Running Secure, Interactive HPC Services and Notebooks on Expanse

Presented at the SDSC 2022 HPC/CI User Training Series

February 18, 2022

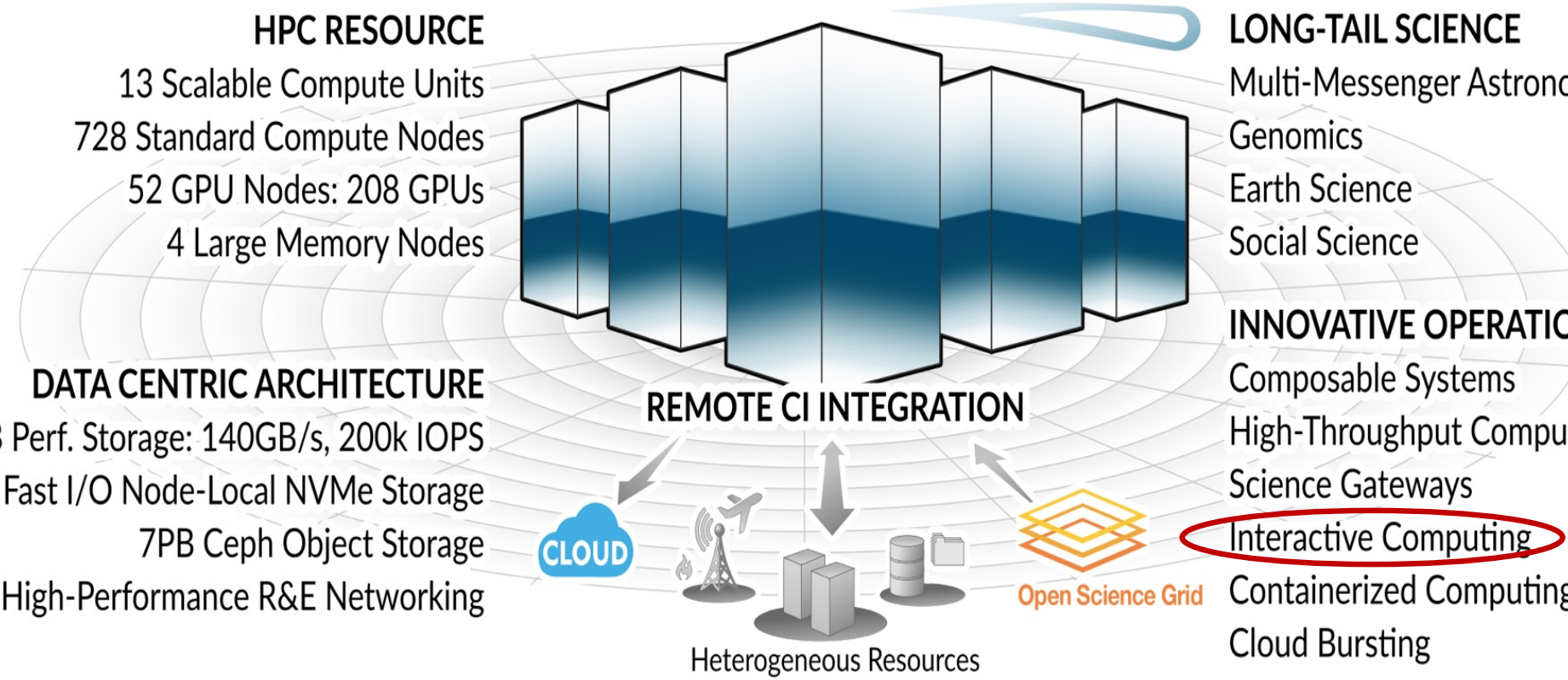
Mary Thomas
(SDSC)

EXPANSE

COMPUTING WITHOUT BOUNDARIES
5 PETAFLOP/S HPC and DATA RESOURCE

HPC RESOURCE
13 Scalable Compute Units
728 Standard Compute Nodes
52 GPU Nodes: 208 GPUs
4 Large Memory Nodes

DATA CENTRIC ARCHITECTURE
12PB Perf. Storage: 140GB/s, 200k IOPS
Fast I/O Node-Local NVMe Storage
7PB Ceph Object Storage
High-Performance R&E Networking



LONG-TAIL SCIENCE
Multi-Messenger Astronomy
Genomics
Earth Science
Social Science

INNOVATIVE OPERATIONS
Composable Systems
High-Throughput Computing
Science Gateways
Interactive Computing
Containerized Computing
Cloud Bursting

For more details see the Expanse user guide @ https://www.sdsc.edu/support/user_guides/expanse.html
and the "Introduction to Expanse" webinar @ https://www.sdsc.edu/event_items/202006_Introduction_to_Expanse.html

Outline

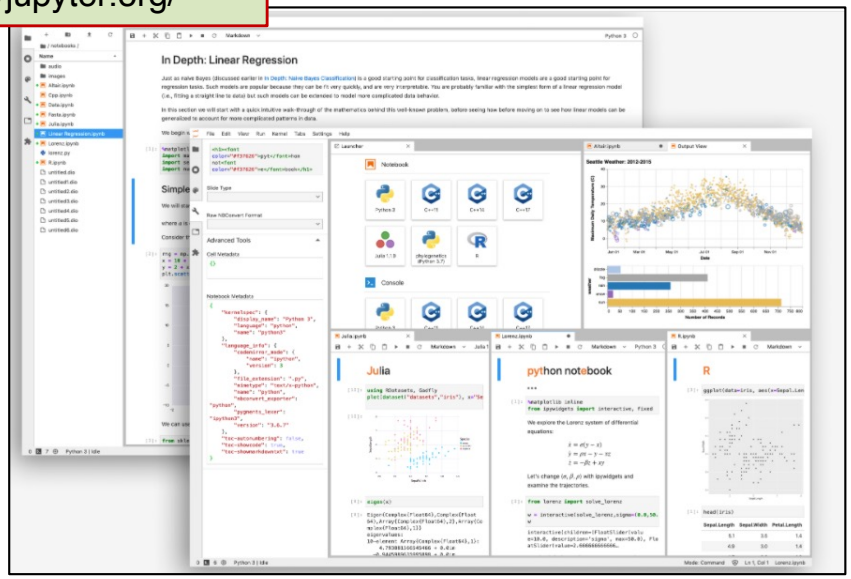
- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galyleo
- Notebook examples

What is Interactive HPC Computing

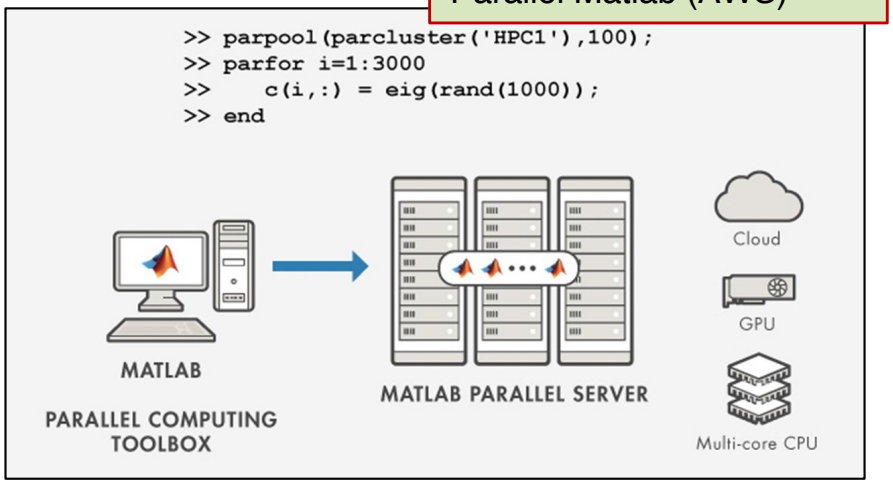
- In **computer** science, **interactive computing** refers to software which accepts input from the user as it runs.
  - **Interactive** software includes commonly used programs, such as word processors or spreadsheet applications.
- **Interactive HPC computing** involves *real-time* user inputs to 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.

Interactive HPC Computing

https://jupyter.org/

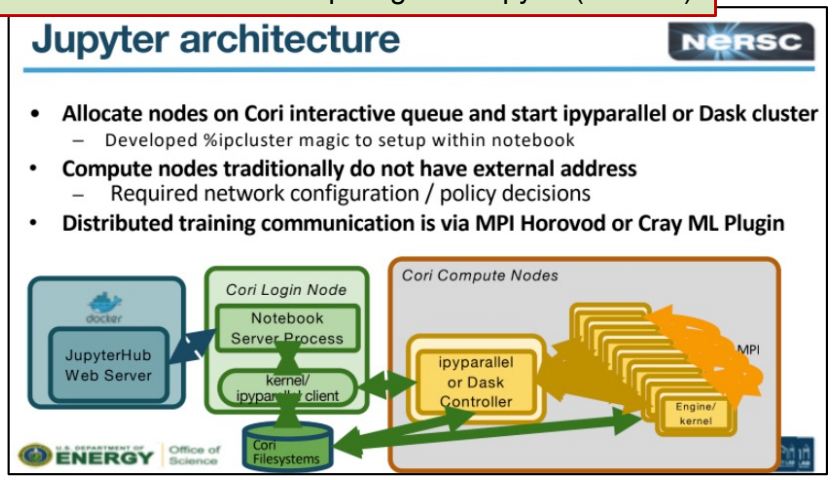


Parallel Matlab (AWS)



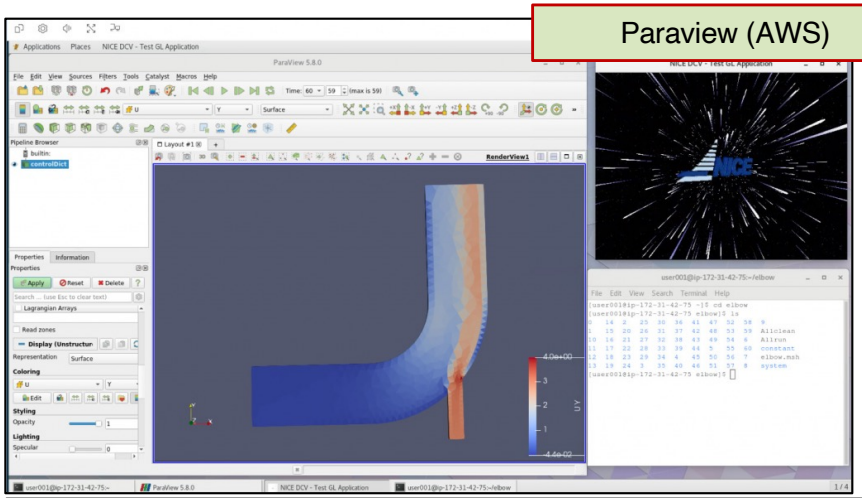
https://azuremarketplace.microsoft.com/en-us/marketplace/apps/mathworks-inc.matlab-parallel-server-listing?tab=Overview

