ASPIRE2A+ General QuickStart Guide

National Supercomputing Centre

Table of Contents

1. Introduction 2
2. User Policy 2
3. ASPIRE2A+ Overview 3
4. Direct Access to ASPIRE2A+ 3
5. VPN Access to ASPIRE2A+ 4
6. Software Environment 5
7. Job Submission 6
8. Need Help? 11
9. Frequently Asked Questions (FAQs) 11

Appendix A (Creating container image outside of 2A+) 15

1. Introduction

This guide contains information on how to access and start submitting jobs on **ASPIRE2A+**. Please take note of the following:

1. **The ASPIRE 2A+ system is being provided to the research community for early access in its current state. While we are excited to offer this opportunity, we seek your understanding that the system may not meet some of your expectations, as the NSCC Systems team is working closely with vendors to fine-tune the system.**
2. **User may experience system design complications, implementation, or operational issues that could lead to system failures, miscalculations, or data loss during this phase.**
3. **Support from the Managed Services Team may also be limited during this early use period, including available training and documentation. However, we are committed to assisting you as much as possible and will strive to address any challenges or needs that arise during your trial use of the system.**

2. User Policy

- All computational jobs must be submitted and run via the workload manager/scheduler.
- No computational jobs are allowed on the login nodes.
- Use the project directory assigned (**/data/projects/<Project ID>/)** for sharing and storing project-specific data (dataset, database, results, input-output, etc.) and for installing controlled/licensed or project-specific applications.
- Users are welcome to install applications/software under their own `$HOME` or project directory.
- Directories are ACL-controlled. If you are unable to access the directory, please send an email to help@nscc.sg.
- Project members can access the project directory based on project group permission (each project directory will be assigned to a group with the same name as the project ID). Project owners are given full access and will be able to adjust group permission within the project folder.
- Users are responsible for the management and control of their own data. NSCC accepts no liability for any data loss or unauthorized data access.
- All terms of use are subject to the prevailing **NSCC Acceptable Use Policy** and **NSCC Data Management and Retention Policy**

3. ASPIRE2A+ Overview

ASPIRE2A+ Supercomputer is powered by an **NVIDIA DGX SuperPOD™ system consisting of 40 DGX H100 nodes.** It is an AI powerhouse that features the groundbreaking NVIDIA H100 Tensor Core GPU. It comes with a high-performance parallel file systems solution that provides 2.5PB usable capacity on NVMe for scratch storage and 27.5PB usable capacity of home storage on DDN Lustre File storage and NVIDIA InfiniBand high-speed interconnect.

3.1 NVIDIA DGX H100 SuperPOD Compute Node Specification

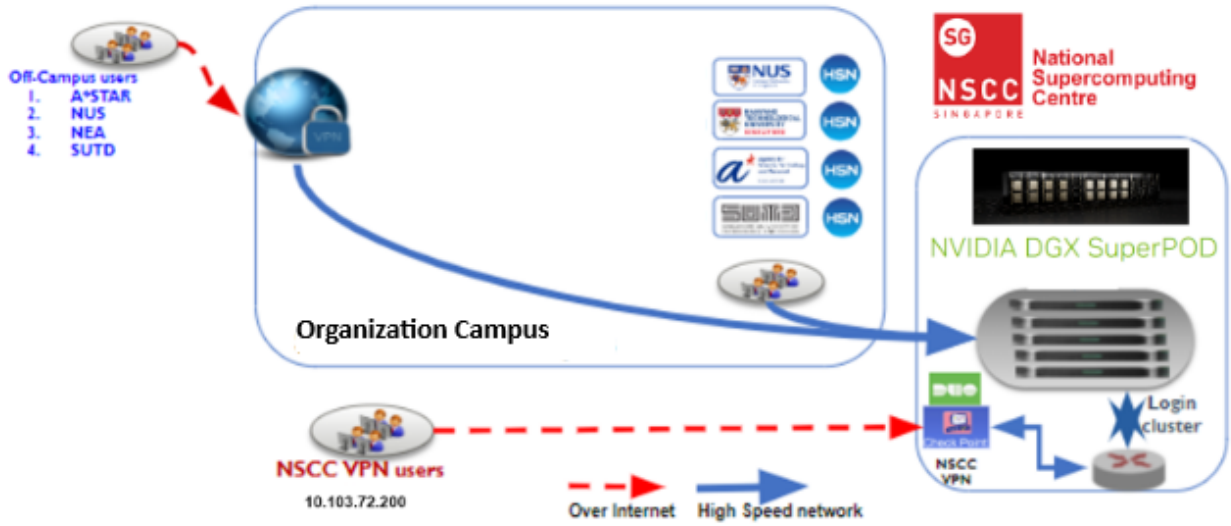
Compute Nodes	CPU Model	System Memory	GPU	Storage
NVIDIA DGX SuperPOD 40 Nodes of DGX H100	Dual Intel® Xeon® Platinum 8480C Processors Total Cores=2 x 56 cores = 112 Cores	2 TB	8 x NVIDIA H100 GPUs Memory 8 x 80GB = 640 GB (80 GB on each GPU card)	8 x 3.84 TB NVMe drives

