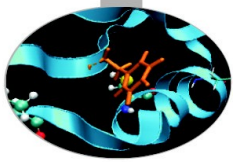
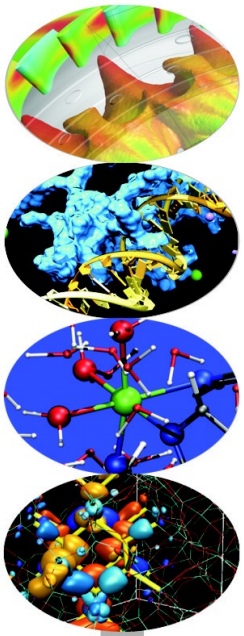
CINECA SCAI
SuperComputing Applications and Innovation



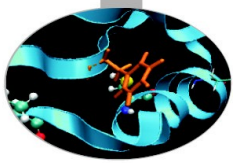


Introduction to AiiDA

Nicola Spallanzani – n.spallanzani@cineca.it
SuperComputing Applications and Innovation Department

CINECA

AiiDA



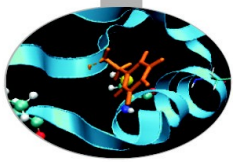
AiiDA is a flexible and scalable informatics' infrastructure to manage, preserve, and disseminate the simulations, data, and workflows of modern-day computational science.

AiiDA gives the user the ability to interact seamlessly with any number of remote HPC resources and codes, thanks to its flexible plugin interface and workflow engine for the automation of complex sequences of simulations.

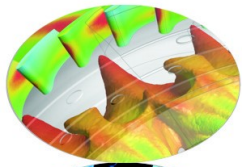
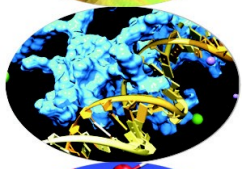
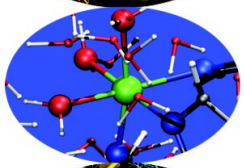
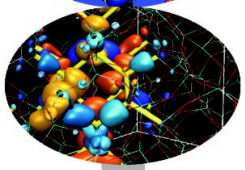
Journal ref: G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, AiiDA: ***A****utomated* ***i****nteractive* ***i****nfrastructure and* ***DA****tabase for computational science,* Comp. Mat. Sci. 111, 218-230 (2016)

http://www.aiida.net/

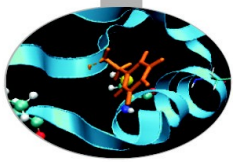
Outline



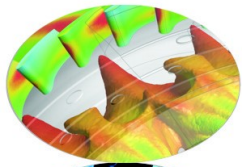
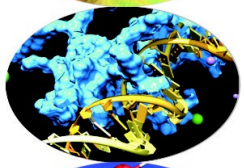
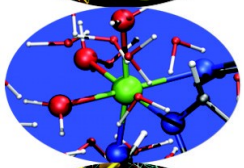
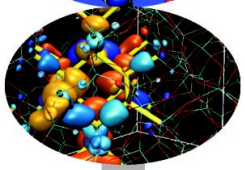
- *Philosophy and implementation*
- *Installation and setup*
- *Usage*

