rook Documentation

Release 0.8.2

Carsten Ehbrecht

May 16, 2022

CONTENTS:

1 Documentation 3

2 Contributing 5

3 Tests 7

4 License 9

5 Credits 11

6 Indices and tables 27

rook (the bird) *The rook belongs to the crow family . . .*

rook *Remote Operations On Klimadaten.*

Rook is a Web Processing Service (WPS) of the roocs project to allow remote operations like subsetting on climate model data. This service provides a one-to-one mapping to the operations available in the daops library based on xarray.

CHAPTER

ONE

DOCUMENTATION

Learn more about rook in its official documentation at https://rook-wps.readthedocs.io.

Submit bug reports, questions and feature requests at https://github.com/roocs/rook/issues

CHAPTER

TWO

CONTRIBUTING

You can find information about contributing in our Developer Guide.

Please use bumpversion to release a new version.

CHAPTER

THREE

TESTS

The `tests` folder includes additional tests for a deployed rook service.

- Smoke test: ensure service is operational. See `tests/smoke/README.md`.
- Storm test: load-test using locust. See `tests/storm/README.md`.

CHAPTER

FOUR

LICENSE

Free software: Apache Software License 2.0

CHAPTER

FIVE

CREDITS

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

5.1 Installation

- *Install from Conda*
- *Install from GitHub*
- *Configure roocs*
- *Start rook PyWPS service*
- *Run rook as Docker container*
- *Use Ansible to deploy rook on your System*

5.1.1 Install from Conda

Warning: TODO: Prepare Conda package.

5.1.2 Install from GitHub

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

```
$ git clone https://github.com/roocs/rook.git
$ cd rook
```

Create Conda environment named *rook*:

```
$ conda env create -f environment.yml
$ source activate rook
```

Install rook app:

```
$ pip install -e .
OR
make install
```

For development you can use this command:

```
$ pip install -e ".[dev]"
OR
$ make develop
```

5.1.3 Configure roocs

rook is using daops for the operations. It needs a `roocs.ini` configuration file. You can overwrite the defaults by setting the environment variable `ROOCS_CONFIG`.

```
$ export ROOCS_CONFIG=~/.roocs.ini
```

There is an example in `etc/sample-roocs.ini`.

For more information on the configuration settings, see https://roocs-utils.readthedocs.io/en/latest/configuration.html

5.1.4 Start rook PyWPS service

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

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

OR

$ rook 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
```

5.1.5 Run rook as Docker container

You can also run rook as a Docker container.

Warning: TODO: Describe Docker container support.

5.1.6 Use Ansible to deploy rook on your System

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

5.2 Configuration

5.2.1 Command-line options

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

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

Start service with different hostname and port:

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

5.2.2 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 rook
$ vim etc/custom.cfg
$ cat etc/custom.cfg
[server]
url = http://demo.org:5000/wps
```

(continues on next page)

(continued from previous page)

```
outputurl = http://demo.org:5000/outputs

[logging]
level = DEBUG
```

Start the service with your custom configuration:

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

5.3 Developer Guide

- *Building the docs*
- *Add pre-commit hooks*
- *Running tests*
- *Run tests the lazy way*
- *Prepare a release*
- *Bump a new version*

Warning: To create new processes look at examples in Emu.

5.3.1 Building the docs

First install dependencies for the documentation:

```
$ make develop
```

Run the Sphinx docs generator:

```
$ make docs
```

5.3.2 Add pre-commit hooks

Before committing your changes, we ask that you install *pre-commit* in your environment. *Pre-commit* runs git hooks that ensure that your code resembles that of the project and catches and corrects any small errors or inconsistencies when you *git commit*:

```
$ conda install -c conda-forge pre_commit
$ pre-commit install
```

5.3.3 Running tests

Run tests using pytest.

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

```
$ source activate rook
$ pip install -r requirements_dev.txt  # if not already installed
OR
$ make develop
```

Configure the pywps configuration with path to test data.

```
$ export PYWPS_CFG=/path/to/test/pywps.cfg
```

Run quick tests (skip slow and online):

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

Run all tests:

```
$ pytest
```

Check pep8:

```
$ flake8
```

5.3.4 Run tests the lazy way

Do the same as above using the `Makefile`.

```
$ make test
$ make test-all
$ make lint
```

5.3.5 Prepare a release

Update the Conda specification file to build identical environments on a specific OS.

