Malleefowl Documentation

Release 0.7.0

Carsten Ehbrecht

Dec 06, 2018

Contents:

Malleefowl (the bird) *Malleefowl is a bird . . .*

A Web Processing Service for Climate Data Access and Workflows.

- Free software: Apache Software License 2.0
- Documentation: https://malleefowl.readthedocs.io.

CHAPTER 1

Credits

This package was created with Cookiecutter and the bird-house/cookiecutter-birdhouse project template.

1.1 Installation

1.1.1 Install from Conda

Warning: TODO: Prepare Conda package.

1.1.2 Install from GitHub

Check out code from the Malleefowl GitHub repo and start the installation:

```
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl
$ conda env create -f environment.yml
$ source activate malleefowl
$ python setup.py develop
```

... or do it the lazy way

The previous installation instructions assume you have Anaconda installed. We provide also a `Makefile` to run this installation without additional steps:

```
$ git clone https://github.com/bird-house/malleefowl.git
$ cd malleefowl
$ make clean    # cleans up a previous Conda environment
$ make install  # installs Conda if necessary and runs the above installation steps
```

1.1.3 Start Malleefowl PyWPS service

After successful installation you can start the service using the `malleefowl` command-line.

```
$ malleefowl --help # show help
$ malleefowl start  # start service with default configuration

OR

$ malleefowl start --daemon # start service as daemon
loading configuration
forked process id: 42
```

The deployed WPS service is by default available on:

http://localhost:5000/wps?service=WPS&version=1.0.0&request=GetCapabilities.

Note: Remember the process ID (PID) so you can stop the service with `kill PID`.

You can find which process uses a given port using the following command (here for port 5000):

```
$ netstat -nlp | grep :5000
```

Check the log files for errors:

```
$ tail -f  pywps.log
```

... or do it the lazy way

You can also use the `Makefile` to start and stop the service:

```
$ make start
$ make status
$ tail -f pywps.log
$ make stop
```

1.1.4 Run Malleefowl as Docker container

You can also run Malleefowl as a Docker container.

Warning: TODO: Describe Docker container support.

1.1.5 Use Ansible to deploy Malleefowl on your System

Use the Ansible playbook for PyWPS to deploy Malleefowl on your system.

1.2 Configuration

Warning: Please read the PyWPS documentation to find details about possible configuration options.

1.2.1 Command-line options

You can overwrite the default PyWPS configuration by using command-line options. See the Malleefowl help which options are available:

```
$ malleefowl start --help
--hostname HOSTNAME        hostname in PyWPS configuration.
--port PORT                port in PyWPS configuration.
```

Start service with different hostname and port:

```
$ malleefowl start --hostname localhost --port 5001
```

1.2.2 Use a custom configuration file

You can overwrite the default PyWPS configuration by providing your own PyWPS configuration file (just modifiy the options you want to change). Use one of the existing `sample-*.cfg` files as example and copy them to `etc/custom.cfg`.

For example change the hostname (*demo.org*) and logging level:

```
$ cd malleefowl
$ vim etc/custom.cfg
$ cat etc/custom.cfg
[server]
url = http://demo.org:5000/wps
outputurl = http://demo.org:5000/outputs

[logging]
level = DEBUG
```

Start the service with your custom configuration:

```
# start the service with this configuration
$ malleefowl start -c etc/custom.cfg
```

Read the PyWPS documentation for futher options and details.

1.2.3 Configure path to data archive

Malleefowl extends the configuration of PyWPS with a *data* section.

[data]

archive_root path to a *read-only* ESGF data archive which is used by the download process to make use of a local ESGF archive. You can configure several archives paths by using a colon `:` as seperator. Default: */tmp/archive*.

cache_path path to a *writeable* cache folder which is used by the download process to store files. Default: *PYWPS_OUTPUTPATH/cache*.

archive_node an option to specify an ESGF data provider for site specfic settings. Possible values: *default*, *dkrz*, *ipsl*. Default: *default*.

Example

```
[server]
url = http://demo.org:5000/wps
outputurl = http://demo.org:5000/outputs
outputpath = /data/pywps/outputs

[data]
archive_root = /data/archive/cmip5:/data/archive/cordex
cache_path = /data/cache
archive_node = default
```

1.3 Developer Guide

1.3.1 Building the docs

First install dependencies for the documentation:

```
$ make bootstrap_dev
$ make docs
```

1.3.2 Running tests

Run tests using pytest.

First activate the `malleefowl` Conda environment and install `pytest`.

```
$ source activate malleefowl
$ conda install pytest flake8  # if not already installed
```

Run quick tests (skip slow and online):

```
$ pytest -m 'not slow and not online'"
```

Run all tests:

```
$ pytest
```

Check pep8:

```
$ flake8
```

1.3.3 Run tests the lazy way

Do the same as above using the `Makefile`.

```
$ make test
$ make testall
$ make pep8
```

1.3.4 Bump a new version

Make a new version of Malleefowl in the following steps:

- Make sure everything is commit to GitHub.
- Update `CHANGES.rst` with the next version.
- Dry Run: `bumpversion --dry-run --verbose --new-version 0.8.1 patch`
- Do it: `bumpversion --new-version 0.8.1 patch`
- ... or: `bumpversion --new-version 0.9.0 minor`
- Push it: `git push`
- Push tag: `git push --tags`

See the bumpversion documentation for details.

