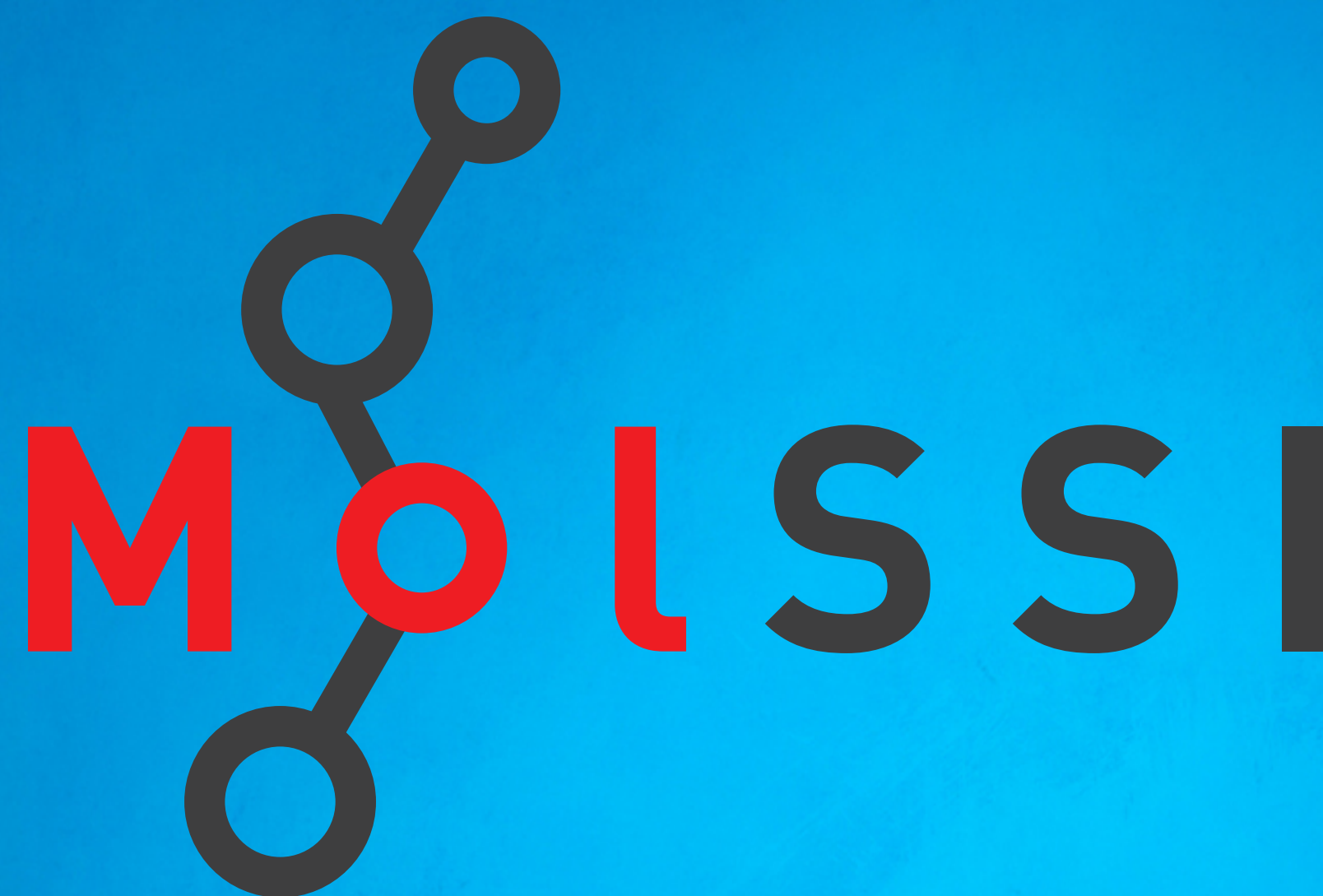


Open Source Best Practices: From Continuous Integration to Static Linters



Daniel G. A. Smith and Ben Pritchard
The Molecular Sciences Software Institute
[@dga_smith](#) / [@molssi_nsf](#)



The Molecular Sciences Software Institute (MolSSI)

- Designed to *serve* and *enhance* the software development efforts of the broad field of computational molecular science (CMS).
- Funded at the institute level under the Software Infrastructure for Sustained Innovation (SI²) initiative by the National Science Foundation.
- Launched August 1st, 2016

- Collaborative effort by:

- Virginia Tech

- Rice U.

- Stony Brook U.

- U.C. Berkeley

- Stanford U.

- Rutgers U.

- U. Southern California

- Iowa State U.



The Molecular Sciences Software Institute (MolSSI)

- Software Infrastructure
 - Build open-source frameworks and packages for the entire CMS community
- Education and training
 - Online materials and training guides
 - Host a number of targeted workshops, summer schools, and other training programs
- Community Engagement and Leadership
 - Interoperability standards and best practices
 - Data and API standardization targets



Talk Motivation

- Many terms thrown around:
 - **Best Practice**
 - **DevOps**
 - **Open Source**
 - **Software Sustainability**
 - **"Incorrect Software"**
 - **Manage Code**
 - **Build and Package**
 - **"Just get the physics working"**

"**Best**" is subjective and depends on your own community.

Lets discuss practical applications and how to make our lives easier!

