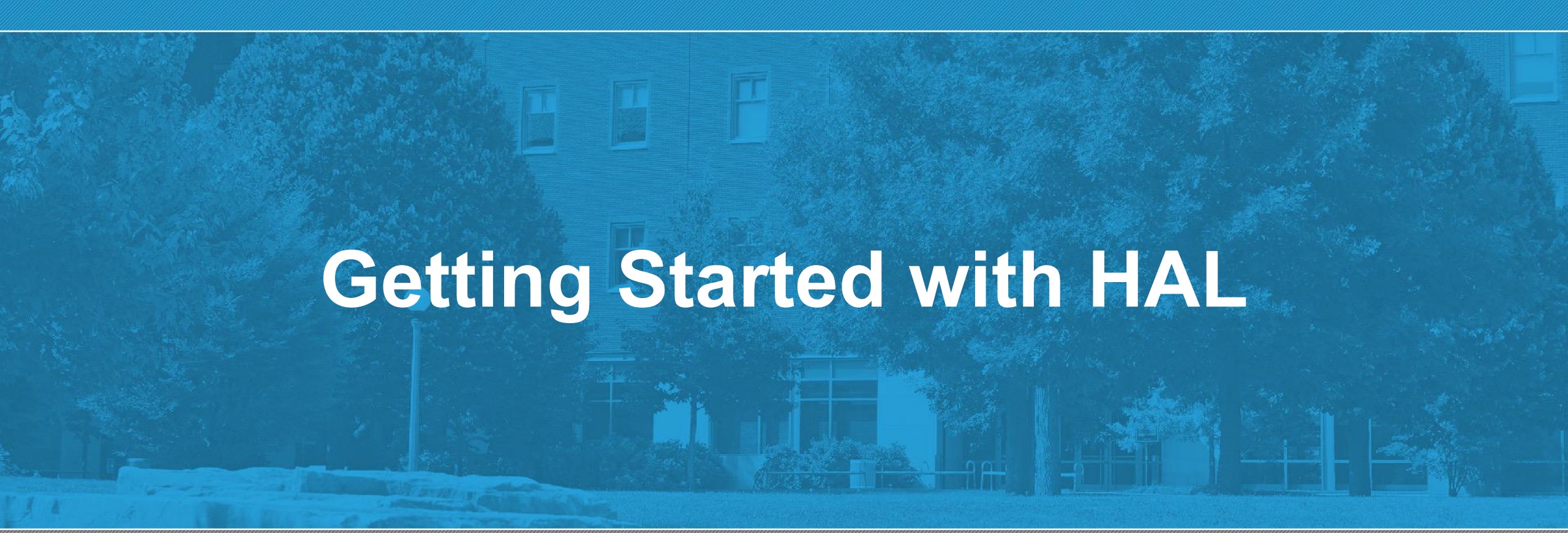


Getting Started with HAL

Dawei Mu

2022.02.02

ILLINOIS
NCSA | National Center for Supercomputing Applications

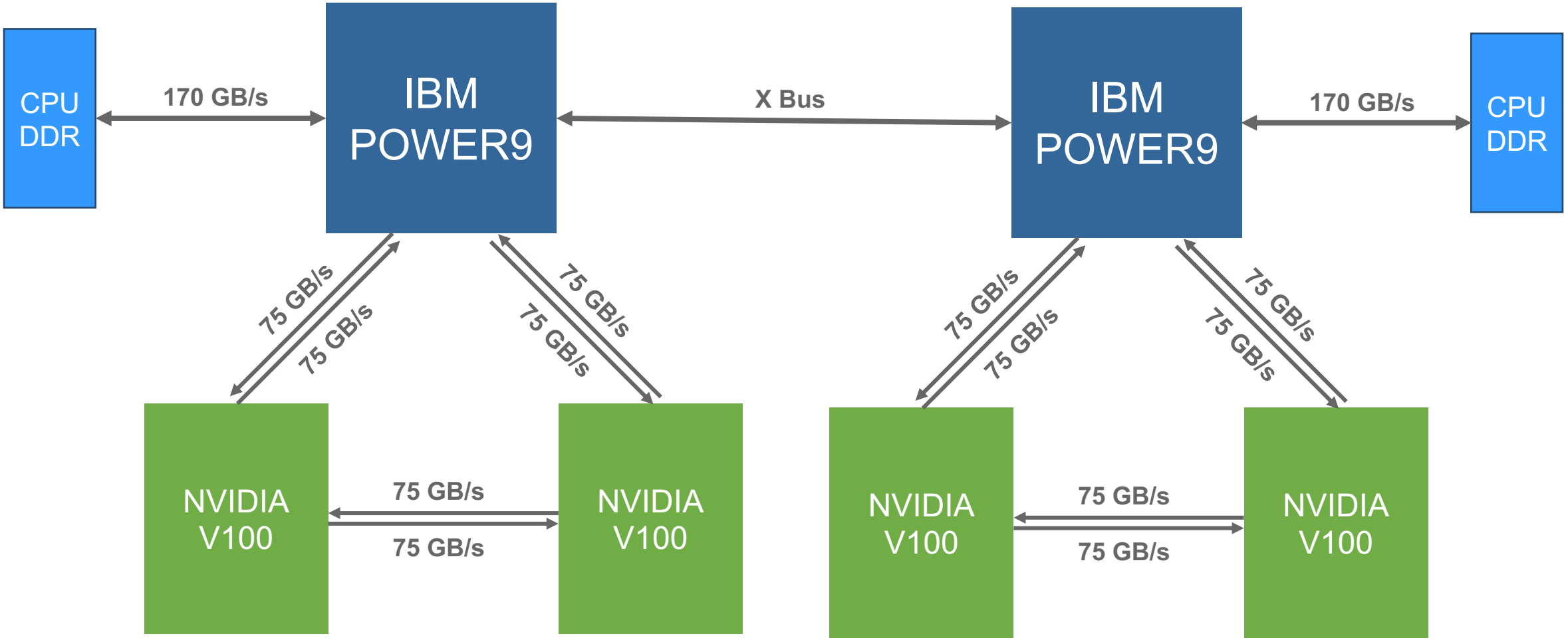
HAL System Overview

- **NSF-funded IBM cluster for Deep Learning applications**
- **16 nodes, 2560 CPU cores, 64 Nvidia GPUs**
- **336 TB of All-Flash Storage**
- **The Origin of the Name**
  - 2001: a space odyssey
  - Early concept of an Artificial Intelligence system

HAL System Overview

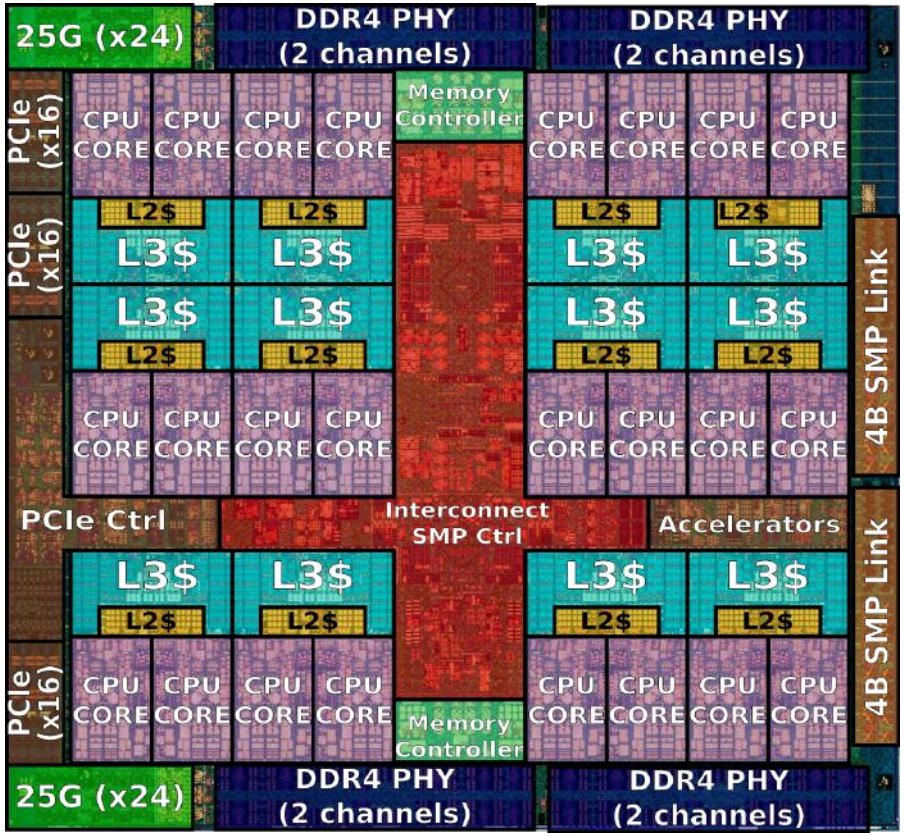
- HAL Hardware
  - 1x IBM 9006-12P LC921 Login Node
    - 2x 16-core IBM POWER9 CPU @ 2.2GHz
    - 256 GB DDR4
  - 16x IBM AC922 Nodes IBM 8335-GTH AC922 Server
    - 2x 20-core IBM POWER9 CPU @ 2.4GHz
    - 256 GB DDR4
    - 4x NVIDIA V100 GPUs
      - 5120 cores
      - 16 GB HBM 2
    - 2-Port EDR 100 Gb IB ConnectX-5 Adapter
  - 3x DDN GS400NVE Flash Arrays
    - 336 TB usable, NVME SSD-based storage
    - Spectrum Scale File System

HAL System Overview



HAL System Overview

- **IBM POWER9 CPUs**
  - 14nm finFET semiconductor
  - Stronger Thread Performance – SMT
  - POWER ISA 3.0 Architecture
  - Enhanced Cache Hierarchy
  - NVIDIA NVLink 2.0
  - I/O System – PCIe Gen4
- **2x 20 Cores with SMT4**
  - Map to OS as 160 CPUs per node



HAL System Overview

- NVIDIA V100 GPUs
  - Peak 7.8 TFLOP/s (double-precision).
  - Peak 15 TFLOP/s (single-precision).
  - SM / Cores : 80 / 5120.
  - HBM2 Memory 16 GB : 900 GB/s.
  - Config up to 128 KB L1 Cache per SM.
  - Config up to 96 KB Shared Memory per SM.
  - Constant memory 64 KB.
  - 65536 32-bit Registers per SM.
  - Clock Frequency : 1530 MHz

HAL Software Overview

- **HAL Software**
  - **OS : Red Hat Enterprise Linux (RHEL) 8.4**
  - **Compilers :**
    - **GNU 8.4.1**
    - **CUDA 11.4.48**
    - **Nvidia HPC SDK 21.5**
  - **Tools :**
    - **OpenCE 1.3.1**
    - **PowerAI 1.7.0 (Watson Machine Learning Community Edition)**
    - **OpenMPI 4.1.1**
    - **CMake 3.20.3**
    - **Singularity 3.8.0**

