

# Superfacility API



Bjoern Enders  
Data Science Workflows Architect  
NERSC/LBNL  
NERSC Data Day, Feb 22, 2024

# NERSC supports a large number of users and projects from DOE SC's experimental and observational facilities



Palomar Transient Factory  
Supernova



Planck Satellite  
Cosmic Microwave Background Radiation



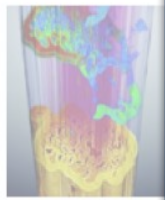
*# of Projects Analyzing Experimental Data or Combining Modeling and Experimental Data by SC Office*

**Superfacility:**  
Ecosystem of connected facilities, software and expertise to enable new modes of discovery

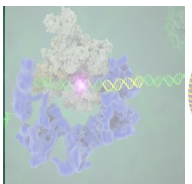
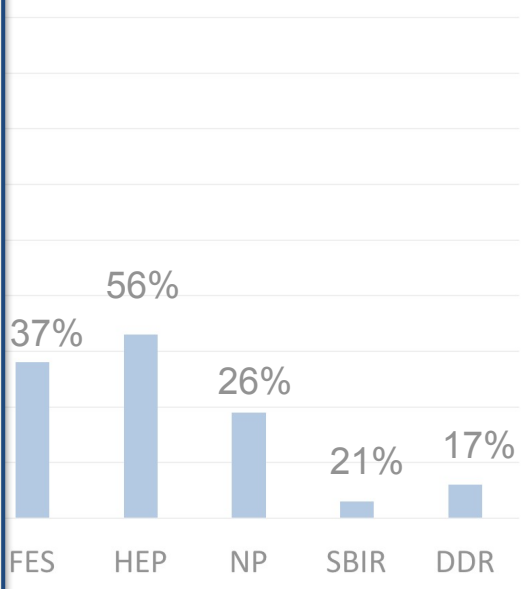
**Superfacility API:**  
An API into NERSC to embed HPC into cross-facility workflows. It is also a general purpose API for all NERSC users and projects.



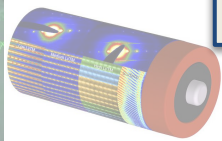
Dayabay Neutrinos



ALS Light Source



Cryo-EM



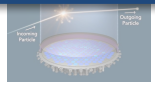
NCEM



DESI



LSST-DESC



LZ

~35% (235) of NERSC projects self identified as confirming the primary role of the project is to 1) analyze experimental data or; 2) create tools for experimental data analysis or; 3) combine experimental data with simulations and modeling

# Model case

Experiments at ext. facilities use high frame rate 2D detectors for their science.

Hosting data & compute on site has become increasingly demanding.

## Requirements

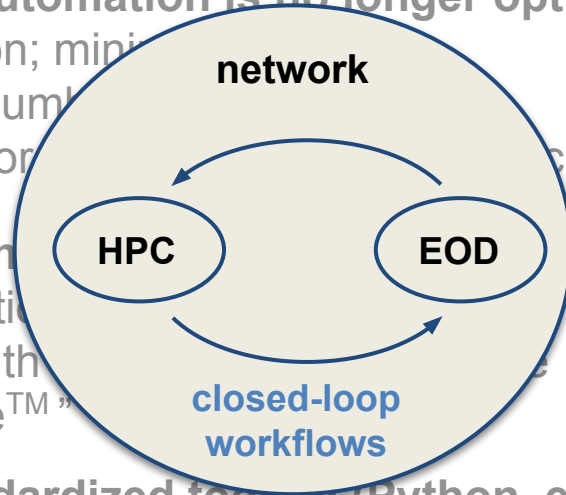
- Planning (HPC as reliable partner)
  - machine-readable status
- Resiliency (needs failover)
  - compatible interfaces
- Realtime (can't wait in queue)
  - workflow endpoint
- Services (portals, data, db)



1. Plan / Check availability of NERSC resource for experiment.
  - check status / accounts
2. Get raw data to NERSC, when experiment is live.
  - move data
3. Start analysis job quasi synchronous with data
  - submit job / monitor job
4. Gather feedback, ideally immediate.
  - download / execute command
5. Move data and results to archive after analysis.
  - move data

# Why an API?

- **Meets a critical need; automation is no longer optional**
  - Unattended operation; minimize human intervention
  - Track/submit large number of jobs
  - Interface with collaborators and other machines
- **NERSC becomes “machine”**
  - Enables easier creation of workflows
  - Allows integration with other systems
    - “NERSC inside™”
- **Less DIY: simpler, standardized tooling (Python, etc)**
  - Stable refactor target for established projects or easier on-ramp for new ones
  - Contribute to HPC interface standards for portability
  - Authentication and security models



