

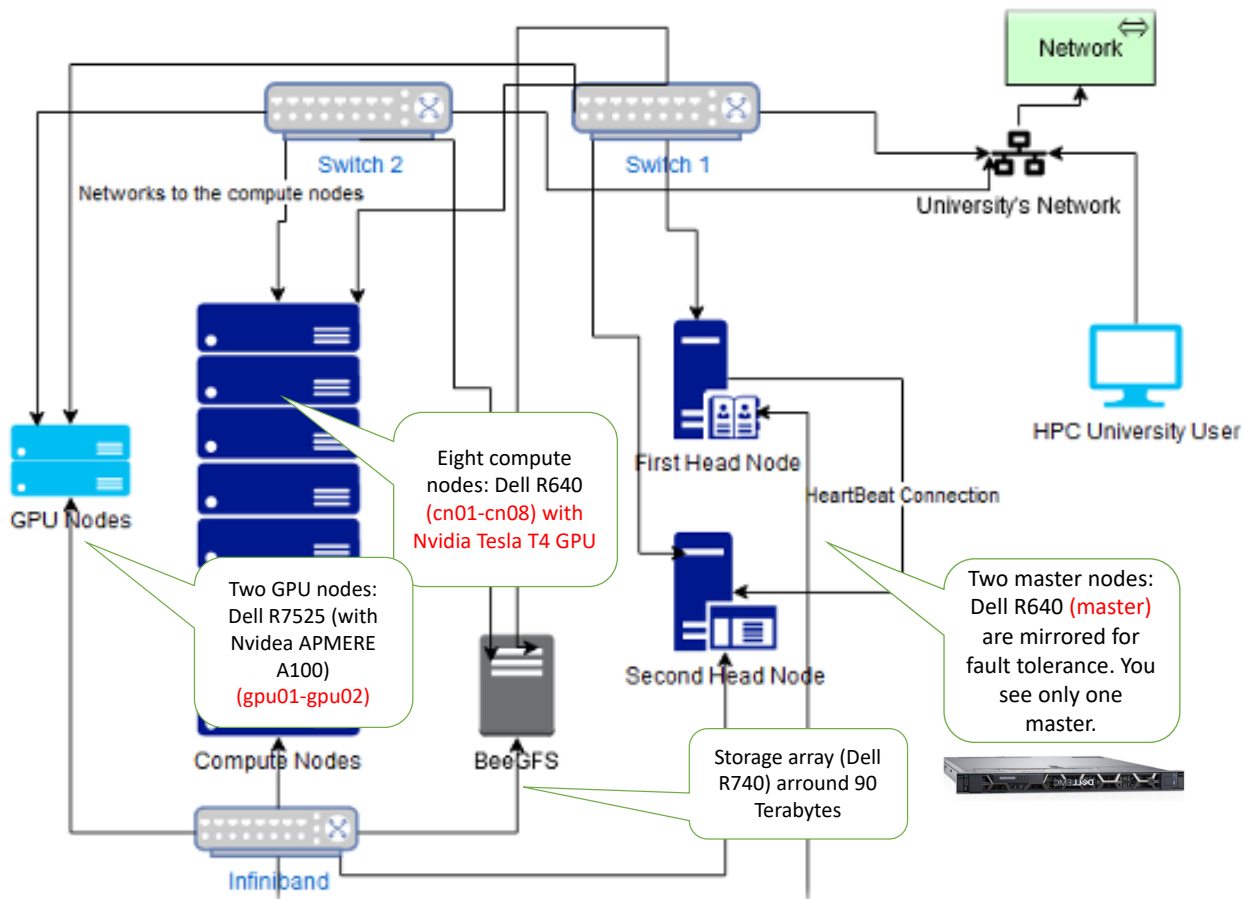
# Guide to Using the Benefit AI Lab HPC Cluster

Prepared by: The Benefit Advanced AI and Computing Lab Committee

November 2022 (version 0.6)

## 1. Cluster Architecture

The Benefit Lab HPC cluster architecture can be seen below in Figure 1.



