

# Python ecosystem for scientific computing with ABINIT: challenges and opportunities

M. Giantomassi and the AbiPy group



Frejus, May 9, 2017



- Python package for:
  - ◆ generating input files automatically
  - ◆ post-processing results stored in netcdf files
  - ◆ creating and executing workflows (relaxations, phonons, GW, BSE...)
- Why Python?
  - ◆ relatively easy to use and to learn
  - ◆ can implement logic to drive high-performance/high-throughput jobs
  - ◆ great support for science (numpy, scipy, pandas, matplotlib...)
  - ◆ interactive environment (ipython, jupyter notebooks, GUIs)
  - ◆ pymatgen ecosystem and materials project database ...

# pymatgen

Computational Materials Science 68 (2013) 314–319



Contents lists available at SciVerse ScienceDirect

Computational Materials Science

journal homepage: [www.elsevier.com/locate/commatsci](http://www.elsevier.com/locate/commatsci)



Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis

Shyue Ping Ong <sup>a,\*</sup>, William Davidson Richards <sup>a</sup>, Anubhav Jain <sup>b</sup>, Geoffroy Hautier <sup>c</sup>, Michael Kocher <sup>b</sup>, Shreyas Cholia <sup>b</sup>, Dan Gunter <sup>b</sup>, Vincent L. Chevrier <sup>d</sup>, Kristin A. Persson <sup>b</sup>, Gerbrand Ceder <sup>a</sup>

- ◆ Classes for the representation of Molecules and Structures
- ◆ Structure manipulation
- ◆ CIF file and XYZ format support
- ◆ IO capabilities to manipulate many VASP and ABINIT input and output files
- ◆ Tools to generate and view compositional and grand canonical phase diagrams
- ◆ Electronic structure analyses
- ◆ Integration with the Materials Project Database (REST API)

## Table Of Contents

- [Getting Started](#)
- [Post-processing](#)
- [Calculations](#)
- [API](#)
- [Indices and tables](#)
- [License](#)

## This Page

[Show Source](#)

## Quick search

Release: 0.2.0  
Date: May 03, 2017

# Getting Started

- [Feature Overview](#)
- [Getting AbiPy](#)
- [What's new in abipy](#)

pip or conda packages

\$ abistruct.py  
\$ abiopen.py  
\$ abicomp.py  
\$ abirun.py  
\$ abidoc.py

# Post-processing

- [Command line tools](#)
- [plot Examples](#)

# Calculations

- [TaskManager](#)

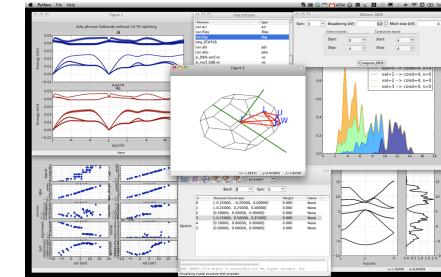
40 examples (electrons, fatbands, phonons, SCR.nc, WFK.nc, grunesein ...)

# API

- [API documentation](#)
- [The AbiPy Developers' Guide](#)

# Indices and tables

- [Index](#)
- [Module Index](#)
- [Search Page](#)



Explains how to connect AbiPy with Abinit

# License

# AbiPy Design

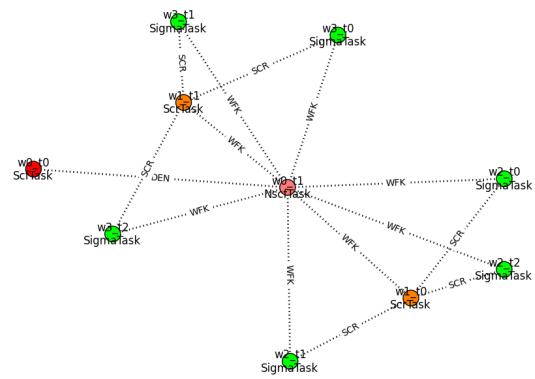
- AbiPy communicates with Abinit via yaml docs and netcdf files:
  - ◆ portable, can support “unconventional” machines e.g. BlueGeneQ
  - ◆ yaml “events” found in the logfile trigger handlers (error-handlers, restart ...)
  - ◆ yaml docs with autoparal configurations, independent perturbations ...
  - ◆ netcdf results + metadata used to implement post-processing/worflows
- Tight integration between Python and Abinit:
  - ◆ AbiPy invokes Abinit at run-time to get critical parameters
  - ◆ Programmatic interface to optimize/modify resources (mem, cpus, timelimit) at runtime to respond to software failure
  - ◆ AbiPy pass runtime parameters to Abinit e.g –time-limit

Two different workflow models (with/without database)

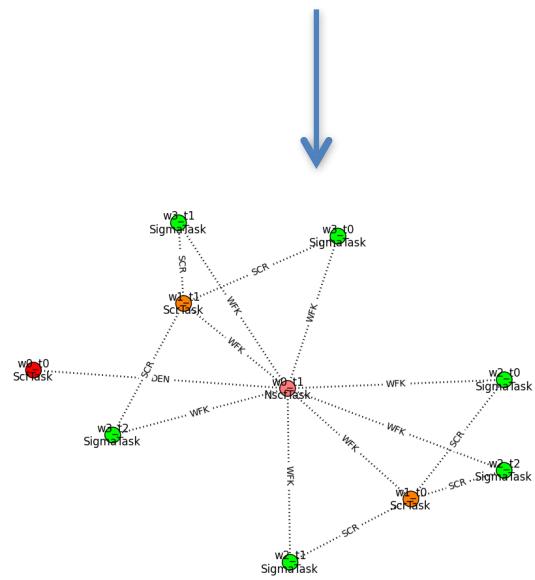
((( abipy )))



