AiiDA v1.0.0: the best version yet

An unapologetic sales-pitch

Sebastiaan Huber

AiiDA plugin migration workshop March 25th 2019

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts

- Daemon

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts

- Daemon
  - Multiple daemons per installation
  - Multiple workers per daemon

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts
- Daemon
  - Multiple daemons per installation
  - Multiple workers per daemon
- Event-based engine

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts
- Daemon
  - Multiple daemons per installation
  - Multiple workers per daemon
- Event-based engine
  - Clear separation between ‘processes’ and ‘nodes’
  - Scalable
  - Lightning fast

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts
- Daemon
  - Multiple daemons per installation
  - Multiple workers per daemon
- Event-based engine
  - Clear separation between ‘processes’ and ‘nodes’
  - Scalable
  - Lightning fast
- Provenance graph implementation that is **not** broken

JUST A TASTE OF SOME THE IMPROVEMENTS

- Improved command line interface
  - Homogeneous and consistent interface for `verdi`
  - Tab-completion of parameter and their values
  - Easy reuse of components in your own scripts
- Daemon
  - Multiple daemons per installation
  - Multiple workers per daemon
- Event-based engine
  - Clear separation between ‘processes’ and ‘nodes’
  - Scalable
  - Lightning fast
- Provenance graph implementation that is **not** broken
  - Determine level of granularity
  - Simplified querying

COMMAND LINE INTERFACE

IMPROVED CLI: HOMOGENEOUS AND CONSISTENT NEW INTERFACE

Reimplemented `verdi` on top of the `click` library

```
(aiida_dev) sphuber@theos:~$ verdi process --help
Usage: verdi process [OPTIONS] COMMAND [ARGS]...

  Inspect and manage processes.

Options:
  -h, --help  Show this message and exit.

Commands:
  kill    Kill running processes.
  list    Show a list of processes that are still running.
  pause   Pause running processes.
  play    Play paused processes.
  report  Show the log report for one or multiple processes.
  show    Show a summary for one or multiple processes.
  status  Print the status of the process.
  watch   Watch the state transitions for a process.
```

IMPROVED CLI: HOMOGENEOUS AND CONSISTENT NEW INTERFACE

Reimplemented `verdi` on top of the `click` library

```
(aiida_dev) sphuber@theos:~$ verdi process --help
Usage: verdi process [OPTIONS] COMMAND [ARGS]...

  Inspect and manage processes.

Options:
  -h, --help  Show this message and exit.

Commands:
  kill    Kill running processes.
  list    Show a list of processes that are still running.
  pause   Pause running processes.
  play    Play paused processes.
  report  Show the log report for one or multiple processes.
  show    Show a summary for one or multiple processes.
  status  Print the status of the process.
  watch   Watch the state transitions for a process.
```

- Consistent naming, validation of parameters
- Automatically generated help messages and documentation
- Improved tab-completion
- Simplify development through component reuse

IMPROVED CLI: TAB-COMPLETION OF COMMANDS AND PARAMETERS

All commands and sub commands are tab completed

```
(aiida_dev) sphuber@theos:~$ verdi profile
delete      list        setdefault  show
```

IMPROVED CLI: TAB-COMPLETION OF COMMANDS AND PARAMETERS

All commands and sub commands are tab completed

```
(aiida_dev) sphuber@theos:~$ verdi profile
delete      list        setdefault  show
```

But also options are completed...

```
(aiida_dev) sphuber@theos:~$ verdi -
--help      -p          --profile   --version
```

IMPROVED CLI: TAB-COMPLETION OF COMMANDS AND PARAMETERS

All commands and sub commands are tab completed

```
(aiida_dev) sphuber@theos:~$ verdi profile
delete      list        setdefault  show
```

But also options are completed...

```
(aiida_dev) sphuber@theos:~$ verdi -
--help      -p          --profile   --version
```

and not just the options themselves, but their values...

```
(aiida_dev) sphuber@theos:~$ verdi -p
django  sqla
```

IMPROVED CLI: TAB-COMPLETION OF COMMANDS AND PARAMETERS

All commands and sub commands are tab completed

```
(aiida_dev) sphuber@theos:~$ verdi profile
delete      list        setdefault  show
```

But also options are completed...

```
(aiida_dev) sphuber@theos:~$ verdi -
--help      -p          --profile   --version
```

and not just the options themselves, but their values...

```
(aiida_dev) sphuber@theos:~$ verdi -p
django  sqla
```

...same goes for arguments

```
(aiida_dev) sphuber@theos:~$ verdi profile setdefault
django  sqla
```

IMPROVED CLI: REUSABLE

CLI parameter types and pre-built options and parameters can easily be reused

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.NODE(required=True)
@decorators.with_dbenv()
def cli(node):
    click.echo('Received node<{}>'.format(node.uuid))

if __name__ == '__main__':
    cli()
```

IMPROVED CLI: REUSABLE

CLI parameter types and pre-built options and parameters can easily be reused

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.NODE(required=True)
@decorators.with_dbenv()
def cli(node):
    click.echo('Received node<{}>'.format(node.uuid))

if __name__ == '__main__':
    cli()
```

Will automatically detect if arguments are missing

```
$ ./cli.py
Usage: cli.py [OPTIONS] NODE
Try "cli.py --help" for help.

Error: Missing argument "NODE".
```

IMPROVED CLI: REUSABLE

CLI parameter types and pre-built options and parameters can easily be reused

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.NODE(required=True)
@decorators.with_dbenv()
def cli(node):
    click.echo('Received node<{}>'.format(node.uuid))

if __name__ == '__main__':
    cli()
```

Will automatically validate values passed

```
$ ./cli.py 1000
Usage: cli.py [OPTIONS] NODE

Error: Invalid value for "NODE": no Node found with ID<1000>: No result was found
```

IMPROVED CLI: REUSABLE

CLI parameter types and pre-built options and parameters can easily be reused

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.NODE(required=True)
@decorators.with_dbenv()
def cli(node):
    click.echo('Received node<{}>'.format(node.uuid))

if __name__ == '__main__':
    cli()
```

Will automatically parse values into target types

```
$ ./cli.py 100
Received node<7795f726-4f44-47be-b89f-581e0f59d967>
```

IMPROVED CLI: REUSABLE

CLI parameter types support various customizations

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import options, types
from aiida.cmdline.utils import decorators

@click.command()
@options.CODE(required=True, type=types.CodeParamType(entry_point='arithmetic.add'),
    help='Code configured to run `arithmetic.add` calculation.')
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code<{}>'.format(code.full_label))

if __name__ == '__main__':
    cli()
```

IMPROVED CLI: REUSABLE

CLI parameter types support various customizations

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import options, types
from aiida.cmdline.utils import decorators

@click.command()
@options.CODE(required=True, type=types.CodeParamType(entry_point='arithmetic.add'),
    help='Code configured to run `arithmetic.add` calculation.')
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code<{}>'.format(code.full_label))

if __name__ == '__main__':
    cli()
```

