Co-funded by
the European Union

Human Brain Project

Tools and services offered by HPAC Platform and Fenix with hands-on

2nd HPAC Platform Training, 26-28 Nov 2019

Colin McMurtrie (CSCS)

Alex Upton (CSCS)

Agenda

- 09:30 – 10:00 Overview of Tools and Services
- 10:00 – 10:30 Pollux OpenStack VM Tutorial
- 10:30 – 11:00 Coffee break
- 11:00 – 12:00 Piz Daint Scalable Compute Tutorial including Sarus

- Tutorial sheet and slides can be downloaded from: bit.ly/HPAC_tutorial
- Slides can be downloaded from: bit.ly/HPAC_slides

Before we begin…

- Who is in the room?
  - Masters students/PhD students/Postdocs/Professors/Other?
- How many people have a HBP collab/wiki account?
  - If not, why?
- How many people have an ICEI account?
  - If not, why?
- How many people in the room find their research is limited by lack of computational resources?
  - e.g. not enough computing power/unable to share workflows/unsure on how to share data etc.

HPAC/Fenix/ICEI – what are they?

- The High Performance Analytics and Computing (HPAC) Platform develops and provides supercomputing, storage, visualisation and simulation technology that can run on supercomputers. This allows scientists to:
  - Run large-scale, data intensive, interactive multi-scale brain simulations up to the size of a full human brain
  - Manage the large amounts of data used and produced by simulations and in experiments
  - Manage complex workflows comprising concurrent simulation, data analysis and visualisation workloads
- The ICEI project delivers a set of e-infrastructure services that will be federated across five European supercomputing centres that include BSC (Spain), CEA (France), CINECA (Italy), CSCS (Switzerland) and JSC (Germany), to form the Fenix Infrastructure.

What Services does Fenix/ICEI/HPAC provide?

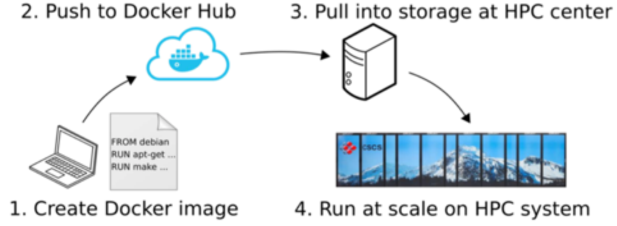
- **End-user Services**
  - Scalable Compute Services (both hybrid CPU+GPU nodes and multicore CPU-only nodes)
  - Interactive Compute Services (including hybrid nodes)
  - SWIFT Object Storage
  - Data Storage Services
  - (Data Transfer Service) ← HPAC
  - (Continuous Integration Services) ← HPAC
  - (Software Packaging and Deployment Services) ← HPAC
  - (Visualisation Services) ← HPAC
- **Platform Services**
  - Infrastructure Services (middleware access to HPC resources via Rest APIs)
  - Infrastructure as a Service (e.g. OpenStack) for Virtual Machine Services
  - Data Management Services
  - User and Resource Management Services
  - Service Accounts (currently not available at all sites)

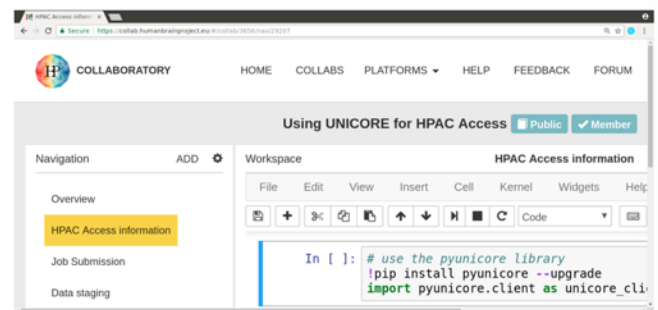
Co-funded by the European Union

Fenix/ICEI provides the Base Infrastructure for HPAC

- Support of user workflows
- Job Submission
- Data access
- Site-to-site Data Transfer
- Single sign-on to HPC Allocations
- Launch simulations and data analysis tasks from Jupyter notebooks
- Software deployment via Docker Containers

HPAC Infrastructure





Platform Services (PaaS)	• Data Services Portal • HPC Portal • Supported Scientific Libraries • Externally supported portals	HBP Platforms / Collaboratory
		REST APIs
Infrastructure Services (IaaS)	• Virtualization • Containers • Web interfaces • Custom middleware	UNICORE — Infrastructure Services — openstack CLOUD SOFTWARE
IT infrastructure	• Computing • Storage • Networking • Security	

FENIX RESEARCH INFRASTRUCTURE — JÜLICH Forschungszentrum — BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación — cea — CINECA — CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre — ETH zürich

ICEI Resources for HBP

- ICEI resources have already been made available to HBP and PRACE by CSCS
- There are currently 21 HBP projects with compute allocations at CSCS
  - More are in the approval stages
- More resources are available than are being consumed so HBP users are encouraged to apply for a compute allocation
  - More on this in the next session and tomorrow

					Quarterly Distribution		
Component	ICEI Service Type	ICEI Total Allocation (Raw Resource)	Allocatable Unit	ICEI (100%)	HBP (25%)	PRACE (15%)	National (60%)
OpenStack Cluster	VM	35 servers	Servers	35	8.75	5.25	21.00
Piz Daint Multicore	SCC	250 nodes	Node-Hours	465375	116,344	69,806	279,225
Piz Daint Hybrid	SCC + IAC	400 nodes	Node-Hours	744600	186,150	111,690	446,760
Store POSIX and Object	ARD	1000 TB	TB	1000	250	150	600
Tape library	ARD	3000 TB	TB	3000	750	450	1,800
Low latency storage tier*	NVM	80 TB	TB	80	20	12	48

* Early access technology. User workflows need to be adapted/augmented.

Resources currently available

Component	Site	Total ICEI	Minimum Request	Technical Details
Scalable Computing Services				
Piz Daint Multicore	CSCS (CH)	250 nodes	1 node	• Memory per node 64/128 GB • Compute nodes/processors: 1813 Cray XC40 nodes with Two Intel® Xeon® E5-2695 v4 @ 2.10GHz (2 x 18 cores) CPUs • Interconnect configuration: Cray Aries
Interactive Computing Services				
Piz Daint Hybrid	CSCS (CH)	400 nodes	1 node	• Memory per node: 64 GB • GPU memory: 16 GB CoWoS HBM2 • Compute nodes/processors: 5704 Cray XC50 nodes with Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores) CPUs and NVIDIA® Tesla® P100 GPUs • Interconnect configuration: Cray Aries
VM Services				

Pollux OpenStack Cluster	CSCS (CH) *	35 servers	1 VM	• 2 types of compute node:
				  • Type 1 - CPU: 2x Intel E5-2660 v4 14C/RAM: 512 GB
				  • Type 2 - CPU: 2x Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz 8C/RAM: 768 GB
				• VMs can be of various flavours and use up to 16 cores

Archival Data Repositories				
Store POSIX and Object, including backup on Tape library (2x)	CSCS (CH)	4000 TB	1 TB	
Active Data Repositories				
Low latency storage tier (DataWarp)	CSCS (CH)	80 TB	1 TB	• Non-volatile Memory

*Julich OpenStack Cluster currently in development

How do I use ICEI Resources? (1)

- Firstly, you will need to have obtained an account via an ICEI request application
  - More info with request form can be found here: https://collab.humanbrainproject.eu/#/collab/28520/nav/203167
  - Application form shown on next slide, lightweight document with only 5 short sections
  - Detailed overview about applying for resources in session tomorrow afternoon, including hands-on walkthrough of the application process

Resource Application Form

FENIX
RESEARCH INFRASTRUCTURE

Request for HBP Resources in ICEI

Project duration[1] (YYYY/MM-YYYY/MM)	
Project name	
Type of project (new or extension)	
Project ID (in case of extension)	
PI name (please name only one)	
PI Organisation	
PI Email	
Names, organisation and Email of other involved persons	
Date	

Note: The resource request form will be shared within the HBP Consortium and information on resource requests received will be included in ICEI deliverables with dissemination level "Confidential, only for members of the consortium (including the Commission Services)".

Summary

Please provide one paragraph summarizing the scientific question(s) that you intend to address using these resources. What is the scientific goal?

Contents

[1] Start of the project may be adjusted by the Infrastructure Allocation Committee (IAC)

1. Relation to HBP DoA and relevance to HBP call

Please provide information on the related work packages, tasks, CDPs, etc. and explain how the project relates to the goals and objectives of HBP.
How does the project relate to the published HBP call for resources in ICEI?

2. Preliminary Work (in case of a project extension)

Please provide a brief summary of project results obtained from your first resource allocation.

3. IT resources requested

3.1 Resources

Resource	Units	Quantity (required in total)
Piz Daint Multicore	node×hour	
Piz Daint Hybrid	node×hour	
OpenStack Cluster	servers	
Store POSIX and Object	TByte	
Tape library	TByte	
Low latency storage tier	TByte×day	

3.2 Technical implementation plans

Please explain why the requested resources are needed to achieve the scientific goal.
What kind of jobs are planned (number and type of nodes, typical job duration)? How much storage needs to be available to execute the jobs? Which software, HBP platform tools and services are needed?

3.3 Does this project involve processing of personal data as defined by GDPR?

Please select "Yes" or "No", if you selected "Yes", please specify what kind of data is processed.

☐ NO
☐ YES

4. Scientific methodology, goals and impact

4.1 Scientific implementation plans

Please explain the methodology that will be used to achieve the scientific goal of the project, highlighting scientific excellence, novelty and potential for high European and international impact of the project.
What are possible transformative aspects and expected advances?

4.2 Resource management and work plan

Please describe how you intend to manage the requested resources.

4.3 Dissemination

Please describe planned channels and resources for dissemination and knowledge exchange. If the requested resources are used to provide EBRAINS services then describe plans for attracting users for these services.

5. References

Please provide recent/most important bibliographic references that are relevant to the project.

[<ref number>] <reference>

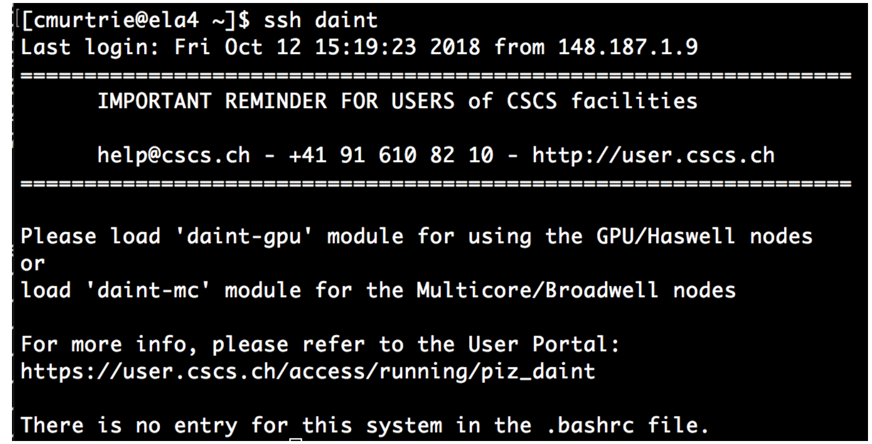
Co-funded by the European Union

How do I use ICEI Resources? (2)

Scalable Compute Resources:

The *Piz Daint* system is available as a state-of-the-art scalable compute resource for use by HBP users

- Accessible globally via Command-line Interface (more about this later)
- Via the UNICORE GUI
- Via the RESTful API offered via UNICORE for platforms
  - Use of Service Accounts for Platforms is also acceptable at some sites (e.g. CSCS)
  - See next slide for some more details



How do I use ICEI Resources? (3)

<u>Interactive Compute Resources:</u>

The *Piz Daint* system supports the use of Jupyter Notebooks for interactive supercomputing, powered by JupyterHub

- This is a multi-user Hub that spawns, manages and proxies multiple instances of the single-user Jupyter notebook server
  - More details below
- Subsequent sessions will demonstrate the use of this environment

Piz Daint and other HPAC HPC systems are also accessible from the Jupyter Notebooks service of the *Collaboratory*

- This relies on the RESTful API offered via UNICORE for platforms
- The sessions tomorrow will go into the details of how to do this

How do I use ICEI Resources? (4)

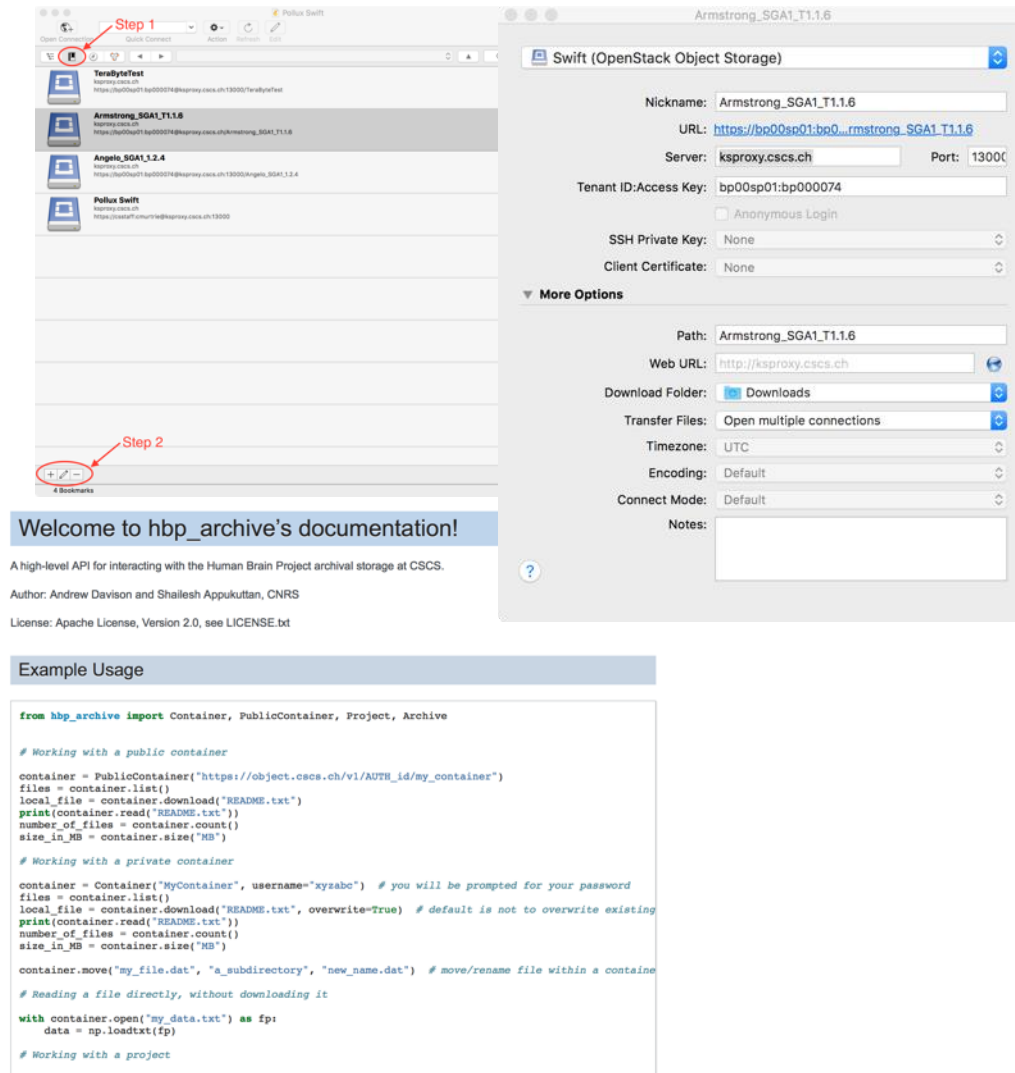
<u>Archival Data Storage:</u>

SWIFT OS can be accessed directly from your personal computer

- GUI clients e.g. CyberDuck
- SP5 Python Library
  - Better for mgmt. of access control lists (ACLs) and Object Buckets
  - https://hbp-archive.readthedocs.io/en/latest/
  - More on this later

Reachable from inside the *Collaboratory*

- Get/Put from Jupyter Notebooks
- More capabilities coming soon



How do I use ICEI Resources? (5)

Active Data Repositories:

- Come as part of the compute allocation (= $SCRATCH)
- Low-latency storage tier (Cray DataWarp with SSDs) in *Piz Daint* can also be requested

Archival Data Repositories:

- Are available either as part of a computing request (your proposal should state how much you need)
- Or separately in a data-storage only use case (in which case a separate proposal is needed)

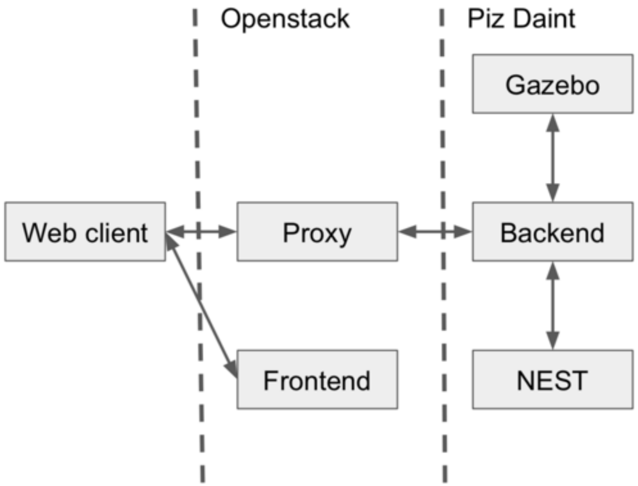
Co-funded by the European Union

How do I use ICEI Resources? (6)

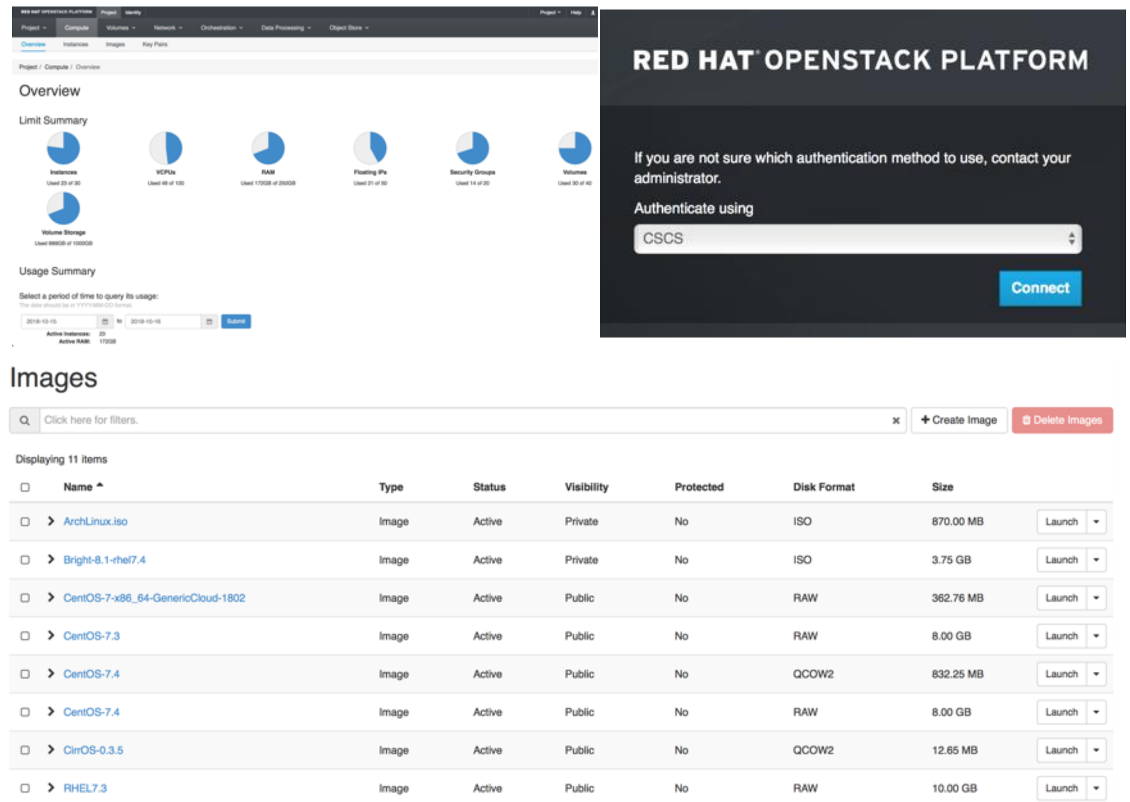
Virtual Machine Resources:

The *Pollux* OpenStack IaaS is available as a VM resource to host your platform VMs:

- Accessible globally via the Horizon GUI interface
- RESTful API can be used for automation



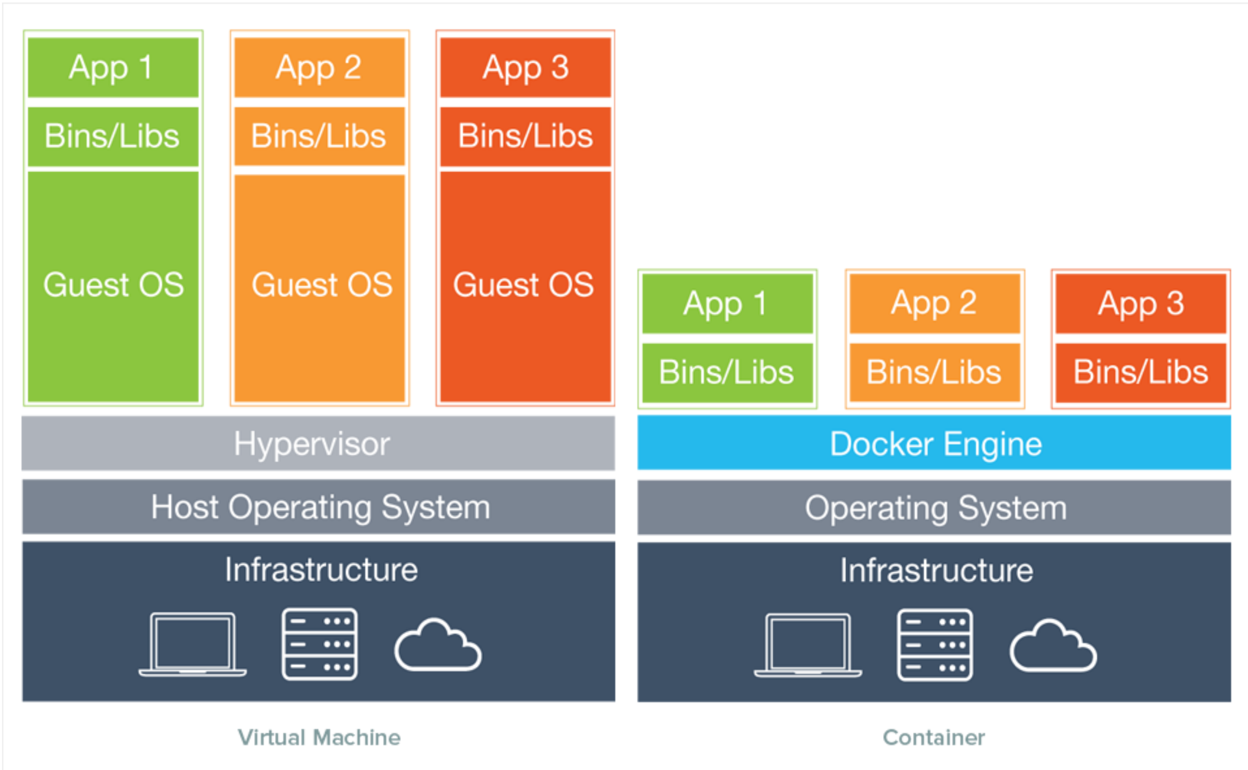
Example of a Platform service (NRP) using VMs AND HPC resources.



