allowing them also to fly at speeds exceeding (54 km/h; 34 mph), backwards or upside down. [..]. (`Wikipedia https://en.wikipedia.org/wiki/Hummingbird`_).

Hummingbird provides CDO and cfchecker as a Web Processing Service. For installation and configuration read documentation.

Hummingbird is part of the Birdhouse project.

Using docker-compose

Start hummingbird with docker-compose (port 8092) on localhost:

```
$ docker-compose run --service-ports -e HOSTNAME=localhost hummingbird
```

Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz

cdo

(Executable) QA components

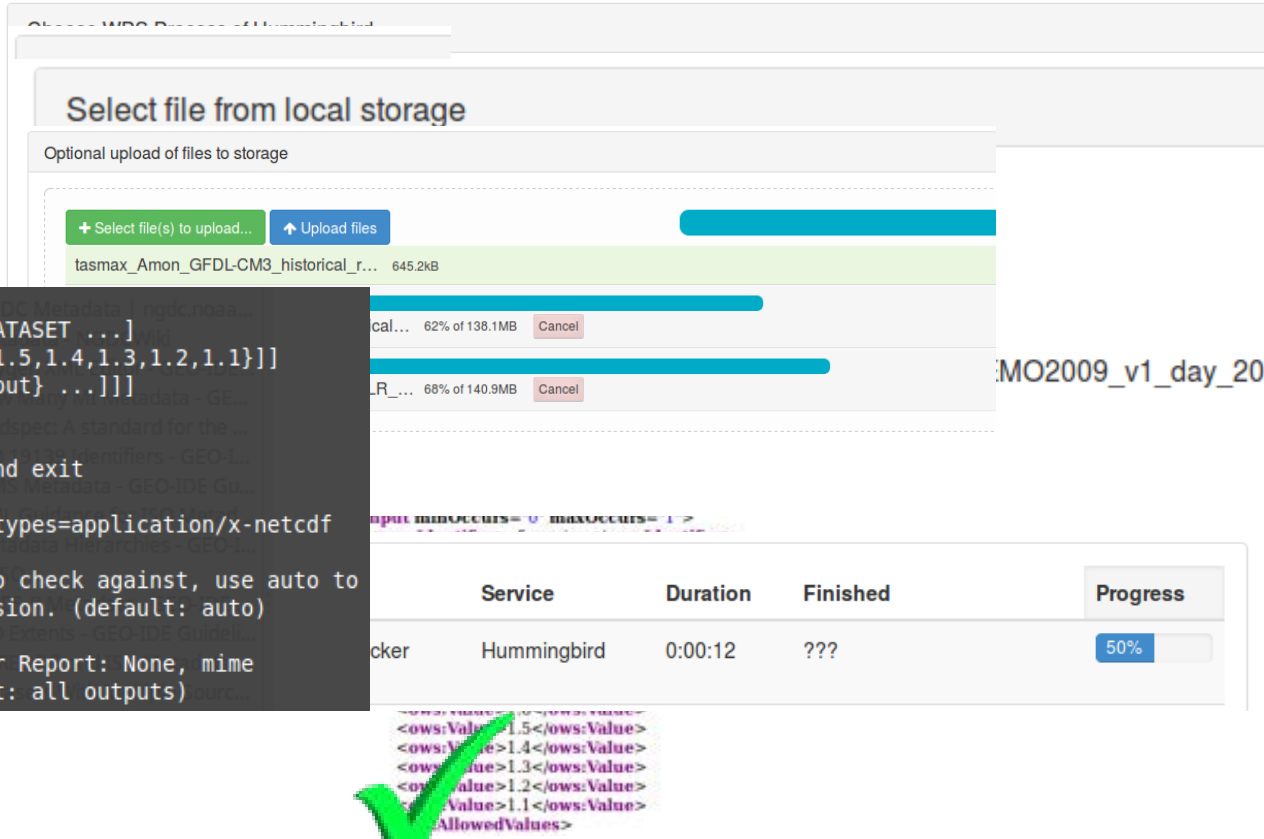
- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

Processing Approach – Example bird in birdhouse

Client Interfaces:

