

Step by step guide on how to access NSCC supercomputer and open up jupyter notebook

This guide is done by Lim Zi Xiong and Bryan Wong Wen Ping from Singapore Polytechnic.

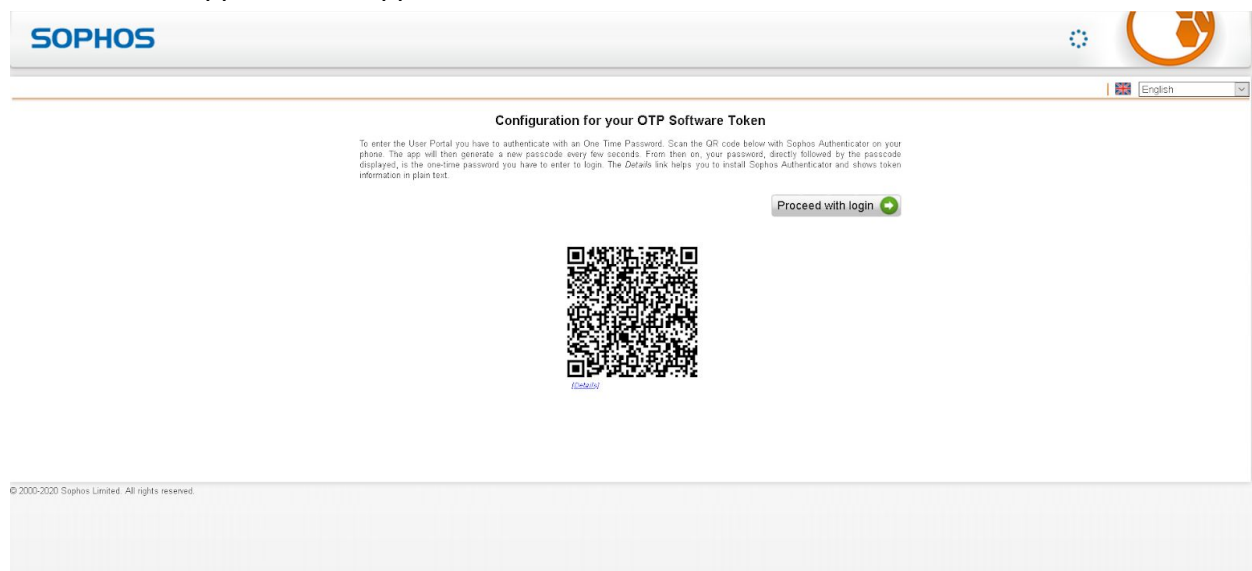
Required software

- Sophos SSL Vpn - download via NSCC website
- Sophos authenticator app - download on the app store on your mobile device.
- A terminal capable of ssh eg putty - <https://www.putty.org/>
- Remote file access software eg winscp - <https://winscp.net/eng/download.php>
- A linux environment if using windows, ubuntu for windows is available to download at windows store.

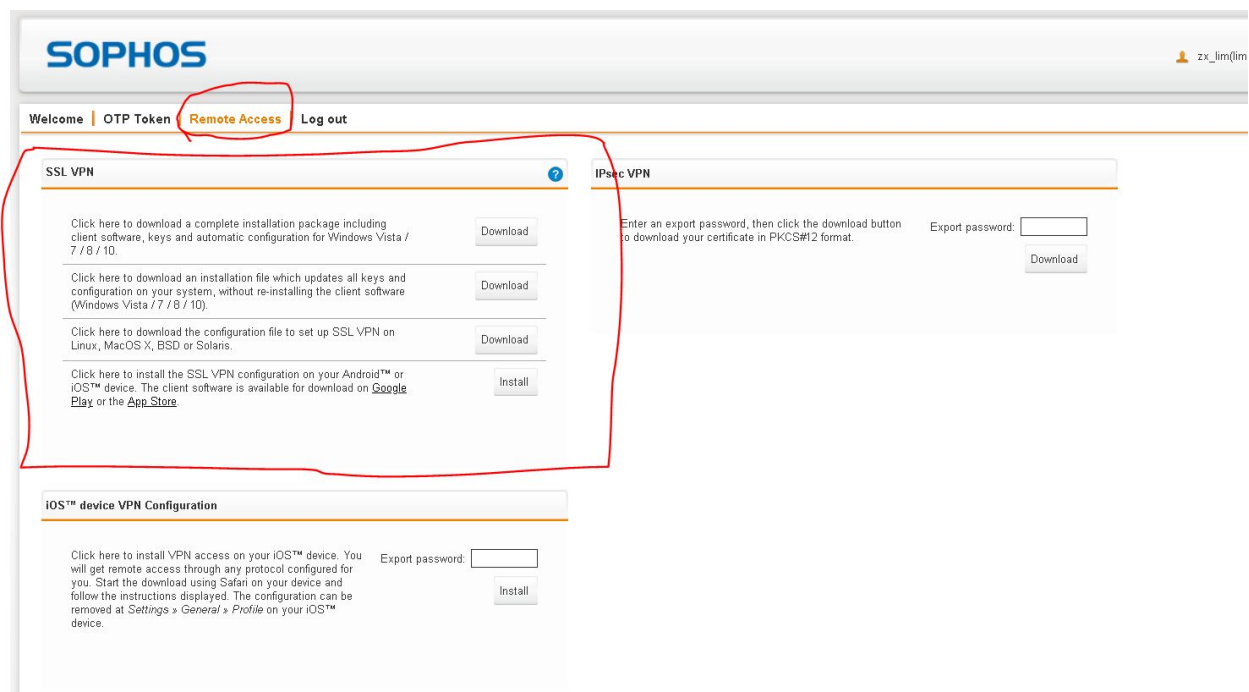
Step 1 - download the vpn and login to the vpn

-Visit <https://vpn.nsc.sg/> to download the vpn client.

