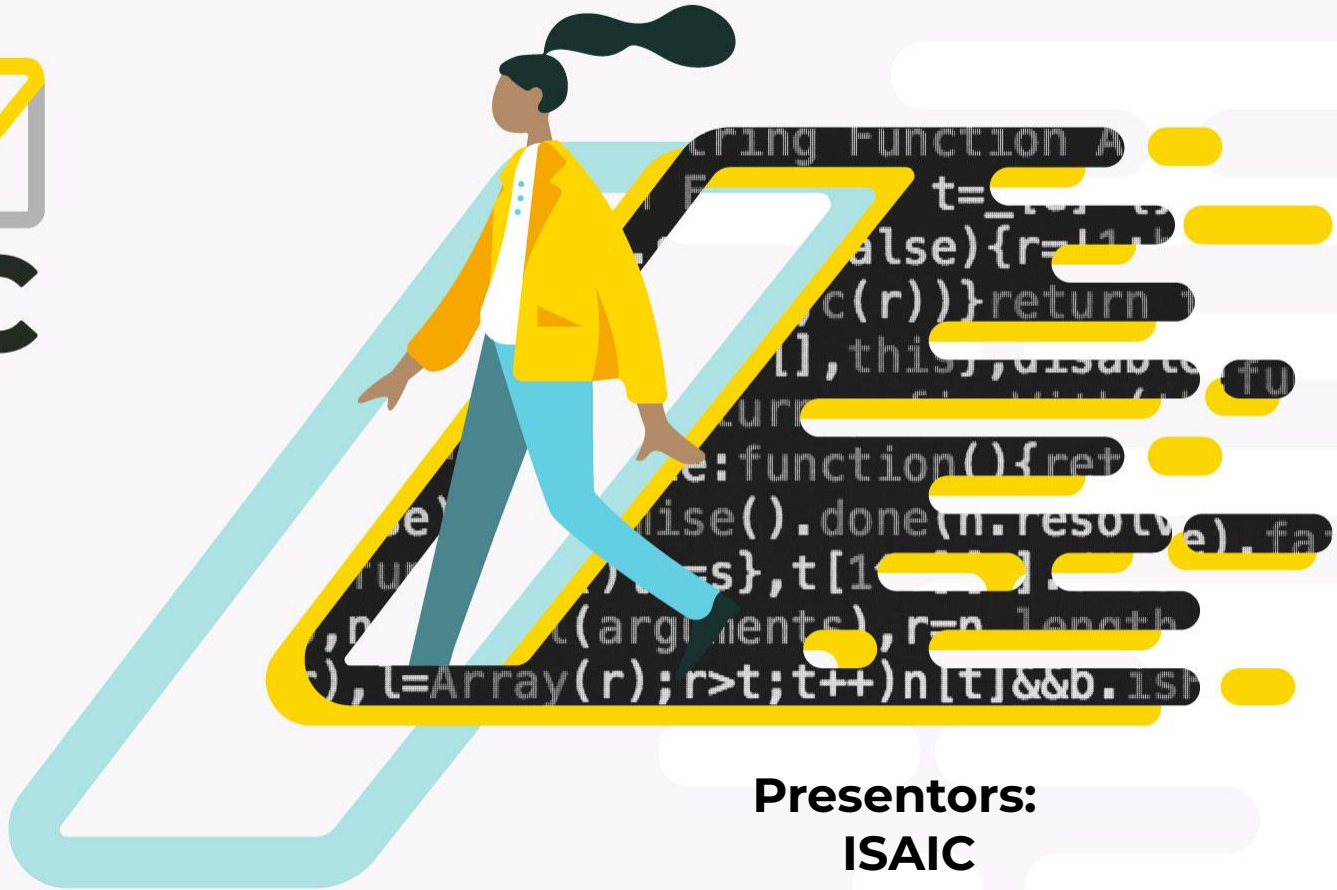
ISAIC

Industry Sandbox
& AI Computing

JupyterHub



Presentors:
ISAIC

Today’s Discussion

- **About JupyterHub and Jupyter Notebooks**
- **How to Access JupyterHub**
- **Login into JupyterLab**
- **JupyterLab Environment**
- **How to install different Libraries in JupyterLab**
- **Difference between Jupyter Notebooks and Python Files**
- **Different Python Modules in Data Science and Neural Networks**

Today’s Discussion

- **Shell Command in Jupyter Notebook**
- **GPU Information**
- **Out Of Memory Error**
- **Convenient Features/Commands**
- **Download a Notebook**
- **Upload files in the File Directory**
- **ISAIC’s Role in JupyterHub**
- **Admin Controls**

About JupyterHub and Jupyter Notebooks

- JupyterHub is the best way to serve Jupyter notebook for multiple users. It can be used in a class of students, a corporate data science group, or scientific research group. It is a multi-user Hub that spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server.

