GPU Cluster setup

Ajith, Abhishek & Patanjali

Outline

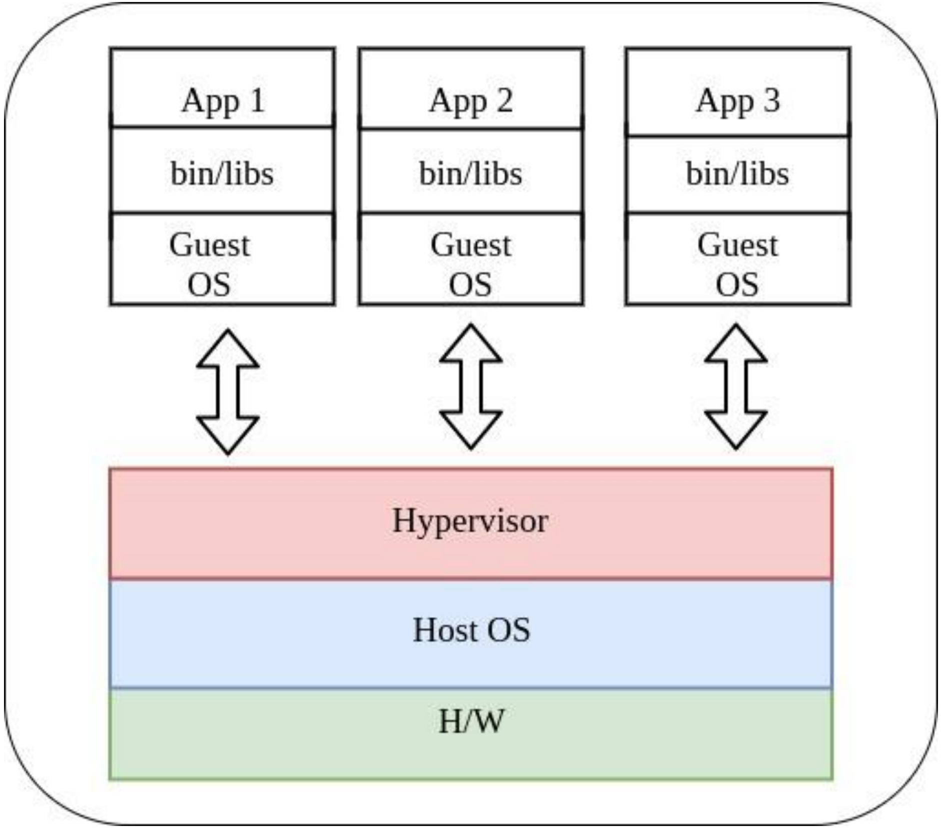
- Overview of cluster manager.
- Accessing the cluster
- Job Management
- Libra Tutorial
- AWS setup

Overview: Why cluster manager?

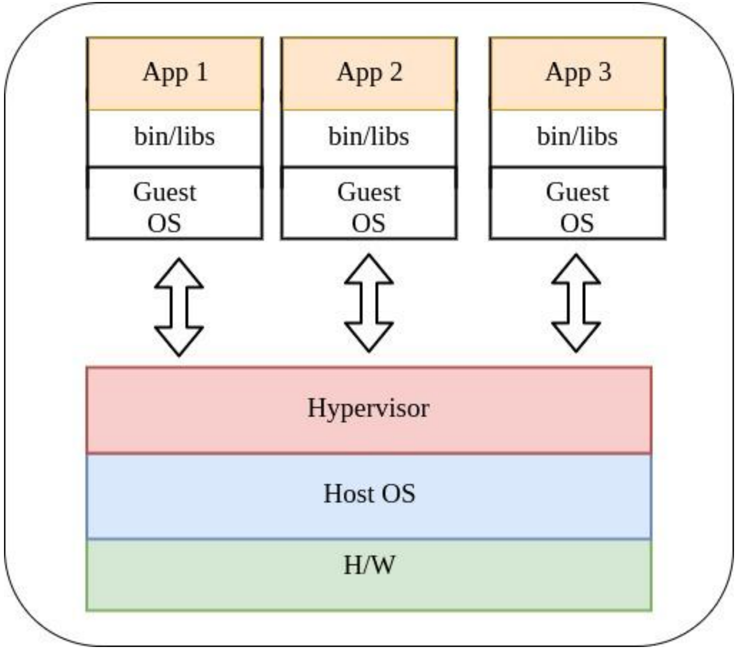
- Different applications, different requirements.
- Need an envt where all can be deployed seamlessly and interact with each other if necessary.
- Typical scenario for an application:
    - What the User wants: More features, zero downtime.
    - What the Developer wants: Needs time to update the application, to debug, keep the application safe from buggy OS updates.
- Can we manage this scenario seamlessly i.e., handle both constraints ?