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/

Accessing Interactive Compute Nodes on Expanse

- Connect via terminal using SSH → secure connections
- Use the *srun* command to obtain nodes for 'live,' command line interactive access:

CPU	srun --partition=debug --qos=debug-normal --pty --nodes=1 --ntasks-per-node=128 --mem=248 -t 00:30:00 --wait=0 --export=ALL --account=abc123 /bin/bash
GPU	srun --partition=gpu-shared --pty --nodes=1 --ntasks-per-node=16 --mem=374 --gpus=1 -t 00:30:00 --wait=0 --export=ALL --account=abc123 /bin/bash

- Note: you don't need an interactive node to launch secure Jupyter notebooks. Use SRPS/galyleo

Using An Interactive CPU node

```
[username@login01 openmp]$ module purge
[username@login01 openmp]$ module load slurm
[username@login01 openmp]$ module load cpu
[username@login01 openmp]$ module load gcc/10.2.0
[username@login01 openmp]$ module load openmpi/4.0.4
[username@login01 openmp]$ srun --partition=debug  --pty --account=use300--nodes=1 --ntasks-per-node=64  --mem=128G -t 00:30:00 --wait=0 --export=ALL /bin/bash
```

Request an interactive node for 30 minutes

```
[mthomas@exp-9-55 openmp]$ export OMP_NUM_THREADS=16
[mthomas@exp-9-55 openmp]$ ./hello_openmp
 HELLO FROM THREAD NUMBER =         0
 HELLO FROM THREAD NUMBER =         8
 HELLO FROM THREAD NUMBER =         9
 HELLO FROM THREAD NUMBER =        10
 HELLO FROM THREAD NUMBER =        11
 HELLO FROM THREAD NUMBER =        12
 HELLO FROM THREAD NUMBER =        13
 HELLO FROM THREAD NUMBER =        14
 HELLO FROM THREAD NUMBER =        15
 HELLO FROM THREAD NUMBER =         4
 HELLO FROM THREAD NUMBER =         7
 HELLO FROM THREAD NUMBER =         6
 HELLO FROM THREAD NUMBER =         5
 HELLO FROM THREAD NUMBER =         3
 HELLO FROM THREAD NUMBER =         2
 HELLO FROM THREAD NUMBER =         1
```

- Exit interactive session when your work is done or you will be charged 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 An Interactive GPU node