Note: You should run this on your target OS, in our case Linux.

```
$ conda env create -f environment.yml
$ source activate rook
$ make clean
$ make install
$ conda list -n rook --explicit > spec-list.txt
```

5.3.6 Bump a new version

Make a new version of rook 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.

5.4 Notebooks

You can use the rooki Python client to use the rook service. See the online notebooks with examples.

5.5 Processes

- *Subset*
- *Average*
- *Orchestrate*

5.5.1 Subset

5.5.2 Average

5.5.3 Orchestrate

5.6 Provenance

- *Introduction*
- *Overview of PROV*
- *Example: Workflow with Subsetting Operators*
- *Related work in other Projects*

5.6.1 Introduction

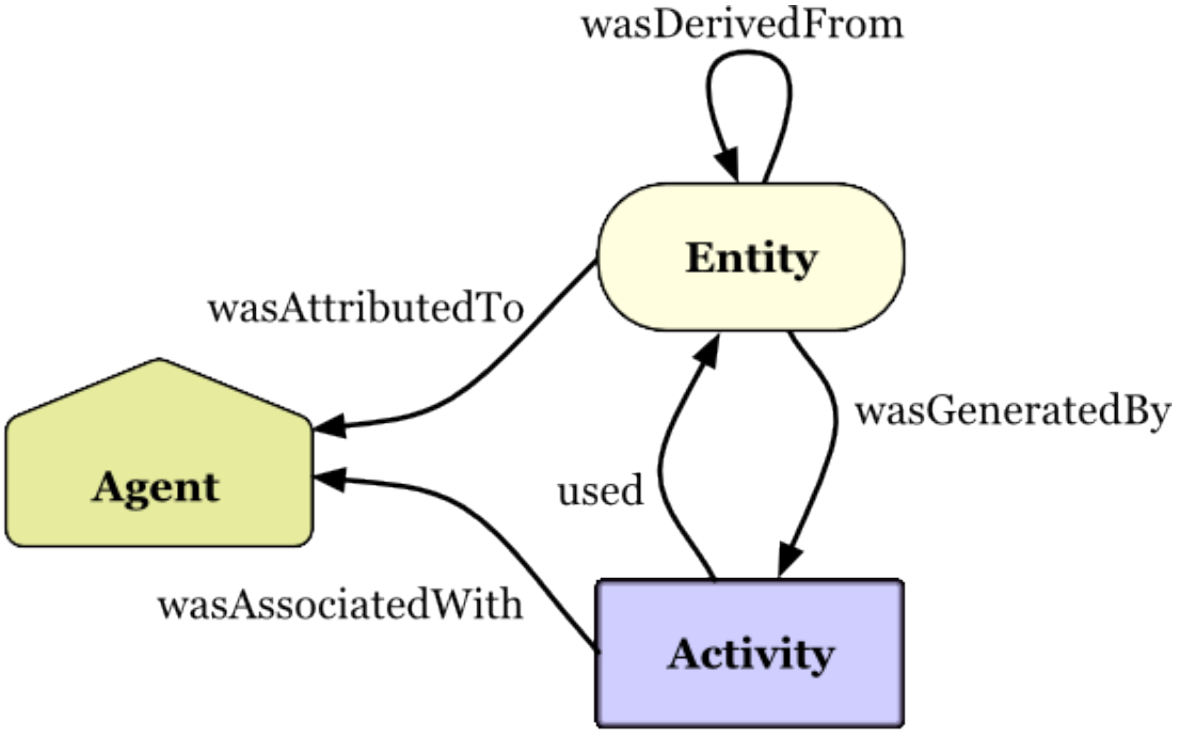
The *rook* processes are recording provenance information about the process execution details. This information includes:

- used software and versions (`rook`, `daops`, ...)
- applied operators like `subset` and `average`
- used input data and parameters (cmip6 dataset, time, area)
- generated outputs (NetCDF files)
- execution time (start-time and end-time)

This information is described with the W3C PROV standard and using the Python PROV Library

5.6.2 Overview of PROV

The W3C PROV Primer document gives an overview of the W3C PROV standard.



A PROV document consists of *agents*, *activities* and *entities*. These can be connected via PROV *relations* like *wasDerivedFrom*.

Entities

W3C PROV In PROV, physical, digital, conceptual, or other kinds of thing are called *entities*.

In *rook* we use *entities* for:

- workflow description,
- input datasets and
- resulting output NetCDF files.

