SoC GPU

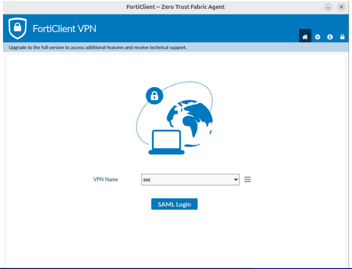
Jiaming, Xavier

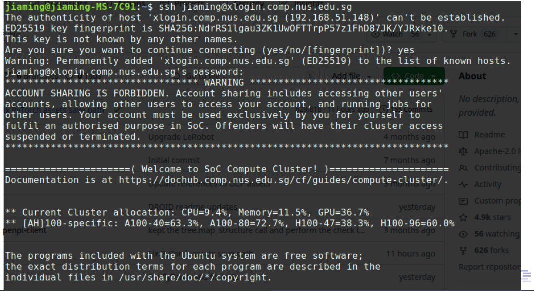
September 12, 2025

Accessing SoC Compute Cluster

- **Create SoC account**: `https://mysoc.nus.edu.sg/~newacct`
- **Enable service**: `https://mysoc.nus.edu.sg/~myacct/services.cgi`
- **SoC VPN is required if not in SoC network**: `https://dochub.comp.nus.edu.sg/cf/guides/network/vpn/`
- **After account is settup, SSH to login node**:

```
ssh username@xlogin.comp.nus.edu.sg
```


FortiClient -- Zero Trust Fabric Agent
FortiClient VPN
Upgrade to the full version to access additional features and receive technical support.
VPN Name
soc
SAML Login



```
jiaming@jiaming-MS-7C91:~$ ssh jiaming@xlogin.comp.nus.edu.sg
The authenticity of host 'xlogin.comp.nus.edu.sg (192.168.51.148)' can't be established.
ED25519 key fingerprint is SHA256:NdrRS1lgau3ZK1UwOFTTrpP57z1FhD87JK/YJRxke10.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'xlogin.comp.nus.edu.sg' (ED25519) to the list of known hosts.
jiaming@xlogin.comp.nus.edu.sg's password:
********************************** WARNING ***********************************
ACCOUNT SHARING IS FORBIDDEN. Account sharing includes accessing other users'
accounts, allowing other users to access your account, and running jobs for
other users. Your account must be used exclusively by you for yourself to
fulfil an authorised purpose in SoC. Offenders will have their cluster access
suspended or terminated.
******************************************************************************

======================( Welcome to SoC Compute Cluster! )=====================
Documentation is at https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/.


** Current Cluster allocation: CPU=9.4%, Memory=11.5%, GPU=36.7%
** [AH]100-specific: A100-40=63.3%, A100-80=72.7%, H100-47=38.3%, H100-96=60.0%


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

Slurm Basics

- Scheduler: **Slurm** (submit jobs; avoid heavy compute on xlogin)
- Common commands:

```
sinfo              # cluster/partition status
squeue -u $USER    # your jobs in queue
sbatch job.slurm   # submit batch job
salloc             # interactive allocation
scancel <jobid>    # cancel job
sacct              # finished jobs accounting
```

Example Workflow Overview

1. Install/activate your own Conda.
2. Create environment and install dependencies (e.g. PyTorch)
3. Write training script (`train_toy.py`).
4. Write Slurm script (`run_toy.slurm`).
5. Submit job with `sbatch` and monitor.
6. Retrieve logs/results (e.g., `scp`).

Create Environment & Install PyTorch

```
# if conda is not installed, download & run installer (on xlogin)
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh

# Initialize shell so ‘conda‘ works
source ~/.bashrc

# Create & activate an environment
conda create -n torch-env python=3.9 -y
conda activate torch-env

# Install PyTorch (CPU by default; GPU if CUDA present)
pip install torch numpy
```

Toy PyTorch Training Script (train_toy.py)

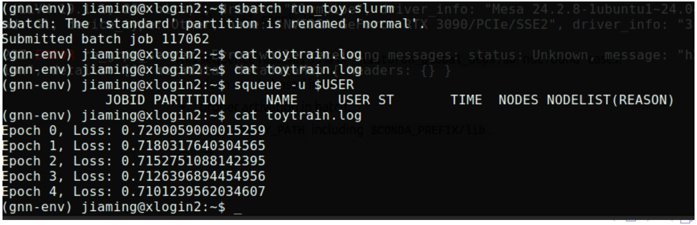
```python
import torch
import torch.nn as nn
import torch.optim as optim

# Toy dataset
x = torch.randn(100, 10)
y = torch.randint(0, 2, (100,))

# Simple MLP
model = nn.Sequential(nn.Linear(10, 32), nn.ReLU(), nn.Linear(32, 2))
loss_fn = nn.CrossEntropyLoss()
opt = optim.Adam(model.parameters(), lr=1e-3)

for epoch in range(5):
    opt.zero_grad()
    out = model(x)
    loss = loss_fn(out, y)
    loss.backward()
    opt.step()
    print(f"Epoch {epoch}: loss={loss.item():.4f}")
```

Slurm Batch Script (run_toy.slurm)

```
#!/bin/bash
#SBATCH --job-name=toytrain
#SBATCH --output=toytrain.log
#SBATCH --time=00:05:00
#SBATCH --partition=standard
#SBATCH --gpus=1        # remove or change if GPU not needed

# Ensure conda is available in batch context
source ~/.bashrc
conda activate torch-env

python train_toy.py
```

Submit, Monitor, and Inspect Logs

```
# Submit
sbatch run_toy.slurm

# Monitor your job(s)
squeue -u $USER
sacct --format=JobID,State,Elapsed,MaxRSS,ExitCode

# Inspect training logs
tail -f toytrain.log   # or: cat toytrain.log
```



```
(gnn-env) jiaming@xlogin2:~$ sbatch run_toy.slurm
sbatch: The 'standard' partition is renamed 'normal'.
Submitted batch job 117062
(gnn-env) jiaming@xlogin2:~$ cat toytrain.log
(gnn-env) jiaming@xlogin2:~$ cat toytrain.log
(gnn-env) jiaming@xlogin2:~$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
(gnn-env) jiaming@xlogin2:~$ cat toytrain.log
Epoch 0, Loss: 0.7209059000015259
Epoch 1, Loss: 0.7180317640304565
Epoch 2, Loss: 0.7152751088142395
Epoch 3, Loss: 0.7126396894454956
Epoch 4, Loss: 0.7101239562034607
(gnn-env) jiaming@xlogin2:~$ _
```

Retrieve Results to Your Laptop

```
# From your laptop/desktop terminal
scp username@xlogin.comp.nus.edu.sg:~/toytrain.log .

# Copy a directory of results
scp -r username@xlogin.comp.nus.edu.sg:~/results ./results
```

Optional: Interactive Debug Session

```
# Request an interactive allocation
salloc --gpus=1 --time=00:10:00

# Get a shell on the compute node
srun --pty bash

# Inside the node: run the script
source ~/.bashrc && conda activate torch-env
python train_toy.py

# Exit to release resources
exit   # leaves node shell
exit   # ends allocation
```

Tips & Good Practices

- Keep jobs short while testing; extend time after validation.
- Prefer `sbatch` for reproducibility; use `salloc` only for debugging.
- Write outputs to your $HOME/$WORK; clean up temporary files.