

# Reproducible Quantum Chemistry in JupyterLab

Chris Harris (Kitware)

@openchem

O'REILLY®

jupytercon

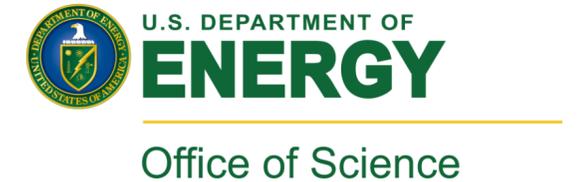
# Overview

- Scientific Use Case
- Why Jupyter?
- Approach
- Demo
- Architecture
  - Backend
  - Frontend
- Deployment
- Future



# Project and Team

- Department of Energy SBIR Phase II (Office of Science contract DE-SC0017193)
- Marcus D. Hanwell (Kitware)
  - Background in physics, experimental data, nanomaterials, visualization
- Chris Harris (Kitware)
  - Computer science, AI, HPC
- Bert de Jong (Berkeley Lab)
  - Developer of NWChem computational chemistry code, machine learning, quantum computing
- Johannes Hachmann (SUNY Buffalo)
  - Expertise in chemistry, machine learning, chemical library generation



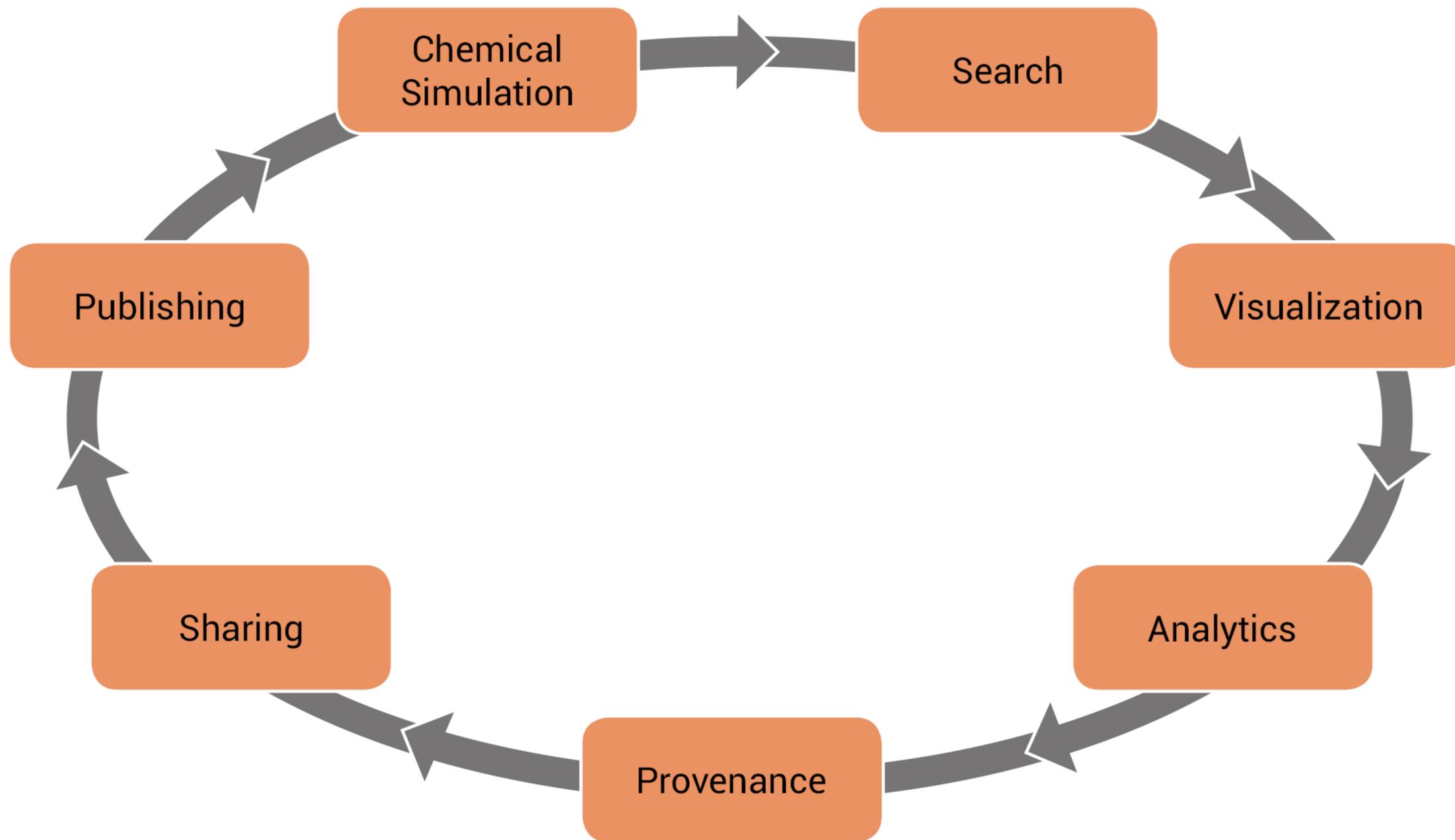
# Scientific Use Case

- Using quantum mechanics to characterize chemical systems
- Has seen vast improvements in both veracity and volume of data
- Lack of transparent and reproducible workflow

- Ad-hoc data management
- Complexity associated with codes
- The intricacies of HPC