- Ipython notebooks
- Birdhouse GUI
- Birdhouse command line




```
pingu@adelie: ⟫ birdy cfchecker -h
usage: birdy cfchecker [-h] --dataset DATASET [DATASET ...]
                       [--cf_version [{auto,1.6,1.5,1.4,1.3,1.2,1.1}]]
                       [--output [{output} [{output} ...]]]

optional arguments:
  -h, --help            show this help message and exit
  --dataset DATASET [DATASET ...]
                        NetCDF File: None, mime types=application/x-netcdf
  --cf_version [{auto,1.6,1.5,1.4,1.3,1.2,1.1}]
                        CF version: CF version to check against, use auto to
                        auto-detect the file version. (default: auto)
  --output [{output} [{output} ...]]
                        Output: output=CF Checker Report: None, mime
                        types=text/plain (default: all outputs)
```

Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz
cdo

(Executable) QA components

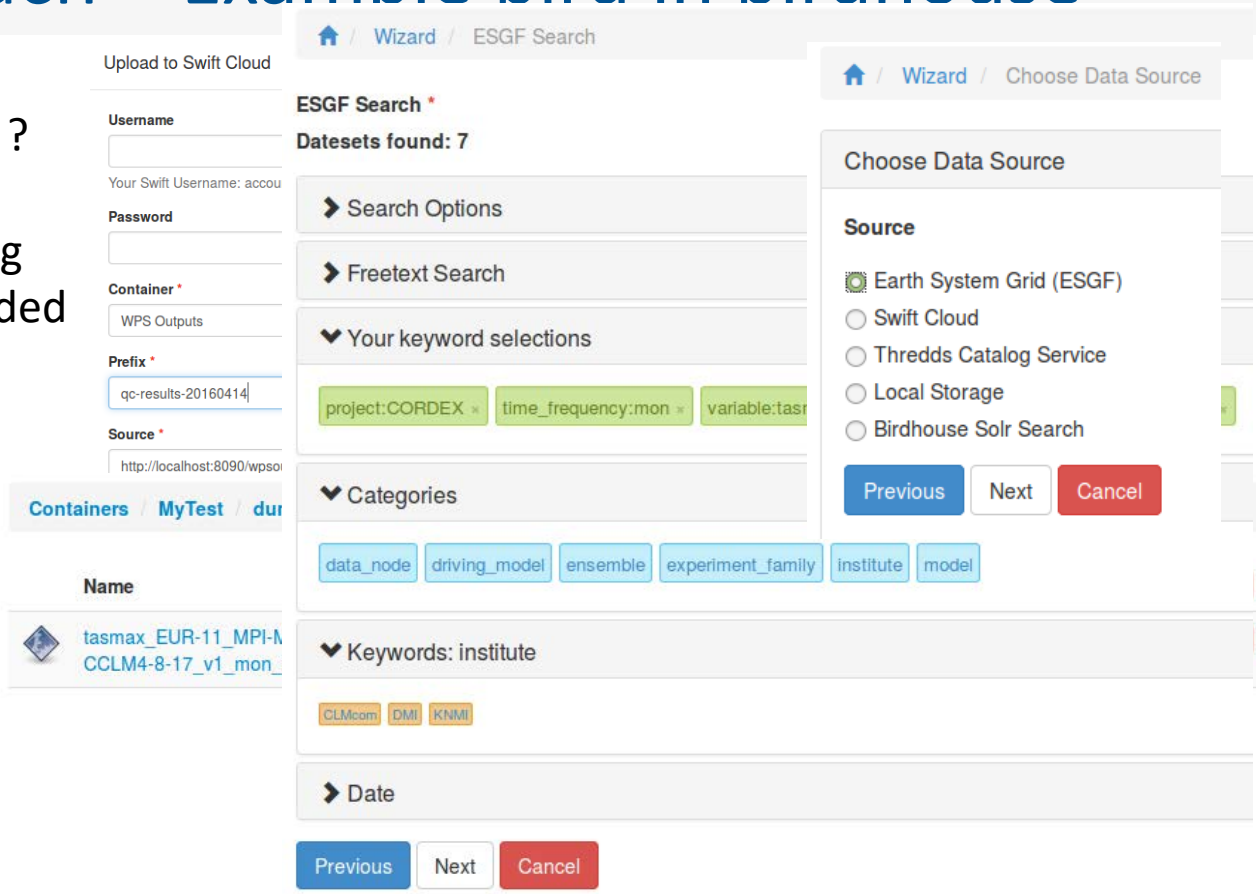
- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

Processing Approach – Example bird in birdhouse

? Real big climate data analysis ?

