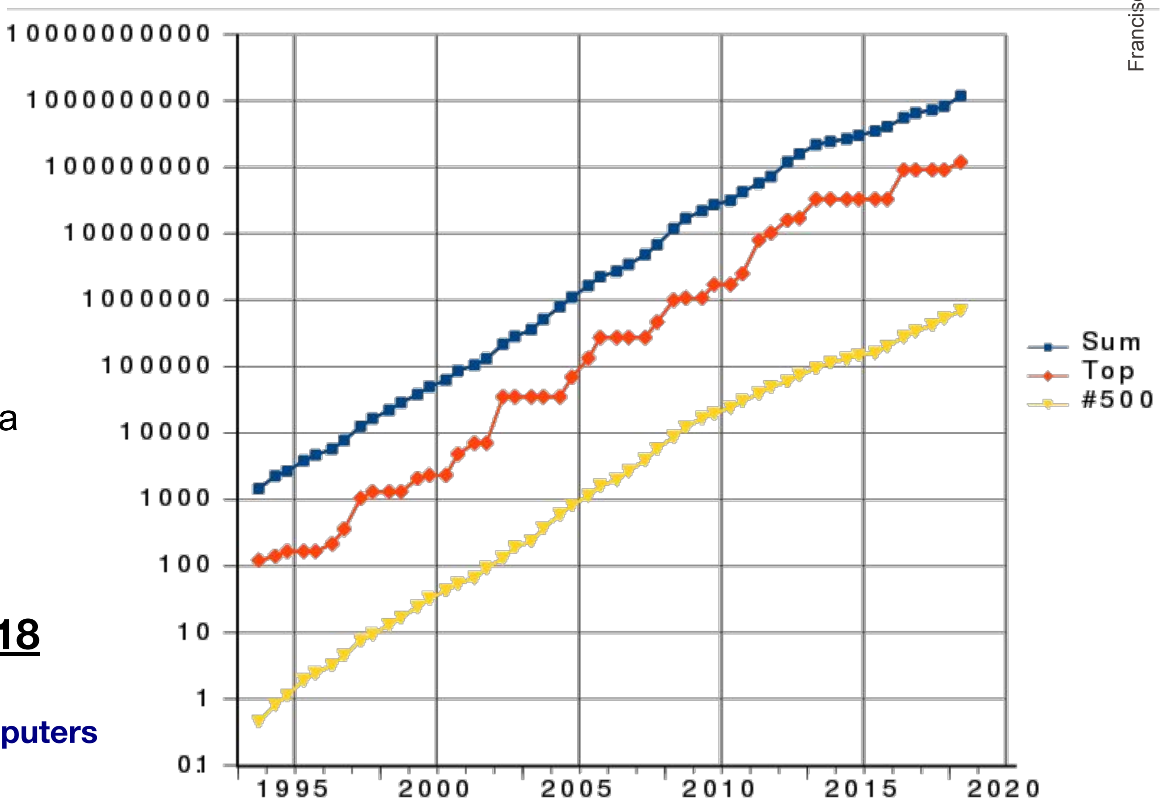
Introduction to AiiDA for HPC

Presenter: Francisco F. Ramirez (THEOS @ EPFL)

CHALLENGES OF HIGH-THROUGHPUT COMPUTING

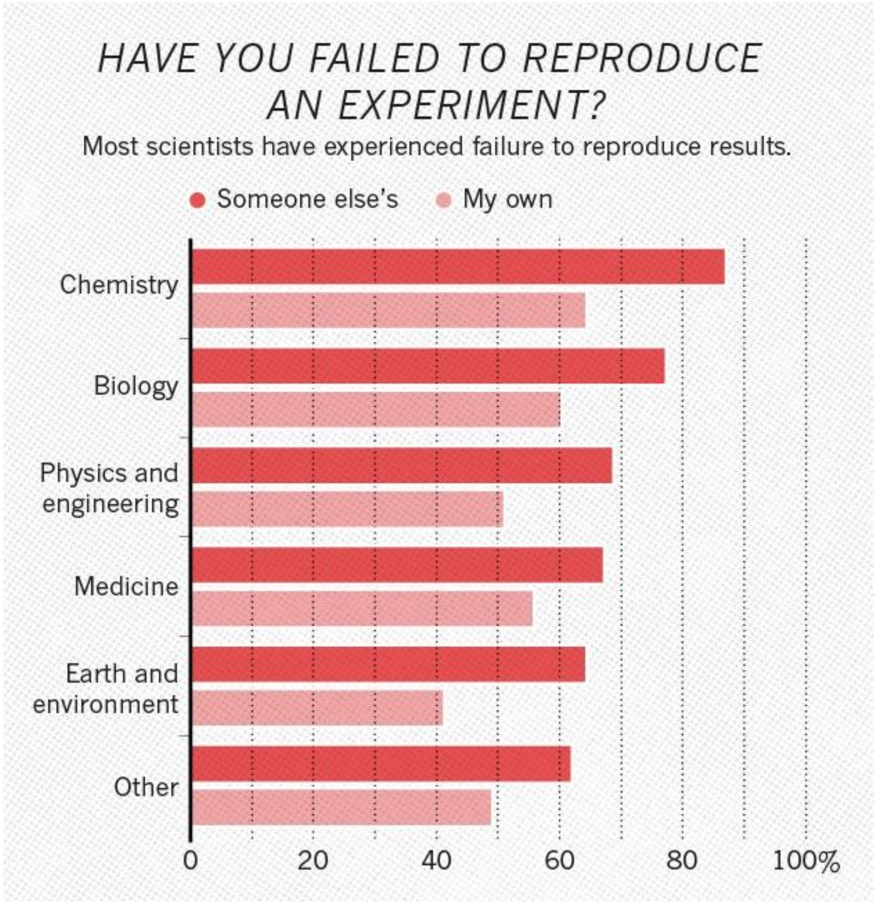
- Moore's Law
- More complex systems (HPC)
- More extensive sampling (HTC)
- Exponential increase in resource utilization and data generation...
  - how to manage it?



Computing power 1993-2018

- Sum of the top 500 supercomputers
- Number 1
- Number 500

CHALLENGES OF HIGH-THROUGHPUT COMPUTING



- **Reproducibility crisis** - even for the experiments performed by the same person/group.
- Experimental disciplines have partial knowledge of their system and the unreliability of the environment.
- Computational disciplines have a tighter control of their systems and a more deterministic methodology
  - Mostly an issue of data management
  - Will only get worse as we increase the amount of data to manage.

Nature 533, 452–454 (2016)

EPFL

CHALLENGES OF HIGH-THROUGHPUT COMPUTING

Efficient data management

- Automated tracking of the full **data provenance**.
- Data discovery and analysis enabled by a simplified querying language.
- Logging of calculations and their computational environment.
- Flexible integration of databases and file repositories.

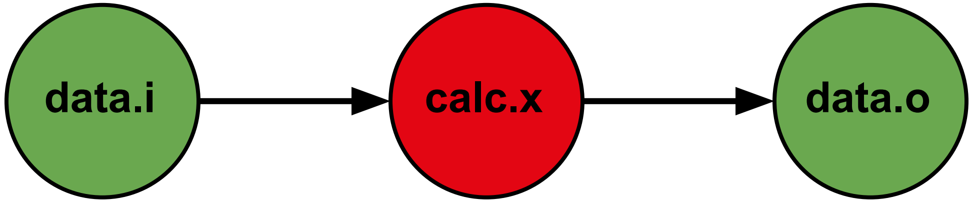
Robust workflow automation

- Automated, high-performant scheduling and execution on local and remote resources.
- Language to define complex workflows that codify scientific workflows and include built-in error handling.
- An expandable system with independent plugins that are easy to design, package and distribute.

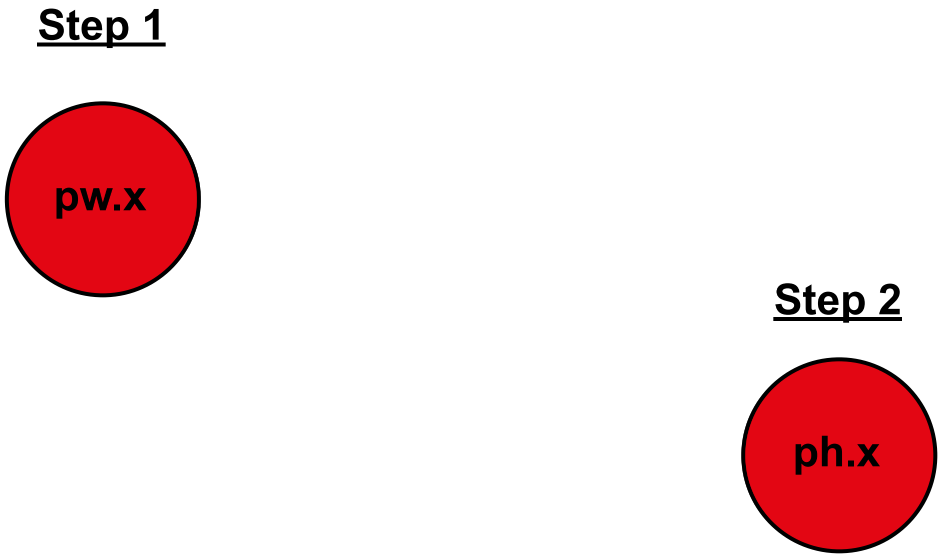
DATA PROVENANCE

- Inspiration from the open provenance model
- Each independent piece of data is represented as a **data node** (a crystal structure, a charge distribution, a set of parameters for a program)
- Each transformation of a group of data nodes into another is represented as a **calculation node** (a simulation, a script that expands or contracts a crystal cell, a post-processing tool that calculates a property from the outputs)

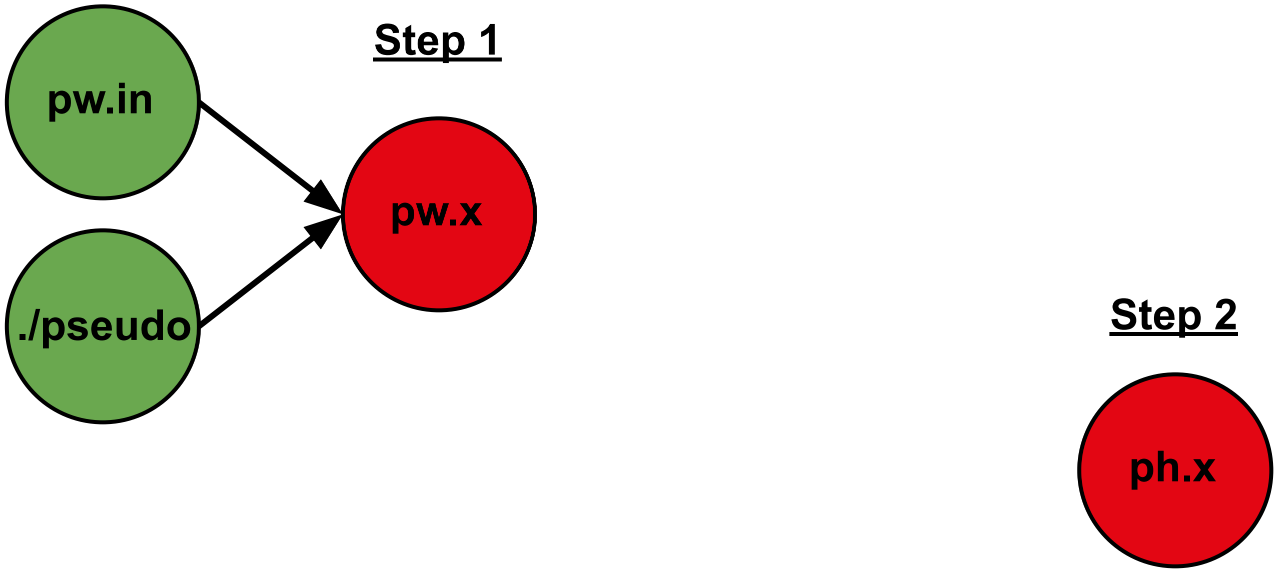
Therefore, data that is derived from pre-existing information has a record of its origin thanks to the connection through a calculation node.