Open Source

Open Source software is software that can be freely accessed, used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition. – Open Source Initiative



- **Free Redistribution** – Program is free of charge and distributable
- **Source Code** – Source code is freely available
- **Derived Works** – Allows modification of source in derived works

Open Source

- **Community** – Building engaged cooperative communities
- **Open Collaboration**– Anyone is free to join the community
- **Development Costs** – Allows common goal execution



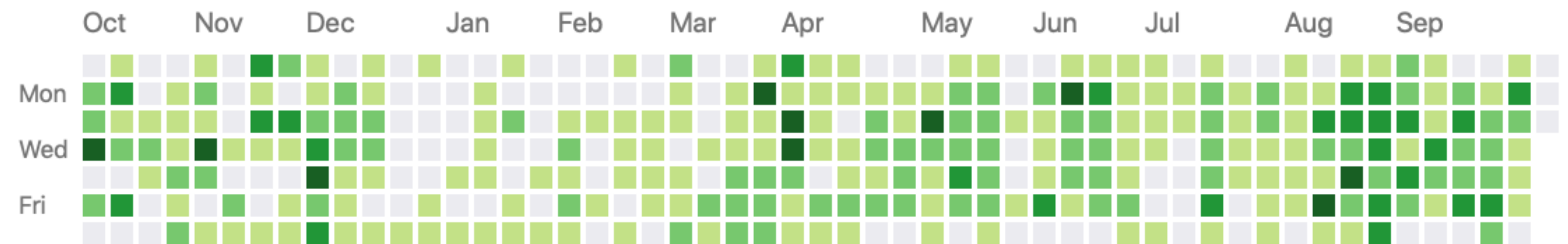
Open Source

Personal Reasons

- **Improves Coding Skills** – Code reviews and feedback
- **New Technologies**– Large base of users to provide unique experience
- **Peer Recognition** – Complete portfolio of what a person has done
- **Job Prospects** – Excellent for resumes

2,412 contributions in the last year

Contribution settings ▼



[Learn how we count contributions.](#)

Less More

Version Control

- Tracks changes over time (and who made them)
- Allow independent development on branches, and then merging later
- Can revert inappropriate changes
- Most popular: git
- Comments at the top of source files **do not** count
- Foundation of a community

GitHub



 Bitbucket

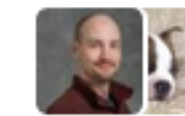
Commits on Oct 3, 2018

Merge pull request [#1278](#) from raimis/fix_py36 ...



robertodr committed 5 days ago ✓

Dft fixes ([#1260](#)) ...



susilehtola authored and andysim committed 5 days ago ✗

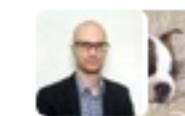
Use Python 3.6 for Windows builds



raimis committed 6 days ago ✓

Commits on Oct 1, 2018

Switch MSVC to LLVM compiler for Windows builds ([#1264](#)) ...



raimis authored and andysim committed 8 days ago ✗

Testing

“Mistakes in software erode trust in research and researchers” – Neil Chue Hong

We can improve reliability and trust in software by testing it

- **Unit Testing** – test a small unit (function)
- **Integration Testing** – test interoperation of units
- **Regression Testing** – ensure a fixed bug remains fixed
- **Smoke Testing** – sanity test of whole system
- **Acceptance Testing** – test that a feature is correctly implemented
- Testing of failure modes is also encouraged



Testing Frameworks

- Pick a common, appropriate testing framework
 - Depends on language and personal preference
 - Do **not** write your own unless you have to!

