



Uniform Resource Access Compute and Cloud Resources at JSC

2025-13-05 | Bernd Schuller, Björn Hagemeier | Juelich Supercomputing Centre



Workflows



Jobs

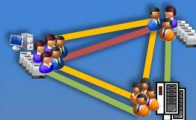


Data Management



Discovery

Services



Federations



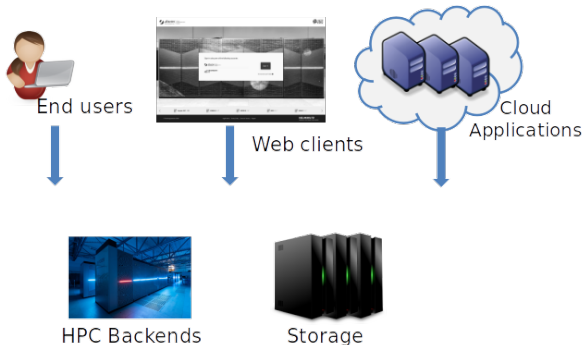
Part I: UNICORE

Motivation

Secure, flexible way for accessing HPC systems

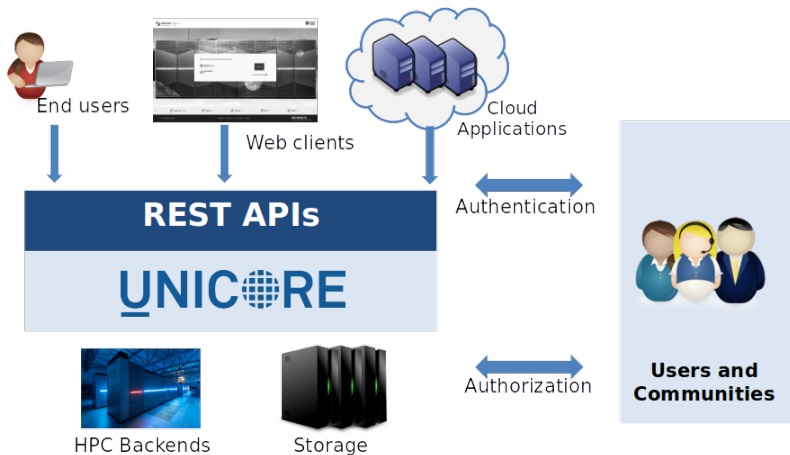
For many applications, SSH is not suitable

- Key management, MFA, ...
- Inconvenient for automation tasks
- Inconvenient for web/cloud applications



UNICORE

UNiform Interface to COmpute REsources



Application example: Jupyter @ JSC

HPC in your web browser

- Start Jupyter Labs on JSC's HPC systems or on a Cloud VM
- `https://jupyter-jsc.fz-juelich.de/`
- Other instances like Jupyter4NFDI
- UNICORE is an integral part of the architecture
 - Used to launch Jupyter kernels on login or compute nodes
 - Correctly assign user's groups and projects



The screenshot shows the JupyterLab web interface in a browser. The address bar shows `https://jupyter-jsc.fz-juelich.de/hub/home`. The page header includes the Jülich Forschungszentrum logo, the text "JÜLICH SUPERCOMPUTING CENTRE", and navigation links for "JupyterLab" and "Documentation". A user profile dropdown for "Bernd Schuller" is visible in the top right. Below the header, a message states: "You can configure your existing JupyterLabs by expanding the corresponding table row." A table with the following structure is displayed:

	Name	Configuration	Status	Action
+	New JupyterLab			
▼	cloud	System JSC-Cloud Option 4.3	running	Open Stop

At the bottom of the interface, a horizontal bar shows the status of various systems: Jupyter-JSC (104), JEDI (2), JUWELS (74), and JURECA (38). The footer contains copyright information for Forschungszentrum Jülich, legal notices, and the Helmholtz logo with the tagline "RESEARCH FOR GRAND CHALLENGES".

