Web Processing Services (WPS) over IPSL Earth System Grid Federation (ESGF) node

Nikolay Kadygrov[2,1], Sebastien Denvil[1], Nils Hempelmann, Carsten Ehbrecht[3], Soulivanh Thao[2] and Pascal Yiou[2]

(1) Institute Pierre Simon Laplace, Paris, France

(2) Climate and Environment Sciences Laboratory (LSCE), Gif-sur-Yvette, France

(3) German Climate Computing Center (DKRZ), Hamburg, Germany

And:

Birdhouse developers: https://github.com/orgs/bird-house/people

IPSL WPS Training, 15 March 2018

Motivation and Objectives

- The amount of climate data archives are huge and will continuously increase during the next 5 years
- IPSL locally holds ~ 450 Tb of model replicas along with observations and reanalysis data (CMIP5, CORDEX, obs4MIPs etc.)
- The aim is to let scientist do research and perform calculation not locally, after downloading vast amount of data, but remotely at HPC close to the data archives. For that purpose, WPS were installed at IPSL in test mode.

Workshop topics

- What is WPS?
- IPSL Earth System Grid Federation (ESGF) Node
- Birdhouse WPS: Existing processes and scientific needs
- TP: Installation, configuration, add your own WPS process

What is WPS?

- The very short answer – WPS is acronym for Web Processing Services

- The slightly longer answer: Say you have a function (maybe written in Python) which might calculate the “summer days in Finland since 1990”. Then this function has probably input parameters (region, from-date, to-date, NetCDF files, ...) and an output (or even more ...) which might be just an integer number or a text document or even a nice diagram. Now, you would like to provide this function as a web service, so that other people can call it with just a simple URL like: http://myhost/wps/identifier=summer_days&region=finland&from=1990

What is WPS?

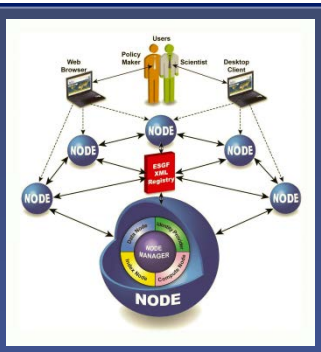
- WPS offers a simple web-based method of finding, accessing, and using all kinds of calculations and models.
- WPS is an OGC standard that defines how to implement geographic calculations or models (i.e. "processes") as a web service. Processes can include any algorithm, calculation or model that operates on spatially referenced data.
- WPS uses standard HTTP and XML as a mechanism for describing processes and the data to be exchanged and provides rules for inputs and outputs (requests-responses).
- The data required by the WPS can be delivered across a network or they can be available at the server.

Scientist: "*I tuned my model and I want to compare results with other models and/or observations.*"
OR
"I need to do research on different models"
OR
"I need specific models output on my custom grid"
OR…

Search
(CMIP5, temperature, 1850-2005,…)

download

On client side calculations

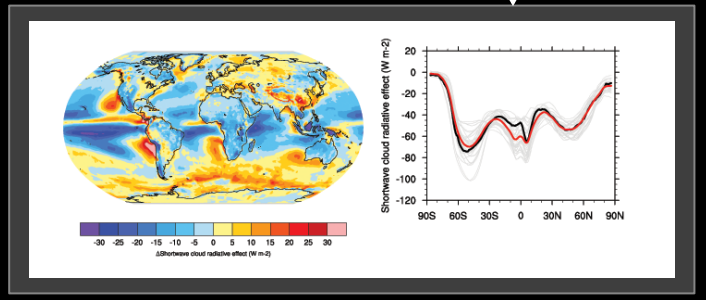


IPSL ESGF node:

esgf-node.ipsl.upmc.fr
vesg.ipsl.upmc.fr
vesg.ipsl.polytechnique.fr
esgf.extra.cea.fr

aims3.llnl.gov
cordexesg.dmi.dk
esg-dn1.nsc.liu.se
esg.cnrm-game-meteo.fr
esgf-data.jpl.nasa.gov
esgf-data1.ceda.ac.uk
…

- Earth System Grid Federation (ESGF) provides access to climate data for the international climate community. ESGF is a system of distributed and federated nodes that dynamically interact with each other.
- One could search and download data geographically distributed over the world through standardized API.



WPS at IPSL

synda

IPSL **LOCAL** ESGF node:

esgf-local.ipsl.upmc.fr

/prodigfs/project ~450 Tb

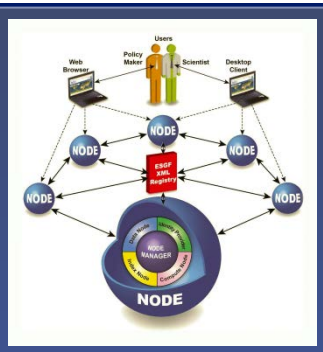
./obs4MIP/

./CORDEX/

…

./CMIP5/

../BCC: bcc-csm1-1, bcc-csm1-1-m
BNU: BNU-ESM
CCCma:CanAM4,CanCM4,CanESM2
CMCC:CMCC-CESM,CMCC-CM,CMCC-CMS
…
NIMR-KMA:HadGEM2-AO
NOAA-GFDL:GFDL-CM2p1,GFDL-CM3,GFDL-ESM2G
GFDL-ESM2M,GFDL-HIRAM-C180



IPSL ESGF node:

esgf-node.ipsl.upmc.fr

vesg.ipsl.upmc.fr
vesg.ipsl.polytechnique.fr
esgf.extra.cea.fr

aims3.llnl.gov
cordexesg.dmi.dk
esg-dn1.nsc.liu.se
esg.cnrm-game-meteo.fr
esgf-data.jpl.nasa.gov
esgf-data1.ceda.ac.uk
…

Web Application

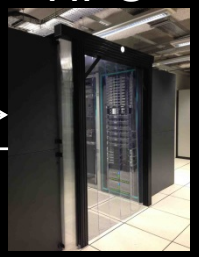
wps-test.ipsl.jussieu.fr

WPS

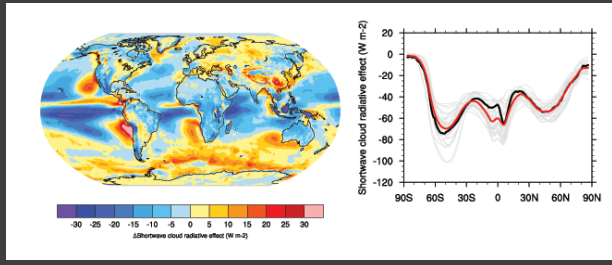
Processing

search

HPC



Results



WPS cache
Other (not IPSL esgf-local) indexed files

Birdhouse Web Processing Service (WPS)

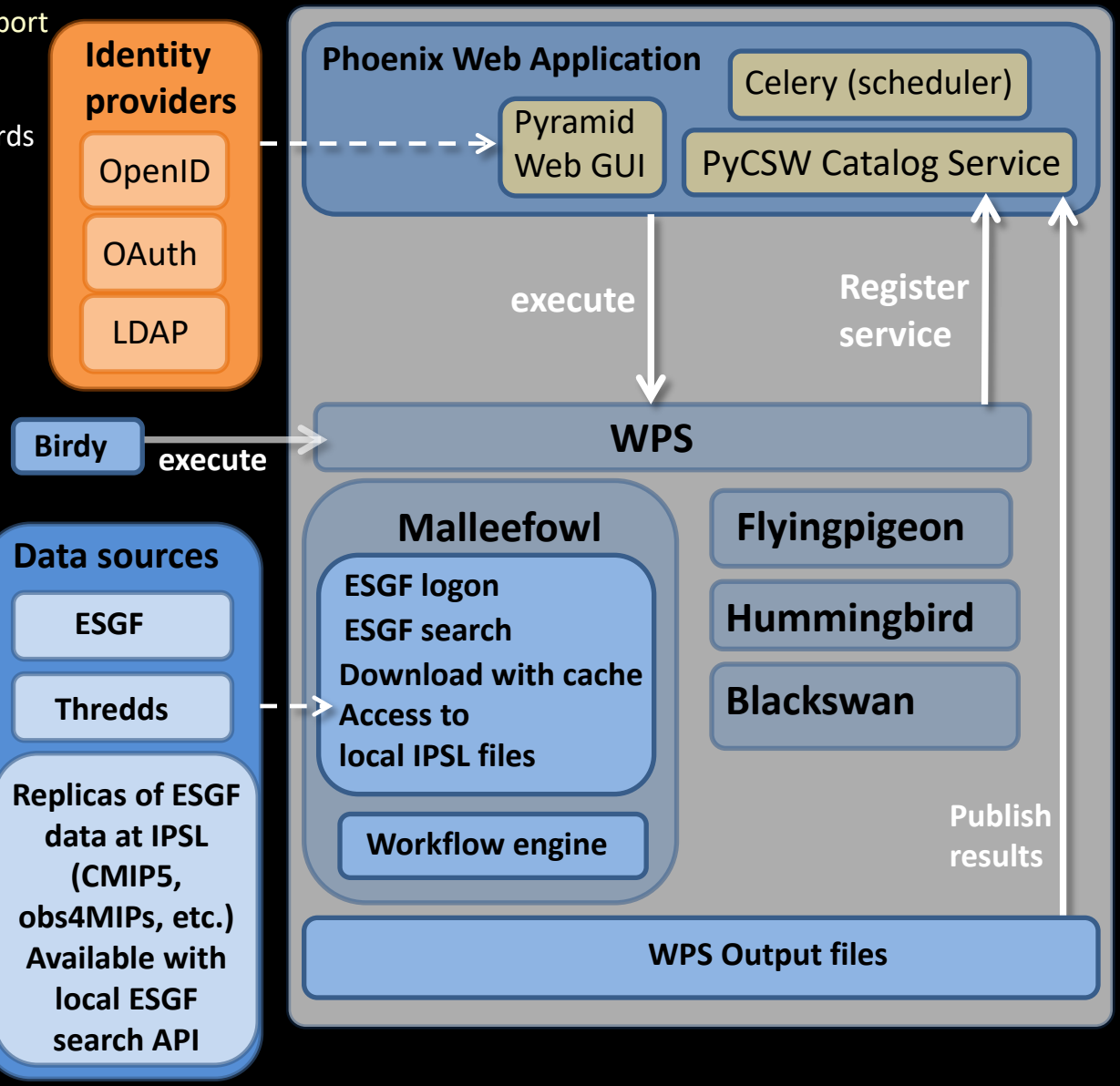
Birdhouse is the home of Web Processing Services used in climate science and components to support them (the birds)

- Based on open source
- Open GeoSpatial Consortium (OGC) Standards
- Easy to implement

Main components of birdhouse are:

- **Phoenix**: Web-based WPS client.
- **Malleefowl** - backend service. Data access processes, workflow engine…
- **Flyingpigeon**: processes for climate data, indices, satellite imaginary.
- **Hummingbird**: data processing based on CDO, core functions (ensemble mean, interpolation, regriding etc.)
- **Birdy**: Command line tool. May be installed on scientist desktop and run WPS processes at IPSL remotely.
- **Blackswan**: service with the processes focusing on extreme weather event assessments.

```
Supervisor status ...
celery           RUNNING   pid 9193, uptime 1 day, 22:08:42
emu              RUNNING   pid 9187, uptime 1 day, 22:08:42
esmvalwps        RUNNING   pid 9192, uptime 1 day, 22:08:42
flyingpigeon     RUNNING   pid 9188, uptime 1 day, 22:08:42
hummingbird      RUNNING   pid 9204, uptime 1 day, 22:08:41
malleefowl       RUNNING   pid 9201, uptime 1 day, 22:08:41
mongodb          RUNNING   pid 9186, uptime 1 day, 22:08:42
nginx            RUNNING   pid 9197, uptime 1 day, 22:08:41
phoenix          RUNNING   pid 9190, uptime 1 day, 22:08:42
pycsw            RUNNING   pid 9199, uptime 1 day, 22:08:41
redis            RUNNING   pid 9191, uptime 1 day, 22:08:42
solr             RUNNING   pid 9203, uptime 1 day, 22:08:41
tomcat           RUNNING   pid 9189, uptime 1 day, 22:08:42
```