→ Climate (meta-)data handling components / services are needed



Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz
cdo

(Executable) QA components

- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

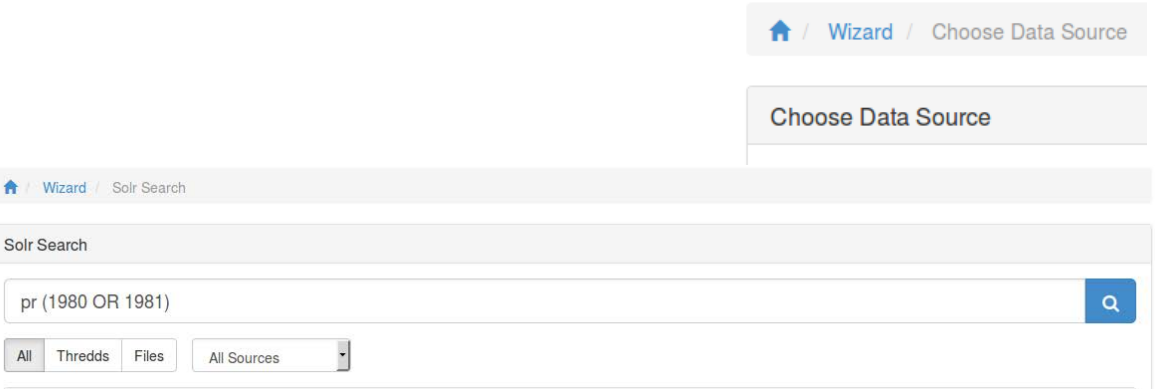
Client Interfaces
(GUI, cmd, jupyter notebook,..)

Processing Approach – Example bird in birdhouse

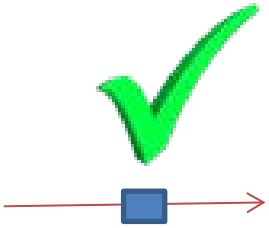
? Real big climate data analysis ?

→ Climate (meta-)data handling components / services are neede

→ Adhere to same birdhouse principles (recipes, packaging,distribution,..)




```
-<wps:ProcessDescriptions xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wp
version="1.0.0" xml:lang="en-CA">
 -<ProcessDescription wps:processVersion="0.2" storeSupported="true" statusSupported="true">
    <ows:Identifier>swift_download</ows:Identifier>
    <ows:Title>Download files from Swift Cloud</ows:Title>
   -<ows:Abstract>
      Downloads files from Swift Cloud and provides file List as JSON Document.
    </ows:Abstract>
   -<DataInputs>
     -<Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>storage_url</ows:Identifier>
        <ows:Title>Storage URL</ows:Title>
       -<LiteralData>
          <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">string</ows:DataType>
          <ows:AnyValue/>
        </LiteralData>
      </Input>
```