- C/C++ – CTest, Catch2

- Python – PyTest

- Fortran – CTest

```
52/54 Test #52: gtoeri_gtoeri_4center_water_sto-3g.inp_testcreate.dat_exact ..... Passed 149.47 sec
      Start 53: gtoeri__create_reference
53/54 Test #53: gtoeri__create_reference ..... Passed 15.31 sec
      Start 54: reference_gtoeri_gtoeri_testref.ref
54/54 Test #54: reference_gtoeri_gtoeri_testref.ref ..... Passed 15.30 sec

100% tests passed, 0 tests failed out of 54

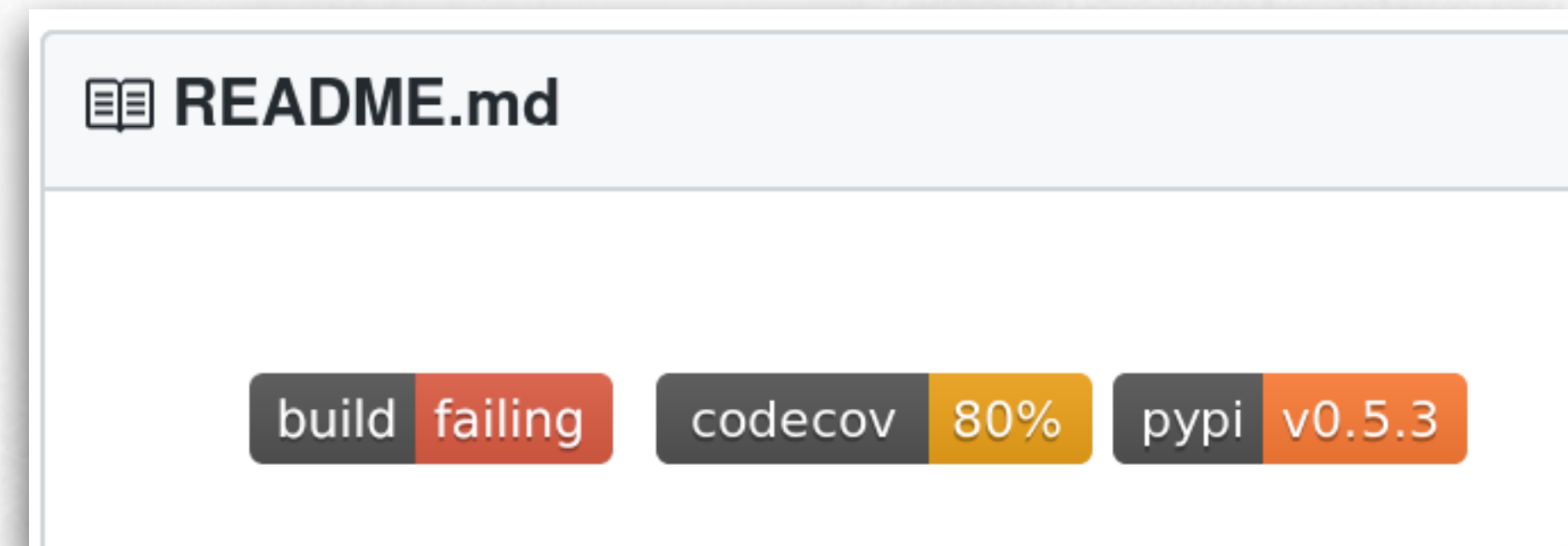
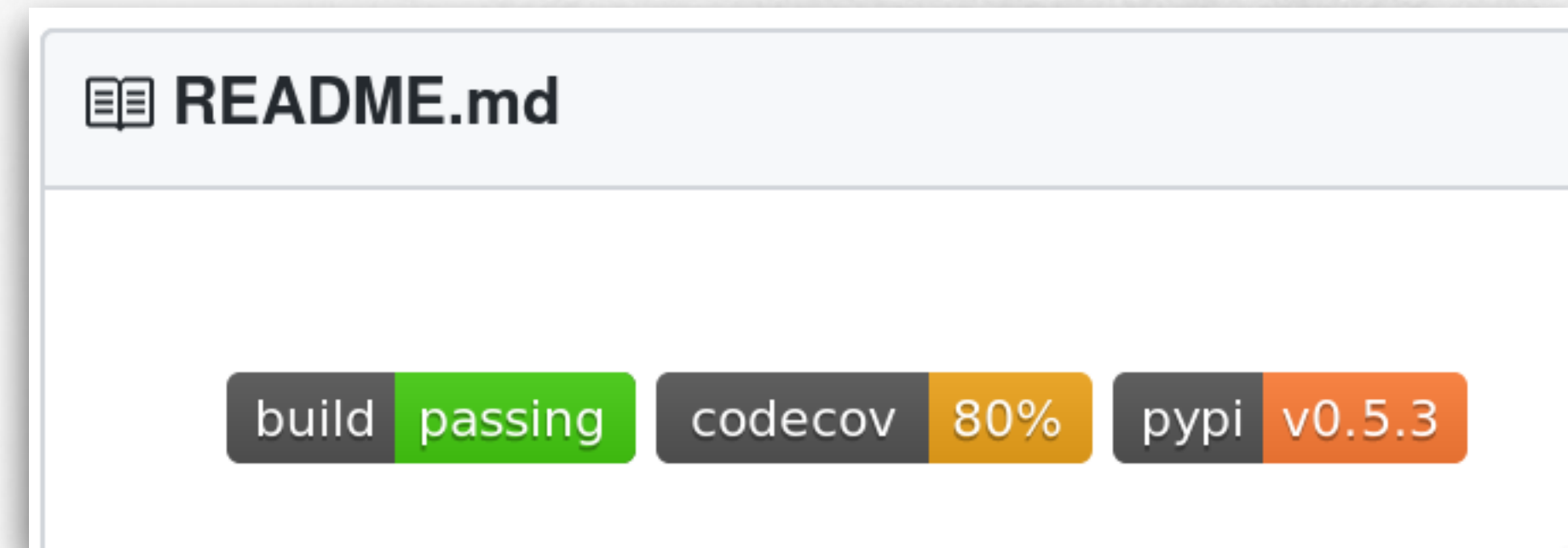
Total Test time (real) = 1073.07 sec
```

```
basis_set_exchange/tests/test_converters.py ..... [ 47%]
basis_set_exchange/tests/test_io.py ..... [ 89%]
basis_set_exchange/tests/test_metadata.py .. [100%]

===== 19 passed, 15996 deselected in 7.86 seconds =====
```


Continuous Integration (CI)

- Testing is done automatically
- Results are displayed publicly
- Develops trust in software package
- Developers feel safer contributing
- Badges!