How PyWPS works. Example process code

wps_cdo_sinfo.py

PyWPS 4

```python
"""
Processes with cdo commands
"""
from cdo import Cdo
cdo_version = Cdo().version()

from pywps import Process
from pywps import LiteralInput
from pywps import ComplexInput, ComplexOutput
from pywps import Format, FORMATS
from pywps.app.Common import Metadata

import logging
LOGGER = logging.getLogger("PYWPS")


class CDOInfo(Process):
    """This process calls cdo sinfo on netcdf file"""
    def __init__(self):
        inputs = [
            ComplexInput('dataset', 'NetCDF File',
                         abstract='You may provide a URL or upload a NetCDF file.',
                         metadata=[Metadata('Info')],
                         min_occurs=1,
                         max_occurs=100,
                         supported_formats=[Format('application/x-netcdf')]),
        ]
        outputs = [
            ComplexOutput('output', 'CDO sinfo result',
                          abstract='CDO sinfo result document.',
                          as_reference=True,
                          supported_formats=[Format('text/plain')]),
        ]

        super(CDOInfo, self).__init__(
            self._handler,
            identifier="cdo_sinfo",
            title="CDO sinfo",
            abstract="Apply CDO sinfo on NetCDF file and return document with metadata information.",
            version=cdo_version,
            metadata=[
                Metadata('Birdhouse', 'http://bird-house.github.io/'),
                Metadata('User Guide', 'http://birdhouse-hummingbird.readthedocs.io/en/latest/'),
                Metadata('CDO Homepage', 'https://code.zmaw.de/projects/cdo'),
                Metadata('CDO Documentation', 'https://code.zmaw.de/projects/cdo/embedded/index.html'),
            ],
            inputs=inputs,
            outputs=outputs,
            status_supported=True,
            store_supported=True,
        )
```

```python
def _handler(self, request, response):
    datasets = [dataset.file for dataset in request.inputs['dataset']]

    cdo = Cdo()

    outfile = 'out.txt'
    with open(outfile, 'w') as fp:
        response.outputs['output'].file = outfile
        for ds in datasets:
            sinfo = cdo.sinfo(input=[ds], output=outfile)
            for line in sinfo:
                fp.write(line + '\n')
            fp.write('\n\n')
    response.update_status("cdo sinfo done", 100)
    return response
```

```python
from .wps_ncdump import NCDump
from .wps_spotchecker import SpotChecker
from .wps_ioos import IOOSCChecker
from .wps_cdo_op import CDOOperation
from .wps_cdo_sinfo import CDOInfo
from .wps_cdo_bbox import CDOBBox
from .wps_ensembles import Ensembles
from .wps_cfchecker import CFChecker
from .wps_hdh_cfchecker import HDHCFChecker
from .wps_hdh_qachecker import QualityChecker

processes = [
    NCDump(),
    SpotChecker(),
    IOOSCChecker(),
    CFChecker(),
    HDHCFChecker(),
    QualityChecker(),
    CDOOperation(),
    CDOInfo(),
    CDOBBox(),
    Ensembles(),
]
```

__init__.py

Adding your R script in PyWPS

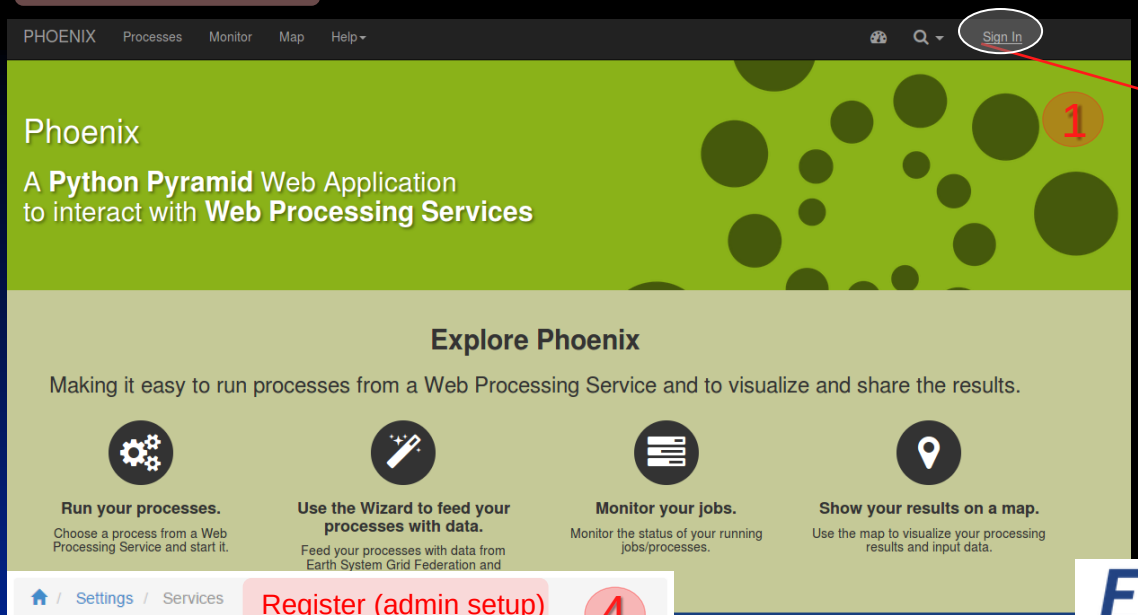
```
class WeatherregimesmodelProcess(Process):
    def __init__(self):
        inputs = [
            ComplexInput('resource', 'Resource',
                         abstract='NetCDF Files or archive
                         metadata=[Metadata('Info')],
                         min_occurs=1,
                         max_occurs=1000,
                         supported_formats=[
                             Format('application/x-netcdf'),
                             Format('application/x-tar'),
                             Format('application/zip'),
                         ]),

            LiteralInput("season", "Time region",
                         abstract="Select the months to def
                         default="DJF",
                         data_type='string',
                         min_occurs=1,
                         max_occurs=1,
                         allowed_values=_TIMEREGIONS_.keys(
                         ),

            LiteralInput('BBox', 'Bounding Box',
                         data_type='string',
                         abstract="Enter a bbox: min_lon, m
                            " min_lon=Western longitude,"
                            " max_lon=Eastern longitude,"
                            " min_lat=Southern or northern
                            " max_lat=Northern or southern
                            " For example: -80,50,20,70",
                         min_occurs=1,
                         max_occurs=1,
                         default='-80,50,20,70',
                         ),

            LiteralInput("period", "Period for weatherregim
                         abstract="Period for analysing the
                         default="19700101-20051231",
                         data_type='string',
                         min_occurs=1,
                         max_occurs=1,
                         ),

            LiteralInput("anualcycle", "Period for anualcyc
                         abstract="Period for anual cycle c
                         default="19700101-19991231",
                         data_type='string',
                         min_occurs=1,
                         max_occurs=1,
                         ),
```

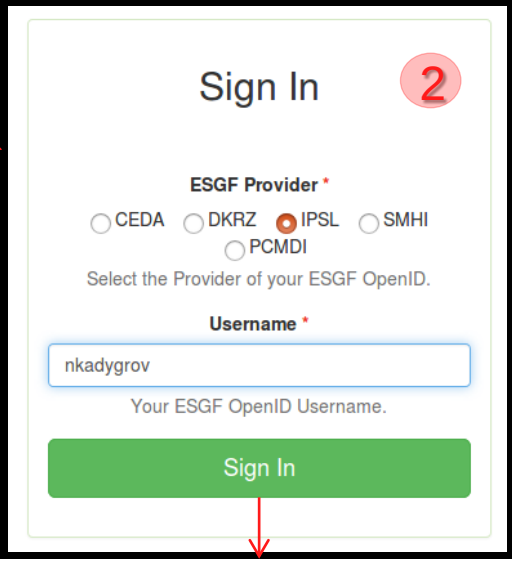
wps_weatherregimes_model.py

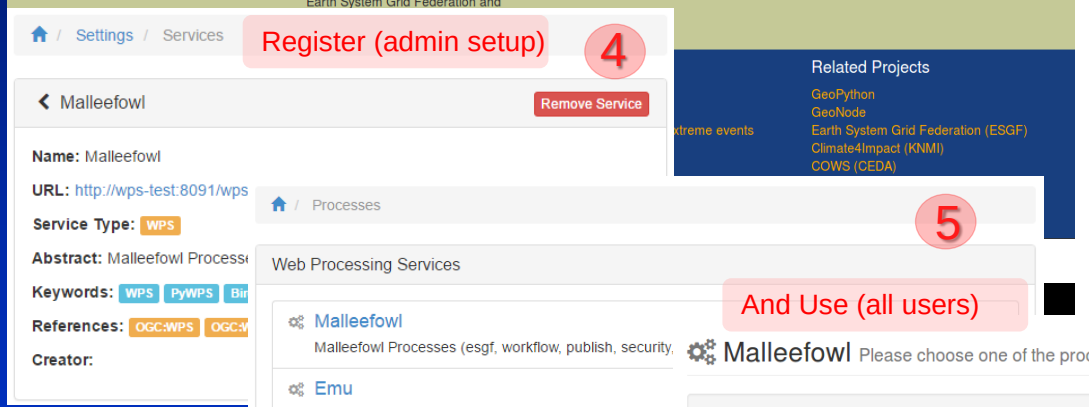
```
####################
# call the R scripts
####################
response.update_status('Start weather regime clustering ', 50)
import shlex
import subprocess
from blackswan import config
from os.path import curdir, exists, join

try:
    rworkspace = curdir
    Rsrc = config.Rsrc_dir()
    Rfile = 'weatherregimes_model.R'

    infile = model_season  # model_subset #model_ponderate
    modelname = 'MODEL'
    yr1 = start.year
    yr2 = end.year
    ip, output_graphics = mkstemp(dir=curdir, suffix='.pdf')
    ip, file_pca = mkstemp(dir=curdir, suffix='.txt')
    ip, file_class = mkstemp(dir=curdir, suffix='.Rdat')

    args = ['Rscript', join(Rsrc, Rfile), '%s/' % curdir,
            '%s/' % Rsrc, '%s' % infile, '%s' % variable,
            '%s' % output_graphics, '%s' % file_pca,
            '%s' % file_class, '%s' % season,
            '%s' % start.year, '%s' % end.year,
            '%s' % 'MODEL', '%s' % kappa]
    LOGGER.info('Rcall builded')
    LOGGER.debug('ARGS: %s'%(args))
except Exception as e:
    msg = 'failed to build the R command %s' % e
    LOGGER.error(msg)
    raise Exception(msg)
try:
    output, error = subprocess.Popen(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE).communicate()
    # ,shell=True
    LOGGER.info('R outlog info:\n %s ' % output)
    LOGGER.debug('R outlog errors:\n %s ' % error)
    if len(output) > 0:
        response.update_status('**** weatherregime in R suceeded', 90)
    else:
        LOGGER.error('NO! output returned from R call')
except Exception as e:
    msg = 'weatherregime in R %s ' % e
    LOGGER.error(msg)
    raise Exception(msg)

response.update_status('Weather regime clustering done ', 92)
############################################
# set the outputs
############################################
response.update_status('Set the process outputs ', 95)
#bla=bla
response.outputs['Routput_graphic'].file = output_graphics
response.outputs['output_pca'].file = file_pca
response.outputs['output_classification'].file = file_class
response.outputs['output_netcdf'].file = model_season
response.update_status('done', 100)
return response
```