Activities

W3C PROV *Activities* are how entities come into existence and how their attributes change to become new entities, often making use of previously existing entities to achieve this.

In *rook* we use *activities* for:

- operators like `subset` and `average`.
- processes like `orchestrate` to run a workflow.

Agent

W3C PROV An *agent* takes a role in an activity such that the agent can be assigned some degree of responsibility for the activity taking place. An agent can be a person, a piece of software or an organisation.

In *rook* we use *agents* for:

- software like *rook* and *daops*,
- organisations like *Copernicus Climate Data Store*.

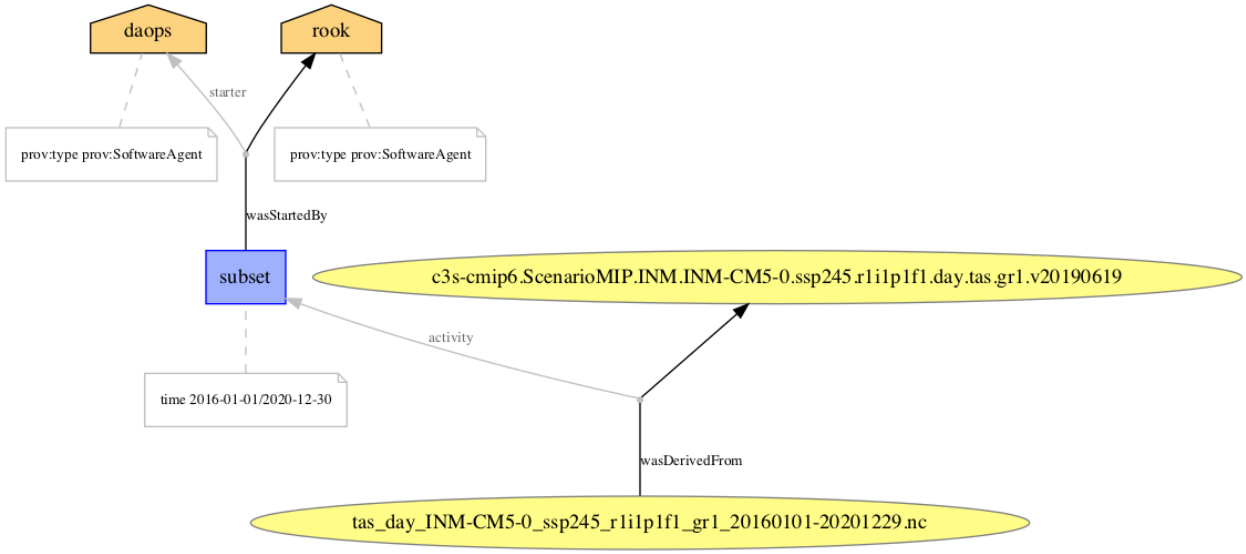
Namespaces

W3C PROV Using URIs and namespaces, a provenance record can draw from multiple sources on the Web.

We use namespaces to use existing PROV vocabularies like `prov:SoftwareAgent`. These are for example:

- PROV (by W3C): https://www.w3.org/ns/prov/
- PROVONE (by DataONE): https://purl.dataone.org/provone/2015/01/15/ontology
- dcterms (Dublin Core Metadata): https://dublincore.org/specifications/dublin-core/dcmi-terms/