4. Direct Access to ASPIRE2A+

All users from the following organizations are recommended to connect via high-speed access link to access the system as shown below.

Organization Direct	Domain name of Login Node	If connecting from outside campus network
NUS	aspire2p.nus.edu.sg	Connect to NUS VPN first
NTU	aspire2pntu.nscc.sg	Connect to NTU VPN first
A*STAR	aspire2p.a-star.edu.sg	Connect to A*STAR VPN first
SUTD	aspire2p.sutd.edu.sg	Connect to SUTD VPN first

For example, users from NUS can access the login node using SSH client via their campus. Refer to the diagram below.



5. VPN Access to ASPIRE2A+

All other users are required to connect via the NSCC VPN to access the system. Please follow the steps below to gain access.

Step 1: Users are required to enroll in DUO 2FA. Users will receive an instruction email from DUO to install the DUO mobile app on their mobile phones. Once installed, users may proceed to connect to the VPN.

Please refer to the following guide for the DUO 2FA enrollment steps.

DUO 2FA Enrollment Guide

Note: User MUST complete Step 1 before proceeding to Step 2 to successfully connect to the VPN.

Step 2: Set up and connect to NSCC VPN

Please refer to the following guides for the setup and connection of the VPN for different Operating Systems:

ASPIRE2A+ VPN for Window

ASPIRE2A+ VPN for MAC

ASPIRE2A+ VPN for Linux

Note: Please check your 2FA mobile device pop-up request for the 2FA approval while logging into the VPN using the ASPIRE2A+ credentials. **VPN connection would fail without the DUO**

2FA approval.

Step 3: Access the system using SSH

Login node FQDN: `aspire2p.nscc.sg`

```
$ ssh -Y <userid>@aspire2p.nscc.sg
```

Upon successful login, you will be presented with a screen similar to Figure 1 displaying the current status of the system.