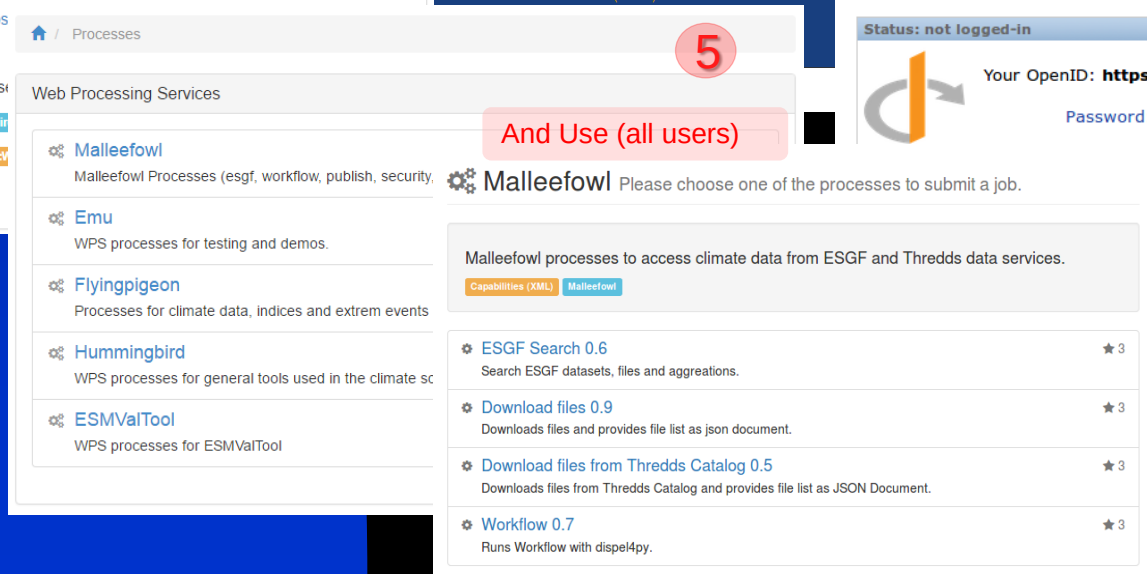
Phoenix web-based WPS client

wps-test.ipsl.jussieu.fr

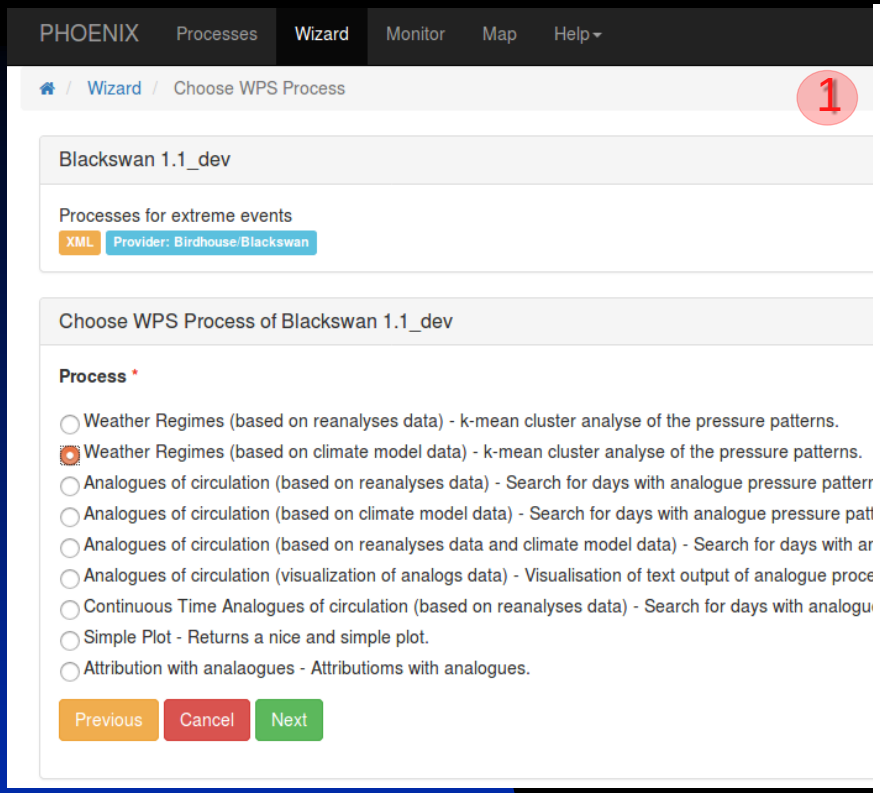


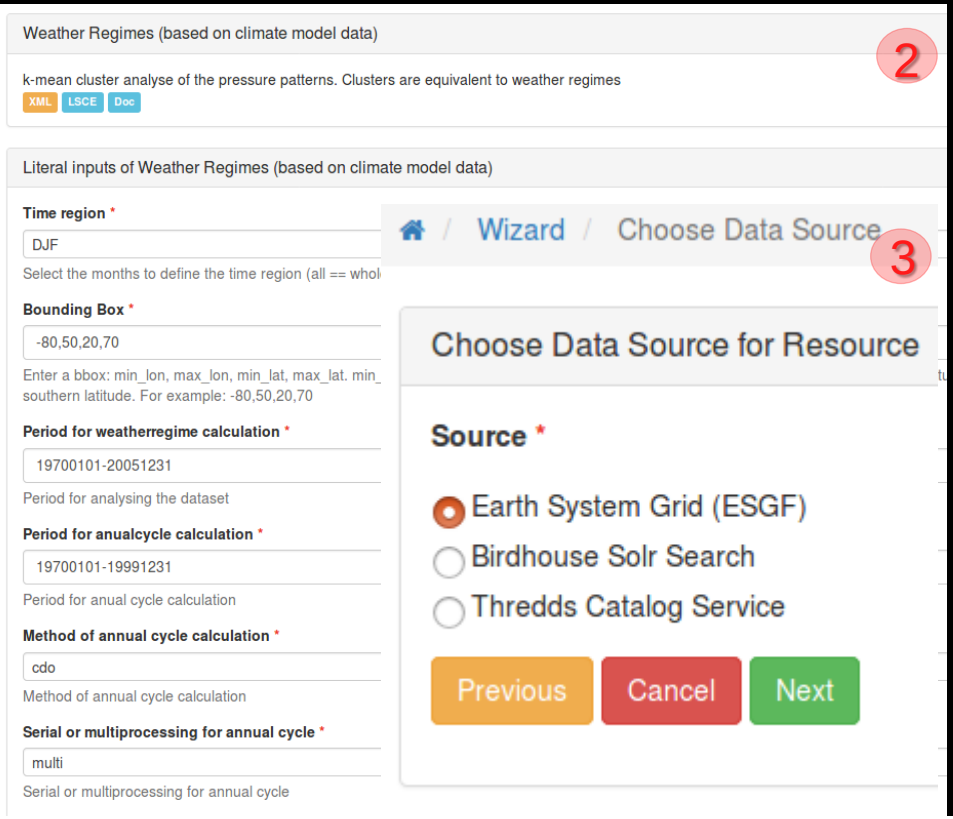


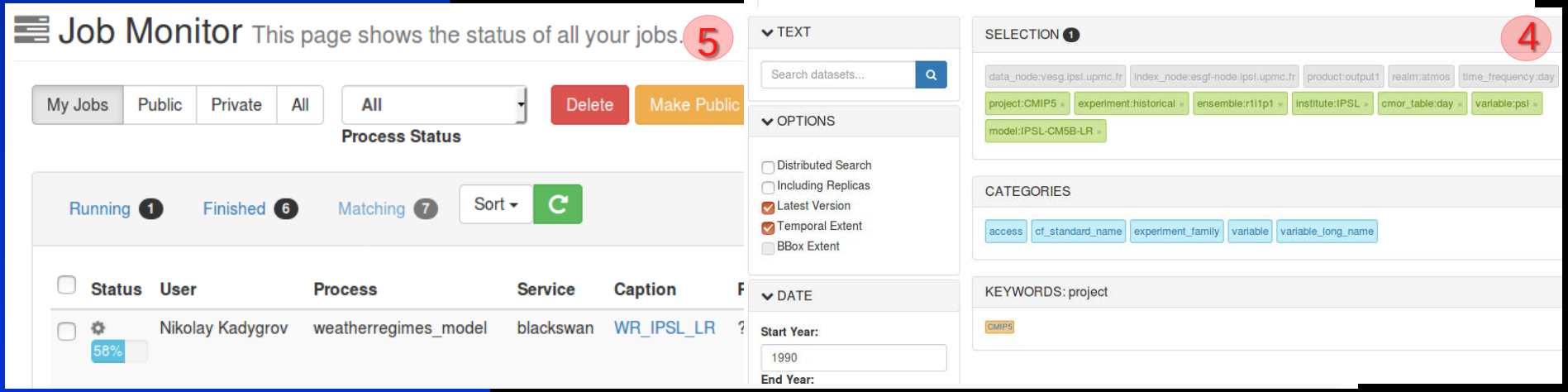




Try Weather regimes process







WPS processes use local archive with ESGF-local search

Log | Inputs | Outputs | </> View as XML

```
 1 0:00:08   0%: PyWPS Process workflow accepted
 2 0:00:10   0%: PyWPS Process workflow accepted
 3 0:00:12  10%: download: status_location=http://wps-test:8090/wpsoutputs/malleefowl/1f486bf6-089a-11e7-8758-00505680073b.xml
 4 0:00:14  10%: download: output=http://wps-test:8090/wpsoutputs/malleefowl/outHmY9Zz.json (application/json)
 5 0:00:16  50%: weatherregimes_model: status_location=http://wps-test:8090/wpsoutputs/flyingpigeon/pywps-212b20ee-089a-11e7-9afd-00505680073
 6 0:00:21  50%: weatherregimes_model: Process weatherregimes_model accepted
 7 0:00:26  50%: weatherregimes_model: Process weatherregimes_model accepted
 8 0:00:31  58%: weatherregimes_model: processstarted start subsetting
 9 0:00:36  58%: weatherregimes_model: processstarted start subsetting
10 0:00:41  58%: weatherregimes_model: processstarted start subsetting
11 0:00:51  58%: weatherregimes_model: processstarted start subsetting
```

wps-test:8090/wpsoutputs/malleefowl/1f486bf6-089a-11e7-8758-00505680073b.xml

```
−<wps:ExecuteResponse xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute_response.xsd" service="WPS" version="1.0.0" xml:lang="en-US" serviceInstance="http://wps-test:8091
/wps?service=WPS&request=GetCapabilities" statusLocation="http://wps-test:8090/wpsoutputs/malleefowl/1f486bf6-089a-11e7-8758-00505680073b.xml">
 −<wps:Process wps:processVersion="0.7">
    <ows:Identifier>download</ows:Identifier>
    <ows:Title>Download files</ows:Title>
   −<ows:Abstract>
      Downloads files and provides file list as json document.
    </ows:Abstract>
  </wps:Process>
 −<wps:Status creationTime="2017-03-14T10:39:30Z">
    <wps:ProcessSucceeded>PyWPS Process Download files finished</wps:ProcessSucceeded>
  </wps:Status>
 −<wps:DataInputs>
   −<wps:Input>
      <ows:Identifier>resource</ows:Identifier>
      <ows:Title>Resource</ows:Title>
      <ows:Abstract>URL of your resource.</ows:Abstract>
     −<wps:Data>
       −<wps:LiteralData dataType="string">
          http://esgf-local.ipsl.upmc.fr/thredds/fileServer/cmip5/output/IPSL/IPSL-CM5B-LR/historical/day/atmos/day/r1i1p1/latest/psl/psl_day_IPSL-CM5B-LR_historical_r1i1p1_18500101-20051231.nc
        </wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
 −<wps:OutputDefinitions>
   −<wps:Output>
      <ows:Identifier>output</
      <ows:Title>Downloaded f
     −<ows:Abstract>
        Json document with list
      </ows:Abstract>
    </wps:Output>
  </wps:OutputDefinitions>
 −<wps:ProcessOutputs>
   −<wps:Output>
      <ows:Identifier>output</ows:Identifier>
      <ows:Title>Downloaded files</ows:Title>
     −<ows:Abstract>
        Json document with list of downloaded files with file url.
      </ows:Abstract>
      <wps:Reference xlink:href="http://wps-test:8090/wpsoutputs/malleefowl/outHmY9Zz.json" mimeType="application/json"/>
    </wps:Output>
  </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

wps-test:8090/wpsoutputs/malleefowl/outHmY9Zz.json

```
[
    "file:///prodigfs/project/CMIP5/output/IPSL/IPSL-CM5B-LR/historical/day/atmos/day/r1i1p1/latest/psl/psl_day_IPSL-CM5B-LR_historical_r1i1p1_18500101-20051231.nc"
]
```

Try Weather regimes process

Details Please note that offline jobs are scheduled and may take a long time to run. This page will continue to poll offline jobs when they are running.

6

weatherregimes_model ✓ Delete Job

No Summary

Status ProcessSucceeded
Duration 0:12:14
Finished 4 minutes ago

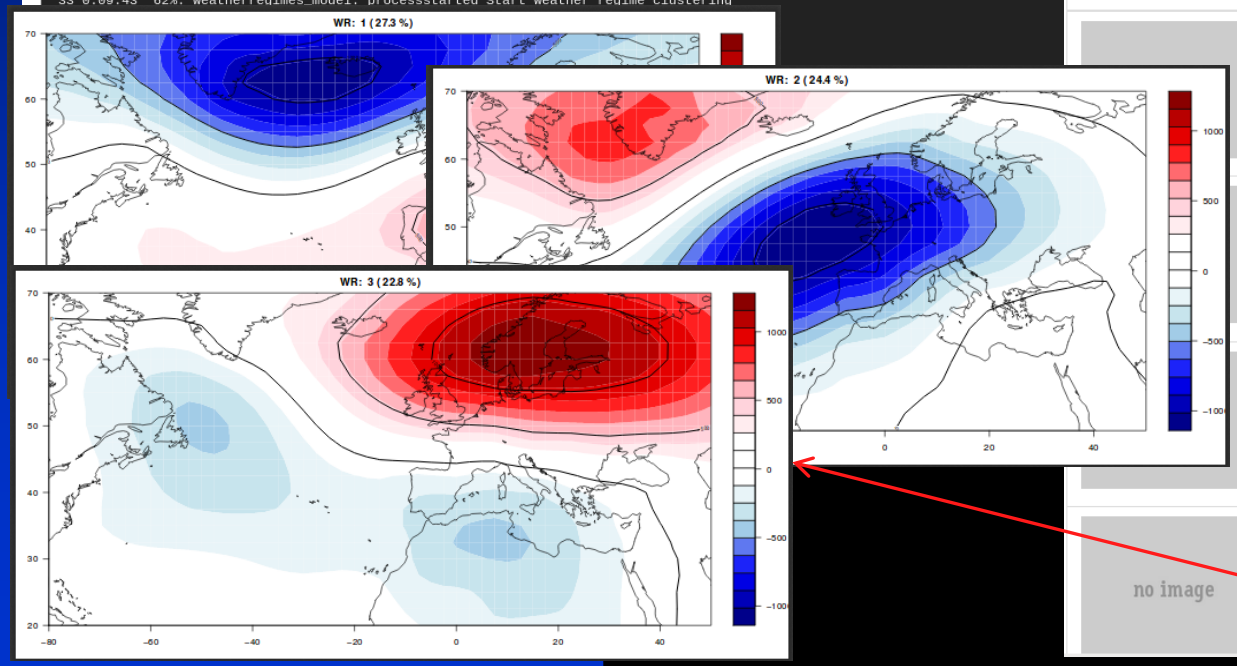
Progress 100%

Labels dev workflow async

Status Message PyWPS Process Workflow finished
Caption wregimes_model

Log | Inputs | Outputs | View as XML