- The Jupyter Notebook is an open-source web application that allows you to interact with data, create and share documents that contain live code, equations, visualizations, narrative text and program using python, Julia, Spark etc.

- We provide ISAIC branded the Littlest JupyterHub which is a branch of JupyterHub. It is small scale, easy to install, deploy and manage JupyterHub which is available for Ubuntu 18.04 or Ubuntu 20.04 on a amd64 or arm64 CPU architecture.

- With the proxies and how it handles port forwarding one IP address can be representative of the Hub and hence multiple users can login and use Jupyter Notebooks from that one IP address.

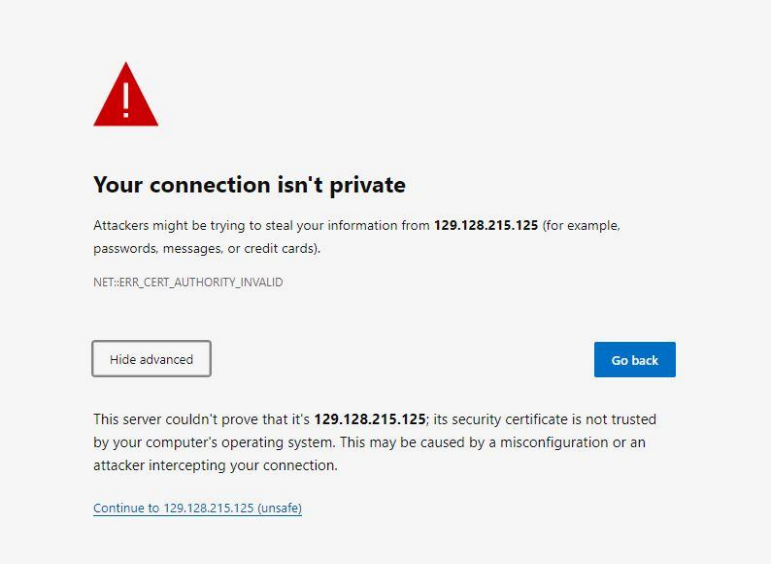
How to Access JupyterHub

- Go to the browser and enter the IP address Link that has been provided to you

- Note that It says the connection is not secure. This is because we use a self signed SSL certificate and the browser doesn't recognise it

- Click on the "Advanced" button and then click on the "proceed to <IP address> (unsafe)" link at the bottom

  In case there is no Advanced option Just follow the instructions in this link https://dev.to/brettimus/this-is-unsafe-and-a-bad-idea-5ej4



