rtd-tutorial

Emmanuel Preciado

Oct 22, 2021

HPRC

High Performance Research Computing

A Resource for Research and Discovery

TEXAS A&M UNIVERSITY

- **Grace Cluster Status:** Cluster deployed, currently in testing and early user access mode.

CHAPTER

ONE

GETTING AN ACCOUNT

- **Understanding HPRC:** For a brief overview of what services HPRC offers, see this video in our getting started series on YouTube.
- **New to HPRC's resources?** This page explains the HPRC resources available to the TAMU community. Also see the Policies Page to better understand the rules and etiquette of cluster usage..
- **Accessing the clusters:** All computer systems managed by the HPRC are available for use to TAMU faculty, staff, and students who require large-scale computing capabilities. The HPRC hosts the Ada , Terra , and Grace clusters at TAMU. To apply for or renew an HPRC account, please visit the Account Applications page. For information on how to obtain an allocation to run jobs on one of our clusters, please visit the Allocations Policy page. *All accounts expire and must be renewed in September of each year.*

CHAPTER

TWO

USING THE CLUSTERS

- **QuickStart Guides:** For just the "need-to-know" information on getting started with our clusters, visit our QuickStart pages. Topics discussed include cluster access, file management, the batch system, setting up a software environment using modules, creating your own job files, and project account management. Grace Quickstart Guide , Terra Quickstart Guide .
- **Batch Jobs:** As a shared resource between many users, each cluster must employ a batch system to schedule a time for each user's job to run. Without such a system, one user could use a disproportionate amount of resources, and cause other users' work to stall. Ada's batch system is called LSF, and Terra's batch system is called SLURM. While similar in function, they differ in their finer details, such as job file syntax. Information relevant to each system can be found below.

Grace / Slurm Batch Pages	Terra / Slurm Batch Pages
Complete Grace Batch Page	Complete Terra Batch Page
Job Submission (sbatch)	Job Submission (sbatch)
Grace Queue Structure	Terra Queue Structure

Creating your own batch jobs: the tamubatch Page provides information on how to use tamubatch to create and submit jobs easily.

- **Troubleshooting:** While we cannot predict all bugs and errors, some issues on our clusters are common enough to catalog. See the Common Problems and Quick Solutions Page for a small collection of the most prevalent issues. For further assistance, users can contact **help@hprc.tamu.edu** to open a support ticket.

CHAPTER THREE

HPRC'S YOUTUBE CHANNEL

- **Prefer visual learning?** HPRC has launched its official YouTube channel where you can find video versions of our help guides, recordings of our short courses, and more! Subscribe here.

CHAPTER FOUR

FURTHER READING

- Ada User Guide
- Terra User Guide
- Grace User Guide
- Workstations
- Hardware Overview
- Ada Hardware
- Terra Hardware
- Grace Hardware
- TAMU OnDemand Portal
- Software Overview
- Loading Software
- Check Software License Availability
- Software Policies
- Usage Policies
- Account Application
- Manage SUs (Transfers)
- Contact Us
- Youtube Channel

4.1 HPRC

- HPRC Home Page
- HPRC Policies Page
- HPRC New User Info
- HPRC Contact Us

4.2 Grace

4.2.1 Deployment Status

Cluster deployed, currently in testing and early user access mode.

4.2.2 General Usage Policies

Access to Grace is granted with the condition that you will understand and adhere to all TAMU HPRC and Grace-specific policies.

General policies can be found on the HPRC Policies Page .

4.2.3 Accessing Grace

Most access to Grace is done via a secure shell session. In addition, **two-factor authentication** is required to login to any cluster.

Users on **Windows** computers use either Putty or MobaXterm . If MobaXterm works on your computer, it is usually easier to use. When starting an ssh session in PuTTY, choose the connection type 'SSH', select port 22, and then type the hostname 'grace.hprc.tamu.edu'. For MobaXterm, select 'Session', 'SSH', and then remote host 'grace.hprc.tamu.edu'. Check the box to specify username and type your NetID. After selecting 'Ok', you will be prompted for Duo Two Factor Authentication. For more detailed instructions, visit the Two Factor Authentication page.

Users on **Mac** and **Linux/Unix** should use whatever SSH-capable terminal is available on their system. The command to connect to Grace is as follows. Be sure to replace [NetID] with your TAMU NetID.

```
[user1@localhost ~]$ ssh [NetID]@grace.hprc.tamu.edu
```

Note: In this example [user1@localhost ~]$ represents the command prompt on your local machine.

Your login password is the same that used on Howdy . You will not see your password as your type it into the login prompt.

Off Campus Access

Please visit this page to find information on accessing Grace remotely.

For more detailed instructions on how to access our systems, please see the HPRC Access page .

4.2.4 Navigating Grace and Storage Quotas

When you first access Grace, you will be within your *home* directory. This directory has smaller storage quotas and should not be used for general purpose.

You can navigate to your *home* directory with the following command:

```
[NetID@grace1 ~]$ cd /home/NetID
```

Your *scratch* directory has more storage space than your *home* directory and is recommended for general purpose use. You can navigate to your *scratch* directory with the following command:

```
[NetID@grace1 ~]$ cd /scratch/user/NetID
```

You can navigate to *scratch* or *home* easily by using their respective environment variables.

Navigate to *scratch* with the following command:

```
[NetID@grace1 ~]$ cd $SCRATCH
```

Navigate to *home* with the following command:

```
[NetID@grace1 ~]$ cd $HOME
```

Note: Your *scratch* directory is restricted to 1TB/250,000 files of storage. This storage quota is **expandable** upon request. A user's *scratch* directory is **NOT** backed up.

Your *home* directory is restricted to 10GB/10,000 files of storage. This storage quota is **not expandable**. A user's *home* directory is backed up on a nightly basis.

You can see the current status of your storage quotas with:

```
[NetID@grace1 ~]$ showquota
```

If you need a storage quota increase, please contact us with justification and the expected length of time that you will need the quota increase.

4.2.5 Transferring Files

Files can be transferred to Grace using the scp command or a file transfer program.

Our users most commonly utilize:

- WinSCP - Straightforward, legacy
- FileZilla Client - Easy to use, additional features, available on most platforms
- MobaXterm Graphical SFTP - Included with MobaXterm

Tip: While GUIs are acceptable for file transfers, the cp and scp commands are much quicker and may significantly benefit your workflow.

Reliably Transferring Large Files

For files larger than several GB, you will want to consider the use of a more fault-tolerant utility such as rsync.

```
[NetID@grace1 ~]$ rsync -av [-z] localdir/ userid@remotesystem:/path/to/remotedir/
```

An rsync example can be seen on the Ada Fast Transfer page.

4.2.6 Managing Project Accounts

The batch system will charge SUs from the either the account specified in the job parameters, or from your default account (if this parameter is omitted). To avoid errors in SU billing, you can view your active accounts, and set your default account using the myproject command.

4.2.7 Finding Software

Software on Grace is loaded using **hierarchical modules**.

A list of the most popular software on our systems is available on the HPRC Available Software page.

To list all software installed as a module on Grace, use the mla utility:

```
[NetID@grace1 ~]$ mla
```

To search for a specific piece of software installed as a module on Grace using the mla utility:

```
[NetID@grace1 ~]$ mla keyword
```

To **search for** particular software by keyword, use:

```
[NetID@grace1 ~]$ module spider keyword
```

To see how to load a module, use the full module name:

```
[NetID@grace1 ~]$ module spider Perl/5.32.0
```

You will see a message like the following:

```
You will need to load all module(s) on any one of the lines below before the "Perl/5.32.0
↪" module is available to load.

    GCCcore/10.2.0
```

Load the base dependency module(s) first then the full module name

```
[NetID@grace1 ~]$ module load GCCcore/10.2.0  Perl/5.32.0
```

To list all currently loaded modules, use:

```
[NetID@grace1 ~]$ module list
```

To see what other modules can be loaded with the base dependency module (for example when GCCcore/10.2.0 is loaded)

```
[NetID@grace1 ~]$ module avail
```

To remove all currently loaded modules, use:

```
[NetID@grace1 ~]$ module purge
```

If you need **new software** or **an update**, please contact us with your request.

