

## Creating a TensorFlow Environment in HPC

Loading the conda module:

```
module load miniconda3
```

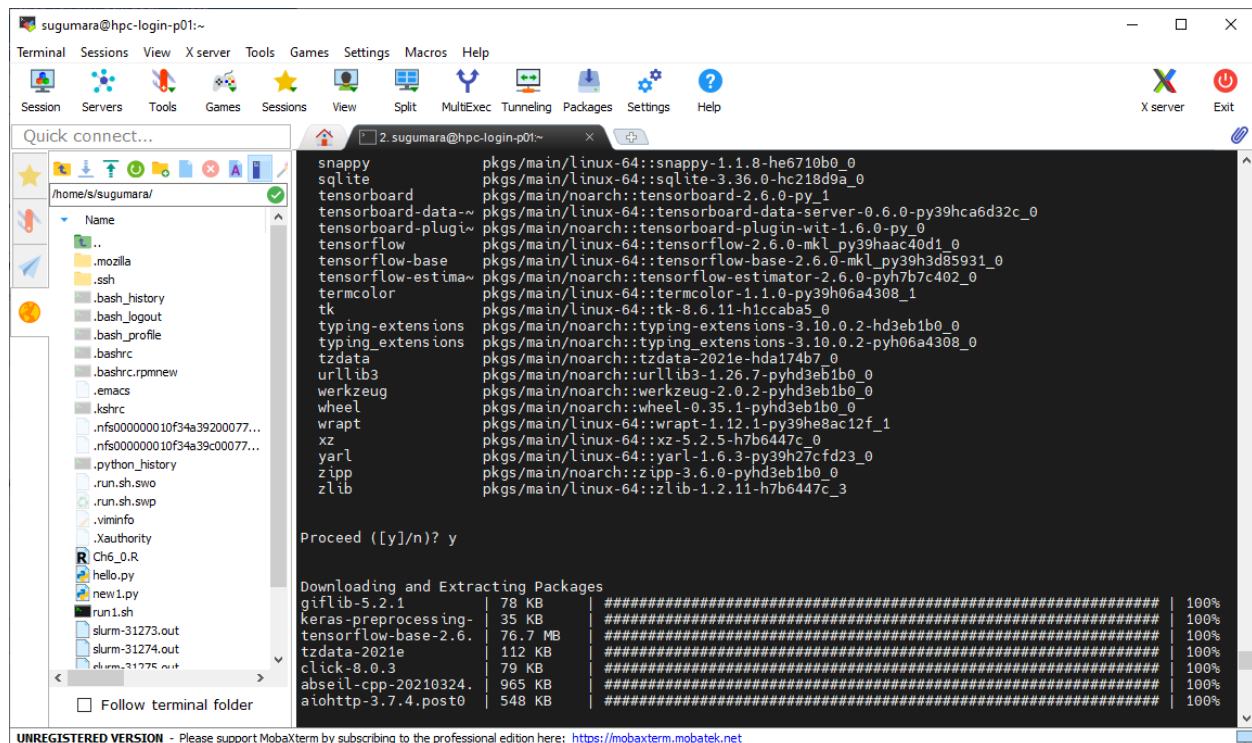
Initialize bash shell to run conda commands:

```
conda init bash  
source ~/.bashrc
```

For creating the tensorflow environment, type the following command:

```
conda create -n tf tensorflow
```

Type “y” once the relevant packages have been identified to install. It takes a bit of time to install all the packages.



Once the installation is complete, for the changes to take effect, close and re-open your current shell.

After starting a new terminal, execute the following command to activate the **tf** environment:

```
conda activate tf
```

## Installing keras

To install the keras package in the **tf** environment, type the following command:

```
conda install keras
```