Option 1: Use a VM

- Guest OS needed.
- Hypervisor support required.
- Too much overhead for one single application.
- If the underlying resource goes down, manual intervention is required.



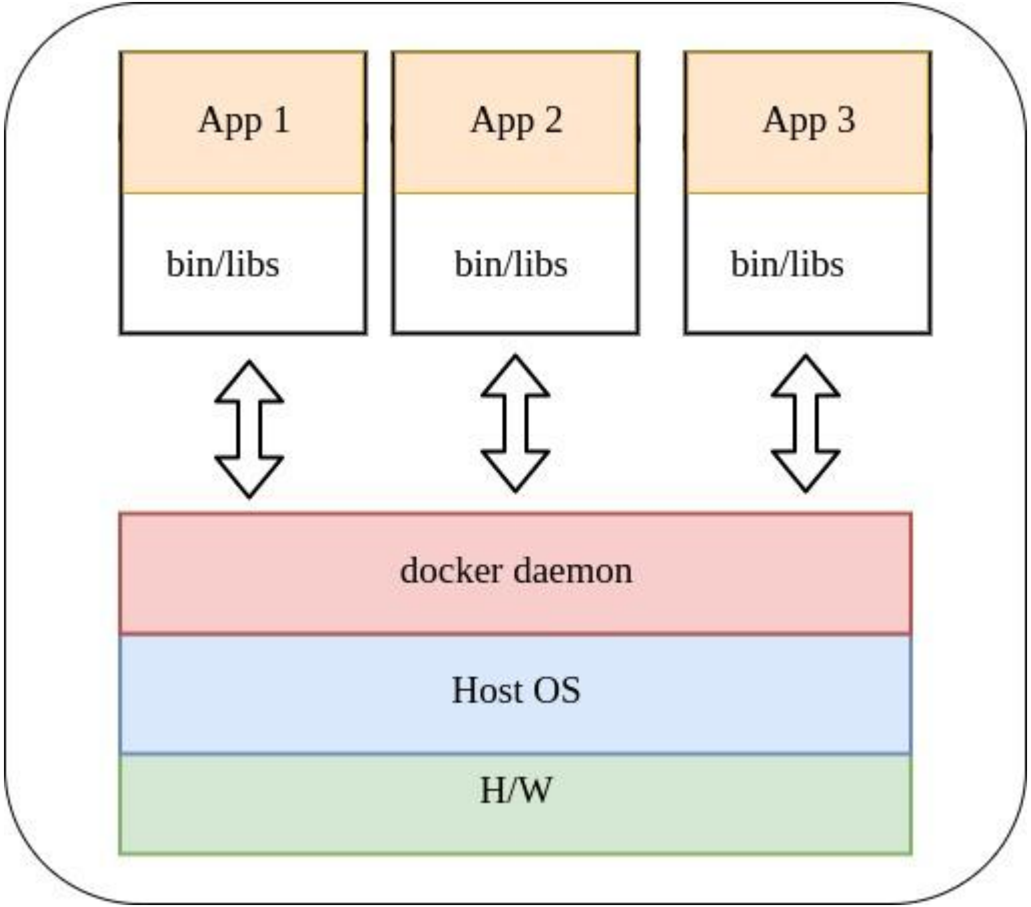
What we wanted



Solution: Kubernetes.

