

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

EXPANSE

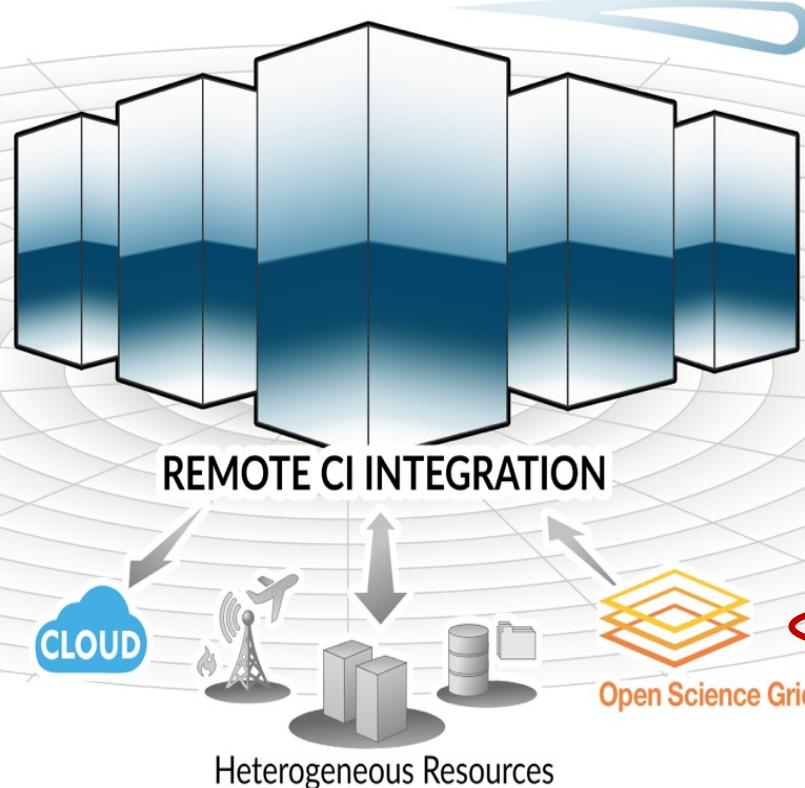
COMPUTING WITHOUT BOUNDARIES
5 PETAFLOP/S HPC and DATA RESOURCE

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

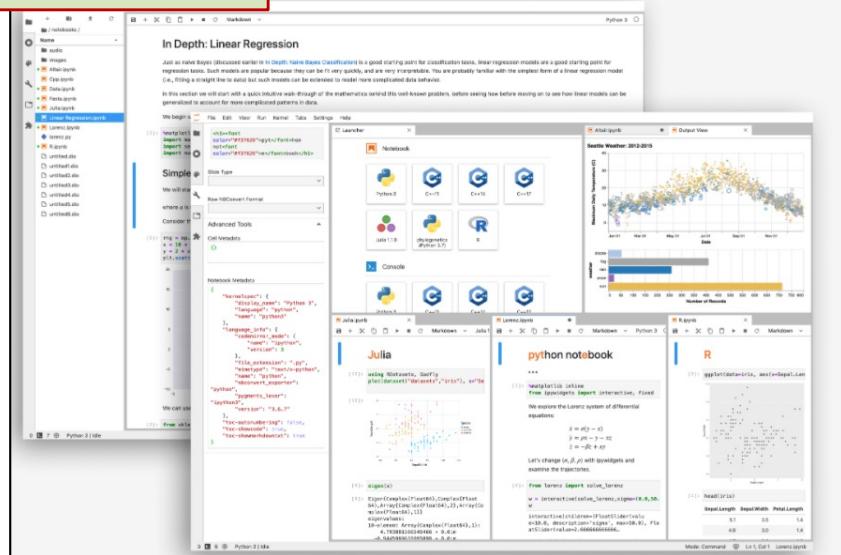
- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galileo
- Notebook examples

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

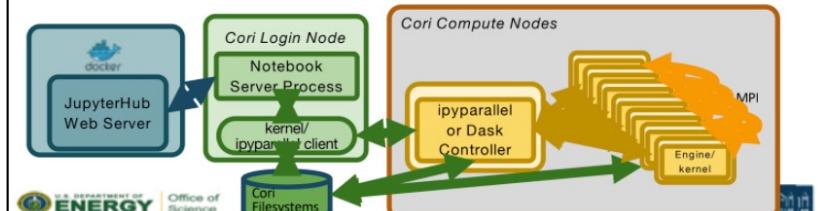
<https://jupyter.org/>



Interactive Distributed Computing with Jupyter (NERSC)

Jupyter architecture

- Allocate nodes on Cori interactive queue and start ipyparallel or Dask cluster
 - Developed %ipcluster magic to setup within notebook
- Compute nodes traditionally do not have external address
 - Required network configuration / policy decisions
- Distributed training communication is via MPI Horovod or Cray ML Plugin

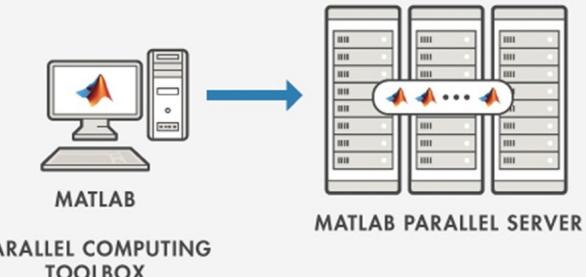


<https://drive.google.com/file/d/1-OFjrk1q3L1d3uakr2xkozrPn2c2VZpZ/view>

SDSC SAN DIEGO SUPERCOMPUTER CENTER

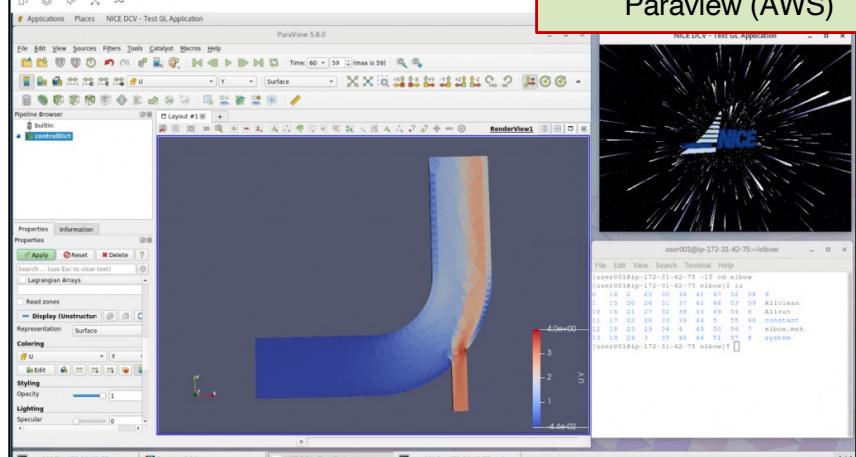
Parallel Matlab (AWS)

```
>> 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>

Paraview (AWS)



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

UC San Diego

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	<code>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</code>
GPU	<code>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</code>

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

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

```
[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
```

Request an interactive node
for 30 minutes

NVIDIA-SMI 460.32.03			Driver Version: 460.32.03		CUDA Version: 11.2		
Fan	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
		Pwr:Usage/Cap		Memory-Usage	GPU-Util	Compute M.	MIG M.
0	Tesla V100-SXM2...	On	00000000:86:00.0	Off	0%	0	
N/A	34C	P0	41W / 300W	0MiB / 32510MiB		Default	N/A

Verify you are on a GPU node

Processes:			GPU Memory		
GPU ID	GI ID	CI ID	PID	Type	Process name
No running processes found					

