

October 10-12, 2023



ALCF Hands-on HPC Workshop

Python, Jupyter Notebook and Containers

Aditya Tanikanti and Archit Vasani

Outline

- Using Python on Polaris
 - Managing conda environments
 - Running python and multi-rank jobs
- Using Jupyter Notebooks on Polaris
 - Creating a new notebook and running programs
 - Working with machine learning modules
- Containers at ALCF
 - Introduction
 - Advantages of containers
 - Containers at ALCF

Using Python on Polaris

Logging in to interactive nodes

- If you are an ALCF user, login to Polaris via:

```
ssh user@polaris.alcf.anl.gov
```

- Else, check:

<https://alcf.anl.gov/support-center/get-started>

- Request an interactive node:

```
qsub -I -A fallws23single -l select=1 -l walltime=01:00:00  
-l filesystems=home:grand:eagle -q debug
```

- **-I** : interactive node
- **-A fallws23single** : fallws23single is the allocation
- **-l select=1** : request 1 compute node
- **-l walltime=01:00:00** : request for 1 hour
- **-l filesystems=home:grand:eagle** : request to use these filesystems
- **-q debug**: request to use debug queue

Managing conda environments

- Polaris uses modules to control loading of software environments
- There are prebuilt environments containing GPU-supported builds of torch, tensorflow, jax, etc.
- To use these:

```
module load conda/2023-10-04
conda activate
```

- To use an older conda version search for available conda environments and load that version:

```
module avail conda
module load conda/2022-09-08
conda activate
```

Managing conda environments

- If you need more flexibility to install your own packages (e.g. using conda install, pip install)
 - Clone the base conda environment:

```
module load conda/2023-10-04
conda activate
conda create -clone base -prefix /path/to/envs/base-clone
conda activate /path/to/envs/base-clone
```

- Note: make sure to change /path/to/envs/base-clone to where you want to install the environment
 - Also, to ensure proper functioning of your environment, install within an interactive job not on login node

Running python

- To run a single rank python job simply activate your conda environment and then run your process

```
module load conda/2023-10-04
conda activate
python example_script.py
```

- If you need additional packages for your application, activate your created conda environment and install via pip/conda

```
module load conda/2023-10-04
conda activate /path/to/envs/base-clone
pip install example_module
python example_script.py
```


Running multi-rank jobs

- Polaris has 64 CPUs and 4 A100 GPUs on each compute node.
- To parallelize across these, use MPI:

```
module load conda/2023-10-04
conda activate
mpiexec -n NPROC -ppn PROC_PER_NODE yourrun
```

- To use MPI with python, use the mpi4py module:

