
Northeastern University Research Computing

Release 2.0.0

Jun 22, 2023

1	Welcome	1
1.1	What is Discovery	1
2	Getting Help	3
2.1	Email	3
2.2	Submit a ticket	3
2.3	Consultations (online)	4
2.4	Status page	4
3	Getting Access	5
3.1	Request an account	5
4	Connecting to Discovery with a Mac	7
4.1	Mac	7
4.2	Using X11	8
4.3	Next steps	9
5	Connecting to Discovery using Windows	11
5.1	Passwordless ssh	11
6	Accessing Open OnDemand (OOD)	13
6.1	Next steps	13
7	Shell environment on Discovery	15
7.1	The Discovery Shell environment and .bashrc	15
7.2	About your .bashrc file	15
7.3	Conda and .bashrc	16
8	Hardware overview	19
8.1	CPU nodes	19
8.2	Using the <code>--constraint</code> flag	20
9	Partitions	21
9.1	Introduction	21
9.2	Viewing partition information	23
9.3	Allocating partitions in your jobs	23
9.4	Partition Access Request	24

10 Software overview	25
10.1 Using Make	25
10.2 Requesting Software Installation Assistance	25
11 Using Module	27
11.1 Module commands	27
11.2 Module show example	28
11.3 Module load and unload example	29
11.4 Using software applications with X11 Forwarding	29
12 Working with MATLAB on Discovery	31
12.1 Installing MATLAB toolboxes	31
12.2 Using MATLAB Parallel Server	32
13 Working with Conda/Miniconda/Anaconda	37
13.1 Creating a Conda virtual environment with Anaconda	37
13.2 Working with a Miniconda environment	38
13.3 Conda best practices	38
14 Using Spack	39
14.1 Getting started with Spack	39
14.2 Installing LAMMPS with Spack example	40
15 Working with R	41
15.1 Creating a Packrat Environment (for R)	41
15.2 A few Packrat tips	43
16 Message Passing Interface (MPI) Overview	45
16.1 MPI libraries on Discovery	45
17 Using Slurm	47
17.1 Viewing Cluster Information	47
17.2 Submitting Jobs	48
17.3 Monitoring Jobs	48
17.4 Account information	48
18 Using sbatch	51
18.1 SBATCH Examples	51
18.2 Example Parallel Job Scripts	52
18.3 Using Arrays	53
19 Using srun	55
19.1 srun examples	55
20 Working with GPUs	57
20.1 Requesting GPUs with srun or sbatch	58
20.2 Using CUDA	59
20.3 Using GPUs with PyTorch	59
20.4 Using GPUs with TensorFlow	60
21 Transferring Data	63
21.1 Data transfer node using Mac	63
21.2 Data transfer node using Windows	64
22 Using Globus	65
22.1 Globus Account Set Up	65

22.2	Install Globus Connect Personal (GCP)	66
22.3	Working with Globus	66
23	Storage Accessible on Discovery	69
24	General Data Storage Options	71
24.1	Connecting to /research	72
25	Checkpoint/Restart Discovery Jobs	73
25.1	The Checkpointing technique	73
25.2	Checkpointing types	74
26	Introduction to Open OnDemand (OOD)	79
27	Using OOD's File Explorer	81
28	Using OOD's Interactive Apps	83
28.1	Available Apps (November 2021)	83
28.2	Working with Jupyter Notebook [Custom Anaconda Environment]	85
29	Using Discovery with your class FAQ	87
30	CPS class-specific instructions	91
30.1	Open the CPS JupyterLab environment	91
30.2	Access CPS class directories	93
30.3	Submit CPS class assignments	95

CHAPTER 1

Welcome

Welcome to the official online help system for Northeastern University's Research Computing team! This documentation will help get you started with using Northeastern's Discovery cluster system. This help system contains the most up-to-date information and instructions on getting an account, connecting to the system, loading modules, running jobs, and storing data. If you are not familiar with high performance computing, you should take our training courses before working on the system. You can find information about the training and services that the Research Computing team provides at our website: <http://www.rc.northeastern.edu>.

This help system is updated frequently to reflect the latest information about Discovery, as well as to add new topics to help you with your research. Check back often to view our updates.

We want your feedback! If you have any comments or suggestions for topics you'd like to see covered in this documentation, submit a [Research Computing Documentation request](#).

1.1 What is Discovery

Discovery is a high performance computing (HPC) resource for the Northeastern University research community. The Discovery cluster is located in the Massachusetts Green High Performance Computing Center (<https://www.mghpcc.org/>) in Holyoke, MA. MGHPC is a 90,000 square foot, 15 megawatt research computing and data center facility that houses computing resources for five institutions: Northeastern, BU, Harvard, MIT, and UMass.



CHAPTER 2

Getting Help

You can get help from the Research Computing team by sending an email, submitting a ticket on ServiceNow, or making an appointment to meet with one of our staff members through our Bookings page.

2.1 Email

You can email the Research Computing team at rchelp@northeastern.edu. This will automatically generate a ticket for you in ServiceNow. Make sure to include details about your question or issue, including any commands or scripts that you are trying to use, so that we can best connect you with the right person to help you.

2.2 Submit a ticket

You can submit a ticket on ServiceNow by selecting from the [Research Computing ServiceNow catalog](#). You might need to sign in with your Northeastern username and password to view the ServiceNow catalog page.

You can also use the short URLs below to go directly to a ticket with the service that you need:

- Get Assistance with Research Computing <https://bit.ly/NURC-Assistance>
- Research Computing Access Request <https://bit.ly/NURC-AccessRequest>
- Discovery Cluster Software Request <https://bit.ly/NURC-Software>
- Research Computing Documentation Request <https://bit.ly/NURC-Documentation>
- Discovery Cluster Partition Access <https://bit.ly/NURC-PartitionAccess>
- New Storage Space request <https://bit.ly/NURC-NewStorage>
- Storage Space Extension Request <https://bit.ly/NURC-StorageExtension>
- Data Transfer Consultation <https://bit.ly/NURC-DataTransfer>
- Discovery Classroom Use Request - <https://bit.ly/NURC-Classroom>

- Unsubscribe from Mailing list - Discovery <https://bit.ly/NURC-Unsubscribe>

2.3 Consultations (online)

We encourage you to schedule a consultation with one of our staff members to get personal, one-on-one assistance for your research computing and data storage needs. Consultations are available to any Northeastern student, faculty, or staff member. We can assist you with getting up and running with Discovery, help to optimize your code, help you with benchmarking, provide assistance with installing and using software packages, detailing data storage options, and much more.

We offer scheduled consultation hours most weekdays during normal business hours (9AM to 5PM). Note that we follow the Northeastern University holiday schedule, so there are no consultation hours on holidays or during breaks. All of our consultations are conducted as online meetings through the Teams app.

Use our Bookings page to see our availability and to schedule an appointment <https://rc.northeastern.edu/support/consulting/>. You will need to sign in using your /@northeastern.edu email (for example, a.student/@northeastern.edu).

2.4 Status page

We post important information, such as power outages and maintenance windows, to the ITS Systems Status page: <https://northeastern.statuspage.io/> You can subscribe to this page to receive email updates on the status of ITS systems.

3.1 Request an account

You must first have an account before you can access Discovery. You request an account through ServiceNow. You will need a Northeastern username and password to access ServiceNow. If you are new to the university or a visiting researcher, work with your sponsor to get a Northeastern username and password.

Valid NU Credentials

Access to Discovery is limited to people affiliated with Northeastern who have a valid Northeastern username and password. Research Computing is not able to create or renew Northeastern accounts. You must work with your sponsor to obtain or update a valid Northeastern username and password.

To request an account, follow these steps:

1. Go to the [ServiceNow Research Computing Access Request form](#).
2. Fill out the form, check the acknowledgement box, and click Submit.

Your request can take up to 24 hours to process after it has been approved by your sponsor (see Sponsor Approval Process below). You will receive a confirmation email when your access has been granted. After you have access, if you are not familiar with using Discovery, high performance computing, or Linux, you should take one of our training courses. See the [Research Computing website](#) for more information on our training and services.

Important: If you previously had access to Discovery, but you are now working with a different PI, submit a [ServiceNow Research Computing Access Request form](#) and enter the name of your current PI in the Sponsor field. This will link your account to your current PI. This is important to help expedite the process for updating your account with any of your current PI's resources on Discovery, such as shared storage or a private partition.

3.1.1 Principal Investigator(PI)/Professor/Instructor Access

If you are a PI, professor, or instructor at Northeastern, and you need access to Discovery, you should use the access form in the above procedure, and in the Sponsor name field on the form, enter your own name.

3.1.2 Sponsor Approval Process

Every user on Discovery needs to have a sponsor approve their request, usually a PI or professor at Northeastern. PIs, professors, and instructors can sponsor themselves. Students (undergraduate or graduate), visiting researchers, or staff members must have a sponsor approve their request. When you fill out the ServiceNow form and enter the name of your sponsor, an email goes to the sponsor when you submit the request. Sponsors will continue to receive email reminders until they approve the request through the link in the email to ServiceNow. We recommended that before you submit an access request, let your sponsor know to look for the email with the link to the approval page to help expedite the process.

Connecting to Discovery with a Mac

You connect to Discovery using a [secure shell](#) program to initiate an SSH session to sign in to Discovery. If you usually launch software from the command line that uses a graphical user interface (GUI), see [Using X11](#) for tips and troubleshooting information.

4.1 Mac

Mac computers come with a Secure Shell (SSH) program called [Terminal](#) that you use to connect to Discovery using SSH. If you need to use software that uses a GUI, such as Matlab or Maestro, make sure to use the `-Y` option in the second step below (see [Using X11](#) for more tips and troubleshooting information).

Note: If you use Mac OS X version 10.8 or higher, and you have [XQuartz](#) running in the background to do X11 forwarding, you should execute the following command in Terminal once before connecting to Discovery:

```
defaults write org.macosforge.xquartz.X11 enable_iglx -bool true
```

You should keep XQuartz running in the background. If you close and restart XQuartz, you will need to execute the above command again after restarting. Do not use the Terminal application from within XQuartz to sign in to Discovery. Use the default Terminal program that comes with your Mac (see Step 1 in the procedure below).

To connect to Discovery on a Mac:

1. Go to Finder > Applications > Utilities, and then double click Terminal.
2. At the prompt, type `ssh <username>@login.discovery.neu.edu`, where `<username>` is your Northeastern username. If you need to use X11 forwarding, type `ssh -Y <username>@login.discovery.neu.edu`.
3. Type your Northeastern password and press Enter.

You are now connected to Discovery at a login node.

Watch this video of how to connect to Discovery on a Mac. If you do not see any controls on the video, right click on the video to see viewing options.

4.2 Using X11

When you launch a software application that uses a graphical user interface (GUI) from the command line, this is completed through X11 forwarding. If you use MobaXterm on Windows, X11 forwarding is turned on by default. If you use the Terminal program on Mac, you'll need to log in using the `-Y` option (`ssh -Y <yourusername>@login.discovery.neu.edu`).

Tip: If you used the `-Y` option to enable X11 forwarding on your Mac, you can test to see if it is working by typing `xeyes`. This will run a small program that makes a pair of eyes appear to follow your cursor.

4.2.1 Passwordless ssh

You need to setup passwordless ssh to ensure that GUI-based applications will launch without any issues. You also need to make sure that your keys are added to the “authorized_key” file. This needs to be done anytime you regenerate your keys. If you're having an issue with opening an application that need X11 forwarding, such as MATLAB or Schrodinger, and you recently regenerated your keys, make sure to add your keys to the “authorized_key” file.

Note:

Errors that you can see on both Mac and Windows when launching a GUI-based program include the following:

```
Error:  unable to open display localhost:19.0
```

```
Launch failed:  non-zero return code
```

If you are getting these types of errors, follow the steps below to set up passwordless ssh.

To setup passwordless ssh:

1. On a Mac, not yet connected to Discovery, open Terminal and type, `cd ~/.ssh`. This moves you to the ssh folder on your local computer. **Note:** Make sure you're on your local computer for steps 1 through 4. If you are connected to Discovery, type `exit` to return to your local computer.
2. Type `ssh-keygen -t rsa` to generate two files: `id_rsa` and `id_rsa.pub`.
3. Press `Enter` to all of the prompts (do not generate a passphrase).
4. Type `ssh-copy-id -i ~/.ssh/id_rsa.pub <yourusername>@login.discovery.neu.edu` to copy `id_rsa.pub` to your `/home/.ssh` folder on Discovery. This step automatically copies your token from the `id_rsa.pub` file to a “authorized_keys” file which will either be generated or appended if it already exists. You will be prompted to enter your NU password.
5. Connect to Discovery again by typing `ssh <yourusername>@login.discovery.neu.edu`. You should now be connected without having to enter your password.

Note: If you are using a Windows machine using MobaXterm, sign in to Discovery as usual, then complete steps 6 through 9 to complete the passwordless ssh setup.

6. Type `cd ~/.ssh` to move to your ssh folder.
7. Type `ssh-keygen -t rsa` to generate your key files.
8. Press `Enter` to all of the prompts (do not generate a passphrase). If prompted to overwrite a file, type `Y`.
9. Type `cat id_rsa.pub >> authorized_keys`. This adds the contents of your public key file to a new line in the `~/.ssh/authorized_keys` file.

4.3 Next steps

After you are connected, you can run jobs either in interactive mode with `srun` or submit a script using `sbatch`. See [Using `srun`](#) and [Using `sbatch`](#) for more information.

To load and run software, see [Software overview](#). To find out more about the hardware and partitions on Discovery, see [Hardware overview](#) and [Partitions](#).

To watch an introductory training video, go to [Northeastern's LinkedIn Learning page](#).

Connecting to Discovery using Windows

You connect to Discovery using a [secure shell](#) program to initiate an SSH session to sign in to Discovery. This topic is about how to connect using Windows. See [Connecting to Discovery with a Mac](#) for information about connecting with a Mac.

Before you can connect to Discovery on a Windows computer, you'll need to download a terminal program, such as [MobaXterm](#) or PuTTY. We recommend MobaXterm, as you can also use it for file transfer, whereas with other SSH programs, you would need a separate file transfer program.

To connect to Discovery with MobaXterm:

1. Open MobaXterm.
2. Click **Session**, then click **SSH** as the connection type.
3. In **Remote Host**, type `login.discovery.neu.edu`, make sure **Port** is set to 22, and click **OK**. (OPTIONAL: You can type your Northeastern username and password on MobaXterm, and it will save that information every time you sign in. If you opt to do this, you will be connected to Discovery after you click **OK**.)
4. At the prompt, type your Northeastern username and press Enter.
5. Type your Northeastern password and press Enter. Note that the cursor does not move as you type your password. This is expected behavior.

You are now connected to Discovery at a login node.

Watch this video to see how to connect to Discovery with MobaXterm. If you do not see any controls on the video, right click on the video to see viewing options.

5.1 Passwordless ssh

You need to setup passwordless ssh to ensure that GUI-based applications will launch without any issues. You also need to make sure that your keys are added to the `authorized.key` file. This needs to be done anytime you regenerate your keys. If you're having an issue with opening an application that need X11 forwarding, such as MATLAB or Schrodinger, and you recently regenerated your keys, make sure to add your keys to the `authorized.key` file.

Note:

Errors that you can see on Windows when launching a GUI-based program include the following:

```
Error:  unable to open display localhost:19.0
```

```
Launch failed:  non-zero return code
```

If you are getting these types of errors, follow the steps below to set up passwordless ssh.

To setup passwordless ssh:

1. Sign into Discovery using MobaXterm.
2. Type `cd ~/ .ssh` to move to your ssh folder.
3. Type `ssh-keygen -t rsa` to generate your key files.
4. Press `Enter` to all of the prompts (do not generate a passphrase). If prompted to overwrite a file, type `Y`.
5. Type `cat id_rsa.pub >> authorized_keys`. This adds the contents of your public key file to a new line in the `~/ .ssh/authorized_keys` file.

The following video tutorial goes through this process.

