National Supercomputing Centre

NSCC ASPIRE 2A AI System

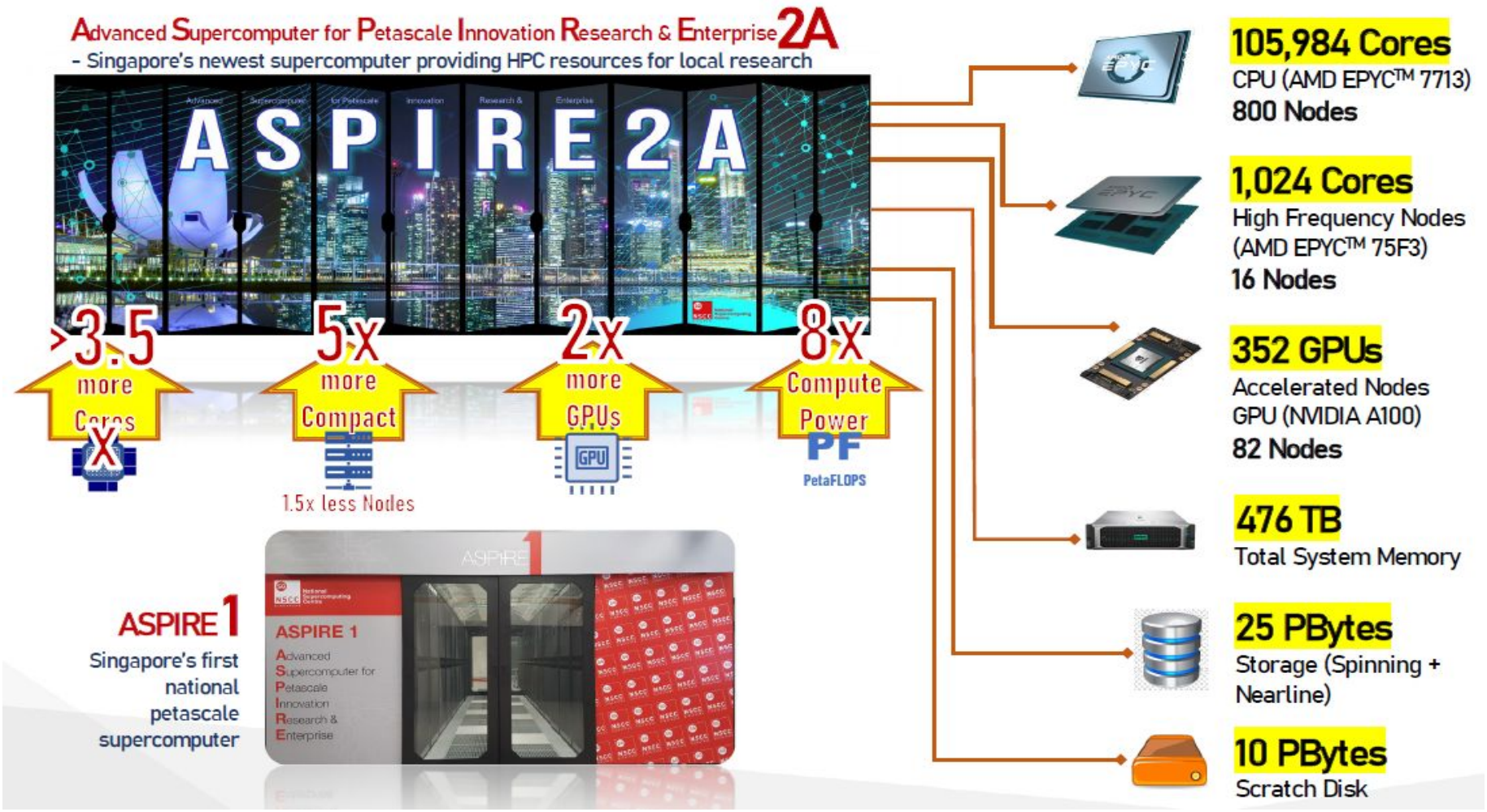
ASP2A - AI QuickStart Guide - 2024

National Supercomputing Centre

Table of Contents

National Supercomputing Centre

Summary of ASPIRE 2A



National Supercomputing Centre

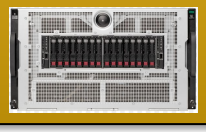
AI Nodes of ASPIRE 2A

AI Cluster

18 nodes (96 x A100)

➔ 12 nodes with 4x A100 40GB with **12TB** nvme.