```
23 0:04:41  58%: weatherregimes_model: processstarted start subsetting
24 0:05:12  58%: weatherregimes_model: processstarted start subsetting
25 0:05:42  59%: weatherregimes_model: processstarted computing anomalies
26 0:06:12  62%: weatherregimes_model: processstarted Start weather regime clustering
27 0:06:43  62%: weatherregimes_model: processstarted Start weather regime clustering
28 0:07:13  62%: weatherregimes_model: processstarted Start weather regime clustering
29 0:07:43  62%: weatherregimes_model: processstarted Start weather regime clustering
30 0:08:13  62%: weatherregimes_model: processstarted Start weather regime clustering
31 0:08:43  62%: weatherregimes_model: processstarted Start weather regime clustering
32 0:09:13  62%: weatherregimes_model: processstarted Start weather regime clustering
33 0:09:43  62%: weatherregimes_model: processstarted Start weather regime clustering
```



weatherregimes_model ✓ Delete Job

7

No Summary

Status ProcessSucceeded
Duration 0:12:14
Finished 6 minutes ago

Progress 100%

Labels dev workflow async

Status Message PyWPS Process Workflow finished
Caption wregimes_model

Log | Inputs | Outputs | View as XML

no image

R - datafile Parameter `output_pca`, a WPS ComplexType
Principal components (PCA)
output_pca-212b20ee-089a-11e7-9afd-00505680073b.txt
text/plain
Download | Share

netCDF reference Parameter `output_netcdf`, a WPS ComplexType
Prepared netCDF file as input for weatherregime calculation
output_netcdf-212b20ee-089a-11e7-9afd-00505680073b.nc
application/x-netcdf
Download | Share | Show on Map

Logging information Parameter `output_log`, a WPS ComplexType
Collected logs during process run.
output_log-212b20ee-089a-11e7-9afd-00505680073b.txt
text/plain
Download | Share

R - workspace Parameter `output_classification`, a WPS ComplexType
Weather regime classification
output_classification-212b20ee-089a-11e7-9afd-00505680073b.Rdat
application/octet-stream
Download | Share

no image

Weather Regime Pressure map Parameter `Routput_graphic`, a WPS ComplexType
Weather Classification
Routput_graphic-212b20ee-089a-11e7-9afd-00505680073b.pdf
image/pdf
Download | Share

Analogs with Reanalysis or Model data
Based on Fortran Castf90 program

config.txt

```
!Configuration file for CASTf90 analogs processes deployed in blackswan
!Created : 20180314_102458
!Version : 0.1.5
&FILES
 my_files%archivefile = "base_slp_1948-01-01_2016-12-31_-20.0_40.0_30.0_70.0.nc"
 my_files%simulationfile = "sim_slp_2018-01-01_2018-01-02_-20.0_40.0_30.0_70.0.nc"
 my_files%outputfile = "output.txt"
/
&PARAM
 my_params%timewin = 1
 my_params%varname = "slp"
 my_params%seacyc = .FALSE.
 my_params%cycsmooth = 91
 my_params%nanalog = 20
 my_params%seasonwin = 30
 my_params%distfun = "euclidean"
 my_params%calccor = .TRUE.
 my_params%oformat = ".txt"
 my_params%silent = .FALSE.
/
&ATTS
 my_atts%simsource = "NCEP"
 my_atts%predictorvar = "slp"
 my_atts%archisource = "NCEP"
 my_atts%archiperiod = "1948-01-01,2016-12-31"
 my_atts%predictordom = "-20.0,40.0,30.0,70.0"
/
```

Input for the Fortran program
analogue.out

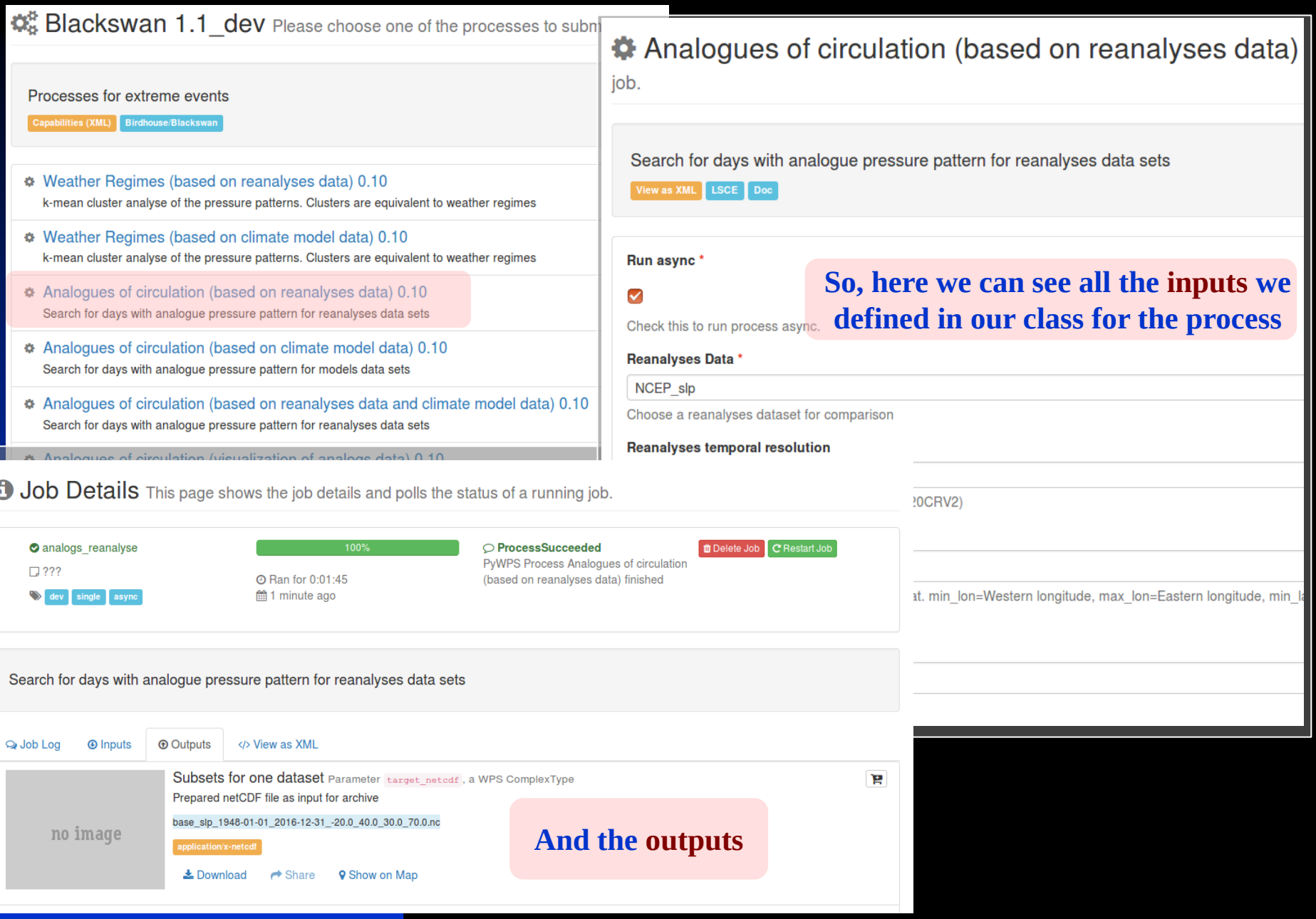
```
class AnalogsreanalyseProcess(Process):
    def __init__(self):
        inputs = [

            LiteralInput("reanalyses", "Reanalyses Data",
                         abstract="Choose a reanalyses dataset for comparison",
                         default="NCEP_slp",
                         data_type='string',
                         min_occurs=1,
                         max_occurs=1,
                         allowed_values=_PRESSUREDATA_
                         ),

            LiteralInput("timeres", "Reanalyses temporal resolution",
                         abstract="Temporal resolution of the reanalyses (only for 20CRV2)",
                         default="day",
                         data_type='string',
                         min_occurs=0,
                         max_occurs=1,
                         allowed_values=['day', '6h']
                         ),

            LiteralInput('BBox', 'Bounding Box',
                         data_type='string',
                         abstract="Enter a bbox: min_lon, max_lon, min_lat, max_lat."
```

```
response.update_status('Start CASTf90 call', 30)
try:
    # response.update_status('execution of CASTf90', 50)
    cmd = ['analogue.out', config_file]
    LOGGER.debug("castf90 command: %s", cmd)
    output = subprocess.check_output(cmd, stderr=subprocess.STDOUT)
    LOGGER.info('analogue output:\n %s', output)
    response.update_status('**** CASTf90 suceeded', 70)
except CalledProcessError as e:
    msg = 'CASTf90 failed:\n{0}'.format(e.output)
    LOGGER.exception(msg)
    raise Exception(msg)
LOGGER.debug("castf90 took %s seconds.", time.time() - start_time)

# TODO: Add try - except for pdfs
if plot == 'Yes':
    analogs_pdf = analogs.plot_analogs(configfile=config_file)
else:
    analogs_pdf = 'dummy_plot.pdf'
    with open(analogs_pdf, 'a'): os.utime(analogs_pdf, None)

response.update_status('preparing output', 75)

response.outputs['analog_pdf'].file = analogs_pdf
response.outputs['config'].file = config_file
response.outputs['analogs'].file = output_file
response.outputs['output_netcdf'].file = simulation
response.outputs['target_netcdf'].file = archive
```