Will automatically detect if arguments are missing

```
$ ./cli.py
Usage: cli.py [OPTIONS]
Try "cli.py --help" for help.

Error: Missing option "-X" / "--code".
```

IMPROVED CLI: REUSABLE

CLI parameter types support various customizations

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import options, types
from aiida.cmdline.utils import decorators

@click.command()
@options.CODE(required=True, type=types.CodeParamType(entry_point='arithmetic.add'),
    help='Code configured to run `arithmetic.add` calculation.')
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code<{}>'.format(code.full_label))

if __name__ == '__main__':
    cli()
```

Will automatically validate values passed

```
$ ./cli.py doubler@localhost
Usage: cli.py [OPTIONS]

Error: Invalid value for "-X" / "--code": the retrieved Code<1> has
       plugin type "templatereplacer" while "arithmetic.add" is required
```

IMPROVED CLI: REUSABLE

CLI parameter types support various customizations

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import options, types
from aiida.cmdline.utils import decorators

@click.command()
@options.CODE(required=True, type=types.CodeParamType(entry_point='arithmetic.add'),
    help='Code configured to run `arithmetic.add` calculation.')
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code<{}>'.format(code.full_label))

if __name__ == '__main__':
    cli()
```

Will automatically parse values into target types

```
$ ./cli.py add@localhost
Received code<add@localhost>
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Using the `ID`

```
$ ./cli.py 15
Received code: Remote code 'add' on localhost, pk: 15, uuid: 222509c1-9fa0-479d-95f2-86d839392696
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Using the `UUID`

```
$ ./cli.py 222509c1-9fa0-479d-95f2-86d839392696
Received code: Remote code 'add' on localhost, pk: 15, uuid: 222509c1-9fa0-479d-95f2-86d839392696
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Partial `UUID` is supported, but can lead to ambiguity with an `ID`

```
$ ./cli.py 2225
Usage: cli.py [OPTIONS] CODE

Error: Invalid value for "CODE": no Code found with ID<add>: No result was found
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Include first hyphen to break ambiguity

```
$ ./cli.py 222509c1-
Received code: Remote code 'add' on localhost, pk: 15, uuid: 222509c1-9fa0-479d-95f2-86d839392696
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Labels, when hexadecimal, will be potentially incorrectly intepreted as a `UUID`

```
$ ./cli.py add
Usage: cli.py [OPTIONS] CODE

Error: Invalid value for "CODE": no Code found with UUID<add>: No result was found
```

IMPROVED CLI: CONSISTENCY

All ORM 'identifiers' support `ID`, `UUID` and `LABEL`
Type interpreted consecutively in that order until successful or all fail

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import click

from aiida.cmdline.params import arguments
from aiida.cmdline.utils import decorators

@click.command()
@arguments.CODE(required=True)
@decorators.with_dbenv()
def cli(code):
    click.echo('Received code: {}'.format(code))

if __name__ == '__main__':
    cli()
```

Exclamation mark serves as ambiguity breaker

```
$ ./cli.py add!
Received code: Remote code 'add' on localhost, pk: 15, uuid: 222509c1-9fa0-479d-95f2-86d83939269
```

DAEMON

DAEMON: MULTIPLE DAEMONS

Originally, each installation had a single daemon

```
(aiida_dev) sphuber@theos:~$ verdi profile list
Info: configuration folder: /home/sphuber/.virtualenvs/aiida_dev/.aiida
* django
  sqla
```

DAEMON: MULTIPLE DAEMONS

Originally, each installation had a single daemon

```
(aiida_dev) sphuber@theos:~$ verdi profile list
Info: configuration folder: /home/sphuber/.virtualenvs/aiida_dev/.aiida
* django
  sqla
```

Now, a single installation has one daemon per profile

```
(aiida_dev) sphuber@theos:~$ verdi daemon status
Profile: django
Daemon is running as PID 2809 since 2019-03-15 16:27:52
Active workers [1]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 2814    0.137        0  2019-03-15 16:27:52
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
```

DAEMON: MULTIPLE DAEMONS

Originally, each installation had a single daemon

```
(aiida_dev) sphuber@theos:~$ verdi profile list
Info: configuration folder: /home/sphuber/.virtualenvs/aiida_dev/.aiida
* django
  sqla
```

Both can run at the same time in parallel

```
(aiida_dev) sphuber@theos:~$ verdi daemon status --all
Profile: sqla
Daemon is running as PID 7284 since 2019-03-21 18:11:31
Active workers [1]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 7288    0.121        0  2019-03-21 18:11:31
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
Profile: django
Daemon is running as PID 2809 since 2019-03-15 16:27:52
Active workers [1]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 2814    0.137        0  2019-03-15 16:27:52
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
```

DAEMON: MULTIPLE WORKERS

By default each daemon has a single worker

```
(aiida_dev) sphuber@theos:~$ verdi daemon status
Profile: django
Daemon is running as PID 2809 since 2019-03-15 16:27:52
Active workers [1]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 2814    0.137        0  2019-03-15 16:27:52
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
```

DAEMON: MULTIPLE WORKERS

By default each daemon has a single worker

```
(aiida_dev) sphuber@theos:~$ verdi daemon status
Profile: django
Daemon is running as PID 2809 since 2019-03-15 16:27:52
Active workers [1]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 2814    0.137        0  2019-03-15 16:27:52
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
```

Can easily add and remove workers that each work independently

```
(aiida_dev) sphuber@theos:~$ verdi daemon incr 5
OK
(aiida_dev) sphuber@theos:~$ verdi daemon status
Profile: django
Daemon is running as PID 2809 since 2019-03-15 16:27:52
Active workers [6]:
  PID    MEM %    CPU %  started
-----  -------  -------  -------------------
 2814    0.137        0  2019-03-15 16:27:52
10602    0.12         0  2019-03-21 18:21:40
10603    0.12         0  2019-03-21 18:21:40
10601    0.121        0  2019-03-21 18:21:40
10604    0.121        0  2019-03-21 18:21:40
10605    0.121        0  2019-03-21 18:21:41
Use verdi daemon [incr | decr] [num] to increase / decrease the amount of workers
```

ENGINE

ENGINE: EVERYTHING IS A PROCESS

New processes to define calculations and workflows

Process class	Node class	Used for
`CalcJob`	`CalcJobNode`	Calculations performed by external codes
`WorkChain`	`WorkChainNode`	Workflows that run multiple sub processes
`FunctionProcess`	`CalcFunctionNode`	Python functions decorated with the `calcfunction` decorator
`FunctionProcess`	`WorkFunctionNode`	Python functions decorated with the `workfunction` decorator

ENGINE: EVERYTHING IS A PROCESS

New processes to define calculations and workflows

Process class	Node class	Used for
`CalcJob`	`CalcJobNode`	Calculations performed by external codes
`WorkChain`	`WorkChainNode`	Workflows that run multiple sub processes
`FunctionProcess`	`CalcFunctionNode`	Python functions decorated with the `calcfunction` decorator
`FunctionProcess`	`WorkFunctionNode`	Python functions decorated with the `workfunction` decorator