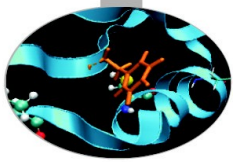
Outline

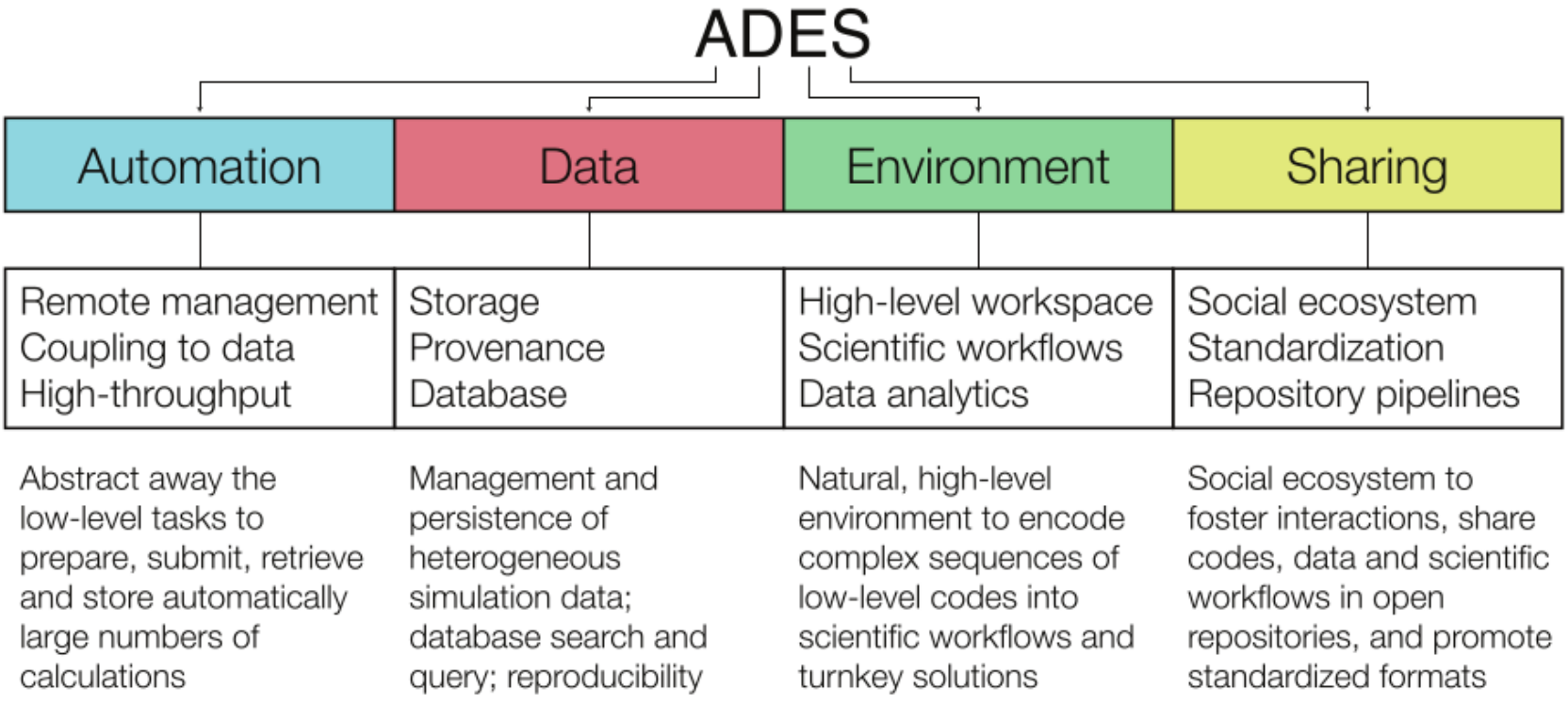


- *Philosophy and implementation*
- *Installation and setup*
- *Usage*

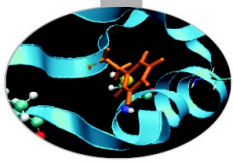





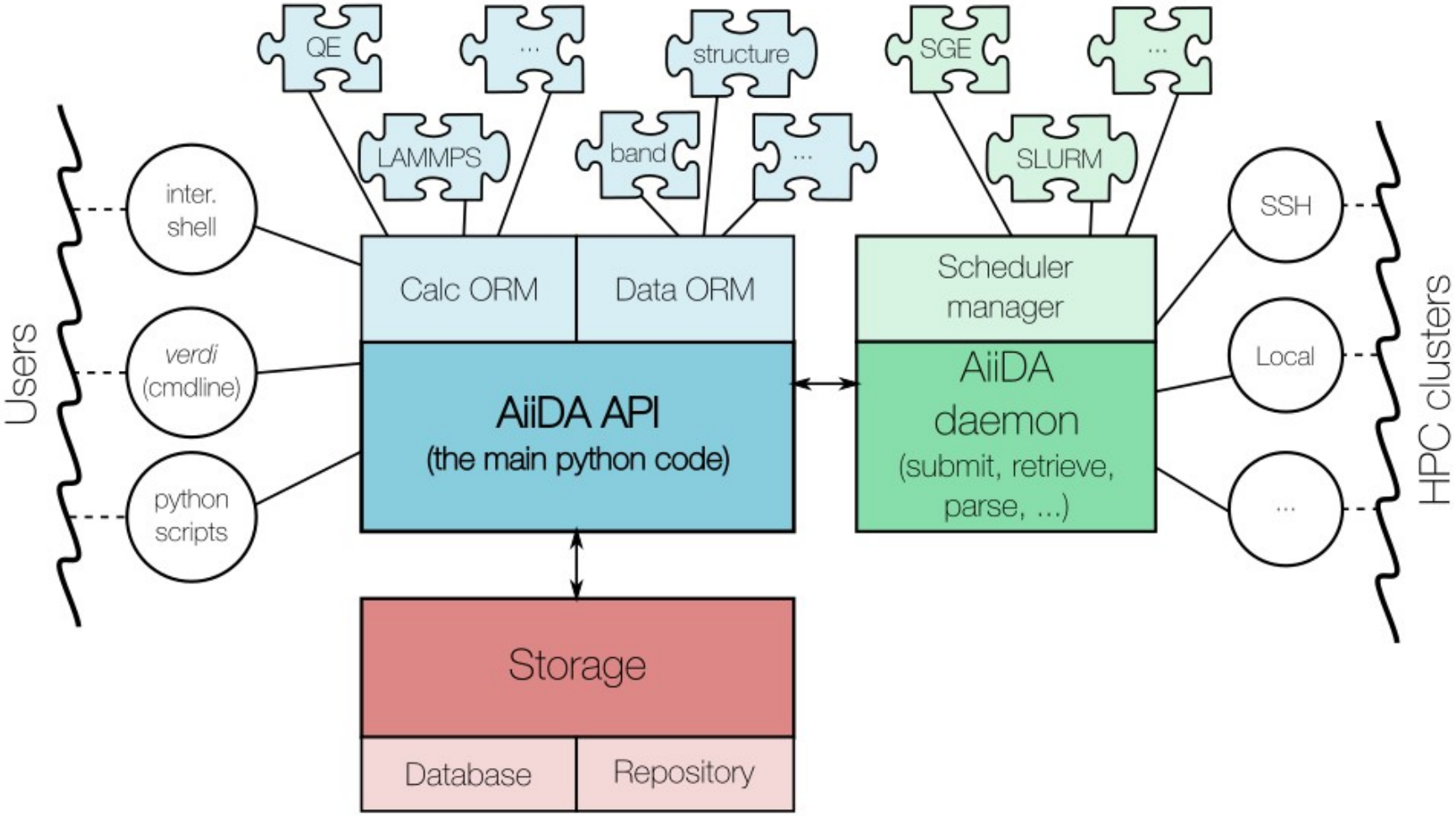
ADES model



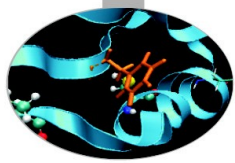

ADES
Automation
Data
Environment
Sharing
Remote management
Coupling to data
High-throughput
Storage
Provenance
Database
High-level workspace
Scientific workflows
Data analytics
Social ecosystem
Standardization
Repository pipelines
Abstract away the low-level tasks to prepare, submit, retrieve and store automatically large numbers of calculations
Management and persistence of heterogeneous simulation data; database search and query; reproducibility
Natural, high-level environment to encode complex sequences of low-level codes into scientific workflows and turnkey solutions
Social ecosystem to foster interactions, share codes, data and scientific workflows in open repositories, and promote standardized formats