```
############################
# generate the config file
############################
config_file = analogs.get_configfile(
    files=files,
    seasoncyc_base=seasoncyc_base,
    seasoncyc_sim=seasoncyc_sim,
    base_id=model,
    sim_id=model,
    timewin=timewin,
    varname=var,
    seacyc=seacyc,
    cycsmooth=91,
    nanalog=nanalog,
    seasonwin=seasonwin,
    distfun=distance,
    outformat=outformat,
    calccor=True,
    silent=False,
    period=[dt.strftime(refSt, '%Y-%m-%d'), dt.strftime(refEn, '%Y-%m-%d')],
    bbox="{0[0]},{0[2]},{0[1]},{0[3]}".format(bbox))
response.update_status('generated config file', 25)
############################
# CASTf90 call
############################
```

Prepare the config file in the process

Examples of different available WPS processes, Analogs in NCEP



Examples of different available WPS processes, Analogs in NCEP

no image

Analogues Viewer html page Parameter `output`, a WPS ComplexType

Interactive visualization of calculated analogues

analogviewer.html

text/html

Download Share

no image

Formated Analogues File Parameter `formated_analogs`, a WPS ComplexType

Formated analogues file for viewer

modified-analogfile.tsv

text/plain

Download Share

no image

Config File Parameter `config`, a WPS ComplexType

Config file used for the Fortran process

config.txt

text/plain

Download Share

no image

Base Seasonal cycle Parameter `base_netcdf`, a WPS ComplexType

Base seasonal cycle netCDF

dummy_base.nc

application/x-netcdf

Download Share Show on Map

no image

Analogues File Parameter `analogs`, a WPS ComplexType

mulit-column text file

output.txt

text/plain

Download Share

no image

Maps with mean analogs and simu

Analogs Maps

tmp7Siuof.pdf

image/pdf

Download Share

Analogues Viewer Help Contact

Calculated analogues file: "output.txt"
Number of analogues: 20
Var name: "slp"
Simulation source: "NCEP"
Archive source: "NCEP"
bbox: "-20.0,40.0,30.0,70.0"
Reference period: 01/01/1948 - 31/12/2016
Full config file: click here

All records selected. Please click on the graph to apply filters.

Analysis period (manual selection, format dd/mm/yyyy)

Start Date (min: 01/01/2018) 01/01/2018 End Date (max: 10/01/2018) 10/01/2018

Analysis period (chart selection)

reset

20 18 16 14 12 10 8 6 4 2 0

Tue 02 Wed 03 Thu 04 Fri 05 Sat 06 Jan 07 Mon 08 Tue 09 Wed 10

Season

DJF

Archive decade

1948-1955
1956-1965
1966-1975
1976-1985
1986-1995
1996-2005
2006-2015
2016

0 10 20 30

Correlation

Count

70 60 50 40 30 20 10 0.0

0.0 0.2 0.4 0.6 0.8 1.0

Correlation

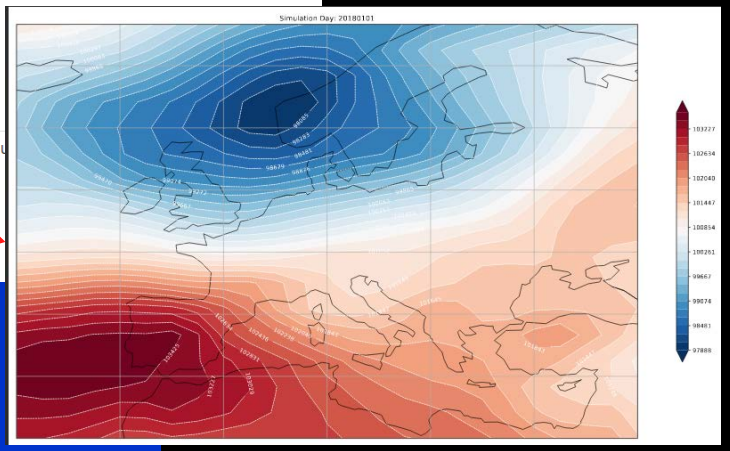
Distance

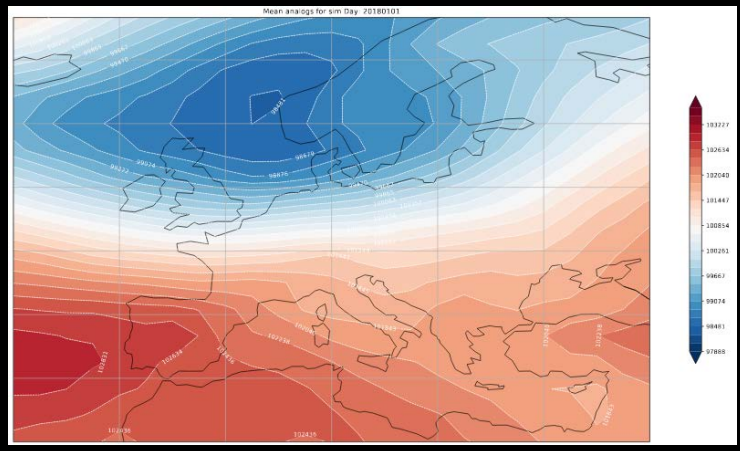
Count

90 80 70 60 50 40 30 20 10 0.0

400 500 600 700 800

Distance





WPS services could be accessed not only through web-application, Phoenix, but also from scripts or birdy cmd-client

```
#Here could be any process from registered WPS - 8091 corresponds to Malleefowl, and esgsearch is one of it's processes

$ export WPS_SERVICE=http://wps-test.ipsl.jussieu.fr:8091/wps

$ birdy –h
gives you all available processes for $WPS_SERVICE
```

```
birdy esgsearch --url https://esgf-node.ipsl.upmc.fr/esg-search  --distrib False --replica False --temporal True --offset 0 --search_type File \
 --constraints project:CMIP5,model:IPSL-CM5A-LR,variable:tas,experiment:historical , ensemble:r1i1p1,time_frequency:mon --start 1900-01-11T12:00:00Z \
--end 2010-12-31T12:00:00Z
```

```
INFO:Execution status: ProcessAccepted

INFO:Execution status: ProcessSucceeded

INFO:Output:INFO:facet_counts=http://wps-test:8090/wpsoutputs/malleefowl/facet_counts-12edb5ea-03dd-11e6-bbcc-3be3bbcfb726.json (application/json)

INFO:output=http://wps-test:8090/wpsoutputs/malleefowl/output-12edb5ea-03dd-11e6-bbcc-3be3bbcfb726.json (application/json)

INFO:summary=http://wps-test:8090/wpsoutputs/malleefowl/summary-12edb5ea-03dd-11e6-bbcc-3be3bbcfb726.json (application/json)
```

```
[
  "http://esgf.extra.cea.fr/thredds/fileServer/work_cmip5/output1/IPSL/IPSL-CM5A-LR/historical/mon/atmos/Amon/r1i1p1/v20110406/tas/tas_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc“
]
```

WPS services could be accessed not only through web-application, Phoenix, but also from scripts or birdy cmd-client

Script language

```
from owslib.wps import WebProcessingService, monitorExecution

wps = WebProcessingService(url="http://wps-test.ipsl.jussieu.fr:8091/wps", verbose=False, skip_caps=False)

#Here could be any process as well!
execute = wps.execute(identifier="esgsearch", inputs=[("url","https://esgf-node.ipsl.upmc.fr/esg-search"),
    ("constraints","project:CMIP5,model:IPSL-CM5A-LR,
      experiment:historical,ensemble:r1i1p1,time_frequency:mon"),
    ("search_type","File")],output=[("output",True)])

for o in execute.processOutputs:
    print o.reference
```

http://wps-test.ipsl.jussieu.fr:8090/wpsoutputs/malleefowl/output-ed807284-03dc-11e6-bbcc-3be3bbcfb726.json


```
[    "http://esgf.extra.cea.fr/thredds/fileServer/work_cmip5/output1/IPSL/IPSL-CM5A-LR/historical/mon
     /atmos/Amon/r1i1p1/v20110406/rlds/rlds_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc",
     "http://esgf.extra.cea.fr/thredds/fileServer/work_cmip5/output1/IPSL/IPSL-CM5A-LR/historical/mon/atmos
     /Amon/r1i1p1/v20110406/ccb/ccb_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512.nc",  ... ]
```

Also the log file with debugging info for every process is available

wps@wps-test:/tmp

Fichier Édition Affichage Rechercher Terminal Onglets Aide

wps@wps-test:~/ESMValTool | wps@wps-test:/tmp | wps@wps-test:~