5.1.1 Next steps

After you are connected, you can run jobs either in interactive mode with `srun` or submit a script using `sbatch`. See [Using srun](#) and [Using sbatch](#) for more information.

To load and run software, see [Software overview](#). To find out more about the hardware and partitions on Discovery, see [Hardware overview](#) and [Partitions](#).

To watch an introductory training video, go to [Northeastern's LinkedIn Learning page](#).

Accessing Open OnDemand (OOD)

Open OnDemand (OOD) is a web portal to the Discovery cluster.

This topic is for connecting to the Discovery cluster through the browser application, Open OnDemand. If you are looking to access Discovery directly on your system rather than through a browser, please see *Connecting to Discovery with a Mac* for connecting with a Mac and *Connecting to Discovery using Windows* for connecting with Windows.

A Discovery account is necessary for you to access OOD. If you need an account, see *Request an account*. After you have created a Discovery account, proceed with the following steps in order to access Discovery.

To Access Discovery through Open OnDemand (OOD), do the following

1. In a web browser, go to <http://ood.discovery.neu.edu>.
2. At the prompt, enter your Northeastern username and password. Note that your username is the first part of your email without the @northeastern, such as **j.smith**.
3. Press **Enter** or click **Sign in**.

Watch the following video for a short tutorial. If you do not see any controls on the video, right click on the video to see viewing options.

6.1 Next steps

After you are connected, you use the interactive apps in OOD. See *Using OOD's Interactive Apps* for more information.

Shell environment on Discovery

7.1 The Discovery Shell environment and .bashrc

Discovery uses a Linux-based Operating System (CentOS), where the Shell program is used to interface with the user. Bash (Bourne Again SHell) is one of the most popular Shell implementations which is the default Shell on Discovery.

The Shell script .bashrc is used by Bash to initialize your Shell environment. For example, it is typically used to define aliases, functions and load modules. Note that environment variables settings (such as `PATH`) generally go in the .bash_profile or .profile files. Your .bashrc, .bash_profile and .profile files live in your /home directory. You can make changes to your .bashrc with a text editor like [nano](#).

Caution: Making edits to your .bashrc file can result in many issues. Some changes may prevent you from launching apps or executing commands. Modifying your `PATH` variable may result in the inability to use basic Shell commands (such as `cd` or `ls`) if not done correctly. Before making changes to your .bashrc file, make a backup of the default .bashrc file so you can restore it if necessary. If you need help with editing your .bashrc file, reach out to rchelp@northeastern.edu or [schedule a consultation with a staff member](#) who can help suggest edits and troubleshoot any issues you might be having.

7.2 About your .bashrc file

You have a default .bashrc file in your home directory when your account is created. See the figure below for an example of a default .bashrc file.

```
[ju.cho@login-01 ~]$ cat .bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

At a command prompt in your /home directory, type `cat .bashrc` to view the contents of your `bashrc` file. This is an example of a default `bashrc` file (it has no modifications).

Important: We recommend to keep `.bashrc` unchanged when using Discovery. You can source environment Shell scripts or load modules directly inside your job instead. This approach can prevent some runtime errors from loading incompatible modules, setting environment variables incorrectly, or from mixing multiple software and Conda environments.

7.3 Conda and `.bashrc`

In addition to editing your `.bashrc` file as outlined in the example above, programs that you install can also modify your `.bashrc` file. For example, if you follow the procedure outlined in [Working with a Miniconda environment](#), there may be a section added to your `.bashrc` file (if you didn't use the `-b` batch option) that automatically loads your conda environment every time you sign in to Discovery. See the figure below for an example of this:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$(('/home/ju.cho/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/ju.cho/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/ju.cho/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/ju.cho/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

You should not modify this section in the `.bashrc` file directly. If it was changed, remove this section manually using a file editor.

Caution: We recommend removing the conda initialization section from your `.bashrc` as it may interfere with the correct startup environment when using Open OnDemand apps. You should always load your Conda environment after your job already started.

If you need help with your `.bashrc` file or would like it restored to its default, reach out to the RC team at rchelp@northeastern.edu, and we can provide you with a new, default `.bashrc` file and/or help troubleshoot issues with the file.

7.3.1 Editing your `.bashrc` file

The basic workflow for editing your `.bashrc` file is to sign in to Discovery, go to your `/home` directory, open the file in a text editor on the command line, make your edits, save the file, sign out of Discovery, then sign back in again. Your changes will take effect when you have signed back in again.

Example procedure for editing your `.bashrc` file:

1. Sign in to Discovery.
2. (Optional) Type `PWD` to make sure you are in your `/home` directory.
3. (Optional) Type `ls -a` to view the contents of your `/home` directory, including hidden files. Your `.bashrc` file is a hidden file (hidden files are preceded by a `.`). Using the `-a` option with `ls` displays hidden files.
4. (Recommended) Type `cp .bashrc .bashrc-default` to make a copy of your `.bashrc` file called `.bashrc-default`.
5. Type `nano .bashrc` to open your `.bashrc` file in the nano text editor.
6. Type the edits that you want to make to your file. In this example, an alias was added to create a shortcut to the user's `/scratch` space.

```

GNU nano 2.3.1 File: .bashrc Modified
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions

# >>> conda initialize >>>
# ! Contents within this block are managed by 'conda init' !!
__conda_setup="$(/home/ju.cho/miniconda3/bin/conda 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/ju.cho/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/ju.cho/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/ju.cho/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<

# alias
alias scratch='cd /scratch/ju.cho'
  
```

Alias section added with an alias called scratch.

7. Save the file and exit the editor.
8. Sign out of Discovery and sign back in for the changes to take effect.

7.3.2 Sourcing a Shell script example

A safe alternative to using `.bashrc` is to source a Shell script inside your runtime job environment. Below is an example script to load an Anaconda module and source a Conda environment which will then be used inside the Slurm script.

Create a Shell script `myenv.bash`:

```
#!/bin/bash
module load anaconda3/2021.05
module load cuda/11.1
source activate pytorch_env_training
```

Then, source the Shell script inside your sbatch Slurm script (see [Using sbatch](#)):

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --gres=gpu:1
#SBATCH --time=01:00:00
#SBATCH --job-name=gpu_run
#SBATCH --mem=4GB
#SBATCH --ntasks=1
#SBATCH --output=myjob.%j.out
#SBATCH --error=myjob.%j.err

source myenv.bash
python <myprogram>
```

Hardware overview

The Discovery computing cluster provides you with access to over 1024 CPU nodes, 50,000 CPU cores, and over 200 GPUs and is connected to the university network over 10 Gbps Ethernet (GbE) for high-speed data transfer. Compute nodes are connected to each other with either 10 GbE or a high-performance HDR200 InfiniBand (IB) interconnect running at 200 Gbps (with some nodes running HDR100 IB, if HDR200 IB is not supported on those nodes).

8.1 CPU nodes

Table 1 below shows the feature names, number of nodes by partition type (public and private), and the RAM memory range per node. The feature name follows archspec microarchitecture [specification](#).

Feature Name	Number of Nodes	RAM memory
	public, private	per node
skylake	0, 170	186 - 3094 GB
zen2	40, 292	256 - 2000 GB
zen	40, 300	256 - 2000 GB
ivybridge	64, 130	31 - 1031 GB
sandybridge	8, 0	384 GB
haswell	230, 62	109 - 1031 GB
broadwell	756, 226	128 - 515 GB
cascadelake	260, 88	186 - 3094 GB

If you are looking for information about GPUs, see [Working with GPUs](#).

If you are interested in more information about the different partitions on Discovery, including the number of nodes per partition, running time limits, job submission limits, and RAM limits, see [Partitions](#).

8.2 Using the `--constraint` flag

When using `srun` or `sbatch`, you can specify hardware features as part of your job by using the `--constraint=` flag. This may be particularly useful when benchmarking, optimizing, or if you're using code that was compiled on a certain micro-architecture. Currently, you can use the `--constraint=` flag to restrict your job to a specific feature name (e.g., `haswell`, `ivybridge`) or you can use the flag: `ib` to only include nodes that are connected by InfiniBand (IB) with a job that needs to use multiple nodes.

A few examples using `srun`:

```
1 srun --constraint=haswell --pty /bin/bash
2 srun --constraint=ivybridge --pty /bin/bash
3 srun --constraint=ib --pty /bin/bash
4 srun --constraint="[ivybridge|zen2]" --pty /bin/bash #this uses the OR operator
↪ | to select either an ivybridge or zen2 node.
```

You can add these same flags as an additional line in your `sbatch` script via (`#SBATCH --constraint=haswell`)

Note: Using the `--constraint` flag can mean that you will wait longer for your job to start, as the scheduler (Slurm) will need to find and allocate the appropriate hardware that you have specified for your job. For more information about running jobs, see [Using Slurm](#). Finally, at this time only the OR operator `|` is supported when using `--constraint`.

test

Updated July 31, 2020

9.1 Introduction

Partitions are logical collections of nodes that comprise different hardware resources and limits to help meet the wide variety of jobs that get scheduled on Discovery. Occasionally, the Research Computing team might need to make updates to the partitions based on monitoring job submissions to help reduce job wait times. As our cluster grows, changes to the partitions also help to ensure the fair, efficient distribution of resources for all jobs being submitted to the cluster.

On Discovery, there are several different partitions:

- General access (`debug`, `express`, `short`, `gpu`)
- Application only (`long`, `large`, `multigpu`)
- PI owned (accessed only by members of the PIs' group)

The general access and application only partitions span the hardware on Discovery, with `gpu` and `multigpu` spanning the GPUs on Discovery and the other partitions spanning the CPUs. For example, if you use the `debug` partition you're using the same hardware as `short`, just with different time, job, and core limits. Refer to the tables below for detailed information on the current partitions. Note that PI-owned partitions only include the hardware that those PIs own and are only accessible to the members of the PI's group.

Note: In the following table, the Running Jobs Per User/Per Research Group. Core and RAM limits are set per user, across all running jobs (not pending).

Name	Requires approval?	Time limit (default/max)	Running jobs	Submitted jobs	Core limit (per user)	RAM limit	Use Case
debug	No	20 minutes/20 minutes	10/25	5000	128	256GB	Best for serial and parallel jobs that can run under 20 minutes. Good for testing code.
express	No	30 minutes/60 minutes	50/250	5000	2048	25TB	Best for serial and parallel jobs that can run under 60 minutes.
short	No	4 hours/24 Hours	50/500	5000	1024	25TB	Best for serial or small parallel jobs (<code>--nodes=2</code> max) that need to run for up to 24 hours.
long	Yes	1 day/5 Days	25/250	1000 user/5000 per group	1024	25TB	Primarily for serial or parallel jobs that need to run for more than 24 hours. Need to prove that your code cannot be checkpointed to use this partition.
large	Yes	6 hours/6 Hours	100/100	1000 user/5000 per group	N/A	N/A	Primarily for running parallel jobs that can efficiently use more than 2 nodes. Need to demonstrate that your code is optimized for running on more than 2 nodes.

Name	Requires approval?	Time limit (default/max)	Running jobs	Submitted jobs	GPU per job limit	GPU per user limit	Use Case
gpu	No	4 hours/8 Hours	25/250	50/100	1	8	For jobs that can run on a single GPU processor.
multigpu	Yes	4 hours/24 Hours	25/100	50/100	12	12	For jobs that require more than one GPU and take up to 24 hours to run.

9.2 Viewing partition information

Slurm commands allow you to view information about the partitions. Three commands that can show you partition information are `sinfo`, `sacct`, and `scontrol`. The following are common options to use with these commands:

```
sinfo -p <partition name> #displays the state of the nodes on a specific partition
sinfo -p <partition name> --Format=time,nodes,cpus,socketcorethread,memory,nodeai,
↪features #displays more detailed information using the Format option, including
↪features like the type of processors
sacct --partition <partition name> #displays the jobs that have been run on this
↪partition
scontrol show partition <partition name> #displays the Slurm configuration of the
↪partition
```

For more information about these commands, see the Slurm documentation site <https://slurm.schedmd.com/>.

9.3 Allocating partitions in your jobs

To specify a partition when running jobs, use the option `--partition=<partition name>` with either `srun` or `sbatch`. When using a partition with your job and specifying the options of `--nodes=` and `--ntasks=`, make sure that you are requesting options that best fit your job. **Requesting the maximum number of nodes or tasks will not make your job run faster or give you higher priority in the job queue.** It can actually have the opposite effect on jobs that are better suited to running with smaller requirements, as you have to wait for the extra resources that your job will not use. See *Using Slurm* for more information on using Slurm to run jobs.

Tip: You should always try to have job requests that will attempt to allocate the best resources for the job you want to run. For example, if you are running a job that is not parallelized, you only need to request one node (`--nodes=1`). For some parallel jobs, such as a small MPI job, you can also use one node (`--nodes=1`) with the `--ntasks=` option set to correspond to the number of MPI ranks (tasks) in your code. For example, for a job that has 12 MPI ranks, request 1 node and 12 tasks within that node (`--nodes=1 --ntasks=12`). If you request 12 nodes, Slurm is going to run code between those nodes, which could slow your job down significantly if it isn't optimized to run between nodes.

If your code is optimized to run on more than 2 nodes and needs less than one hour to run, you can use the express partition. If your code needs to run on more than 2 nodes for more than one hour, you should apply to use the large partition. See the section Partition Access Request below for more information.

9.4 Partition Access Request

If you need access to the large, long, or multigpu partition, you need to submit a [ServiceNow ticket](#). Access is not automatically granted. You will need to provide details and test results that demonstrate your need for access for these partitions. If you need temporary access to multigpu to perform testing before applying for permanent access, you should also submit a [ServiceNow ticket](#). All requests are evaluated by members of the RC team, and multigpu requests are also evaluated by two faculty members.

CHAPTER 10

Software overview

Discovery offers you many options for working with software. Two of the easiest and most convenient ways are using the `module` command on the command line and using the interactive apps on Open onDemand (OOD), Discovery's web portal. If you need a specific software package, first check to see if it is already available through one of the preinstalled modules on Discovery. The Research Computing team adds new modules regularly, so use the `module avail` command to view the most up to date list. You can also try using Spack, a software package manager available on Discovery. Spack has over 5000 packages that you can install.

See *Using Module* for more information about working with module.

See *Using OOD's Interactive Apps* for more information about OOD.

See *Using Spack* for more information about Spack.

You can also use Conda, Miniconda, and Anaconda to manage software packages. See *Working with Conda/Miniconda/Anaconda* for more information.

10.1 Using Make

If you want to use `make` to add software locally to your path you must first download the software package from its source (such as a webpage or Github), and you need to unpack it or unzip it, if it is archived/zipped. Then, you must set the installation path to a directory where you have write access on Discovery, such as your home directory. You can use `./configure` to do this, such as `./configure --prefix=/home/<yourusername>/software`. After you have set the install path, you need to compile the code using `make` and then install the software using `make install`.

10.2 Requesting Software Installation Assistance

If the software that you need is not a module on Discovery, cannot be installed through Spack, or is not available through another way of self-installation (such as using `make`), you can submit a [ServiceNow software request ticket](#). Be aware that there might be packages that cannot be installed on Discovery due to incompatibility with the hardware on Discovery.

CHAPTER 11

Using Module

The module system on Discovery includes many commonly used scientific software packages that you can load in your path when you need it and unload it when you no longer need it.

Use the `module avail` command to show a list of the most currently available software on Discovery.

Note: Some modules might conflict with each other, resulting in the software not behaving as expected. Also, if there are multiple versions of software, and you load more than one version of the software, only the latest version will be used. Use `module list` to view the modules you currently have loaded in your path.

Tip: Use the `which` command to display which version of a software package you have in your path. For example, `which python` will display the version of python that you have in your path.

11.1 Module commands

The following are common module commands that are useful for interacting with software packages on Discovery.

Module Command	Function
<code>module avail</code>	View a list of all of the available software packages on Discovery that you can load
<code>module list</code>	Displays a list of the software packages currently loaded in your path
<code>module show <module name></code>	View the details of a software package (see the section “Module Show” below for more information)
<code>module load <module name></code>	Load a software package into your environment
<code>module unload <module name></code>	Remove a single software package from your environment
<code>module purge</code>	Removes all of the loaded software packages from your environment.