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
size = comm.Get_size()
rank = comm.Get_rank()
print(f'My rank is {rank} of {size} total ranks')
```

- This program creates an MPI World, gets number of ranks (size), and specific rank.
- It then outputs the rank id across all processes.

Using Jupyter Notebooks on Polaris

Logging in

- Access JupyterHub at: <https://jupyter.alcf.anl.gov/>
- Select **Login Polaris** and use ALCF credentials + Multi-factor Authentication to login

ALCF Jupyter Instances
Argonne Leadership Computing Facility

About Jupyter

Jupyter notebook is a web application that allows users to create, execute, and share notebooks containing text, code, equations, and visualized data within the ALCF Cooley and Theta compute environment.

[User Guide](#) [Login Cooley](#) [Login Theta](#) [Login ThetaGPU](#) [Login Polaris](#)

NOTICE TO USERS

This is for authorized use only. Users (authorized or unauthorized) have no explicit or implicit expectation of privacy.

Any or all uses of this system and all files on this system may be intercepted, monitored, recorded, copied, audited, inspected, and disclosed to authorized site, Department of Energy, and law enforcement personnel, as well as authorized officials of other agencies, both domestic and foreign. By using this system, the user consents to such interception, monitoring, recording, copying, auditing, inspection, and disclosure at the discretion of authorized site or Department of Energy personnel.

Unauthorized or improper use of this system may result in administrative disciplinary action and civil and criminal penalties. By continuing to use this system you indicate your awareness of and consent to these terms and conditions of use. **LOG OFF IMMEDIATELY if you do not agree to the conditions stated in this warning .**

Starting a new server

- You want to setup your server Job Options as follows.
- Pressing start will submit to the batch queue

Select a job profile:

Polaris Compute Node

Queue Name

debug

Project List

fallwkshp23

Number of Nodes

1

Runtime (minutes:second)

30:00

File Systems

Home

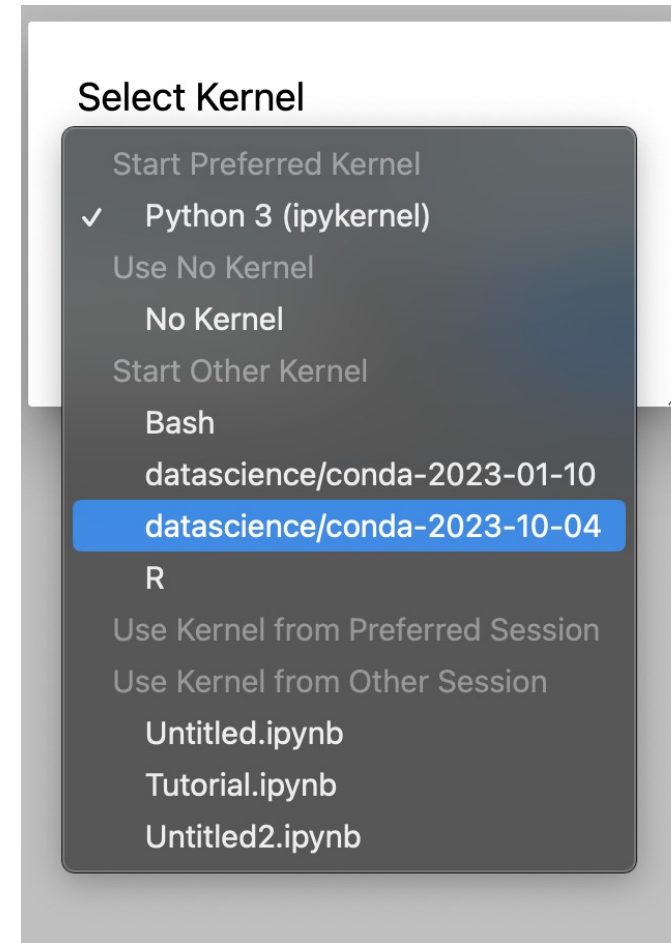
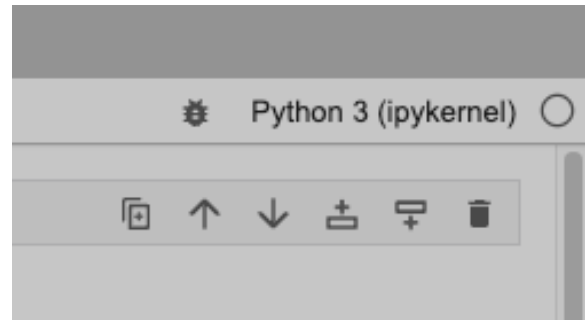
Grand

Eagle

Start

Creating new notebook

- Once the job begins start a new notebook.
- To use a conda environment with several necessary python modules, change kernel to datascience/conda-2023-10-04



Simple example

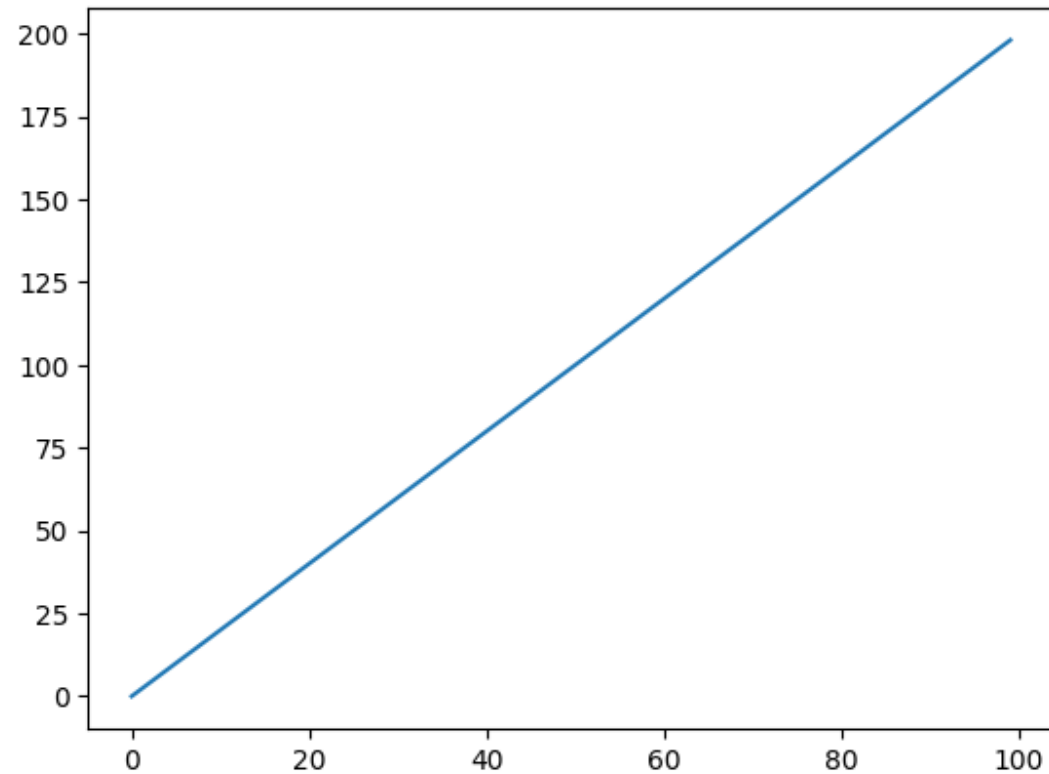
```
[1]: import numpy as np
```