Continuous Deployment – CI + deploy a service

CI Features

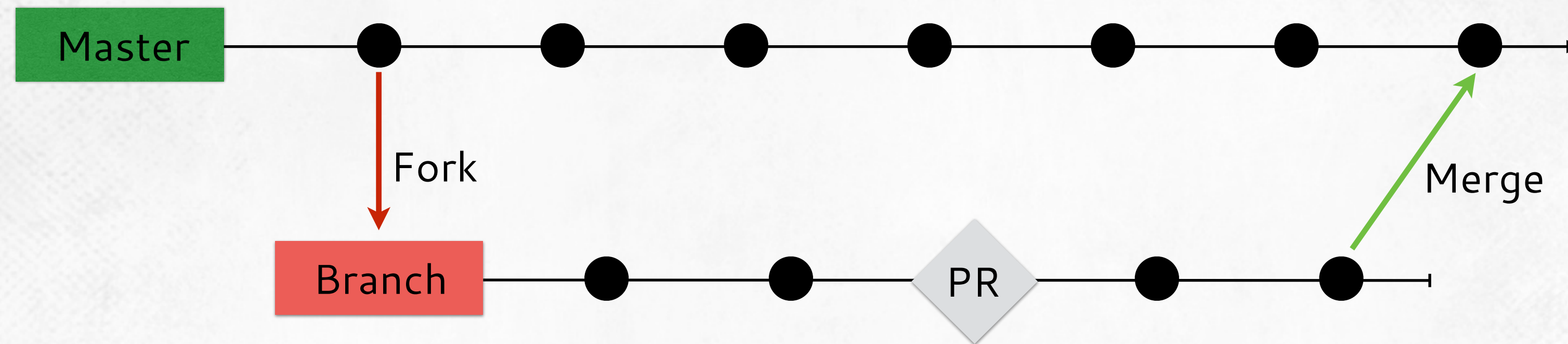
- Matrix builds

✓ # 2969.1	 </> Compiler: clang C++	 CXX_COMPILER='clang++-3.6' PYTHON_VER='...' ⌚ 30 min 50 sec
✓ # 2969.2	 </> Compiler: gcc C++	 CXX_COMPILER='g++-4.9' PYTHON_VER='3.7' ... ⌚ 33 min 5 sec
✓ # 2969.3	 </> Compiler: gcc C++	 CXX_COMPILER='g++-6' PYTHON_VER='3.6' C... ⌚ 35 min 7 sec

- Different:

- Distributions
- Operating systems
- Dependency versions
- Integrated with version control (git, GitHub, GitLab etc)
- Integrated with social services such as Slack, etc

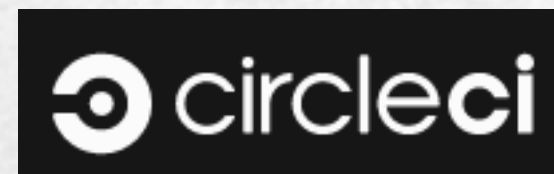
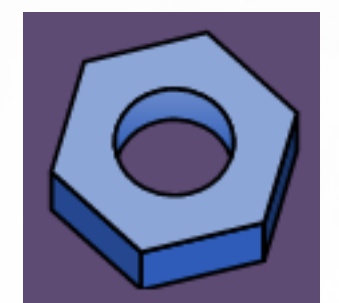
Development Workflow (Fork/Pull Model)



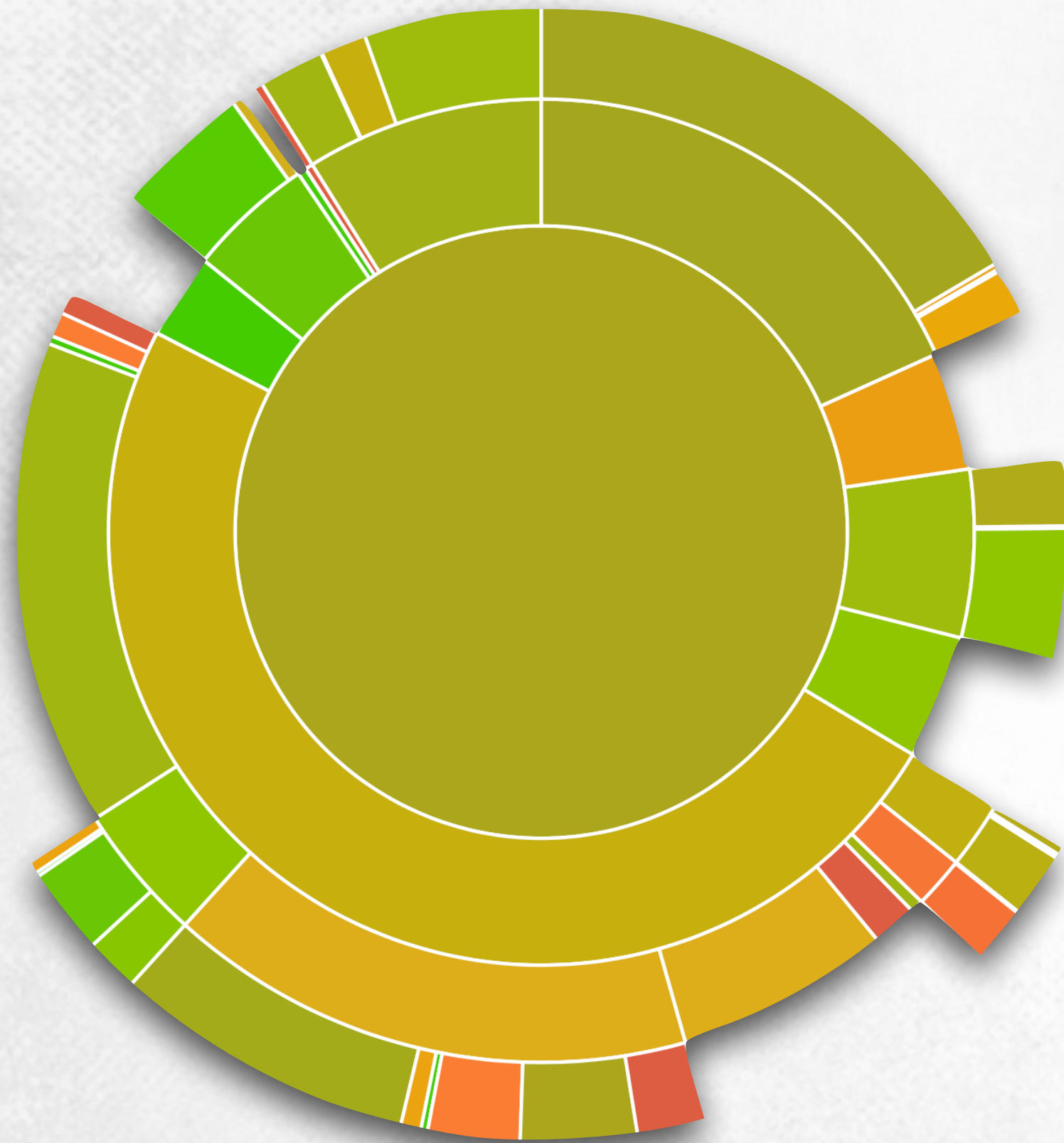
- Development happens on an independent branch/repo
- Code is reviewed/**tested** during the Pull Request (PR)
- If accepted, code is merged into the *master* branch

CI Services and Software

- Services can be cloud-based or you can use your own resources
- Some are free for open-source projects
- Cloud hosted:
 - TravisCI ([Linux/OSX](#))
 - CircleCI ([Linux/OSX](#))
 - Appveyor ([Windows](#))
 - Azure Pipelines ([Linux/OSX/Windows](#))
- Locally Hosted
 - Jenkins
 - Buildbot
 - Team City



Code Coverage



codecov.io

- Tendency to write many tests that cover a single area while completely neglecting others.
- 100% code coverage does not imply perfect code!
- General metric that informs testing competency that is difficult to fake.