PROCESS STATE

Active	Terminated
Created	Killed
Running	Excepted
Waiting	Finished

ENGINE: EVERYTHING IS A PROCESS

New processes to define calculations and workflows

Process class	Node class	Used for
`CalcJob`	`CalcJobNode`	Calculations performed by external codes
`WorkChain`	`WorkChainNode`	Workflows that run multiple sub processes
`FunctionProcess`	`CalcFunctionNode`	Python functions decorated with the `calcfunction` decorator
`FunctionProcess`	`WorkFunctionNode`	Python functions decorated with the `workfunction` decorator

PROCESS STATE

Active	Terminated
Created	Killed
Running	Excepted
Waiting	Finished

PROCESS NODE ATTRIBUTES

Property	Meaning
`process_state`	Returns the current process state
`exit_status`	Returns the exit status, or None if not set
`exit_message`	Returns the exit message, or None if not set
`is_terminated`	Returns `True` if the process was either `Killed`, `Excepted` or `Finished`
`is_killed`	Returns `True` if the process is `Killed`
`is_excepted`	Returns `True` if the process is `Excepted`
`is_finished`	Returns `True` if the process is `Finished`
`is_finished_ok`	Returns `True` if the process is `Finished` and the `exit_status` is equal to zero
`is_failed`	Returns `True` if the process is `Finished` and the `exit_status` is non-zero

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

Submit to the daemon

```
from aiida import orm
from aiida.engine import submit

ArithmeticAddCalculation = CalculationFactory('arithmetic.add')
node = submit(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
```

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

Run blockingly in local interpreter

```
from aiida import orm
from aiida.engine import run

ArithmeticAddCalculation = CalculationFactory('arithmetic.add')
result = run(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
```

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

Run variants to get the process node or pk in addition to the result

```
from aiida import orm
from aiida.engine import run_get_node, run_get_pk

ArithmeticAddCalculation = CalculationFactory('arithmetic.add')
result, node = run_get_node(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
result, pk = run_get_pk(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
```

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

Variants are available as attributes on `run` launcher requiring only single import

```
from aiida import orm
from aiida.engine import run

ArithmeticAddCalculation = CalculationFactory('arithmetic.add')
result = run(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
result, node = run.get_node(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
result, pk = run.get_pk(ArithmeticAddCalculation, x=orm.Int(1), y=orm.Int(2))
```

ENGINE: EVERYTHING IS A PROCESS

Four processes launchers with identical interface

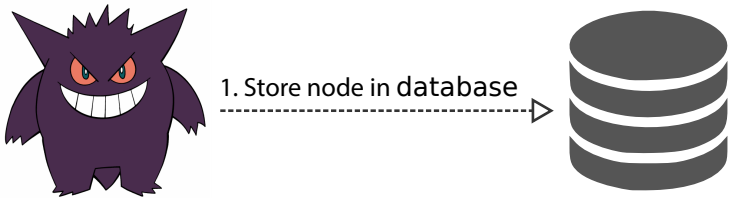
`run`	Run blockingly and return result
`run_get_node`	Run blockingly and return result + node
`run_get_pk`	Run blockingly and return result + pk
`submit`	Submit to daemon and return node

Syntactic keyword expansion for big input dictionaries

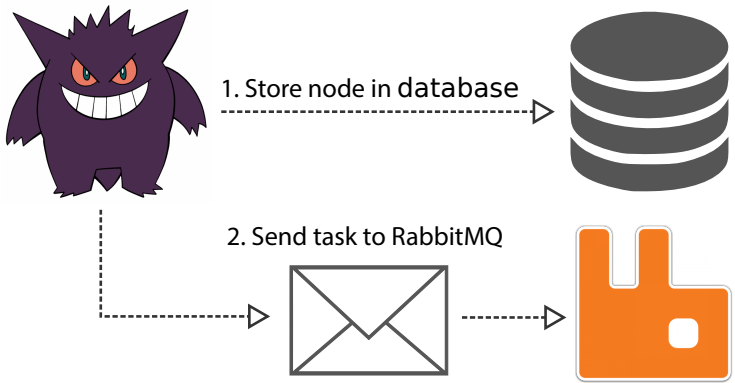
```
from aiida import orm
from aiida.engine import submit

ArithmeticAddCalculation = CalculationFactory('arithmetic.add')
inputs = {
    'x': orm.Int(1),
    'y': orm.Int(2)
}
node = submit(ArithmeticAddCalculation, **inputs)
```

What happens when we submit a process?

What happens when we submit a process?

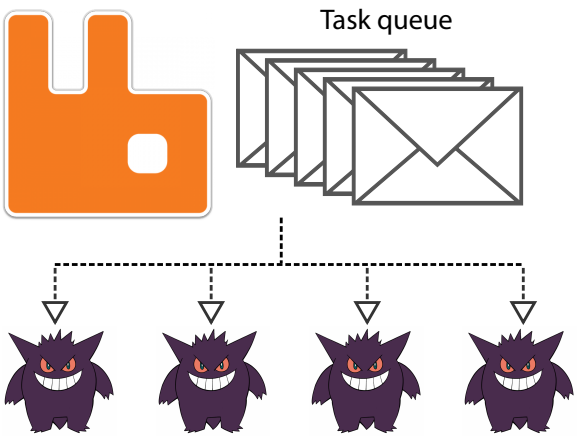


What happens when we submit a process?



What happens with those tasks?

What happens with those tasks?



ENGINE: RELYING ON A RESILIENT AND ROBUST RABBIT

The promise of RabbitMQ

- All tasks are persisted to disk
- Each task is guaranteed to be delivered
- Each task is guaranteed to be sent to only one listener at a time
- Each task is guaranteed to be completed

ENGINE: RELYING ON A RESILIENT AND ROBUST RABBIT

The promise of RabbitMQ

- All tasks are persisted to disk
- Each task is guaranteed to be delivered
- Each task is guaranteed to be sent to only one listener at a time
- Each task is guaranteed to be completed

This ensures that:

1. We can run multiple processes in parallel independently

2. Each launched task or "process" will eventually be completed

No matter what happens

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process list`: list active and terminated processes

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process list -l 10
     PK  Created    State      Process label     Process status
-------  ---------  ---------  ---------------  --------------------------------
1285287  2D ago     ▸ Waiting  PwRelaxWorkChain
1285535  2D ago     ▸ Waiting  PwRelaxWorkChain
1286517  2D ago     ▸ Waiting  PwRelaxWorkChain
1286913  2D ago     ▸ Waiting  PwRelaxWorkChain
1287288  2D ago     ▸ Waiting  PwBaseWorkChain
1287486  2D ago     ▸ Waiting  PwRelaxWorkChain
1287517  2D ago     ▸ Waiting  PwBaseWorkChain
1287937  2D ago     ▸ Waiting  PwRelaxWorkChain
1289927  2D ago     ▸ Waiting  PwCalculation     Waiting for transport task: update
1291148  2D ago     ▸ Waiting  PwBaseWorkChain

