# aiida-diff

***Release 1.0.0***

**Nov 06, 2019**

# Contents

`aiida-diff` is available at http://github.com/aiidateam/aiida-diff

CHAPTER 1

# User guide

## 1.1 Getting started

This page should contain a short guide on what the plugin does and a short example on how to use the plugin.

### 1.1.1 Installation

Use the following commands to install the plugin:

```
git clone https://github.com/aiidateam/aiida-diff .
cd aiida-diff
pip install -e .  # also installs aiida, if missing (but not postgres)
#pip install -e .[pre-commit,testing] # install extras for more features
verdi quicksetup  # better to set up a new profile
verdi calculation plugins  # should now show your calclulation plugins
```

Then use `verdi code setup` with the `diff` input plugin to set up an AiiDA code for aiida-diff.

### 1.1.2 Usage

A quick demo of how to submit a calculation:

```
verdi daemon start         # make sure the daemon is running
cd examples
verdi run test_submit.py        # submit test calculation
verdi calculation list -a  # check status of calculation
```

If you have already set up your own aiida_diff code using `verdi code setup`, you may want to try the following command:

```
diff-submit  # uses aiida_diff.cli
```

### 1.1.3 Available calculations

**calcjob**aiida_diff.calculations.**DiffCalculation**

AiiDA calculation plugin wrapping the diff executable.

Simple AiiDA plugin wrapper for 'diffing' two files.

**Inputs:**

- **code**, *Code*, required – The *Code* to use for this job.
- **file1**, *SinglefileData*, required – First file to be compared.
- **file2**, *SinglefileData*, required – Second file to be compared.
- **metadata**, *Namespace*
  - **call_link_label**, *(basestring)*, optional, *non_db* – The label to use for the *CALL* link if the process is called by another process.
  - **computer**, *Computer*, optional, *non_db* – When using a "local" code, set the computer on which the calculation should be run.
  - **description**, *(basestring)*, optional, *non_db* – Description to set on the process node.
  - **dry_run**, *bool*, optional, *non_db* – When set to *True* will prepare the calculation job for submission but not actually launch it.
  - **label**, *(basestring)*, optional, *non_db* – Label to set on the process node.
  - **options**, *Namespace*
    - **account**, *(basestring)*, optional, *non_db* – Set the account to use in for the queue on the remote computer
    - **append_text**, *(basestring)*, optional, *non_db* – Set the calculation-specific append text, which is going to be appended in the scheduler-job script, just after the code execution
    - **custom_scheduler_commands**, *(basestring)*, optional, *non_db* – Set a (possibly multiline) string with the commands that the user wants to manually set for the scheduler. The difference of this option with respect to the *prepend_text* is the position in the scheduler submission file where such text is inserted: with this option, the string is inserted before any non-scheduler command
    - **environment_variables**, *dict*, optional, *non_db* – Set a dictionary of custom environment variables for this calculation
    - **import_sys_environment**, *bool*, optional, *non_db* – If set to true, the submission script will load the system environment variables
    - **input_filename**, *(basestring)*, optional, *non_db* – Filename to which the input for the code that is to be run will be written.
    - **max_memory_kb**, *int*, optional, *non_db* – Set the maximum memory (in KiloBytes) to be asked to the scheduler
    - **max_wallclock_seconds**, *int*, optional, *non_db* – Set the wallclock in seconds asked to the scheduler
    - **mpirun_extra_params**, *(list, tuple)*, optional, *non_db* – Set the extra params to pass to the mpirun (or equivalent) command after the one provided in computer.mpirun_command. Example: mpirun -np 8 extra_params[0] extra_params[1] … exec.x
    - **output_filename**, *(basestring)*, optional, *non_db*
    - **parser_name**, *(basestring)*, optional, *non_db*

        - **prepend_text**, *(basestring)*, optional, *non_db* – Set the calculation-specific prepend text, which is going to be prepended in the scheduler-job script, just before the code execution
        - **priority**, *(basestring)*, optional, *non_db* – Set the priority of the job to be queued
        - **qos**, *(basestring)*, optional, *non_db* – Set the quality of service to use in for the queue on the remote computer
        - **queue_name**, *(basestring)*, optional, *non_db* – Set the name of the queue on the remote computer
        - **resources**, *dict*, optional, *non_db*
        - **scheduler_stderr**, *(basestring)*, optional, *non_db* – Filename to which the content of stderr of the scheduler will be written.
        - **scheduler_stdout**, *(basestring)*, optional, *non_db* – Filename to which the content of stdout of the scheduler will be written.
        - **withmpi**, *bool*, optional, *non_db* – Set the calculation to use mpi
      - **store_provenance**, *bool*, optional, *non_db* – If set to *False* provenance will not be stored in the database.
  - **parameters**, *DiffParameters*, required – Command line parameters for diff