## Drivers:

- **Complex workflows**
- **Data-driven projects**
- **Real-time compute and streaming data from instruments**
- **Automation**

# What is the API good for?

**Vision: all NERSC interactions are callable to facilitate seamless, automated "NERSC inside" workflows without a human in the loop.**

## Endpoints

`/status`  
`/account`  
`/compute`  
`/storage`  
`/tasks`  
`/storage`  
`/utilities`

### Example request

```
curl -X 'GET' \  
  'https://api.nersc.gov/api/v1.2/ status/perlmutter' \  
  -H 'accept: application/json'
```

### Result:

```
{  
  "name": "perlmutter",  
  "full_name": "Perlmutter",  
  "description": "System is active",  
  "system_type": "compute",  
  "notes": [],  
  "status": "active",  
  "updated_at": "2023-03-02T18:00:00-08:00"  
}
```

**API is used both internally at NERSC and externally for our users**



The following NERSC resources that Jupyter depends on appear to be in maintenance or having issues. This may impact Jupyter. See the [NERSC MOTD](#) for further information.

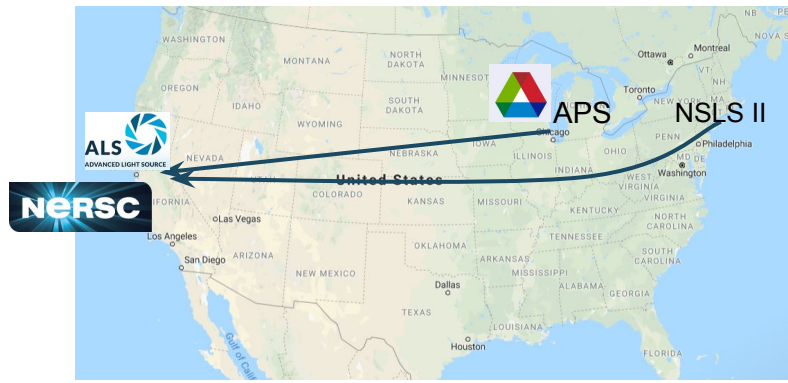
Perlmutter status: degraded

Action	Manual steps	With SuperFacility API
Check status	Test SSH or ping specific services for status	Query the <code>/status</code> API endpoint if resources are active.
Submit job	SSH in and submit jobs with <code>sbatch ...</code>	Create jobs using POST calls from a script or Spin service to the <code>/compute</code> endpoint.
Monitor job	SSH in (again) and do <code>queue   grep   sort   ...</code>	Consult the <code>/compute</code> and <code>/tasks</code> endpoints.
Plan ahead	Read the NERSC MOTD to see if any down time is planned	Query the <code>/status/outages/planned</code> API endpoint for planned outages
Move data	SSH in and run file transfer tools to move data	POST to the <code>/storage</code> API endpoint.
Check account	Log into "Iris" (our accounting web app) and check allocation account balance.	Query the <code>/account</code> API to get the same information.

# Model use

Experiments at ext. facilities use high frame rate 2D detectors for their science.

Hosting data & compute on site has become increasingly demanding.

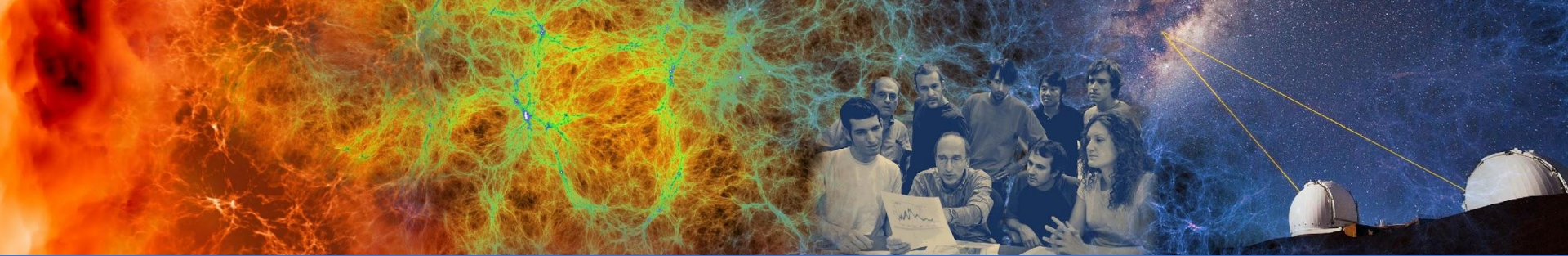


## Requirements

- Planning (HPC as reliable partner)
  - machine-readable status
- Resiliency (needs failover)
  - compatible interfaces
- Realtime (can't wait in queue)
  - workflow endpoint
- Services (portals, data, db)

