arctool Documentation

Release 0.13.2

Tjelvar Olsson and Matthew Hartley

May 01, 2017

Contents

CHAPTER 1

Content

Archive scientific data sets

- Documentation: http://arctool.readthedocs.io
- GitHub: https://github.com/JIC-CSB/arctool
- PyPI: https://pypi.python.org/pypi/arctool
- Free software: MIT License

Overview

The arctool project project provides tools for archiving (scientific) data. It aims to help in three areas:

1. Adding structure and meta data to your project and files
2. Verifying the integrity of the files in your project
3. Creating archives for long term storage

Design philosophy

The tools in this project should produce outputs that can be understood without access to these tools. This is important as it is likely that the outputs of from these tools may outlive these tools.

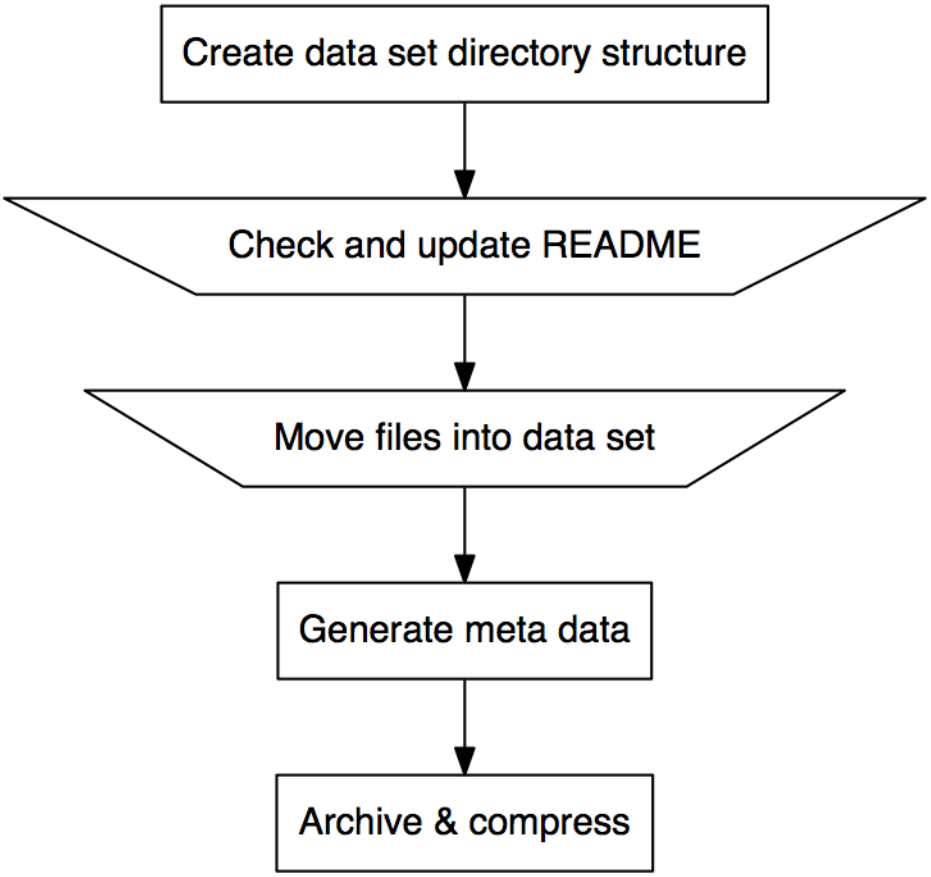
Usage

Overview

The `arctool` command line tool does several things.

1. Provides templates for meta data associated with a project in the plain text yaml file format
2. Provides a means to generate a manifest with meta data for all files in a data directory
3. Provides directory structure templates for archiving data
4. Provides wrappers to create tar archives
5. Provides wrappers to gzip tar archives
6. Provides wrappers to verify the integrity of files in gzipped tar archives

Below is a high level overview of the workflow.



Walkthrough

`arctool` is a tool for archiving data.

Install or load the tool

JIC users can load the `arctool` program on the cluster using the command below.

```
source /common/software/linuxbrew/Cellar/lmod/5.9.3/lmod/5.9.3/init/bash
module use /common/modulefiles/Core
module load arctool
```

To install the tool manually see installation_notes.

Creating a staging area

First you will need to create an archiving staging area.

```
$ mkdir archive_staging_area
$ cd archive_staging_area
```

Making a new archive

Warning: this section of the documentation assumes functionality that will be added in a future release. Specifically, the creation of project part of the directory structure is not yet implemented.