Caution: Using `module purge` will purge all modules from your environment, including the default module `discovery/2019-02-21`. This module contains the http proxy needed for nodes to have internet access. If you accidentally purge this module, it will be automatically reloaded the next time you log out and log back in again. You can also load it manually if you have purged it by using the `module load` command.

11.2 Module show example

Before loading a module, type `module show <name of module>` to see if there are any dependencies or commands that you need to execute before loading the module. In some cases, a module might depend on having other modules loaded to work as expected. While modules are a convenient way of loading software to use on Discovery, scientific software can come with many packages and dependencies. In addition to `module`, you should review other ways of loading software on Discovery. See [Software overview](#) for more information on different ways you can install software on Discovery. The figure below shows an example of `module show` with the software package called `amber`.

```
[ju.cho@login-01 ~]$ module show amber
-----
/shared/centos7/modulefiles/amber/18-mpi:

module-whatis    loads the modules environment for Amber 18 MPI parallel executable on CPU nodes.

Please load the following modules:
module load openmpi/3.1.2
module load amber/18-mpi
module load python/2.7.15

setenv          AMBER_HOME /shared/centos7/amber/amber18-cpu
prepend-path    PYTHONPATH /shared/centos7/amber/amber18-cpu/lib/python2.7/site-packages
prepend-path    PATH /shared/centos7/amber/amber18-cpu/bin
prepend-path    LD_LIBRARY_PATH /shared/centos7/amber/amber18-cpu/lib
prepend-path    C_INCLUDE_PATH /shared/centos7/amber/amber18-cpu/include
prepend-path    CPLUS_INCLUDE_PATH /shared/centos7/amber/amber18-cpu/include
-----
```

11.3 Module load and unload example

In the figure below, the software module stata/15 was loaded and then unloaded. After loading and unloading, module list was used to check that the STATA was loaded and unloaded.

```
[ju.cho@login-00 ~]$ module load stata/15
[ju.cho@login-00 ~]$ module list
Currently Loaded Modulefiles:
  1) discovery/2019-02-21  2) stata/15
[ju.cho@login-00 ~]$ module unload stata/15
[ju.cho@login-00 ~]$ module list
Currently Loaded Modulefiles:
  1) discovery/2019-02-21
```

11.4 Using software applications with X11 Forwarding

If you are attempting to open a GUI-based software application that uses X11 forwarding to display, such as MATLAB or Maestro, and you get an error such as `Error: unable to open display localhost:19.0`, this is most likely due to an issue with passwordless SSH. See [Using X11](#) for tips and troubleshooting information opening applications that use X11 forwarding.

Working with MATLAB on Discovery

MATLAB is available as a module on Discovery (see *Using Module* for more information), and it is also an interactive app on Open onDemand (see *Introduction to Open OnDemand (OOD)* for more information). You can also download MATLAB for use with your personal computer through the [Northeastern portal on the MATLAB website](#). Note that the procedures detailed below are specific to using MATLAB on Discovery and not with using MATLAB on your personal computer.

12.1 Installing MATLAB toolboxes

Use the following procedure if you need to install a MATLAB toolbox:

1. Download the toolbox from its source website.
2. Connect to Discovery.
3. Create a directory in your /home directory. We recommend creating a directory called `matlab` by typing:

```
mkdir /home/<username>/matlab #where <username> is your username
```

4. Go to the directory you just created by typing:

```
cd /home/<username>/matlab
```

5. Unzip the toolbox file by typing:

```
unzip <toolboxname>
```

6. Load MATLAB by typing:

```
module load matlab
```

7. Start MATLAB by typing:

```
matlab
```

8. Add the toolbox to your PATH by typing:

```
addpath('/home/<username>/matlab/<toolbox>') #where <toolbox> is the name of the_  
↪ toolbox you just unzipped
```

9. If this is a toolbox you want to use more than once, you should save it to your path by typing:

```
savepath()
```

10. You can now use the toolbox within MATLAB. When you are done, type `quit`.

12.2 Using MATLAB Parallel Server

The Discovery cluster has MATLAB Parallel Server installed. This section details an example of how you can setup and use the MATLAB Parallel Computing Toolbox. This walkthrough uses MATLAB 2020a launched as an interactive app on the Open onDemand web portal. There are several parts to this walkthrough. We suggest that you read it through completely before starting. The parameters presented represent only one scenario.

This walkthrough will use Open onDemand, the web portal on Discovery, to launch MATLAB. You'll then create a cluster profile. This allows you to define cluster properties that will be applied to your jobs. Supported functions are *batch*, *parpool*, and *parcluster*. The Parallel Computing Toolbox comes with a cluster profile called *local*, which you will change in the walkthrough below.

Note: This walkthrough details submitting jobs through Discovery's Open onDemand web portal. Some parameters will vary if you are using MATLAB from the command line. This walkthrough does not apply to other versions of MATLAB.

Before starting, you should create a folder in your `/scratch/<yourusername>` directory. This folder is where you'll save your job data.

1. Go to your `/scratch` directory: `cd /scratch/<yourusername>` where `<yourusername>` is your NU username
2. Make a new folder. We suggest calling it *matlab-metadata*: `mkdir matlab-metadata`

To start MATLAB and add a Cluster Profile, do the following:

1. Go to <http://ood.discovery.neu.edu>. If prompted, sign in with your Discovery username and password.
2. Click **Interactive Apps**, and select **MATLAB**.
3. Select **MATLAB version 2020a**, and keep the default time of 1 hour and default memory of 2GB. Click **Launch**.
4. If necessary, adjust the **Compression** and **Image Quality**, and then click **Launch MATLAB**.
5. On the MATLAB Home tab, in the **Environment** section, select **Parallel**, then click **Create and Manage Clusters**. This opens the Cluster Profile Manager window.
6. On the Cluster Profile Manager window, select **Add Cluster Profile**, then click **Slurm**. If prompted, click **OK** to the notice about needing Parallel Server.
7. Double click the new profile name in the Cluster Profile column, and type a name such as **TestProfile**. Press **Enter** to save the change.

8. Select **Edit** in the **Manage Profile** section. This lets you edit the options on the **Properties** tab. For this walkthrough, make the following edits:
 - a. In the **Folder where job data is stored on the client** option, type `/scratch/<yourusername>/matlab-metadata` (this is the directory that you created in the first procedure above).
 - b. In the **Number of workers available to cluster** option, type a number between 1 and 10. This field is the number of MPI processes you intend to run. This corresponds to the `--ntasks` Slurm option. The maximum is 128 per job; however, for this task, we recommend keeping it lower and use threading inside the nodes. The number you set here will be the default maximum for the job. You can set it for less than or equal to this number in the MATLAB Command Window when submitting your job.
 - c. In the **Number of computational threads to use on each worker** option, type a number between 1 and 10. This field represents the number of threads that each worker will possess. This corresponds to `cpus-per-task` in Slurm. Do not exceed the number of available cores on the node.
9. When you have finished editing your properties, click **Done**.
10. (Optional) If you want to validate your setup, click the **Validation** tab (next to the Properties tab). Ensure all of the stages are checked, then click the **Validate** button at the bottom of the page. This will check the properties of your profile. You might need to wait a minute or two for this to complete.

Caution: Do not click the green **Validate** button. This will attempt validation using the maximum number of workers, which can cause the validation to hang or fail. If you accidentally click the green Validate button, click **Stop** to end the validation process.

(OPTIONAL) In the **Cluster Profile** column, right-click on the TestProfile name and select **Set as Default**. This sets your profile to be the default.

Now that you have set up your profile, you can use the default cluster profile you just created (*TestProfile*) with the following commands:

```
#with parpool
parallel.defaultClusterProfile('TestProfile')
parpool

#with parcluster
c = parcluster('TestProfile')
```

12.2.1 Using parcluster example

This section will detail how to submit batch jobs to the cluster to perform scaling calculations for an integer factorization sample problem. It's a computationally intensive problem, where the complexity of the factorization increases with the magnitude of the number. We'll use the `myParallelAlgorithmFcn.m` MATLAB function. This section assumes you have configured a MATLAB Cluster Profile according to the procedure above.

On Discovery, there are benchmarking scripts and examples located in the `/shared/centos7/matlab/R2020a/examples/parallel/main` folder. To add the path to this folder to the list of available paths, do one of the following:

- On the MATLAB Home tab, in the **Environment** section, click **Set Path** and add the path to the script.
- Alternatively, provide the full path of the script in the MATLAB command line.

The contents of `myParallelAlgorithmFcn` is as follows:

```
function [numWorkers,time] = myParallelAlgorithmFcn ()

complexities = [2^18 2^20 2^21 2^22];
numWorkers = [1 2 4 6 16 32 64];

time = zeros(numel(numWorkers),numel(complexities));

% To obtain obtain predictable sequences of composite numbers, fix the seed
% of the random number generator.
rng(0,'twister');

for c = 1:numel(complexities)

    primeNumbers = primes(complexities(c));
    compositeNumbers = primeNumbers.*primeNumbers(randperm(numel(primeNumbers)));
    factors = zeros(numel(primeNumbers),2);

    for w = 1:numel(numWorkers)
        tic;
        parfor (idx = 1:numel(compositeNumbers), numWorkers(w))
            factors(idx,:) = factor(compositeNumbers(idx));
        end
        time(w,c) = toc;
    end
end
```

To submit myParallelAlgorithmFcn as a batch job, in the MATLAB Command Window, type:

```
totalNumberOfWorkers = 65;
cluster = parcluster('TestProfile');
job = batch(cluster,'myParallelAlgorithmFcn',2,'Pool',totalNumberOfWorkers-1,
↳ 'CurrentFolder','.');
```

This specifies the totalNumberOfWorkers as 65, where 64 workers will be issued to run *parfor* in parallel (so the pool is set as 64), and the additional worker will run the main process.

To monitor the job after you submit it, click **Parallel**, then **Monitor Jobs** to open the Job Monitor. You can view some job information, such as the state of the job (i.e. running, failed, finished etc.), as well as the ability to fetch outputs if you right-click on the job line.

You can close MATLAB after you submit the job the scheduler. The job monitor tool will keep track of the jobs.

If you want to block MATLAB until the jobs are finished, type `Wait(job)`.

When the jobs complete, you can transfer the outputs of the function using the `fetchOutputs` command:

```
outputs = fetchOutputs(job);
numWorkers = outputs{1};
time = outputs{2};
```

You can plot the performance (speedup) by typing:

```
figure
speedup = time(1,:)./time;
plot(numWorkers,speedup);
legend('Problem complexity 1','Problem complexity 2','Problem complexity 3','Problem_
↳ complexity 4','Location','northwest');
title('Speedup vs complexity');
```

(continues on next page)

(continued from previous page)

```
xlabel('Number of workers');  
xticks(numWorkers(2:end));  
ylabel('Speedup');
```

Working with Conda/Miniconda/Anaconda

Conda is an open source environment and package manager. Miniconda is a free installer for Conda, Python, and a few other useful packages. Anaconda is also a package manager that has a much larger number of packages that you can install. A question that frequently comes up is “Should I use Anaconda or Miniconda?” The Conda documentation site has a topic that can help you to decide which package manager to use: <https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html#anaconda-or-miniconda>.

13.1 Creating a Conda virtual environment with Anaconda

Using a locally installed Conda virtual environment is highly recommended so that you can install the specific packages that you need. You can also have more than one environment with different packages for testing purposes. This procedure uses the Anaconda module already loaded on Discovery.

1. If you are on a login node, move to a compute node by typing `srun --partition=short --nodes=1 --cpus-per-task=1 --pty /bin/bash`. In the above example, we request for 1 node with 1 cpu core, but you can request for additional resources as per your requirements.
2. To check what version of Python you have installed, type `which python`.
3. To load anaconda, type `module load anaconda3/2022.01`.
4. To create your environment, type `conda create -n <yourenvironmentname> python=3.7 anaconda`, where `<yourenvironmentname>` is the name you want to give your environment. Tip: to see a list of all of your conda environments, type `conda info -e`. To save space, also use the `--prefix=/work/<mygroup>/<mydirectory>` flag to build in your work directory.
5. Follow the prompts to complete the Conda install.
6. To activate your Conda environment, type `source activate <yourenvironmentname>`. Note that `conda activate` will not work on Discovery with this version.
7. To install a specific package, type `conda install -n <yourenvironmentname> [package]`.
8. To deactivate the current, active Conda environment, type `conda deactivate`.

9. To delete a Conda environment and all of its related packages, type `conda remove -n <yourenvironmentname> --all`.

13.2 Working with a Miniconda environment

This procedure assumes that you have not installed Miniconda previously. If you need to update Miniconda, don't use the installation procedure. Use the `conda update` command. This procedure uses the Miniconda3 version with Python version 3.8 in step 2, although there are other versions you can install, such as Miniconda3 with Python 3.7.

To install Miniconda:

1. If you are on a login node, move to a compute node by typing `srun --partition=short --nodes=1 --cpus-per-task=1 --pty /bin/bash`.
2. Type `wget --quiet https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh` to get the latest version of Miniconda.
3. Type `sha256sum Miniconda3-latest-Linux-x86_64.sh` to check the hash key of the package.
4. Type `bash Miniconda3-latest-Linux-x86_64.sh -b -p <dir>` to start the installation, where `<dir>` should be replaced with the full path to your desired installation directory. For example, set it to `/work/<mygroup>/<mydirectory>/miniconda3` (recommended).
5. Type `source <dir>/bin/activate` to activate the miniconda environment.
6. Another recommended step is to update your Conda version (possible only when using conda you own): `conda update conda -y`

After installing, activating and updating Miniconda, you can create a new virtual Conda environment. In the example below, the Conda environment is named "my-python38environment" and installs Python version 3.8.

1. After completing steps 1 through 6 in the previous procedure, type `conda create --name my-python38environment python=3.8`.
2. Type `y` if asked to proceed with the installation.
3. Type `conda activate my-python38environment` to activate the environment.

To deactivate the environment, type `conda deactivate`. You can type this command again to deactivate the Miniconda environment.

13.3 Conda best practices

1. Your `.conda` directory may get very large if you install multiple packages and create many virtual Conda environments. Make sure to clean the Conda cache and clean unused packages with: `conda clean --all`.
2. Clean unused Conda environments by first listing the environments with: `conda env list`, and then removing unused ones: `conda env remove --name <yourenvironmentname>`.
3. You can build Conda environments in different locations to save space on your home directory (see [Storage Accessible on Discovery](#)). You can use the `--prefix` flag when building your environment. For example: `conda create myenv --prefix=/work/<mygroup>/<mydirectory>`.

Note: It is not recommended to build your Miniconda and virtual Conda environments inside your `/home` directory due to its limited space quota (see [Storage Accessible on Discovery](#)). Use the `/work` file system instead. If your group needs access to `/work`, the group PI can request it using: [New Storage Space request](#).

Research Computing recommends using [Spack](#) to conveniently install software packages locally to your path. Please refer to the [Spack documentation](#) for the latest information about the [packages](#) that Spack contains. To use Spack, you first need to copy it to your /home directory or a /work directory, then you need to add it to your local environment.

14.1 Getting started with Spack

These instructions will demonstrate how to install Spack in your local /home directory (step 2) and then how to add Spack to your local environment while on a compute node so you have access to the Spack commands (steps 4-5).