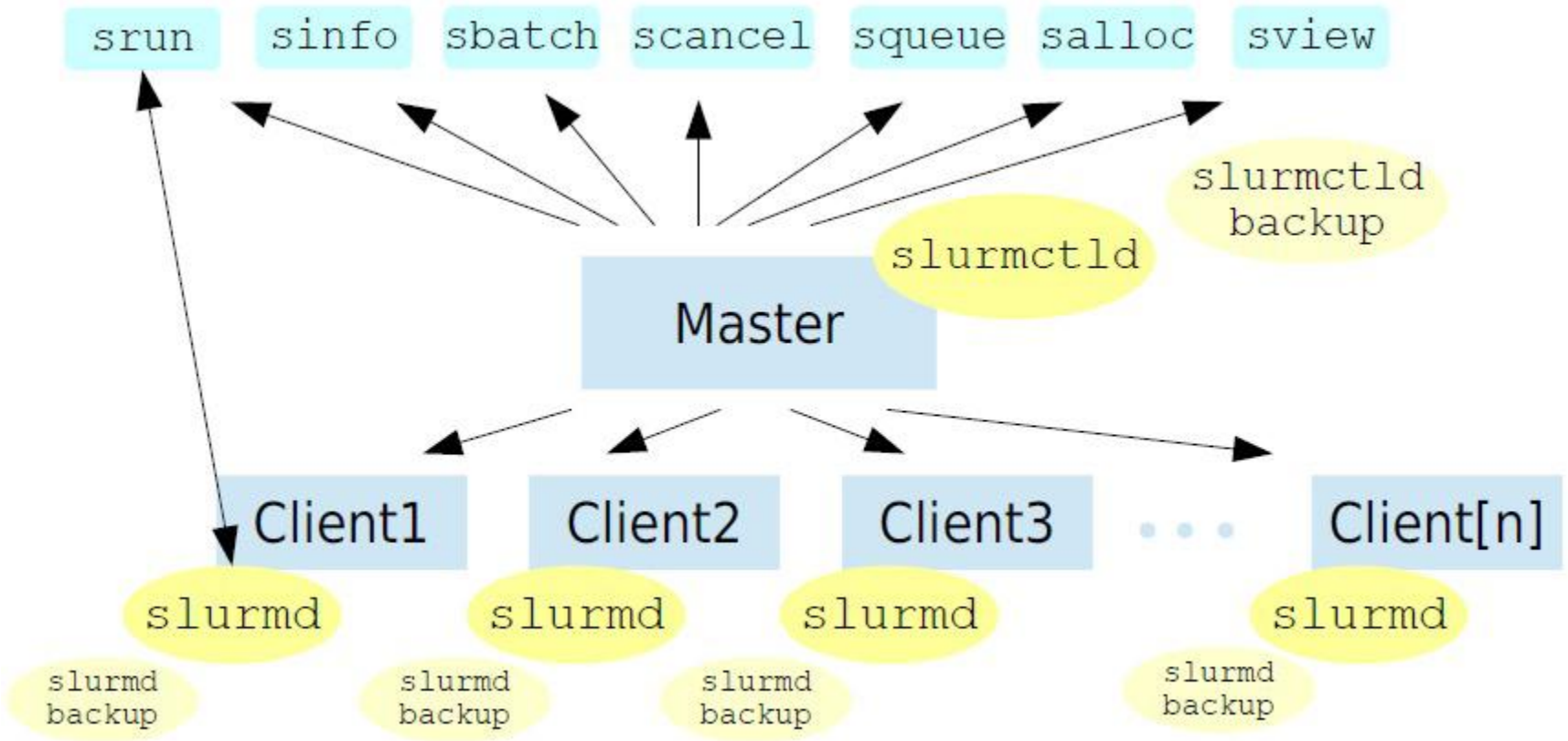
Programming Environment

- **Environment Management**
  - Environment modules are provided through Lmod, a Lua-based module system for dynamically altering environments.
- **General Usage**
  - module avail : shows a table of the currently available modules.
  - module list : shows a list of the currently loaded modules.
  - module purge : unloads all loaded modules
  - module unload : unloads module into the current environment
  - module load : loads module into the current environment

Connection

- **Login Node**
  - **1x IBM 9006-12P Login Node**
  - **1x IBM 9006-22P Login Node**
- **Connect with SSH**
  - ssh <userid>@hal.ncsa.illinois.edu
  - ssh <userid>@hal-login2.ncsa.illinois.edu
- **Connect with SFTP**
  - sftp <userid>@hal.ncsa.illinois.edu
  - sftp <userid>@hal-login2.ncsa.illinois.edu

Slurm Workload Manager



Slurm Policy on HAL

- Maximum 5 running jobs per user;
- Maximum 5 activate nodes per user;
- Maximum 16 activate GPUs per user;
- Maximum 24 hours per job.

Slurm Workload Manager

- Original Slurm command could be complex
  - srun --partition=gpu --time=24:00:00 --nodes=1 --ntasks-per-node=160 --sockets-per-node=2 --cores-per-socket=20 --threads-per-core=4 --mem-per-cpu=1200 --wait=0 --export=ALL --gres=gpu:v100:4 --pty /bin/bash

Slurm Wrapper Suite

- **Rule of Thumb**
  - Minimize the required input options.
  - Consistent with the original "slurm" run-script format.
  - Submits job to suitable partition based on the number of GPUs.

Slurm Wrapper Suite

- “swrun” Usage
  - Only 4 options
    - Partition *(required)*
    - CPUs Per GPU *(optional)*
    - Wall Time *(optional)*
    - Singularity Container *(optional)*
  - Restrictions
    - Partitions vary by GPU number (gpux1, gpux2, gpux3, …)
    - CPU Per GPU (16 <= c <= 40, default 16)
    - Wall Time (1 <= t <= 24, default 4 hours)
  - Default if selecting 1 gpu
    - gpux1(required), 16x CPUs, 19.2 GB Memory, 1x GPU, 4 Hrs

Slurm Wrapper Suite

- “swrun” Usage
  - Debug queue
    - Only need to set time to 4 hours or less on queue “gpux1/2/3/4” and “cpu”
    - Multi-nodes queue less than 4 hours still go to normal queue
  - Singularity
    - swrun -p gpux4 -s powerai -c 40 -t 24
    - “-s”: using singularity image for this job
    - “powerai”: the name of singularity image “powerai.sif”
    - export “HAL_CONTAINER_REGISTRY” to your own directory
  - Example workflow:
    - Export HAL_CONTAINER_REGISTRY=$HOME/container/pool
    - Then specify the image as above or in your batch script as
    - #SBATCH --singularity=powerai

Slurm Wrapper Suite

- Original Slurm command could be complex
  - srun --partition=gpu --time=24:00:00 --nodes=1 --ntasks-per-node=160 --sockets-per-node=2 --cores-per-socket=20 --threads-per-core=4 --mem-per-cpu=1200 --wait=0 --export=ALL --gres=gpu:v100:4 --pty /bin/bash

    <=>
  - swrun -p gpux4 -c 40 -t 24

Slurm Wrapper Suite

- “swbatch” Usage
  - Only 7 options
    - Partition *(required)*
    - CPUs Per GPU *(optional)*
    - Wall Time *(optional)*
    - Job name *(optional)*
    - Output file *(optional)*
    - Error file *(optional)*
    - Singularity Container *(optional)*
  - Restrictions
    - Partitions vary by GPU number (x1, x2, x3, …)
    - CPU Per GPU (16 <= c <= 40, default 16)
    - Wall Time (1 <= t <= 24, default 4 hours)

Slurm Wrapper Suite

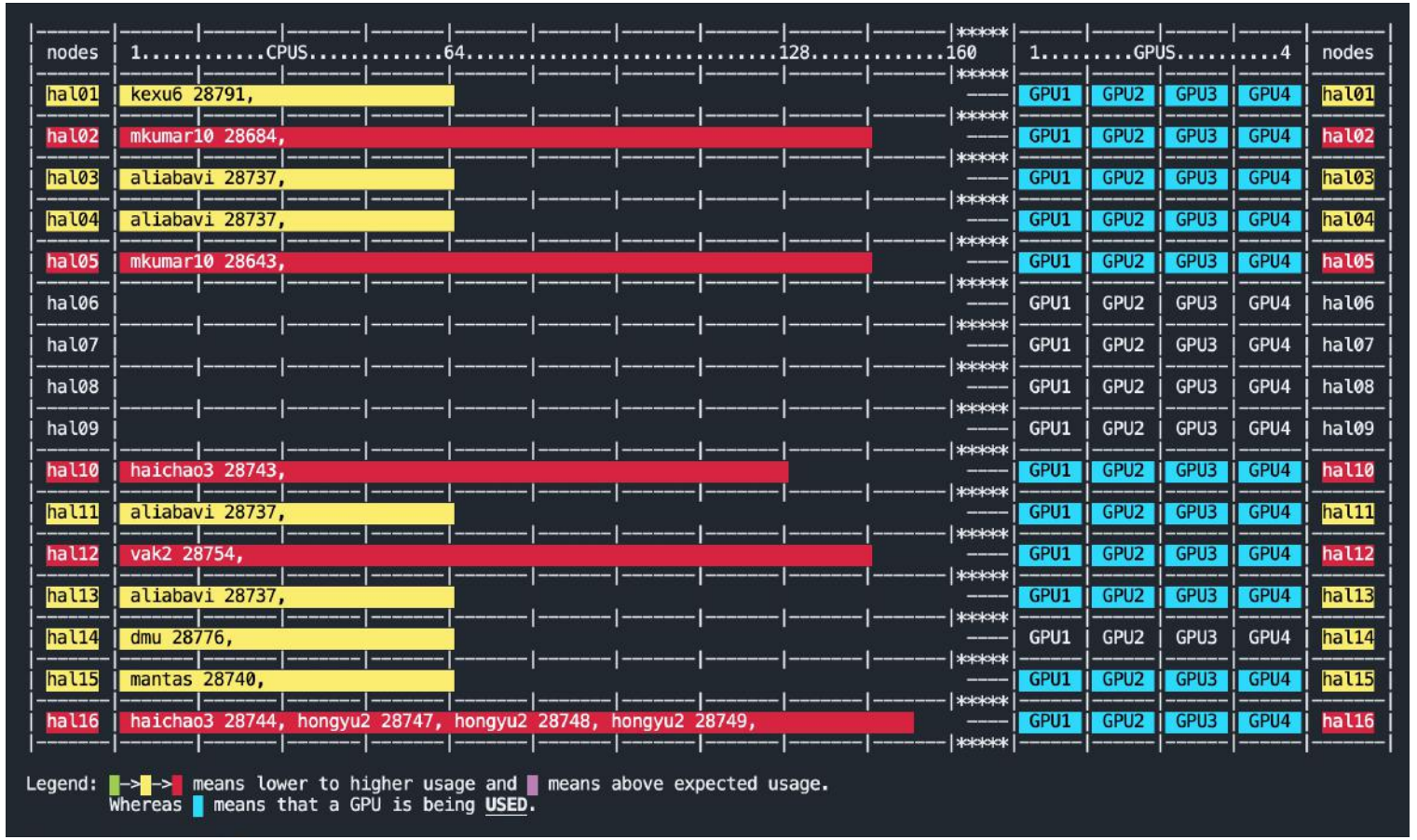
- “swbatch” example:
  - Sample run script - sample.sb vs sample.swb

