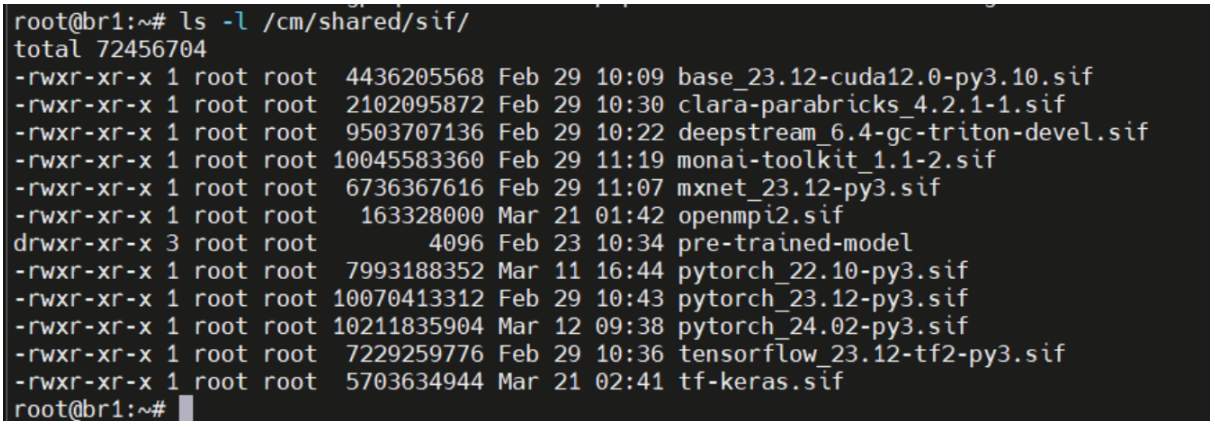
AI_KU_A100 คู่มือการใช้งาน Singularity

Singularity คือ ระบบ Container ที่ใช้งานบนระบบ HPC ซึ่งทำงานคล้าย Docker และสามารถนำ Image จาก Docker มารันได้

ในขณะที่รันภายใน Singularity container ผู้ใช้งานมีระดับของสิทธิ์เหมือนกันกับภายนอก container โดยทั่วไปแล้ว Singularity image จะต้องถูกพัฒนาและสร้างขึ้นมาจากพื้นฐานเครื่อง Linux ซึ่งต้องมีสิทธิ์เทียบเท่า Administrator โดยในระบบ หรือใช้งาน Feature Fakeroot โดยตัวอย่างไฟล์ image และไฟล์ .def สำหรับผู้ใช้งานทุกคนถูกวางไว้ที่ /cm/shared/sif/



1. ก่อนเริ่มใช้งาน Singularity ทุกครั้ง

ก่อนเริ่มใช้งาน Singularity ให้ทำการรันคำสั่งด้านล่างก่อนทุกครัง

```
module load singularity-ce
```

```
(base) wpeeranon@br1:~$ module load singularity-ce/
(base) wpeeranon@br1:~$ singularity version
3.9.2
(base) wpeeranon@br1:~$
```

2. การสร้าง Singularity Image ด้วย Fakeroot feature

ในกรณีที่คุณมีเครื่องที่ติดตั้ง Singularity บนเครื่อง Linux ไว้แล้ว เราก็สามารถที่จะสร้าง Singularity image ได้ โดยสร้าง Singularity Definition file โดยให้มีรายละเอียดดังตัวอย่างต่อไปนี้

```
(base) [peeranon@              . ~]$ cat ,                    'tensorflow-keras_20.12-tf2-py3.def
Bootstrap: docker
From: nvcr.io/nvidia/tensorflow:20.12-tf2-py3

%runscript
        echo "Singularity image with TensorFlow 2 and Keras running on python 3.5 with GPU support."

%post
        # Set up some e required environment defaults
        export LC_ALL=C
        export PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH

        # Install other commonly-needed packages
        pip install -U keras

        mkdir /cm
```

ภายในไฟล์ เรากำลังต้องการให้ Singularity สร้าง image บน Ubuntu 18.04 โดยดึง base image จาก Docker Hub ภายใต้ %runscript จะใส่สิ่งที่ต้องการติดตั้งลงไป โดยจะทำเพียงครั้งเดียวในตอนสร้าง image ในกรณีนี้ เราใช้คำสั่ง apt-get ในการติดตั้ง Python 3 พร้อมกับ NumPy และ SciPy และสุดท้ายถ้าเราต้องการให้ image สามารถใช้งานพื้นที่จัดเก็บภายใน Cluster หรือบน NFS เราจำเป็นต้องประกาศ directories ต่าง ๆ ภายใน container ได้แก่ /cm ถ้าไม่ประกาศ directories เหล่านี้ภายใน container เราอาจจะได้รับ warning เมื่อรัน container นี้

ในการ Build image สามารถรันคำสั่งด้านล่างนี้ได้เลย พร้อมระบุ flag fakeroot และตำแหน่งของ image และ Definition file

```
singularity build --fakeroot example.sif singularity.def
```

```
(base) [peeranon@                ~]$ singularity build --fakeroot tf-keras.sif /data/container_image/tensorflow-keras_20.12-tf2-py3.def
INFO:    Starting build...
Getting image source signatures
Copying blob 6a5697faee43 [================================>-----] 24.0MiB / 27.2MiB
```

Examples

Build from a definition file:

```
singularity build --fakeroot /tmp/test.sif /tmp/test.def
```

Ping from container:

```
singularity exec --fakeroot --net docker://alpine ping -c1 8.8.8.8
```

HTTP server:

```
singularity run --fakeroot --net --network-args="portmap=8080:80/tcp" -w docker://nginx
```