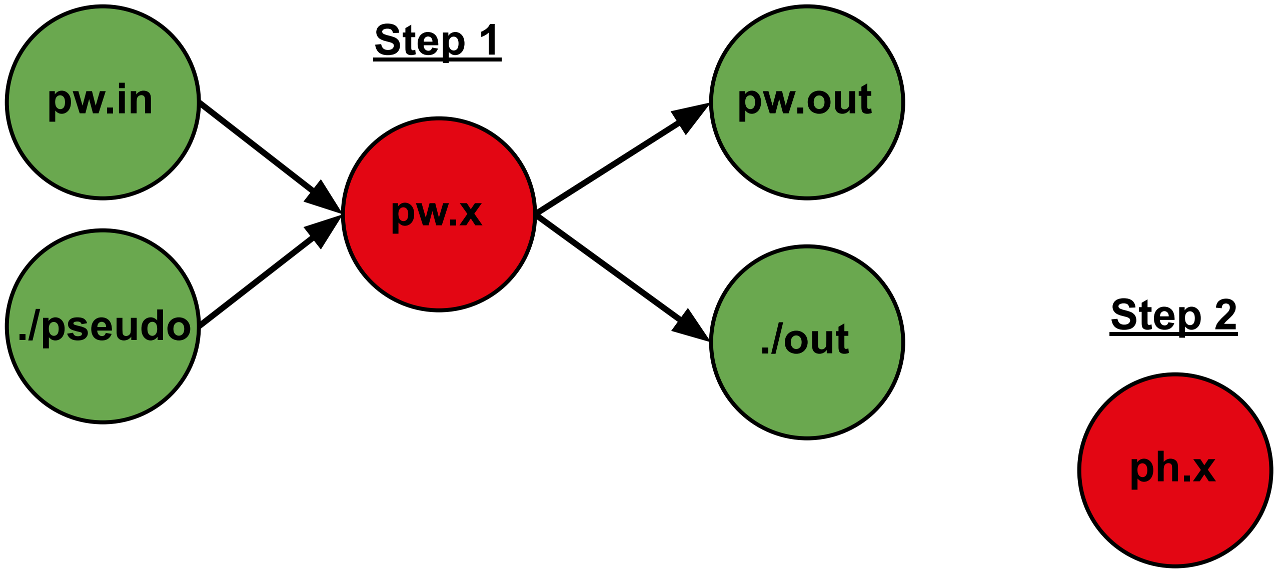
DATA PROVENANCE



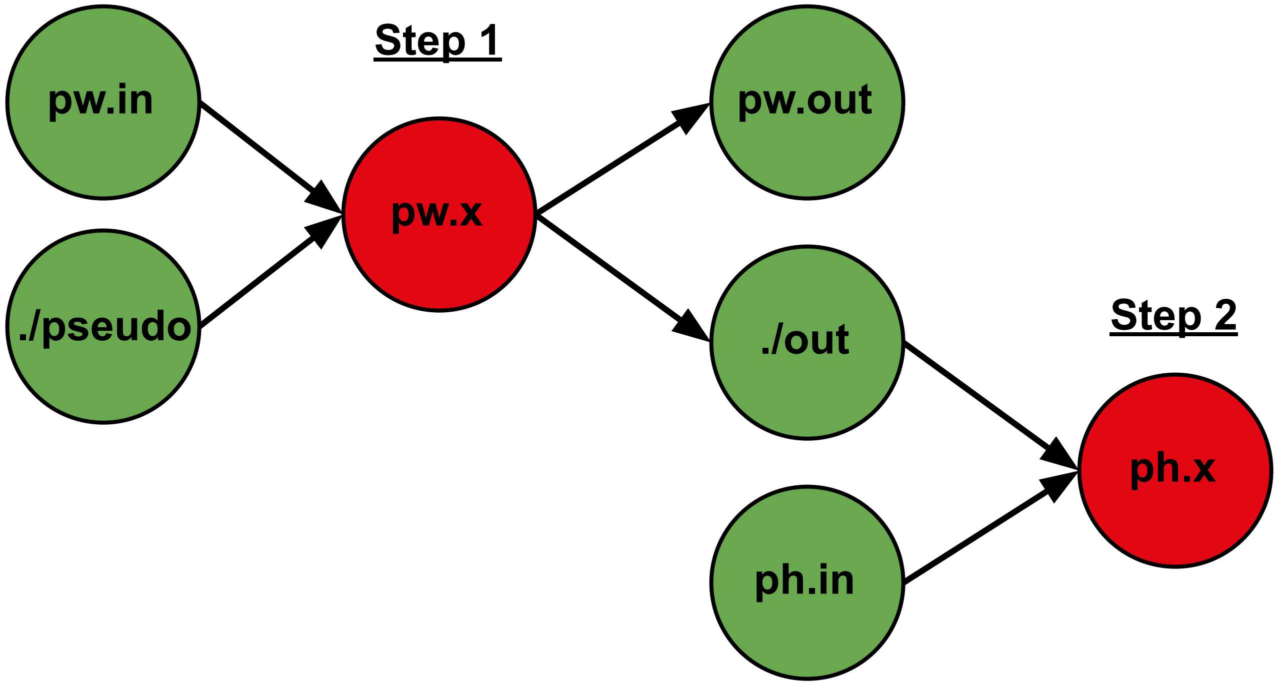
DATA PROVENANCE



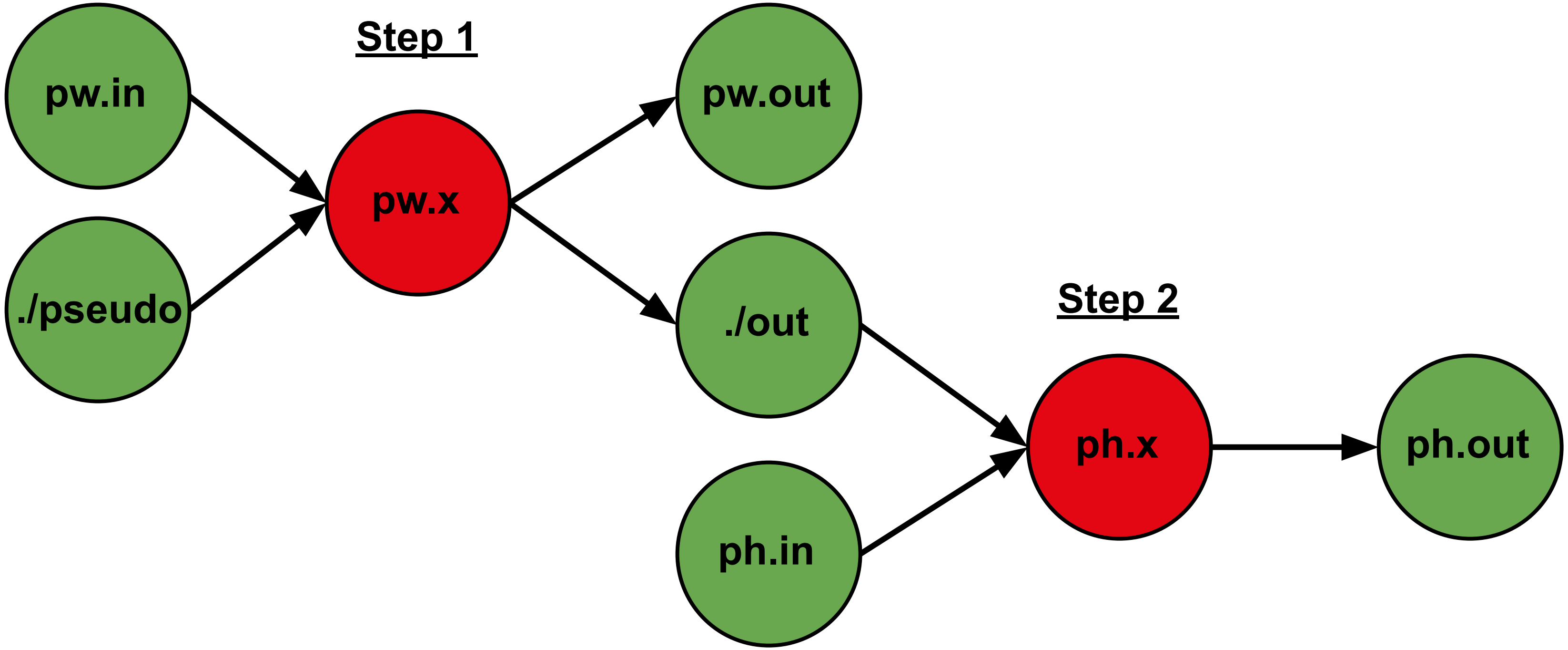
DATA PROVENANCE



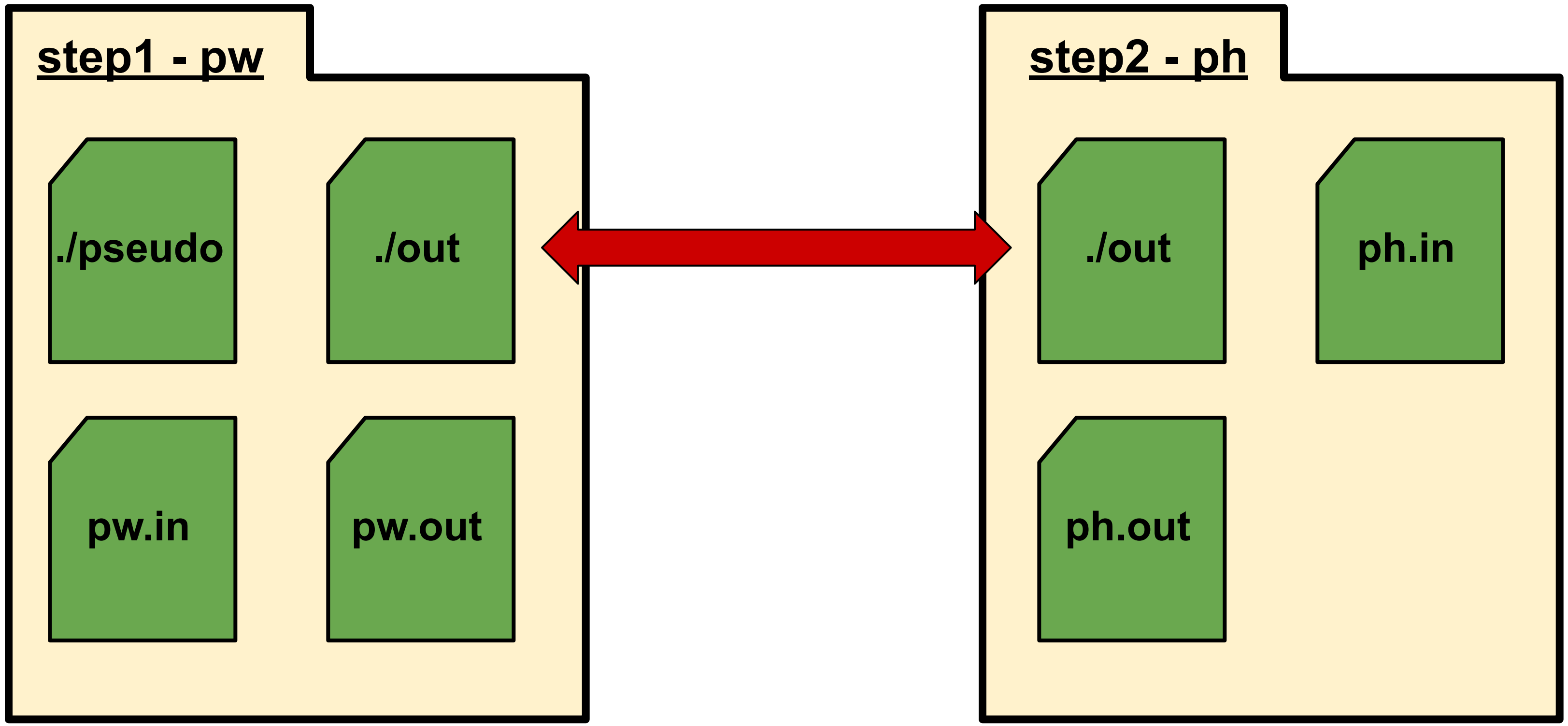
DATA PROVENANCE



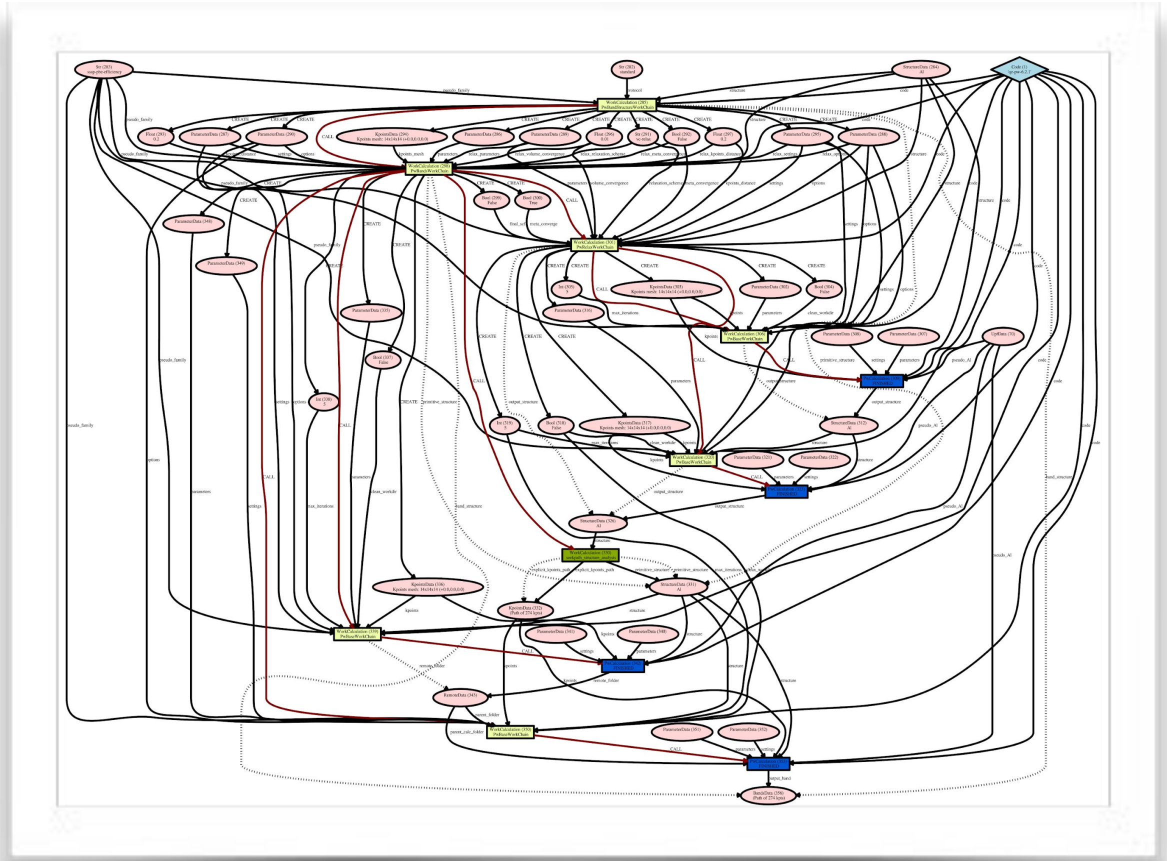
DATA PROVENANCE



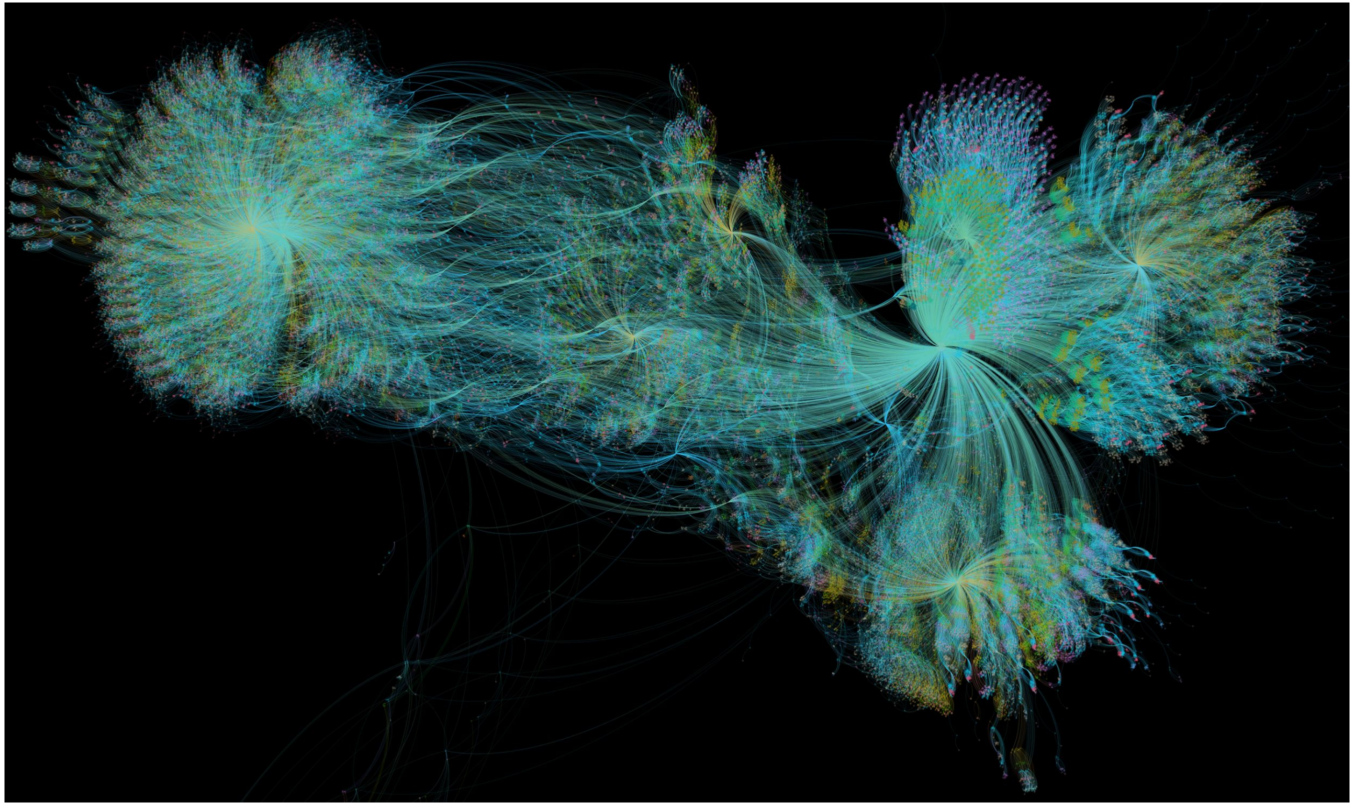
DATA PROVENANCE



DATA PROVENANCE


Str (283) sssp-pbe-efficiency
Str (282) standard
StructureData (284) Al
Code (1) qe-pw-6.2.1
WorkCalculation (285) PwBandStructureWorkChain
WorkCalculation (298) PwBandsWorkChain
WorkCalculation (301) PwRelaxWorkChain
WorkCalculation (306) PwBaseWorkChain
WorkCalculation (320) PwBaseWorkChain
WorkCalculation (330) seekpath_structure_analysis
WorkCalculation (339) PwBaseWorkChain
WorkCalculation (350) PwBaseWorkChain
RemoteData (345)
BandsData (356) (Path of 274 kpts)

DATA PROVENANCE


Provenance graph of a high-throughput study (courtesy of Jens Broeder).

AiiDA CORE

- Python Library (pip installable)
- Open Source (MIT License)
  [Numfocus affiliated project as of Feb 2020]