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --time=4:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --sockets-per-node=1
#SBATCH --cores-per-socket=4
#SBATCH --threads-per-core=4
#SBATCH --mem-per-cpu=1200
#SBATCH --export=ALL
#SBATCH --gres=gpu:v100:1
python3 mnist_train_pytorch.py
    =>
#!/bin/bash
#SBATCH --partition=gpux1
python3 mnist_train_pytorch.py
```

  - Simply run as: sbatch sample.sb => swbatch sample.swb

Slurm Wrapper Suite

- swqueue (new): show cluster usage within terminal, color coded utilization level.



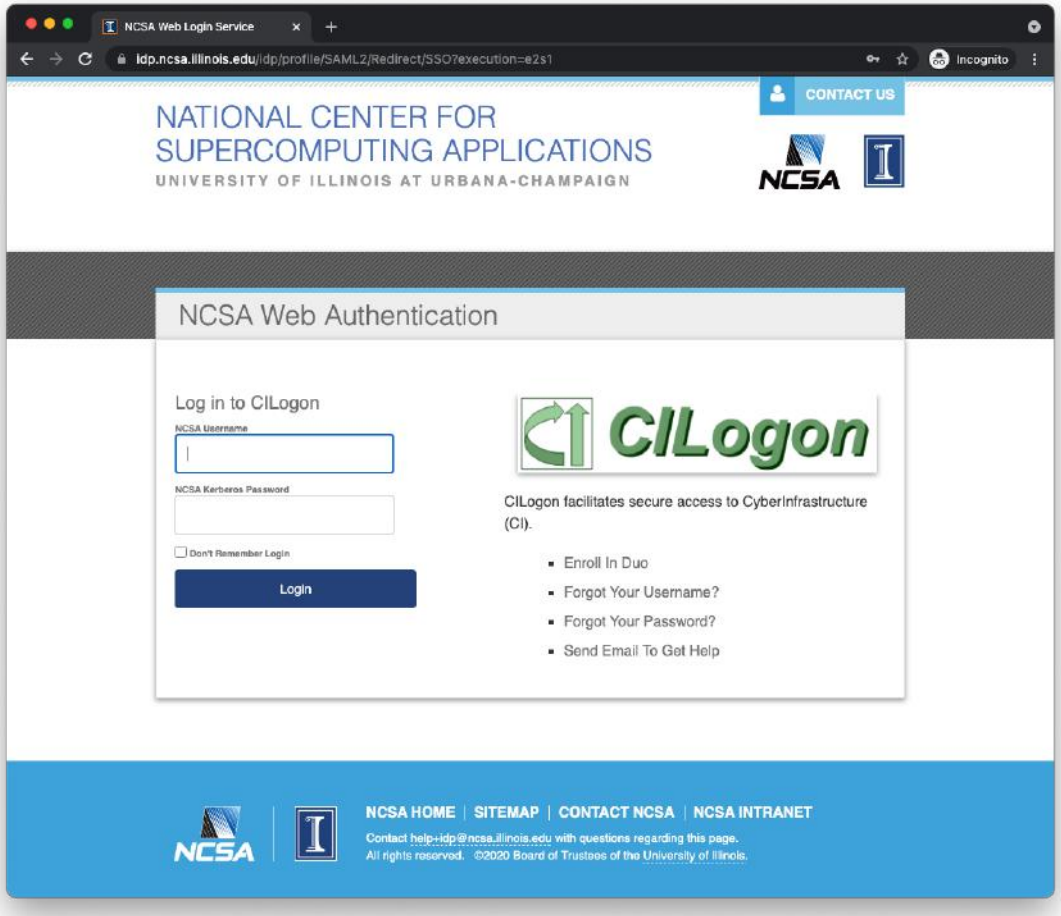
Frequently Ask Questions

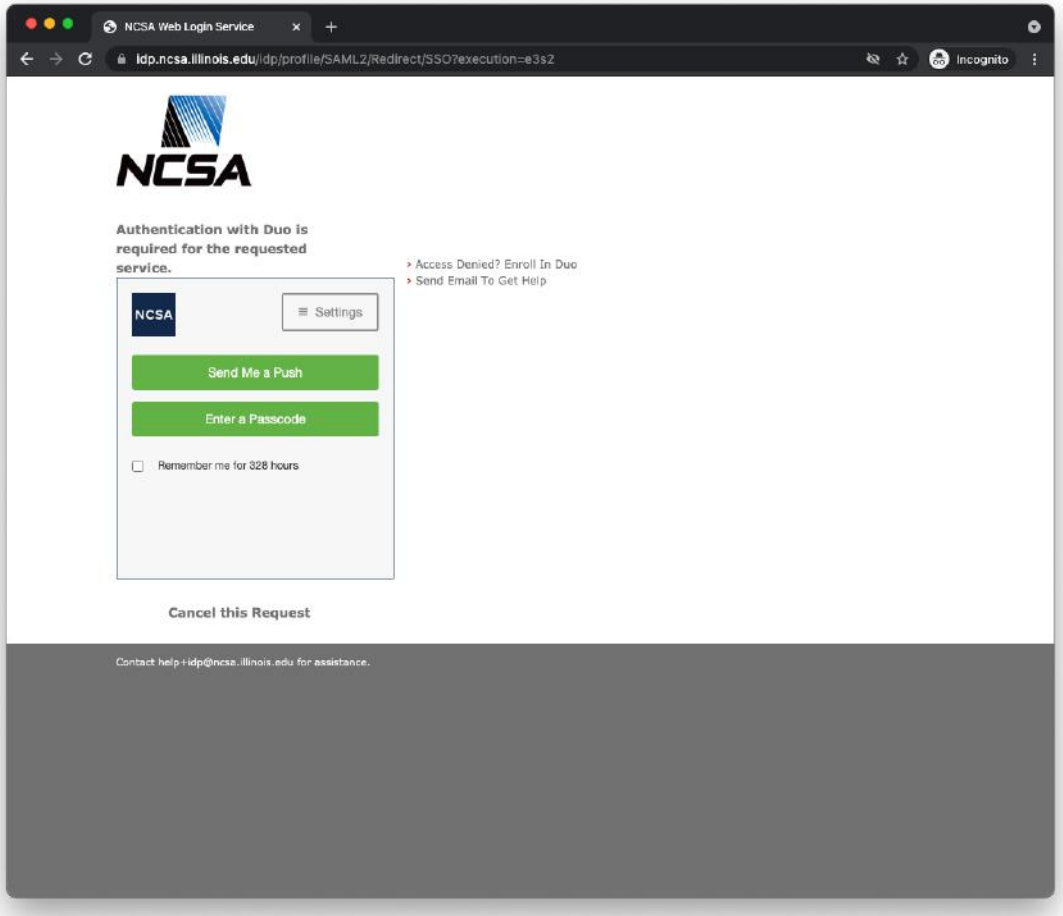
1. Why partition “gpux1” doesn’t work with “sbatch”?
   - “gpux1”, “gpux2”, … works only with slurm wrapper suite.
   - if use “sbatch”, the valid partitions include “gpu” and “cpu”.
2. Why I get “IndexError: string index out of range” error when using “swbatch”?
   - slurm wrapper suite requires some packages like pyyaml.
   - users should submit your job within the default python env.