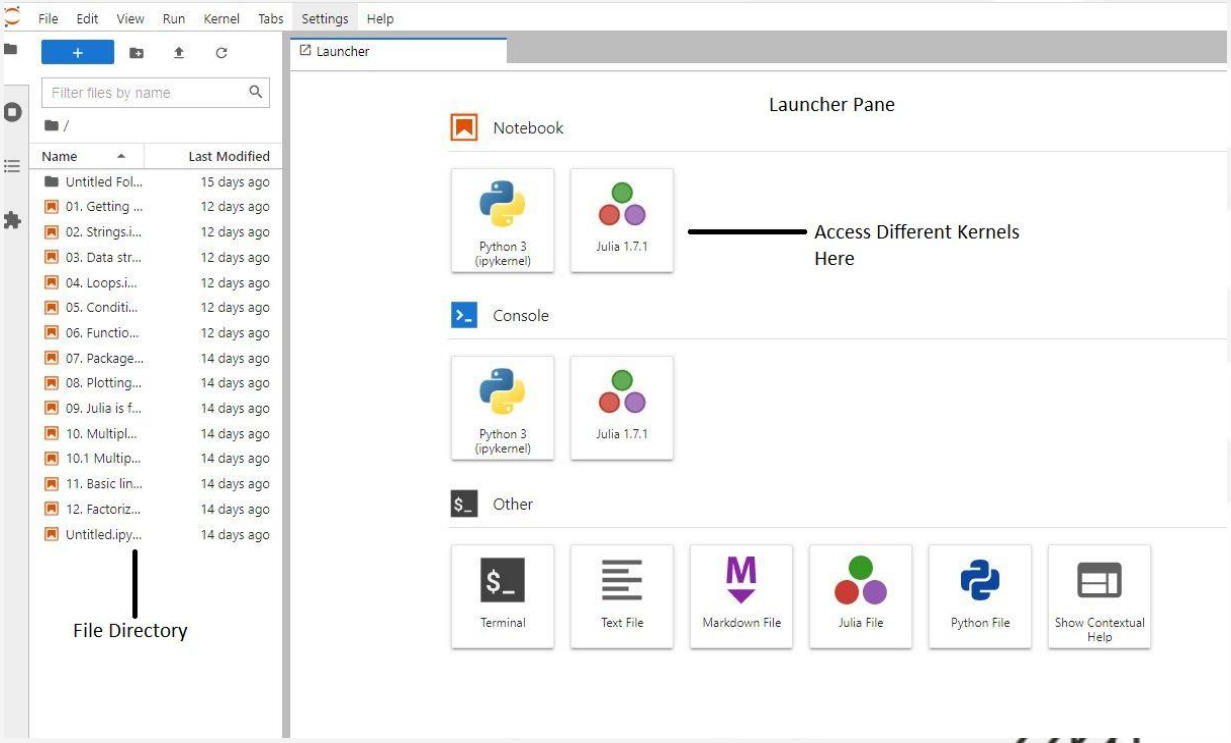
Login into JupyterLab

- This should take you to the login page of the JupyterHub
- Here you enter the username that has been assigned to you and create your own password that you will use to login from the next time
- This should take you to JupyterLab home page.

JupyterLab Environment

JupyterLab is the **latest web-based interactive development environment for notebooks, code, and data**

- In the launcher pane you can create a jupyter notebook
- In the file directory you can access the newly created notebook or you can upload existing notebooks
- When you right click on a notebook in the file directory you are also able to:
  1. Rename the notebook
  2. Delete the notebook
  3. Download the notebook
  4. And, Other basic functions
- To create a new python notebook you have to click the python3 kernel under the Notebook title. It creates an untitled.ipynb notebook.(IPYthon NoteBook)



How to install different Libraries in JupyterLab

- To run any libraries in the notebook you have to download the libraries in the terminal. For example if you cannot run Numpy in the notebook then you have to download Numpy in the terminal that you can find in the launcher page using this code:

  $ sudo -E pip install numpy

  ```
  jupyter-aryan@jupytertest:~$ sudo -E pip install numpy
  ```

  Or you can run the same command in the Jupyter Notebook by preceding it with an exclamation mark “!” as follows:

  ! sudo -E pip install numpy

  ```
  [1]: ! sudo -E pip install numpy
  ```

- Now if you restart your kernel from the kernel tab you will see that you are able to run numpy

- This way you can download any libraries that are not pre installed

- In the JupyterHub launcher you can also open a python file under the “Other” section

Difference between Jupyter Notebook and Python File

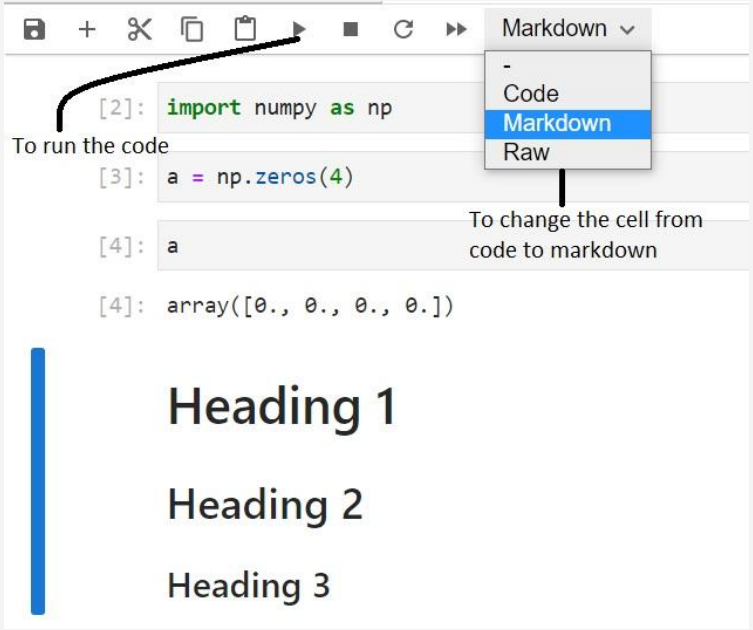
- The main difference between a Jupyter Notebook and a python file is that in the notebook you can run each block of code separately and this makes it easier to debug your code and only run the cell that you want, whereas in a python file it is harder to locate the bugs and even after solving the bugs you will have to run the whole code each time making it inefficient