-When you first login using the password of your NSCC account, you will be prompted to scan the QR code. The QR code has to be scanned via the sophos vpn authenticator app so download the app from the App store.



-Once the above step is done, you will have to log out and login again but this time in the password section you will have to include the number generated from the authenticator app. For example, my password from the NSCC is supercomputer123 and the code generated is 123456, my password will be supercomputer123123456. Once you have successfully logged in, you will see a new interface. Click on remote access on the top left corner and download the vpn client that is suitable for your OS.



-Run the installer. Once the installer is completed, look for Sophos SSL VPN client. You will be prompted to login and the format is the same as the above step, username will be your NSCC username and the password will be the password with the code from the authenticator app.

SSL VPN - User Authentication

Username:

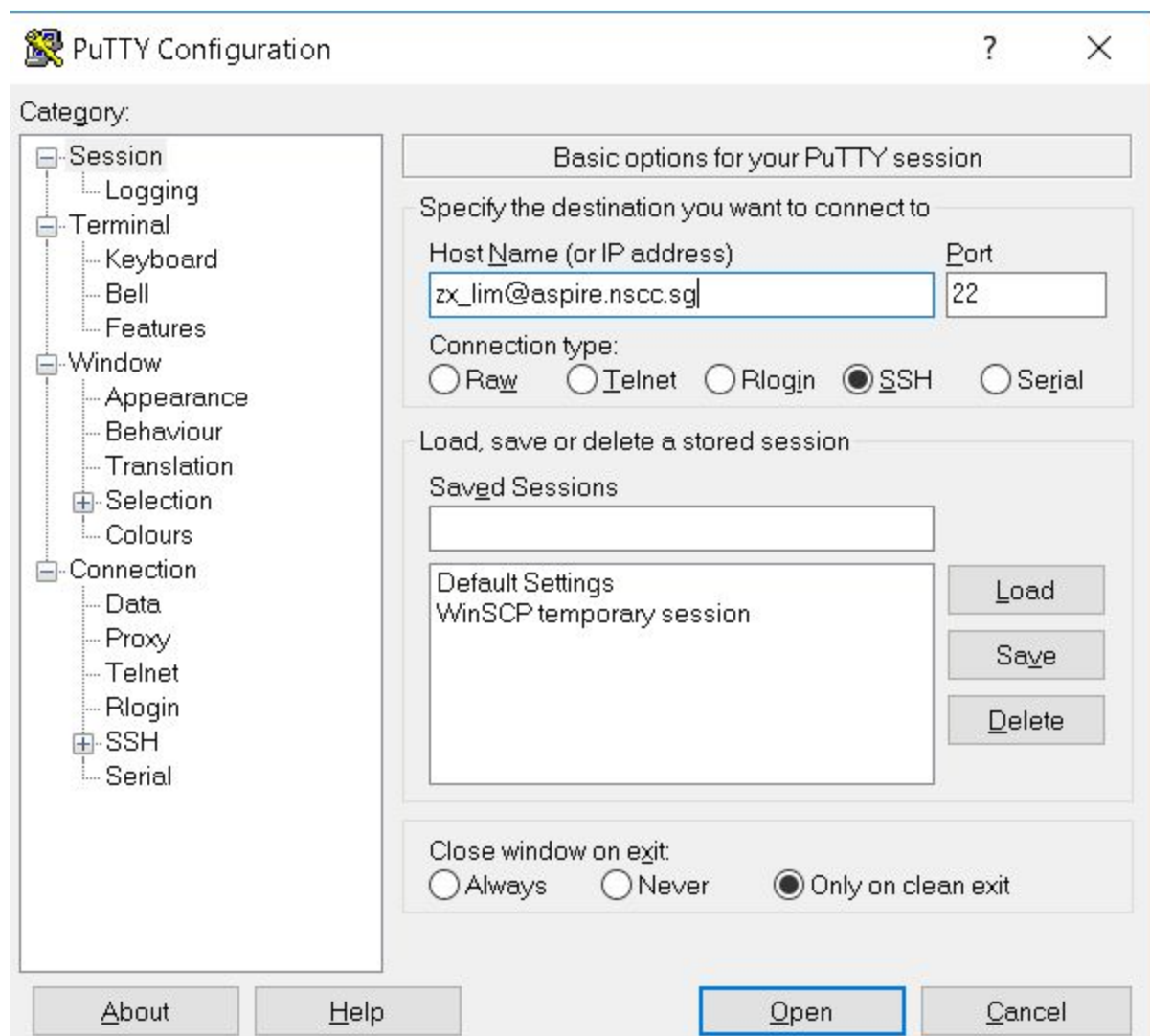
Password:

-If the login credentials are correct, the icon will turn from yellow-orange to green and this means that you have successfully connected to the NSCC vpn.