Service Detail: Software Packaging and Deployment

Containers

- Isolated environments to run applications/services
- Images include all software dependencies
- Prescriptive, portable, easy to build, quick to deploy



Container implementations

OpenVZ Linux Containers

docker

LXC

rkt

ORACLE SOLARIS

HPC focused

Singularity

SARUS

Charliecloud

SHIFTER

Docker

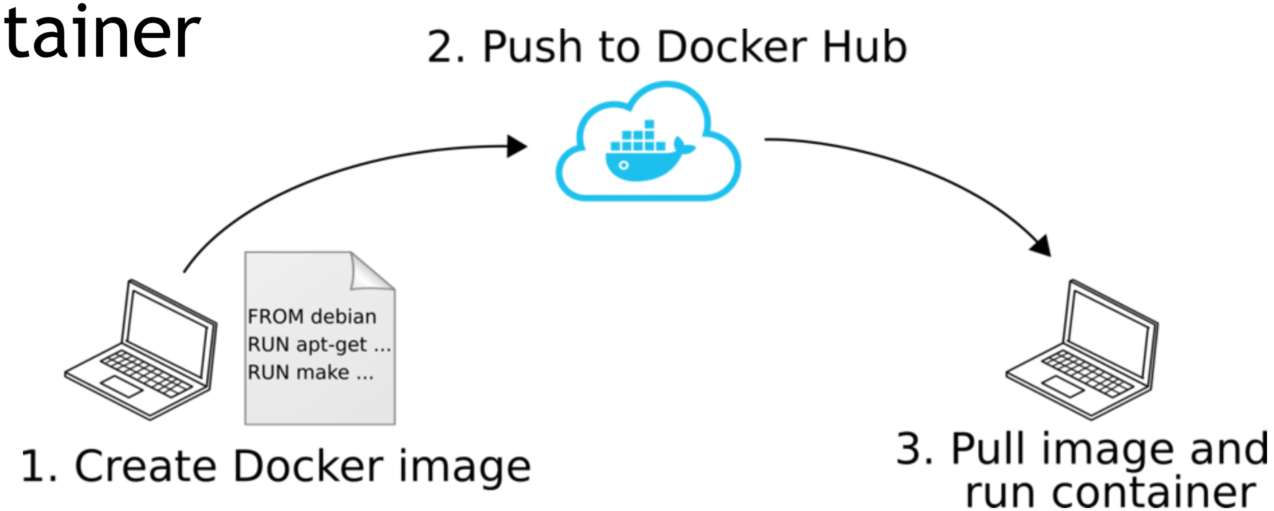
- Extremely popular container implementation

- Easy to use authoring tools
  - Container images are created from recipe-like files
  - Images can be named, tagged and built on top of other images

- Cloud-based image distribution strategy
  - Several remote registries available (e.g. Docker Hub)
  - Client includes facilities to authenticate, push and pull images

Docker workflow

1. An image is created locally from a *Dockerfile*
2. Push (i.e. upload) the image to a remote registry
   - DockerHub is the public registry maintained from the Docker company
3. Pull (i.e. download) the image on a target machine and run the container



Key terms

- **Image**: standalone, executable package that includes everything needed to run a piece of software
  - code, runtime libraries, environment variables, configuration files
- **Container**: runtime *instance* of an image
  - What the image becomes in memory when actually executed
  - Runs completely isolated from the host environment by default
    - only accessing host resources if configured to do so

So… how are containers useful?

Containers give the possibility to create (scientific) applications that are:

1. Portable
2. Reproducible
3. Easy to deploy
4. Easy to test

Unfortunately Docker containers are not a panacea for HPC environments because of:

- Security concerns
  - root in the container means root on shared parallel file systems
- Performance Portability
  - Performance is important in HPC (it's in the name…)

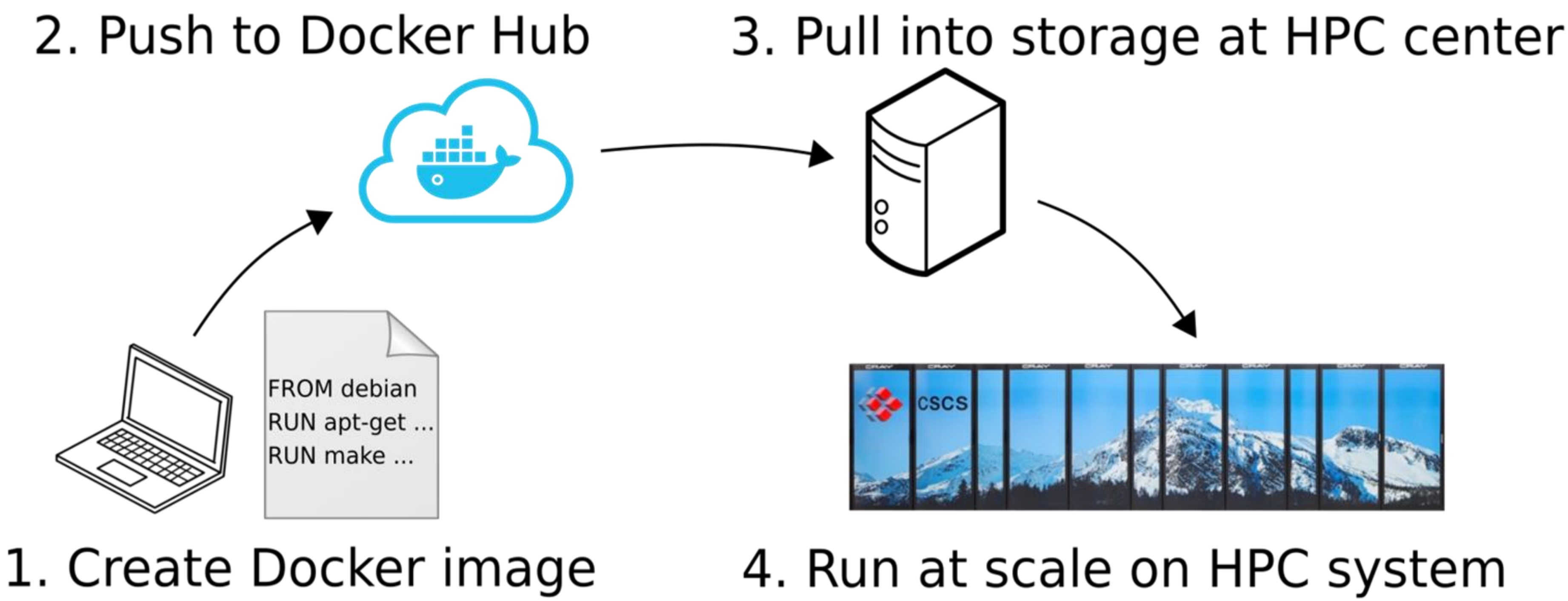
Co-funded by the European Union

Sarus container engine

- OCI-compliant container engine engineered by CSCS
- Designed for the requirements of HPC
- Consistent UX with widely-used Docker: small learning curve
- Transparent native performance through hooks e.g. MPI, GPU
- Enables use of standard, open, upstream components on HPC systems
- Extensible architecture encourages vendor engagement and improves maintainability

Co-funded by the European Union

User workflow using scalable compute



Sarus from the user perspective

- Consistent experience
  - With Docker: closely resembles CLI (see next slide)
  - With host environment: env variables, uid/gid, file permissions, working directory
- Pull images from Docker registries (e.g. Docker Hub, NVIDIA NGC)
- Import images from local tar archives (no cloud upload required in case someone does not want to put image on Docker Hub)
- Integration with the workload manager (Slurm)
- Native performance from GPUs and Cray high-speed interconnect
- Access to parallel filesystems inside containers

Sarus CLI

- Sarus

```
$ sarus pull [options] <image>[<:tag>]

$ sarus load [options] <file> <image>

$ sarus images

$ sarus rmi <image>[<:tag>]

$ sarus run [options] <image>[<:tag>] <command> <args>
```

- Docker

```
$ docker pull [options] <image>[<:tag>]

$ docker load [options] -i <file>

$ docker images [options] [repo[<:tag>]]

$ docker rmi [options] <image> [image…]

$ docker run [options] <image>[<:tag>] <command> <args>
```

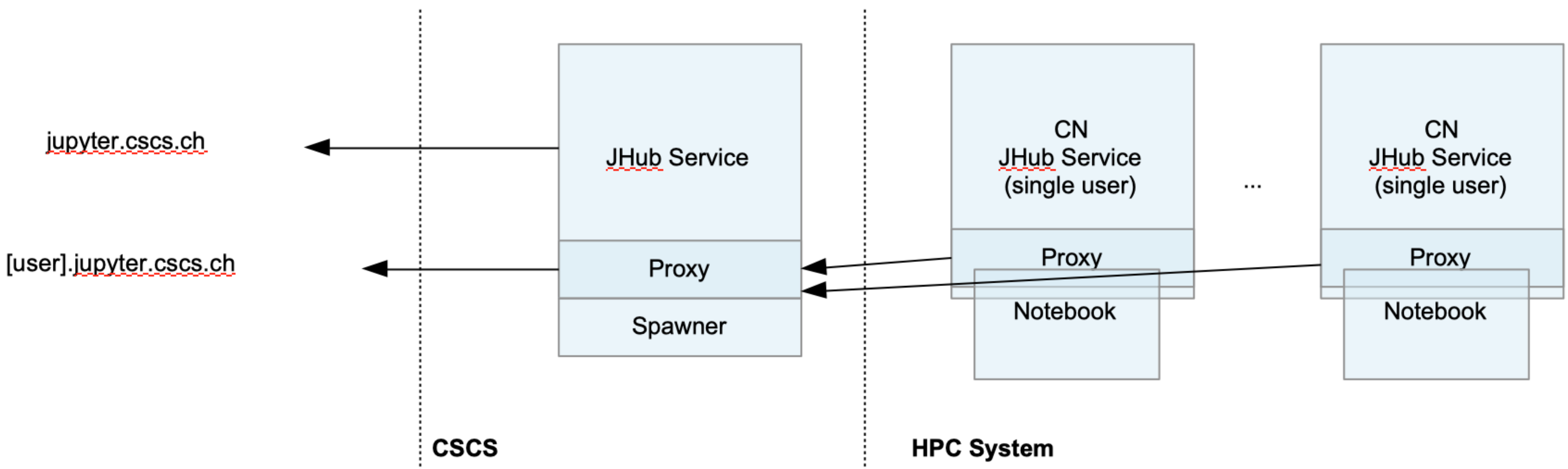
Service Detail: JupyterHub Service at CSCS

Using JupyterHub at CSCS

- This service enables the interactive execution of Jupyter Notebook on *Piz Daint* over both single and multiple nodes.
  - The supported python version is python3.
- The service is accessed through the address
  - https://jupyter.cscs.ch
  - users should provide their HPAC credentials in order to login
- Once logged in, the user is redirected to a job setup page
  - Allows typical job configuration options to be selected in order to allocate the resources that are going to be used to run Jupyter
    - account
    - type of *Piz Daint* node type (gpu or mc)
    - number of nodes
    - wall-clock time limit
- More information at: https://user.cscs.ch/tools/interactive/

JupyterHub Service Architecture (1)

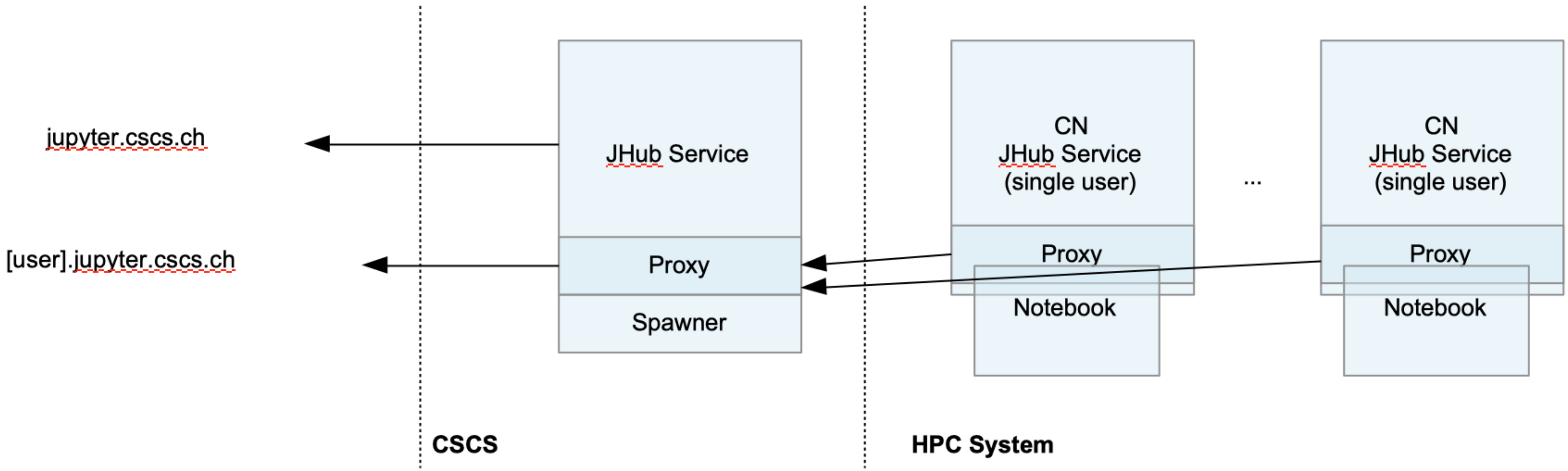
- The **current** architecture protects the notebook in each compute node (CN) by launching a JupyterHub Service along with it



JupyterHub Service Architecture (2)

Notebooks v4.3 and newer are protected with a per-session tokens

- Avoids the creation of several custom spawners
  - Ideally we want one CSCS spawner only
- Will be integrated with an Infrastructure Services API (UNICORE or similar)
- The frontend will be kept outside of the HPC system



VM 101

1. Login via Horizon Web GUI
2. Create small network
3. Create router
4. Create key pair
5. Launch instance
6. Assign floating IP
7. Ping instance to verify connection and login
8. Sharing data between workshop participants using ACLs (if time allows)

Cloud Computing - OpenStack

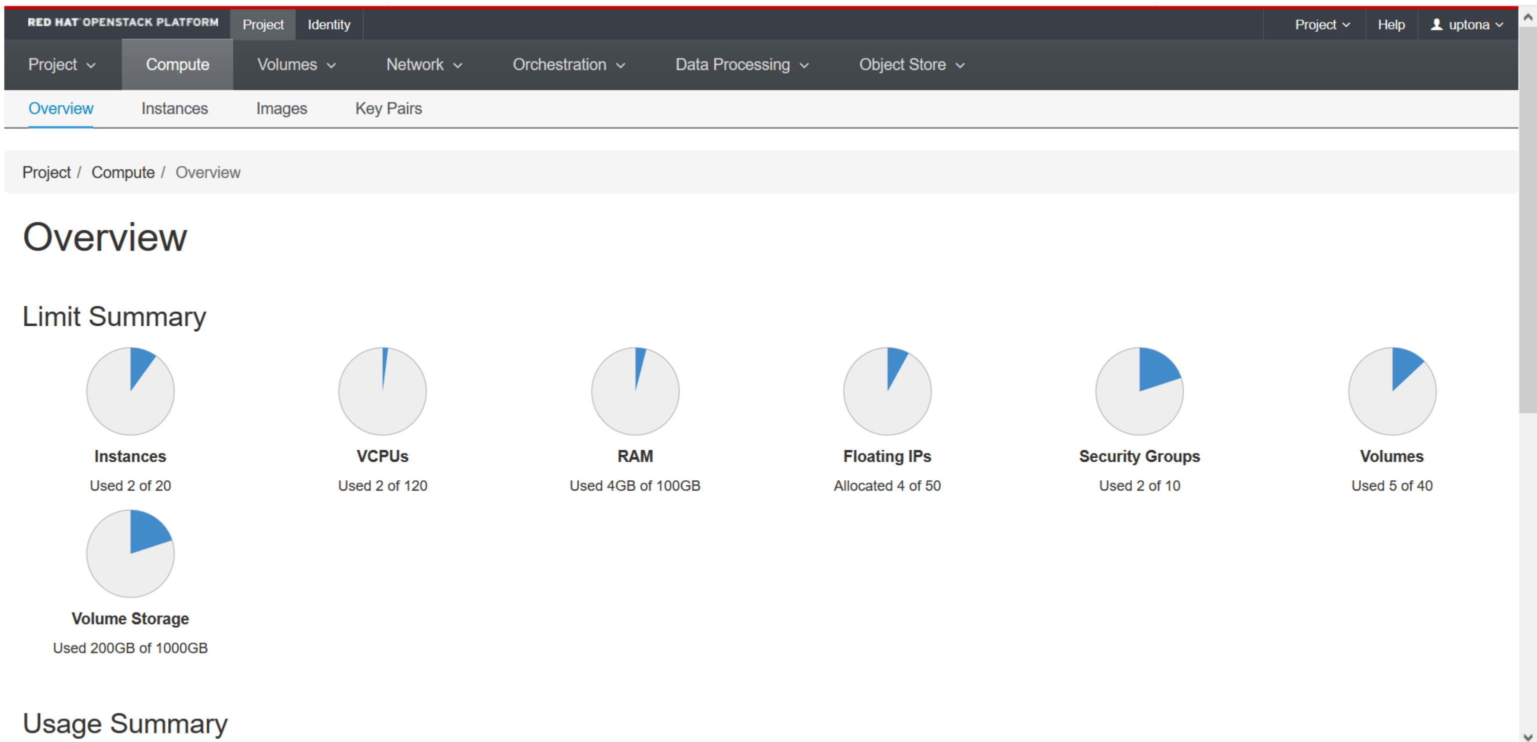
"OpenStack is a free and **open-source software platform** for **cloud computing**, mostly deployed as infrastructure-as-a-service (**IaaS**), whereby **virtual servers and other resources** are made available to customers."

OpenStack @CSCS - Pollux

- **Pollux** is CSCS's general purpose OpenStack system
  - since summer 2017
- **~433 VMs**
- **~338 users**
- VM uptime In 2018:
  - **99.93%** unplanned
  - **99.76%** unplanned and planned
- Currently at version Queens
  - RedHat OpenStack Platform 13

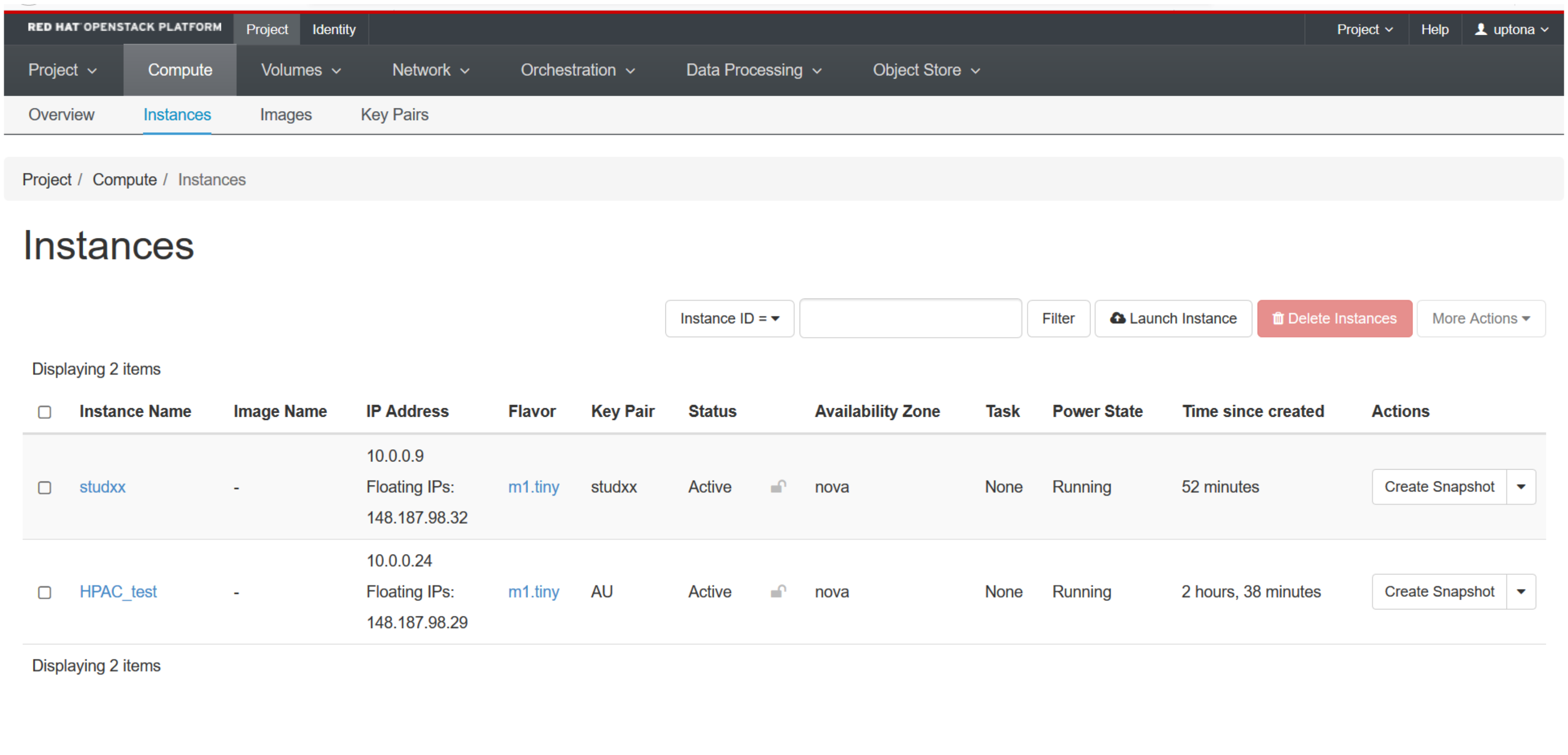
Horizon OpenStack GUI

- You can **create VMs** and execute other actions using the OpenStack **graphical interface Horizon**, can be reached at the URL: https://pollux.cscs.ch