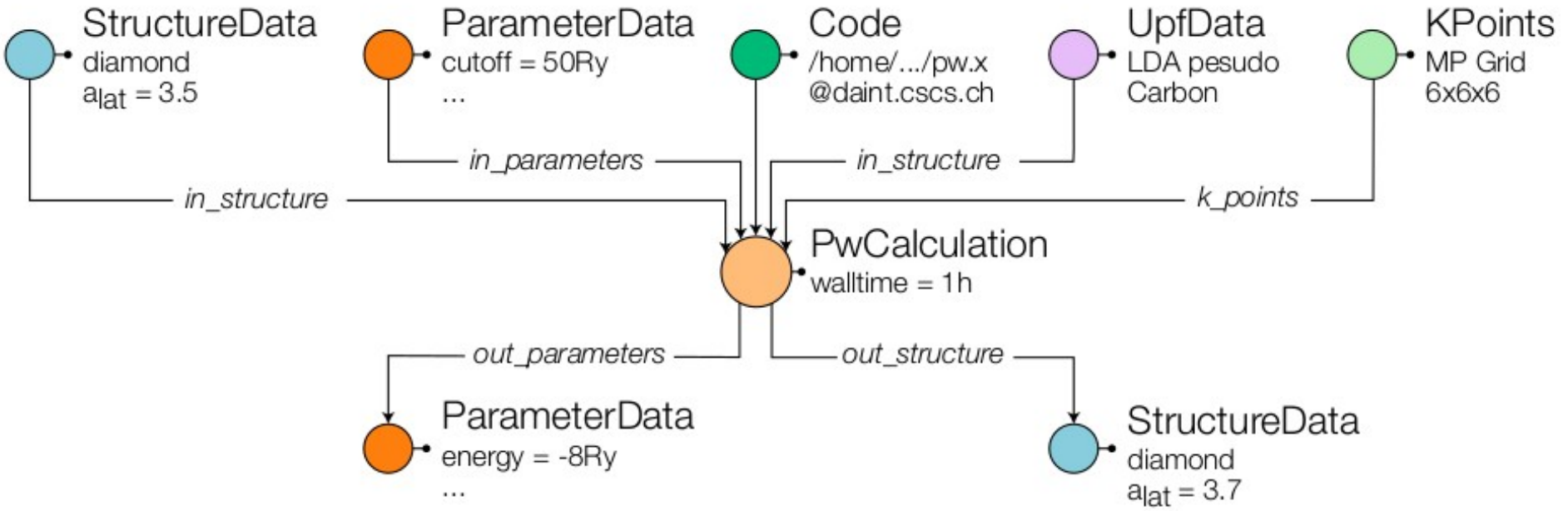
Infrastructure



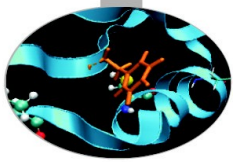


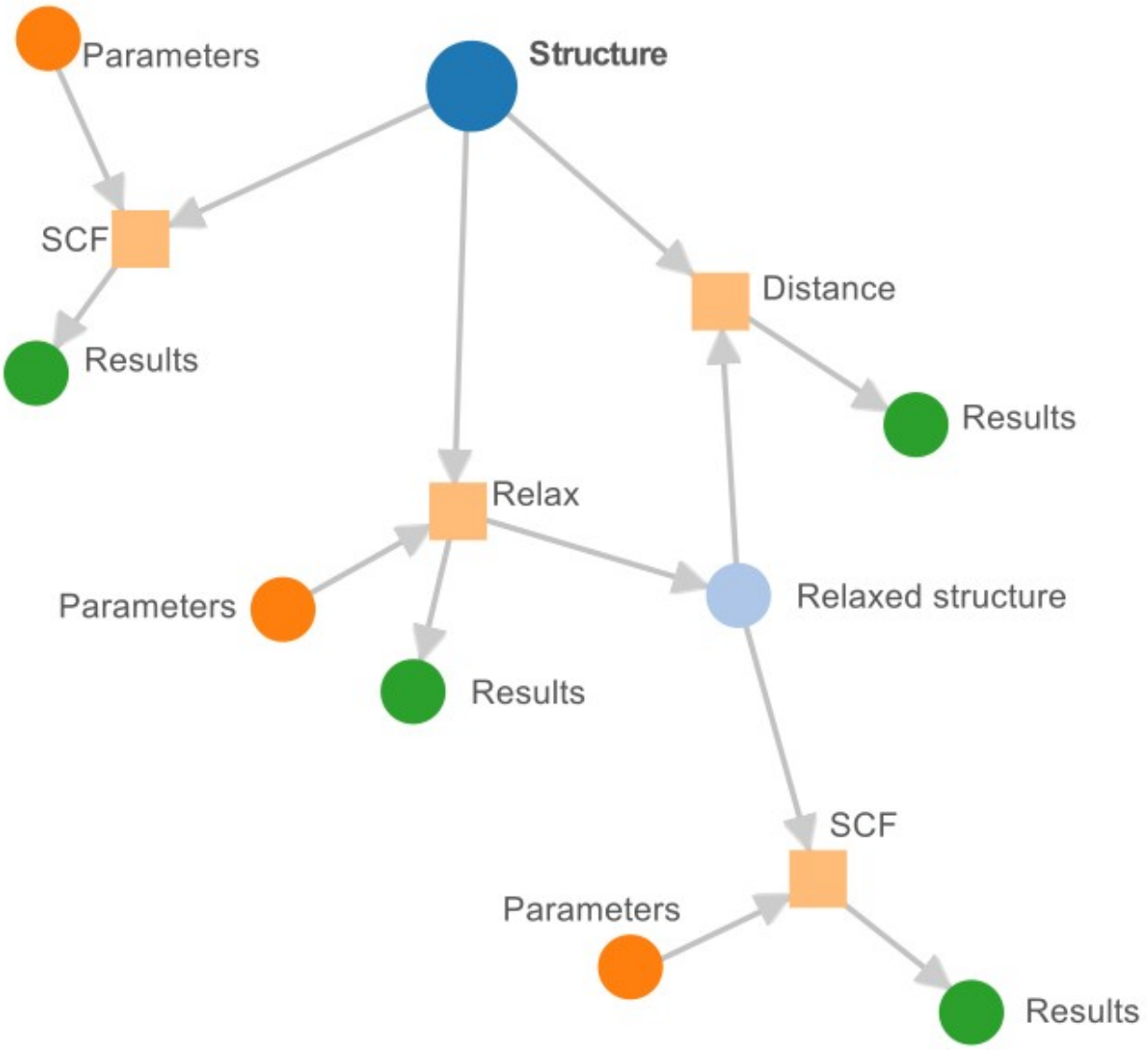
Calculation



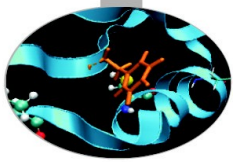


Graph

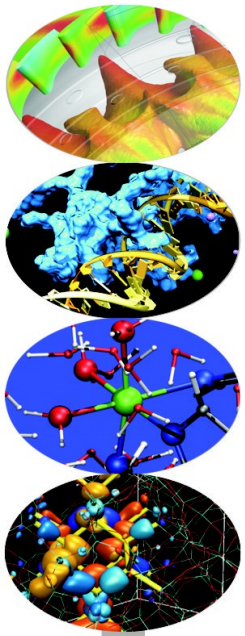



Parameters
Structure
SCF
Distance
Results
Results
Relax
Parameters
Relaxed structure
Results
SCF
Parameters
Results

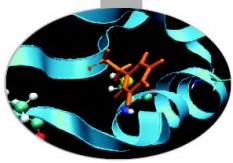
Outline



- *Philosophy and implementation*
- *Installation and setup*
- *Usage*

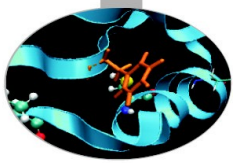


Core dependencies



- *Python 2.7 (with development files)*
- *iPython (optional, but suggested by AiiDA developers)*
- *Virtualenv (optional, but suggested by me)*
- *Git (optional, for download some dependencies)*
- *Development files of a supported database:*
  - *SQLite3 (only if you just want to try AiiDA)*
  - *PostgreSQL (suggested by AiiDA developers)*
  - *MySQL*

VIRTUALENV

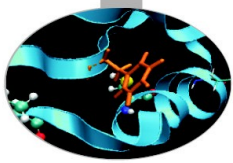


Virtualenv is a tool to create isolated Python environments.

Main feature: avoids the problem of module dependencies and versions.

```
$ mkdir envs
$ virtualenv envs/aiida
$ . envs/aiida/bin/activate
(aiida)$ pip install -U pip
(aiida)$ deactivate
$
```

Python dependencies



http://www.aiida.net/download/

```
(aiida)$ tar zxvf aiida_epfl-0.7.0.tar.gz
(aiida)$ cd aiida_epfl
(aiida)$ pip install -U -r requirements.txt
```

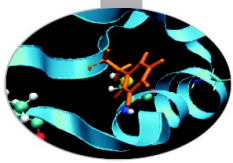
- eventually -

```
(aiida)$ pip install psycopg2==2.6.1
```