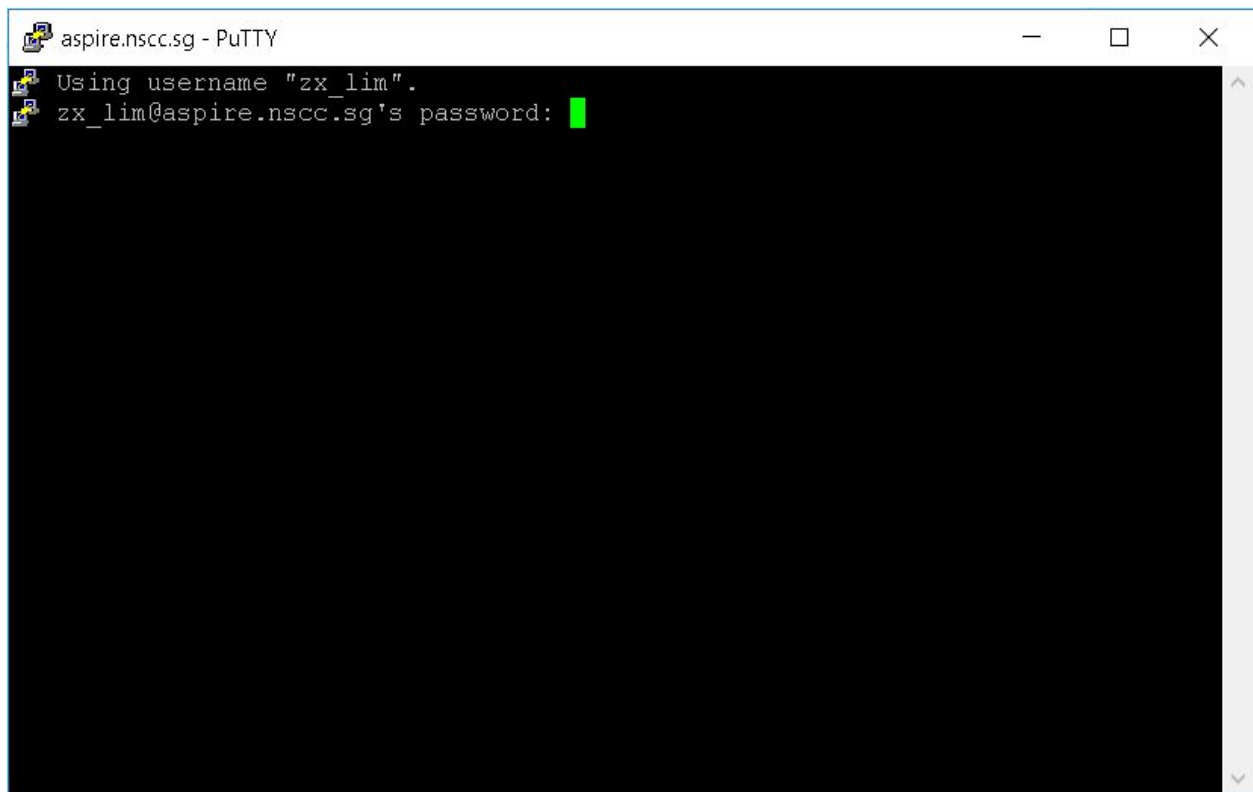


Step 2 - connecting to the nscs aspire facility via SSH

-In step 2, open up your terminal capable of SSH in this case i will be using putty.



-As you can see above, you are required to connect to a host name/address. The host name format is <NSCC username>@aspire.nscs.sg. After that click on open.



-You can see from above that once you click on open, you are prompted to key in your NSCC password. The password will just be the password of your NSCC account without the authenticator code.

-Once you have successfully logged in, you will see the overview of your account eg usage the last 7 days. In this terminal we can do several things by sending a particular command. To start a new job, we use qsub, to delete a job, we use qdel, to view the status of the job, we use qstat.

Step 3 - submitting a job script

-Submitting a job requires you to use a script file in the format of SH or pbs. When submitting a job, a few things must be specified inside.

-Look for runjupyter Singularity.pbs. Inside this script will have comments about what each line does so you can adjust accordingly if required.

-Before we move the script to the NSCC facility, we have to generate a hash in order to use the jupyter notebook.

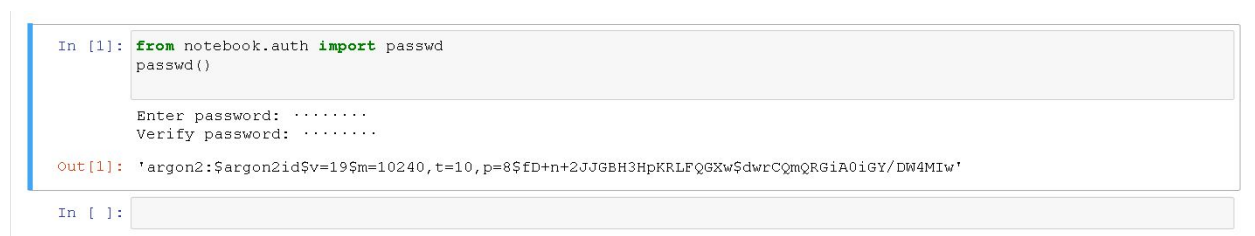
-Go into the jupyter notebook in your local computer and input this code and click on run.

```
from notebook.auth import passwd  
passwd()
```



```
In [*]: from notebook.auth import passwd  
passwd()  
  
Enter password:   
  
In [ ]:
```

-You will be prompted to key in a password just key in what you one but make sure you remember and note it somewhere. After entering and confirming the password, a hash will be generated.



```
In [1]: from notebook.auth import passwd  
passwd()  
  
Enter password: .....  
Verify password: .....  
  
Out[1]: 'argon2:$argon2id$v=19$m=10240,t=10,p=8$FD+n+2JJGBH3HpKRLFQGXw$dwrcQmQRGiA0iGY/DW4MIw'  
  
In [ ]:
```

