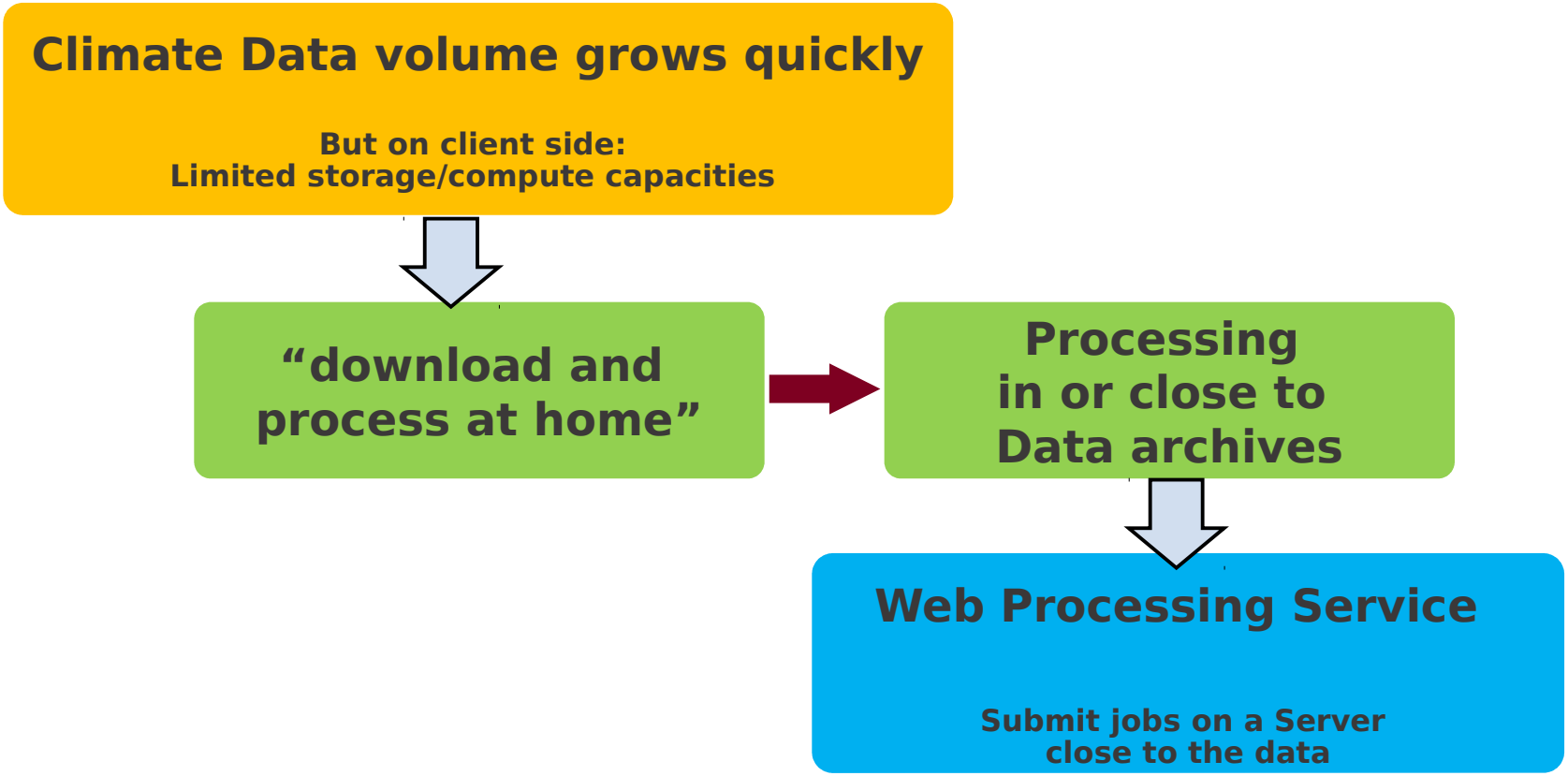
Birdhouse
Supporting Web Processing Services for Climate Data

Stephan Kindermann[1], Carsten Ehbrecht[1], Nils Hempelmann[2] et. al.

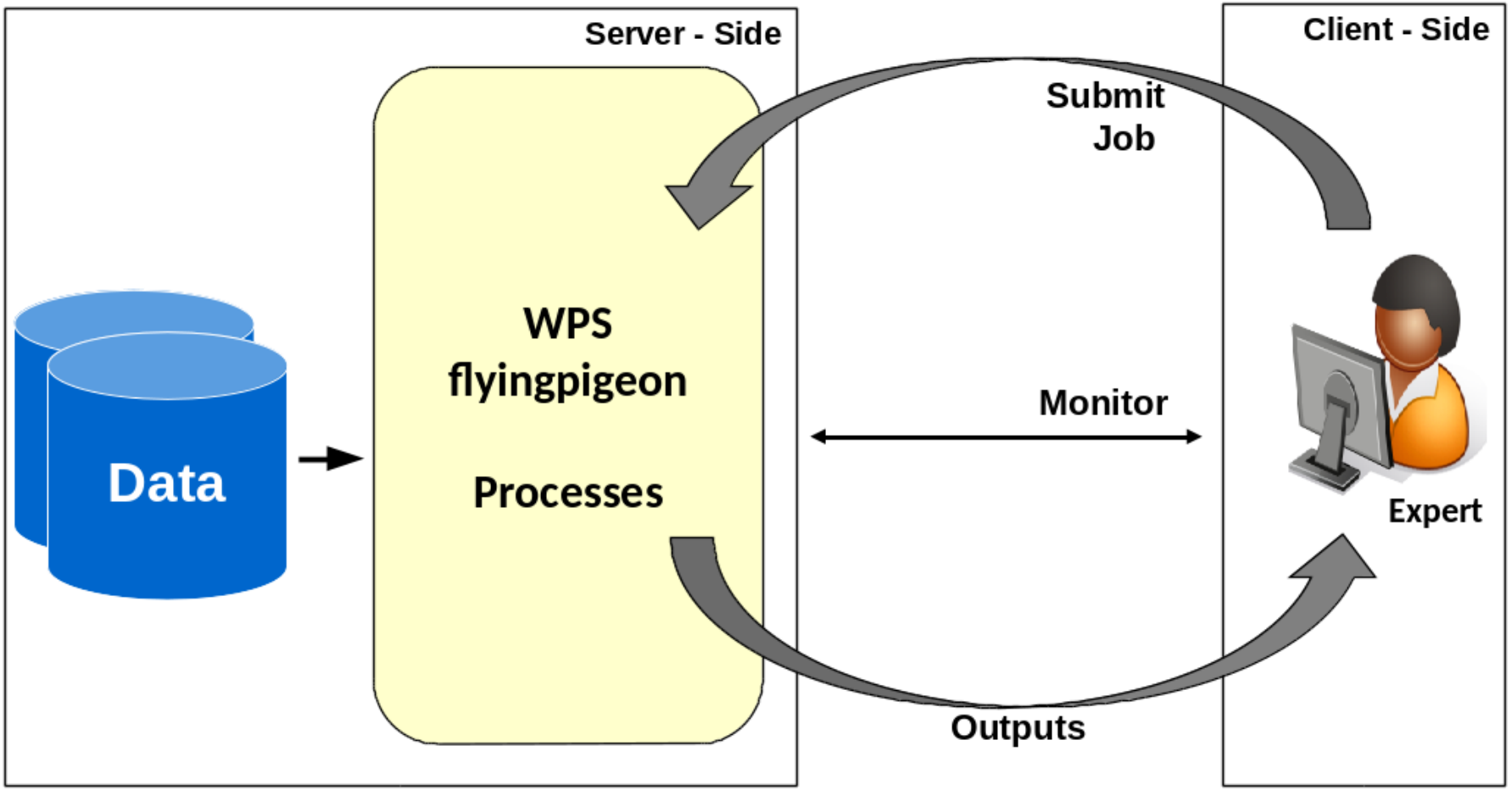
1. German Climate Computing Center, Germany (DKRZ)
2. Le Laboratoire des Sciences du Climat et de l'Environnement, France