UNICORE

Secure, flexible way for accessing HPC systems

- Provides APIs for securely accessing to compute and storage
- Create, submit and monitor **jobs** (i.e. computational tasks)
- Data access, transfer and management features
- Transparently handles user identities, accounts and projects
- Multi-site workflows
- Abstract system specifics



The basic unit of execution

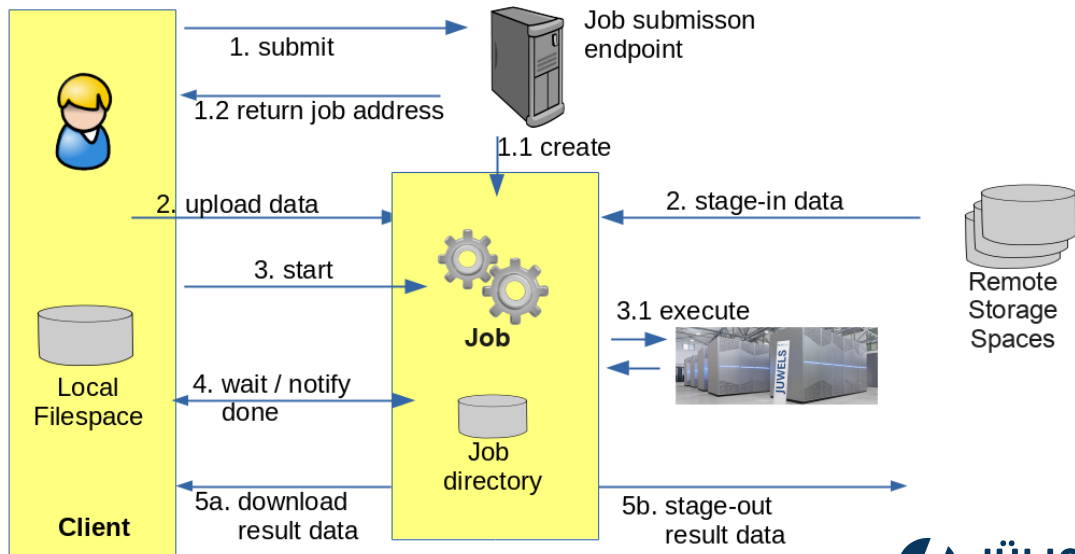
- Each job automatically gets its own working directory
- Job types
 - BATCH (default) run user task via Slurm (sbatch)
 - ON_LOGIN_NODE
 - RAW: like BATCH but with low-level option to specify resource requests
 - ALLOCATE: create a Slurm allocation
- Data stage-in
- Execution (pre, main, post)
- Data stage-out

`https://unicore-docs.readthedocs.io/en/latest/user-docs/rest-api/
job-description`

UNICORE Job example

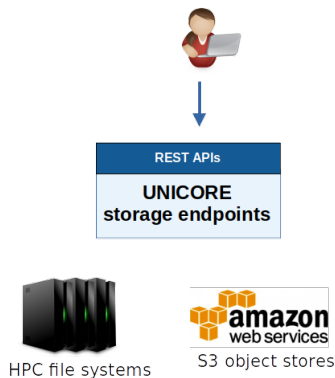
```
{  
  "Executable": "/bin/bash myscript.sh",  
  "Imports": [  
    { "To": "myscript.sh",  
      "Data": "hostname"  
    }  
  ],  
  "Resources": {  
    "Queue": "prod",  
    "Runtime": "4h",  
    "Nodes": "16"  
  }  
}
```


UNICORE's job execution workflow



Data access and file transfer

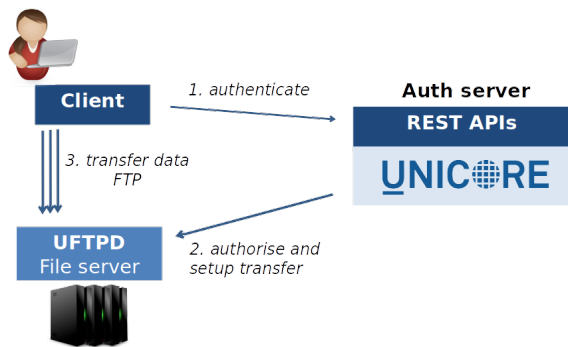
- Backends
 - HPC file systems
 - (S3 object stores)
- Comprehensive access APIs
- Server-to-server transfers
- Data stage-in/out from many sources (other UNICORE servers, https, git, ...)
- Default transfer mechanism is https



UFTP (UNICORE FTP)