   - check “.bashrc” to remove conda related setting.
3. Why my jobs always under queueing?
   - check jobs status with “squeue” for detailed reason.
   - check your recent usage to verify your fair share.

Open OnDemand

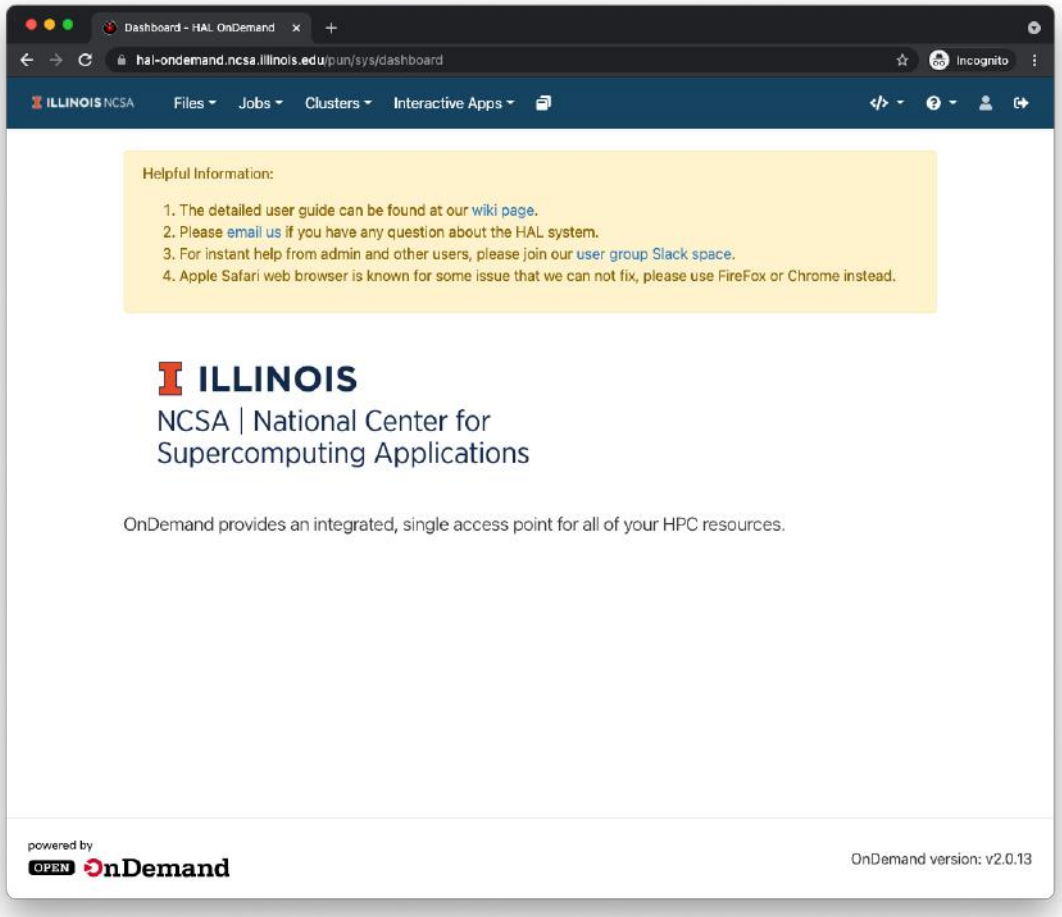
- We have implemented the Open OnDemand on HAL system as a web-based portal.
- The OOD service including
  - Manage data with Files app.
  - Submit and monitor jobs with Jobs app.
  - Access the cluster with Shell Access app under Clusters.