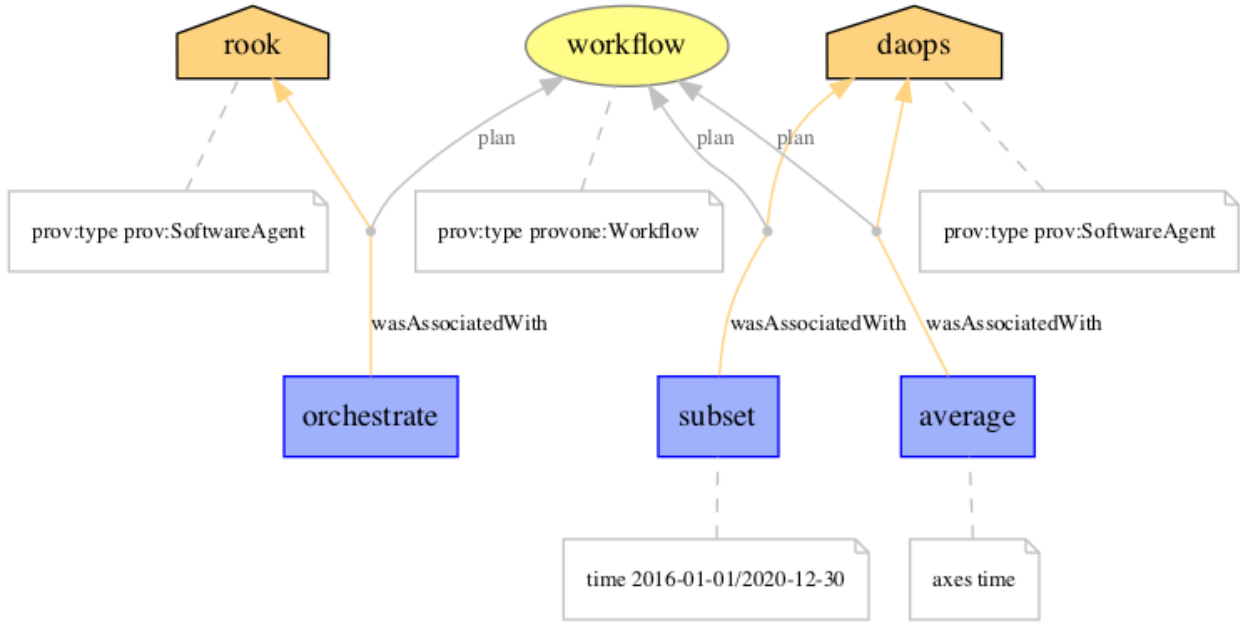
Subset Example



The *activity* `subset` is started by the software *agent* `daops` (Python library) which was triggered by `rook` (data-reduction service).

The NetCDF file `tas_day_...nc` *entity* was derived from `c3s-cmip6` dataset *entity* using the *activity* `subset`.

Workflow Example



W3C PROV Plans Activities may follow pre-defined procedures, such as recipes, tutorials, instructions, or workflows. PROV refers to these, in general, as *plans*.

In W3C PROV workflows are named *plans*.

The *activity* `orchestrate` is started by the *agent* `rook`. It uses a workflow document `entity` (*plan*) which consists of a `subset` and `average` *activity*. These activities are started by the software *agent* `daops`.

5.6.3 Example: Workflow with Subsetting Operators

The rooki client for rook has example notebooks for process executions and displaying the provenance information.

You can run the `orchestrate` process to execute a workflow with subsetting operators and show the provenance document:

```
from rooki import operators as ops
wf = ops.Subset(
      ops.Subset(
          ops.Input(
              'tas', ['c3s-cmip6.ScenarioMIP.INM.INM-CM5-0.ssp245.r1i1p1f1.day.tas.gr1.
↪v20190619']
          ),
          time="2016-01-01/2020-12-30",
      ),
      time="2017-01-01/2017-12-30",
)
resp = wf.orchestrate()
# show URLs of output files
resp.download_urls()
# show URL to provenance document
resp.provenance()
# show URL to provenance image
resp.provenance_image()
```

The response of the process includes a provenance document in PROV-JSON format:

```
{
  "prefix": {
    "provone": "http://purl.dataone.org/provone/2015/01/15/ontology#",
    "dcterms": "http://purl.org/dc/terms/",
    "default": "http://purl.org/roocs/prov#"
  },
  "agent": {
    "copernicus_CDS": {
      "prov:type": "prov:Organization",
      "dcterms:title": "Copernicus Climate Data Store"
    },
    "rook": {
      "prov:type": "prov:SoftwareAgent",
      "dcterms:source": "https://github.com/roocs/rook/releases/tag/v0.2.0"
    },
    "daops": {
      "prov:type": "prov:SoftwareAgent",
      "dcterms:source": "https://github.com/roocs/daops/releases/tag/v0.3.0"
    }
  },
  "wasAttributedTo": {
    "_:id1": {
      "prov:entity": "rook",
      "prov:agent": "copernicus_CDS"
    }
  },
```

(continues on next page)

(continued from previous page)