```
PyWPS [2017-03-14 10:39:39,071] DEBUG: DLL: <CDLL 'libc.so.6', handle 7fa9ddcc04c8 at 7fa9c5c1ded0>
PyWPS [2017-03-14 10:39:40,250] INFO: Following processes are imported: ['subset_continents', 'subset_countries', 'subset_regionseurope', 'subset_points', 'landseamask', 'indice
herregimes_reanalyse', 'weatherregimes_model', 'weatherregimes_projection', 'analogs_detection', 'analogs_model', 'analogs_compare', 'analogs_viewer', 'segetalflora', 'sdm_gbiff
es', 'sdm_csv', 'sdm_allinone', 'robustness', 'plot_timeseries', 'climatefactsheet', 'fetch', 'wps_c4i_simple_indice', 'cdo_inter']
PyWPS [2017-03-14 10:39:40,253] INFO: Status [processpaused]: Getting input resource of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,253] DEBUG: Missing ComplexDataInput mimeType in: resource, adopting default mimeType (first in formats list)
PyWPS [2017-03-14 10:39:40,253] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,253] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,255] INFO: Status [processpaused]: Getting input BBox of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,258] INFO: Status [processpaused]: Getting input season of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,262] INFO: Status [processpaused]: Getting input period of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,266] INFO: Status [processpaused]: Getting input anualcycle of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,268] INFO: Status [processpaused]: Getting input kappa of process weatherregimes_model
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Missing ComplexDataOutput mimeType in output_pca, adopting default mimeType text/plain (first in formats list)
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Missing ComplexDataOutput mimeType in Routput_graphic, adopting default mimeType image/pdf (first in formats list)
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Missing ComplexDataOutput mimeType in output_classification, adopting default mimeType application/octet-stream (first in formats list)
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Missing ComplexDataOutput mimeType in output_netcdf, adopting default mimeType application/x-netcdf (first in formats list)
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding schema: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Adding encoding: None
PyWPS [2017-03-14 10:39:40,269] DEBUG: Missing ComplexDataOutput mimeType in output_log, adopting default mimeType text/plain (first in formats list)
PyWPS [2017-03-14 10:39:40,272] INFO: Status [processstarted][0.0]: Process weatherregimes_model started
PyWPS [2017-03-14 10:39:40,273] INFO: Start process
PyWPS [2017-03-14 10:39:40,276] INFO: Status [processstarted][5.0]: processstarted execution started at : 2017-03-14 10:39:40.273260
PyWPS [2017-03-14 10:39:40,276] INFO: read in the arguments
PyWPS [2017-03-14 10:39:40,294] DEBUG: failed to read in the arguments local variable 'bbox' referenced before assignment
PyWPS [2017-03-14 10:39:40,294] INFO: bbox_obj=([-180.0, -90.0], [180.0, 90.0])
PyWPS [2017-03-14 10:39:40,294] INFO: bbox=[-180.0, -90.0, 180.0, 90.0]
PyWPS [2017-03-14 10:39:40,295] INFO: Status [processstarted][17.0]: processstarted start subsetting
PyWPS [2017-03-14 10:39:40,410] INFO: Start ocgis module call function
PyWPS [2017-03-14 10:39:40,422] DEBUG: spatial_reorder: True and spatial_wrapping: wrap
PyWPS [2017-03-14 10:39:40,422] INFO: Execute ocgis module call function
PyWPS [2017-03-14 10:39:40,422] DEBUG: call module curdir = /home/wps/birdhouse/var/lib/pywps/tmp/flyingpigeon/pywps-instanceWbdDM7
PyWPS [2017-03-14 10:39:41,265] INFO: OcgOperations set
PyWPS [2017-03-14 10:43:34,401] INFO: data_mb  = 198.987037659 ; memory_limit = 478.306640625
PyWPS [2017-03-14 10:43:34,402] INFO: ocgis module call as ops.execute()
(END)
```

Some more examples, Remapping core function

- Robustness - Calculates the robustness as the ratio of noise to signal in an ensemle of tim
- Plots -- timeseries - Outputs some timeseries of the file field means. Spaghetti and uncertai
- Climate Fact Sheet Generator - Returns a pdf with a short overview of the climatological sit
- Download Resources - This process downloads resources (limited to 50GB) to the local fil
- c4i -- Simple Climate Indices - Computes single input indices of temperature TG, TX, TN, T RR1, SDII, R10mm, R20mm, RX1day, RX5day; and of snowfall: SD, SD1, SD5, SD50. This
- (•) CDO Remapping - CDO Remapping of NetCDF File(s).
- Atmospheric Modes of Variability (NAO) - North Atlantic Oscillation (NAO)
- QBO from selected models - Plotting QBO

Previous | Cancel | Next

Literal inputs of CDO Remapping

CDO Operator

remapbil

Choose a CDO Operator

Grid

custom

Select an grid

longitude

144

New nx Longitude)

Latitude

73

New ny Latitude)

Previous | Next | Cancel

tarout-7

Fichier Édition Affichage Aide

Nouveau Ouvrir Extraire Ajoutez des fichiers Ajouter un dossier Arrêter

Précédent Emplacement : /

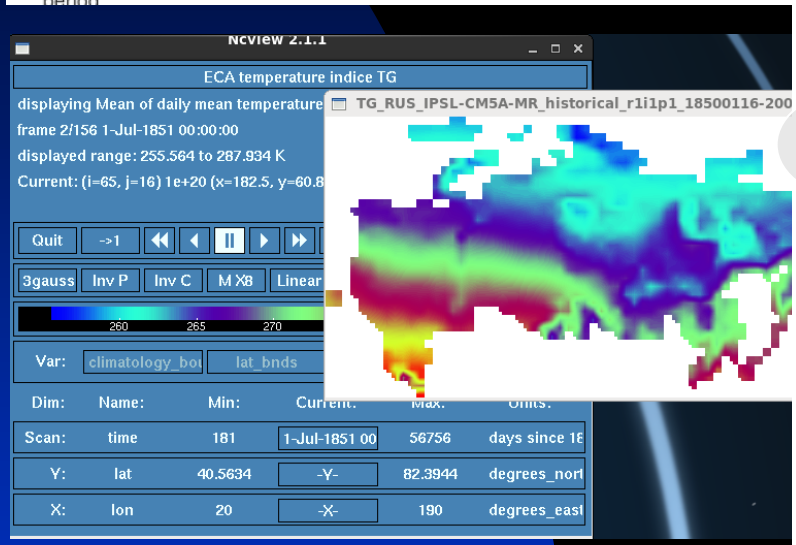
Nom	Taille	
.	225,3 Mio	Dossier
tas_Amon_IPSL-CM5A-LR_historical_r1i1p1_185001-200512_remapbil_r144x73.nc	75,1 Mio	document U... 16 avril 2016, 16:49
tas_Amon_IPSL-CM5A-MR_historical_r1i1p1_185001-200512_remapbil_r144x73.nc	75,1 Mio	document U... 16 avril 2016, 16:49
tas_Amon_IPSL-CM5B-LR_historical_r1i1p1_185001-200512_remapbil_r144x73.nc	75,1 Mio	document U... 16 avril 2016, 16:50

Some more examples, Climate indices.

Process 1

- ○ Visualisation of netcdf files - Just testing a nice script to visualise some variables
- ○ Species distribution model - Species distribution model (SDM)
- ○ Weather Regimes - Weather Regimes based on pressure patterns (kmean method)
- ○ Extract Coordinate Points - Extract Timeseries for specified coordinates from grid da
- ○ Segetal Flora - Species biodiversity of segetal flora. Imput files: variable:tas , domai
- ◉ Calculation of climate indice (single variable) - This process calculates climate indice
- ○ Calculation of percentile based climate indices (single variable) - This process calcu

period



Literal inputs of Calculation of climate indice (single variable) 2

Grouping

yr

Select an time grouping (time aggregation)

Indice

TG

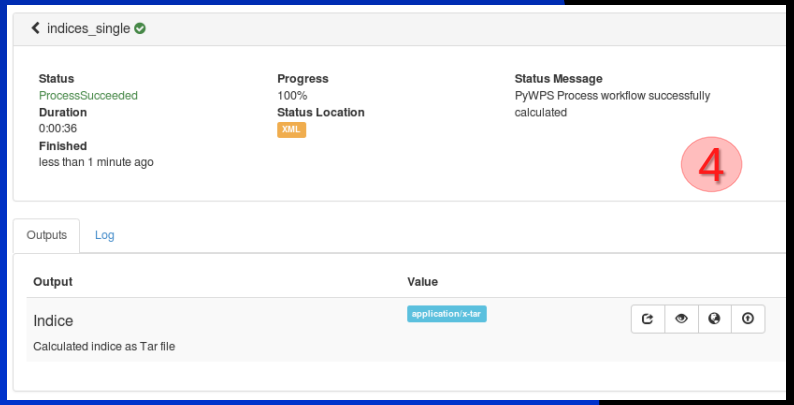
CDD: Consecutive dry days (pr as input files) CFD: Nr of consecutive frost days (tasmin as input files) CSU: Nr of consecutive summer days (tasmax as input files) CWD: Consecutive wet days (pr as input files) FD: Nr of frost days (tasmin as input files) GD4: Growing degree days [sum of TG >= 4 degrees] (tas as input files) HD17: Heating degree days [sum of 17 degrees - mean temperature] (tas as input files) ID: Nr of Ice days (tasmax as input files) PRCPTOT: Precipitation flux mean (mon / year) (pr as input files) R10mm: Nr of days >10mm (pr as input files) R20mm: Nr of days with precipitation >= 20 mm (pr as input files) RR1: Nr of days with precipitation > 1 mm (pr as input files) RX1day: Highest 1-day precipitation amount (pr as input files) RX5day: Highest 5-day precipitation amount (pr as input files) SD: Nr of snow days (prsn as input files) SD1: Nr of days with snow >= 1cm (prsn as input files) SD50cm: Nr of days with snow >= 50 cm (prsn as input files) SD5cm: Nr of days with snow >= 5cm (prsn as input files) SDII: Simple daily intensity index for wet days [mm/wet day] (pr as input files) SU: Nr of summer days (tasmax as input files) TG: Mean of mean temperatur (tas as input files) TN: Mean of daily min temperatur (tasmin as input files) TNn: Min of daily min temperatur (tasmin as input files) TNx: Max of daily min temperatur (tasmin as input files) TR: Tropical nights - number of days where daily minimum temperature >= 20 degrees.(tasmin as input files) TX: Mean of max temperatur (tasmax as input files) TXn: Min of daily min temperatur (tasmax as input files) TXx: Max of daily max temperatur (tasmax as input files)

Country subset

RUS

ABW : Aruba AFG : Afghanistan AGO : Angola ALB : Albania ALD : Aland Islands AND : Andorra ARE : United Arab Emirates ARG : Argentina ARM : Armenia ASM : American Samoa ATA : Antarctica ATF : French Southern and Antarctic Lands ATG : Antigua and Barbuda AUS : Australia AUT : Austria AZE : Azerbaijan BDI : Burundi BEL : Belgium BEN : Benin BFA : Burkina Faso BGD : Bangladesh BGR : Bulgaria BHR : Bahrain BHS : Bahamas BIH : Bosnia and Herzegovina BLR : Belarus BLZ : Belize BOL : Bolivia BRA : Brazil BRB : Barbados BRN : Brunei Darussalam BTN : Bhutan BWA : Botswana CAF : Central African Republic CAN : Canada CHE : Switzerland CHL : Chile CHN : China CIV : CÃ´te d'Ivoire CMR : Cameroon CNM : Cyprus U.N. Buffer Zone COD : Democratic Republic of the Congo COG : Republic of Congo COL : Colombia COM : Comoros CPV : Cape Verde CRI : Costa Rica CUB : Cuba CUW : CuraÃ§ao CYM : Cayman Islands CYN : Northern Cyprus CYP : Cyprus CZE : Czech Republic DEU : Germany DJI : Djibouti DMA : Dominica DNK : Denmark DOM : Dominican Republic DZA : Algeria ECU : Ecuador EGY : Egypt ERI : Eritrea ESB : Dhekelia ESP : Spain EST : Estonia ETH : Ethiopia FIN : Finland FJI : Fiji FLK : Falkland Islands FRA : France FRO : Faeroe Islands FSM : Federated States of Micronesia GAB : Gabon GBR : United Kingdom GEO : Georgia GHA : Ghana GIN : Guinea GMB : The Gambia GNB : Guinea-Bissau GNQ : Equatorial Guinea GRC : Greece GRD : Grenada GRL : Greenland GTM : Guatemala GUM : Guam GUY : Guyana HKG : Hong Kong HMD : Heard I. and McDonald Islands HND : Honduras HRV : Croatia HTI : Haiti HUN : Hungary IDN : Indonesia IMN : Isle of Man IND : India IRL : Ireland IRN : Iran IRQ : Iraq ISL : Iceland ISR : Israel ITA : Italy JAM : Jamaica JOR : Jordan JPN : Japan KAB : Baikonur Cosmodrome KAS : Siachen Glacier KAZ : Kazakhstan KEN : Kenya KGZ : Kyrgyzstan KHM : Cambodia KIR : Kiribati KNA : Saint Kitts and Nevis KOR : Republic of Korea KOS : Kosovo KWT : Kuwait LAO : Lao PDR LBN : Lebanon LBR : Liberia LBY : Libya LCA : Saint Lucia LIE : Liechtenstein LKA : Sri Lanka LSO : Lesotho LTU : Lithuania LUX : Luxembourg LVA : Latvia MAR : Morocco MDA : Moldova MDG : Madagascar MEX : Mexico MKD : Macedonia MLI : Mali MLT : Malta MMR : Myanmar MNE : Montenegro MNG : Mongolia MNP : Northern Mariana Islands MOZ : Mozambique MRT : Mauritania MUS : Mauritius MWI : Malawi MYS : Malaysia NAM : Namibia NCL : New Caledonia NER : Niger NGA : Nigeria NIC : Nicaragua NIU : Niue NLD : Netherlands NOR : Norway NPL : Nepal NZL : New Zealand OMN : Oman PAK : Pakistan PAN : Panama PER : Peru PHL : Philippines PLW : Palau PNG : Papua New Guinea POL : Poland PRI : Puerto Rico PRK : Dem. Rep. Korea PRT : Portugal PRY : Paraguay PSX : Palestine PYF : French Polynesia QAT : Qatar ROU : Romania RUS : Russian Federation RWA : Rwanda SAH : Western Sahara SAU : Saudi Arabia SDN : Sudan SDS : South Sudan SEN : Senegal SGP : Singapore SGS : South Georgia and South Sandwich Islands SHN : Saint Helena SLB : Solomon Islands SLE : Sierra Leone SLV : El Salvador SOL : Somaliland SOM : Somalia SPM : Saint Pierre and Miquelon SRB : Serbia STP : SÃ£o TomÃ© and Principe SUR : Suriname SVK : Slovakia SVN : Slovenia SWE : Sweden SWZ : Swaziland SYR : Syria TCD : Chad TGO : Togo THA : Thailand TJK : Tajikistan TKM : Turkmenistan TLS : Timor-Leste TON : Tonga TTO : Trinidad and Tobago TUN : Tunisia TUR : Turkey TWN : Taiwan TZA : Tanzania UGA : Uganda UKR : Ukraine URY : Uruguay USA : United States UZB : Uzbekistan VCT : Saint Vincent and the Grenadines VEN : Venezuela VIR : United States Virgin Islands VNM : Vietnam VUT : Vanuatu WSB : Akrotiri WSM : Samoa YEM : Yemen ZAF : South Africa ZMB : Zambia ZWE : Zimbabwe

Previous | Next | Cancel



Status	Job	Process	Service	Duration	Finished	Progress
⚙	a2aa1780-03d0-11e6-afd0-fdf570512394	indices_single	Flyingpigeon	0:00:21	???	50%
✓	6bd21552-03ce-11e6-afd0-fdf570512394	mydiag	ESMValTool	0:01:17	14 minutes ago	100%

3

Demonstration and TPs

- TP: After the lunch we will check how-to:

- Install WPS service from the sources on gitgub
- Run WPS process as HTTP request
- Install command-line tool to work with WPS
- Run the process from cmd
- Configure and run processes using Phoenix GUI
- Study how to use Jupyter notebooks with Python, and call WPS processes from it.

- Create your first WPS process

- Check how to use WPS call for event attribution using analogues

Thank you!

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

Appendix

Birdy

wps@wps-test:~/birdhouse/pyramid-phoenix

Fichier Édition Affichage Rechercher Terminal Onglets Aide

wps@wps-test:~/birdhouse/pyramid-phoenix

```
(birdhouse)[wps@wps-test pyramid-phoenix]$
(birdhouse)[wps@wps-test pyramid-phoenix]$
(birdhouse)[wps@wps-test pyramid-phoenix]$ export WPS_SERVICE=http://wps-test:8093/wps
(birdhouse)[wps@wps-test pyramid-phoenix]$ 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,weatherregimes,extractpoints,segetalflora,indices_single,indices_perc
                        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
                        (SDM)
    weatherregimes      Weather Regimes: Weather Regimes based on pressure
                        patterns (kmean method)
    extractpoints       Extract Coordinate Points: Extract Timeseries for
                        specified coordinates from grid data
    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.
    indices_percentile  Calculation of percentile based climate indices
                        (single variable): This process calculates climate
                        indices based on one single variable and based on
                        percentils of a referece period.
    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 CORDEX
                        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 days
                        with analog pressure pattern
    fetch               Download Resources: This process downloads resources
                        (limited to 50GB) to the local file system of the
                        birdhouse compute provider
    qbo                 QBO from selected models: Plotting QBO
    nao                 Atmospheric Modes of Variability (NAO): North Atlantic
                        Oscillation (NAO)
    nam                 Atmospheric Modes of Variability (NAM): Northern
                        Annular Mode (NAM)
(birdhouse)[wps@wps-test pyramid-phoenix]$
```

wps@

Fichier Édition Affichage Rechercher Terminal Onglets Aide

wps@wps-test:~/birdhouse/pyramid-phoenix

```
(birdhouse)[wps@wps-test pyramid-phoenix]$ birdy nao -h
usage: birdy nao [-h] --resource RESOURCE [RESOURCE ...] [--styear [STYEAR]]
                 [--enyear [ENYEAR]] [--refdata [{NCEP,ERA-I}]]
                 [--season [{DJF,MAM,JJA,SON,ANN}]]
                 [--output [{plotout_nao_pc_ref,plotout_nao_ref,plotout_nao_patt,plotout_na
lotout_nao_pc,plotout_nao_refW,plotout_nao_pattW} ...]]]