**Outputs:**

- **diff**, *SinglefileData*, required – diff between file1 and file2.
- **remote_folder**, *RemoteData*, required – Input files necessary to run the process will be stored in this folder node.
- **retrieved**, *FolderData*, required – Files that are retrieved by the daemon will be stored in this node. By default the stdout and stderr of the scheduler will be added, but one can add more by specifying them in *CalcInfo.retrieve_list*.

## 1.2 Tutorial

This page can contain a simple tutorial for your code.

### 1.2.1 What we want to achieve

#### Step 1

Some text

#### Step 2

Some other text

### 1.2.2 The final result

Some text

CHAPTER 2

# Developer guide

## 2.1 Running the tests

The following will discover and run all unit test:

```
pip install -e .[testing]
pytest -v
```

## 2.2 Automatic coding style checks

Enable enable automatic checks of code sanity and coding style:

```
pip install -e .[pre-commit]
pre-commit install
```

After this, the yapf formatter, the pylint linter and the prospector code analyzer will run at every commit.

If you ever need to skip these pre-commit hooks, just use:

```
git commit -n
```

## 2.3 Continuous integration

`aiida-diff` comes with a `.travis.yml` file for continuous integration tests on every commit using Travis CI. It will:

1. run all tests for the `django` and `sqlalchemy` ORM
2. build the documentation
3. check coding style and version number (not required to pass by default)

Just enable Travis builds for the `aiida-diff` repository in your Travis account.

`aiida-diff` also includes an `azure-pipelines.yml` file for continuous integration tests using Azure Pipelines.

## 2.4 Online documentation

The documentation of `aiida-diff` is ready for ReadTheDocs:

Simply add the `aiida-diff` repository on your RTD profile, preferably using `aiida-diff` as the project name - that's it!

## 2.5 PyPI release

Your plugin is ready to be uploaded to the Python Package Index. Just register for an account and:

```
pip install twine
python setup.py sdist bdist_wheel
twine upload dist/*
```

After this, you (and everyone else) should be able to:

```
pip install aiida-diff
```

CHAPTER 3

# aiida_diff package

## 3.1 Subpackages

### 3.1.1 aiida_diff.data package

#### Module contents

Data types provided by plugin

Register data types via the "aiida.data" entry point in setup.json.

**class** aiida_diff.data.**DiffParameters**(*dict=None*, ***kwargs*)

Bases: aiida.orm.nodes.data.dict.Dict

Command line options for diff.

This class represents a python dictionary used to pass command line options to the executable.

**__abstractmethods__ = frozenset([])**

**__init__**(*dict=None*, ***kwargs*)

Constructor for the data class

Usage: `DiffParameters(dict{'ignore-case': True})`

**Parameters**

- **parameters_dict** (*type*) – dictionary with commandline parameters
- **parameters_dict** – dict

**__module__ = 'aiida_diff.data'**

**__str__**()

String representation of node.

Append values of dictionary to usual representation. E.g.:

```
uuid: b416cbee-24e8-47a8-8c11-6d668770158b (pk: 590)
{'ignore-case': True}
```

**_abc_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache_version = 97**

**_abc_registry = <_weakrefset.WeakSet object>**

**_logger = <logging.Logger object>**

**_plugin_type_string = 'data.diff.DiffParameters.'**

**_query_type_string = 'data.diff.'**

**cmdline_params** (*file1_name*, *file2_name*)

Synthesize command line parameters.

e.g. [ '–ignore-case', 'filename1', 'filename2']

**Parameters**

- **file_name1** (*type*) – Name of first file
- **file_name1** – str
- **file_name2** (*type*) – Name of second file
- **file_name2** – str

**schema = <Schema({'ignore-case': <type 'bool'>, 'ignore-tab-expansion': <type 'bool'**

**validate** (*parameters_dict*)

Validate command line options.