Flow generation

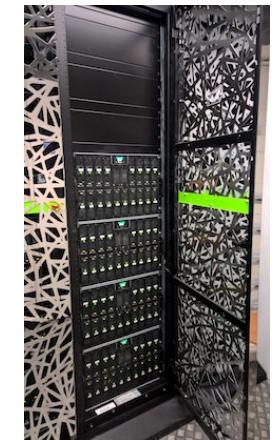


# abipy



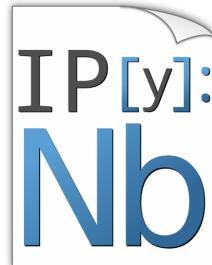
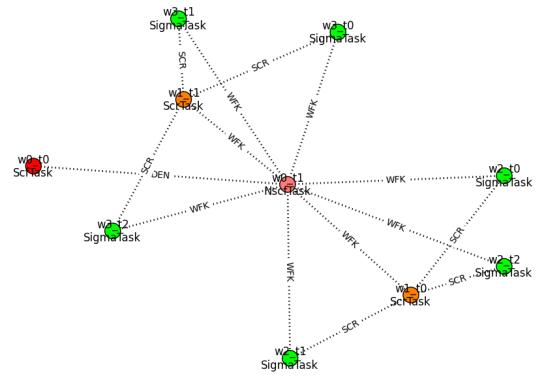
The logo for Abipy, featuring the word "abipy" in a bold, lowercase sans-serif font. To the left of the text are three curved orange-red lines of varying lengths, and to the right are three curved orange-red lines of varying lengths.

Scheduling the flow on the cluster  
Error handling  
Restart/increase resources ...