## Installing sklearn

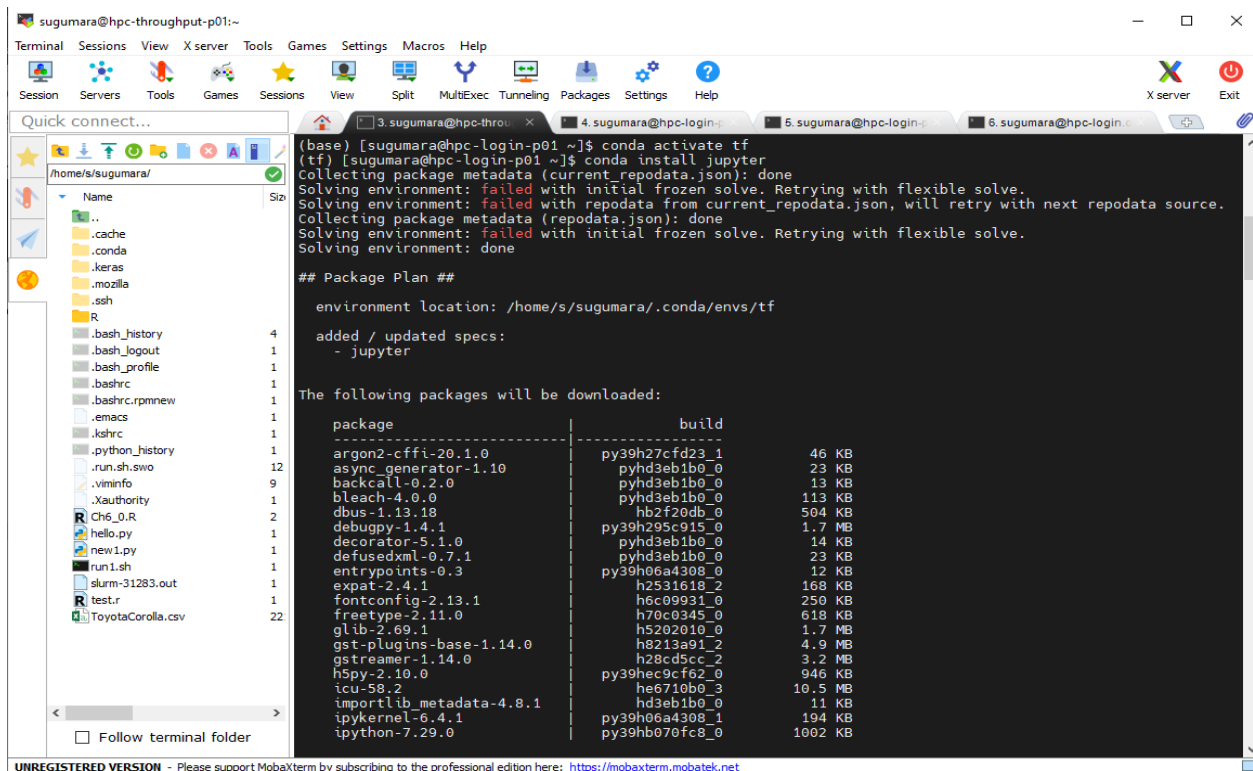
To install the sklearn package in the **tf** environment, type the following command:

```
pip install sklearn
```

## Installing Jupyter Notebook

In the **tf** environment, to install Jupyter Notebook, type the following command:

```
conda install jupyter
```



```
(base) [sugumara@hpc-login-p01 ~]$ conda activate tf
(tf) [sugumara@hpc-login-p01 ~]$ conda install jupyter
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: done

## Package Plan ##

environment location: /home/s/sugumara/.conda/envs/tf

added / updated specs:
- jupyter

The following packages will be downloaded:

package                                     build
-----
argon2-cffi-20.1.0                         py39h27cfd23_1      46 KB
async_generator-1.10                      pyhd3eb1b0_0        23 KB
backcall-0.2.0                            pyhd3eb1b0_0        13 KB
bleach-4.0.0                              pyhd3eb1b0_0       113 KB
dbus-1.13.18                              hb2f20db_0         504 KB
debugpy-1.4.1                             py39h295c915_0      1.7 MB
decorator-5.1.0                           pyhd3eb1b0_0        14 KB
defusedxml-0.7.1                         pyhd3eb1b0_0        23 KB
entrypoints-0.3                           py39h06a4308_0       12 KB
expat-2.4.1                               h2531618_2         168 KB
fontconfig-2.13.1                         h6c09931_0         250 KB
freetype-2.11.0                           h70c0345_0         618 KB
glib-2.69.1                               h5202010_0          1.7 MB
gst-plugins-base-1.14.0                  h3213a91_2          4.9 MB
gstreamer-1.14.0                         h28cd5ec_2          3.2 MB
h5py-2.10.0                               py39hec9cf62_0      946 KB
icu-58.2                                  he6710b0_3         10.5 MB
importlib_metadata-4.8.1                  hd3eb1b0_0          11 KB
ipykernel-6.4.1                           py39h06a4308_1      194 KB
ipython-7.29.0                            py39hb070fc8_0     1002 KB
```

After the relevant packages are gathered, type “y” to proceed with the installation. This will also take some time.

Once the jupyter notebook installation is complete in the **tf** environment, you will get the “done” messages.