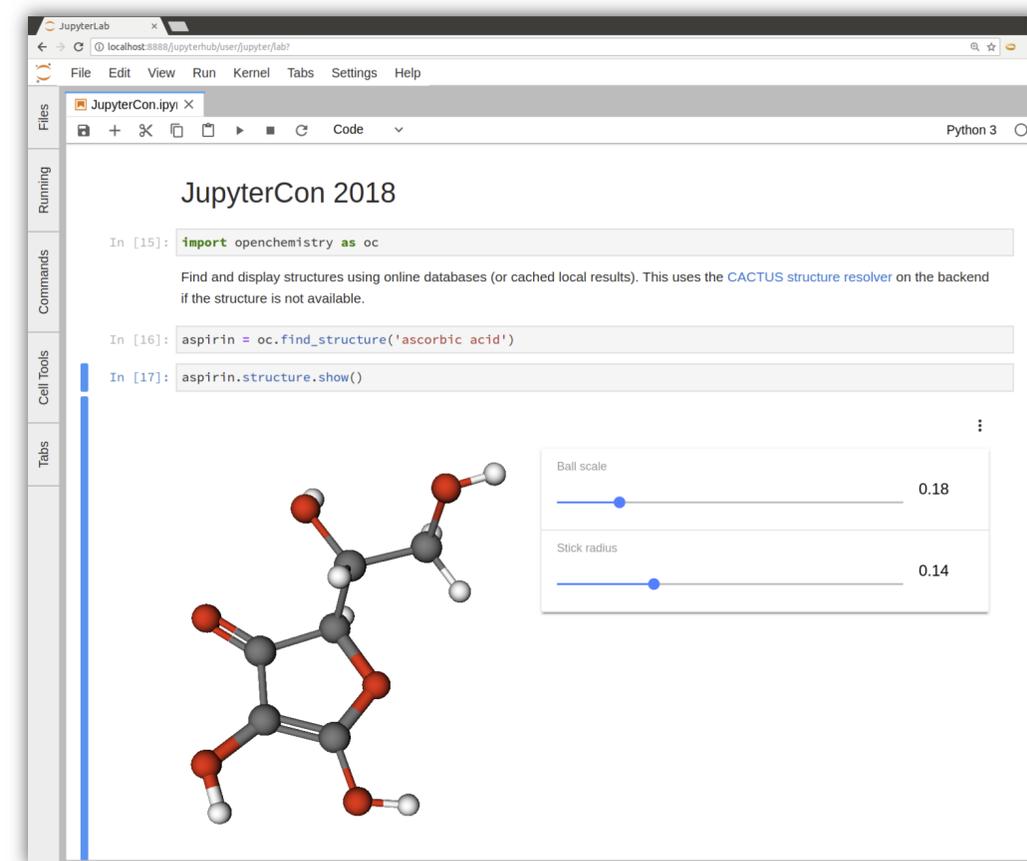
$$\left[ \frac{-\hbar^2}{2\mu} \nabla^2 + V(\mathbf{r}) \right] \Psi(\mathbf{r}) = E\Psi(\mathbf{r})$$

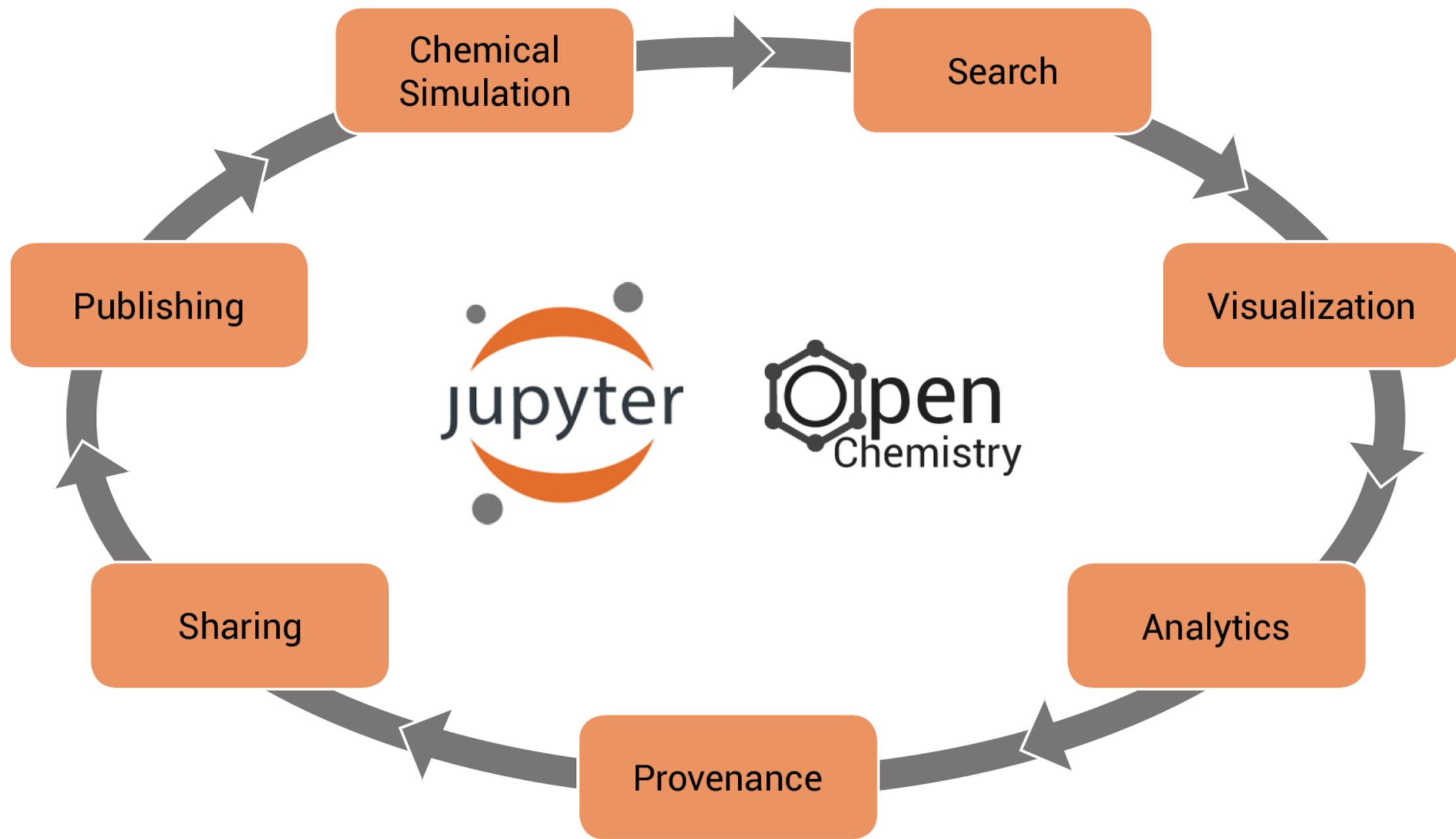
- Lack of integration with environments for visualization and analysis
- Need a platform to enable **end-to-end workflows** from simulation setup, simulation submission, right through to analytics and visualization of the result