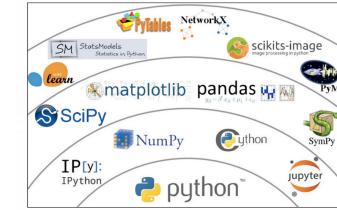


SLURM  
PBS/Torque  
Sun Grid Engine  
IBM LoadLeveler

 abipy

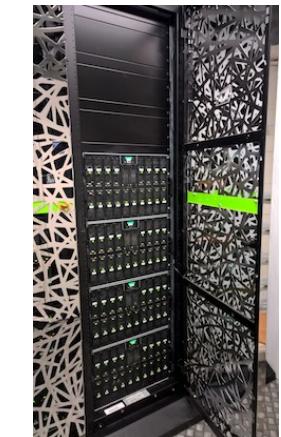


Interactive analysis  
and presentation of  
results

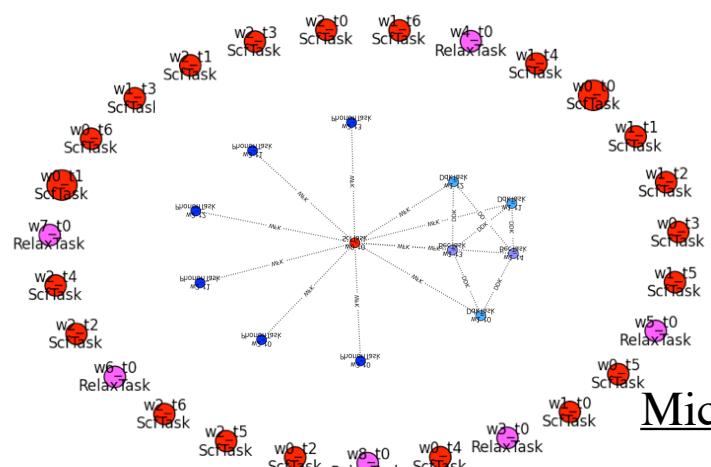


Storing results

 abipy

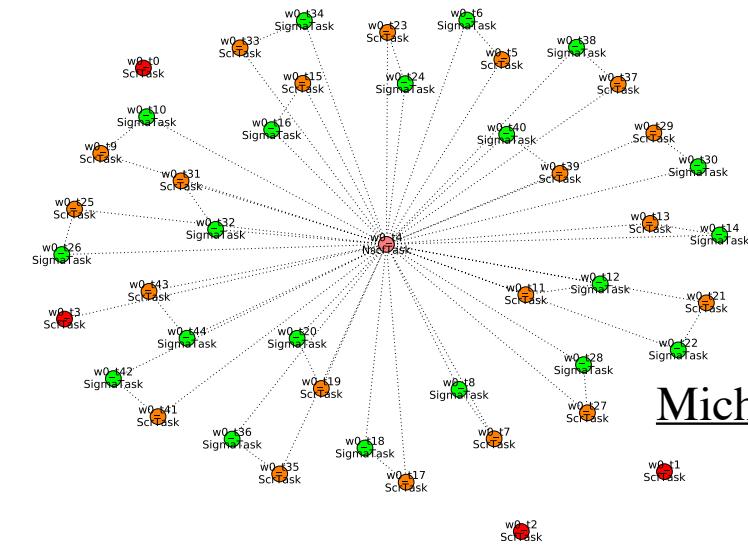


## PseudoDojo



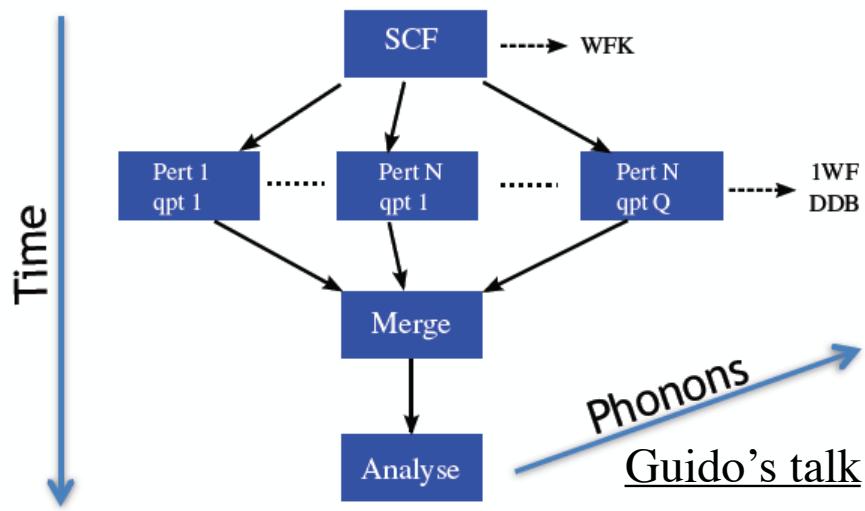
Michiel's talk

## G<sub>0</sub>W<sub>0</sub> for ~100 solids

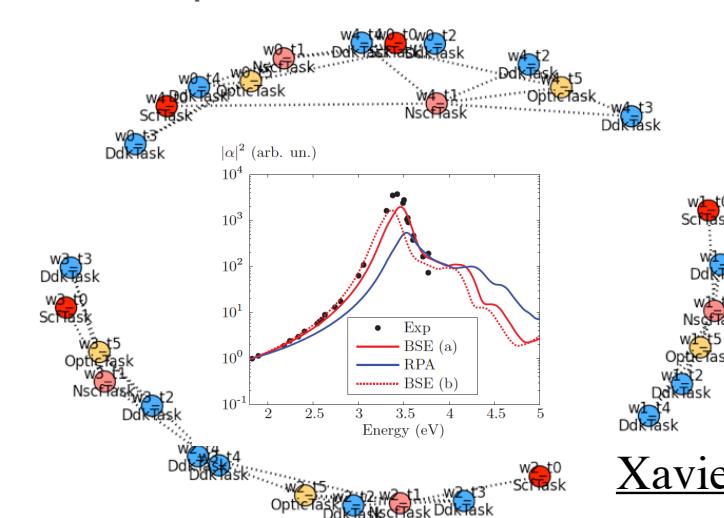


Michiel's talk

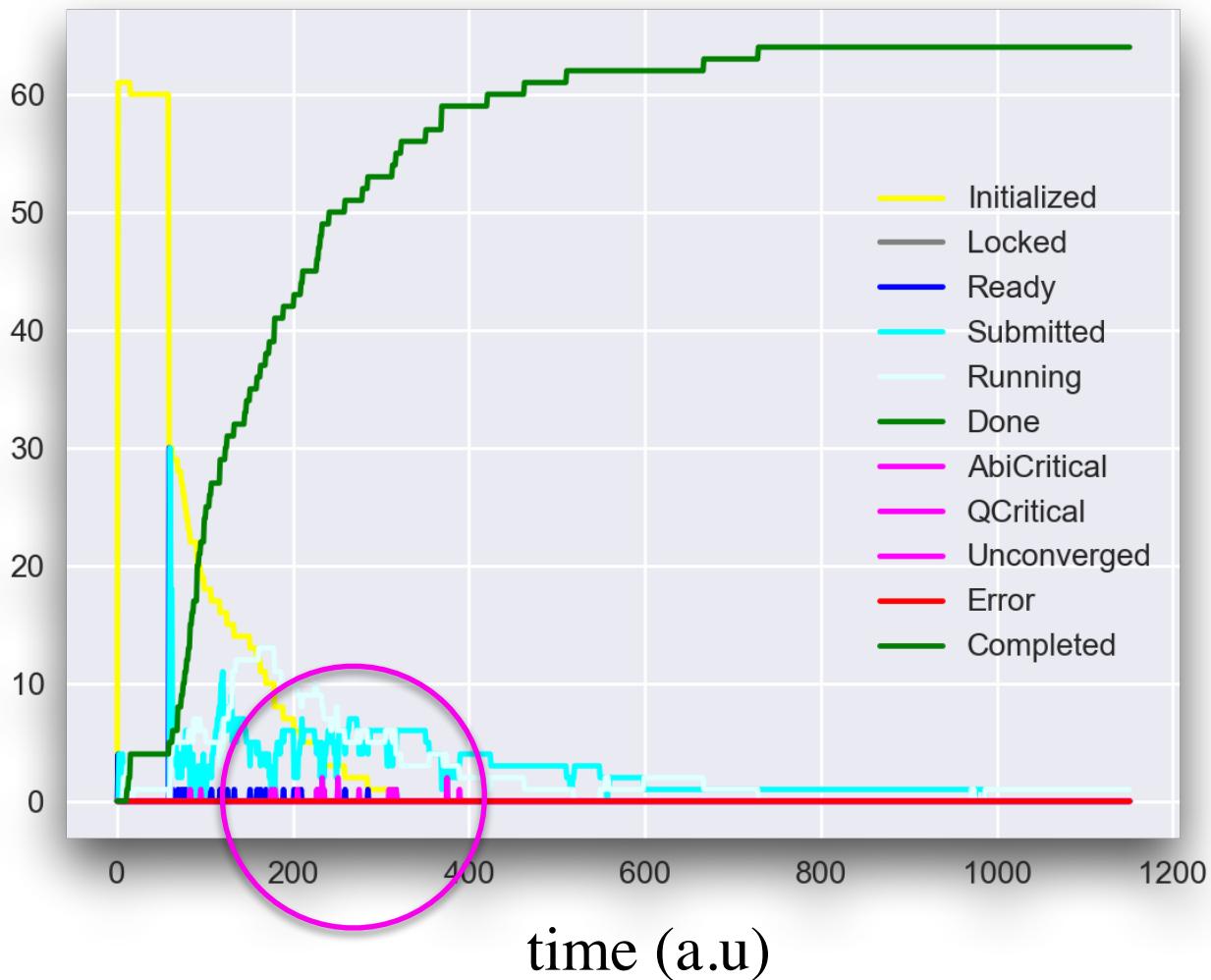
## HT DFPT



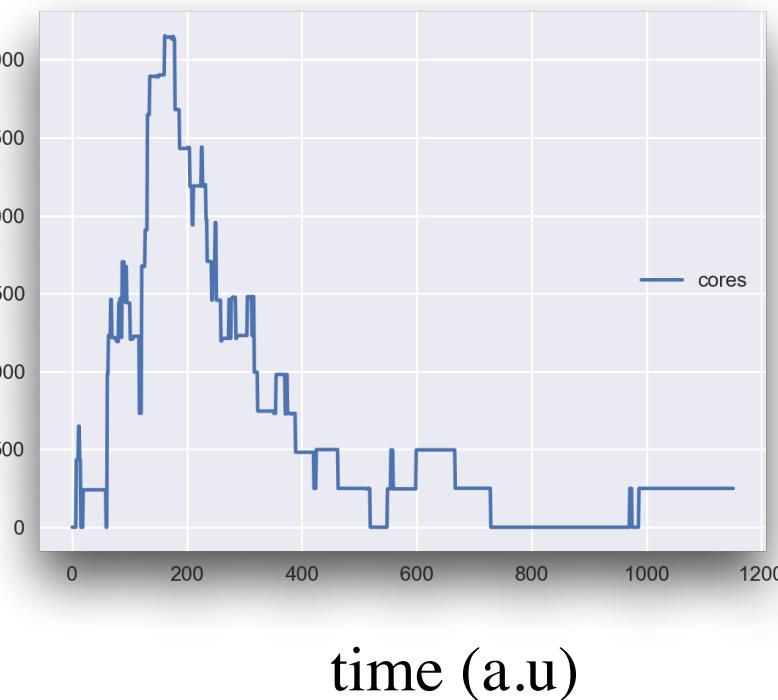