```
[2]: import matplotlib.pyplot as plt
```

```
[3]: a = [i * 2 for i in range(0, 100)]
```

```
[4]: plt.plot(a)
```

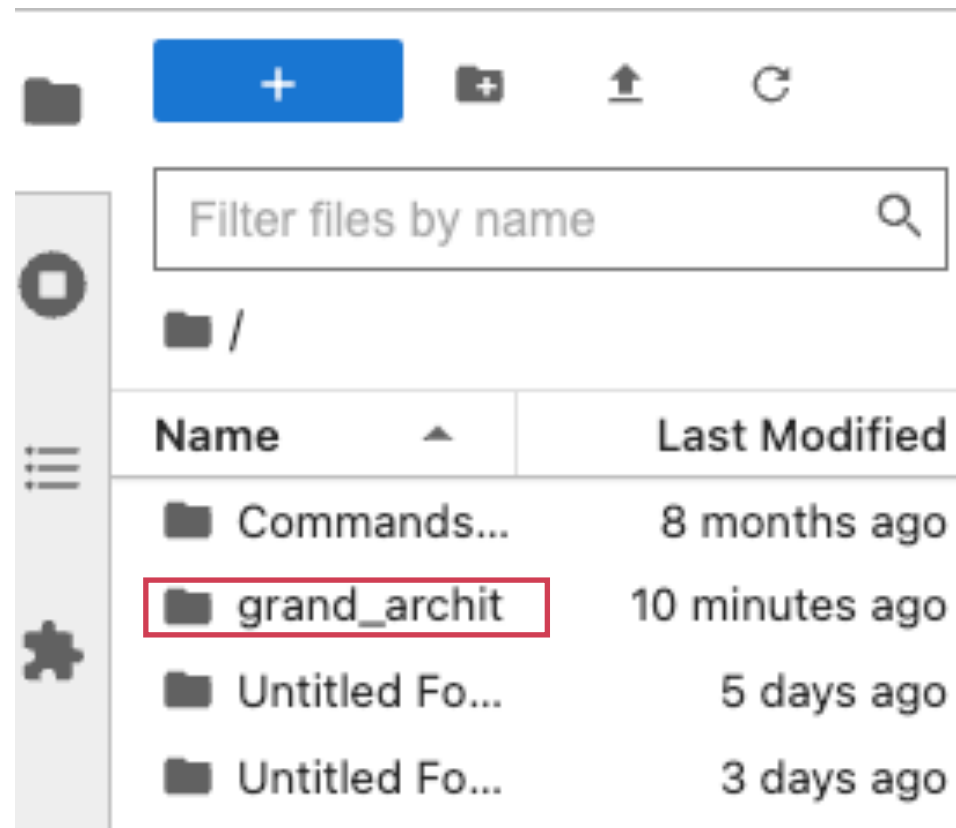
```
[4]: [<matplotlib.lines.Line2D at 0x14e74c605ba0>]
```



Accessing project folders

- JupyterHub starts on your home folder
- Need to create symbolic link to access projects

```
!ln -s /grand/datascience/avasan grand_archit
```



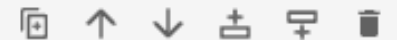
Working with machine learning modules

- Machine learning modules Tensorflow and Pytorch are installed in datascience/conda-2023-10-04 module
- Here is how to check GPU usage on these modules:

```
import tensorflow as tf
```

```
2023-10-09 19:14:58.095004: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

```
tf.config.list_physical_devices('GPU')
```



```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU'),  
 PhysicalDevice(name='/physical_device:GPU:1', device_type='GPU'),  
 PhysicalDevice(name='/physical_device:GPU:2', device_type='GPU'),  
 PhysicalDevice(name='/physical_device:GPU:3', device_type='GPU')]
```

```
import torch
```

```
torch.cuda.is_available()
```