ESGF Earth System Grid Federation

Motivation


Climate Data volume grows quickly
But on client side:
Limited storage/compute capacities
“download and process at home”
Processing in or close to Data archives
Web Processing Service
Submit jobs on a Server close to the data

Server-Client Side

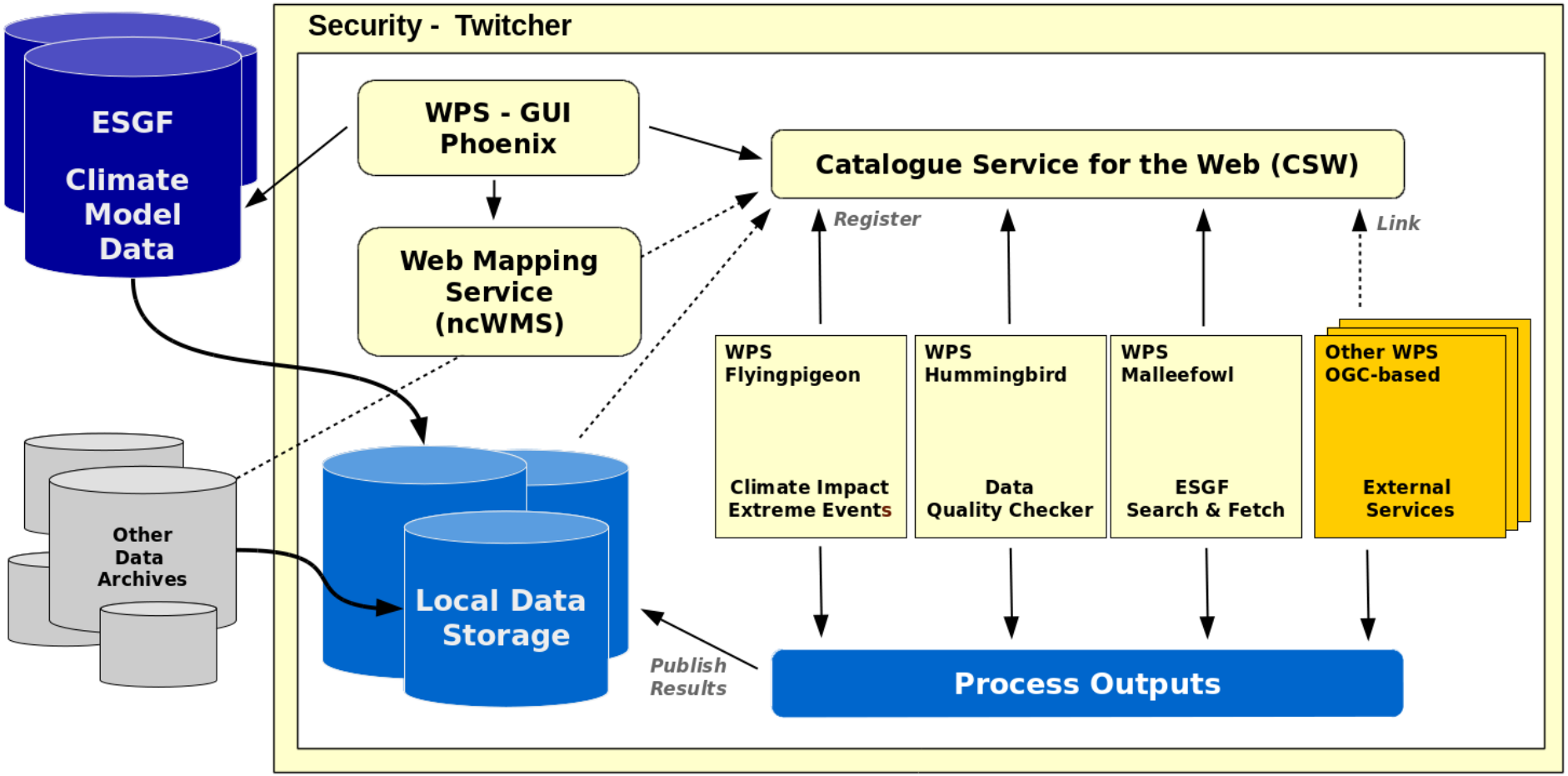


What does Birdhouse provide?

- Customizable installation of Web Processing Services using conda, buildout and ansible.
- Provides WPS as Docker Container.
- Web-based and Terminal WPS clients.
- Security Proxy Twitcher for OGC/WPS services.
- Data Access: ESGF, Thredds, OpenStack, ...
- WPS for compliance checks and climate impact.
- Supports PyWPS 3. and 4.x … but not restricted to it (others: Zoo, GeoServer, COWS, 52North).
- Supports to setup a WPS for your own use cases.