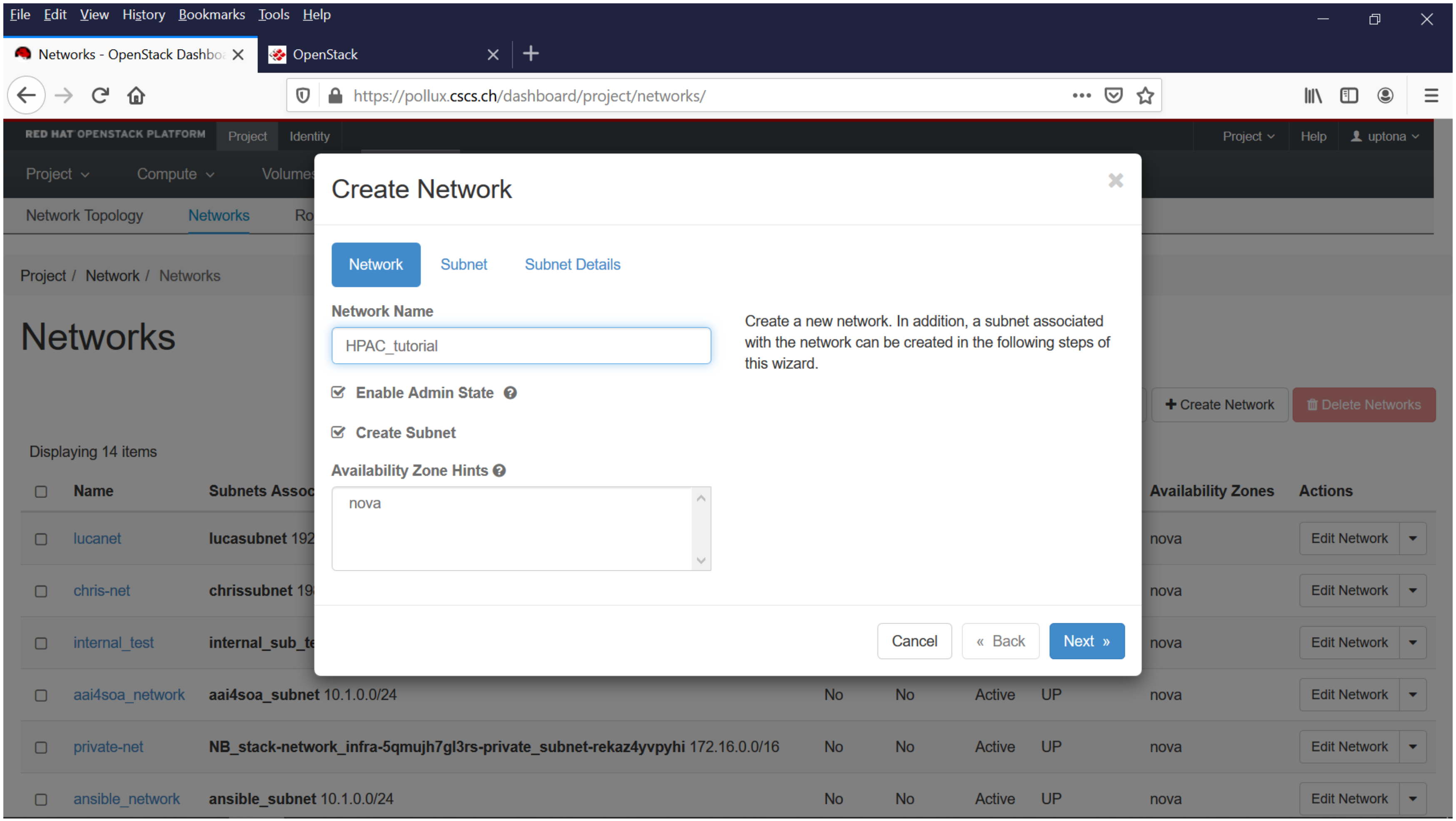
View Instances in the Project

- Can see all current instances in the project



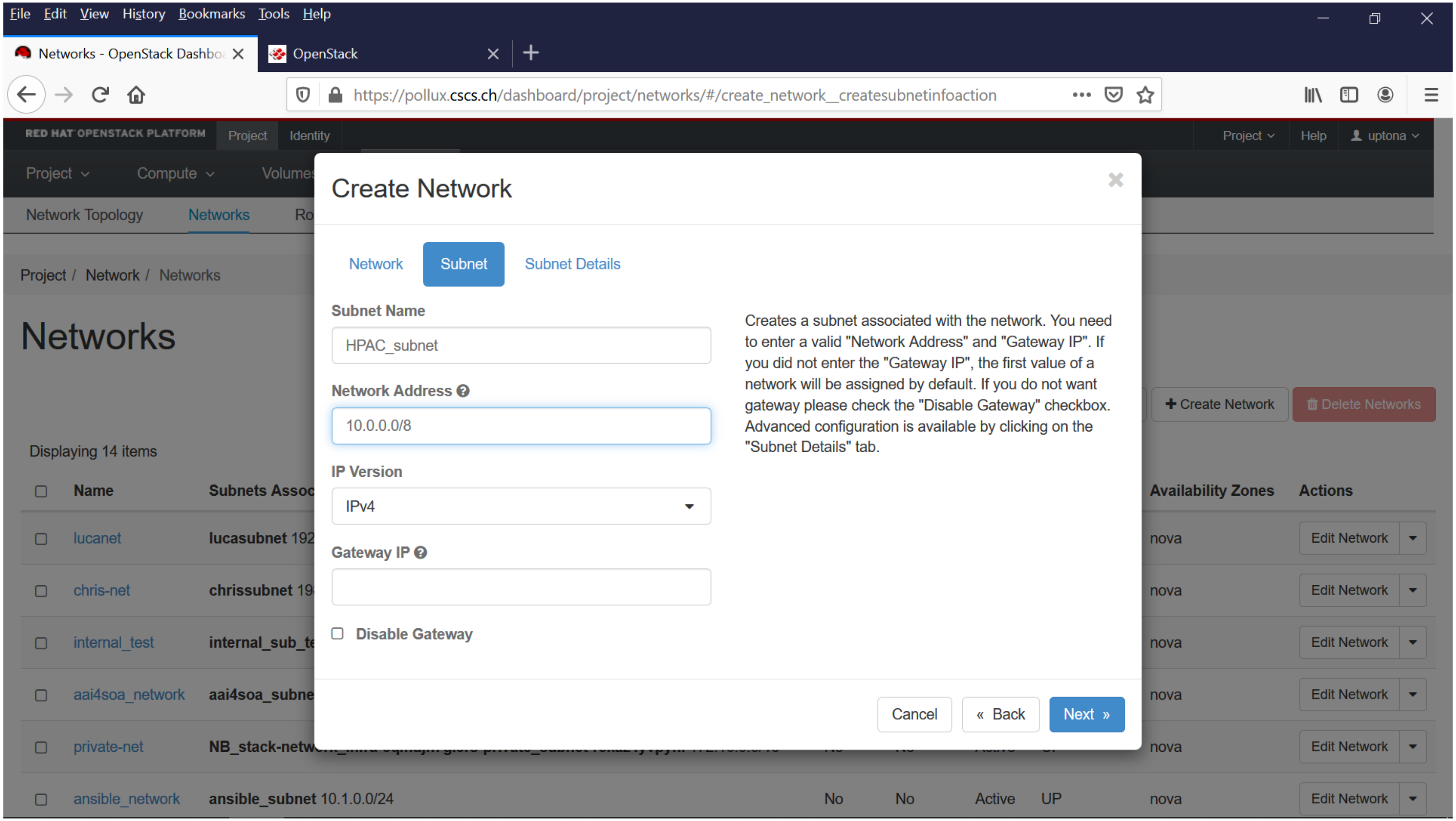
Create network

- First we want to create a network to connect to



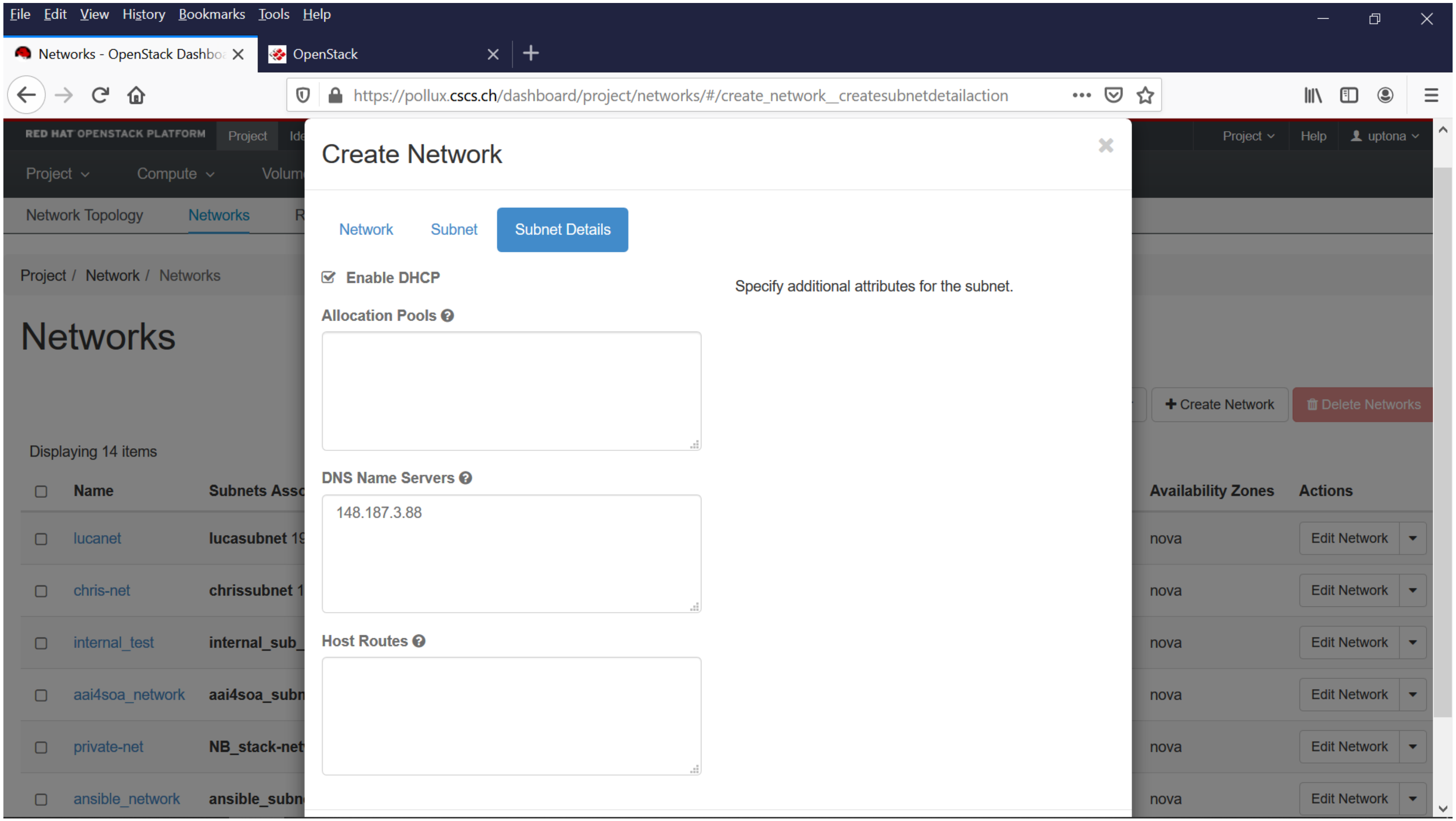
Create subnet

- Next create subnet



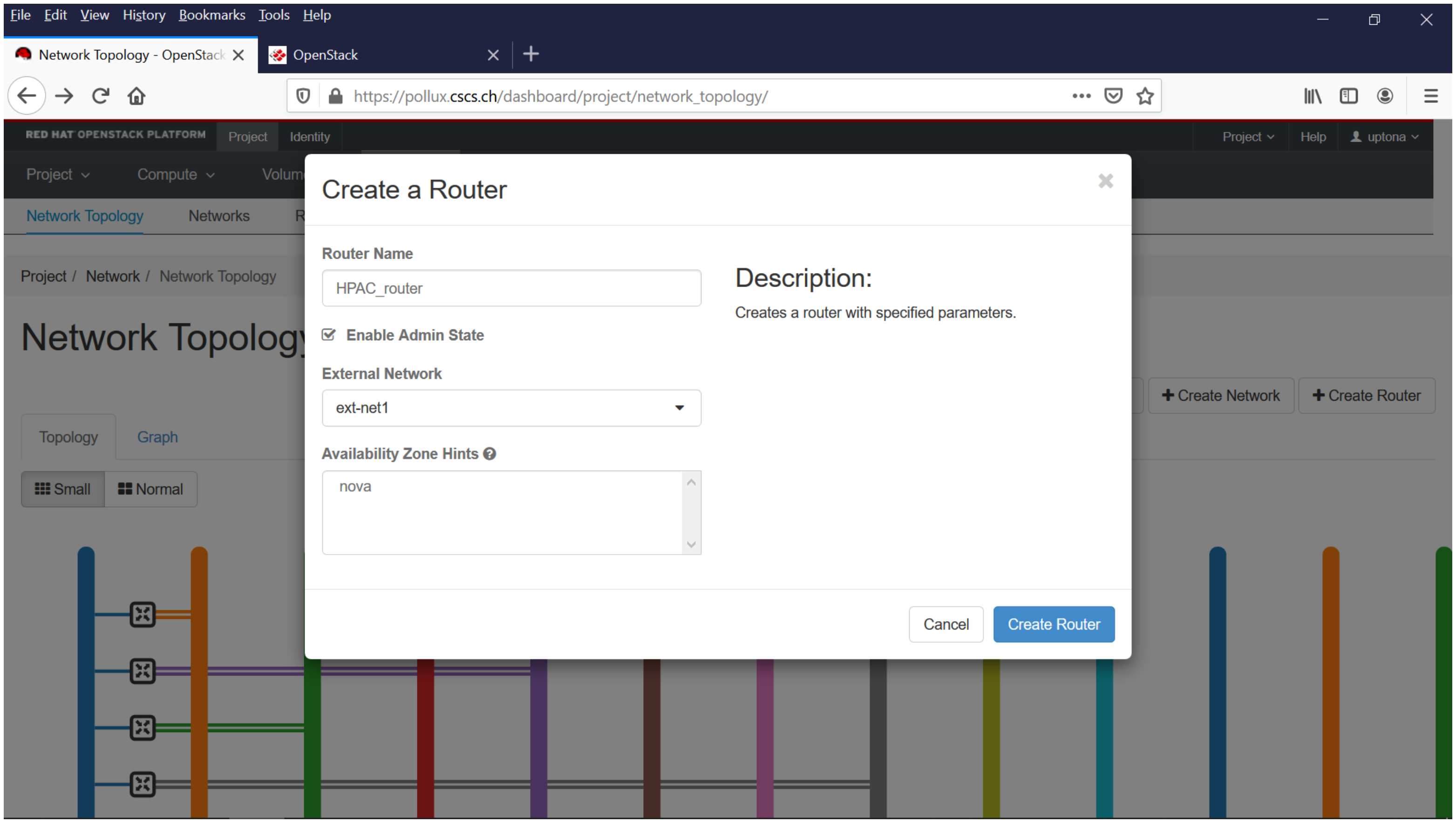
Subnet details

- Finally define the subnet details

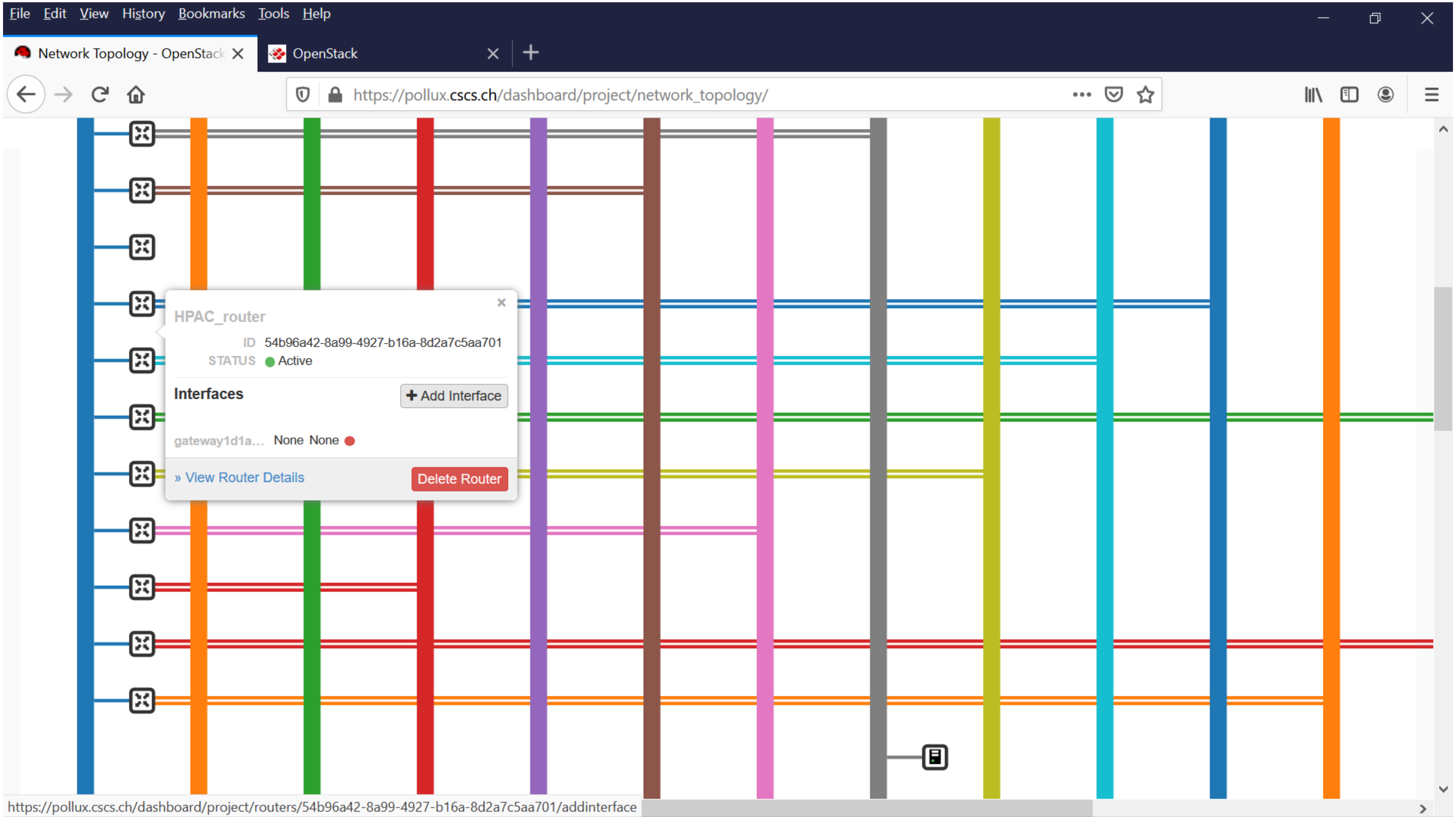


Create router

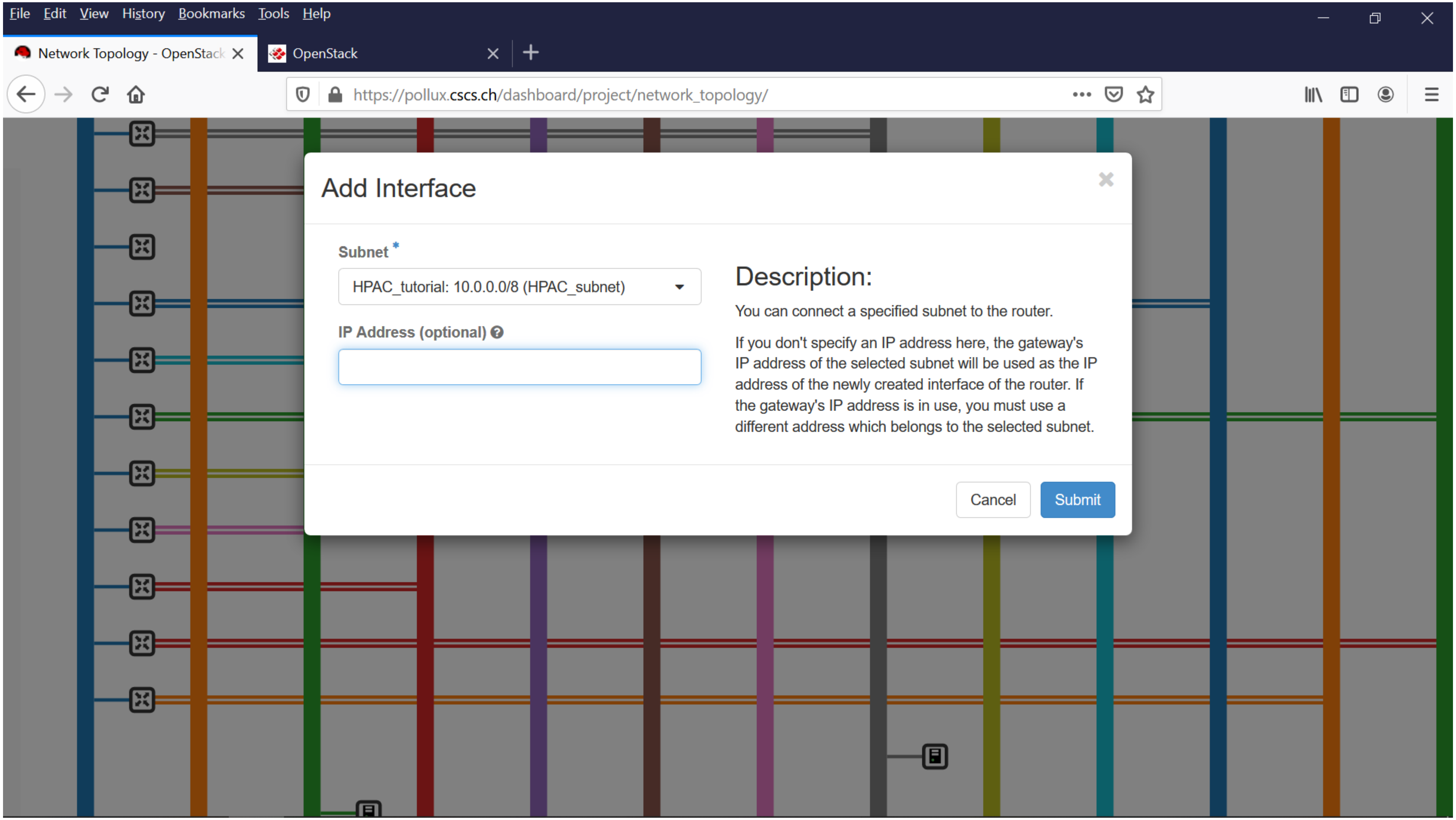
- Previously we created a private isolated network so now we want to connect it to the internet via a router