1. Plan / Check availability of NERSC resource for experiment.
  - `/status (/reservations)`
2. Get data to NERSC, when experiment is live.
  - `/storage`
3. Start analysis job quasi synchronous with data
  - `/compute /tasks`
4. Gather feedback, ideally immediate.
  - `/utilities /storage`
5. Move data and results to archive after analysis.
  - `/storage`





# The API



BERKELEY LAB



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



# Projects using the SF API

Since release in 2022 (Feb 2023 numbers)

- 27 non-staff users made clients
- members of 40 different non-staff projects.

~ 12M logged requests from May 2022 - Feb 2023  
= one request every 2 sec

## Examples



automated data processing for serial diffraction/scattering workflows

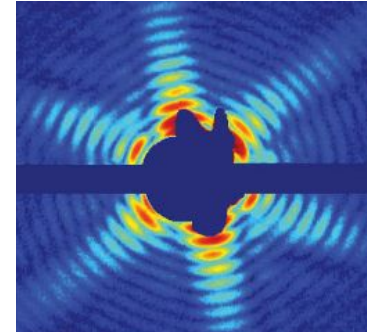


automated processing (HTC) of 4D STEM scan data with the [Distiller](#) app.

9



automated data processing between each of their plasma shots to reconstruct the shape and properties of the plasma.



## Linac Coherent Light Source, a BES User Facility

- How does photosynthesis happen?
  - How do drugs dock with proteins in our cells?
  - Why do jet engines fail?
- 
- By ~2024, 5% of LCLS experiments will produce >10PB data/year, and some need >128PFlops for real-time analysis



# Linac Coherent Light Source - Serial Diffraction/Scattering Workflows

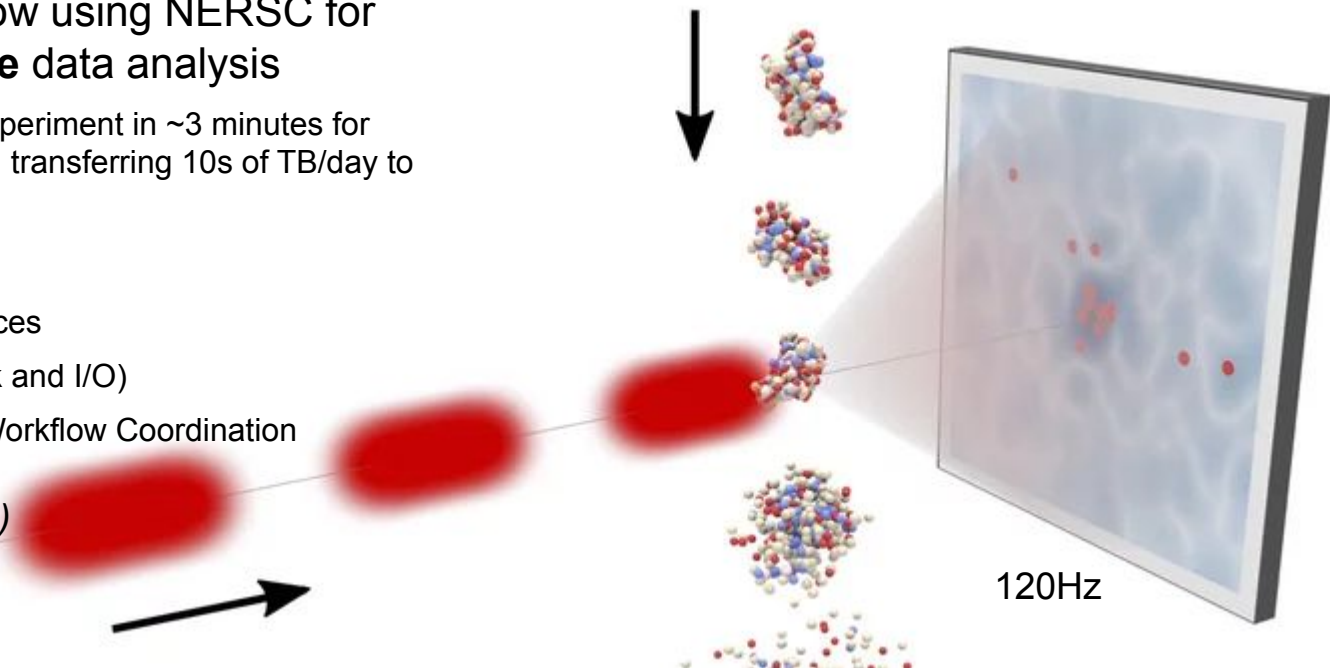
- LCLS experiments are now using NERSC for semi-automated **real-time** data analysis
  - Can analyze a 5 minute experiment in ~3 minutes for feedback to beamline staff, transferring 10s of TB/day to NERSC
- LCLS workflow requires
  - Urgent Computing Resources
  - High-Speed Data (Network and I/O)
  - Realtime Monitoring and Workflow Coordination