-Copy the output and paste it in the pbs script

```
#PBS -N JupyterSingularity

### Start of commands to be run

# Singularity image to use for container
image="/app/singularity/images/tensorflow/tensorflow_2.3.0_gpu_py3.simg"

### We need export these environment variables to container
# NoteBook Port Between 8000 and 9999
export NOTEBOOKPORT=8899
# Hash
export Hash='Put the hash output here|'

# Please note that when you start a container then it will start
in a directory defined by the image
# You will also need to change to the correct directory inside
the container
echo
echo Job should start in your home directory:
pwd
echo Change to directory where job was submitted:
cd "$PBS_O_WORKDIR" || exit $?
pwd
echo

# See which node job is running on and GPU status
echo Shell hostname:
hostname
echo
echo Shell nvidia-smi:
nvidia-smi
echo

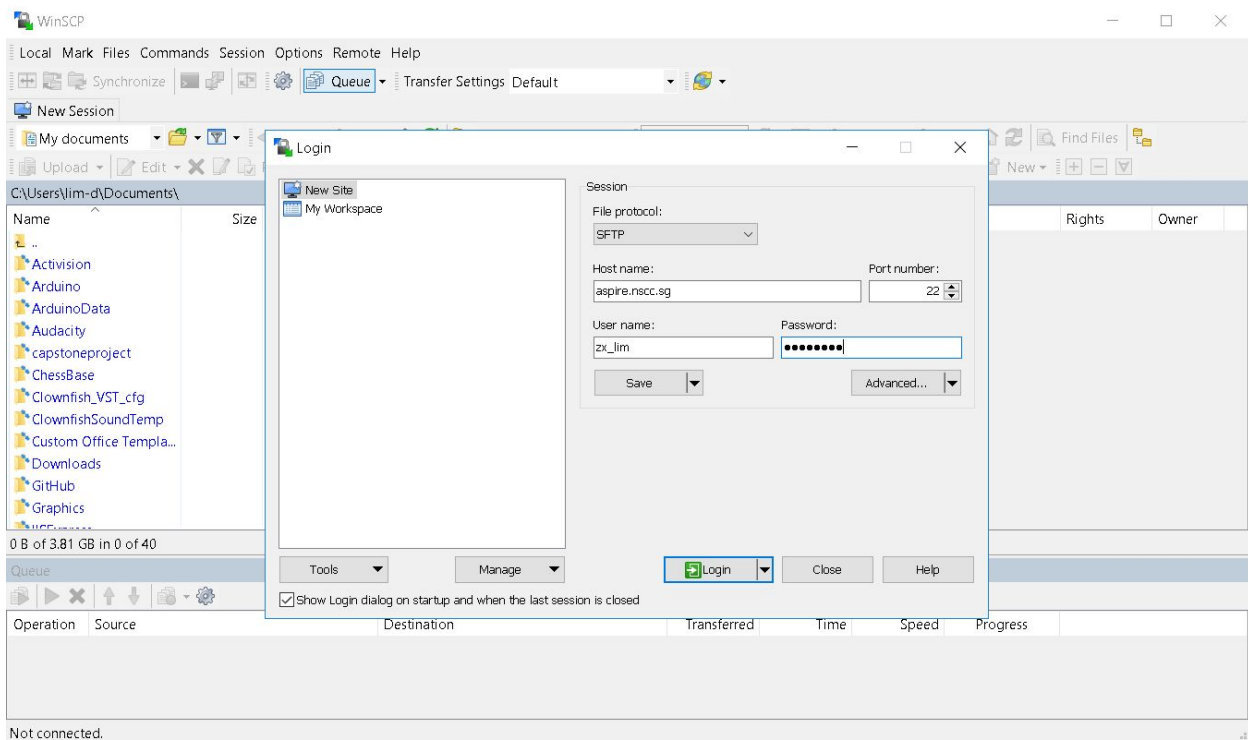
echo Loading singularity module:
module load singularity
echo

echo View Environment Variables of Shell:
env
echo

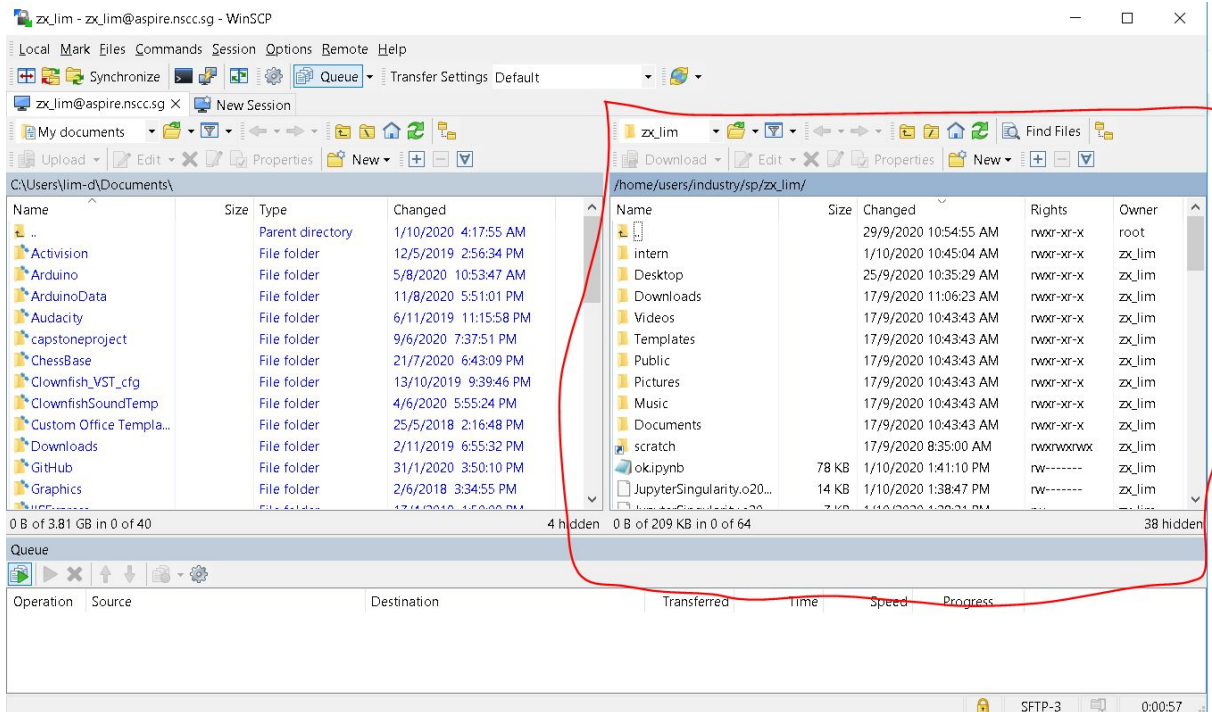
echo Variables defined in script:
echo image:
echo $image
echo NOTEBOOKPORT:
```

-Now that we have done that, we can move the script over to NSCC facility in order to submit a job.