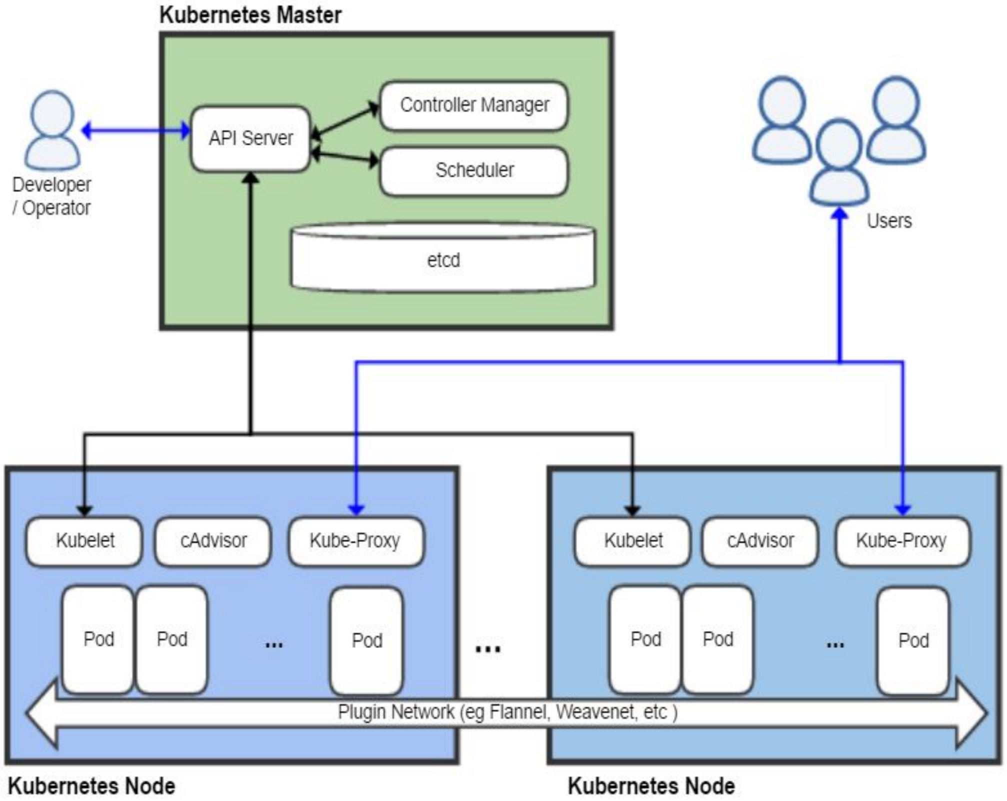
Kubernetes

- Lightweight: No Guest OS, no hypervisor.
- Portable: The application is self contained and hence can be scaled and deployed easily.