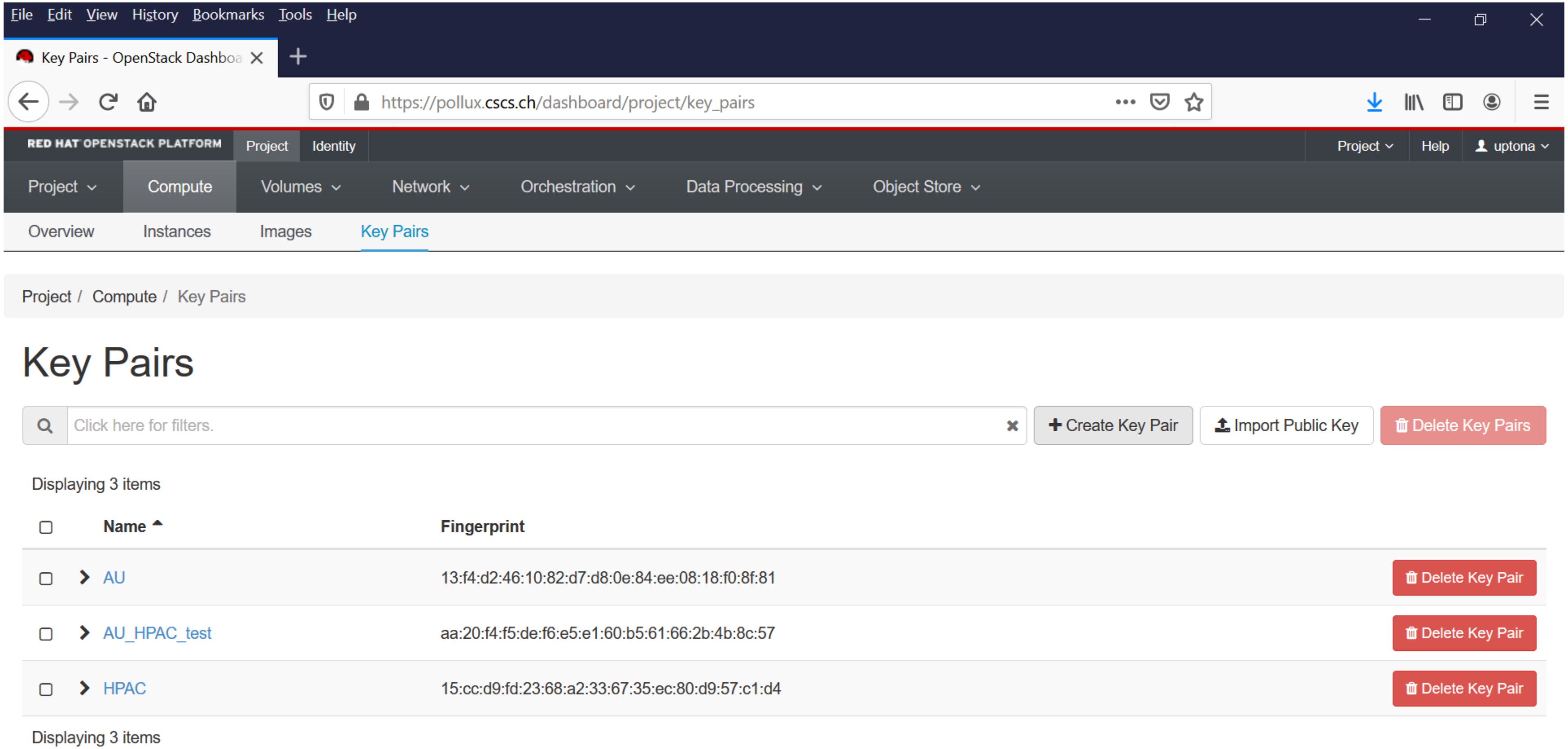
Create router



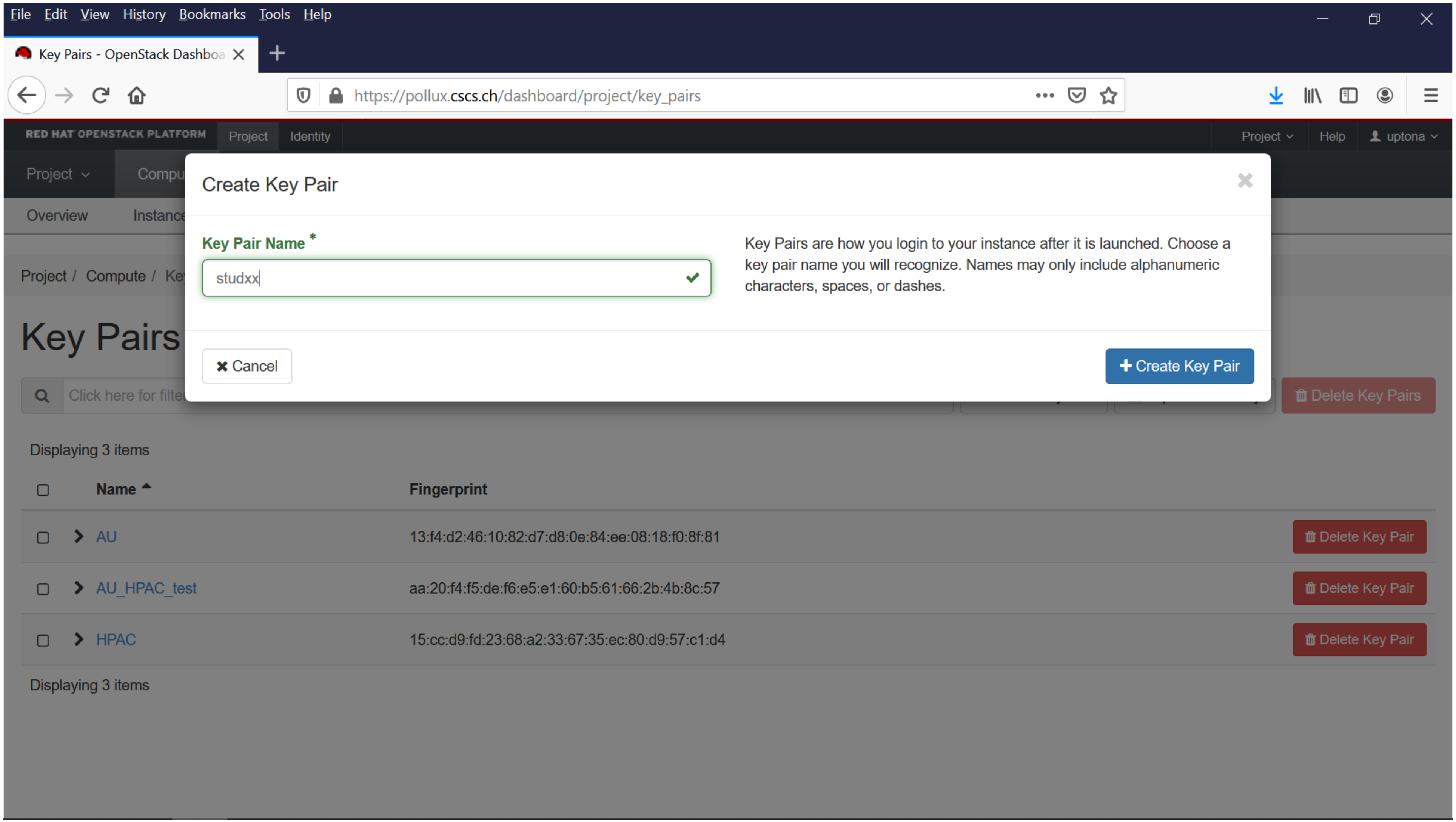
Create router



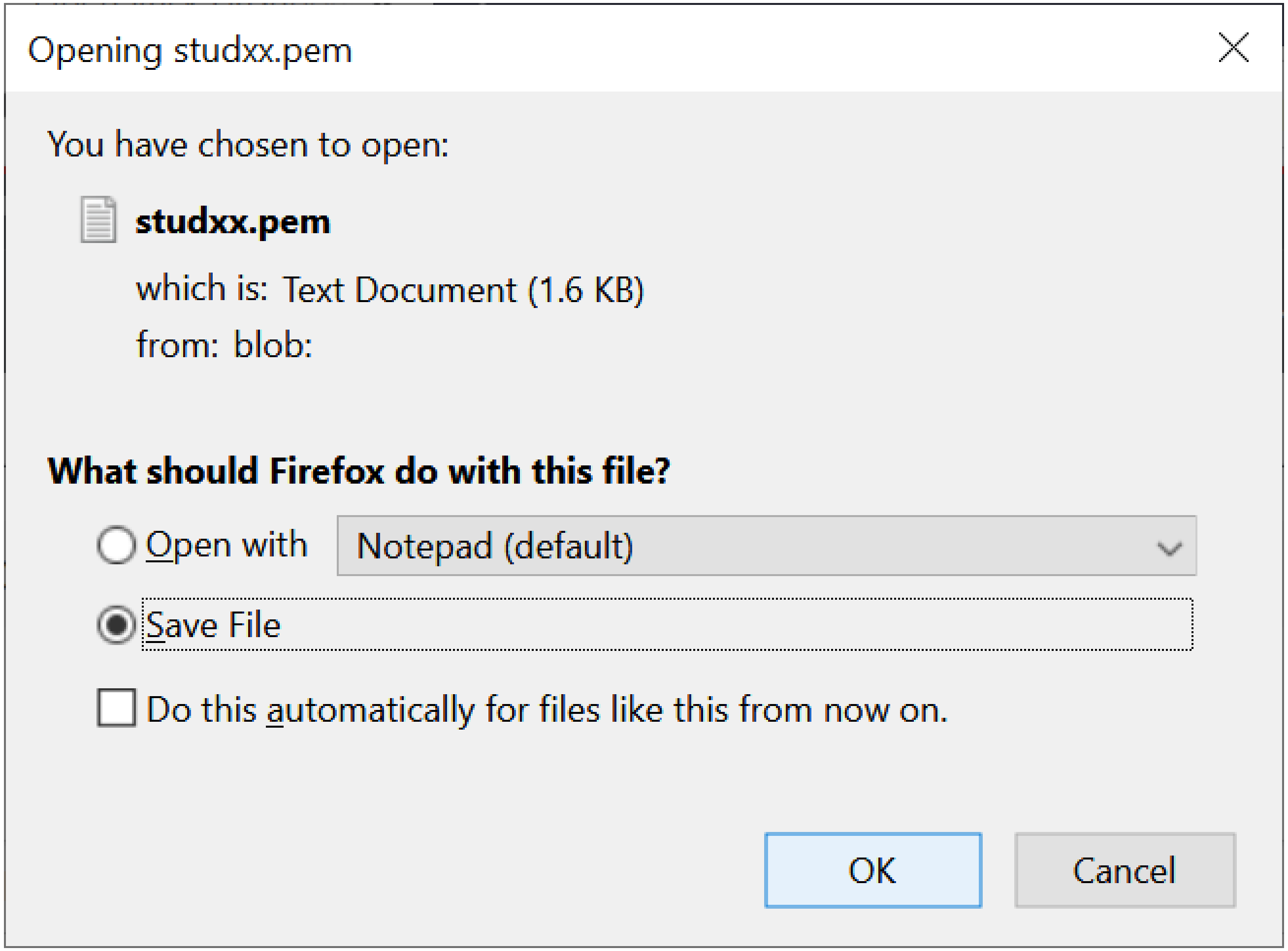
Generate Key Pair



Generate Key Pair

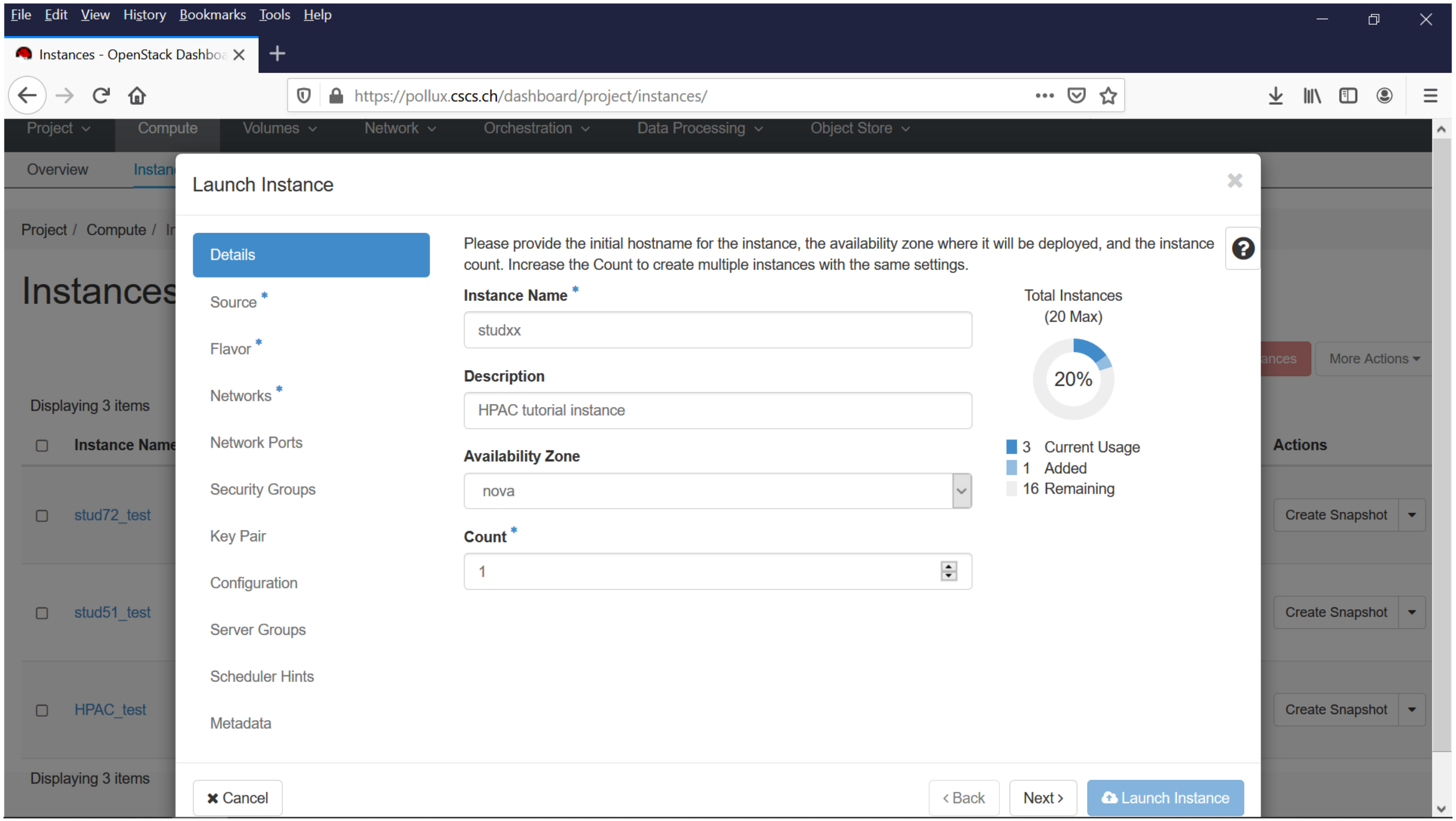


Generate Key Pair



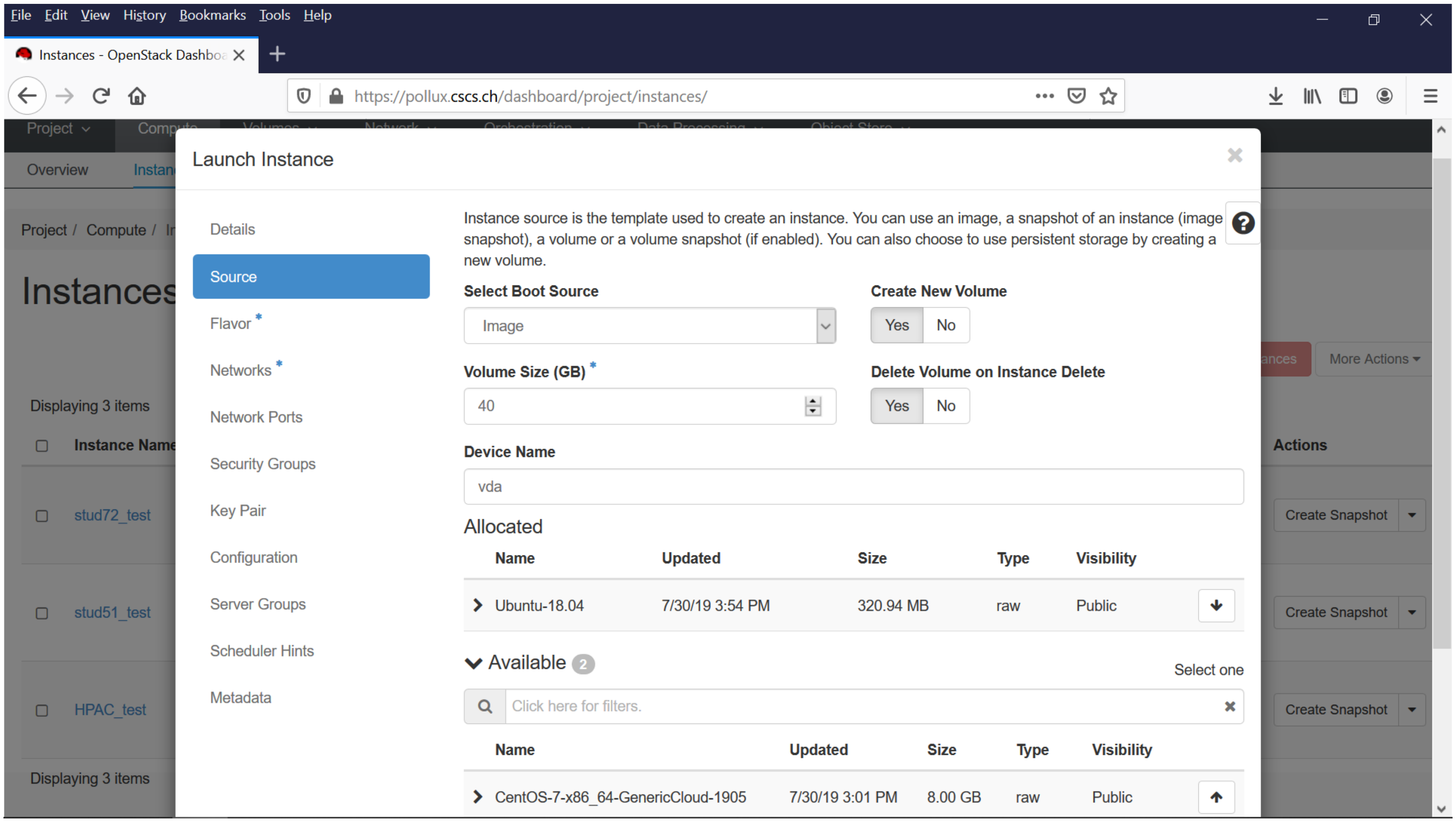
Launch Instance

- Launch a new instance called studxx



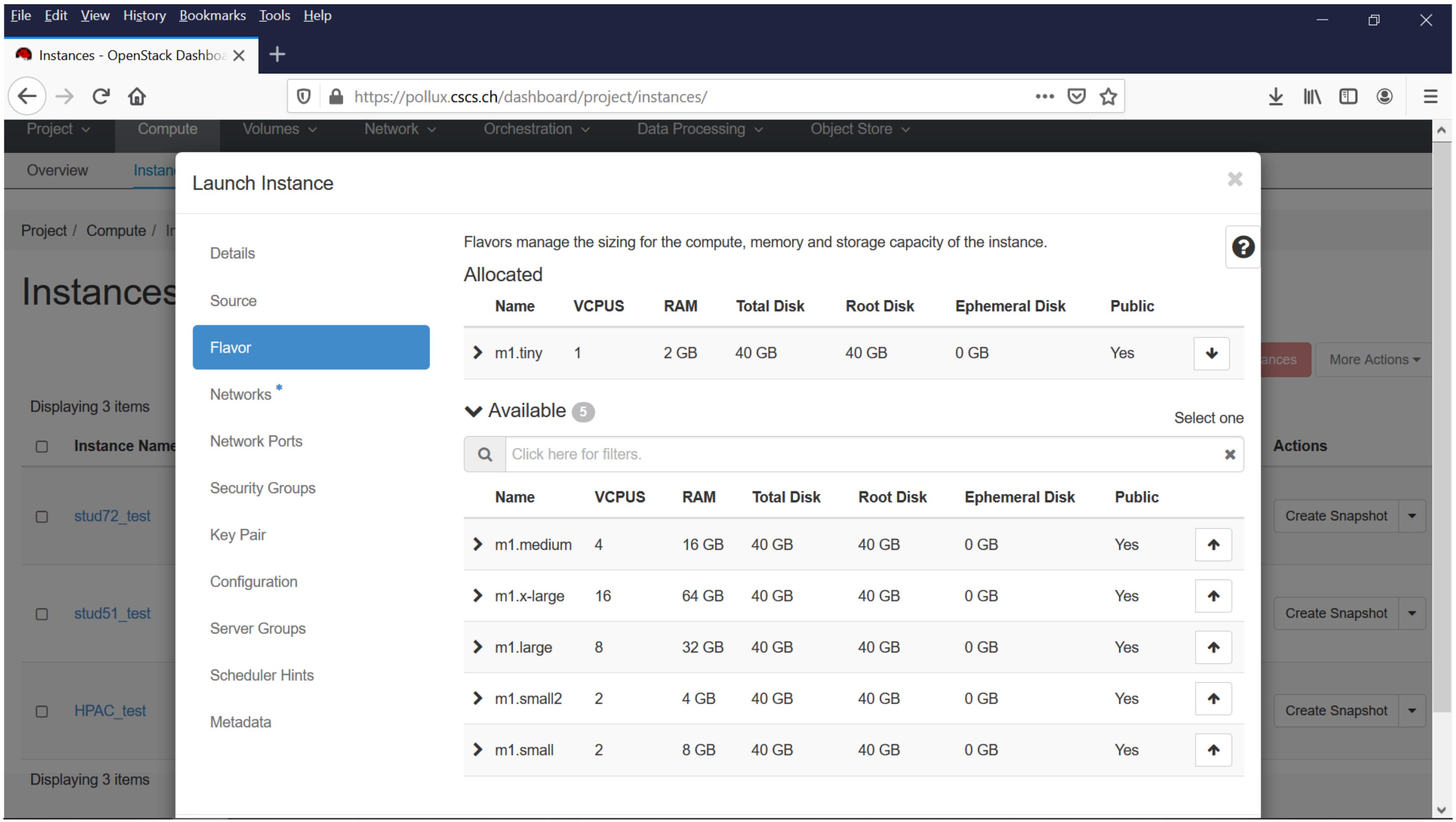
Launch Instance

- Next choose instance source and volume size



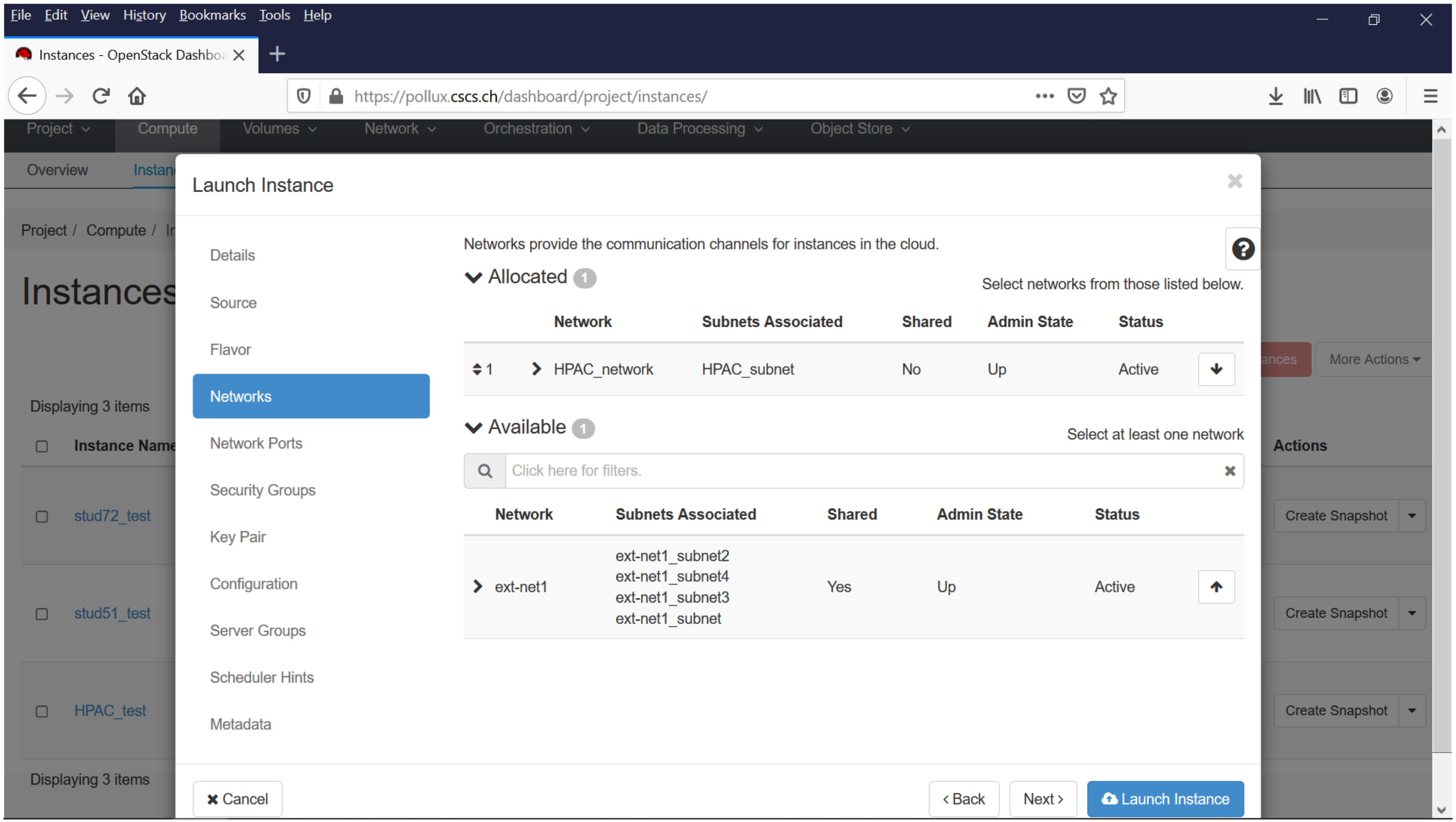
Specify hardware required

- Choose m1.tiny ‘flavour’ hardware config



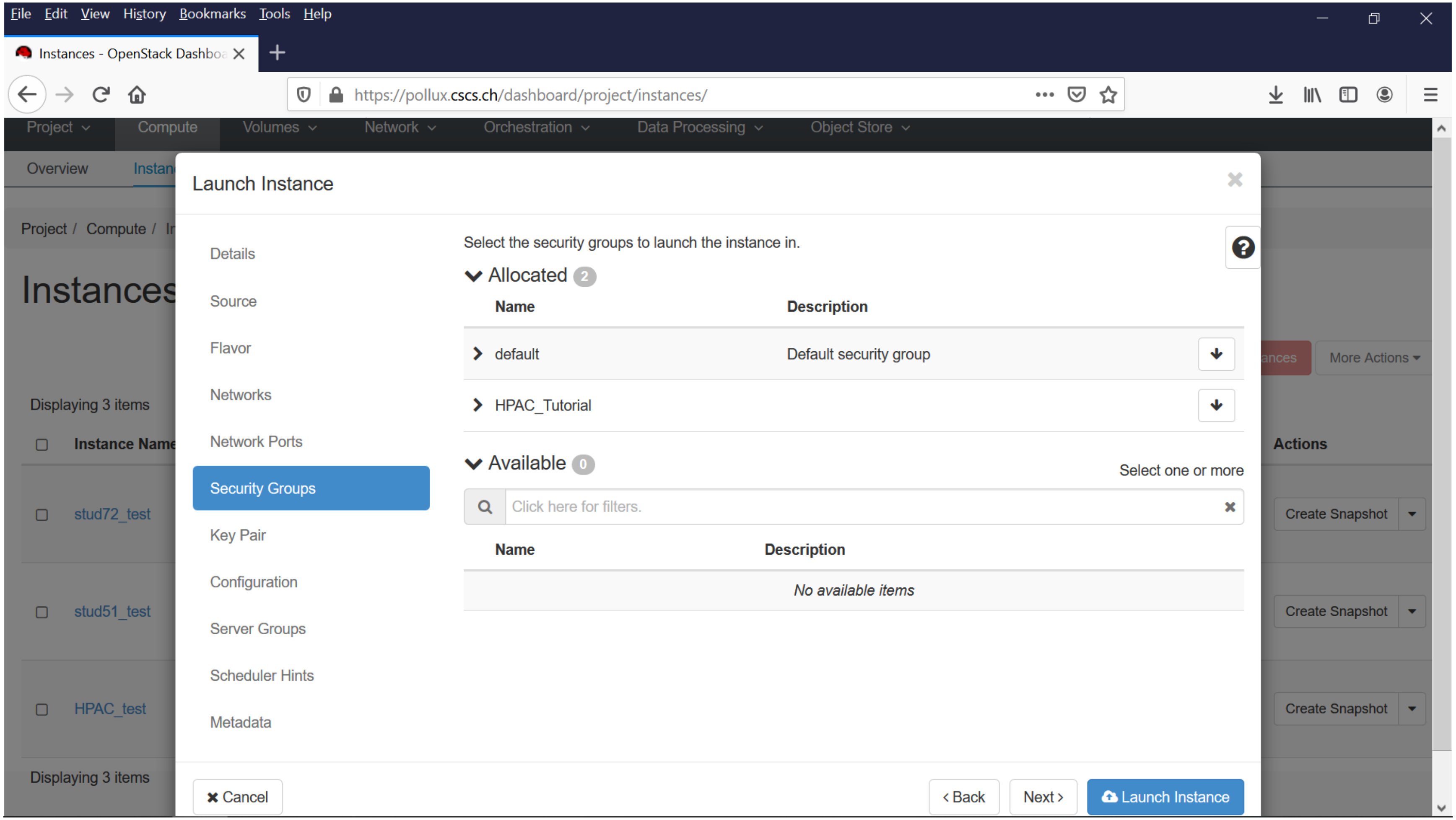
Connect to Network

- Connect to the HPAC_network



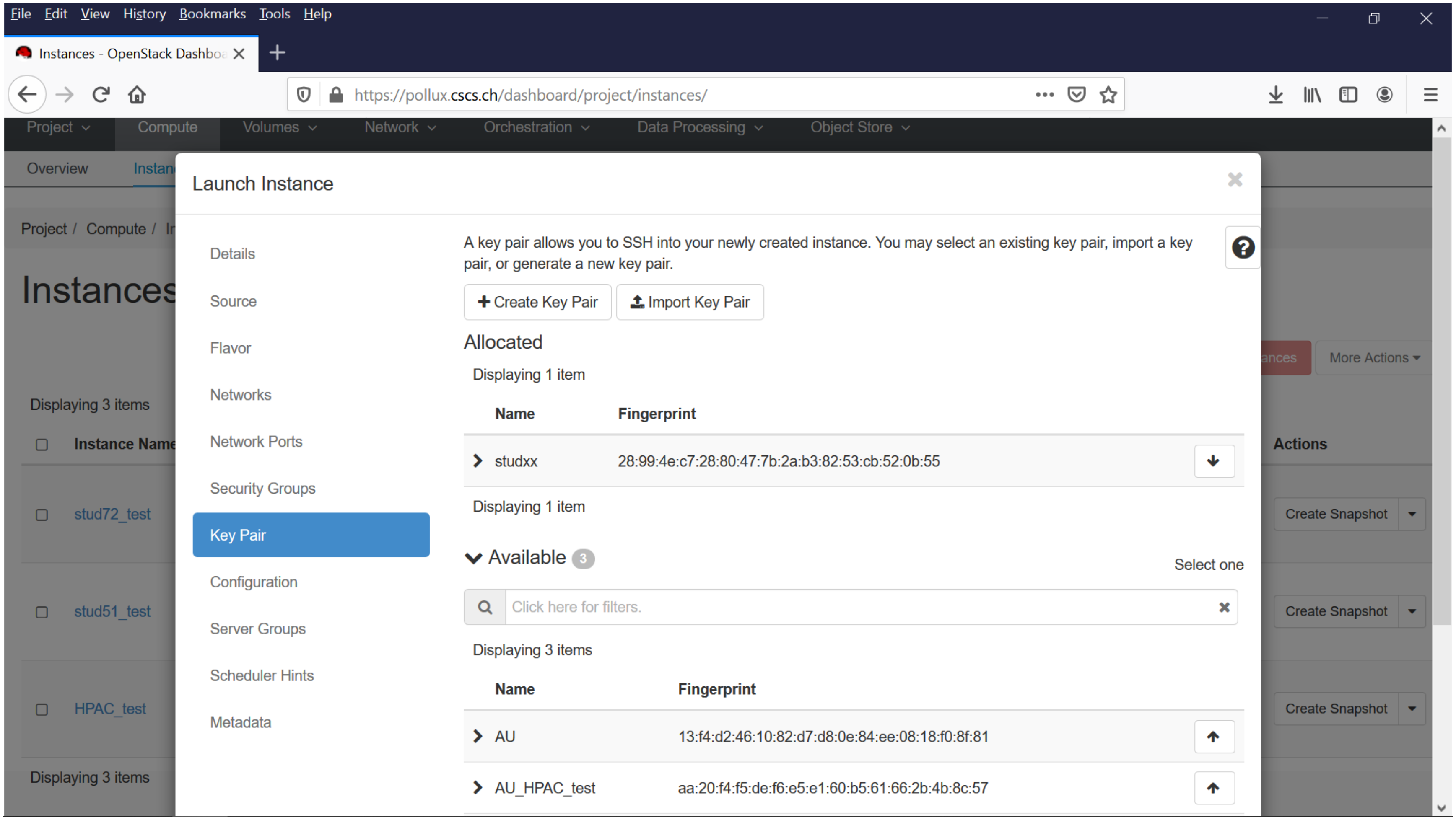
Select Security Groups

- Make sure to also select HPAC_tutorial security group

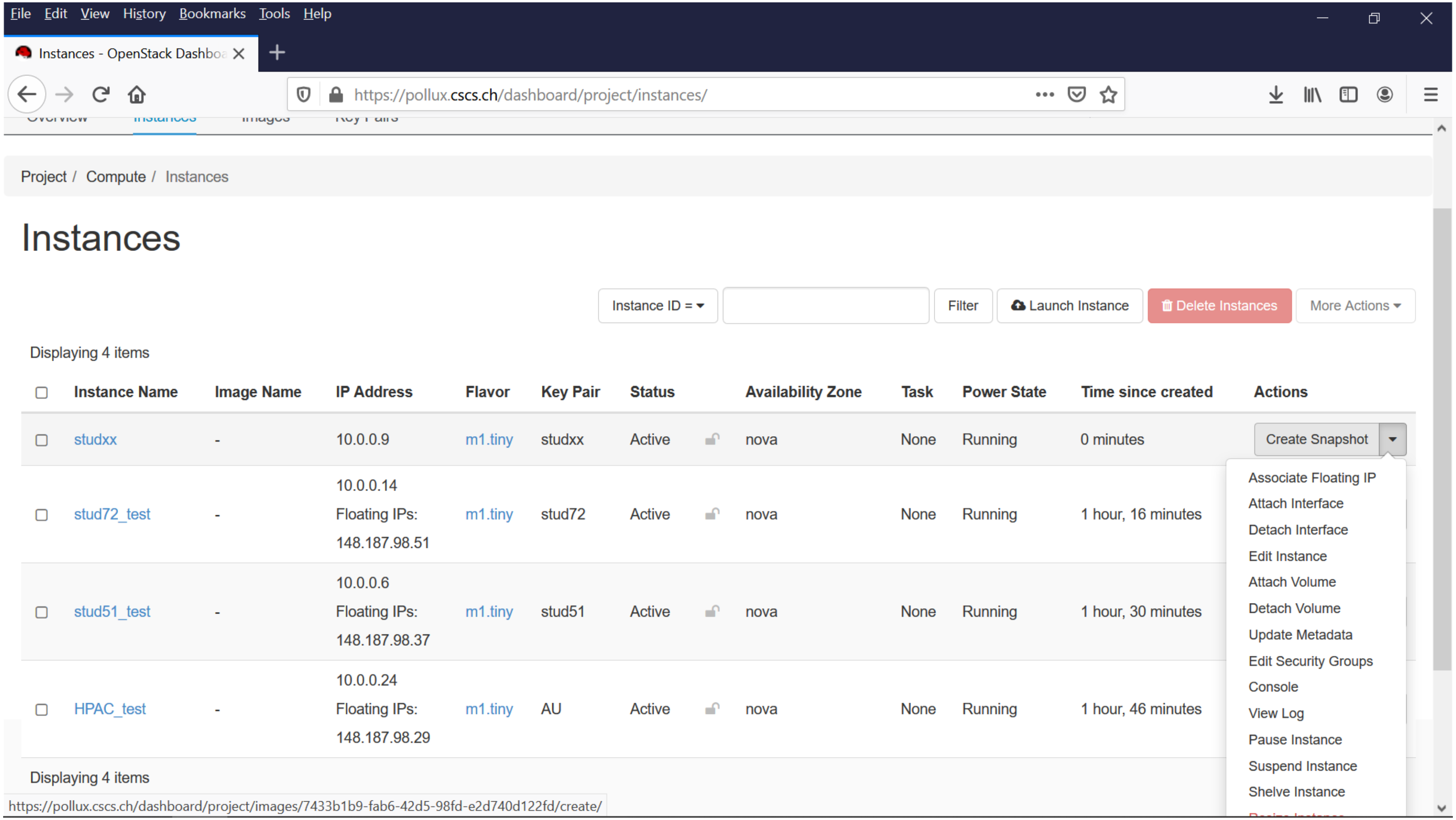


Specify key pair

- Choose the studxx key pair you created previously

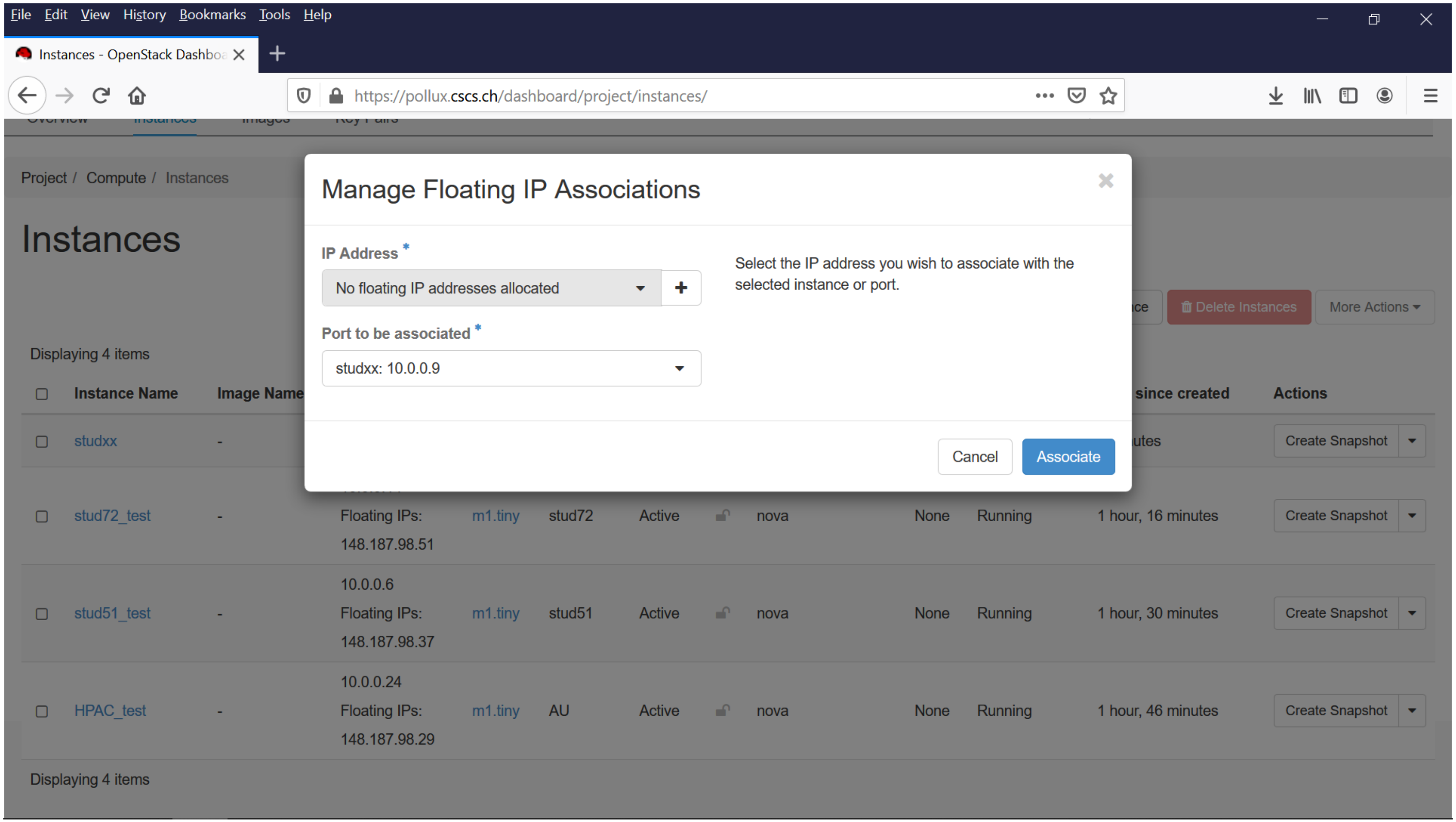


Associate floating IP



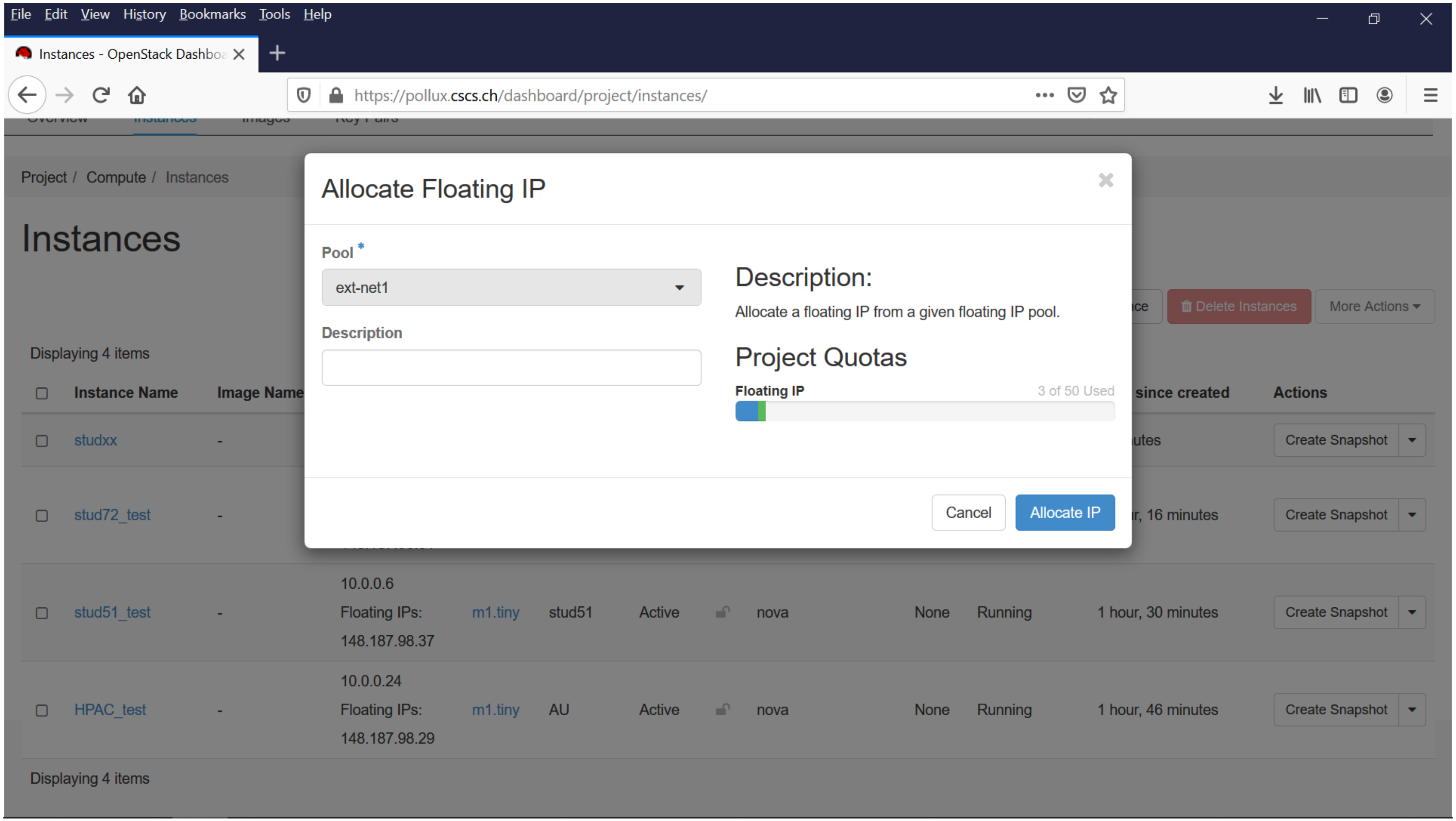
Associate floating IP

- Click the plus symbol



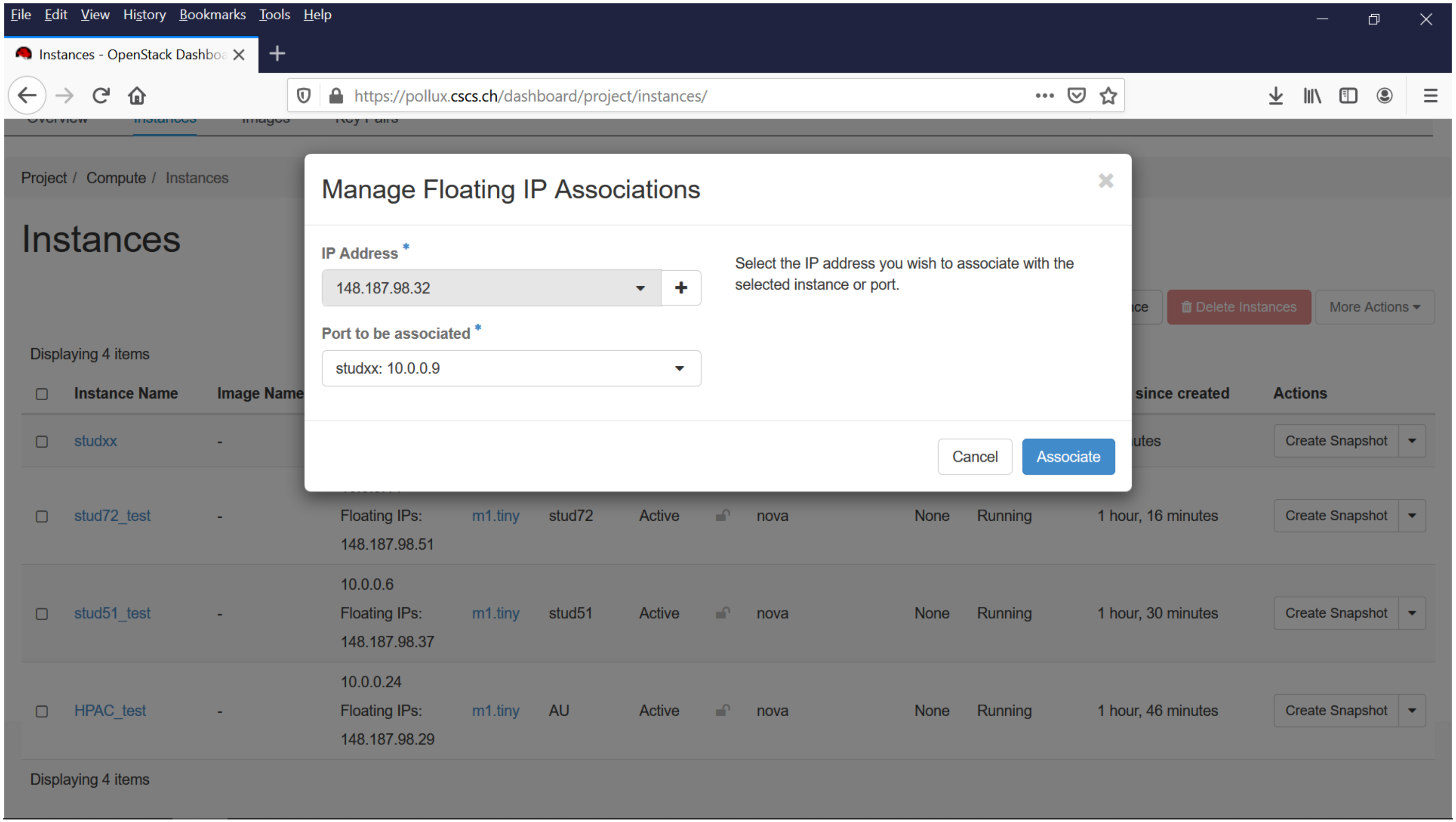
Associate floating IP

- Click Allocate IP



Associate floating IP

- An IP address has been allocated, click associate



Ping instance and login

ubuntu@studxx: ~

```
C:\Users\uptona\.ssh>ping 148.187.98.32

Pinging 148.187.98.32 with 32 bytes of data:
Reply from 148.187.98.32: bytes=32 time=21ms TTL=53
Reply from 148.187.98.32: bytes=32 time=19ms TTL=53
Reply from 148.187.98.32: bytes=32 time=16ms TTL=53
Reply from 148.187.98.32: bytes=32 time=15ms TTL=53

Ping statistics for 148.187.98.32:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 15ms, Maximum = 21ms, Average = 17ms

C:\Users\uptona\.ssh>ssh -i studxx.pem ubuntu@148.187.98.32
The authenticity of host '148.187.98.32 (148.187.98.32)' can't be established.
ECDSA key fingerprint is SHA256:Ig4VmISLXJ7PJ1zo185mObpP5L45YEQTZNv83HnWm9A.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '148.187.98.32' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-22-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Nov 21 17:52:15 UTC 2019

  System load:  0.0               Processes:           82
  Usage of /:   2.4% of 38.60GB   Users logged in:     0
  Memory usage: 6%                IP address for ens3: 10.0.0.9
  Swap usage:   0%

 * Overheard at KubeCon: "microk8s.status just blew my mind".

     https://microk8s.io/docs/commands#microk8s.status

  Get cloud support with Ubuntu Advantage Cloud Guest:
```

Set console password

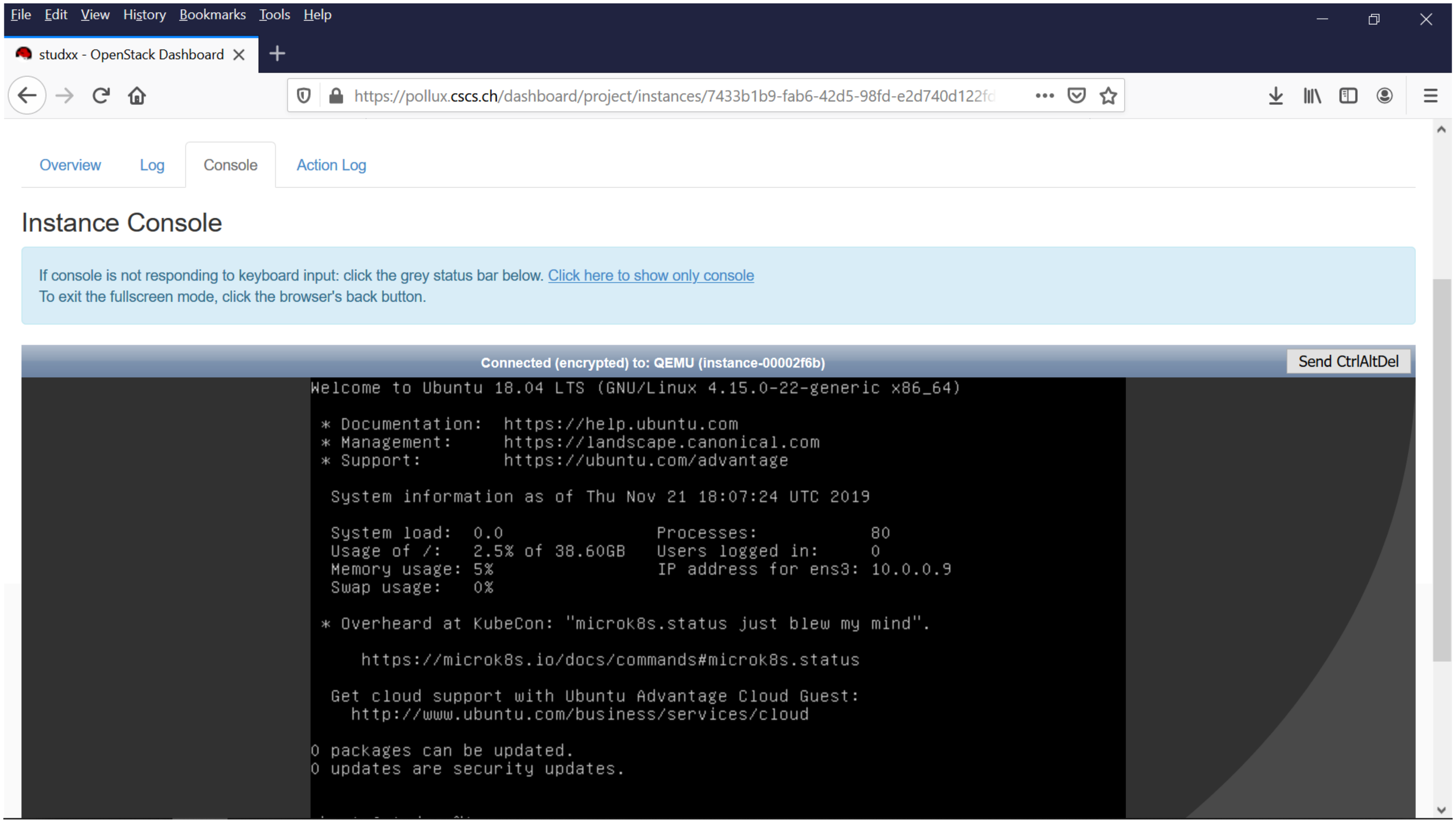
ubuntu@studxx: ~

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@studxx:~$ sudo passwd ubuntu
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@studxx:~$
```

Login from console

- Login using password just created for user ubuntu



Object Store

- "The **OpenStack Object Store** project, known as **Swift**, offers **cloud storage software** so that you can **store and retrieve lots of data with a simple API**. It's built for **scale** and optimized for **durability**, **availability**, and **concurrency** across the entire data set. Swift is ideal for storing **unstructured data** that can grow without bound."

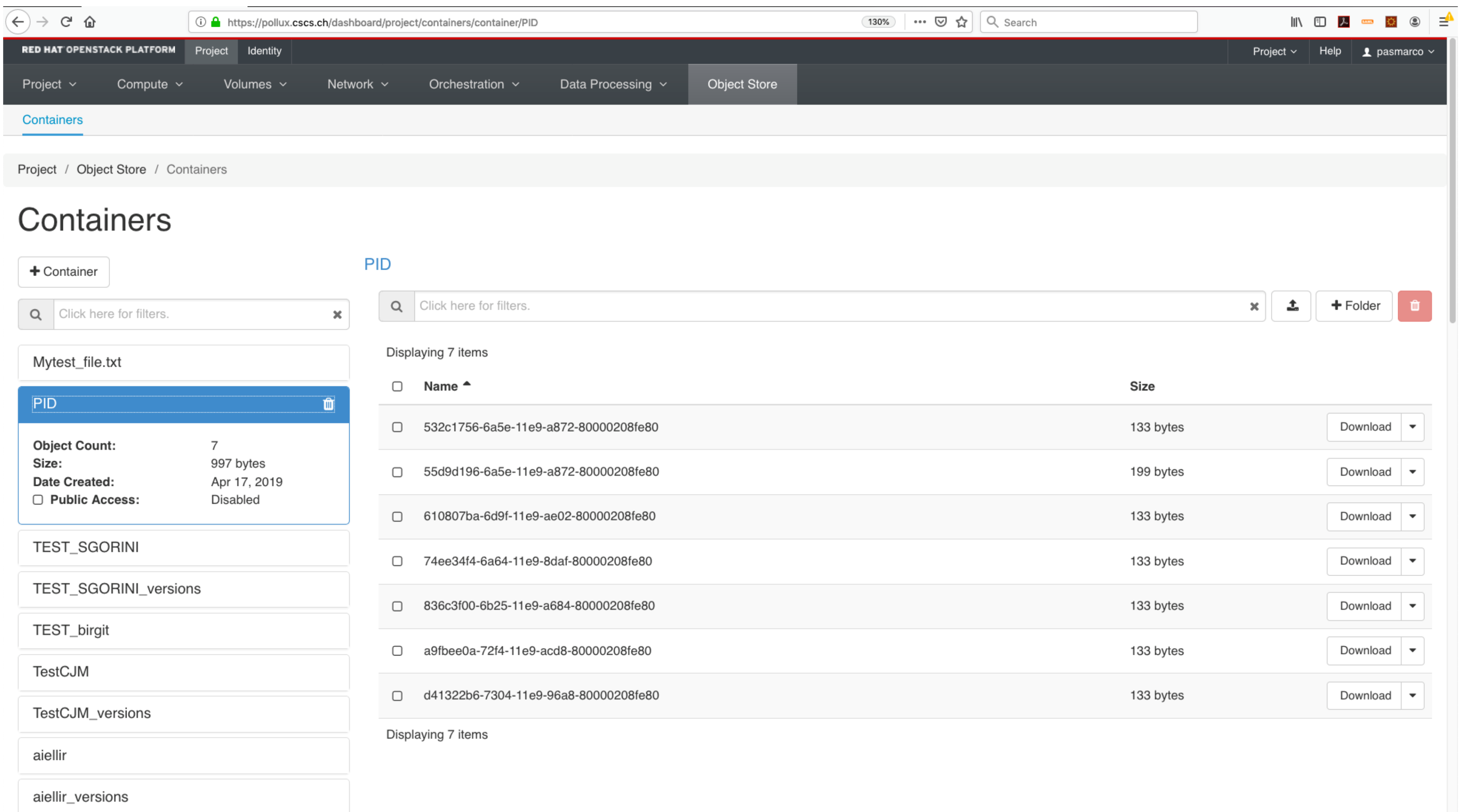
SWIFT

an OpenStack Community Project

Object Store

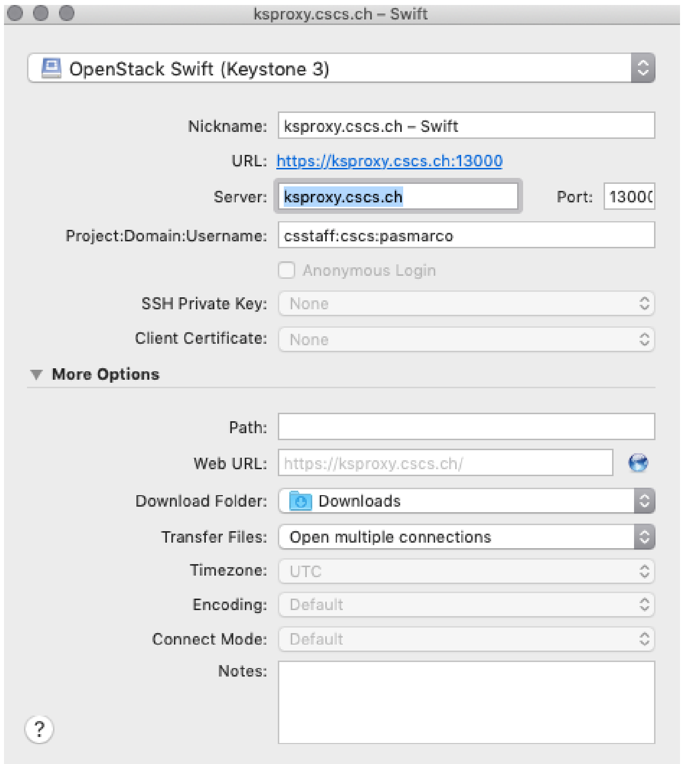
- Two entities:
  - **Objects** (~files)
  - **Containers** (~directories)
- Access through **REST APIs**, but there are many clients available (explained in the next slides)
- **Role based access control** (RBAC)
