How to request GPU and CPU resources

To understand how our cluster is configured

- **To request GPU or GPU nodes you must first understand how your cluster is configured**, for this reason we recommend that you refer to the home page where we have a cluster description that may help you.
- Basically there are 2 architectures: ***icelake*** (GPU nodes) and ***sandy*** (CPU nodes).
- Based on these architectures, the resources within the cluster are requested.

Mainly, there are 2 ways to specify resource to use when submitting a job using ***constraints*** and using ***gres***.

Slurm *Constrains* at TeideHPC

Nodes can have **features** assigned to them by the TeideHPC administrators. Users can specify which of these features are required by their job using the **--constraint** option. **Only nodes having features matching the job constraints will be used to satisfy the request.**

For a more detailed explanation of what constrains are you can see the official slurm documentation

To find out list of constrains at TeideHPC look at the column *AVAIL FEATURES* after execute this command.

```
sinfo -o "%40N  %10c  %10m  %35f  %30G "
```

```
NODELIST                                CPUS       MEMORY     AVAIL_FEATURES                      GRES
node18109-1                             64         257214     ilk,gpu,a100                        gpu:a100:8
node2204-[3-4]                          20         31906      ivy                                 (null)
node17109-1,node17110-1,node18110-1,node  64         257214     ilk,viz,t4                          gpu:t4:1
node0303-2,node0304-[1-4],node1301-[1-4]  16         30000+     sandy                               (null)
node17102-1                             64         257214     ilk,gpu,a100,3g.20gb,2g.10gb,1g.5gb gpu:3g.20gb:1(S:0),gpu:2g.10gb
node17101-1,node17103-1,node17104-1,node  64         257214
ilk,gpu,a100                        gpu:a100:4(S:0-1)
```

constraints	Type	Definition
ilk	nodes architecture	Allows you to select an icelake type node (they have GPU)
sandy	nodes architecture	Allows you to select a computing node with sandy-bridge architecture
ivy	nodes architecture	Allows you to select a computing node with ivy-bridge architecture
viz	display node	Allows you to select a display-specialized GPU node (Nodes with Nvidia T4)
gpu	gpu nodes	Allows you to select a gpu node (Nodes with NVidia A100)
a100	gpu model	Allows you to select a gpu node with NVidia A100 directly
t4	gpu model	Allows you to select a gpu node with NVidia T4 directly

Constraints usage example.

The following commands in interactive session have their equivalent in *batch*. You only need to write this directive with `#SBATCH --constraint=<constraint>` in your scripts.

- For a computing node with sandy architecture (16 cores | 32/64 GB).

```
salloc -n 1 --cpus-per-task 16 -p express --mem 29000 --constraint=sandy
```

- For a compute node with icelake architecture (64 cores | 256 GB).

```
salloc -n 1 --cpus-per-task 8 -p express --mem 10000 --constraint=ilk
```

- For a GPU node (icelake) with one GPU (Nvidia A100).

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --constraint=gpu[,a100]
```

- For a display-specialized GPU node (Nvidia Tesla T4)

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --constraint=viz[,t4]
```

Generic Resource *GRES*

In Slurm, **GRES stands for Generic Resource**. GRES is a feature that allows you to specify and manage various types of generic resources such as GPUs (Graphics Processing Units) within a computing cluster.

Slurm's GRES functionality enables efficient allocation, scheduling, and tracking of these resources for jobs submitted to the cluster. It helps ensure that the requested resources are available and properly utilized by the jobs that require them.

To use GRES effectively, you need to understand how your cluster is configured, how available GRES types and quantities are defined and specify GRES requirements when submitting jobs.

To get type of GRES defined in the cluster you can use:

```
scontrol show config | grep Gres
```

```
GresTypes = gpu
```

To find out list of *GRES* at TeideHPC look at the column *GRES* after execute this command.

```
sinfo -o "%40N  %10c  %10m  %35f  %30G"
```

```
NODELIST                            CPUS       MEMORY     AVAIL_FEATURES                      GRES
node18109-1                         64         257214     ilk,gpu,a100                        gpu:a100:8
node2204-[3-4]                      20         31906      ivy                                 (null)
node17109-1,node17110-1,node18110-1,node  64         257214     ilk,viz,t4                          gpu:t4:1
node0303-2,node0304-[1-4],node1301-[1-4]  16         30000+     sandy                               (null)
node17102-1                         64         257214     ilk,gpu,a100,3g.20gb,2g.10gb,1g.5gb gpu:3g.20gb:1(S:0),gpu:2g.10gb
node17101-1,node17103-1,node17104-1,node  64         257214
ilk,gpu,a100                        gpu:a100:4(S:0-1)
```

Alternatively, it can also be viewed by listing and filtering the nodes:

```
scontrol show nodes | egrep "NodeName|gres"
```

```
....
NodeName=node1315-4 Arch=x86_64 CoresPerSocket=8
NodeName=node2204-3 Arch=x86_64 CoresPerSocket=10
...
NodeName=node17101-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
NodeName=node17102-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=7,gres/gpu:1g.5gb=2,gres/gpu:
2g.10gb=1,gres/gpu:3g.20gb=1,gres/gpu:a100=3
NodeName=node17103-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
NodeName=node17104-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
...
```

As you can see, each node in cluster may have a different definition. For example this node has 4 GPUs NVidia A100.

```
NodeName=node17104-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=4,gres/gpu:a100=4
```

and this node has 3 NVidia A100 and 1 partitioned GPU (Nvidia A100).

```
NodeName=node17102-1 Arch=x86_64 CoresPerSocket=32
   CfgTRES=cpu=64,mem=257214M,billing=64,gres/gpu=7,gres/gpu:1g.5gb=2,gres/gpu:
2g.10gb=1,gres/gpu:3g.20gb=1,gres/gpu:a100=3
```

To understand, what is the meaning of `gres/gpu:1g.5gb=2,gres/gpu:2g.10gb=1,gres/gpu:3g.20gb=1,gres/gpu:a100=3` you must understand what is MIG

GRES usage example.

- For a GPU node (icelake) with one GPU (Nvidia A100).

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --gres=gpu:a100=1
```

- For a MIG partition of GPU (Nvidia A100) .

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --gres=gpu:2g.10gb=1
```

- For a display-specialized GPU (Nvidia Tesla T4)

```
salloc -n 1 --cpus-per-task 8 -p express --mem 8000 --gres=gpu:t4=1
```