Overview: Kubernetes

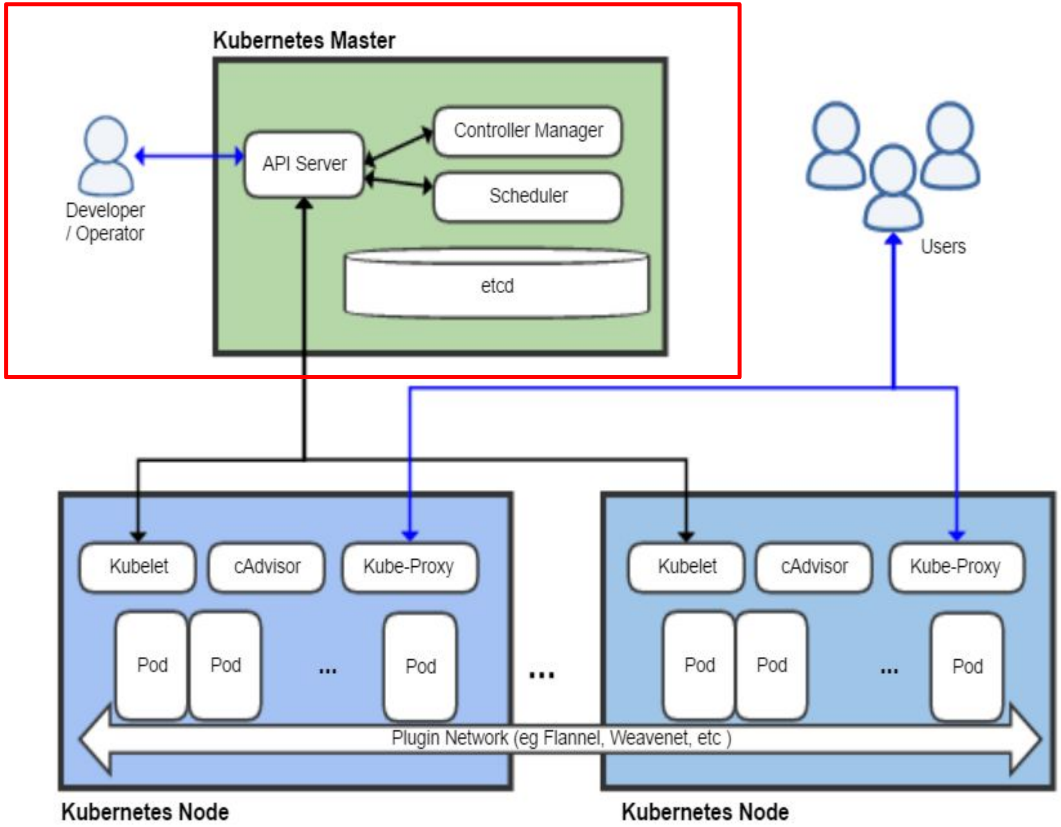
- A typical Kubernetes cluster has the following 3 components
  - Master
  - Node
  - Pod