  - Users can have different roles on different projects (admin, normal user, data operator etc.)
- **Access control lists** (ACL)
  - **R and/or W** access can be potentially granted **to specific users on specific projects**
  - **Public** access is available
  - **Temporary URLs** for anonymous access
- **Metadata** can be associated to objects and containers
- **Documentation** for CSCS' Swift: https://user.cscs.ch/storage/object_storage/ (perhaps remove this/change to make HBP specific)

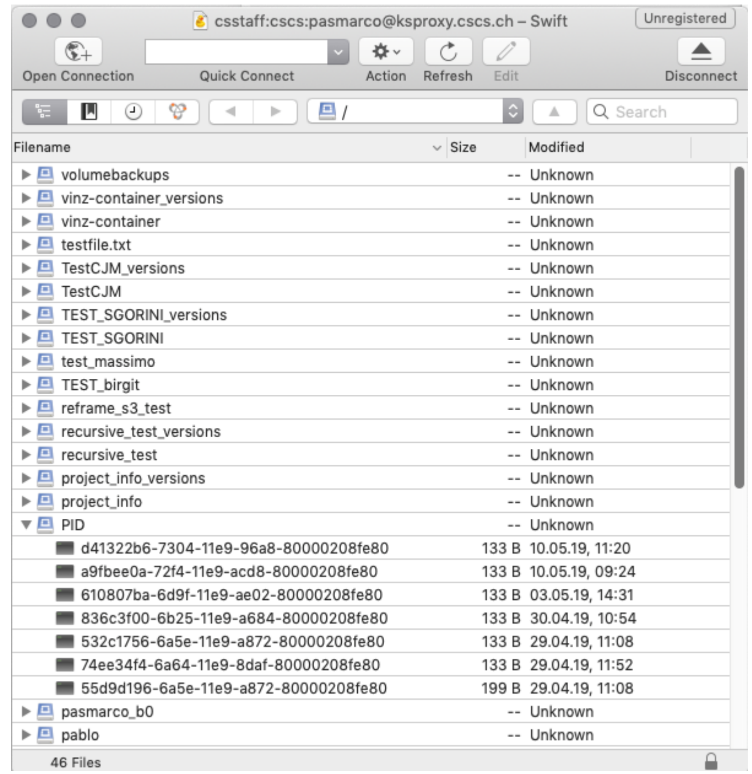
Object Store Web Interface



CyberDuck access

- Documented at https://user.cscs.ch/storage/object_storage/cyberduck/
- Let’s give this a go…





Access control lists

- Access through **REST APIs**, but there are many clients available (explained in the next slides)
- **Role based access control** (RBAC)
  - Users can have different roles on different projects (admin, normal user, data operator etc.)
- **Access control lists** (ACL)
  - **R and/or W** access can be potentially granted **to specific users on specific projects**
  - **Public** access is available
  - **Temporary URLs** for anonymous access
- **Metadata** can be associated to objects and containers
- **Documentation** for CSCS' Swift: https://user.cscs.ch/storage/object_storage/

Other Access options

- It's also possible to access Swift from:
  - **Command line interface (see tutorial sheet)**
  - **S3 interface** e.g. with s3curl
  - **Python OpenStack libraries**
  - **REST APIs**

Piz Daint – Scalable Compute

- 6th most powerful supercomputer in the world (TOP500 list November 2019)

Model	Cray XC50/XC40
XC50 Compute Nodes (Intel Haswell processor)	Intel® Xeon® E5-2690 v3 @ 2.60GHz (12 cores, 64GB RAM) and NVIDIA® Tesla® P100 16GB
XC40 Compute Nodes (Intel Broadwell processor)	Intel® Xeon® E5-2695 v4 @ 2.10GHz (18 cores, 64/128 GB RAM)
Login Nodes	Intel® Xeon® E5-2650 v3 @ 2.30GHz (10 cores, 256 GB RAM)
Interconnect Configuration	Aries routing and communications ASIC Dragonfly network topology
Scratch capacity	Piz Daint scratch filesystem: 8.8 PB

TOP 500 The List.

HOME LISTS STATISTICS RESOURCES ABOUT MEDIA KIT

Home / Lists / November 2019 / List

TOP500 List - November 2019

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

R_{peak} values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

previous 1 2 3 4 5 next

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,414,592	148,600.0	200,794.9	10,096
2	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125,712.0	7,438
3	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125,435.9	15,371
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100,678.7	18,482
5	Texas Advanced Computing Center/Univ. of Texas United States	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR Dell EMC	448,448	23,516.4	38,745.9	
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray/HPE	387,872	21,230.0	27,154.3	2,384

Co-funded by the European Union

Accessing Piz Daint

You should have already obtained an account ☺

The front end Ela is accessible via **ssh** as **ela.cscs.ch:**

- It provides a minimal Linux environment

```
$ ssh ela.cscs.ch
```

- ssh already available via MacOS/Linux, also now in Windows 10
- You can **ssh** the computing systems from Ela

```
$ ssh daint.cscs.ch
```

- Can use either command line or optionally GUI tools such as WinSCP to transfer data from laptop to/from filesystems on Piz Daint and Ela

Please note the following:

- **No programming environments** on the front end system