There are restrictions on what software we can install. There is also regularly a queue of requested software installations. Please account for **delays **in your installation request timeline.

4.2.8 Running Your Program / Preparing a Job File

In order to properly run a program on Grace, you will need to create a job file and submit a job to the batch system. The batch system is a load distribution implementation that ensures convenient and fair use of a shared resource. Submitting jobs to a batch system allows a user to reserve specific resources with minimal interference to other users. All users are required to submit resource-intensive processing to the compute nodes through the batch system - **attempting to circumvent the batch system is not allowed.**

On Grace, **Slurm** is the batch system that provides job management. More information on **Slurm** can be found in the Grace Batch page.

The simple example job file below requests 1 core on 1 node with 2.5GB of RAM for 1.5 hours. **Note that typical nodes on Grace have 48 cores with 384 GB of usable memory and ensure that your job requirements will fit within these restrictions.** Any modules that need to be loaded or executable commands will replace the *"#First Executable Line"* in this example.

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE        #Do not propagate environment
#SBATCH --get-user-env=L     #Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1     #Set the job name to "JobExample1"
#SBATCH --time=01:30:00            #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1                 #Request 1 task
#SBATCH --ntasks-per-node=1        #Request 1 task/core per node
#SBATCH --mem=2560M                #Request 2560MB (2.5GB) per node
#SBATCH --output=Example1Out.%j    #Send stdout/err to "Example1Out.[jobID]"

#First Executable Line
```

Note: If your job file has been written on an older Mac or DOS workstation, you will need to use "dos2unix" to remove certain characters that interfere with parsing the script.

```
[NetID@grace1 ~]$ dos2unix MyJob.slurm
```

More information on **job options** can be found in the Building Job Files section of the Grace Batch page.

More information on **dos2unix** can be found on the dos2unix section of the HPRC Available Software page.

4.2.9 Submitting and Monitoring Jobs

Once you have your job file ready, it is time to submit your job. You can submit your job to slurm with the following command:

After the job has been submitted, you are able to monitor it with several methods. To see the status of all of your jobs, use the following command:

To see the status of one job, use the following command, where XXXX is the JobID:

To cancel a job, use the following command, where XXXX is the JobID:

```
[NetID@grace1 ~]$ scancel XXXX
```

More information on Job Submission and Job Monitoring Slurm jobs can be found at the Grace Batch System page.

4.2.10 tamubatch

tamubatch is an automatic batch job script that submits jobs for the user without the need of writing a batch script on the clusters. The user just needs to provide the executable commands in a text file and tamubatch will automatically submit the job to the cluster. There are flags that the user may specify which allows control over the parameters for the job submitted.

tamubatch is still in beta and has not been fully developed. Although there are still bugs and testing issues that are currently being worked on, tamubatch can already submit jobs to both the clusters if given a file of executable commands.

For more information, visit this page.

4.2.11 Graphic User Interfaces (Visualization)

The use of GUIs on Grace is a more complicated process than running non-interactive jobs or doing resource-light interactive processing.

You have **two options** for using GUIs on Grace.

The **first option** is to use the Open On Demand Portal ,which is a web interface to our clusters. Users must be connected to the campus network either directly or via VPN to access the portal. More information can be found here , or on our YouTube channel

The **second option** is to run on the login node. When doing this, you **must** observe the fair-use policy of login node usage. Users commonly violate these policies by accident, resulting in terminated processes, confusion, and warnings from our admins.

4.3 Terra

4.3.1 Terra Usage Policies

Access to Terra is granted with the condition that you will understand and adhere to all TAMU HPRC and Terra-specific policies.

General policies can be found on the HPRC Policies Page .

Terra-specific policies, which are similar to Ada, can be found on the Terra Policies Page

4.3.2 Accessing Terra

Most access to Terra is done via a secure shell session. In addition, **two-factor authentication** is required to login to any cluster.

Users on **Windows** computers use either Putty or MobaXterm . If MobaXterm works on your computer, it is usually easier to use. When starting an ssh session in PuTTY, choose the connection type 'SSH', select port 22, and then type the hostname 'terra.tamu.edu'. For MobaXterm, select 'Session', 'SSH', and then remote host 'terra.tamu.edu'. Check the box to specify username and type your NetID. After selecting 'Ok', you will be prompted for Duo Two Factor Authentication. For more detailed instructions, visit the Two Factor Authentication page.

Users on **Mac** and **Linux/Unix** should use whatever SSH-capable terminal is available on their system. The command to connect to Terra is as follows. Be sure to replace [NetID] with your TAMU NetID.

```
[user1@localhost ~]$ ssh [NetID]@terra.tamu.edu
```

Note: In this example [user1@localhost ~]$ represents the command prompt on your local machine.

Your login password is the same that used on Howdy . You will not see your password as your type it into the login prompt.

Off Campus Access

Please visit this page to find information on accessing Terra remotely.

For more detailed instructions on how to access our systems, please see the HPRC Access page .

4.3.3 Navigating Terra and Storage Quotas

When you first access Terra, you will be within your *home* directory. This directory has smaller storage quotas and should not be used for general purpose.

You can navigate to your *home* directory with the following command:

```
[NetID@terra1 ~]$ cd /home/NetID
```

Your *scratch* directory has more storage space than your *home* directory and is recommended for general purpose use. You can navigate to your *scratch* directory with the following command:

```
[NetID@terra1 ~]$ cd /scratch/user/NetID
```

You can navigate to *scratch* or *home* easily by using their respective environment variables.

Navigate to *scratch* with the following command:

```
[NetID@terra1 ~]$ cd $SCRATCH
```

Navigate to *home* with the following command:

```
[NetID@terra1 ~]$ cd $HOME
```

Note: Your *scratch* directory is restricted to 1TB/250,000 files of storage. This storage quota is **expandable** upon request. A user's *scratch* directory is **NOT** backed up.

Your *home* directory is restricted to 10GB/10,000 files of storage. This storage quota is **not expandable**. A user's *home* directory is backed up on a nightly basis.

You can see the current status of your storage quotas with:

```
[NetID@terra1 ~]$ showquota
```

If you need a storage quota increase, please contact us with justification and the expected length of time that you will need the quota increase.

4.3.4 Transferring Files

Files can be transferred to Terra using the scp command or a file transfer program.

Our users most commonly utilize:

- WinSCP - Straightforward, legacy
- FileZilla Client - Easy to use, additional features, available on most platforms
- MobaXterm Graphical SFTP - Included with MobaXterm

Tip: While GUIs are acceptable for file transfers, the cp and scp commands are much quicker and may significantly benefit your workflow.

Reliably Transferring Large Files

For files larger than several GB, you will want to consider the use of a more fault-tolerant utility such as rsync.