—

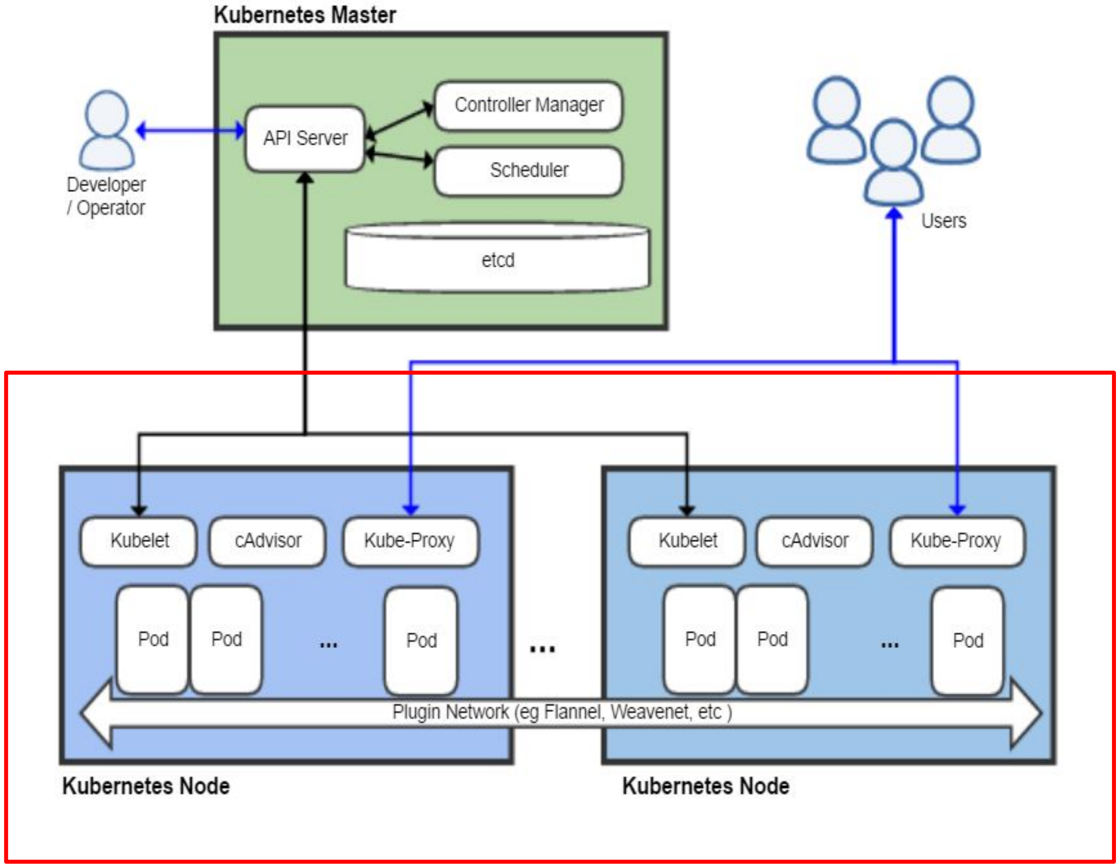
Overview: Kubernetes

- Master:
  - Maintains the cluster.
  - Schedules jobs on the cluster.



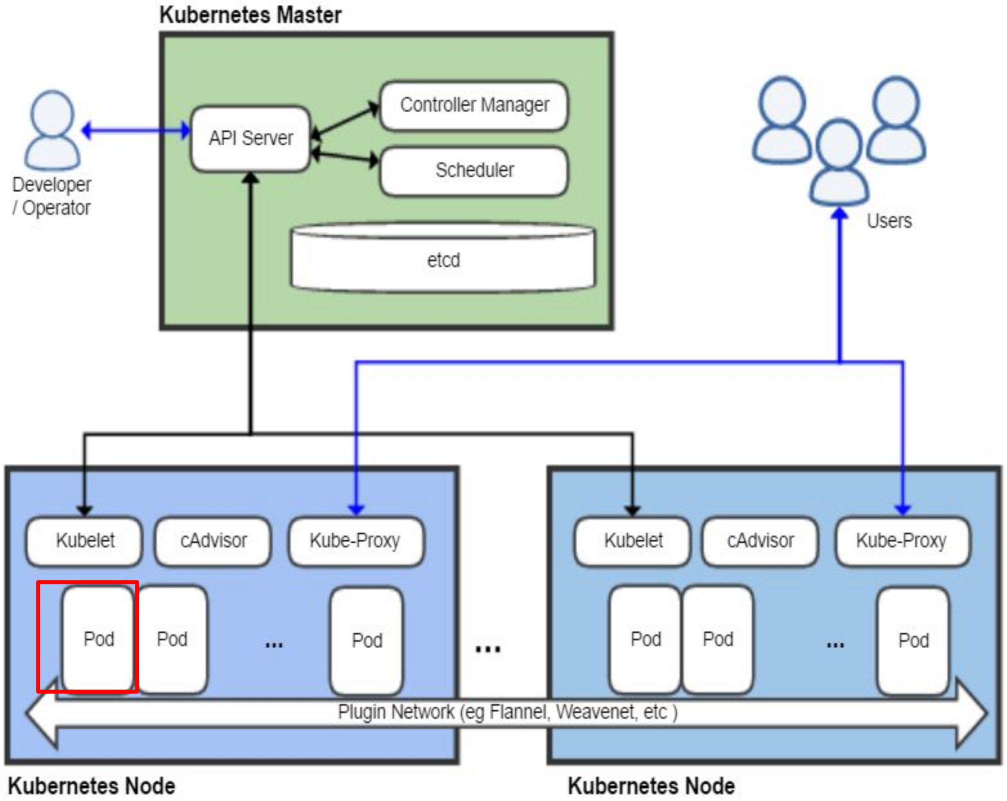
Overview: Kubernetes

- Node:
    - Where the jobs actually run.
    - Interacts with the master through a service called **kubelet**

