Interactive High-performance (HPC) Computing

HPC & Data Science Summer Institute 2024

By: Mary Thomas

https://github.com/sdsc-complecs/interactive-computing/

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

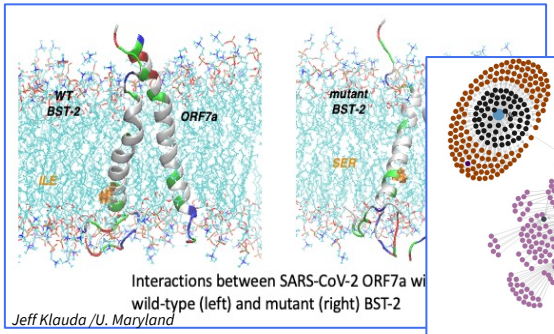
Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

You can run amazing jobs on supercomputers!

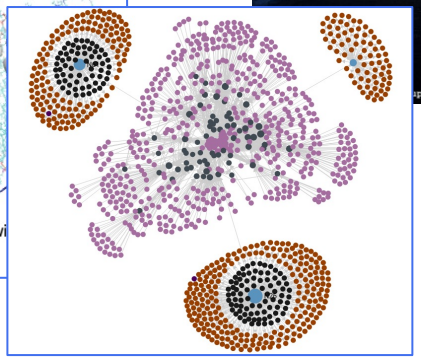
Typically parallel or very large memory

Pelagic fish communities Shapes (Jerome Guiet, UCLA)

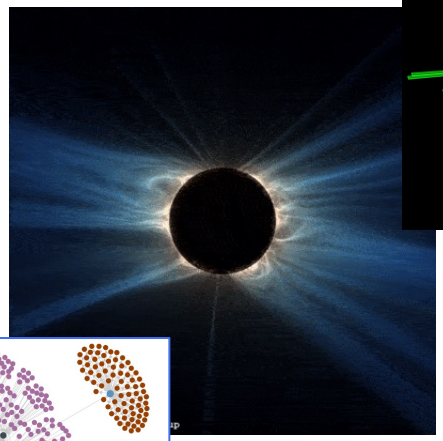


Interactions between SARS-CoV-2 ORF7a wi
wild-type (left) and mutant (right) BST-2

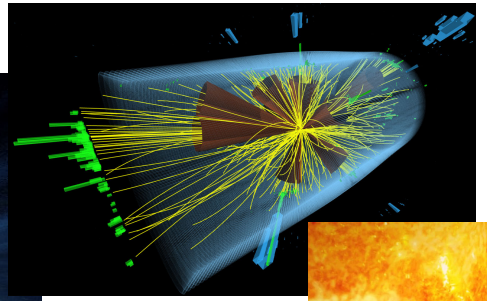
Jeff Klauda /U. Maryland



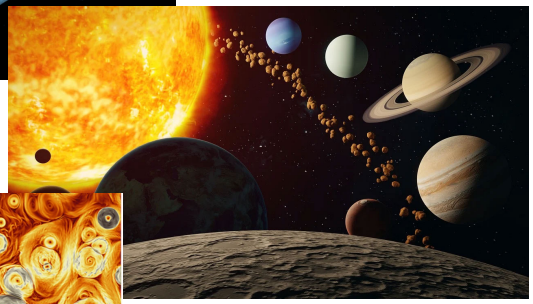
Sample of Internet structure from CAIDA data (Mark Burgess, 12/16/21)



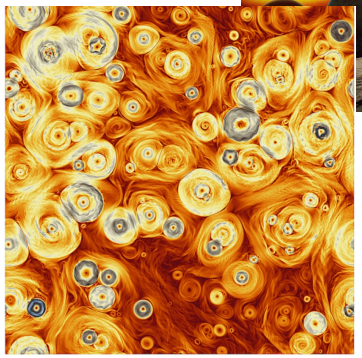
Cooper Downs, Predictive Science Inc



Compact Muon Solenoid (**CMS**)experiment at the LHC, CERN. *Image courtesy of CMS Collaboration; Mc Cauley, Thomas*

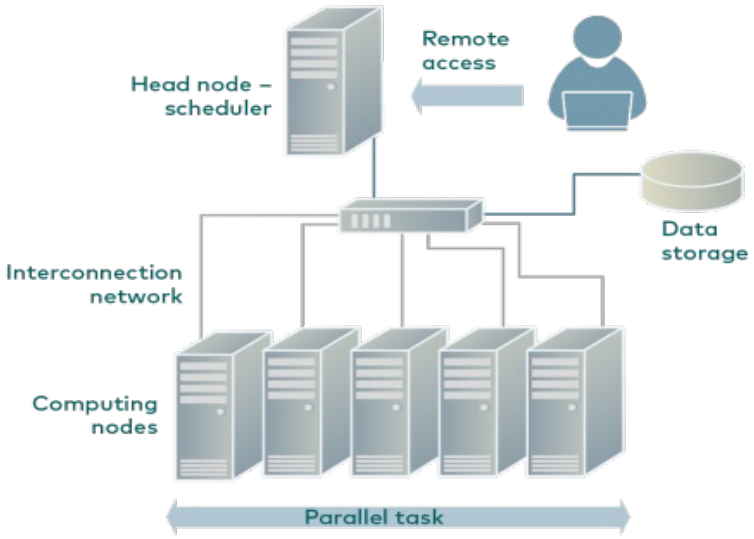


Model of Both Inner and Outer Solar System M. S.Clement (Carnegie Institution for Science)



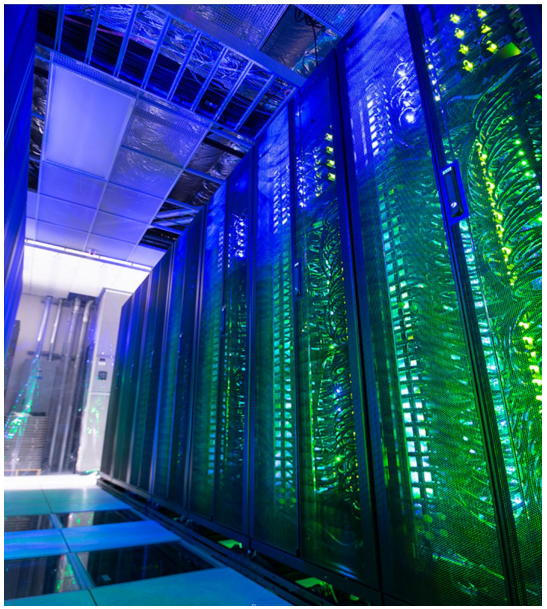
Modeling the sun's corona, Alfred Mallet (UC Berkeley)

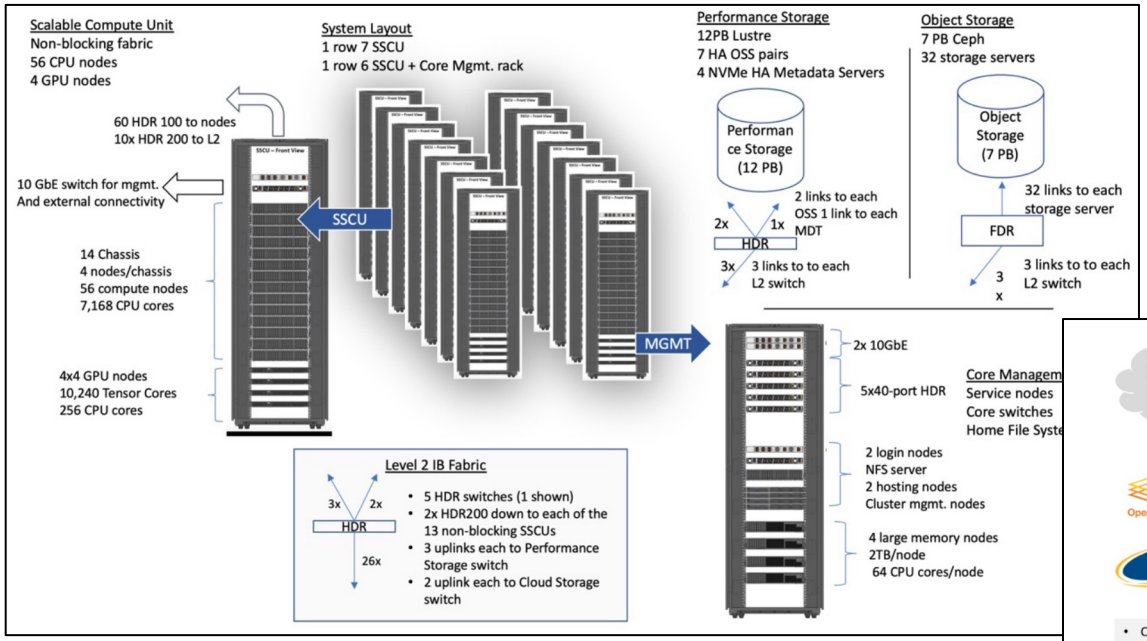
However, you have to run your jobs on HPC clusters

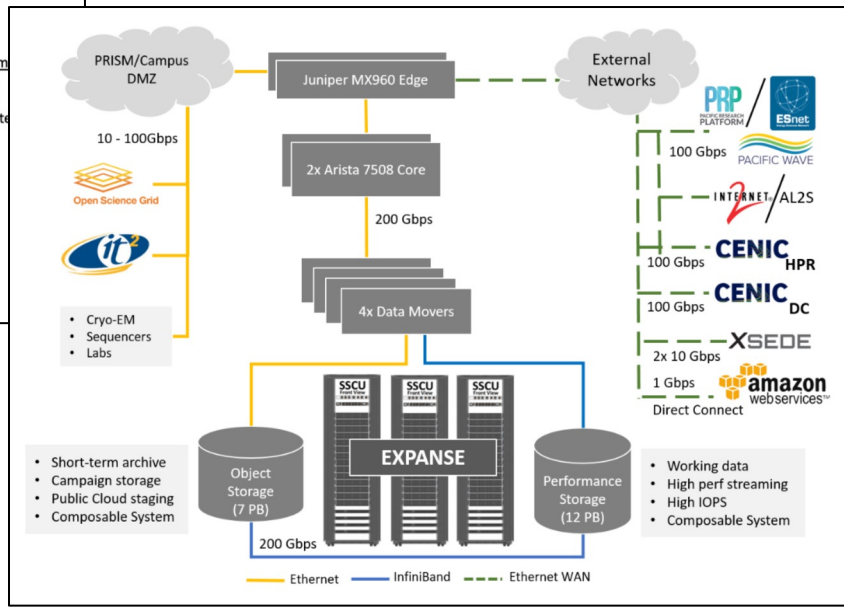


- Systems powerful but complex
- Jobs can be run from:
  - Command line interface (CLI)
  - The batch queue system
  - Application clients (PyTorch)
- Job scaling:
  - Parallel: 1 core → 1000s of nodes
- For large jobs, schedulers are needed to coordinate tasks