➔ 6 nodes with 8x A100 40GB with **14TB** nvme



35 PB Storage

- I/O bandwidth up to 500GB/s
- GPFS and Lustre File System

Slingshot 100G Interconnect

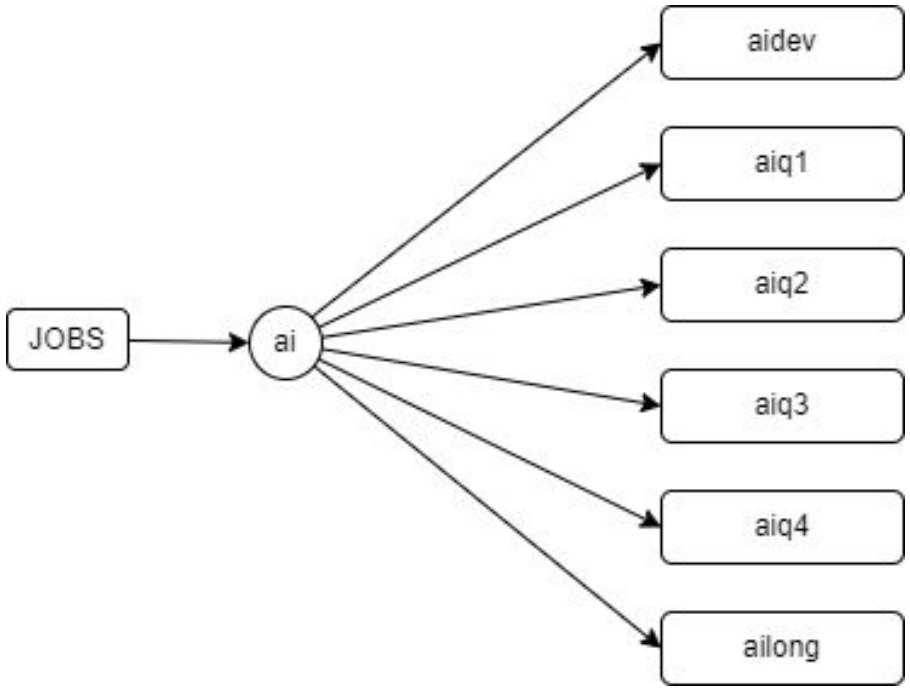
- Dragonfly Topology

National Supercomputing Centre

ASPIRE 2A job queues

There are two route queues:-

1. normal -> pbs101
2. ai -> pbs102



National Supercomputing Centre

Selection rules of resources

Route Queue	Execution Queue	GPU Qty.	Max Walltime (Hour)	Max Run Jobs
ai	aidev	[1,4]	(0,2]	-
	aiq1	1	(2,24]	2
	aiq2	[2,3]	(2,24]	6
	aiq3	4	(2,24]	2
	aiq4	[5,64]	(2,24]	2
	ailong	[1,4]	(2,120]	2

National Supercomputing Centre

Best Practices

- The AI nodes are most suited to large, batch workloads
  - e.g. training complex models with large datasets
- We encourage users to do development and preliminary testing on local resources or aidev queue
- Users are encouraged to use the optimized NVIDIA GPU Cloud Docker images. And convert them to Singularity images.

National Supercomputing Centre

Filesystems

There are multiple filesystems available on the NSCC systems:-

`/home` and `/data/projects` GPFS file system

`/scratch` high-performance Lustre filesystem

`/raid` Local nvme disks on each AI node

Lustre($SCRATCH) and raid($TMPDIR) are better for I/O intensive workloads.