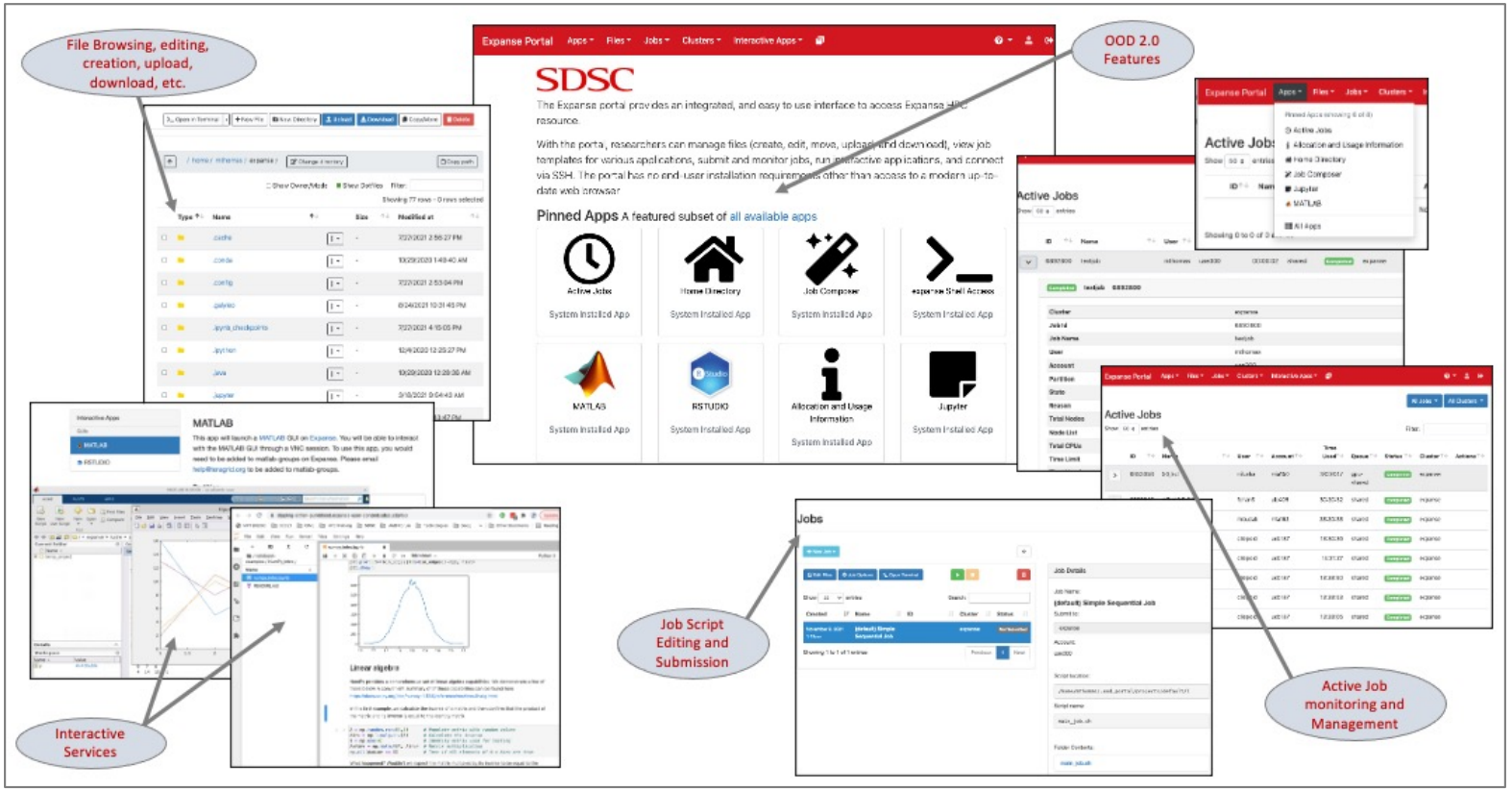
Request an interactive node for 30 minutes

```
[snip]
Last login: Fri Feb 18 12:58:32 2022 from 76.176.117.51
[username @login02 ~]$
[username @login02 ~]$ srun --partition=gpu-shared --pty  --nodes=1  --ntasks-per-node=16 --
mem=374   --gpus=1  -t 00:30:00 --wait=0 --export=ALL --account=use300  /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

Expanse User Portal



- https://portal.expanse.sdsc.edu
- Access using XSEDE credentials
- Securely hosts batch job submission & monitoring, and interactive applications

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

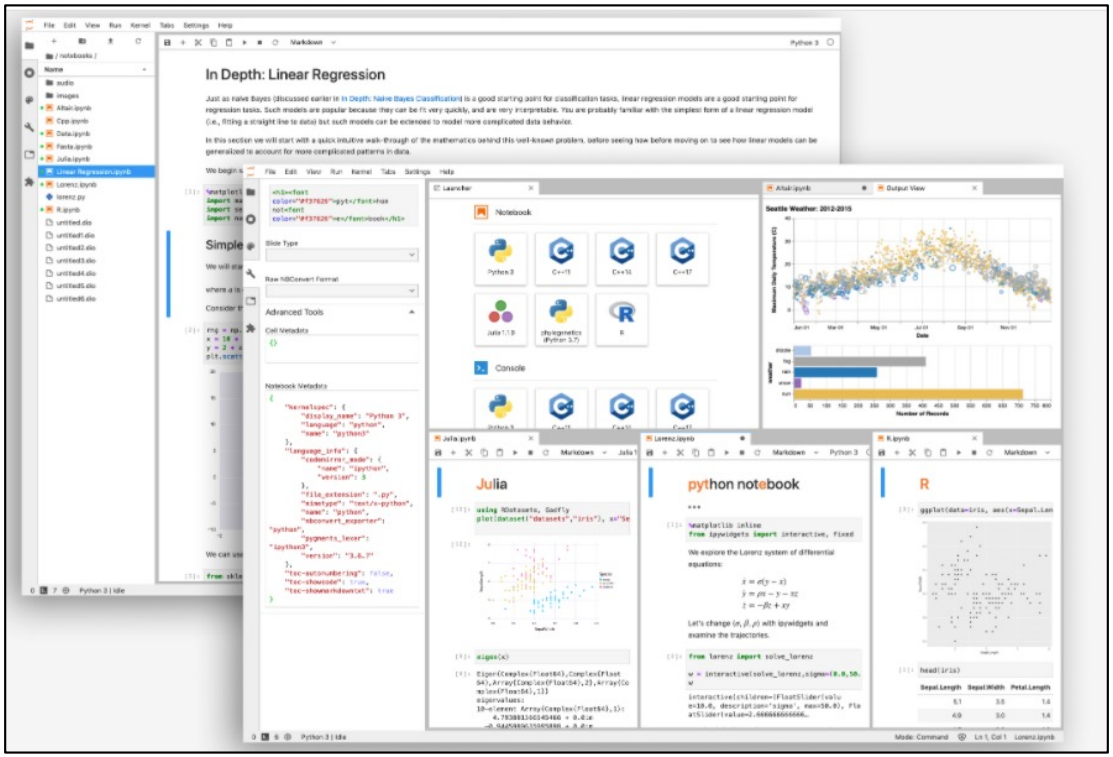
Juptyer Notebooks

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



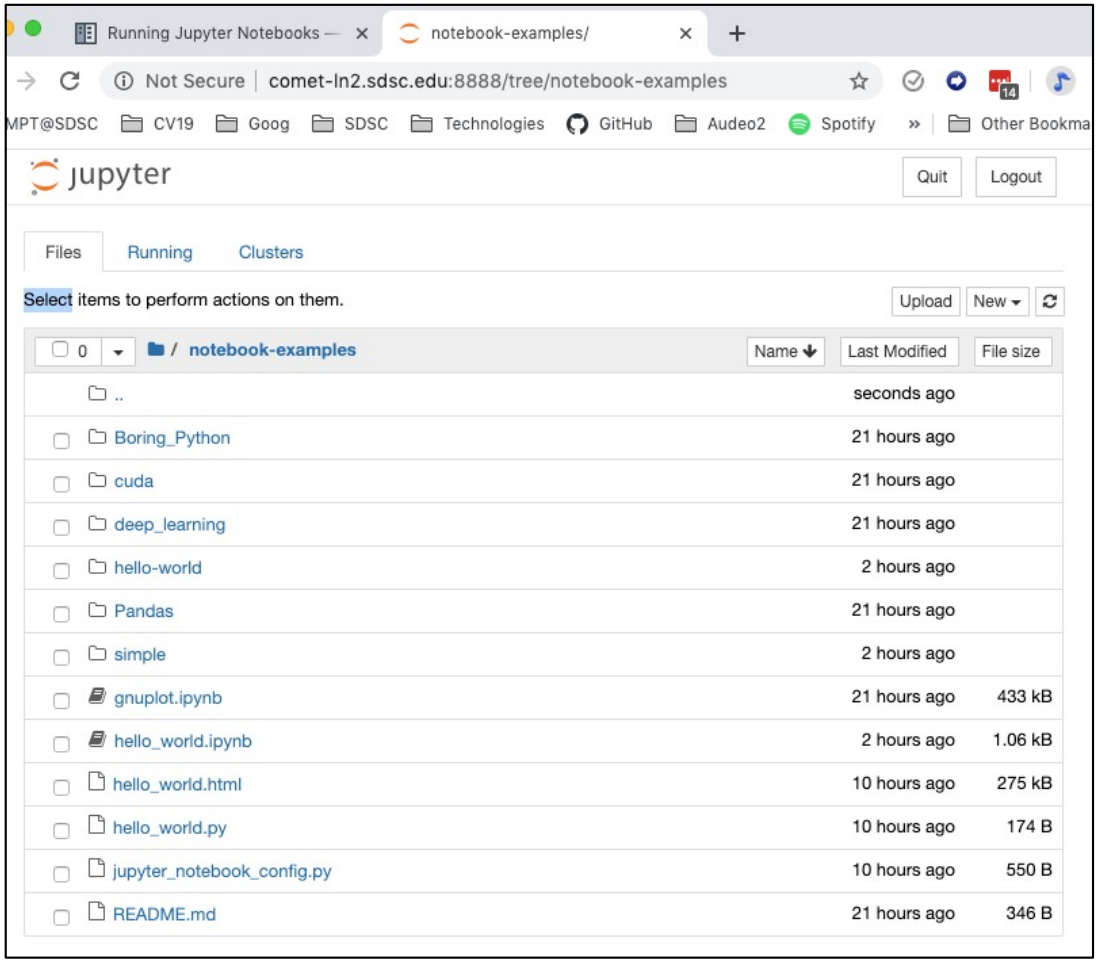
Interactive Services have a Key Vulnerability: They Provide Access to HPC File Systems, often over HTTP

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



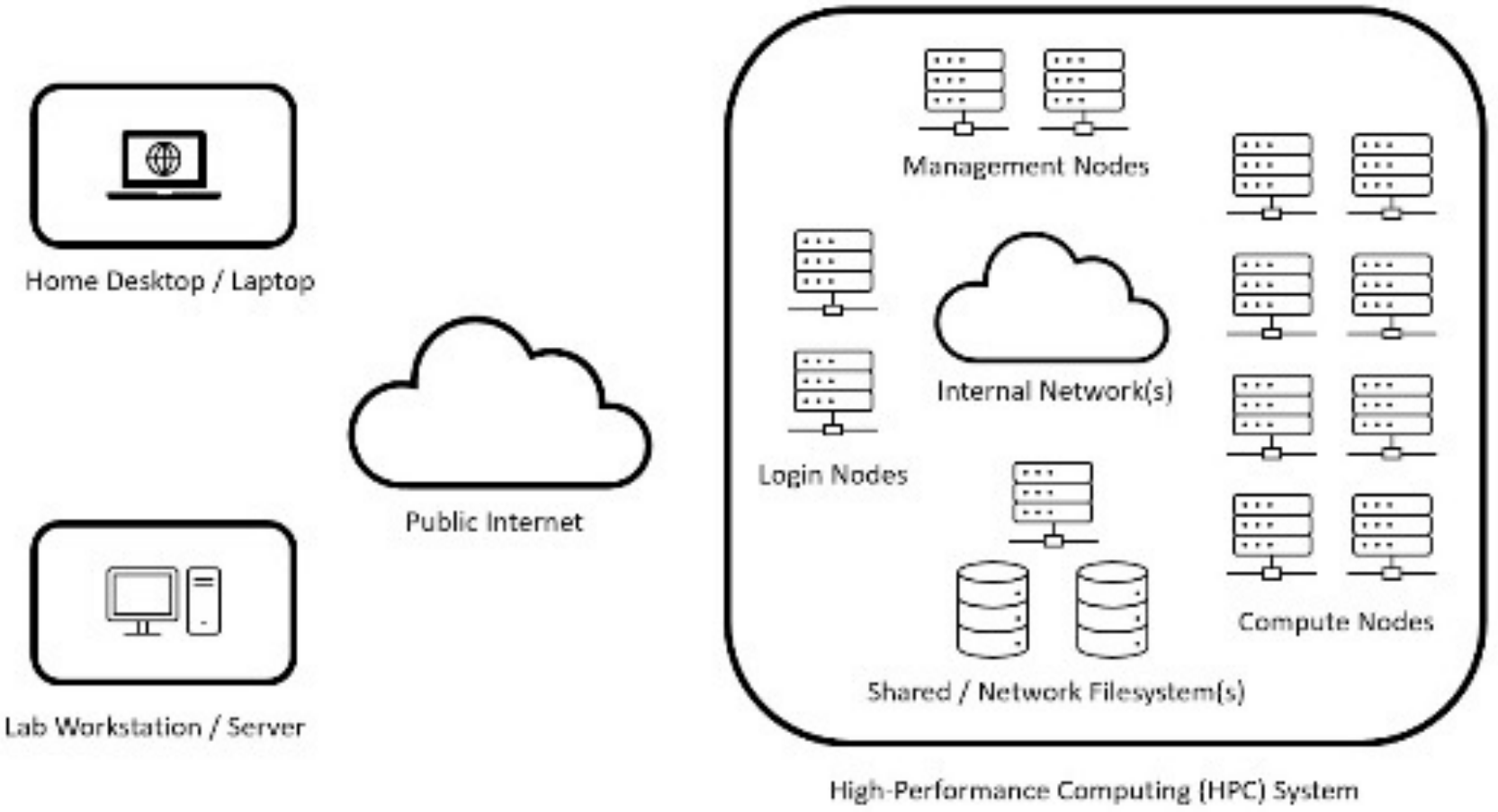
Outline

- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galyleo