-In order to submit a job, we have to move the script from our local computer to the NSCC facility. To do so, we will open up winscp.



-Click on new session and you should see a new window popup prompting you to input hostname, username and password. Host name will be in the format of <username>@aspire.nscg.sg. Click on login and you should see that you have access to the files in your NSCC.



-Once we are here, move the pbs script into any directory and note down the directory. In this case I will place my script in /home/users/industry/sp/zx_lim/intern.

-Now that we have moved over the script, we can now submit a job using the script. Go to your terminal eg putty and type in the following

`qsub /home/users/industry/sp/zx_lim/intern/nameofscript.pbs`



-You should see a 7 digit number .wlm01 and this means that a job is submitted with that ID.

-Now that we have submitted a job, we have to track the status of the job and ensure that the job is running. The resources at NSCC are used by people all over singapore so there may be an instance where there is a shortage of CPU or GPU. To track our job status, type qstat -s on your putty terminal.

```
zx_lim@nsc04:~  
[zx_lim@nsc04 ~]$ qsub /home/users/industry/sp/zx_lim/intern/lol.pbs  
2046537.wlm01  
[zx_lim@nsc04 ~]$ qstat -s  
  
wlm01:  


| Job ID        | Username | Queue    | Jobname    | SessID | NDS | TSK | Req'd<br>Memory | Req'd<br>Time | Elap<br>S | Time  |
|---------------|----------|----------|------------|--------|-----|-----|-----------------|---------------|-----------|-------|
| 2046537.wlm01 | zx_lim   | gpunorma | JupyterSin | 19641  | 1   | 1   | 105gb           | 02:00         | R         | 00:01 |

  
Job run at Thu Oct 01 at 14:08 on (gpu2035:ncpus=1:ngpus=1)  
[zx_lim@nsc04 ~]$
```

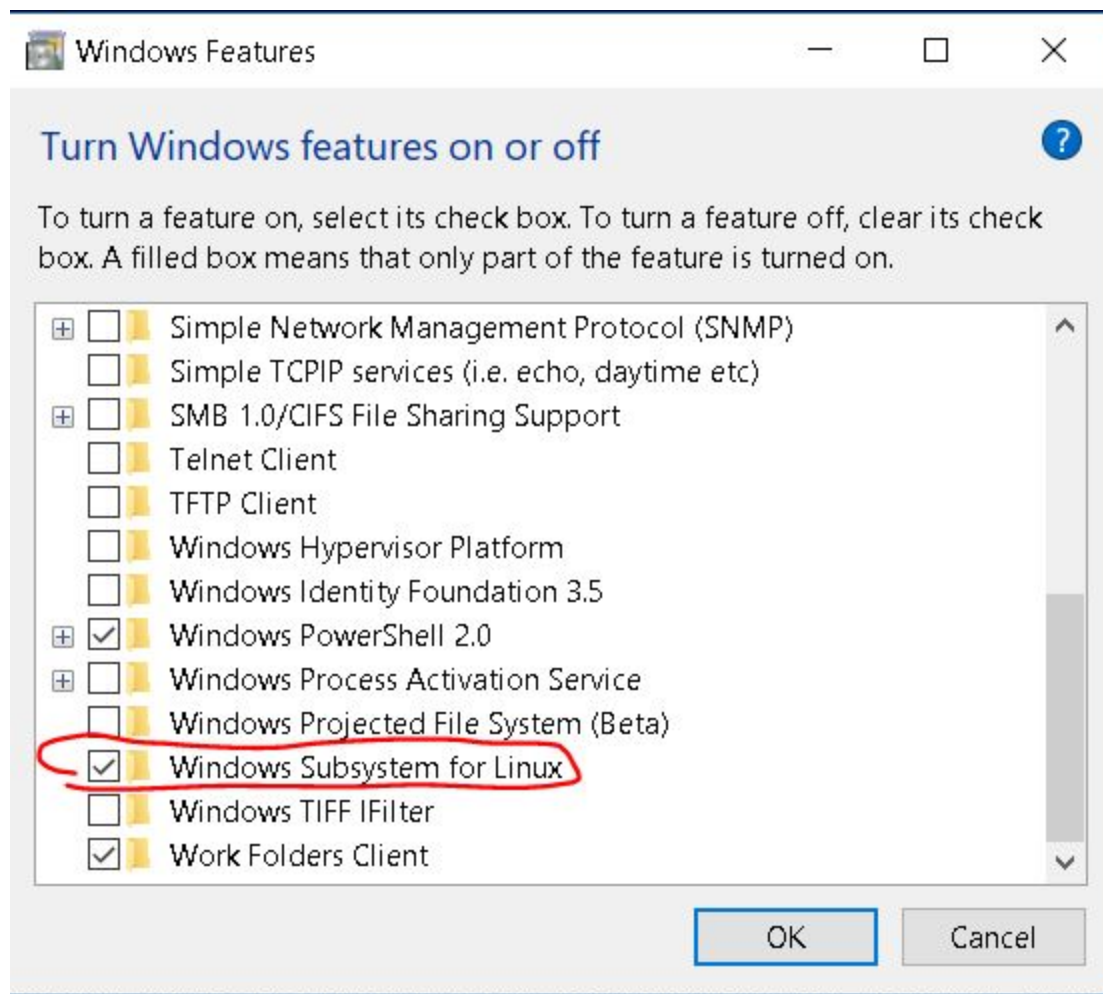
-You can see that our job is running as shown in the column 'S' which is good news but there may be an instance where in the 'S' column it shows 'Q', this means that the job is on queue and you will have to wait sometime before the system provision the resources you requested.