Data policy: https://help.nscc.sg/wp-content/uploads/Appendix-1_Data-Management-and-Retention-Policy_v2_final.pdf

File System	Mount point	Total Capacity	Quota Per user [Fixed]	Data Retention Policy
Lustre	/scratch	10 PB	100TB	Purge
GPFS	/home/users	15PB	50GB	Yes
GPFS	/home/project		**Based on project**	Yes
Local $TMPDIR	/raid/pbs.<jobid>.pbs101	12/14TB	**Auto-created**	Auto-removed

National Supercomputing Centre

ALTAIR ONLY FORWARD

2 -The job management with PBS Pro

- Submit jobs
- Job script example
- Check job and node states
- Resources selection(GPUs)
- Best Practices

National Supercomputing Centre

Submit jobs: an interactive job

```
qsub -I -l select=1:gpus=1 -l walltime=1:00:00 -P <project> -q ai
```

What an interactive job can do?

- ❑ For debugging
- ❑ For testing before batch jobs

National Supercomputing Centre

Batch Job Submission

Accessing the batch scheduler generally involves 3 commands:

To submit a batch job	`qsub jobscript`
To query a job state	`qstat @pbs102`
To Kill a job	`qdel <jobid> @pbs102`

See https://help.nscc.sg/user-guide/ for more information on how to use the PBS scheduler
Introductory workshops are held regularly, more information at https://www.nscc.sg/hpc-calendar/

National Supercomputing Centre

Example: PBS Job Script (Headers)

```
#!/bin/sh
## Lines which start with #PBS are directives for the scheduler

## The following line requests the resources for 1 gpu, ngpus=4 for 4 GPUs.
#PBS -l select=1:ngpus=1

## Run for 1 hour, modify as required
#PBS -l walltime=1:00:00

## Submit to correct queue for AI cluster access
#PBS -q ai

## Specify project ID
#PBS -P <projectId>

## Job name
#PBS -N <jobName>

## Merge standard output and error from PBS script
#PBS -j oe
```

National Supercomputing Centre

Example PBS Script (Commands)

```
# Change to directory where job was submitted
cd "$PBS_O_WORKDIR" || exit $?

# Specify which singularity image to use for container
image="/app/apps/containers/pytorch/pytorch-nvidia-22.02-py3.sif"

# Pass the commands that you wish to run inside the container
singularity run --nv $image python scripts.py <args>
or
singularity exec --nv $image python scripts.py <args>
```

National Supercomputing Centre SINGAPORE

Check job and node states

3 available options to see which host a job is running on:

`$ qstat -f JOBID @pbs102`

```
Job Id: 1850335.pbs101
    Job_Name = STDIN
    Job_Owner = michaelqi@asp2a-login-nscc02.head.cm.asp2a.nscc.sg
    resources_used.cpupercent = 0
    resources_used.cput = 00:00:00
```

`$ qstat -wan JOBID @pbs102`

```
pbs102:
                                                                                  Req'd  Req'd   Elap
Job ID                          Username        Queue           Jobname         SessID   NDS  TSK   Memory Time  S Time
------------------------------- --------------- --------------- --------------- -------- ---- ----- ------ ----- - -----
1850335.pbs101                  michaelqi       aidev           STDIN             415837    1    16  110gb 02:00 R 00:06
   asp2a-gpu002/3*16
```

`$ pbsnodes -Sj asp2a-gpu0{04..05}`

```
                                                          mem        ncpus    nmics    ngpus
vnode           state           njobs   run    susp       f/t         f/t      f/t      f/t    jobs
--------------- --------------- ------ ----- ------ ------------ ------- ------- ------- -------
asp2a-gpu004    job-busy             1     1      0    126gb/1tb   0/128     0/0     0/8 1801705
asp2a-gpu005    job-busy             1     1      0    126gb/1tb   0/128     0/0     0/8 1801705
```

National Supercomputing Centre

Resources selection (GPUs)