# Why Jupyter?

- Supports interactive analysis while preserving the analytic steps
  - Preserves much of the provenance
- Familiar environment and language
  - Many are already familiar with the environment
  - Python is the language of scientific computing
- Simple extension mechanism
  - Particularly with JupyterLab
  - Allows for complex domain specific visualization
- Vibrant ecosystem and community





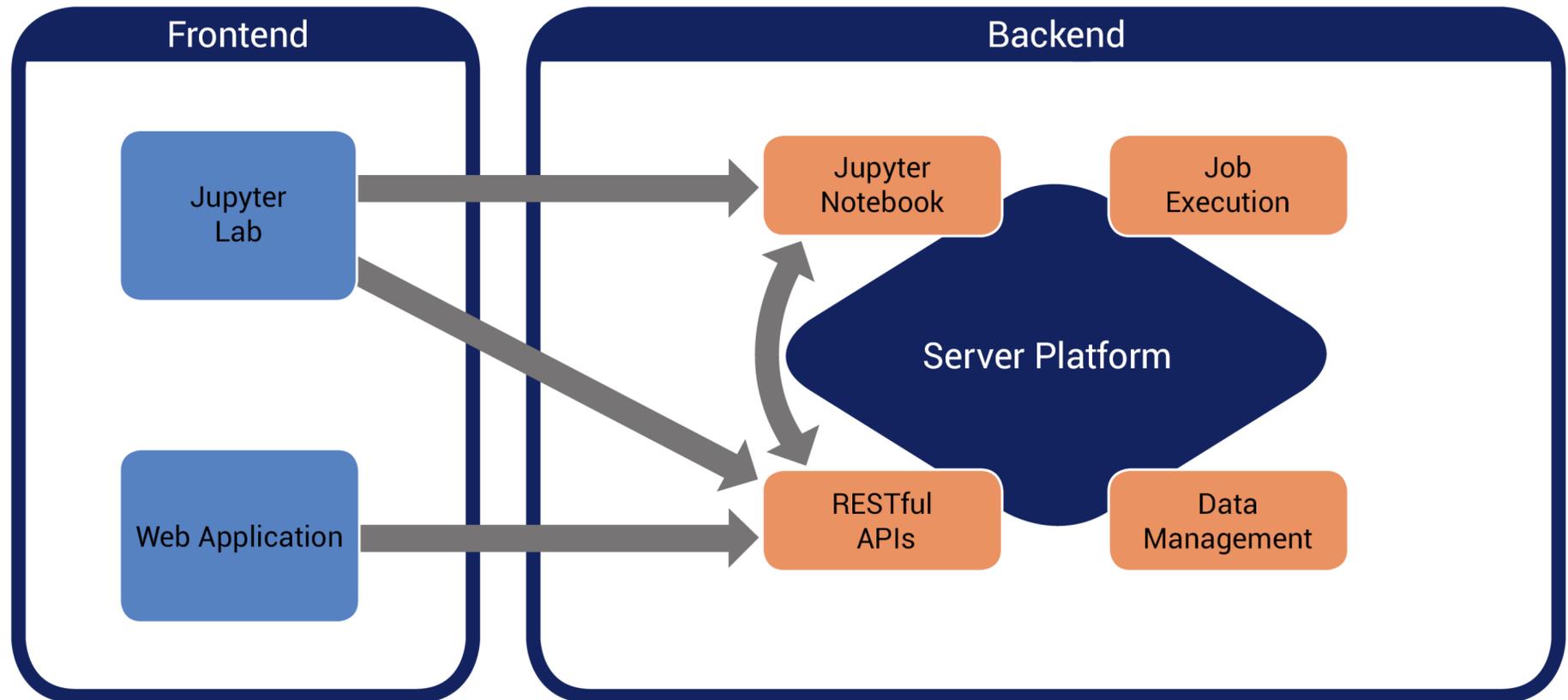
# Approach

- Data is the core of the platform
  - Start with simple but powerful data model and data server
- RESTful APIs everywhere
  - Allows access anywhere
    - Notebooks, web apps, command line, desktop applications, etc
- Jupyter notebooks for interactive analysis
  - Provide a simple high-level domain specific Python API for use within the notebooks
- Web application
  - Authentication, access control and user management
  - Launching/managing notebooks
  - Enable users to interact with data without having to launch notebooks