```
[username@login02 ~]$ exit
```

Exit when tasks are done

Expanse User Portal

The screenshot displays the Expanse User Portal interface, which integrates various research computing services. Key features shown include:

- File Browsing, editing, creation, upload, download, etc.**: A file browser interface showing a list of files in a directory, with arrows pointing to the interface.
- SDSC**: The portal's logo and a brief description of its purpose.
- Pinned Apps**: A grid of eight system-installed applications: Active Jobs, Home Directory, Job Composer, Expanse Shell Access, MATLAB, RSTUDIO, Allocation and Usage Information, and Jupyter.
- OOD 2.0 Features**: A callout pointing to the "Allocation and Usage Information" app.
- Interactive Services**: A callout pointing to a MATLAB application window showing a plot.
- Job Script Editing and Submission**: A callout pointing to a "Jobs" submission interface.
- Active Job monitoring and Management**: A callout pointing to a detailed "Active Jobs" monitoring interface.

- <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>

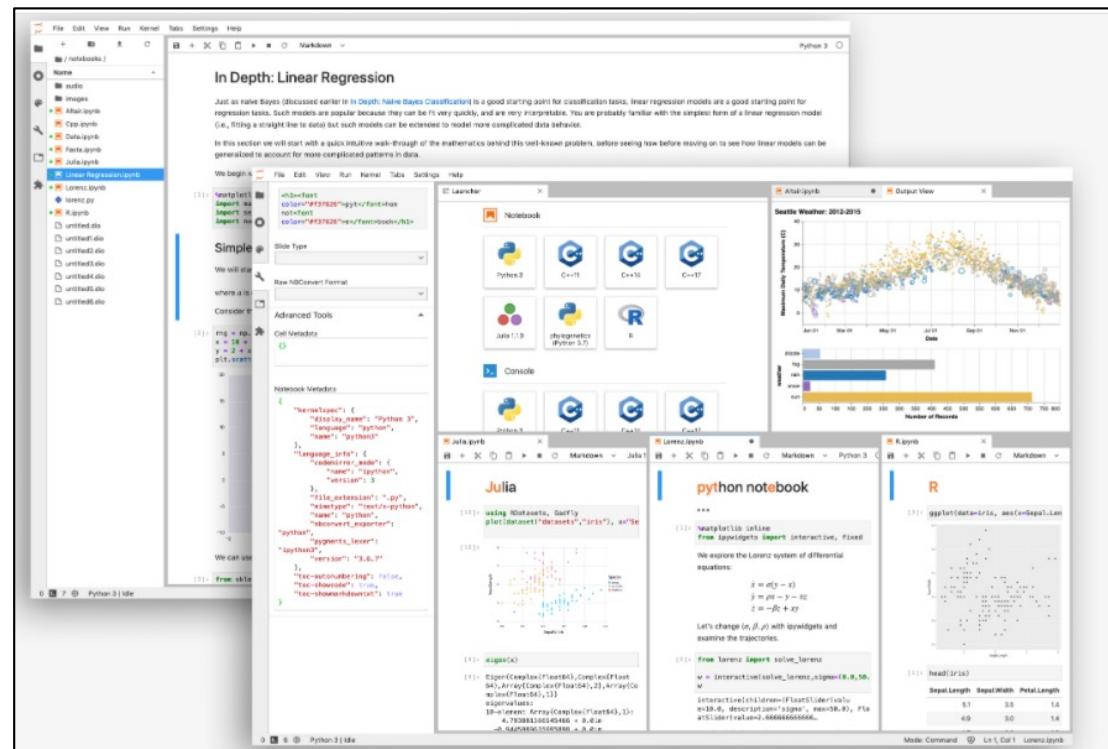
Jupyter Notebooks

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



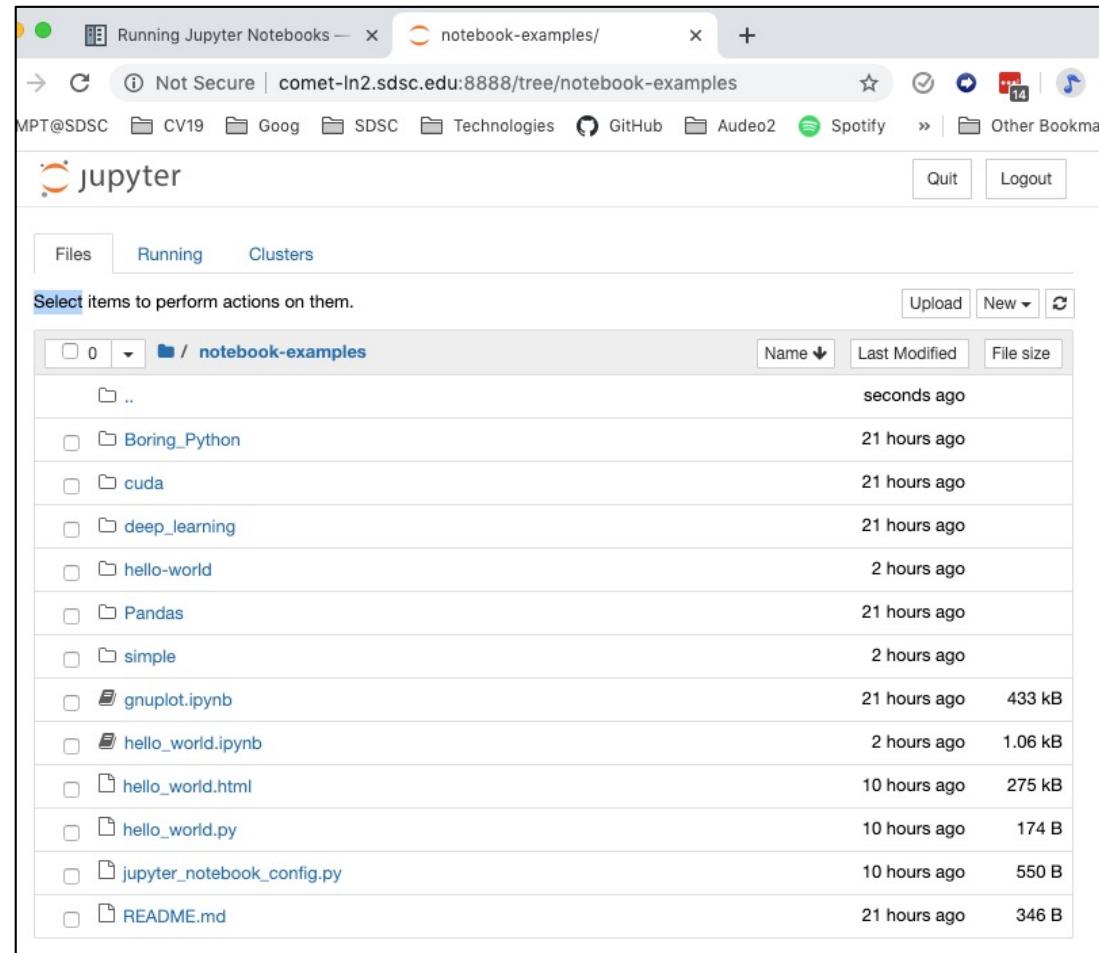
Interactive Services have a Key Vulnerability: They Provide Access to HPC File Systems, often over HTTP

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



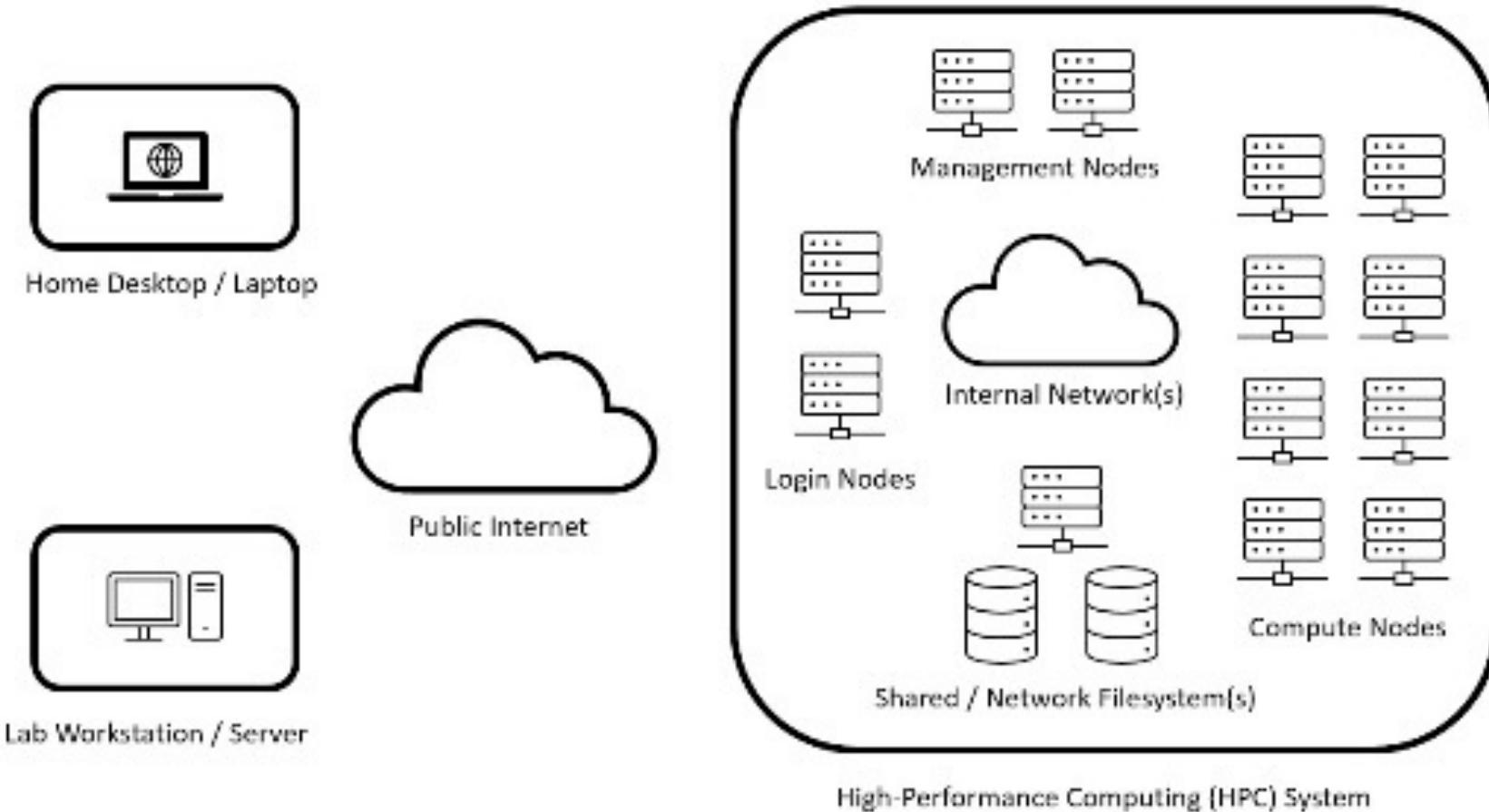
Outline

- Defining Interactive High-Performance Computing
- Notebook Security
- SDSC Solution: the Satellite Reverse Proxy Service (SRPS)
- SRPS Client: galileo
- 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/*galileo* -- **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 *galileo 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!**

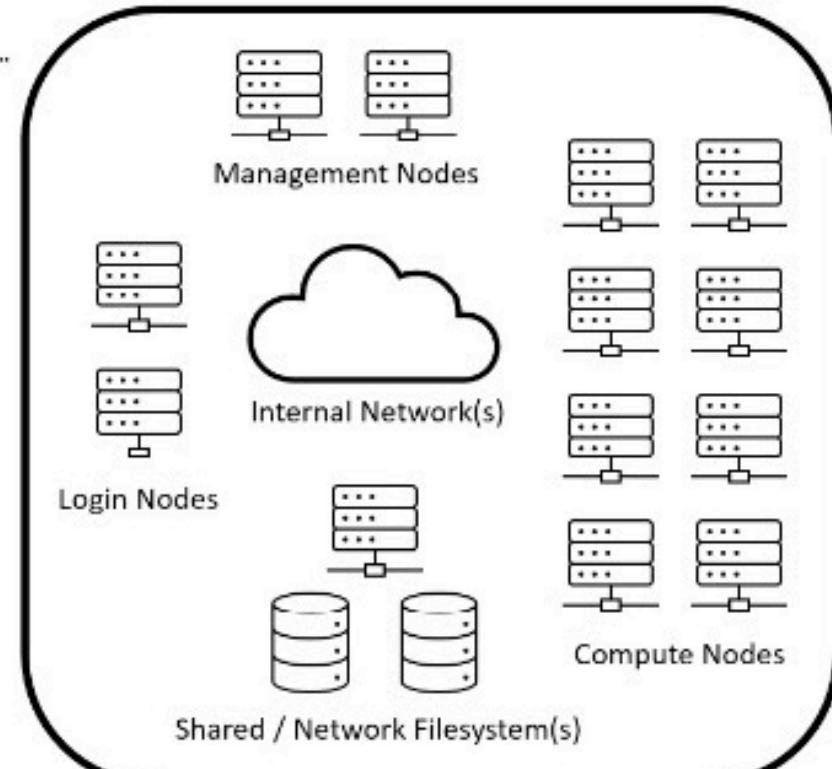