  - Utilize the Jupyter Notebook, TensorBoard, VS Code and H2O-AI apps under Interactive Apps.

Open OnDemand (main page)

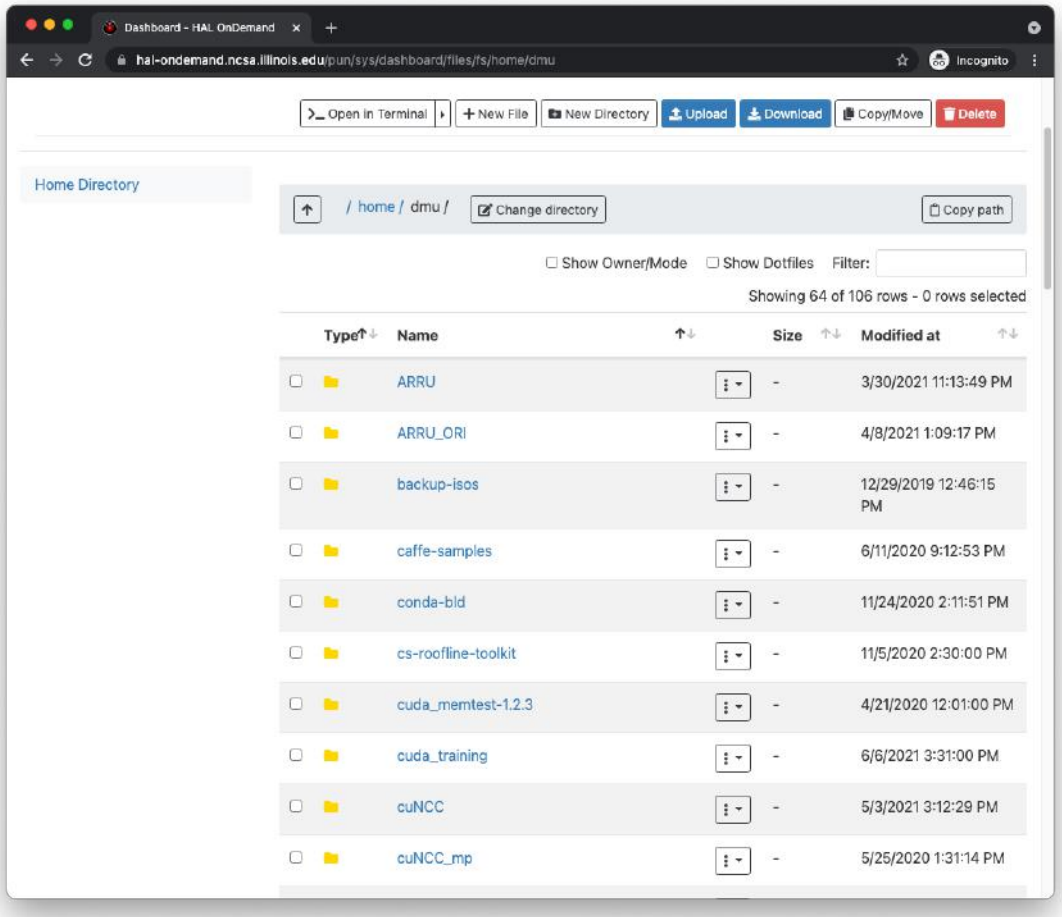




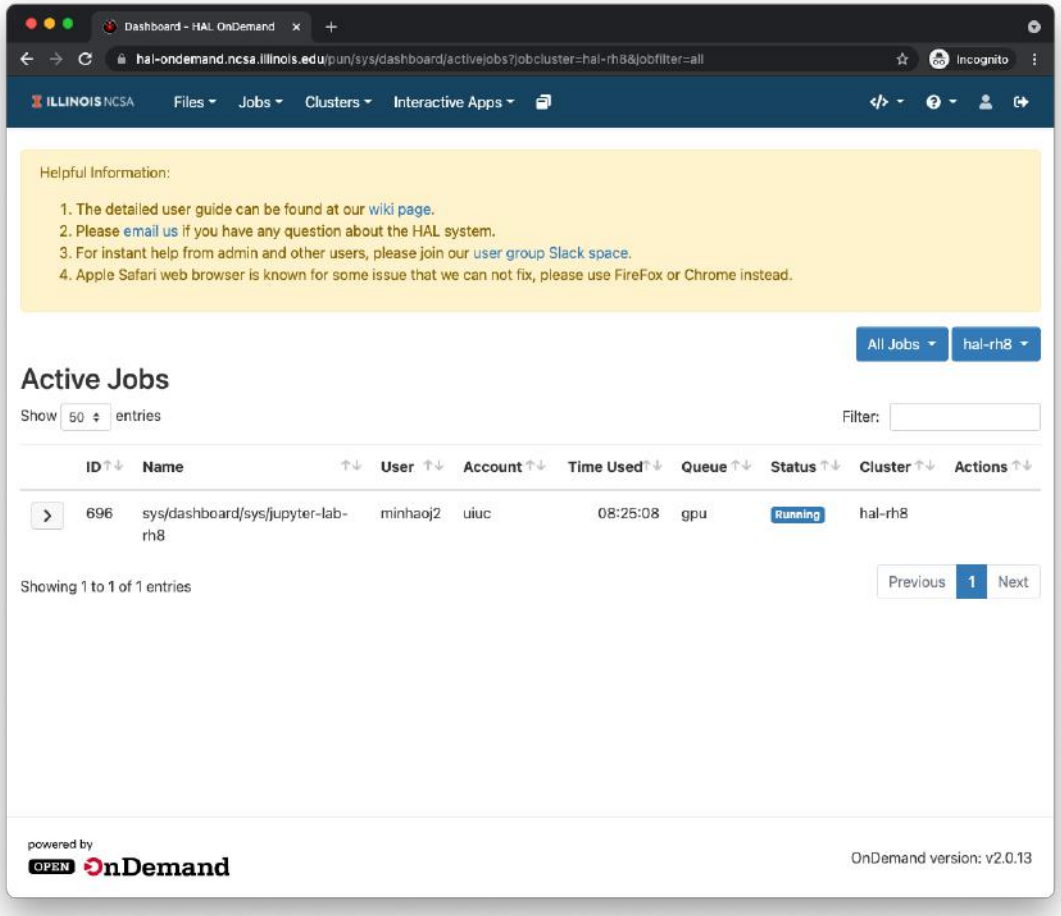
Open OnDemand (main page)

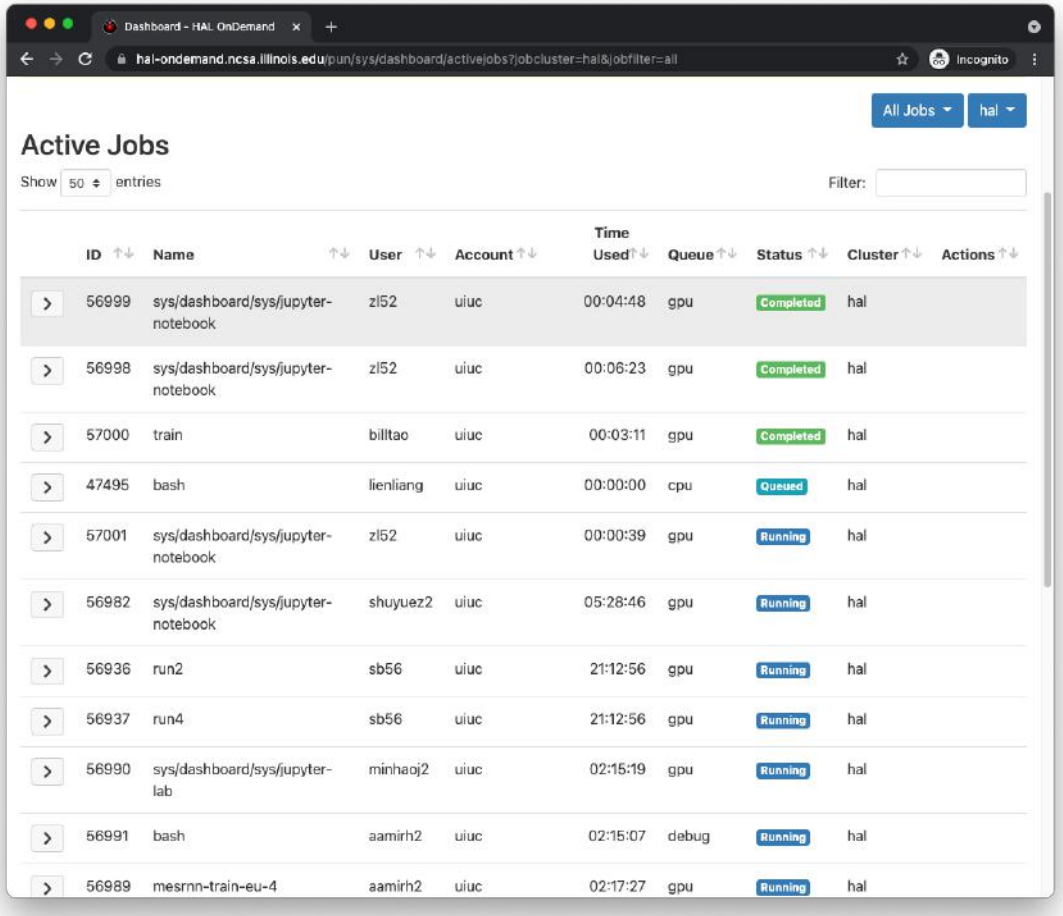


Open OnDemand (files)

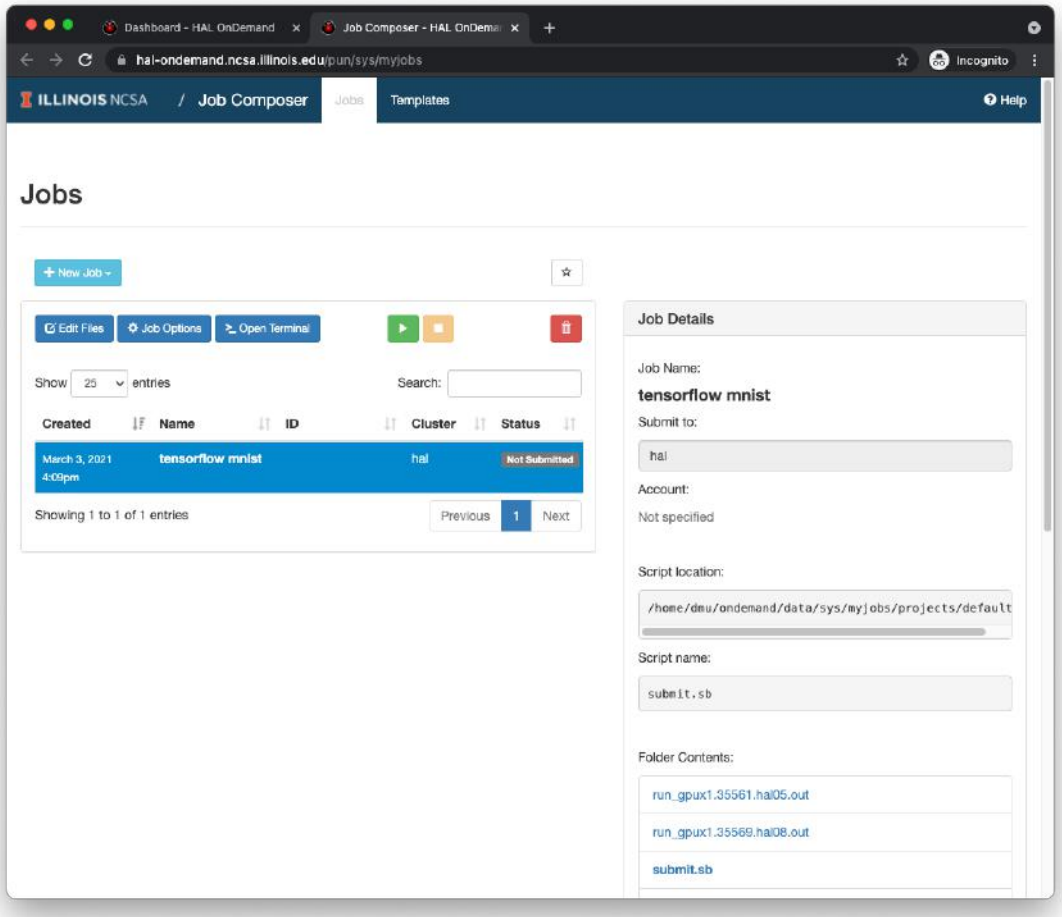


Open OnDemand (activate jobs)

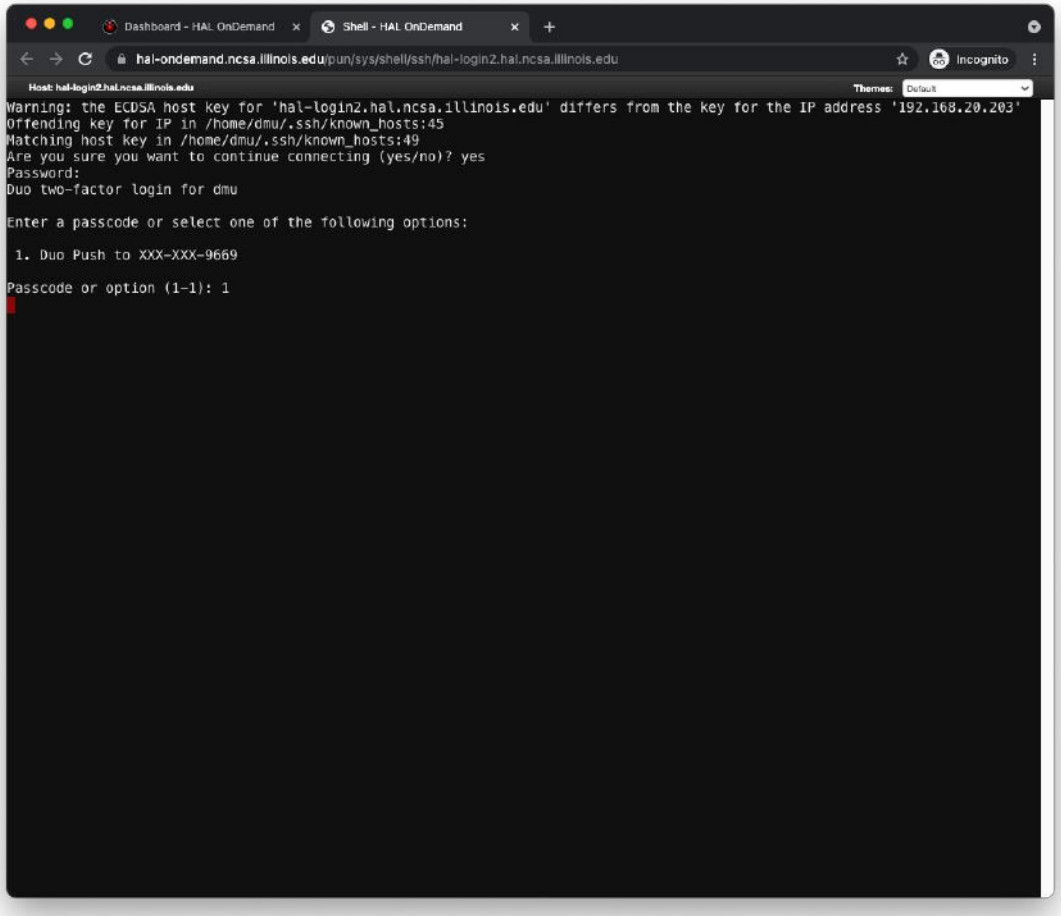


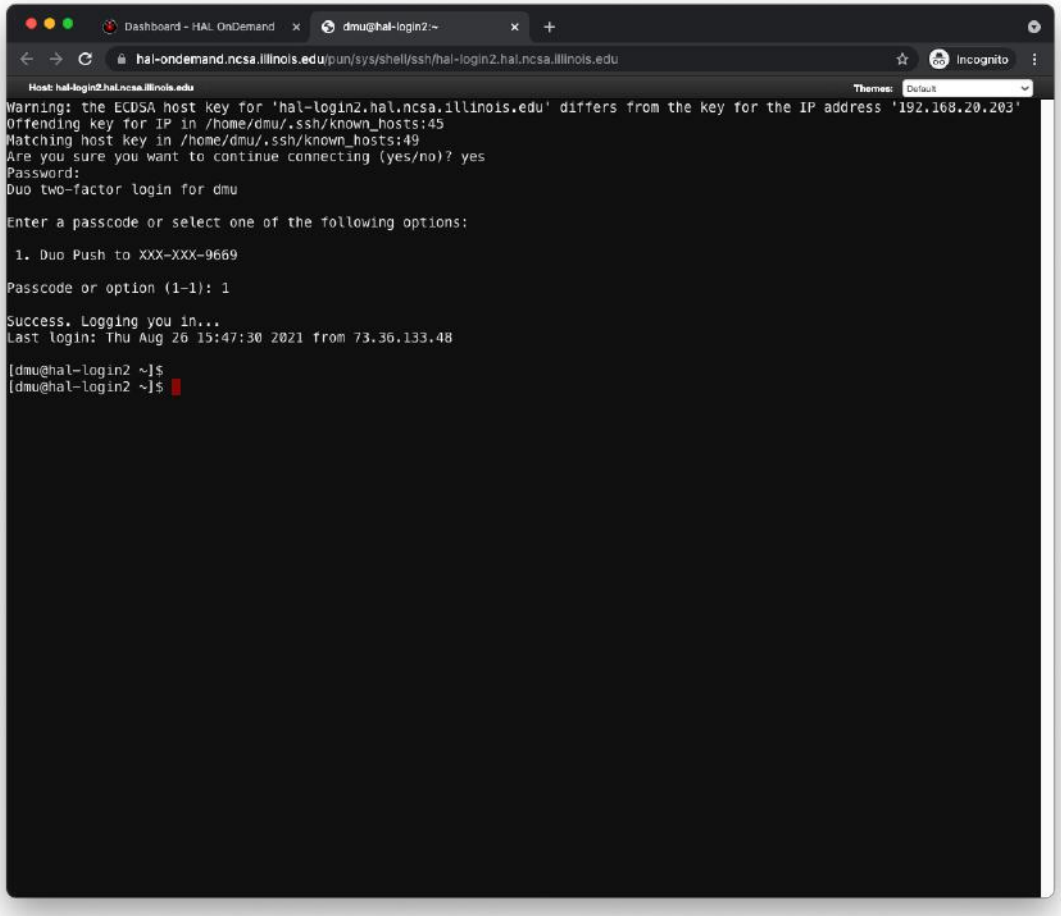


Open OnDemand (compose jobs)

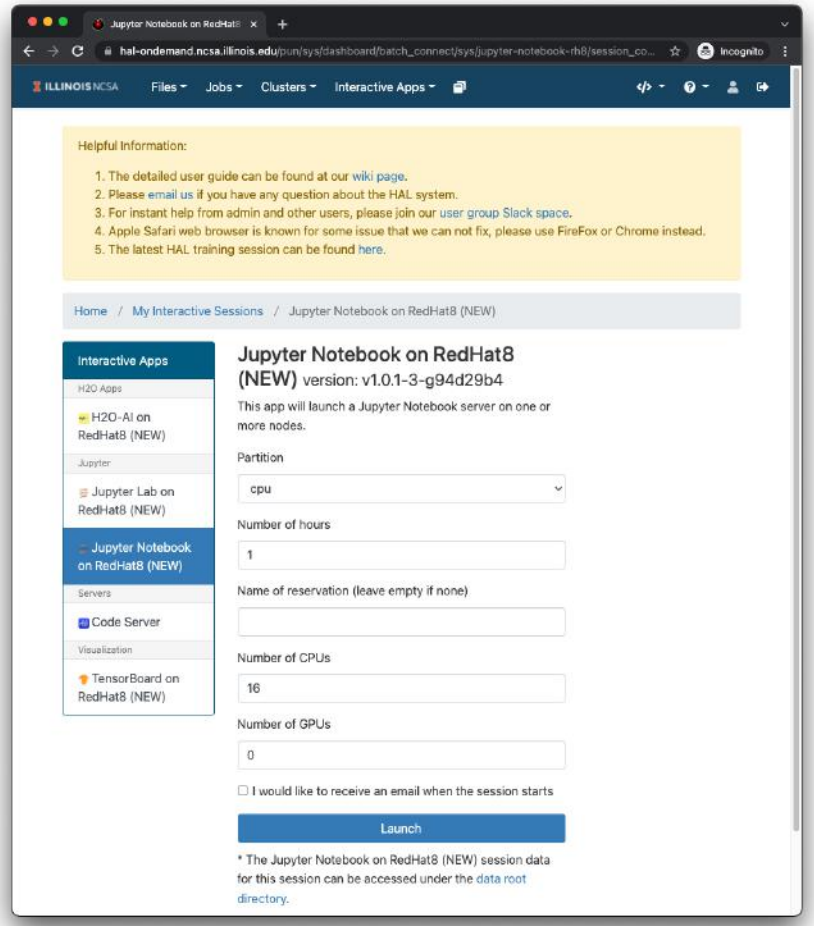


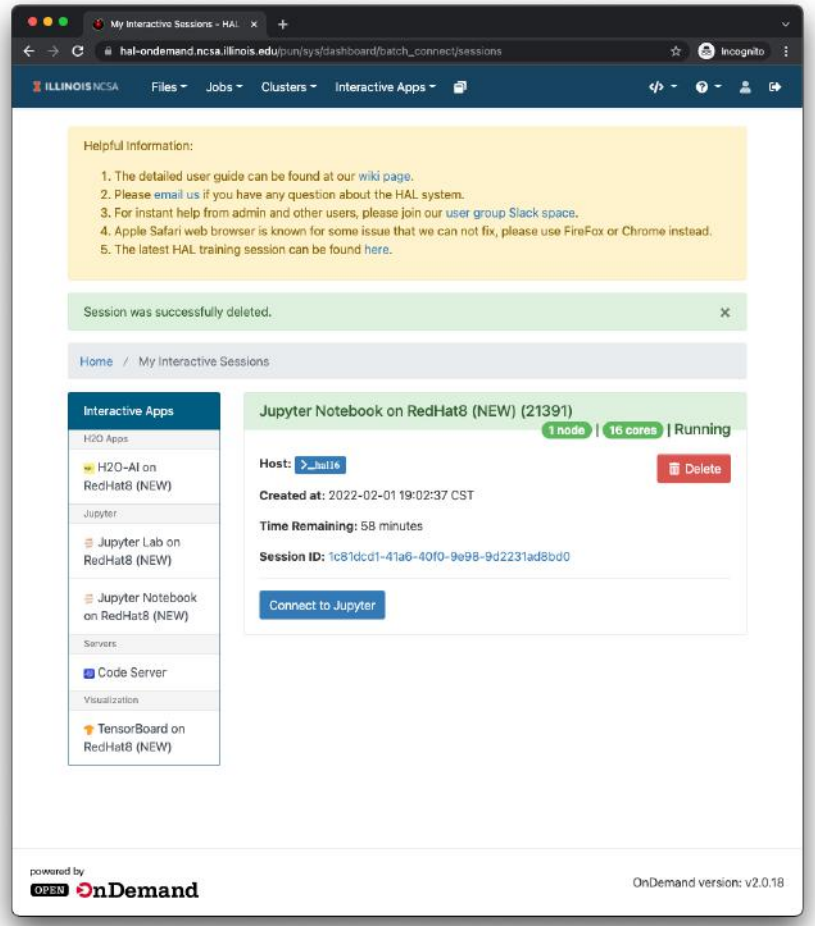