High performance file transfer

- Direct, parallelizable = fast!
- Combines standard FTP compliance with UNICORE-style authentication
- Client-to-server and server-to-server data transfers
- Create "share links" for public, anonymous access to files/datasets on HPC
- Many extras, e.g. optional compression and encryption of data streams
- <https://apps.fz-juelich.de/jsc/hps/judac/uftp.html>



Commandline Client (UCC)

- Full-featured end-user client application (requires Java)
- Suitable for interactive use and automation tasks
- Use in scripts, e.g. "ucc run ..." to launch a job
- Interactive shell with powerful command completion

```
$ ucc shell
```

UCC 10.1.4, <https://www.unicore.eu>

PyUNICORE API

Features

- Offers most UNICORE features to Python programmers
- Authentication
- Jobs, workflows, data access
- FUSE driver for mounting remote file systems via UFTP
- Port forwarding / tunneling to applications on HPC
- Includes a commandline tool ('unicore') similar to UCC
- <https://pyunicore.readthedocs.io>

```
$ pip install pyunicore[crypto,fs,fuse]
```

```
import pyunicore.client as uc_client
import pyunicore.credentials as uc_credentials

base_url = "https://unicore.fz-juelich.de/JEDI/rest/core"
credential = uc_credentials.BearerToken(_get_token())
client = uc_client.Client(credential, base_url)

job_description = {}
job_description["Executable"] = "id"
job_description["Job type"] = "ON_LOGIN_NODE"

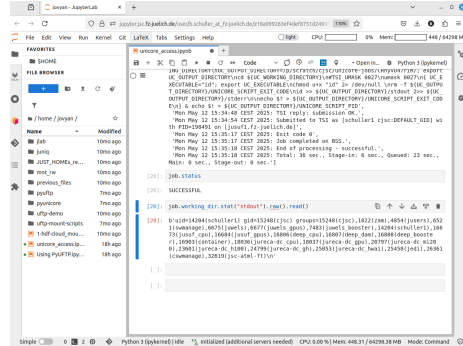
job = client.new_job(job_description)
job.poll()

stdout = job.working_dir.stat("/stdout")
print( stdout.raw().read() )
```

Example: use UNICORE in your notebooks

Hybrid Cloud / HPC applications

- Use UNICORE from within your Jupyter notebooks
- Launch HPC jobs, access files, etc



```
from IPython.display import Image, HTML
import os
import subprocess
import time

def submit_job(job_name, script_name, output_dir):
    # Submit the job to the UNICORE system
    cmd = f"unicore job-submit {job_name} {script_name} {output_dir}"
    result = subprocess.run(cmd, shell=True, capture_output=True)
    return result.stdout.decode('utf-8')

# Example usage
job_name = "test_job"
script_name = "test_script.py"
output_dir = "/tmp/output"

result = submit_job(job_name, script_name, output_dir)
print(result)
```

Job submission details:

```
Job ID: 123456789
Script: test_script.py
Output: /tmp/output
Status: Submitted to T2E via [schuler1 cjs:DEFAULT_020] via
PID=123456789 on [jussuf1.fz-juelich.de].
Main: May 12 15:35:17 EST 2025: Exit code 0.
Main: May 12 15:35:17 EST 2025: Job completed on BSS.
Main: May 12 15:35:18 EST 2025: End of processing - successful.
Main: May 12 15:35:18 EST 2025: Total: 36 sec., Stage-in: 0 sec., Queued: 23 sec.,
Main: 0 sec., Stage-out: 0 sec.

[20]: Job status
[20]: SUCCESSFUL
[20]: Job working_dir stat["stdout"] -> .read()
[20]: b'ids=14204[schuler1] gid=15248[cjs] groups=15248[cjs],1922[ram],4054[jussuf],652
1[swanago],6075[jussuf1],6077[jussuf],7483[jussuf booster],16384[schuler1],166
73[jussuf_cpu],16684[jussuf_gpu],16890[deep_cpu],16897[deep_dam],16898[deep_booste
r],16899[centralizer],16896[jureca-dc_cpu],16897[jureca-dc_gpu],20707[jureca-dc_wl20
0],25083[jureca-dc_h108],24795[jureca-dc_gpl],25053[jureca-hc_hmali],25458[jeff],26301
[cswmanage],32819[jsc-atal-f1]n'
```