## Raman with frozen phonon and RPA/BSE



# Evolution of tasks in a single GW flow

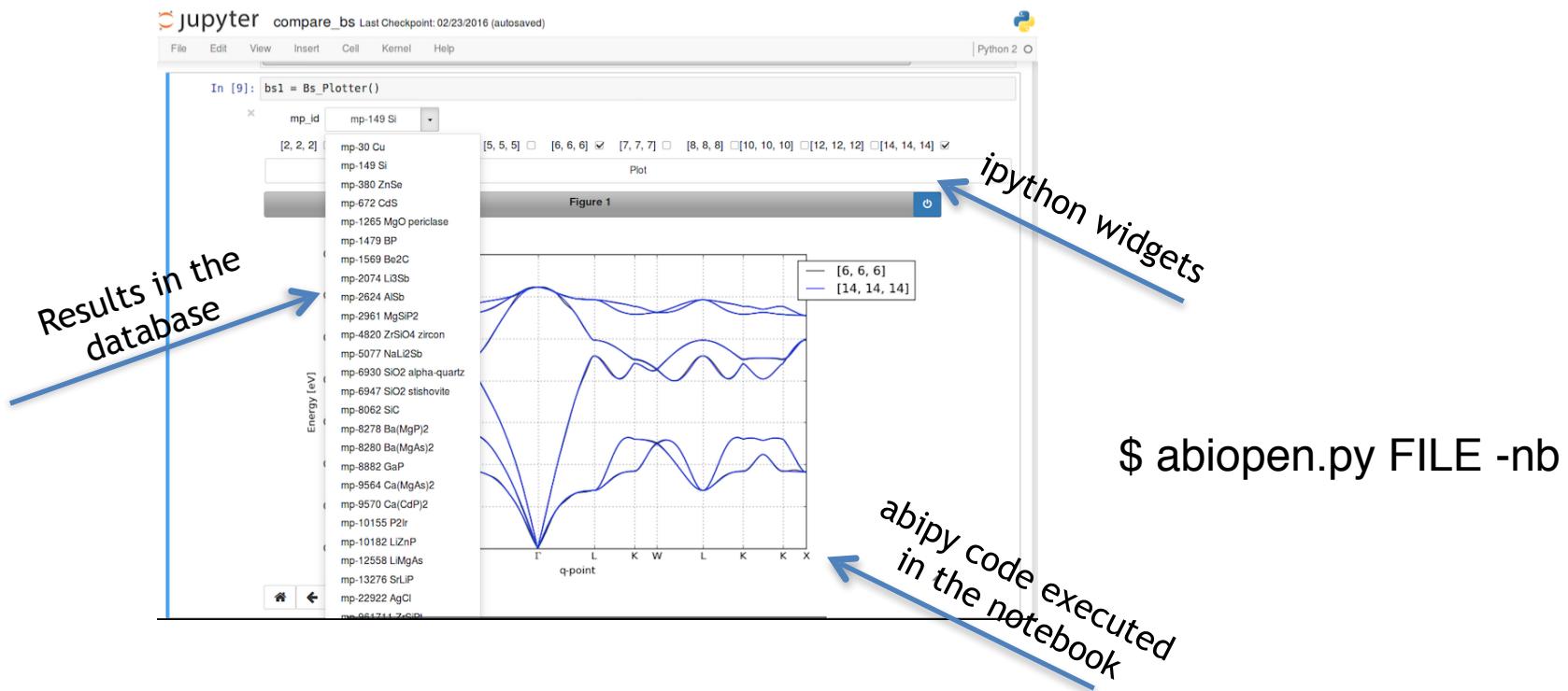


Tot #cpus used by a single GW flow



# jupyter

- Web application to create and share documents with live code, equations, visualizations and explanatory text
- Can produce rich output such as images, videos, LaTeX, and slides for talks!
- Make data analysis easier to share and reproduce. Keep detailed records of work
- Interactive widgets can be used to manipulate and visualize data in realtime
- Some researchers are even using notebooks as supplementary material (e.g. pseudo-dojo paper)