**FIGURE 1. HPC CLUSTER ARCHITECTURE OVERVIEW**

The cluster architecture has the following main components:

- Two **master nodes** (Dell R640) working in mirrored synchrony (seen by the user as a single master node) in order to avoid having the master node be a single point of failure.
- Eight **computer nodes** (Dell R640) that each has an Nvidia Tesla T4 GPU.
- Two **GPU nodes** (Dell R7525) with AMD CPUs and two A100 GPUs in each server.

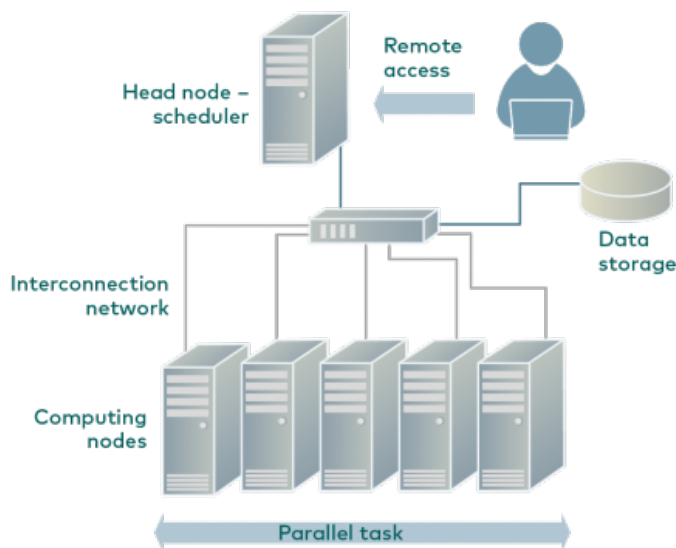
The domain names and IP addresses of these nodes can be seen in Table 1 below.

**TABLE 1. NODE NAMES AND IP ADDRESSES**

Server Name	Description
master	Dell R640 used for accessing the cluster as a login node. IP address: 10.240.240.11
cn01	Eight compute nodes Dell R640 server with Intel processors and Nvidia Tesla T4 GPUs, used as compute nodes. 24 cores/node 384 Gb combined memory
cn02	
cn03	
cn04	
cn05	
cn06	
cn07	
cn08	
gpu01	Two GPU Dell R7525 servers with AMD CPUs and two A100 GPUs each.
gpu02	

In August 2022, to test the speed of the Benefit Lab cluster, the Linpack Benchmark was tested and it reached 14.25 Tera FLOPS for the combined 8 compute nodes, and 35.25 Tera FLOPS on the two GPU AMD nodes.<sup>22</sup>

From the user’s perspective, Figure 2 shows the logical view of the cluster architecture.



**FIGURE 1. HPC CLUSTER ARCHITECTURE OVERVIEW**

## 2. Requesting an Account and Terms of Use

Before you start you should request an account. Staff members can request an account directly from the **system administrator** [ailab@uob.edu.bh](mailto:ailab@uob.edu.bh), while students should have a staff member approve their request. All users should read and agree to the [Terms of Use](#) document.

The system administrator will create an account for you based on the category you belong to. Table 2 shows the basic categories available, with their privileges.