# Demo

# Architecture

- Backend
  - Data Management
  - Job Execution
  - Notebook management
- Frontend
  - Web components
  - JupyterLab Extensions
  - Web application



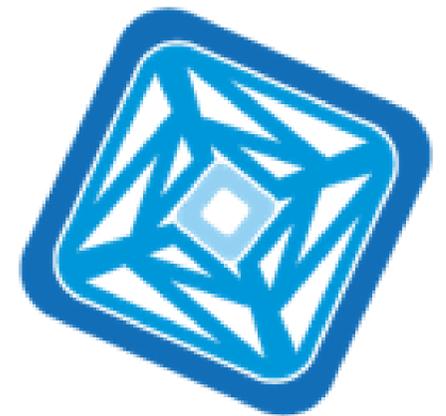
# Data Management

- Computational chemistry codes produce a wide variety of output
  - Often non-standard, even non-structured
  - Need to convert to single format
- Chemical JSON (CJSON)
  - Simple JSON format for representing chemical information
  - Efficient binary representation
  - MolSSI standard being developed
- Support export in multiple standard formats
  - Facilitate integration

```
1 {  
2   "chemical.json": 0,  
3   "name": "Ethane",  
4   "inchi": "1/C2H6/c1-2/h1-2H3",  
5   "formula": "C2H6",  
6   "atoms": {  
7     "elements": {  
8       "number": [1, 6, 1, 1, 6, 1, 1, 1]  
9     },  
10    "coords": {  
11      "3d": [1.185080, -0.003838, -0.987524,  
12            0.751621, -0.022441, -0.020839,  
13            1.166929, 0.833015, -0.569312,  
14            1.115519, -0.932892, -0.514525,  
15            -0.751587, -0.022496, -0.020891,  
16            -1.166882, -0.833372, -0.568699,  
17            -1.115691, -0.932608, -0.515082,  
18            -1.184988, 0.004424, -0.987522]  
19    },  
20  },  
21  "bonds": {  
22    "connections": {  
23      "index": [0, 1,  
24              1, 2,  
25              1, 3,  
26              1, 4,  
27              4, 5,  
28              4, 6,  
29              4, 7]  
30    },  
31    "order": [1, 1, 1, 1, 1, 1, 1]  
32  },  
33  "properties": {  
34    "molecular mass": 30.0690,  
35    "melting point": -172,  
36    "boiling point": -88  
37  }  
38 }  
39
```

# Data Management

- Girder
  - Web-based data management platform
    - Enable quick and easy construction of web applications:
      - Data organization and dissemination
      - User management & authentication
      - Authorization management
  - Extended via the development of plugins
    - Expose new data models and RESTful endpoints



# Job Execution

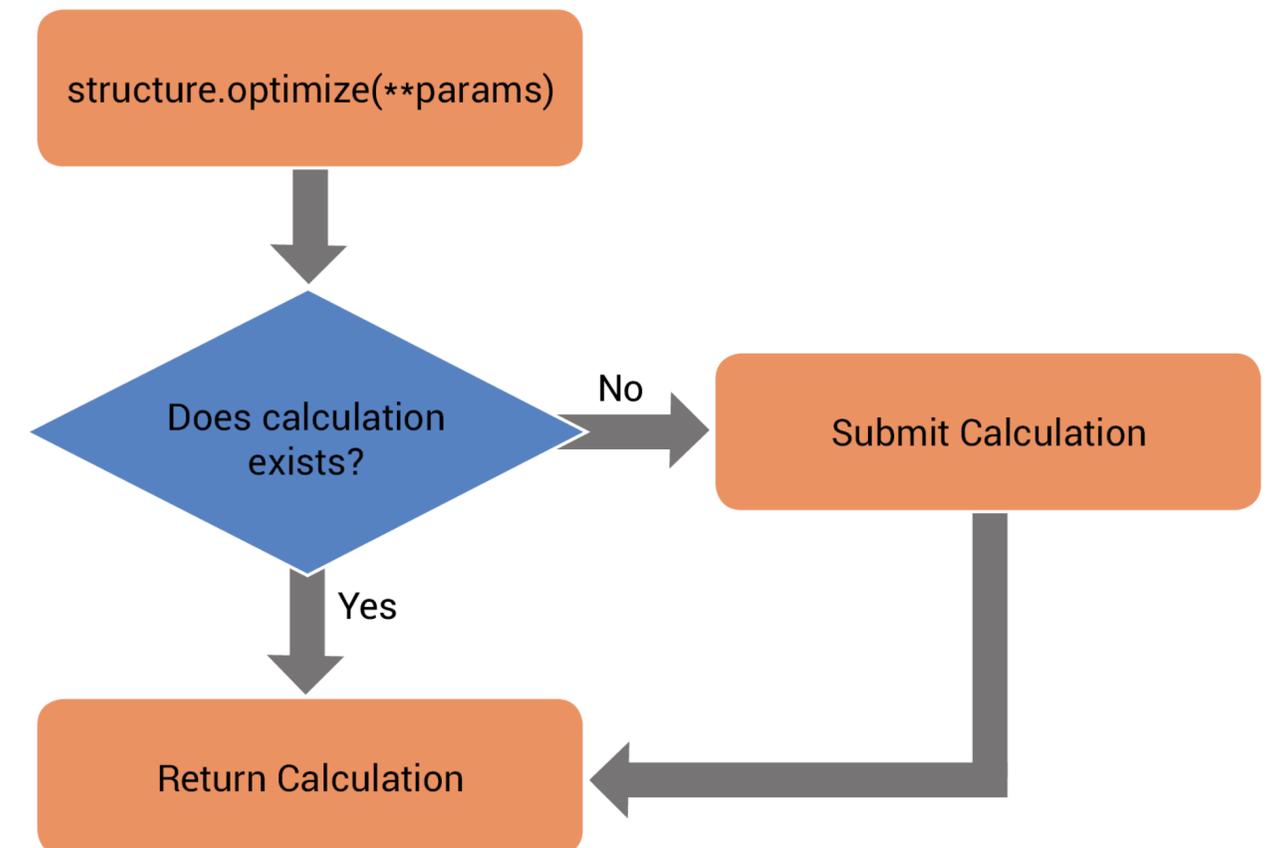
- What's involved in submitting a job to run on HPC resource?
  - Input generation
    - Code specific and often pretty esoteric
  - Moving the required data onto the resource
  - Generate submission script
    - Scheduler specific
  - Submit and monitor job
    - Scheduler specific
  - Post-processing or ingestion of result