Additional information and support

UNICORE

- Project web site: <https://www.unicore.eu/>
- Product documentation: <https://unicore-docs.readthedocs.io>

UNICORE at JSC

- User support email: ds-support@fz-juelich.de
- JSC specific documentation: <https://www.fz-juelich.de/en/ias/jsc/services/user-support/jsc-software-tools/unicore>



Part II: JSC Cloud

Overview

- OpenStack Infrastructure-as-a-Service (IaaS) environment
 - Compute, storage, network, orchestration, load balancing
 - Run VMs to provide services **linked to DATA**
 - Orchestration using OpenStack Heat
 - Load Balancer as a Service (LBaaS) using OpenStack Octavia
- Intended and existing workloads
 - Scientific services connected to HPC
 - Jupyter JSC
- Further information and reference:
<https://go.fzj.de/jsc-cloud>

JSC Cloud	
vCPU	7680
Memory	30.5 TB
GPU	12x NVIDIA A100 80GB 16x NVIDIA V100
NVMe	32x 1TB

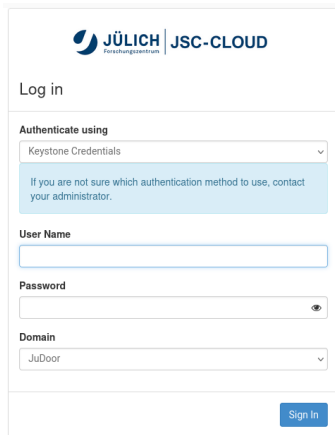


- OpenStack
- <https://docs.openstack.org/yoga/user/>
- Services
 - Keystone – authentication and service registry
 - Horizon dashboard – convenient Web UI appropriate for many simple tasks
 - Nova compute – virtual machine (VM) service
 - Neutron networking – software defined networks
 - Cinder volume – virtual block devices
 - Glance images – template images for VMs
 - Heat orchestration – infrastructure management
 - Octavia load balancing – load balancing as a service
 - Neutron VPNaaS – cross-site (or project) VPNs
 - Sahara – data processing through virtual clusters

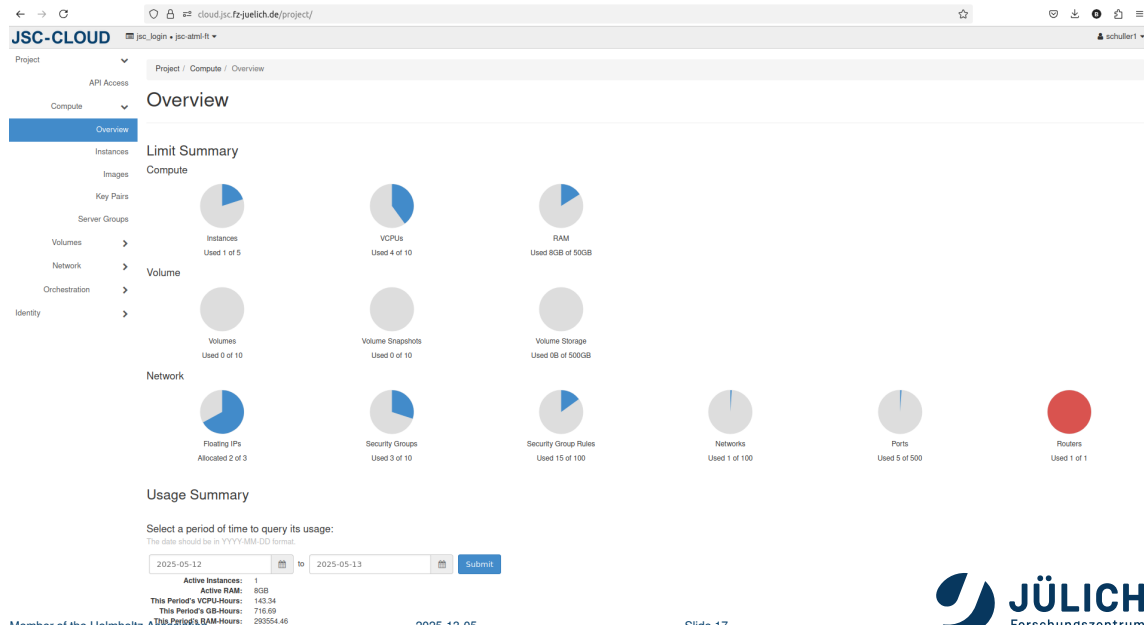
Authentication

There are two ways to authenticate

- JSC account
 - username and password
 - usable from both commandline interface and Web UI
- Helmholtz login
 - directly usable only from Web UI
 - commandline access through application credentials
- **However:** you need a project and allocated resources before using JSC Cloud