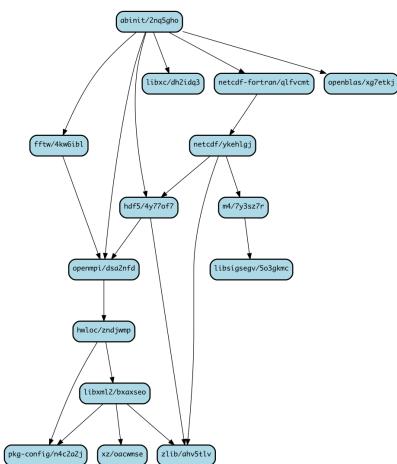


# Challenges Issues

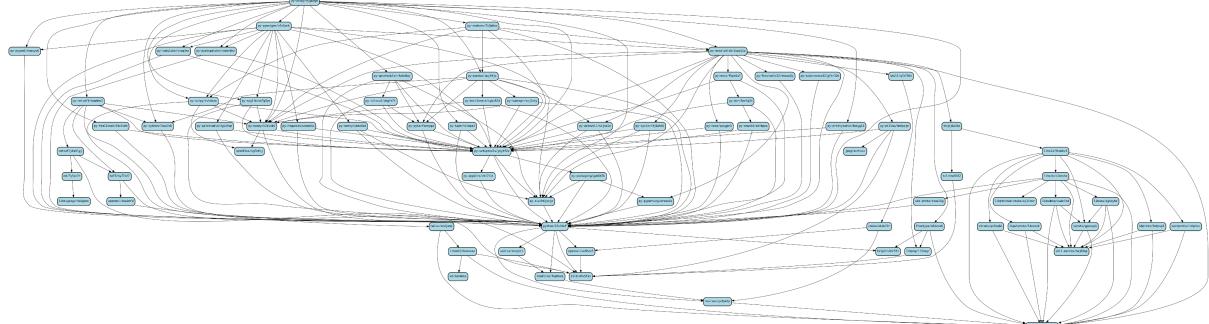


- Improve the accessibility of these tools to researchers and end-users
- Test/validate the Abinit/AbiPy interface and the workflows to guarantee:
  - i) Backward compatibility, software quality, scientific reproducibility
  - ii) That new developments don't break what is already available
- Make the python ecosystem coexist with the HPC software stack

Abinit (HPC libraries)



Abipy + Pymatgen  
(usually gcc-toolchain)



Vendor-provided HPC libraries are not always compatible with PyData stack  
(unless you love `configure && make` all the libraries you need!)

From: <https://computation.llnl.gov/newsroom/flexible-package-manager-hpc-software>

## A Flexible Package Manager for HPC Software

FRIDAY, FEBRUARY 19, 2016

High-performance computing (HPC) software is becoming **increasingly complex**, quickly outpacing the capabilities of existing software management tools. To support scientific applications, system administrators and developers frequently build, install, and support **different configurations of math and physics libraries and other software** on the same HPC system. Those applications are later rebuilt to fix bugs and to support new operating system versions, compilers (special programs that turn programming language into code that can be used by a computer's processor), message passing interface (MPI) versions, and other dependency libraries. **Forcing all application teams to use a single, standard software stack [...] is infeasible, but managing many software configurations and versions for all users on a single system is a time-consuming task for supercomputing staff.**

Existing tools can automate portions of this process, but they either cannot manage installation of multiple versions and configurations, or they require numerous configuration files for each software version, leading to organizational and maintenance issues.



- Handle multiple versions via modules
- Module load abinit\_version\_gcc-toolchain

<https://github.com/hpcugent/easybuild>



<https://github.com/llnl/spack>

- Uses RPATH linking (each package knows where to find its dependencies)
- \$ spack activate abinit (abinit+hdf5@8.2.2)

- ★ Support for multiple versions and configurations of software
- ★ Installing a new version does not break existing installations, many configurations can coexist

- Recipes for Abinit/AbiPy are already available
- The Abinit test farm could use these tools to extend the set of tests and/or keep an history of the different builds
- What about desktop computers? What if I want to have full control of my software stack without having to install these HPC tools?



## ANACONDA

- Package manager + environment manager (conda)
- Python distribution and collection of over 720 open source packages

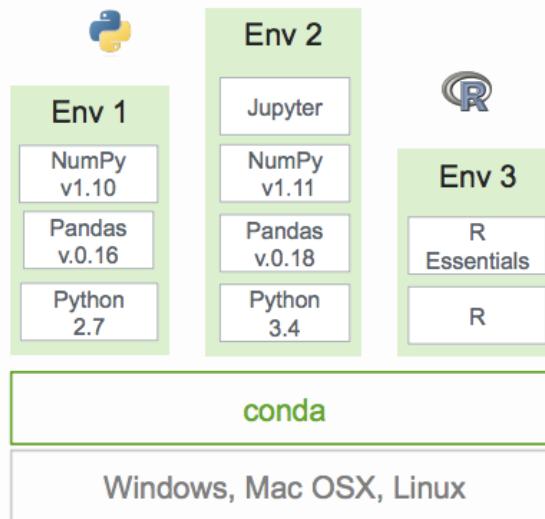
## CONDA

- Created for Python programs, but can package and distribute any software
- Keeps track of the dependencies between packages and platforms
- A conda environment is a directory that contains a specific collection of conda packages that you have installed.
- You can easily activate or deactivate (switch between) these environments.
- Conda packages are downloaded from remote channels, which are simply URLs to directories containing conda packages