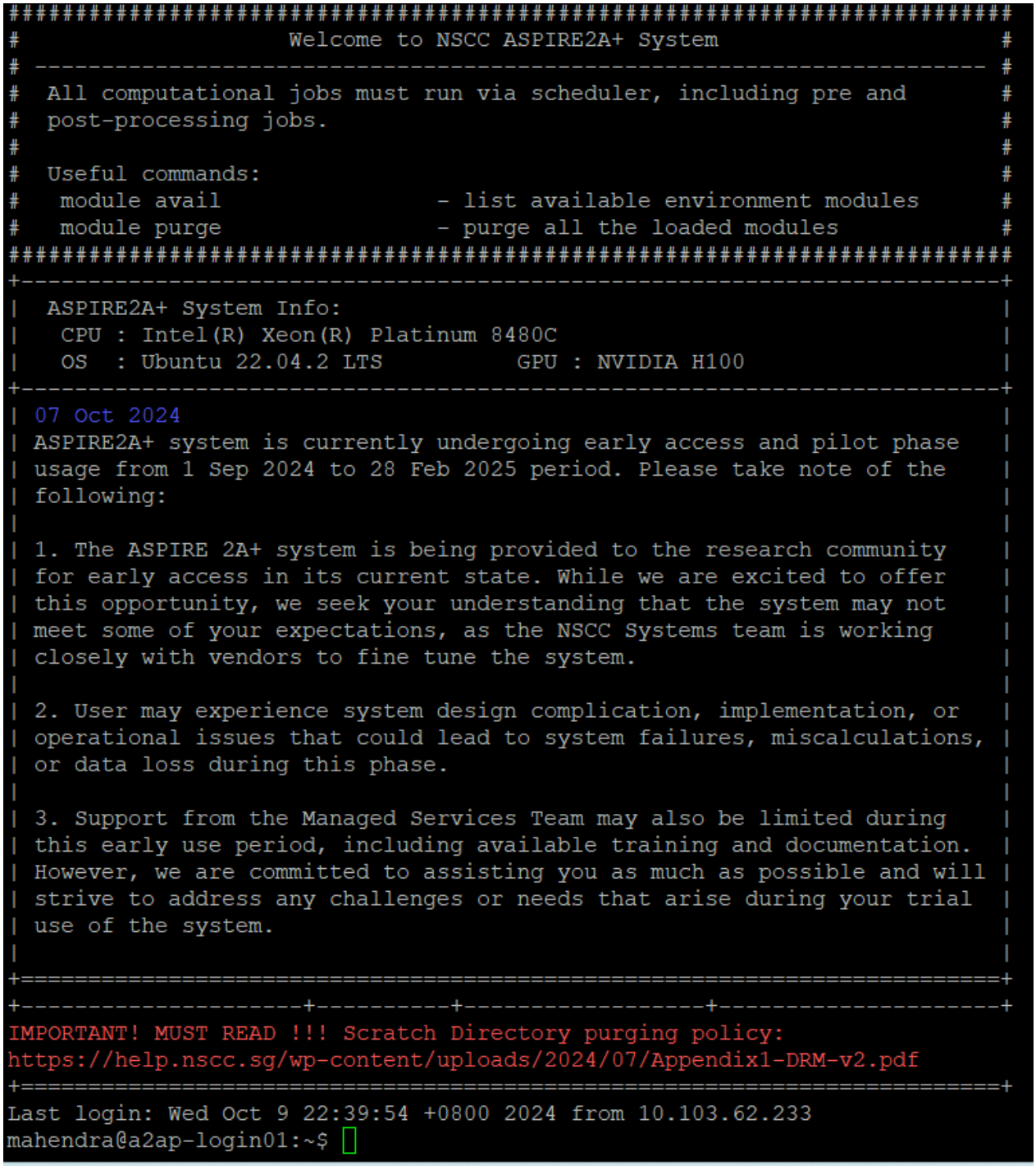
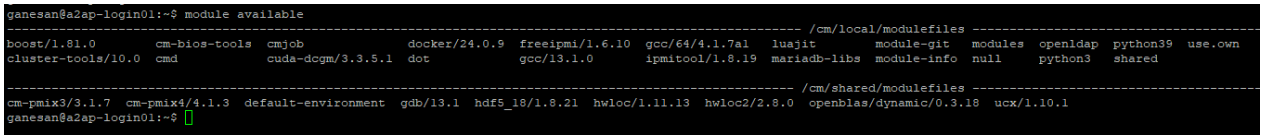


Figure 1: Successful login to ASPIRE2A+ login node

6. Software Environment

ASPIRE2A+ uses the Ubuntu 22.04.2 LTS as its operating system (OS). Users can list the available modules by typing `module avail`.

Figure 2 shows the available modules on ASPIRE2A+.



Although you can run your job in a bare-metal environment, we highly recommend using enroot containers. If you choose to use the bare-metal environment, you will need to install all the required software by yourself. Since we do not provide root permissions to users, you will need to build packages from scratch or set all the required paths to your own directory. Using containers can greatly simplify these processes and improve efficiency and reliability

Enroot

Enroot is a simple, yet powerful tool to turn traditional container/OS images into unprivileged sandboxes. Enroot can be thought of as an enhanced unprivileged chroot. It uses the same underlying technologies as containers but removes much of the isolation they inherently provide while preserving filesystem separation. This approach is generally preferred in high-performance environments or virtualized environments where portability and reproducibility is important, but extra isolation is not warranted.

Enroot is also similar to other tools like Proot or Fakeroot but instead relies on more recent features from the Linux kernel (i.e. user and mount namespaces), and provides facilities to import well-known container image formats (e.g. Docker).

7. Job Submission

This guide provides instructions for users to submit jobs to ASPIRE2A+ using PBS pro scheduler and details on how to schedule Enroot container workloads with various options. Users will learn how to specify Enroot-related resources in their job scripts to leverage the Enroot adapter features.

7.1 To submit an interactive job:

```
$ qsub -I -q normal -l select=1:ncpus=112:ngpus=8:mem=1887gb -l
walltime=12:00:00 -P <project_ID>
```

Interactive jobs can be requested using the “-I” option of qsub. When the resources are available, the job is started and, user’s terminal input and output are connected to the job

similarly to a login session. It appears that the user is logged into one of the available execution machines, and the resources requested by the job are reserved for that job. This is useful for debugging applications or for computational steering.