HPC System Architecture: Expanse @ SDSC







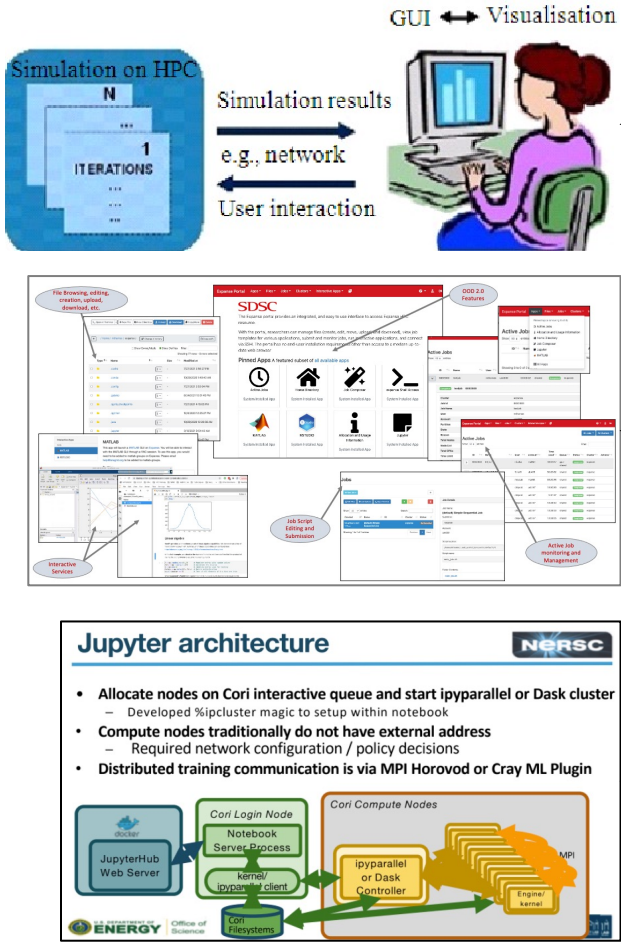
https://expanse.sdsc.edu
https://www.youtube.com/watch?v=uNZyg6X_t3s
https://access-ci.org

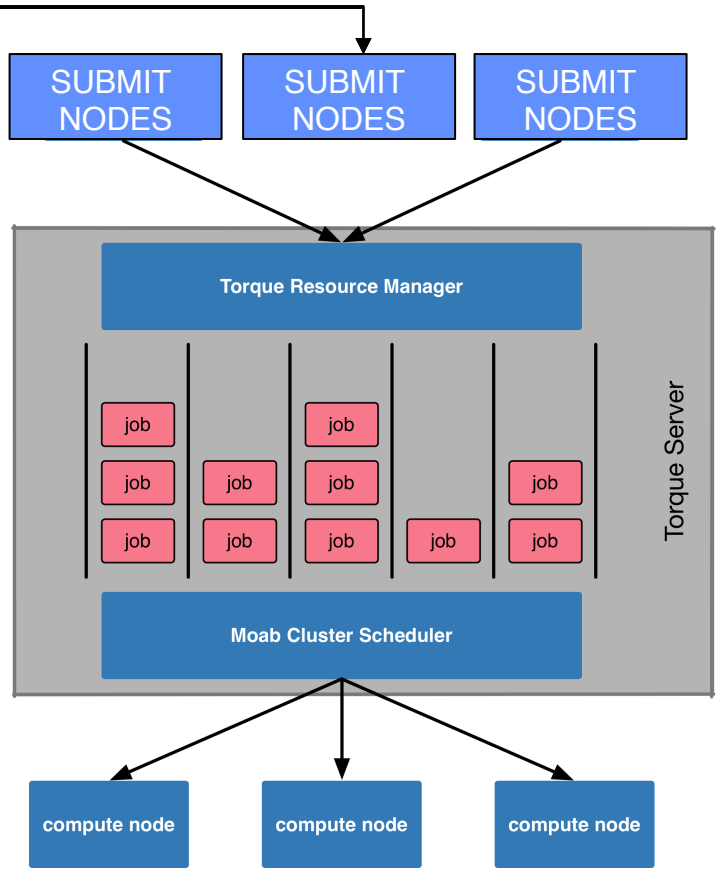
HPC Jobs can be run as batch jobs (background) or interactively (real time)

Terminal shell – submit batch

https://portal.expanse.edu

Interactive Distributed Computing with Jupyter





Src: https://hpc.dccn.nl/_images/torque_moab_arch.png

Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

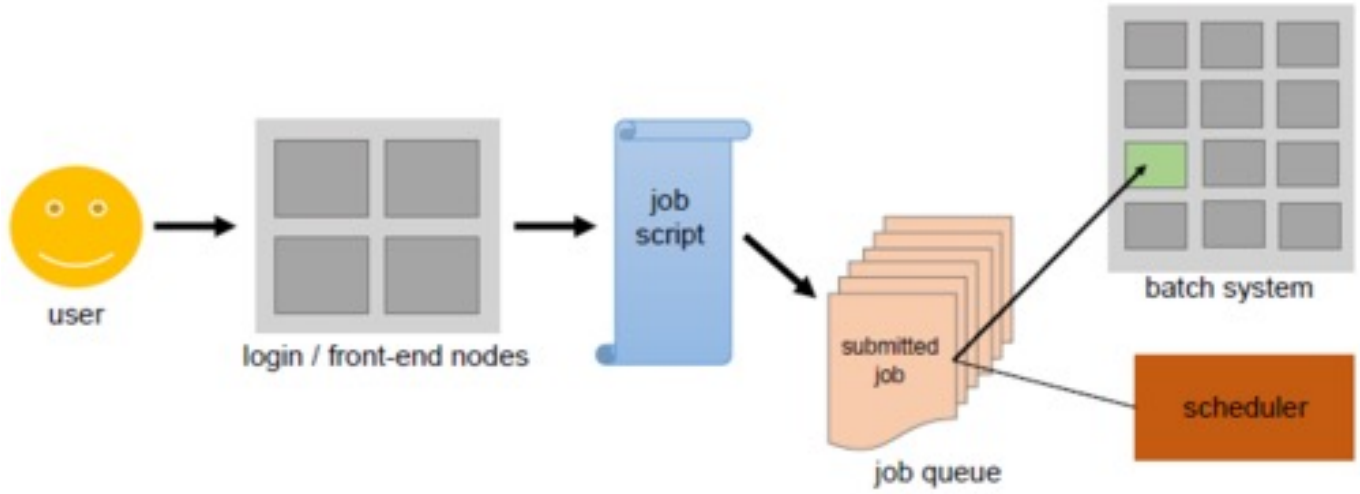
What is a batch job scheduler & how are they used?

- Any HPC (or HTC) system — usually a cluster of machines — needs a means of sharing computational resources fairly between users; without this, there would be anarchy.
- Batch-queueing systems — usually abbreviated to simply *batch systems* — are intended to do this.
- All batch systems have at least these features:
  - *a scheduler* for allocating resources (CPUs!) to jobs and for prioritizing jobs;
  - *one or more queues* to which jobs are submitted
  - Job partitions (or job queues): queue can be configured for a particular type of job (serial or parallel jobs; long or short jobs; or those requiring particularly high memory); members of a group or project).
- You need a basic understanding of batch jobs work in order to run interactive jobs on HPC Clusters (ironic?)

Batch Scheduler: main goals

- Minimize time between job submission and completion:
  - No job should stay in queue for extensive periods of time.
- Optimize CPU utilization:
  - Algorithms focus on minimizing CPU idle times.
- Maximize job throughput:
  - Manage as many jobs per time unit as possible.
- Support running jobs automatically in the background
- Finalizes jobs: ensures job data and results are recorded & stored where you want them.

Simple “Batch Scheduler” Architecture



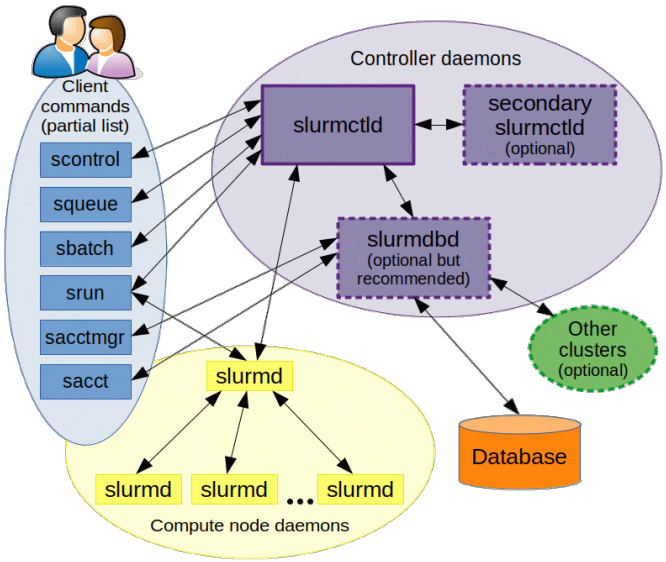
- Batch scheduler: software that implements a *batch system* on a cluster.
- Users do not run calculations interactively -- instead they submit *non-interactive batch jobs* to the *scheduler*.
- All work about the same: some are open source; some cost money; some are very expensive.

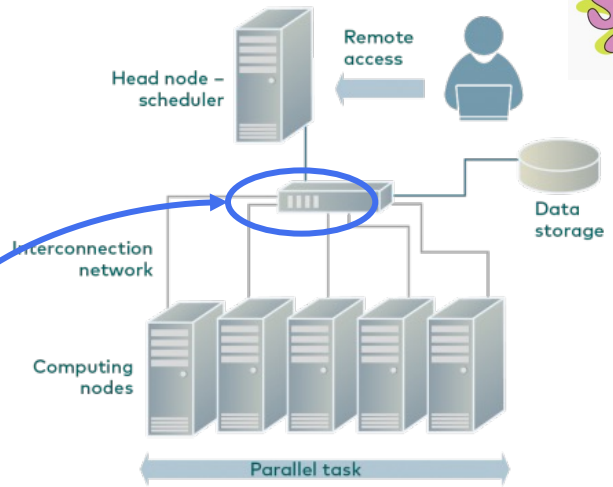
https://hpc-wiki.info/hpc/Scheduling_Basics

Batch jobs: Slurm resource manager

Simple Linux Utility for Resource Management

- Open Source, runs on many systems
- “Glue” for parallel computer to schedule and execute jobs
- Role: Allocate resources within a cluster
  - Nodes (unique IP address)
  - Interconnect/switches
  - Generic resources (e.g. GPUs)
  - Launch and otherwise manage jobs





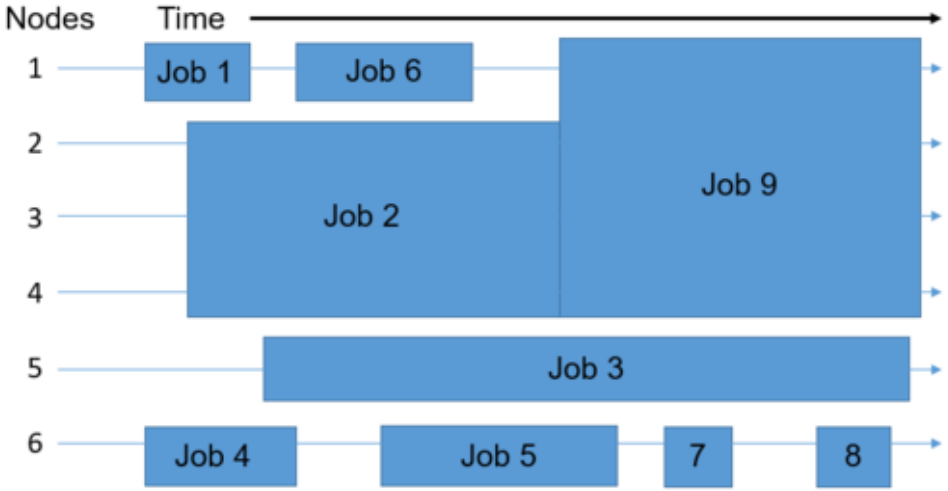
Functionality:

- Prioritize queue(s) of jobs;
- decide when and where to start jobs;
- terminate job when done;
- Appropriate resources;
- manage accounts for jobs

How a scheduler schedules jobs

- Simple example:
  - a 6-node system
  - user wants to run 9 jobs.
- Scheduler places the jobs in the queue and then onto the available nodes as they open up.