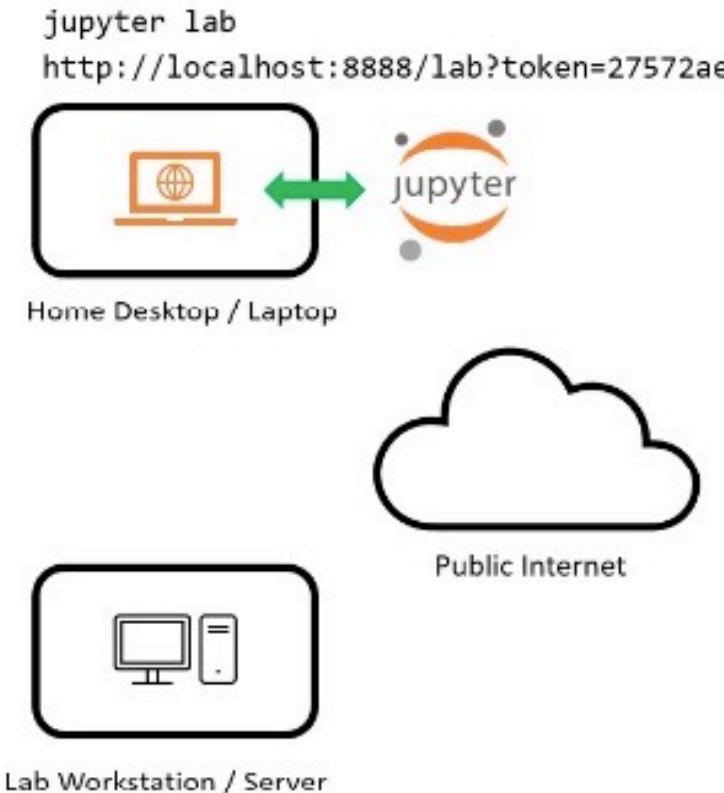
Standard HPC Ecosystem



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/

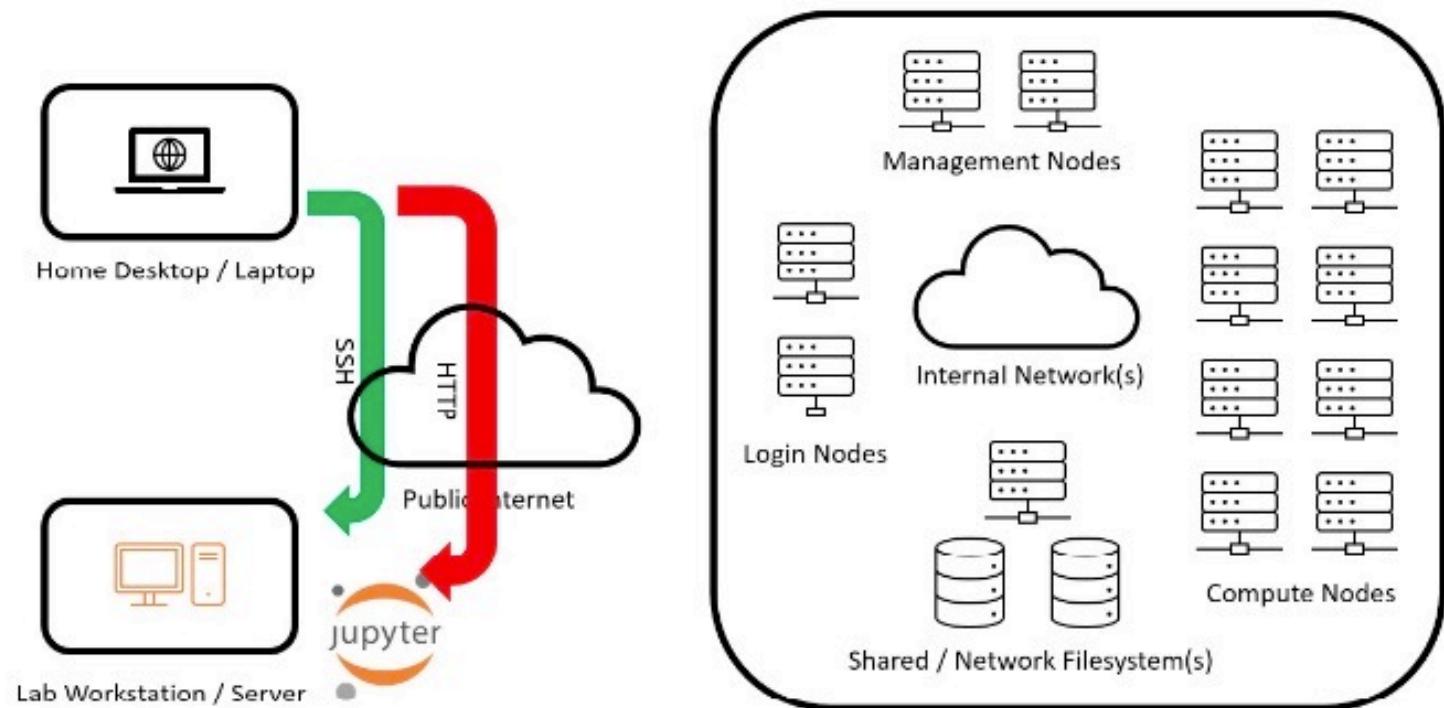
Anaconda on Local Host or Laptop

The image shows three windows illustrating the use of Anaconda on a local host or laptop:

- Left Window:** A terminal session on `localhost:8920/lab` showing file listing and system logs.
- Middle Window:** The Anaconda Navigator interface. It features a sidebar with a tree view of files and a main panel displaying application icons. The "Jupyter" icon is highlighted with a red oval. Below it, the "Launch" button for both JupyterLab and Notebook is also circled in red.
- Right Window:** A browser window on `localhost:8807/tree/dev/sdsc...` showing a list of files and a terminal log window at the bottom.

Running Notebooks Remotely Over HTTP

- Messages over HTTP
- Not encrypted
- Vulnerable to hacking

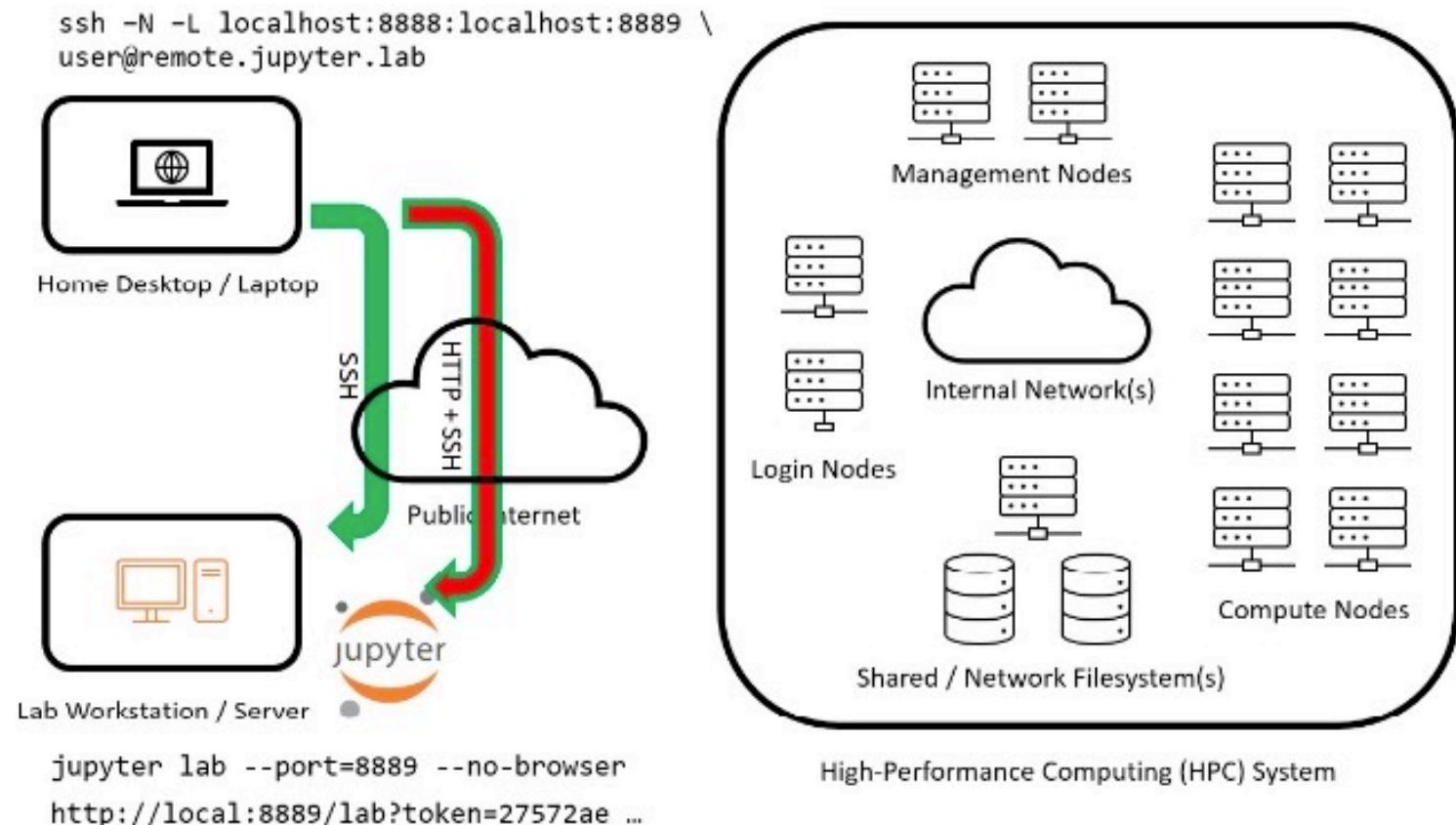


