Working with Jupyter on the Clusters

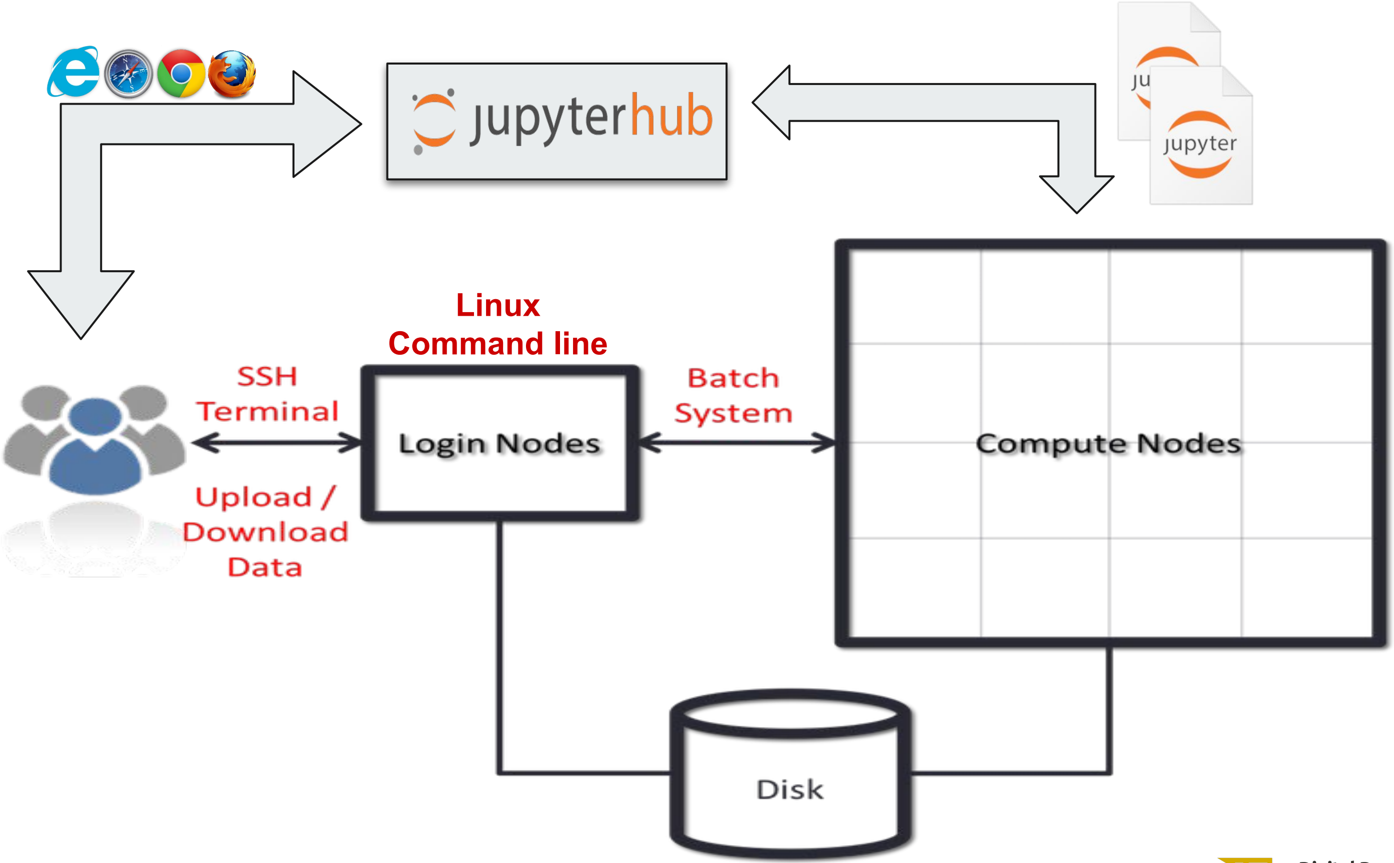
Jinhui Qin

SHARCNET | Compute Ontario | Digital Research Alliance of Canada
Western University