Specify required ngpus resource in job script: Example: 1_ai_environment/gpu.selection.sh

```
#PBS -l select=1:ngpus=N
```

where ***N*** is the number of GPUs required, 16xN CPU cores and memory will be selected by PBSpro

```
$ echo nvidia-smi | qsub -l select=1:ngpus=1 -l walltime=0:05:00 -q ai -P <projID>
1852792.pbs101
$ grep A100 STDIN.o1852792
|   0  NVIDIA A100-SXM...  On   | 00000000:88:00.0 Off |                    0 |

$ echo nvidia-smi | qsub -l select=1:ngpus=2 -l walltime=0:05:00 -q ai -P <projID>
1852793.pbs101
|   0  NVIDIA A100-SXM...  On   | 00000000:46:00.0 Off |                    0 |
|   1  NVIDIA A100-SXM...  On   | 00000000:C7:00.0 Off |                    0 |


$ echo nvidia-smi | qsub -l select=1:ngpus=4 -l walltime=0:05:00 -q ai -P <projID>
1852794.pbs101
|   0  NVIDIA A100-SXM...  On   | 00000000:07:00.0 Off |                    0 |
|   1  NVIDIA A100-SXM...  On   | 00000000:46:00.0 Off |                    0 |
|   2  NVIDIA A100-SXM...  On   | 00000000:85:00.0 Off |                    0 |
|   3  NVIDIA A100-SXM...  On   | 00000000:C7:00.0 Off |                    0 |
```

National Supercomputing Centre

GPU Environment variable

echo $CUDA_VISIBLE_DEVICES

```
CUDA_VISIBLE_DEVICES=GPU-50ee0fc4-bb3d-920c-8039-da7054e1496b
```

When the job gets the GPU resources, the $CUDA_VISIBLE_DEVICES is assigned by PBSpro.

For some apps, if this value cannot be accepted. then export it by the ids to replace the UUIDs

```
export CUDA_VISIBLE_DEVICES=0
```

If select 2 GPUs,

```
export CUDA_VISIBLE_DEVICES=0,1
```

National Supercomputing Centre

Best Practices

- Access is through **PBS job scheduler**
- We encourage workloads which can scale up to utilise all 8 GPUs on a node or run across multiple nodes
- Users can request fewer than 8 GPUs
  - Multiple jobs will run on a node with GPU resource isolation (using cgroups)
  - You will only see the number of GPUs you request
  - Select whole GPUs in a node

```
-l select=1:ngpus=4  VS  -l select=4:ngpus=1
```

National Supercomputing Centre

Building AI environment

- Python virtual environment (using conda for example.)
- Singularity image

National Supercomputing Centre

Python virtual environment

Python is a popular programming language for artificial intelligence (AI) research. And it has a large library of AI-related libraries and frameworks.

Python virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. With Python virtual environment, the multi-research environment can be created and switch easily to adapt different AI models.

Conda is an open-source package manager and environment management system. Conda as a package manager helps you find and install packages of AI projects.

National Supercomputing Centre

Create the virtual env.

- **modules and create env:-**

```
module load miniforge3
conda create -n myenv python=3.11
```

- **Activate env. and install packages:-**

```
conda activate myenv
conda install -y mamba -c conda-forge
mamba install pytorch torchvision torchaudio pytorch-cuda=11.6 -c
pytorch -c nvidia
```

- **Deactivate env:-**

```
conda deactivate
```

Note: Do not add conda init in ~/.bashrc, in some case, it will cause login very slow. And double load the conda default environment when the miniforge3 module file is loaded.

National Supercomputing Centre

In job script

```
# load miniforge3
module load miniforge3

# activate the env
conda activate myenv

# run python scripts
python scripts.py <args>
```

National Supercomputing Centre

Singularity images

A Singularity image is a container image that is used to run software in a reproducible and isolated environment. It is an open-source project that is often used in HPC research environments.

Singularity images are similar to Docker images, but they have a few key differences:-

1) Singularity images are not tied to a specific operating system. This means that they can be run on a variety of different machines, regardless of the operating system that is running on the machine.

2) Singularity images are more secure than Docker images. This is because Singularity images are sandboxed, which means that they are isolated from the host machine.

3) Singularity images are more portable than Docker images. This is because Singularity images can be easily moved from one machine to another.

National Supercomputing Centre

Ways to Create Singularity Image

- Singularity containers (https://sylabs.io/docs/)
- Docker hub, NGC Cloud (https://catalog.ngc.nvidia.com/)
- Images are created from your workstation or PC.