- There are many interfaces to interact with AiiDA:
  - Python ORM for designing workflows and pre/post-processing scripts.
  - Verdi CLI for submitting and controlling running processes.
  - REST-API for creating services that interact with AiiDA (Mat Cloud)
  - Jupyter-lab widgets (AiiDAlab)

https://pypi.org/project/aiida/

github.com/aiidateam/aiida-core

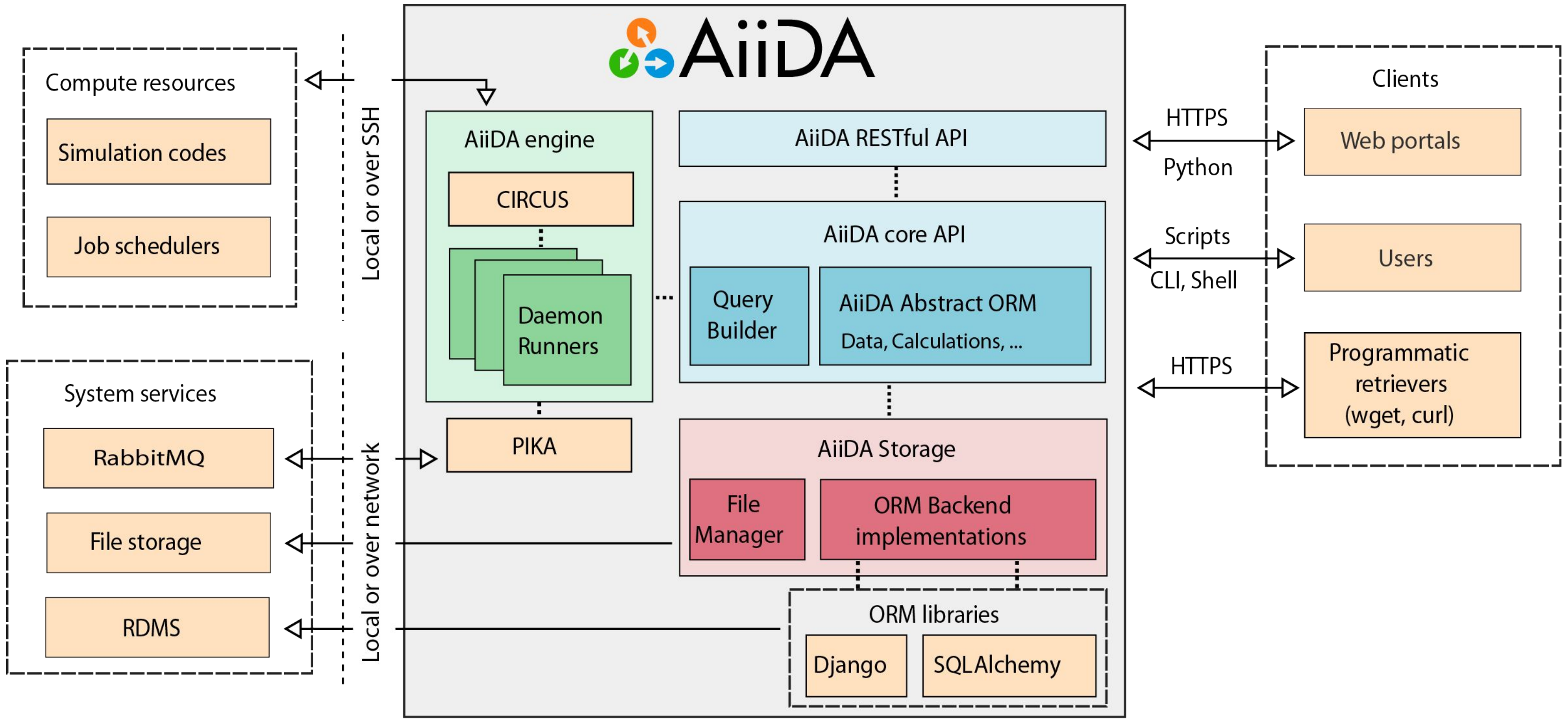
MIT LICENSED

open source initiative

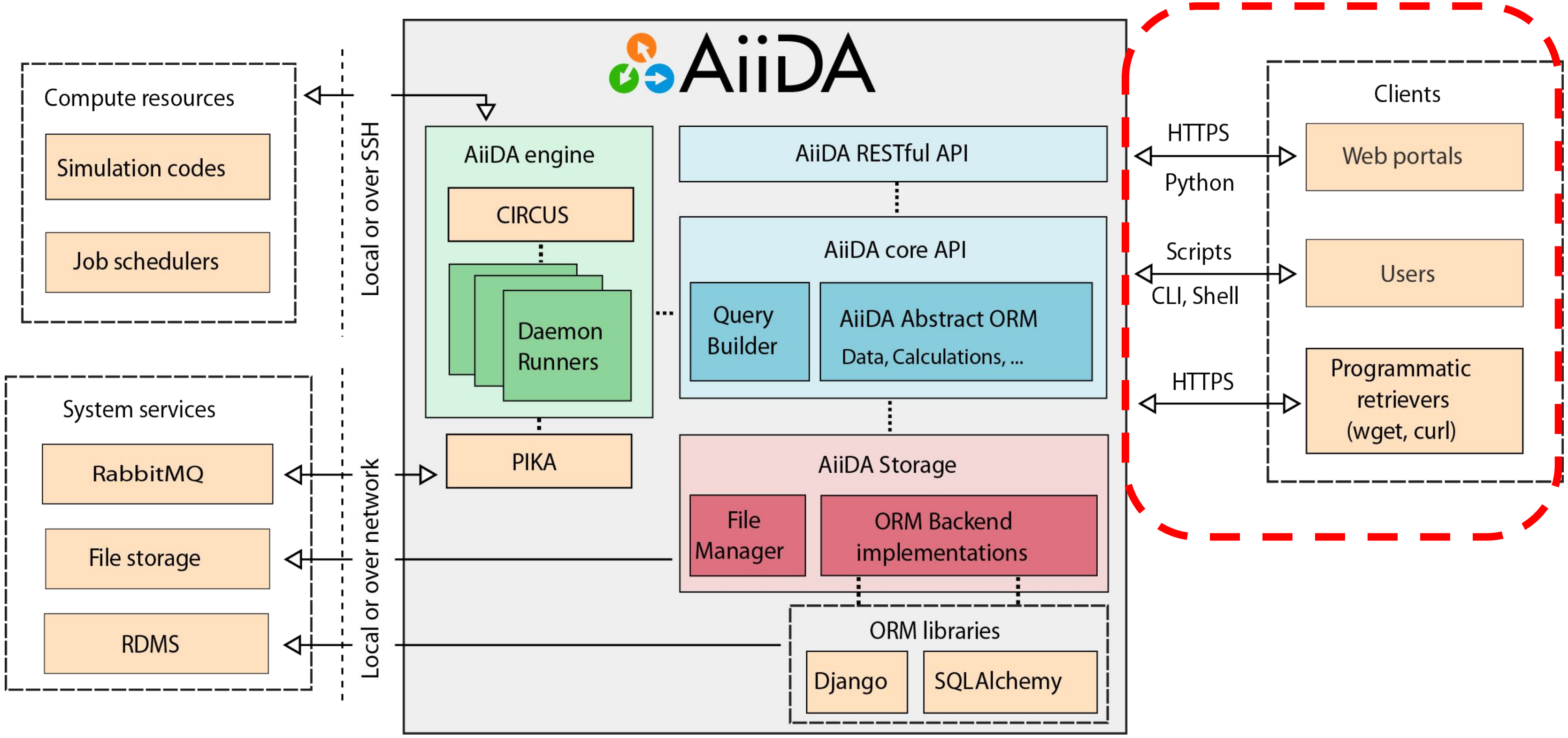
NUMFOCUS
OPEN CODE = BETTER SCIENCE

EPFL

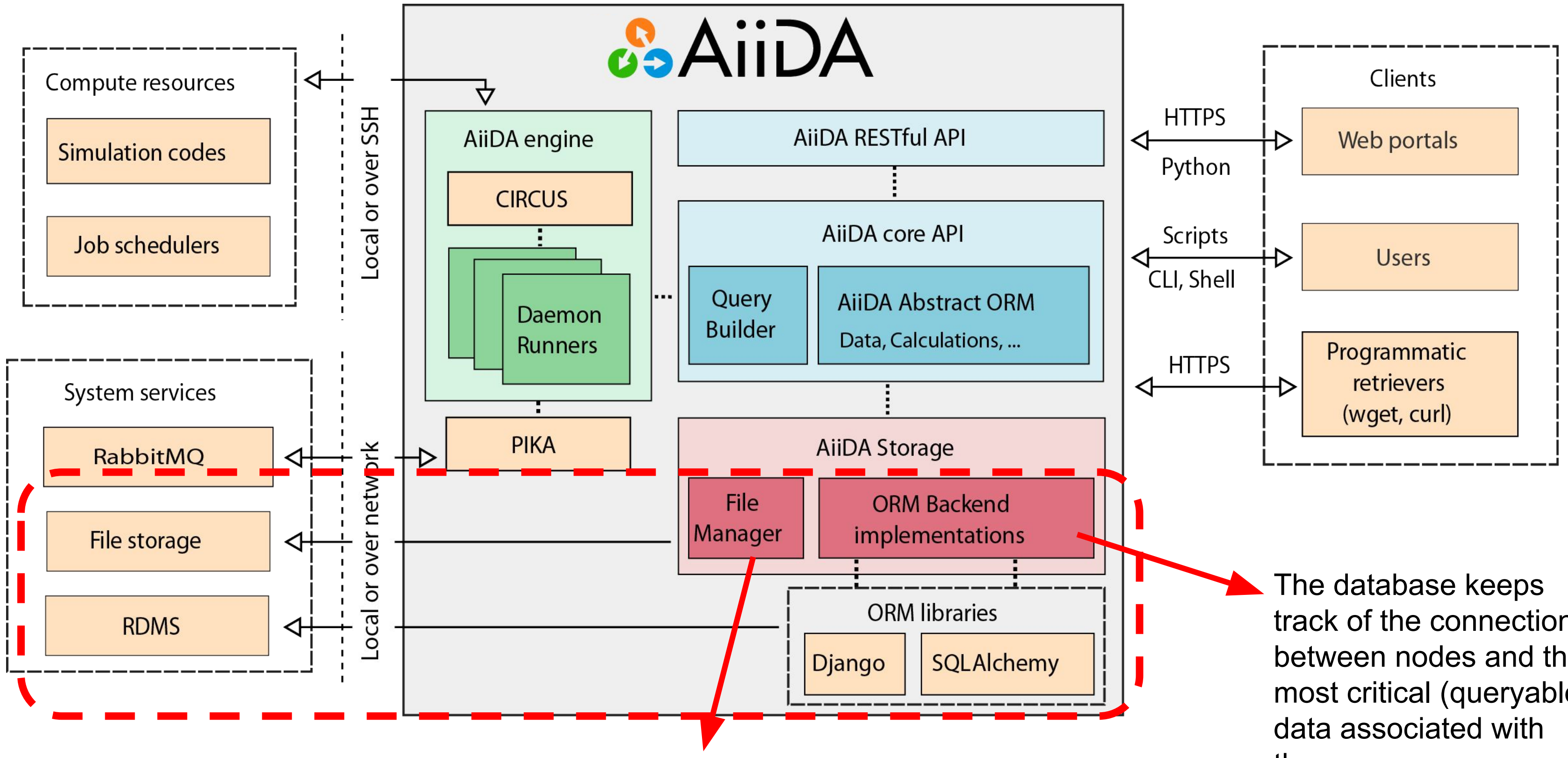
AiiDA CORE



AiiDA CORE


Compute resources
Simulation codes
Job schedulers
Local or over SSH
System services
RabbitMQ
File storage
RDMS
Local or over network
AiiDA
AiiDA engine
CIRCUS
Daemon Runners
PIKA
AiiDA RESTful API
AiiDA core API
Query Builder
AiiDA Abstract ORM
Data, Calculations, ...
AiiDA Storage
File Manager
ORM Backend implementations
ORM libraries
Django
SQLAlchemy
HTTPS
Python
Scripts
CLI, Shell
HTTPS
Clients
Web portals
Users
Programmatic retrievers (wget, curl)