Birdhouse - Ecosystem

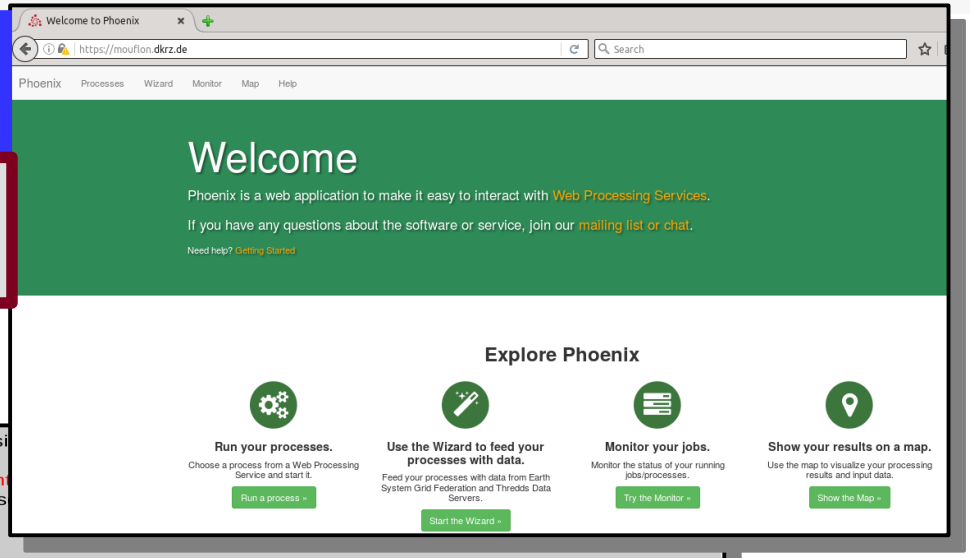
http://bird-house.github.io/



Client Side

Web Browser GUI

Authentication with OAuth or OpenID



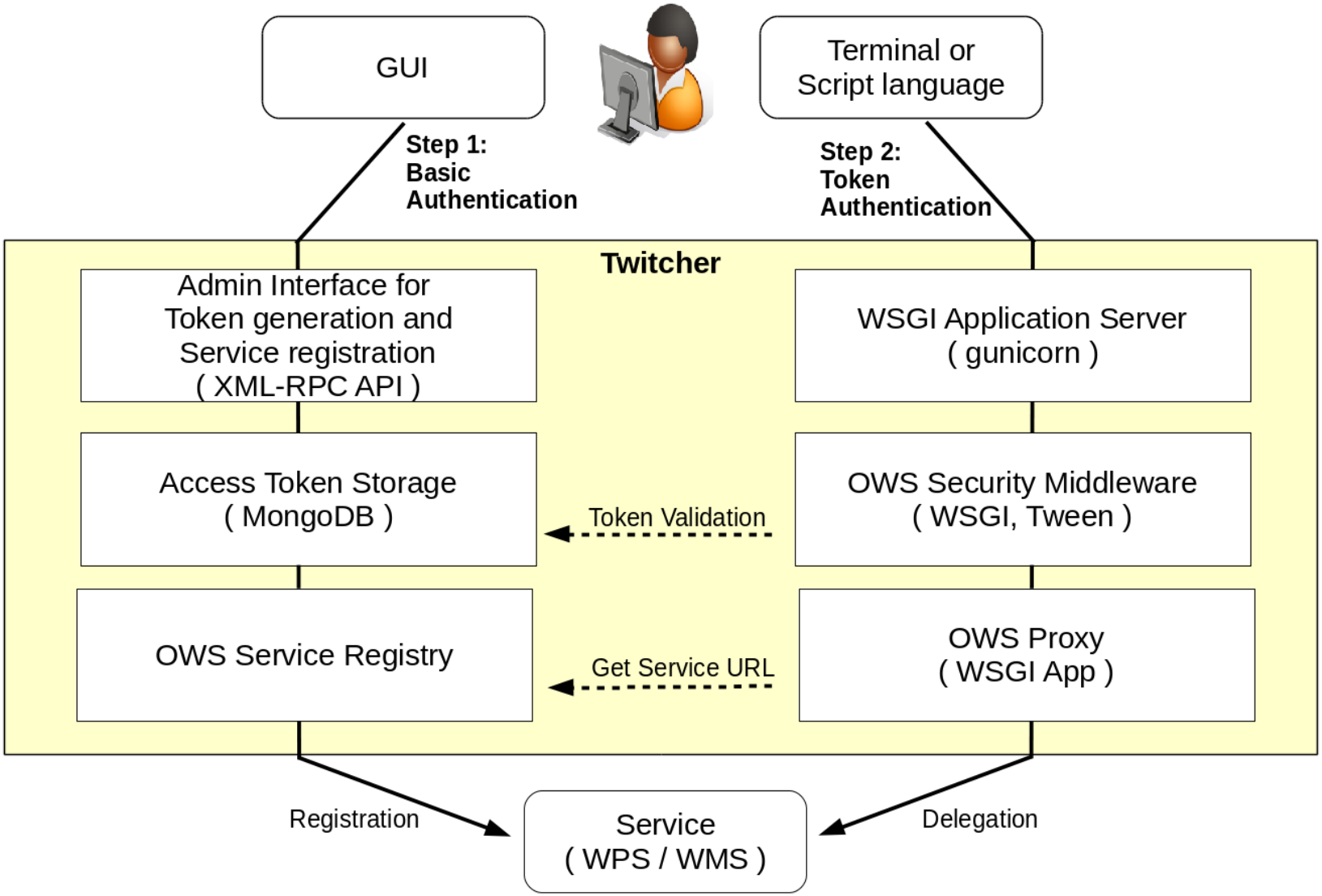
Script Language Terminal Call

Token Authentication

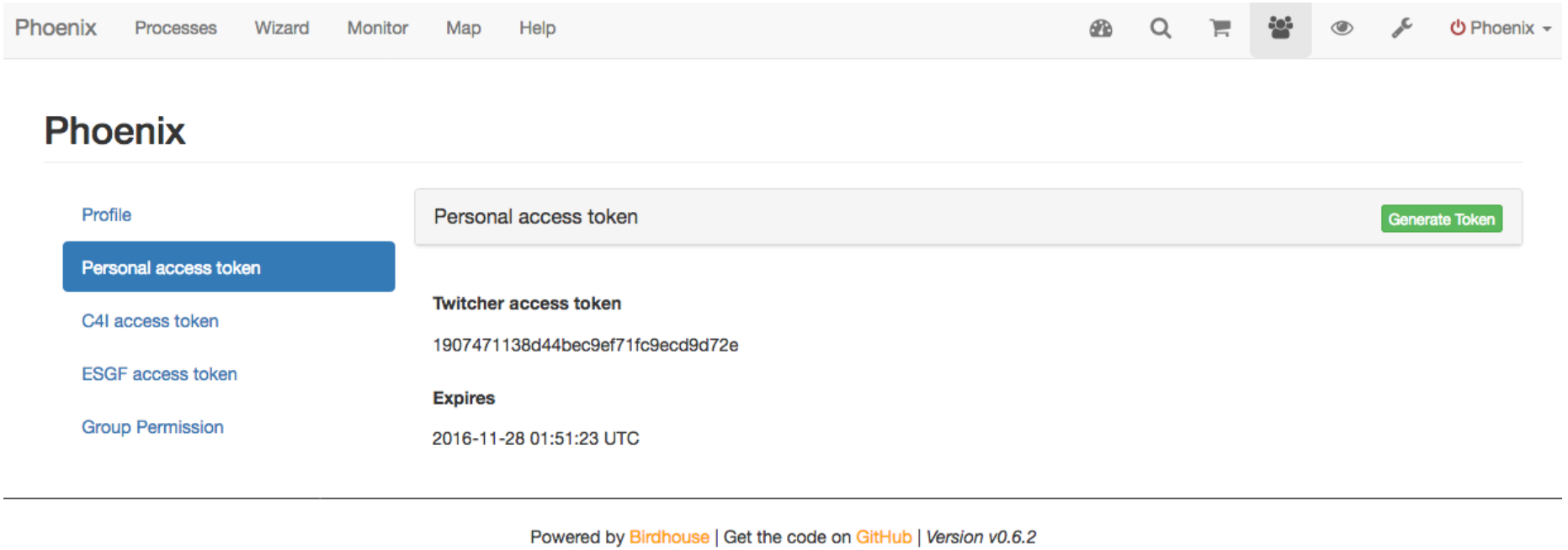
```
from owslib.wps import WebProcessi

wps = WebProcessingService(url="ht
                           verbose=Fals

execute = wps.execute(
    identifier="niceprocess",
    inputs=[
    ("parameter_1", "argument"),
    ("parameter_2", "42"),
#     ("parameter_3", "0.987"), # use the default value
    ("file_identifier", "https://thredds/fileServer1/test/file1.nc"),
    ("file_identifier", "https://thredds/fileServer1/test/file2.nc"),
    ("file_identifier", "https://thredds/fileServer2/test/file3.nc")],
    output=[("output", True)])

# time for a coffee

for o in execute.processOutputs:
    print o.reference

https://mouflon.dkrz.de:8090/wpsoutputs/flyingpigeon/output_graphic-697dee76-d722-93ae-9789bf75cf44.png
https://mouflon.dkrz.de:8090/wpsoutputs/flyingpigeon/output_netCDF-697dee76-d722-93ae-9789bf75cf44.nc
https://mouflon.dkrz.de:8090/wpsoutputs/flyingpigeon/output_text-697dee76-d722-93ae-9789bf75cf44.txt
```

```
[nhempel@lsce3199 ~]$ export WPS_SERVICE=http://birdhc

[nhempel@lsce3199 ~]$ birdy -h

usage: birdy [<options>] <command> [<args>]

Flyingpigeon: Processes for climate data, indices and extrem events

optional arguments:
  -h, --help            show this help message and exit
  --debug               enable debug mode

command:
  List of available commands (wps processes)

  {visualisation,sdm,segetalflora,indices_single,subset_countries,eobs_to_cordex,ensembleRobustness,analogs,fetch}
                        Run "birdy <command> -h" to get additional help.

    visualisation       Visualisation of netcdf files:                  Just testing a nice script to visualise some variables
    sdm                 Species distribution model:                     Species distribution model
    segetalflora        Segetal Flora:                                  Species biodiversity of segetal flora.  Imput files: variable:tas , domain: EUR-
                                                                        11 or EUR-44
    indices_single      Calculation of climate indice (single variable): This  process calculates climate indices based on one single variable.
    subset_countries    Subset netCDF files:                            This process returns only the given polygon from input netCDF files.
    eobs_to_cordex      EOBS to CORDEX:                                 downloads EOBS data in adaped CORDE format
    ensembleRobustness  Calculation of the robustness of an ensemle:    Calculates the robustness as the ratio of noise to
                                                                         signal in an ensemle of timeseries
    analogs             Days with analog pressure pattern:              Search for day with analog pressure pattern
    fetch               Download Resources:                             This process downloads resources (limited to 50GB) to the local file system
                                                                        and returns a textfile with appropriate pathe
```