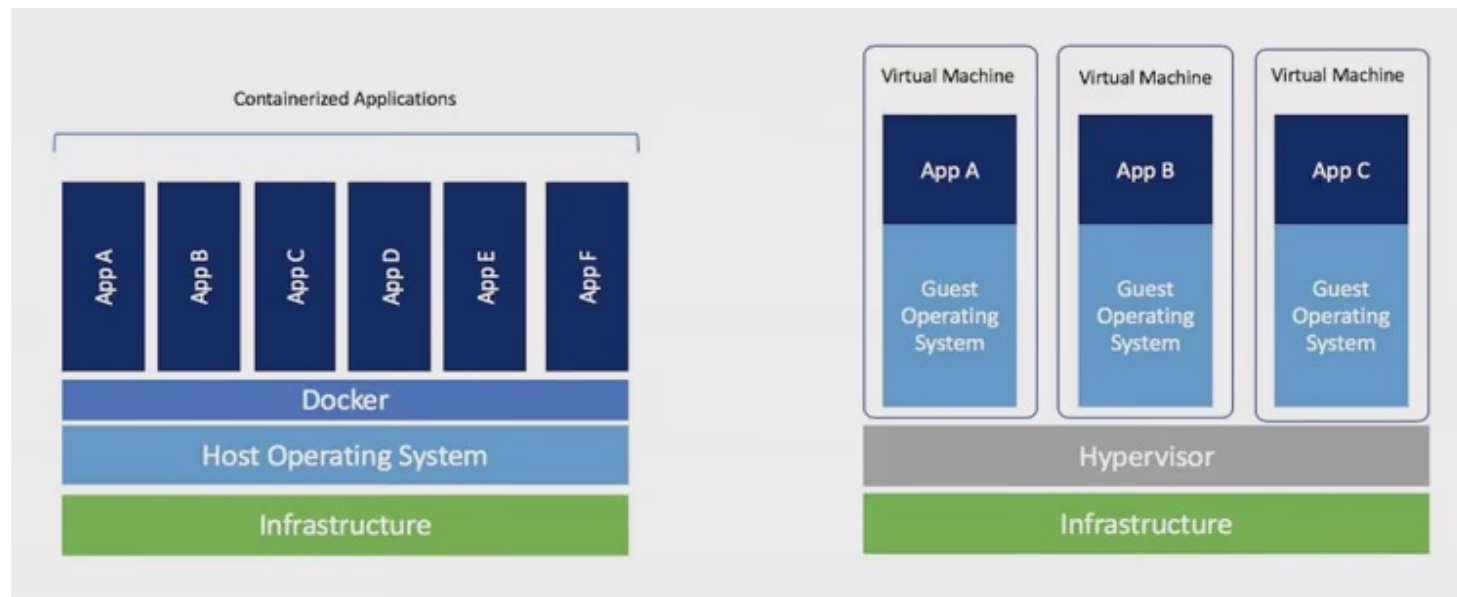
```
True
```


Containers at ALCF

Introduction to Containers

A container is a software package that wraps a software process or microservice to make it executable in all computing environments. It encapsulates an application and its dependencies into a "container". It runs natively on the operating system's kernel, sharing the kernel with other containers.

Ideally, a container can be copied from one system to another, and the contained software runs without changes to the installation. [Containers are often compared to virtual machines \(VMs\)](#)



Advantages of Containers:

Portability: Consistent behavior across different environments.

Lightweight: Quick startups and efficient resource use.

Isolation: Secure and conflict-free application environments.

Efficiency: Maximizes system resource utilization.

Microservices: Supports breaking apps into smaller, scalable services.

Scalability: Easily scales with tools like Kubernetes.

Version Control: Infrastructure can be tracked and managed like code.

CI/CD: Simplifies continuous deployment and integration.

Developer Productivity: Consistent local development setup.

Strong Ecosystem: Vast community and third-party tool support.

Containers at ALCF

- Several container technologies like Docker, podman, containerd. At ALCF, users must run [Singularity](#) containers. Singularity is a container technology built for supercomputers with security in mind. Singularity has now joined the Linux Foundation and has been renamed [Apptainer](#)
- Either build a singularity container from scratch or build a [docker](#) container locally on your machine and subsequently convert it to a singularity container. An example to build a docker container locally can be found in our user [docs](#).
- We have a [registry](#) for different containers at ALCF. A walkthrough of running an MPI container on Polaris is [here](#)
- Reach out to support@alcf.anl.gov if you have any questions.

Thank you!