- You can run the Jupyter Notebook by just clicking the run button where as to run the python file you have to use the terminal and run the file similar to:

    - $ python3 test.py

- You can add headings in the Notebook as well by changing the cell to a markdown and using '#' to change the size of the heading. For eg: '# Heading 1' will be larger than '## Heading 2' in the markdown.

Jupyter Notebook



Python File

```python
import numpy as np
a = np.zeros(4)
print(a)
```

To run a Python File in the terminal:

```
jupyter-aryan@jupytertest:~$ python3 isaictest.py
[0. 0. 0. 0.]
```

(Here isaictest.py is the name of the python file.)

Different Modules for Data Science and Neural Networks

- Python is adapted by data scientists more than any other language

- It has good support for neural networks and data science modules
  - For e.g.:
    1. Apache MXNet
    2. PyTorch
    3. Tensorflow
    4. Keras
    5. Scikit-Learn
    6. Etc.

Shell Command in Jupyter Notebook

To use shell command in the notebook you can use the '!' before your command and it will work.

For e.g.:

```
!nvidia-smi
```

```
Fri Jan 28 08:39:22 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.29.05    Driver Version: 495.29.05    CUDA Version: 11.5     |
|-------------------------------+----------------------+----------------------+
| 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-PCIE...  Off  | 00000000:00:06.0 Off |                    0 |
| N/A   32C    P0    35W / 250W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-PCIE...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   31C    P0    35W / 250W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

GPU information

At ISAIC when we provide JupyterHub we sometimes include GPUs for the users.
If your service comes with GPUs then the 'nvidia-smi' command that was used in the last slide gives you the information on our GPUs.

It tells us:

- The driver version
- CUDA version
- Memory usage (out of 16160MB)
- Number of GPUs (2 for this example server)
- Processes running on the GPUs
- Power usage of the GPUs
- ....

```
jupyter-aryan@jupytertest:~$ nvidia-smi
Fri Jan 28 09:41:40 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.29.05    Driver Version: 495.29.05    CUDA Version: 11.5     |
|-------------------------------+----------------------+----------------------+
| 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-PCIE...  Off  | 00000000:00:06.0 Off |                    0 |
| N/A   32C    P0    35W / 250W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-PCIE...  Off  | 00000000:00:07.0 Off |                    0 |
| N/A   31C    P0    35W / 250W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Out Of Memory Error

- In some cases when the users are running their codes in the GPUs, based on the configuration of the model for eg: PyTorch the code by default runs on GPU 0. Hence if there are a lot of users using this configuration then they are all using the same GPU meaning GPU number 0. So once the memory of the GPU is completely used then any users who try running their code will get an “out of memory” error.

- Solution: When this happens you need to tell the python kernel to use a different GPU that has compute resources available.

- We can specify which PCI Express devices are available to each kernel hence specifying which GPU to ultimately used as they are PCI Connected devices on the servers.

Command for specifying which PCIE connection to use on the OS:

- For python file (in the terminal)- CUDA_VISIBLE_DEVICES=x python ./prog.py

  Here ‘x’ is the free GPU index that the user wants to use that you can get by running and observing the resource usage through the Nvidia-smi command. The user can get the GPU index from the nvidia-smi command and ‘prog.py’ is the name of your python file. This parameter can be passed on through the command line as follows:

```
jupyter-aryan@jupytertest:~$ CUDA_VISIBLE_DEVICES=1 python ./isaictest.py
[0. 0. 0. 0.]
```

  Here the code was executed on the second GPU since the indexing starts from 0.