Outline

- Working on a cluster
- Notebook interface
  - Jupyter Notebook
  - JupyterLab
- Launching a Jupyter server
  - Via JupyterHub
  - Via Command line
- Use cases to demo
  - Case 1 : via JupyteHub without command line
  - Case 2: work with a commercial software package
  - Case 3: work with a container

Working on a cluster



Notebook Interface

Jupyter Notebook
— the classic interface

JupyterLab
— the new interface

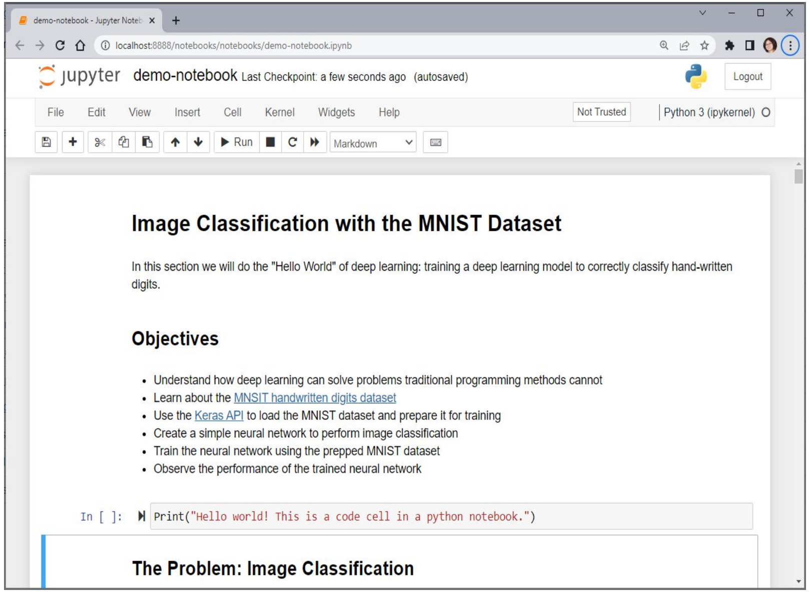

demo-notebook - Jupyter Noteb
localhost:8888/notebooks/notebooks/demo-notebook.ipynb
jupyter
demo-notebook Last Checkpoint: a few seconds ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

Image Classification with the MNIST Dataset

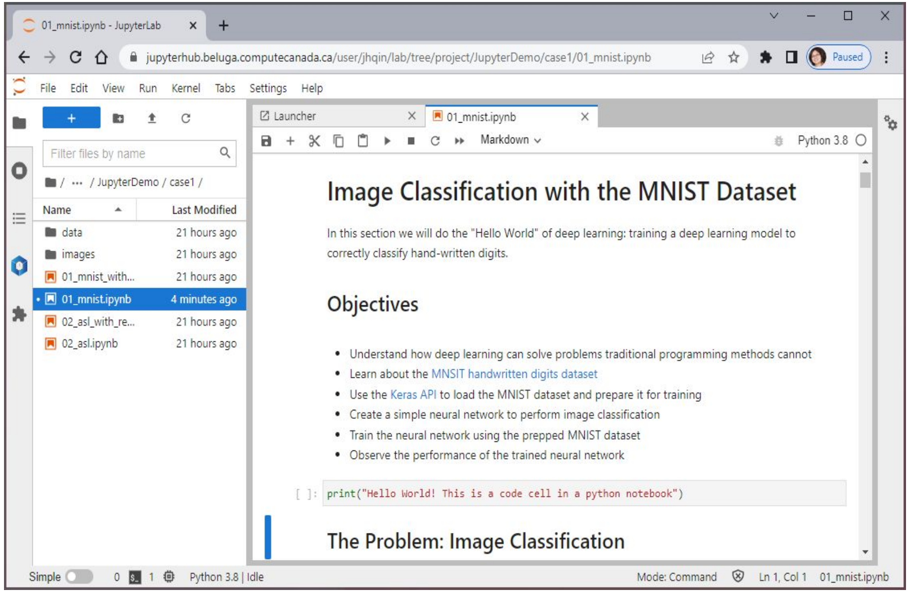
In this section we will do the "Hello World" of deep learning: training a deep learning model to correctly classify hand-written digits.