**TABLE 2. DEFAULT USER GROUPS**

Group Name	Description	Initial storage quota	SLURM compute node quota	Default user period
ugstudents	Undergraduate students	2 GB	4 compute nodes 0 GPU nodes	4 months (1 semester)
pgstudents	Postgraduate students	6 GB	8 compute nodes 2 GPU nodes	4 months (1 semester)
faculty	UOB faculty members	6 GB	8 compute nodes 2 GPU nodes	4 months (1 semester)
sysadmin	System administrators	4 GB	All available nodes	-

By submitting this form, users agree to the following policies general policies detailed in the Terms of Use:

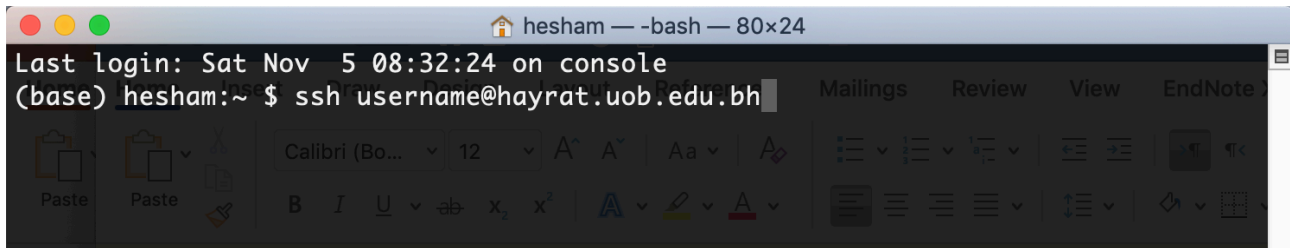
1. Use the batch submission system of the scheduler (SLURM).
  - A. Users may not run applications interactively from the login node.
  - B. Users may not log into the compute nodes for the purpose of running a job directly without permission from the Benefit AI Lab committee.
  - C. To request access other than batch submission, please send a request explaining why special access is necessary for your research to [ailab@uob.edu.bh](mailto:ailab@uob.edu.bh) .
2. Unauthorized Interactive jobs or applications found running on the login/master node(s) or on the compute nodes outside of Benefit AI Lab Terms of Use are subject to immediate termination.
3. Respond in a timely way to correspondence from the Benefit AI Lab Committee.
4. All correspondence will be sent to the e-mail address on file with the Benefit AI Lab committee.
5. Users must not attempt to access other users' data or install unauthorized or malicious software on the cluster.
6. Users agree to ensure that all publications or other outputs, which involved the use of Benefit AI Lab resources credit them appropriately as described below.
7. Users agree to provide on request reports detailing their accomplishments achieved with involvement of Benefit AI Lab resources and expertise.

- Citing the Benefit Advanced AI and Computing Lab in an acknowledgment section of a publication

Acknowledgment: The experiments presented in this paper were carried out using the facilities of the Benefit Advanced AI and Computing Lab at the University of Bahrain {\small -- see \url{<https://ailab.uob.edu.bh>}} with support from Benefit Bahrain Company {\small -- see \url{<https://benefit.bh>}}.

### 3. Logging-In to the Master Node

You can use secure shell to log in to the master node `hayrat.uob.edu.bh` (IP address 10.240.240.11). Figure 3 shows a basic login from the shell command line (users of Windows 10 or before can use **putty** to login if **ssh** is not available).



**FIGURE 3. SSH COMMAND TO LOGIN TO MASTER NODE**

Once logged in please change your password using the `passwd` command if it is your first time. Use a strong password that you can remember in order to make sure your account is secure.

The master node and all compute nodes are running CentOS Linux. Please refer to tutorials on Linux to see how you can use the command line (CLI) for loading your data and running programs. This guide will show you the details of running Jupyter on a compute node in a separate section below.

## 4. Slurm Workload Manager Basics

The Benefit AI Lab Cluster uses slurm as a scheduler and workload manager. As a warning, note that **on a cluster, you do not run the computations on the login node**. Computations belong on the compute nodes, when, and where they will be run is decided by the scheduler (like slurm).

In the Benefit AI Lab cluster, this is the master node: `hayrat`.

After logging in to `hayrat` you can submit a job using slurm, and it will run it on the **compute** or **gpu** nodes that you specify in the submission script.