- For Jupyter Notebook code to specify which GPU- import and use the os Module;

  os.environ[‘CUDA_VISIBLE_DEVICES’]=”x”

  Here ‘x’ is the free GPU index that the user wants to use.

```
import os;
os.environ['CUDA_VISIBLE_DEVICES']="0"
import numpy as np
```

(Note: This code should be the first block of code should be executed in the notebook)

Convenient Features/Commands

- When we use nvidia-smi command it gives us the details for a specific time stamp. Instead we can use a command to automatically call nvidia-smi in regular time intervals: watch -n t nvidia-smi
  Here, “t” is time in seconds

```
jupyter-aryan@jupytertest:~$ watch -n 2 nvidia-smi
```

- This command will update the nvidia-smi table every 2 seconds. You can set “t” to whatever interval you want.

- Another convenient feature is that you can drag the terminal to the side or bottom of your screen and run your Notebook or python file simultaneously while having the terminal open.

- Just drag by your mouse until the panes rearrange on the next slide there is an example of this.

isaic.ipynb | isaictest.py | jupyter-aryan@jupytertest: ~

Code | Python 3 (ipykernel)

[13]:
```
import os;
os.environ['CUDA_VISIBLE_DEVICES']="1"
import numpy as np
```

[8]:
```
a = np.zeros(4)
```

[9]:
```
a
```

[9]:
```
array([0., 0., 0., 0.])
```

[10]:
```
!nvidia-smi
```

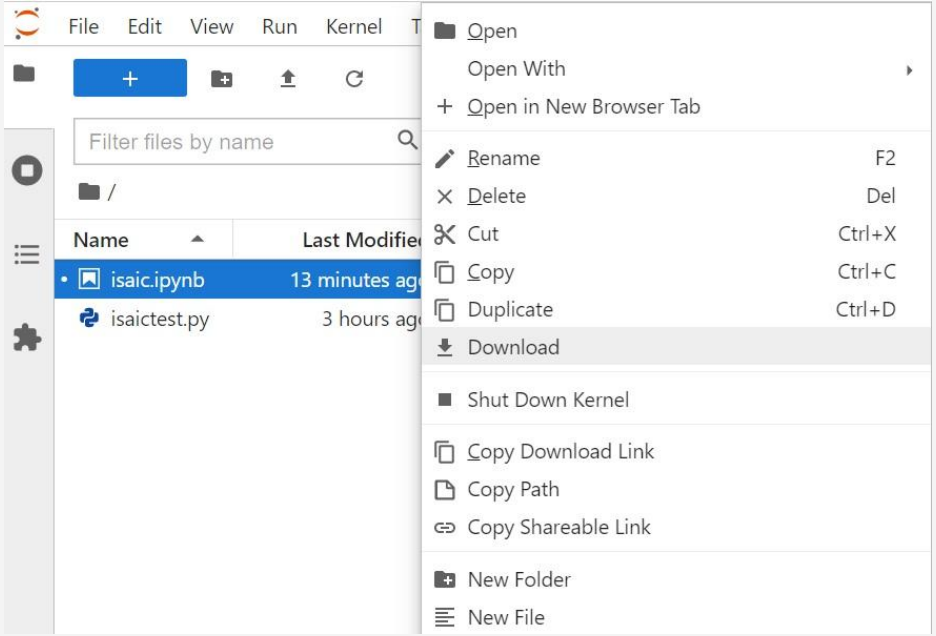
jupyter-aryan@jupytertest: ~

```
Every 2.0s: nvidia-smi                                          jupytertest: Fri Jan 28 11:04:26 2022

Fri Jan 28 11:04:26 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.29.05    Driver Version: 495.29.05    CUDA Version: 11.5     |
|-------------------------------+----------------------+----------------------+
| 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-PCIE...  Off  | 00000000:00:06.0 Off |                    0 |
| N/A   33C    P0    35W / 250W |      0MiB / 16160MiB |      0%      Default |
|                               |                      |                  N/A |
+--------2--------------------------------------------------------------------+
```

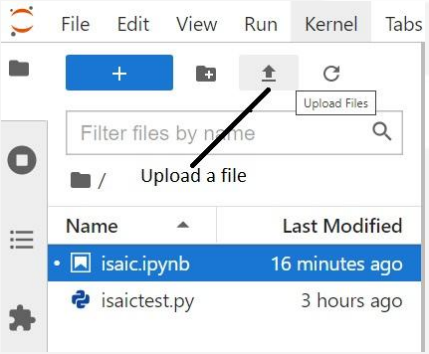
Download Files

- When you download your notebook (or any files), it gets downloaded in your downloads folder of your computer. Following you can see how one can download their notebook by right clicking on the file for the corresponding notebook in the File browser pane