1.4 Processes

- *Download*
- *ESGSearch*
- *ThreddsDownload*
- *Workflow*

1.4.1 Download

class malleefowl.processes.wps_download.**Download**

download Download files (v0.9)

Downloads files and provides file list as json document.

Parameters **resource** (*string*) – URL pointing to your resource which should be downloaded.

Returns **output** – Json document with list of downloaded files with file url.

Return type *application/json*

References

- Birdhouse
- User Guide

which should be downloaded.

The downloader first checks if the file is available in the local ESGF archive or cache. If not then the file will be downloaded and stored in a local cache. As a result it provides a list of local `file://` paths to the requested files.

The downloader does not download files if they are already in the ESGF archive or in the local cache.

1.4.2 ESGSearch

class malleefowl.processes.wps_esgsearch.**ESGSearchProcess**

esgsearch ESGF Search (v0.6)

Search ESGF datasets, files and aggreations.

Parameters

- **url** (*string, optional*) – URL of ESGF Search Index which is used for search queries. Example: http://esgf-data.dkrz.de/esg-search
- **distrib** (*boolean, optional*) – If flag is set then a distributed search will be run.
- **replica** (*boolean, optional*) – If flag is set then search will include replicated datasets.
- **latest** (*boolean, optional*) – If flag is set then search will include only latest datasets.
- **temporal** (*boolean, optional*) – If flag is set then search will use temporal filter.
- **search_type** (*{'Dataset', 'File', 'Aggregation'}, optional*) – Search on Datasets, Files or Aggregations.
- **constraints** (*string, optional*) – Constraints as list of key/value pairs.Example: project:CORDEX, time_frequency:mon, variable:tas
- **query** (*string, optional*) – Freetext query. For Example: temperatue
- **start** (*dateTime, optional*) – Startime: 2000-01-11T12:00:00Z
- **end** (*dateTime, optional*) – Endtime: 2005-12-31T12:00:00Z

- **limit** (*{'0', '1', '2', '5', '10', '20', '50', '100', '200'}, optional*) – Maximum number of datasets in search result
- **offset** (*integer, optional*) – Start search of datasets at offset.

Returns

- **output** (*application/json*) – JSON document with search result, a list of URLs to files on ESGF archive nodes.
- **summary** (*application/json*) – JSON document with search result summary
- **facet_counts** (*application/json*) – JSON document with facet counts for constraints.

References

- Birdhouse
- User Guide

to get a list of matching files on ESGF data nodes. It is using esgf-pyclient Python client for the ESGF search API.

In addition to the esgf-pyclient the process checks if local replicas are available and would return the replica files instead of the original one.

The result is a JSON document with a list of `http://` URLs to files on ESGF data nodes.

TODO: bbox constraint for datasets

1.4.3 ThreddsDownload

class malleefowl.processes.wps_thredds.**ThreddsDownload**
thredds_download Download files from Thredds Catalog (v0.5)

Downloads files from Thredds Catalog and provides file list as JSON Document.

Parameters **url** (*string*) – URL of the catalog.

Returns **output** – JSON document with list of downloaded files with file url.

Return type *application/json*

References

- Birdhouse
- User Guide

1.4.4 Workflow

class malleefowl.processes.wps_workflow.**DispelWorkflow**
workflow Workflow (v0.7)

Runs Workflow with dispel4py.

Parameters **workflow** (*text/yaml*) – Workflow description in YAML.

Returns

- **output** (*text/yaml*) – Workflow result document in YAML.
- **logfile** (*text/plain*) – Workflow log file.

References

- Birdhouse
- User Guide

run WPS process for climate data (like cfchecker, climate indices with ocgis, . . .) with a given selection of input data (currently NetCDF files from ESGF data nodes).

Currently the Dispel4Py workflow engine is used.

The Workflow for ESGF input data is as follows:

Search ESGF files -> Download ESGF files -> Run choosen process on local (downloaded) ESGF files.

1.5 Changes

1.5.1 0.7.0 (2018-12-05)

Converted malleefowl to the new deployment without buildout (#36).

Changes:

- regenerated using cookiecutter.
- updated dependencies (pywps, dispel4py, . . .).
- works on Python 2.7 and 3.6.
- integrated previous processes.

1.5.2 0.6.8 (2018-09-06)

Bugfixes:

- Updated Buildout 2.12.1 (#33, #37).
- Fix pluggy dependency (#34, #35).
- Other fixes: #24, #31.

1.5.3 0.6.7 (2018-04-04)

- added demo service using werkzeug for testing.
- cleaned up documentation.
- added tutorial.
- removed esgf_logon process.
- fixed os.link error in download module.

1.5.4 0.6.6 (2017-08-10)

- fixed headers in solr and thredds workflow.
- updated pywps recipe 0.9.2.
- postgres db can be configured.
- added config option archive_node to handle different archive root paths.
- removed unused files (todo, examples, . . .)