The workload manager tries to distribute the resources based on the cluster rules. Resources available for slurm include:

- CPU cores
- RAM
- GPUs

You can request these resources nicely through **slurm** using the shell script and **slurm** `sbatch` or `srun` commands. But the ultimate decision is taken by the workload manager.

The system administrator also divides the cluster into partitions, and each user group will have some of these partitions available to them based on their privileges. A partition is a set of compute nodes (computers dedicated to... computing) grouped logically based on either physical properties of the hardware or job scheduling policies. Once the submitted job is executed, the output of the jobs is then written into disk (or storage).

### Gathering Cluster Information

Slurm offers the `sinfo` command to get an overview of the resources offered by the cluster.

By default, `sinfo` lists the partitions that are available on the cluster.

```
[halammal@master ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
standard* up       infinite   4      idle  cn[01-04]
compute   up       infinite   8      idle  cn[01-08]
gpu       up       infinite   2      idle  gpu[01-02]
```

As you can see from the result of the basic `sinfo` command you can see that there are three partitions in this cluster: **standard** with 4 compute nodes `cn01` to `cn04` (which is the default), then **compute** with eight nodes, and finally **gpu** with the two GPU nodes.

You can output node information using `sinfo -N -l`. With the `-l` argument, more information about the nodes is provided, among which the number of “CPUs” (CPUS), which is the number of processing units that the jobs can use. It should generally correspond to the number of sockets (S) times number of cores per socket (C) times number of hardware threads per core (T in the S:C:T column) but can be lower in the case some CPUs are reserved for system use.

```
[halammal@master ~]$ sinfo -N -l
Sun Nov 13 11:11:17 2022
NODELIST      NODES PARTITION STATE CPUS  S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
cn01          1 standard*  idle 24   2:12:1 385448      0      1 (null) none
cn01          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn02          1 standard*  idle 24   2:12:1 385448      0      1 (null) none
cn02          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn03          1 standard*  idle 24   2:12:1 385448      0      1 (null) none
cn03          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn04          1 standard*  idle 24   2:12:1 385448      0      1 (null) none
cn04          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn05          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn06          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn07          1 compute   idle 24   2:12:1 385448      0      1 (null) none
cn08          1 compute   idle 24   2:12:1 385448      0      1 (null) none
gpu01         1 node[cpu]  idle 192  2:48:2 103164     0     16 Int 1 gpu,cent none
gpu02         1 node[cpu]  idle 192  2:48:2 103164     0     16 Int 1 gpu,cent none
```

The other columns report the volatile working memory (RAM – MEMORY), the size of the local temporary disk (also called local scratch space – TMP\_DISK), and the node “weight” (an internal parameter specifying preferences in nodes for allocations when there are multiple possibilities).

## Running Jobs on slurm

Jobs are made of one or multiple sequential steps, each consisting in one or multiple parallel tasks that will be dispatched to possibly distinct nodes. Each task will be allocated CPUs, memory, and possible other generic resources in an exclusive manner by **slurm**.

Two jobs cannot share the same resource unless explicitly forced by the admins, but that is generally not the case. Therefore, jobs can only start when all needed resources are free and not needed by another higher priority job. Jobs are indeed assigned a priority when they are submitted, which can depend upon multiple factors.

For the scheduling process to work properly, you will need to describe your job before you submit it:

- what are the steps (i.e. which program must be run and how) ;
- how many tasks there will be ;
- what resource each task needs (CPU, memory, etc.), and
- for how long the job is supposed to run.

All of these, along with potentially additional job parameters submission options, can be described in a submission script. The expression `#SBATCH` is used to specify these parameters in the script.

**Important:** The `#SBATCH` directives must appear at the top of the submission file, before any other line except for the very first line which should be the shebang (e.g. `#!/bin/bash`).

