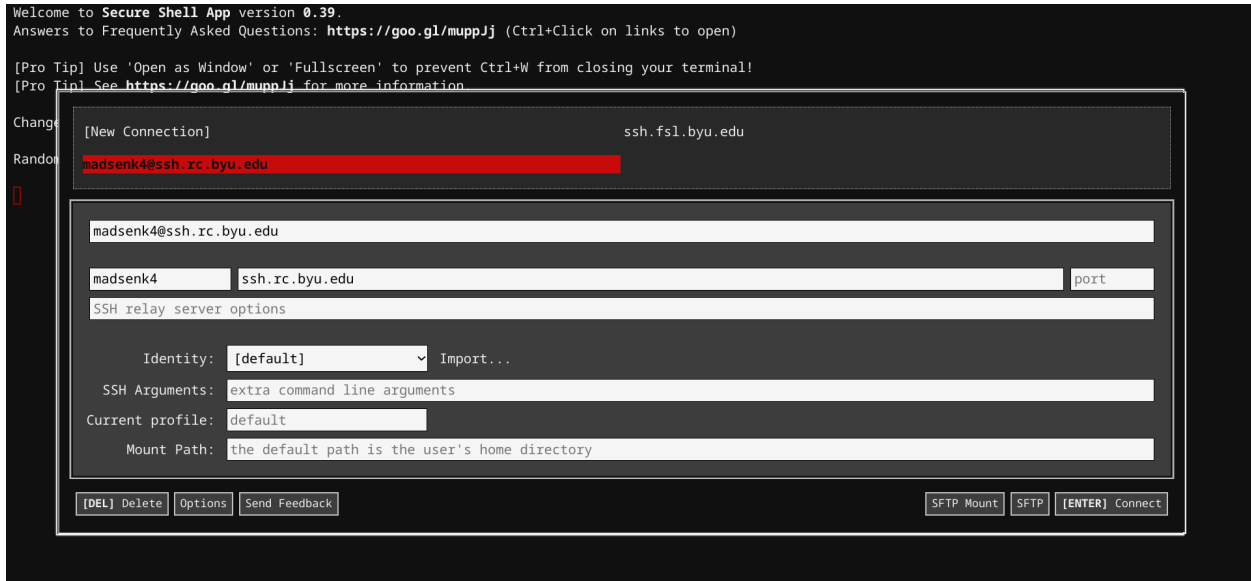


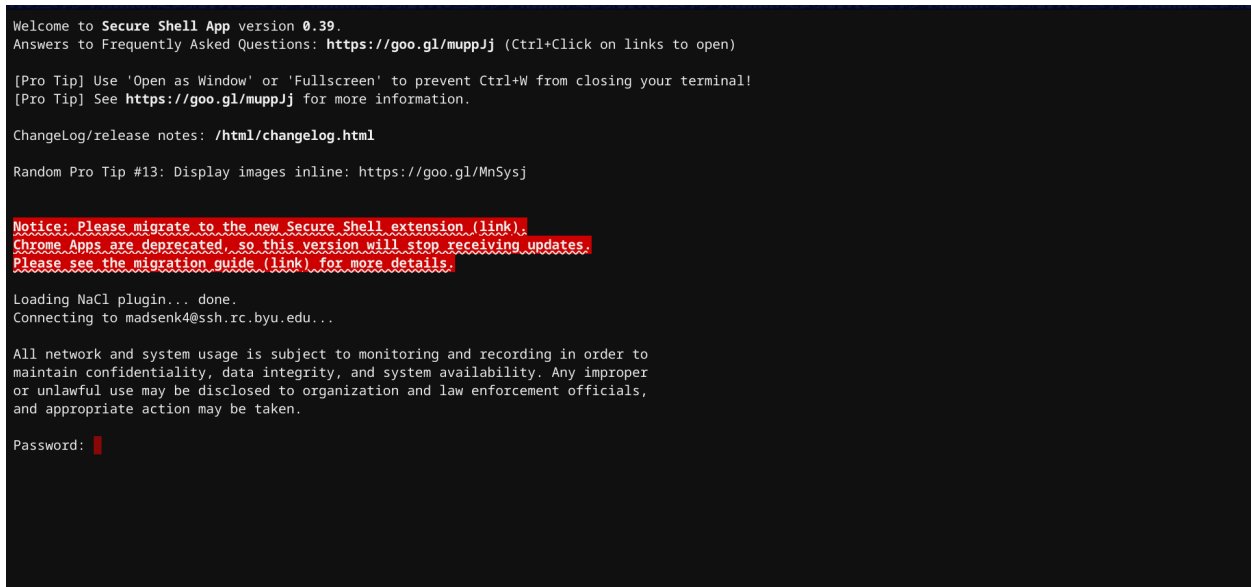
Introduction to the Supercomputer

Logging into the Supercomputer

Fill out the first three boxes with [your netID]@ssh.rc.byu.edu, and hit enter



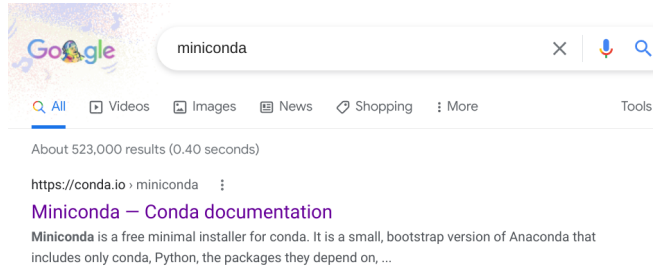
Enter the same password you use for the fulton supercomputing lab and the authentication code. No letters will appear on the screen - don't worry, it's still receiving the input.



Importing CONDA

Step 1: Get miniconda

Google miniconda and click on the conda documentation.



Scroll down to 'Linux Installers' and right-click on Miniconda3-Linux64-bit to copy the link.

Conda
latest

Search docs

Conda
Conda-build

Miniconda

- System requirements
- Latest Miniconda Installer Links
- Windows installers
- MacOSX installers
- Linux installers
- Installing
- Other resources
- Help and support
- Contributing
- Conda license

Python version	Name	Size	SHA256 hash
Python 3.9	Miniconda3 Linux 64-bit	63.6 MIB	1ea2f885b4dbc3898662845560bc64271eb17085387a70c2ba3f29fffe78d52f
	Miniconda3 Linux-aarch64 64-bit	62.6 MIB	4879829a19718743f945d88ef142c3a4b38dfc8e448d1ca98e019586374b773f
	Miniconda3 Linux-ppc64le 64-bit	60.6 MIB	fa92ee4773611f50ed933f977d32bb64769292f685d518732183be1f3321fa
	Miniconda3 Linux-s390x 64-bit	57.1 MIB	1faed9abecf4a4dd4e0d8891fc2cdaa3394c51e877af14ad6b9d4aad4e90d8
Python 3.8	Miniconda3 Linux 64-bit	98.8 MIB	935d72deb16e42739d69644977290395061b7a6db059b316958d97939e9bdf3d
	Miniconda3 Linux-aarch64 64-bit	94.8 MIB	19584b4fb5c0656e0cf9de72aaa8b0a7991fbd0f1254d12e2119048c9a47e5cc