Open OnDemand (terminals)



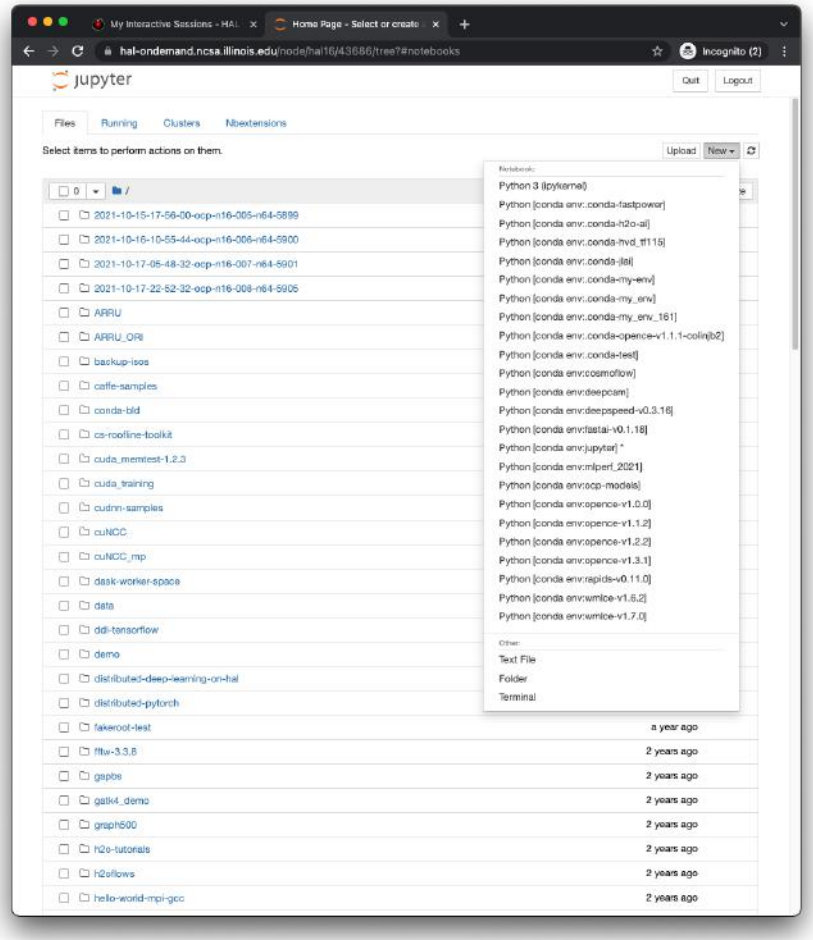


Open OnDemand (jupyter-notebook)

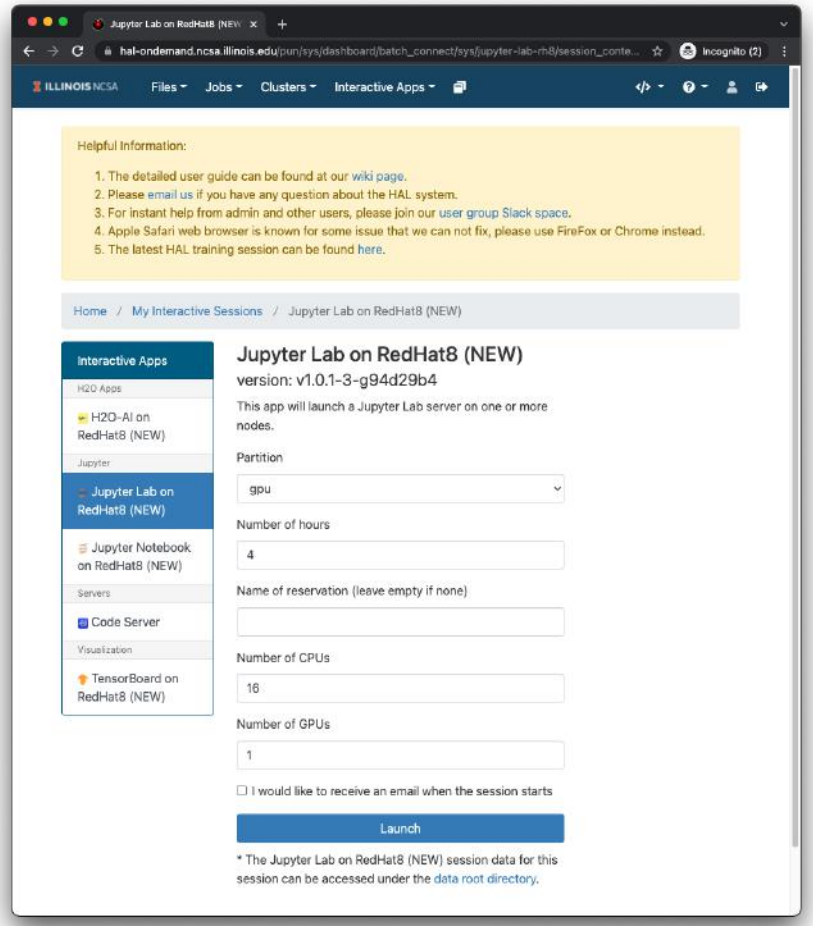


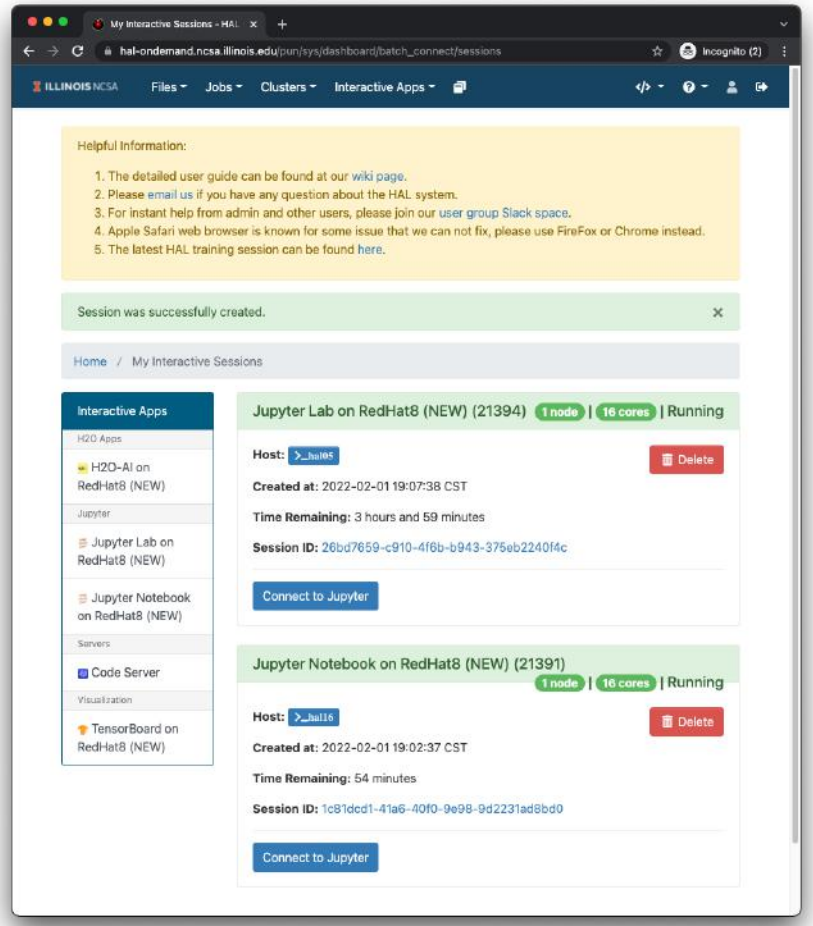


Open OnDemand (jupyter-notebook)

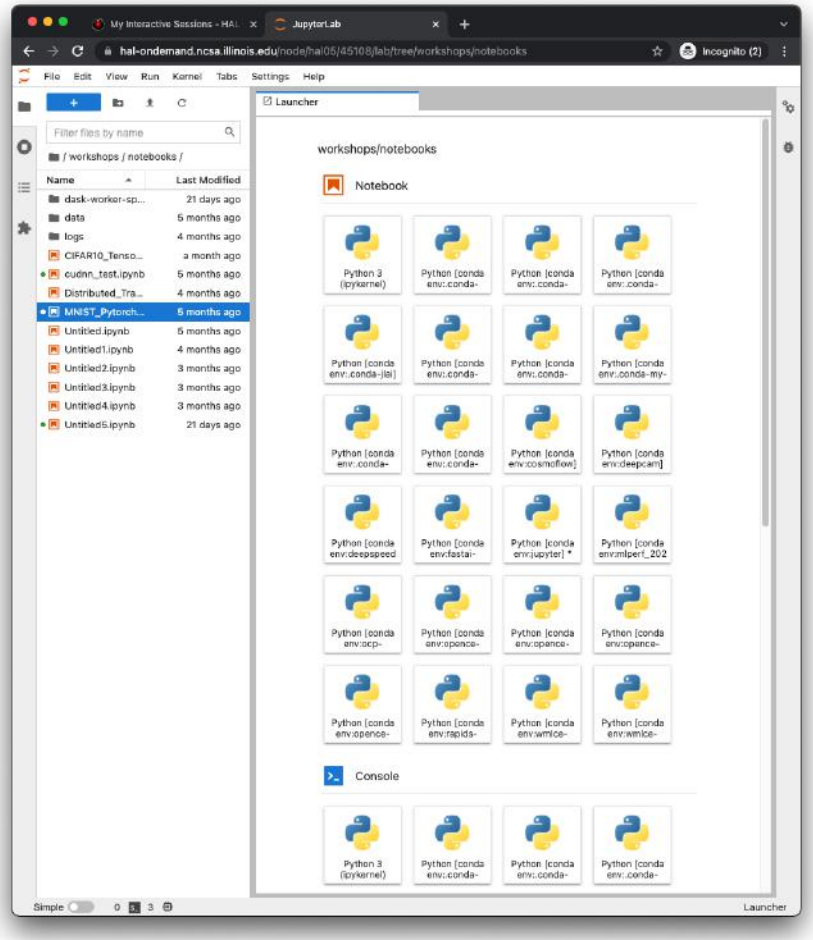


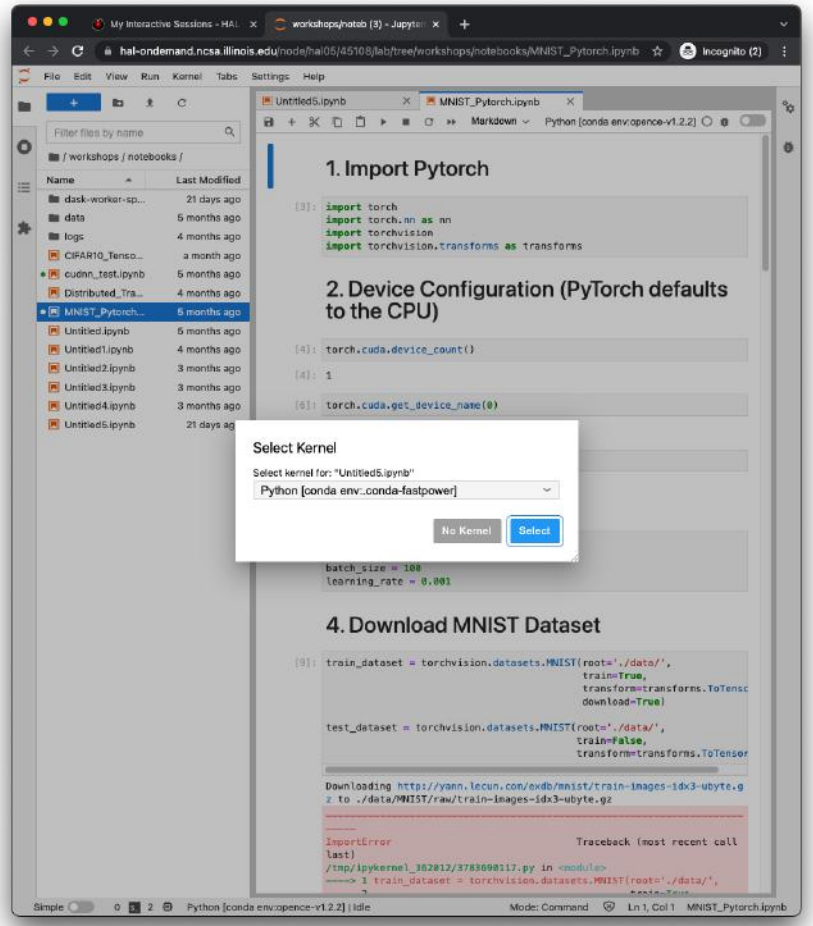
Open OnDemand (jupyter-lab)



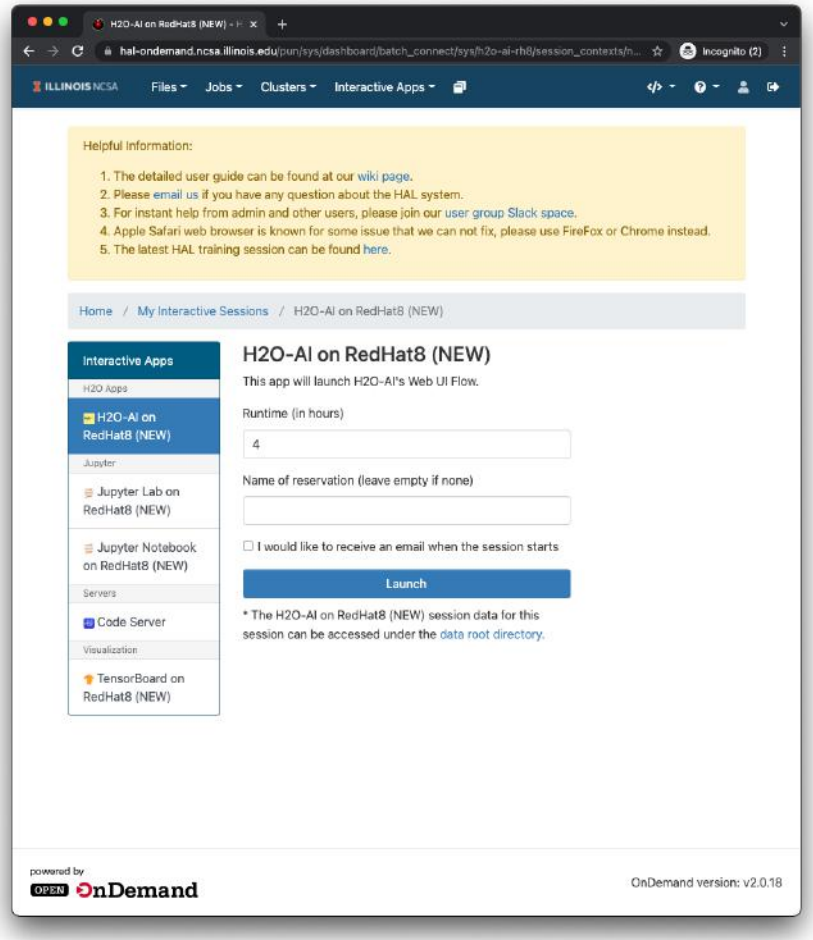


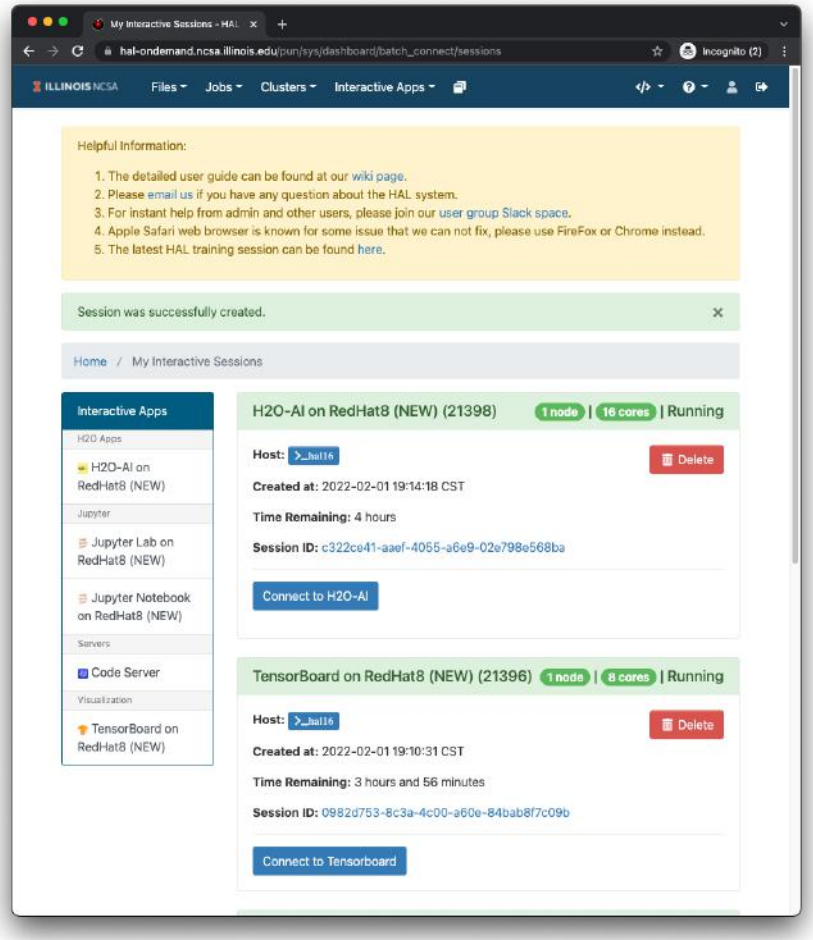
Open OnDemand (jupyter-lab)



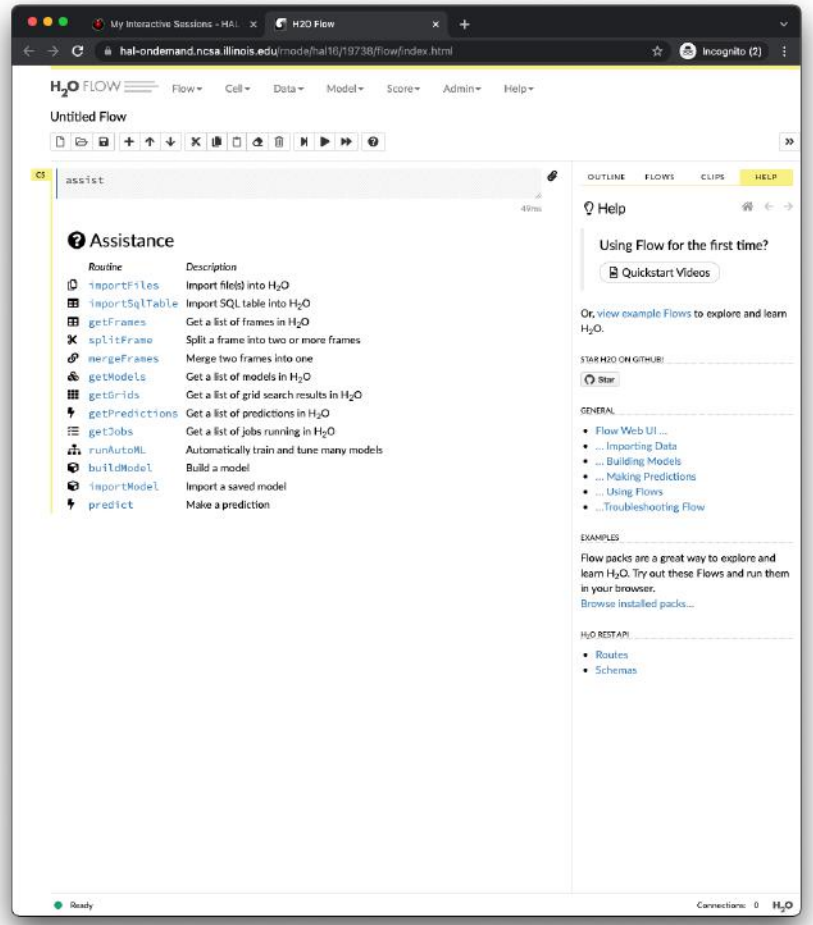


Open OnDemand (h2o-flow)