- Notebook examples

Accessing and Running Secure Notebooks on HPC Systems

- Install notebook application:
  - Locally: install Anaconda on your laptop
  - Remotely:
    - Install Anaconda/conda on the remote machine (default is HTTP) – not recommended
    - Launch securely (HTTPS) using SRPS/`galyleo` -- *`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:
  - CPU and GPUs
  - Interactive nodes: command line
  - Slurm batch script
- Treat the Notebook URL like a Password!

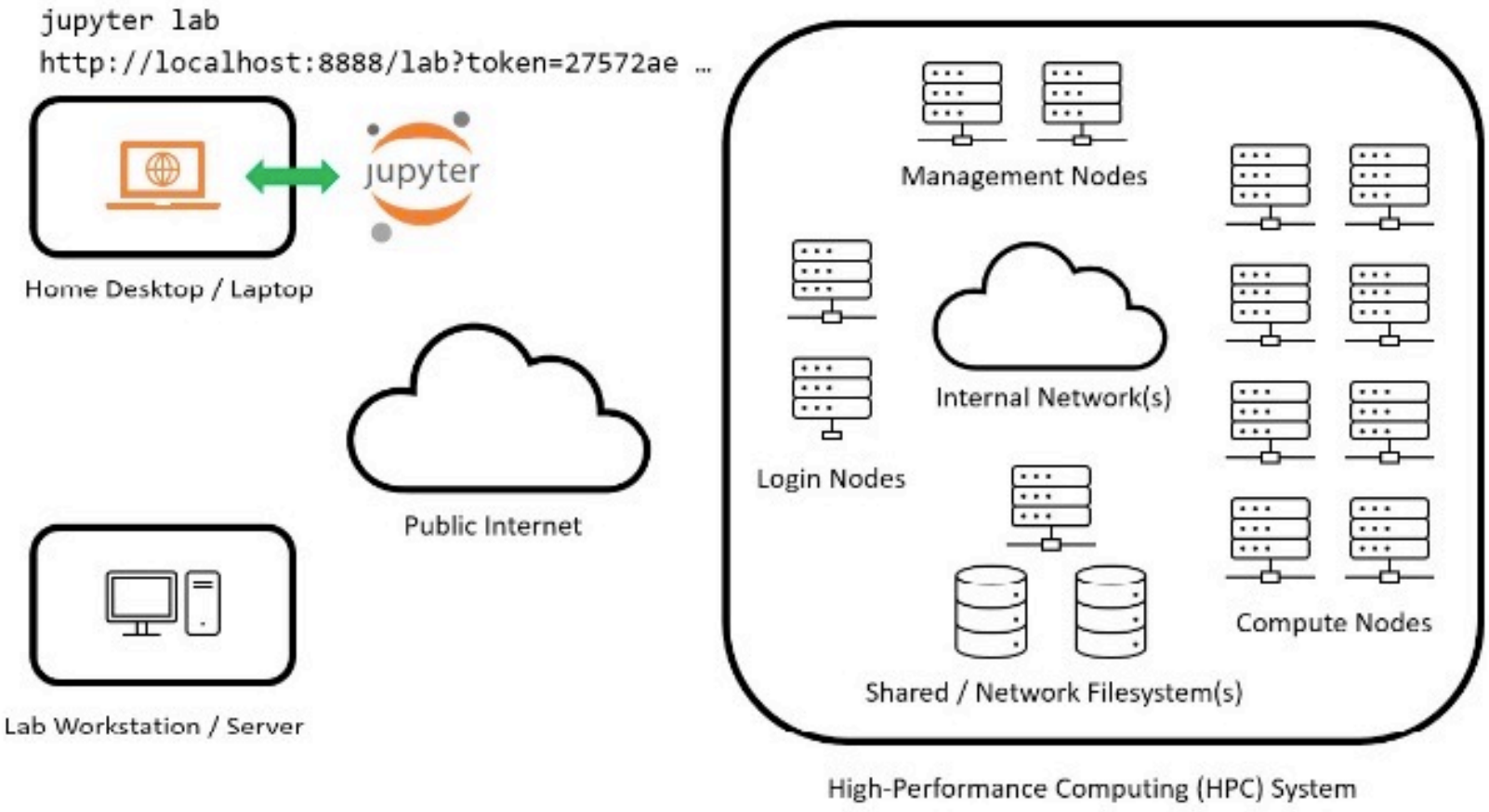
Standard HPC Ecosystem



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

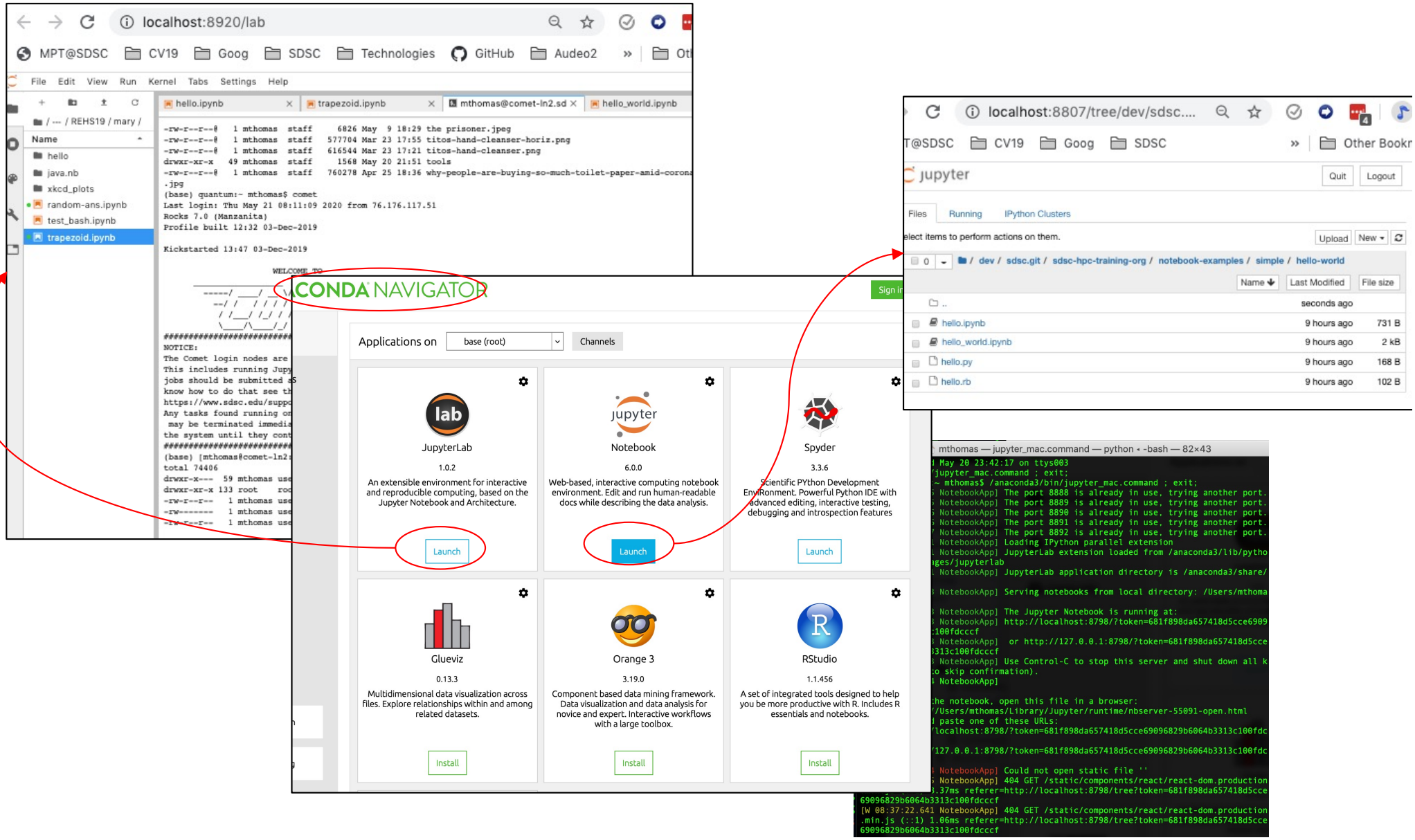
Running Notebooks Locally Over HTTP

Install the Anaconda application and use it to launch your notebooks



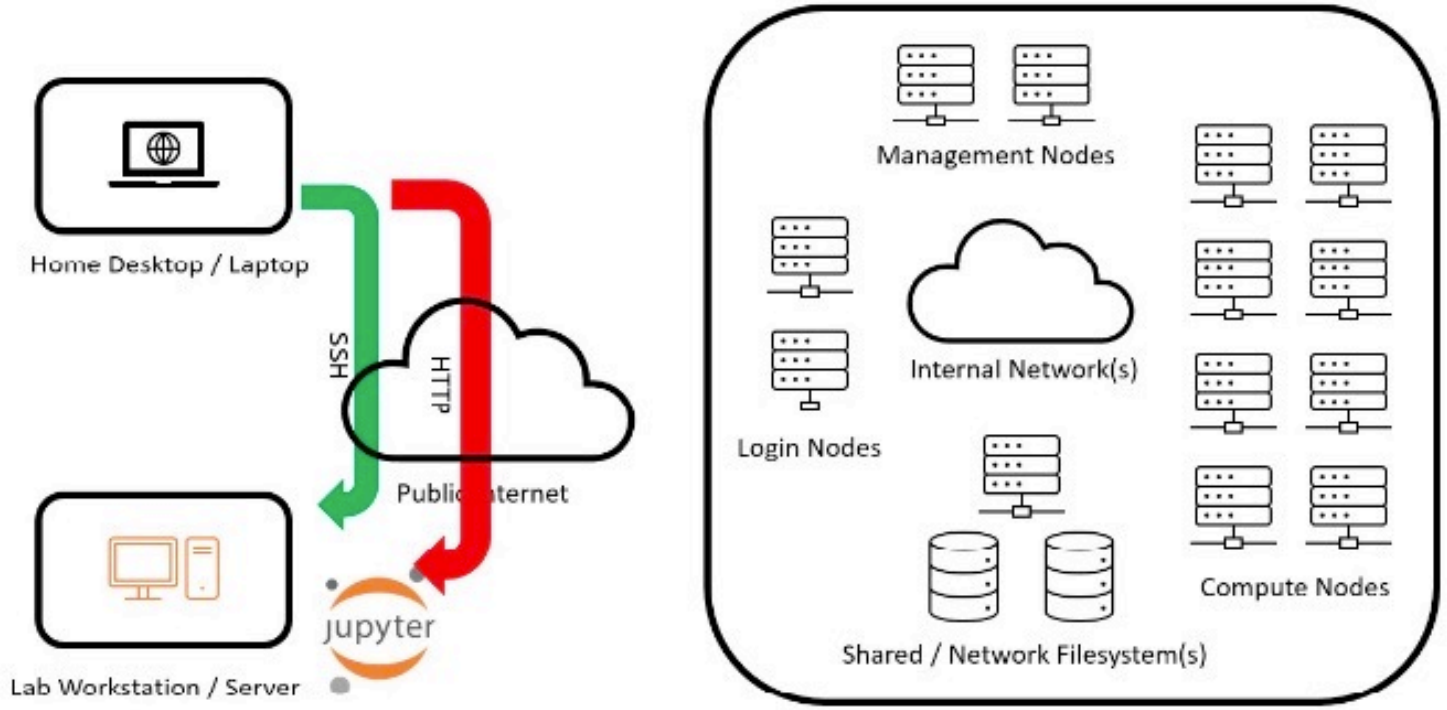
Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Anaconda on Local Host or Laptop



Running Notebooks Remotely Over HTTP

- Messages over HTTP
- Not encrypted
- Vulnerable to hacking

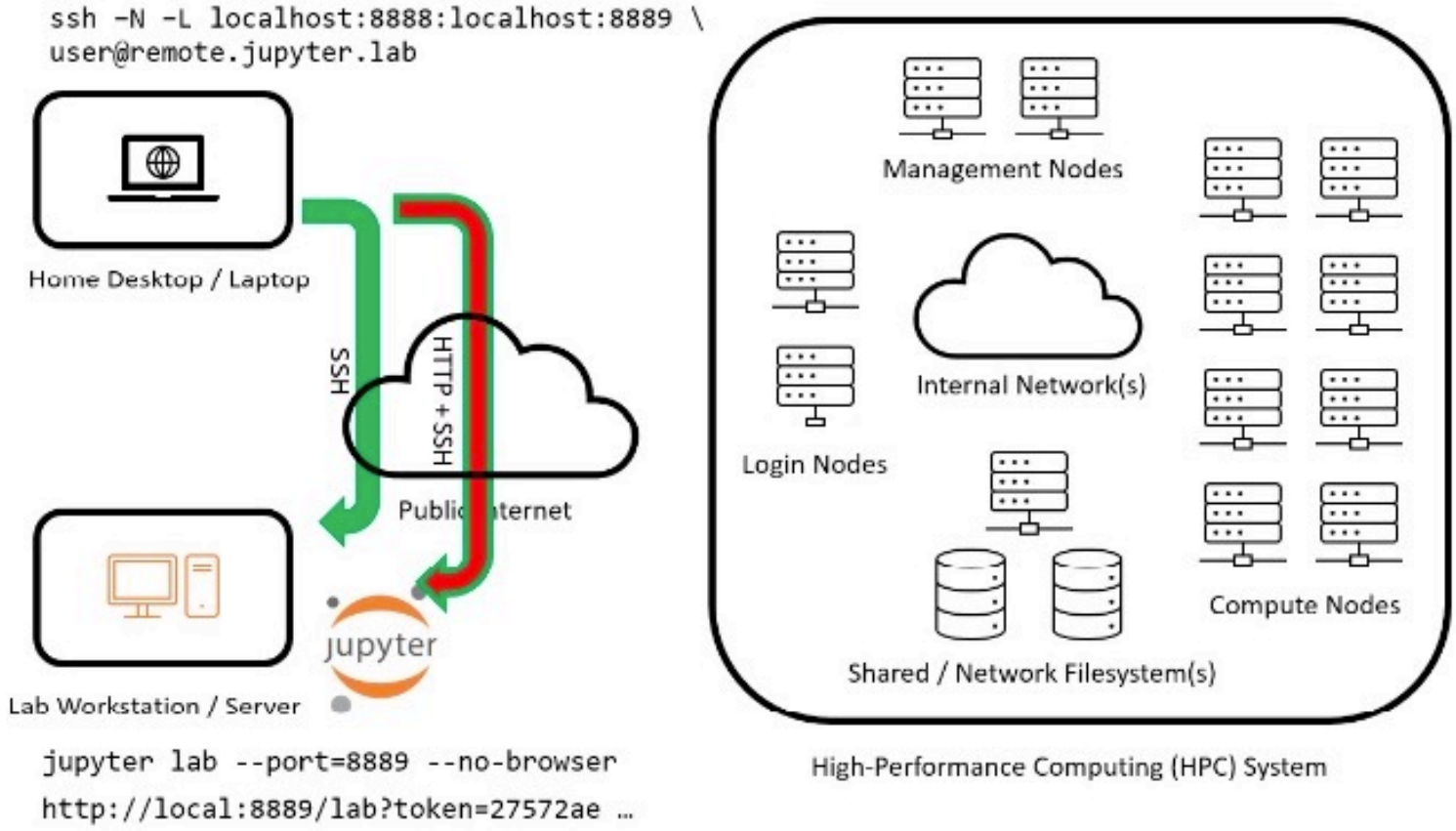