- or -

```
(aiida)$ pip install MySQL-python==1.2.5
```

AiiDA Configuration



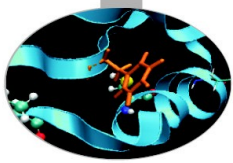
Edit ~/.bashrc file:

```
# AiiDA paths
export PYTHONPATH=~/aiida_epfl:${PYTHONPATH}
export PATH=~/aiida_epfl/bin:${PATH}
```

Edit ~/envs/aiida/bin/activate file:

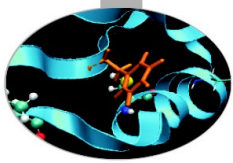
```
# AiiDA verdi completion
eval "$(verdi completioncommand)"
```

AiiDA Setup



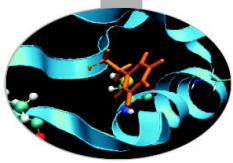
```
(aiida)$ verdi install
[...]
Database engine: postgresql_psycopg2
AiiDA Database password: *********
AiiDA Database name: aiidadb
Database host: localhost
AiiDA backend: django
Database port: 5432
Default user email: n.spallanzani@cineca.it
Timezone: Europe/Rome
AiiDA repository directory:
file:///home/aiida/.aiida/repository-default/
AiiDA Database user: aiida
```

AiiDA Setup



```
(aiida)$ verdi daemon configureuser
(aiida)$ verdi daemon start

(aiida)$ verdi daemon status
# Most recent daemon timestamp:0h:00m:06s ago
## Found 1 process running:
   * aiida-daemon[aiida-daemon] RUNNING
pid 31890, uptime 3 days, 4:21:29

(aiida)$ verdi daemon stop
```

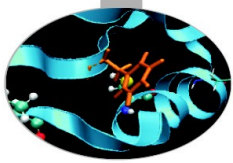
Computer



Requirements for a computer are:

- It must run a Unix-like operating system
- The default shell must be bash
- It should have a batch scheduler installed (see here for a list of supported batch schedulers)
- It must be accessible from the machine that runs AiiDA using one of the available transports
  - Local
  - Ssh (remote)

Password-less login



```
$ ssh-keygen -t rsa
$ ssh-copy-id nspalla1@login.marconi.cineca.it
```

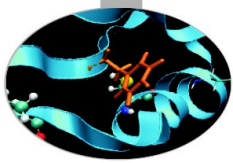
Edit ~/.ssh/config file:

```
Host login.marconi.cineca.it
  User nspalla1
  HostKeyAlgorithms ssh-rsa
  IdentityFile ~/.ssh/id_rsa
```

Try it:

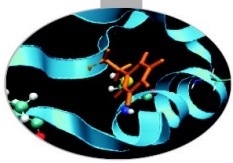
```
$ ssh login.marconi.cineca.it
$ sftp login.marconi.cineca.it
```

Computer setup



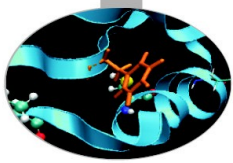
```
(aiida)$ verdi computer setup
[...]
=> Computer name: marconi_bdw
=> Fully-qualified hostname: login.marconi.cineca.it
=> Description: Lenovo NeXtScale, 1.512 nodes with 2 x
18-cores Intel Xeon E5-2697 v4 (Broadwell) at 2.30 GHz
and 128 GB
=> Enabled: True
=> Transport type: ssh
=> Scheduler type: pbspro
=> AiiDA work directory:
/marconi_scratch/userinternal/nspalla1/aiida_run
=> mpirun command: mpirun -np {tot_num_mpiprocs}
=> Default number of CPUs per machine: 36
=> Text to prepend to each command execution:
=> Text to append to each command execution:
Computer 'marconi_bdw' successfully stored in DB.
pk: 2, uuid: 5b503859-9f93-49db-aec4-a04c370dfa8c
```

Computer setup



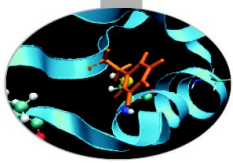
```
(aiida)$ verdi computer configure marconi_bdw
[...]
=> username = nspalla1
=> port = 22
=> look_for_keys = True
=> key_filename =
=> timeout = 60
=> allow_agent =
=> proxy_command =
=> compress = True
=> load_system_host_keys = True
=> key_policy = RejectPolicy
Configuration stored for your user on computer
'marconi'.
```