Objectives

- Understand how deep learning can solve problems traditional programming methods cannot
- Learn about the MNSIT handwritten digits dataset
- Use the Keras API to load the MNIST dataset and prepare it for training
- Create a simple neural network to perform image classification
- Train the neural network using the prepped MNIST dataset
- Observe the performance of the trained neural network

```
In [ ]:  Print("Hello world! This is a code cell in a python notebook.")
```

The Problem: Image Classification



Name	Last Modified
data	21 hours ago
images	21 hours ago
01_mnist_with...	21 hours ago
01_mnist.ipynb	4 minutes ago
02_asl_with_re...	21 hours ago
02_asl.ipynb	21 hours ago

Image Classification with the MNIST Dataset

In this section we will do the "Hello World" of deep learning: training a deep learning model to correctly classify hand-written digits.

Objectives

- Understand how deep learning can solve problems traditional programming methods cannot
- Learn about the MNSIT handwritten digits dataset
- Use the Keras API to load the MNIST dataset and prepare it for training
- Create a simple neural network to perform image classification
- Train the neural network using the prepped MNIST dataset
- Observe the performance of the trained neural network

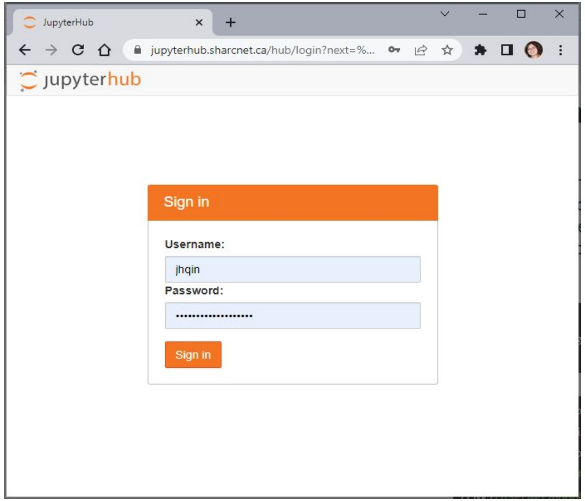
```
[ ]:  print("Hello World! This is a code cell in a python notebook")
```

The Problem: Image Classification

Launching a Jupyter server on a cluster

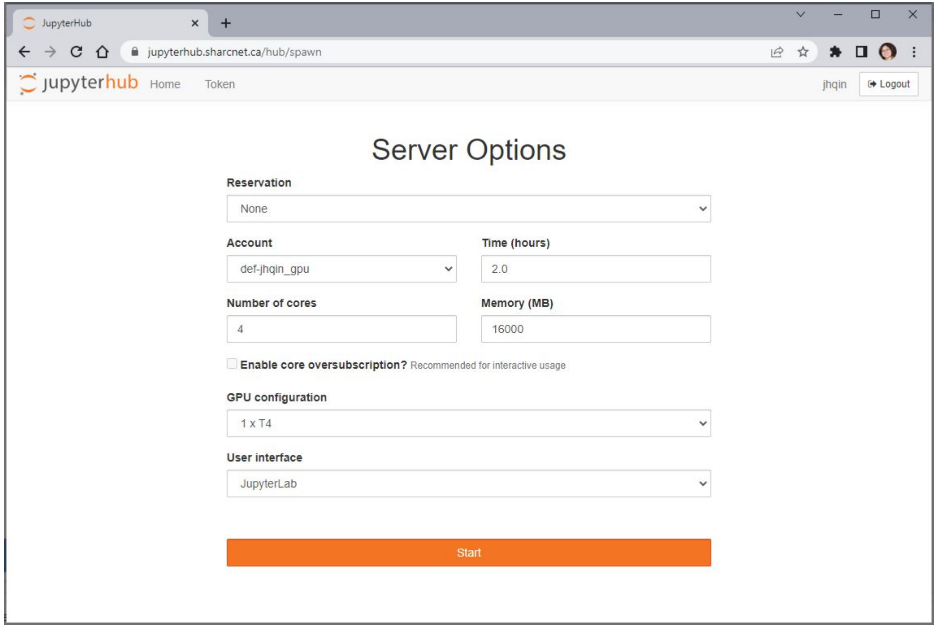
Via JupyterHub -- a web portal for launching a Jupyter server