Many parameters affect scheduling: number of jobs submitted, required runtime, required number of cores, required main memory, accelerators, libraries, etc.



Scheduler needs to play kind of "multidimensional tetris" to fill the cluster's nodes evenly and efficiently.

https://hpc-wiki.info/hpc/Scheduling_Basics

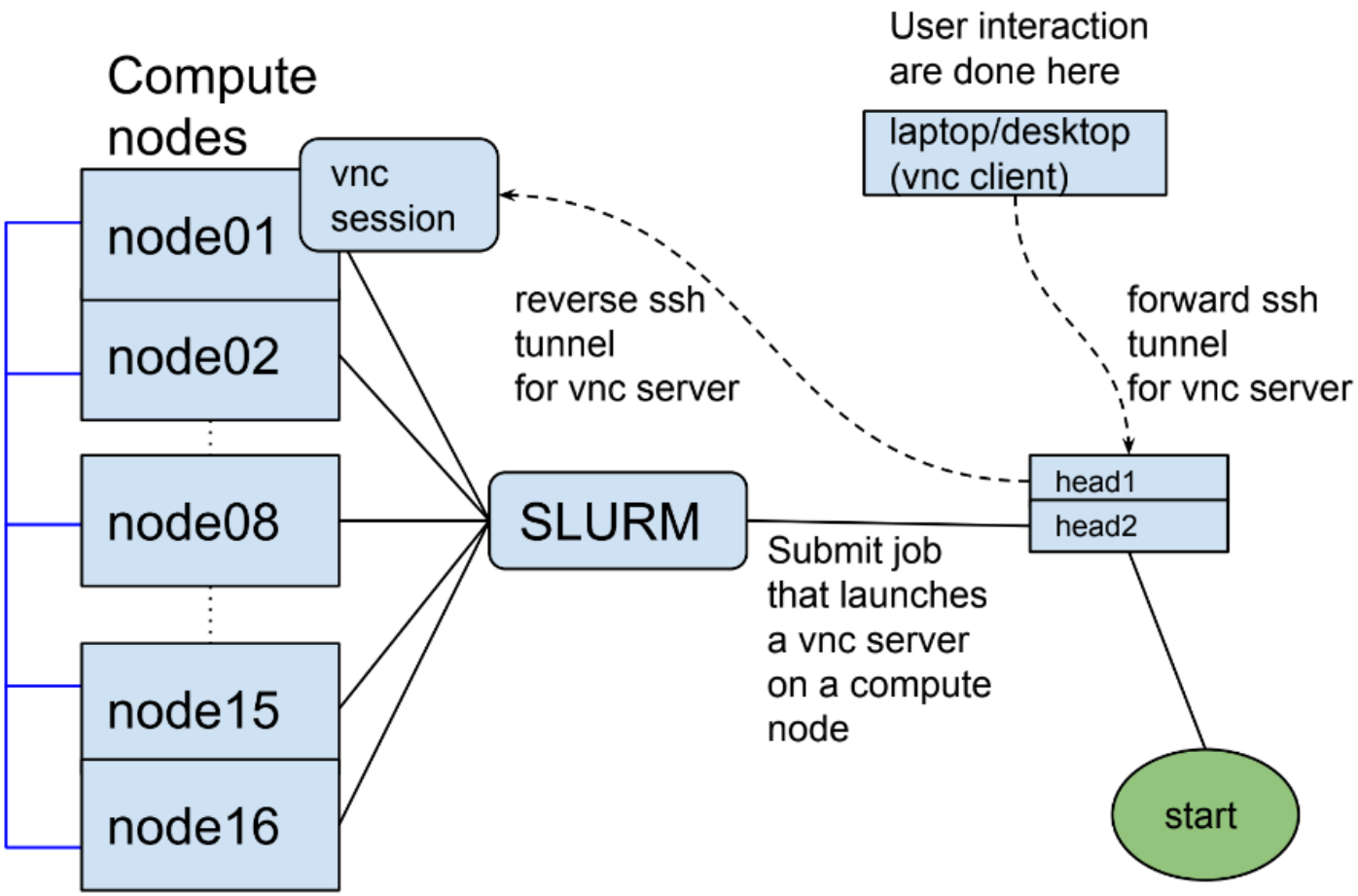
Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

What is Interactive HPC Computing

- In **computer** science, **interactive computing** refers to software which accepts input from the user as it runs.
  - **Interactive** software includes commonly used programs, such as word processors or spreadsheet applications.
- **Interactive HPC computing** involves *real-time* user inputs to perform tasks on a set of compute node(s) including:
  - Code development, real-time data exploration, and visualizations.
  - Used when applications have large data sets or are too large to download to local device, software is difficult install, etc.
  - User inputs come via command line interface or application GUI (Jupyter Notebooks, Matlab, R-studio).
  - Actions performed on remote compute nodes as a result of user input or program out.

Real-time User Interactions



real-time user inputs, perform tasks on a set of compute nodes

Interactive HPC Computing- Motivation

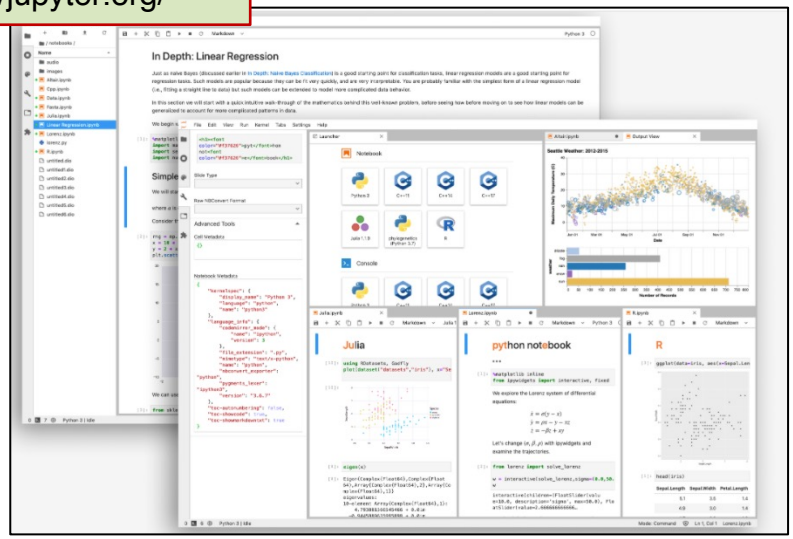
- Need more memory: Your jobs no longer fit onto the CPU/system you have been using:
  - That is why Expanse has 768 nodes (128 cores per node), with 256 GB DDR memory on each node
  - My MacBook Pro: 1 node, 8 cores, 16 GB DDR
- Too much data: your application needs more room:
  - Expanse has 1TB NVME/node, and 12 PB file system.
  - My MBP: 500 GB, no NVME
- Your network is too slow:
  - Expanse has connections to ~ 150GB/sec (or faster) networks
  - My MBP: 300 Mbps download; 11 Mbps upload
  - Bioinfo lab, running analysis on PC: FastQ dataset ~ 500 MB: would need 360+ seconds to upload 1 run.

Interactive HPC Methods & Applications

- Interact with data after job is done:
  - Unix: query file info, location, output, grep, awk, sed
  - Cat the file contents from batch job or raw data
  - Data browser
- Plot results:
  - From within the code/model using libraries
  - Command-line driven graphing utility : Gnuplot
- Data visualization apps
  - NetCDF, HPF, TAU, ParaView
  - Notebooks, others
- Data Analysis Platforms: Matlab, R

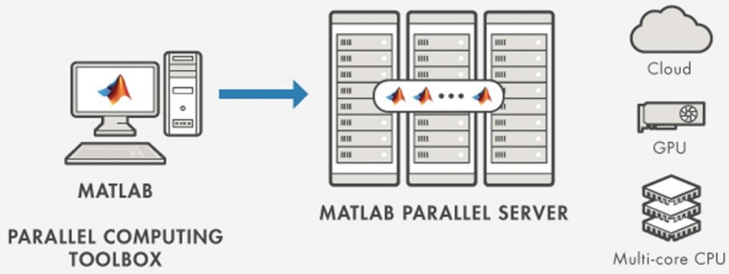
Interactive HPC Computing

https://jupyter.org/



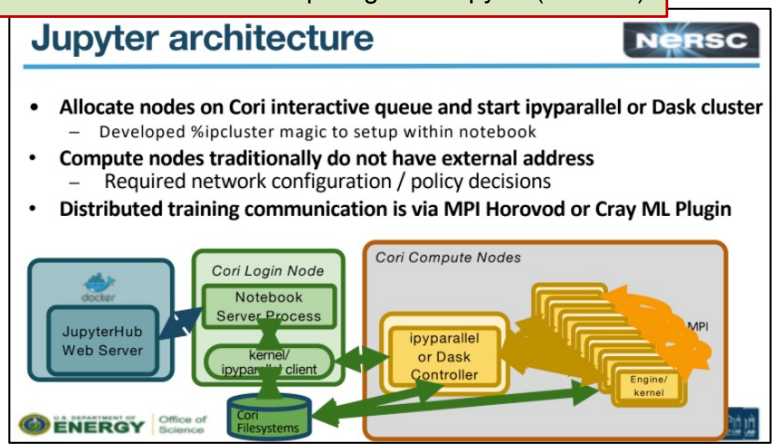
Parallel Matlab (AWS)

```
>> parpool(parcluster('HPC1'),100);
>> parfor i=1:3000
>>     c(i,:) = eig(rand(1000));
>> end
```



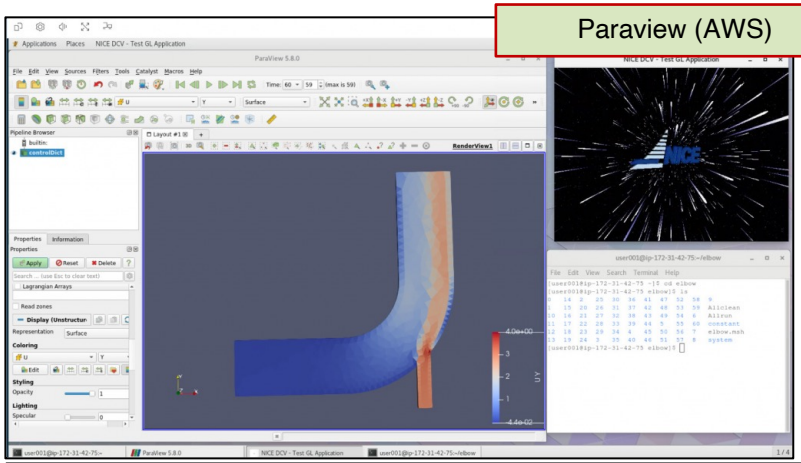
https://azuremarketplace.microsoft.com/en-us/marketplace/apps/mathworks-inc.matlab-parallel-server-listing?tab=Overview