Computer setup



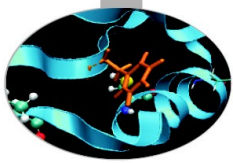
```
(aiida)$ verdi computer list
# List of configured computers:
# (use 'verdi computer show COMPUTERNAME' to
see the details)
* galileo
* marconi_bdw

(aiida)$ verdi computer show marconi_bdw

(aiida)$ verdi computer test marconi_bdw
```

Code setup



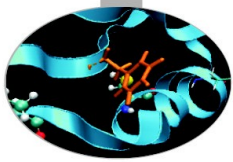
```
(aiida)$ verdi code setup
[...]
=> Label: qe/5.4.0/pw.x
=> Description: Quantum Espresso v.5.4.0, pw.x code, compiled
with intelmpi (2017)
=> Local: False
=> Default input plugin: quantumespresso.pw
=> Remote computer name: marconi_bdw
=> Remote absolute path:
/cineca/prod/opt/applications/qe/5.4.0/intelmpi--2017--
binary/bin/pw.x
=> Text to prepend to each command execution:
module load profile/phys
module load autoload qe/5.4.0
=> Text to append to each command execution:

Code 'qe/5.4.0/pw.x' successfully stored in DB.
pk: 96, uuid: fb863032-c0cb-42f0-8e62-434d1490ef66
```

Code setup

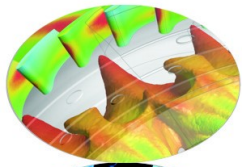
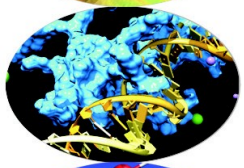
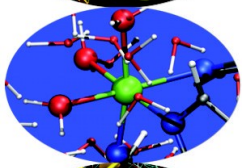
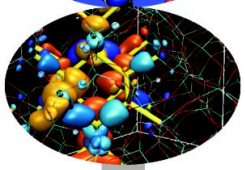


```
(aiida)$ verdi code list
# List of configured codes:
# (use 'verdi code show CODEID' to see the
details)
* pk 1 - qe/5.4.0/pw.x@galileo
* pk 31 - qe/5.4.0/ph.x@galileo
* pk 96 - qe/5.4.0/pw.x@marconi_bdw
* pk 97 - qe/5.4.0/ph.x@marconi_bdw

(aiida)$ verdi code show 96
```

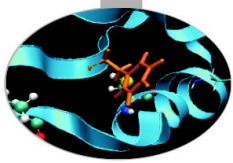
Outline



- *Philosophy and implementation*
- *Installation and setup*
- *Usage*

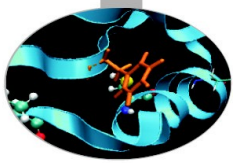
Quantum Espresso
PWscf user-tutorial



Using a python script (pw_short_example.py):

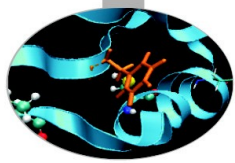
```
#!/usr/bin/env python
from aiida import load_dbenv
load_dbenv()

# Code
codename = 'qe/5.4.0/pw.x@marconi_bdw'
from aiida.orm import Code
code = Code.get_from_string(codename)
```

Quantum Espresso
PWscf user-tutorial



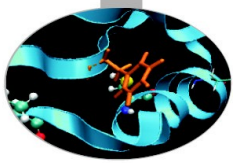
```
# Structure
from aiida.orm import DataFactory
StructureData = DataFactory('structure')

alat = 4. # angstrom
cell = [[alat, 0., 0.,],
        [0., alat, 0.,],
        [0., 0., alat,],
       ]

# BaTiO3 cubic structure
s = StructureData(cell=cell)
s.append_atom(position=(0.,0.,0.),symbols='Ba')
s.append_atom(position=(alat/2.,alat/2.,alat/2.),symbols='Ti')
s.append_atom(position=(alat/2.,alat/2.,0.),symbols='O')
s.append_atom(position=(alat/2.,0.,alat/2.),symbols='O')
s.append_atom(position=(0.,alat/2.,alat/2.),symbols='O')
```

