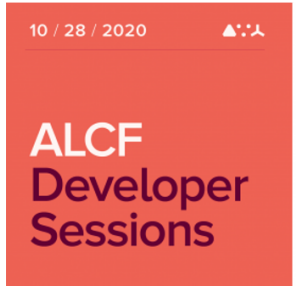
Argonne NATIONAL LABORATORY

Towards Interactive High-Performance Computing with ALCF JupyterHub

Murat Keçeli

keceli@anl.gov

Computational Science Division, Argonne National Laboratory



Outline

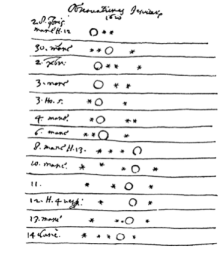
- Project Jupyter
- What you can do with Jupyter?
- Jupyter/IPython basics
- Introduction to markdown, magic, widgets
- Introduction to ALCF JupyterHub
- Live Demos
    - New kernel installation
    - ezCobalt: how to submit jobs
    - ezBalsam: how to use Balsam

Disclaimer

- This webinar will not cover:
    - low level details about queuing or ensembling jobs or creating Balsam workflows, etc. covered in a previous webinar
    - using Jupyter through an ssh tunnel, reverse proxy, or remote kernels
    - using Dask, Spark, Kubernetes, or a container for distributed computing
    - accessing compute nodes directly
- ALCF JupyterHub is a new service and improving rapidly. You can send an email to support@alcf.anl.gov (cc: keceli@anl.gov) for problems and suggestions.

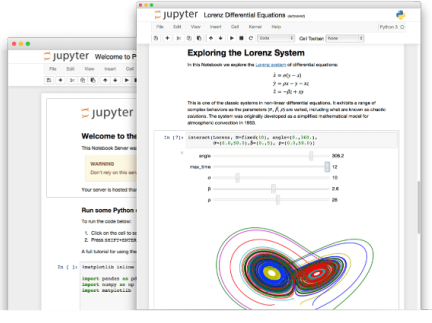
Project Jupyter

- Started in 2014, as an IPython spin-off project led by Fernando Perez to “develop open-source software, open-standards, and services for interactive computing”.
- Inspired by Galileo’s notebooks and languages used in scientific software: Julia, Python, and R.

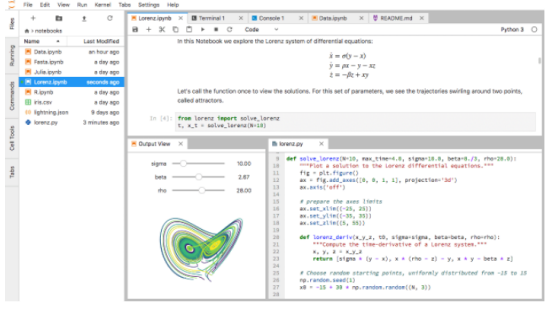


Jupyter X

Jupyter Notebook



JupyterLab



jupyterhub

jupyter {book}

What you can do?

- Interactive development environment
    - Fast code prototyping, test new ideas easily
    - Most languages are supported through Jupyter kernels
- Learn or teach with notebooks
    - Prepare tutorials, run demos
- Data analysis and visualization
- Presentations with Reveal.js
- Interactive work on HPC centers or cloud
    - JupyterHub
    - Google Colab
    - Binder

Basics (Shortcuts)

- `Esc/Enter` get in command/edit mode

Command mode		Edit mode	
`h`	show (edit) all shortcuts	`shift enter`	Run cell, select below
`a/b`	insert cell above/below	`cmd/ctrl enter`	Run cell
`c/x`	copy/cut selected cell	`tab`	completion or indent
`V/v`	paste cell above/below	`shift tab`	tooltip
`d,d`	delete cell	`cmd/ctrl d`	delete line
`y/m/r`	code/markdown/raw mode	`cmd/ctrl a`	select all
`f`	search, replace	`cmd/ctrl z`	undo
`p`	open the command palette	`cmd/ctrl /`	comment

Basics (Shortcuts)

- `Esc/Enter` get in command/edit mode

Command mode		Edit mode	
`h`	show (edit) all shortcuts	`shift enter`	Run cell, select below
`a/b`	insert cell above/below	`cmd/ctrl enter`	Run cell
`c/x`	copy/cut selected cell	`tab`	completion or indent
`V/v`	paste cell above/below	`shift tab`	tooltip
`d,d`	delete cell	`cmd/ctrl d`	delete line
`y/m/r`	code/markdown/raw mode	`cmd/ctrl a`	select all
`f`	search, replace	`cmd/ctrl z`	undo
`p`	open the command palette	`cmd/ctrl /`	comment