- User scratch space is **not accessible** from Ela

Piz Daint 101

1. Login via ssh to Ela
2. Login from Ela to Daint
3. See modules that are loaded
4. See available modules and search for specific modules
5. Load/unload module and select either hybrid or multicore nodes (DONE)
6. Transfer data to Ela/Piz Daint via command line
7. Toy example – submit batch job (https://user.cscs.ch/access/running/jobscript_generator/)
8. Graphical tools for transferring data e.g. WinSCP

ssh ela.cscs.ch – login node

OpenSSH SSH client

```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\uptona>ssh uptona@ela.cscs.ch
Warning: Permanently added the ECDSA host key for IP address '2001:620:808:4001::21' to the list of known
 hosts.
uptona@ela.cscs.ch's password:
Last login: Wed Nov 13 09:12:08 2019 from dhcp-133-237.cscs.ch

  =========================================================================
             IMPORTANT REMINDER FOR USERS of CSCS facilities

                   help@cscs.ch - http://user.cscs.ch
  =========================================================================



  =========================================================================
             IMPORTANT REMINDER FOR USERS of CSCS facilities

                   help@cscs.ch - http://user.cscs.ch
  =========================================================================


[uptona@ela6 ~]$ _
```

ssh daint - connecting to Piz Daint

OpenSSH SSH client

```
  ========================================================================



  ========================================================================
               IMPORTANT REMINDER FOR USERS of CSCS facilities

                     help@cscs.ch - http://user.cscs.ch
  ========================================================================


[uptona@ela6 ~]$ ssh daint
Password:
=================================================================
      IMPORTANT REMINDER FOR USERS of CSCS facilities

      help@cscs.ch - +41 91 610 82 10 - http://user.cscs.ch
=================================================================

Please load 'daint-gpu' module for using the GPU/Haswell nodes
or
load 'daint-mc' module for the Multicore/Broadwell nodes

For more info, please refer to the User Portal:
https://user.cscs.ch/access/running/piz_daint

uptona@daint102:~> _
```

module list - modules currently loaded

OpenSSH SSH client

```
uptona@daint102:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                              12) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
  2) cce/9.0.2                                     13) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  3) craype-network-aries                          14) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  4) craype/2.6.1                                  15) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  5) cray-libsci/19.06.1                           16) atp/2.1.3
  6) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari         17) perftools-base/7.1.1
  7) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari      18) PrgEnv-cray/6.0.5
  8) pmi/5.0.14                                    19) cray-mpich/7.7.10
  9) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari         20) slurm/19.05.3-2
 10) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari  21) craype-haswell
 11) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari        22) xalt/2.7.10
uptona@daint102:~>
```

module avail – see modules available

OpenSSH SSH client