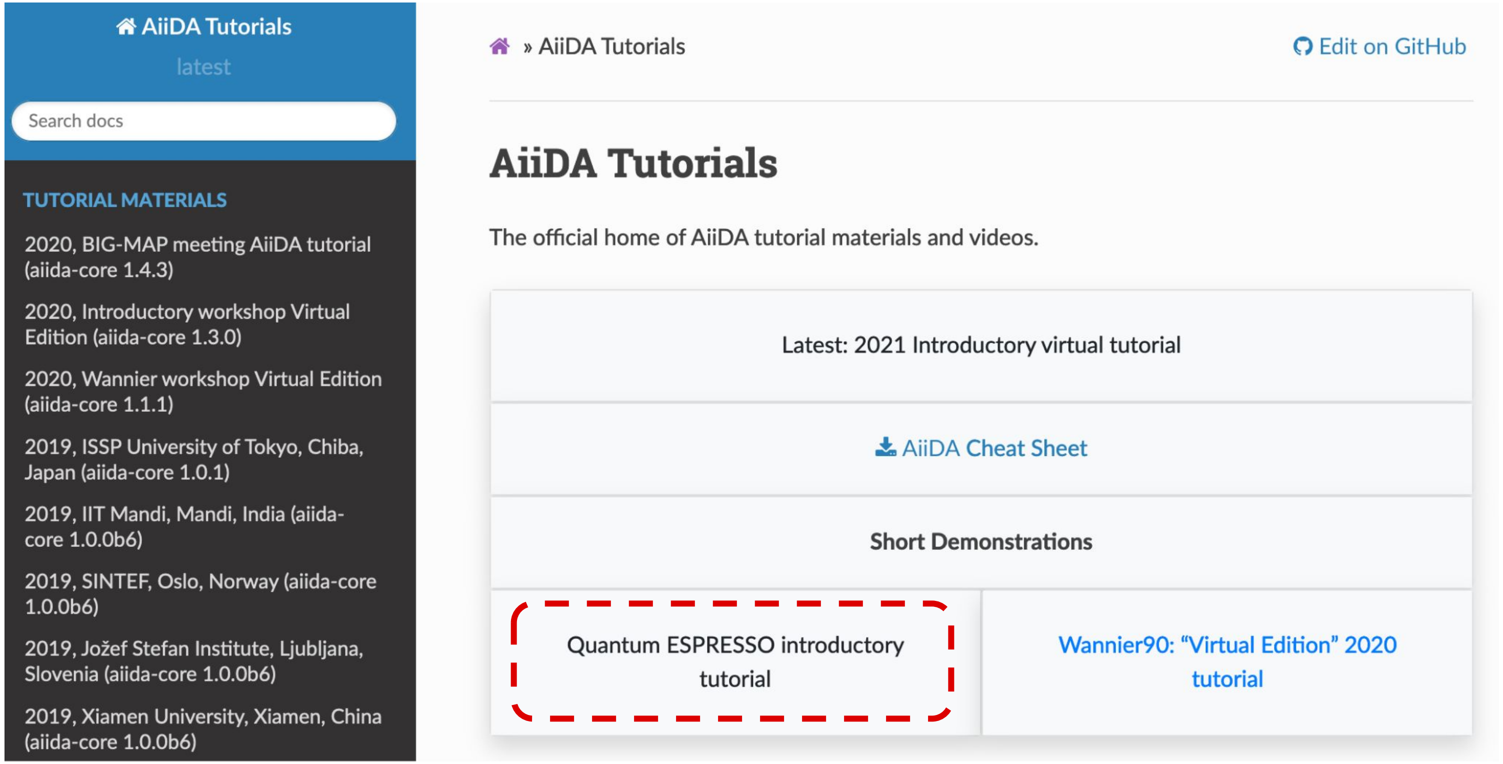
AiiDA CORE



The database keeps track of the connections between nodes and the most critical (queryable) data associated with them

The file repository stores all other relevant (bigger) data pieces

USAGE EXAMPLE



TUTORIAL MATERIALS

- 2020, BIG-MAP meeting AiiDA tutorial (aiida-core 1.4.3)
- 2020, Introductory workshop Virtual Edition (aiida-core 1.3.0)
- 2020, Wannier workshop Virtual Edition (aiida-core 1.1.1)
- 2019, ISSP University of Tokyo, Chiba, Japan (aiida-core 1.0.1)
- 2019, IIT Mandi, Mandi, India (aiida-core 1.0.0b6)
- 2019, SINTEF, Oslo, Norway (aiida-core 1.0.0b6)
- 2019, Jožef Stefan Institute, Ljubljana, Slovenia (aiida-core 1.0.0b6)
- 2019, Xiamen University, Xiamen, China (aiida-core 1.0.0b6)

» AiiDA Tutorials — Edit on GitHub

AiiDA Tutorials

The official home of AiiDA tutorial materials and videos.

Latest: 2021 Introductory virtual tutorial

AiiDA Cheat Sheet

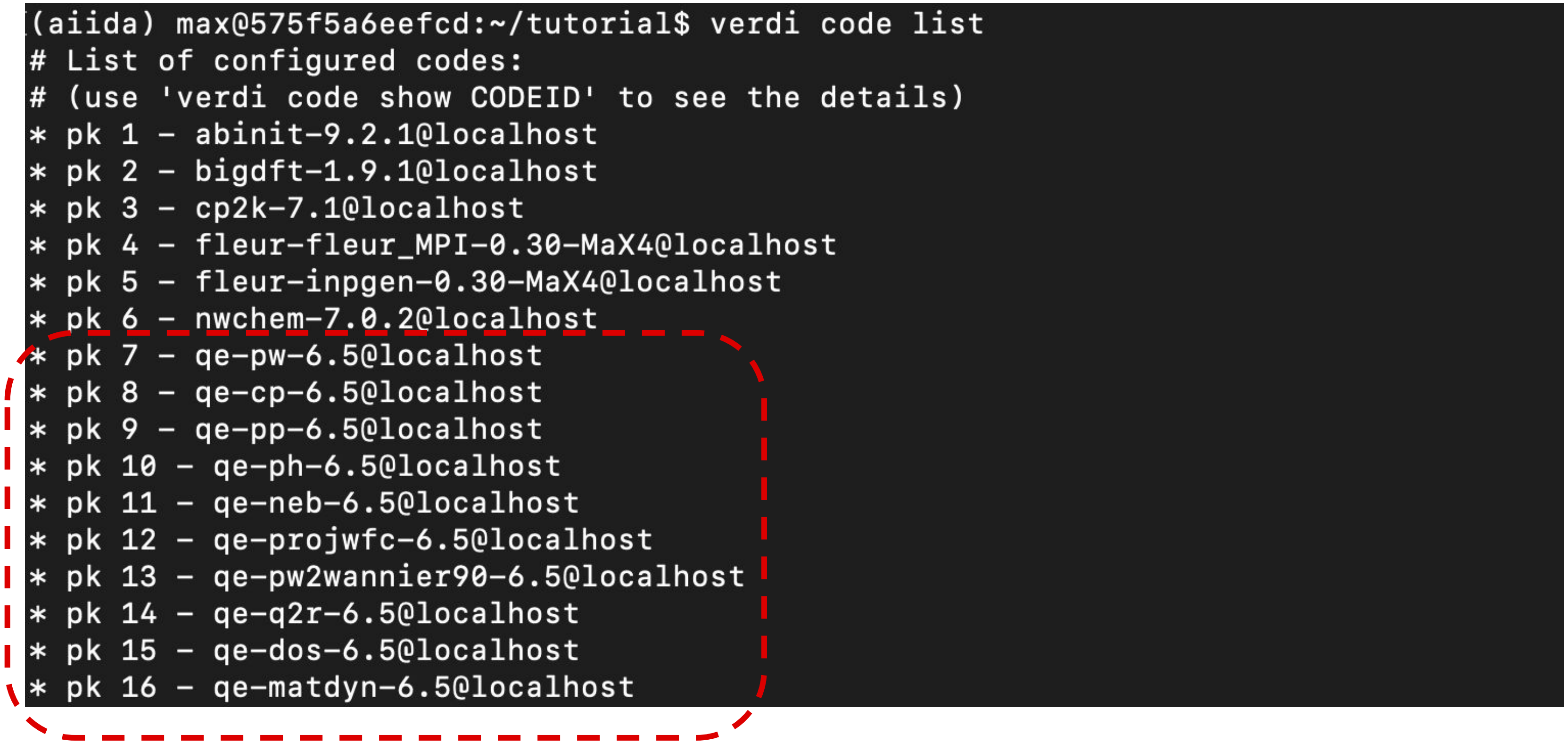
Short Demonstrations

Quantum ESPRESSO introductory tutorial	Wannier90: “Virtual Edition” 2020 tutorial

AiiDA Tutorials $\longrightarrow$ https://aiida-tutorials.readthedocs.io/en/latest/

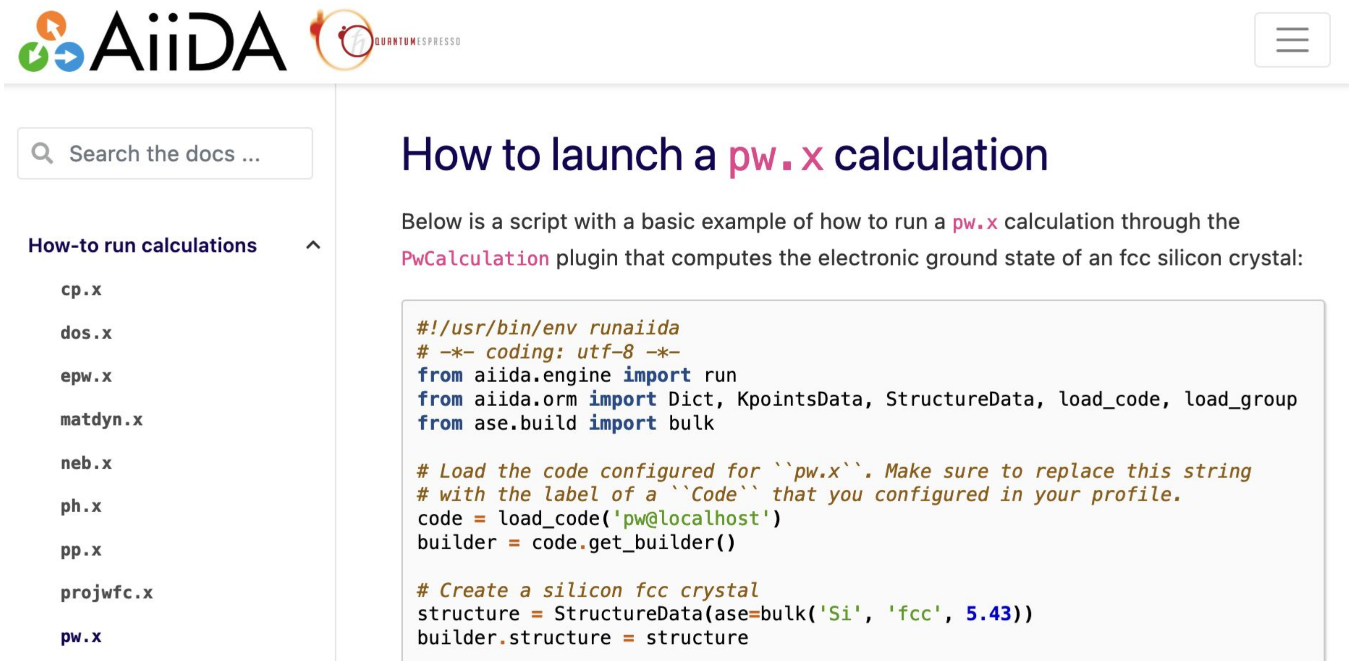
USAGE EXAMPLE

The virtual machine comes with some codes already set up…



```
(aiida) max@575f5a6eefcd:~/tutorial$ verdi code list
# List of configured codes:
# (use 'verdi code show CODEID' to see the details)
* pk 1 - abinit-9.2.1@localhost
* pk 2 - bigdft-1.9.1@localhost
* pk 3 - cp2k-7.1@localhost
* pk 4 - fleur-fleur_MPI-0.30-MaX4@localhost
* pk 5 - fleur-inpgen-0.30-MaX4@localhost
* pk 6 - nwchem-7.0.2@localhost
* pk 7 - qe-pw-6.5@localhost
* pk 8 - qe-cp-6.5@localhost
* pk 9 - qe-pp-6.5@localhost
* pk 10 - qe-ph-6.5@localhost
* pk 11 - qe-neb-6.5@localhost
* pk 12 - qe-projwfc-6.5@localhost
* pk 13 - qe-pw2wannier90-6.5@localhost
* pk 14 - qe-q2r-6.5@localhost
* pk 15 - qe-dos-6.5@localhost
* pk 16 - qe-matdyn-6.5@localhost
```