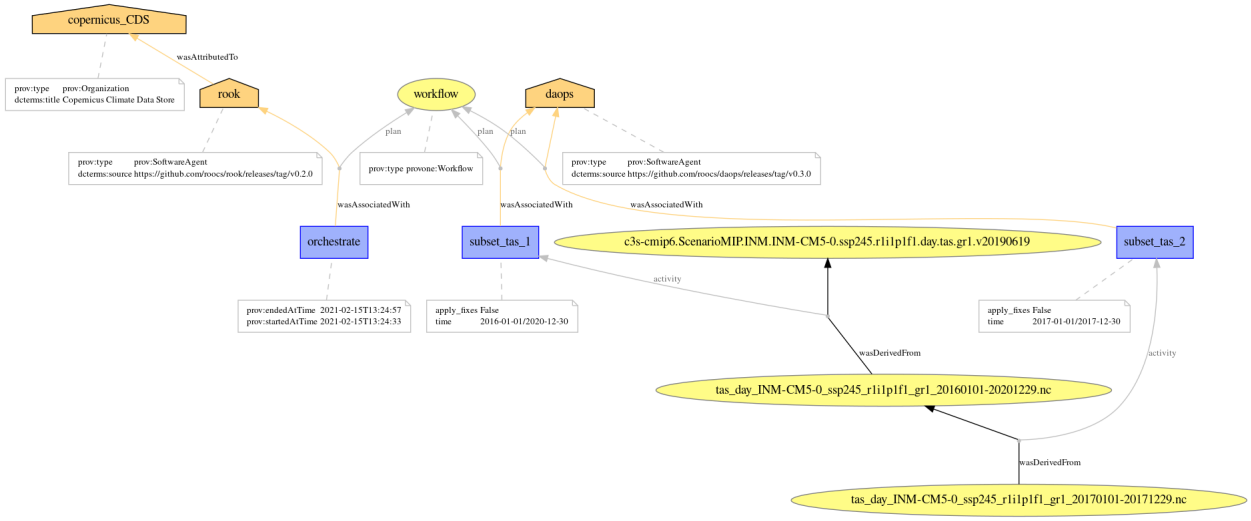

```
  "entity": {
    "workflow": {
      "prov:type": "provone:Workflow"
    },
    "c3s-cmip6.ScenarioMIP.INM.INM-CM5-0.ssp245.r1i1p1f1.day.tas.gr1.v20190619": {},
    "tas_day_INM-CM5-0_ssp245_r1i1p1f1_gr1_20160101-20201229.nc": [{}, {}],
    "tas_day_INM-CM5-0_ssp245_r1i1p1f1_gr1_20170101-20171229.nc": {}
  },
  "activity": {
    "orchestrate": [{
      "prov:startedAtTime": "2021-02-15T13:24:33"
    }, {
      "prov:endedAtTime": "2021-02-15T13:24:57"
    }],
    "subset_tas_1": {
      "time": "2016-01-01/2020-12-30",
      "apply_fixes": false
    },
    "subset_tas_2": {
      "time": "2017-01-01/2017-12-30",
      "apply_fixes": false
    }
  },
  "wasAssociatedWith": {
    "_:id2": {
      "prov:activity": "orchestrate",
      "prov:agent": "rook",
      "prov:plan": "workflow"
    },
    "_:id3": {
      "prov:activity": "subset_tas_1",
      "prov:agent": "daops",
      "prov:plan": "workflow"
    },
    "_:id5": {
      "prov:activity": "subset_tas_2",
      "prov:agent": "daops",
      "prov:plan": "workflow"
    }
  },
  "wasDerivedFrom": {
    "_:id4": {
      "prov:generatedEntity": "tas_day_INM-CM5-0_ssp245_r1i1p1f1_gr1_20160101-20201229.nc
↪",
      "prov:usedEntity": "c3s-cmip6.ScenarioMIP.INM.INM-CM5-0.ssp245.r1i1p1f1.day.tas.
↪gr1.v20190619",
      "prov:activity": "subset_tas_1"
    },
    "_:id6": {
      "prov:generatedEntity": "tas_day_INM-CM5-0_ssp245_r1i1p1f1_gr1_20170101-20171229.nc
↪",
      "prov:usedEntity": "tas_day_INM-CM5-0_ssp245_r1i1p1f1_gr1_20160101-20201229.nc",
```


(continues on next page)

(continued from previous page)

```
      "prov:activity": "subset_tas_2"
    }
  }
}
```

This provenance document can also be displayed as an image:



5.6.4 Related work in other Projects

The ESMValTool project is recording provenance information of scientific workflows run as diagnostics.

The Climate4Impact project is using provenance to record the workflow of data staging and creating Jupyter notebooks.