Interactive Distributed Computing with Jupyter (NERSC)



https://drive.google.com/file/d/1-OFjrk1q3L1d3uakr2xkozrPn2c2VZpZ/view

Paraview (AWS)

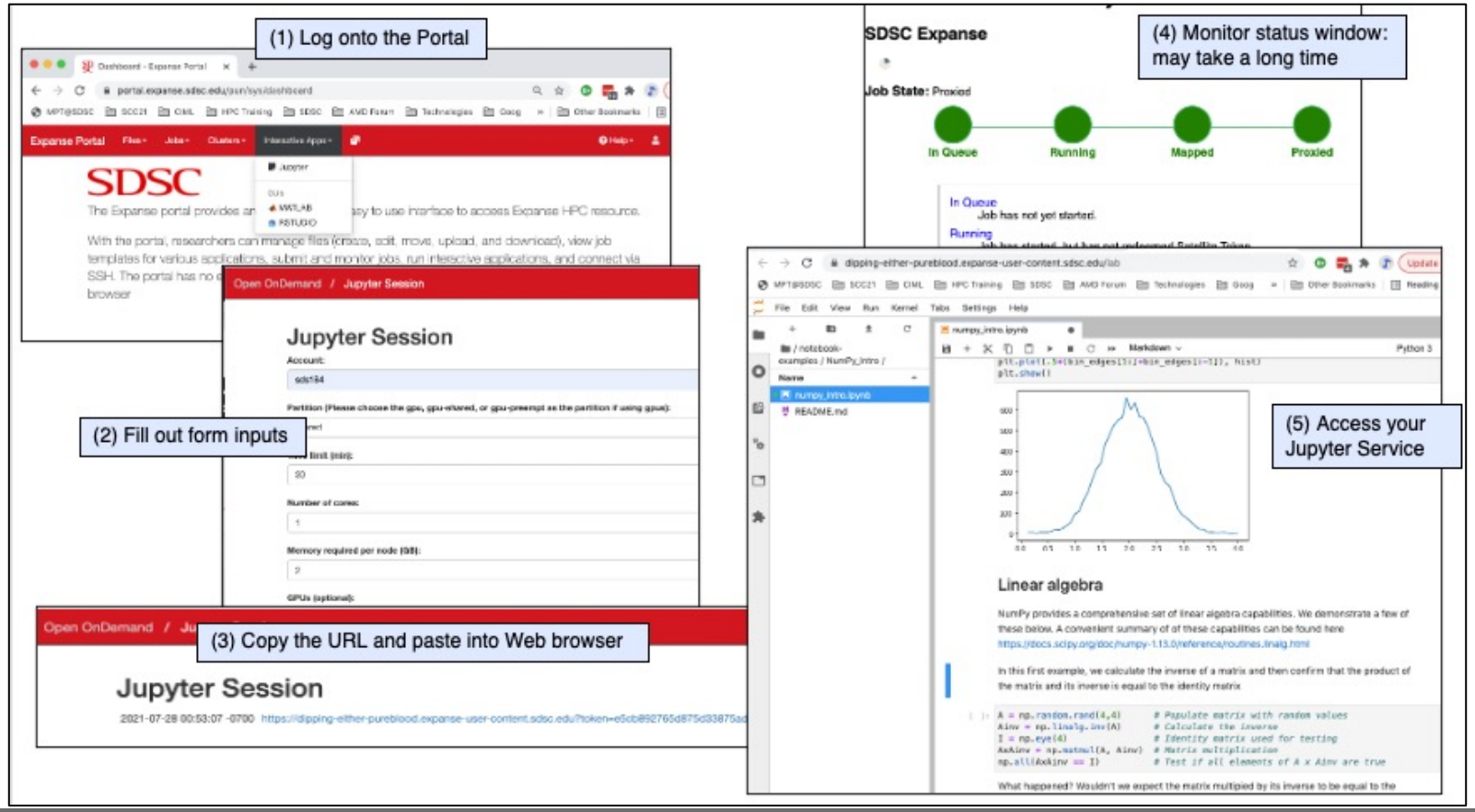


https://aws.amazon.com/blogs/compute/how-to-run-3d-interactive-applications-with-nice-dcv-in-aws-batch/

Expanse User Portal

https://portal.expanse.sdsc.edu

- Provides Web based access to interactive applications including Jupyter Notebooks & Jupyter Lab, Matlab, Rstudio.
- Access using your ACCESS portal account



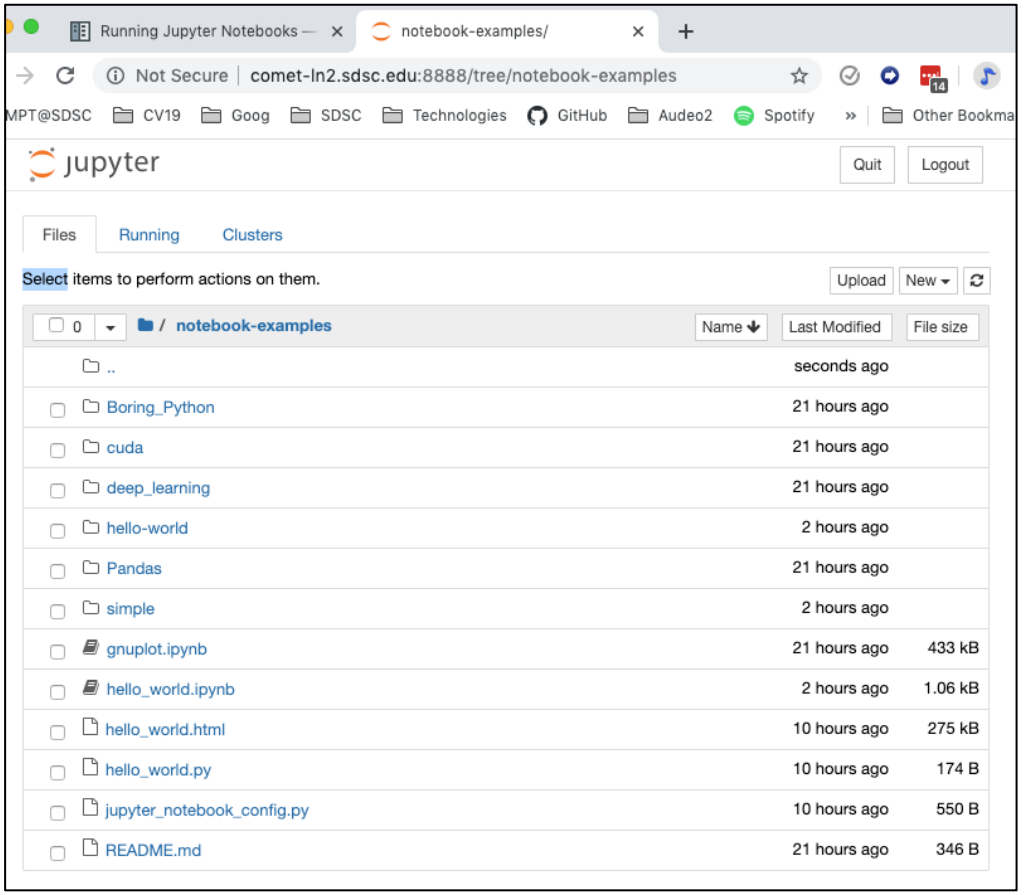
Interactive Services have a Key Vulnerability: They Provide Access to HPC File Systems, often over HTTP

SDSC Interactive Services Policy:

- Portals, JupyterHub, and other services cannot be mounted directly to disk (must be on VM or external)
  - Many use root in vulnerable ways
  - If a user launches Jupyter Lab or Notebooks, the jobs will be killed.
- Applications cannot run on login nodes

SDSC recommendation:

- Use secure connections: when you choose insecure connections your account is vulnerable to hacking
- Use portal.expanse.sdsc.edu



Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

When would you need to use an Interactive node?

- The application is interactive (Matlab, Notebooks)
- Need to stage large amounts of data
- AI applications (Pytorch, Tensorflow)
- Need to compile a very large, complex application that may take a long time.
  - Java code, 1 million lines, > 30 minutes on DELL 5511 laptop

Accessing Interactive Compute Nodes on Expanse

- Connect to HPC system (e.g. Expanse) via terminal using SSH → secure connections
- Use the *srun* command to obtain nodes for 'live,' command line interactive access:

CPU	srun --partition=debug --pty --account=--account=<<project>> --nodes=1 --ntasks-per-node=4 --mem=8G -t 00:30:00 --wait=0 --export=ALL /bin/bash
GPU	srun --partition=gpu-debug --pty --account=<<project>> --ntasks-per-node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash

(Tested 04/17/2024)

Using An Interactive CPU node

```
[mthomas@login01 calc-prime]$ srun --partition=gpu-debug --pty --
account=use300 --ntasks-per-node=10  --nodes=1 --mem=96G --gpus=1 -t
00:30:00 --wait=0 --export=ALL /bin/bash

srun: job 24457429 has been allocated resources
[mthomas@exp-9-55 calc-prime]$ module purge
[mthomas@exp-9-55 calc-prime]$ module load slurm
[mthomas@exp-9-55 calc-prime]$ module load cpu
[mthomas@exp-9-55 calc-prime]$ module load gcc/10.2.0
[mthomas@exp-9-55 calc-prime]$ module load openmpi/4.1.1
[mthomas@exp-9-55 calc-prime]$ mpirun -n 64 ./mpi_prime
06 August 2023 11:10:26 PM
PRIME_MPI   n_hi= 5000000      C/MPI version
An MPI example program to count the number of primes: #
processes is 64
         N        Pi          Time
         1         0        0.013258
         2         1        0.001058
         4         2        0.000101
         8         4        0.000101
     [SNIP]
    131072     12251        0.110848
    262144     23000        0.410792
    524288     43390        1.527210
   1048576     82025        5.733612
   2097152    155611       21.725862
PRIME_MPI - Master process: Normal end of execution.
06 August 2023 11:12:26 PM
```

Request an interactive node for 30 minutes

- Exit interactive session when your work is done or you will be charged more CPU time.
- Beware of oversubscribing your job: don't ask for more cores than you have requested.
- Intel compiler allows this, but your performance will be degraded.