```
jupyter lab --ip="$(hostname)" --no-browser      High-Performance Computing (HPC) System  
http://remote.jupyter.lab:8888/lab?token=27572ae ...
```

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

Running Notebooks Remotely Over HTTP Using SSH Tunneling

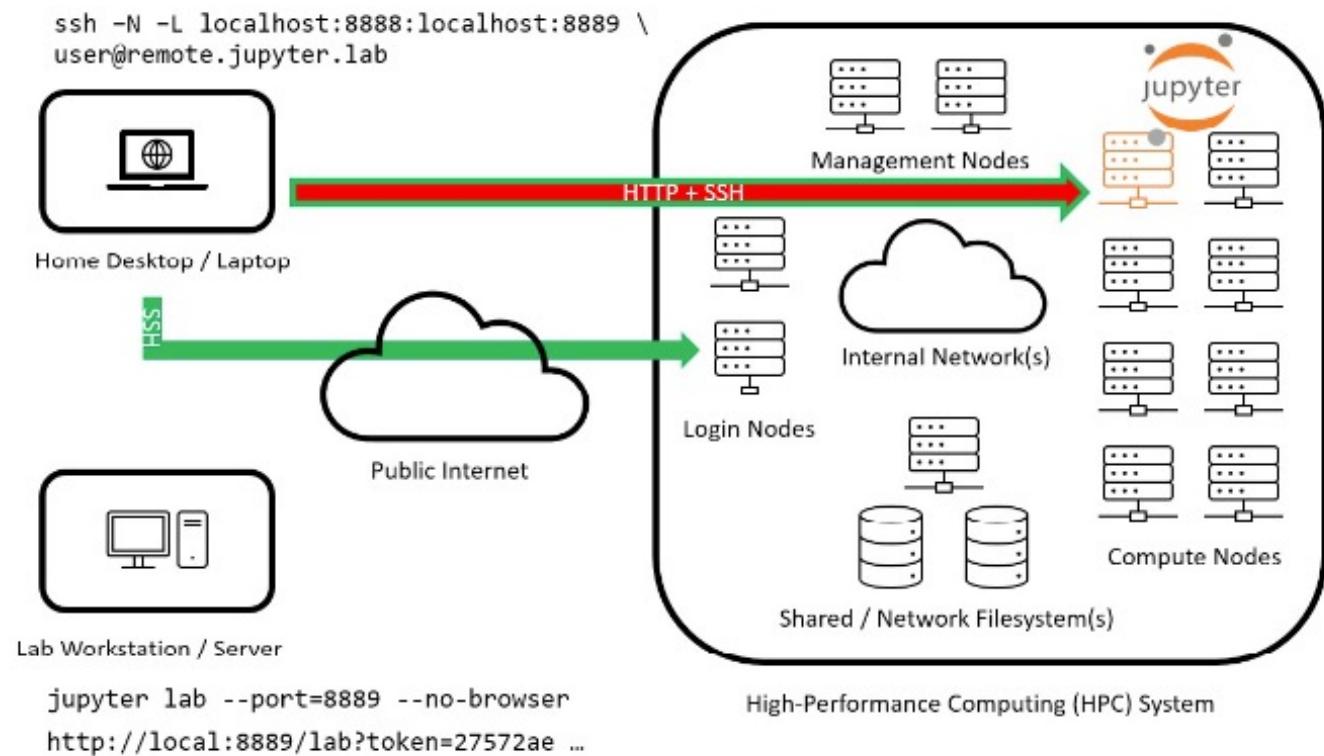
- Notebooks running on Remote host
- Messages over HTTP, but sent via secure tunnel



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

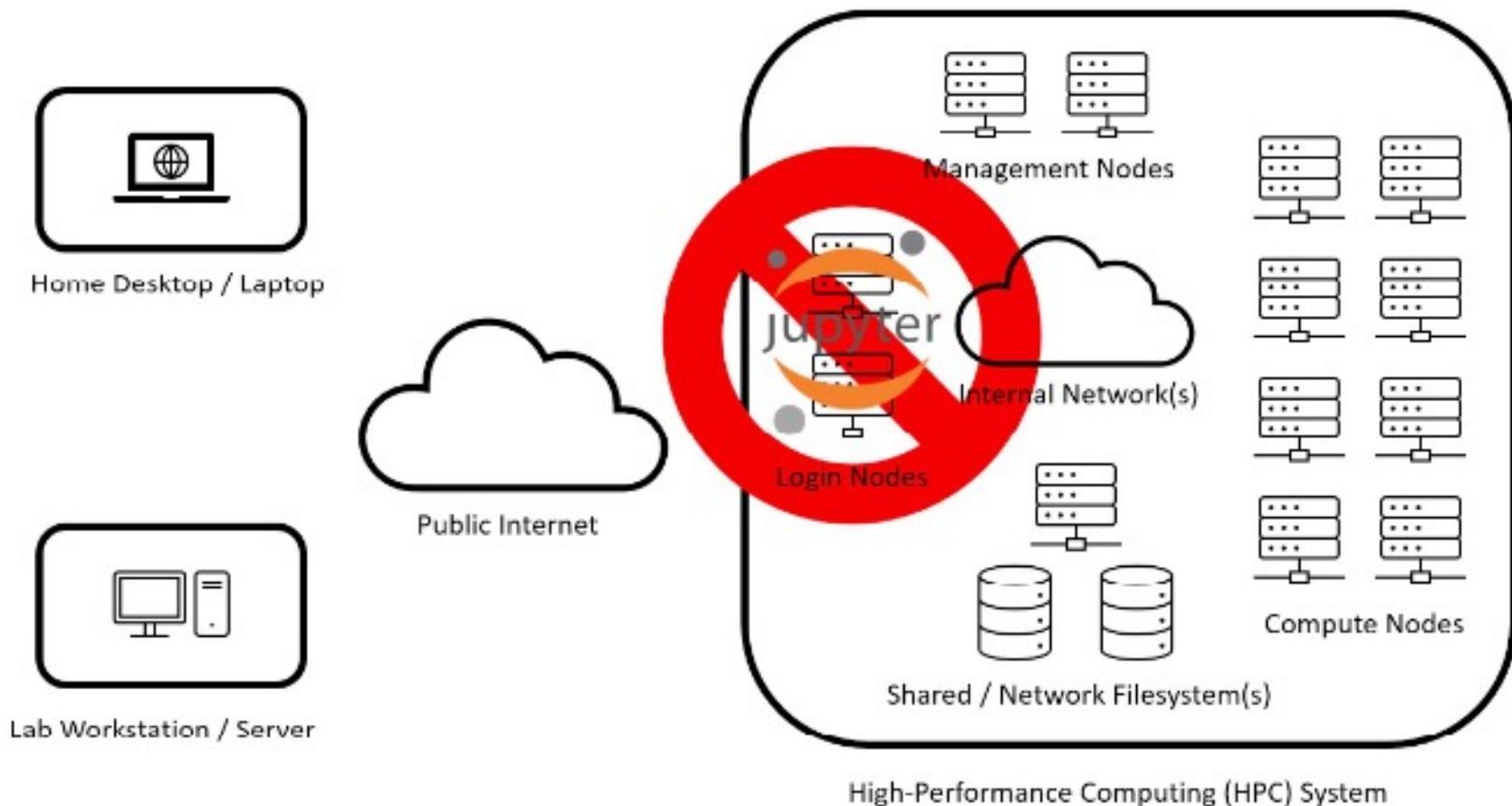
- Notebook running on HPC System
- Messages over HTTP, but passing via secure tunnel
- But do not run on compute nodes!



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: galileo
- Notebook examples

SDSC Satellite Reverse Proxy Service

- SRPS: prototype system that allows users to launch secure standard Jupyter Notebooks 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:

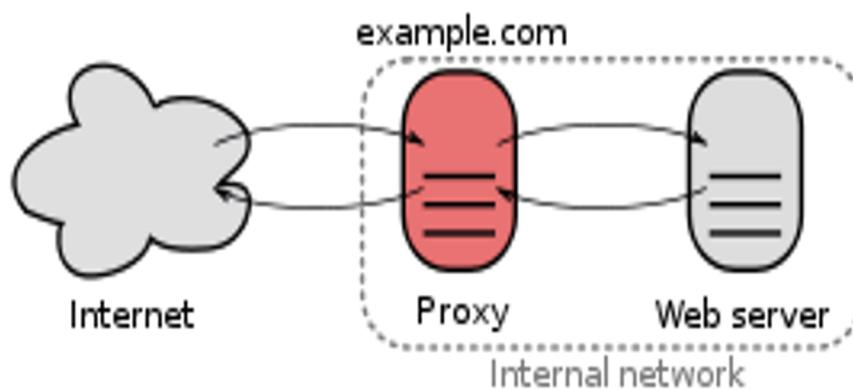
- Satellite: a self-service HTTP(s) reverse-proxy.
- Satellite Client: a shell-based utility to orchestrate a user's interaction with both Satellite and Slurm to start a Jupyter session within a batch job.

GitHub Repo:

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

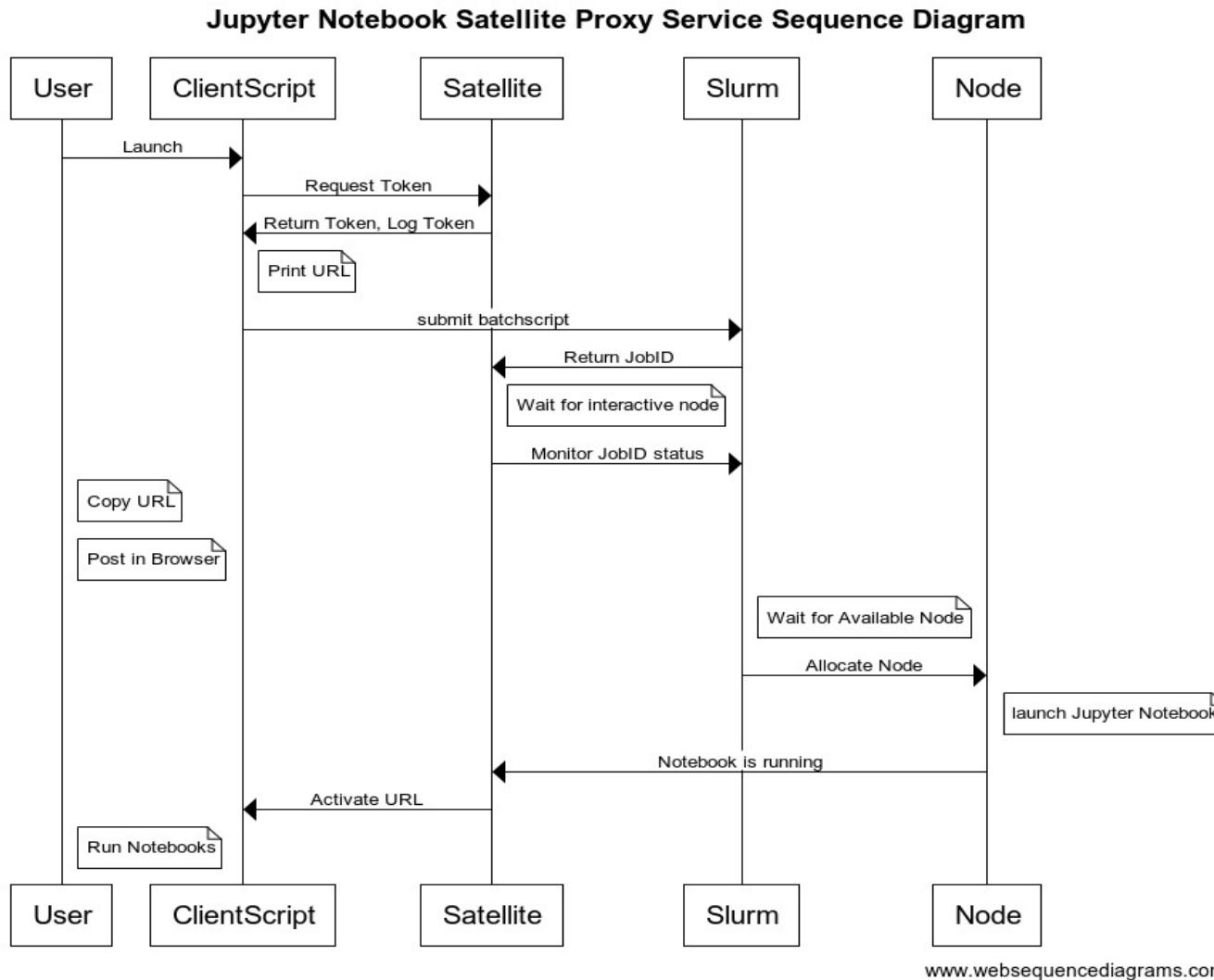
What is a Reverse Proxy?

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.



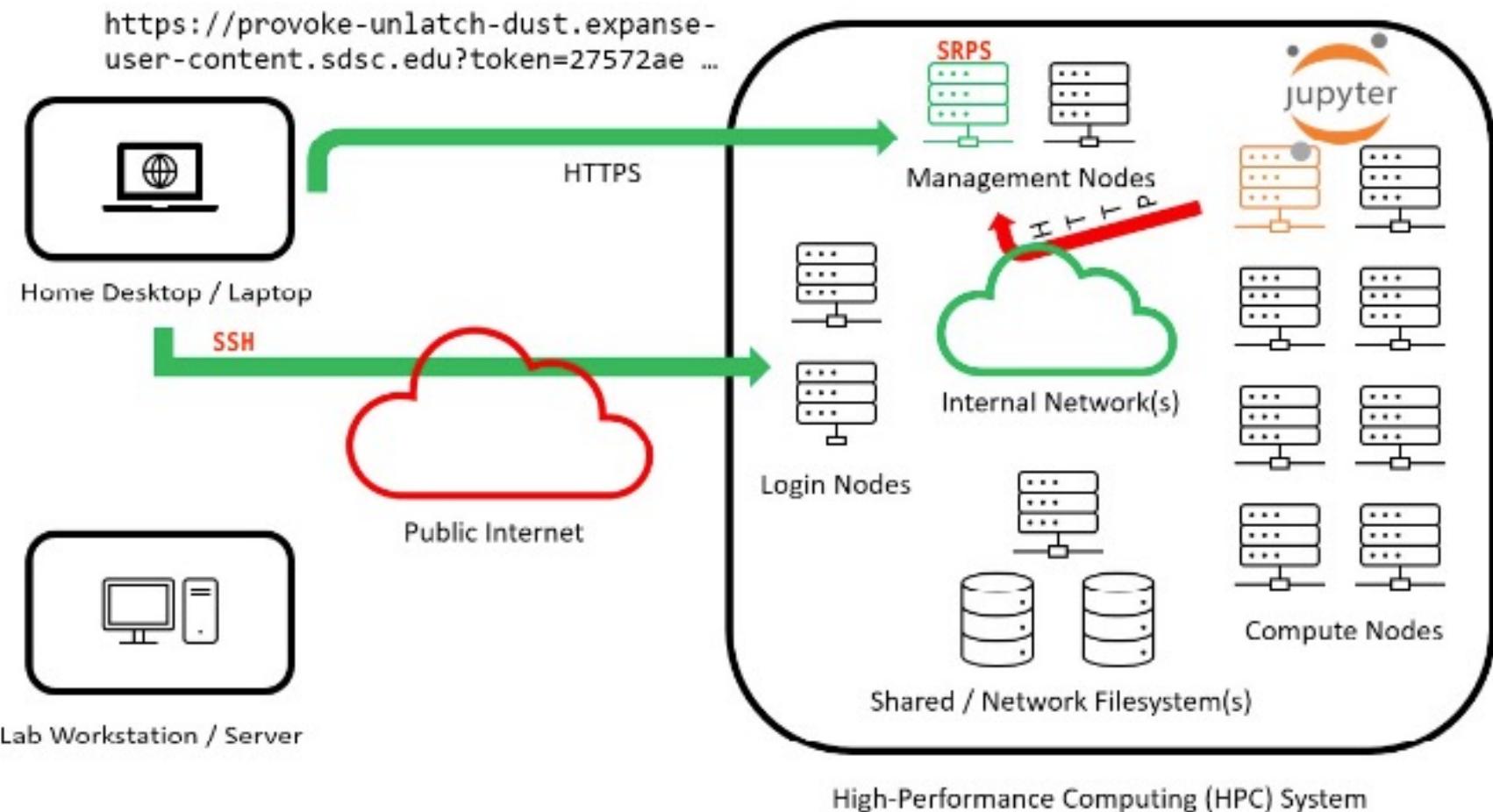
Img Source: [Wikipedia reverse proxy](#)

Satellite Life Cycle



www.websequencediagrams.com

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: `galileo`
- 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 galileo

Step 1:
Login and setup user environment