Total results: 10

Info: last time an entry changed state: 43s ago (at 11:09:50 on 2019-03-22)
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process status`: tree representation of call stack

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process status 1338418
PwRelaxWorkChain <pk=1338418> [ProcessState.FINISHED] [3:results]
    ├── PwBaseWorkChain <pk=1338525> [ProcessState.FINISHED] [4:results]
    │   ├── create_kpoints_from_distance <pk=1338529> [ProcessState.FINISHED]
    │   └── CalcJobNode <pk=1338569> [FINISHED]
    ├── PwBaseWorkChain <pk=1341443> [ProcessState.FINISHED] [4:results]
    │   ├── create_kpoints_from_distance <pk=1341445> [ProcessState.FINISHED]
    │   └── CalcJobNode <pk=1341463> [FINISHED]
    └── PwBaseWorkChain <pk=1341701> [ProcessState.FINISHED] [4:results]
        ├── create_kpoints_from_distance <pk=1341703> [ProcessState.FINISHED]
        └── CalcJobNode <pk=1341730> [FINISHED]
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process report`: complete report of log messages and scheduler stdout/stderr

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process report 1338418
2019-03-22 05:26:39 [553130 | REPORT]: [1338418|PwRelaxWorkChain|run_relax]: launching PwBaseWorkChain<1338525>
2019-03-22 05:28:02 [553147 | REPORT]:   [1338525|PwBaseWorkChain|run_calculation]: launching PwCalculation<1338569> iteration #1
2019-03-22 07:45:52 [554706 | REPORT]:   [1338525|PwBaseWorkChain|inspect_calculation]: PwCalculation<1338569> completed successfully
2019-03-22 07:45:52 [554707 | REPORT]:   [1338525|PwBaseWorkChain|results]: workchain completed after 1 iterations
2019-03-22 07:45:52 [554708 | REPORT]:   [1338525|PwBaseWorkChain|on_terminated]: remote folders will not be cleaned
2019-03-22 07:45:53 [554709 | REPORT]: [1338418|PwRelaxWorkChain|inspect_relax]: after iteration 1 cell volume of relaxed structure is 427.27585978
2019-03-22 07:45:54 [554710 | REPORT]: [1338418|PwRelaxWorkChain|run_relax]: launching PwBaseWorkChain<1341443>
2019-03-22 07:47:21 [554713 | REPORT]:   [1341443|PwBaseWorkChain|run_calculation]: launching PwCalculation<1341463> iteration #1
2019-03-22 08:00:47 [554854 | REPORT]:   [1341443|PwBaseWorkChain|inspect_calculation]: PwCalculation<1341463> completed successfully
2019-03-22 08:00:47 [554855 | REPORT]:   [1341443|PwBaseWorkChain|results]: workchain completed after 1 iterations
2019-03-22 08:00:47 [554856 | REPORT]:   [1341443|PwBaseWorkChain|on_terminated]: remote folders will not be cleaned
2019-03-22 08:00:48 [554857 | REPORT]: [1338418|PwRelaxWorkChain|inspect_relax]: after iteration 2 cell volume of relaxed structure is 427.275859777
2019-03-22 08:00:48 [554858 | REPORT]: [1338418|PwRelaxWorkChain|inspect_relax]: relative cell volume difference 5.54830030607e-12 smaller than convergence threshold 0.01
2019-03-22 08:00:49 [554859 | REPORT]: [1338418|PwRelaxWorkChain|run_final_scf]: launching PwBaseWorkChain<1341701> for final scf
2019-03-22 08:02:13 [554881 | REPORT]:   [1341701|PwBaseWorkChain|run_calculation]: launching PwCalculation<1341730> iteration #1
2019-03-22 08:11:19 [554931 | REPORT]:   [1341701|PwBaseWorkChain|inspect_calculation]: PwCalculation<1341730> completed successfully
2019-03-22 08:11:19 [554932 | REPORT]:   [1341701|PwBaseWorkChain|results]: workchain completed after 1 iterations
2019-03-22 08:11:19 [554933 | REPORT]:   [1341701|PwBaseWorkChain|on_terminated]: remote folders will not be cleaned
2019-03-22 08:11:20 [554934 | REPORT]: [1338418|PwRelaxWorkChain|results]: workchain completed after 2 iterations
2019-03-22 08:11:30 [554935 | REPORT]: [1338418|PwRelaxWorkChain|on_terminated]: cleaned remote folders of calculations: 1341730 1341463 1338569
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process pause`: pause an active process

```
(aiida_dev) sphuber@theos:~$ verdi process pause 804
Success: scheduled pause Process<804>
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process pause`: pause an active process

```
(aiida_dev) sphuber@theos:~$ verdi process pause 804
Success: scheduled pause Process<804>
```

`verdi process play`: resume a paused process

```
(aiida_dev) sphuber@theos:~$ verdi process play 812
Success: scheduled play Process<812>
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process pause`: pause an active process

```
(aiida_dev) sphuber@theos:~$ verdi process pause 804
Success: scheduled pause Process<804>
```

`verdi process play`: resume a paused process

```
(aiida_dev) sphuber@theos:~$ verdi process play 812
Success: scheduled play Process<812>
```

`verdi process kill`: kill an active process

```
(aiida_dev) sphuber@theos:~$ verdi process kill 820
Success: scheduled kill Process<820>
```

ENGINE: EVERYTHING IS A PROCESS

`verdi process`: your one-stop-shop for inspecting and interacting with processes

`verdi process pause`: pause an active process

```
(aiida_dev) sphuber@theos:~$ verdi process pause 804
Success: scheduled pause Process<804>
```

`verdi process play`: resume a paused process

```
(aiida_dev) sphuber@theos:~$ verdi process play 812
Success: scheduled play Process<812>
```

`verdi process kill`: kill an active process

```
(aiida_dev) sphuber@theos:~$ verdi process kill 820
Success: scheduled kill Process<820>
```

`verdi process pause/play/kill`: fails if process is already terminated

```
(aiida_dev) sphuber@theos:~$ verdi process kill 812
Error: Process<812> is already terminated
```

ENGINE: ROBUSTNESS

Automatic retry for transport tasks with exponential backoff

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process list --paused
     PK  Created    State      Process label    Process status
-------  ---------  ---------  ---------------  ------------------------------------------------------------------------------------------
1343914  5h ago     ⏸ Waiting  PwCalculation    Pausing after failed transport task: retrieve_calculation failed 5 times consecutively

Total results: 1

Info: last time an entry changed state: 21s ago (at 16:10:08 on 2019-03-22)
```

ENGINE: ROBUSTNESS

Automatic retry for transport tasks with exponential backoff

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process list --paused
     PK  Created    State      Process label    Process status
-------  ---------  ---------  ---------------  ------------------------------------------------------------------------------------------
1343914  5h ago     ⏸ Waiting  PwCalculation    Pausing after failed transport task: retrieve_calculation failed 5 times consecutively

Total results: 1

Info: last time an entry changed state: 21s ago (at 16:10:08 on 2019-03-22)
```