Using Interactive GPU nodes

Request an interactive node for 30 minutes

```
[snip]
Last login: Fri Feb 18 12:58:32 2022 from 76.176.117.51
[username @login02 ~]$
[username @login02 ~]$ srun --partition=gpu-debug --pty --account=use300 --ntasks-per-
node=10 --nodes=1 --mem=96G --gpus=1 -t 00:30:00 --wait=0 --export=ALL /bin/bash
srun: job 9794018 queued and waiting for resources
srun: job 9794018 has been allocated resources
[mthomas@exp-14-57 ~]$
[mthomas@exp-14-57 ~]$ nvidia-smi
Fri Feb 18 13:04:19 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================
|   0  Tesla V100-SXM2...  On   | 00000000:86:00.0 Off |                    0 |
| N/A   34C    P0    41W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================|
|  No running processes found
+-----------------------------------------------------------------------------+
[username @exp-14-57 ~]$ exit
```

Verify you are on a GPU node

Exit when tasks are done

Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

Visualizing Data Methods

- Viewing Data: unix file ops (grep, awk, cat),
  - Note: gnuplot, NetCDF – apps don't work well on Expanse: can be very slow
- Programming & Visualization Platforms:
  - Matlab, R, Jupyter Notebooks, Paraview
  - Many require X11 servers on the local host; which is slow and unstable (especially MacOC)
- Jupyter notebooks have many packages that can be used to visualize data.
- Gateways & Portals:
  - simplify access to interactive apps

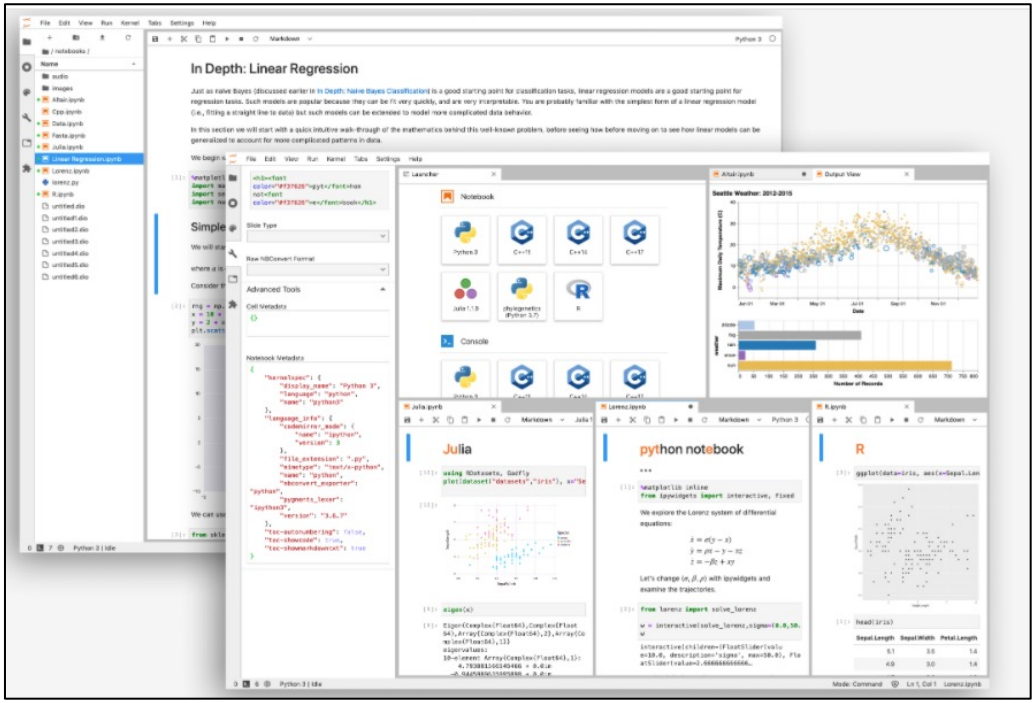
Juptyer Notebooks

What is Jupyter?

> ***Jupyter*** *is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document.* *(J. Perkel, https://www.nature.com/articles/d41586-018-07196-1)*

Common Jupyter Services:

- Jupyter Notebooks (single user)
- JupyterLab: advanced version of notebook
- JupyterHub: multiuser Jupyter service



Accessing and Running Secure Notebooks on SDSC HPC Systems

- Install notebook application:
  - Locally: install Anaconda on your laptop
  - Remotely:
    - Install Anaconda/conda on the remote machine (default is HTTP) – not recommended
- Running remotely:
  - Connect over HTTP (default, insecure)
  - Connect over HTTP + SSH tunneling (secure, but inconvenient)
  - Connect over HTTPS + using the *Satellite Reverse Proxy Service* (SRPS) and *`galyleo` client (secure, convenient)*
- ➢ You can launch Jupyter services on SDSC:
  - Launch securely (HTTPS) using SRPS/*`galyleo`* -- *`recommended`*
  - CPU and GPUs
  - Interactive nodes: command line *`or`* Slurm batch script
- Treat the Notebook URL like a Password!

SDSC Satellite Reverse Proxy Service

- SRPS: prototype system that allows users to launch secure standard Jupyter Notebooks on on any Expanse compute node using a reverse proxy server.
  - Notebooks will be hosted on the internal cluster network as an HTTP service using standard Jupyter commands.
  - Service available to the user outside of the cluster firewall over HTTPS connection between the external users web browser and the reverse proxy server.
- Goal: minimize software changes for users, improve security of user notebooks running on SDSC systems.
- SRPS can run on any HPC system capable of supporting Apache on internal network.

galyleo

- 2nd generation shell utility developed to orchestrate a user's interaction with both Satellite and Slurm to start a Jupyter session within a batch job.
- Developed while reviewing start-jupyter (prototype client) codebase to sort out how best to support Expanse (OOD) Portal and HPC User Services Group long-term; integrated into an existing SSH tunneling orchestration utility to use Satellite proxy service instead

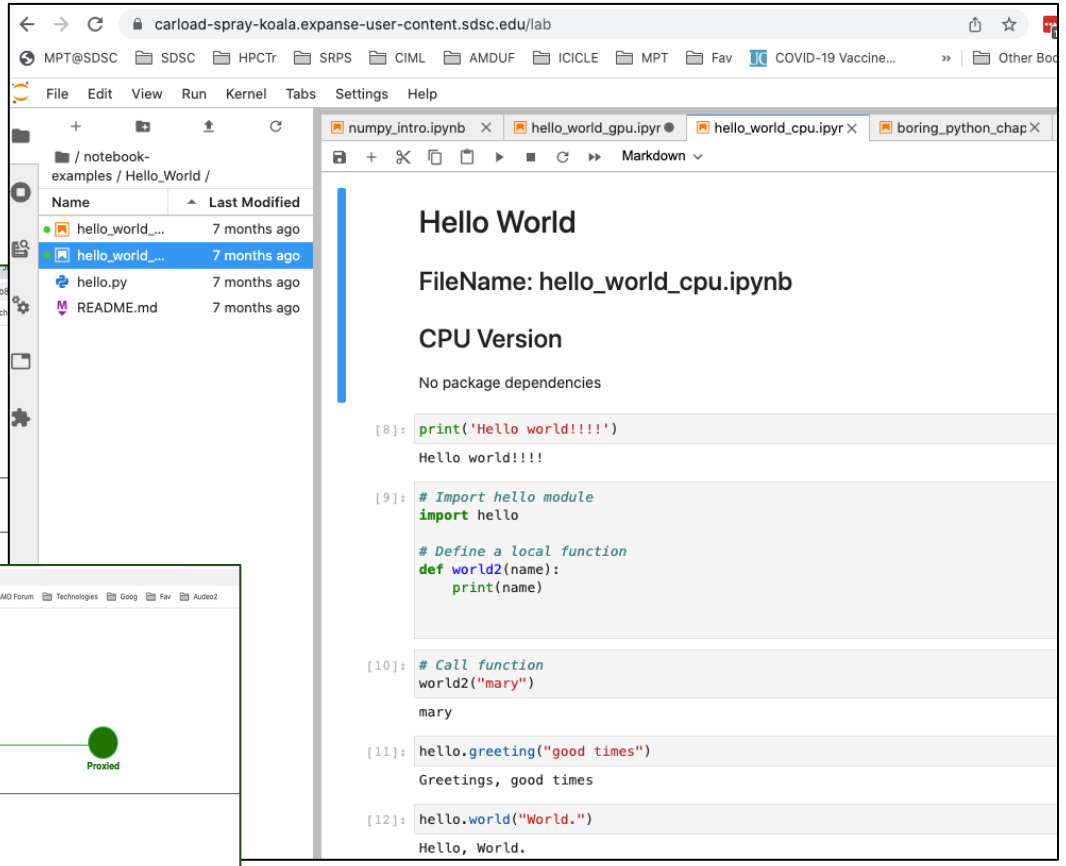
- Key features in design:
  - HTTPS URL
  - Supports containers (Singularity on Expanse) → **GPU environment**
  - No need to install conda environment or update packages
  - Increases flexibility for users to configure software environment; but also try to makes it simpler for them to do this themselves
  - Batch job script is generated completely on-the-fly.
  - Command-line argument driven.
  - Quiet mode for OOD portal

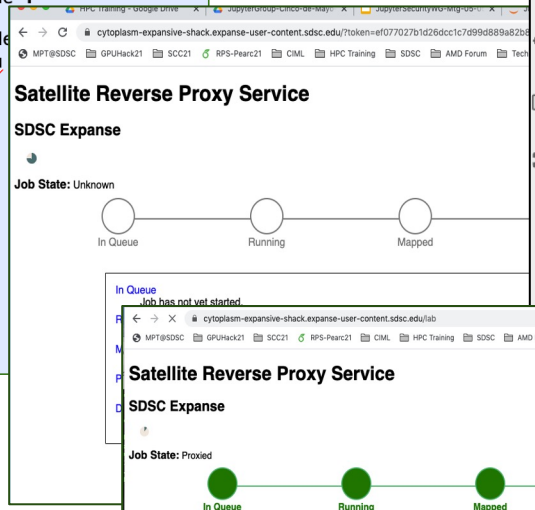
https://github.com/mkandes/galyleo

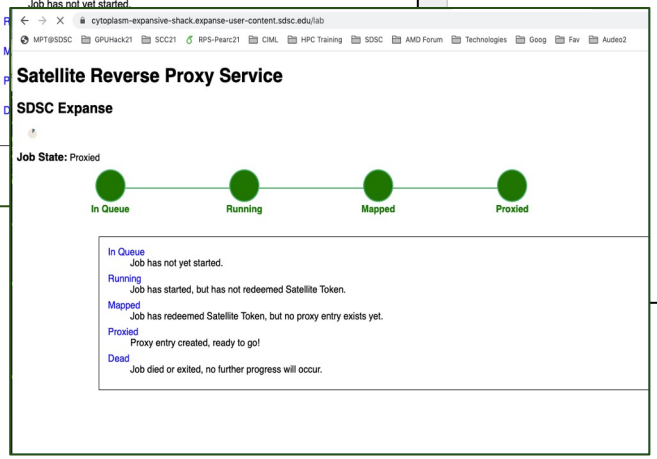
Satellite-Galyleo System

```
[username@login01 ~]export
PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"

[username@login01 ~]$ galyleo.sh --help
USAGE: galyleo.sh launch [command-line
option] {value}
command-line option      : value
    -A | --account         :
    -R | --reservation     :
    -p | --partition       :
    -q | --qos             :
    -N | --nodes           :
    -n | --ntasks-per-node :
    -c | --cpus-per-task
    -M | --memory-per-node
    -m | --memory-per-cpu
    -G | --gpus
       | --gres
    -t | --time-limit
    -j | --jupyter
    -d | --notebook-dir
    -r | --reverse-proxy
    -D | --dns-domain
    -s | --sif
    -B | --bind
       | --nv
    -e | --env-modules
       | --conda-env
    -Q | --quiet
```







Satellite Client: galyleo

Key features in design:

- User calls galyleo.sh launch script, which requests token from Satellite, passes token to batch job script and submits the job to Slurm; token redeemed from batch job once it runs
- Increase flexibility for users to configure software environment; but also try to make it simpler for them to do themselves
- Batch job script is generated completely on-the-fly.
- Command-line argument driven.
- Quiet mode for OOD portal

```
[username@login01 ~]export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"

[username@login01 ~]$ galyleo.sh --help
USAGE: galyleo.sh launch [command-line option] {value}
command-line option      : value
    -A | --account         :
    -R | --reservation     :
    -p | --partition       :
    -q | --qos             :
    -N | --nodes           :
    -n | --ntasks-per-node :
    -c | --cpus-per-task   :
    -M | --memory-per-node :  GB
    -m | --memory-per-cpu  :  GB
    -G | --gpus            :
       | --gres            :
    -t | --time-limit      :
    -j | --jupyter         :
    -d | --notebook-dir    :
    -r | --reverse-proxy   :
    -D | --dns-domain      :
    -s | --sif             :
    -B | --bind            :
       | --nv              :
    -e | --env-modules     :
       | --conda-env       :
    -Q | --quiet           : 1
```

https://github.com/mkandes/galyleo

Launching CPU notebooks using galyleo

Step 1:
Login and setup user environment

Step 2:
Run command to launch secure notebook

Step 3:
Copy URL; paste into browser on local system

Step 4:
Monitor Slurm queue; wait for job to start

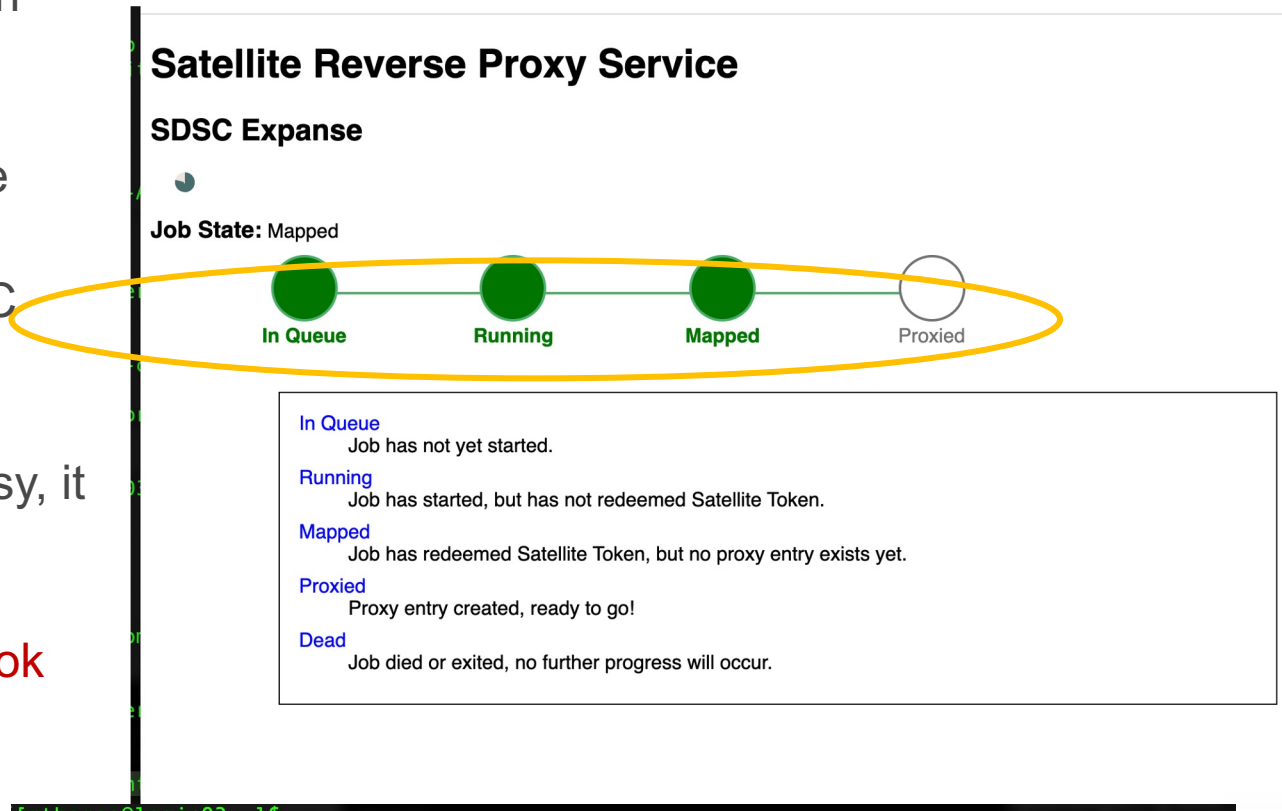
```
[username@login01 ~]$ which galyleo.sh
/usr/bin/which: no galyleo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-
centos8-zen/gcc-8.3.1
[SNIP]
home/username/.local/bin:/home/username/bin)
[username@login01 ~]$ export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"
[username@login01 ~]$ which galyleo
/cm/shared/apps/sdsc/galyleo/galyleo
[username@login01 ~]$ galyleo launch --account abc123 --partition shared --cpus 2
--memory 4 --time-limit 00:30:00 --env-modules cpu/0.17.3b,anaconda3/2021.05
[snip]
Submitted Jupyter launch script to Slurm. Your SLURM_JOB_ID is 9773665
[snip]
Your Jupyter notebook session will begin once compute resources are allocated to
your job by the scheduler.

https://carload-spray-koala.expanse-user-content.sdsc.edu/lab

[username@login01 ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
           9773665 gpu-debug galyleo-  username PD       0:00      1 (None)
[username@login01 ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
           9773665 gpu-debug galyleo-  username  R       0:20      1 exp-7-59
```

Satellite Server Status/Pending Page

- Load notebook URL in browser; wait for it to launch
- Monitor pending page
- Run the “squeue” command on the HPC system to check job status
- If the job queue is busy, it may take a while to launch the notebook
- Treat Jupyter Notebook URL as a password!




```
[mthomas@login02 ~]$
[mthomas@login02 ~]$ squeue -u mthomas
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9774239    shared galyleo-  mthomas  R       3:49      1 exp-1-13
           9774274     debug galyleo-  mthomas  R       0:12      1 exp-9-55
[mthomas@login02 ~]$
```

Launching GPU notebooks using galyleo

- GPU Notebooks run better when using containers. SDSC maintains several containers on Expanse
- See: /cm/shared/apps/containers/singularity

Step 1:
Login and setup user environment

Step 2:
Run command to launch secure notebook

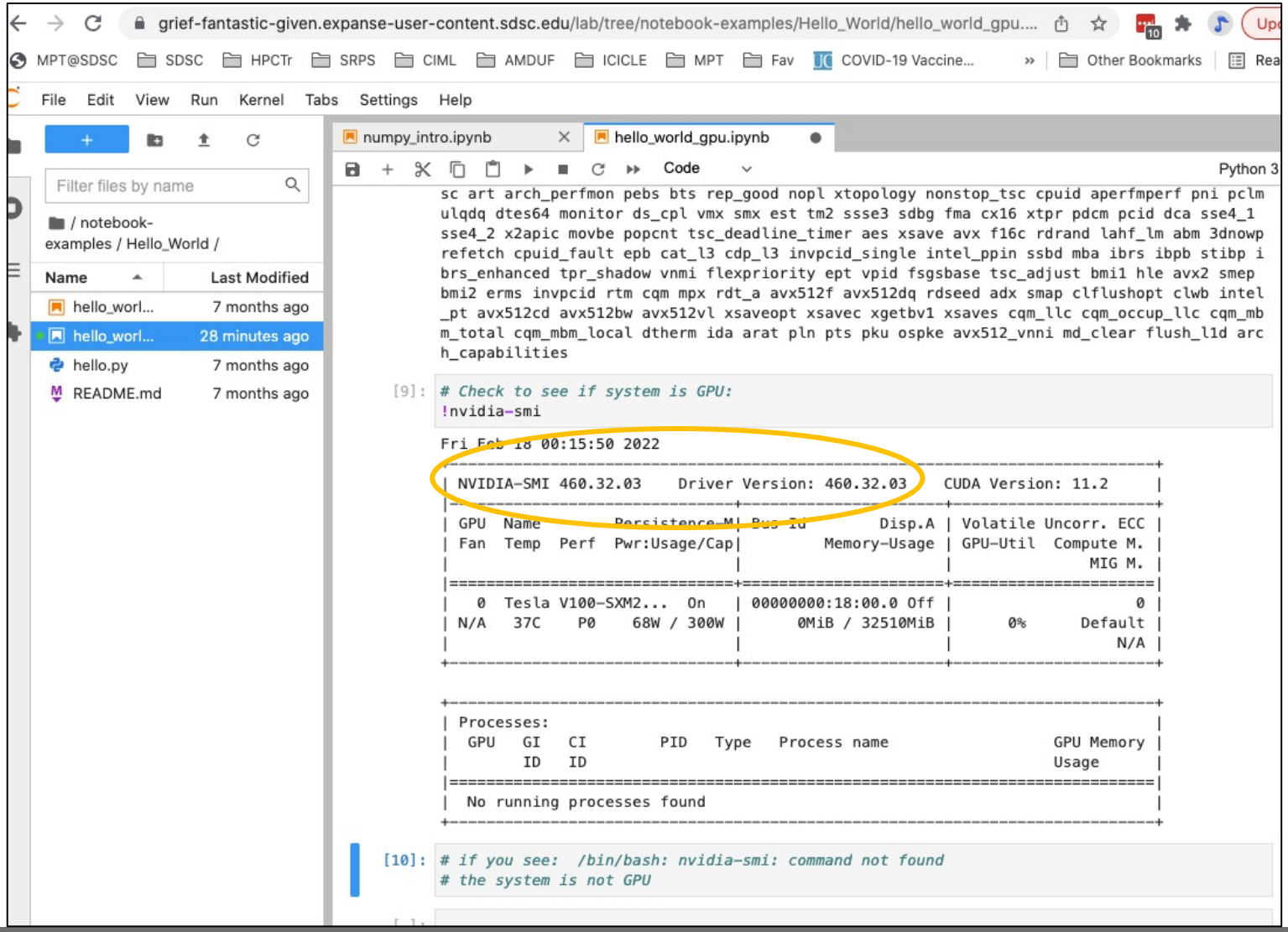
Step 3:
Copy URL; paste into browser on local system

Step 4:
Monitor Slurm queue; wait for job to start

```
[username@login01 ~]$ which galyleo.sh
/usr/bin/which: no galyleo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-
zen/gcc-8.3.1
[SNIP]
home/username/.local/bin:/home/username/bin)
[username@login01 ~]$ export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"
[username@login01 ~]$ which galyleo
/cm/shared/apps/sdsc/galyleo/galyleo
[username@login01 ~]$ galyleo launch --account use300 --partition gpu-debug --cpus 1
--memory 93 --gpus 1 --time-limit 00:5:00 --env-modules singularitypro --sif
/cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif --bind
/expanse,/scratch --nv
[snip]
Submitted Jupyter launch script to Slurm. Your SLURM_JOB_ID is 9773912
[snip]
Your Jupyter notebook session will begin once compute resources are allocated to
your job by the scheduler.
https://grief-fantastic-given.expanse-user-
content.sdsc.edu?token=5097acb6f32ab82dd51b4524c267d2fd

[username@login01 ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9773912 gpu-debug galyleo-  username PD       0:00      1 (None)
[username@login01 ~]$ squeue -u username
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9773912 gpu-debug galyleo-  username  R       0:20      1 exp-7-59
```