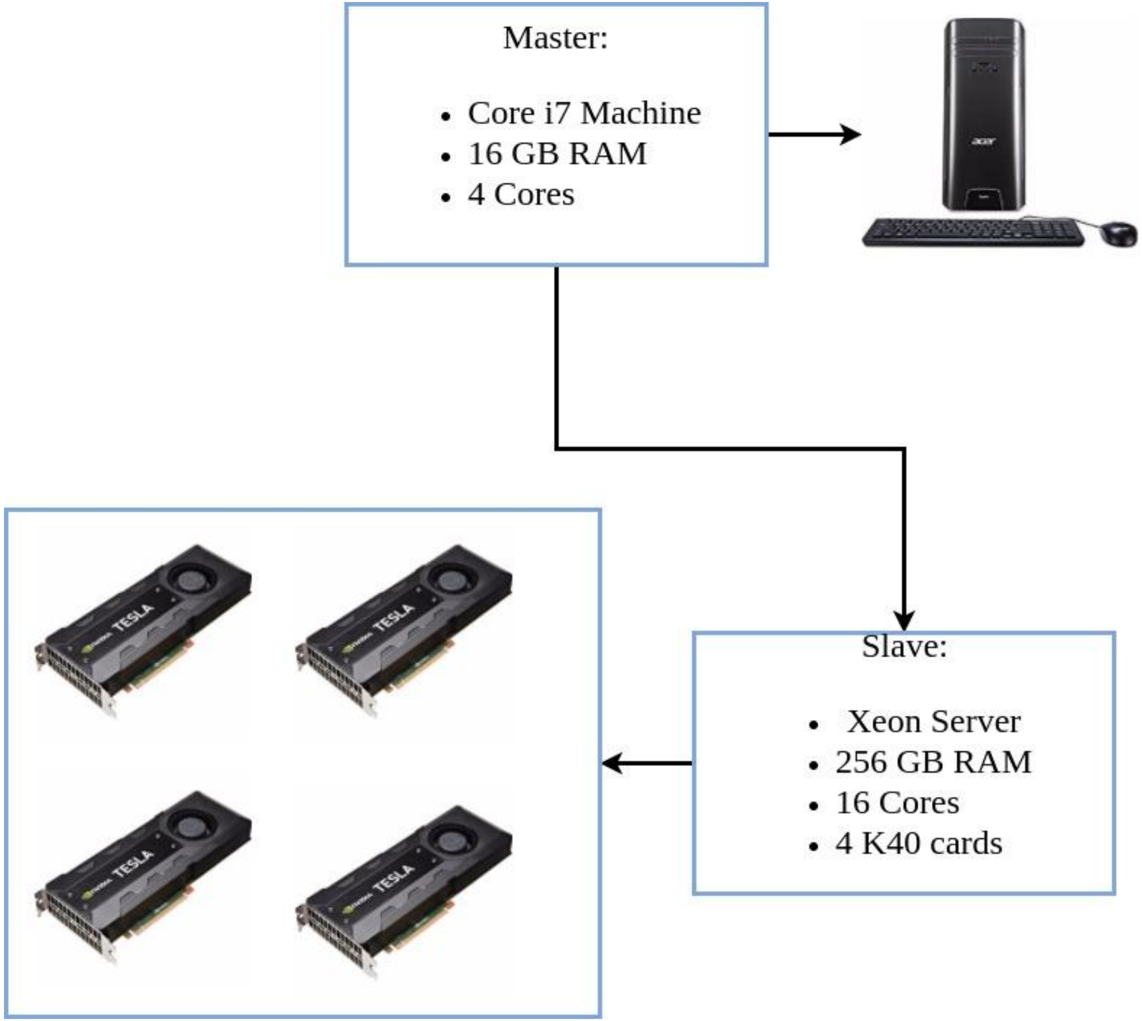


Overview: Kubernetes

- Pod:
  - Jobs are referred to as containers.
  - Container is a single entity that comprises of the application and all its dependencies.
  - A pod is a collection of containers.
  - A Pod runs on a Node.
  - If Node goes down, another similar node is automatically reallocated.