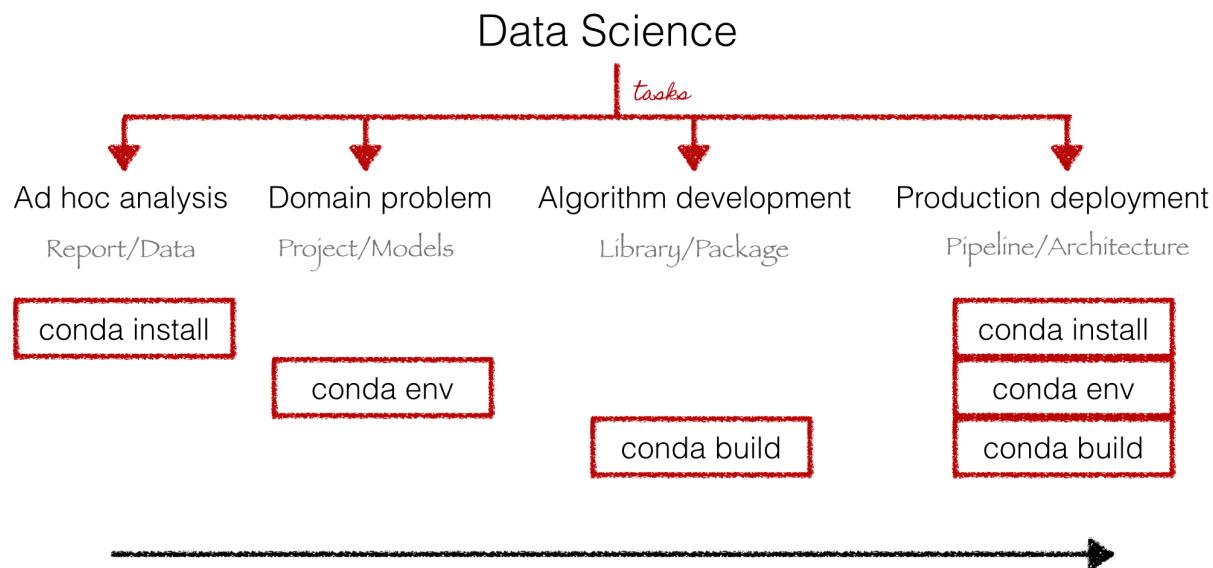
# Conda: Package and Environment Management

<http://chdoig.github.io/pydata2015-dallas-conda>

- Install packages
- Update packages
- Create sandboxes: Conda environments
- Conda environments: Critical for reproducibility, collaboration & scale



conda	pip (+ virtualenv)
language agnostic handles environments natively installs binaries general purpose environments	python packages virtualenv compiles from source python environments

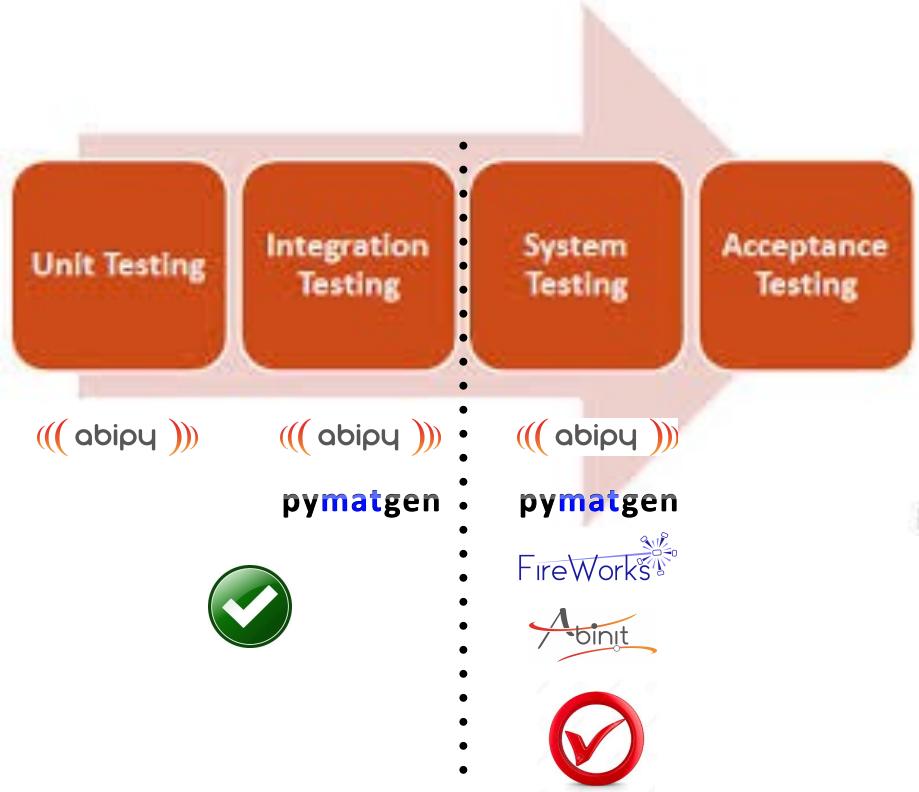
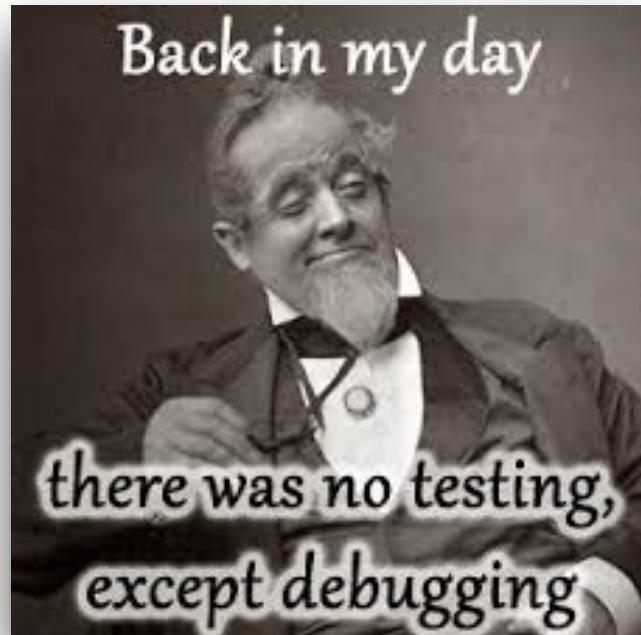