1. Connect to Discovery via ssh ([Connecting to Discovery with a Mac](#) or [Connecting to Discovery using Windows](#)).
2. From the terminal, type `git clone -c feature.manyFiles=true https://github.com/spack/spack.git` to copy Spack to your /home directory.
3. Type `srun -p short --pty -N 1 -n 28 /bin/bash` to allocate an interactive job on a compute node. Spack will attempt to run `make` in parallel when Spack builds the software you choose to install, so this `srun` request is for 28 cores on one node (-N 1 -n 28). To use Spack, it needs to add it to your local environment on the compute node, which is why this is completed after step 3.
4. To use a newer version of python for compatibility with Spack, type: `module load python/3.8.1`.
5. To add Spack in your local environment so you can use the Spack commands, type `export SPACK_ROOT=/home/<yourusername>/spack`.
6. Next, type `. $SPACK_ROOT/share/spack/setup-env.sh`.
7. After you have the Spack commands in your environment, type `spack help` to ensure Spack is loaded in your environment and to see the commands you can use with Spack. You can also type `spack list` to see all of the software that you can install with Spack, but note this command can take a few moments to populate the list.
8. To see information about a specific software package, including options and dependencies, type `spack info <software name>`. Make sure to note the options and/or dependencies that you want to add or not add before installing the software.

9. To install a software package plus any dependencies or options, type `spack install <software name> +<any dependencies or options>;` you can specify `-<any dependencies or options>.` You can also list `+` or `-` different options and dependencies within the same line. Do not put a space between each option/dependency that you list.
10. To view your installed software packages, type `spack find`. If you want to view information about a specific installed package, type `spack find <software package name>.`

When you have installed a software package, you can add it to the module system by typing: `. $SPACK_ROOT/share/spack/setup-env.sh`

Note: Spack can be installed in a `/work` directory, which enables members of the `/work` directory to use the programs that are installed with Spack in that directory.

14.2 Installing LAMMPS with Spack example

This section details how to install the LAMMPS application with the KOKKOS and User-reaxc packages using Spack. This example assumes that you do not have any previous versions of LAMMPS installed. If you have any previous versions of LAMMPS, you must uninstall them before using this procedure. To see if you have any previous versions of LAMMPS, type `spack find lammps`. If you do have a previous version, you will need to uninstall LAMMPS by typing `spack uninstall --dependents lammps`. Then, you can try the example procedure below. Note that the installation can take about two hours to complete. As part of the procedure, we recommend that you initiate a [screen session](#) so that you can have the installation running as a separate process if you need to do other work on Discovery. If you decide to use screen, make note of the compute node number (compute node numbers start with c or d with four numbers, such as c0123) to make it easier to check on the progress of the installation.

1. Install Spack by following steps 1 through 5 in the Getting started with Spack procedure above.
2. Type `exit` to exit from the compute node you requested in step 2 above.
3. Type the following to request a GPU node for 8 hours:

```
srunk --partition=gpu --nodes=1 --ntasks=14 --pty --export=All --gres=gpu:1 --  
↪mem=0 --time=08:00:00 /bin/bash
```

4. (Optional) Initiate a screen session:
 - a. Type `screen -S lammps-install` to create a screen session.
 - b. Type `screen -ls` to check to see if the session was created (you'll see it listed if it was successfully created).
 - c. Type `screen -rd lammps-install` to enter that screen session.
 - d. Type `echo $STY` to check that you are in the screen session.
 - e. Type `CTRL+A+D` to exit the screen.

5. Type:

```
spack install lammps +asphere +body +class2 +colloid +compress +coreshell +cuda_  
↪cuda_arch=70 +cuda_mps +dipole +granular +kokkos +kspace +manybody +mc +misc_  
↪+molecule +mpio +peri +python +qeq +replica +rigid +shock +snap +spin +srd_  
↪+user-reaxc +user-misc
```

6. Type `spack find LAMMPS` to view your installed software package.
7. Type `spack load lammps`.

R is available as a module (see *Using Module* for more information) and Rstudio is also an interactive app on Open OnDemand (see *Introduction to Open OnDemand (OOD)* for more information). You can also use R with Anaconda. See *Working with Conda/Miniconda/Anaconda* and the [Anaconda documentation](#) for more information

If you work with R packages, using a Packrat environment can be helpful. Use the procedure below to create a Packrat environment on Discovery.

15.1 Creating a Packrat Environment (for R)

Packrat is an application that helps you manage packages for R. Packrat allows for greater reproducibility because it saves the exact package version that you installed, it's portable from one machine to another (or node-to-node in HPC land), and it's isolated, allowing you to create separate project directories and avoid potential package conflicts. After you create a new directory for your R project, you can then use Packrat to store your package dependencies inside it. For more information about Packrat, see the website: <https://rstudio.github.io/packrat/>.

When using R in the command line:

1. Connect to Discovery.
2. Type `module load R/4.2.1`
3. Create a new directory for your R project by typing, `mkdir /scratch/<yourusername>/<directoryname>` where `yourusername` is your user name, and `directoryname` is the name of the directory you want to create for your R project. For example, `/scratch/j.smith/packrat_r`
4. Open the R interface and install Packrat:

```
install.packages("packrat") #during the installation, you will be prompted to  
↪install in a local directory, as you cannot install as root
```

5. Initialize the environment where R packages will be written to:

```
packrat::init("/scratch/<yourusername>/<directoryname>")
```

You can then install R packages that you need. For example, to install a package called `rtracklayer`, type the following:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("rtracklayer")
```

When using Rstudio in the OOD App:

The instructions below can be applied on any RStudio “flavor” available (i.e., RStudio, Geospatial, and Tidyverse). Once a packrat snapshot is created it can easily be transferred between flavors and even machines (e.g., personal laptop, Discovery).

1. Launch an RStudio instance on the OOD. Specify the flavor and other parameters as usual.
2. In the Rstudio console type:

```
install.packages("packrat")
```

Note: This will install by default in `$HOME/R/x86_64-pc-linux-gnu-library/<version>/` as long as you don’t have previous environments or those have been turned off (see below). For packrat installation, it is best to specify a “project folder” in your `$HOME`, `/scratch` or `/work` directory (if you do not have `/work` please see [here](#) for access). The location `/tmp/Rtmp8CbQCA/downloaded_packages` would not work because `/tmp` corresponds to the compute node that you were on while running the R session. Optimally, you would like to have the packrat location in a persistent place so that all packages and libraries are available to you at all times regardless of the compute node you are on.

3. Create a packrat project directory at the desired location by selecting “New Folder” in the “Files” tab in the lower right hand side of the RStudio screen. Alternatively, you can use the command `mkdir` which can be executed in the terminal tab on the lower left hand side of the RStudio screen. For example:

```
mkdir projectfolder
```

4. In the Rstudio console, initialize the packrat. If your current directory is the project folder (i.e., `getwd() == “packrat project folder”`) you can omit the path here.

```
packrat::init("<path-to-project-folder>")
```

5. You can now record all the currently installed packages to your packrat with the `snapshot` command. This may take some time if you have installed a lot of packages.

```
packrat::snapshot()
```

6. And now you can check on the status of your packrat with:

```
packrat::status()
```

7. Now turn packrat on. Packrat will now stay on for all your Rstudio sessions and across the Rstudio flavors (Rstudio, geospatial, and tidyverse).

```
packrat::on()
```

8. You can now install packages as normal. You should see the install location for your packrat project folder. E.g., “Installing package into ‘/work/groupname/username/packrat_R/”

```
install.packages("viridis")
```

15.2 A few Packrat tips

- At any time you can check the status of your packrat with `packrat::status()`
- Packrat can be toggled on and off with `packrat::on()` and `packrat::off()` respectively.
- To disconnect packrat and allow for package installation outside of your packrat project folder: `packrat::disable(project = NULL, restart = TRUE)` Where `project` refers to the current packrat project folder, and `restart = TRUE` will restart the R session.
- To re-initialize packrat run: `packrat::init("<path-to-packrat-project-folder>")` This will automatically restart your R session.
- A package can be removed from packrat via: `remove.packages("viridis")`, but will remain in your packrat snapshot and can be restored with: `packrat::restore()`
- The function `packrat::clean(dry.run=T)` will list any unused packages that were installed in your snapshot. You can remove them with: `packrat::clean()`

Note: For most cases, having a single packrat directory is sufficient, unless you notice specific package conflicts or need different versions of the same package. A single packrat directory also saves from having to install the same dependencies multiple times in different locations.

To turn-off previously set environments

If you find the install location is not setting to your project folder you may need to turn-off these environments. In some cases, these folders could also be present in your `/work/groupname/<project-name>` directory.

```
mv ~/.rstudio ~/.rstudio-off
mv ~/.local ~/.local-off
mv ~/ondemand ~/ondemand.off
mv ~/.Rprofile ~/.Rprofile.off
mv ~/.Rhistory ~/.Rhistory.off
```

Message Passing Interface (MPI) Overview

Message Passing Interface (MPI) is a standard for writing message-passing programs. MPI itself is not a language. It defines a library interface, aimed at establishing a practical and easy-to-use standard for message passing. There are several implementations of MPI, such as [Open MPI](#) and [MVAPICH](#).

16.1 MPI libraries on Discovery

These are the current versions of Open MPI, MVAPICH, and [MPICH](#) that are available on Discovery as modules.

```
openmpi/4.1.0-zen2-gcc10.1 openmpi/4.1.0-amd-intel2021 openmpi/4.0.5-skylake-gcc7.3 openmpi/4.1.0-gcc10.1-  
cuda11.2 openmpi/4.0.4-amd-intel2020u2 openmpi/4.0.5 openmpi/4.0.5-skylake-gcc10.1 openmpi/4.0.3-cuda  
mpich/3.3.2-skylake-gcc7.3 mvapich2/2.3.4-intel2020
```

Use the `module show` command to view information about the compilers you need to use with these libraries and if they support InfiniBand (IB) or not. For example, `module show openmpi/4.1.0-zen2-gcc10.1`.

For assistance with getting started with using MPI or troubleshooting using MPI libraries on Discovery, reach out to us at rchelp@northeastern.edu or [schedule a consultation](#) with one of our team members.

CHAPTER 17

Using Slurm

Slurm (Simple Linux Utility Resource Management) is the software on Discovery that lets you do the following:

- view information about the cluster
- monitor your jobs
- schedule your jobs on Discovery
- view information about your account

Using srun and *Using sbatch* provide you with a few examples to help get you familiar with Slurm and be able to submit basic jobs on Discovery.

Important: Slurm commands have numerous options to help your jobs run efficiently by requesting specific resources. Options also usually have both short and verbose versions, such as `--nodes` and `-n` (both mean the same thing). The examples in this documentation all use the verbose version of the options for clarity, but you can use the short version if you prefer. For example, `srun -p short -N 1 -n 1` means the exact same thing as `srun --partition=short --nodes=1 --ntasks=1`. Refer to the official Slurm documentation to get in-depth information about these commands and their options: [Slurm documentation website](#).

17.1 Viewing Cluster Information

Use the following commands to view information about the cluster. This information can help you better understand the hardware that is available in order to customize your job scripts. Also see *Hardware overview* for more information.

Slurm Command	Function
<code>sinfo</code> <code><options></code>	View partition and node information. Use option <code>-a</code> to view all partitions.
<code>smap</code> <code><options></code>	View details about the cluster in a visual format

17.2 Submitting Jobs

There are two main commands for submitting jobs to Discovery: `srun` and `sbatch`. To run a job interactively, use `srun`. To submit a job to run in the background with a script, use `sbatch`.

Slurm Command	Function
<code>srun</code>	Run an interactive job on the cluster. See Using srun
<code>sbatch</code> <code><scriptname></code> <code>script</code>	Submit a script to the scheduler for running a job. See Using sbatch
<code>scancel</code> <code><jobid></code>	Cancel a pending or running job on the cluster

17.3 Monitoring Jobs

Slurm Command	Function
<code>seff</code> <code><jobid></code>	Reports the computational efficiency of your calculations.
<code>squeue -u</code> <code><your user name></code>	Displays your job status in the job queue. Good to use with <code>sbatch</code> .
<code>scontrol</code> <code>show jobid</code> <code>-d <JOBID></code>	Displays your job information. Good to use with <code>srun</code> .

17.4 Account information

Some Discovery users have more than one Discovery group account associated with their username. For example, a student might be in a class that is using Discovery, and also be in a student club that is using Discovery for a club project. In this case, the student would have two group accounts associated with their username. When running a job with either `srun` or `sbatch`, if you have more than one account associated with your username, we recommend that you use the `--account=` flag and specify the account that corresponds to the project you are working on. In the example with a student associated with a class and a student club, if the student is on Discovery submitting a job for a project for his or her class, set the `account=` flag to the name of the class account. If the student is working on a project for the club, set the `account=` flag to the name of the student club account.

To find out what account(s) your username is associated with, use the following command:

```
sacctmgr show associations user=<yourusername>
```

After you have determined what accounts your username is associated with, if you have more than one account association, you can use the `account=` flag with your usual `srun` or `sbatch` commands.

For example, if you are associated with an account named `dataclub` and an account named `info7500`, and you're currently doing work that should be associated with the `dataclub` account, in your `srun` command, you can add the `--account=dataclub` flag to specify that account.:

```
srun --account=dataclub --partition=short --nodes=1 --ntasks=28 --mem=0 --pty /bin/  
↵bash
```

Note: If you do not have more than one account associated with your username, you do not need to use the `--account=` flag. Most users on Discovery have only one account associated with their username.

Using sbatch

You use the `sbatch` command with a bash script to specify the resources you need to run your jobs, such as the number of nodes you want to run your jobs on and how much memory you'll need. Slurm then schedules your job based on the availability of the resources you've specified. The general format for submitting a job to the scheduler is as follows:

```
sbatch example.script
```

Where `example.script` is a script detailing the parameters of the job you want to run.

Note: The default time limit depends on the partition that you specify in your submission script using the `--partition=<partition name>` option. If your job does not complete within the requested time limit, Slurm will automatically terminate the job. See [Partitions](#) for the most up-to-date partition names and parameters.

18.1 SBATCH Examples

18.1.1 Job requesting one node

Run a job on one node for 4 hours on the short partition:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --partition=short
<commands to execute>
```

18.1.2 Job requesting one node and additional memory

The default memory per allocated core is 1GB. If your calculations try to use more memory than what is allocated, Slurm automatically terminates your job. You should request a specific amount of memory in your job script if your calculations need more memory than the default. The example script below is requesting 100GB of memory (`--mem=100G`). Use one capital letter to abbreviate the unit of memory (K,M,G,T) with the `--mem=` option, as that is what Slurm expects to see.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100G
#SBATCH --partition=short
<commands to execute>
```

18.1.3 Job requesting one node with exclusive use of a node

If you need exclusive use of a node, such as when you have a job that has high I/O requirements, you can use the exclusive flag. The example script below specifies exclusive use of 1 node in the short partition for four hours.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --exclusive
#SBATCH --partition=short
<commands to execute>
```

18.2 Example Parallel Job Scripts

Parallel jobs should be used with code that is configured to use the reserved resources. If your code is not optimized for running in parallel, your job could fail. The following script examples all allocate additional memory. The default memory per allocated core is 1GB. If your calculations try to use more memory than what is allocated, Slurm automatically terminates your job. You should request a specific amount of memory in your job script if your calculations need more memory than the default.