*J. Blaschke et. al (ExaFEL team)*

<https://arxiv.org/abs/2106.11469>

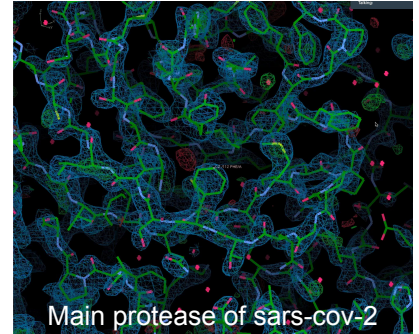
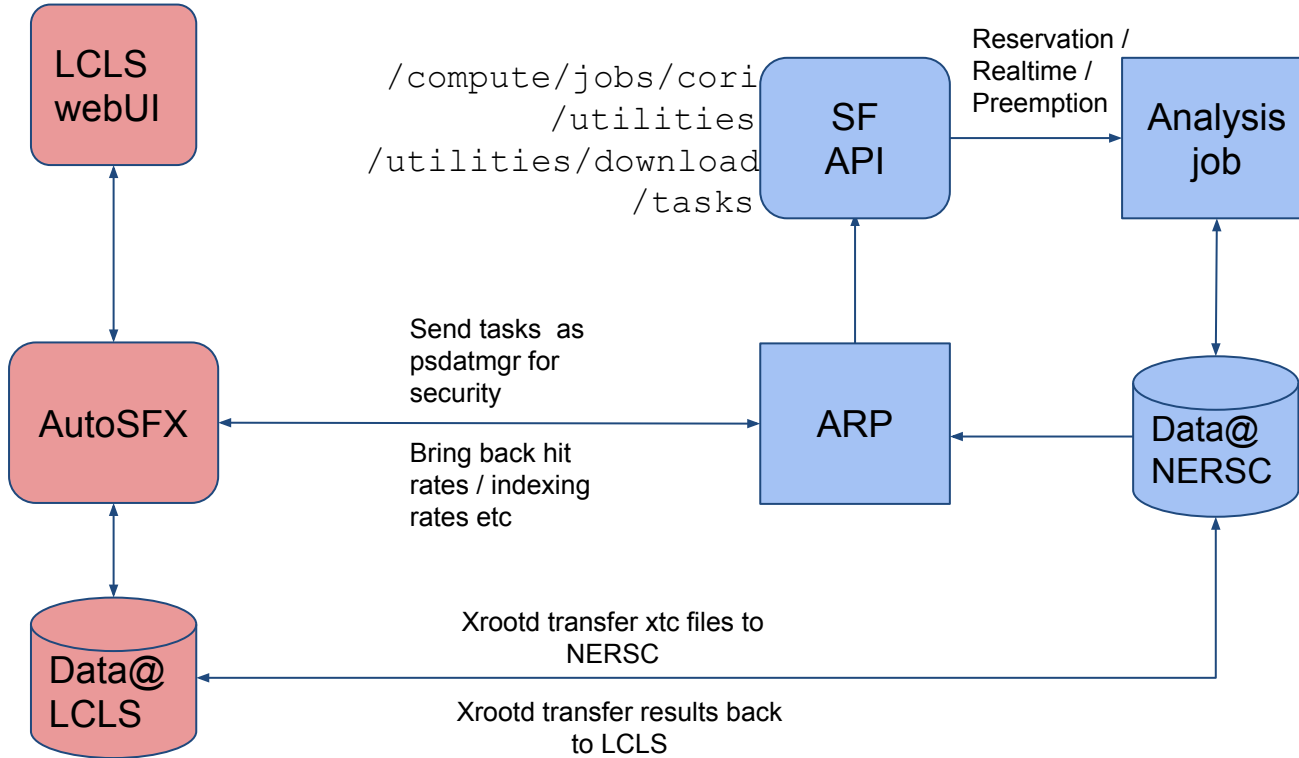