The screenshot shows the Anaconda Cloud interface for the abinit package repository. At the top, there's a navigation bar with the Anaconda Cloud logo, a search bar, and links for Docs, Contact, and abinit. Below the header, the title "Package Repository for abinit" is displayed. A navigation bar with tabs for Packages, Files, Install Instructions, and History is shown, with "Packages" being the active tab. Underneath is a "Filters" section with dropdowns for Type (set to conda), Access (all), and Label (main). Below the filters are two buttons: "Delete" and "Set access". The main content area displays a table of packages:

Package Name	Access	Summary	Updated
abipy	public	No Summary	2017-03-27
apscheduler	public	No Summary	2017-03-27
html2text	public	No Summary	2017-03-27
abinit_seq	public	No Summary	2017-03-21
abinit	public	No Summary	2017-03-20
oncwpsp	public	No Summary	2017-03-20
atompaw	public	No Summary	2017-03-20
libxc	public	No Summary	2017-03-20
fftw	public	No Summary	2017-03-20

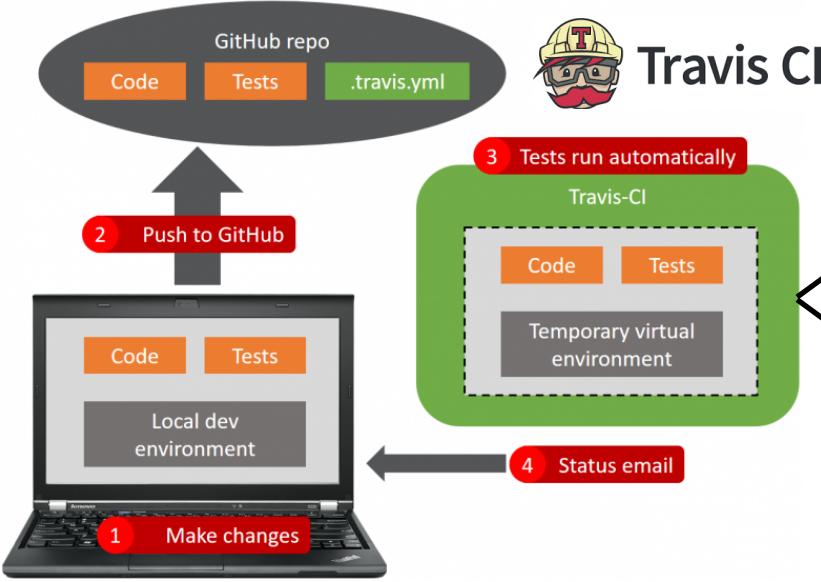
At the bottom of the table, there are navigation links: « Previous, showing 1 of 1, and Next ».

- ◆ Install abinit from scratch in 5 steps (see <https://github.com/abinit/abiconda>)
- ◆ Linux-64 and Osx-64 (sequential or parallel version)
- ◆ Can be used for schools and tutorials (not recommended for HPC)
- ◆ Cornerstone of the continuous integration infrastructure used by AbiPy ...



- Testing scientific software has never been easy (roundoff errors, parallelism ...)
- Testing frameworks for automatic calculations is even more complicated:
  - Lots of “futures”
  - python code assumes a well-defined “response protocol” (output files in a given format, error messages ...)
  - In principle one should validate the code in a typical production env (databases, resource managers, MPI execution in “exotic” architectures ...)

# Level1: unit tests and CI



Travis CI Dashboard Screenshot:

My Repositories

- abinit/abipy (Activity · Dashboard · GitHub) - PR #128 Change compatibility for API\_KEY (failed, 1 hr 11 min 14 sec, 76d7714, 20 days ago)
- abinit/abiconda-tests (Activity · GitHub) - PR #127 updates and tests (passed, 1 hr 10 min 46 sec, db5c3f4, 27 days ago)
- materialsproject/custodian (Activity · GitHub) - PR #127 updates and tests (failed, 1 hr 5 min, 9efb676, 28 days ago)
- abinit/pseudo\_doj (Activity · GitHub) - PR #126 testing runlevel. (passed, 1 hr 16 min 54 sec, d25cc6e, about a month ago)
- abinit/abiflows (Activity · GitHub) - PR #125 testing lessons (passed, 1 hr 6 min 18 sec, cf05c8, about a month ago)
- PR #124 more tests : gw convergence inputs (errored, 43 min 49 sec, #811 errored, 43 min 49 sec)

Coveralls Dashboard Screenshot:

ABINIT / ABIPY / 880

82%  
MASTER: 63%

COVERAGE INCREASED (+0.3%) TO 82.205%

BUILD # 880  
BUILD TYPE push  
COMMITTED BY gmatteo  
COMMIT MESSAGE Fix bug in skw in systems without spatial inversion