Our Cluster



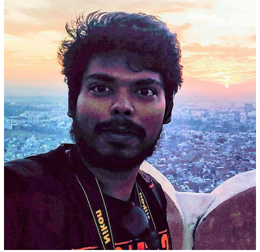
Master:

- Core i7 Machine
- 16 GB RAM
- 4 Cores

Slave:

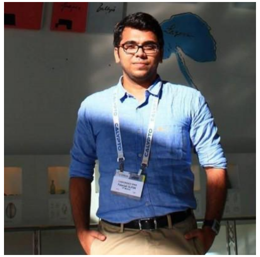
- Xeon Server
- 256 GB RAM
- 16 Cores
- 4 K40 cards

Meet the Cluster Team



Ajith Kumar
ajith@cse.iitm.ac.in

Abhishek Chakraborty
abhic@cse.iitm.ac.in



Patanjali SLPSK
slpskp@cse.iitm.ac.in

TA hours: Thursday 10:00-10:50 am.

Basic Information

1. Every user has to create an account on the cluster.

   Fill the form: https://goo.gl/forms/OSL4zv5DvLs3w6l22

2. You have **time limits** on the cluster.

Accessing the cluster

- **Linux / macOS**

```
>ssh username@10.21.230.1
```

- **Windows**

  Try **PuTTY** & **WinSCP**

You will now have access to the master node of the cluster.

Job Management

1. You could either write your source code in the master node itself or copy your source code from your computer to the master node using **scp.**
2. Compile the source code using **nvcc.**
3. Submit the executable as a pod to the cluster. (How?)

Basic cluster commands

- **`gsub`** - to submit a job

```
user3@gpumaster-machine:~$ gsub job.sh

pod "user3-pod" created
```

- **`gstat`** - to view the status of the job

```
user3@gpumaster-machine:~$ gstat

NAME         READY        STATUS       RESTARTS     AGE

user3-pod    0/1          Completed    0            3h
```

job.sh

```
echo "hello world"
```

Basic cluster commands (contd.)

- **`gdel`** - to delete a job

```
user3@gpumaster-machine:~$ gdel

pod "user3-pod" deleted
```

- **`gtime`** - to view the remaining quota

```
user3@gpumaster-machine:~$ gtime

Remaining quota : 18368s
```

A few important notes

- Only one pod per user will be running at any time.
- If you want to redirect the output of your program to a file, make sure you write to `/home/<username>/<your_dir_or_file>`
- ‘`gstat -o`’ will show the output of your program if not redirected to a file
- If (and only if) ‘`gdel`’ doesn’t kill the pod, use ‘`-f`’ flag with ‘`gdel`’ to force kill the pod.
- A template ‘job.sh’ and CUDA program will be put to every user’s home, as an example.

Viewing all running pods

- A web interface is setup to view the status of all running pods on the cluster.
- Available at: 10.21.230.1:6277
- Can be viewed from any web browser.

Script to submit jobs on GNR cluster

```
#!/bin/bash
#PBS -o outputfile.log
#PBS -e errorfile.err
#PBS -l cput=800:00:00
mkdir -p $HOME/jobs
tpdir=`echo $PBS_JOBID | cut -f 1 -d .`
tempdir=$HOME/work/job$tpdir
mkdir -p $tempdir
cd $tempdir
cp -R $PBS_O_WORKDIR/* .
make
./transpose
mv ../job$tpdir $HOME/jobs/.
```