Winner of CUG21  
Best Paper award

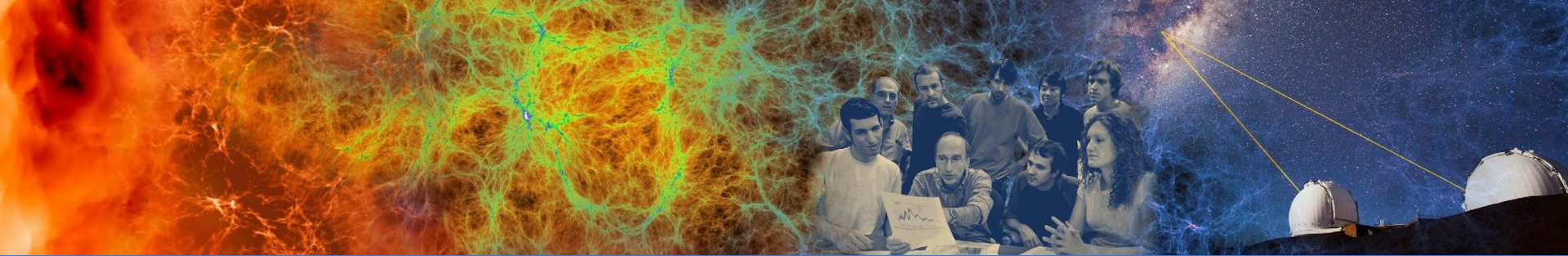


Source: <https://www.mpibpc.mpg.de/9660732/SM-Ultrafast-XRay-Diffraction>

# SF API in LCLS workflow



Adapted from  
Chuck Yoon



# Technical overview



BERKELEY LAB



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science

# Interface documentation

<https://api.nersc.gov/api/v1.2>

- Interactive, up-to-date and self-documenting
- See endpoints, payloads, example code
- Works with any dev environment
- End user docs and examples:  
<https://docs.nersc.gov/services/sfapi/>

## Authorization

- Some endpoints are public and don't need an access token (no lock icon)
- For authorized endpoints (with lock icon), click the "Authorize" button at the top page and paste the access token



The screenshot shows the NERSC SuperFacility API documentation page. At the top, it says "NERSC SuperFacility API 1.2 OAS3" with a link to the OpenAPI spec. Below that, there's a description of the API and links for terms of service and contact information. A "Servers" dropdown menu is set to "/api/v1.2", and an "Authorize" button is visible. The main content is a list of API endpoints grouped into sections: "meta", "status", "account", and "compute". Each endpoint is shown with its HTTP method, path, description, and a lock icon indicating if it requires authorization. The "account" section is expanded, showing endpoints like "/account/projects", "/account", "/account/roles", "/account/groups", and "/account/groups/{group}". The "compute" section is also expanded, showing endpoints like "/compute/jobs/{machine}" and "/compute/jobs/{machine}".

# Users need to create a SF API client in Iris.

- Some endpoints (e.g. /status) are public and don't require authentication.
- For the authenticated part of the API, users need to create client credentials (via MFA in iris.nersc.gov) and exchange these for short-lived API access tokens.

**Create a New SuperFacility API Client**

Read/Write/Execute Client:  
30 day lifetime (until 12/9/2022), full endpoint access  
Useful for automating complex workflows requiring all API functions: job submission and monitoring, file uploads/downloads, data transfer, etc. Intended for use with workflow orchestration systems installed on institutionally-managed computer systems or clusters.

This option requires NERSC approval. Submit a request for access.

Read-only Client:  
180 day lifetime (until 5/8/2023), limited endpoint access  
Useful for monitoring scripts or dashboards that require read-only API functions only: job monitoring, file download, etc.

