Superfacility API
NERSC supports a large number of users and projects from DOE SC’s experimental and observational facilities.

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.

~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; minimizing HITL
  - Track/submit large number of jobs
  - Interface with collaborations, workflows and machines

- NERSC becomes “machine readable”
  - Enables easier creation of UIs, portals, etc.
  - Allows integration with control/analysis software
  - “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
<table>
<thead>
<tr>
<th>Action</th>
<th>Manual steps</th>
<th>With SuperFacility API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check status</td>
<td>Test SSH or ping specific services for status</td>
<td>Query the /status API endpoint if resources are active.</td>
</tr>
<tr>
<td>Submit job</td>
<td>SSH in and submit jobs with <code>sbatch ...</code></td>
<td>Create jobs using POST calls from a script or Spin service to the /compute endpoint.</td>
</tr>
<tr>
<td>Monitor job</td>
<td>SSH in (again) and do `squeue</td>
<td>grep</td>
</tr>
<tr>
<td>Plan ahead</td>
<td>Read the NERSC MOTD to see if any down time is planned</td>
<td>Query the /status/outages/planned API endpoint for planned outages</td>
</tr>
<tr>
<td>Move data</td>
<td>SSH in and run file transfer tools to move data</td>
<td>POST to the /storage API endpoint.</td>
</tr>
<tr>
<td>Check account</td>
<td>Log into &quot;Iris&quot; (our accounting web app) and check allocation account balance.</td>
<td>Query the /account API to get the same information.</td>
</tr>
</tbody>
</table>
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
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.

Examples

automated data processing for serial diffraction/scattering workflows

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

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)*


Winner of CUG21 Best Paper award

Source: [https://www.mpibpc.mpg.de/9660732/SM-Ultrafast-XRay-Diffraction](https://www.mpibpc.mpg.de/9660732/SM-Ultrafast-XRay-Diffraction)
SF API in LCLS workflow

LCLS webUI

AutoSFX

Data@LCLS

SF API

Analysis job

ARP

Data@NERSC

Reservation / Realtime / Preemption

Send tasks as psdatmgr for security

Bring back hit rates / indexing rates etc

Xrootd transfer xtc files to NERSC

Xrootd transfer results back to LCLS

Main protease of sars-cov-2

Adapted from Chuck Yoon
Technical overview
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
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.

1. Iris > Profile > Superfacility API clients > new client
2. Store oauth client credentials in safe place
3. Sign client assertion with private key
4. Use access token for API

https://oidc.nersc.gov
client assertion (JWT) (10 min)

access token

api.nersc.gov/api/v1.2

jose/openssl authlib (Python)
Live walkthrough
Check queue status, wait for job to complete.

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

Read from the slurm output file

```python
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()
```
The sfapi_client Python library
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
- Documentation Page
  - https://nersc.github.io/sfapi_client
- Nersc User Slack
  - https://www.nersc.gov/users/NUG/nersc-users-slack
  - #sfapi_client
- 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!
Technical overview
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