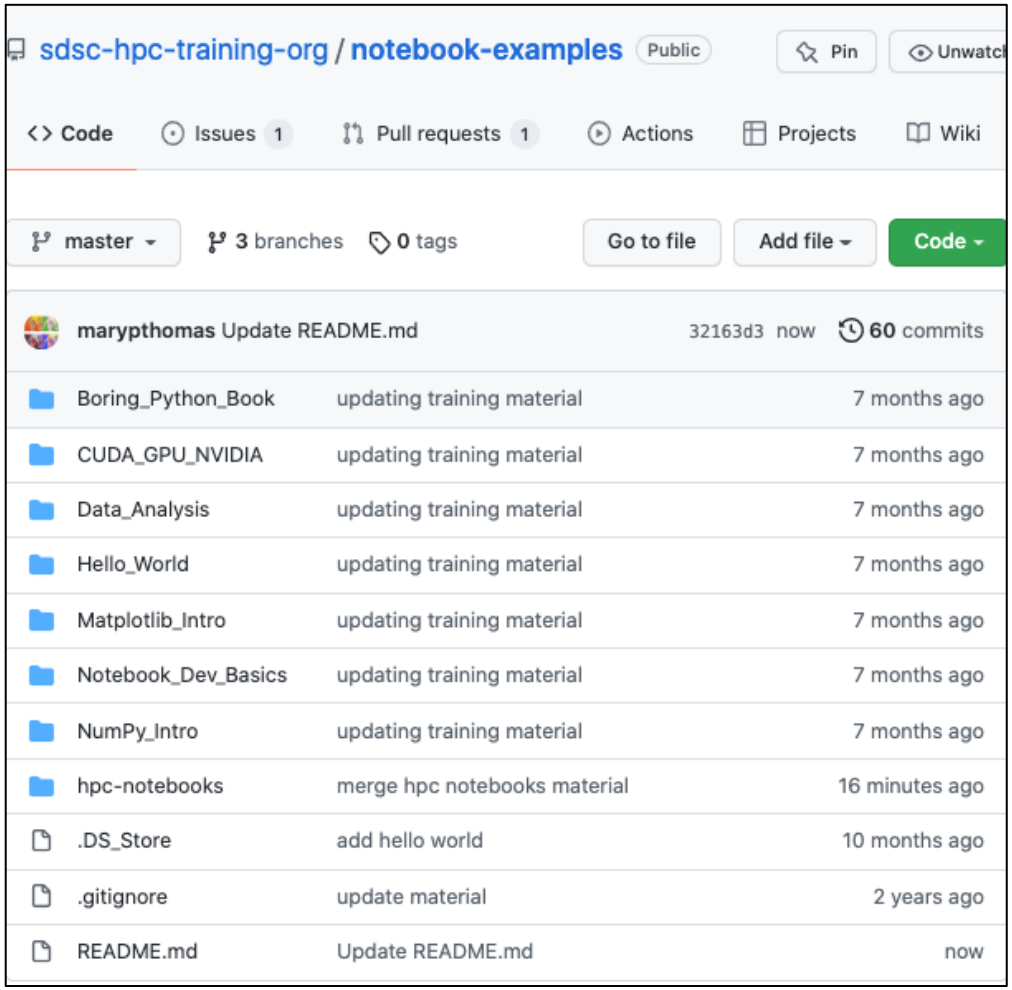
Verify notebook launched on GPU device



Notebook Examples

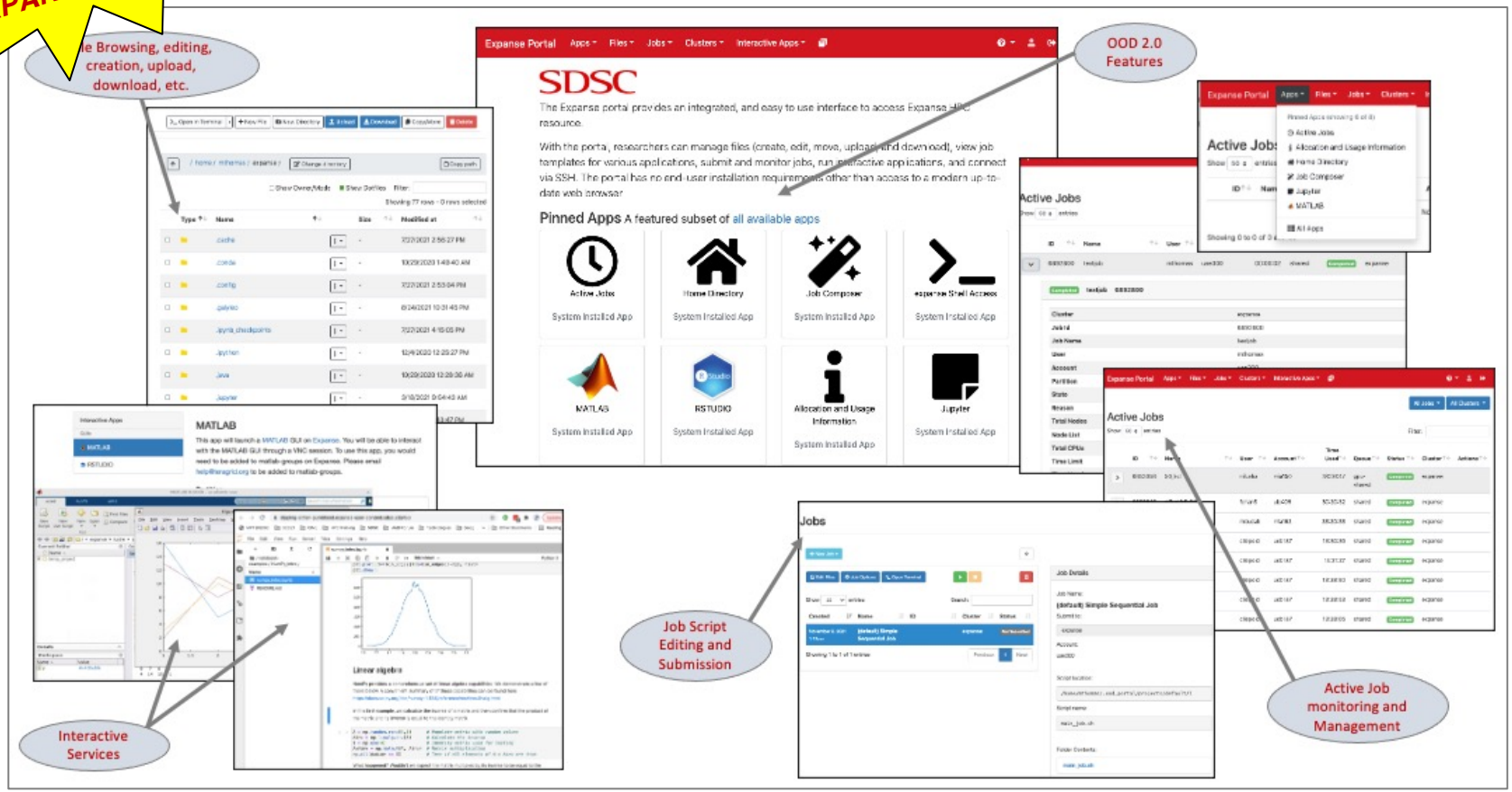
https://github.com/sdsc-hpc-training-org/notebook-examples

- Collection of tested, working notebooks tested on Expanse and other SDSC HPC systems
- Includes range of materials from “hello world” to Spark ML notebooks.
- Note: collection changes often, based on testing and contributions



Expanse User Portal

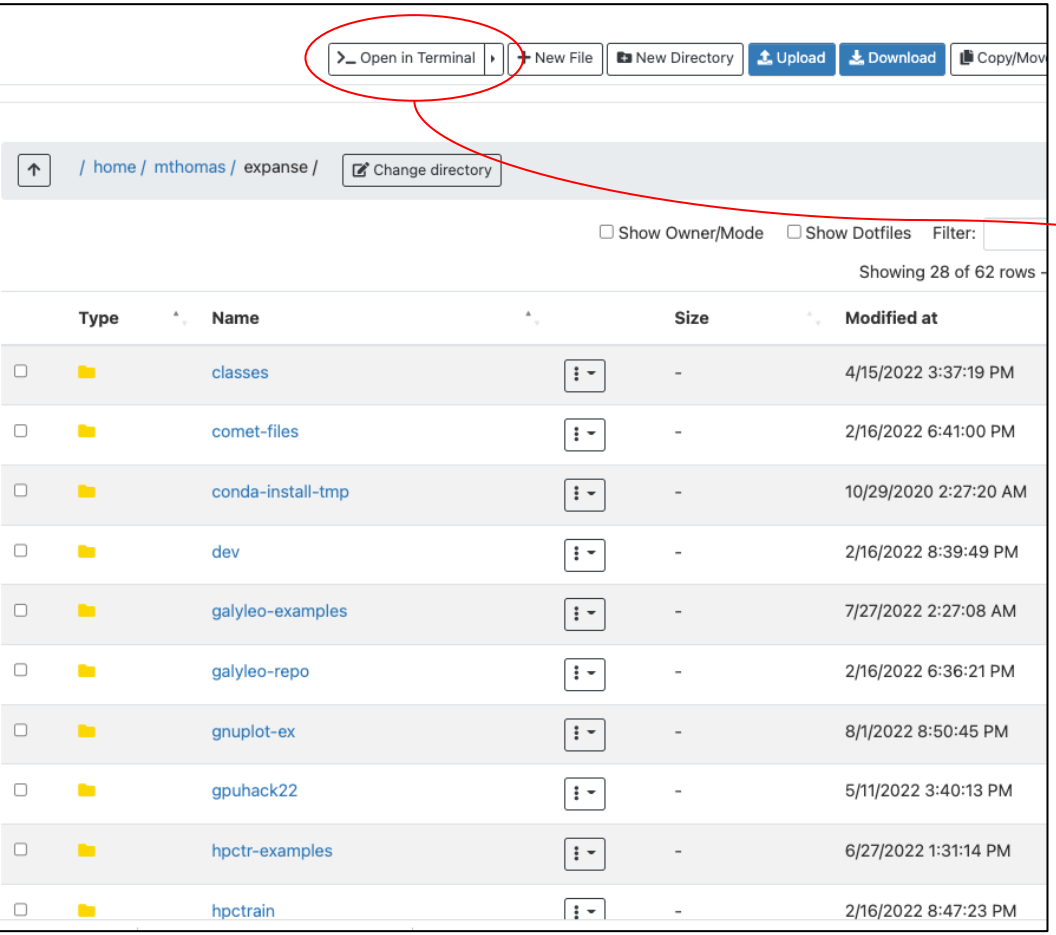
SIMPLIFY! EXPANSE PORTAL

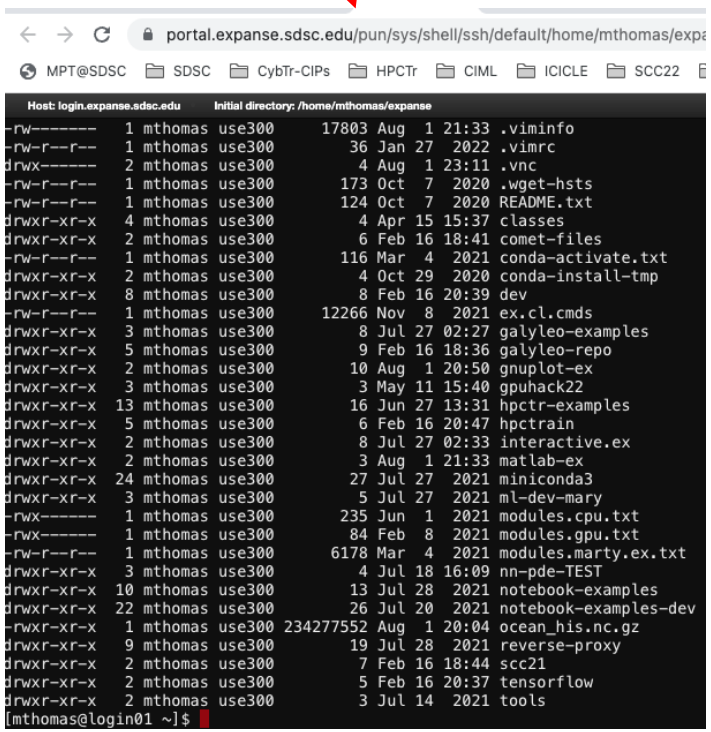


- https://portal.expanse.sdsc.edu; authenticate using ACCESS credentials
- Securely hosts batch job submission & monitoring, and interactive applications
- Simplifies launching supported interactive applications → manages software dependencies