```
[username@login01 ~]$ which galileo.sh  
/usr/bin/which: no galileo.sh in (/cm/shared/apps/spack/cpu/opt/spack/linux-centos8-zen/gcc-8.3.1  
[SNIP]
```

Step 2:
Run command to launch secure notebook

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

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

```
[username@login01 ~]$ galileo 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.

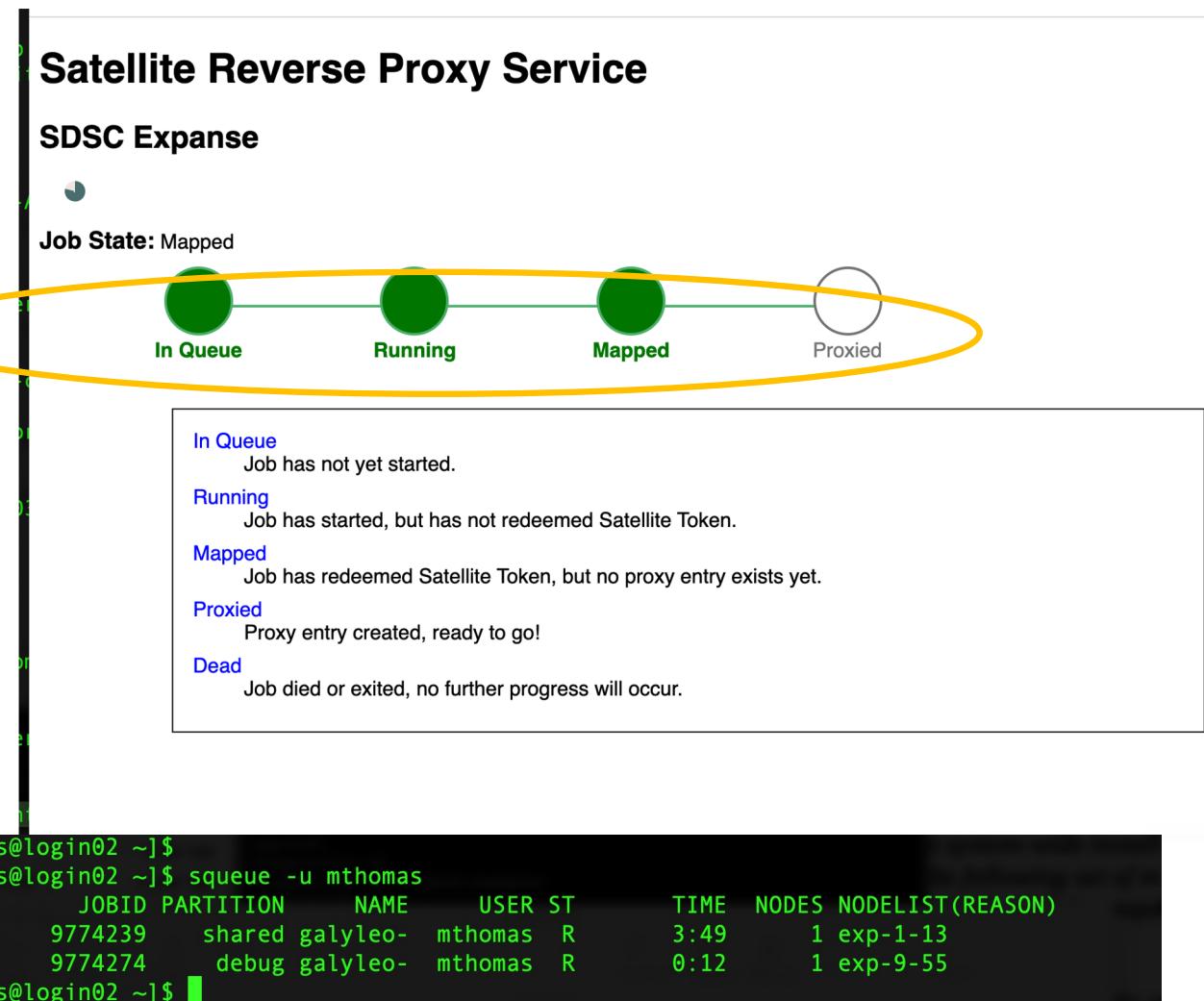
```
https://carload-spray-koala.expanse-user-content.sdsc.edu/lab
```