- https://docs.alliancecan.ca/wiki/JupyterHub#JupyterHub_on_clusters



Username: jhqin

Password: ••••••••••••••••••••



Server Options

Reservation: None

Account: def-jhqin_gpu

Time (hours): 2.0

Number of cores: 4

Memory (MB): 16000

□ Enable core oversubscription? Recommended for interactive usage

GPU configuration: 1 x T4

User interface: JupyterLab

Launching a Jupyter server on a cluster

Via Command line

- ssh to a cluster
- load a python module and other modules if needed, e.g. scipy-stack
- create a python virtual env and Install jupyerlab and other libs if needed
- request an interactive session on a compute node
- launch a Jupyter server via command jupyter-lab from the virtual env.

```
[you@local]$ ssh user@cluster.alliancecan.ca
```

```
[user@cluster-login]$ module load python scipy-stack <other-modules>
[user@cluster-login]$ virtualenv --no-download ENV
[user@cluster-login]$  source ENV/bin/activate
(ENV)[user@cluster-login]$ pip install jupyterlab <other-libs>
(ENV)[user@cluster-login]$ deactivate
```

```
user@cluster-login]$ salloc --account=def-user --cpus-per-task=4 \
--mem=16G --gpus-per-node=t4:1 --time=1:0:0
[user@node###]$ module load python scipy-stack <other-modules>
[user@node###]$ source ENV/bin/activate
(ENV)[user@node###]$ jupyter-lab --no-browser --ip $(hostname -f)
```

Features

	JupyterHub	Command line
Connecting	Automatically	*Need to setup a SSH tunnel*
Default environment	Pre-configured, *temporary*	Self-made, *persistent*
Internet access from a notebook	No	No
Accessing files on a cluster	Yes	Yes
Installing packages	Only those from system wheelhouse, i.e. with *--no-index*	Possible from external, e.g. PyPI, on a login node
Working with a self-built virtual env	Possible	Yes
Using system modules	Yes	Yes
Using GPUs	Yes	Yes
Working with a container	No	Yes

Demo on Cluster
Case 1

Use Cases

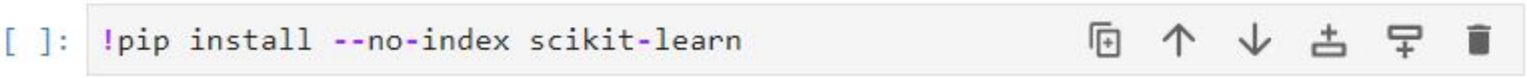
Case 1: run a python notebook for image classification on GPUs

- Launch a JupyterLab on a GPU node from a JupyterHub
- Work in the pre-configured environment
- Load modules, e.g. scipy-stack for using numpy, pandas, matplotlib, etc.
- Install additional python libraries, e.g. tensorflow
- Check GPU status if available
- Upload / Download / Export files
- No need to work in command line on a terminal

JupyteHub on clusters: https://docs.alliancecan.ca/wiki/JupyterHub#JupyterHub_on_clusters

Notes for working via JupyterHub

- If any system module is needed, always load the system modules before installing any additional python packages, and kill all running kernels
    - Click ⬡ to open the module panel and load a system module
    - Click ⏹ to open the kernel panel and kill all running kernels
- Before working on a second notebook, always kill any previous running kernel
    - Click ⏹ to open the kernel panel and kill any previous running kernel
- After a python package installation, always restart the kernel before working on a notebook
    - Click ↻ on top of the notebook panel to restart the kernel
- You can only install a python package with --no-index option on a notebook, e.g.



```
[ ]: !pip install --no-index scikit-learn
```