	Miniconda3 Linux-ppc64le 64-bit	93.3 MIB	c1ac79540cb77b2e0ca5b9f78b3b36767d810118500a167dea4a0cab50963
Python 3.7	Miniconda3 Linux-s390x 64-bit	89.0 MIB	55f514110a50e98549a68912cbb03e43a36193940a1889e1c8be39089b4da19
	Miniconda3 Linux 64-bit	84.9 MIB	a1a7285dea8edc439b2bc7951d89b39a2a1b32926d2a7b02aacaaa95cf69c7c
	Miniconda3 Linux-aarch64 64-bit	89.2 MIB	65f480a906e3132d0bba35a38d619478be77d32210a2acab05133d92ba08f111
	Miniconda3 Linux-ppc64le 64-bit	88.1 MIB	e4f8b4a5eb0da1badf060c91fd7ee25e39120d4d77443e7a1ef3661f6439a997
	Miniconda3 Linux-s390x 64-bit	84.1 MIB	7ab9f813d084cb0951a2d755c084708263ce4e03c65e6e5e2fa79e08f024f0f7

Make sure you're in the compute directory: `cd ~/compute`

Then use wget with the copied url:

```
-bash-4.2$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-...
```

Your screen should look something like this:

```
Resolving repo.anaconda.com (repo.anaconda.com)... 104.16.131.3, 104.16.130.3
, 2606:4700::6810:8203, ...
Connecting to repo.anaconda.com (repo.anaconda.com)|104.16.131.3|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 93052469 (89M) [application/x-sh]
Saving to: 'Miniconda3-latest-Linux-x86_64.sh'

100%[=====>] 93,052,469    210MB/s   in 0.4s

2020-09-17 14:50:09 (210 MB/s) - 'Miniconda3-latest-Linux-x86_64.sh' saved [9
3052469/93052469]
```

Step 2: Initialize CONDA

Type in `bash Miniconda3-latest-Linux-x86_64.sh` and hit enter. When prompted, accept any terms and conditions / questions (say yes by hitting the Y key or hit enter).