Step 3 - connecting to jupyter notebook via local computer

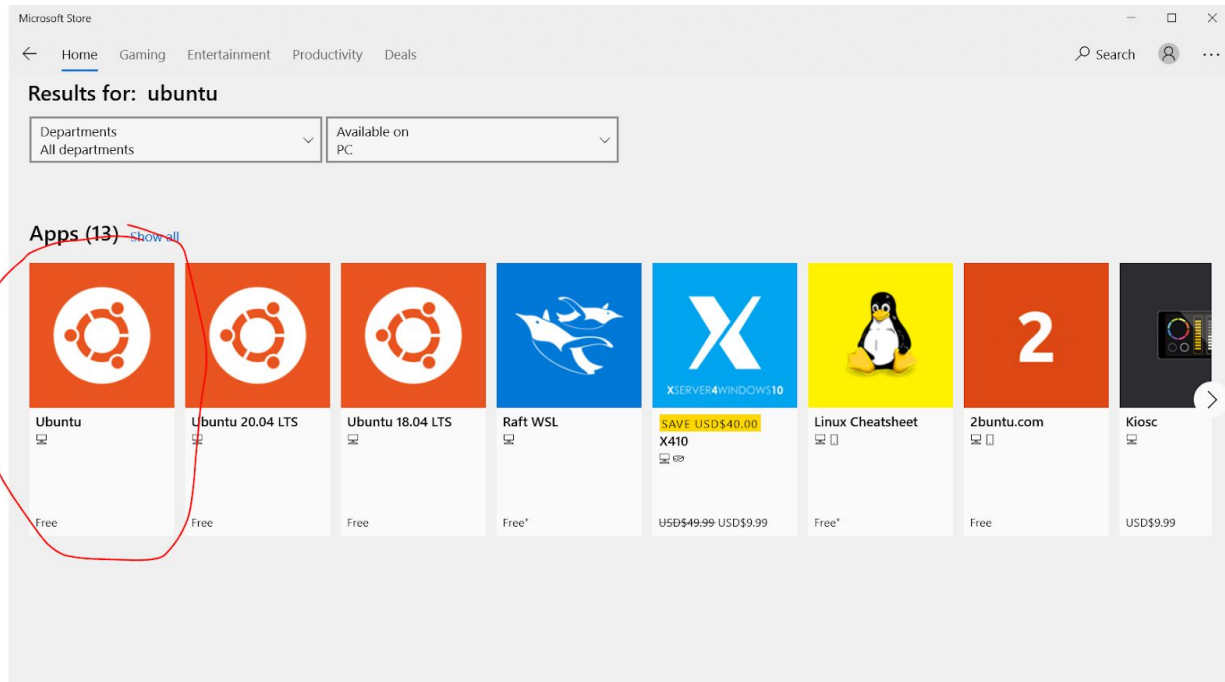
-In step 3, we will set up and run the jupyter notebook on our local computer but using the gpu and cpu from the NSCC to run our code.

-This step requires a system that is capable of running linux command line. Since we are running windows, we need to download ubuntu for windows in the microsoft store.

-Go to your windows start button and type in 'Turn windows features on and off' and look for Windows subsystem for linux. Enable it and close the window.



-You will be asked to restart but do not restart first as we have to install ubuntu for windows. On your start button type microsoft store and search for ubuntu. Install it and restart your computer.



-Now that we have successfully installed ubuntu on our windows pc, we can proceed to establishing a tunnel connection from our local pc to NSCC facilities. In this step, open up your notepad++ and name it tunnel.sh. Inside, paste the content below to your notepad.

```
ssh -N -L localhost:<LOCAL_HOST_PORT>:<HOST_NAME>-ib0:<NOTEBOOK_PORT>  
<USER_NAME>@aspire.nsc.sg
```