```
[NetID@terra1 ~]$ rsync -av [-z] localdir/ userid@remotesystem:/path/to/remotedir/
```

An rsync example can be seen on the Ada Fast Transfer page.

4.3.5 Managing Project Accounts

The batch system will charge SUs from the either the account specified in the job parameters, or from your default account (if this parameter is omitted). To avoid errors in SU billing, you can view your active accounts, and set your default account using the myproject command.

4.3.6 Finding Software

Software on Terra is loaded using **modules**.

A list of the most popular software on our systems is available on the HPRC Available Software page.

To list all software installed as a module on Terra, use the following command:

```
[NetID@terra1 ~]$ module avail
```

To **search for** particular software by keyword, use:

```
[NetID@terra1 ~]$ module spider keyword
```

To load a module, use:

```
[NetID@terra1 ~]$ module load moduleName
```

To list all currently loaded modules, use:

```
[NetID@terra1 ~]$ module list
```

To remove all currently loaded modules, use:

```
[NetID@terra1 ~]$ module purge
```

If you need **new software** or **an update**, please contact us with your request.

There are restrictions on what software we can install. There is also regularly a queue of requested software installations. Please account for **delays **in your installation request timeline.

4.3.7 Running Your Program / Preparing a Job File

In order to properly run a program on Terra, you will need to create a job file and submit a job to the batch system. The batch system is a load distribution implementation that ensures convenient and fair use of a shared resource. Submitting jobs to a batch system allows a user to reserve specific resources with minimal interference to other users. All users are required to submit resource-intensive processing to the compute nodes through the batch system - **attempting to circumvent the batch system is not allowed.**

On Tera, **Slurm** is the batch system that provides job management. More information on **Slurm** can be found in the Terra Batch page.

The simple example job file below requests 1 core on 1 node with 2.5GB of RAM for 1.5 hours. Note that typical nodes on Terra have 28 cores with 120GB of usable memory and ensure that your job requirements will fit within these restrictions. Any modules that need to be loaded or executable commands will replace the *"#First Executable Line"* in this example.

```
#!/bin/bash
##ENVIRONMENT SETTINGS; CHANGE WITH CAUTION
#SBATCH --export=NONE        #Do not propagate environment
#SBATCH --get-user-env=L     #Replicate login environment

##NECESSARY JOB SPECIFICATIONS
#SBATCH --job-name=JobExample1     #Set the job name to "JobExample1"
#SBATCH --time=01:30:00            #Set the wall clock limit to 1hr and 30min
#SBATCH --ntasks=1                 #Request 1 task
#SBATCH --ntasks-per-node=1        #Request 1 task/core per node
#SBATCH --mem=2560M                #Request 2560MB (2.5GB) per node
#SBATCH --output=Example1Out.%j    #Send stdout/err to "Example1Out.[jobID]"

#First Executable Line
```

Note: If your job file has been written on an older Mac or DOS workstation, you will need to use "dos2unix" to remove certain characters that interfere with parsing the script.

```
[NetID@terra1 ~]$ dos2unix MyJob.slurm
```

More information on **job options** can be found in the Building Job Files section of the Terra Batch page.

More information on **dos2unix** can be found on the dos2unix section of the HPRC Available Software page.

4.3.8 Submitting and Monitoring Jobs

Once you have your job file ready, it is time to submit your job. You can submit your job to slurm with the following command:

After the job has been submitted, you are able to monitor it with several methods. To see the status of all of your jobs, use the following command:

To see the status of one job, use the following command, where XXXX is the JobID:

To cancel a job, use the following command, where XXXX is the JobID:

```
[NetID@terra1 ~]$ scancel XXXX
```

More information on Job Submission and Job Monitoring Slurm jobs can be found at the Terra Batch System page.

4.3.9 tamubatch

tamubatch is an automatic batch job script that submits jobs for the user without the need of writing a batch script on the clusters. The user just needs to provide the executable commands in a text file and tamubatch will automatically submit the job to the cluster. There are flags that the user may specify which allows control over the parameters for the job submitted.

tamubatch is still in beta and has not been fully developed. Although there are still bugs and testing issues that are currently being worked on, tamubatch can already submit jobs to both the clusters if given a file of executable commands.

For more information, visit this page.

4.3.10 Graphic User Interfaces (Visualization)

The use of GUIs on Terra is a more complicated process than running non-interactive jobs or doing resource-light interactive processing.

You have **two options** for using GUIs on Terra.

The **first option** is to use the Open On Demand Portal ,which is a web interface to our clusters. Users must be connected to the campus network either directly or via VPN to access the portal. More information can be found here , or on our YouTube channel

The **second option** is to run on the login node. When doing this, you **must** observe the fair-use policy of login node usage. Users commonly violate these policies by accident, resulting in terminated processes, confusion, and warnings from our admins.

4.4 OOD Portal

4.4.1 What is TAMU HPRC OnDemand Portal

The TAMU HPRC OnDemand portal is based on Open OnDemand , an open source web platform through which users can access HPC clusters and services with a web browser. The portal provides an intuitive and easy-to-use interface and allows new users to be instantly productive at using the HPC resources for their research, and at the same time, provides an alternative convenient way for experienced users to access the HPC resources. The portal has a flexible and extensible design that makes it easy to deploy new services as needed.

4.4.2 Services Provided

- Job submission and monitoring
- File transfer and management
- File editing
- Shell access
- Interactive applications
- Abaqus
- Ansys
- IGV
- LS-PREPOST
- Matlab
- Jupyter Notebook
- Paraview
- VNC
- Rstudio
- JupyterLab
- JBrowse

4.4.3 How to Access

We recommend you access the Grace or Terra portal through their landing page at

> https://portal.hprc.tamu.edu

Click the portal you want to connect. The portals are CAS authenticated. All active HPRC users have access to both portals using their NetID and password. You will only be authenticated once, and before your session expires, you can freely access both portals without further authentication.

If accessing from off-campus, the TAMU VPN is needed.

You can go directly to the Grace or Terra portal using one of the following URLs:

> https://portal-terra.hprc.tamu.edu
>
> https://portal-grace.hprc.tamu.edu

Two-Factor Authentication Requirement

Starting October 1, 2018, the Division of Information Technology will require use of Duo NetID Two Factor Authentication on its Virtual Private Network (VPN) (connect.tamu.edu) service.

Duo provides a second layer of security to Texas A&M accounts.

If you are not already enrolled in Duo and plan to use VPN, you can enroll now at duo.tamu.edu. Enrolling is as easy as 1-2-3:

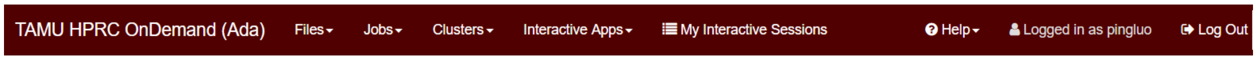
1. Choose your device and download the Duo Mobile app. (We strongly recommend the mobile app as the most user-friendly option.)

2. Start your enrollment at https://gateway.tamu.edu/duo-enroll/;
3. Remember: Once you sign up, you will need your Duo-enrolled device when you log in to most Texas A&M resources.

For more information, consult IT's knowledge base article for Duo: https://u.tamu.edu/KB0012105

4.4.4 Using the Portal

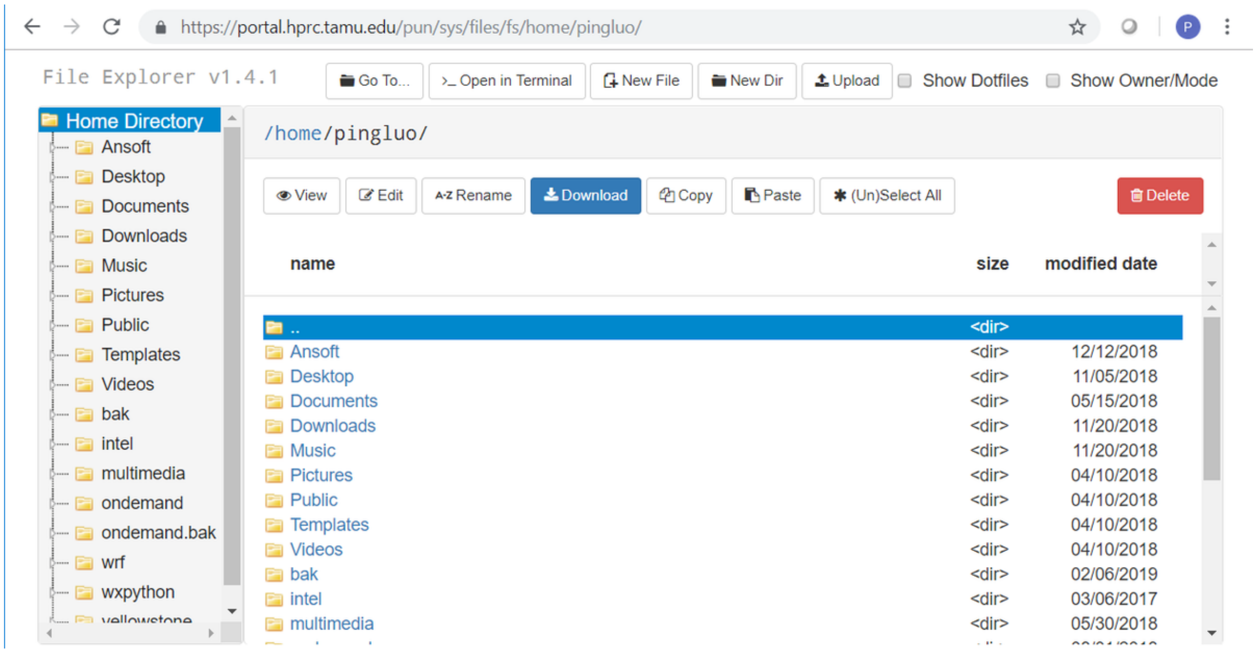
Each service provided by the portal is available at the navigation bar at the top of the page.



Files

The first option in the navigation bar is the "Files" drop down menu. From this menu, a user can view a file explorer at either their home directory or scratch directory.

Some users may find the visual interface of the file explorer more intuitive than shell based file exploring. All files in the directory are shown on screen, along with the file tree or hierarchy.



name	size	modified date
..	<dir>	
Ansoft	<dir>	12/12/2018
Desktop	<dir>	11/05/2018
Documents	<dir>	05/15/2018
Downloads	<dir>	11/20/2018
Music	<dir>	11/20/2018
Pictures	<dir>	04/10/2018
Public	<dir>	04/10/2018
Templates	<dir>	04/10/2018
Videos	<dir>	04/10/2018
bak	<dir>	02/06/2019
intel	<dir>	03/06/2017
multimedia	<dir>	05/30/2018

Normal file management commands are available with the click of a button. These include:

- Viewing files
- Text editing
- Copy/Paste
- Renaming files
- Creating files
- Creating directories
- Deleting files

- File upload/download

The ‘View’ button will display the highlighted file in the browser, as long as the file type is supported by the browser. Luckily modern browsers support many different types of files, from simple text files, to image files, to complicated multimedia files. This feature can be very convenient and useful if you want to quickly review a file, since you don’t have to download the file to your local machine first and then review it, as what you would be doing if you had connected to a cluster using putty or mobaxterm.

File Editor

File editor allows you to edit a file selected. It cannot be accessed from the main menu, but is available through the File app or Job Composer. In File app, you first select a file, then click ‘Edit’ from the File app interface. Then a new tab will be opened and you can edit the file in the editor. In Job Composer, you can edit the job script by clicking ‘Open Editor’ at the bottom of Job Composer.

Cluster Shell Access

Shell access to any of the three clusters is available from this drop down menu with one click. Shell access app is similar to ssh client such as Putty and MobaXterm. it allows users to login to a cluster with their NetID and password.

Copy/Paste can be done with hot keys. To copy text from the shell access terminal, highlight the text with a mouse, then the highlighted text will be coped into the clipboard. To paste a text from the clipboard to the terminal, type ‘Ctrl+v’.

Shell access works with Firefox and Chrome only.

Copy/Paste in VNC

If launching an interactive session in the portal, there are a few extra steps that need to be taken. Please reference the media below, or the summary of steps below that for more information.

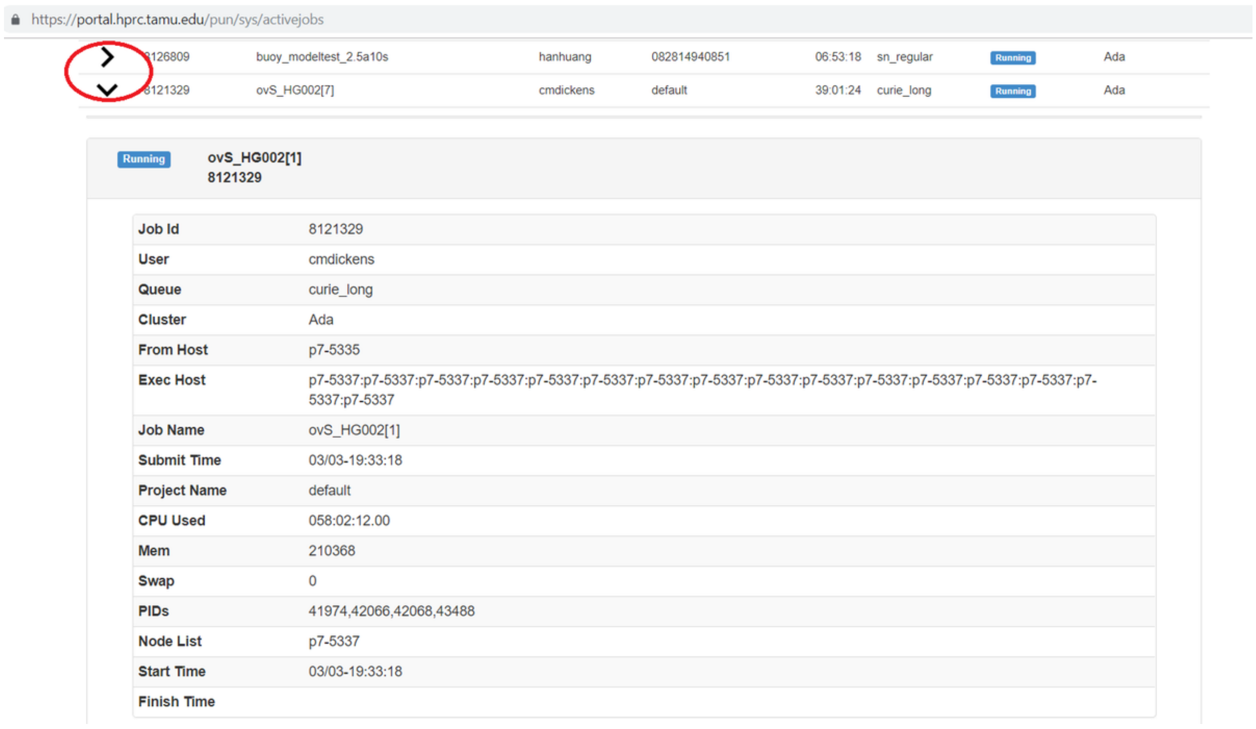
1. Open the toolbar on the left of the screen and select “Clipboard”.
2. If you want to paste text from your host computer to the remote session, paste the text in the clipboard box. You can then use the middle-mouse button (MMB) to paste it in your terminal.
3. If you want to copy text from the remote session to your host computer’s clipboard, simply highlight the text in the terminal. It will appear in the Clipboard toolbar pop-out where you can copy it to your host clipboard.

Jobs

From the jobs drop down menu, a user can view their active jobs or compose and submit jobs using the job composer.

Active Jobs

The active jobs menu provides information about running jobs the cluster, including their JobID, name, user, account, time used, queue, and status. Clicking the arrow to the left of a given job will reveal more details, such as where it was submitted from, which node it’s running on, when it was submitted, process IDs, memory, and CPU time.


https://portal.hprc.tamu.edu/pun/sys/activejobs

8126809	buoy_modeltest_2.5a10s	hanhuang	082814940851	06:53:18	sn_regular	Running	Ada
8121329	ovS_HG002[7]	cmdickens	default	39:01:24	curie_long	Running	Ada

Running ovS_HG002[1]
8121329

Job Id	8121329
User	cmdickens
Queue	curie_long
Cluster	Ada
From Host	p7-5335
Exec Host	p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337:p7-5337
Job Name	ovS_HG002[1]
Submit Time	03/03-19:33:18
Project Name	default
CPU Used	058:02:12.00
Mem	210368
Swap	0
PIDs	41974,42066,42068,43488
Node List	p7-5337
Start Time	03/03-19:33:18
Finish Time	

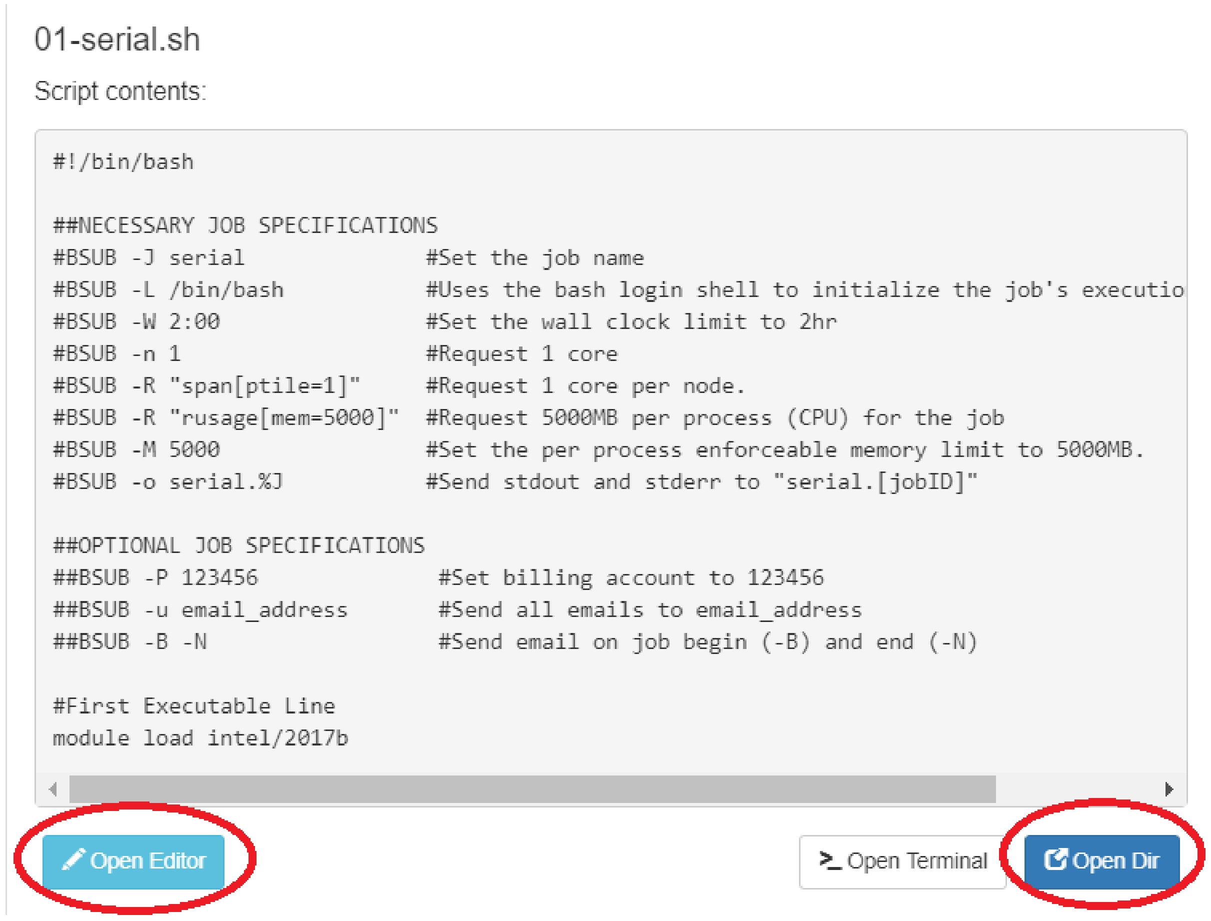
Job Composer

When first launched, the job composer will walk the user through each of its features, covering the whole process of creating, editing, and submitting a job.

The job composer provides some template job scripts the user can choose from. Once a template is selected, you need to edit the template to provide customized job content. This can be done by clicking 'Open Editor' underneath the job script contents.

The job composer has a specific directory in the user's scratch to store the jobs it has created. We call the directory the job composer's root directory. New jobs created by the job composer will have a sub-directory in the root directory. The name of the sub-directory is same as the index of the job, which is an integer maintained by the job composer. The first job has an index 1, the second job has an index 2, and so on. Knowing this is very important to help us using the job composer more effectively.

There are two ways to cope with the default directory created by the job composer.



01-serial.sh

Script contents:

```
#!/bin/bash

##NECESSARY JOB SPECIFICATIONS
#BSUB -J serial              #Set the job name
#BSUB -L /bin/bash           #Uses the bash login shell to initialize the job's executio
#BSUB -W 2:00                #Set the wall clock limit to 2hr
#BSUB -n 1                   #Request 1 core
#BSUB -R "span[ptile=1]"     #Request 1 core per node.
#BSUB -R "rusage[mem=5000]"  #Request 5000MB per process (CPU) for the job
#BSUB -M 5000                #Set the per process enforceable memory limit to 5000MB.
#BSUB -o serial.%J           #Send stdout and stderr to "serial.[jobID]"

##OPTIONAL JOB SPECIFICATIONS
##BSUB -P 123456              #Set billing account to 123456
##BSUB -u email_address       #Send all emails to email_address
##BSUB -B -N                  #Send email on job begin (-B) and end (-N)

#First Executable Line
module load intel/2017b
```

Method 1: using the default directory as the working directory of your job. This means you need to upload all input files to that directory before you can click the submit button. This can be easily done by clicking 'Open Dir' right beneath the job script contents. A file explorer will open the job directory in a new tab where you can do file transfers.

Method 2: if you already have the input files stored somewhere in the cluster and don't want to move them around, or you prefer to have an organized directories by yourself, you can simply add one command line in the job script before any other command line, where /path/to/job_working_dir is the directory you want all the commands to be executed:

```
cd /path/to/job_working_dir
```

Common Problems

1. The session starts and quits immediately.

   Check your quota in your home and scratch. If you see a full or close to full usage, clean your disk space and try again.

2. In ANSYS Workbench, not all windows are available in the foreground.

   Right click the bottom panel title bar "Unsaved Project - Workbench" and select maximize

Log out

To properly log out the portal, you must do two things: (1) log out the portal by clicking 'Log out' from the top navigation bar; (2) close the browser to completely terminate the session.

Be aware that only logout of the portal is not enough. You must also close the entire browser (not just the tab), a side effect of CAS. This is very important if you are using a public computer.

Cleanup

The portal stores temporary files for interactive apps in $SCRATCH/ondemand/data/sys/dashboard/. Although the disk space used by those files accumulate slowly, it is a good habit to clean this directory periodically.