optional arguments:
  -h, --help            show this help message and exit
  --resource RESOURCE [RESOURCE ...]
                        NetCDF Files for psl(Monthly): NetCDF Files for
                        psl(Monthly), mime types=application/x-netcdf
  --styear [STYEAR]     First Year: First Year (default: 1980)
  --enyear [ENYEAR]     Last Year: Last Year (default: 2000)
  --refdata [{NCEP,ERA-I}]
                        Reference reanalysis Dataset: Reference reanalysis
                        Dataset (default: ERA-I)
  --season [{DJF,MAM,JJA,SON,ANN}]
                        Season: Season (default: DJF)
  --output [{plotout_nao_pc_ref,plotout_nao_ref,plotout_nao_patt,plotout_naoeof,plotout_nao
otout_nao_refW,plotout_nao_pattW} ...]]
                        Output: plotout_nao_pc_ref=PC timeseries for Reference
                        Dataset: Visualisation of PC timeseries for Reference
                        Dataset, mime types=application/html (default: all
                        outputs)plotout_nao_ref=NAO pattern from Reference
                        dataset on stereographic map: NAO pattern from
                        Reference dataset on stereographic map, mime
                        types=application/html (default: all
                        outputs)plotout_nao_patt=Model NAO pattern on
                        stereographic map: Visualisation of Model NAO pattern
                        on stereographic map, mime types=application/html
                        (default: all outputs)plotout_naoeof=1st EOF of psl:
                        Visualisation of 1st EOF of psl, mime
                        types=application/html (default: all
                        outputs)plotout_nao_pc=NAO PC timeseries for Model:
                        Visualisation of NAO PC timeseries, mime
                        types=application/html (default: all
                        outputs)plotout_nao_refW=NAO PSL pattern from
                        Reference dataset on global map: NAO PSL pattern from
                        Reference dataset on global map, mime
                        types=application/html (default: all
                        outputs)plotout_nao_pattW=Model NAO pattern on global
                        map: Visualisation of Model NAO pattern on global map,
                        mime types=application/html (default: all outputs)
(birdhouse)[wps@wps-test pyramid-phoenix]$
```

Tutorial: WPS post request

Documentation: http://geoprocessing.info/wpsdoc/1x0ExecutePOST

XML for chosmky execute request ...

In [6]:
```
execute_xml = """
<wps:Execute
    xmlns:wps="http://www.opengis.net/wps/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ows="http://www.opengis.net/ows/1.1"
    service="WPS"
    version="1.0.0"
    xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute_r
equest.xsd">
    <ows:Identifier>wordcount</ows:Identifier>
    <wps:DataInputs>
        <wps:Input>
            <ows:Identifier>text</ows:Identifier>
            <wps:Data>
                <wps:LiteralData>https://gnu.org/licenses/gpl-3.0.txt</wps:LiteralData>
            </wps:Data>
        </wps:Input>
    </wps:DataInputs>
    <wps:ResponseForm>
        <wps:ResponseDocument status="true" storeExecuteResponse="true">
            <wps:Output asReference="true">
                <ows:Identifier>output</ows:Identifier>
            </wps:Output>
        </wps:ResponseDocument>
    </wps:ResponseForm>
</wps:Execute>
"""
```

In [7]:
```
from urllib2 import urlopen, Request
req = Request("http://localhost:8094/wps", execute_xml)
req.add_header('Content-Type', "text/xml")
response = urlopen(req)
```

In [8]:
```
from lxml import etree
tree = etree.parse(response)
#print etree.tostring(tree)
print tree.getroot().get("statusLocation")
```

```
http://localhost:8090/wpsoutputs/emu/pywps-385e44b4-a7d8-11e4-be4c-68f72837e1b4.xml
```

Tutorial: WPS *async execute* request

Template for async execute request

In [1]:
```
req_url = "{wps_url}?" +\
    "request=Execute" +\
    "&service=WPS" +\
    "&version=1.0.0" +\
    "&identifier={identifier}" +\
    "&DataInputs={inputs}" +\
    "&storeExecuteResponse=true" +\
    "&status=true"
```

Execute the *ultimate question* process again ...

In [2]:
```
url=req_url.format(
    wps_url="http://localhost:8094/wps",
    identifier="ultimatequestionprocess",
    inputs="")
print url
```

```
http://localhost:8094/wps?request=Execute&service=WPS&version=1.0.0&identifier=ultimatequestionprocess&DataInputs=&storeExecuteResponse=true&status=true
```

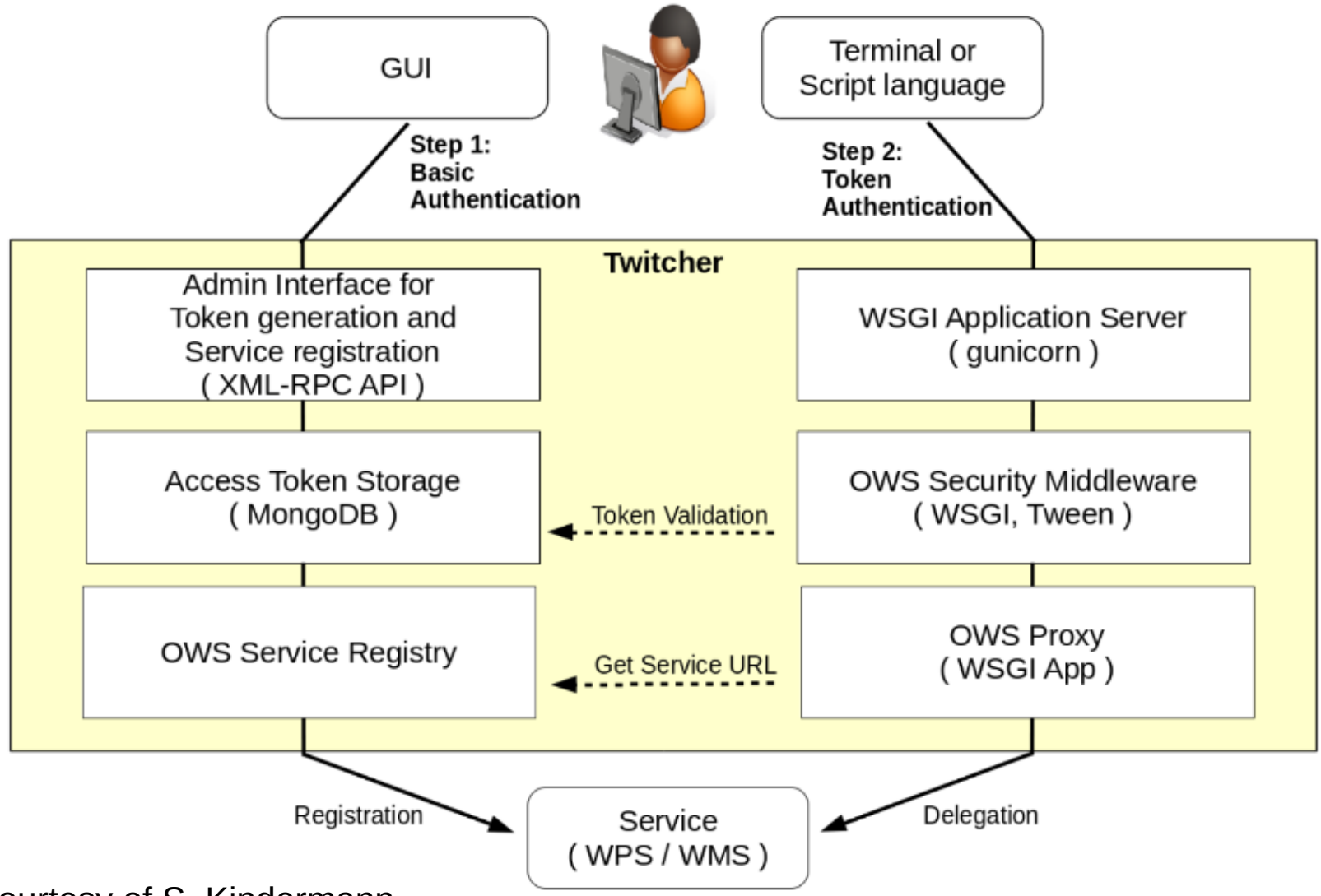
Get the *statusLocation* link from the response ...

In [3]:
```
import urllib
response = urllib.urlopen(url)
from lxml import etree
tree = etree.parse(response)
#print etree.tostring(tree)
print tree.getroot().get("statusLocation")
```