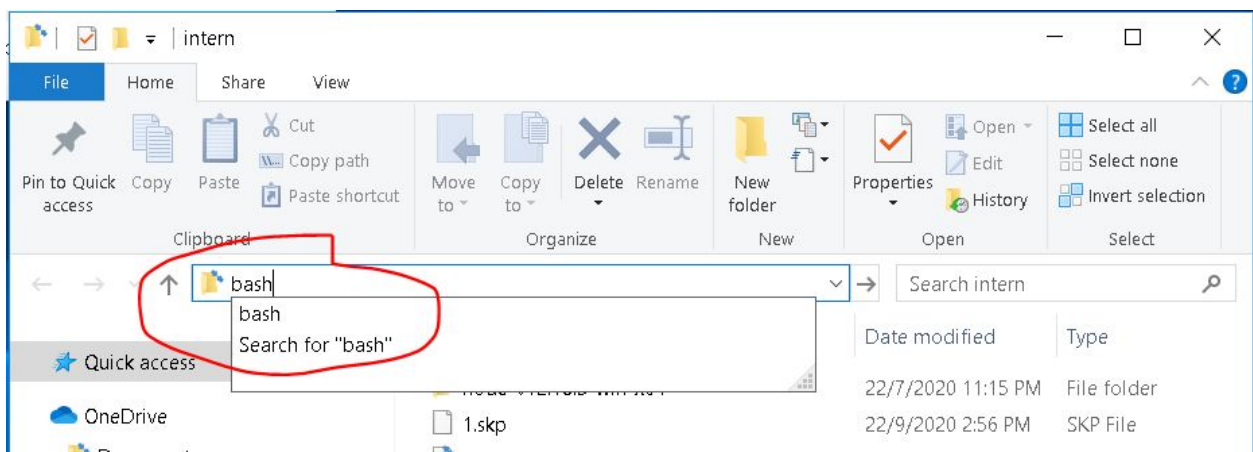
-Since in the above PBS script we specified that we want to run the jupyter notebook on port 8899 and we want to use 8899 on our local computer as well, we will key in 8899 as our local host port and notebook port. For the hostname, go to putty and type `qstat -f <id of job>` and look for 'exec_host'.

```
zx_lim@nscc04:~  
[zx_lim@nscc04 ~]$ qstat -f  
Job Id: 2046537.wlm01  
Job_Name = JupyterSingularity  
Job_Owner = zx_lim@nscc04  
resources_used.cput = 98  
resources_used.cput = 00:00:17  
resources_used.mem = 245864kb  
resources_used.ncpus = 1  
resources_used.vmem = 1642480kb  
resources_used.walltime = 00:14:47  
job_state = R  
queue = gpunormal  
server = wlm01  
Checkpoint = u  
ctime = Thu Oct 1 14:07:46 2020  
Error_Path = nscc04:/home/users/industry/sp/zx_lim/JupyterSingularity.e2046537  
exec_host = gpu2035/0  
exec_vnode = (gpu2035:ncpus=1:ngpus=1)  
Hold_Types = n  
Join_Path = n  
Keep_Files = n  
Mail_Points = a  
mtime = Thu Oct 1 14:22:53 2020
```

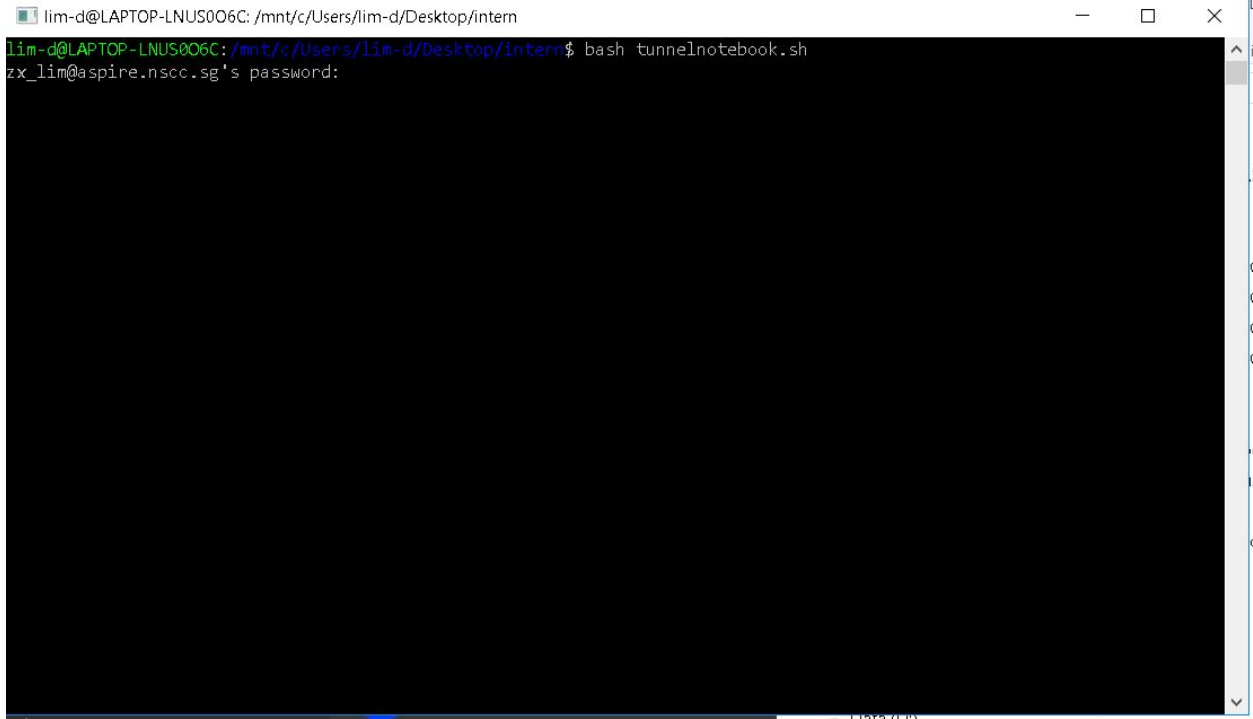
-We will ignore the /0 so our host name will be gpu2035. In this case, my script content will look like this

ssh -N -L localhost:8899:gpu2035-ib0:8899 zx_lim@aspire.nsc.sg

-Save the file in a folder. Now we have to run the script to establish the connection. Go to the folder and type bash.