As an example, the following should be entered in a file named `submit.sh`:

```
#!/bin/bash
#
#SBATCH --job-name=test
#SBATCH --output=res.txt
#SBATCH --partition=standard
#
#SBATCH --time=10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=100

srun hostname
srun sleep 60
```

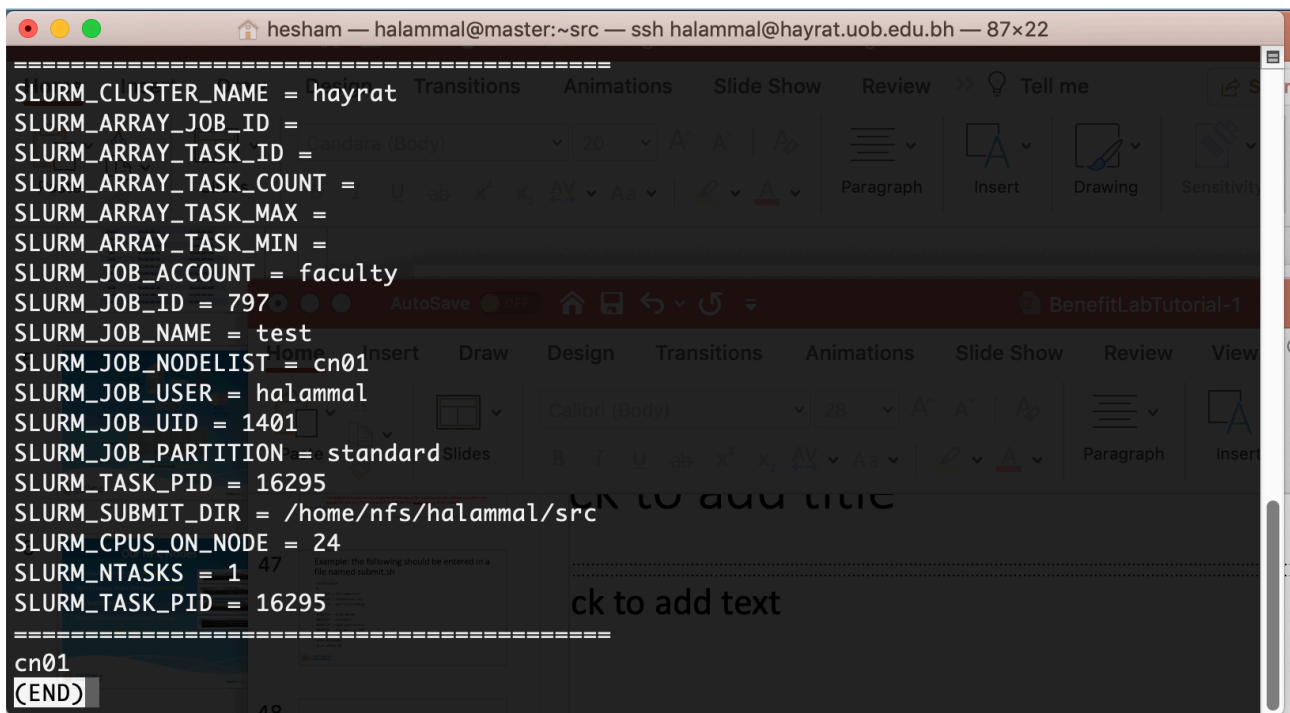
This job does not do a lot. It will only display the hostname on the compute node it is running on and then sleep for 60 minutes. Note that these programs are run using the slurm command `srun`. You should use an editor like nano to write this job in the file (here called `submit.sh`) and save. Then, to run this job, you can use the `sbatch` command as follows:

```
[halammal@master src]$ sbatch submit.sh
Submitted batch job 797
[halammal@master src]$ squeue --me
      JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
       797  standard     test  halammal  R           0:15      1  cn01
```

Note that the `squeue` command can show you the job queue and its current compute node and status.