7.2 To submit a batch job:

Here are some examples of job scripts utilizing the PBS Pro Enroot adapter. Users are required to create a job script file and submit using qsub command as follows:

```
$ qsub myjobscript.sh
```

7.2 (a) Single node job script:

```
#!/bin/bash
#PBS -N pytorch_dp_job

#PBS -l select=1:ncpus=28:ngpus=2:mem=470gb:container_engine=enroot
#PBS -l walltime=24:00:00
#PBS -q normal
#PBS -P <project_id>
#PBS -j oe
#PBS -l container_image=~/images/nvidia+pytorch+23.10-py3.sqsh
#PBS -l container_name=nvidia+pytorch+23.10-py3
#PBS -l enroot_env_file=~/sample_jobs/container_env.conf

#Start enroot container and run the command
cd $PBS_O_WORKDIR

enroot start nvidia+pytorch+23.10-py3 python
~/pytorch_examples/torch_data_parallel_model.py --model simplenet --epochs 10
--batch_size 128 --lr 0.0001   >> ${PBS_O_WORKDIR}/pytorch_dp_job_${PBS_JOBID}.out
```

7.2 (b) Multi node job

```
#!/bin/bash
#PBS -N my_enroot_job
### Here select=3 is to select 3 node with 8 gpus each
#PBS -l select=2:ncpus=112:ngpus=8:mpiprocs=8:mem=1880gb:container_engine=enroot
#PBS -l walltime=1:00:00
#PBS -q normal
#PBS -P <project_id>
#PBS -j oe
#PBS -l container_image=$HOME/NCCL/nvidia+tensorrt+23.12-py3.sqsh
#PBS -l container_name=nvidianccl
#PBS -l enroot_mounts="/source/path:/dest/path;/another/source:/another/dest"
```

```
#PBS -l enroot_env_file=~/NCCL/myenv.conf

#Start enroot container and run the command
cd $PBS_O_WORKDIR

enroot start nvidianccl mpirun -hostfile $PBS_NODEFILE
/usr/local/bin/all_reduce_perf_mpi -b 8 -f 2 -g 1 -e 16G -n 8000
```

Sample environment file:

Below is an example of an environment file to be used if you are using the "#PBS -l enroot_env_file=/path/to/env_file" directive.
Use it to add your own environment variables that should be made available inside your containers.:

```
$ cat ~/NCCL/myenv.conf

# Sample environment file for NCCL
UCX_TLS=rc
OMPI_MCA_pml=ucx
OMPI_MCA_coll_hcoll_enable=0
OMPI_MCA_btl=^openib,smcuda
NVSHMEM_ENABLE_NIC_PE_MAPPING=1
NVSHMEM_HCA_LIST=mlx5_0:1,mlx5_3:1,mlx5_4:1,mlx5_5:1,mlx5_6:1,mlx5_9:1,mlx5_10:1,mlx5
_11:1
RDMA_IF="mlx5_0,mlx5_3,mlx5_4,mlx5_5,mlx5_6,mlx5_9,mlx5_10,mlx5_11"
NCCL_IB_QPS_PER_CONNECTION=2
NCCL_IB_SPLIT_DATA_ON_QPS=0
NCCL_IB_HCA="mlx5_0,mlx5_3,mlx5_4,mlx5_5,mlx5_6,mlx5_9,mlx5_10,mlx5_11"
NCCL_DEBUG=INFO
```

Explanations of the parameters used:

a) Container Resource Declaration
container_engine=enroot: This directive is used by PBS to recognize containerized jobs and specify the container type, which is "enroot" in this case.

```
#PBS -l select=3:ncpus=112:mpiprocs=8:mem=1887gb:ngpus=8:container_engine=enroot
```

b) Specifying the Container Image
This directive specifies the container image that PBS should use when creating the container. The path must be absolute, with support for user home directories using $HOME

or ~."

```
#PBS -l container_image=/absolute/path/to/my_image.sqsh
```

c) Defining the Container Name

This allows users to define a custom name for their enroot container. It should be provided according to the user's preference.

```
#PBS -l container_name=my_container
```

d) Defining Custom Mounts (optional)

This option allows users to specify custom files or directories to mount inside the container. Only absolute paths are supported, using the format <source>:<destination>. Multiple mounts can be defined, separated by semicolons.

```
#PBS -l enroot_mounts="/source/path:/dest/path;/another/source:/another/dest"
```

e) Specifying an Environment File (optional)

This directive optionally defines a file that will be sourced inside the container. Variables defined in this file will be available across all container instances on all nodes.

```
#PBS -l enroot_env_file=/path/to/env_file
```

f) Starting the Container

To start the container and invoke the command script within its name space, use the following command:

```
$ enroot start <container-name> <binary> <arguments>
```

Notes:

Default Mounts: All essential default mounts are already predefined in the configuration. Users only need to specify workload-specific mounts.