Security – Twitcher Security Proxy



Security Token



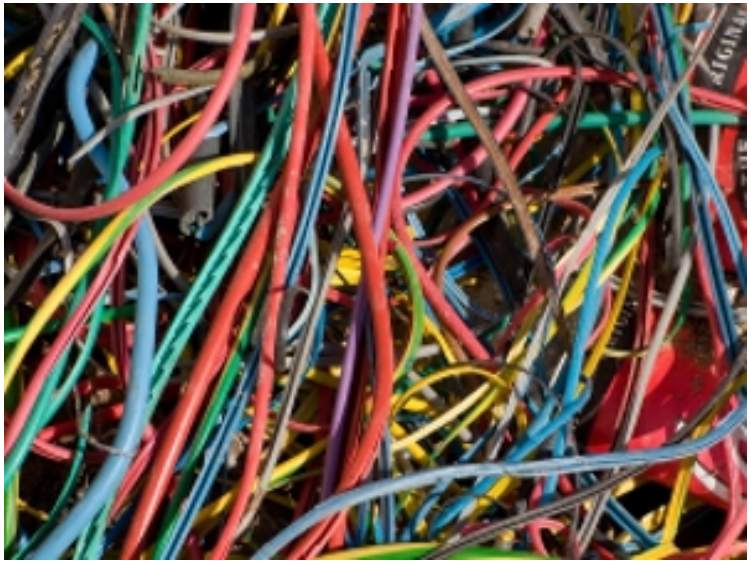
The token-based Security Proxy is similar to the Climate4Impact approach:
https://dev.knmi.nl/projects/impactportal/wiki/API

Script language

```
from owslib.wps import WebProcessingService, monitorExecution

#  using wps url with access token db6c...
wps = WebProcessingService(
                url="https://mouflon.dkrz.de/ows/proxy/flyingpigeon/db6c1293d0444d919dcc3ce48fa610f7 ", \
                verify=False,
                verbose=False, skip_caps=False,
                )

execute = wps.execute(
   identifier="niceprocess",
   inputs=[
   ("parameter_1", "argument"),
   ("parameter_2", "42"),
#    ("parameter_3", "0.987"), # use the default value
   ("file_identifier", "https://thredds/fileServer1/test/file1.nc"),
   ("file_identifier", "https://thredds/fileServer1/test/file2.nc"),
   ("file_identifier", "https://thredds/fileServer2/test/file3.nc")],
   output=[("output", True)])

for o in execute.processOutputs:
   print o.reference

https://mouflon.dkrz.de:8090/wpsoutputs/flyingpigeon/output_graphic-697dee76-d722-93ae-9789bf75cf44.png
https://mouflon.dkrz.de:8090/wpsoutputs/flyingpigeon/output_netCDF-697dee76-d722-93ae-9789bf75cf44.nc
```

Deployment with conda and buildout

“Manage the Chaos”



Using conda package manager to setup an environment with all used software components (Python, R, matplotlib, PyWPS, …).

Using Buildout to setup PyWPS with all services (supervisor, gunicorn, nginx) and configuration files.

To install a *Bird* just run:

```
$ git clone https://github.com/bird-house/emu
$ cd emu
$ make clean install
$ make start
$ http://localhost:8094/wps
```

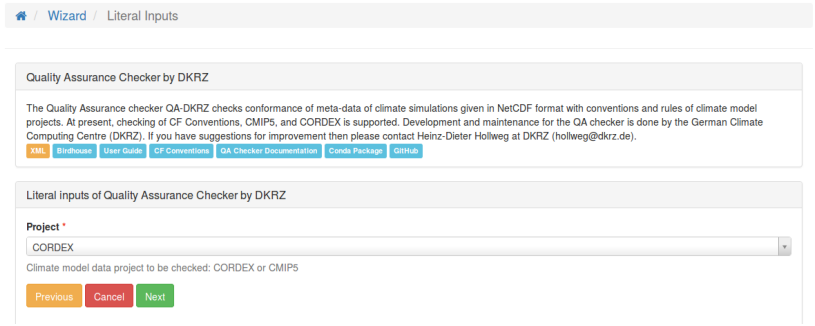
docker

A Dockerfile is generated with the Buildout setup for each WPS.

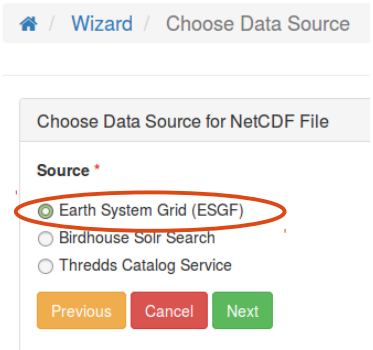
http://conda.pydata.org/docs/
http://www.buildout.org/en/latest/
http://birdhouse.readthedocs.io/en/latest/installation.html

Wizard: Using ESGF-Search

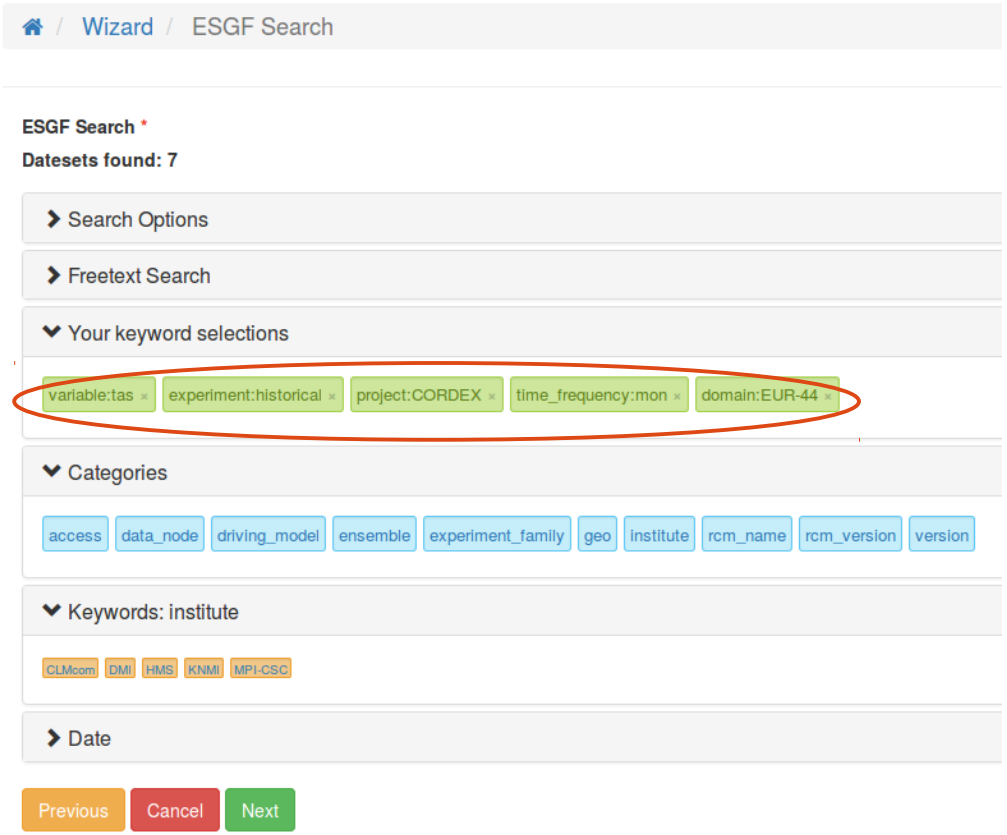
1. Use the Wizard to select Process and enter Parameters