As we specified in the job script that we want our output to be stored in a file called res.txt, after the job is finished, you can view the output as follows:



```
=====  
SLURM_CLUSTER_NAME = hayrat  
SLURM_ARRAY_JOB_ID =  
SLURM_ARRAY_TASK_ID =  
SLURM_ARRAY_TASK_COUNT =  
SLURM_ARRAY_TASK_MAX =  
SLURM_ARRAY_TASK_MIN =  
SLURM_JOB_ACCOUNT = faculty  
SLURM_JOB_ID = 797  
SLURM_JOB_NAME = test  
SLURM_JOB_NODELIST = cn01  
SLURM_JOB_USER = halammal  
SLURM_JOB_UID = 1401  
SLURM_JOB_PARTITION = standard  
SLURM_TASK_PID = 16295  
SLURM_SUBMIT_DIR = /home/nfs/halammal/src  
SLURM_CPUS_ON_NODE = 24  
SLURM_NTASKS = 1  
SLURM_TASK_PID = 16295  
=====  
cn01  
(END)
```

The res.txt file contains a brief report on the job, plus the output which is in this case just the name of the compute node (result from running the `hostname` command).

This was just a basic example, and there are many other commands that you can use to control your submitted jobs such as: `srun`, `scancel`, and `sview`. For more information please refer to the **slurm** user documentation <https://slurm.schedmd.com/documentation.html> .

## 5. Running Jupyter Lab

You can run Jupyter Lab interactively in one of the nodes using the following steps.

Step 1. Initiate conda from the **master** node using the command: `conda init`

```
[halammal@master halammal]$ nano cluster_jupyter_instructions.sh
[halammal@master halammal]$ conda init
no change      /data/software/miniconda3/condabin/conda
no change      /data/software/miniconda3/bin/conda
no change      /data/software/miniconda3/bin/conda-env
no change      /data/software/miniconda3/bin/activate
no change      /data/software/miniconda3/bin/deactivate
no change      /data/software/miniconda3/etc/profile.d/conda.sh
no change      /data/software/miniconda3/etc/fish/conf.d/conda.fish
no change      /data/software/miniconda3/shell/condabin/Conda.psm1
no change      /data/software/miniconda3/shell/condabin/conda-hook.ps1
no change      /data/software/miniconda3/lib/python3.9/site-packages/xontrib/conda.xsh
no change      /data/software/miniconda3/etc/profile.d/conda.csh
no change      /home/nfs/halammal/.bashrc
No action taken.
[halammal@master halammal]$
```

Step 2. Prepare a shell script to submit for **slurm** that will run jupyter in one of the nodes like the following. Notice the time limit and the specification of the node. Also, choose the partition and specify the output file, which will be necessary to get the URL that will run Jupyter.