18.2.1 8-task job, one node and additional memory

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=1
#SBATCH --time=4:00:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100G
#SBATCH --partition=short
<commands to execute>
```

18.2.2 8-task job, multiple nodes and additional memory

```
#!/bin/bash
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=1
#SBATCH --time=00:30:00
#SBATCH --job-name=MyJobName
#SBATCH --mem=100G
#SBATCH --partition=express
<commands to execute>
```

18.3 Using Arrays

Using a job array can often help in situations where you need to submit multiple similar jobs. To use an array with your jobs, in your `sbatch` script, use the `array=` option.

For example, if you want to run a 10 job array, one job at a time, you would add the following line to your `sbatch` script:

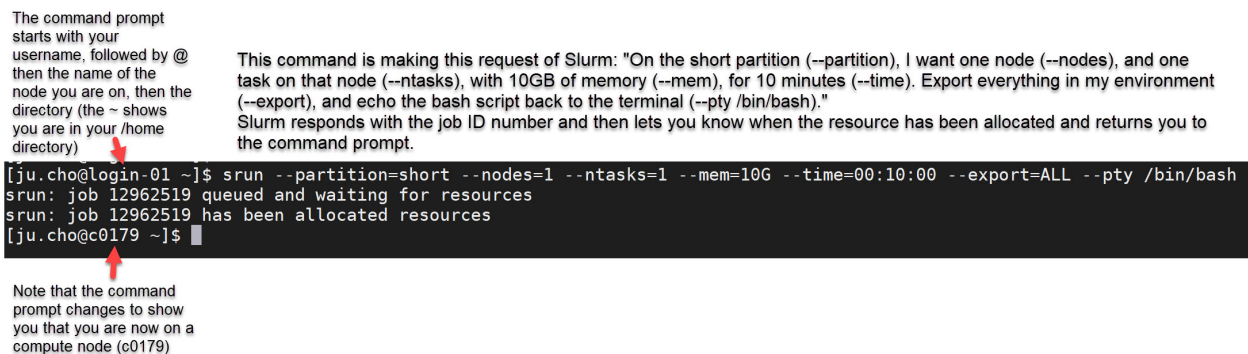
```
#SBATCH --array=1-10%1
```

For more information on this command, go to the [Slurm documentation page](#).

Using srun

You can use the Slurm command `srun` to allocate an interactive job. This means you use specific options with `srun` on the command line to tell Slurm what resources you need to run your job, such as number of nodes, amount of memory, and amount of time. After typing your `srun` command and options on the command line and pressing enter, Slurm will find and then allocate the resources you specified. Depending on what you specified, it can take a few minutes for Slurm to allocate those resources. You can view all of the `srun` options on the [Slurm documentation website](#).

The following image shows an example of an `srun` command as run on a command line.



19.1 srun examples

This section details a few examples using `srun`. You should first review the [Hardware overview](#) and [Partitions](#) sections to be familiar with the available hardware and partition limits on Discovery. This way, you can tailor your request to fit both the needs of your job and the limits of the partitions. For example, if you specify `--partition=debug` and `--time=01:00:00`, you'll get an error because the time you've specified exceeds the limit for that partition. Also keep in mind that while these examples are all valid, general examples, they might not work for your particular job.

simple `srun` example is to move to a compute node after you first log into Discovery.

```
srun --pty /bin/bash
```

To request one node and one task for 30 minutes with X11 forwarding on the short partition, type:

```
srun --partition=short --export=ALL --nodes=1 --ntasks=1 --x11 --mem=10G --  
↪time=00:30:00 --pty /bin/bash
```

To request one node, with 10 tasks and 2 CPUs per task (a total of 20 CPUs), 1GB of memory, for one hour on the express partition, type:

```
srun --partition=express --nodes 1 --ntasks 10 --cpus-per-task 2 --pty --export=ALL -  
↪-mem=1G --time=01:00:00 /bin/bash
```

To request two nodes, each with 10 tasks per node and 2 CPUs per task (a total of 40 CPUs), 1GB of memory, for one hour on the express partition, type:

```
srun --partition=express --nodes=2 --ntasks 10 --cpus-per-task 2 --pty --export=ALL -  
↪-mem=1G --time=01:00:00 /bin/bash
```

To allocate a GPU node, you should specify the `gpu` partition and use the `-gres` option:

```
srun --partition=gpu --nodes=1 --ntasks=1 --export=ALL --gres=gpu:1 --mem=1Gb --  
↪time=01:00:00 --pty /bin/bash
```

For more information about working with GPUs, see [Working with GPUs](#).

19.1.1 Monitor your jobs

You can monitor your jobs by using the Slurm `scontrol` command. Type `scontrol show jobid -d <JOBID>`, where `JOBID` is the number of your job. In the figure at the top of the page, you can see that when you submit your `srun` command, Slurm displays the unique ID number of your job (job 12962519). This is the number you use with `scontrol` to monitor your job.

CHAPTER 20

Working with GPUs

The Discovery cluster has a number of NVIDIA Graphics Processing Units (GPUs) available, as detailed in the table below.

Note: The tables on this page slide from left-to-right. Make sure to swipe to right to see the content on the right side of the table

GPU Type	GPU Memory	Tensor Cores	CUDA Cores	Nodes in Public GPUs	Nodes in Private GPUs
p100 (Pascal)	12GB	N/A	3,584	12 with 3-4 GPUs each	3 with 4 GPUs each
v100-pcie (Volta)	32GB	640	5,120	4 with 2 GPUs each	1 with (16GB) 2 GPUs
v100-sxm2 (Volta)	32GB	640	5,120	24 with 4 GPUs each	10 with 4 GPUs each & 16GB GPU memory; 8 with 4 GPUs & 32GB GPU memory
t4 (Turing)	15GB	320	2,560	2 with 3-4 GPUs each	1 with 4 GPUs
quadro (Quadro RTX 8000)	46GB	576	4,608	0	2 with 3 GPUs each
a30 (Ampere)	24GB	224	3,804	0	1 with 3 GPUs
a100 (Ampere)	41 & 82GB	432	6,912	3 nodes with 4 GPUs each	15 nodes with 2-8 GPUs each
a5000 (Ampere RTX A5000)	24GB	256	8,192	0	6 with 8 GPUs each
a6000 (Ampere RTX A6000)	49GB	336	10,752	0	3 with 8 GPUs each

The public GPUs are available within two partitions, named `gpu` and `multigpu`. The differences between the two partitions are the number of GPUs that one can request per job and the time limit on each job. Both partitions give access to all of the public GPU types mentioned above. The table below shows the differences between the two partitions. For more information about the partitions on Discovery, see [Partitions](#).

Note: All user limits are subject to the availability of cluster resources at the time of submission and will be honored according to that.

Name	Requires Approval?	Time limit (Default/Max)	Submitted Jobs	GPU per job Limit	Max GPUs per user Limit
<code>gpu</code>	No	4 hours/8 Hours	50/100	1	8
<code>multigpu</code>	Yes	4 hours/24 Hours	50/100	12	12

Anyone with a Discovery account can use the `gpu` partition. However, you must submit a [ServiceNow ticket](#) to request temporary access to `multigpu` for testing, or to request full access to the `multigpu` partition. Your request will be evaluated by members of the RC team to ensure that the resources in this partition will be used appropriately.

20.1 Requesting GPUs with `srun` or `sbatch`

Use `srun` for interactive and `sbatch` for batch mode. The `srun` example below is requesting 1 node and 1 GPU with 4GB of memory in the `gpu` partition. You must use the `--gres=` option to request a GPU:

```
srun --partition=gpu --nodes=1 --pty --gres=gpu:1 --ntasks=1 --mem=4GB --
↪time=01:00:00 /bin/bash
```

Note: On the `gpu` partition, requesting more than 1 GPU (`--gres=gpu:1`) will cause your request to fail. Additionally, one cannot request all the CPUs on that `gpu` node as they are reserved for other GPUs.

The `sbatch` example below is similar to the `srun` example above, but it submits the job in the background, gives it a name, and directs the output to a file:

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --gres=gpu:1
#SBATCH --time=01:00:00
#SBATCH --job-name=gpu_run
#SBATCH --mem=4GB
#SBATCH --ntasks=1
#SBATCH --output=myjob.%j.out
#SBATCH --error=myjob.%j.err
<your code>
```

20.1.1 Specifying a GPU type

You can add a specific type of GPU to the `--gres=` option (with either `srun` or `sbatch`). For a list of available GPU types, refer to the GPU Types column in the table, at the top of this page, that are listed as `Public`. The following is an example for requesting a single `p100` GPU:

```
--gres=gpu:p100:1
```

Note: Requesting a specific type of GPU could result in longer wait times, based on GPU availability at that time.

20.2 Using CUDA

There are several versions of CUDA Toolkits on Discovery, including:

```
cuda/9.0
cuda/9.2
cuda/10.0
cuda/10.2
cuda/11.0
cuda/11.1
cuda/11.2
cuda/11.3
cuda/11.4
cuda/11.7
cuda/11.8
cuda/12.1
```

Use the `module avail` command to check for the latest software versions on Discovery. To see details on a specific CUDA toolkit version, use `module show`. For example, `module show cuda/11.4`.

To add CUDA to your path, use `module load`. For example, type `module load cuda/11.4` to load version 11.4 to your path.

Use the command `nvidia-smi` (NVIDIA System Management Interface) inside a GPU node to get the CUDA driver information and monitor the GPU device.

20.3 Using GPUs with PyTorch

You should use PyTorch with a conda virtual environment if you need to run the environment on the Nvidia GPUs on Discovery. The following example demonstrates how to build PyTorch inside a conda virtual environment for CUDA version 11.7.

Note: Make sure to be on a GPU node before loading the environment. Additionally, the latest version of PyTorch is not compatible with GPUs with CUDA version 11.7 or less. Hence, the installation does not work on k40m or k80 GPU's. In order to see what non-Kepler GPUs might be available, one can execute this command:

```
sinfo -p gpu --Format=nodes,cpus,memory,features,statecompact,nodelist,gres
```

This will indicate the state (idle or not) of a certain gpu-type that could be helpful in requesting an idle gpu. However, the command does not give real-time information of the state and should be used with caution.

PyTorch installation steps (with a specific GPU-type):

```

srun --partition=gpu --nodes=1 --gres=gpu:v100-sxm2:1 --cpus-per-task=2 --mem=10GB --
↪time=02:00:00 --pty /bin/bash
module load anaconda3/2022.05 cuda/11.7
conda create --name pytorch_env python=3.9 -y
source activate pytorch_env
conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia -y
python -c'import torch; print(torch.cuda.is_available())'

```

Note: If the installation times out, please ensure that your `.condarc` file doesn't contain additional channels. Also, consider cleaning your conda instance using the `conda clean` command. See [Conda best practices](#).

If CUDA is detected by PyTorch, you should see the result, `True`.

As the latest version of PyTorch often depends on the newest CUDA available, please refer to the [PyTorch documentation page](#) for the most up to date instructions on installation.

The above PyTorch installation instructions will not include `jupyterlab` and few other commonly used datascience packages in the environment. In order to include those one can execute the following command after activating the `pytorch_env` environment:

```
conda install pandas scikit-learn matplotlib seaborn jupyterlab -y
```

20.4 Using GPUs with TensorFlow

We recommend that you use CUDA 11.2 (latest supported version) when working on a GPU with the latest version of TensorFlow (TF). TensorFlow provides information on the [compatibility of CUDA and TensorFlow versions](#), and [detailed installation instructions](#).

For the latest installation, use the TensorFlow pip package, which includes GPU support for CUDA-enabled devices:

```

srun --partition=gpu --gres=gpu:1 --nodes=1 --cpus-per-task=2 --mem=10GB --
↪time=02:00:00 --pty /bin/bash
module load anaconda3/2022.05 cuda/11.2
conda create --name TF_env python=3.9 -y
source activate TF_env
conda install -c conda-forge cudatoolkit=11.2.2 cudnn=8.1.0 -y
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/
mkdir -p $CONDA_PREFIX/etc/conda/activate.d
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CONDA_PREFIX/lib/' > $CONDA_PREFIX/etc/
↪conda/activate.d/env_vars.sh
pip install --upgrade pip
pip install tensorflow==2.11.*

```

Verify the installation:

```

# Verify the CPU setup (if successful, then a tensor is returned):
python3 -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000,
↪1000])))"

# verify the GPU setup (if successful, then a list of GPU device is returned):
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"

# test if a GPU device is detected with TF (if successful, then True is returned):
python3 -c 'import tensorflow as tf; print(tf.test.is_built_with_cuda())'

```

To get the name of the GPU, type:

```
python -c 'import tensorflow as tf; print(tf.test.gpu_device_name())'
```

If the installation is successful, then, for example, you should see the following as an output,:

```
2023-02-24 16:39:35.798186: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1613]
↳ Created device /device:GPU:0 with 10785 MB memory: -> device: 0, name: Tesla K80,
↳ pci bus id: 0000:0a:00.0, compute capability: 3.7 /device:GPU:0
```

Note: Ignore the Warning messages that get generated after executing the above commands.

test

CHAPTER 21

Transferring Data

Discovery has a dedicated transfer node that you must use to transfer data to and from the cluster. You are not allowed to transfer data from any other node. The node name is `<username>@xfer.discovery.neu.edu`: where `<username>` is your Northeastern username.

You can also transfer files using Globus. This is highly recommended if you need to transfer large amounts of data. See *Using Globus* for more information.

Caution: The `/scratch` directory is for temporary file storage only. It is not backed up. If you have directed your output files to `/scratch`, you should transfer your data from `/scratch` to another location as soon as possible. See *Storage Accessible on Discovery* for more information.

21.1 Data transfer node using Mac

You can use Terminal to transfer data to and from Discovery.

For example, you can use this command to transfer a file to your `/scratch` space:

```
scp <filename> <yourusername>@xfer.discovery.neu.edu:/scratch/<yourusername>
```

Where `<filename>` is the name of the file you want to transfer and `<yourusername>` is your Northeastern username.

21.1.1 Using SSHFS

If you want to use `sshfs`, you will need to use it with the dedicated transfer node `xfer.discovery.neu.edu`. It will not work on the login or compute nodes.

Use this syntax to perform file transfers with `sshfs`:

```
sshfs <yourusername>@xfer.discovery.neu.edu:</your/remote/path> <your/local/path> -  
↪<options>
```

21.2 Data transfer node using Windows

You can use MobaXterm to transfer data to and from Discovery. You can also use a file transfer program, such as FileZilla. The Files menu in Open onDemand (OOD) also allows you to transfer files. See *Using OOD's File Explorer* for more information.

To transfer data using MobaXterm:

1. Open MobaXterm.
2. Click **Session**, then select **SFTP**.
3. In the **Remote host** field, type `xfer.discovery.neu.edu`
4. In the **Username** field, type your Northeastern username.
5. In the **Port** field, type 22.
6. In the **Password** box, type your Northeastern password and click **OK**. Click **No** if prompted to save your password.

You will now be connected to the transfer node and can transfer files through MobaXterm.

To transfer data using FileZilla:

1. Open FileZilla.
2. In the **Host** field, type `sftp://xfer.discovery.neu.edu`
3. In the **Username** field, type your Northeastern username.
4. In the **Password** field, type your Northeastern password.
5. In the **Port** field, type 22.

You will now be connected to the transfer node and can transfer files through FileZilla.

Globus is a data management system that you can use to transfer and share files. Northeastern has a subscription to Globus, and you can setup a Globus account with your Northeastern credentials. If you have another account, either personal or through another institution, you can also link your accounts.

To use Globus, you must first set up an account as detailed below. Then, you must install Globus Connect to create an endpoint on your local computer, as also detailed below. After you have completed these two initial setup procedures, you can then use the Globus web app to perform file transfers. See [:ref: using_nuendpoint](#) for a walkthrough of using the Northeastern endpoint on Globus.

22.1 Globus Account Set Up

Use the following instructions to setup an account with Globus using your Northeastern credentials.

1. Go to [Globus](#).
2. Click **Log In**.
3. From the Use your existing organizational login, select **Northeastern University**, and then click **Continue**.
4. Enter your Northeastern username and password.
5. If you do not have a previous Globus account, click **Continue**. If you have a previous account, click Link to existing account.
6. Check the agreement checkbox, and then click **Continue**.
7. Click **Allow** to give Globus permissions to access your files.

You will then be able to access the [Globus File Manager](#) app.

Tip: If you received an account identity that includes your NUID number (for example [000123456@northeastern.edu](#)), you can follow the “Creating and linking a new account identity” instructions below to get a different account identity if you want a more user-friendly account identity. You can then link the two accounts together.

22.1.1 Creating and linking a new account identity (Optional)

If you created an account through the Northeastern University existing organizational login and received a username that includes your NUID, you can create a new identity with a different username and then link the two accounts together. A username that you have selected, as opposed to one with your NUID, can make it easier for you to remember your login credentials.

1. Go to [Globus](#).
2. Click **Log In**.
3. Click **Globus ID** to sign in.
4. Click **Need a Globus ID? Sign up**.
5. Enter your Globus ID information.
6. Enter the verification code that Globus sends to your email.
7. Click **Link to an existing account** to link this new account with your primary account.
8. Select Northeastern University from the dropdown box, and click **Continue** to be taken to the Northeastern University single sign on page.
9. Enter your Northeastern username and password.