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Now you are ready to run jupyter notebook.

## Using Jupyter Notebook on Matilda

There are two basic methods for running Jupyter Notebook on the Matilda HPC compute nodes:  
a) Interactive scheduled job, and b) SBATCH Job.

### Interactive Scheduled Job using SRUN

The first method involves invoking a schedule interactive job to begin the process. Run the following command:

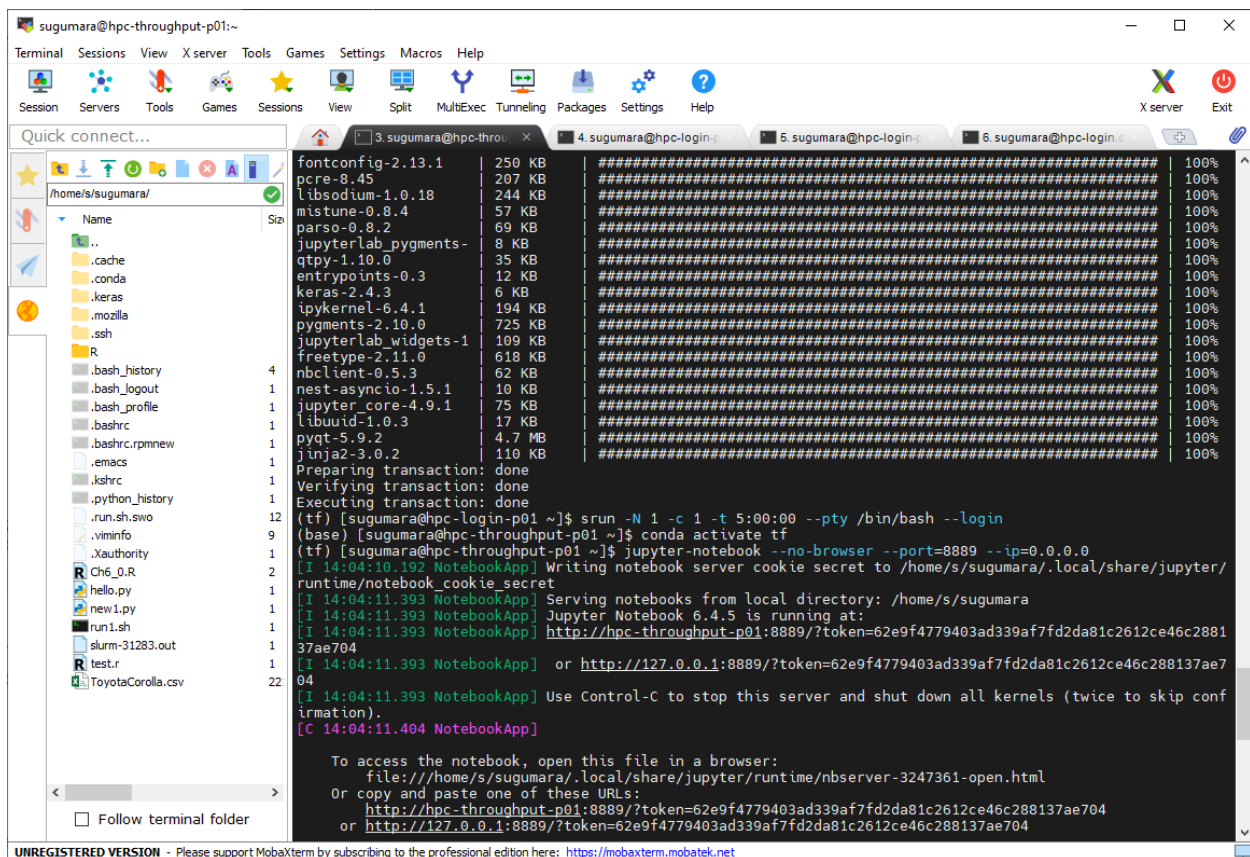
```
srun -N 1 -c 1 -t 5:00:00 --pty /bin/bash --login
```

Then, activate the **tf** environment.  
`conda activate tf`

Start the Jupyter Notebook Server

```
jupyter-notebook --no-browser --port=8889 --ip=0.0.0.0
```