Change the settings then use `sbatch` to run the shell script.

```
#!/bin/bash
#SBATCH -J jupyterlab
#SBATCH --output=jupyterlab_config.out
#SBATCH --nodes=1
#SBATCH --time=30:00 # Set the time you want you jupyter lab to be running, if you remove this line it will take the partition default which is infinity.
#SBATCH --partition=compute # Change the partition to gpu if you want to connect to AMD/A100 machines.
source ~/.bashrc # Make conda available to slurm

echo 'hostname: ' `hostname` # Print the hostname to the output file

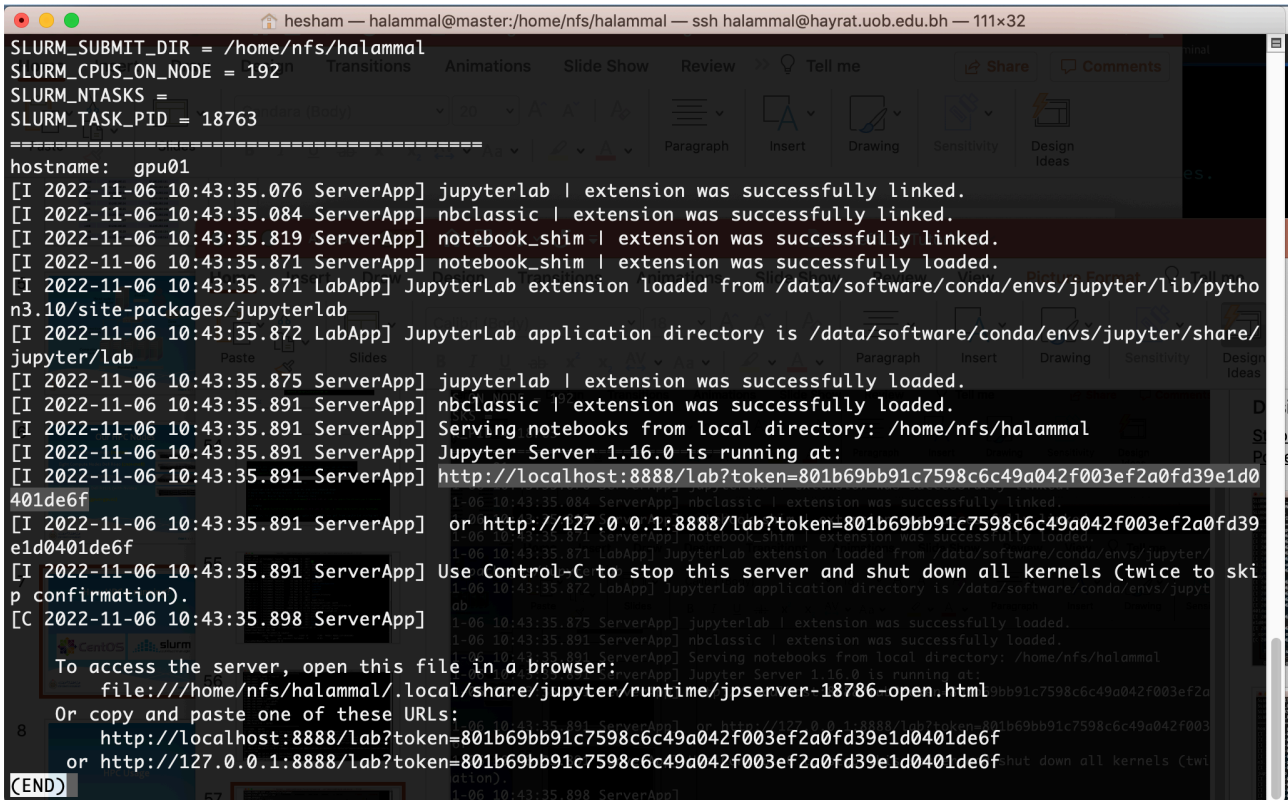
conda activate /data/software/conda/envs/jupyter # Activate the environment
jupyter-lab --no-browser --port 8888 # Run the server on port 8888
```

Step 3. Check the contents of `jupyter_config.out` to get the server details.

```
# Run the batch file, refer to the file for more details
sbatch launch_jupyterlab.sh
```

```
# Check the content of jupyter_config.out to get your server details.
# Take note of the hostname and the server URL, and port number.
cat jupyterlab_config.out
```