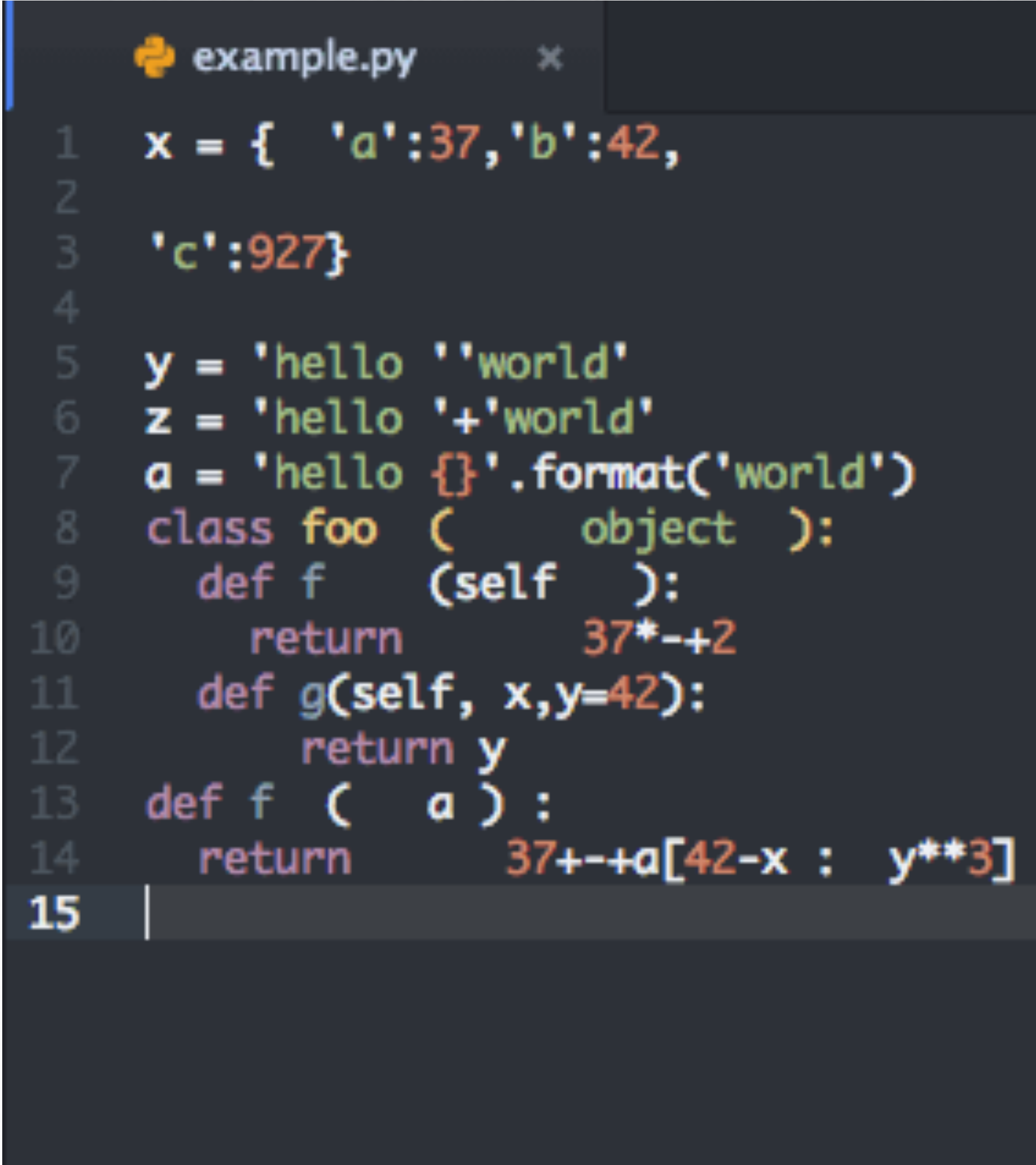
Code Coverage

238	7	<code>if "queue_socket" in self.objects:</code>
239		<code># Add canonical queue callback</code>
240		<code>nanny = tornado.ioloop.PeriodicCallback(self.objects["queue_nanny"].update, 2000)</code>
241		<code>nanny.start()</code>
242		<code>self.periodic["queue_nanny_update"] = nanny</code>

- Common case of missing "if" statements
- Exceptions are often not tested!
- Services: Codecov, Coveralls
- Tools: gcov (C++), coverage (Python), etc
- All languages (C, C++, Fortran, Python, Java, Julia, etc)

Code Formatters

- Automatic application of a standard format
 - Most language have a single standard (Python)
 - Several language have multiple standard (C++)
- Provides consistent code
 - Within your own software stack
 - And amongst other software stacks
- Tools: clang-format (C/C++), yapf (Python)
- Most languages (C, C++, Python, Java, Julia, etc)



```
example.py
1 x = { 'a':37,'b':42,
2
3 'c':927}
4
5 y = 'hello ' + 'world'
6 z = 'hello ' + 'world'
7 a = 'hello {}'.format('world')
8 class foo ( object ):
9     def f (self ):
10         return 37*--+2
11     def g(self, x,y=42):
12         return y
13 def f ( a ) :
14     return 37+--+a[42-x : y**3]
15
```

YAPF

Code Quality



lgtm.co

- Statically examine the code for a variety of errors and potential errors
- Catch bugs that do not have unit tests for them.
- Automation of code review:
 - Always catches “known” errors
 - Bad news is easier from a bot

Code Quality

Actual errors

```
2150     for (iop = 0; iop < nop; ++iop) {  
2151         int axis, op_ndim, op_axis;
```

Local variable 'axis' hides a parameter of the same name.