```
jupyter lab --ip="$(hostname)" --no-browser
http://remote.jupyter.lab:8888/lab?token=27572ae …
```

Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Running Notebooks Remotely Over HTTP Using SSH Tunneling

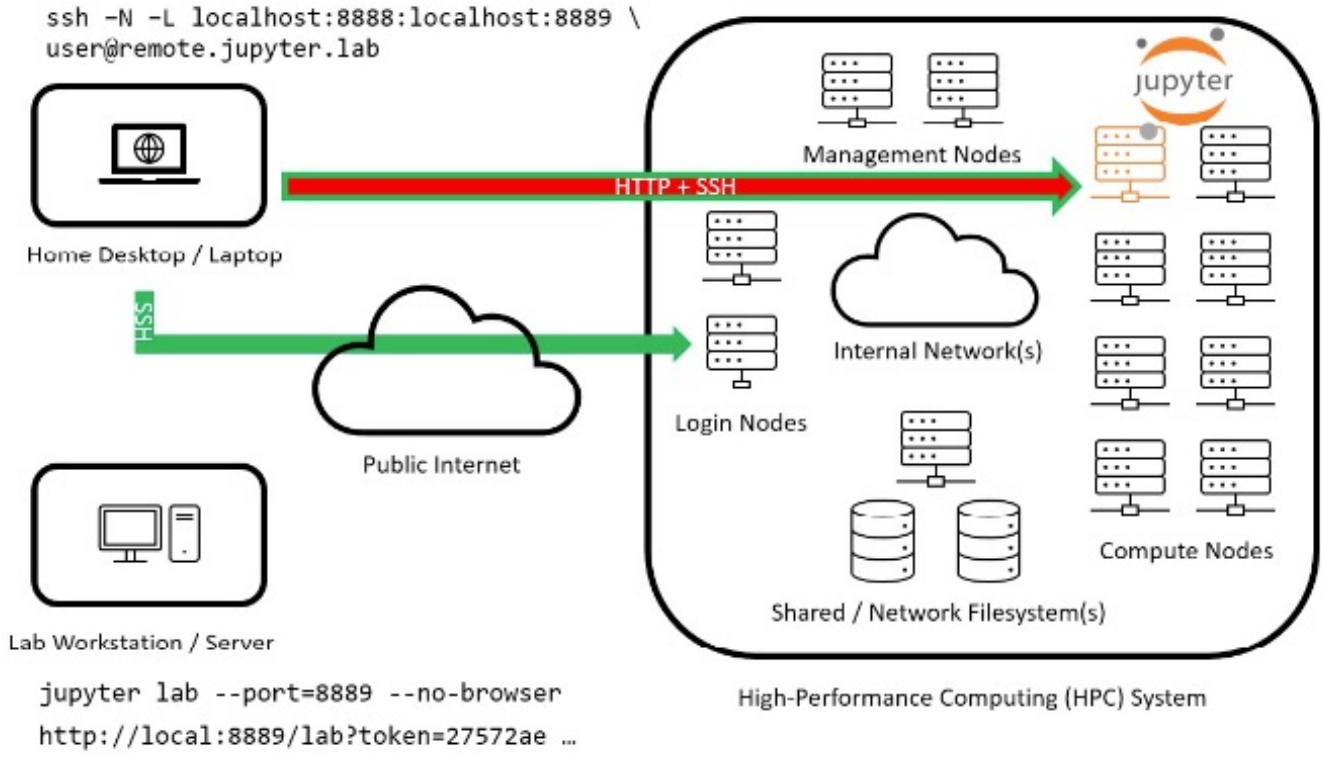
- Notebooks running on Remote host
- Messages over HTTP, but sent via secure tunnel



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Running Notebooks Remotely Over HTTP Using SSH Tunneling on HPC System

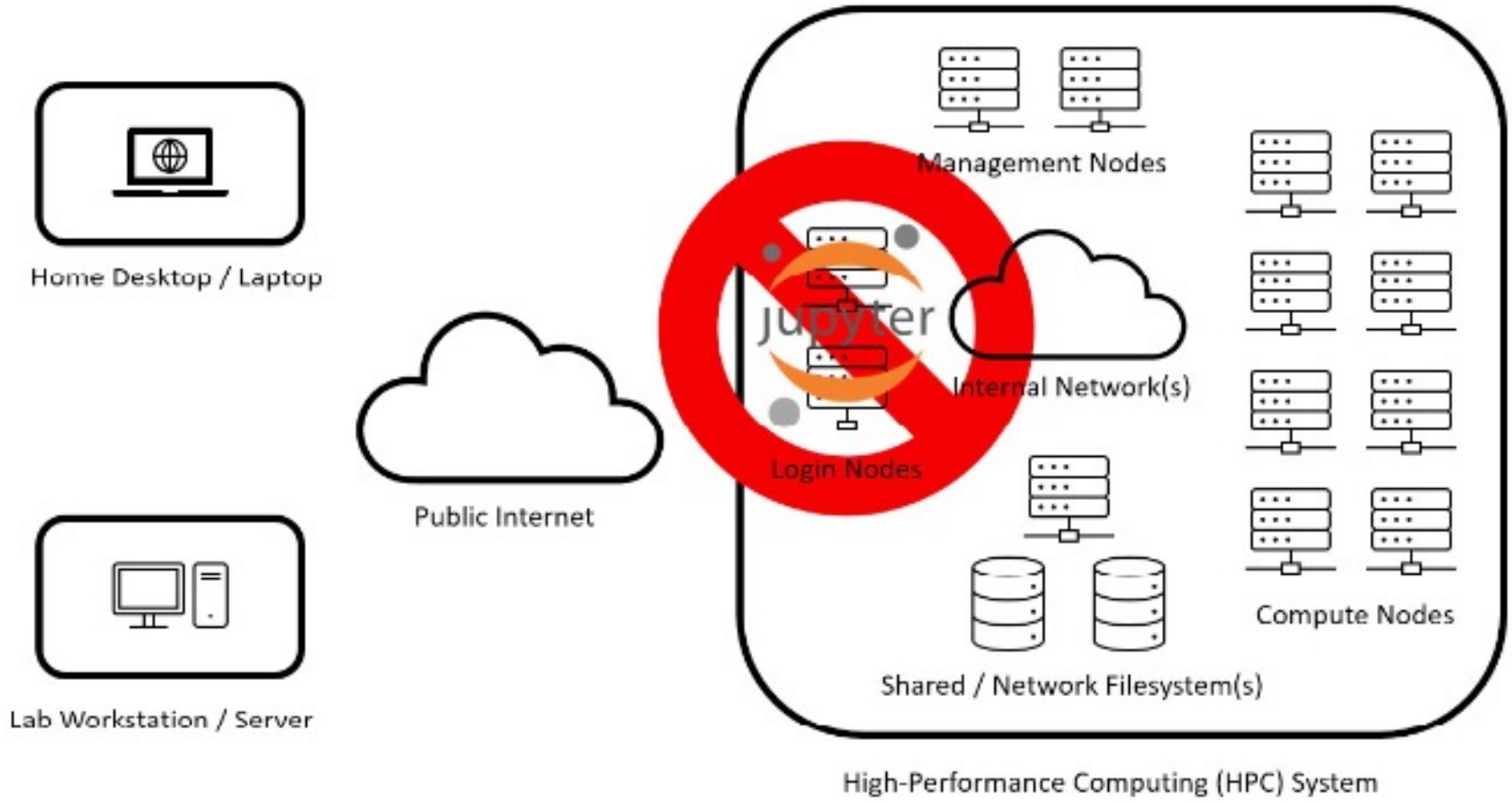
- Notebook running on HPC System
- Messages over HTTP, but passing via secure tunnel
- But do not run on compute nodes!



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Warning

Do not run notebooks/interactive services on the login nodes!



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Warning:

The Jupyter Notebook URL is essentially a password

Treat it very carefully
Do NOT Share

Outline

- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galyleo
- Notebook examples

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.

SRPS Components

Just two:

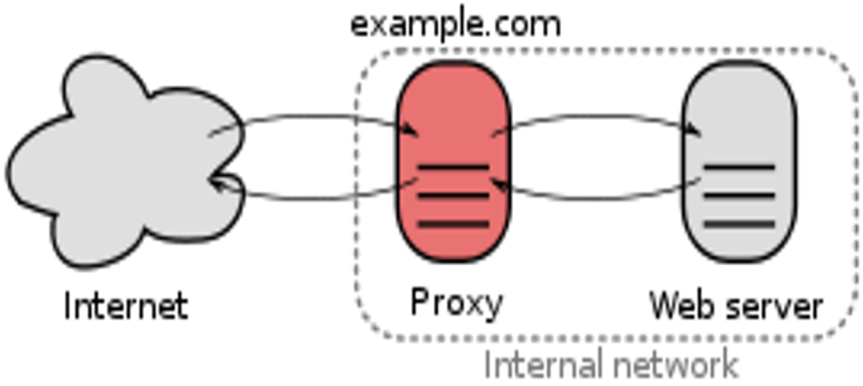
- Satellite: a self-service HTTP(s) reverse-proxy.
- Satellite Client: a shell-based utility to orchestrate a user's interaction with both Satellite and Slurm to start a Jupyter session within a batch job.

GitHub Repo:

- https://github.com/sdsc-hpc-training-org/satellite

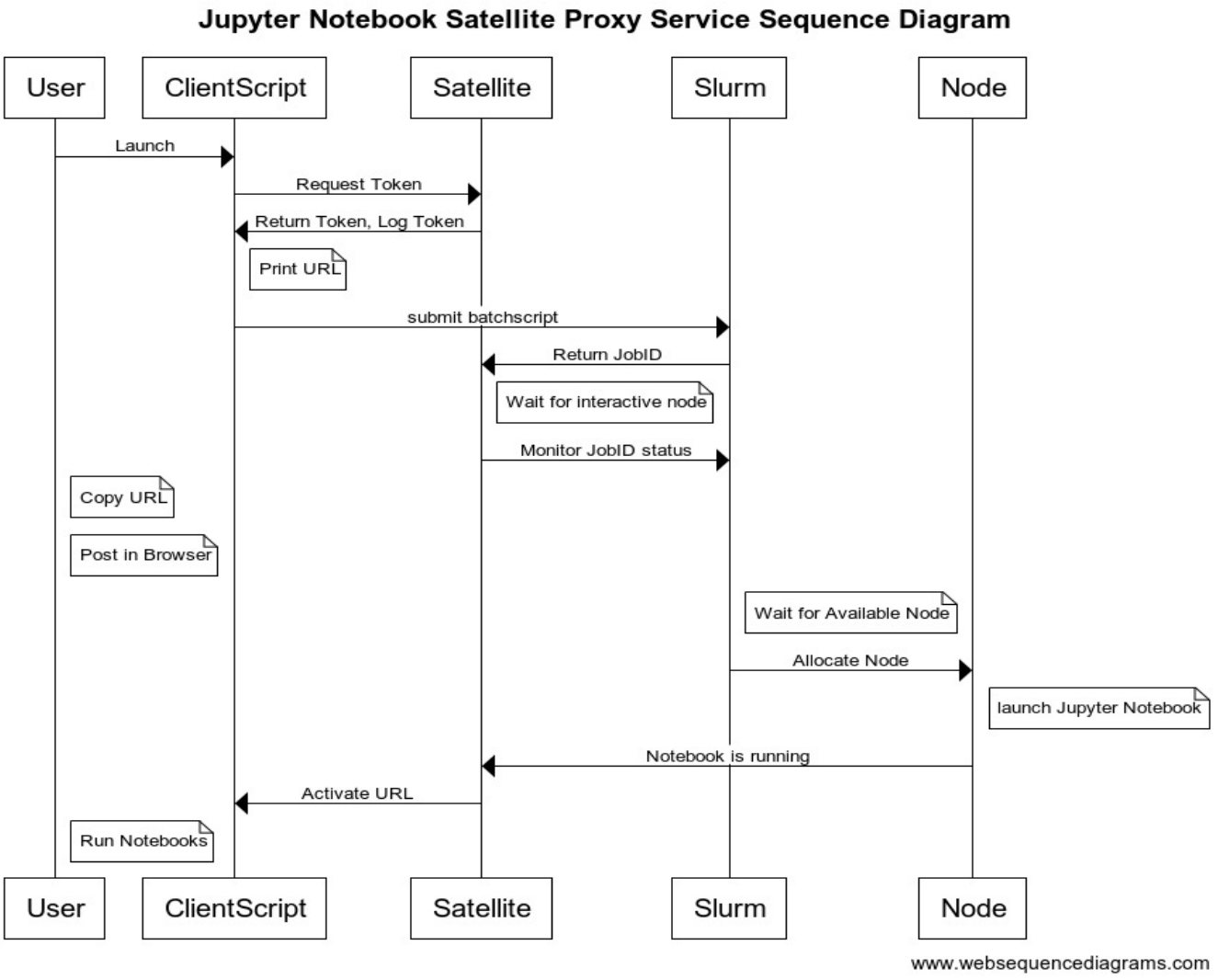
What is a Reverse Proxy?

A reverse proxy takes requests from the Internet and forwards them to servers in an internal network. Those making requests to the proxy may not be aware of the internal network.

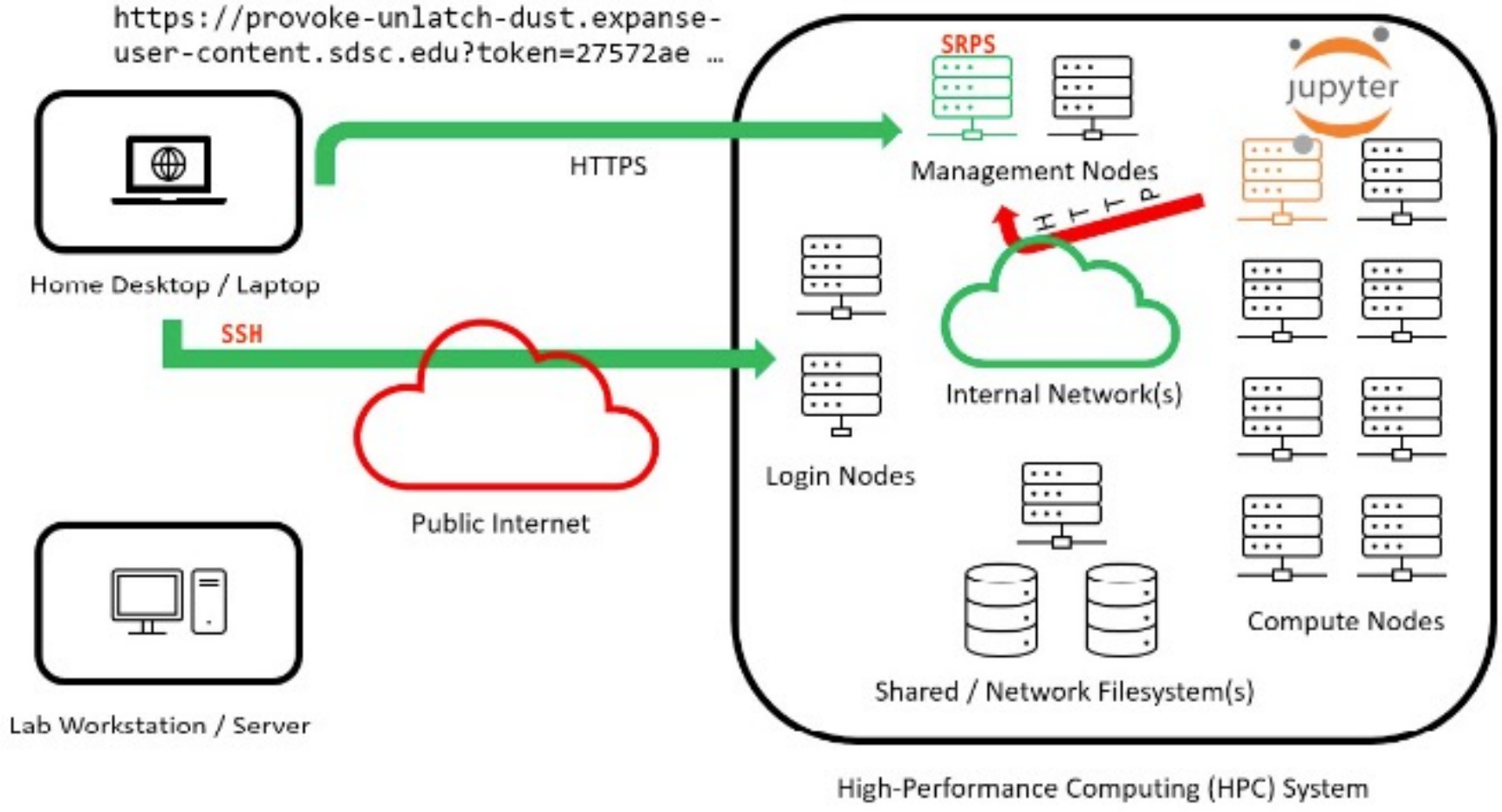


Img Source: Wikipedia reverse proxy

Satellite Life Cycle



Running Notebooks Remotely Using SRPS



Src: M Kandes: https://education.sdsc.edu/training/interactive/202112_running_jupyter_notebooks_on_expanse/

Outline

- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galyleo
- Notebook examples

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:

    - 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 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 ~]$ 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

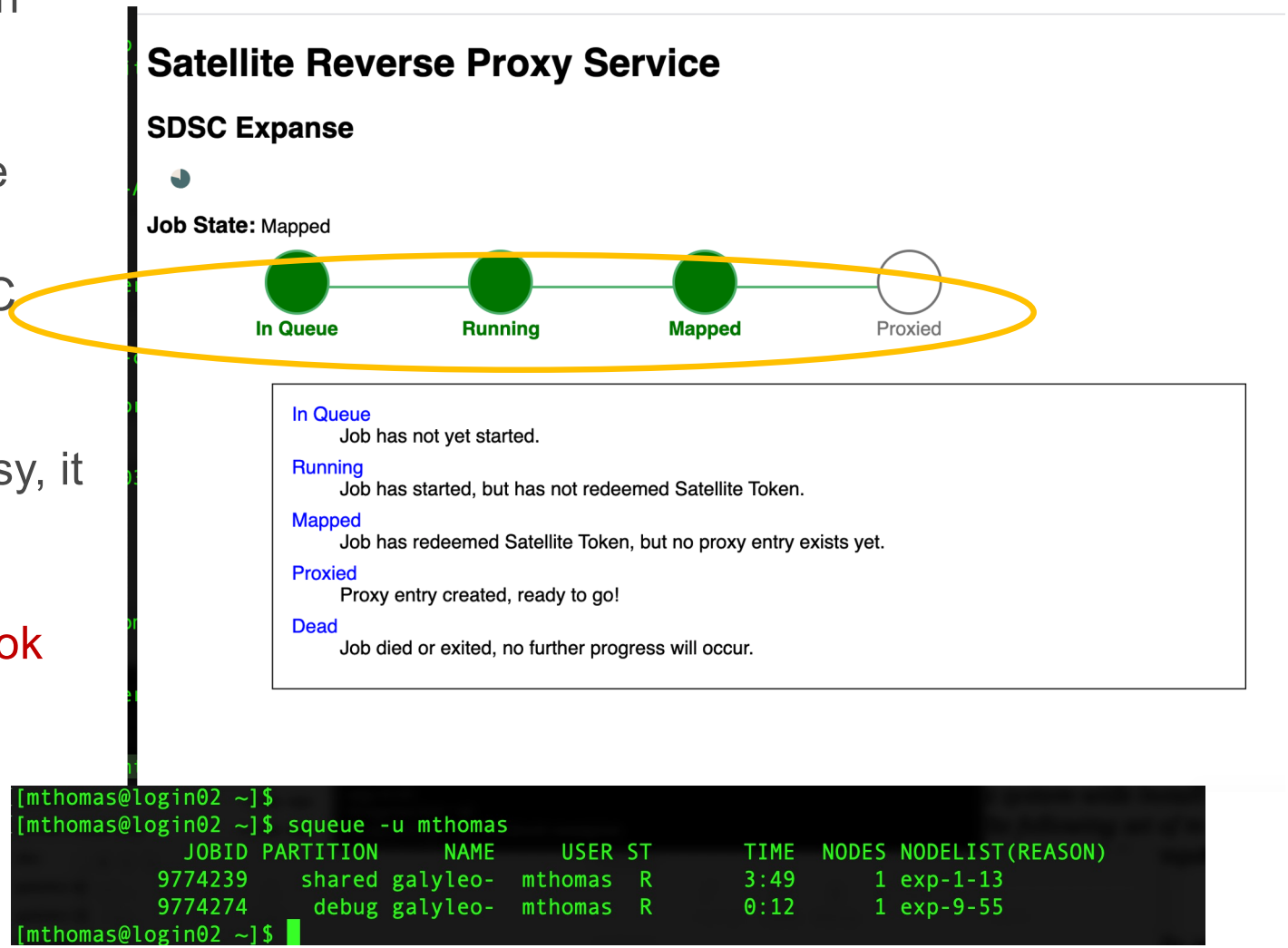
Step 3:
Copy URL; paste into browser on local system

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

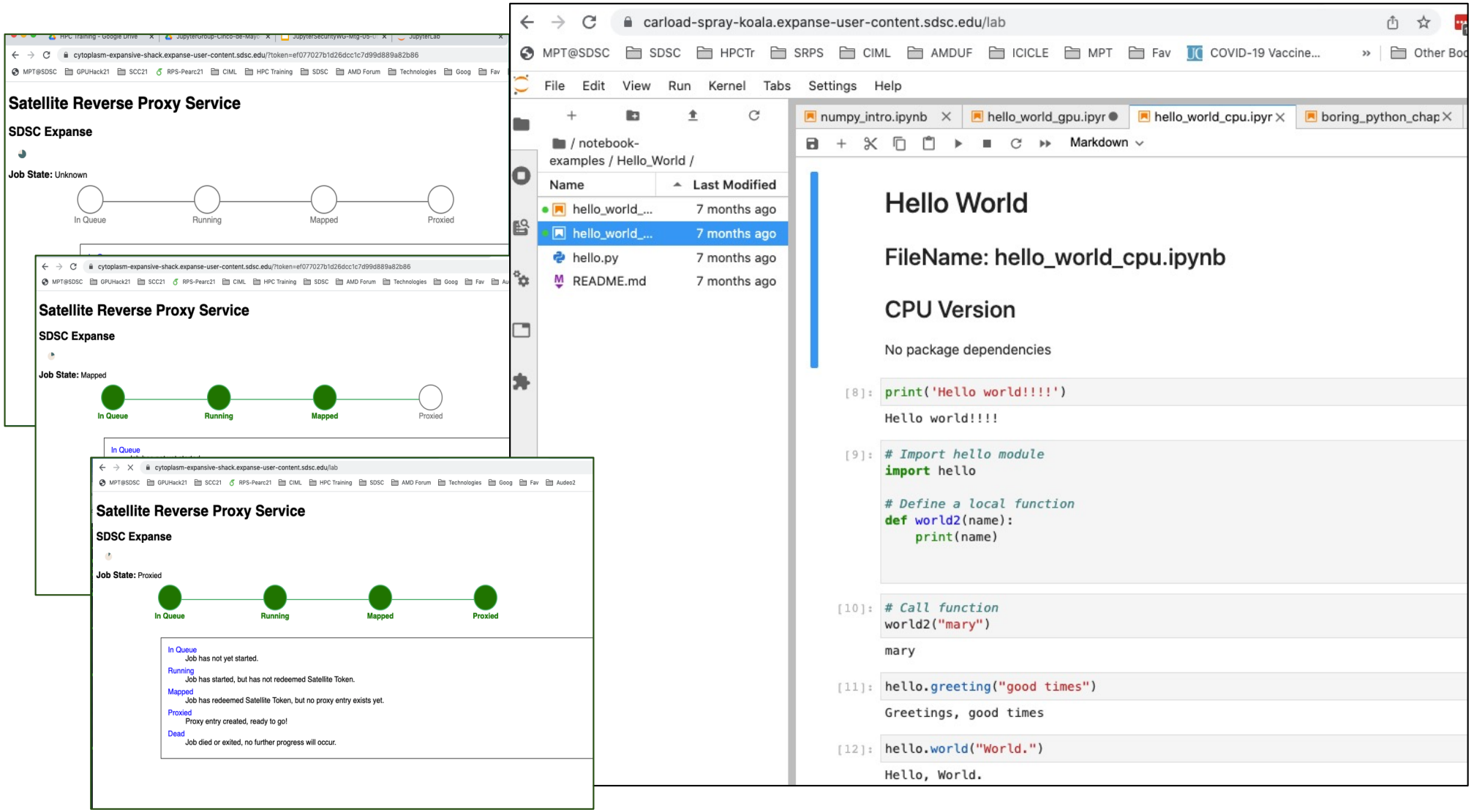
```
[username@login01 ~]$ which galyleo.sh
/usr/bin/which: no galyleo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-
zen/gcc-8.3.1
[SNIP]
home/username/.local/bin:/home/username/bin)
[username@login01 ~]$ export PATH="/cm/shared/apps/sdsc/galyleo:${PATH}"
[username@login01 ~]$ which galyleo
/cm/shared/apps/sdsc/galyleo/galyleo