Quantum Espresso
PWscf user-tutorial



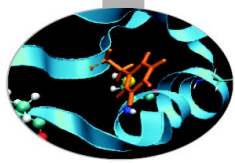
```
# Parameters
ParameterData = DataFactory('parameter')

parameters = ParameterData(dict={
          'CONTROL': {
              'calculation': 'scf',
              'restart_mode': 'from_scratch',
              'wf_collect': True,
              },
          'SYSTEM': {
              'ecutwfc': 30.,
              'ecutrho': 240.,
              },
          'ELECTRONS': {
              'conv_thr': 1.e-6,
              }})

KpointsData = DataFactory('array.kpoints')
kpoints = KpointsData()
kpoints.set_kpoints_mesh([4,4,4])
```

Quantum Espresso
PWscf user-tutorial



```
# Calculation
calc = code.new_calc()

calc.set_max_wallclock_seconds(30*60) # 30 min
calc.set_resources({"num_machines": 1,
                    "num_cores_per_machine": 16,
                    "num_mpiprocs_per_machine": 8})
calc.set_max_memory_kb(120*1024*1024) # 120gb
calc.set_custom_scheduler_commands("#PBS -A cin_staff")
# calc.set_queue_name("the_queue_name")

calc.set_environment_variables({"OMP_NUM_THREADS": "2"})
calc.set_withmpi(True) # default
calc.label = "A generic title"
calc.description = "A much longer description"
```

Quantum Espresso
PWscf user-tutorial

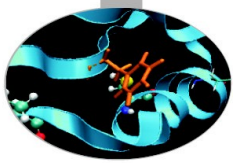


```
calc.use_structure(s)
calc.use_code(code)
calc.use_parameters(parameters)
calc.use_kpoints(kpoints)
calc.use_pseudos_from_family('pbesol-uspp')
```

How to create a pseudopotential family:

```
(aiida)$ verdi data upf uploadfamily path/to/folder \
         pbesol-uspp "some description for your convenience"

(aiida)$ verdi data upf listfamilies
* pbesol-uspp [3 pseudos]
* pz-hgh [3 pseudos]
```

Quantum Espresso
PWscf user-tutorial

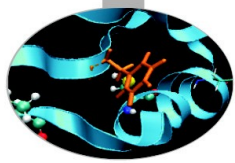


```
# Execute
calc.store_all()
print "created calculation; with uuid='{}' and
       PK={}".format(calc.uuid,calc.pk)
calc.submit()
```

Submission:

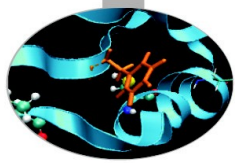
```
(aiida)$ python pw_short_example.py
```

Quantum Espresso
PWscf user-tutorial



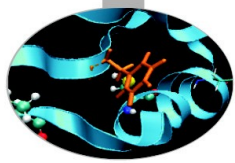
```
(aiida)$ verdi calculation list -a -p1
# Last daemon state_updater check: 0h:00m:18s ago (at 11:05:48 on 2016-12-02)
  PK  State              Creation    Sched. state    Computer    Type
----  ---------------  ----------  --------------  ----------  ------------------
  26  WITHSCHEDULER    24s ago     None            galileo     quantumespresso.pw

Number of rows: 1

(aiida)$ verdi calculation list -a -p1
# Last daemon state_updater check: 0h:00m:28s ago (at 11:07:19 on 2016-12-02)
  PK  State              Creation    Sched. state    Computer    Type
----  ---------------  ----------  --------------  ----------  ------------------
  26  FINISHED         2m ago      DONE            galileo     quantumespresso.pw

Number of rows: 1
```

Quantum Espresso
PWscf user-tutorial

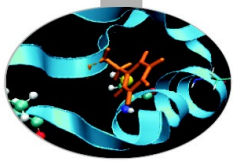


```
(aiida)[aiida@aiida benchmark]$ verdi shell
Python 2.7.5 (default, Sep 15 2016, 22:37:39)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: calc = load_node(26)

In [2]: calc.res.energy
Out[2]: -3900.8472554484

In [3]: calc.res.energy_units
Out[3]: u'eV'
```

Quantum Espresso
PWscf user-tutorial



```
(aiida)$ verdi calculation res 26 -k energy energy_units
{
  "energy": -3900.8472554484,
  "energy_units": "eV"
}
```



Thanks for your attention!