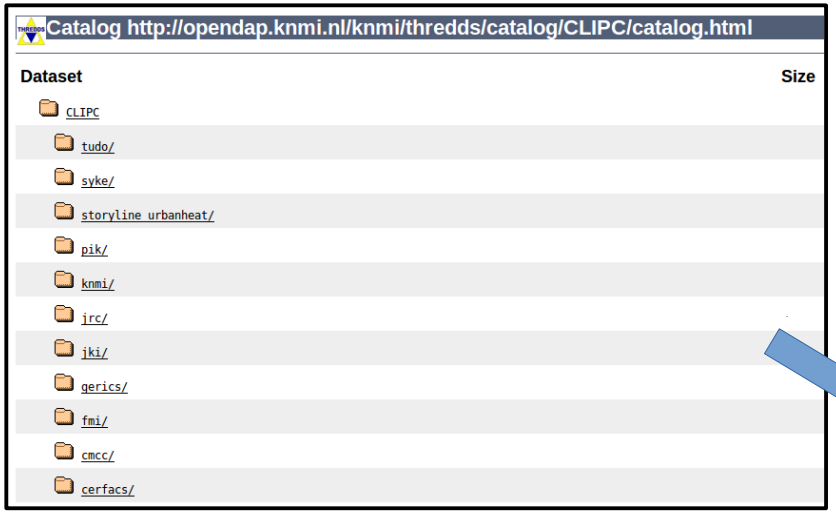
2. Choose ESGF as Input Source for NetCDF Files



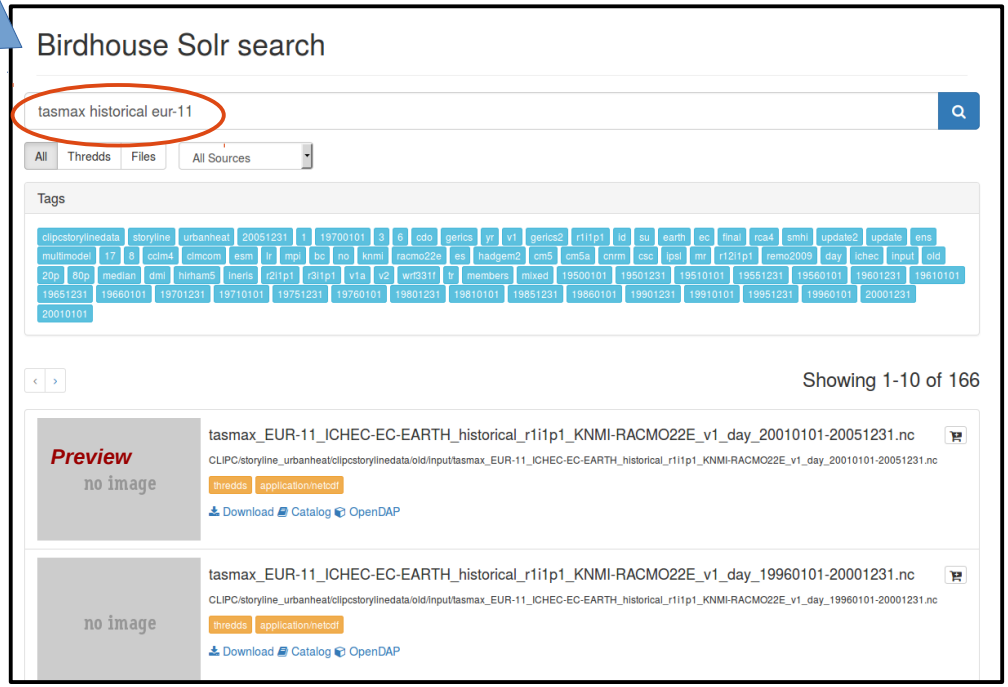
3. Select Datasets with ESGF Search Widget



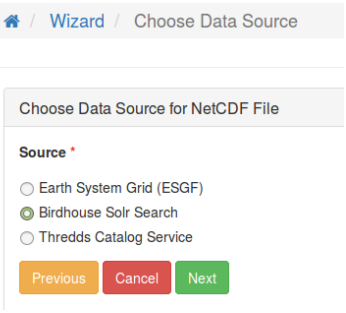
Solr Index for Thredds Catalogs (bird-feeder)



Run bird-feeder to create Solr Search Index for Thredds Data Catalogs and local filesystems.