```
rm -rf $SCRATCH/ondemand/data/sys/dashboard/batch_connect/sys/*
```

4.4.5 Interactive Apps

Each piece of software listed above in the "services provided" section is directly available to launch from this menu. When a piece of software is selected, you will see the interface for job parameters such as number of cores, wall time, memory, and type of node. If you are not sure what to change, the default values work fine. Once you fill out the form, click 'Launch' and the app will be launched as a job. It will first go into a queue, and when ready, a button will become available to view the interface of the chosen software.

Interactive sessions can be managed via the "My Interactive Sessions" button on the navigation bar.

We have tried to provide the most commonly used GUI software packages on the Interactive Apps drop-down menu. If a software is not available, you can always run it within VNC, which is provided on the drop-down menu. To run a GUI application in the VNC session on the portal, follow these steps.

1. Click 'VNC' from 'Interactive Apps' and start a vnc session.
2. In the terminal within the new tab, load the module for the software you want to run.
3. If you have chosen a GPU node, please run

```
vglrun app_name
```

Otherwise, type the app_name from the command line directly.

RStudio

To install CRAN packages, start RStudio with enough memory for the install process: 10 cores and 2GB per core for example.

RStudio automatically enables a web proxy so you only need to run the install.packages command or other commands for Github and Bioconductor package installations.

If you are having problems with Bioconductor or github R packages installations, contact the HPRC helpdesk to request installation.

JupyterLab

Anaconda

Default conda environment

You can use the default conda environment in the JupyterLab portal app by selecting Anaconda/3-5.0.0.1 or Anaconda3/2020.07 and leaving the 'Optional Conda Environment to be activated' field blank.

The default environment for Anaconda/3-5.0.0.1 is jupyterlab-v1.2.3_R-3.6.1 which has the R console installed.

The default environment for Anaconda3/2020.07 is /sw/hprc/sw/Anaconda3/2020.07/envs/jupyterlab_v2.2.9_R-3.6.1 which has the R console installed.

Custom Anaconda/3-5.0.0.1 conda environment

You can create your own JupyterLab conda environment using Anaconda for use on the HPRC portal but you must use one of the Anaconda versions that are on the JupyterLab HPRC portal webpage .

Notice that you will need to make sure you have enough available file quota (~30,000) since conda creates thousands of files.

To to create an Anaconda conda environment called jupyterlab, do the following on the command line:

```
module purge
module load Anaconda/3-5.0.0.1
conda create -n jupyterlab
```

After your jupyterlab environment is created, you will see output on how to activate and use your jupyterlab environment

```
#
# To activate this environment, use:
# > source activate jupyterlab
#
# To deactivate an active environment, use:
# > source deactivate
#
```

Then you can install jupyterlab (specifying a version if needed) and add packages to your jupyterlab environment

```
source activate jupyterlab
conda install -c conda-forge jupyterlab
conda install -c conda-forge package-name
```

You can specify a specific package version with the install command. For example to install pandas version 1.1.3:

```
conda install -c conda-forge pandas=1.1.3
```

To remove downloads after packages are installed.

```
conda clean -t
```

When using Anaconda/3-5.0.0.1, use just the environment name in the 'Optional Environment to be activated' field which in this example will be **jupyterlab**

Custom Anaconda3/2020.07 conda environment

You can create your own JupyterLab conda environment using Anaconda for use on the HPRC portal but you must use one of the Anaconda versions that are on the JupyterLab HPRC portal webpage .

Notice that you will need to make sure you have enough available file quota (~30,000) since conda creates thousands of files.

When using Anaconda3/2020.07, you will need to move your ~/.conda directory to $SCRATCH and make a symbolic link since Anaconda3 may fill up your $HOME disk quota:

```
cd
mv .conda $SCRATCH
ln -s $SCRATCH/.conda
```

To to create an Anaconda conda environment called jupyterlab, do the following on the command line:

```
module purge
mkdir -p /scratch/user/your_netid/Anaconda3/2020.07/envs
module load Anaconda3/2020.07
conda create --prefix /scratch/user/your_netid/Anaconda3/2020.07/envs/jupyterlab
```

After your jupyterlab environment is created, you will see output on how to activate and use your jupyterlab environment. You can use 'source activate' instead of 'conda activate'

```
#
# To activate this environment, use:
# > conda activate /scratch/user/your_netid/Anaconda3/2020.07/envs/jupyterlab
#
# To deactivate an active environment, use:
# > conda deactivate
#
```

Then you can install jupyterlab (specifying a version if needed) and add packages to your jupyterlab environment

```
source activate /scratch/user/your_netid/Anaconda3/2020.07/envs/jupyterlab
conda install -c conda-forge jupyterlab
conda install -c conda-forge package-name
```

You can specify a specific package version with the install command. For example to install pandas version 1.1.3:

To remove downloads after packages are installed.

```
conda clean -t
```

When using Anaconda3/2020.07, you must use the full path to the environment in the 'Optional Environment to be activated' field. In this example it will be **/scratch/user/your_netid/Anaconda3/2020.07/envs/jupyterlab**

NOTE: When using Anaconda3/2020.07 to create a virtualenv, the installation will add lines to your ~/.bashrc file that you should delete since these lines which automatically load your virtualenv which will interfere with other jobs and modules.

Python

Default python virtualenv

You can use the default virtualenv in the JupyterLab portal app by selecting Python/3.7.4-GCCcore-8.3.0 and leaving the 'Optional Conda Environment to be activated' field blank.

The default virtualenv has Jupyterlmod installed which allows you to load compatible software modules to use in your notebook.

Type 'toolchains' on the Terra command line to see a table of compatible toolchains.

To load additional software modules, click the 'Softwares' icon in the left most part of your JupyterLab notebook. Search for modules with a compatible toolchain (such as TensorFlow/2.2.0-foss-2019b-Python-3.7.4) and click 'Load' once and wait for the LOADED MODULES section to refresh.

If you have already started your notebook before loading modules, you will need to restart the kernel in order for the loaded module to be available by clicking Kernel -> Restart Kernel… in the top JupyterLab menu or click the 'Restart the kernel' icon at the top of the notebook.

If you get 'Server Connection Error' messages after restarting the kernel, stop all other notebooks you have running by clicking the 'Running Terminals and Kernels' button in the left panel menu and then 'SHUT DOWN' all other running KERNEL SESSIONS.

Custom python virtualenv

You can create your own virtualenv to use with the JupyterLab portal app but in most cases the default virtualenv should work for you.

You must create your virtualenv using one of the Python modules listed on the JupyterLab HPRC portal webpage.

Here is an example of creating your own virtualenv on a login node.

```
module load Python/3.7.4-GCCcore-8.3.0
mkdir -p /scratch/user/your_netid/pip_envs/Python/3.7.4-GCCcore-8.3.0
cd /scratch/user/your_netid/pip_envs/Python/3.7.4-GCCcore-8.3.0
virtualenv jupyterlab
source /scratch/user/your_netid/pip_envs/Python/3.7.4-GCCcore-8.3.0/jupyterlab/bin/
↪activate
pip install juypter
pip install jupyterlab
pip install additional_packages
```

Then in the JupyterLab portal app, select the Python/3.7.4-GCCcore-8.3.0 Module and enter the **full path** of the activate command found in your virtualenv into the 'Optional Conda Environment to be activated' field.

Example of what to enter in the 'Optional Conda Environment to be activated' field:

```
/scratch/user/your_netid/pip_envs/Python/3.7.4-GCCcore-8.3.0/jupyterlab/bin/activate
```

Web Access

Although compute nodes do not have access to the internet, the JupyterLab app uses a proxy server by default which allows your JupyterLab session to have access to the internet.

Jupyter Notebook

HPRC supports three kinds of environment for Jupyter Notebooks: Conda, Module + Python virtualenv, and Singularity.

All three of these allow some customization by the user, to varying degrees. Broadly speaking:

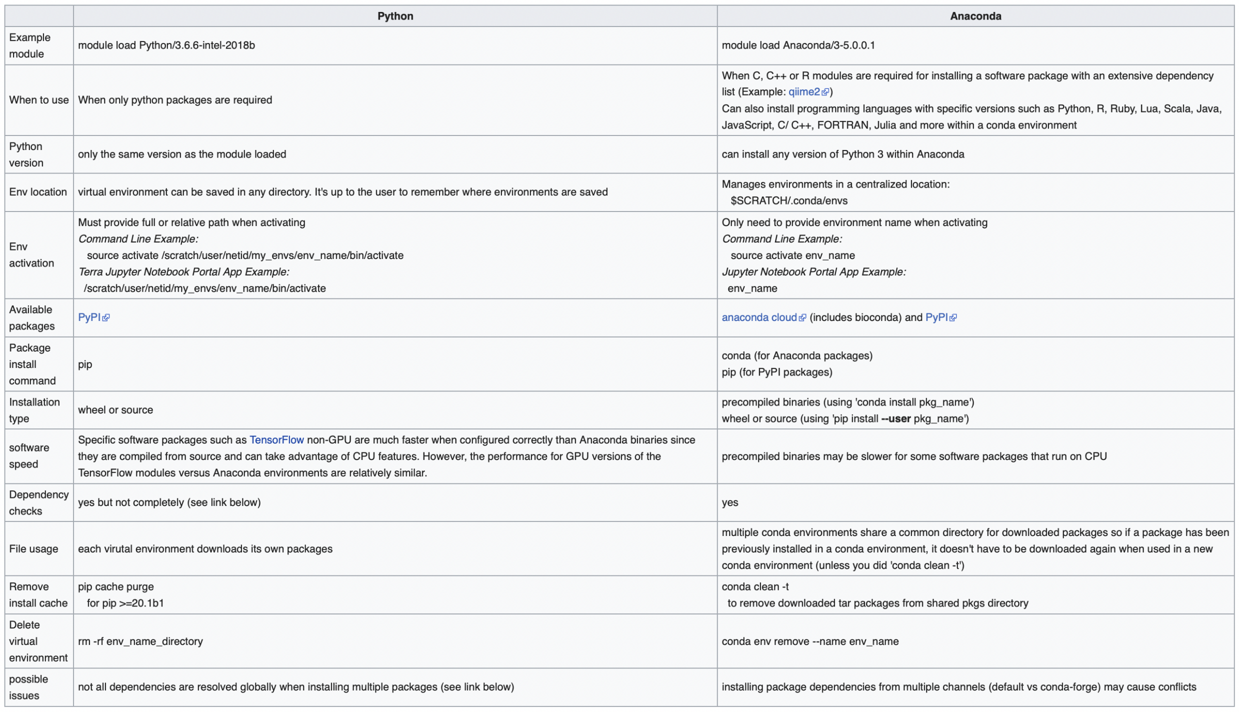
- Conda: software built by external repository. Provides quick access to commonly-used python packages. Can be extended by the user. Version choices are limited. Recommended for novice users.
- Module + Python virtualenv: software built and maintained by HPRC, optimized for use on our cluster. Can be extended by the user. New software can be requested. Recommended for experienced users.
- Singularity: software built by anyone, anywhere. Fully customizable by the user. Recommended for research groups who collaborate on software builds across multiple clusters.

HPRC provides Jupyter Notebook installations for use with our Conda and Python modules. You can also create your own Jupyter Notebook environment using either a Python environment or Anaconda environment for use on the HPRC Portal, but you must use one of the Module versions that are available on the Jupyter Notebook HPRC portal web page.

Your custom Notebook environment must be created on the command line for later use on the Jupyter Notebook portal app.

Notice that you will need to make sure you have enough available file quota (~10,000) since conda and pip creates thousands of files.

This table can help you decide when to use a Python module and when to use an Anaconda module for installing python packages.



	Python	Anaconda
Example module	module load Python/3.6.6-intel-2018b	module load Anaconda/3-5.0.0.1
When to use	When only python packages are required	When C, C++ or R modules are required for installing a software package with an extensive dependency list (Example: qiime2) Can also install programming languages with specific versions such as Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, Julia and more within a conda environment
Python version	only the same version as the module loaded	can install any version of Python 3 within Anaconda
Env location	virtual environment can be saved in any directory. It's up to the user to remember where environments are saved	Manages environments in a centralized location: $SCRATCH/.conda/envs
Env activation	Must provide full or relative path when activating *Command Line Example:* source activate /scratch/user/netid/my_envs/env_name/bin/activate *Terra Jupyter Notebook Portal App Example:* /scratch/user/netid/my_envs/env_name/bin/activate	Only need to provide environment name when activating *Command Line Example:* source activate env_name *Jupyter Notebook Portal App Example:* env_name
Available packages	PyPI	anaconda cloud (includes bioconda) and PyPI
Package install command	pip	conda (for Anaconda packages) pip (for PyPI packages)
Installation type	wheel or source	precompiled binaries (using 'conda install pkg_name') wheel or source (using 'pip install **--user** pkg_name')
software speed	Specific software packages such as TensorFlow non-GPU are much faster when configured correctly than Anaconda binaries since they are compiled from source and can take advantage of CPU features. However, the performance for GPU versions of the TensorFlow modules versus Anaconda environments are relatively similar.	precompiled binaries may be slower for some software packages that run on CPU
Dependency checks	yes but not completely (see link below)	yes
File usage	each virutal environment downloads its own packages	multiple conda environments share a common directory for downloaded packages so if a package has been previously installed in a conda environment, it doesn't have to be downloaded again when used in a new conda environment (unless you did 'conda clean -t')
Remove install cache	pip cache purge for pip >=20.1b1	conda clean -t to remove downloaded tar packages from shared pkgs directory
Delete virtual environment	rm -rf env_name_directory	conda env remove --name env_name
possible issues	not all dependencies are resolved globally when installing multiple packages (see link below)	installing package dependencies from multiple channels (default vs conda-forge) may cause conflicts

understanding-conda-and-pip

Note: you must activate the python virtualenv or anaconda environment before installing packages with 'pip install –user' or 'conda install'

Python

A Python module can be used to create a virtual environment to be used in the portal Jupyter Notebook app when all you need is Python packages.

You can use a default Python virtual environment in the Jupyter Notebook portal app by leaving the "Optional Environment to be activated" field blank.

To to create a Python virtual environment called my_notebook-python-3.6.6-intel-2018b (you can name it whatever you like), do the following on the command line. You can save your virtual environments in any $SCRATCH directory you want. In this example a directory called /scratch/user/mynetid/pip_envs is used but you can use another name instead of pip_envs

```
mkdir /scratch/user/mynetid/pip_envs
```

A good practice is to name your environment so that you can identify which Python version is in your virtualenv so that you know which module to load.

The next three lines will create your virtual environment. You can use any one of the modules listed on the Jupyter notebook portal app. Python/3.6.6-intel-2018b is used in this example

```
module purge
module load Python/3.6.6-intel-2018b
virtualenv --system-site-packages /scratch/user/mynetid/pip_envs/my_notebook-python-3.6.
↪6-intel-2018b
```

We recommend enabling –system-site-packages so that any modules you load will continue to function.

Then you can activate the virtual environment by using the full path to the activate command inside your virtual environment and install Python packages.