Rate limited connections to remote clusters per daemon worker

```
(aiida_3dd) aiida@theossrv5:~$ verdi computer configure show localhost
* safe_interval  0
```

ENGINE: ROBUSTNESS

Automatic retry for transport tasks with exponential backoff

```
(aiida_3dd) aiida@theossrv5:~/code/aiida/env/3dd$ verdi process list --paused
     PK  Created    State      Process label    Process status
-------  ---------  ---------  ---------------  ------------------------------------------------------------------------------------------
1343914  5h ago     ⏸ Waiting  PwCalculation    Pausing after failed transport task: retrieve_calculation failed 5 times consecutively

Total results: 1

Info: last time an entry changed state: 21s ago (at 16:10:08 on 2019-03-22)
```

Rate limited connections to remote clusters per daemon worker

```
(aiida_3dd) aiida@theossrv5:~$ verdi computer configure show localhost
* safe_interval  0
```

Rate limited scheduler state queries per daemon worker

```
(aiida_3dd) aiida@theossrv5:~$ verdi shell

In [1]: from aiida.orm import Computer

In [2]: computer = Computer.get(name='localhost')

In [3]: computer.get_minimum_job_poll_interval()
Out[3]: 30

In [4]: computer.set_minimum_job_poll_interval(60)

In [5]: computer.get_minimum_job_poll_interval()
Out[5]: 60
```

PROVENANCE REDESIGN

Two clearly distinct types of processes

CALCULATIONS

Can *create* new data

WORKFLOWS

Can *call* other processes

Can *return* existing data

PROVENANCE REDESIGN: CALCULATION FUNCTIONS

To transform simple function into process

```
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

PROVENANCE REDESIGN: CALCULATION FUNCTIONS

To transform simple function into process

```
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

Just apply the `calcfunction` decorator...

```
from aiida.engine import calcfunction

@calcfunction
def add(x, y):
    return x + y

@calcfunction
def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

PROVENANCE REDESIGN: CALCULATION FUNCTIONS

To transform simple function into process

```
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

... and pass storable data types when calling

```
from aiida.engine import calcfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return x + y

@calcfunction
def multiply(x, y):
    return x * y

result = multiply(add(Int(1), Int(2)), Int(3))
```

Just apply the `calcfunction` decorator...

```
from aiida.engine import calcfunction

@calcfunction
def add(x, y):
    return x + y

@calcfunction
def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

PROVENANCE REDESIGN: CALCULATION FUNCTIONS

To transform simple function into process

```python
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

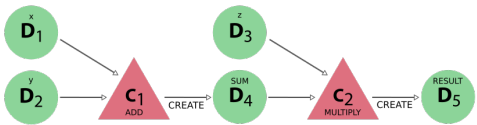
... and pass storable data types when calling

```python
from aiida.engine import calcfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return x + y

@calcfunction
def multiply(x, y):
    return x * y

result = multiply(add(Int(1), Int(2)), Int(3))
```

Just apply the `calcfunction` decorator...

```python
from aiida.engine import calcfunction

@calcfunction
def add(x, y):
    return x + y

@calcfunction
def multiply(x, y):
    return x * y

result = multiply(add(1, 2), 3)
```

Provenance is automatically stored in the graph



PROVENANCE REDESIGN: WORK FUNCTIONS

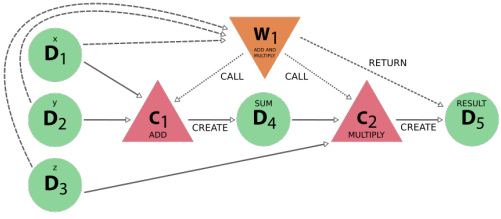
Work function can be used to store *logical* provenance

```
from aiida.engine import calcfunction, workfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

@workfunction
def add_and_multiply(x, y, z):
    sum = add(x, y)
    product = multiply(sum, z)
    return product

result = add_and_multiply(Int(1), Int(2), Int(3))
```

PROVENANCE REDESIGN: WORK FUNCTIONS

Work function can be used to store *logical* provenance

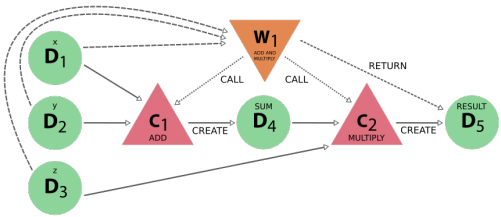
```python
from aiida.engine import calcfunction, workfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

@workfunction
def add_and_multiply(x, y, z):
    sum = add(x, y)
    product = multiply(sum, z)
    return product

result = add_and_multiply(Int(1), Int(2), Int(3))
```



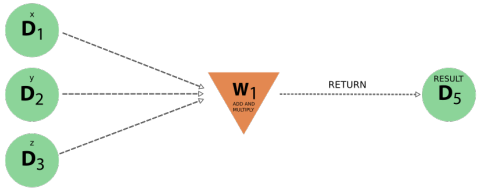
PROVENANCE REDESIGN: WORK FUNCTIONS

Work function can be used to store *logical* provenance

```python
from aiida.engine import calcfunction, workfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

@workfunction
def add_and_multiply(x, y, z):
    sum = add(x, y)
    product = multiply(sum, z)
    return product

result = add_and_multiply(Int(1), Int(2), Int(3))
```



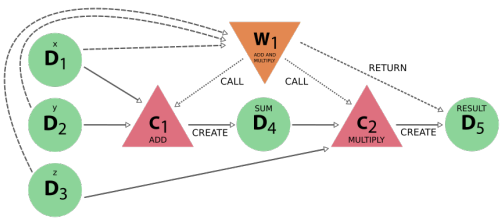
Logical provenance allows to 'hide' complexity



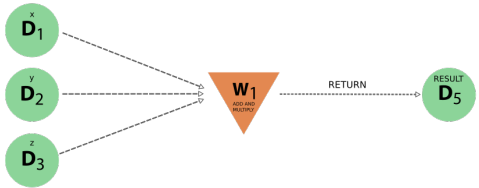
PROVENANCE REDESIGN: WORK FUNCTIONS

Work function can be used to store *logical* provenance

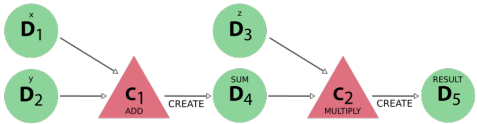
```python
from aiida.engine import calcfunction, workfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

@workfunction
def add_and_multiply(x, y, z):
    sum = add(x, y)
    product = multiply(sum, z)
    return product

result = add_and_multiply(Int(1), Int(2), Int(3))
```