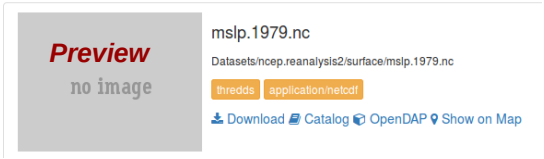


Wizard:
Choose Solr Search of Thredds Catalog as Input Source

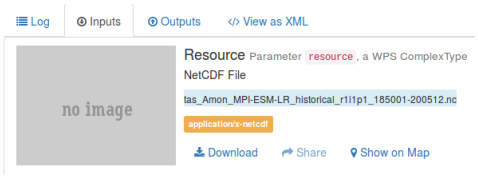


Web Map Service: Show NetCDF Files on Map

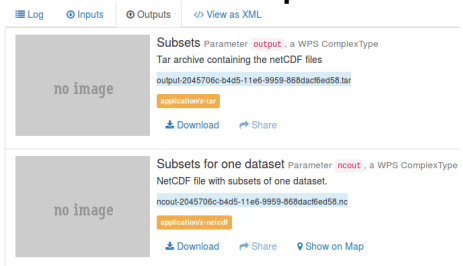
Browse Catalog



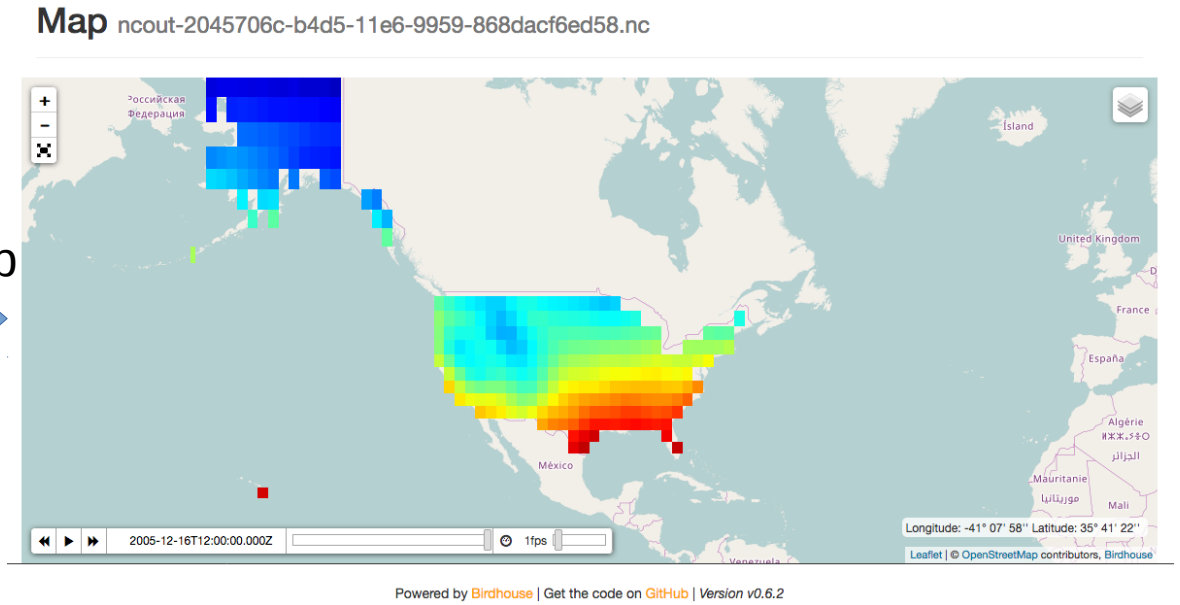
Process Inputs



Process Outputs

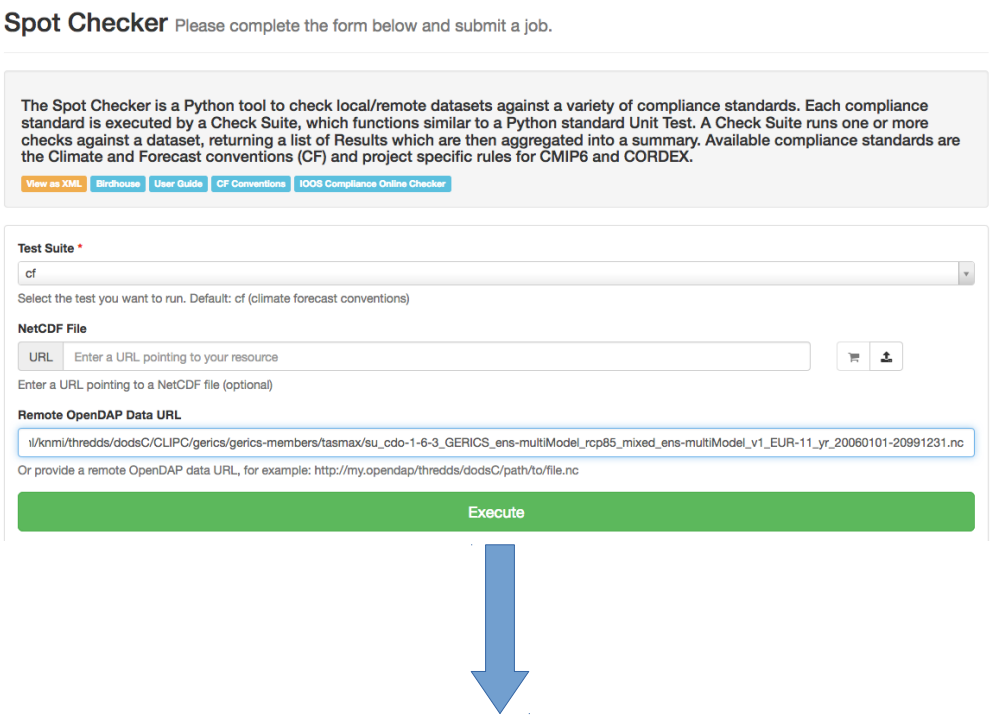


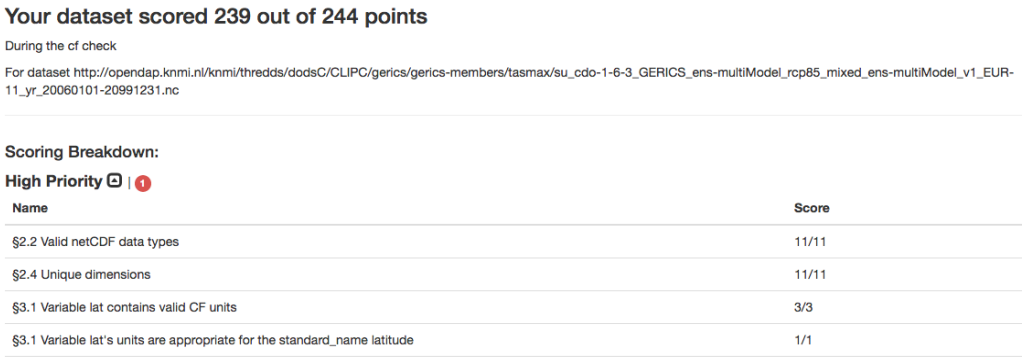
Show Map



Possible Web Map Services:
ncWMS, Adaguc (KNMI), sci-wms (planned)

SpotChecker: Metadata Compliance Checks





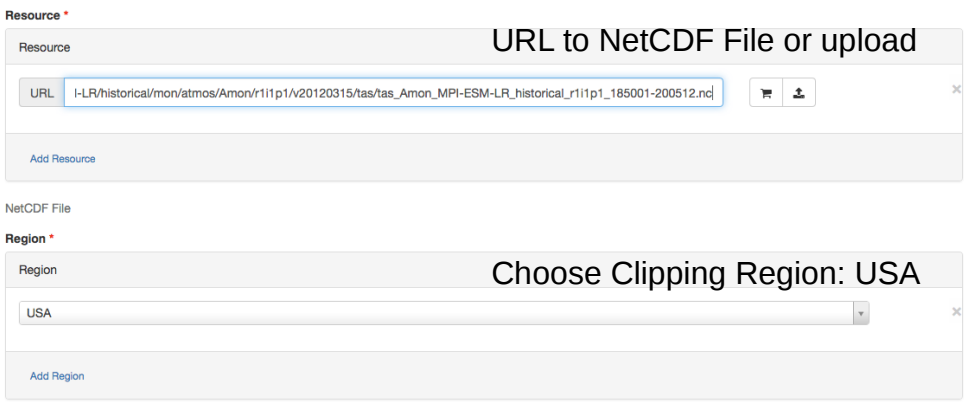
Run SpotChecker on NetCDF File:
File URL, OpenDAP URL or upload File

Perform Compliance Checks:
CF Conventions, CORDEX, CMIP5, …

Using IOOS Compliance-Checker and DKRZ Quality Assurance Checker.

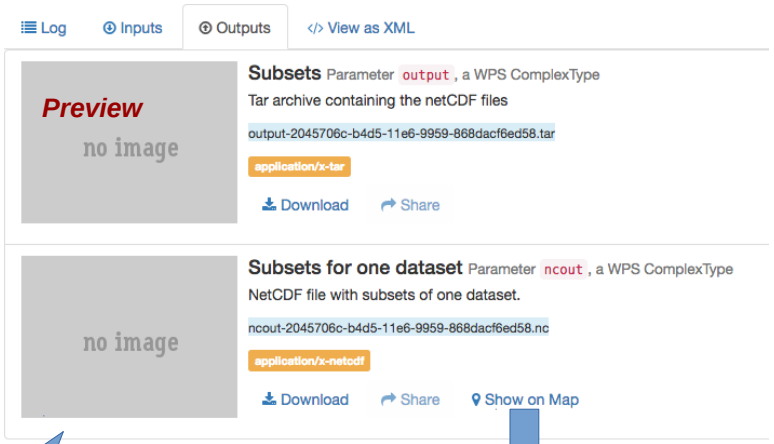
Check Reports in HTML and YAML Format.