Client Name  
Client name

Comments  
Notes about this client

User to create client for  
perclus

Source IP Range (CIDR)  
IP range in CIDR format IP Presets  
IPv4 ranges must have /24 or greater suffix

Create Client Cancel



**Create a New SuperFacility API Client**

New Client Id  
36vggkqzuvk

Your Private Key  
[\*ky\*: "RSA", "n": "z6acqby7O3kUfX2wCx8VPr00B86wSS4-JTUy83aOKkzZsOQB\_c1rnbHc2Yc1r2h-M87849gdcdfra2g9IQ\_B5cGWf7LjH0cxt4WQFS6zfggereVDW2Qt1Ulx0i8J5V4k4amhb2JQ  
Please store your private key in a secure location. **It will not be displayed again.**

Your Private Key (PEM format)  
-----BEGIN RSA PRIVATE KEY-----  
MIIEogIBAAKAAQEAz6acqby7O3kUfX2wCx8VPr00B86wSS4+JTUy83aOKkzZsOQ  
Please store your private key in a secure location. **It will not be displayed again.**

Your Public Key (PEM format)  
-----BEGIN PUBLIC KEY-----  
MIBjANBgkqhkiG9w0BAQEFAAOCAQAMIBICgKAAQEAz6acqby7O3kUfX2wCx8

Create Client Close



jose/openssl  
authlib (Python)

**3** Sign client assertion  
with private key

https://oidc.nersc.gov

client  
assertion  
(JWT)

access  
token  
(10 min)

api.nersc.gov/api/v1.2

**4** Use access token for API

- 1** Iris > Profile  
> Superfacility API clients  
> new client

- 2** Store oauth client  
credentials in safe place



# Live walkthrough





```
Poll 04:{'id': '2236', 'status': 'completed', 'result': '{"status": "ok", "jobid": "484275", "error": null}'}  
{'status': "ok", "jobid": "484275", "error": null}
```

Check queue status, wait for job to complete.

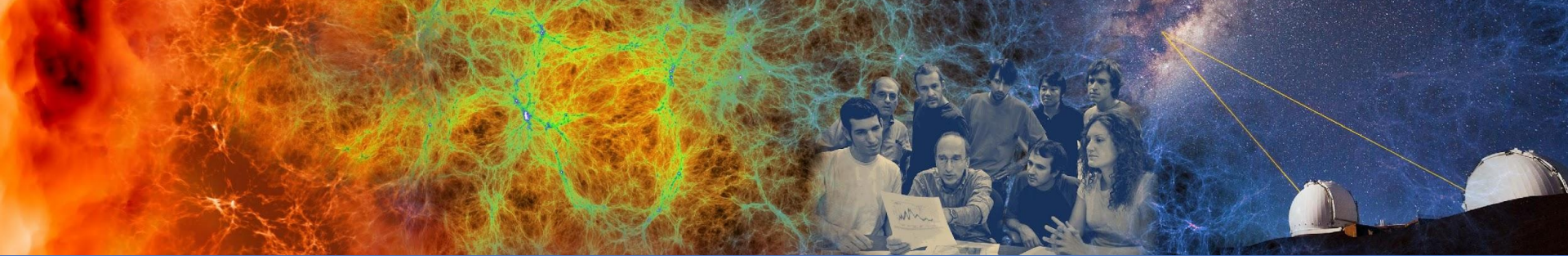
```
[338]: #print(json.dumps(api("compute/jobs/perlmutter/"+jobid+"?sacct=true"), indent=2))  
print(api("compute/jobs/perlmutter/"+jobid+"?sacct=true")['output']['state'])
```

RUNNING

Read from the slurm output file

```
[334]: slurmfile = home+"/apidemo/slurm-"+jobid+".out"  
response = api("utilities/command/perlmutter", {"executable": "tail -n 20 "+slurmfile })  
if isinstance(response, Task):  
    print(response.wait_for_result()['output'].strip())
```

```
Poll 03:{'id': '2237', 'status': 'completed', 'result': '{"status": "ok", "output": " * subpix          : linear\n * update_object_first : linear\n * overlap_converge... : 0.05\n * overlap_max_itera... : 10\n * probe_inertia       : 1e-09\n * object_inertia      : 0.0001\n * fourier_power_bound : None\n * fourier_relax_factor : 0.05\n * obj_smooth_std      : None\n * clip_object         : None\n * probe_center_tol    : None\n * compute_log_likel... : True\n * probe_update_cuda... : False\n * object_update_cuda... : True\n * fft_lib              : reikna\n * alpha                : 1.0\n * name                 : DM_pycuda\n=====  
\\nIteration #10 of DM_pycuda :: Time 1.249\nErrors :: Fourier 5.94e+01, Photons 4.73e+01, Exit 3.51e+01\n", "error": null}'}  
{'status': "ok", "output": " * subpix          : linear\n * update_object_first : linear\n * overlap_converge... : 0.05\n * overlap_max_it  
era... : 10\n * probe_inertia       : 1e-09\n * object_inertia      : 0.0001\n * fourier_power_bound : None\n * fourier_relax_factor : 0.05\n * obj_smooth_std      : None\n * clip_object         : None\n * probe_center_tol    : None\n * compute_log_likel... : True\n * probe_upda  
te_cuda... : False\n * object_update_cuda... : True\n * fft_lib              : reikna\n * alpha                : 1.0\n * name                 :  
DM_pycuda\n=====  
\\nIteration #10 of DM_pycuda :: Time 1.249\nErrors :: Fo  
urier 5.94e+01, Photons 4.73e+01, Exit 3.51e+01\n", "error": null}  
 * subpix          : linear  
  * update_object_first : True  
  * overlap_converge... : 0.05  
  * overlap_max_itera... : 10  
  * probe_inertia       : 1e-09  
  * object_inertia      : 0.0001  
  * fourier_power_bound : None  
  * fourier_relax_factor : 0.05  
  * obj_smooth_std      : None  
  * clip_object         : None  
  * probe_center_tol    : None  
  * compute_log_likel... : True
```