Possible errors

```
1440     printf("| IterIndex: %d\n", (int)NIT_ITERINDEX(iter));  
1441     printf("| Iterator SizeOf: %d\n",  
1442           (int)NIT_SIZEOF_ITERATOR(itflags, ndim, nop));
```

Multiplication result may overflow 'int' before it is converted to 'unsigned long'.



Potential code duplication

```
186     def list_final_energies(self, fragments=None, refresh_cache=False):
```

All statements in list_final_energies are similar in list_final_molecules.



Code Quality

- Still experimenting with code quality analysis
- Many are overzealous in their warnings
- LGTM seems to hit the “right” balance

✓		LGTM analysis: C/C++ — No code changes detected	Details
✓		LGTM analysis: JavaScript — No code changes detected	Details
✓		LGTM analysis: Python — No alert changes	Details

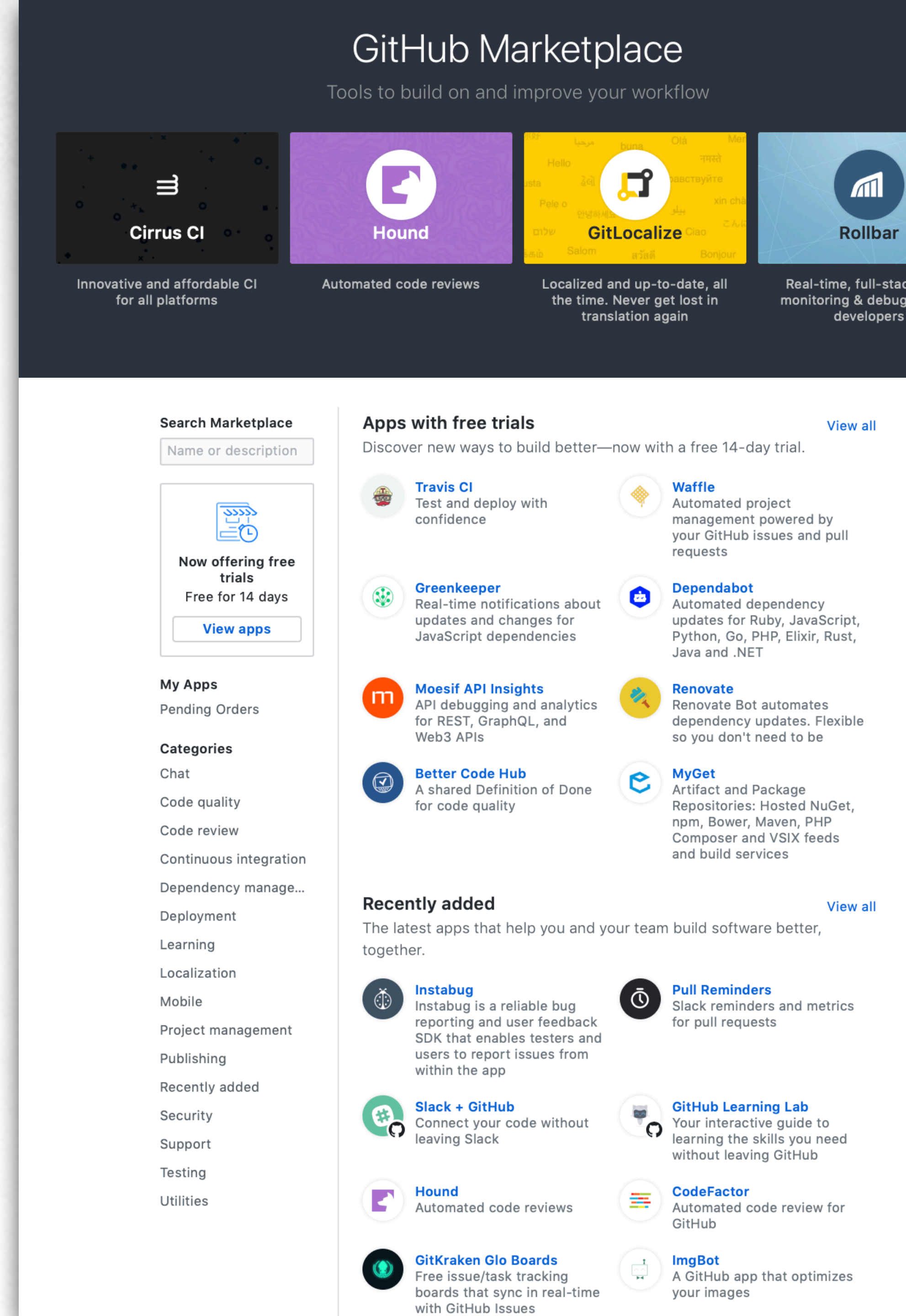
- LGTM, Code Climate, PyFlakes, ...