Environments
Libraries
Source code

Netcdf4
Udunits
..
python

qa_dkrz

cdo

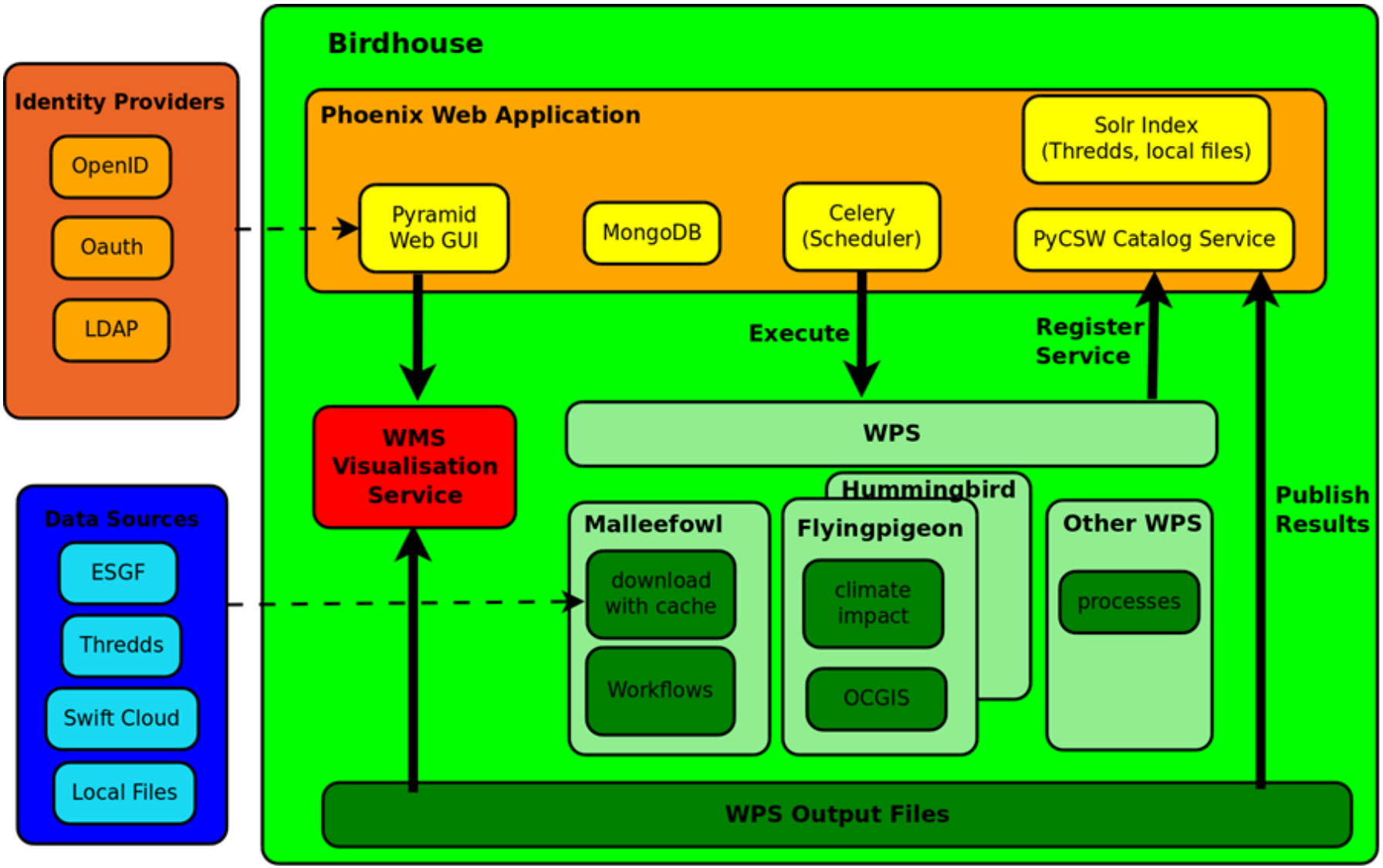
(Executable) QA components

- qa_dkrz
- cf_checker
- cdo_info

Data quality assurance (QA) service
E.g. hummingbird

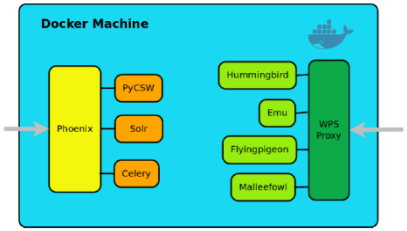
Client Interfaces
(GUI, cmd, jupyter notebook,..)

The Birdhouse approach



Status and Outlook

- **Birdhouse** provides modular system to develop and deploy web processing services
  - HPC center, Data center, (cloud) service provider, scientist
    - code, recipes: https://github.com/bird-house
    - binstar channel: https://conda.anaconda.org/birdhouse,
    - Docker hub: https://hub.docker.com/u/birdhouse
    - documentation: http://birdhouse.readthedocs.org
    - Demo installation: http://mouflon.dkrz.de



Concrete deployment plans:

- DKRZ: generic data services, e.g. quality control
- DKRZ, IPSL: ESGF data processing
- DKRZ, IPSL, BADC: ESGF data processing for Copernicus

Integration plans:

- ESGF: integration with other ESGF OGC WPS deployments at PCMDI, NASA, ..
- EUDAT: collaboration in context of EUDAT generic execution framework (GEF)
- ENVRI+: cross-community harmonization of OGC-WPS processing approaches

Thank You !

Questions ?

Info / Contact:

- http://birdhouse.readthedocs.org
- kindermann@dkrz.de , ehbrecht@dkrz.de