USAGE EXAMPLE



How to launch a pw.x calculation

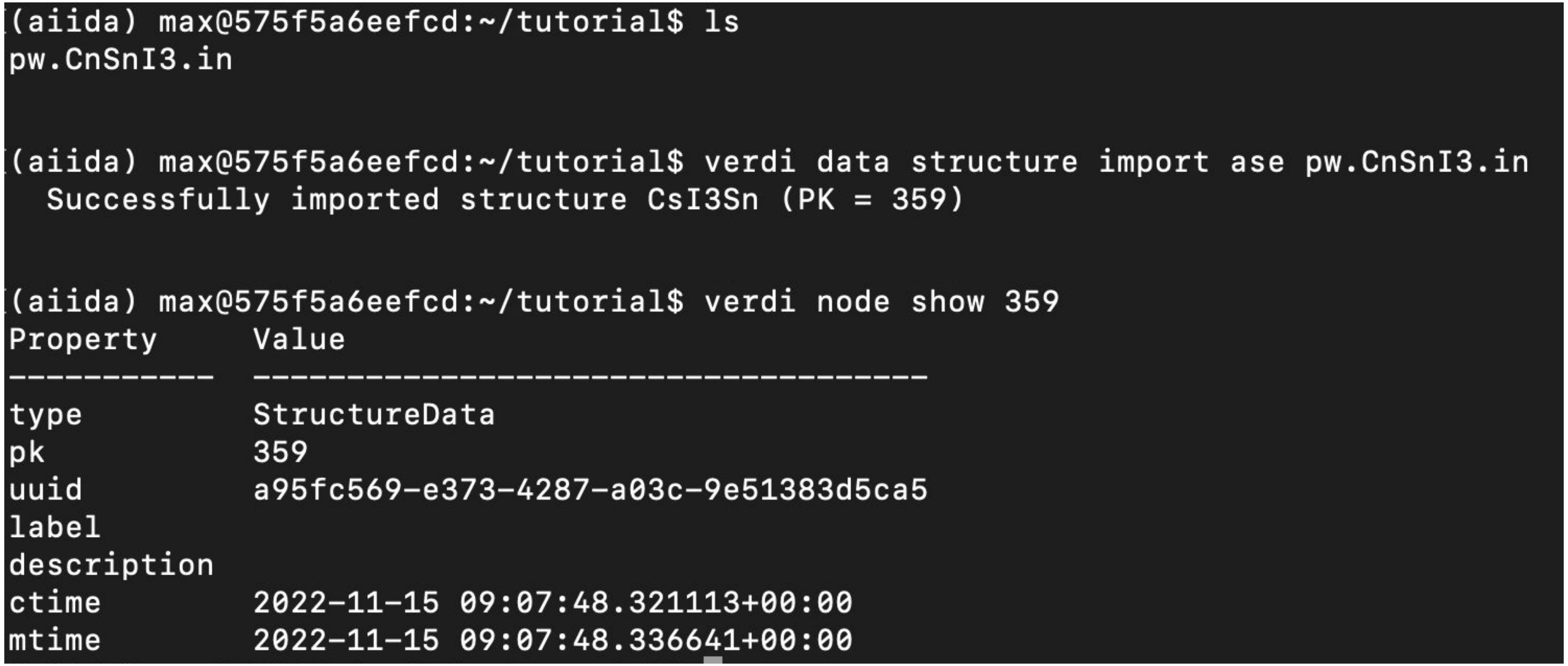
Below is a script with a basic example of how to run a pw.x calculation through the PwCalculation plugin that computes the electronic ground state of an fcc silicon crystal:

```
#!/usr/bin/env runaiida
# -*- coding: utf-8 -*-
from aiida.engine import run
from aiida.orm import Dict, KpointsData, StructureData, load_code, load_group
from ase.build import bulk

# Load the code configured for ``pw.x``. Make sure to replace this string
# with the label of a ``Code`` that you configured in your profile.
code = load_code('pw@localhost')
builder = code.get_builder()

# Create a silicon fcc crystal
structure = StructureData(ase=bulk('Si', 'fcc', 5.43))
builder.structure = structure
```

More information → https://aiida-quantumespresso.readthedocs.io

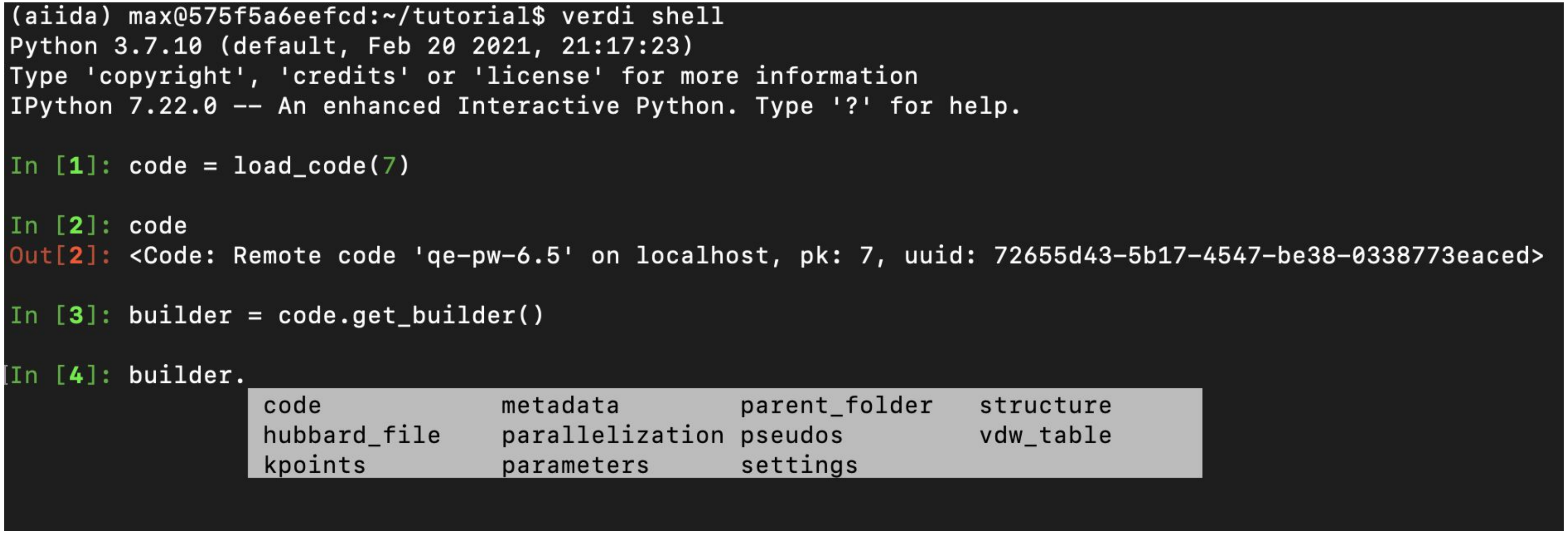
USAGE EXAMPLE

The structures we want to use need to be “uploaded” into AiiDA as a node

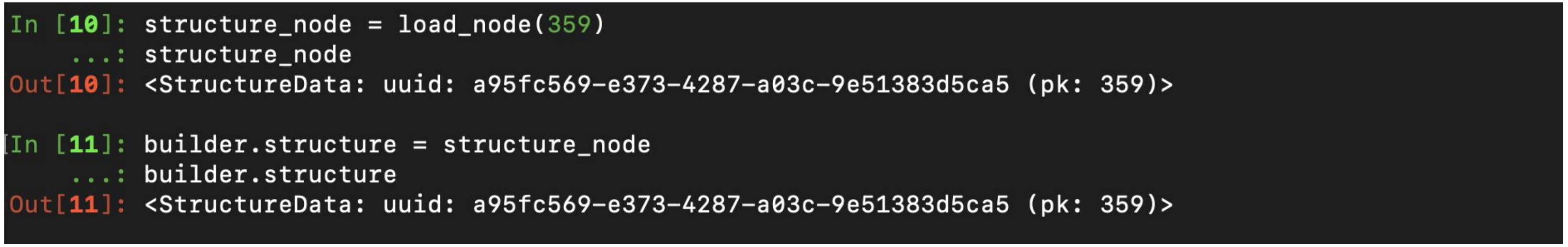


```
(aiida) max@575f5a6eefcd:~/tutorial$ ls
pw.CnSnI3.in


(aiida) max@575f5a6eefcd:~/tutorial$ verdi data structure import ase pw.CnSnI3.in
  Successfully imported structure CsI3Sn (PK = 359)


(aiida) max@575f5a6eefcd:~/tutorial$ verdi node show 359
Property     Value
-----------  ------------------------------------
type         StructureData
pk           359
uuid         a95fc569-e373-4287-a03c-9e51383d5ca5
label
description
ctime        2022-11-15 09:07:48.321113+00:00
mtime        2022-11-15 09:07:48.336641+00:00
```

USAGE EXAMPLE



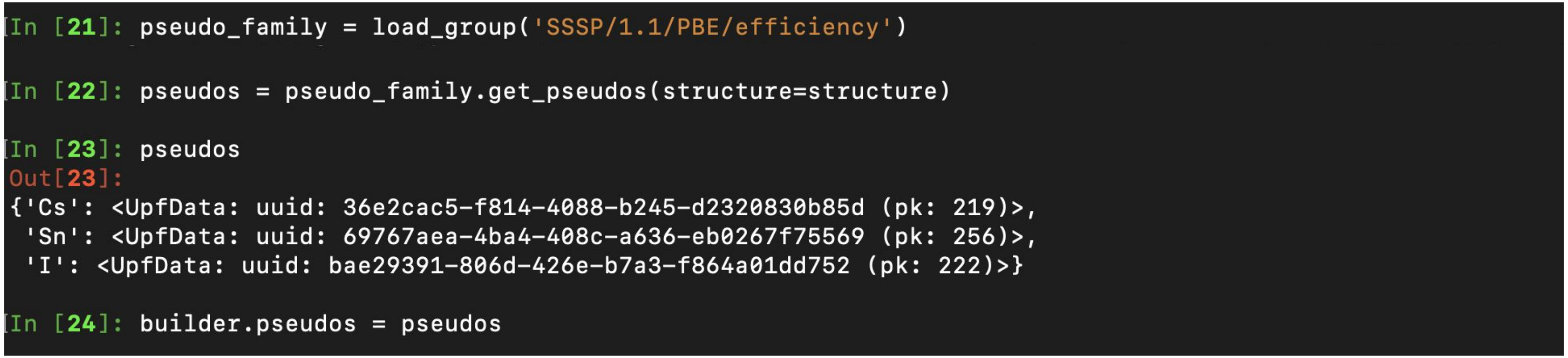
```
(aiida) max@575f5a6eefcd:~/tutorial$ verdi shell
Python 3.7.10 (default, Feb 20 2021, 21:17:23)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.22.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: code = load_code(7)

In [2]: code
Out[2]: <Code: Remote code 'qe-pw-6.5' on localhost, pk: 7, uuid: 72655d43-5b17-4547-be38-0338773eaced>

In [3]: builder = code.get_builder()

In [4]: builder.
         code             metadata          parent_folder   structure
         hubbard_file     parallelization   pseudos         vdw_table
         kpoints          parameters        settings
```



```
In [10]: structure_node = load_node(359)
    ...: structure_node
Out[10]: <StructureData: uuid: a95fc569-e373-4287-a03c-9e51383d5ca5 (pk: 359)>

In [11]: builder.structure = structure_node
    ...: builder.structure
Out[11]: <StructureData: uuid: a95fc569-e373-4287-a03c-9e51383d5ca5 (pk: 359)>
```

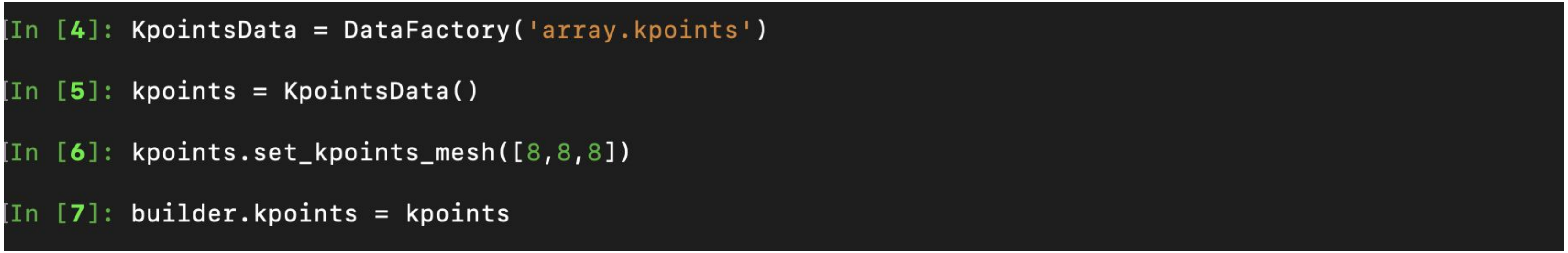
USAGE EXAMPLE

Selecting and setting up the pseudo-potentials…



```
In [21]: pseudo_family = load_group('SSSP/1.1/PBE/efficiency')

In [22]: pseudos = pseudo_family.get_pseudos(structure=structure)

In [23]: pseudos
Out[23]:
{'Cs': <UpfData: uuid: 36e2cac5-f814-4088-b245-d2320830b85d (pk: 219)>,
 'Sn': <UpfData: uuid: 69767aea-4ba4-408c-a636-eb0267f75569 (pk: 256)>,
 'I': <UpfData: uuid: bae29391-806d-426e-b7a3-f864a01dd752 (pk: 222)>}

In [24]: builder.pseudos = pseudos
```