*Focus on knowledge discovery, not job execution...*

# Job Execution

- Shield the end-user from the complexities
- Job execution is implicit with sane defaults
  - A result of requesting a given data set that doesn't exist
  - Concentrate on the data and analysis



# Job Execution

- Provide a scheduler abstraction
  - SGE, PBS and Slurm (+NEWT)
- Template input decks
- Distributed task queue to support long running operations
  - Job submission and monitoring
  - Support "offline" execution of jobs

```
In [12]: oc.show_free_energies(my_reactions, **calc_setup)
```

Pending Calculations

ID	Code	Type	Status
5b71947481798300016299bb	NWChem (version 27327)	vibrationalenergy	 RUNNING
5b7194728179830001629939	NWChem (version 27327)	vibrationalenergy	 COMPLETE
5b7194738179830001629975	NWChem (version 27327)	vibrationalenergy	 ERROR
5b71947381798300016299ae	NWChem (version 27327)	vibrationalenergy	 ERROR
5b7194718179830001629932	NWChem (version 27327)	vibrationalenergy	 RUNNING
5b719473817983000162998f	NWChem (version 27327)	vibrationalenergy	 ERROR
5b719472817983000162995b	NWChem (version 27327)	vibrationalenergy	 ERROR

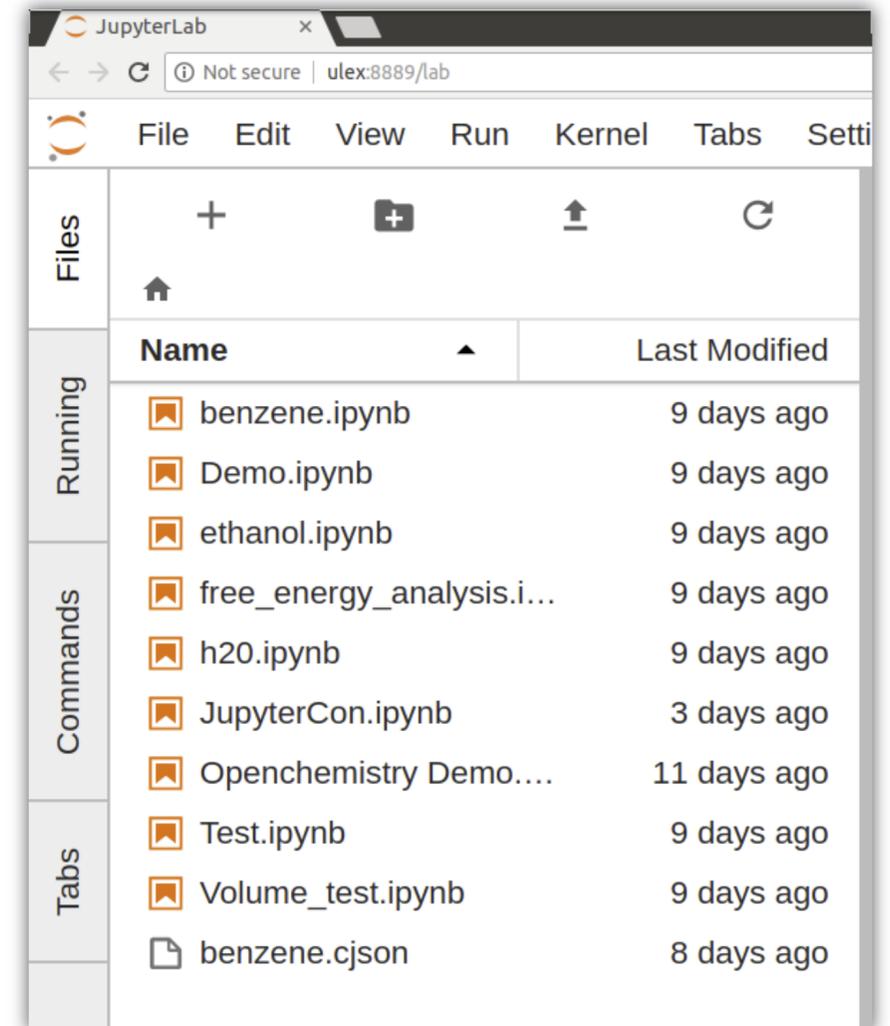
# Notebook Management

- JupyterHub to enable multi-user environment
  - DockerSpawner
    - Users do not need to have account on server
    - Simple deployment of complex Jupyter configurations
  - JupyterHub Girder authenticator
    - Allows cross-site authentication
  - Jupyter servers are launched with a simple redirect