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

JOBLIST	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REAON)
	gpu-debug	galileo-	username	PD	0:00	1	(None)
	gpu-debug	galileo-	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

The screenshot shows a Jupyter Notebook interface with several tabs open. The main tab displays a 'Hello World' example in a CPU version notebook. Below the code, the output shows the text 'Hello, World.'.

```
[8]: print('Hello world!!!!')
Hello world!!!!
[9]: # Import hello module
import hello

# Define a local function
def world2(name):
    print(name)

[10]: # Call function
world2("mary")
mary
[11]: hello.greeting("good times")
Greetings, good times
[12]: hello.world("World.")
Hello, World.
```

On the left, there are three smaller windows showing the 'Satellite Reverse Proxy Service' status for different job states: 'In Queue', 'Running', and 'Mapped'. Each window has a detailed description of the state:

- In Queue:** Job has not yet started.
- Running:** Job has started, but has not redeemed Satellite Token.
- Mapped:** Job has redeemed Satellite Token, but no proxy entry exists yet.
- Proxied:** Proxy entry created, ready to go!
- Dead:** Job died or exited, no further progress will occur.

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

```
[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  
/expanses,/scratch --nv
```

Step 2:
Run command to launch secure notebook

```
[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
```

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

```
[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

The screenshot shows a Jupyter Notebook interface with two tabs: "numpy_intro.ipynb" and "hello_world_gpu.ipynb". The "hello_world_gpu.ipynb" tab is active, showing Python 3 code. The code includes a command to check for a GPU system and the output of the nvidia-smi command. The output of the nvidia-smi command is highlighted with a yellow circle, showing the GPU is an NVIDIA-SMI 460.32.03. The error message for the if you see command is also highlighted with a yellow circle.

```
sc art arch_perfmon pebs rep_good nopl xtopology nonstop_tsc cpuid aperfmpfperf pni pclm ulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpc pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowp refetch cpuid_fault epb cat_l3 cdp_l3 invpcid_single intel_ppin ssbd mba ibrs ibpb stibp i brs_enhanced tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mb_m_total cqm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni md_clear flush_l1d arc_h_capabilities

[9]: # Check to see if system is GPU:
!nvidia-smi

Fri Feb 18 00:15:50 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. |
|          |          |          |          |          |          |          |
|-----+
|  0  Tesla V100-SXM2... On | 00000000:18:00.0 Off |           0 | | | |
| N/A   37C     P0    68W / 300W |      0MiB / 32510MiB |     0%      Default |
|          |          |          |          |          |          |
+-----+
+-----+
| Processes:
| GPU  GI CI      PID  Type  Process name          GPU Memory |
|          ID  ID                  Usage          |
|-----+
| No running processes found
+-----+


[10]: # if you see: /bin/bash: nvidia-smi: command not found
# the system is not GPU
```

Using Expanse Portal to Launch Notebooks

The screenshot shows the Expanse Portal interface for launching Jupyter notebooks. On the left, a configuration form for a Jupyter session is displayed, including fields for Account (use300), Partition (debug), Time limit (min) (30), Number of cores (1), Memory required per node (GB) (2), and GPUs (optional) (0). Below these are sections for Singularity Images and Environment modules, both currently empty. On the right, a terminal window titled "Terminal 1" shows a command-line session using Singularity, listing various container names and their last modified times. A smaller window at the bottom left shows a preview of the Jupyter session URL.

Open OnDemand / Jupyter Session

Jupyter Session

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 Images /cm/shared/applications

Environment modules

Conda Environments

File Edit View Run Kernel Tabs Settings Help

+

Name Last Modified

Running

comet-files a day ago

conda-install-tmp a year ago

dev a day ago

galileo-examples a day ago

galileo-repo a day ago

hpctrain a day ago

interactive.ex a day ago

miniconda3 7 months ago

ml-dev-mary 7 months ago

Commands

Tabs

Terminal 1

```
Singularity> pwd  
/home/mthomas  
Singularity> hostname  
exp-9-55  
Singularity> cd  
Singularity> cd notebook-exam  
Singularity> ll  
11: command not found  
Singularity> ls -al  
total 228  
drwxr-xr-x 10 mthomas use300  
drwxr-x--- 33 mthomas use300  
-rw-r--r-- 1 mthomas use300  
drwxr-xr-x 8 mthomas use300  
-rw-r--r-- 1 mthomas use300  
drwxr-xr-x 3 mthomas use300  
drwxr-xr-x 2 mthomas use300  
drwxr-xr-x 2 mthomas use300  
drwxr-xr-x 4 mthomas use300  
drwxr-xr-x 2 mthomas use300  
drwxr-xr-x 2 mthomas use300  
drwxr-xr-x 3 mthomas use300  
-rw-r--r-- 1 mthomas use300  
Singularity>
```

Open OnDemand / Jupyter Session

Jupyter Session

2022-02-18 00:35:10 -0800 <https://crushing-props-backslid.expanse-user-content.sdsc.edu?token=e96f6928f048fa9a000d372590982cee>

Outline

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

The screenshot shows the GitHub repository page for 'sdsc-hpc-training-org / notebook-examples'. The repository is public, as indicated by the 'Public' badge. The master branch is selected, showing 3 branches and 0 tags. The commit history lists 60 commits, all made by 'marypthomas' and dated 'now'. The commits are: 'Update README.md', 'Boring_Python_Book', 'CUDA_GPU_NVIDIA', 'Data_Analysis', 'Hello_World', 'Matplotlib_Intro', 'Notebook_Dev_Basics', 'NumPy_Intro', 'hpc-notebooks', '.DS_Store', '.gitignore', and 'README.md'. The last commit was 'Update README.md'.

Commit	Message	Date
marypthomas Update README.md	32163d3 now	60 commits
Boring_Python_Book	updating training material	7 months ago
CUDA_GPU_NVIDIA	updating training material	7 months ago
Data_Analysis	updating training material	7 months ago
Hello_World	updating training material	7 months ago
Matplotlib_Intro	updating training material	7 months ago
Notebook_Dev_Basics	updating training material	7 months ago
NumPy_Intro	updating training material	7 months ago
hpc-notebooks	merge hpc notebooks material	16 minutes ago
.DS_Store	add hello world	10 months ago
.gitignore	update material	2 years ago
README.md	Update README.md	now

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/galileo:/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-expans>
- 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 Galileo:
 - <https://github.com/sdsc-hpc-training-org/satellite>
 - <https://github.com/mkandes/galileo>
- 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