Logical provenance allows to 'hide' complexity



Or by ignoring it, retrieve the original data provenance



PROVENANCE REDESIGN: WORK CHAINS

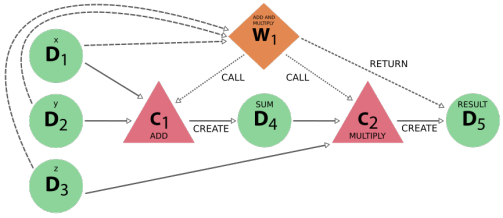
Work chains achieve the same but save progress in between steps

```python
from aiida.engine import WorkChain, calcfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

class AddAndMultiplyWorkChain(WorkChain):

    @classmethod
    def define(cls, spec):
        super(AddAndMultiplyWorkChain, cls).define(spec)
        spec.input('x')
        spec.input('y')
        spec.input('z')
        spec.outline(
            cls.add,
            cls.multiply,
            cls.results,
        )
        spec.output('result')

    def add(self):
        self.ctx.sum = add(self.inputs.x, self.inputs.y)

    def multiply(self):
        self.ctx.product = multiply(self.ctx.sum, self.inputs.z)

    def results(self):
        self.out('result', self.ctx.product)
```

PROVENANCE REDESIGN: WORK CHAINS

Work chains achieve the same but save progress in between steps

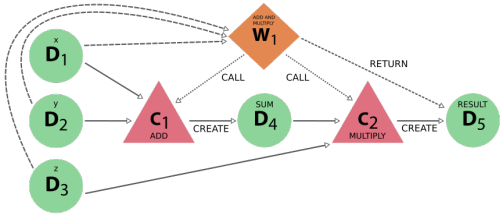
```python
from aiida.engine import WorkChain, calcfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

class AddAndMultiplyWorkChain(WorkChain):

    @classmethod
    def define(cls, spec):
        super(AddAndMultiplyWorkChain, cls).define(spec)
        spec.input('x')
        spec.input('y')
        spec.input('z')
        spec.outline(
            cls.add,
            cls.multiply,
            cls.results,
        )
        spec.output('result')

    def add(self):
        self.ctx.sum = add(self.inputs.x, self.inputs.y)

    def multiply(self):
        self.ctx.product = multiply(self.ctx.sum, self.inputs.z)

    def results(self):
        self.out('result', self.ctx.product)
```



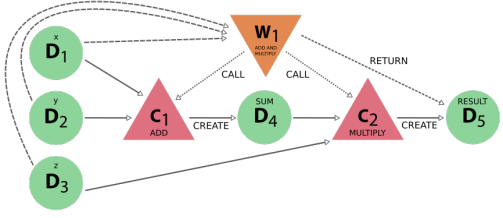
PROVENANCE REDESIGN: WORK CHAINS

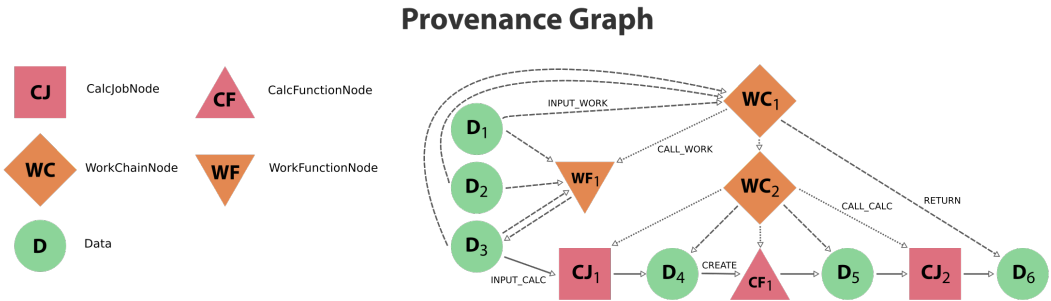
Work chains achieve the same but save progress in between steps

```python
from aiida.engine import WorkChain, calcfunction
from aiida.orm import Int

@calcfunction
def add(x, y):
    return Int(x + y)

@calcfunction
def multiply(x, y):
    return Int(x * y)

class AddAndMultiplyWorkChain(WorkChain):

    @classmethod
    def define(cls, spec):
        super(AddAndMultiplyWorkChain, cls).define(spec)
        spec.input('x')
        spec.input('y')
        spec.input('z')
        spec.outline(
            cls.add,
            cls.multiply,
            cls.results,
        )
        spec.output('result')

    def add(self):
        self.ctx.sum = add(self.inputs.x, self.inputs.y)

    def multiply(self):
        self.ctx.product = multiply(self.ctx.sum, self.inputs.z)

    def results(self):
        self.out('result', self.ctx.product)
```

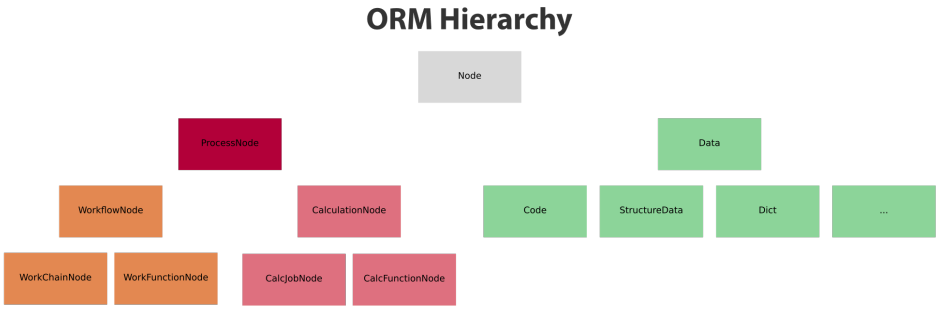


Only difference with work function solution is node type



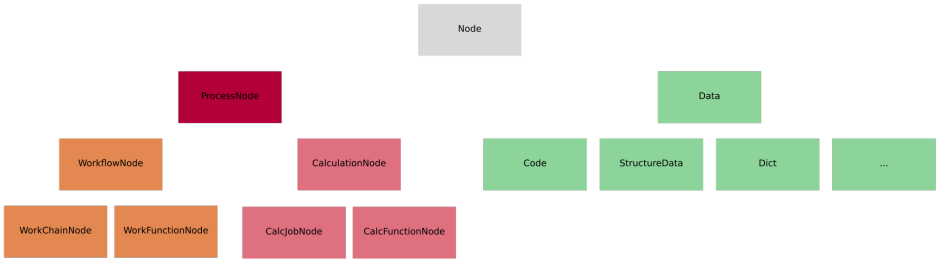

Provenance Graph
CJ
CalcJobNode
CF
CalcFunctionNode
WC
WorkChainNode
WF
WorkFunctionNode
D
Data
INPUT_WORK
CALL_WORK
RETURN
CALL_CALC
CREATE
INPUT_CALC
WC1
D1
D2
D3
WF1
WC2
CJ1
D4
CF1
D5
CJ2
D6

PROVENANCE REDESIGN: THE RULES

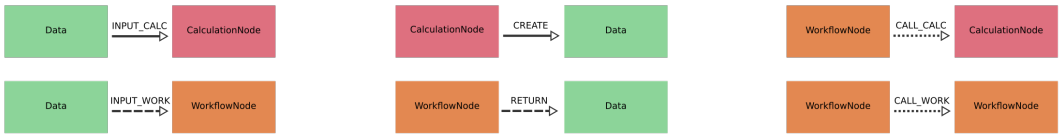


PROVENANCE REDESIGN: THE RULES

ORM Hierarchy



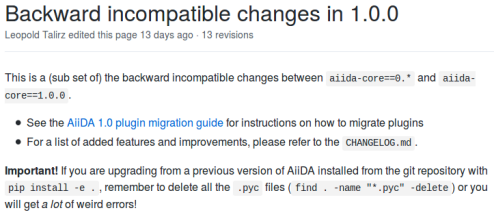
Link Hierarchy



WHAT CHANGED?