# Notebooks as data

- The notebooks encode the workflow
  - Are as valuable as the calculation output
- Store in the data management system along with the output
  - Make them searchable
  - Make them available to others
  - Version
- Girder Contents Manager
  - Implements Jupyter Contents API
  - Notebooks can be stored in Girder



# Frontend

- Users have two interaction modes
  - Web application
  - JupyterLab

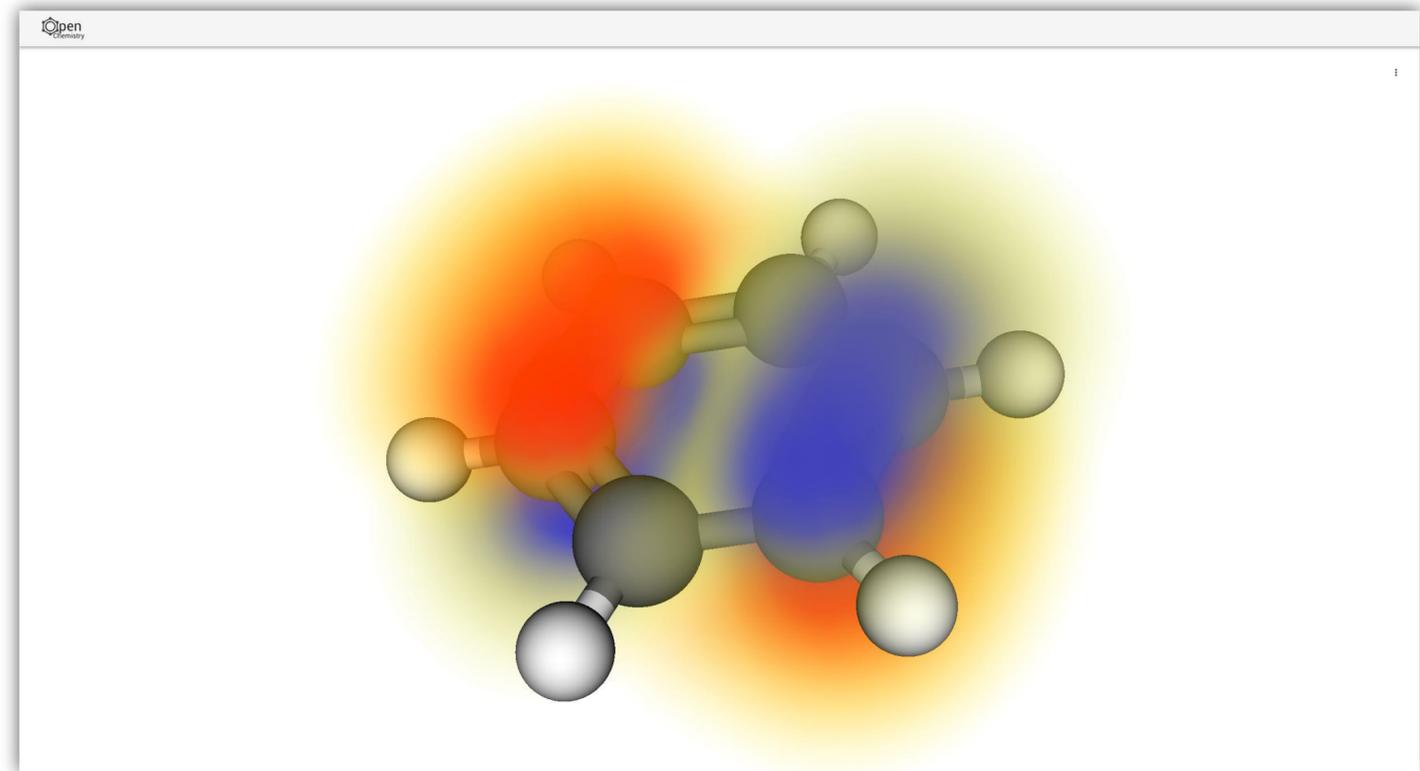
The image displays two overlapping browser windows. The background window is titled 'Open Chemistry' and shows a file browser interface with a list of files including 'benzene.ipynb', 'Demo.ipynb', 'ethanol.ipynb', 'free\_ener...', 'h2O.ipynb', 'JupyterCo...', 'Openche...', 'Test.ipynb', and 'Volume\_t...'. The foreground window is titled 'JupyterLab' and shows a Jupyter notebook titled 'JupyterCon 2018'. The notebook contains three code cells:

```
In [1]: import openchemistry as oc
Find and display structures using online databases (or cached local results). This uses the CACTUS structure resolver on the backend if the structure is not available.
In [2]: aspirin = oc.find_structure('aspirin')
In [3]: aspirin.structure.show()
```

Below the code cells, a 3D ball-and-stick model of an aspirin molecule is displayed, showing a benzene ring with a carboxylic acid group and an ester group.