You should now be able to see your two accounts linked in the Account section on the [Globus web app](#).

22.2 Install Globus Connect Personal (GCP)

Use Globus Connect Personal (GCP) to use your personal laptop as an endpoint. You first need to install GCP using the following procedure. You need to be logged into Globus before you can install GCP.

1. Go to [Globus File Manager](#).
2. Enter a name for your endpoint in the Endpoint Display Name field.
3. Click **Generate Setup Key** to generate a setup key for your endpoint.
4. Click the **Copy** icon next to the setup key that was generated to copy the key to your clipboard. You will need this key during the installation of GCP in step 6.
5. Click the appropriate OS icon for your computer to download the installation file.
6. After the installation file has been downloaded to your computer, double click on the file to launch the installer.

Accept the defaults on the install wizard. After the install completes, you can now use your laptop as an endpoint within Globus.

Note: You can't modify an endpoint after you have created it. If you need an endpoint with different options, you'll need to completely delete the endpoint and recreate it. Follow the instructions on the Globus website for [deleting and recreating an endpoint](#).

22.3 Working with Globus

After you have an account and set up a personal endpoint using Globus Connect personal, you can perform basic file management tasks using the Globus File Manager interface such as transferring files, renaming files, and creating new

folders. You can also download and use the Globus Command Line Interface (CLI) tool. Globus also has extensive documentation and training files for you to practice with.

22.3.1 Using the Northeastern endpoint

To access the Northeastern endpoint on Globus, on the Globus web app, click **File Manager**, then in the **Collection** text box, type Northeastern. The endpoints owned by Northeastern University display in the collection area. The general Northeastern endpoint is `northeastern#discovery`. Using the File Manager interface, you can easily change directories, switch the direction of transferring to and from, and specify options such as transferring only new or changed files. Below is a procedure for transferring files from Discovery to your personal computer, but with the flexibility of the File Manager interface, you can adjust the endpoints, file view, direction of the transfer, and many other options.

To transfer files from Discovery to your personal computer, do the following

1. Create an endpoint on your computer using the procedure above “Install Globus Connect”, if you have not done so already.
2. In the File Manager on the Globus web app, in the **Collections** textbox, type Northeastern, then in the collection list click the `northeastern#discovery` endpoint.
3. In the right-pane menu, click **Transfer or Sync to**.
4. Click in the **Search** text box, and then in on the **Your Collections** tab, click the name of your personal endpoint. You now can see the list of your files on Discovery on the left and a list of your files on your personal computer on the right.
5. Select the file or files from the right-side list of Discovery files that you want to transfer to your personal computer.
6. Select the destination folder from the left-side list of the files on your personal computer.
7. (Optional) Click **Transfer & Sync Options** and select the transfer options that you need.
8. Click **Start**.

22.3.2 Connecting to Google Drive

The version of Globus currently on Discovery allows you to connect to Google Drive by first setting up the connection in GCP. This will add your Google Drive to your current personal endpoint. You’ll need to first have a personal endpoint, as outlined in the procedure above. This procedure is slightly different from using the Google Drive Connector with Globus version 5.5. You’ll need your Google Drive [downloaded to your local computer](#).

To add Google Drive to your personal endpoint, do the following

1. Open the GCP app. On Windows, right click on the **G** icon in your taskbar and select **Options**. On Mac, click the **G** icon in the menu bar and select **Preferences**.
2. On the **Access** tab, click the + button to open the **Choose a directory** dialog box.
3. Navigate to your Google Drive on your computer and click **Choose**.
4. Click the **Shareable** checkbox to make this a shareable folder in Globus File Manager, and then click **Save**.

You can now go to Globus File Manager and see that your Google Drive is available as a folder on your personal endpoint.

22.3.3 Command Line Interface (CLI)

The Globus Command Line Interface (CLI) tool allows you to access Globus from a command line. It is a stand-alone app that requires a separate download and installation. Please refer to the [Globus CLI documentation](#) for working with this app.

22.3.4 Globus documentation and test files

Globus provides detailed instructions on using Globus and also has test files for you to practice with. These are free for you to access and use. We encourage you to use the test files to become familiar with the Globus interface. You can access the Globus documentation and training files on the [Globus How To website](#).

Storage Accessible on Discovery

There are two main storage systems connected to Discovery: `/home` and `/scratch`. These options have specific quotas and limitations. The list below details the storage options available to you on Discovery if you have an account on Discovery. These are storage options that are connected to Discovery, and you should use when working on Discovery. Every individual with an account on Discovery has both a `/home` and `/scratch` directory. Research groups can request additional storage on the `/work` storage system. Note that currently `/work` storage is not provisioned to individuals.

Important: The `/scratch` space is only for temporary storage. It is not backed up, and there is a purge policy for data older than 28 days on `/scratch`. Please review the `/scratch` policy on our Policy page: <https://rc.northeastern.edu/policy/>

NAME: `/home/<yourusername>` where `yourusername` is your username, e.g. `/home/j.smith`

- **DESCRIPTION:** You are given a `/home` directory automatically when your Discovery account is created. This storage is mainly intended for storing relatively small files such as script files, source code, software installation files, and other small files that you need for your work on Discovery. While it is permanent storage that is backed up and replicated, it is not performant storage. It also has a small quota, so you should frequently check your space usage (use a command such as `du -h /home/<yourusername>` where `<yourusername>` is your user name, to see the total space usage). For running jobs and directing output files, you should use your `/scratch` directory.
- **QUOTA:** 75GB

NAME: `/scratch/<yourusername>`

- **DESCRIPTION:** You are given a `/scratch` directory automatically when your Discovery account is created. Scratch is a shared space for all users. The total storage available is 1.8PB; however, while this is performant storage, it is for temporary use only. **It is not backed up.** Data on `/scratch` should be moved as soon as possible to another location for permanent storage. You should run your jobs from `/scratch` and direct your output files to your `/scratch` directory for best performance, but it is best practice to move your files off of scratch to avoid any potential data loss.
- **QUOTA:** N/A

NAME: `/work/<groupname>`

- **DESCRIPTION:** Research groups can request additional storage on `/work`. A PI can request this extra storage through the [New Storage Space request](#) . This is a performant, persistent, and long-term storage that is meant for storing data being actively used for research. It can be accessed by all members of the research group who have access permissions to this directory.

Note: Your group can also request additional general data storage if needed. See [:ref:general_storage](#) for details about the storage options that are not associated with Discovery but are available to anyone affiliated with Northeastern University. Each group can request up to 35TB of free storage across all supplemental storage tiers: `/work`, `/research` and `/nese`.

General Data Storage Options

The Research Computing (RC) team is responsible for the procurement and ongoing maintenance of several data storage options, including active and archive storage solutions. If you are affiliated with Northeastern, you can request one or more storage solutions to meet your storage needs. It is highly recommended that if you anticipate needing storage as part of a grant requirement, schedule a consultation with a Research Computing staff member to understand what storage options would best meet your research needs. You can schedule an online consultation on the [RC website](#).

Note: If you have an account on Discovery, see [Storage Accessible on Discovery](#) for details on the storage available to you specifically for use with Discovery’s compute resources. The options listed below are not connected to Discovery.

NAME: `/research`

- **DESCRIPTION:** This storage tier is intended to be a repository for data derived from equipment such as lab machines, instruments, etc. The performance capabilities are not intended for parallel or high performance workloads. Data are backed up and a second copy is created. You can request storage on `/research` by submitting a [New Storage Space request](#). See the section “Connecting to `/research`” below for information on how to connect to this storage.
- **QUOTA:** Each group can request up to **35TB** of free storage across all supplemental storage tiers: `/work/<groupname>`, `/research()` and `/nese`.

NAME: `/nese`

- **DESCRIPTION:** This is archival, non-performant storage that is intended for researchers who need to have a long-term storage option for their data.
- **QUOTA:** Each group can request up to **35TB** of free storage across all supplemental storage tiers: `/work/<groupname>`, `/research()` and `/nese`.

Important: If you are not connected to the campus internet, you must be connected to the university’s VPN (GlobalProtect) before you can access these storage systems. You can find detailed information about downloading and using the GlobalProtect VPN in the [ServiceNow Knowledge Base](#).

24.1 Connecting to /research

After you have received notification that your shared storage folder on /research is ready for you to use, you can connect to it on your laptop. Use the following procedures to connect to /research from either a Windows PC or a Macbook. If you are not on the campus internet, you will need to first be connected to Northeastern's VPN before you can connect to /research.

Windows

1. Open **File Explorer**.
2. In the left pane, select **This PC**.
3. On the **Computer** tab, select **Map network drive**.
4. Select any letter not currently mapped to a drive. It does not matter what letter.
5. In the **Folder** box, type `\\nunet.neu.edu\rc-shares\<sharename>`, where `<sharename>` is the name of your shared storage. If you want this drive to reconnect automatically every time you log in, check **Reconnect at sign in**.
6. Click **Finish**. Your shared folder now be available in File Explorer.

Mac

1. Click the **Finder** icon, and in the **Finder** window, select **Go**, then click **Connect to Server**.
2. In the **Server Address** field type `smb://nunet.neu.edu/rc-shares/<sharename>` where `<sharename>` is the name of the share you want to connect to.
3. Click **Connect**.
4. If you are prompted to enter your name and password for the nunet server, select **Connect as Registered User**, in the **Name** field type `NUNET\<yourusername>`, (where `<yourusername>` is your NU username) and type your NU password in the **Password** field. If you want to connect to /research automatically when you log in to your computer, select **Remember this password in my keychain**.
5. Click **Connect**. Your share should appear as a folder in the Finder window.

Checkpoint/Restart Discovery Jobs

The complexity of HPC systems may introduce unpredictable behaviors and may result in job failures due to hardware or software. Applying fault tolerance techniques to your HPC workflows allows your jobs to become more resilient to crashes, partition time limits and hardware failures.

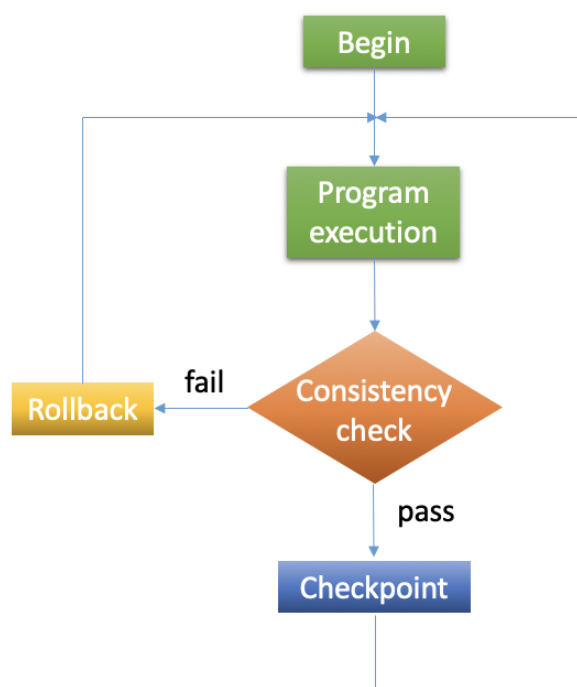
Checkpointing will allow you to:

- Create resilient workflows in the existence of faults.
- Overcome most scheduler resource time limitations.
- Implement an early error detection approach by inspecting intermediate results.

25.1 The Checkpointing technique

Checkpointing is a fault tolerance technique designed to overcome the “fail-stop” failure type (interruption of the execution of a job). It is based on the BER technique (Backward error recovery or Rollback-recovery algorithm):

- Use data redundancy - create checkpoint files saving all of the necessary calculation state data. Checkpoint files are generally created at constant time intervals during the run.
- If a failure occurs - start from an error-free state, check for consistency and restore the algorithm to the previous error-free state.



25.2 Checkpointing types

Checkpointing can be implemented in different levels of your workflow:

- User-level checkpointing - suitable if you develop your own code, or have sufficient knowledge of the application code to integrate checkpointing techniques yourself. Generally, this approach is not recommended for most Discovery users.
- Application-level checkpointing - recommended for most Discovery users. Utilize the checkpointing tool that is already available in your software application. Most software designed for HPC have a checkpointing option, and information on proper usage is often available in the software user manual.
- System-level checkpointing - done on the system side, where the state of the entire process is being saved. This option is less efficient than User-level or Application-level checkpointing as it introduces a lot of redundancy.

Note: If you are developing code using Python, Matlab or R, there are packages and functions that can be used to implement checkpointing easily. Some examples include [Python PyTorch checkpointing](#), [TensorFlow checkpointing](#), [Python Pickle checkpointing](#), [MATLAB checkpointing](#) and [R checkpointing](#). Additionally, many Computational Chemistry and Molecular Dynamics software have built-in checkpointing options, such as [GROMACS](#) and [LAMMPS](#).

Implementing checkpointing can be achieved by:

- Some save-and-load mechanism of your calculation state.
- The use of [Slurm Job Arrays](#).

Note: To overcome partition time limits, replace your single long job with multiple shorter jobs. Using job arrays, set each job to run one after the other. Each job will write a checkpoint file if checkpointing is implemented. The next

job in line will be the latest checkpoint file to continue from the latest state of the calculation.

25.2.1 GROMACS checkpointing example

This example demonstrates how to implement a longer [GROMACS](#) job of 120 hours by using multiple shorter jobs on the **short** partition. We use Slurm job arrays and the GROMACS built-in checkpointing option (read more [here](#)) to implement checkpointing.

The following script **submit_mdrun_array.bash** creates a Slurm job array of 10 individual array jobs:

```
#!/bin/bash
#SBATCH --partition=short
#SBATCH --constraint=cascadelake
#SBATCH --nodes=1
#SBATCH --time=12:00:00
#SBATCH --job-name=myrun
#SBATCH --ntasks=56
#SBATCH --array=1-10%1 #execute 10 array jobs, 1 at a time.
#SBATCH --output=myrun-%A_%a.out
#SBATCH --error=myrun-%A_%a.err

module load cuda/10.2
module load gcc/7.3.0
module load openmpi/4.0.5-skylake-gcc7.3
module load gromacs/2020.3-gpu-mpi
source /shared/centos7/gromacs/2020.3-gcc7.3/bin/GMXRC.bash

srun --mpi=pmi2 -n $SLURM_NTASKS gmx_mpi mdrun -ntomp 1 -s myrun.tpr -v -dlb yes -cpi_
↪state
```

In the above script, we use the checkpoint flag `-cpi state` followed by the file name to be used for checkpointing. This directs `mdrun` to use the checkpoint file named `state.cpt` when loading the state. The Slurm option `--array=1-10%1` will create 10 Slurm array tasks, and will run one task job at a time for 12 hours. Note that the saved variable `%A` denotes the main job ID, while variable `%a` denotes the task ID (spanning values 1-10).

To submit this array job to the scheduler, use the following command:

```
sbatch submit_mdrun_array.bash
```

25.2.2 Python TensorFlow checkpointing example

This example demonstrates how to implement a longer TensorFlow ML job by training using the **tf.keras** checkpointing [API](#) and multiple shorter Slurm job arrays on the **gpu** partition. Below the example **submit_tf_array.bash** script:

```
#!/bin/bash
#SBATCH --job-name=myrun
#SBATCH --time=00:10:00
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --gres=gpu:1
#SBATCH --mem=10Gb
#SBATCH --output=%A-%a.out
#SBATCH --error=%A-%a.err
```

(continues on next page)

(continued from previous page)

```
#SBATCH --array=1-10%1 #execute 10 array jobs, 1 at a time.

module load miniconda3/2020-09
source activate tf_gpu

##Define the number of steps based on the job id:
numOfSteps=$(( 500 * SLURM_ARRAY_TASK_ID ))

# run the python code, save all output to a log file corresponding the the current_
↪ job task that is running:
python train_with_checkpoints.py $numOfSteps &> log.$SLURM_ARRAY_TASK_ID
```

Where the checkpointing implementation is given in this code snippet of `train_with_checkpoints.py`:

```
checkpoint_path = "training_2/{epoch:d}.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,
    verbose=1,
    save_weights_only=True,
    period=5)
```

The full scripts can be found [here](#) and were modified from [TensorFlow Save and load models](#).