WHAT CHANGED?

Summary of backwards-compatible changes curated and maintained[1]



Backward incompatible changes in 1.0.0

Leopold Talirz edited this page 13 days ago · 13 revisions

This is a (sub set of) the backward incompatible changes between `aiida-core==0.*` and `aiida-core==1.0.0`.

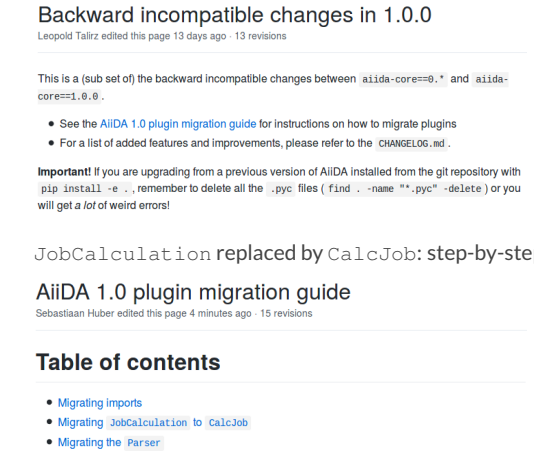
- See the AiiDA 1.0 plugin migration guide for instructions on how to migrate plugins
- For a list of added features and improvements, please refer to the `CHANGELOG.md`.

Important! If you are upgrading from a previous version of AiiDA installed from the git repository with `pip install -e .`, remember to delete all the `.pyc` files (`find . -name "*.pyc" -delete`) or you will get *a lot* of weird errors!

[1] https://github.com/aiidateam/aiida_core/wiki/Backward-incompatible-changes-in-1.0.0
[2] https://github.com/aiidateam/aiida_core/wiki/AiiDA-1.0-plugin-migration-guide

WHAT CHANGED?

Summary of backwards-compatible changes curated and maintained[1]



Backward incompatible changes in 1.0.0

Leopold Talirz edited this page 13 days ago · 13 revisions

This is a (sub set of) the backward incompatible changes between `aiida-core==0.*` and `aiida-core==1.0.0`.

- See the AiiDA 1.0 plugin migration guide for instructions on how to migrate plugins
- For a list of added features and improvements, please refer to the `CHANGELOG.md`.

Important! If you are upgrading from a previous version of AiiDA installed from the git repository with `pip install -e .`, remember to delete all the `.pyc` files (`find . -name "*.pyc" -delete`) or you will get *a lot* of weird errors!

`JobCalculation` replaced by `CalcJob`: step-by-step guide on wiki[2]

AiiDA 1.0 plugin migration guide

Sebastiaan Huber edited this page 4 minutes ago · 15 revisions

Table of contents

- Migrating imports
- Migrating `JobCalculation` to `CalcJob`
- Migrating the `Parser`

[1] https://github.com/aiidateam/aiida_core/wiki/Backward-incompatible-changes-in-1.0.0
[2] https://github.com/aiidateam/aiida_core/wiki/AiiDA-1.0-plugin-migration-guide

WHAT CHANGED: MODULE HIERARCHY AND API GUARANTEES

Significant restructuring and renaming of second-level modules

backends	Release `v1.0.0b1` (#2593)	21 hours ago
calculations	Release `v1.0.0b1` (#2593)	21 hours ago
cmdline	Merge branch 'master' of github.com:aiidateam/aiida_core into merge_m…	a day ago
common	Release `v1.0.0b1` (#2593)	21 hours ago
engine	Release `v1.0.0b1` (#2593)	21 hours ago
manage	Release `v1.0.0b1` (#2593)	21 hours ago
orm	Release `v1.0.0b1` (#2593)	21 hours ago
parsers	Release `v1.0.0b1` (#2593)	21 hours ago
plugins	Release `v1.0.0b1` (#2593)	21 hours ago
restapi	Merge branch 'master' of github.com:aiidateam/aiida_core into merge_m…	a day ago
schedulers	Release `v1.0.0b1` (#2593)	21 hours ago
sphinxext	Simplify imports from the `aiida.orm` (#2534)	15 days ago
tools	Release `v1.0.0b1` (#2593)	21 hours ago
transports	Release `v1.0.0b1` (#2593)	21 hours ago
__init__.py	Release `v1.0.0b1` (#2593)	21 hours ago
settings.py	Formalizing importable resources on second level modules (#2528)	16 days ago

WHAT CHANGED: MODULE HIERARCHY AND API GUARANTEES

Significant restructuring and renaming of second-level modules

backends	Release `v1.0.0b1` (#2593)	21 hours ago
calculations	Release `v1.0.0b1` (#2593)	21 hours ago
cmdline	Merge branch 'master' of github.com:aiidateam/aiida_core into merge_m…	a day ago
common	Release `v1.0.0b1` (#2593)	21 hours ago
engine	Release `v1.0.0b1` (#2593)	21 hours ago
manage	Release `v1.0.0b1` (#2593)	21 hours ago
orm	Release `v1.0.0b1` (#2593)	21 hours ago
parsers	Release `v1.0.0b1` (#2593)	21 hours ago
plugins	Release `v1.0.0b1` (#2593)	21 hours ago
restapi	Merge branch 'master' of github.com:aiidateam/aiida_core into merge_m…	a day ago
schedulers	Release `v1.0.0b1` (#2593)	21 hours ago
sphinxext	Simplify imports from the `aiida.orm` (#2534)	15 days ago
tools	Release `v1.0.0b1` (#2593)	21 hours ago
transports	Release `v1.0.0b1` (#2593)	21 hours ago
__init__.py	Release `v1.0.0b1` (#2593)	21 hours ago
settings.py	Formalizing importable resources on second level modules (#2528)	16 days ago

- `aiida.utils` merged into `aiida.common`
- `aiida.scheduler` → `aiida.schedulers`
- `aiida.transport` → `aiida.transports`
- `aiida.work` → `aiida.engine`

WHAT CHANGED: MODULE HIERARCHY AND API GUARANTEES

You should only ever (have to) import directly from a second-level package

```
from aiida.orm import Data
data = Data()
```

[3] https://github.com/aiidateam/aiida_core/wiki/AiiDA-public-modules,-classes-and-functions

WHAT CHANGED: MODULE HIERARCHY AND API GUARANTEES

You should only ever (have to) import directly from a second-level package

```
from aiida.orm import Data
data = Data()
```

```
from aiida import orm
data = orm.Data()
```