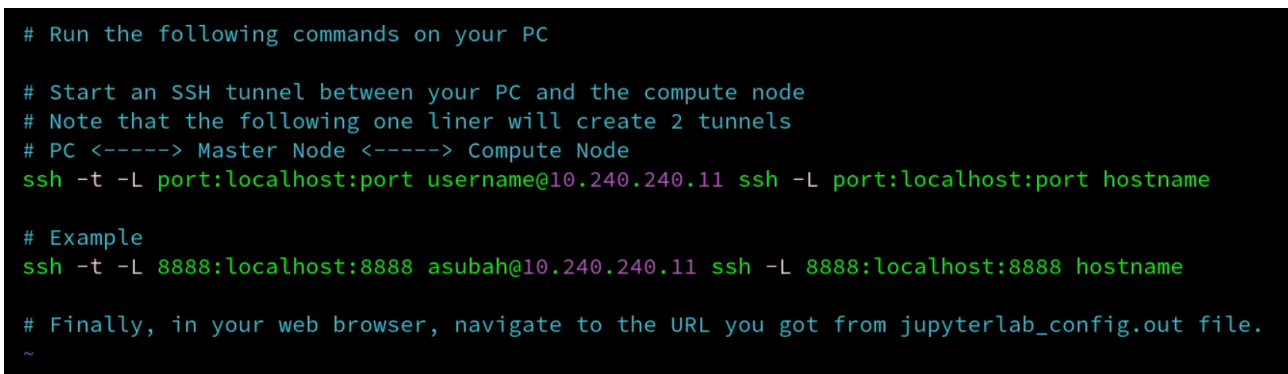
Copy the URL you find in the output file.



```
SLURM_SUBMIT_DIR = /home/nfs/halammal
SLURM_CPUS_ON_NODE = 192
SLURM_NTASKS =
SLURM_TASK_PID = 18763
=====
hostname: gpu01
[I 2022-11-06 10:43:35.076 ServerApp] jupyterlab | extension was successfully linked.
[I 2022-11-06 10:43:35.084 ServerApp] nbclassic | extension was successfully linked.
[I 2022-11-06 10:43:35.819 ServerApp] notebook_shim | extension was successfully linked.
[I 2022-11-06 10:43:35.871 ServerApp] notebook_shim | extension was successfully loaded.
[I 2022-11-06 10:43:35.871 LabApp] JupyterLab extension loaded from /data/software/conda/envs/jupyter/lib/python3.10/site-packages/jupyterlab
[I 2022-11-06 10:43:35.872 LabApp] JupyterLab application directory is /data/software/conda/envs/jupyter/share/jupyter/lab
[I 2022-11-06 10:43:35.875 ServerApp] jupyterlab | extension was successfully loaded.
[I 2022-11-06 10:43:35.891 ServerApp] nbclassic | extension was successfully loaded.
[I 2022-11-06 10:43:35.891 ServerApp] Serving notebooks from local directory: /home/nfs/halammal
[I 2022-11-06 10:43:35.891 ServerApp] Jupyter Server 1.16.0 is running at:
[I 2022-11-06 10:43:35.891 ServerApp] http://localhost:8888/lab?token=801b69bb91c7598c6c49a042f003ef2a0fd39e1d0401de6f
or http://127.0.0.1:8888/lab?token=801b69bb91c7598c6c49a042f003ef2a0fd39e1d0401de6f
[I 2022-11-06 10:43:35.891 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2022-11-06 10:43:35.898 ServerApp]

To access the server, open this file in a browser:
file:///home/nfs/halammal/.local/share/jupyter/runtime/jpserver-18786-open.html
Or copy and paste one of these URLs:
http://localhost:8888/lab?token=801b69bb91c7598c6c49a042f003ef2a0fd39e1d0401de6f
or http://127.0.0.1:8888/lab?token=801b69bb91c7598c6c49a042f003ef2a0fd39e1d0401de6f
(END)
```

Step 4. On your laptop or PC run the following ssh command to create an ssh tunnel. Enter the password for the Benefit AI Lab cluster when prompted (substitute the “asubah” with your account id).



```
# Run the following commands on your PC

# Start an SSH tunnel between your PC and the compute node
# Note that the following one liner will create 2 tunnels
# PC <----> Master Node <----> Compute Node
ssh -t -L port:localhost:port username@10.240.240.11 ssh -L port:localhost:port hostname

# Example
ssh -t -L 8888:localhost:8888 asubah@10.240.240.11 ssh -L 8888:localhost:8888 hostname

# Finally, in your web browser, navigate to the URL you got from jupyterlab_config.out file.
~
```

Step 5. Copy the URL into a browser on your laptop and Jupyter Lab should appear.