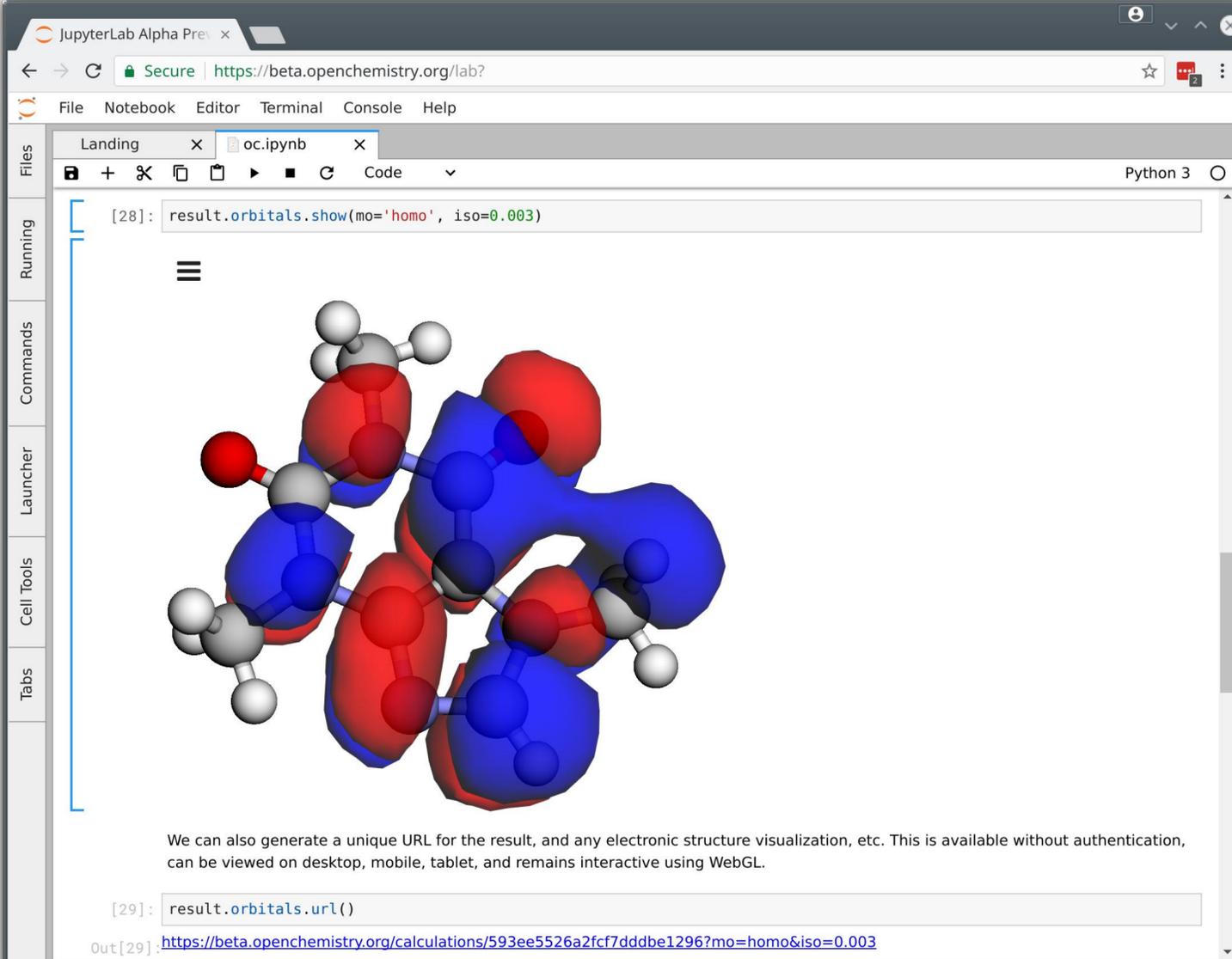
# Web components

- Allows the creation of new custom, reusable, encapsulated HTML tags
- stenciljs web component compiler
- Low level visualization components
  - Shared between JupyterLab extensions and web application
  - VTK.js for volume rendering
  - 3DMol.js for 3D chemical structures



# JupyterLab Extensions

- MIME renderer extensions
  - React/Redux components
  - Fetch data direct from data server
- Components are "thin" by design
- How to store "interactive" provenance?
- Adopted TypeScript



The screenshot shows a JupyterLab notebook interface. The browser address bar displays `https://beta.openchemistry.org/lab?`. The notebook has a menu bar with `File`, `Notebook`, `Editor`, `Terminal`, `Console`, and `Help`. The left sidebar contains navigation options: `Files`, `Running`, `Commands`, `Launcher`, `Cell Tools`, and `Tabs`. The main area shows a code cell with the following code:

```
[28]: result.orbitals.show(mo='homo', iso=0.003)
```

The output of the cell is a 3D molecular orbital visualization of a molecule, with red and blue lobes representing the positive and negative phases of the orbital. Below the visualization, there is a text block:

We can also generate a unique URL for the result, and any electronic structure visualization, etc. This is available without authentication, can be viewed on desktop, mobile, tablet, and remains interactive using WebGL.