National Supercomputing Centre

From Docker Images

- Find the image

  https://hub.docker.com/

  https://catalog.ngc.nvidia.com/

```
$singularity build <imageName>.sif docker://docker/Image:tag
$singularity build pytorch_23.06.sif
docker://nvcr.io/nvidia/pytorch:23.06-py3
$singularity build pytorch_23.06.sif docker-daemon://pytorch:23.06-py3
```

- Remove the cache if necessary

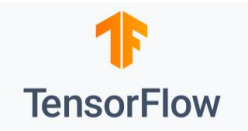
```
$rm -rf ~/.singularity/cache
```

National Supercomputing Centre

Local images on ASPIRE 2a

/app/apps/containers


```
pytorch-nvidia-22.04-py3.sif
pytorch-nvidia-22.12-py3.sif
```




```
tensorflow_2.3.0_gpu_py3_nltk_3.6.7.sif
tensorflow-nvidia-22.04-tf2-py3.sif
tensorflow-nvidia-22.12-tf2-py3.sif
```

National Supercomputing Centre

Hands-on

```
git clone /app/workshops/introductory/ai
```

Example PBS job scripts to demonstrate how to:

1. create AI environment
2. conda
3. singularity image
4. submit jobs with GPU selection
5. run a standard MXNet training job
6. run a jupyter-note job

See https://help.nscc.sg/user-guide/ for more information on how to use the NSCC systems

National Supercomputing Centre

Hands-on

Step 1: Log on to NSCC ASPIRE 2A

Step 2: Run the following commands to clone the hands-on examples

```
git clone /app/workshops/introductory/ai
ls -l ai
drwxr-xr-x 2 michaelqi fujitsu 4096 Jul 12 16:39 1_ai_environment
drwxr-xr-x 2 michaelqi fujitsu 4096 Jul 12 16:39 2_singularity_image
drwxr-xr-x 2 michaelqi fujitsu 4096 Jul 12 16:39 3_job_with_multi-gpus
drwxr-xr-x 2 michaelqi fujitsu 4096 Jul 12 16:39 3_pytorch_data_parallel
```

Use "`qstat @pbs102`" to check job status and when jobs have finished examine output files to confirm everything is working correctly

National Supercomputing Centre

Example 1

Create AI environment

- Create a pytorch environment with miniforge3
  - Location: ai/1_ai_environment
  - file: conda.env.md

```
module load miniforge3
conda create -n myenv python=3.11
# Activate env and install packages
conda activate myenv
conda install -y mamba -c conda-forge
mamba install pytorch torchvision torchaudio pytorch-cuda=11.6 -c pytorch -c
nvidia
```

National Supercomputing Centre

Example 2

Build singularity image from NGC

- Example folder: ai/2_singularity_image

```
qsub -I -l select=1:ngpus=1 -q ai -P <projectId> -l walltime=02:00:00
module load singularity
singularity build pytorch_22.02-py3.sif
docker://nvcr.io/nvidia/pytorch:22.02-py3
```

- Submit a job which using pre-created singularity image:

```
qsub singularity-tensorflow.pbs
```

National Supercomputing Centre

Example 3

Run a multi-gpu AI codes

Location: ai/3_job_with_multi_gpus

submit a job with 4 GPUs

```
qsub train.pbs
```

National Supercomputing Centre

Example 4: Jupyter-lab

Jupyter lab job:-

```
#!/bin/bash

#PBS -q ai
#PBS -l select=1:ngpus=1
#PBS -l walltime=2:00:00
#PBS -P <projectid>
#PBS -N jupyter
#PBS -j oe
```

```
# Change directory to where job was submitted
cd $PBS_O_WORKDIR || exit $?

# get a random port
PORT=$(shuf -i8000-8999 -n1)
module load singularity

echo -e "ssh -N -L $PORT:`hostname`:$PORT $USER@aspire2a.nus.edu.sg\n">>sshtunnel.$PBS_JOBID

singularity  exec --nv -B /scratch,/app \
     /app/apps/containers/pytorch/pytorch-nvidia-22.04-py3.sif jupyter-lab \
       --no-browser --ip=0.0.0.0 --port=$PORT \
       >> sshtunnel.$PBS_JOBID 2> jpylab.$PBS_JOBID
```

National Supercomputing Centre

Example 4: (continue)