In [6]:

```
import os
os.getenv??
#help('modules')
#help('modules mpi4py')
```

Markdown

- bullet list
    - subbullet
- equation: $E = mc^2$
- inline code `echo hello jupyter` `
- A link
- Table

Col 1	Col 2	Col 3
1, 1	1,2	1,3
2, 1	2,2	2,3
3, 1	3,2	3,3

-  A kitten

IPython Magic

- Magic functions are prefixed by `%` (line magic) or `%%` (cell magic)
- Cell magic `%%` should be at the first line
- Shell commands are prefixed by `!`
- `%quickref`: Quick reference card for IPython
- `%magic`: Info on IPython magic functions
- `%debug`: Interactive debugger
- `%timeit`: Report time execution
- `%prun`: Profile (%lprun is better, `pip install lprun` and `%load_ext line_profiler`)

In [7]:

```
%magic
```

In [7]:
```
%magic
```

In [60]:
```
import numpy as np
a = [1]*1000
%timeit sum(a)
b = np.array(a)
%timeit np.sum(a)
%timeit np.sum(b)
```

```
10000 loops, best of 5: 7.51 µs per loop
The slowest run took 145.08 times longer than the fastest. This could mean
that an intermediate result is being cached.
10000 loops, best of 5: 106 µs per loop
The slowest run took 5.23 times longer than the fastest. This could mean t
hat an intermediate result is being cached.
100000 loops, best of 5: 7.14 µs per loop
```

Jupyter Widgets (ipywidgets)

- Widgets are basic GUI elements that can enhance interactivity on a Jupyter notebook
- Enables using sliders, text boxes, buttons, and more that can link input and output.

Jupyter Widgets (ipywidgets)

- Widgets are basic GUI elements that can enhance interactivity on a Jupyter notebook
- Enables using sliders, text boxes, buttons, and more that can link input and output.

In [1]:
```
import ipywidgets
ipywidgets.IntSlider()
```

Jupyter Widgets (ipywidgets)

- Widgets are basic GUI elements that can enhance interactivity on a Jupyter notebook
- Enables using sliders, text boxes, buttons, and more that can link input and output.

In [1]:
```
import ipywidgets
ipywidgets.IntSlider()
```

In [2]:
```
ipywidgets.Text(value='Hello Jupyter!', disabled=False)
```

Jupyter Widgets (ipywidgets)

- Widgets are basic GUI elements that can enhance interactivity on a Jupyter notebook
- Enables using sliders, text boxes, buttons, and more that can link input and output.

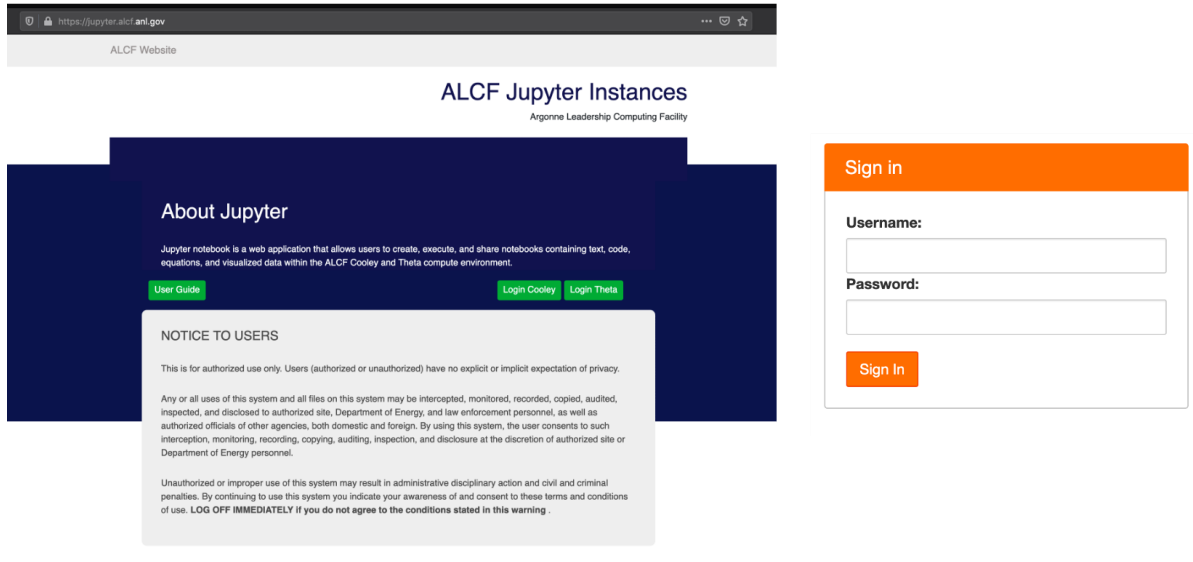
In [1]:
```
import ipywidgets
ipywidgets.IntSlider()
```