Subsetting: Region USA

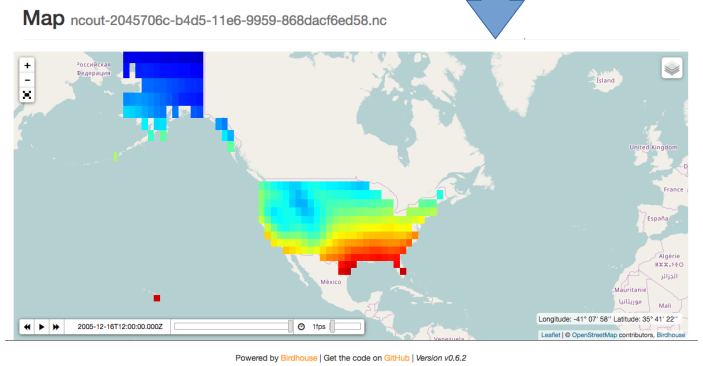


Execute Job

Details



Show Map

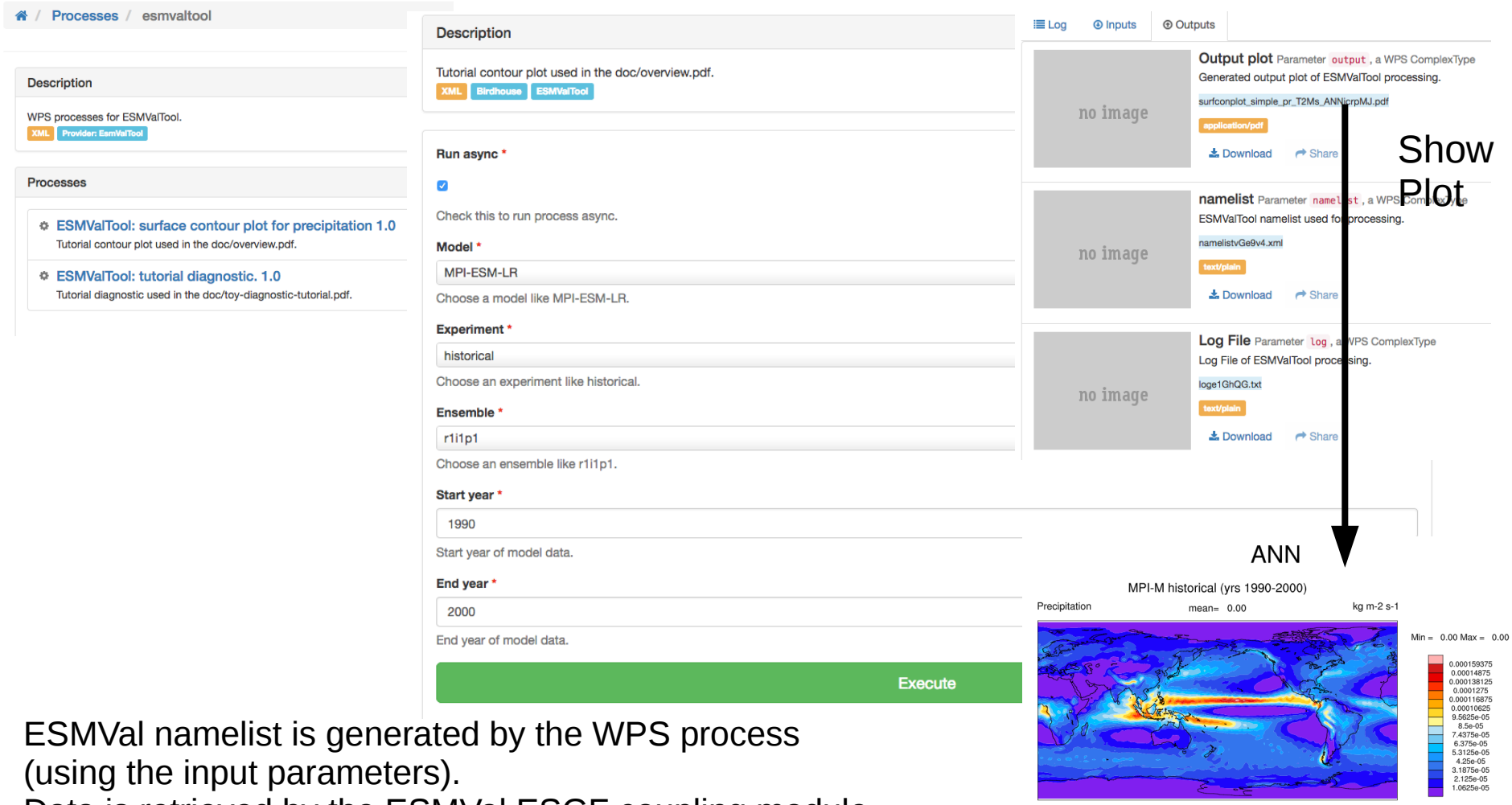


ESMValTool Diagnostices as Web Processing Service

1. Choose ESMVal Process

2. Enter Input Parameters

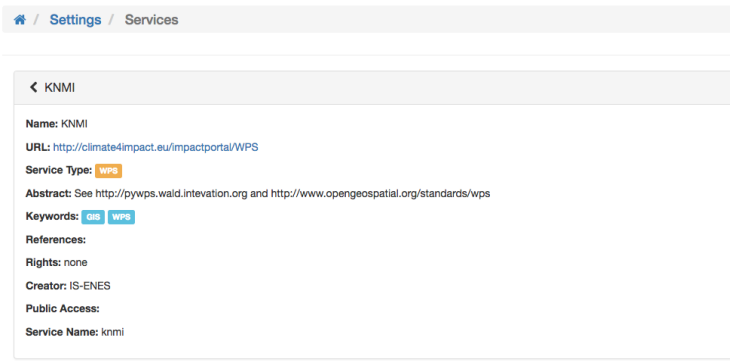
3. Outputs: plot, namelist, log



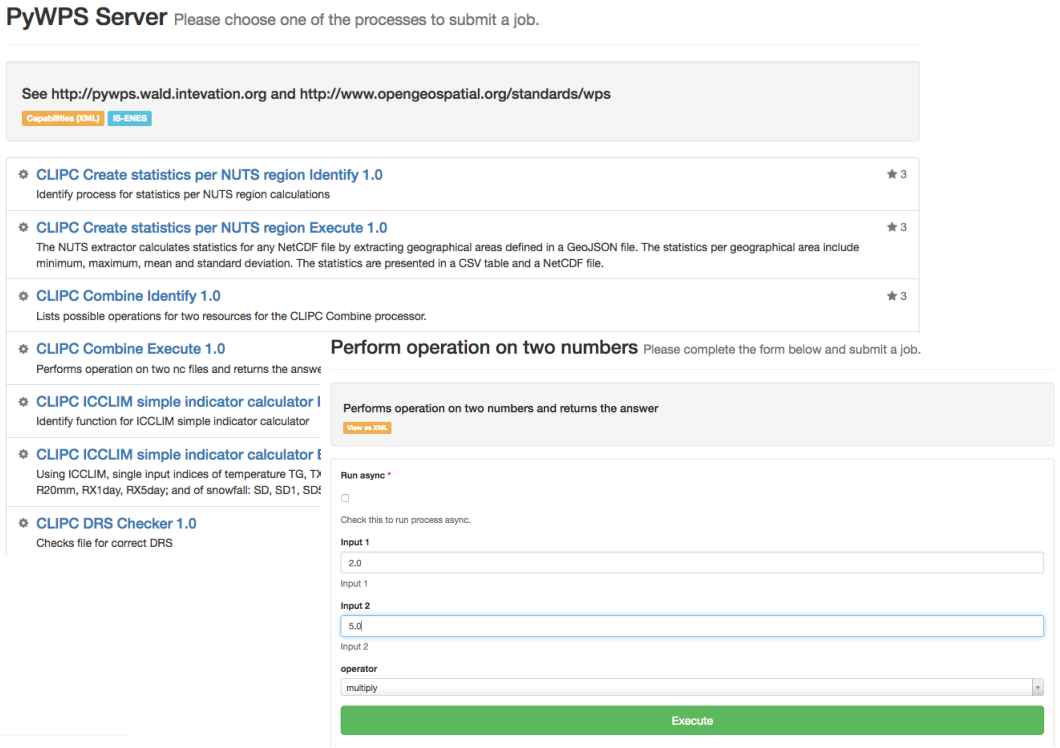
ESMVal namelist is generated by the WPS process (using the input parameters).
Data is retrieved by the ESMVal ESGF coupling module.