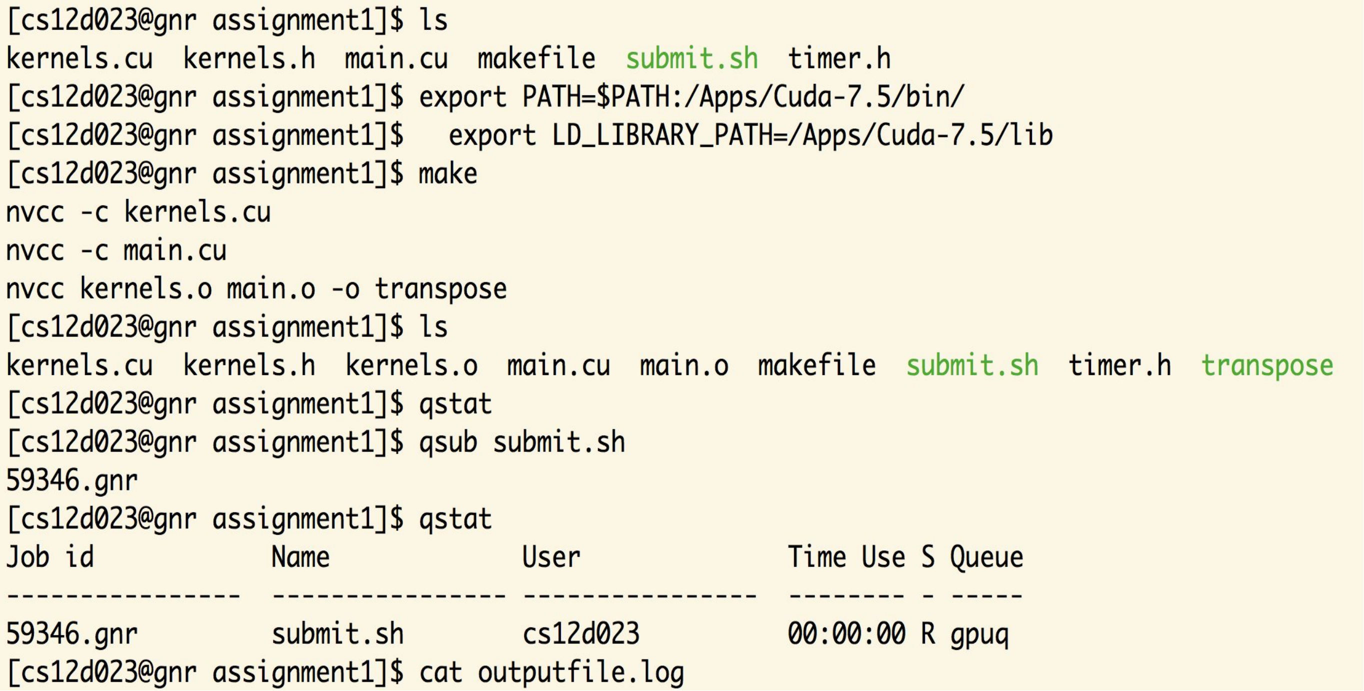
PBS Directives

Portable Batch System (PBS)

Using the PBS job scheduler:

- **qsub: Submit a job**
- **qdel: Delete a batch job**
- **qstat: Show status of batch**

Running your assignments on GNR



```
[cs12d023@gnr assignment1]$ ls
kernels.cu  kernels.h  main.cu  makefile  submit.sh  timer.h
[cs12d023@gnr assignment1]$ export PATH=$PATH:/Apps/Cuda-7.5/bin/
[cs12d023@gnr assignment1]$  export LD_LIBRARY_PATH=/Apps/Cuda-7.5/lib
[cs12d023@gnr assignment1]$ make
nvcc -c kernels.cu
nvcc -c main.cu
nvcc kernels.o main.o -o transpose
[cs12d023@gnr assignment1]$ ls
kernels.cu  kernels.h  kernels.o  main.cu  main.o  makefile  submit.sh  timer.h  transpose
[cs12d023@gnr assignment1]$ qstat
[cs12d023@gnr assignment1]$ qsub submit.sh
59346.gnr
[cs12d023@gnr assignment1]$ qstat
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
59346.gnr        submit.sh        cs12d023         00:00:00 R gpuq
[cs12d023@gnr assignment1]$ cat outputfile.log
```

Thank You!