*the easiest thing to do is use tab complete: type "bash Mi", then hit tab, and the rest should fill in

```
(base) -bash-4.2$ bash Miniconda3-latest-Linux-x86_64.sh
```

When the miniconda installation prompts you if you want to initiate conda and it it to your ~/.bashrc, type "y"

When you're done, the command line will appear with a (base) after typing in `source ~/.bashrc`

```
(base) -bash-4.2$
```

Writing a Job File

We'll be using the [Job Script Generator](https://rc.byu.edu/documentation/slurm/script-generator) (<https://rc.byu.edu/documentation/slurm/script-generator>).

The screenshot shows the BYU Office of Research Computing website. The 'Documentation' menu is open, displaying a list of links. The 'Job Submission' link is highlighted, and the 'Job Script Generator' link is selected. The page also features a 'Calendar' section with no upcoming events, a 'News / Notices' section with no notices, and a 'Current Utilization' section with a message about RHEL 7 transition.

To get here, go to Documentation > Job Submission > Job Script Generator

By default, mine looks something like this:

Video Tutorials

- [How to use the Script Generator](#)
- [How to manage jobs using SLURM](#)

Parameters (video tutorial)	
Limit this job to one node:	<input checked="" type="checkbox"/>
Number of processor cores across all nodes : <small>#nodes * #cores</small>	<input type="text" value="1"/>
Number of GPUs: <small>Very limited number of GPUs available.</small>	<input type="text" value="0"/> <small>Only use this if your code actually utilizes GPUs.</small>
Memory per processor:	<input type="text" value="1"/> GB
Walltime:	<input type="text" value="01"/> hours <input type="text" value="00"/> mins <input type="text" value="00"/> secs
Job is a test job:	<input type="checkbox"/>
Job is preemptable :	<input type="checkbox"/>
I am in a file sharing group and my group members need to read/modify my output files:	<input type="checkbox"/>
Need licenses?	<input type="checkbox"/>
Job name:	<input type="text"/>
Receive email for job events:	<input type="checkbox"/> begin <input type="checkbox"/> end <input type="checkbox"/> abort
Email address:	<input type="text" value="kimmermadsen@gmail.com"/>

If you would like to enter your email, you can opt to receive an email when the job has begun, ended, or if it aborted. It can be nice to get these kinds of alerts instead of checking the job status every few minutes from the command line.

Most labs will specify ranges for the number of processors, memory per processor, and walltime.

For practice, since we're submitting a very small job, you can just say 1 processor core with 1 GB memory and 10 minutes for simple practice code. If it aborts, you can modify these numbers. It's also helpful to name your job. In this case, I named mine "hello_there"

Job Script Generator

Video Tutorials

- [How to use the Script Generator](#)
- [How to manage jobs using SLURM](#)

Parameters (video tutorial)	
Limit this job to one node:	<input checked="" type="checkbox"/>
Number of processor cores across all nodes: <small>#nodes * #cores</small>	<input type="text" value="1"/>
Number of GPUs: <small>Very limited number of GPUs available.</small>	<input type="text" value="0"/>
Memory per processor:	<input type="text" value="1"/> GB
Walltime:	<input type="text" value="00"/> hours <input type="text" value="10"/> mins <input type="text" value="00"/> secs
Job is a test job:	<input type="checkbox"/>
Job is preemptable :	<input type="checkbox"/>
I am in a file sharing group and my group members need to read/modify my output files:	<input type="checkbox"/>
Need licenses?	<input type="checkbox"/>
Job name:	<input type="text" value="hello_there"/>
Receive email for job events:	<input checked="" type="checkbox"/> begin <input checked="" type="checkbox"/> end <input checked="" type="checkbox"/> abort
Email address:	<input type="text" value="kimmermadsen@gmail.com"/>

When you're satisfied with the settings, scroll down to where it says 'Job Script' and click "copy script to clipboard" (you can also highlight and copy - it doesn't really matter).