```
uptona@daint104:~> module avail

--------------------------------------------- /opt/cray/pe/perftools/7.1.1/modulefiles ---------------------------------------------
perftools              perftools-lite-events perftools-lite-hbm    perftools-nwpc
perftools-lite         perftools-lite-gpu    perftools-lite-loops  perftools-preload

---------------------------------------------------- /opt/cray/ari/modulefiles ----------------------------------------------------
aeld/1.3.3-7.0.1.1_4.10__g6016d48.ari(default)             logcb/1.3.1-7.0.1.1_3.4__ga55b812.ari(default)
alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari(default)           lustre-utils/2.3.5-7.0.1.1_5.8__g0e6e9b2.ari(default)
alpscomm/1.3.12-7.0.1.1_4.10__ga8f75ce.ari(default)        ncmd/1.3.6-7.0.1.1_4.13__g036045e.ari(default)
apptermd/1.3.1-7.0.1.1_3.8__g4a70d82.ari(default)          nhm/5.7.2-7.0.1.1_3.4__g8895b35.ari(default)
ccm/2.5.7-7.0.1.1_5.8__g83c42ff.ari(default)               nodehealth/5.6.19-7.0.1.1_5.8__gb89faf6.ari(default)
codbc/2.5.105-7.0.1.1_1.4__g811bbf2.ari(default)           nodestat/2.3.85-7.0.1.1_3.5__gc6218bb.ari(default)
comm_msg/1.2.3-7.0.1.1_1.6__g60fcb6d.ari(default)          pdsh/2.27-7.0.1.1_5.4__g70b69a8.ari(default)
configparse/2.4.41-7.0.1.1_1.4__g8c68499.ari(default)      rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari(default)
daemontools/1.3.1-7.0.1.1_1.5__gbf01d9d.ari(default)       rdma-credentials/1.2.25-7.0.1.1_4.8__ga0a409f.ari(default)
datawarp/3.0.9-7.0.1.1_4.21__gb93afd5.ari(default)         sdb/3.3.795-7.0.1.1_3.8__gd16b6f4.ari(default)
dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari(default)             socketauth/1.3.1-7.0.1.1_1.5__gfdd1da0.ari(default)
dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703(default)           swrap/1.3.1-7.0.1.1_1.5__gc085a9f.ari(default)
dw_wlm/3.0.11-7.0.1.1_3.7__g1462d48.ari(default)           sysadm/2.4.136-7.0.1.1_4.7__g4685a09.ari(default)
dws/3.0.27-7.0.1.1_5.1__g8583f3e.ari(default)              system-config/3.5.3009-7.0.1.1_3.7__gb85ddb0b.ari(default)
gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari(default)    sysutils/2.5.72-7.0.1.1_1.4__g0dd4e0a.ari(default)
hosts/2.5.111-7.0.1.1_5.9__g62e13b4.ari(default)           udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari(default)
imps/3.8.4291-7.0.1.1_5.6__g47ffcd82.ari(default)          udwfs/3.0.5-7.0.1.1_4.19__g0828810.ari(default)
isvaccel/6.0.0-7.0.1.1_3.5__gf0c05b4.ari                   ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari(default)
job/2.2.4-7.0.1.1_3.8__g36b56f4.ari(default)               wlm_detect/1.3.3-7.0.1.1_4.6__g7109084.ari(default)
kdreg/2.2.5-7.0.1.1_4.7__ge6d8d0e.ari(default)             wlm_trans/1.5.6-7.0.1.1_4.8__gad40448.ari(default)
krca/2.2.6-7.0.1.1_5.8__gb641b12.ari(default)              xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari(default)
linux-nvme-ctl/0.0_2.1.4-7.0.1.1_3.8__g375a019.ari(default) xtgetconfig/2.3.90-7.0.1.1_4.5__g3d11e7d.ari(default)
llm/21.4.596-7.0.1.1_5.7__g8b75441.ari(default)

----------------------------------------------- /opt/cray/pe/craype/2.6.1/modulefiles -----------------------------------------------
craype-accel-host      craype-haswell        craype-hugepages16M   craype-intel-knc      craype-x86-cascadelake
craype-accel-nvidia20  craype-hugepages1G    craype-hugepages32M   craype-ivybridge      craype-x86-skylake
craype-accel-nvidia35  craype-hugepages2G    craype-hugepages64M   craype-mic-knl
craype-accel-nvidia52  craype-hugepages2M    craype-hugepages128M  craype-network-aries
craype-accel-nvidia60  craype-hugepages4M    craype-hugepages256M  craype-network-none
craype-broadwell       craype-hugepages8M    craype-hugepages512M  craype-sandybridge

-------------------------------------------------- /apps/daint/system/modulefiles --------------------------------------------------
cksys/2.3.0                              intel-opencl/1.2(default)                    shifter/17.08.00(default)
```

module avail – search for Python

- Rather than go through all the modules, can return a list with all the modules that match a specific argument, e.g. in this case we look for Cray Python modules

```
uptona@daint102:~> module avail cray-python

-------------------------------------------------------- /opt/modulefiles --------------------------------------------------------
cray-python/2.7.15.6         cray-python/2.7.15.7(default) cray-python/3.6.5.6         cray-python/3.6.5.7
uptona@daint102:~>
```

module show - Info about specific module

- In addition to listing modules that match our interest, we can also find out more about them
- In this case, we see that it is a module for Python on the Cray XC system

```
uptona@daint102:~> module show cray-python/3.6.5.7
-------------------------------------------------------------------
/opt/modulefiles/cray-python/3.6.5.7:

prepend-path     PATH /opt/python/3.6.5.7/bin
prepend-path     MANPATH /opt/python/3.6.5.7/share/man
prepend-path     PYTHONPATH /opt/python/3.6.5.7
prepend-path     LD_LIBRARY_PATH /opt/cray/pe/gcc-libs
module-whatis    python for XC
-------------------------------------------------------------------

uptona@daint102:~> _
```

module daint-gpu – use Hybrid nodes

- Hybrid CPU/GPU nodes – load if you want to use GPUs

```
uptona@daint105:~> module load daint-gpu
uptona@daint105:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                          13) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  2) cce/9.0.2                                 14) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  3) craype-network-aries                      15) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  4) craype/2.6.1                              16) atp/2.1.3
  5) cray-libsci/19.06.1                       17) perftools-base/7.1.1
  6) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari     18) PrgEnv-cray/6.0.5
  7) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari  19) cray-mpich/7.7.10
  8) pmi/5.0.14                                20) slurm/19.05.3-2
  9) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari     21) craype-haswell
 10) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari  22) xalt/2.7.10
 11) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari    23) daint-gpu
 12) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
uptona@daint105:~>
```

See in the list that the GPU module is loaded

module daint-mc – use multicore nodes

- Multicore nodes with 2x18 cores – ideal for OpenMP

OpenSSH SSH client

```
uptona@daint105:~> module unload daint-gpu
uptona@daint105:~> module load daint-mc
uptona@daint105:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.11.3                          13) dvs/2.12_2.2.151-7.0.1.1_5.6__g7eb5e703
  2) cce/9.0.2                                 14) alps/6.6.56-7.0.1.1_4.10__g2e60a7e4.ari
  3) craype-network-aries                      15) rca/2.2.20-7.0.1.1_4.9__g8e3fb5b.ari
  4) craype/2.6.1                              16) atp/2.1.3
  5) cray-libsci/19.06.1                       17) perftools-base/7.1.1
  6) udreg/2.3.2-7.0.1.1_3.9__g8175d3d.ari     18) PrgEnv-cray/6.0.5
  7) ugni/6.0.14.0-7.0.1.1_7.10__ge78e5b0.ari  19) cray-mpich/7.7.10
  8) pmi/5.0.14                                20) slurm/19.05.3-2
  9) dmapp/7.1.1-7.0.1.1_4.8__g38cf134.ari     21) xalt/2.7.10
 10) gni-headers/5.0.12.0-7.0.1.1_6.7__g3b1768f.ari  22) craype-broadwell
 11) xpmem/2.2.19-7.0.1.1_3.7__gdcf436c.ari    23) daint-mc
 12) job/2.2.4-7.0.1.1_3.8__g36b56f4.ari
uptona@daint105:~> _
```

See in the list that the multicore module is loaded

Filesystems

Filesystems

	/scratch (Piz Daint)	/scratch (Clusters)	/users	/project	/store
Type	Lustre	GPFS	GPFS	GPFS	GPFS
Quota	Soft quota 1 M files	None	10 GB/user 100K files	Maximum 50K files/TB	Maximum 50K files/TB
Expiration	30 days	30 days	Account closure	End of the project	End of the contract
Data Backup	None	None	90 days	90 days	90 days
Access Speed	Fast	Fast	Slow	Medium	Slow
Capacity	8.8 PB	1.4 PB	86 TB	4.7 PB	3.6 PB

Soft quota:

- Soft quota on **scratch** to prevent excessive loads on the Lustre filesystem
- Quota reached: **warning at submit time**, **no job submission** allowed

Filesystems

/users and /project storage

Shared parallel filesystems based on the IBM GPFS software:

- Accessible from the login nodes using native GPFS client
- Storage space for datasets, shared code or configuration scripts
- Better performance with larger files (archive small files with **tar**)

Users are NOT supposed to run jobs here:

- The emphasis is on **reliability over performance**
- All directories are **backed up** with [GPFS snapshots]
- No cleaning policy until **3-months** after the end of the project

Environment variables pointing to personal folders:

- **$HOME** points to **/users/$USER**
- **$PROJECT** points to **/project/<group_id>/$USER**

Filesystems

/scratch filesystem

Fast workspace for running jobs:

- Designed for **performance** rather than reliability
- **Cleaning policy**: files **older than 30 days deleted** daily
- **No backup:** transfer data after job completion

Performance of Piz Daint scratch (Lustre filesystem):

- **Soft quota on inodes** (files and folders) **to avoid** large numbers of small files
- Occupancy impacts performance:
  - **> 60%**: we will ask you to **remove unnecessary data immediately**
  - **> 80%**: we will free up disk space **manually removing data**

All CSCS systems provide a scratch personal folder:

- the variable **$SCRATCH** points to the user space

Filesystems

What this means in practice:

1) Store important input/output data in $HOME in ela
2) Move data needed for running job to $SCRATCH in Piz Daint
3) Run job on data in $SCRATCH using batch script
4) Move output data such as results from job to $HOME or optionally local file filesystem

Let's look at how we do this with a toy example…

Copy data to $HOME using SCP

- First we copy data from the local filesystem to the $HOME directory

Command Prompt

```
C:\CSCS_work_files\HBP\HPAC_handson>scp C:\CSCS_work_files\HBP\HPAC_handson\HPAC_tutorial_1.txt uptona@ela.cscs.ch:/users/uptona/
HPAC_tutorial_1.txt                                                                      100%   75     13.8KB/s   00:00

C:\CSCS_work_files\HBP\HPAC_handson>scp C:\CSCS_work_files\HBP\HPAC_handson\HPAC_tutorial_1.txt uptona@ela.cscs.ch:/users/uptona/
```

Copy input data to $SCRATCH

- Next we copy input data from $HOME to $SCRATCH

```
 ========================================================================
             IMPORTANT REMINDER FOR USERS of CSCS facilities

                   help@cscs.ch - http://user.cscs.ch
 ========================================================================


[uptona@ela1 ~]$ ls -a
.              .bashrc  .config   .fonts      hbp_logo.jpg         .inputrc  .muttrc   .ssh      .xemacs
..             bin      .dvipsrc  .forward    HPAC_tutorial_1.txt  .local    .profile  .viminfo  .xim.template
.bash_history  .cache   .emacs    .gnu-emacs  HPAC_tutorial.txt    .mozilla  .ros      .vimrc    .xinitrc.template
[uptona@ela1 ~]$ ls
bin  hbp_logo.jpg  HPAC_tutorial_1.txt  HPAC_tutorial.txt
[uptona@ela1 ~]$
```

OpenSSH SSH client

```
[uptona@ela1 ~]$ ssh daint
Last login: Mon Nov 18 11:18:03 2019 from 148.187.1.6
=================================================================
      IMPORTANT REMINDER FOR USERS of CSCS facilities

      help@cscs.ch - +41 91 610 82 10 - http://user.cscs.ch
=================================================================

Please load 'daint-gpu' module for using the GPU/Haswell nodes
or
load 'daint-mc' module for the Multicore/Broadwell nodes

For more info, please refer to the User Portal:
https://user.cscs.ch/access/running/piz_daint

uptona@daint105:~> scp $HOME/HPAC_tutorial_1.txt $SCRATCH
uptona@daint105:~>
```

Batch script

- Instead of entering the information about allocations and jobs through the command line, a *batch script* is used
- All the details can be written and submitted through the sbatch command
- This is the preferred way of running complex jobs or making a large amount of submissions
- It may seem daunting, but to help you there is a jobscript generator that you can use found here: https://user.cscs.ch/access/running/jobscript_generator/

Run job on $SCRATCH

- For our toy example we run a simple Python program to print 'Hello from Python' on different tasks

/scratch/snx3000/uptona/hello_python.sbatch - Daint connection - Editor - WinSCP

```
#!/bin/bash -l
#SBATCH --job-name="HPAC_test"
#SBATCH --mail-type=ALL
#SBATCH --mail-user=alex.upton@cscs.ch
#SBATCH --time=00:01:00
#SBATCH --nodes=2
#SBATCH --ntasks-per-core=2
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=1
#SBATCH --partition=normal
#SBATCH --constraint=gpu
#SBATCH --hint=multithread

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export CRAY_CUDA_MPS=1

srun python -c \
'import os; \
print("Hello from Python task {} on node {}".format \
(os.environ["SLURM_PROCID"], os.environ["HOSTNAME"]))'
```

```
uptona@daint106:/scratch/snx3000/uptona> sbatch hello_python.sbatch
Submitted batch job 18178790
uptona@daint106:/scratch/snx3000/uptona>
```

Inspect output file

- Output file named slurm-18178790.out is created
- Can open it using e.g. vim to see results

OpenSSH SSH client

```
-rw-r--r--    1 uptona csstaff     44 Nov 18 00:03 test.txt
uptona@daint104:/scratch/snx3000/uptona> ls -all
total 452
drwxr-xr-x    3 uptona csstaff   4096 Nov 18 18:07 .
drwxr-xr-x 2613 root   root    135168 Nov 15 15:36 ..
lrwxrwxrwx    1 root   root        29 Oct 31 10:34 .sarus -> /scratch/snx3000/sarus/uptona
drwxr-xr-x    4 uptona csstaff   4096 Sep  5 17:35 .shifter
-rw-r--r--    1 uptona csstaff     75 Nov 18 17:40 HPAC_tutorial_1.txt
-rw-r--r--    1 uptona csstaff 294882 Nov 13 10:24 hbp_logo.jpg
-rw-r--r--    1 uptona csstaff    321 Nov 18 17:54 hello.sbatch
-rw-r--r--    1 uptona csstaff    520 Nov 18 18:02 hello_python.sbatch
-rw-r--r--    1 uptona csstaff    678 Nov 18 18:07 slurm-18178790.out
-rw-r--r--    1 uptona csstaff     44 Nov 18 00:03 test.txt
uptona@daint104:/scratch/snx3000/uptona> vi slurm-18178790.out
```

OpenSSH SSH client

```
Hello from Python task 2 on node nid03384
Hello from Python task 8 on node nid03384
Hello from Python task 7 on node nid03384
Hello from Python task 1 on node nid03384
Hello from Python task 5 on node nid03384
Hello from Python task 3 on node nid03384
Hello from Python task 6 on node nid03384
Hello from Python task 4 on node nid03384
Hello from Python task 0 on node nid03384
Hello from Python task 14 on node nid03384
Hello from Python task 15 on node nid03384
Hello from Python task 9 on node nid03384
Hello from Python task 10 on node nid03384
Hello from Python task 11 on node nid03384
Hello from Python task 12 on node nid03384
Hello from Python task 13 on node nid03384
```

Copy output file to $HOME

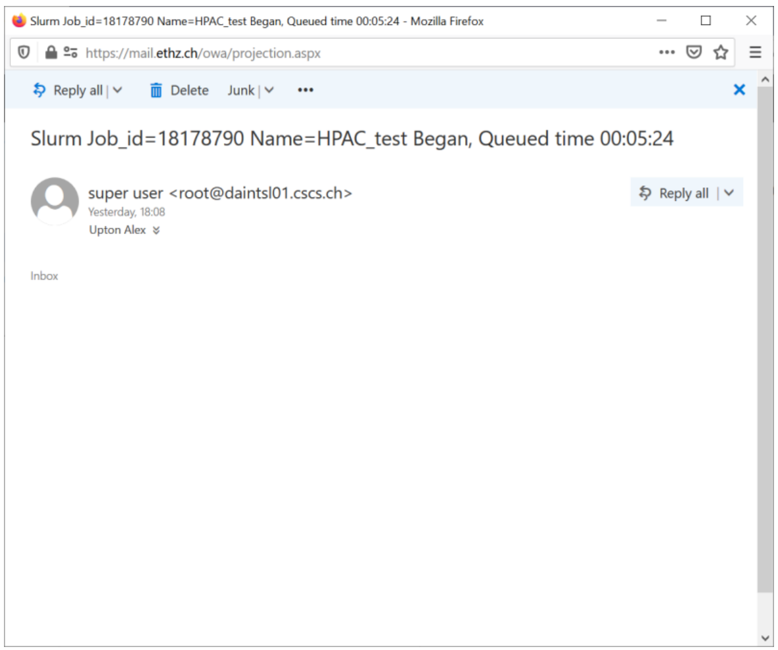
- In the final step, we copy the output file to the home directory, and can optionally open again with e.g. Vim to see that it is the same

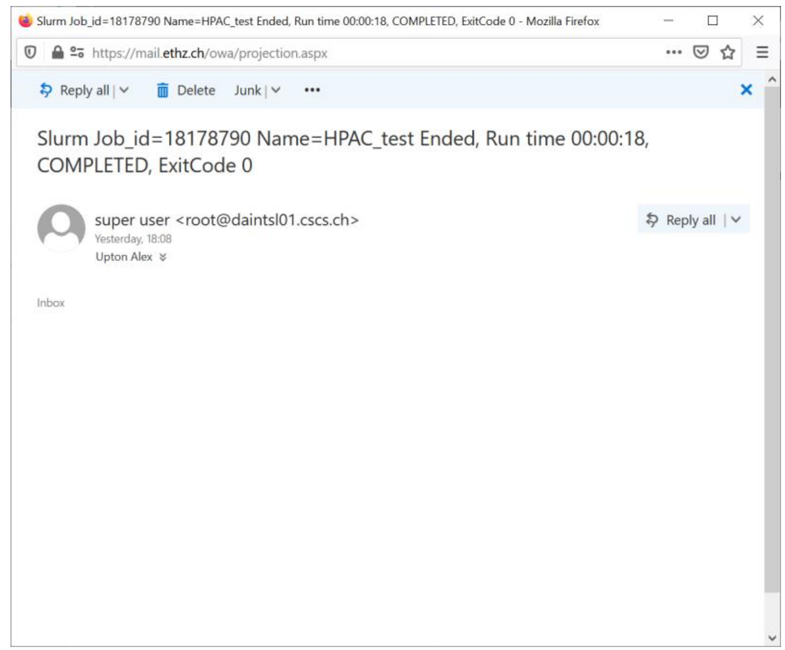
OpenSSH SSH client