Demo on Cluster
Case 2

Use Cases

Case 2: run a notebook to work with a commercial package via a *persistent* python virtual environment

- An example notebook to work with Gurobi optimizer with machine learning
- Prepare a python env with required packages installed
  - Required python packages:
    - pandas, numpy, matplotlib i.e. module load scipy-stack
    - seaborn, scikit-learn   i.e. pip install --no-index seaborn scikit-learn
    - gurobipy, i.e. follow the instructions on docs wiki
- Two options for launching a Jupyter Server, ie.
  - via JupyterHub or via Command line

Use Cases

Case 2: run a notebook to work with a commercial package via a *persistent* python virtual environment

- Option 1. launching a Jupyter server from a JupyterHub
  - Add a customized kernel from *command line on the cluster*

```
(gurobiENV)[user@cluster-login]$ pip install --no-index ipykernel
(gurobiENV)[user@cluster-login]$ python -m ipykernel install --user --name gurobiEnv
```

  - Check available Jupyter kernels, remove a kernel if no longer needed

```
(gurobiENV)[user@cluster-login]$ jupyter kernelspec list
(gurobiENV)[user@cluster-login]$ jupyter kernelspec uninstall <kernel-name>
```

Use Cases

Case 2: running a notebook for using a commercial package via a python virtual environment

- Option 2. launching a Jupyter server from command line on a compute node
    - Need to install *jupyterlab in the same virtual env*

```
(gurobiENV)[user@cluster-login]$ pip install --no-index jupyterlab
(gurobiENV)[user@cluster-login]$ deactivate

[user@cluster-login]$ salloc --account=def-user --cpus-per-task=4 --mem=16G --time=1:0:0

[user@node###]$ module load python scipy-stack gurobi
[user@node###]$ source gurobiENV/bin/activate
(gurobiENV)[user@node####]$ jupyter-lab --no-browser --ip $(hostname -f)
```

Use Cases

Case 2: running a notebook for using a commercial package via a python virtual environment

- Option 2. launching a Jupyter server via command `jupyter-lab` on a compute node

```
(gurobiENV)[user@node###]$ jupyter-lab --no-browser --ip $(hostname -f)
... ...
    http://<node-hostname>:8888/lab?token=109c369980ab346bcd129eb…
```

  - Setup a SSH tunnel on a second terminal from your local computer

```
[you@local]$ ssh -L 9999:<node-hostname>:8888 user@cluster.computecanada.ca
[user@cluster-login]$
```

  - The URL to connect on a local web browser would be:

    `http://localhost:9999/lab?token=109c369980ab346bcd129eb…`

Demo on Cluster
Case 3

Use Cases

Case 3: run a notebook using packages in a container

- Working with Anaconda via an Apptainer container
  - Build an Apptainer container based from an Anaconda3 Docker container on Dockerhub
    - Docker pull command: *docker pull continuumio/anaconda3*

```
[user@gra-login]$ module load apptainer
[user@gra-login]$ apptainer build anaconda3.sif docker://continuumio/anaconda3
```

  - Request an interactive session on a compute node via salloc
  - Load the apptainer module and access the container shell

```
[user@gra-login]$ salloc --account=def-user --cpus-per-task=4 --mem=16G --time=1:0:0
[user@gra-####]$ module load apptainer
[user@gra-####]$ apptainer shell anaconda3.sif
Apptainer>
```

Use Cases

Case 3: run a notebook using packages in a container

- Launch a Jupyter server via `jupyer-lab` command from the container shell

```
Apptainer> jupyter-lab --no-browser --ip $(hostname -f)
... ...
    http://<node-hostname>:8888/lab?token=109c369980ab346bcd129eb…
```

- Setup a SSH tunnel on a second terminal from your local computer

```
[you@local]$ ssh -L 9999:<node-hostname>:8888 user@cluster.computecanada.ca
[user@cluster-login]$
```

- The URL to connect on a local web browser would be:

  `http://localhost:9999/lab?token=109c369980ab346bcd129eb…`