Uses the voluptuous package for validation. Find out about allowed keys using:

```
print(DiffParameters).schema.schema
```

**Parameters**

- **parameters_dict** (*type*) – dictionary with commandline parameters
- **parameters_dict** – dict

**Returns** validated dictionary

### 3.1.2 aiida_diff.tests package

#### Subpackages

#### aiida_diff.tests.input_files package

#### Module contents

#### Submodules

### aiida_diff.tests.test_calculations module

Tests for calculations

aiida_diff.tests.test_calculations.**test_process**(*diff_code*)
Test running a calculation note this does not test that the expected outputs are created of output parsing

### aiida_diff.tests.test_cli module

Tests for command line interface.

**class** aiida_diff.tests.test_cli.**TestDataCli**(*methodName='runTest'*)
Bases: aiida.manage.tests.unittest_classes.PluginTestCase

Test verdi data cli plugin.

**__module__ = 'aiida_diff.tests.test_cli'**

**setUp**()
Hook method for setting up the test fixture before exercising it.

**test_data_diff_export**()
Test 'verdi data diff export'

Tests that it can be reached and that it shows the contents of the node we have set up.

**test_data_diff_list**()
Test 'verdi data diff list'

Tests that it can be reached and that it lists the node we have set up.

### Module contents

Tests for the plugin.

Includes both tests written in unittest style (test_cli.py) and tests written in pytest style (test_calculations.py).

## 3.2 Submodules

## 3.3 aiida_diff.calculations module

Calculations provided by aiida_diff.

Register calculations via the "aiida.calculations" entry point in setup.json.

**class** aiida_diff.calculations.**DiffCalculation**(**args*, ***kwargs*)
Bases: aiida.engine.processes.calcjobs.calcjob.CalcJob

AiiDA calculation plugin wrapping the diff executable.

Simple AiiDA plugin wrapper for 'diffing' two files.

**__abstractmethods__ = frozenset([])**

**__module__ = 'aiida_diff.calculations'**

**_abc_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache_version = 97**

**_abc_registry = <_weakrefset.WeakSet object>**

**classmethod define**(*spec*)
Define inputs and outputs of the calculation.

**prepare_for_submission**(*folder*)
Create input files.

**Parameters** **folder** – an *aiida.common.folders.Folder* where the plugin should temporarily place all files needed by the calculation.

**Returns** *aiida.common.datastructures.CalcInfo* instance

## 3.4 aiida_diff.cli module

Command line interface (cli) for aiida_diff.

Register new commands either via the "console_scripts" entry point or plug them directly into the 'verdi' command by using AiiDA-specific entry points like "aiida.cmdline.data" (both in the setup.json file).

## 3.5 aiida_diff.parsers module

Parsers provided by aiida_diff.

Register parsers via the "aiida.parsers" entry point in setup.json.

**class** aiida_diff.parsers.**DiffParser**(*node*)
Bases: aiida.parsers.parser.Parser

Parser class for parsing output of calculation.

**__abstractmethods__ = frozenset([])**

**__init__**(*node*)
Initialize Parser instance

Checks that the ProcessNode being passed was produced by a DiffCalculation.

**Parameters**

- **node** (*type*) – ProcessNode of calculation
- **node** – aiida.orm.ProcessNode

**__module__ = 'aiida_diff.parsers'**

**_abc_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache = <_weakrefset.WeakSet object>**

**_abc_negative_cache_version = 97**

**_abc_registry = <_weakrefset.WeakSet object>**

**parse**(***kwargs*)
Parse outputs, store results in database.

**Returns** an exit code, if parsing fails (or nothing if parsing succeeds)

## 3.6 Module contents

aiida_diff

AiiDA demo plugin that wraps the *diff* executable for computing the difference between two files.

If you use this plugin for your research, please cite the following work:

> Author Name1, Author Name2, *Paper title*, Jornal Name XXX, YYYY (Year).

If you use AiiDA for your research, please cite the following work:

> Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky, *AiiDA: automated interactive infrastructure and database for computational science*, Comp. Mat. Sci 111, 218-230 (2016); https://doi.org/10.1016/j.commatsci.2015.09.013; http://www.aiida.net.

`aiida-diff` is released under the MIT license.

CHAPTER 4

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## a

# Index

## Symbols



## A



## C



## D



## P

## S



## T



## V