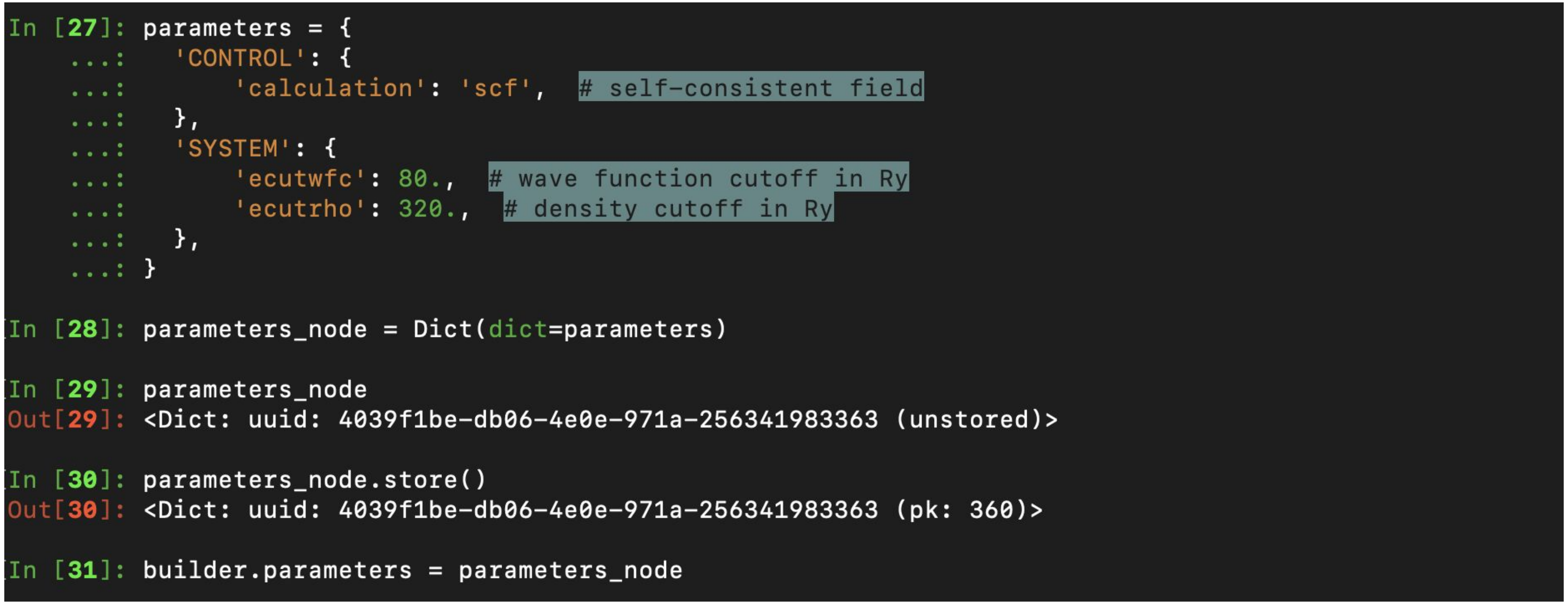
Generating and setting up the Kpoints…



```
In [4]: KpointsData = DataFactory('array.kpoints')

In [5]: kpoints = KpointsData()

In [6]: kpoints.set_kpoints_mesh([8,8,8])

In [7]: builder.kpoints = kpoints
```

USAGE EXAMPLE

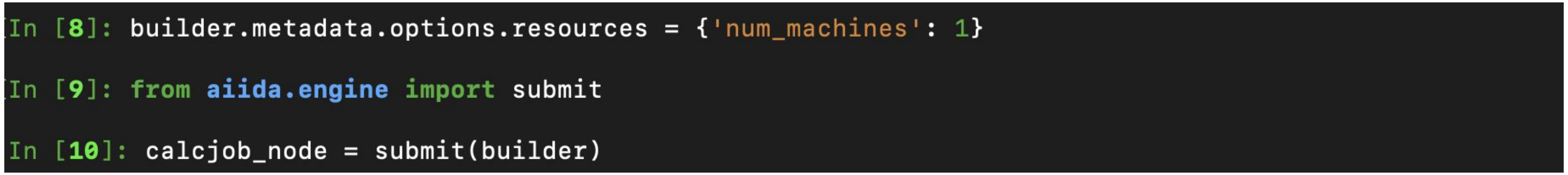
Generating and setting up the calculation parameters…



```
In [27]: parameters = {
    ...:     'CONTROL': {
    ...:         'calculation': 'scf',  # self-consistent field
    ...:     },
    ...:     'SYSTEM': {
    ...:         'ecutwfc': 80.,  # wave function cutoff in Ry
    ...:         'ecutrho': 320.,  # density cutoff in Ry
    ...:     },
    ...: }

In [28]: parameters_node = Dict(dict=parameters)

In [29]: parameters_node
Out[29]: <Dict: uuid: 4039f1be-db06-4e0e-971a-256341983363 (unstored)>

In [30]: parameters_node.store()
Out[30]: <Dict: uuid: 4039f1be-db06-4e0e-971a-256341983363 (pk: 360)>

In [31]: builder.parameters = parameters_node
```

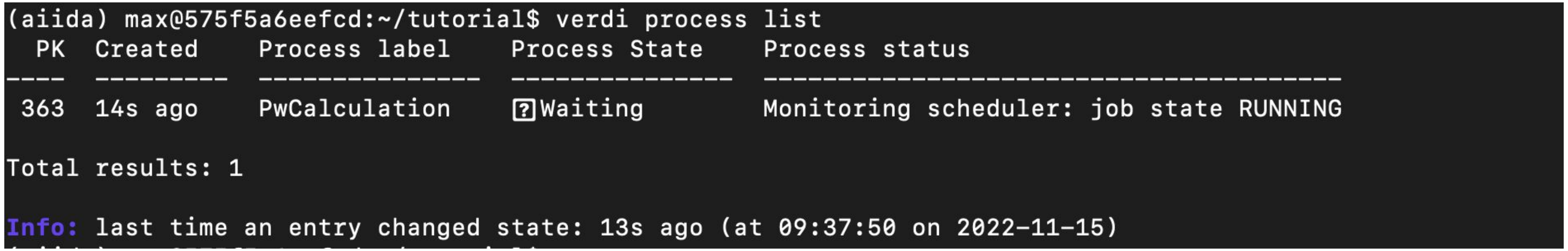
USAGE EXAMPLE

Setting up resources and it is ready to submit!



```
In [8]: builder.metadata.options.resources = {'num_machines': 1}

In [9]: from aiida.engine import submit

In [10]: calcjob_node = submit(builder)
```

This is running locally, but it is equivalent for submitting to a cluster

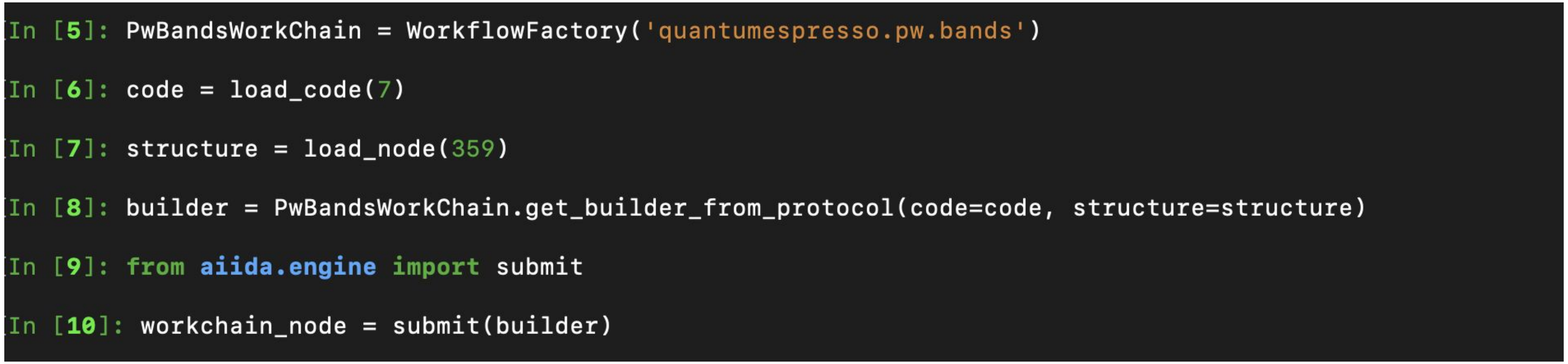


```
(aiida) max@575f5a6eefcd:~/tutorial$ verdi process list
  PK  Created     Process label     Process State     Process status
----  ----------  ----------------  ----------------  ---------------------------------------
 363  14s ago     PwCalculation     ⍰Waiting          Monitoring scheduler: job state RUNNING

Total results: 1

Info: last time an entry changed state: 13s ago (at 09:37:50 on 2022-11-15)
```

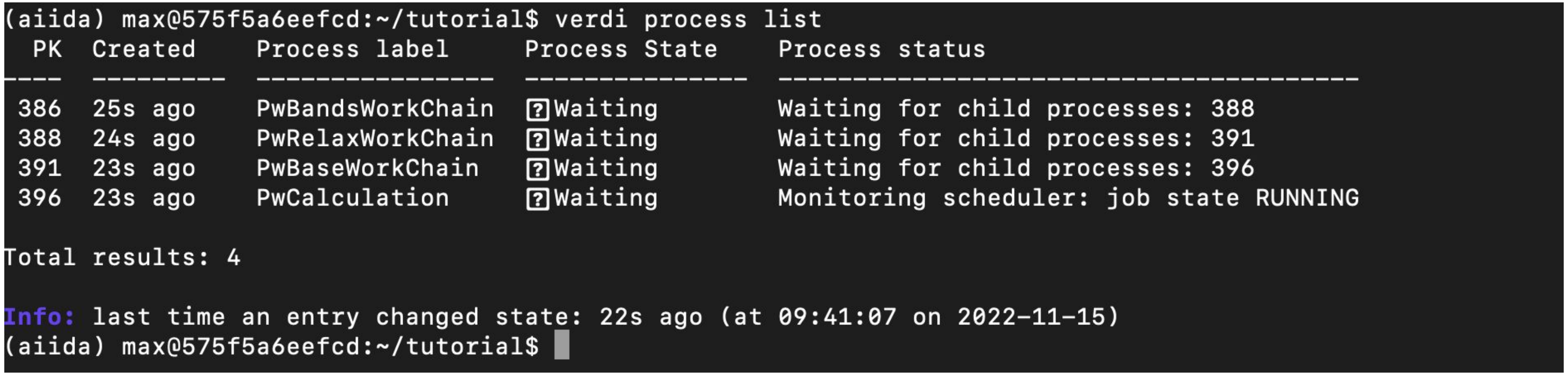
Seems more complicate to launch a single calculation but… it is scriptable!

USAGE EXAMPLE

Getting a band structure



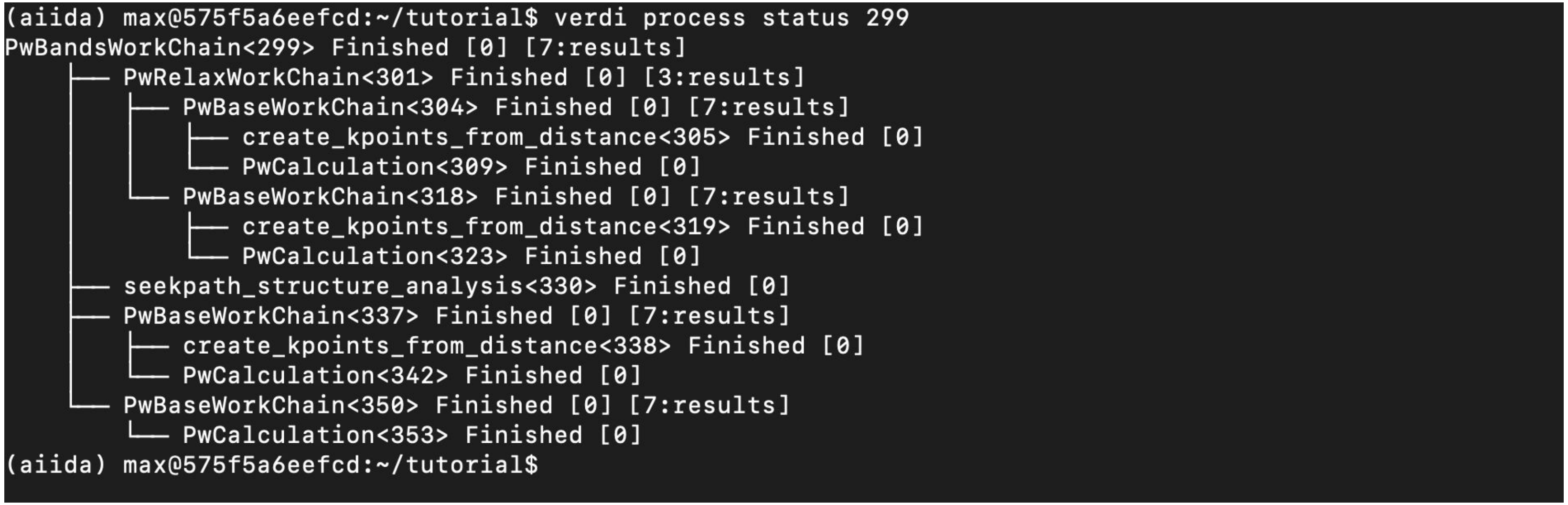
```
In [5]: PwBandsWorkChain = WorkflowFactory('quantumespresso.pw.bands')

In [6]: code = load_code(7)

In [7]: structure = load_node(359)

In [8]: builder = PwBandsWorkChain.get_builder_from_protocol(code=code, structure=structure)

In [9]: from aiida.engine import submit

In [10]: workchain_node = submit(builder)
```



```
(aiida) max@575f5a6eefcd:~/tutorial$ verdi process list
  PK  Created     Process label      Process State     Process status
----  ----------  -----------------  ----------------  ---------------------------------------
 386  25s ago     PwBandsWorkChain   ⍰Waiting          Waiting for child processes: 388
 388  24s ago     PwRelaxWorkChain   ⍰Waiting          Waiting for child processes: 391
 391  23s ago     PwBaseWorkChain    ⍰Waiting          Waiting for child processes: 396
 396  23s ago     PwCalculation      ⍰Waiting          Monitoring scheduler: job state RUNNING

Total results: 4

Info: last time an entry changed state: 22s ago (at 09:41:07 on 2022-11-15)
(aiida) max@575f5a6eefcd:~/tutorial$
```