The Slurm option `--array=1-10%1` will create 10 Slurm array tasks, and will run one task job at a time. Note that the saved variable `%A` denotes the main job ID, while variable `%a` denotes the task ID (spanning values 1-10). Note that also the output/error files are unique in order to prevent different jobs writing to the same files. The Shell variable `SLURM_ARRAY_TASK_ID` holds the unique task ID value and can be used within the Slurm Shell script to point to different files or variables.

To submit this job to the scheduler, use the command:

```
sbatch submit_tf_array.bash
```

25.2.3 Checkpointing using DMTCP

DMTCP (Distributed MultiThreaded checkpointing) is a checkpointing tool that lets you checkpoint without the need to change your code. It Works with most Linux applications such as Python, Matlab, R, GUI, MPI etc. The program runs in the background of your program, without significant performance loss, and saves the process states into checkpoint files. DMTCP is available on the cluster

```
module avail dmtcp
module show dmtcp
module load dmtcp/2.6.0
```

As DMTCP runs in the background, it requires some changes to your Shell script. For examples of how to checkpoint with DMTCP visit [here](#). The example demonstrates how to use DMTCP with a simple C++ program (scripts modified from [RSE-Cambridge](#)).

25.2.4 Checkpointing tips

What data to save?

- Non-temporary application data

- Any application data that has been modified since the last checkpoint
- Delete checkpoints that are no longer useful - keep only the most recent checkpoint file.

How frequently to checkpoint?

- Too often – will slow down your calculation, may be I/O heavy and memory-limited.
- Too infrequently – leads to large/long rollback times.
- Consider how long it takes to checkpoint and restart your calculation.
- In most cases a rate of every 10-15 minutes is ok.

Which checkpointing method to use?

- If your software already comes with built-in checkpointing, it is often the preferred option. It is probably the most optimized and efficient way to checkpoint.
- Application-level checkpointing is the easiest to use as it is already integrated in your application. Does not require major changes to your scripts.
- Application-level checkpointing will save only the relevant data for your specific application.
- If you're writing your own code - use DMTCP or implement your own checkpointing.

Introduction to Open OnDemand (OOD)

Open OnDemand (OOD) is a web portal to the Discovery cluster. A Discovery account is necessary for you to access OOD. If you need an account, see *Request an account*. If you already have an account, in a web browser, go to <http://ood.discovery.neu.edu> and sign in with your Northeastern username and password.

OOD provides you with a number of resources for interacting with the Discovery cluster:

- Launch a terminal that runs within your web browser without needing a separate terminal program. This is an advantage if you use Windows, as otherwise you need to download and use a separately installed program, such as MobaXterm.
- Use software applications, such as SAS Studio, that run in your browser without needing any further configuration. See *Using OOD's Interactive Apps* for more information.
- View, download, copy, and delete files using the File Explorer feature. See *Using OOD's File Explorer* for more information.

Note: OOD is a web-based application. You access it by using a web browser. Like many web-based applications, it has compatibility issues with certain web browsers. For optimal results, use OOD with newer versions of Chrome, Firefox, or Internet Explorer. OOD does not currently have support for Safari or mobile devices (phones and tablets).

Using OOD's File Explorer

When working with the resources in OOD, your files are stored in your home directory on the storage space on the Discovery cluster. Similar to any file navigation system, you can work with your files and directories through the OOD Files feature as detailed below. For example, you can download a Jupyter Notebook file in OOD that you've been working on to your local hard drive, rename a file, or delete a file that you no longer need.

Note: Your home directory has a file size limit of 100GB. Be sure to check your home directory regularly, and remove any files you do not need to ensure you do not run out of space.

1. In a web browser, go to ood.discovery.neu.edu. If prompted, enter your myNortheastern username and password.
2. Select **Files > Home Directory**. The contents of your home directory display in a new tab.
3. To download a file to your hard drive, navigate to the file you want to download, select the file, and click **Download**. If prompted by your browser, click **OK** to save your file to your hard drive.
4. To navigate to another folder on the Discovery file system, click **Go To**, enter the path to the folder you want to access, and click **OK**.

Note: From the **Files > Home Directory** view, the **Edit** button will not launch your .ipynb file in a Jupyter Notebook. It will open the file in a text editor. You have to be in Jupyter Notebook in order to launch a .ipynb file from your /home directory. See *Using OOD's Interactive Apps* for how to access a Jupyter Notebook through OOD.

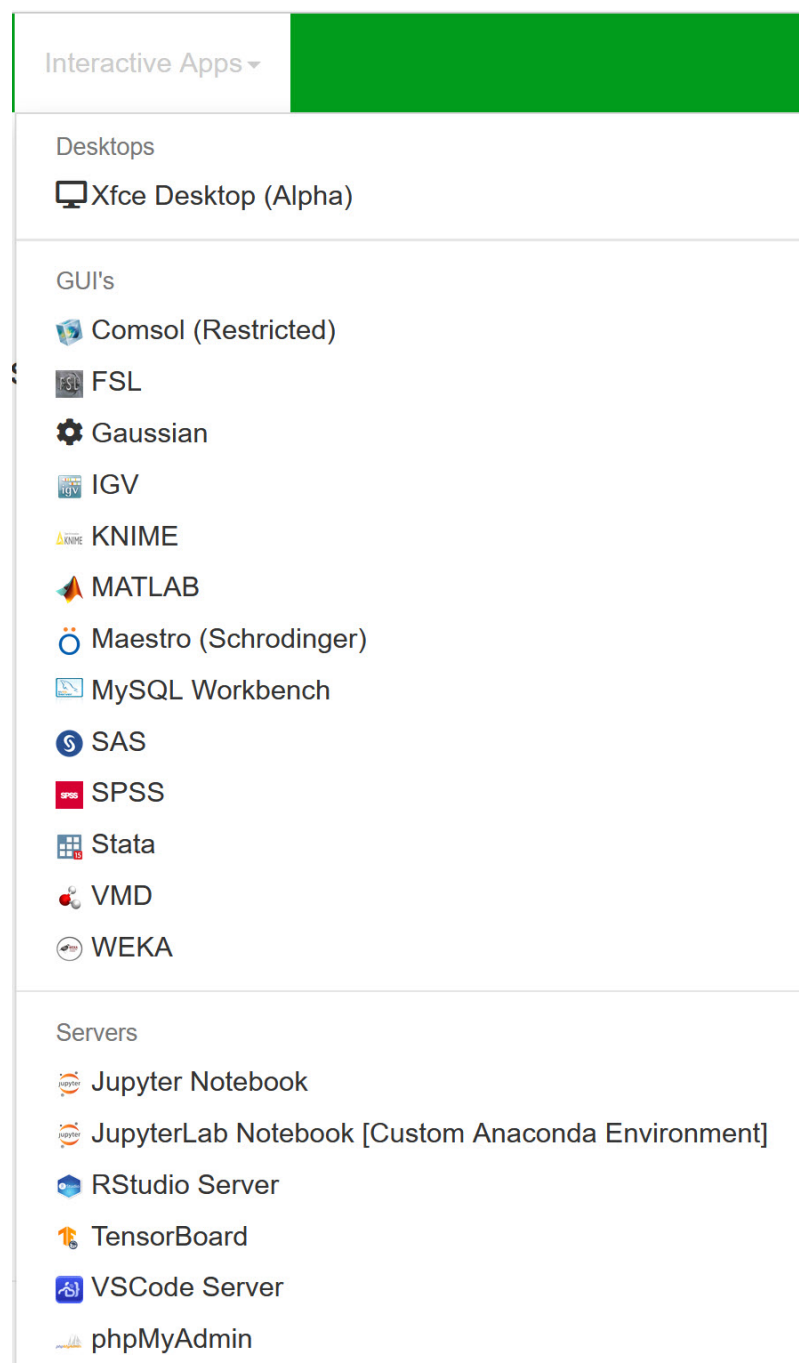
Using OOD's Interactive Apps

There are a number of applications that you can access and use through the OOD web portal. When you select an application and click launch, the scheduler (Slurm) allocates a compute node with a predetermined number of cores and amount of memory. The default time for running any of the applications is one hour. It is recommended that you keep the default. If you ask for more than one hour, you will need to wait for Slurm to allocate a resource that can run for your requested time, which could take a long time to be allocated, depending on how busy the cluster is.

If you are attempting to run a job on one of the interactive apps on OOD that launches a graphical user interface (GUI), such as Maestro, you might get an error if your passwordless ssh is not set up correctly. See [Using X11](#) for tips and troubleshooting information opening applications that use X11 forwarding.

28.1 Available Apps (November 2021)

On the Interactive Apps tab, you can view the list of available interactive apps through the OOD web interface.



Note: Some apps are reserved for specific research groups and are not for general access. If you get an access error when attempting to open an app, and you believe that you should have access to it, please email rchelp@northeastern.edu with your username, research group, the app you are trying to access, a screenshot of the error that you are getting, and we will look into the issue.

1. In a web browser, go to ood.discovery.neu.edu. If prompted, enter your myNortheastern username and password.
2. Select **Interactive Apps**, then select the application you want to use.
3. For most apps, keep the default options, and then click **Launch**. You might have to wait a minute or two for a

compute node to be allocated with your requested time and resource.

4. If you selected Jupyter Notebook, click **Connect to Jupyter**. A Jupyter Notebook opens in a new tab on your browser.

Tip: On the Jupyter Notebook Home tab, click **New**, then select the kernel you want to use, such as Python or R. Your selected kernel opens in a new tab on your browser. Click the **Running** tab to see a list of what Notebooks you already have running. If you already have a Notebook saved to your home directory, on the **Files** tab, click the name of the file to restart the Notebook in a new tab on your browser.

28.2 Working with Jupyter Notebook [Custom Anaconda Environment]

One of the interactive apps on OOD is Jupyter Notebook Custom Anaconda Environment. You need to do some initial setup in order to use this app effectively. This section will provide a walkthrough of setting up and using this app. The general workflow is to first create a virtual python environment; ensure that Jupyter Notebook is installed in your virtual environment; and then reference this environment when you start the Jupyter Notebook [Custom Anaconda Environment] OOD interactive app.

Virtual python environment

1. First set up a virtual python environment. See [Creating a Conda virtual environment with Anaconda](#) for how to set up a virtual python environment on Discovery using the terminal.
2. When your environment is ready, in the terminal type `source activate` to activate the base environment.
3. Type `which conda` to get the full path of your environment. Make a note of the full path name.
4. Type `source activate <yourenvironmentname>` where `<yourenvironmentname>` is the name of your custom environment to activate the environment you just created.
5. Type `conda install jupyterlab` to install jupyterlab in your environment.

Using OOD to launch Jupyter Notebook

1. Go to www.ood.discovery.neu.edu.
2. Click **Interactive Apps**, then select **Jupyter Notebook [Custom Anaconda Environment]**.
3. Select the compute node features for the job:
 - In the **Time** field, enter the number of hour(s) needed for the job. Minimum is one hour, maximum is 8 hours.
 - In the **Memory (in Gb)** field, enter the amount of memory you need for the job. Minimum is 2GB, maximum is 128GB.
 - If you need GPU, check **Use GPU** box.
4. Select the Anaconda version that you used to create your virtual python environment in the **System-wide Conda Module** field.
5. Check **Custom Anaconda Environment** box, and enter the name of your custom virtual python environment in the **Name of Custom Conda Environment** field.
6. Click **Launch**. This will put you in the queue for a compute node. Note that this might take a few minutes, depending on the resources you requested.
7. When you have been allocated a compute node, click **Connect to Jupyter**.

When your Jupyter Notebook is running and open, type `conda list` in a cell and run the cell to confirm that the environment is your custom conda environment (you should see this on the first line). This command will also list all of your available packages.

Using Discovery with your class FAQ

There are several use cases for teaching and learning with the Discovery cluster in a classroom. Discovery offers both a command line and web interface, allowing flexibility in access and instruction level. Using Discovery gives you and your students access to many popular scientific applications, and also allows you and your students to install other packages as needed. The easy-to-use Open onDemand web portal also offers a built-in visual file explorer for file viewing and transfer.

The following Frequently Asked Questions section should help to answer most of your questions about how you can use the Discovery cluster for classroom use. You can also reach out to us at rchelp@northeastern.edu or submit a ServiceNow ticket if you need further information.

How can I use Discovery with my class?

There are several ways you can use Discovery in your classroom. Your class can use Discovery for accessing specific software packages and working environments, as well as learning how to utilize high performance computing (HPC) resources for large and complex data processing, such as machine learning; AI and molecular simulations; and more.

How do I get my class access to Discovery?

You fill out a [Discovery Classroom Use Request](#) ticket. The form asks you to submit a list of your students' names and emails. We will create accounts on Discovery for them. Follow the steps in the article [How to Download a List of Student Email Addresses from Canvas](#) to get a list of your students' emails. If your class roster is not complete, you can attach a preliminary list to the ticket, and then follow up with us on the same ticket with an updated roster when the add/drop period is over.

Is there any training on Discovery for my class?

Yes, we currently provide online training for your class on using Discovery. In the past, we have also given in-person presentations during a class period on the Boston campus. We hope to provide in-person training again in 2021. We can customize the training to focus on the resources you will be using with them for the class. Email us at rchelp@northeastern.edu and provide us with some details about your class and what training you would like us to provide.

Do my students have to learn Linux to work with Discovery?

Depending on your class assignments, many students can work with the Open onDemand web portal, which does not require any knowledge of Linux to use. In cases where you want them to work on the command line, they should have

a basic understanding of Linux commands. Please see the question “Is there any training on Discovery for my class” above if you would like us to provide your class with a basic training course prior to using Discovery.

What software is available to use with my class on Discovery?

There are many software packages available, including popular software apps such as Jupyter Notebook, RStudio, and MATLAB. If you have an account on Discovery, you can see the list of available software by using the `module avail` command. See [Using Module](#) for more information. Students have access to all of the modules on Discovery. They can also use the interactive apps available on Open onDemand. See [Introduction to Open OnDemand \(OOD\)](#) for more information. We can also install additional modules and libraries specifically for your class as needed.

My class needs access to a specific software application that I do not see installed on Discovery or Open onDemand (OOD). What should I do?

If your class requires software not currently installed on Discovery or OOD, follow the procedure below to request that software be installed on Discovery. You must be a professor or instructor to initiate this request. Students in your class should not make this request. If your students need a specific software application, you must complete the form for them. This is to ensure that we only get one request for the software. Students in one class often make multiple requests for the same software, so having all requests go through the professor or instructor reduces this overlap.

To request additional software (instructors only):

1. Go to the [Discovery Cluster Software Request](#). If prompted, sign in to ServiceNow with your Northeastern username and password to access the form.
2. In the Sponsor’s Name field, enter your name.
3. Make sure to follow the instructions on the form regarding either providing the URL of the open source software library or uploading the installation package in your home directory if it requires you to register it first. The request will not be completed without this information.
4. Select the acknowledgement checkbox, and click **Submit**.

Note: Software requests can take up to 24 hours to verify, and then additional time is needed to complete the installation. We might not be able to install every software application requested. If this is the case we will notify you and try to provide alternative software to meet your needs.

Tip: You and your students can install software locally to your PATH on Discovery, which may be a better option in some cases, such as installing multiple conda environments. See [Software overview](#) for more information.

I just need my class to access Open onDemand. How do I request that?

Open onDemand is a web portal that lets you access the resources on Discovery through an easy-to-navigate web browser interface. You request course access the same way you would to get access to Discovery, as outlined above. Please specify that you’d like your course to be on Open onDemand for your students, in addition to the information we request above.

I’d like my class to use specific resources on Discovery. Can you create a reservation on Discovery for my class?