```
source /scratch/user/mynetid/pip_envs/my_notebook-python-3.6.6-intel-2018b/bin/activate
pip install notebook
pip install optional-python-package-name
```

You can use your Python/3.6.6-intel-2018b environment in the Jupyter Notebook portal app by selecting the Python/3.6.6-intel-2018b module in the portal app page and providing the name including full path to the activate command for your Python/3.6.6-intel-2018b virtual environment in the "Optional Environment to be activated" box (i.e. /scratch/user/mynetid/pip_envs/my_notebook-python-3.6.6-intel-2018b/bin/activate). The activate command is found inside the bin directory of your virtual env. An example of what to put in the "Optional Environment to be activated" box is the full path used in the source command above.

Loading additional Lmod modules

The default Python/3.7.4-GCCcore-8.3.0 virtualenv for JupyterLab on Terra has Jupyterlmod installed which allows the user to load any module built with the GCCcore-8.3.0 or 2019b toolchains after starting jupyter notebook in JupyterLab.

You can install the jupyterlmod package in your python virtual environment on Terra which will allow you to load additional system modules that you may need or may have used during the creation of your virtual environment on Terra for use with the Terra Jupyter Notebook portal app.

To add this feature to your existing Terra virtual environment, do the following on the command line prior to launching Jupyter Notebook on the portal (you can use Python/3.6.6-foss-2018b if the additional module(s) you need are not available with the intel-2018b toolchain):

```
module purge
module load Python/3.6.6-intel-2018b
source /scratch/user/mynetid/pip_envs/my_notebook-python-3.6.6-intel-2018b/bin/activate
pip install jupyter jupyterlmod
```

Then launch the Terra Jupyter Notebook portal app using your optional environment and click the 'Softwares' tab in your notebook and search for system modules.

Select a module or multiple modules that match the toolchain and python version that you used in creating your virtual environment and then click enter to load the module.

The 'Loaded Modules' list will update in a few seconds to reflect the additional module(s) loaded.

You can save your modules loaded using the 'collection' button at the right side of the notebook 'softwares' page so that you just have to select the collection instead of searching for modules each time you want to use your python virtual environment.

Anaconda

Anaconda is different than Python's virtualenv in that you can install other types of software such as R and R packages in your environment.

Anaconda also manages the installation path and installs in your $SCRATCH/.conda directory so you don't have to create a directory prior to creating an environment.

The recommended Anaconda module to use for Python 3 is Anaconda/3-5.0.0.1

To to create an Anaconda conda environment called my_notebook (you can name it whatever you like), do the following on the command line:

```
module purge
module load Anaconda/3-5.0.0.1
conda create -n my_notebook
```

After your my_notebook environment is created, you will see output on how to activate and use your my_notebook environment

```
#
# To activate this environment, use:
# > source activate my_notebook
#
# To deactivate an active environment, use:
# > source deactivate
#
```

Then you need to install notebook and then you can add optional packages to your my_notebook environment

```
source activate my_notebook
conda install -c conda-forge notebook
conda install -c conda-forge optional-package-name
```

You can use your Anaconda/3-5.0.0.1 environment in the Jupyter Notebook portal app by selecting the Anaconda/3-5.0.0.1 module in the portal app page and providing just the name (without the full path) of your Anaconda/3-5.0.0.1 environment in the "Optional Environment to be activated" box. In the example above, the value to enter is: **my_notebook**

Errors importing a python package

Sometimes the latest or specific versions of a python package does not work in a specific Anaconda environment due to incompatibility with the Python version or with other packages.

Default Python version

You can try installing an older version of a python package by searching available versions on anaconda.org and specify a specific version to install.

For example, if you have Python 3.6.11 and numpy 1.19.2 and see the following error:

```
import numpy

ModuleNotFoundError: No module named 'numpy.core._multiarray_umath'
```

Install an older version of numpy (the following command automatically overwrites the currently installed numpy version)

```
conda install -c conda-forge numpy=1.15.4
```

In the above example, this resolved an 'import numpy' error in an environment with Python 3.6.11 with numpy 1.19.2 by downgrading numpy which also downgraded other packages:

```
The following packages will be DOWNGRADED:

      matplotlib:       3.3.2-0                  conda-forge --> 3.2.0-1                 ␣
↪ conda-forge
      matplotlib-base:  3.3.2-py36h2451756_0     conda-forge --> 3.2.0-py36h250f245_1    ␣
↪ conda-forge
      numpy:            1.19.2-py36he0f5f23_1    conda-forge --> 1.15.4-py36h8b7e671_
↪1002 conda-forge
      scipy:            1.5.2-py36h832618f_0     conda-forge --> 1.4.1-py36h921218d_0    ␣
↪ conda-forge
```

Web Access

Jupyter Notebook runs on the compute nodes which do not have internet access.

If you need internet access for your notebook then enable the proxy using the following.

Terra

Run the following lines in your notebook:

```
import os
os.environ['http_proxy'] = '10.76.5.24:8080'
os.environ['https_proxy'] = '10.76.5.24:8080'
```

Grace

Run the following lines in your notebook:

```
import os
os.environ['http_proxy'] = '10.73.132.63:8080'
os.environ['https_proxy'] = '10.73.132.63:8080'
```

Spark Jupyter Notebook

default notebook

You can use a default Python virtual environment in the Spark Jupyter Notebook portal app by leaving the "Optional Python Environment to be activated" field blank.

The default Spark notebook uses the module Spark/2.4.0-intel-2018b-Hadoop-2.7-Java-1.8-Python-3.6.6 and the following python packages jupyter, numpy, sklearn, pandas, seaborn, pyarrow

create your own notebook

You can create your own Spark Jupyter Notebook python virtual environment for use on the HPRC Portal but you must use the following module to create your Python virtualenv

```
GRACE:
module load iccifort/2019.5.281  impi/2018.5.288  Spark/2.4.5-Python-3.7.4-Java-1.8
```

```
TERRA:
module load Spark/2.4.0-intel-2018b-Hadoop-2.7-Java-1.8-Python-3.6.6
```

Notice that you will need to make sure you have enough available file quota (~10,000) since pip creates thousands of files.

To create a Python virtual environment called my_spark_notebook-python-3.6.6-foss-2018b (you can name it whatever you like), do the following on the command line. You can save your virtual environments in any $SCRATCH directory you want. In this example a directory called /scratch/user/mynetid/pip_envs is used but you can use another name instead of pip_envs

```
mkdir -p /scratch/user/mynetid/pip_envs
```

A good practice is to name your environment so that you can identify which Python version is in your virtualenv so that you know which module to load.

The next three lines will create your virtual environment using the Spark module on Terra.

```
module purge
module load Spark/2.4.0-intel-2018b-Hadoop-2.7-Java-1.8-Python-3.6.6
export SPARK_HOME=$EBROOTSPARK
virtualenv /scratch/user/mynetid/pip_envs/my_spark_notebook-python-3.6.6-foss-2018b
```

Then you can activate the virtual environment by using the full path to the activate command inside your virtual environment and install Python packages.

First install the required dependencies (jupyter, numpy, sklearn, pandas, seaborn, pyarrow) then you can install your additional packages.

```
source /scratch/user/mynetid/pip_envs/my_spark_notebook-python-3.6.6-foss-2018b/bin/
↪activate
python3 -m pip install jupyter
python3 -m pip install numpy
python3 -m pip install sklearn
python3 -m pip install pandas
python3 -m pip install seaborn
python3 -m pip install pyarrow
```

When you are finished installing your python packages, go to the Spark Jupyter Notebook portal app and enter the full path of your virtualenv in the field 'Optional Python Environment to be activated' such as /scratch/user/mynetid/pip_envs/my_spark_notebook-python-3.6.6-foss-2018b in this example.

4.4.6 Additional Information

Ohio Supercomputing Center has video for OOD at:

https://youtu.be/DfK7CppI-IU

4.5 Galaxy

4.5.1 Maroon Galaxy Accounts

The new Maroon Galaxy (v21.01) on Grace is available to students, faculty and staff for research use.

See the Maroon Galaxy usage slides

Before you request an account on Maroon Galaxy, you must do the following:

- Go to usegalaxy.org and get familiar with Galaxy. You can start with a free account and learn about Galaxy tools.
- Request a Grace Maroon Galaxy account only if you have data to analyze, otherwise use usegalaxy.org Galaxy for training and practice.
- **If you decide that Galaxy is a good choice for your research project then do the following**
  - Establish an HPRC account by sending a request. See the NewUser page for details on how to request an account.

  - After you have your HPRC account approved, send an email to help@hprc.tamu.edu requesting an account on Maroon Galaxy
  - Send us information on what type of data you will be analyzing and which tools you expect to use for your research project.

If you are off campus then you will have to install and run the TAMU VPN to connect to Maroon Galaxy.

Maroon Galaxy can be accessed using your favorite web browser such as Firefox, Chrome or IE.

https://galaxy-grace.hprc.tamu.edu/maroon