[username@login01 ~]$ galyleo launch --account use300 --partition shared --cpus 1 --
memory 2 --time-limit 00:30:00 --env-modules cpu,gcc,anaconda3
[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 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!



Satellite Pending Page



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

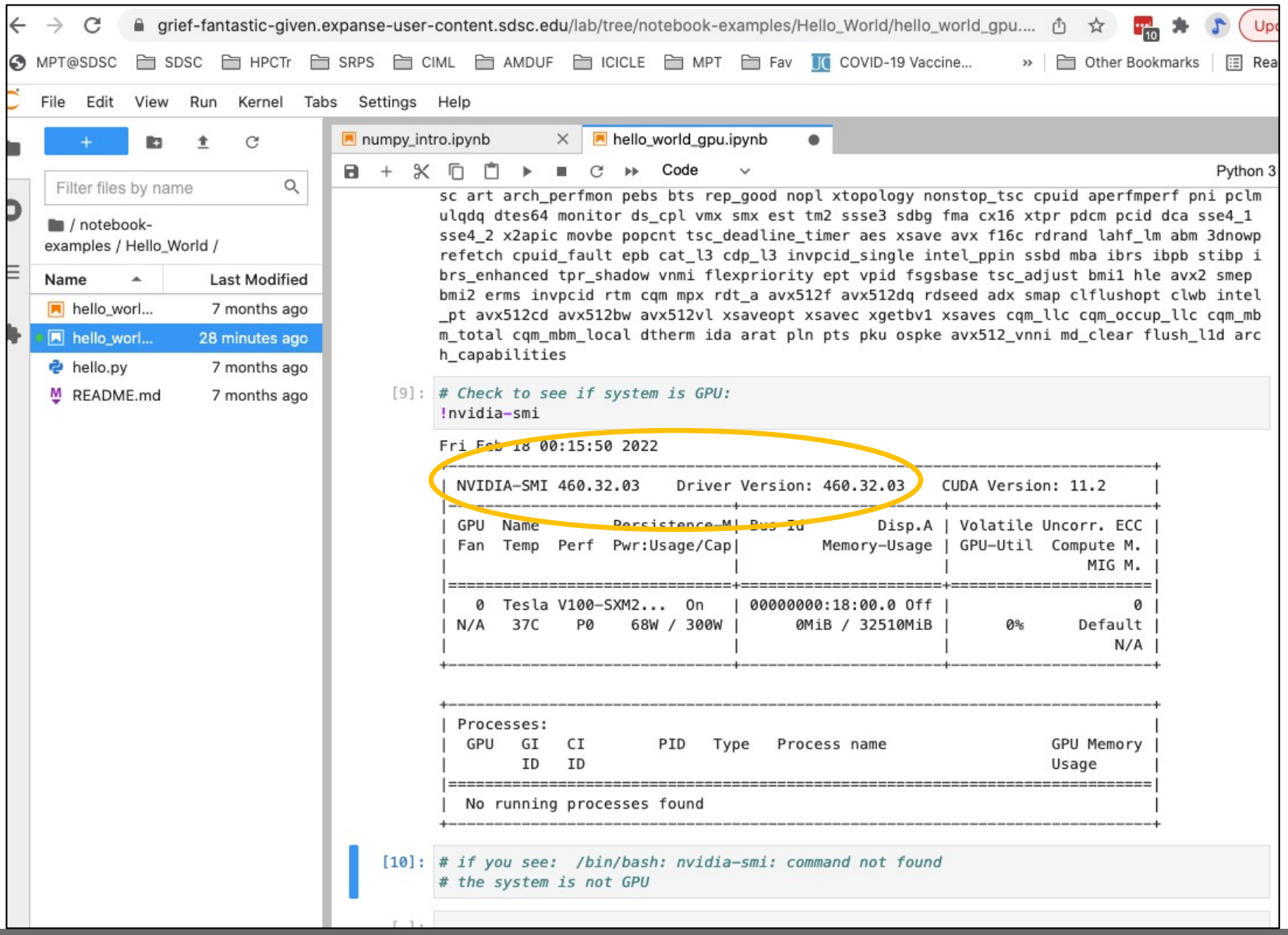
Step 2:
Run command to launch secure notebook