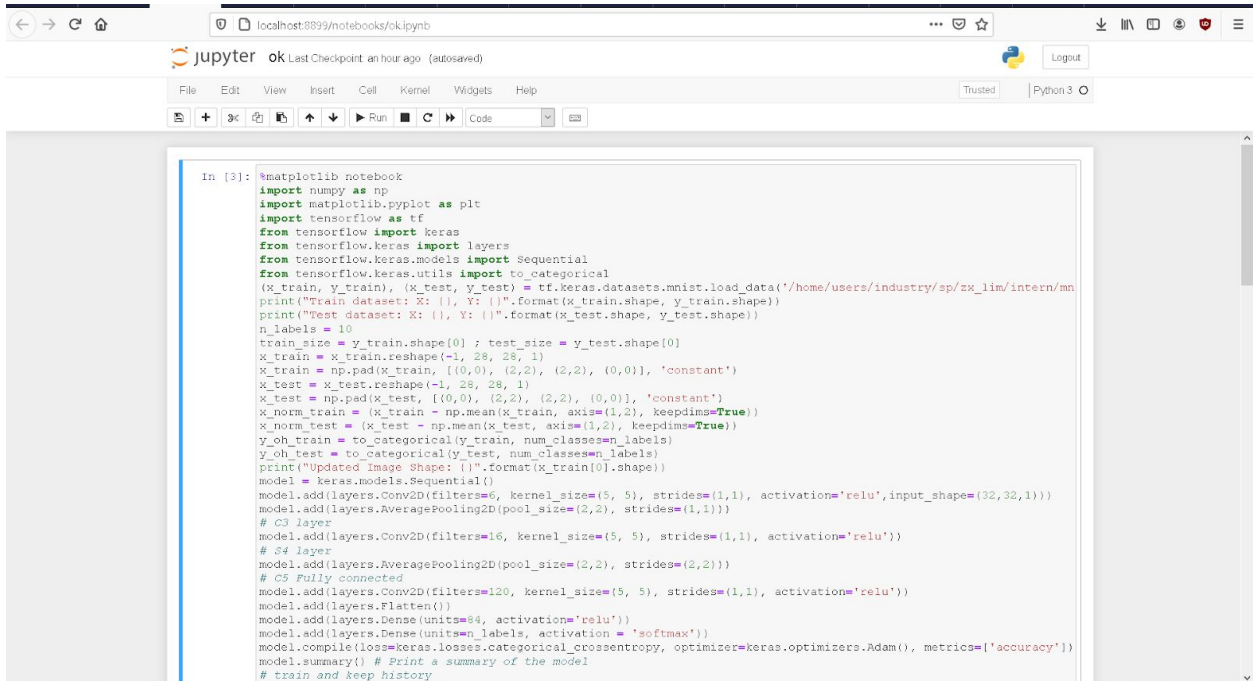


-A new window will popup. Type in 'bash tunnel.sh'. You will be prompted for a password and the password will be your NSCC account password. Now that we have established a connection between NSCC and your local pc, it is time to test out if the connection is successful. Go to your browser and type localhost:8899. You should see jupyter notebook in the browser. You may be prompted to key in a password in order to access the notebook, the password will be the one that you have set on the earlier step in your local jupyter notebook.

A terminal window titled 'lim-d@LAPTOP-LNUS006C: /mnt/c/Users/lim-d/Desktop/Intern' is shown. The terminal has a black background with green text. The first line shows the prompt 'lim-d@LAPTOP-LNUS006C: /mnt/c/Users/lim-d/Desktop/Intern\$' followed by the command 'bash tunnelnotebook.sh'. The second line shows the prompt 'zx_lim@aspire.nsc.sg's password:' followed by a series of asterisks representing a password. The terminal window has standard window controls (minimize, maximize, close) in the top right corner.

```
lim-d@LAPTOP-LNUS006C: /mnt/c/Users/lim-d/Desktop/Intern$ bash tunnelnotebook.sh
zx_lim@aspire.nsc.sg's password:
```

-If all is successful, you should be able to run the code in your jupyter notebook.



The screenshot shows a Jupyter Notebook interface in a web browser. The browser address bar shows 'localhost:8899/notebooks/ok.ipynb'. The Jupyter Notebook header indicates 'ok Last Checkpoint an hour ago (autosaved)' and includes a 'Logout' button. The interface has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The main area contains a code cell with the following Python code:

```
In [3]: %matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data('/home/users/industry/ap/zx_lim/intern/mn
print("Train dataset: X: {}, Y: {}".format(x_train.shape, y_train.shape))
print("Test dataset: X: {}, Y: {}".format(x_test.shape, y_test.shape))
n_labels = 10
train_size = y_train.shape[0] ; test_size = y_test.shape[0]
x_train = x_train.reshape(-1, 28, 28, 1)
x_train = np.pad(x_train, [(0,0), (2,2), (2,2), (0,0)], 'constant')
x_test = x_test.reshape(-1, 28, 28, 1)
x_test = np.pad(x_test, [(0,0), (2,2), (2,2), (0,0)], 'constant')
x_norm_train = (x_train - np.mean(x_train, axis=(1,2), keepdims=True))
x_norm_test = (x_test - np.mean(x_test, axis=(1,2), keepdims=True))
y_oh_train = to_categorical(y_train, num_classes=n_labels)
y_oh_test = to_categorical(y_test, num_classes=n_labels)
print("Updated Image Shape: {}".format(x_train[0].shape))
model = keras.models.Sequential()
model.add(layers.Conv2D(filters=6, kernel_size=(5, 5), strides=(1,1), activation='relu', input_shape=(32,32,1)))
model.add(layers.AveragePooling2D(pool_size=(2,2), strides=(1,1)))
# C3 layer
model.add(layers.Conv2D(filters=16, kernel_size=(5, 5), strides=(1,1), activation='relu'))
# C4 layer
model.add(layers.AveragePooling2D(pool_size=(2,2), strides=(2,2)))
# C5 Fully connected
model.add(layers.Conv2D(filters=120, kernel_size=(5, 5), strides=(1,1), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=n_labels, activation = 'softmax'))
model.compile(loss=keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(), metrics=['accuracy'])
model.summary() # Print a summary of the model
# train and keep history
```