```
http://localhost:8090/wpsoutputs/emu/pywps-afd42c3a-a7d7-11e4-9a2a-68f72837e1b4.xml
```

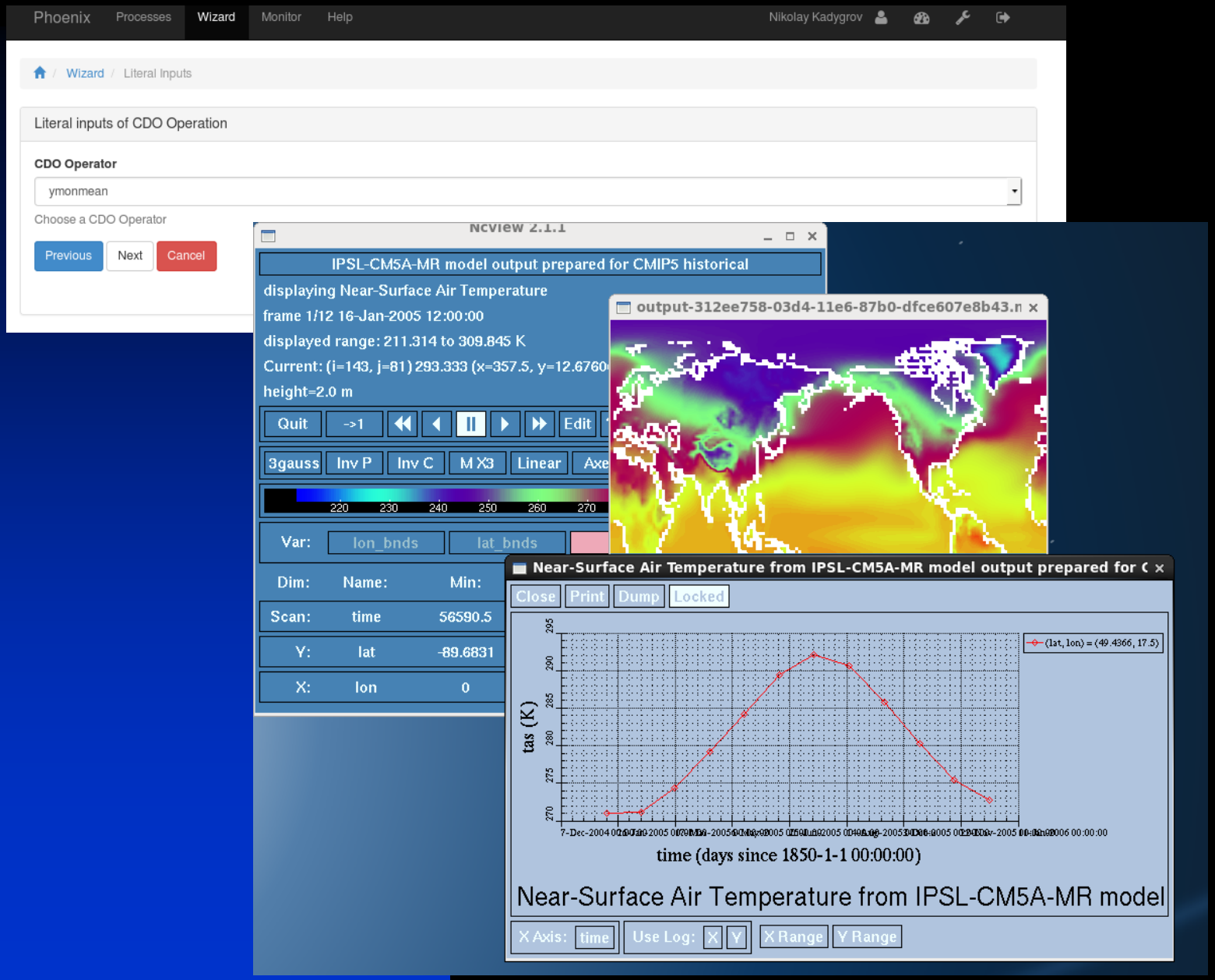
... check the url until process is finished.

Security – Twitcher Security Proxy

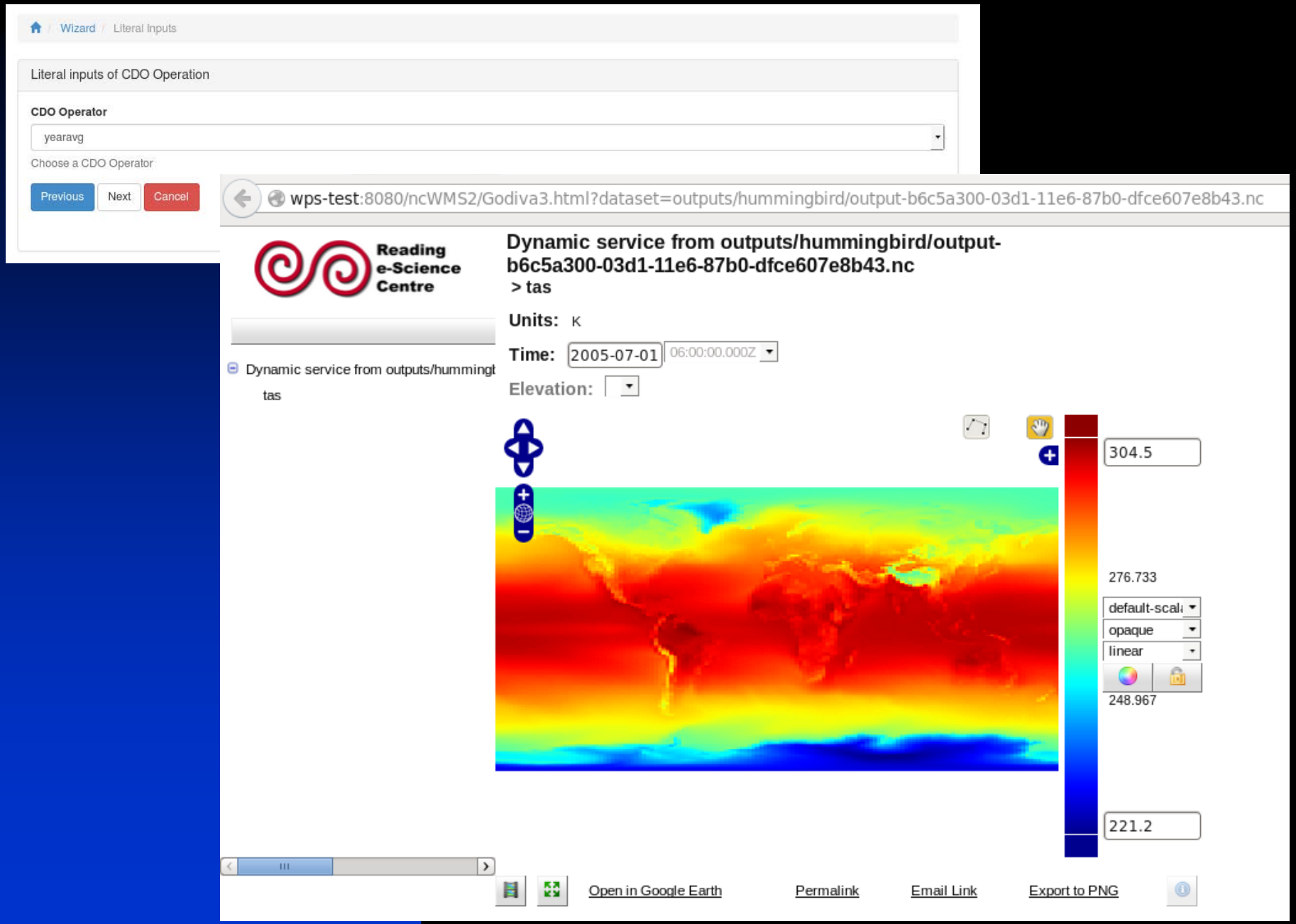


Slide courtesy of S. Kindermann

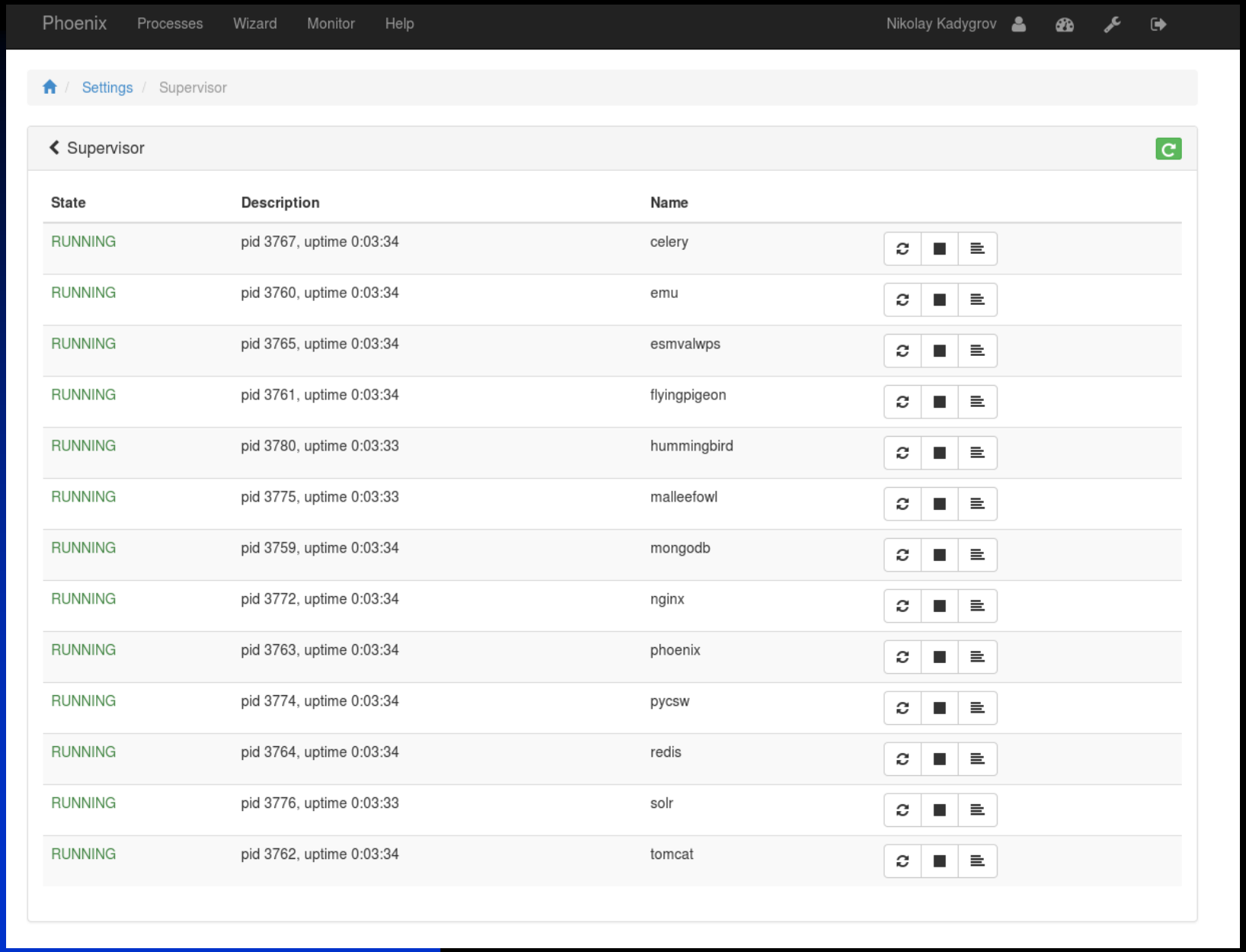
CDO seasonal cycle



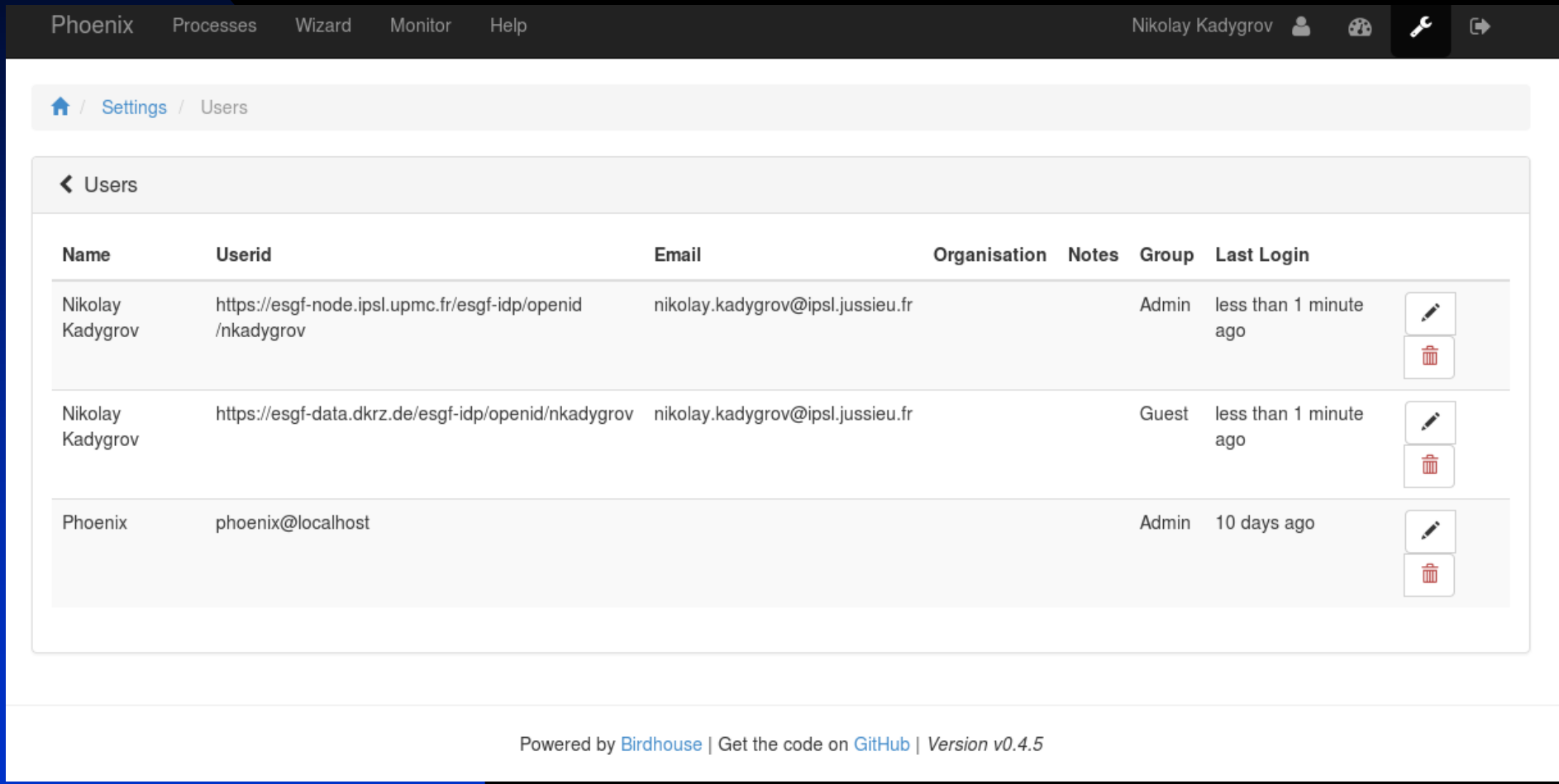
CDO yearavg + ncWMS visualisation



Supervisor


Phoenix
Processes
Wizard
Monitor
Help
Nikolay Kadygrov
Settings
Supervisor
Supervisor
State
Description
Name
RUNNING
pid 3767, uptime 0:03:34
celery
RUNNING
pid 3760, uptime 0:03:34
emu
RUNNING
pid 3765, uptime 0:03:34
esmvalwps
RUNNING
pid 3761, uptime 0:03:34
flyingpigeon
RUNNING
pid 3780, uptime 0:03:33
hummingbird
RUNNING
pid 3775, uptime 0:03:33
malleefowl
RUNNING
pid 3759, uptime 0:03:34
mongodb
RUNNING
pid 3772, uptime 0:03:34
nginx
RUNNING
pid 3763, uptime 0:03:34
phoenix
RUNNING
pid 3774, uptime 0:03:34
pycsw
RUNNING
pid 3764, uptime 0:03:34
redis
RUNNING
pid 3776, uptime 0:03:33
solr
RUNNING
pid 3762, uptime 0:03:34
tomcat

Users


Phoenix
Processes
Wizard
Monitor
Help
Nikolay Kadygrov
Settings
Users
Users
Name
Userid
Email
Organisation
Notes
Group
Last Login
Nikolay Kadygrov
https://esgf-node.ipsl.upmc.fr/esgf-idp/openid/nkadygrov
nikolay.kadygrov@ipsl.jussieu.fr
Admin
less than 1 minute ago
Nikolay Kadygrov
https://esgf-data.dkrz.de/esgf-idp/openid/nkadygrov
nikolay.kadygrov@ipsl.jussieu.fr
Guest
less than 1 minute ago
Phoenix
phoenix@localhost
Admin
10 days ago
Powered by Birdhouse | Get the code on GitHub | Version v0.4.5