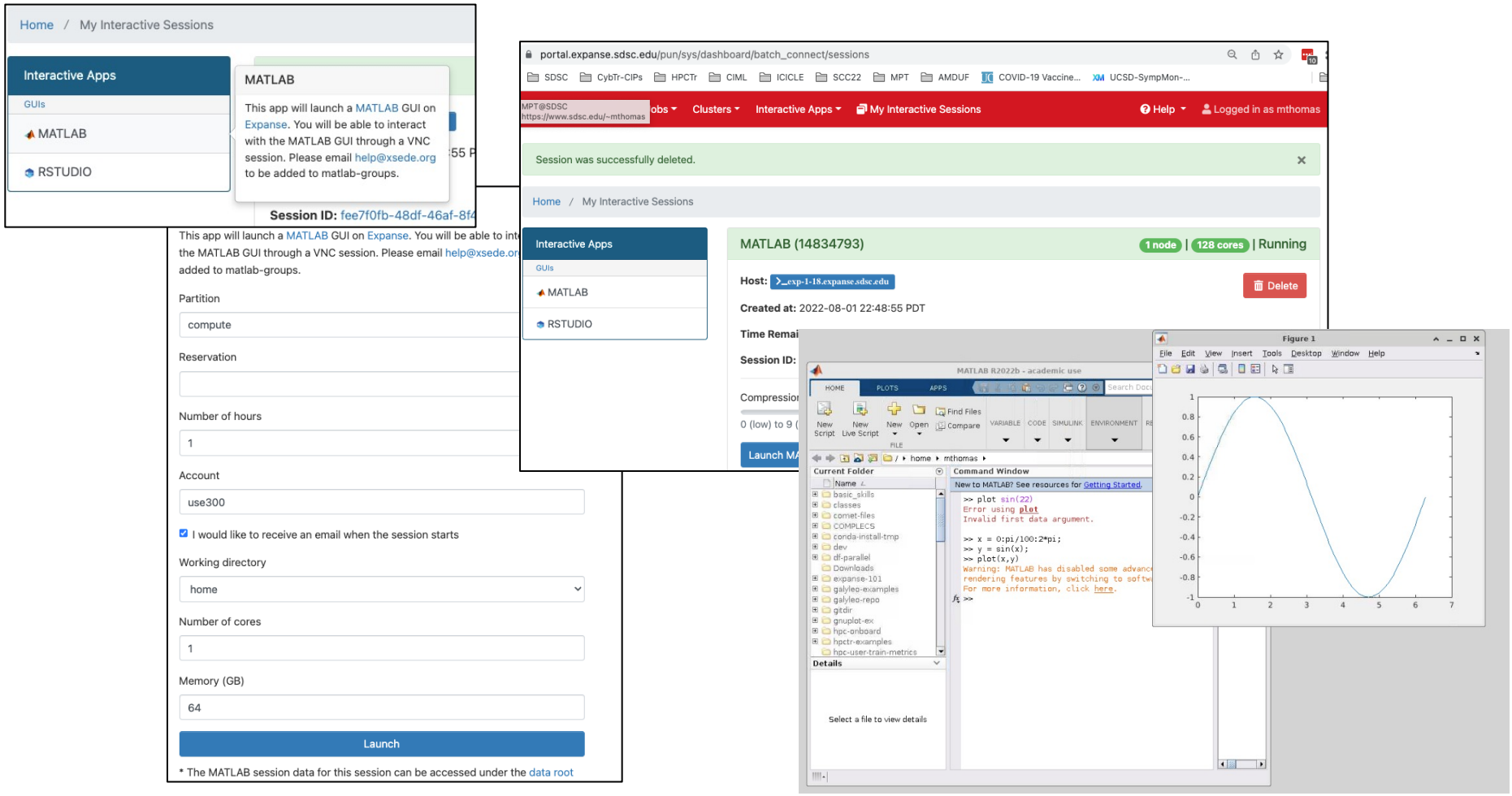
Sivagnanam, S., Irving, C., Kandes, M., Mishin, D., Sakai, S., Strande, S., Tatineni, M., Thomas, M., Norman, M. (2021). Experiences in building a user portal for Expanse supercomputer. *PEARC21: Practice and Experience in Advanced Research Computing*, Accepted. ACM. https://doi.org/10.1145/3437359.3465590

Expanse Portal: File Management

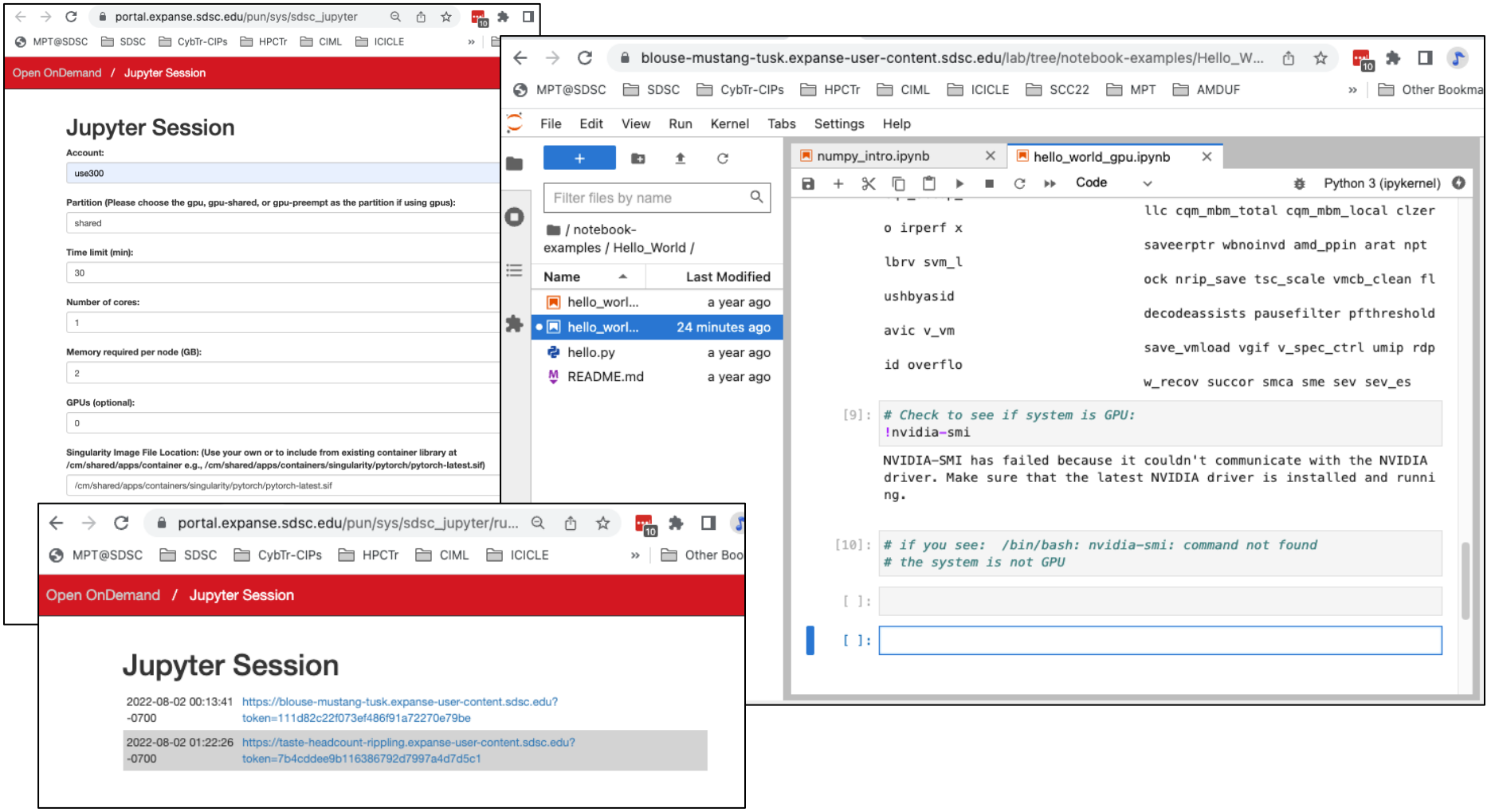




Expanse Portal: Running Matlab



Expanse Portal: Launching Notebooks



Expanse Portal: Input data example for Jupyter Notebook

Account: use300
Partition: *(Please choose the gpu, gpu-shared, or gpu-preempt as the partition if using gpus): debug*
Time limit (min): 30
Number of cores: 1
Memory required per node (GB): 2
GPUs (optional): 0
Singularity Image File Location: *(Use your own or to include from existing container library at /cm/shared/apps/container e.g., /cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif)*
/cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif
Environment modules to be loaded *(E.g., to use latest version of system Anaconda3 include cpu,gcc,anaconda3):* singularitypro
Conda Environment (Enter your own conda environment if any):
Conda Init (Provide path to conda initialization scripts):
Conda Yaml *(Upload a yaml file to build the conda environment at runtime) No file chosen:*
Turn on use of mamba for speeding up conda-yml installs:
Enable use of new caching mechanism that will store and reuse conda-yml created environments using conda-pack !????
Reservation:
QoS:
Working directory: HOME
Type: JupyterLab

Outline

- Defining Interactive HPC
  - High-performance computing (HPC)
  - HPC batch computing
  - Interactive computing
- Accessing Interactive HPC Nodes
  - Launching nodes
  - Running GUIs using X11 forwarding
- Interactive Application Examples
  - Viewing Data: unix file ops (grep, awk, cat), gnuplot, NetCDF
  - Programming & Visualization Platforms: Matlab, R, Jupyter Notebooks
  - Gateways & Portals: simplify access to interactive apps
- Q&A

Thank You!

Q&A

If you have problems, please contact **consult@sdsc.edu**

https://github.com/sdsc-complecs/interactive-computing/

Resources

- GitHub Repo for this presentation:
  - https://github.com/sdsc-complecs/interactive-computing/
- SDSC Training Resources
  - https://www.sdsc.edu/education_and_training/training
  - https://github.com/sdsc-hpc-training-org/notebook-examples
  - https://github.com/mkandes/galyleo
- Expanse :
  - Landing page: expanse.sdsc.edu
  - User Guide: https://expanse.sdsc.edu/support/user_guides/expanse.html
  - Training: https://www.sdsc.edu/education_and_training/training_hpc.html
- Problems? Contact consult@sdsc.edu