[Update Script](#) [Copy Script to Clipboard](#) [Save Script](#)

Job Script

```
#!/bin/bash

#SBATCH --time=00:10:00 # walltime
#SBATCH --ntasks=1 # number of processor cores (i.e. tasks)
#SBATCH --nodes=1 # number of nodes
#SBATCH --mem-per-cpu=1024M # memory per CPU core
#SBATCH -J "hello_there" # job name
#SBATCH --mail-user=kimmermadsen@gmail.com # email address
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --mail-type=FAIL

# Set the max number of threads to use for programs using OpenMP. Should be <= ppn. Does nothing if the program doesn't use OpenMP.
export OMP_NUM_THREADS=SLURM_CPUS_ON_NODE

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
```

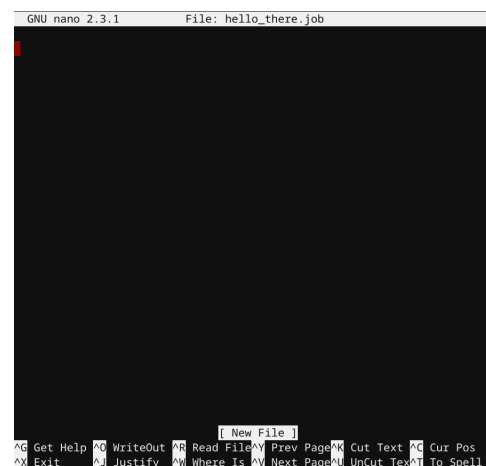
In the command line, create a text file called "hello_there.job" using the 'nano' command:

nano hello_there.job

(hit enter)

You'll be brought to a screen that looks like this:

Paste in the job script you just copied, then type in the command: echo "hello there!", sleep 75, then echo "You're finished!". This will tell the computer to print out 'hello there!', wait 75 seconds, then print out "you're finished!"



Mine looks like this:

```
GNU nano 2.3.1      File: hello_there.job      Modified

#!/bin/bash

#SBATCH --time=00:10:00    # walltime
#SBATCH --ntasks=1        # number of processor cores (i.e. tasks)
#SBATCH --nodes=1         # number of nodes
#SBATCH --mem-per-cpu=1024M    # memory per CPU core
#SBATCH -J "hello_there"     # job name
#SBATCH --mail-user=kimmermadsen@gmail.com    # email address
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --mail-type=FAIL

# Set the max number of threads to use for programs using OpenMP. Should be
export OMP_NUM_THREADS=$SLURM_CPUS_ON_NODE

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE

echo "hello there!"
sleep 75
echo "You're finished!"
```

Use `^O` to save or rename your text file, and `^X` to exit. After hitting `^X`, be sure to hit 'Y' and enter to save the changes you've made

```
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No          ^C Cancel
```

You can check that it saved by using the cat command to view the text file:

```
-bash-4.2$ nano hello_there.job
-bash-4.2$ cat hello_there.job
#!/bin/bash

#SBATCH --time=00:10:00    # walltime
#SBATCH --ntasks=1        # number of processor cores (i.e. tasks)
#SBATCH --nodes=1          # number of nodes
#SBATCH --mem-per-cpu=1024M # memory per CPU core
#SBATCH -J "hello_there"   # job name
#SBATCH --mail-user=kimmermadsen@gmail.com # email address
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --mail-type=FAIL

# Set the max number of threads to use for programs using OpenMP. Should be
# <= ppn. Does nothing if the program doesn't use OpenMP.
export OMP_NUM_THREADS=$SLURM_CPUS_ON_NODE

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE

echo "hello there!"
sleep 75
echo "You're finished!"
-bash-4.2$
```

Submitting a Job

Once you have the job file created, use the 'sbatch' command to submit your job.

```
-bash-4.2$ sbatch hello_there.job
Submitted batch job 43392291
-bash-4.2$
```

Checking the Job Status

Use the squeue command to check job status:

squeue -u [your netID or username]

```
-bash-4.2$ sbatch hello_there.job
Submitted batch job 43392292
-bash-4.2$ squeue -u madsenk4
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
43392292	m8n	hello_th	madsenk4	R	0:02	1	m8-19-3

You can do this as much as you'd like:

```
-bash-4.2$ squeue -u madsenk4
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
43392292	m8n	hello_th	madsenk4	R	0:02	1	m8-19-3

```
-bash-4.2$ squeue -u madsenk4
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
43392292	m8n	hello_th	madsenk4	R	0:25	1	m8-19-3

```
-bash-4.2$ squeue -u madsenk4
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
43392292	m8n	hello_th	madsenk4	R	0:25	1	m8-19-3

```
-bash-4.2$
```

If you signed up for email alerts, you'll be notified if the job is begun, finished, or aborted.

Remember to properly exit out of the super computer by typing in 'exit':

```
-bash-4.2$ exit
logout
Connection to ssh.rc.byu.edu closed.
```