Below this text, another code cell is shown:

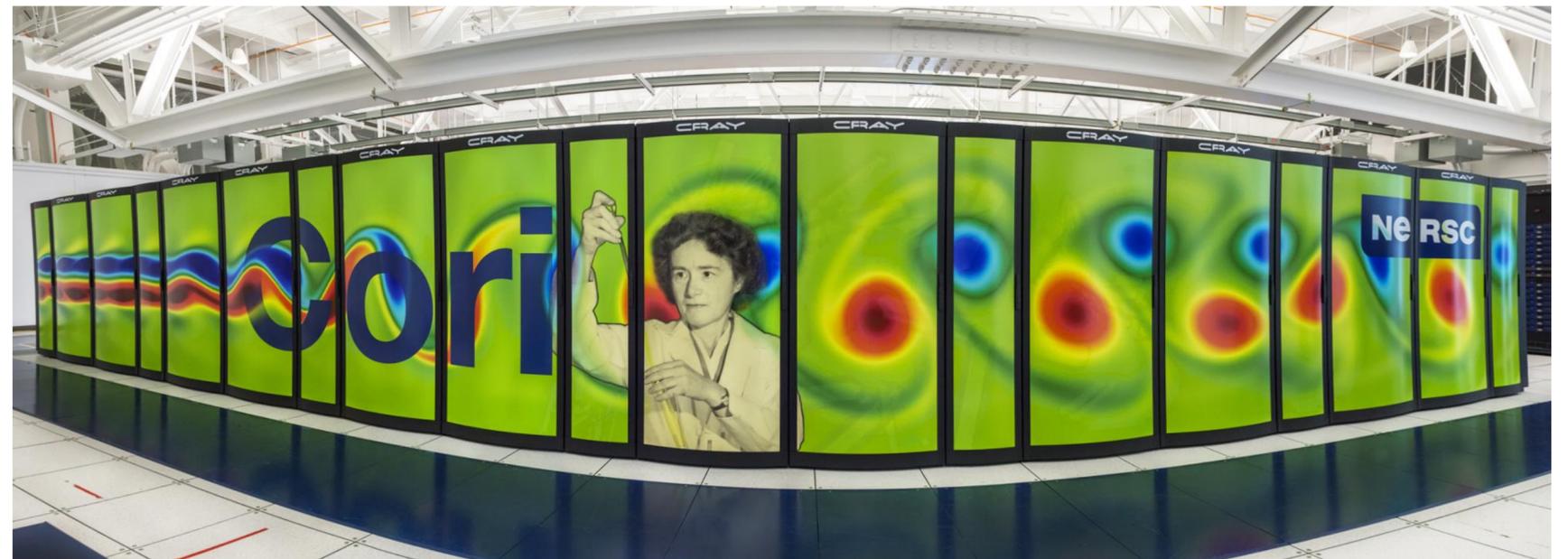
```
[29]: result.orbitals.url()
```

The output of this cell is a URL:

```
Out[29]: https://beta.openchemistry.org/calculations/593ee5526a2fcf7dddbe1296?mo=homo&iso=0.003
```

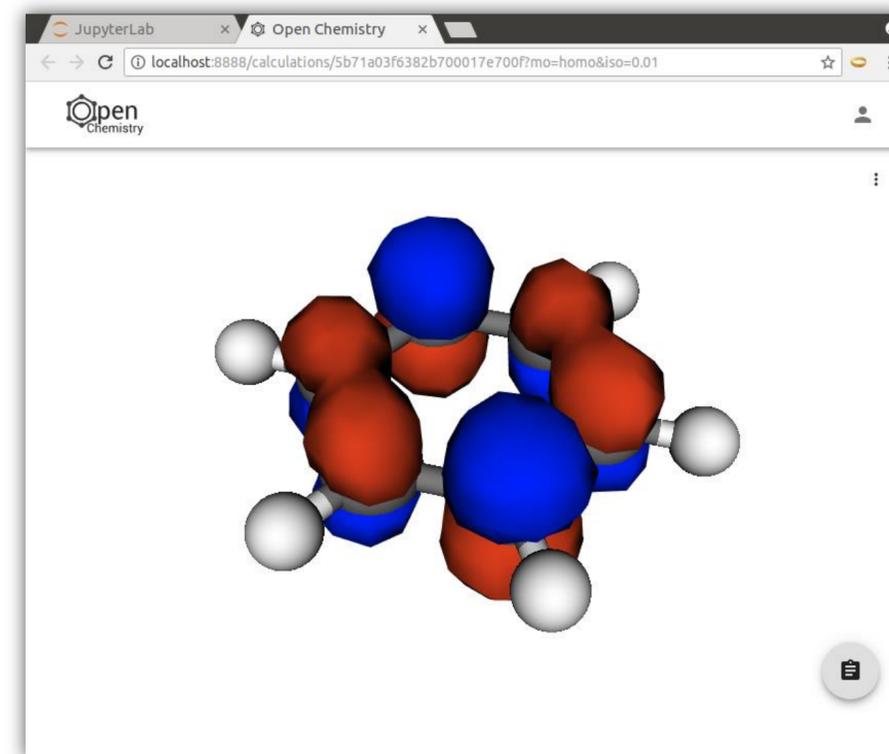
# Deployment

- docker-compose
- Ansible for runtime configuration
- AWS
  - Running jobs on small cloud cluster
- National Energy Research Scientific Computing Center (NERSC)
  - Uses NERSC login credentials
  - Jobs run on Cori



# Future Work

- Extend collaboration features
  - Fork notebooks
  - Real time editing of notebooks
- Integrate more computational chemistry and materials codes
  - Psi4, NWChemEx, Orca
- Add machine learning capabilities
  - Bulk downloads for training datasets
- Semantic web
  - Enriching data and make it more discoverable



# Thank you!

- Please come visit!
  - <https://openchemistry.org/>
  - <https://github.com/openchemistry/>