To start building an archive use `arctool new`, this will create a directory structure in the working directory (`archive_staging_area`) and prompt you to specify some meta data associated with the project.

```
$ arctool new

# Add ouput here
```

This results in the directory structure below.

```
$ tree some_project

# Add output here
```

Editing dataset meta data

Inspect and extend the `some_project/data_set_1/README.yml` as necessary. This file is meant to provide overall meta data of the data set.

```
$ cat some_project/data_set_1/README.yml

# Add output here
```

Moving data into the dataset in the staging area

Move your data to be archived into the `some_project/data_set_1/archive` directory.

```
$ mv ~/my_old_project/data_set_1/* some_project/data_set_1/archive/
```

Generating file meta data

Generate meta data for the files that you just moved into the `some_project/data_set_1/archive` directory.

```
$ arctool manifest create some_project/data_set_1
```

Creating the archive file

Create a tar ball of the data set.

```
$ arctool archive create some_project/data_set_1

# Add output here
```

Compressing the archive

Compress the archive using gzip compression.

```
$ arctool archive compress some_project/data_set_1

# Add output here
```

Moving the archive into long term storage

Finally move the gzipped tarball archive into your long term storage.

Deleting original data

Now you can delete the original data.

Tab completion

arctool supports tab completion for bash. To enable it, enter the following in your shell:

```
$ eval "$(_ARCTOOL_COMPLETE=source arctool)"
```

or add it to your .bashrc to enable permanently. You can also generate a sourceable bash shell with:

```
$ _ARCTOOL_COMPLETE=source arctool > arctool-complete.sh
```

Logging with fluentd

arctool sends logs to fluentd, a logging system. The fluentd server can be set in two ways:

1. By setting the FLUENTD_HOST environment variable, e.g.:

   ```
   $ export FLUENTD_HOST=my_host.domain
   ```

2. By specifying the host on the command line, e.g.:

   ```
   $ arctool --fluentd-host my_host.domain <command>
   ```

API documentation

arctool

arctool package.

arctool.archive

Module wrapping tar and gzip.

class arctool.archive.**ArchiveDataSet**(*name*)
Class for creating specific archive datasets.

class arctool.archive.**ArchiveFile**
Class for working with tarred/gzipped archive datasets.

Initialising using a dataset is used for creating archives, while initialising from a file is used for extracting and verifying.

classmethod from_file(*path*)
Read archive from file, either .tar or .tar.gz

summarise()
Return dictionary with summary information about an archive.

Returns dictionary of summary information about the archive

verify_all()
Verify all files in archive.

Returns True if all files verify, False otherwise.

verify_file(*file_in_archive*)
Verify single file in archive.

Parameters **file_in_archive** – file to verify

Returns True if checksum matches, False otherwise.

class arctool.archive.**ArchiveFileBuilder**
Class for building up tarred archive datasets.

classmethod from_path(*path*)
Return *arctool.archive.ArchiveFileBuilder*.

Parsed from a archive dataset directory.

persist_to_tar(*path*)
Write archive dataset to tarball.

arctool.archive.**compress_archive**(*path*, *n_threads=8*)
Compress the (tar) archive at the given path.

Uses pigz for speed.

Parameters

- **path** – path to the archive tarball
- **n_threads** – number of threads for pigz to use

Returns path to created gzip file

arctool.utils

Module containing arctool API.

arctool.utils.**new_archive_dataset**(*staging_path*, *descriptive_metadata*)
Create new archive in the staging path.

This creates an initial skeleton directory structure that includes a top level README.yml file.

Parameters

- **staging_path** – path to archiving staging area
- **descriptive_metadata** – dictionary with information which will populate README.yml

Returns (dataset, path to newly created data set archive in the staging area)

arctool.utils.**readme_yml_is_valid**(*yml_string*)
Return True if string representing README.yml content is valid.

Parameters **yml_string** – string representing content of readme file

Returns bool

CHANGELOG

This project uses semantic versioning. This change log uses principles from keep a changelog.

[Unreleased]

Added

Changed

Deprecated

Removed

Fixed

Security

[0.13.2] - 2017-02-28

Fixed

- Templates are now included in the package

[0.13.1] - 2017-02-23

Fixed

- Fix use of `arctool.__version__` rather than `dtool.__version__`

[0.13.0] - 2017-02-23

Changed

- Made to work with dtool version >= 0.12.1

[0.12.1] - 2017-02-23

Added

- Stringent constrain on dtool version == 0.11.0

[0.12.0] - 2017-02-09

Added

- `arctool.slurm` module moved across from `dtool` package

[0.11.0] - 2017-02-09

Changed

- Package split out of dtool github.com/JIC-CSB/dtool

MIT License

Python Module Index

a

Index