Accessing Remote WPS: Climate4Impact WPS at KNMI

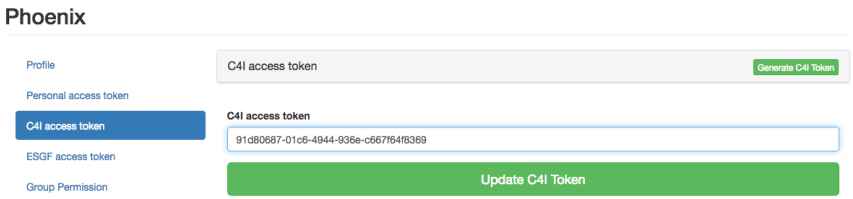
Register C4I WPS URL

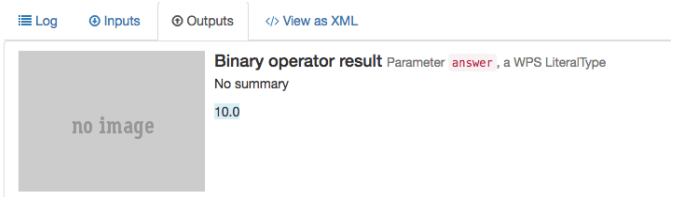


Run a Process: operation on two numbers



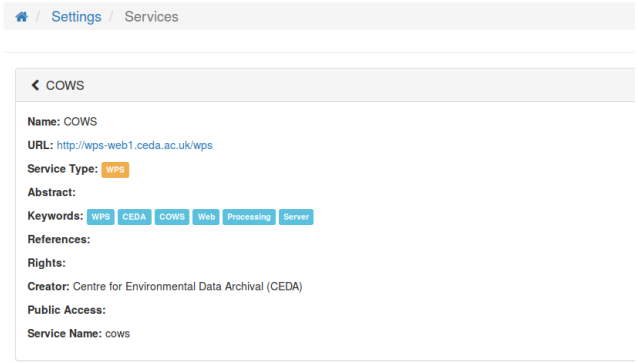
Update C4I Access Token



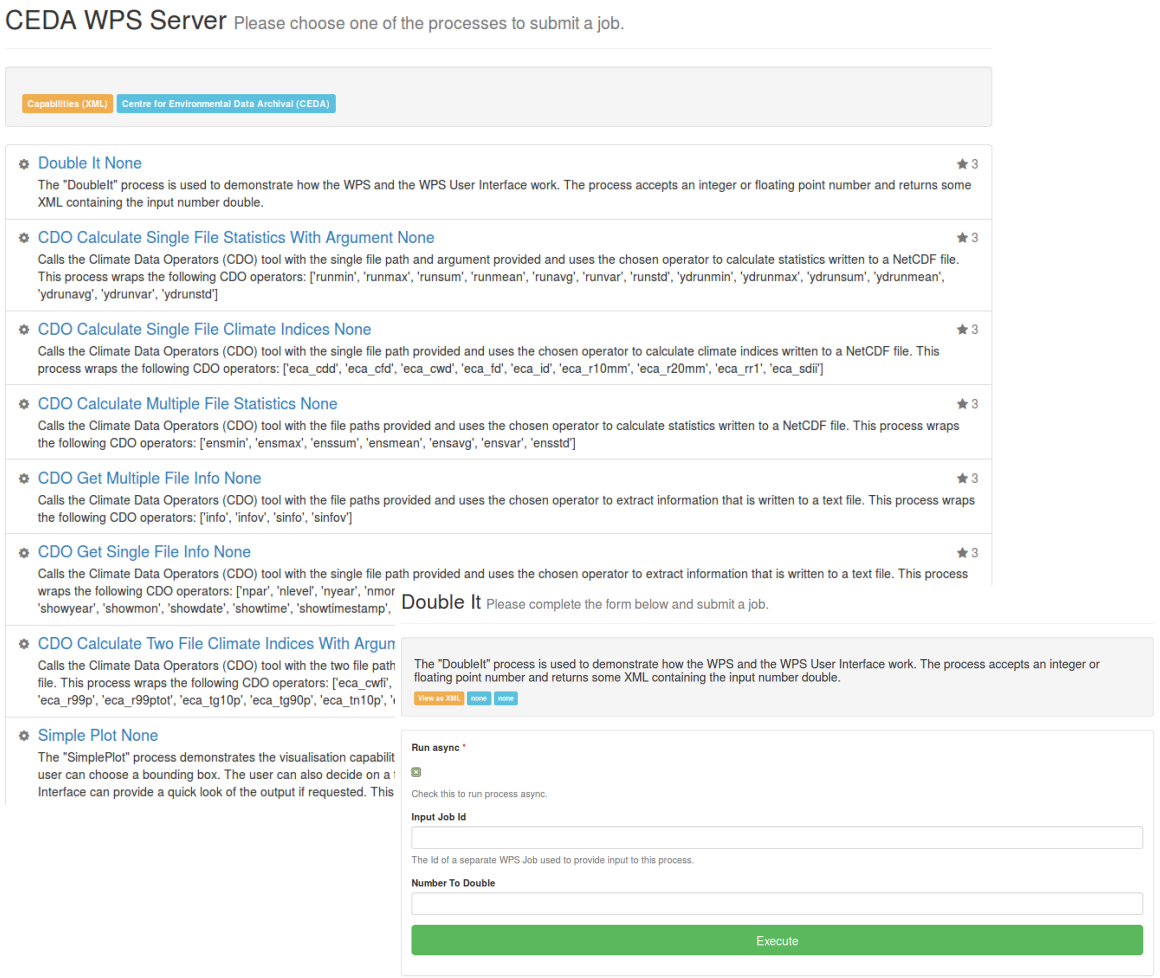


Accessing Remote WPS: COWS WPS at CEDA

Register COWS WPS URL



Run a Process: Double It



Copernicus Extensions for PyWPS

- Using PyWPS-4: ready for WPS 2.0 (pause, resume, delete)
- Attach batch processing with SLURM etc ..
- Optionally run processes in Docker Container.
- Delegation to SLURM and Docker is handled internally of PyWPS (new feature in PyWPS-4).
- WPS Process definition and code is not changed when run as batch job or in a docker container.

Links

- http://bird-house.github.io/
- http://birdhouse.readthedocs.io/en/latest/

- https://gitter.im/bird-house/birdhouse
- https://lists.dkrz.de/mailman/listinfo/wps
- https://lists.dkrz.de/mailman/listinfo/wps-dev

- Demo: https://mouflon.dkrz.de/

Contact:

wps@dkrz.de

Thanks to:

Carmen Alvarez-Castro, Katharina Berger, Patrick Brockmann, Carsten Ehbrecht, Wolfgang Falk, Nils Hempelmann, Heinz-Dieter Hollweg, Jörg Hoffmann, Nikolay Kadygrov, Stephan Kindermann, Florian Klemme, Nikolay Koldunov, Ben Koziol, Cathy Nangini, Sabine Radanovics, Seckmag, Robert Vautard, Pascal Yiou, …., et. al.

..

Additional slides