USAGE EXAMPLE

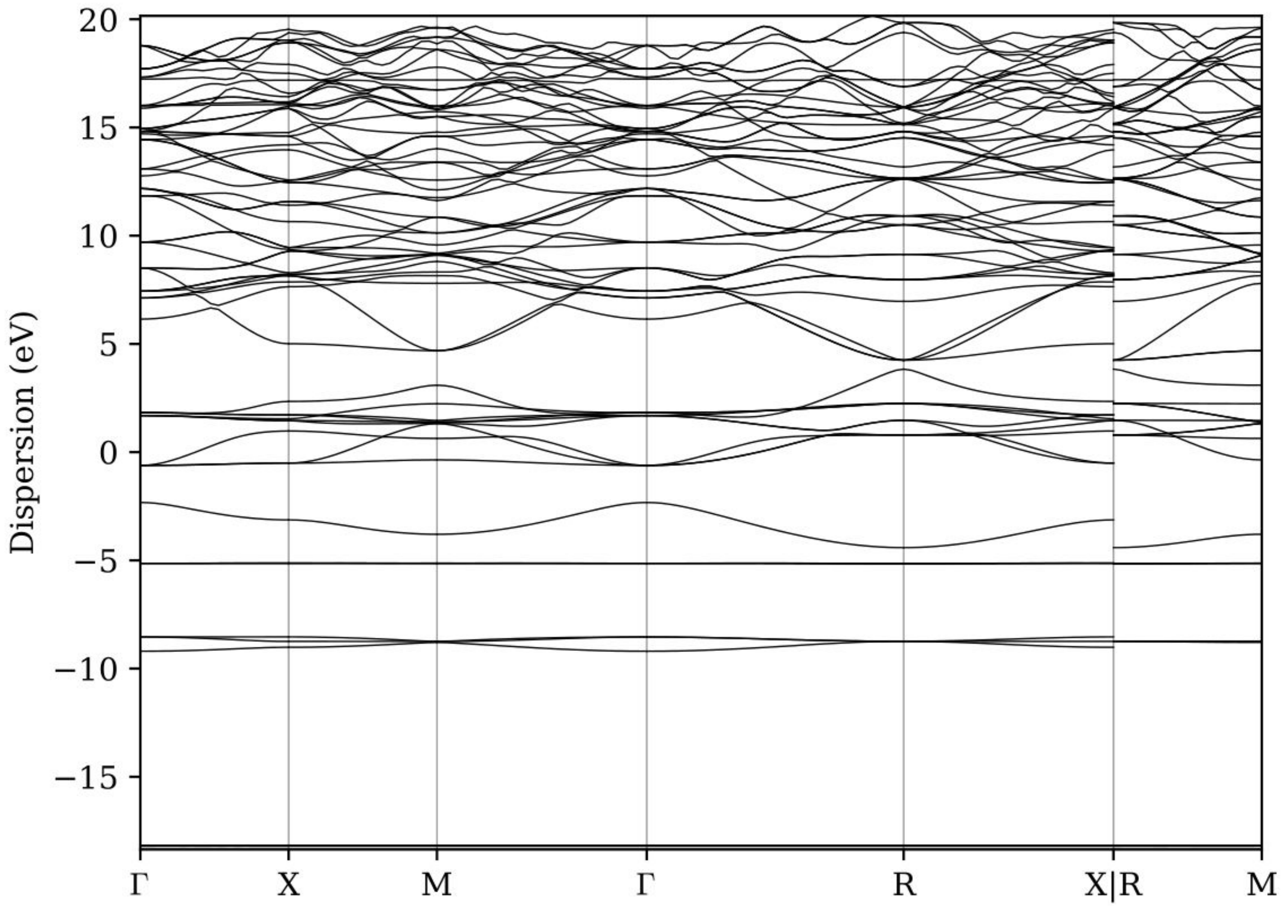
Automated complex procedure!



```
(aiida) max@575f5a6eefcd:~/tutorial$ verdi process status 299
PwBandsWorkChain<299> Finished [0] [7:results]
    ├── PwRelaxWorkChain<301> Finished [0] [3:results]
    │   ├── PwBaseWorkChain<304> Finished [0] [7:results]
    │   │   ├── create_kpoints_from_distance<305> Finished [0]
    │   │   └── PwCalculation<309> Finished [0]
    │   └── PwBaseWorkChain<318> Finished [0] [7:results]
    │       ├── create_kpoints_from_distance<319> Finished [0]
    │       └── PwCalculation<323> Finished [0]
    ├── seekpath_structure_analysis<330> Finished [0]
    ├── PwBaseWorkChain<337> Finished [0] [7:results]
    │   ├── create_kpoints_from_distance<338> Finished [0]
    │   └── PwCalculation<342> Finished [0]
    └── PwBaseWorkChain<350> Finished [0] [7:results]
        └── PwCalculation<353> Finished [0]
(aiida) max@575f5a6eefcd:~/tutorial$
```

Note: we submitted workchain pk = 386 but this one with pk = 299 is an older one that already finished.

USAGE EXAMPLE



AiiDA COMMON WORKFLOWS

Objective: To have a toolkit of *modular* turn-key solutions that can be used by non-experts to obtain physical properties *using their available / preferred software* without needing to manually tweak any parameter.

AUTOMATED SETUPS
(kpoints, basis sets, cutoffs)

ROBUSTNESS
(Both from the orchestrator and the underlying codes)

AiiDA COMMON WORKFLOWS

This task required the coordinated effort of the developers of all the different plugins and simulation codes involved in the project.

- Analyze and study the commonalities between codes.
- Distill the “essence” of the methods.
- Agree on common set of inputs (structure, protocol)
- How to translate abstract inputs (“protocol”) for each code to obtain compatible results / outputs.

Abinit
Guido Petretto
Samuel Poncé
Austin Zadoks
Gian-Marco Rignanese

BigDFT
Augustin Degomme
Luigi Genovese

CASTEP
Bonan Zhu

CP2K
Aliaksandr V. Yakutovich
Berend Smit

VASP
Espen Flage-Larsen

AiiDA
Sebastiaan P. Huber
Giovanni Pizzi
Emanuele Bosoni
Nicola Marzari
Dominik Gresch
Christopher J. Sewell
Martin Uhrin

QUANTUMESPRESSO
Marnik Bercx

www.flapw.de
Fleur
Jens Bröder
Vasily Tseplyaev
Daniel Wortmann

siesta
Vladimir Dikan
Alberto Garcia

ORCA
Pezhman Zarabadi-Poor

NWCHEM
HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE
Conrad Johnston

Kristjan Eimre

AiiDA COMMON WORKFLOWS

```
from aiida.plugins import WorkflowFactory, from aiida.engine import submit

CommonRelaxWorkChain = WorkflowFactory('common_workflows.relax.siesta')
structure = ...
engs = {
    'relax': {
        'code': 'siesta@marenostrum',
        'options': {'resources': {'num_machines': 1}, 'max_wallclock_seconds': 3600}
       }
}

inp_gen = CommonRelaxWorkChain.get_input_generator()
builder = inp_gen.get_builder(structure=structure,engines=engs,protocol='moderate')
(...)
submit(builder)
```

```
$ aiida-common-workflows launch eos siesta --structure=Al --protocol=precise
```

https://github.com/aiidateam/aiida-common-workflows/

AiiDA COMMON WORKFLOWS

Already working and applied to perform a series of comparisons between the performance and accuracy of results obtained with the different codes.

Improvement in the pseudos!

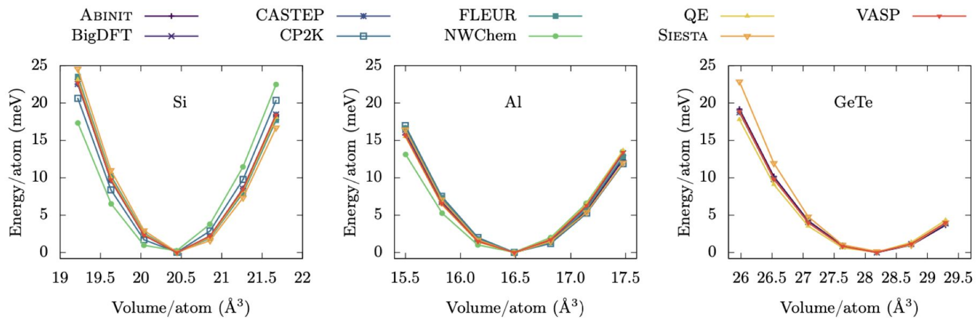
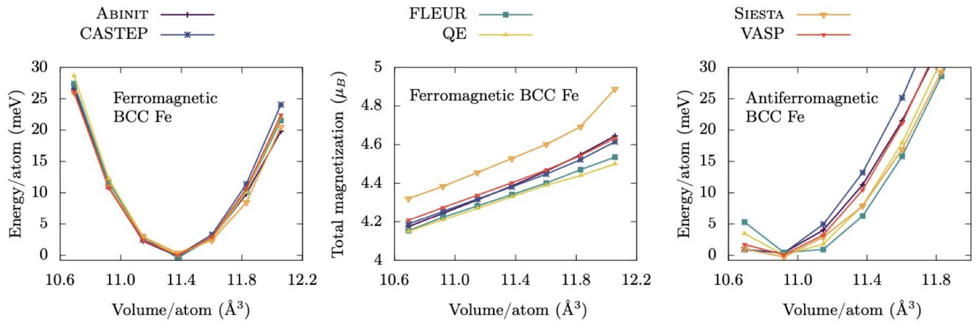
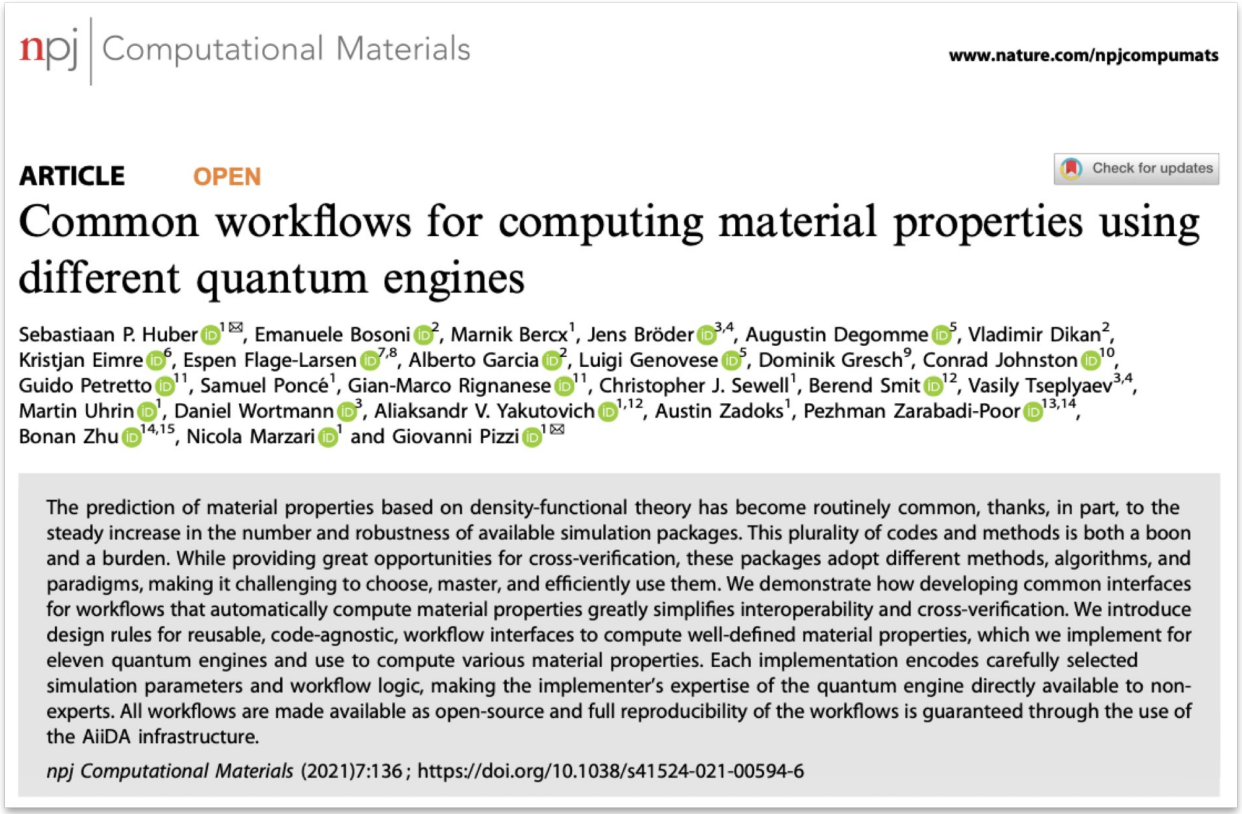


Figure 7. **EOS for Si, Al and GeTe.** Results obtained with the code-agnostic `EquationOfStateWorkflow`. For each code, the energy is shifted to set the minimum energy to zero. The EOS has been computed with all codes discussed in this work, except ORCA and Gaussian, which are mainly specialized for non-periodic systems. In addition, for GeTe, results are missing for BigDFT, CP2K, FLEUR and NWChem (see Table II in the Supplementary Information for more details). The label QE stands for QUANTUM ESPRESSO.