5.7 Changes

5.7.1 0.8.2 (2022-05-16)

- Updated to daops 0.8.1 and clisops 0.9.1 (#211).
- Added tests to check correct metadata (#211).

5.7.2 0.8.1 (2022-04-20)

- Updated to roocs-utils 0.6.1 (#209).
- Fixed *director* for new *average_time* operator (#208).
- Added smoke tests for c3s-cmip5 and c3s-cordex (#208, #209).

5.7.3 0.8.0 (2022-04-14)

- Added “average” and “average_time” operators (#191, #206).
- Removed “diff” operator (#204).
- Cleaned up workflow and tests (#205).
- Added changes for CMIP6 decadal (#202).
- Updated to daops 0.8.0 (#207).
- Updated to clisops 0.9.0 (#207).
- Updated to latest bokeh 2.4.2 in dashboard (#207).
- Updated pre-commit (#207).
- Updated pywps 4.5.2 (#203, #207).

5.7.4 0.7.0 (2021-11-08)

- Added “subset-by-point” (#190).
- Updated to clisops 0.7.0.
- Updated to daops 0.7.0.
- Updated dashboard (#195).
- Updated provenance namespace (#188).

5.7.5 0.6.2 (2021-08-11)

- Update pywps 4.4.5 (#186).
- Updated provenance types and ids (#184).
- Update dashboard (#183).

5.7.6 0.6.1 (2021-06-18)

- Added initial dashboard (#182).
- Update clisops 0.6.5.

5.7.7 0.6.0 (2021-05-20)

- Inventory urls removed from `etc/roocs.ini`. Intake catalog url now lives in daops. (#175)
- Intake catalog base and search functionality moved to daops. Database intake implementation remains in rook. (#175)
- Updated to roocs-utils 0.4.2.
- Updated to clisops 0.6.4.
- Updated to daops 0.6.0.
- Added initial usage process (#178)

5.7.8 0.5.0 (2021-04-01)

- Updated pywps 4.4.2.
- Updated clisops 0.6.3.
- Updated roocs-utils 0.3.0.
- Use `FileMapper` for search results (#169).
- Using intake catalog (#148).

5.7.9 0.4.2 (2021-03-22)

- Updated clisops 0.6.2

5.7.10 0.4.1 (2021-03-21)

- Updated pywps 4.4.1 (#162, #154, #151).
- Use pywps `storage_copy_function=link` (#154).
- Updated director with InvalidCollection error (#153).
- Added locust (storm) tests (#141, #149, #155).
- Updated smoke tests (#134, #137).
- Cleaned requirements (#152).
- Fixed warning in workflow yaml loaded (#142).
- Removed original files option for average and added test (#136).

5.7.11 0.4.0 (2021-03-04)

- Removed cfunits, udunits2, cf-xarray and python-dateutil as dependencies.
- Use daops>=0.5.0
- Renamed axes input of `wps_average.Average` to dims
- Added wps_average to work with daops.ops.average (#126)
- Fixed tests for new inventory (#127)
- Use `apply_fixes=False` for average (#129)
- Added smoke tests (#131, #134)

5.7.12 0.3.1 (2021-02-24)

- Pin `cf_xarray <0.5.0` ... does not work with daops/clisops.

5.7.13 0.3.0 (2021-02-24)

- Fixed testdata using git-python (#123).
- Removed xfail where not needed (#121).
- Updated PyWPS 4.4.0 (#120).
- Updated provenance (#112, #114 ,#119).
- Fixed subset alignment (#117).
- `apply_fixes` and `original_files` option added for WPS processes and the `Operator` class (#111).
- Replaced travis with GitHub CI (#104).
- `director` module added. This makes decisions on what is returned - NetCDF files or original file URLs (#77, #83)
- `python-dateutil>=2.8.1` added as a new dependency.
- Allow no inventory option when processing datasets
- c3s-cmip6 dataset ids must now be identified by the use of `c3s-cmip6` (#87).
- Fixed workflow (#79, #75, #71).

5.7.14 0.2.0 (2020-11-19)

Changes:

- Build on cookiecutter template with `cruft` update.
- Available processes: `subset`, `orchestrate`.
- Using `daops` for subsetting operation.
- Using a simple workflow implementation for combining operators.
- Process outputs are provided as `Metalink` documents.
- Added initial support for provenance documentation.

5.7.15 0.1.0 (2020-04-03)

- First release.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search