Once the jupyter notebook server starts, you will get a URL that can be used to connect to the server through a browser. You can copy and paste this URL into a browser after the next step.



```
sugumara@hpc-throughput-p01:~$ srun -N 1 -c 1 -t 5:00:00 --pty /bin/bash --login
sugumara@hpc-throughput-p01:~$ conda activate tf
(tf) sugumara@hpc-throughput-p01:~$ jupyter-notebook --no-browser --port=8889 --ip=0.0.0.0
[14:04:10.192 NotebookApp] Writing notebook server cookie secret to /home/s/sugumara/.local/share/jupyter/
runtime/notebook_cookie_secret
[14:04:11.393 NotebookApp] Serving notebooks from local directory: /home/s/sugumara
[14:04:11.393 NotebookApp] Jupyter Notebook 6.4.5 is running at:
[14:04:11.393 NotebookApp] http://hpc-throughput-p01:8889/?token=62e9f4779403ad339af7fd2da81c2612ce46c2881
37ae704
[14:04:11.393 NotebookApp] or http://127.0.0.1:8889/?token=62e9f4779403ad339af7fd2da81c2612ce46c288137ae7
04
[14:04:11.404 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip conf
irmation).

To access the notebook, open this file in a browser:
file:///home/s/sugumara/.local/share/jupyter/runtime/nbserver-3247361-open.html
Or copy and paste one of these URLs:
http://hpc-throughput-p01:8889/?token=62e9f4779403ad339af7fd2da81c2612ce46c288137ae704
or http://127.0.0.1:8889/?token=62e9f4779403ad339af7fd2da81c2612ce46c288137ae704
```

Before we can connect to the jupyter notebook server, we need to initiate a port forwarding session.

From another shell on your workstation, initiate a port forwarding session to the compute node as shown below:

```
ssh -N -L 8889:hpc-throughput-p01:8889 username@hpc-login.oakland.edu
```

Make sure to change “username” to your actual username. Also, make sure to substitute in the actual port used as well as the actual node you are logged into (in this example we are running on hpc-throughput-p01).

Now open a browser and enter the URL that was generated when the Jupyter Notebook Server started:

```
http://127.0.0.1:8889/?token=47ef2216d4ce8e14f30967def52d6e8dd6a0db0514692b00
```

You should now be connected to the Jupyter Notebook. Please again note to substitute the actual port used for "8889" above (also your token will be different than the one shown above). Just copy and paste the appropriate link.

## PRECAUTIONS AND NOTES

There are several issues to keep in mind when running Jupyter Notebooks on Matilda. These are highlighted below.

## PORTS

The default port for Jupyter notebooks is "8888". Since others may be running a notebook on the node on which your job is running, it is possible that port will be in-use and an error will occur. If this happens, increment the port number until you find one that is not being used. This can be controlled either in the interactive job session, or by making a change to the job script for port.

## LOGIN NODE

Please exercise care if running Jupyter Notebooks on the login node. Login is not meant for intensive jobs and notebooks consuming inordinate resources may be killed without warning.

When ever practical, you should be running your notebook on a compute node as a schedule job (interactive or batch).

## Executing a Python Script File

**Example Python file: new.py**

```
import tensorflow as tf
from tensorflow import keras as ke
print('\n TensorFlow version:', tf.__version__)
print(' Keras version:', ke.__version__)
print ('\n Hello World \n')
```

**SLURM File to run the new.py Python File: run.sh**

```
#!/bin/bash
#SBATCH --job-name=mySerialjob
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-00:20:00
#SBATCH --mem=3102
source ~/.bashrc
conda activate tf
python new.py
```

## Submitting the batch file

`sbatch run.sh`

## Directing the output to a file (say python.out)

Modify the last line in the run.sh file as follows:

**SLURM File: run1.sh**

```
#!/bin/bash
#SBATCH --job-name=mySerialjob
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-00:20:00
#SBATCH --mem=3102
source ~/.bashrc
conda activate tf
python new.py > python.out
```

## Running a simple Linear Regression model in Python

**SLURM File: run2.sh**

```
#!/bin/bash
#SBATCH --job-name=mySerialjob
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-00:20:00
#SBATCH --mem=3102
source ~/.bashrc
conda activate tf
python LinearRegression.py > LinearRegression.out
```

## Running a Multiple Linear Regression model in Python

**SLURM File: run3.sh**

```
#!/bin/bash
#SBATCH --job-name=mySerialjob
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-00:20:00
#SBATCH --mem=3102
source ~/.bashrc
conda activate tf
python MLR.py > MLR.out
```

## Running an R script file

**SLURM File: run4.sh**

```
#!/bin/bash
#SBATCH --job-name=mySerialjob
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0-00:20:00
#SBATCH --mem=3102
module load R
Rscript test.r > test.out
```