

# Run docker-based workload on HPC with GPU

In this case study, we will walk thru how to convert a docker image into singularity format and import it into the cluster, how to look up appropriate hardware, and finally enqueue a job.

Due to security concerns, OAsis HPC supports Singularity rather than Docker. But you may convert docker images easily using command line statements.

## Convert a docker image to Singularity

If you use containers other than Docker and Singularity, please consult [this page](#) for details.

To communicate with our GPU, your container should have CUDA. Version 11.6 is recommended. Besides packaging CUDA from scratch, you may also extend the built-in image `nvhpc.22.9-devel-cuda_multi-ubuntu20.04.sif` in `/pfss/containers`. It has lots of GPU libraries and utilities pre-built.

We recommend putting your containers in the `containers` folder under one of the `scratch` folders so you can browse inside the portal. Following is an example of converting the `julia` docker image to Singularity to put in the `home containers` folder.

```
singularity pull julia.1.8.2.sif docker://julia:alpine3.16

# move it to the containers folder, then we can run it in the web portal
mkdir -p ~/containers
mv julia.1.8.2.sif ~/containers
```

## Explore available GPUs

There may be more than one GPU available for your account. Head to the [partitions](#) page to look up the best partition for your job.

Find by job requirement

#nodes  #cpu cores per node  #memory per Node (GB)

#gpu instances per node

1 - 2 of 2 < Prev Next >

Name	State	Idle CPU	Idle Memory	GPU	Nodes	Jobs
batch	UP	384 / 384 (100.00%)	1545 / 1545 GB (100.00%)	N/A	3	0 / 0
gpu	UP	32 / 32 (100.00%)	515 / 515 GB (100.00%)	6 / 6 (100.00%)	1	0 / 0

You may also click on the nodes count to check the nodes under this partition. In the following example, only one node is currently available in the **gpu** partition. That node has six idle GPUs ready for use.

a100 is referring to the NVIDIA A100 80GB GPU. 1g.10gb and 3g.40gb are MIG partitions.

Nodes of gpu

1 - 1 of 1 < Prev Next >

Arch	Hostname	Cores	Inuse	Memory	Alloc	Gpu	State
x86_64	oaklbhpcgpu001	32	0	515 GB	0 GB	1g.10gb:0/4 3g.40gb:0/1 a100:0/1	IDLE

## Enqueue job

There are multiple ways in OAsis to enqueue such a container-based job leveraging GPUs. Here we allocate **two 1g.10gb** GPUs and run **nvidia-smi** command on the built-in **nvhpc** container to inspect the allocated GPUs. It should detect two GPU devices and print them out to your console or the log file.

## Enqueue and wait for the output (srun)

```
srun -p gpu --gpus 1g.10gb:2 \  
singularity exec --nv \  
/pfss/containers/nvhpc.22.9-devel-cuda_multi-ubuntu20.04.sif \  
/bin/sh -c nvidia-smi -L
```

## Enqueue and redirect stdout to a file (sbatch)

Create a **myjob.sbatch** text file with the content below:

```
#!/usr/bin/env bash  
  
#SBATCH -p gpu  
#SBATCH --gpus 1g.10gb:2  
  
singularity exec --nv \  
/pfss/containers/nvhpc.22.9-devel-cuda_multi-ubuntu20.04.sif \  
/bin/sh -c nvidia-smi -L
```

Then enqueue your job with the following command:

```
sbatch myjob.sbatch
```

## Run with the quick job GUI

Besides using commands, you may also enqueue container-based jobs in our web portal. Click **Jobs** then **Run Containers** to open the quick job launcher GUI. Select a partition, the preferred CPU cores, memory, GPUs, the container, and the command you want to run.

## Enqueue job



Path: /pfss/toolkit/run\_container.sbatch

Run a container for your workload.

Node\*

Partition: gpu  
Node: oaklbhpcgpu001  
CPU: 4 cores  
Memory: 8 GB  
GPU: 1g.10gb x 2

[Select another](#)

Container\*

/pfss/containers/nvhpc.22.9-devel-cuda\_multi-ubuntu20.04.sif

Command\*

nvidia-smi -L

Output path\*

container.out

Enqueue now

[Download this file](#)  
[Toggle file content](#)

Revision #20

Created 1 February 2023 03:00:57 by Loki Ng

Updated 2 February 2023 15:49:26 by Loki Ng