1.5.5 0.6.5 (2017-05-18)

- handle boundingbox in workflow.
- updated pywps recipe 0.9.0.
- added wsgi application.

1.5.6 0.6.4 (2017-03-27)

- fixed query search parameter.
- fixed replica search parameter.
- added backward compatibility for constraints search parameter.
- update nginx conda package.

1.5.7 0.6.3 (2017-03-21)

- removed certificate parameter in download.
- provide headers to worker process in workflow.

1.5.8 0.6.2 (2017-02-14)

- update pywps recipe and archive-root config.

1.5.9 0.6.1 (2017-01-31)

- using pyesgf.logon.
- using `X509_USER_PROXY` variable in download.
- link downloaded files to download cache.

1.5.10 0.6.0 (2017-01-27)

- updated to pywps 4.0.0.

1.5.11 0.5.0 (2017-01-12)

- moved old swift code to examples.
- removed esgf download with openid.
- pep8 checks on tests.
- using `ignore_facet_check=True` option in esgf.search.

1.5.12 0.4.4 (2017-01-04)

- using __version__ constant.
- fixed install on ubuntu 16.04: updated conda environment (lxml, icu).

1.5.13 0.4.3 (2016-12-06)

- update wget=2.2 from conda defaults.

1.5.14 0.4.2 (2016-10-19)

- enabled pep8 check in travis.
- updated docker recipe.
- updated setuptools and buildout version.

1.5.15 0.4.1 (2016-09-28)

- pep8
- using owslib.wps.ComplexDataInput in workflow
- update conda env: owslib=0.13.0, removed gdal

1.5.16 0.4.0 (2016-07-11)

- using new buildout recipes.
- using conda environment.yml.

1.5.17 0.3.12 (2016-06-15)

- using pytest.
- pinned wget=1.15.

1.5.18 0.3.11 (2016-02-01)

- cleaned up tests.
- fixed thredds download process.

1.5.19 0.3.10 (2016-01-21)

- removed mktempfile and working_dir form malleefowl.process.
- using status.set instead of show_status in processes/

1.5.20 0.3.9 (2016-01-19)

- fixed esgf download.

1.5.21 0.3.8 (2016-01-18)

- fixed esgsearch and esgf_logon process.

1.5.22 0.3.7 (2016-01-05)

- use pywps.process.WPSProcess instead of malleefowl.process.WPSProcess.
- cleaned up malleefowl.config.
- updated dockerfile and recipe.

1.5.23 0.3.6 (2015-07-30)

- download: checks if url has "file" schema. Those files can be returned directly.

1.5.24 0.3.5 (2015-07-28)

- added solr search workflow.
- fixed esgf logon: port = "7512"

1.5.25 0.3.4 (2015-07-23)

- disabled "File_Thredds" search type ... using "File" search instead.

1.5.26 0.3.3 (2015-06-18)

- using python myproxyclient.

1.5.27 0.3.2 (2015-06-17)

- added download with openid.
- renamed myproxy_logon().
- updated tomcat/thredds recipe.

1.5.28 0.3.1 (2015-06-14)

- added thredds workflow
- download with *wget -x* to create directories in cache.
- fixed workflow process output parameter.

1.5.29 0.3.0 (2015-05-22)

- cleaned up processes ... download, esgsearch ...
- refactored workflow with dispel4py ... improved logging.

1.5.30 0.2.1 (2015-05-18)

- fixed adagucserver installation
- using buildout recipes: birdhousebuilder.recipe.adagucserver, birdhousebuilder.recipe.postgres
- swift cloud access processes added.
- log to stderr/supervisor.

1.5.31 0.2.0 (2015-03-24)

- update sphinx docs.
- using birdhouse environment.
- fixed mako_cache path.

1.5.32 0.1.8 (2015-01-17)

- adagucserver with postgres added.
- fixed buildout bootstrap.
- esgf search checks local replica
- esgf archive_path changed

1.5.33 0.1.7 (2014-12-19)

- wget download with thredding.
- added log-level to settings.
- Disabled map processes.
- wget process using local file archive.
- esgsearch process added.
- Disabled restflow.
- Using dispel4py workflow engine.

1.5.34 0.1.6 (2014-11-28)

- Added wpsfetch script to retrieve test data for unit tests.

1.5.35 0.1.5 (2014-11-26)

- changed config for cache_path and cache_url.
- Cleaned up unit tests.
- download method added.

1.5.36 0.1.4 (2014-11-24)

- Using buildout 2.x.

1.5.37 0.1.3 (2014-11-11)

- Fixed LD_LIBRARY_PATH for myproxy-logon. Should not use openssl library from anaconda.
- Replaced install.sh by Makefile.
- Dockerfile added.

1.5.38 0.1.2 (2014-10-21)

- Fixed pyOpenSSL dependency.
- Updated docs.
- Updated dependencies.
- Dockfile for automated builds added.

1.5.39 0.1.1 (2014-08-21)

- Changed default cache path.

1.5.40 0.1.0 (2014-08-18)

- First Release.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index