ภาพแสดงตัวอย่างเพิ่มเติมในการใช้ Fakeroot

3. การ Pull Container ที่ต้องการใช้งาน

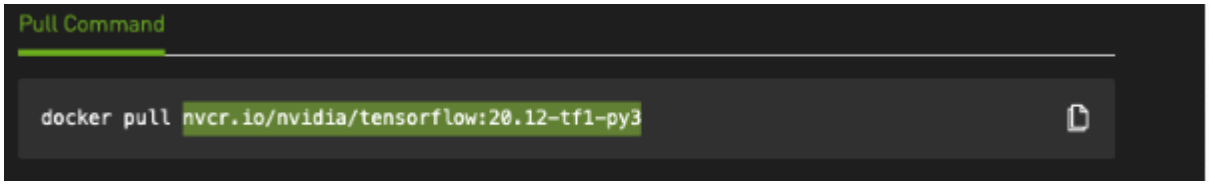
3.1 กรณีมี container image ที่พร้อมใช้งานอยู่แล้ว

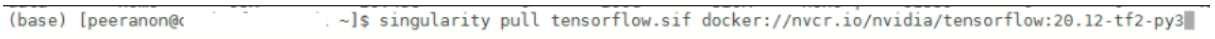
เลือกแหล่งของ Image เพื่อใช้งานจากเว็บไซต์ที่ต้องการ ตัวอย่างเช่น 2 เว็บไซต์ ได้แก่

1. NVIDIA GPU Cloud (NGC) แหล่งรวม Pre-built GPU-Accerelated container https://ngc.nvidia.com/catalog/containers
2. Docker hub แหล่งรวม container ของ Docker https://hub.docker.com/

```
singularity pull <singularity_name>.sif docker://<pull_url>
```

ในกรณีที่ต้องการ Pull image จาก NGC สามารถนำ Pull URL ที่อยู่ใน Pull Command จากเว็บไซต์ของ NGC มาแทนที่ <pull_url> ในตัวอย่างคำสั่งได้เลย ดังตัวอย่างในภาพด้านล่าง





4 การรัน Container

4.1 การรัน Singularity Container (CPU-based)

โดยใช้คำสั่งด้านล่าง การใส่ `--nv` เพื่อเป็นการเรียกใช้งาน GPU

```
singularity exec <singularity_name>.sif <command>
```

ตัวอย่างการเรียกใช้งาน Singularity Image

```
(base) jsorawid@br1:~$ singularity exec --nv /cm/shared/sif/pytorch_24.02-py3.sif python -V
WARNING: Could not find any nv files on this host!
13:4: not a valid test operator: (
13:4: not a valid test operator:
Python 3.10.12
```

4.2 การรัน Singularity GPU-Enabled Containers

การรัน Singularity เพื่อใช้งาน GPU ให้ใส่ `--nv` ไว้ในคำสั่งเสมอ

```
singularity exec --nv /cm/shared/sif/pytorch_24.02-py3.sif
```

```
(base) wpeeranon@dgx-01:~$ module load singularity-ce/
(base) wpeeranon@dgx-01:~$ singularity shell --nv /cm/shared/sif/pytorch_24.02-py3.sif
13:4: not a valid test operator: (
13:4: not a valid test operator: 535.154.05
Singularity> nvidia-smi -L
GPU 0: NVIDIA A100-SXM4-80GB (UUID: GPU-b75bc4e6-1801-7974-70d7-c2847d1ab4b9)
  MIG 3g.40gb     Device  0: (UUID: MIG-d4bdd46f-5611-529d-8152-14acd748b6eb)
  MIG 3g.40gb     Device  1: (UUID: MIG-74094c8d-6148-54c9-bdb2-90fc4153afac)
GPU 1: NVIDIA A100-SXM4-80GB (UUID: GPU-9e3ae35f-aab7-5abc-69d3-4cac9615b9d9)
  MIG 3g.40gb     Device  0: (UUID: MIG-8a4362ee-f2b0-531f-9890-29cd3b7124c9)
  MIG 3g.40gb     Device  1: (UUID: MIG-171d673d-c497-5d02-b21c-5b915e1305d4)
```

5. การเรียกใช้งาน Singularity ผ่าน Slurm

ตัวอย่างการใช้งานคำสั่ง Bash Script ในการเรียกใช้งาน Singularity ใน Slurm Cluster

```
(base) jsorawid@br1:~$ cat slurm_with_sing.sh
#!/bin/bash
#SBATCH -p gpuq
#SBATCH -t 00:30:00
#SBATCH -J simple_cnn
#SBATCH -c 2
#SBATCH --mem=16G
#SBATCH --gres=gpu:1

module load singularity-ce

#singularity exec --nv /cm/shared/sif/tf-keras.sif python simple_cnn.py
singularity exec --nv /cm/shared/sif/tf-keras.sif python -c "import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))"
```

ทำการ Submit งาน ด้วยคำสั่ง sbatch

```
(base) jsorawid@br1:~$ sbatch slurm_with_sing.sh
Submitted batch job 1651
```

ภายหลังจากการ Submit สำเร็จ ให้ลองตรวจสอบงานด้วยคำสั่ง squeue เพื่อตรวจสอบสถานะของงาน

```
(base) jsorawid@br1:~$ squeue  -u jsorawid
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1651      gpuq simple_c jsorawid  R       0:29      1 dgx-02
```

เมื่องานรันเสร็จแล้ว จะพบกับไฟล์ Output แบบ Default name ที่ชื่อ slurm-<JOBID>.out โดย <JOBID> จะเป็นเลข Job ที่เราได้ตอนสั่งรัน sbatch เช่น slurm-1651.out เมื่อลองเปิดไฟล์ดู จะพบกับ Output log จากงานที่เราสั่งรันไปเมื่อสักครู่