The screenshot shows the login page for JÜLICH JSC-CLOUD. At the top, the JÜLICH Forschungszentrum logo is next to the text 'JSC-CLOUD'. Below this is a 'Log in' heading. A dropdown menu labeled 'Authenticate using' currently shows 'Keystone Credentials'. A light blue informational box below the dropdown states: 'If you are not sure which authentication method to use, contact your administrator.' Below this are three input fields: 'User Name' (a text box), 'Password' (a text box with a toggle eye icon), and 'Domain' (a dropdown menu showing 'JuDoor'). A blue 'Sign in' button is located at the bottom right of the form.



OpenStack Nova

Virtual machine service

Nova manages the lifecycle of virtual machines (VMs) that have

- a number of CPUs
- some amount of main memory
- storage: **system**, ephemeral, swap
- data storage: volumes
- network ports
- a template image containing an operating system

Nova

VM Flavors

Flavors define the "hardware characteristics" of the virtual machine. We use Sovereign Cloud Stack (SCS) flavors of the form

Example

SCS-16L:32:20n or SCS-nL:m:l-features

with

- n → number of CPUs (L for low performance or high oversubscription)
- m → amount of main memory (RAM) in GB
- l → size of root disk in GB (n for network shared storage)

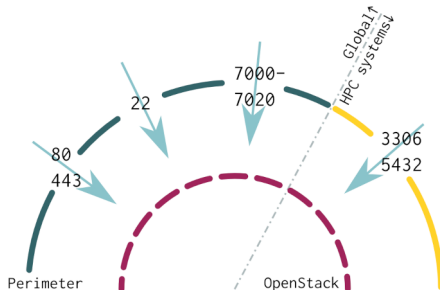
Use `openstack flavor list` or the Web UI to see available flavors. Some examples:

- SCS-2L:4:20n
- SCS-16L:64:20n-z2-nvme
- SCS-1L:1:20n

Network setup

Security groups and perimeter firewall

- OpenStack firewall freely configurable
- Restrictions apply for inbound connections in perimeter firewall
 - Globally available: HTTP (80), HTTPS (443), SSH (22), Kubernetes API (6443), 7000–7020, 8008, 8080, 8443
 - Available from HPC systems: MySQL (3306), PostgreSQL (5432)
- Outbound connections: anything but MTA (25) aka. SMTP



Data access

VMs and the DATA file system

- JSC Cloud / OpenStack cluster
 - Hosts virtual machines (VMs) for communities
 - Potentially administered by externals, bound by acceptable use policy
- Enable access to data beyond perimeter of SC facility
 - Web interfaces, databases, post processing, ...
 - Users of service likely unknown to SC directory information service
- Access Method
 - POSIX file systems (\$DATA) available ON REQUEST in VMs via NFS mount from CES servers
 - Server side UID squashing
 - ensures consistency
 - requires services to manage data accordingly
 - read-write or read-only

/p/largedata/slai
/p/largedata/slbld
/p/largedata/slschem
/p/largedata/slcm
/p/largedata/slfire
/p/largedata/slfse
/p/largedata/slkit
/p/largedata/slmet
/p/largedata/slnpp
/p/largedata/slns
/p/largedata/slpp
/p/largedata/slqip
/p/largedata/slts

a/slfse/...
ta/slkit/...
ta/slmet/xyz
/slnpp/...
tas/

Summary

JSC Cloud

OpenStack

- Project web site: <https://www.openstack.org/>
- Documentation: <https://docs.openstack.org/>

JSC Cloud:

- User support: sc@fz-juelich.de
- Web dashboard: <https://cloud.jsc.fz-juelich.de/>
- Documentation: <https://go.fzj.de/jsc-cloud>