- Most languages (C, C++, Python, Java, Julia, etc)

...and more

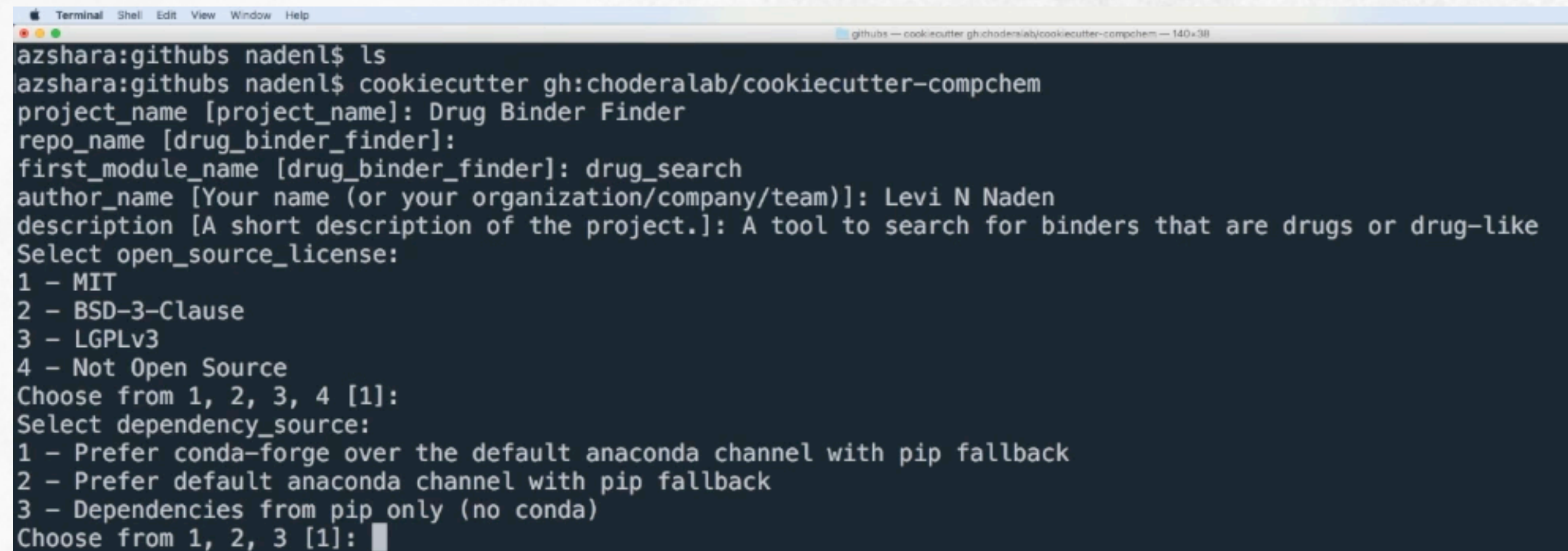
- Many other services out there
 - Dashboards
 - Agile Task Boards
 - Vulnerability Checkers
 - Dependency Management
 -

...you can have too many services!



Cookiecutters [Python]

- Automatic setup:
 - Version Control (git)
 - License (MIT/BSD/LGPL/Other)
 - CI (Travis CI/Appveyor)
 - Testing (PyTest)
 - Code Coverage (Codecov)
 - Documentation (Sphinx)
- Takes ~5 minutes to integrate



```
azshara:githubs nadenl$ ls
azshara:githubs nadenl$ cookiecutter gh:choderalab/cookiecutter-compchem
project_name [project_name]: Drug Binder Finder
repo_name [drug_binder_finder]:
first_module_name [drug_binder_finder]: drug_search
author_name [Your name (or your organization/company/team)]: Levi N Naden
description [A short description of the project.]: A tool to search for binders that are drugs or drug-like
Select open_source_license:
1 - MIT
2 - BSD-3-Clause
3 - LGPLv3
4 - Not Open Source
Choose from 1, 2, 3, 4 [1]:
Select dependency_source:
1 - Prefer conda-forge over the default anaconda channel with pip fallback
2 - Prefer default anaconda channel with pip fallback
3 - Dependencies from pip only (no conda)
Choose from 1, 2, 3 [1]:
```

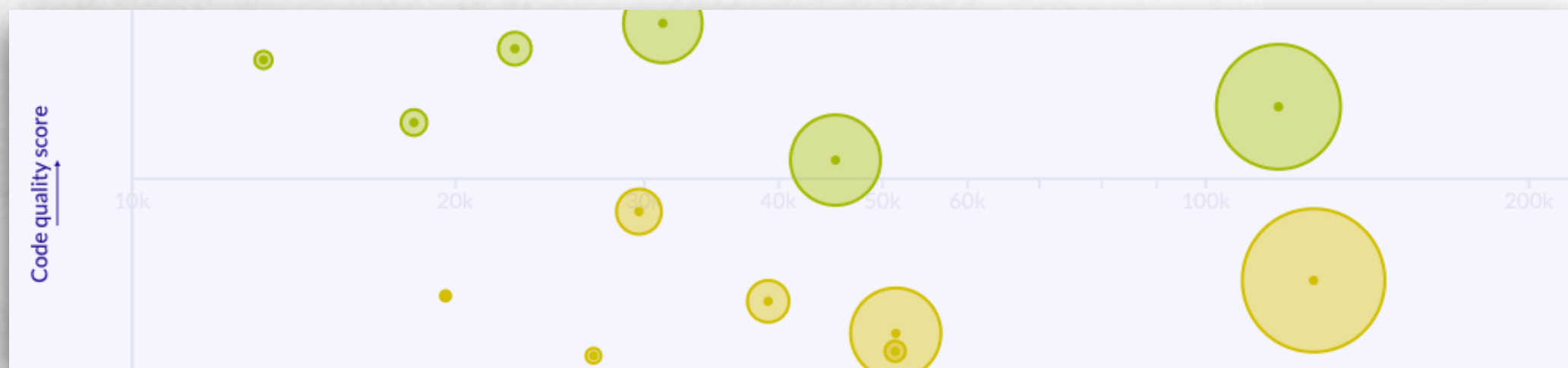
<https://github.com/MolSSI/cookiecutter-cms>

Summary

- Version Control
- Testing
- Continuous Integration
- Code Coverage
- Formatters



- Saves developer time
- Builds trust with your community base
- Makes your code more welcoming to new developers
- Best practices are a balance of developer and user time



Communities of Practice

“**Best**” is subjective and depends on your own community.



- We believe these practices are optimal for our community and are derived from many other communities
- Depends greatly on:
 - Target audience
 - Size of the project
 - Activity level of the project
- Still core “good ideas” that can be widely applied



Example Repositories

- **Molecular Dynamics:**

- LAMMPS – GitHub, CI (Jenkins),
- OpenMM – GitHub, CI (Travis/Appveyor)

- **Quantum Mechanics:**

- NWChem – GitHub, CI (Travis)
- Psi4 – GitHub, CI (Travis/Appveyor), Code Coverage

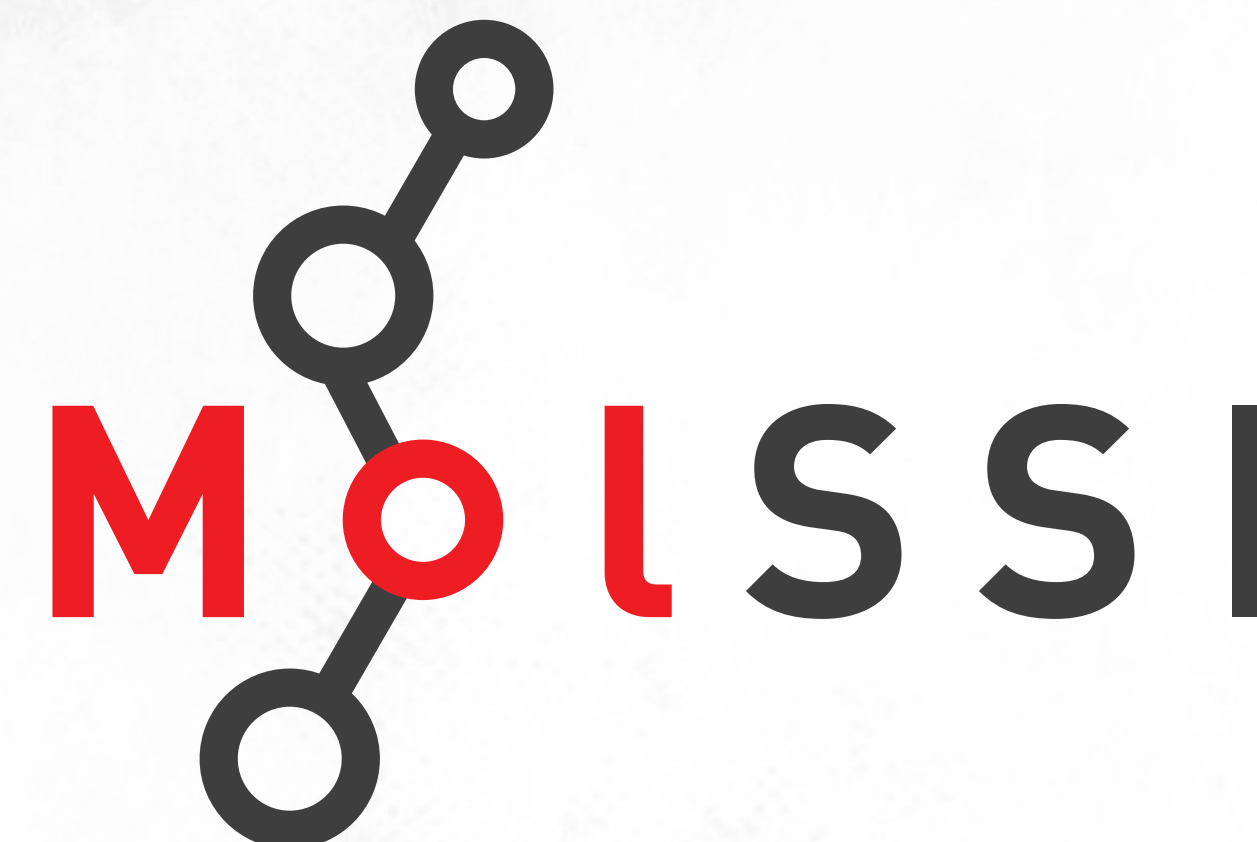
- **Python Utility Codes:**

- opt_einsum – GitHub, CI, Code Coverage
- QCFractal – GitHub, CI, Code Coverage, Static Analysis

Learn More about MolSSI



- Twitter: [@molssi_nsf](https://twitter.com/molssi_nsf)
- Best Practices: <https://molssi.org/education/best-practices/>
- E-Mail List: [MolSSI Mailing List](#)
- Cookiecutter: <https://github.com/MolSSI/cookiecutter-cms>



Watch molssi.org for the latest information!

Tools

Code Coverage

- Services: Codecov, Coveralls
- Tools: gcov (C++), coverage (Python)

Static Analysis

- LGTM, Code Climate, PyFlakes

Version Control

GitHub



GitLab

 **Bitbucket**

Continuous Integration

- Cloud hosted:
 - TravisCI
 - CircleCI
 - Appveyor
 - Azure Pipelines
- Locally Hosted
 - Jenkins
 - Buildbot
 - TeamCity