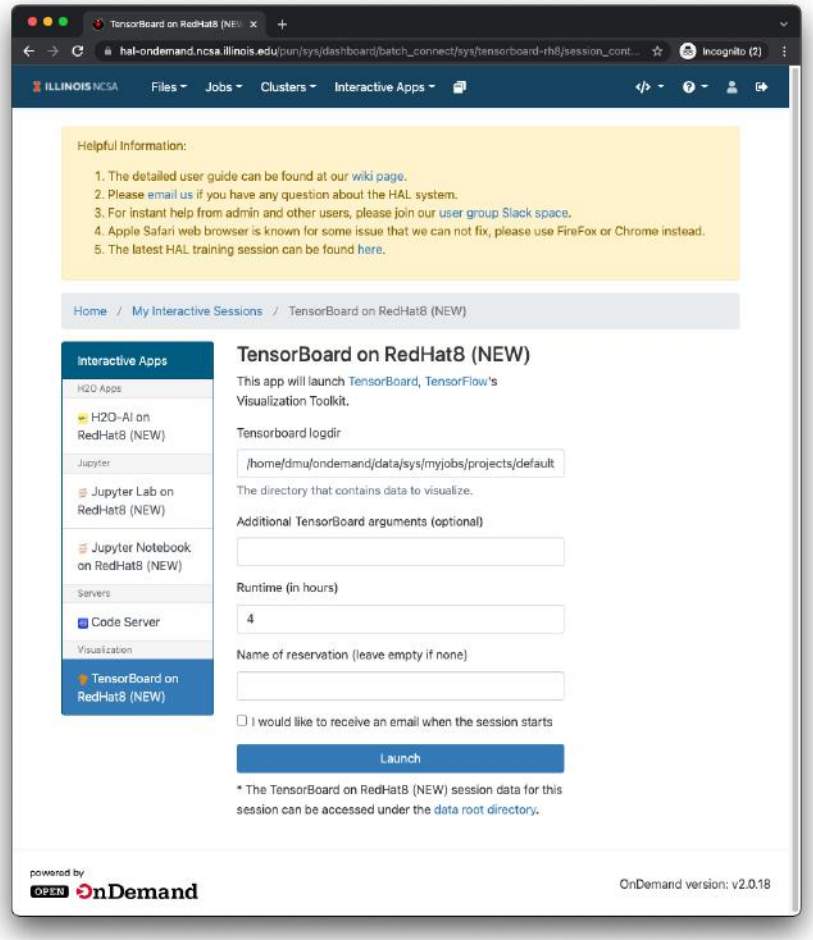


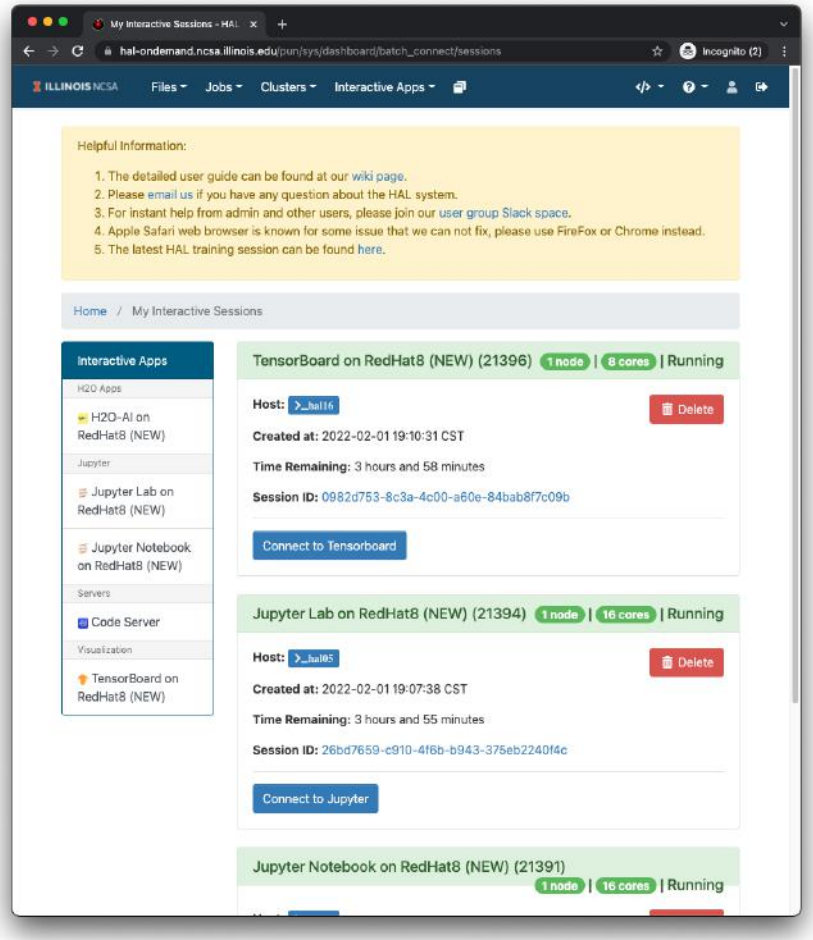


Open OnDemand (h2o-flow)

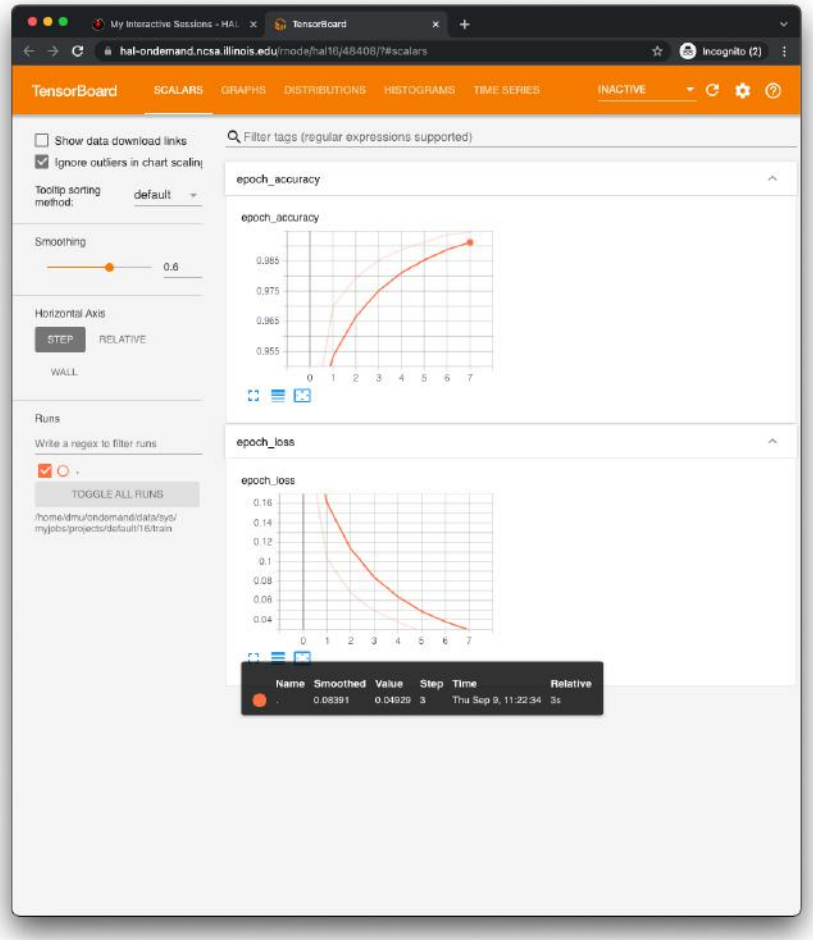


Open OnDemand (tensorboard)

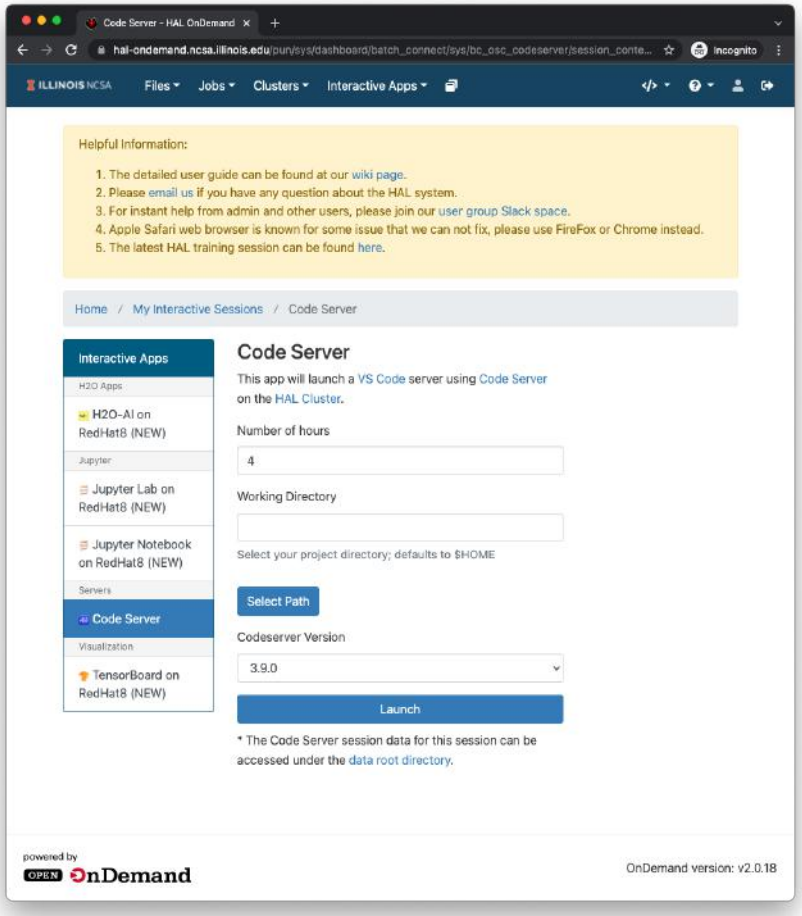


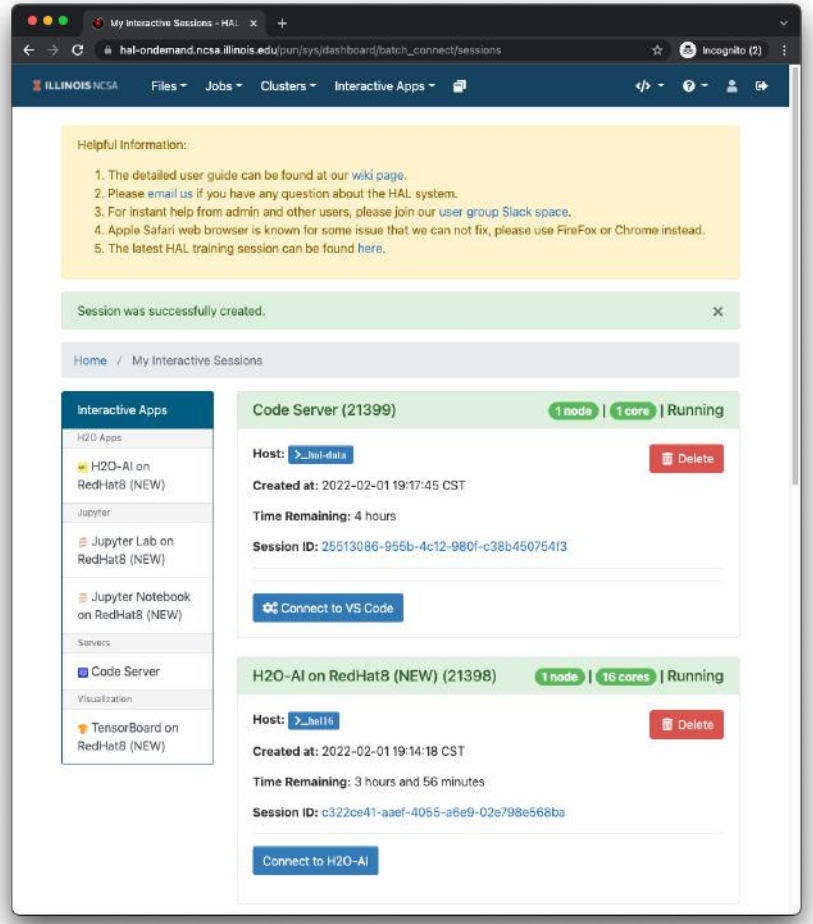


Open OnDemand (tensorboard)

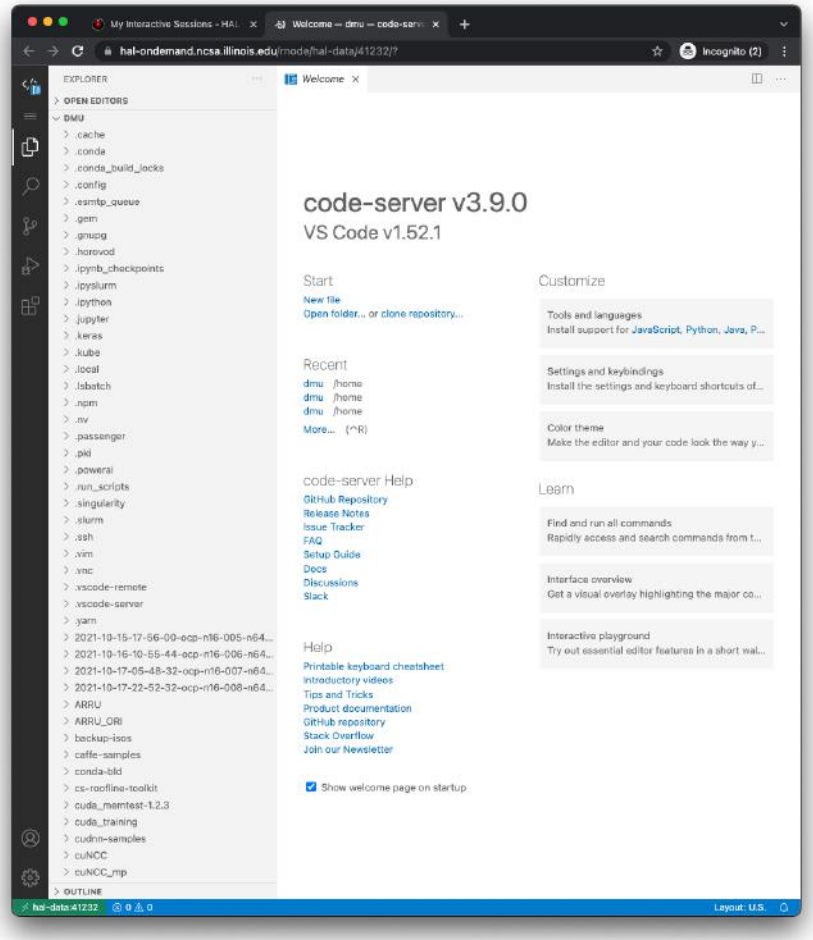


Open OnDemand (vs code server)





Open OnDemand (vs code server)



- Hands-on Demo Time
- uiuc_32

THANK YOU FOR YOUR TIME !

ILLINOIS
NCSA | National Center for
Supercomputing Applications