# The sfapi\_client Python library



BERKELEY LAB



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science

# Using Python to interact with the API

- The `sfapi_client` is a pythonic way to interact with the API
  - Many NERSC users are familiar with python already
- Uses pydantic to take API responses and make equivalent python objects
  - Can easily be updated with changes to the API specification
- Both an Asynchronous and Synchronous Client
  - Allows for a wider range of programming styles
  - More advanced users can interact Asynchronously
  - Still supports all (python) users
  - Easily place the client in existing tools

# Using Python to interact with the API

- Code on Github
  - [https://github.com/NERSC/sfapi\\_client](https://github.com/NERSC/sfapi_client)
- Documentation Page
  - [https://nersc.github.io/sfapi\\_client](https://nersc.github.io/sfapi_client)
- Nersc User Slack
  - <https://www.nersc.gov/users/NUG/nersc-users-slack>
  - #sfapi\_client
- [help.nersc.gov](http://help.nersc.gov)
  - Include `sfapi\_client` in your title or description

# sfapi\_client library needs to be installed

- First load the python module

```
module load python
```

- Then create custom environment and install sfapi\_client

```
conda create -n sfapi-demo python=3.11
```

```
conda activate sfapi-demo
```

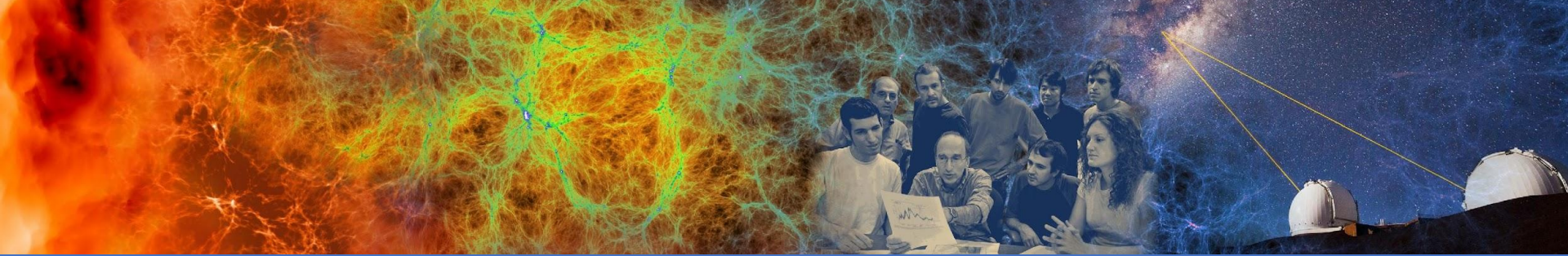
```
pip install "sfapi_client==0.0.8" ipykernel
```

```
python -m ipykernel install \
```

```
--user --name sfapi --display-name sfapi
```

# Live walkthrough





Thank you for your attention!