- Create ssh tunnel with a new terminal, the command in ssh tunnel.<jobid>

  `ssh -N -L 8541:asp2a-gpu003:8541 yourId@<aspire2a.login>`

- Open Jupyter lab with local web browser

`To access the notebook, find the url in "jpylab.<jobid>":`

http://localhost:8541/?token=5839e1dda5899003a05666d059de1552ce67ffd7df49a973

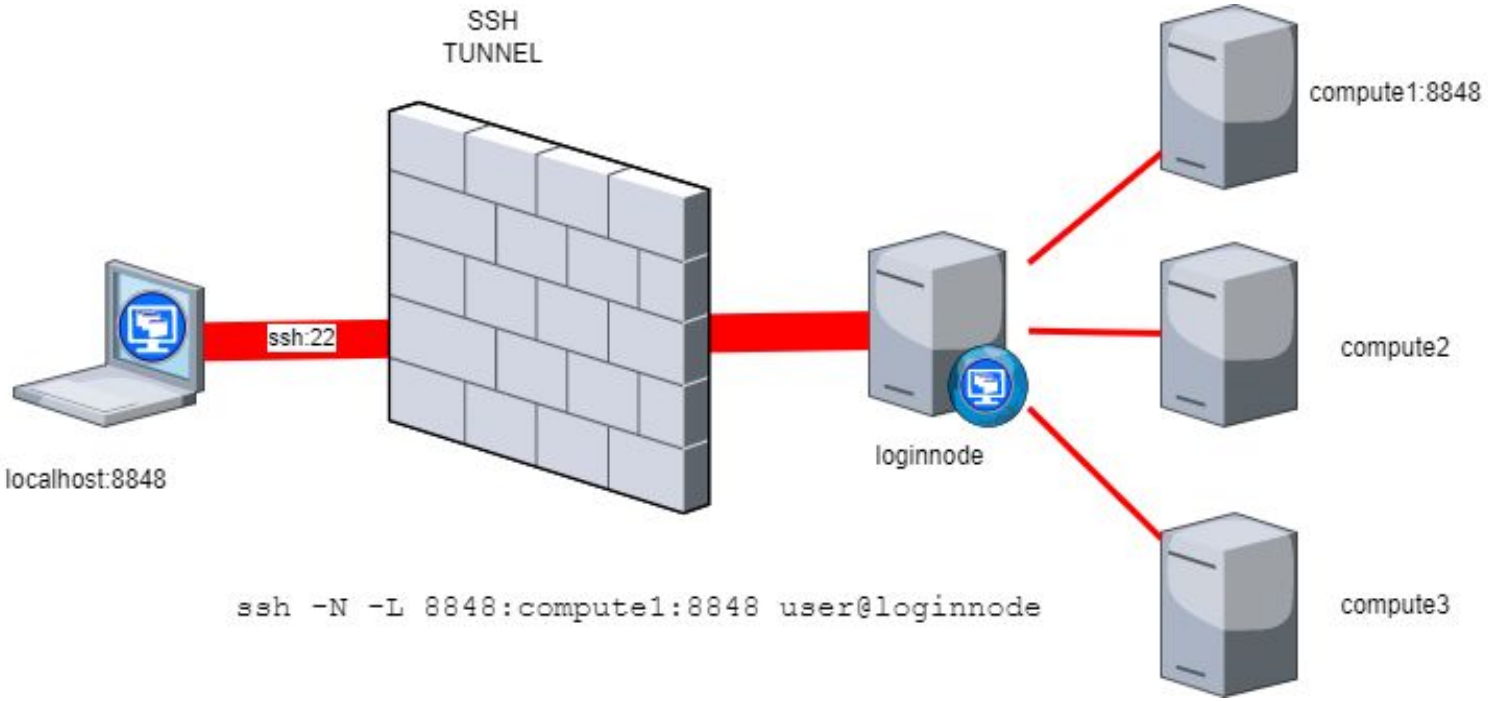
or

http://127.0.0.1:8541/?token=5839e1dda5899003a05666d059de1552ce67ffd7df49a973

National Supercomputing Centre

SSH TUNNEL

- Safety connection
- Bypass firewall



National Supercomputing Centre

Thank you!