Multi-Node Jobs: While enroot start allows you to pass variables and mounts, this will only work on the first node in a multi-node job. We recommend using PBS directives to handle these enroot features in multi-node environments.

Multi-Node Jobs with "Fake Root": Running enroot containers in "fake root" mode is not supported for multi-node jobs.

7.3. Default Values

These values will apply if the user submits a job without mentioning these resources in the job submission.

A. ncpus = 1
B. walltime = 00:05:00
C. mem = 10mb
D. queue = normal

7.4 Maximum values that can be requested:

A. walltime= 120:00:00
B. ngpus in single chunk = 8
C. memory in single chunk = 1887 GB
D. ncpus in single chunk = 112

7.5 Resource ratio implementation:

For each GPU requested, 14 CPUs and 253 GB of memory will be automatically allocated, regardless of the user's resource request.

7.6. Job Monitoring

The users can monitor their jobs using the `qstat` command. There are several ways to show the jobs but here are some common ones:

A. To show all jobs currently in queue or running:

```
$ qstat -ans
```

```
altairsup@a2ap-login02:~$ qstat -ans

pbs111:
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
-------------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
4341.pbs111          altairs* aiq1     STDIN         --    1  14  235gb 01:00 R   --
   a2ap-dgx001/0*14
   Job run at Fri Sep 06 at 18:50 on (a2ap-dgx001:ngpus=1:ncpus=14:mem=24...
4342.pbs111          altairs* aiq1     STDIN         --    1  14  235gb 01:00 R   --
   a2ap-dgx001/4*14
   Job run at Fri Sep 06 at 18:50 on (a2ap-dgx001:ngpus=1:ncpus=14:mem=24...
```

B. To show full details of a job if it is currently in queue or running.

```
$ qstat -f <job-ID>
```

C. To show all the nodes in the cluster, available/total resources and jobs running on those nodes.

```
$ pbsnodes -aSj
```

Note: Users can also append -x flag in qstat commands above to see information for finished jobs in addition to queued and running jobs.

7.7. Access Job Node

A. The environment variable `PBS_JOBID` must be set first to access the node running a job via SSH. If the job ID is 123456.pbs111, execute the following command on the login node:

```
export PBS_JOBID=123456.pbs111
```

B. Use SSH to connect from the login node to the assigned job nodes
C. The SSH connection will be rejected if the job ID doesn't match
D. The SSH session will terminate when the job ends

8. Need Help?

For any issues or enquiries regarding **ASPIRE 2A+**, please contact help@nscc.sg and indicate the following in your email subject heading - **“ASPIRE 2A+: [issue]”**

9. Frequently Asked Questions (FAQs)

1. What is the maximum memory that can be used?

A job can request multiple chunks of memory with a maximum of 1887 GB per chunk. (For example: `-l select=2:ncpus=112:ngpus=8:mem=1887GB` is a job that is requesting 3774 GB memory in total and will run on 2 nodes).

2. What is my disk quota (`/home` & `/scratch`) on ASPIRE2A+?

A. Home Directory (/home) - 50 GB
B. Scratch Directory (/scratch) - 100TB

3. How do I check my home and scratch storage quota on ASPIRE2A+?

To check Home and scratch Quota:

```
$ myquota
```

To check Project Quota:

```
$  myquota -p <Project-ID>
```

4. Are there other options besides installing the Duo 2FA mobile app?

Only the mobile app is supported currently.

5. Is it possible to use VPN clients such as OpenVPN or Tunnelblick?

Only Checkpoint VPN clients are supported.

6. Who should I contact if the Duo 2FA does not work?

Please contact us at help@nscc.sg and indicate the following in your email subject heading - **"ASPIRE 2A+: [issue]**

7. My job failed to get submitted with error message:

A. **"enroot image source is required for running containers"**.

Solution: Please ensure that the enroot image path is provided, is correct and the file with

extension ".sqsh" exists.