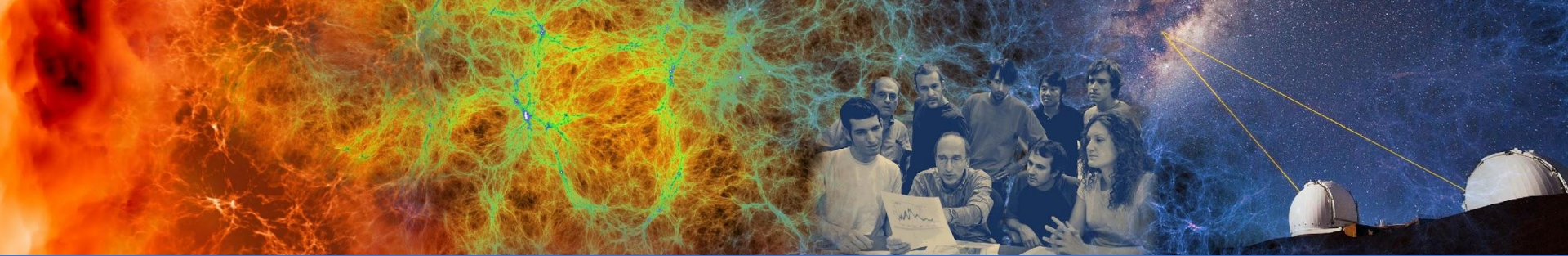


BERKELEY LAB



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



# Technical overview



BERKELEY LAB



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science



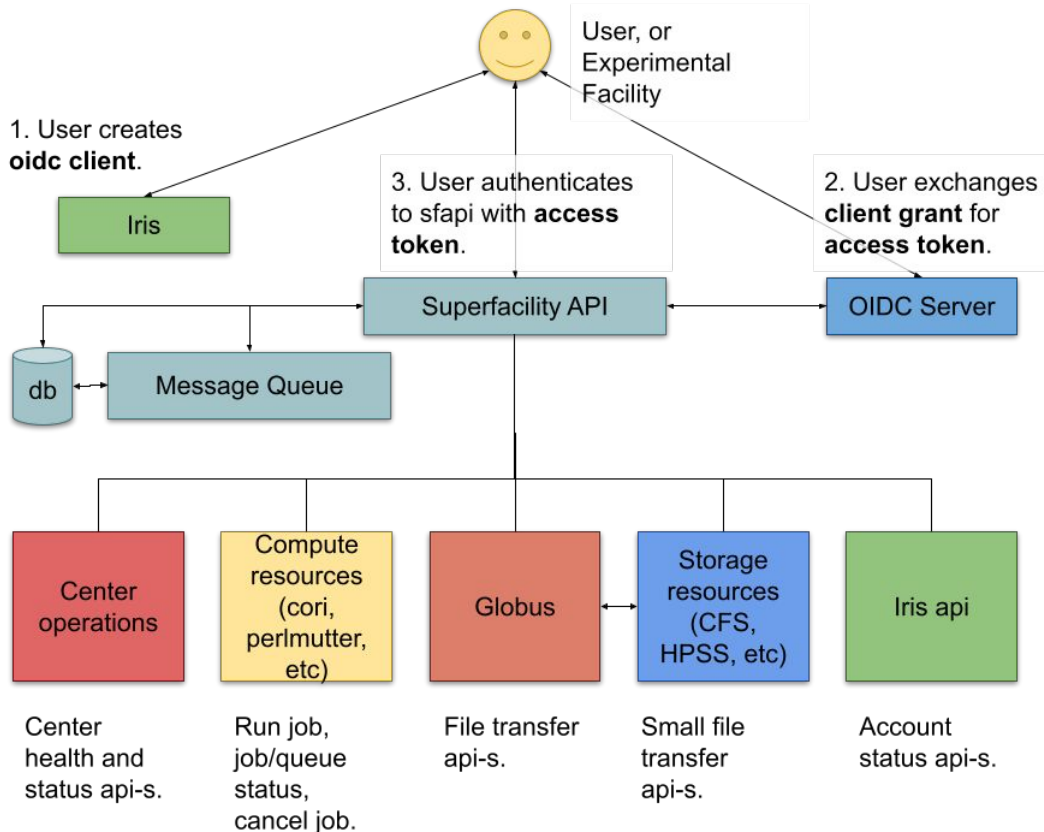
# Superfacility API architecture

The API consists of:

- JWT-based authentication
- Each REST endpoint a thin layer on top of a NERSC resource
- Message Queue (RabbitMQ) to handle Asynchronous tasks
- Redis & PostgreSQL database for application state

Docker images on NERSC cloud service.

Hides actual complexity of interacting with NERSC behind simple API.



# Example API process flow

Using API client and access token for authentication

POST /compute/jobs to start job (async)

GET /task/id to get status of async task (job submission)

GET /compute/job/<job\_id> to see job status