In many cases, we will create a reservation on Discovery for your class for specific hardware resources. For example, if your course assignments require the use of GPUs or nodes with a large amount of memory, we can create a reservation on specific nodes that only your class can access for the duration of the course. However, Discovery is a shared resource, so we ask that you test out your assignments on Discovery so that you are requesting an appropriate amount of resources for your class. If a class needs an increase in resources due to higher than expected use, we can always increase your reservation. But, if a reservation is not being used to capacity, we will ask you to review the need for the requested resources, and adjust the reservation accordingly. We understand that sometimes it is difficult to determine

exactly what your resource needs will be, but we do need to keep a reservation to a reasonable limit so that we keep the shared resources available to all users.

How long do my students have access to Discovery?

Students will have access to Discovery for the full duration of the class. If they want to continue to have access to Discovery after that time period, they'll need to request an individual account.

How do I get an account on Discovery?

If you are a professor or instructor at Northeastern, you can request an account on Discovery. See *Principal Investigator(PI)/Professor/Instructor Access* for more information.

How do my students get help with Discovery?

You and/or your students can either submit a [Get Assistance with Research Computing](#) ticket or email rchelp@northeastern.edu.

CHAPTER 30

CPS class-specific instructions

Caution: Note - the following instructions will only work for CPS classes.

These instructions describe the process of opening a CPS JupyterLab environment on the Open OnDemand (OOD) Discovery web portal and accessing class work directories.

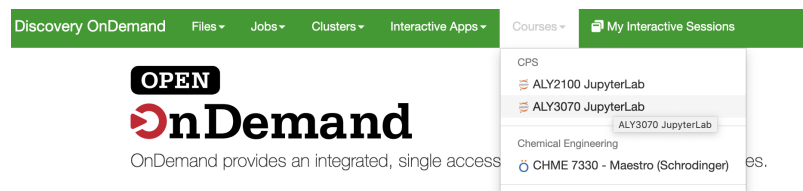
Note: Due to problems with launching OOD on **Safari**, we recommend using **Google Chrome**, **Mozilla Firefox** or **Microsoft Edge** browsers instead for best experience.

30.1 Open the CPS JupyterLab environment

Important: The class instructor needs to fill in the: [Discovery Classroom Use Request](#) You will only be able to find your class resources if a request was already made.

In a web browser, go to <http://ood.discovery.neu.edu>. Login with your NU credentials.

Under the **Courses** menu, select your Class Name (For example: **ALY3070 JupyterLab**):



Select the default options and click **Launch**. Wait until the session is successfully created and ready to be launched (turns green).

ALY3070 JupyterLab version: f9fa07d

This app will launch a Jupyter Lab application on a node with 2 CPUs and up to 64GB of memory for up to 8 hours on a compute node. Please note that this application is accessible only to students in class ALY3070.

Time

1

Enter time in Hours

Memory (In GB)

10

Working Directory

Enter the work directory to launch Jupyter Lab from. If left blank, will launch in the main class directory: `/work/cps/ALY3070`. You can also use your student directory: `/work/cps/ALY3070/students/[yourusername]`, where `[yourusername]` is your username on Discovery.

Additional Jupyter Notebook arguments (optional):

Enter any other optional arguments for Jupyter Notebook.

Launch

* The ALY3070 JupyterLab session data for this session can be accessed under the [data root directory](#).

For more control of the session, modify **Time** for the session time (in hours), **Memory** to get more memory in GB, and the **Working Directory** where JupyterLab will launch.

Note: If **Working Directory** is left blank, the session will launch in the main class folder (in this example `/work/cps/ALY3070`). Alternatively, start the session directly from your personal working directory by entering: `/work/cps/ALY3070/students/[username]`, where `[username]` is your username on Discovery. The instructions below assume the field is left blank.

Click **Connect to Jupyter** to open JupyterLab:

Session was successfully created.

Home / My Interactive Sessions

Clusters

Y_Xtce Terminal (Alpha)

Courses

CPS

ALY2100 JupyterLab

ALY3070 JupyterLab

Chemical Engineering

ALY3070 JupyterLab (22197440) 1 node | 2 cores | Running

Host: >u0978.discovery.neu.edu

Created at: 2021-11-29 16:11:48 EST

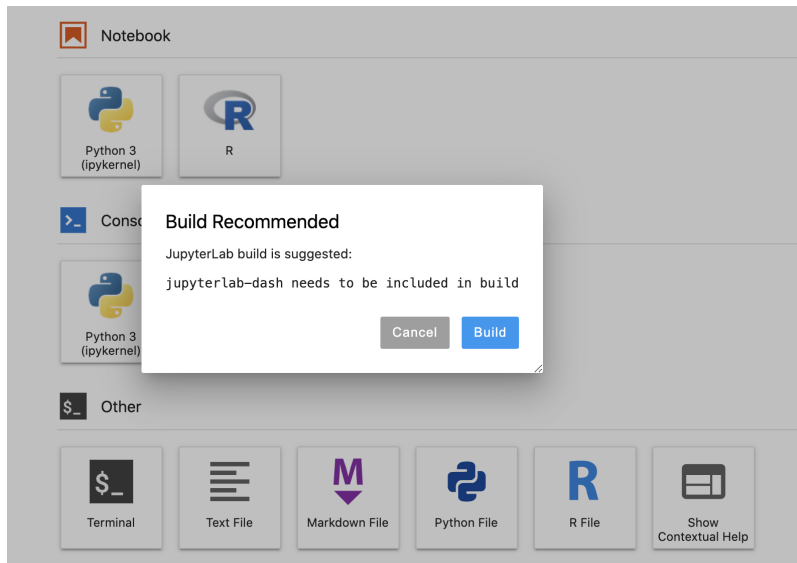
Time Remaining: 58 minutes

Session ID: 8e42e66f-5733-4442-9c37-7ec51452ca48

Connect to Jupyter

This will open a JupyterLab interface in another tab.

Select **Cancel** when prompted with the **Build Recommended** option:

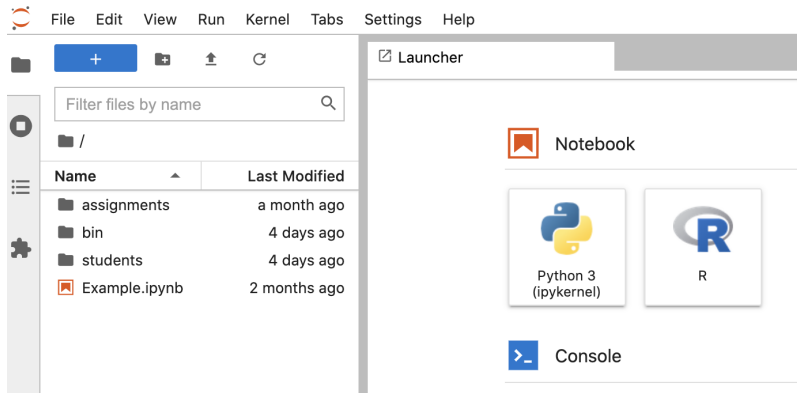


The package `jupyterlab-dash` does not require a build, and will not work when build is enabled.

30.2 Access CPS class directories

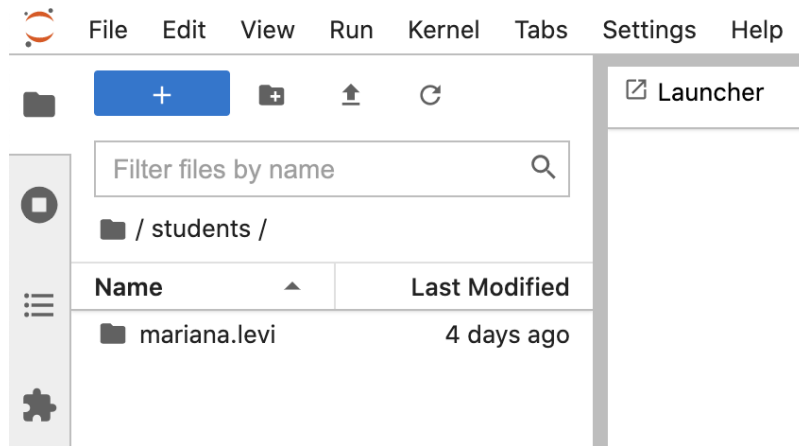
After you are connected to a CPS JupyterLab session on OOD, you can access any shared class directories and your private class directory.

You can navigate between the class folders using the left menu. Your instructor may share files in this directory:

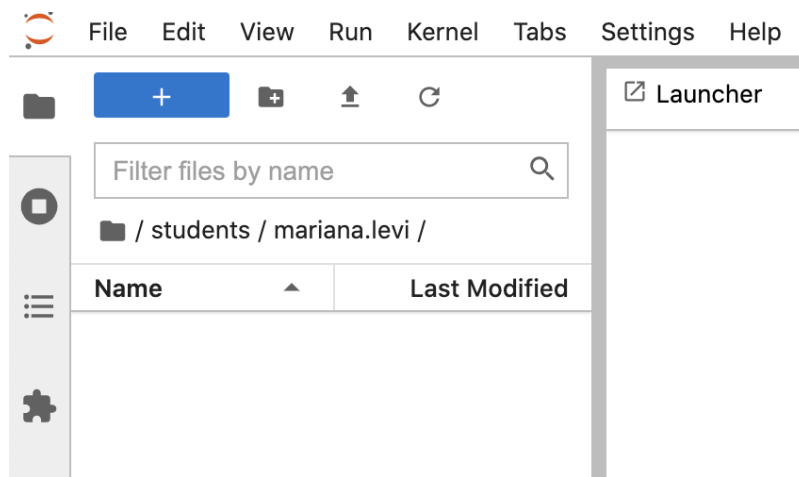


For instance, file **Example.ipynb** can be viewed using Python Jupyter Notebook (but not edited or removed).

Navigate to the **students** directory, where you will see another directory under your username:

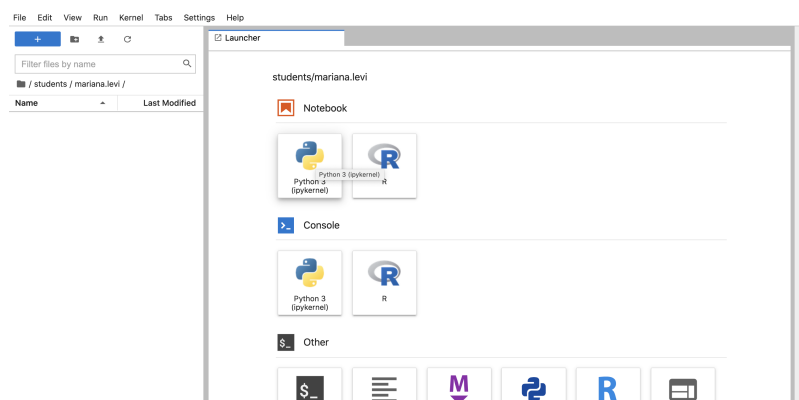


Enter your personal class directory (here, username *mariana.levi* is shown):

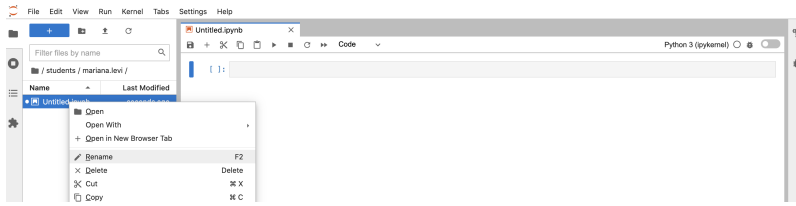


Now you can create and edit Jupyter Notebook files.

Open a new Python Notebook session from the Launcher menu by clicking the **Python 3 (ipykernel)**:



A new file will be created inside your directory called **Untitled.ipynb**. You can rename it by right-clicking on it and using the Rename option:



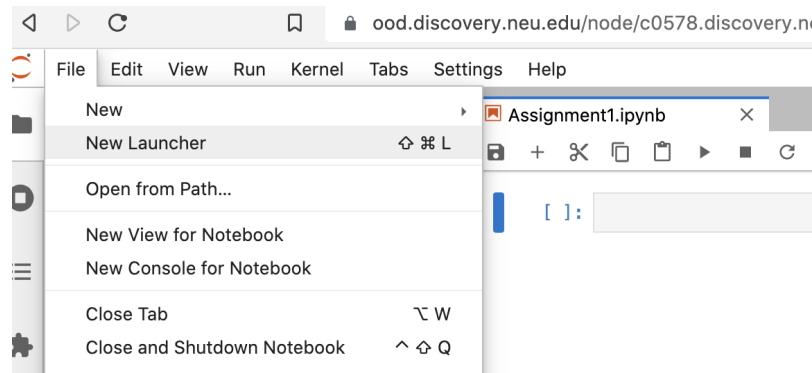
This Python notebook has ready-to-use Python packages needed for your class.

Note: Permission Denied errors: Do not attempt to create, edit or write files that are outside of your personal student directory. Most “Permission Denied” errors are due to directories or files having read-only access permissions.

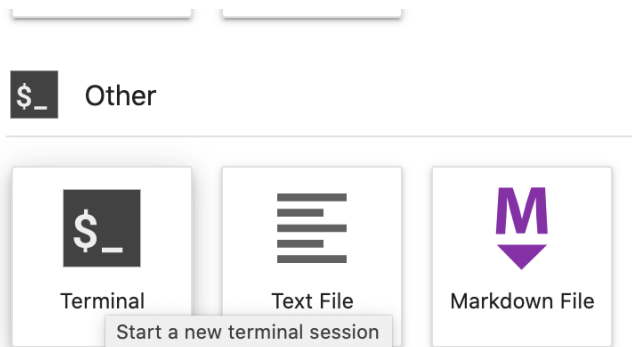
30.3 Submit CPS class assignments

Important: Due to the write-only access permissions on the **assignments** directory, it is required to use the command line interface (Linux Terminal) to submit assignments. **Using other methods, such as the JupyterLab interface or OOD File Explorer, currently does not work.**

To submit your assignment (for example, named: **Assignment1.ipynb**) to the **assignments** directory, open the JupyterLab New Launcher by clicking the **File** top menu option, and then selecting **New Launcher**:



Click on the **Terminal** option under **Other** to open a Linux terminal:



Navigate to your personal directory by typing the following command (change the class name from ALY3070 to your class name accordingly):

```
cd /work/cps/ALY3070/students/$USER
```

Where `$USER` is a saved shell variable for your username. You can optionally also replace it with your username.

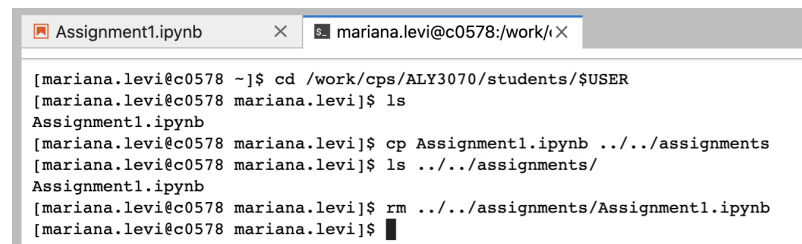
Check that your assignment file is visible in the command line by typing `ls`. Then, Copy the assignment file to the **assignments** directory with this command (replace **Assignment1.ipynb** with your file name):

```
cp Assignment1.ipynb ../../assignments
```

To remove an existing assignment, type (replace **Assignment1.ipynb** with your file name):

```
rm ../../assignments/Assignment1.ipynb
```

Close the Terminal tab when done.



The screenshot shows a terminal window with a tab titled "Assignment1.ipynb". The terminal output shows the following sequence of commands and their results:

```
[mariana.levi@c0578 ~]$ cd /work/cps/ALY3070/students/$USER
[mariana.levi@c0578 mariana.levi]$ ls
Assignment1.ipynb
[mariana.levi@c0578 mariana.levi]$ cp Assignment1.ipynb ../../assignments
[mariana.levi@c0578 mariana.levi]$ ls ../../assignments/
Assignment1.ipynb
[mariana.levi@c0578 mariana.levi]$ rm ../../assignments/Assignment1.ipynb
[mariana.levi@c0578 mariana.levi]$
```