In [2]:
```
ipywidgets.Text(value='Hello Jupyter!', disabled=False)
```

In [3]:
```
ipywidgets.ToggleButton(value=False, description="Don't click",
                        button_style='danger', tooltip='Description',)
```

ALCF JupyterHub

- If you are an ALCF user, you can log in to Jupyter Hub at https://jupyter.alcf.anl.gov using your ALCF credentials.
- If not, check https://alcf.anl.gov/support-center/get-started
- Jupyter Hub instances runs on an external servers, but not on login, mom, or compute nodes.
- Servers have 16 core Intel(R) Xeon(R) CPU E5-2683 and 512 GB memory and reserved for data analytics and visualization, not simulations.



ALCF JupyterHub

- JupyterHub for Cooley :
    - runs on jupyter01.mcp.alcf.anl.gov
    - has access to the user's home folder `/home/$USER`, the Mira projects folder `/projects`, and the Theta project folder `/lus/theta-fs0/projects`
    - submitted jobs will run on Cooley
- JupyterHub for Theta:
    - runs on jupyter02.mcp.alcf.anl.gov
    - has access to your home folder `/home/$USER` and projects folder `/lus/theta-fs0/projects`*
    - does not have access to `/opt/cray`, `/opt/intel`, etc., that is, you cannot use any Theta modules or any Cray libraries.
    - Submitted jobs will run on Theta

Notes

- JupyterHub starts on your home folder, to access project folders, you can create a symbolic link `!ln -s /project/my_project my_project`
- If you have a broken symlink on your home directory, JupyterHub gives a server error with `permission denied` message. You need to clean up / fix the broken symbolink links.
- When you exceed your file quota, you may also experience problems. Check with `myquota`.
- To run JupyterLab on JupyterHub, modify the link to `https://jupyter.alcf.anl.gov/cooley/user/$USER/lab`
- Documentation is available at https://www.alcf.anl.gov/support-center/theta/jupyter-hub

How to install a new Conda environment & Jupyter kernel

Step 0

- Check the names of the existing environments & kernels:

```
!conda env list
!jupyter kernelspec list
```

- Select a name for the new environment & kernel.

- Using a prefix such as `jhub_` is helpful to distinguish JupyterHub environments from others.

```
ENVNAME="jhub_demo"
```

Step 1

- Create a new environment

```
!conda create -y --name $ENVNAME
```

- Or, create a new environment with a different python version

```
!conda create -y --name $ENVNAME python=3.8
```

- Or, create a new environment with a clone of the base environment (recommended)

```
!conda create -y --name $ENVNAME --clone base
```

A step backward

Step 2

- Install new packages with `conda`, or `pip`

```
!source activate $ENVNAME; conda install -y -c conda-forge rise
```

```
!source activate $ENVNAME; pip install balsam-flow
```

- Or, if you didn't clone from the base, you need to install the following packages additionally:

```
!source activate $ENVNAME; conda install -y jupyter nb_conda
ipykernel
```

Step 3

- Install the kernel for Jupyter

```
!source activate $ENVNAME;python -m ipykernel install --user --name
$ENVNAME
```

Final steps

- Refresh the browser or open a new notebook.
- Select the new `Kernel` from the top dropdownlist
- When you need to install another package, you only need to run the following steps

```
ENVNAME='jhub_demo'
!source activate $ENVNAME; conda install -y <any_conda_package>
!source activate $ENVNAME; pip install -c <any_pypi_package>
```

Notes

- Check the installation with

```
!conda list
import <any_package>
print(<any_package>.__file__)
print(<any_package>.__version__)
```

- Do not use environments installed on JupyterHub elsewhere.

Clean up

- You may run out of space quickly, check with `myquota`.
- You can run `conda clean` to remove index cache, lock files, tarballs, unused cache packages, and source cache

```
!conda clean --all -y
```

- To remove an environment and the kernel you don't need:

```
!conda env remove -y -n $ENVNAME
!jupyter kernelspec uninstall -y $ENVNAME
```

Resources

- Fernando Perez's Project Jupyter presentation
- jupyter.org
    - Check out Voilà, Jupyter Lab, Jupyter Book
- Jupyter tutorial
- Version control for Jupyter
- ALCF ML tutorials
- More ALCF notebooks

Acknowledgements

- Thank you all for attending
- Thanks to Misha, Alvaro, and Ray for their feedback and suggestions
- Thanks to Tommie for running and maintaining JupyterHub servers
- Thanks to Gurunath for working together during the summer
- Thanks to Venkat, Tom, and Mike for motivation and support

Live Demo

- All materials are at https://github.com/keceli/ezHPC

```
git clone https://github.com/keceli/ezHPC
```