[3] https://github.com/aiidateam/aiida_core/wiki/AiiDA-public-modules,-classes-and-functions

WHAT CHANGED: MODULE HIERARCHY AND API GUARANTEES

You should only ever (have to) import directly from a second-level package

```python
from aiida.orm import Data
data = Data()
```

```python
from aiida import orm
data = orm.Data()
```

An explicit list is being maintained on the Github wiki[3]

AiiDA public modules, classes and functions

Sebastiaan Huber edited this page 8 days ago · 1 revision

The main package of `aiida-core` is called `aiida`, which contains various sub-packages that we refer to as "second-level packages". These second level packages can have further nested hierarchies. Certain resources (modules, classes, functions and variables) within these packages are intended for internal use, whereas others *are meant* to be used by users of the `aiida-core` package. To make it easier for users to locate these resources that are intended for external use, as well as to distinguish them from internal resources *that are not supposed to be used*, they are exposed directly on the second-level package. This means that any resource that can be directly imported from a second-level package, *is intended for external use*. Below we provide a list of the resources per second-level package that are exposed in this way. If a module is mentioned, then all the resources defined in its `__all__` are included.

[3] https://github.com/aiidateam/aiida_core/wiki/AiiDA-public-modules,-classes-and-functions

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

Retrieving an instance from the 'collection' through the `objects` property

```
from aiida import orm
computer = orm.Computer.objects.get(name='localhost')
```

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

Retrieving an instance from the 'collection' through the `objects` property

```
from aiida import orm
computer = orm.Computer.objects.get(name='localhost')
```

Shortcut directly on the class

```
from aiida import orm
computer = orm.Computer.get(name='localhost')
```

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

Retrieving an instance from the 'collection' through the `objects` property

```
from aiida import orm
computer = orm.Computer.objects.get(name='localhost')
```

Shortcut directly on the class

```
from aiida import orm
computer = orm.Computer.get(name='localhost')
```

Getting all instance from the collection

```
from aiida import orm
computers = orm.Computer.objects.all()
```

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

Retrieving an instance from the 'collection' through the `objects` property

```
from aiida import orm
computer = orm.Computer.objects.get(name='localhost')
```

Shortcut directly on the class

```
from aiida import orm
computer = orm.Computer.get(name='localhost')
```

Getting all instance from the collection

```
from aiida import orm
computers = orm.Computer.objects.all()
```

Finding multiple instances with filters

```
from aiida import orm
computers = orm.Computer.objects.find(filters={'hostname': 'localhost'}, order_by='name')
```

WHAT CHANGED: ORM

Constructing new instances

```
from aiida import orm
computer = orm.Computer(name='localhost', hostname='localhost')
```

Retrieving an instance from the 'collection' through the `objects` property

```
from aiida import orm
computer = orm.Computer.objects.get(name='localhost')
```

Shortcut directly on the class

```
from aiida import orm
computer = orm.Computer.get(name='localhost')
```

Getting all instance from the collection

```
from aiida import orm
computers = orm.Computer.objects.all()
```

Finding multiple instances with filters

```
from aiida import orm
computers = orm.Computer.objects.find(filters={'hostname': 'localhost'}, order_by='name')
```

Deleting an instance

```
from aiida import orm
orm.Computer.objects.delete(computer.pk)
```

WHAT CHANGED: REPOSITORY INTERFACE

In `aiida-core<=0.12.*`, the file repository of a node lives on the local filesystem

Provided the `folder` property to get a folder object to interact with it, e.g.:

- `node.folder.abspath`
- `node.folder.get_abs_path('somefile.txt')`
- `node.folder.add_path('/some/path.txt', 'destination.txt')`

WHAT CHANGED: REPOSITORY INTERFACE

In `aiida-core<=0.12.*`, the file repository of a node lives on the local filesystem

Provided the `folder` property to get a folder object to interact with it, e.g.:

- `node.folder.abspath`
- `node.folder.get_abs_path('somefile.txt')`
- `node.folder.add_path('/some/path.txt', 'destination.txt')`

In the future, repository no longer necessarily lives on local filesystem

- Remote filesystem
- Object store

WHAT CHANGED: REPOSITORY INTERFACE

In `aiida-core<=0.12.*`, the file repository of a node lives on the local filesystem

Provided the `folder` property to get a folder object to interact with it, e.g.:

- `node.folder.abspath`
- `node.folder.get_abs_path('somefile.txt')`
- `node.folder.add_path('/some/path.txt', 'destination.txt')`

In the future, repository no longer necessarily lives on local filesystem

- Remote filesystem
- Object store

Additionally, files may potentially be packed in archives for efficiency reasons

- Tar archive
- Zip compressed

WHAT CHANGED: REPOSITORY INTERFACE

In `aiida-core<=0.12.*`, the file repository of a node lives on the local filesystem

Provided the `folder` property to get a folder object to interact with it, e.g.:

- `node.folder.abspath`
- `node.folder.get_abs_path('somefile.txt')`
- `node.folder.add_path('/some/path.txt', 'destination.txt')`

In the future, repository no longer necessarily lives on local filesystem

- Remote filesystem
- Object store

Additionally, files may potentially be packed in archives for efficiency reasons

- Tar archive
- Zip compressed

In preparation, node repository interface has been significantly changed

Now to interact, go through the `node.repository` property, which has methods to:

- List objects: `list_object_names`, `list_objects`
- Get objects: `get_object`, `get_object_content`, `open`
- Put objects: `put_object_from_tree`, `put_object_from_file`, `put_object_from_filelike`
- Delete objects: `delete_object`

ACKNOWLEDGMENTS

Casper
Andersen
(EPFL)

Marco
Borelli
(EPFL)

Sebastiaan
Huber
(EPFL)

Leonid
Kahle
(EPFL)

Nicola
Marzari
(EPFL)

Martin
Muhrin
(EPFL)

Elsa
Passaro
(EPFL)

Giovanni
Pizzi
(EPFL)

Aliaksandr
Yakutovich
(EPFL)

Snehal
Waychal
(EPFL)

Spyros
Zoupanos
(EPFL)

Martin Uhrin, Rico Häuselmann, Nicolas Mounet, Andrea Cepellotti, Fernando Gargiulo, Riccardo Sabatini, Rico Häuselmann, Valentin Bersier, Jocelyn Boullier, Jens Bröder, Marco Dorigo, Marco Gibertini, Dominik Gresch Eric Hontz, Daniel Marchand , Tiziano Müller, Phillippe Schwaller, Ivano E. Castelli, Ian Lee, Gianluca Prandini, Jianxing Huang, Antimo Marrazzo, Nicola Varini, Mario Zic, Vladimir Dikan, Michael Atambo, Ole Schütt, Y.-W. Fang, Philipp Rüßmann, Bonan Zhu, Andreas Stamminger, Keija Cui, Daniel Hollas, Jianxing Huang, Espen Flage-Larsen

FIN