AiiDA COMMON WORKFLOWS



npj Computational Materials www.nature.com/npjcompumats

ARTICLE OPEN

Check for updates

Common workflows for computing material properties using different quantum engines

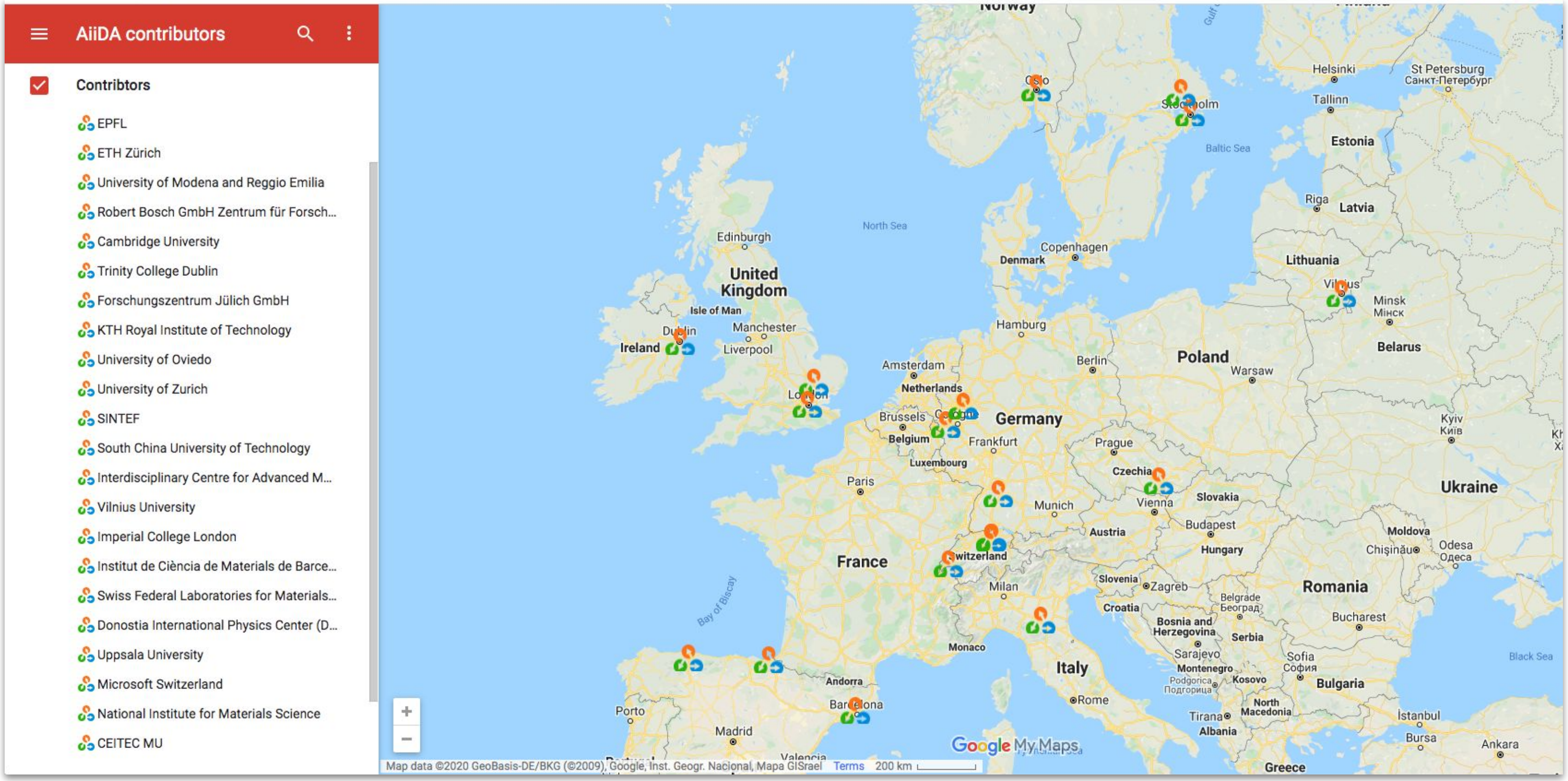
Sebastiaan P. Huber[1✉], Emanuele Bosoni[2], Marnik Bercx[1], Jens Bröder[3,4], Augustin Degomme[5], Vladimir Dikan[2], Kristjan Eimre[6], Espen Flage-Larsen[7,8], Alberto Garcia[2], Luigi Genovese[5], Dominik Gresch[9], Conrad Johnston[10], Guido Petretto[11], Samuel Poncé[1], Gian-Marco Rignanese[11], Christopher J. Sewell[1], Berend Smit[12], Vasily Tseplyaev[3,4], Martin Uhrin[1], Daniel Wortmann[3], Aliaksandr V. Yakutovich[1,12], Austin Zadoks[1], Pezhman Zarabadi-Poor[13,14], Bonan Zhu[14,15], Nicola Marzari[1] and Giovanni Pizzi[1✉]

The prediction of material properties based on density-functional theory has become routinely common, thanks, in part, to the steady increase in the number and robustness of available simulation packages. This plurality of codes and methods is both a boon and a burden. While providing great opportunities for cross-verification, these packages adopt different methods, algorithms, and paradigms, making it challenging to choose, master, and efficiently use them. We demonstrate how developing common interfaces for workflows that automatically compute material properties greatly simplifies interoperability and cross-verification. We introduce design rules for reusable, code-agnostic, workflow interfaces to compute well-defined material properties, which we implement for eleven quantum engines and use to compute various material properties. Each implementation encodes carefully selected simulation parameters and workflow logic, making the implementer's expertise of the quantum engine directly available to non-experts. All workflows are made available as open-source and full reproducibility of the workflows is guaranteed through the use of the AiiDA infrastructure.

npj Computational Materials (2021)7:136 ; https://doi.org/10.1038/s41524-021-00594-6

S. P. Huber et al., npj Comput. Mater. 7, 136 (2021)

AiiDA COMMUNITY



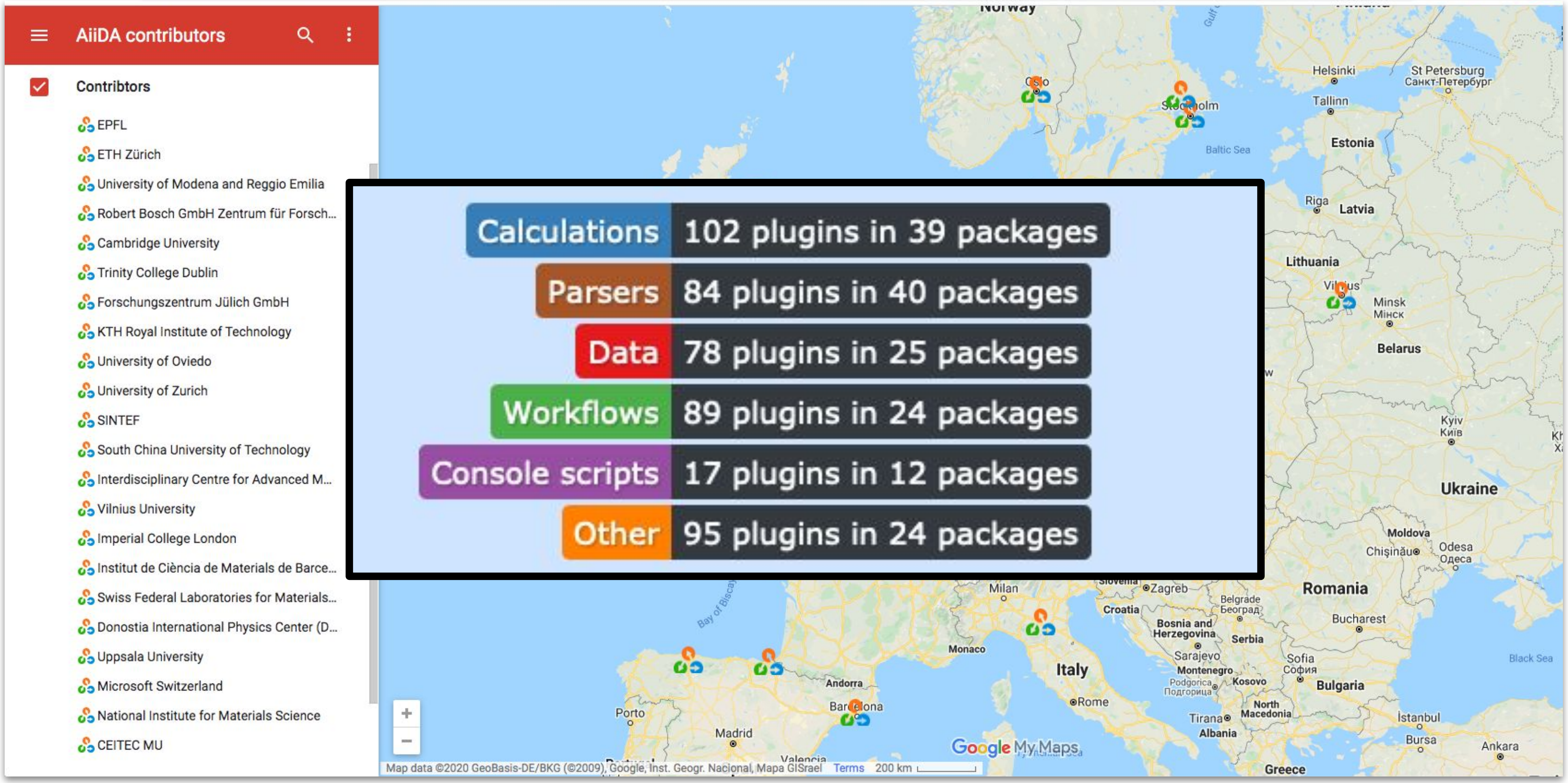
AiiDA contributors

Contribtors

- EPFL
- ETH Zürich
- University of Modena and Reggio Emilia
- Robert Bosch GmbH Zentrum für Forsch...
- Cambridge University
- Trinity College Dublin
- Forschungszentrum Jülich GmbH
- KTH Royal Institute of Technology
- University of Oviedo
- University of Zurich
- SINTEF
- South China University of Technology
- Interdisciplinary Centre for Advanced M...
- Vilnius University
- Imperial College London
- Institut de Ciència de Materials de Barce...
- Swiss Federal Laboratories for Materials...
- Donostia International Physics Center (D...
- Uppsala University
- Microsoft Switzerland
- National Institute for Materials Science
- CEITEC MU

Map data ©2020 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional, Mapa GISrael Terms 200 km

AiiDA COMMUNITY



AiiDA contributors

Contribtors

- EPFL
- ETH Zürich
- University of Modena and Reggio Emilia
- Robert Bosch GmbH Zentrum für Forsch...
- Cambridge University
- Trinity College Dublin
- Forschungszentrum Jülich GmbH
- KTH Royal Institute of Technology
- University of Oviedo
- University of Zurich
- SINTEF
- South China University of Technology
- Interdisciplinary Centre for Advanced M...
- Vilnius University
- Imperial College London
- Institut de Ciència de Materials de Barce...
- Swiss Federal Laboratories for Materials...
- Donostia International Physics Center (D...
- Uppsala University
- Microsoft Switzerland
- National Institute for Materials Science
- CEITEC MU

Calculations	102 plugins in 39 packages
Parsers	84 plugins in 40 packages
Data	78 plugins in 25 packages
Workflows	89 plugins in 24 packages
Console scripts	17 plugins in 12 packages
Other	95 plugins in 24 packages

Map data ©2020 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional, Mapa GISrael Terms 200 km

ACKNOWLEDGMENTS

MARVEL

MARVEL National Centre for Competency in Research

MaX

European Centre of Excellence MaX

EPFL

École Polytechnique Fédérale de Lausanne

FNSNF

SWISS NATIONAL SCIENCE FOUNDATION

swissuniversities

MarketPlace

INTERSECT

erc European Research Council Established by the European Commission

European Research Council

pasc

Platform for Advanced Scientific Computing

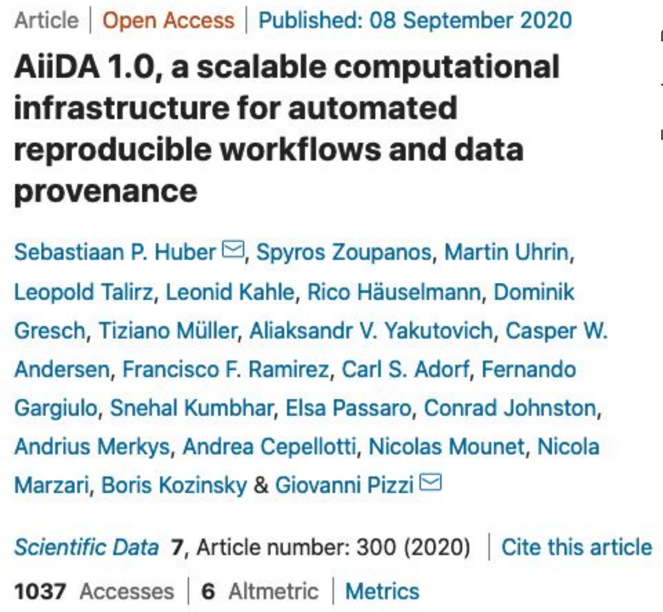
PRACE

PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE

nffa.eu nanoscience foundries & fine analysis

EMMC

The European Materials Modelling Council



Article | Open Access | Published: 08 September 2020

AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance

Sebastiaan P. Huber, Spyros Zoupanos, Martin Uhrin, Leopold Talirz, Leonid Kahle, Rico Häuselmann, Dominik Gresch, Tiziano Müller, Aliaksandr V. Yakutovich, Casper W. Andersen, Francisco F. Ramirez, Carl S. Adorf, Fernando Gargiulo, Snehal Kumbhar, Elsa Passaro, Conrad Johnston, Andrius Merkys, Andrea Cepellotti, Nicolas Mounet, Nicola Marzari, Boris Kozinsky & Giovanni Pizzi

Scientific Data **7**, Article number: 300 (2020) | Cite this article

1037 Accesses | **6** Altmetric | Metrics

We also acknowledge code contributions, bug fixes, documentation, and suggestions from:
Oscar Arbelaez, Michael Atambo, Valentin Bersier, Marco Borelli, Jocelyn Boullier, Jens Bröder, Ivano E. Castelli, Andrea Cepellotti, Keija Cui, Vladimir Dikan, Marco Dorigo, Y.-W. Fang, Fernando Gargiulo, Marco Gibertini, Davide Grassano, Conrad Johnston, Rico Häuselmann, Daniel Hollas, Eric Hontz, Jianxing Huang, Christoph Koch, Espen Flage Larsen, Boris Kozinsky, Ian Lee, Daniel Marchand, Antimo Marrazzo, Andrius Merkys, Nicolas Mounet, Simon Pintarelli, Gianluca Prandini, Philipp Rüßmann, Riccardo Sabatini, Ole Schütt, Philippe Schwaller, Chris Sewell, Andreas Stamminger, Atsushi Togo, Daniele Tomerini, Nicola Varini, Jason Yu, Austin Zadoks, Bonan Zhu and Mario Zic.

ONLINE RESOURCES AND CONTACT

AiiDA

WEBSITE

http://www.aiida.net

SOURCE CODE

github.com/aiidateam/aiida-core

DOCUMENTATION

https://aiida.readthedocs.io

CONTACT INFORMATION

francisco.ramirez@epfl.ch