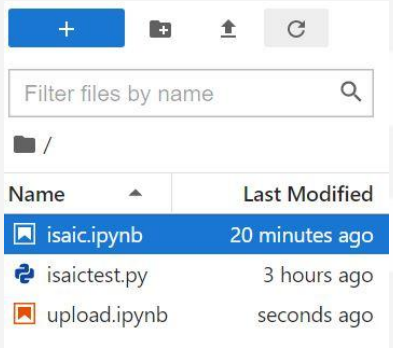


Upload Files in the File Directory

- You can also upload any file that you want into the file directory:



- Once uploaded the file will show up in the current file directory that you have added this to:



ISAIC's Role in JupyterHub

- We can limit users to a specific number of CPU cores and available memory.
- We can set the idle time culling to a certain time which means that if the kernel has been ideal for more than the specified time then it automatically shuts down releasing the memory in the GPU.
- Password Recovery

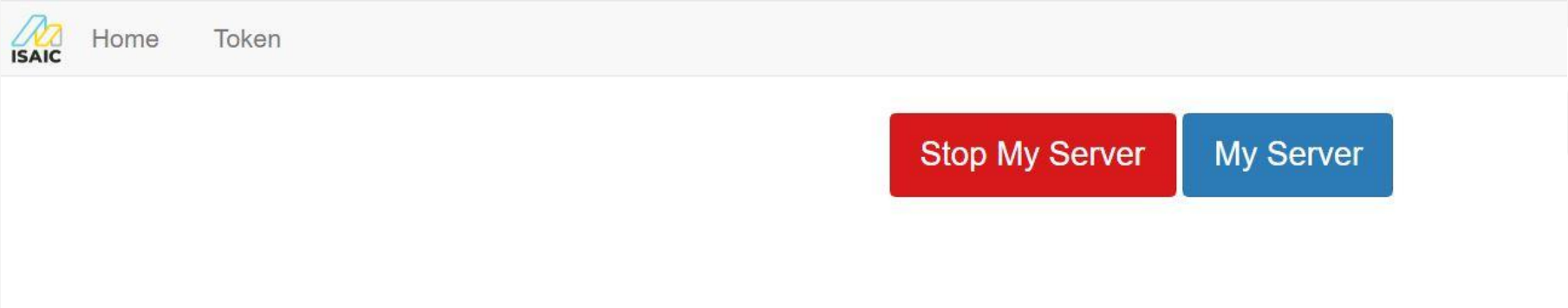
Shutting Down your Kernels

- When you complete running your code and send some data to the GPU, the memory of the GPU device gets utilised and even after the code is finished the kernel is loaded in the GPU hence the kernel keeps using the memory of the GPU for no reason at all.
- After your computations are done and you have saved your models. To free up the memory of the GPU you need to shutdown the kernel. We usually solve it by using an specified timeout for the idle kernels but shutting down the kernel is a good practice to develop when you finish executing your code to free up resources for others to use.

Hub Controls For Users

You can access the controls from the ‘hub control panel’ under the File tab in the JupyterLab.
Here you can:

- Stop/start your own server (Don't ShutDown the HUB on your own and ask ISAIC for this)
- Under the token tab you can set your API. For e.g. you can connect your Visual Studio code



ISAIC Home Token aryan Logout

Request new API token

Note

note to identify your new token

This note will help you keep track of what your tokens are for.

Token expires

Never

You can configure when your token will be expired.

API Tokens

These are tokens with full access to the JupyterHub API. Anything you can do with JupyterHub can be done with these tokens. Revoking the API token for a running server will require restarting that server.

Note	Last used	Created	Expires at	
Server at /user/aryan/	a minute ago	a minute ago	Never	revoke
Server at /user/aryan/	17 days ago	17 days ago	Never	revoke
Server at /user/aryan/	19 days ago	19 days ago	Never	revoke

Hub Controls For Admin

- Under the Admin tab you get access to all the users
- You can see when all the users were last active
- You can add, delete, and edit users
- You can start and stop all the servers

ISAIC Home Token Admin aryan Logout

User	Admin	Last Activity	Running (1)		
Add Users			Start All	Stop All	Shutdown Hub
aryan	admin	3 minutes ago	stop server	edit user	
charles	admin	a month ago	start server	edit user	delete user
omid	admin	17 hours ago	start server	edit user	delete user
sourav	admin	2 days ago	start server	edit user	delete user

Displaying users 1 - 4 of 4