B. **"container_name is not requested for enroot container. Please add, #PBS -l container_name=<name>"**

Solution: As the comment suggests, the container_name resource was not requested but only container_image was requested. Please add
#PBS -l container_name=my_container_name in job script.

8. My job failed without any output file being created.

The job may have failed for several reasons but most likely the container image may have failed to get created on all the nodes requested. You may try to run enroot create command on the image file you are using in an interactive job to see if there are any errors.

9. My job failed with an output file being created.

Please check the output file for errors and if the error suggests a system issue, please report to help@nscc.sg and attach the error log for further troubleshooting.

10. The environment variables I declared in the environment file are not being passed inside the container or have incorrect values.

Please make sure of the following:

a) The environment file is passed to PBS in the following format:

```
#PBS -l enroot_env_file=/path/to/env_file
```

b) The environment declared is not commented with "#" character at the beginning.
c) The value of the variable does NOT contain another variable like:

```
FOO=$BAR
```

This is currently not supported for anything except $PATH and $LD_LIBRARY_PATH

d) The values declared at the time of container start are not overwritten by other bashrc or bash_profile scripts as part of the container package.

11. I do not have a container image on ASPIRE2A+ but on the docker registry online. How do I use it?

Users can submit an interactive job and use the "enroot import" command to download the enroot container images with extension ".sqsh" on ASPIRE2A+ into their project or home directories. These images can be used in PBS enroot batch jobs as usual.

Prior to importing images, authentication for the NGC cloud needs to be set up. You may refer to the following documentation:
https://github.com/NVIDIA/enroot/blob/master/doc/cmd/import.md#description

Eg:

```
$ qsub -I -q normal -l select=1:ncpus=112:ngpus=8:mem=1887gb -l
walltime=12:00:00 -P <project_ID>

$ enroot import docker://nvcr.io/nvidia/pytorch:23.10-py3
```

12. Can I use singularity image file on ASPIRE2A+?

Singularity is not supported on ASPIRE2A+. Only enroot container images with extension ".sqsh" are supported.

13. **Can I use "--root" or "-r" option with "enroot start" to remap my user to be root inside the container**?While we do not prevent users from using this option, it will most likely won't give the expected results if you are trying to run a multi-node container workload running the container as remapped or "fake" root. It may work as expected in a single node - single container environment. If the requirement of running the container as "fake root" is necessary, please let us know and we will check what can be done for the workaround.

14. Can I access the internet from inside the container?

Yes. You may do so by exporting these variables inside the container environment:
no_proxy=localhost,127.0.0.1,10.104.0.0/21
https_proxy=http://10.104.4.124:10104
http_proxy=http://10.104.4.124:10104

Appendix A (Creating container image outside of 2A+)

You might want to create your container image outside of 2A+. The following are the steps if you wish to do so. The example below uses an ubuntu container image.

1. Install enroot on your local machine or server by following the instructions below.

   https://github.com/NVIDIA/enroot/blob/master/doc/installation.md

2. Modify /etc/enroot/enroot.conf to add the following line (to be consistent with ASPIRE 2A+ enroot settings)

   ENROOT_SQUASH_OPTIONS -noI -noD -noF -noX -no-duplicates

3. Convert the docker image to an enroot image.

   ```
   $ enroot import -o ubuntu.sqsh docker://ubuntu:latest
   ```

4. Create an enroot container from the image

   ```
   $ enroot create -n ubuntu ubuntu.sqsh
   ```

5. Start the enroot container to make sure it is working

   ```
   $ enroot start ubuntu
   ```

6. Stop the enroot container

   ```
   $ enroot remove ubuntu
   ```

7. Install squashfuse and fuse-overlaysfs

   ```
   $ sudo apt-get install -y squashfuse

   $ sudo apt-get install -y fuse-overlayfs
   ```

8. Create a base container from the ubuntu.squash you created in the previous exercise

   ```
   $ enroot create --name my_custom_ubuntu ubuntu.sqsh
   ```

9. Start the container with write permission in container filesystem

   ```
   $ enroot start --root -w my_custom_ubuntu
   ```

10. Add packages / codes to the container

    ```
    $ apt install <pkg_name>

    $ exit
    ```

11. Save the container as new image

    ```
    $ enroot export --output my_custom_tf.sqsh my_custom_tf
    ```