```
uptona@daint104:~> scp $SCRATCH/slurm-18178790.out $HOME/
uptona@daint104:~> cd $HOME
uptona@daint104:~> ls -all
total 632
drwx------    11 uptona csstaff   4096 Nov 18 18:29 .
drwxr-xr-x  2365 root   root    131072 Jan 18  2017 ..
-rw-------     1 uptona csstaff  23656 Nov 18 18:22 .bash_history
-rw-r--r--     1 uptona csstaff   1177 May 15  2017 .bashrc
drwxr-xr-x.    3 uptona csstaff   4096 Jul 30 12:19 .cache
drwxr-xr-x.    3 uptona csstaff   4096 Jul 30 12:19 .config
-rw-r--r--     1 uptona csstaff    315 May 12  2010 .dvipsrc
-rw-r--r--     1 uptona csstaff   1637 Feb 15  2010 .emacs
drwxr-xr-x     2 uptona csstaff   4096 May  5  2010 .fonts
-rw-r--r--     1 uptona csstaff     19 Jun 18 08:00 .forward
-rw-r--r--     1 uptona csstaff  18251 Mar 23  2015 .gnu-emacs
-rw-r--r--     1 uptona csstaff    861 May 19  2006 .inputrc
drwxr-xr-x.    3 uptona csstaff   4096 Jul 30 12:19 .local
drwxr-xr-x     2 uptona csstaff   4096 May  5  2010 .mozilla
-rw-r--r--     1 uptona csstaff   6043 Mar 10  2015 .muttrc
-rw-r--r--     1 uptona csstaff   1028 May 15  2017 .profile
drwxr-xr-x     2 uptona csstaff   8192 Aug  6 17:08 .ros
drwx------.    2 uptona csstaff   4096 Nov 15 14:09 .ssh
-rw-------     1 uptona csstaff   2040 Nov 18 18:29 .viminfo
-rw-r--r--     1 uptona csstaff    849 Nov 24  2016 .vimrc
drwxr-xr-x     2 uptona csstaff   4096 Apr 23  2014 .xemacs
-rw-r--r--     1 uptona csstaff   1940 Aug 30  2012 .xim.template
-rwxr-xr-x     1 uptona csstaff   1446 Jun 12  2015 .xinitrc.template
-rw-r--r--.    1 uptona csstaff     75 Nov 13 09:18 HPAC_tutorial.txt
-rw-r--r--.    1 uptona csstaff     75 Nov 18 17:24 HPAC_tutorial_1.txt
drwxr-xr-x     2 uptona csstaff   4096 May  5  2010 bin
-rw-r--r--.    1 uptona csstaff 294882 Nov 12 16:01 hbp_logo.jpg
-rw-r--r--     1 uptona csstaff    678 Nov 18 18:29 slurm-18178790.out
uptona@daint104:~> vi slurm-18178790.out
```

Monitoring Jobs

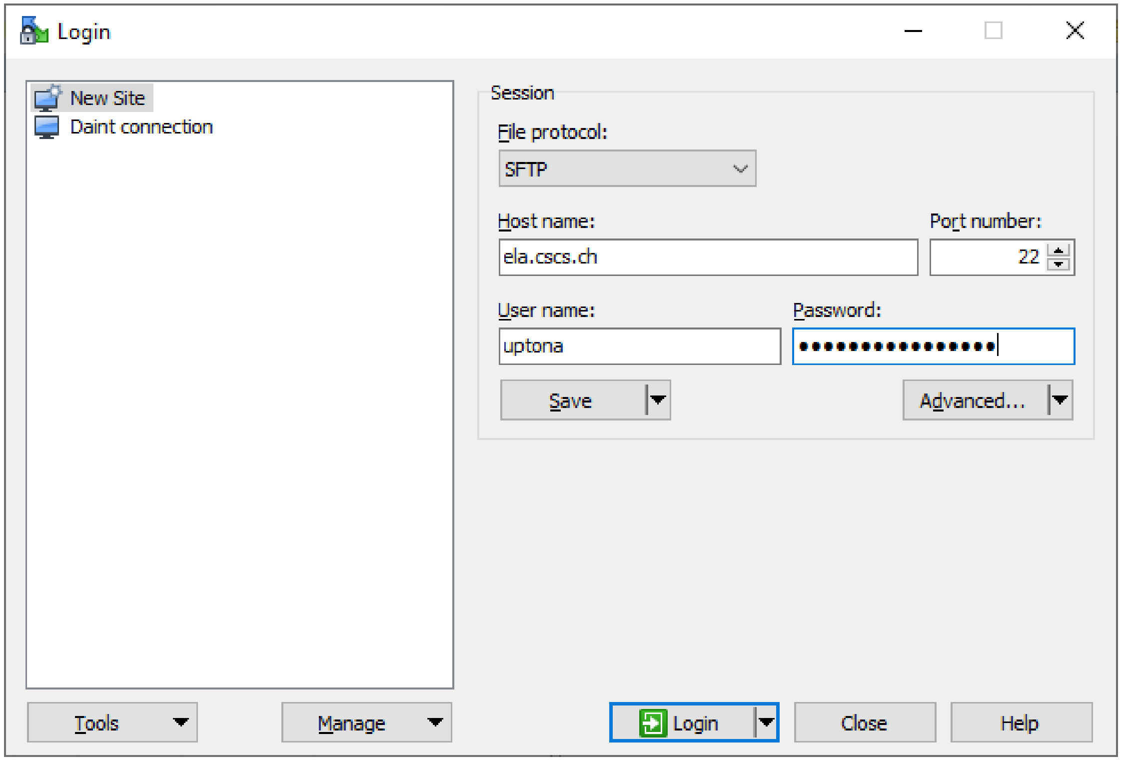
- We can monitor submitted jobs in two ways
- 1st is via the squeue –u <username> command
- More convenient however is via the email notification setup in the batch script, submit the job and we are then sent an email when the job begins and another when it has finished

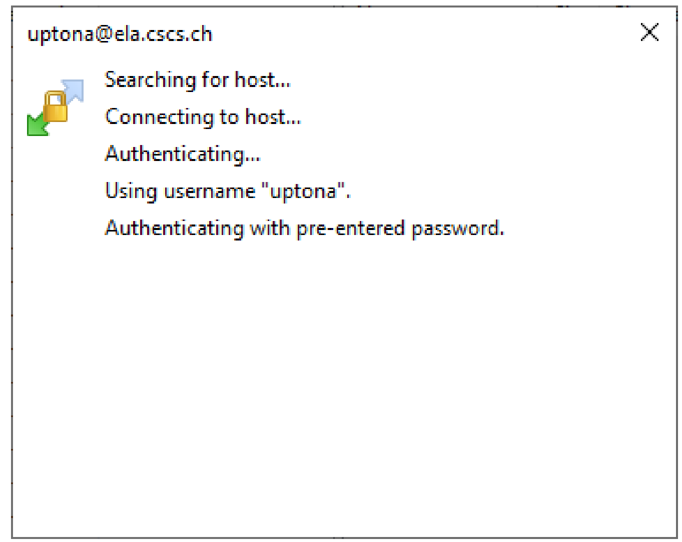




Graphical tools for moving data

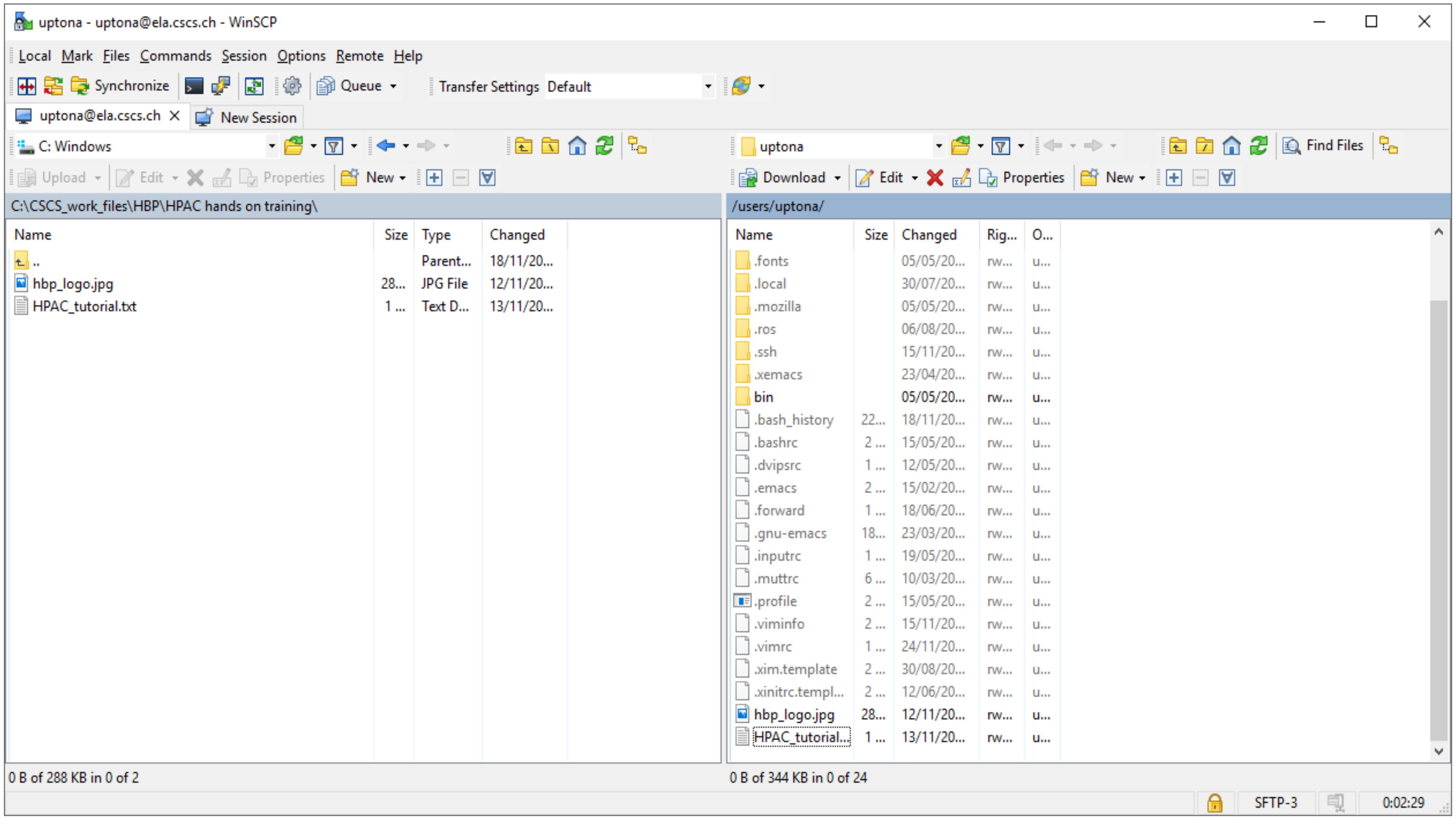
- Also possible to use graphical tools such as WinSCP to move data from local system to Ela/Piz Daint





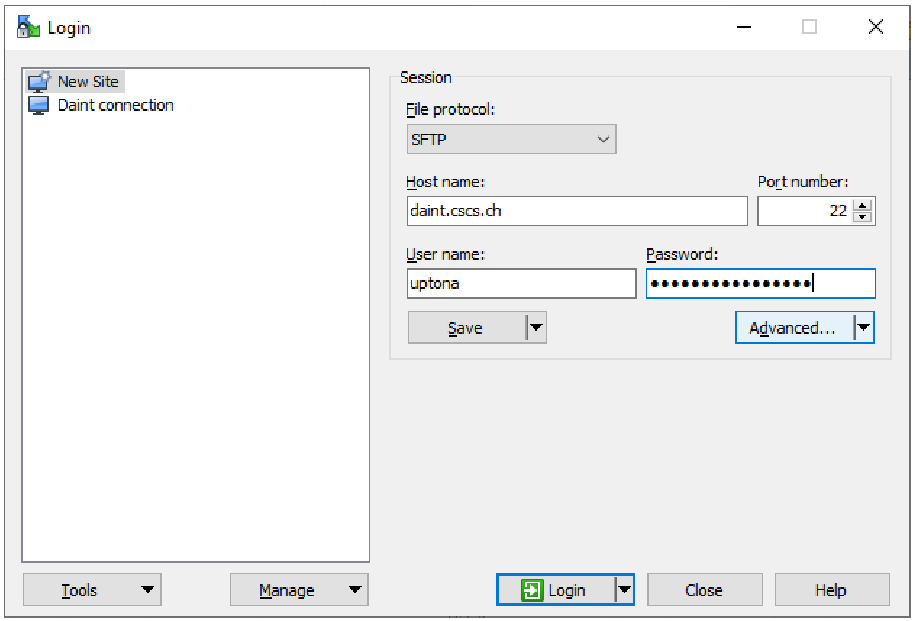
Graphical tools for moving data

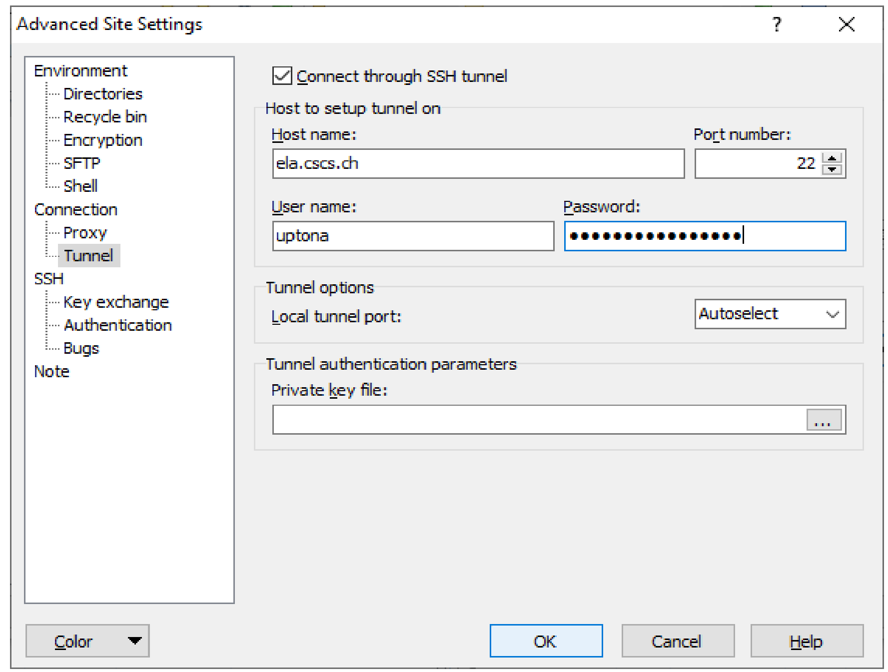
- After entering host name and password, we can now see data in the $HOME directory in the login node



Graphical tools for moving data

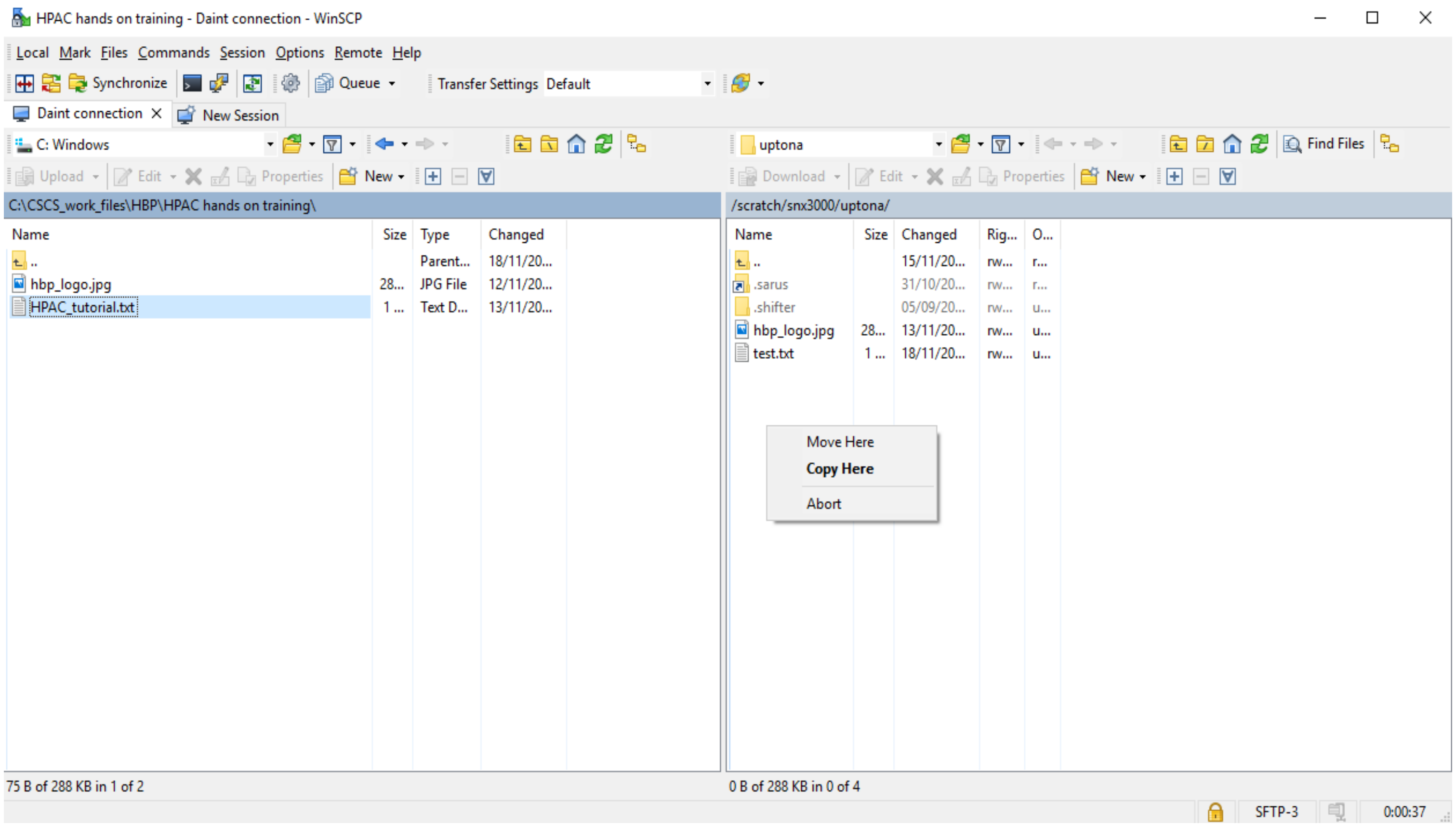
- Can also setup ssh tunnel to also connect to $SCRATCH directory on Piz Daint





Graphical tools for moving data

- Now able to see files in Piz Daint $SCRATCH directory and drag/drop to and from local file system



Sarus hands-on

- Load Sarus and pull image

```
OpenSSH SSH client
uptona@daint102:~> module load sarus
uptona@daint102:~> module unload xalt
uptona@daint102:~> srun -C gpu -N1 -t5 -p debug sarus pull ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04
srun: job 18277343 queued and waiting for resources
srun: job 18277343 has been allocated resources
# image            : index.docker.io/ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04
# cache directory  : "/scratch/snx3000/uptona/.sarus/cache"
# temp directory   : "/tmp"
# images directory : "/scratch/snx3000/uptona/.sarus/images"
> save image layers ...
> pulling        : sha256:cfef435b7966bf9a502bccb37d61cda644411b0a4907188e240d0ee47288374c
> pulling        : sha256:95e67ad685cda25141b83f50847806a0ae064ea2c0592366109f2732960fb261
> pulling        : sha256:5bd89e778a00c521f1c123a64449e4db6244ec1ecbd05cedb532a906240a73e1
> pulling        : sha256:4001a1209541c37465e524db0b9bb20744ceb319e8303ebec3259fc8317e2dec
> pulling        : sha256:6b98dfc1607190243b0938e62c5ba2b7daedf2c56d7825dfb835208344705641
> pulling        : sha256:97f170c87c6f10548068b35cbe9bd00da1278b92f700a5e66c6a16ec04ba456b
> pulling        : sha256:6319fc68c576d6bd3e469b0ae31e9a010bc9b71ed286cf4e632424d82dca70d8
> pulling        : sha256:b24603670dc3e91d00439b5d56701884e4fc313877ef3940183a7e8e25fa5f1b
> completed      : sha256:4001a1209541c37465e524db0b9bb20744ceb319e8303ebec3259fc8317e2dec
> completed      : sha256:5bd89e778a00c521f1c123a64449e4db6244ec1ecbd05cedb532a906240a73e1
> completed      : sha256:97f170c87c6f10548068b35cbe9bd00da1278b92f700a5e66c6a16ec04ba456b
> completed      : sha256:b24603670dc3e91d00439b5d56701884e4fc313877ef3940183a7e8e25fa5f1b
> completed      : sha256:6319fc68c576d6bd3e469b0ae31e9a010bc9b71ed286cf4e632424d82dca70d8
> completed      : sha256:6b98dfc1607190243b0938e62c5ba2b7daedf2c56d7825dfb835208344705641
> completed      : sha256:95e67ad685cda25141b83f50847806a0ae064ea2c0592366109f2732960fb261
> completed      : sha256:cfef435b7966bf9a502bccb37d61cda644411b0a4907188e240d0ee47288374c
> expanding image layers ...
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:6b98dfc1607190243b0938e62c5ba2b7daedf2c56d7825dfb835208344705641.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:4001a1209541c37465e524db0b9bb20744ceb319e8303ebec3259fc8317e2dec.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:6319fc68c576d6bd3e469b0ae31e9a010bc9b71ed286cf4e632424d82dca70d8.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:b24603670dc3e91d00439b5d56701884e4fc313877ef3940183a7e8e25fa5f1b.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:97f170c87c6f10548068b35cbe9bd00da1278b92f700a5e66c6a16ec04ba456b.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:cfef435b7966bf9a502bccb37d61cda644411b0a4907188e240d0ee47288374c.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:95e67ad685cda25141b83f50847806a0ae064ea2c0592366109f2732960fb261.tar"
> extracting     : "/scratch/snx3000/uptona/.sarus/cache/sha256:5bd89e778a00c521f1c123a64449e4db6244ec1ecbd05cedb532a906240a73e1.tar"
> make squashfs image: "/scratch/snx3000/uptona/.sarus/images/index.docker.io/ethcscs/osu-mb/5.3.2-mpich3.1.4-ubuntu18.04.squashfs"
```

Run container

- Run latency test in container with native MPI support (lower is better)

```
uptona@daint102:~> srun -C gpu -N2 -t2 -p debug sarus run --mpi ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04 ./osu_latency
srun: job 18277447 queued and waiting for resources
srun: job 18277447 has been allocated resources
# OSU MPI Latency Test v5.3.2
# Size          Latency (us)
0                       1.14
1                       1.10
2                       1.09
4                       1.08
8                       1.09
16                      1.11
32                      1.09
64                      1.09
128                     1.11
256                     1.12
512                     1.15
1024                    1.37
2048                    1.67
4096                    2.23
8192                    4.17
16384                   4.99
32768                   6.72
65536                  10.03
131072                 16.72
262144                 29.96
524288                 56.62
1048576               109.57
2097152               218.88
4194304               432.07
```

Run container

- Run latency test in container with MPI from the image (lower is better)

```
OpenSSH SSH client
uptona@daint102:~> srun -C gpu -N2 -t2 --mpi=pmi2 -p debug sarus run ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04 ./osu_latency
srun: job 18277462 queued and waiting for resources
srun: job 18277462 has been allocated resources
# OSU MPI Latency Test v5.3.2
# Size          Latency (us)
0                       6.57
1                       6.81
2                       6.81
4                       6.84
8                       6.91
16                      6.83
32                      6.89
64                      6.85
128                     6.97
256                     7.05
512                     7.07
1024                    9.48
2048                   10.34
4096                   10.91
8192                   11.73
16384                  13.41
32768                  16.61
65536                  27.87
131072                 54.16
262144                 81.69
524288                134.96
1048576               239.49
2097152               437.99
4194304               875.63
```

Run container

- Run bandwidth test in container with native MPI support (higher is better)

OpenSSH SSH client

```
uptona@daint102:~> srun -C gpu -N2 -t2 -p debug sarus run --mpi ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04 ./osu_bw
srun: job 18277705 queued and waiting for resources
srun: job 18277705 has been allocated resources
# OSU MPI Bandwidth Test v5.3.2
# Size      Bandwidth (MB/s)
1                       1.54
2                       3.16
4                       6.47
8                      12.93
16                     26.07
32                     52.22
64                    105.35
128                   210.54
256                   412.90
512                   801.95
1024                 1205.34
2048                 1860.49
4096                 2586.78
8192                 6249.71
16384                8720.23
32768                9260.99
65536                9601.41
131072               9773.98
262144               9853.84
524288               9916.61
1048576              9935.88
2097152              9952.86
4194304              9912.66
```

Run container

- Run bandwidth test in container with MPI from the image (higher is better)

OpenSSH SSH client

```
uptona@daint102:~> srun -C gpu -N2 -t2 --mpi=pmi2 -p debug sarus run ethcscs/osu-mb:5.3.2-mpich3.1.4-ubuntu18.04 ./osu_bw
srun: job 18277708 queued and waiting for resources
srun: job 18277708 has been allocated resources
# OSU MPI Bandwidth Test v5.3.2
# Size      Bandwidth (MB/s)
1                       0.43
2                       0.86
4                       1.72
8                       3.43
16                      6.87
32                     13.54
64                     27.15
128                    54.59
256                   109.78
512                   159.73
1024                  179.77
2048                  353.46
4096                  679.09
8192                 1372.05
16384                2459.69
32768                3192.02
65536                3917.08
131072               4006.56
262144               4418.20
524288               4400.59
1048576              4783.34
2097152              4819.47
4194304              4852.71
```

How to get Help or More Information

General Contact for HPAC Platform:

- HPAC Platform: https://collab.humanbrainproject.eu/#/collab/264/nav/2378

How to apply for resources:

- Send your proposals to: icei-coord@fz-juelich.de

Getting help:

- Send emails to: hpac-support@humanbrainproject.eu

Co-funded by
the European Union

Human Brain Project

Thank You

colin@cscs.ch alex.upton@cscs.ch

www.humanbrainproject.eu @HumanBrainProj Human Brain Project