

# User Manual

## สารบัญ

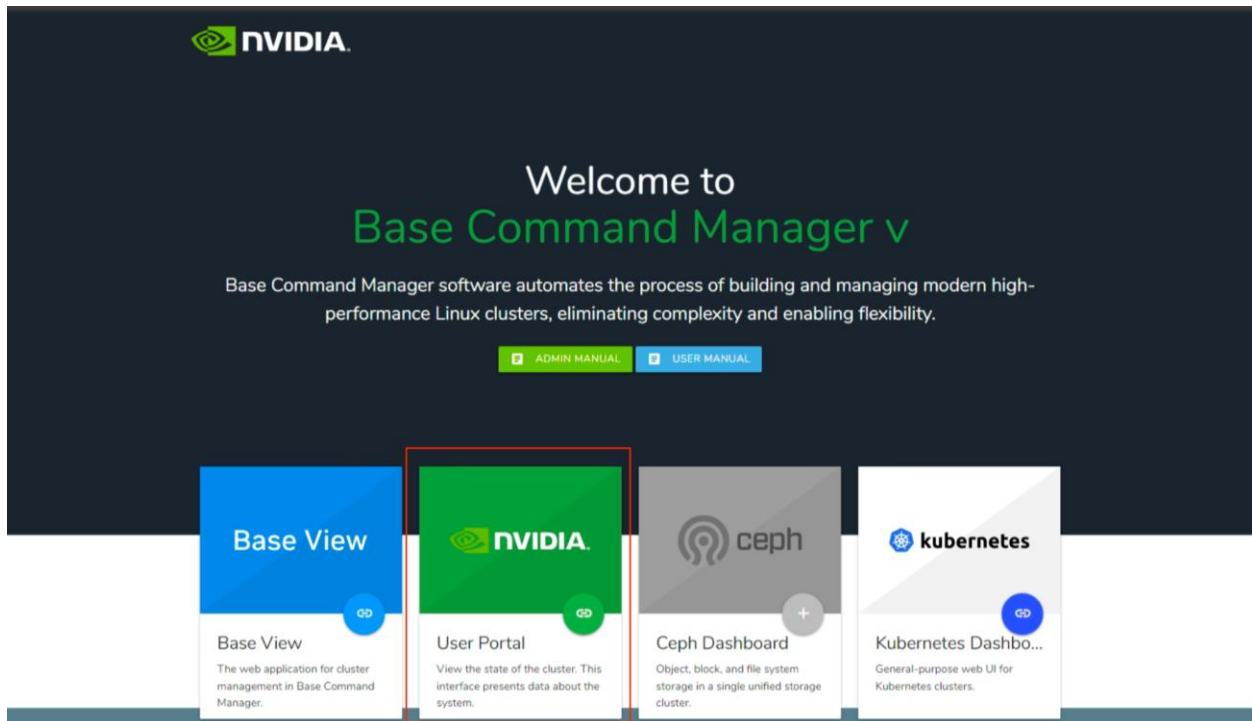
สารบัญ	2
1. Access the system	3
1.1 Web GUI	3
1.2 Command-Line (CLI)	5
2. Base Command Manager	5
2.1 การเข้าใช้งานผ่าน Bright Cluster Manager - User Portal	6
2.1.1 User Portal Overview	6
2.1.2 Workload	7
2.1.3 Nodes	8
2.1.4 Kubernetes	8
3. Module Environment	9
3.1 การแสดง module ที่สามารถใช้งานได้	9
3.2 การเรียกใช้งาน module ด้วยคำสั่ง module load	9
4. Slurm Workload Management	10
4.1 Introduction	10
4.2 การตรวจสอบสถานะของ Node	10
4.3 การรันงาน	11
4.3.1 การรันงานด้วย sbatch	11
4.3.2 การรันงานด้วย srun	11
4.3.3 การรันงานแบบ Interactive Jobs ด้วย srun	11
4.4 การกำหนดทรัพยากรในการรันงาน	12
4.5 การตรวจสอบงานที่รันอยู่	13
4.6 การยกเลิกงาน	13
5. JupyterHub	14
6. Kubernetes	17
6.1 การเข้าใช้งาน	17
6.2 การสร้าง Pod	17
6.3 การลบ Pod	17
6.4 การตรวจสอบ Pod	18
7. Singularity / Apptainer	20

## 1. Access the system

ผู้ใช้งานสามารถเข้าใช้งานระบบได้จาก 2 ช่องทาง คือ Web GUI และ Command-line (CLI)

### 1.1 Web GUI

- 1) เปิด Web browser จากนั้นกรอก URL <http://br1.paas.ku.ac.th/>
- 2) ในกรณีเป็นผู้ใช้งานทั่วไป เลือก User Portal



### Please Sign In

→ Login

- Overview
- Workload
- Nodes
- Kubernetes

#### Overview

#### Message of the day

This is the message of the day. Feel free to edit this to your liking (in assets/config/message-of-the-day.html).

#### Documentation

- NVIDIA Bright Cluster Manager
- Administrator manual
- User manual
- JSON API documentation

#### Cluster overview

Uptime	12 days 13 hours 39 min
Nodes	
Devices	
Cores	1368 up out of 1368 total
Users	1 out of 5 total
Phase Load	N/A
Occupation Rate	3.50

#### Resource utilization

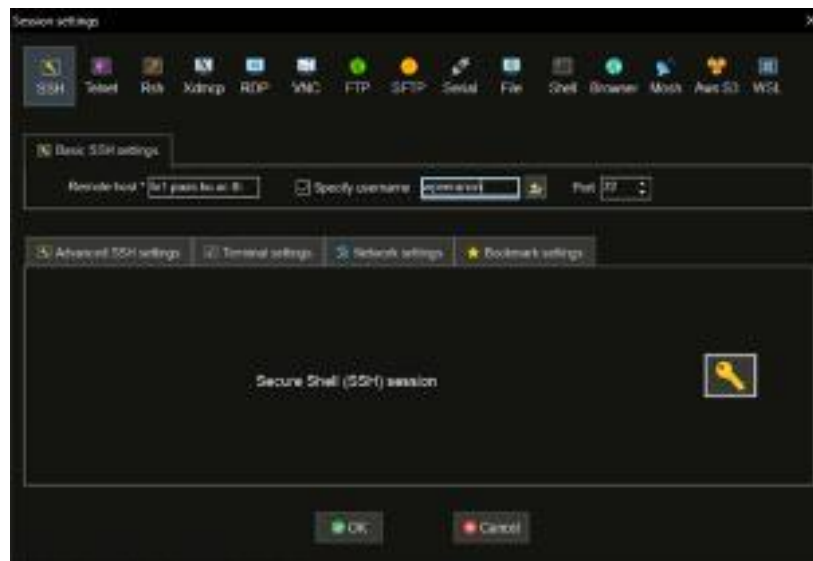
Memory	
Swap	
User usage	
System usage	
Idle usage	
Other usage	

## 1.2 Command-Line (CLI)

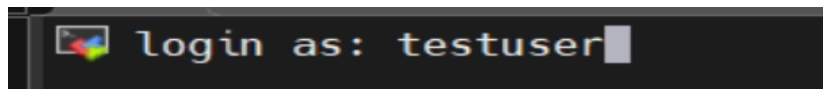
ในตัวอย่างนี้ ใช้โปรแกรม MobaXterm ในการเข้าใช้งานแบบ CLI

(<https://mobaxterm.mobatek.net/download.html>)

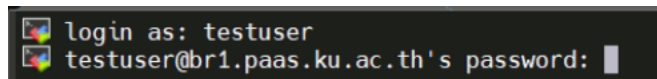
- 1) เปิดโปรแกรม MobaXterm
- 2) เลือก Session -> SSH
  - a) กรอกข้อมูล Remote host = br1.paas.ku.ac.th
  - b) Port = 22
  - c) กด OK



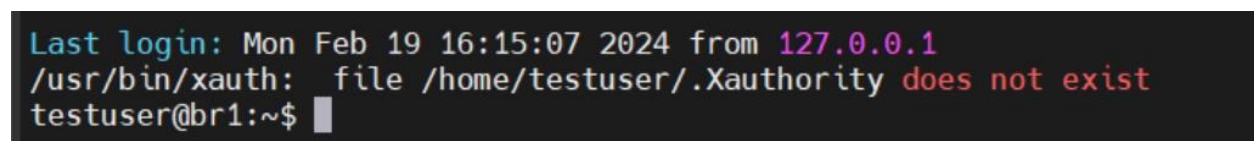
- 3) กรอกชื่อ username ของผู้ใช้งาน แล้วกด Enter



- 4) กรอกรหัสผ่านของผู้ใช้งาน แล้วกด Enter



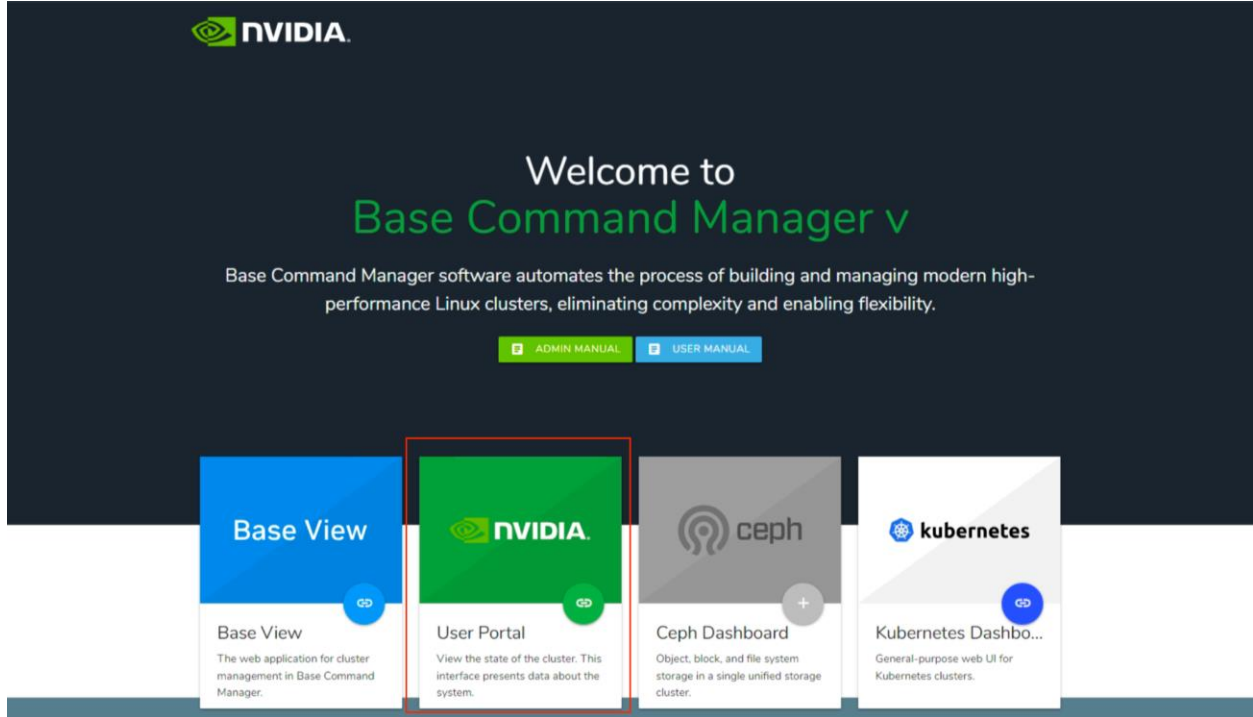
- 5) เมื่อเข้าใช้งานสำเร็จ จะได้ผลดังภาพด้านล่าง



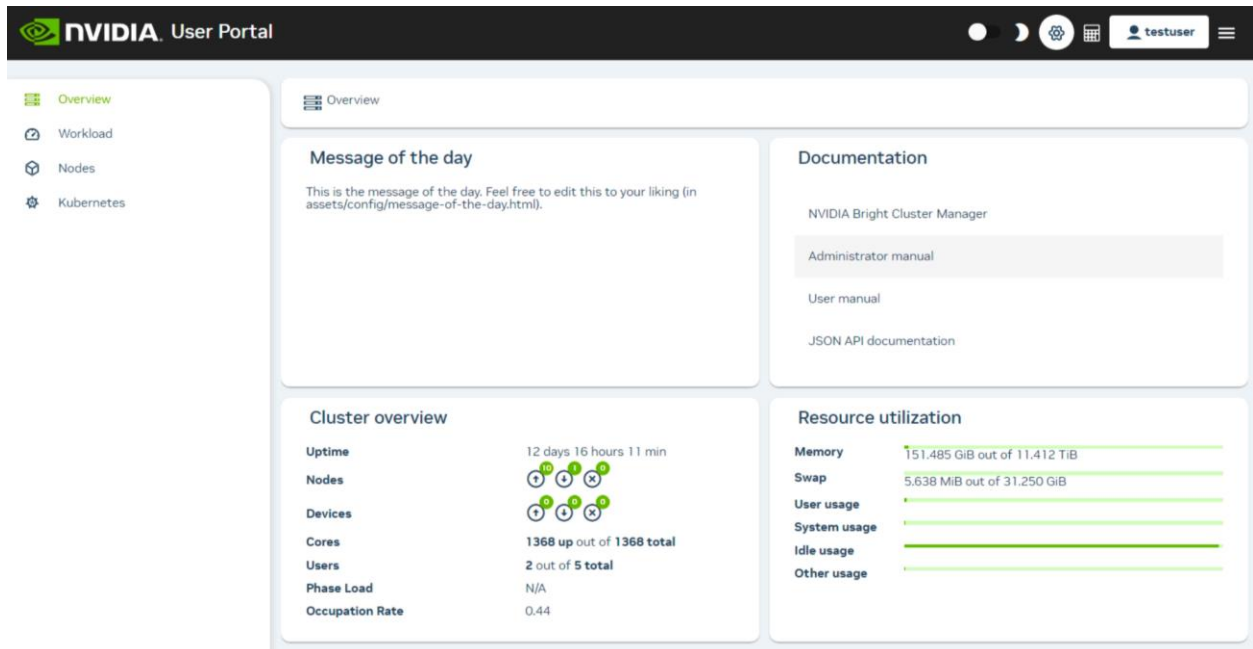
## 2. Base Command Manager

ผู้ใช้งานสามารถเข้าตรวจสอบระบบผ่าน Base Command Manager ได้

## 2.1 การเข้าใช้งานผ่าน Bright Cluster Manager - User Portal



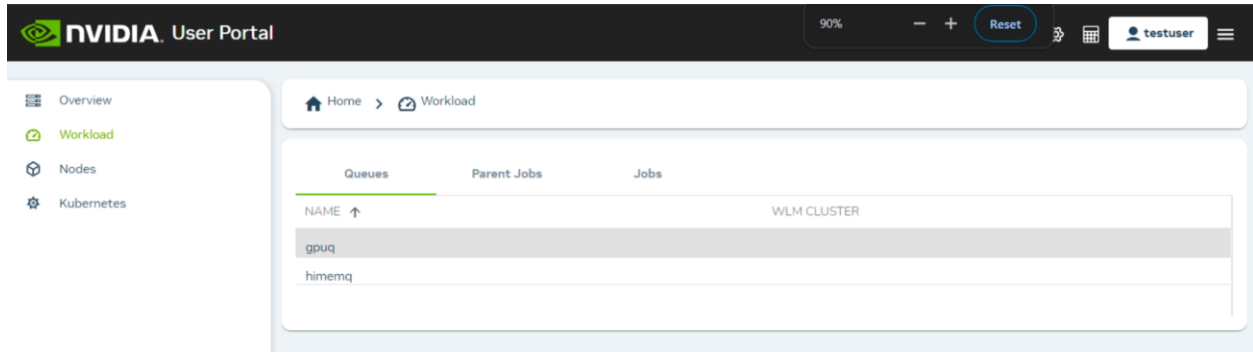
### 2.1.1 User Portal Overview



## 2.1.2 Workload

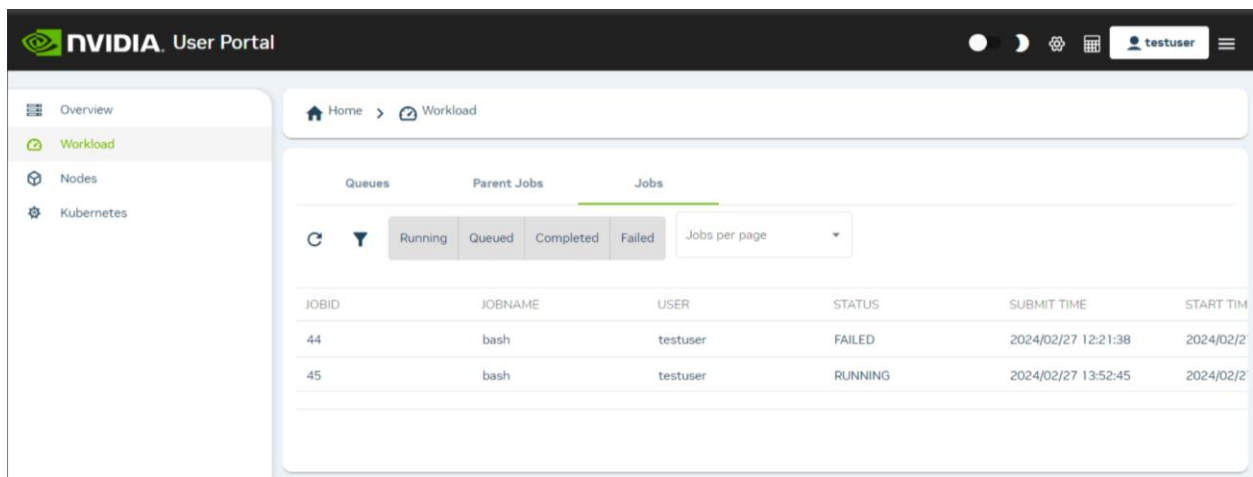
ผู้ใช้งานสามารถตรวจสอบ Workload ในระบบได้ทั้งจากหน้า Web GUI และ CLI การตรวจสอบผ่านหน้า Web GUI

ทำได้โดยเข้าไปที่ Workload



The screenshot shows the NVIDIA User Portal interface. The top navigation bar includes the NVIDIA logo, 'User Portal', a 90% battery indicator, a 'Reset' button, and a user profile for 'testuser'. The left sidebar contains navigation options: Overview, Workload (highlighted), Nodes, and Kubernetes. The main content area is titled 'Home > Workload' and features three tabs: 'Queues', 'Parent Jobs', and 'Jobs'. The 'Queues' tab is active, displaying a table with the following data:

NAME ↑	WLM CLUSTER
gpuq	
himemq	



The screenshot shows the NVIDIA User Portal interface with the 'Jobs' tab selected. The top navigation bar and sidebar are consistent with the previous screenshot. The main content area is titled 'Home > Workload' and features three tabs: 'Queues', 'Parent Jobs', and 'Jobs'. The 'Jobs' tab is active, displaying a table with the following data:

JOBID	JOBNAME	USER	STATUS	SUBMIT TIME	START TIME
44	bash	testuser	FAILED	2024/02/27 12:21:38	2024/02/27 12:21:38
45	bash	testuser	RUNNING	2024/02/27 13:52:45	2024/02/27 13:52:45

## 2.1.3 Nodes

HOS...	STATE	MEMORY	CORES	CPU	SPEED	GPU	NICS	IB	CATEGO...
br1	br1, status...	251.871 GiB	64	Intel(R) Xeo...	2 Hz		3	0	
br2	br2, status...	251.880 GiB	32	Intel(R) Xeo...	2 Hz		3	0	
dgx-01	dgx-01, sta...	1.968 TiB	256	AMD EPYC ...	2 Hz	NVIDIA NVL...	15	8	slurm-dgx
dgx-02	dgx-02, sta...	1.968 TiB	256	AMD EPYC ...	3 Hz	NVIDIA NVL...	15	8	slurm-dgx
dgx-03	dgx-03, sta...	1.968 TiB	256	AMD EPYC ...	2 Hz	NVIDIA NVL...	15	8	slurm-dgx
dgx-04	dgx-04, sta...	1.968 TiB	256	AMD EPYC ...	2 Hz	NVIDIA NVL...	16	8	k8s-dgx
k8s-m1	k8s-m1, st...	125.792 GiB	8	Intel(R) Xeo...	2 Hz		3	0	k8s-catego...
k8s-m2	k8s-m2, st...	125.792 GiB	8	Intel(R) Xeo...	2 Hz		3	0	k8s-catego...
k8s-m3	k8s-m3, st...	125.792 GiB	8	Intel(R) Xeo...	2 Hz		3	0	k8s-catego...
slurm-mem	slurm-me...	2.952 TiB	224	Intel(R) Xeo...	1 Hz		12	2	slurm-cate...
template	template, s...	0.000 B	0	Unknown	0 Hz		0	0	default

## 2.1.4 Kubernetes

NAME	VERSION	NODES	NAMESPACES	SERVICES	REPLICATIO...	PERSISTENT...	PERSISTENT...
default	1.27	4	17	36	0	0	0
		0	0	0	0	0	0

### QuickStart

From your local PC it should be possible to access the Kubernetes Dashboard URL at:

<https://dashboard.br1.ns.ku.ac.th:30443>

If that does not work, then the cluster administrator has probably modified the standard configuration, and should be consulted on how to access the Kubernetes Dashboard.

If the Dashboard is accessible, then a token needs to be obtained in order to authenticate against the Kubernetes Dashboard.

#### Obtaining and using the token

From the head node, the token can be obtained according to the following procedure:

```
$ module load kubernetes/default
$ NS=default
$ SECRET_NAME=$(kubectl get serviceaccount testuser -n $NS -o
  jsonpath='{.secrets[0].name}')
$ kubectl get secret -n $NS $SECRET_NAME -o
  jsonpath='{.data.token}' | base64 -d ; echo
If the namespace is not known to the user, then the cluster administrator
should be asked to provide this.
```

### Kubernetes cluster overview of default

Name	Version	PODs	jobs
default	1.27	0	0
State		0	0
Failed:		0	0
Pending:		67	0
Running:		5	4
Succeeded:		0	0
Unknown:			



### 3. Module Environment

#### 3.1 การแสดง module ที่สามารถใช้งานได้

```
module avail
```

ใช้สำหรับตรวจสอบ application ต่าง ๆ บน Module Environment

```
testuser@br1:~$ module avail
----- /cm/local/modulefiles -----
apptainer/apptainer.module  cm-setup/10.0      dot                mariadb-libs      python3
boost/1.81.0                cmd                freeipmi/1.6.10    module-git        python39
cluster-tools/10.0          cmjob              gcc/13.1.0         module-info       rocm-smi/4.3.0
cm-bios-tools               cmsh               ipmitool/1.8.19    modules           shared
cm-image/10.0               containerd/1.7.13  kubernetes/default/1.27.11-1.1  null              slurm/ai-ku-slurm/23.02.6
cm-scale/cm-scale.module    docker/24.0.9      luajit             openldap          use.own

----- /cm/shared/modulefiles -----
cm-pmix3/3.1.7              gdb/13.1           hwloc/1.11.13      jupyter/15.1.2    openmpi/gcc/64/4.1.5
cm-pmix4/4.1.3              hdf5_18/1.8.21    hwloc2/2.8.0       mvapich2/gcc/64/2.3.7  openmpi4/gcc/4.1.5
default-environment         hpl/2.3            jupyter-eg-kernel-wlm-py39/3.0.2  openblas/dynamic/0.3.18  ucx/1.10.1
```

โดยระบบประกอบไปด้วย 2 paths ได้แก่

- /cm/local/modulefiles เป็น path ที่จัดเก็บ Application ต่าง ๆ อยู่บนเครื่องนั้น ๆ โดยไม่ได้ถูกแชร์ไปยังเครื่องอื่น ๆ
- /cm/shared/modulefiles เป็น path ที่จัดเก็บ Application ต่าง ๆ ที่ถูกแชร์ไปยังทุก ๆ เครื่องในระบบ

#### 3.2 การเรียกใช้งาน module ด้วยคำสั่ง module load

```
module load <module>
```

```
testuser@br1:~$ module load jupyter
Loading jupyter/15.1.2
Loading requirement: python39
```

ใช้คำสั่ง module load แล้วตามด้วยชื่อ module ที่ต้องการ เพื่อโหลด module มาใช้งาน ส่วนในกรณีที่ต้องการทำให้ค่าคงอยู่อย่างถาวร ให้ใช้คำสั่งด้านล่าง

```
module unload <module>
```

```
testuser@br1:~$ module unload jupyter/15.1.2
Unloading jupyter/15.1.2
Unloading useless requirement: python39
```

## 4. Slurm Workload Management

### 4.1 Introduction

ในโครงการมีส่วนของ Workload Management คือ Slurm โดยพื้นฐาน Workflow คือ User จะทำการ submit job มาที่ Queue และระบบจะนำ job ไปประมวลผลใน Node ใน Cluster

ในการใช้งาน Slurm Workload Manager เบื้องต้น จะประกอบด้วย 4 คำสั่งหลัก ๆ ได้แก่

- sinfo
- srun
- sbatch
- squeue
- scancel

### 4.2 การตรวจสอบสถานะของ Node

ใช้คำสั่ง sinfo ในการตรวจสอบสถานะของทุกโหนดใน Cluster

```
root@br1:~# module load slurm
root@br1:~# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
gpuq      up       infinite    3    idle dgx-[01-03]
himemq*   up       infinite    1    mix  slurm-mem
root@br1:~#
```

ตารางรายละเอียดสถานะ sinfo

Status	รายละเอียด
idle	Node Online อยู่และสามารถรับ job ได้
alloc	Node Online อยู่มีงานรันอยู่เต็มแล้ว
drain	Node Online อยู่แต่ไม่สามารถรับjob ได้เนื่องจากมีการตรวจสอบ health ไม่ผ่าน, NHC หรือ admin กำหนด
drng	Node Online อยู่และกำลังจะเข้าสู่สถานะ drain

down	Node Offline
boot	Node ถูก Reboot โดย Slurm

## 4.3 การรันงาน

### 4.3.1 การรันงานด้วย sbatch

```

testuser@br1:~$ cat script.sh
#!/bin/bash
/bin/hostname
sleep 30

testuser@br1:~$ sbatch script.sh
Submitted batch job 48

testuser@br1:~$ squeue
          JOBID PARTITION    NAME    USER  ST       TIME  NODES
NODELIST (REASON)
          41      himemq interact user-tes  R 1-02:37:24      1 slurm-
mem
          48      himemq script.s testuser  R        0:04      1 slurm-
mem
testuser@br1:~$ ls
script.sh slurm-46.out slurm-47.out slurm-48.out
testuser@br1:~$ cat slurm-48.out
slurm-mem
testuser@br1:~$

```

### 4.3.2 การรันงานด้วย srun

ผู้ใช้งานสามารถ submit job เข้าไป Slurm ได้

```

testuser@br1:~$ srun hostname
slurm-mem

```

### 4.3.3 การรันงานแบบ Interactive Jobs ด้วย srun

ผู้ใช้งานสามารถสั่งรันแบบ Interactive Jobs ได้ โดยใส่คำสั่ง `--pty` ลงในการรัน srun

```

testuser@br1:~$ srun --pty bash
testuser@slurm-mem:~$ hostname
slurm-mem
testuser@slurm-mem:~$ exit

testuser@br1:~$ srun -p gpuq --pty bash
testuser@dgx-01:~$ hostname
dgx-01
testuser@dgx-01:~$ exit

```

#### 4.4 การกำหนดทรัพยากรในการรันงาน

ผู้ใช้งานสามารถรันงานแบบกำหนดทรัพยากรในแต่ละ job ได้ดังตัวอย่างต่อไปนี้

```

Request two tasks:
srun -n 2 <cmd>

```

```

Request two nodes, eight tasks per node, and one GPU
per task:
sbatch -N 2 --ntasks-per-node=8 --gpus-per-task=1
<cmd>

```

ตารางรายละเอียดสถานะ sbatch และ srun

Status	รายละเอียด
-N, --nodes=	Specify the total number of nodes to request
-n, --ntasks=	Specify the total number of tasks to request
--ntasks-per-node=	Specify the number of tasks per node
-G, --gpus=	Total number of GPUs to allocate for the job
--gpus-per-task=	Number of GPUs per task
--gpus-per-node=	Number of GPUs to be allocated per node

#### 4.5 การตรวจสอบงานที่รันอยู่

ผู้ใช้งานสามารถตรวจสอบ job ที่รันอยู่ได้ด้วยคำสั่ง squeue

```
testuser@dgx-01:~$ squeue -a -l
Tue Feb 27 14:13:17 2024
      JOBID PARTITION     NAME     USER      STATE       TIME  TIME_LIMI  NODES
NODELIST (REASON)
      53      gpuq      bash testuser  RUNNING      4:47  UNLIMITED    1 dgx-01
      41      himemq  interact user-tes  RUNNING  1-02:50:15  UNLIMITED    1 slurm-mem
```

#### 4.6 การยกเลิกงาน

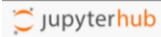
ผู้ใช้งานสามารถยกเลิก job ที่รันอยู่ได้ด้วยคำสั่ง scancel

```
scancel JOBID
```

## 5. JupyterHub

ในโครงการมีส่วนของการ JupyterHub ได้ติดตั้งไว้ในระบบ สำหรับการพัฒนาหรือ compile โปรแกรมที่อยู่บนพื้นฐานของภาษา Python โดยสามารถเข้าใช้งานผ่าน Web GUI ได้ โดย

- 1) เปิด Browser เข้าไปที่ <http://br1.paas.ku.ac.th:8000/>



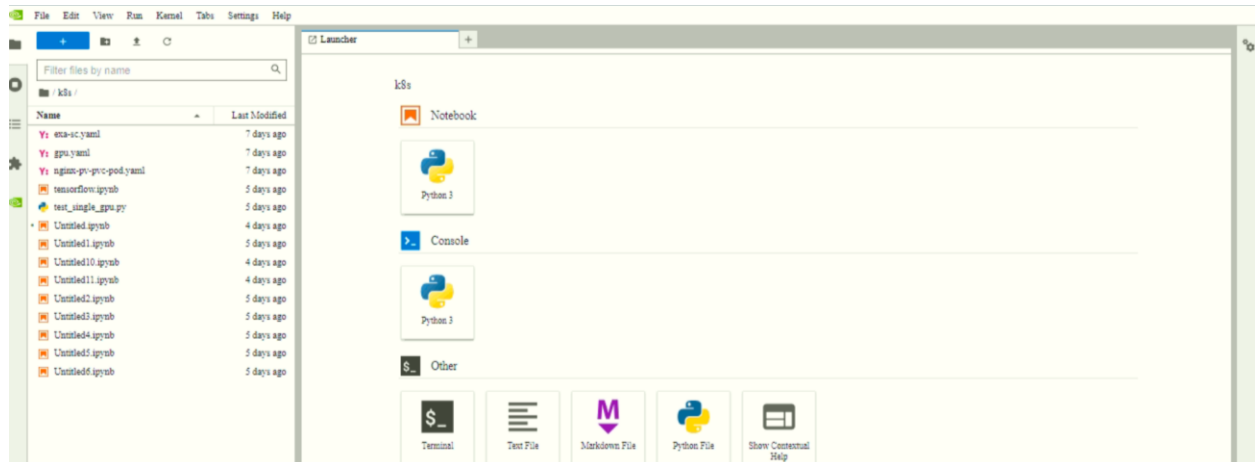
Sign in

Username:

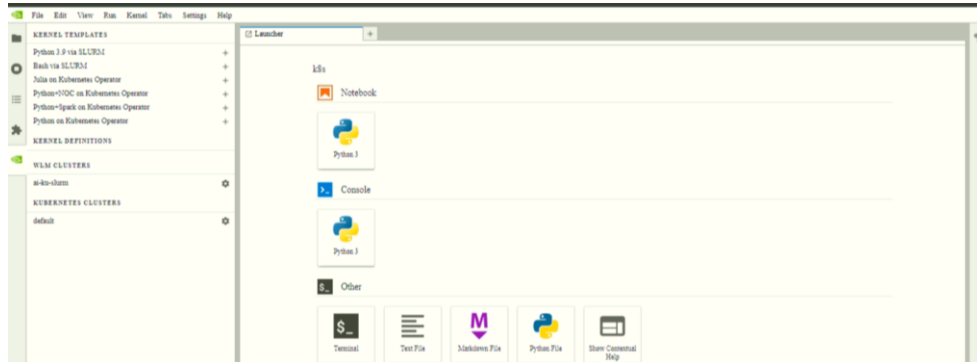
Password:

Sign in

- 2) ทำการ Login เข้าใช้งาน
- 3) จะพบกับหน้าแรกของ JupyterHub



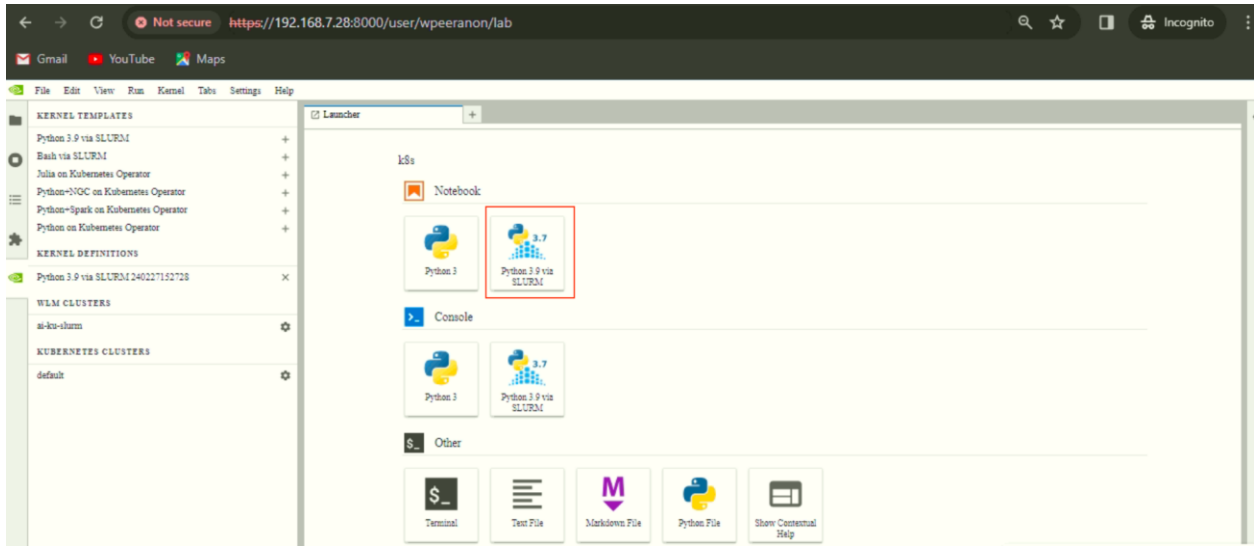
- 4) เมื่อไปที่เมนู Kernel Templates จะพบกับการใช้งาน Jupyter ร่วมกับ Slurm และ Kubernetes



- 5) ผู้ใช้งานสามารถเลือกสร้าง Kernel Template ได้ตามต้องการ โดยคลิกที่เครื่องหมาย + ด้านหลัง Kernel Template นั้น ๆ



- 6) ที่หน้า Launcher จะปรากฏ Kernel Template ที่สร้างขึ้นมา





## 6. Kubernetes

### 6.1 การเข้าใช้งาน

สามารถเข้าใช้งานผ่าน CLI ได้ โดยเมื่อเข้าใช้งานมาแล้ว ให้รัน `module load kubernetes` ก่อนการใช้งานทุกครั้ง สามารถตรวจสอบสถานะของ Kubernetes Cluster ได้ด้วยคำสั่ง `kubernetes get nodes`

```
wpeeranon@br1:~$ module load kubernetes/  
wpeeranon@br1:~$ kubectl get nodes  
NAME          STATUS    ROLES          AGE    VERSION  
dgx-04        Ready    worker         4d23h  v1.27.11  
k8s-m1        Ready    control-plane,master 4d23h  v1.27.11  
k8s-m2        Ready    control-plane,master 4d23h  v1.27.11  
k8s-m3        Ready    control-plane,master 4d23h  v1.27.11
```

### 6.2 การสร้าง Pod

สามารถสร้าง Pod ได้จาก YAML ไฟล์ โดยรันคำสั่งดังตัวอย่างด้านล่าง

```
wpeeranon@br1:~/k8s$ cat gpu.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  name: peeranon-pytorch-gpu-pod  
spec:  
  restartPolicy: Never  
  containers:  
  - name: pytorch-container  
    image: nvcr.io/nvidia/pytorch:22.08-py3  
    command: ["nvidia-smi"]  
    resources:  
      limits:  
        nvidia.com/gpu: 1
```

### 6.3 การลบ Pod

สามารถลบ Pod ได้จาก YAML ไฟล์ โดยรันคำสั่งดังตัวอย่างด้านล่าง

```
wpeeranon@br1:~/k8s$ kubectl delete -f gpu.yaml
```

#### 6.4 การตรวจสอบ Pod

สามารถตรวจสอบการทำงานของ Pod โดยรันคำสั่งดังตัวอย่างด้านล่าง

```
wpeeranon@br1:~/k8s$ kubectl describe pods peeranon-pytorch-gpu-  
Name:                peeranon-pytorch-gpu-pod  
Namespace:           wpeeranon-restricted  
Priority:             0  
Service Account:    default  
Node:                dgx-04/192.168.7.30  
Start Time:         Tue, 27 Feb 2024 15:36:18 +0700  
.br/>.br/>.br/>Events:  
  Type            Reason              Age   From                Message  
  ----            -  
  Normal         Scheduled           86s   default-scheduler  Successfully  
wpeeranon-restricted/peeranon-pytorch-gpu-pod to dgx-04  
  Normal         AddedInterface     80s   multus              Add eth0  
[172.29.133.109/32] from k8s-pod-network  
  Normal         Pulling            80s   kubelet            Pulling image  
"nvcr.io/nvidia/pytorch:22.08-py3"
```