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

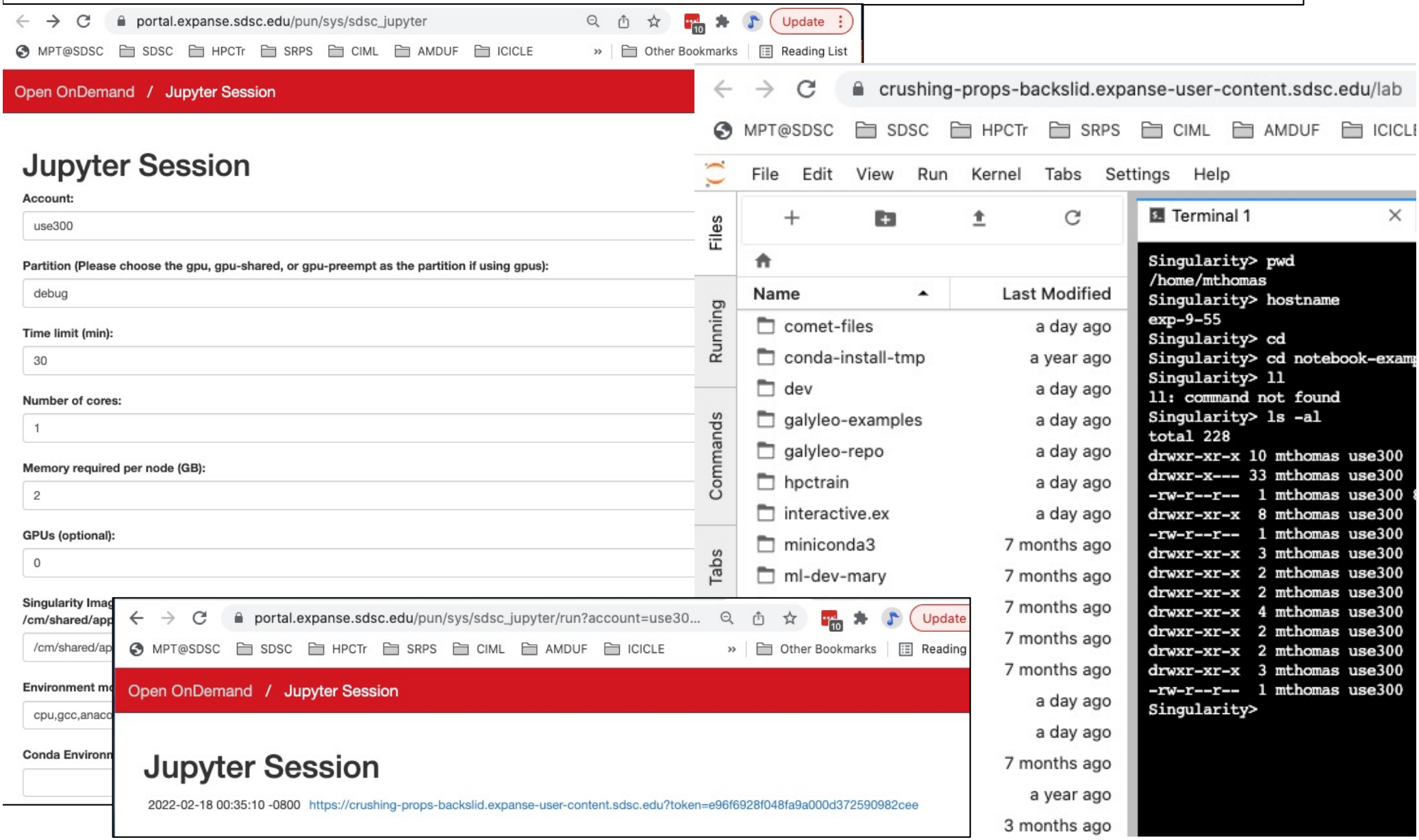
Step 4:
Monitor Slurm queue; wait for job to start

```
[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
10 --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
```

Notebook is launched on GPU device



Using Expanse Portal to Launch Notebooks



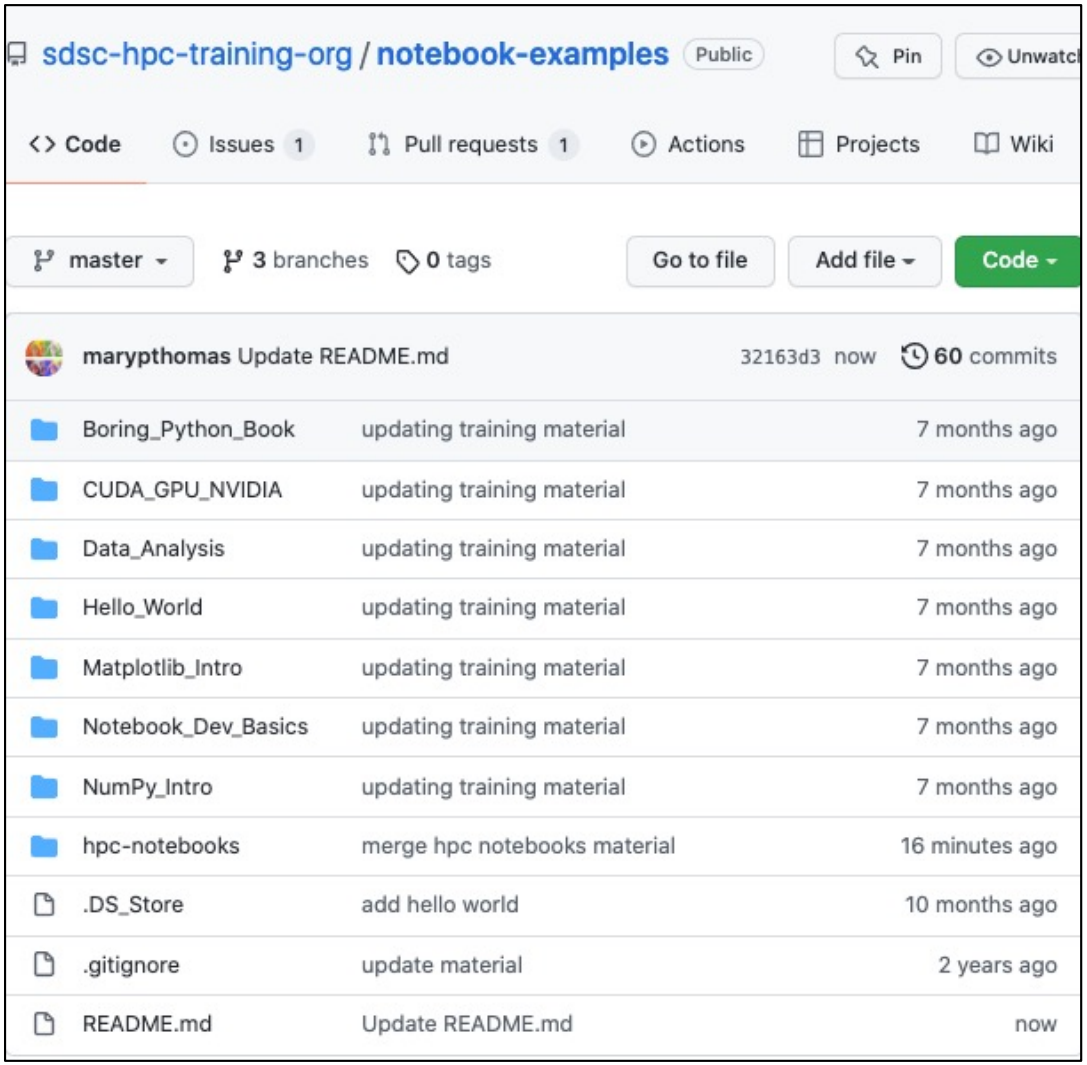
Outline

- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galyleo
- Notebook examples

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



Install Anaconda/conda to Configure Custom Environments

```
quantum:~ username$ ssh username@login.expanse.sdsc.edu
Welcome to Bright release         9.0
                                                    Based on CentOS Linux 8
                                                              ID: #000002
-------------------------------------------------------------------------------
                             WELCOME TO

              _______  __ ____  ___     _   _______ ______
             / ____/ |/ // __ \/   |  / | / / ___// ____/
            / __/  |   // /_/ / /| | /  |/ /\__ \/ __/
           / /___ /   |/ ____/ ___ |/ /|  /___/ / /___
          /_____//_/|_/_/   /_/  |_/_/ |_//____/_____/
-------------------------------------------------------------------------------
Use the following commands to adjust your environment:
'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login
-------------------------------------------------------------------------------
Last login: Fri Feb 18 10:19:37 2022 from 76.176.117.51
[username@login02 ~]$ which conda
/usr/bin/which: no conda in
(/cm/shared/apps/sdsc/galyleo:/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-
8.3.1/intel-19.1.1.217-[snip]

[username@login02 ~]$ module load sdsc cpu anaconda3

[username@login02 ~]$ which conda
/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1/anaconda3-2020.11-
da3i7hmt6bdqbmuzq6pyt7kbm47wyrjp/bin/conda

[username@login02 ~]$
```

Activate your Conda environment

- Miniconda installation recommends that you put these lines into your .bashrc script, however this can take a long time and slow down login process

```
### Activate Conda environment
# . /home/username/miniconda3/etc/profile.d/conda.sh
# conda activate    # commented out by conda initialize
### move conda install info to separate file
```

  - If you don't need conda every time you login, run the activation from the command line

```
[username@login02 ~]$ which jupyter
/usr/bin/which: no jupyter in
(/home/username/miniconda3/bin/conda:/usr/local/bin:/usr/bin:/usr/local/sbin:/us
r/sbin:/opt/dell/srvadmin/bin:/home/username/.local/bin:/home/username/bin)
[username@login01 reverse-proxy]$ conda activate
-bash: conda: command not found
[username@login01 reverse-proxy]$ .
/home/username/miniconda3/etc/profile.d/conda.sh
[username@login01 reverse-proxy]$ conda activate
(base) [username@login01 reverse-proxy]$ which jupyter
~/miniconda3/bin/jupyter
```

Status & Future Plans

- SRPS runs on all SDSC HPC (CPU, GPU) systems:
  - SRPS server is available for download if you want to test it. This is in "beta" development. See:
    - https://github.com/sdsc-hpc-training-org/satellite-expanse
- galyeo/SRPS have been integrated into the Expanse user portal as an interactive service (notebooks, jupyterlab).
- Future plans include:
  - Test SRPS deployment at other organizations
  - Moving to using yml for custom environment configuration
  - Adding richer error handling features
  - Generalize to support other X11/Web based services
  - Development of new clients for other services

Resources

- Expanse :
  - Landing page: expanse.sdsc.edu
  - User Guide: https://expanse.sdsc.edu/support/user_guides/expanse.html
- GitHub Repo for Satellite and Galyleo:
  - https://github.com/sdsc-hpc-training-org/satellite
  - https://github.com/mkandes/galyleo
- SDSC Training Resources
  - https://www.sdsc.edu/education_and_training/training
  - https://github.com/sdsc-hpc-training-org
  - https://github.com/sdsc-hpc-training-org/notebook-examples
- XSEDE Training Resources
  - https://www.xsede.org/for-users/training
  - https://cvw.cac.cornell.edu/expanse/
- Problems? Contact help@xsede.org

Thank You!

if you have problems, please contact help@xsede.org