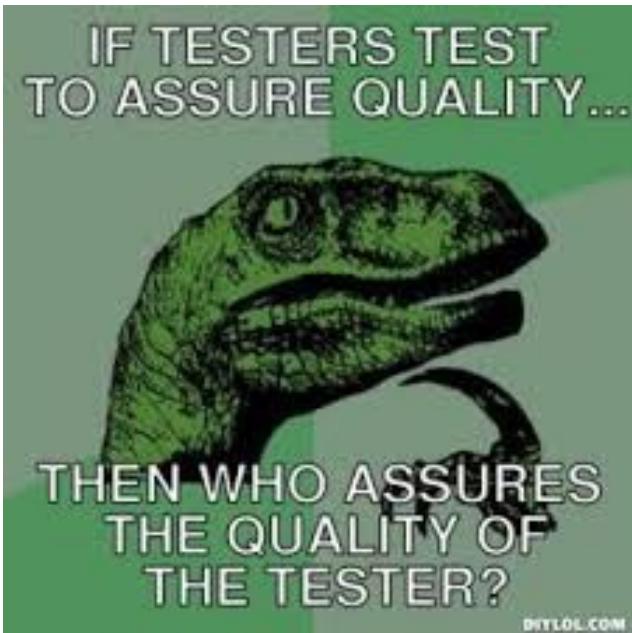
JOB COVERAGE JOB FILES COVERED RAN

JOB	COVERAGE	FILES COVERED	RAN
880.1 (PYTHON_VERSION=2.7)	82.2	185	2 days ago

- 82% coverage
- abinit8.2.2 from conda vs Abipy@github:
  - ✓ validate input files
  - ✓ execute small workflows

# Level2: integration with other python packages and software deployment

abiconda-tests:



## Goals:

- github repository to periodically test software integration
- pip/conda/github-repos with abinit8.2.2 binary from abinit channel

```
Abinitbuild:  
Abinit Build Information:  
  Abinit version: 8.2.2  
  MPI: True, MPI-IO: True, OpenMP: False  
  Netcdf: True  
  
Abipy Scheduler:  
PyFlowScheduler, Pid: 2194  
Scheduler options: {'seconds': 10, 'hours': 0, 'weeks': 0, 'minutes': 0, 'days': 0}  
  
Installed packages:  
Package      Version  
-----  
system        Linux  
python_version 2.7.13  
numpy         1.12.1  
scipy          0.19.0  
netCDF4       1.2.7  
apscheduler   2.1.0  
pydispatch     2.0.5  
yaml           3.12  
pymatgen      4.7.4  
matplotlib    2.0.0 (backend: Qt5Agg)  
  
Abipy requirements are properly configured  
  
Running small flow in workdir: /tmp/tmpWxnGY8  
  
[Mon May 1 14:59:52 2017] Number of launches: 1  
  
Work #0: <BandStructureWork, node_id=1, workdir=../../../../tmp/tmpWxnGY8/w0>, Finalized=False  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| Task | Status | Queue | MPI|Omp|Gb | Warn|Com | Class | Sub|Rest|Corr | Time | Node_ID |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| w0_t0 | Submitted | 2257@travis | 2| 112.0 | NAINA | ScfTask | (1, 0, 0) | 0:00:00Q | 2 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| w0_t1 | Initialized | None | 1| 112.0 | NAINA | NscfTask | (0, 0, 0) | None | 3 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
  
[Mon May 1 15:00:02 2017] Number of launches: 1  
  
Work #0: <BandStructureWork, node_id=1, workdir=../../../../tmp/tmpWxnGY8/w0>, Finalized=False  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| Task | Status | Queue | MPI|Omp|Gb | Warn|Com | Class | Sub|Rest|Corr | Time | Node_ID |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| w0_t0 | Completed | 2257@travis | 2| 112.0 | 4| 0 | ScfTask | (1, 0, 0) | 0:00:02R | 2 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| w0_t1 | Submitted | 2295@travis | 2| 112.0 | NAINA | NscfTask | (1, 0, 0) | 0:00:00Q | 3 |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

# Next step: monitor integration between Abinit@gitlab and AbiPy-stable



## Goals:

- ▶ Buildbot slave to run AbiPy tests with Abinit-dev when merge-request is opened on gitlab
- ▶ Detect possible regressions or incompatibilities before we go in production

- ▶ Scientific code must be allowed to change and evolve to tackle new problems
- ▶ But now there's an entire ecosystem around Abinit and big changes must be rationalised and planned in advance (e.g. Abinit7 —> Abinit8)

**Thank you for your attention**

This screenshot shows the GitHub organization page for ABINIT. The top navigation bar includes links for 'This organization', 'Search', 'Pull requests', 'Issues', 'Gist', and notifications. The organization's logo is a stylized 'A' with a red swoosh, and it is located in Belgium with a website at <http://www.abinit.org>. The pinned repositories are:

- abinit**: GitHub mirror of the Abinit repository (stable branch). We welcome bug fixes and improvements. Note that most of the active developments are hosted on our <https://gitlab.abinit.org/> server. Before ...  
Type: Fortran | Stars: 3 | Forks: 2
- abipy**: Open-source library for analyzing the results produced by ABINIT  
Type: Jupyter Notebook | Stars: 10 | Forks: 27
- pseudo\_dojo**: Python framework for generating and validating pseudo potentials  
Type: Python | Stars: 1 | Forks: 15
- abiconfig**: Configuration files to configure/compile Abinit  
Type: Python | Forks: 3
- abiconda**: Conda recipes for the abiconda channel.  
Type: Shell
- abiflows**: High-throughput calculations with Abinit  
Type: Python | Forks: 5

At the bottom, there are search fields for 'Search repositories...', dropdown menus for 'Type: All' and 'Language: All', and a green 'New' button.

- ♦ **pseudo\_dojo**: python package providing pseudos/workflows for tests
- ♦ **abiconfig**: configuration files for clusters
- ♦ **abiconda**: recipes to build abinit/abipy (matsci.sh provides pymatgen packages)
- ♦ **abiflows**: Fireworks workflows
- ♦ **abinit\_issues**: