# Software Infrastructure

**Open Force Field Initiative**
Consortium meeting, 8 Jan 2019

**Jeff Wagner** (OFF), **Daniel Smith** (MolSSI),
**Simon Boothroyd** (MSKCC), **David Mobley** (UCI),
**John Chodera** (MSKCC)

**#infrastructure** on Slack

1. **What is the OpenForceField Toolkit?**
2. **How will we develop software components?**
3. **How will the components interact?**

1. **What is the OpenForceField Toolkit?**
2. **How will we develop software components?**
3. **How will the components interact?**

# What software do I need for my use case?

“I want to set up simulations using the force fields that the Open Force Field Consortium will make.”

- You need the OpenForceField toolkit

“I want to perform bespoke parameterization using my own machines”

- You need the bespoke workflow (yet to come, will contain ForceBalance, QC driver, etc)

“I want to use my own machines to make my own force field from the ground up, to cover my favorite region of chemical space”

- You need all project components

# The OpenForceField Toolkit

The **SMIRNOFF specification** is a language for defining parameterization strategies.

- Our aim is to contain keywords for every decision that affects system energy
- The spec is a general object model (not restricted to XML)

An **OFFXML file** uses the language laid out by the SMIRNOFF spec to describe a particular parameterization strategy.

- Describes SMIRKS-based parameter application
- Can include parameter libraries (eg. TIP3P water)
- Is modular, and can be extended to support new classes of interactions or charge methods

The **OpenForceField Toolkit** is a program that can read an OFFXML file and your molecules, and output a system ready for simulation.

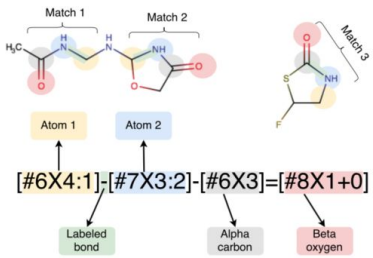
# Exactly how do I use the OpenForceField Toolkit?

```
forcefield = ForceField("smirnoff99Frosst.offxml")
pdbfile = openmm.app.PDBFile("cyclohexane_ethanol_0.4_0.6.pdb")

molecules = []
molecules.append(Molecule.from_file("ethanol.mol2"))
molecules.append(Molecule.from_smiles("C1CCCCC1"))

topology = Topology.from_openmm(pdbfile.topology, unique_molecules=molecules)

openmm_system = forcefield.create_openmm_system(topology)

forcefield.create_amber_system(topology, "amber.prmtop", "amber.inpcrd")
forcefield.create_gromacs_system(topology, "gromacs.top", "gromacs.gro")
```

# OpenForceField Toolkit v 0.1.0

## Used for the initial SMIRNOFF publication

- Parameterizes systems using OpenEye's SMIRKS matching
- Allows users to manually add and change SMIRKS-based parameters
- Loads v 0.1 OFFXML parameter files
- Outputs OpenMM systems ready for simulation
- OpenMM systems can be converted Amber/Gromacs via ParmEd




David Mobley, Caitlin C. Bannan, Andrea Rizzi, Christopher I. Bayly, John D. Chodera, Victoria T Lim, Nathan M. Lim, Kyle A. Beauchamp, Michael R. Shirts, Michael K. Gilson, and Peter K. Eastman
Preprint ahead of publication: [bioRxiv] | CC-BY 4.0
Published: **Journal of Chemical Theory and Computation** . [DOI]
*This paper introduces the SMIRNOFF format in the context of traditional force fields, explains the development and validation of our new small molecule force field smirnoff99Frosst, and highlights some directions the initiative is headed.*

# Upcoming OpenForceField Toolkit 1.0.0

- Can run using only open-source dependencies
- Provides modular cheminformatics toolkit support
  - Implemented alternatives to most OE-dependent tasks using RDKit and AmberTools
  - Automatically checks which toolkits are available and selects backend
- Provides toolkit-independent Molecule and Topology classes
- Handles partial charge calculation
- Uses modularized ParameterHandlers, allowing researchers to experiment with new types of parameters easily without modifying toolkit
- Implements modular ParameterIOHandlers, to support reading files or serialized force fields other than XML
  - Plans to include URL!

**Toolkit 1.0.0 Prototype will be installed during hands-on session**

# Developments in Progress

- Additional flexibility in partial charge/bond order calculations
- Compatibility checks for SMIRNOFF spec version and aromaticity model
- Library charges
- Wiberg bond order calculation
- Support for parameterization strategies that include:
  - Off-center charges
  - Custom bond-charge corrections
  - Valence parameter interpolation based on fractional bond order
- Biopolymer support

# Documentation Overhaul: v0.1.0



Search

## Open Forcefield Toolkit

A modern, extensible library for molecular mechanics forcefield science.

Contents:



## Modules

# Documentation Overhaul: v1.0.0



## Open Force Field Toolkit

A modern, extensible library for molecular mechanics force field science from the Open Force Field Initiative

### User Guide

- Installation
- Release History
- The SMIRks Native Open Force Field (SMIRNOFF) specification v1.0
- Examples using SMIRNOFF with the toolkit
- Using Chemical Environments
- Developing for the toolkit
- Frequently asked questions (FAQ)

### API documentation

- Molecular topology representations
- Forcefield typing tools
- Utilities

# Versioning

The initial SMIRNOFF publication was done using version 0.1.0 of the OpenForceField Toolkit, and version 0.1 of the SMIRNOFF spec.

Today we are completing OpenForceField Toolkit 1.0.0 and version 1.0 of the SMIRNOFF spec.

Future toolkit releases will be labeled as version X.Y.Z, where

- X increments with major, API changing updates
- Y increments with newly added features
- Z increments with bugfixes

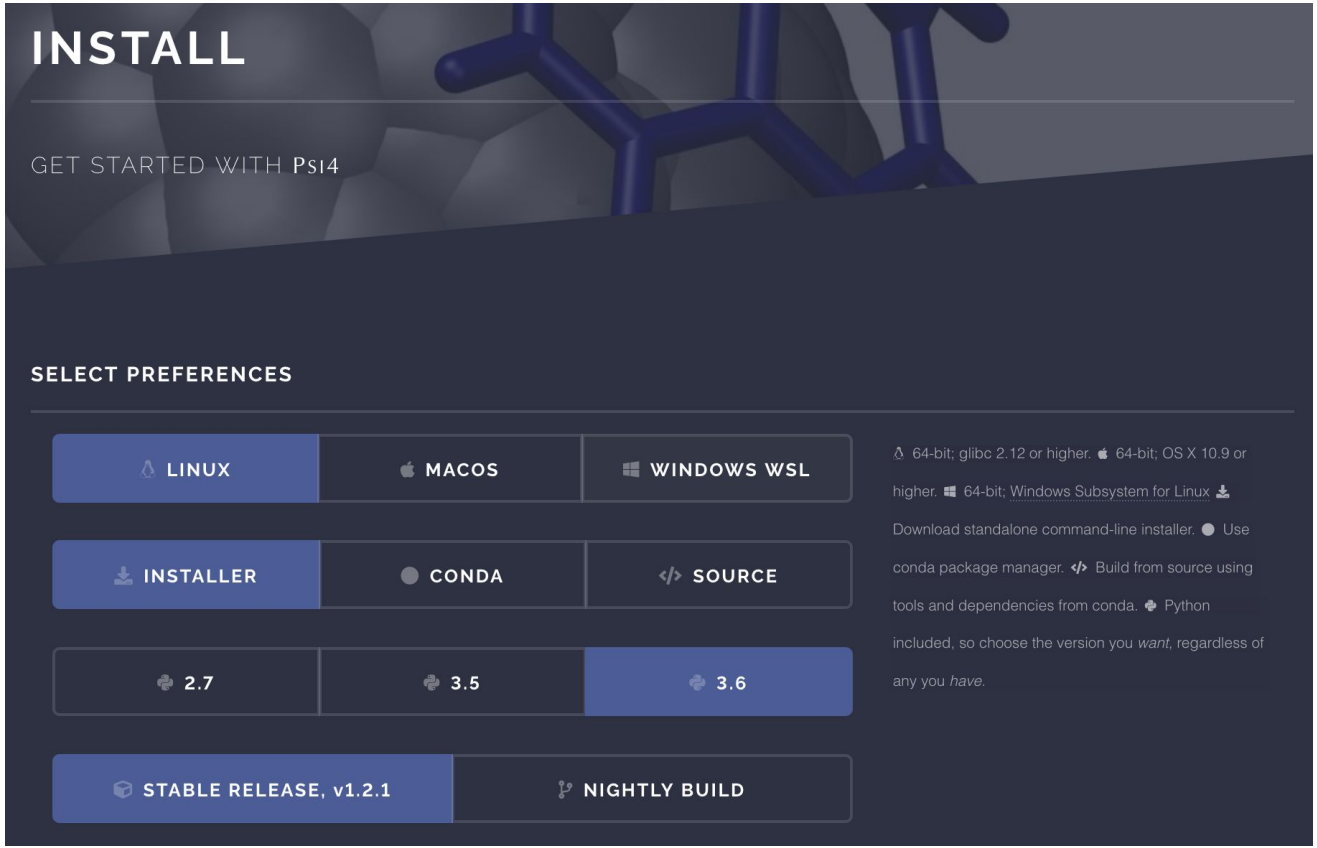
1. What is the OpenForceField Toolkit?
2. **How will we develop software components?**
3. How will the components interact?

# How Do I Install The Software?

- Based on survey results and comments from October meeting, we anticipate that some users will have restricted access to conda-forge

- **We will distribute fully-packaged binary installers for Mac and Linux**
  - Do not require administrator access to install
  - Include all dependencies
  - Are built using conda with Python 3.6 and 3.7
  - Conda is a popular distribution tool used in many industries
  - This isn't the method we'll use this afternoon, but will be available soon

# Installer Example: Psi4

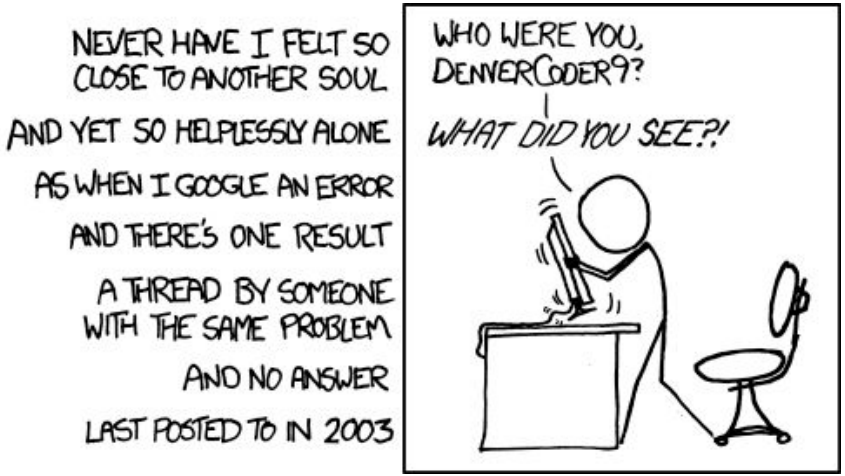


# Cloud-based Conda Install will be Supported

```
$ conda install openforcefield
```

# Migration to Conda-Forge will Improve Reliability

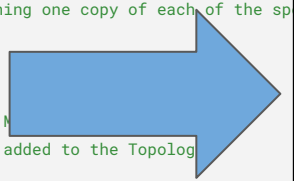
- Conda packages live on specific “channels”
- Packages must be maintained to remain readily distributable
- Omnia maintenance burden is substantial
- Conda-forge has an active community of professional maintainers



# ReadTheDocs

- A free web hosting service for software documentation
- Parses docstrings in code and generates nicely-formatted web pages
- Supports custom text such as examples and commentary mixed with autogenerated API information

```
@staticmethod
def from_molecules(molecules):
    """
    Create a new Topology object containing one copy of each of the sp

    Parameters
    ----------
    molecules : Molecule or iterable of M
            One or more molecules to be added to the Topolog

    Returns
    -------
    topology : Topology
            The Topology created from the specified molecule(s)
    """
```



***static*** **`from_molecules`** **(*molecules*)**

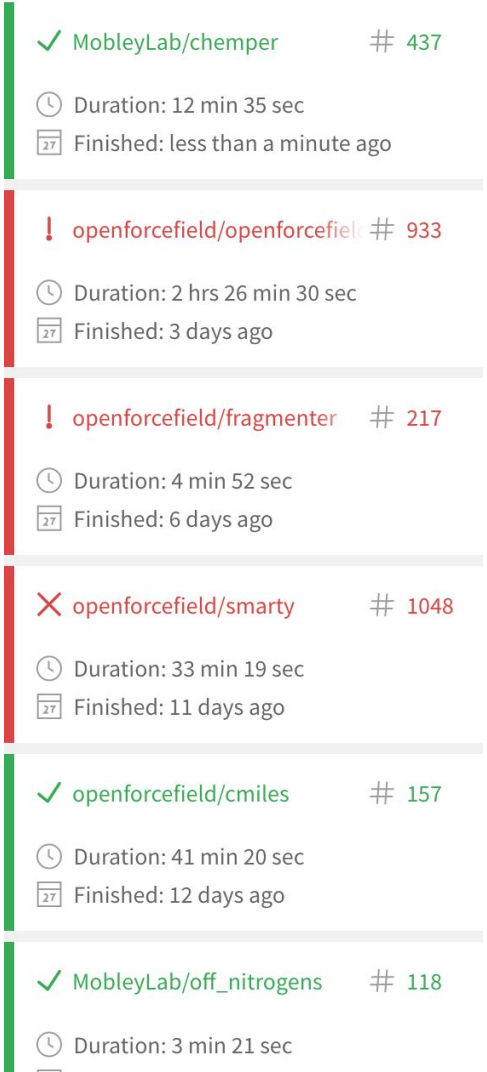
Create a new Topology object containing one copy of each of the specified molecule(s).

**Parameters:** **molecules : *Molecule or iterable of Molecules***
One or more molecules to be added to the Topology

**Returns:** **topology : *Topology***
The Topology created from the specified molecule(s)

# Testing Strategy

- Changes in one component can break others that rely on it
- **Regression tests** for each OpenFF component will be run using the “stable” versions of other OpenFF components
- **Integration tests** will be run nightly using the “development” version of each package
- Integration tests ensure new versions of the ecosystem continue to work together as backend code changes.

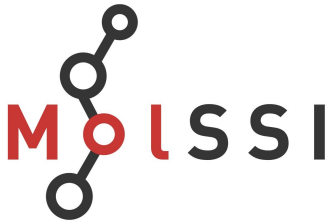


# Code of Conduct

- Open-source projects receive contributions from a global developer community
- Relying on volunteer contributions can leave the door open to negative behavior and create an unproductive atmosphere
- A formal code of conduct will be released soon, setting standards for all contributors

> “Really. Give me *one* reason why it was written in that idiotic way with two different conditionals, and a shiny new nonstandard function that wants particular compiler support to generate even half-way sane code, and even then generates worse code? A shiny function that we have never ever needed anywhere else, and that is just [expletive].”
>
> - Linus Torvalds

# Software Development Best Practices




- Each component in the consortium will need similar packaging/ distribution infrastructure
- By using the **MolSSI project cookiecutter**, we
  - Avoid redundant development work
  - Ensure that components will build using the same tools, allowing for efficient maintenance and platform compatibility management
- On Wednesday, Open Force Field developers are participating in a best practices workshop in Irvine

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

**Cookiecutter for Computational Molecular Sciences (CMS) Python Packages**

build passing build passing docs passing

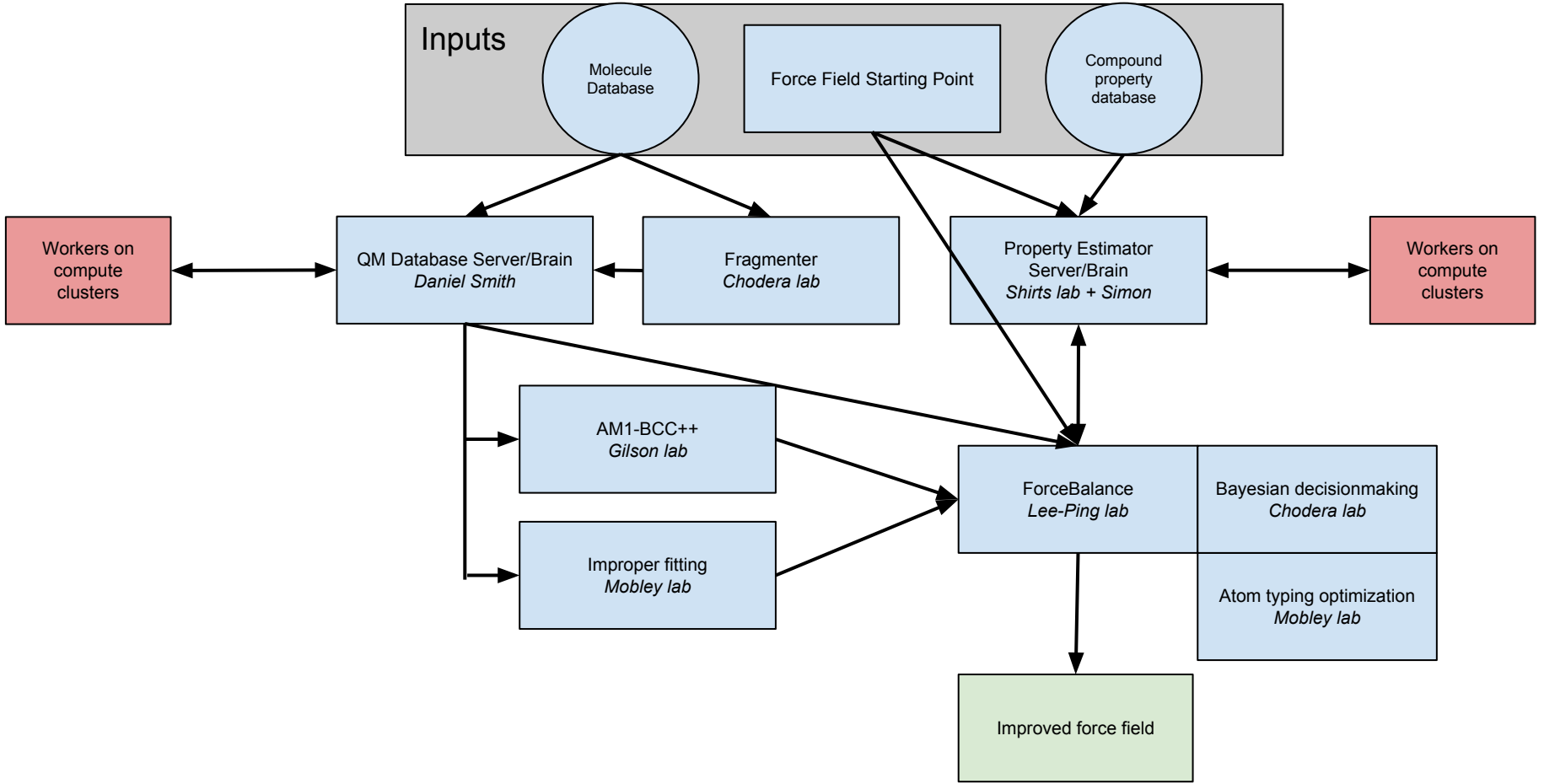
A cookiecutter template for those interested in developing computational molecular packages in Python. Skeletal starting repositories can be created from this template to create the file structure semi-autonomously so you can focus on what's important: the science!

# Individual Repositories in Consortium

| | Python API | Versioneer | Pytest | Travis | Appveyor | Pep8 | Documentation | LGTM | Codecov | RTD | Setup.py | pypi | Conda |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ForceBalance | | N | UnitTest / Integration tests | Build fails | | | Built in 2014 | | | Hosted on GH | Y | Y | Y |
| Openforcefield (SMIRNOFF) | Y | Y | Y | Build fails | N | | Y | N | | Y | Y | N | Y |
| GeomeTRIC | ? | Y | Y | Y | N | YAPF | Examples | | 29% | | Y | Y | Y |
| Respyte | ? | Active? | | Inactive | Inactive | | N | | Inactive | Inactive | Unsure if functional | N | N |
| AM1-BCC++ | ? | Active? | Active? | Y | N | | In progress | N | 2% | N | Unsure if functional | N | Inactive |
| Chemper | Y | N | Y | Y | N | | Y | Y | 87% | Y | Y | N | N |
| OFF Nitrogens | Y | N | Y | Y | N | | N | N | N | N | | | |
| Quantformer | | | | | | | | | | | | | |
| QCFractal | Y | Y | Y | Y | Y | YAPF | Y | Y | 90% | Y | Y | N | N |
| Paprika | | | | | | | | | | | | | |
| Fragmenter | | | | | | | | | | | | | |

1. What is the OpenForceField Toolkit?
2. How will we develop software components?
3. **How will the components interact?**

# The Problem: A Possible FF Optimization Workflow



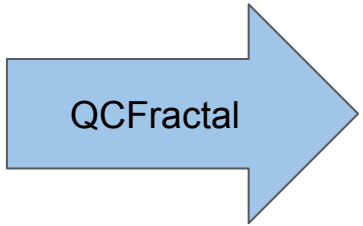
# The Solution: Stable, Defined APIs

- Initial APIs are guided by coordinated planning of component functionality
- Defining and sticking to APIs will allow developers to “plug in” to our components
- Backends can change as development goes on
- Frontends change infrequently with major version releases
- ReadTheDocs permits detailed description of specific function behaviors
- GitHub issues document user communication with developers
- Regression and integration testing will catch incompatibilities early

# Distributed Computing Example: QCFractal

**Input**

```
"molecule":
  "geometry": [
    0.0,  0.0000, -0.1294,
    0.0, -1.4941,  1.0274,
    0.0,  1.4941,  1.0274
  ],
  "symbols": ["O", "H", "H"]

"driver": "energy"

"model":
  "method": "B3LYP",
  "basis": "cc-pVDZ"

"keywords":
  "scf_type": "df",
  "e_congerence": 1.e-7
```



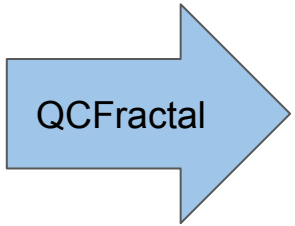

**Output**

```
"success": true,

 "return_result": -76.41876202714

"provenance":
  "creator": "QM Program",
  "version": "1.1",
  "routine": "module.json.run_json"
```

# Distributed Property Calculation (Approx.)

**Input**

```
"module_metadata": "property.density",
"substance":
        "components": [
            "smiles": "CCO",
            "mole_fraction": 1,
            "impurity": false ]
"model":
    "forcefield": "OFFXML text",
    "generate_coordinates": "packmol"
"thermodynamic_state":
    "temperature": "293.15 K",
    "pressure": "101300 Pa",
"experimental_data" {
    "value": "964.88 kg m^-3",
    "uncertainty": "0.05 kg m^-3"
...
```




**Output**

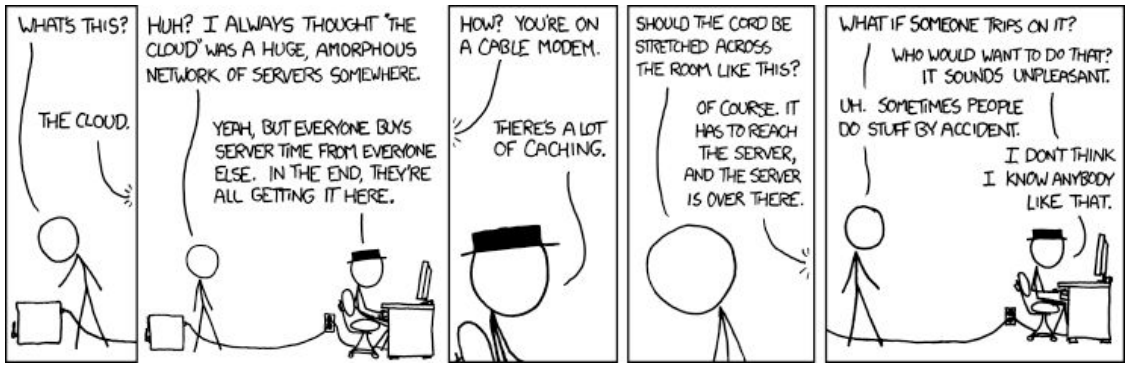
```
 "success": true,

   "return_result": 963.83479
   "return_uncertainty": 0.087241

 "provenance":
    "toolkit": "property_calculator",
    "toolkit_version": "1.2.3",
    "off_toolkit_version": "1.1.1"
    "toolkit_backend": "RDKit",
    "toolkit_backend_version": "45.6",
    "md_engine": "openmm",
    "engine_version": "7.89"
    "routine": "module.json.run_json"
…
```

# Distributed Computing Requirements

- `output = my_function(input_A, input_B)`
  - If `output`, `input_A`, and `input_B` can be serialized, and `my_function` is available on worker nodes, the function can run through QCFractal
- Standardized task descriptions
  - Use a neutral language to call APIs across machines
  - Return only the needed quantities instead of stdout or whole trajectory
  - Write keywords for common customization options
- Object serialization
  - Serialized objects can be safely transported to workers and unambiguously deserialized
  - Serialization will make component interfaces much more reliable
- Robust deployment
  - Conda distribution of all components enables software deployment and updates
- Provenance tracking
  - Tagging results with the software and versions used to generate them

# “OpenForceField Cloud”

- “A set of compute resources that any group can access for the good of OpenFF”
  - The collection of machines where QCFractal has managers/workers running
  - May require periodic updates via conda
  - Contact us if you are interested in contributing compute time
- The property calculator will initially be deployed on one cluster
  - Ensures that previous trajectories will always be available locally for reweighting



# OpenForceField Software Scientists

- Oversee bringing project components up to our software standards
- Maintain core software toolkits
- Guide development of APIs to ensure functionality
- Set up testing and deployment for component interoperability
- Help bring new components into the overall project
- Be a primary point of contact for pharma partners' tech support
    - For technical problems, post an issue on on the OpenForceField GitHub
    - For scientific issues, contact OpenForceField on Slack

**I will be scheduling one-on-ones with industry partners to assess infrastructure needs**

# We’d love your feedback (**#infrastructure**)

# Conversation Starter

- How do we encode things like AMBER “PME” parameters, which are specified in the mdin file for sander?


# Hands-on Session with Toolkit Prototype


If you don’t have conda

{

- Download miniconda3 from https://conda.io/miniconda.html
- Install miniconda: `$ bash Miniconda3-latest-MacOSX-x86_64.sh -b -p ./miniconda3`
- Activate the environment: `$ source miniconda3/bin/activate`

}

Download and install the openforcefield prototype package and dependencies:

- `$ conda create -n openforcefield -c rdkit -c conda-forge -c omnia-dev -c omnia rdkit openforcefield jupyter python=3.6`

Activate the newly created environment

- `$ conda activate openforcefield`

Download the workshop example from https://drive.google.com/open?id=1Im57D9_RhuswVIDZQZ4vHJodCLlZKKYI

Run the notebook

- `$ jupyter-notebook 2019_01_08_smirnoff_hands_on.ipynb`

Look at trajectory.pdb in the local directory

# Hands-on Session Troubleshooting

Preview documentation available at
https://open-forcefield-toolkit.readthedocs.io/en/topology/index.html

CLI-based miniconda download:

```
OSX: curl -s https://repo.continuum.io/miniconda/Miniconda3-latest-MacOSX-x86_64.sh
-o Miniconda3-latest-MacOSX-x86_64.sh
```