และสามารถดู Log การทำงานของ Pod โดยรันคำสั่งดังตัวอย่างด้านล่าง

```
wpeeranon@br1:~/k8s$ kubectl logs peeranon-pytorch-gpu-pod  
Tue Feb 27 08:39:46 2024
```

```
+-----+  
| NVIDIA-SMI 535.154.05                Driver Version: 535.154.05    CUDA Vers  
|-----+-----+-----+  
| GPU   Name                               Persistence-M | Bus-Id        Disp.A | Volatil  
| Fan   Temp   Perf             Pwr:Usage/Cap |      Memory-Usage | GPU-Util  
|-----+-----+-----+  
|    0   NVIDIA A100-SXM4-80GB                On      | 00000000:0F:00.0 Off  |  
| N/A    26C    P0              59W / 400W | 0MiB / 81920MiB |      0%  
|-----+-----+-----+
```

```
+-----+  
| Processes:  
| GPU   GI    CI          PID    Type    Process name  
|      ID    ID  
|-----+-----+-----+  
| No running processes found  
+-----+
```

## 7. Singularity / Apptainer

ผู้ใช้สามารถใช้งาน Singularity/Apptainer ร่วมกับ Slurm ได้ โดย Singular Image File (SIF) ส่วนกลางจะถูกจัดเก็บไว้ที่ /cm/shared/sif นอกจากนี้ ผู้ใช้อาจเลือกที่จะเก็บ SIF file ไว้ที่ home directory ของตนเองก็สามารถทำได้เช่นกัน

วิธีการทำงานมีขั้นตอน ดังนี้

- ทำการจองทรัพยากรโดยใช้ salloc ในที่นี้ขอระบุว่าต้องการใช้งาน 2 GPU ที่ partition gpuq

```
$ salloc -p gpuq --gpus=2 --cpus-per-task=1 --time=2:0:0
salloc: Granted job allocation 97
salloc: Waiting for resource configuration
salloc: Nodes dgx-01 are ready for job
```
- จากนั้นย้ายการทำงานไปที่ Compute node ที่ระบบ Slurm จัดสรรให้ ในที่นี้ คือเครื่อง dgx-01 สังเกตว่า prompt จะขึ้นชื่อเครื่องที่ทำงานอยู่ ณ ขณะนั้น

```
$ srun --pty bash
```
- ทำการ module load singularity แล้ว run คำสั่ง singularity เพื่อ start Container ที่อยู่ในรูป SIF format

```
$ module load singularity
$ singularity shell --nv -B /cm/shared -B /scratch /cm/shared/sif/pytorch_23.12-py3.sif
```

แสดงภาพรวมการทำงานของ Slurm + Singularity/Apptainer

```
testuser@br1:~/1-Slurm$ salloc -p gpuq --gpus=2 --cpus-per-task=1 --time=2:0:0
salloc: Granted job allocation 97
salloc: Waiting for resource configuration
salloc: Nodes dgx-01 are ready for job
testuser@br1:~/1-Slurm$ srun --pty bash
# Now you are in Compute node
testuser@dgx-01:~/1-Slurm$
testuser@dgx-01:~/1-Slurm$ module load singularity
testuser@dgx-01:~/1-Slurm$ singularity shell --nv -B /cm/shared -B /scratch
/cm/shared/sif/pytorch_23.12-py3.sif
```

เมื่อเข้ามาอยู่ภายใน Container แล้ว (สังเกตที่ prompt จะเป็น Apptainer>) ทดลองใช้คำสั่ง 'nvidia-smi' เพื่อดูจำนวน GPU พบว่าระบบมองเห็น GPU ตามที่จองมาในตอนต้น

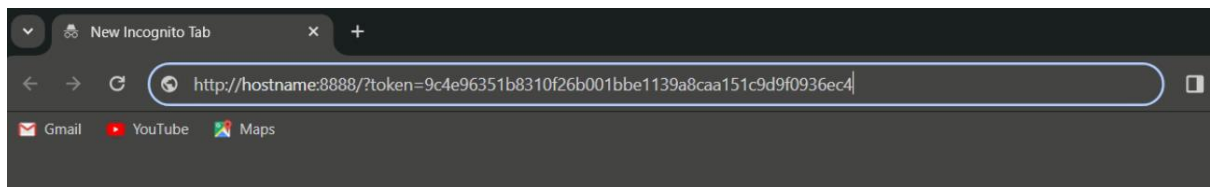
```
(base) wpeeranon@br1:~$ srun -p gpuq --gpus=1 --cpus-per-task=1 --time=2:0:0 --pty bash
(base) wpeeranon@dgx-01:~$ module load sin
ERROR: Unable to locate a modulefile for 'sin'
(base) wpeeranon@dgx-01:~$
(base) wpeeranon@dgx-01:~$
(base) wpeeranon@dgx-01:~$
(base) wpeeranon@dgx-01:~$
(base) wpeeranon@dgx-01:~$ module av
----- /cm/local/modulefiles -----
apptainer/apptainer.module  cmd      freeipmi/1.6.10  luajit      modules  python39
boost/1.81.0                cmjob    gcc/13.1.0      mariadb-libs  null     shared
cluster-tools/10.0         cuda-dcgm/3.1.8.1  gcc/64/4.1.7a1  module-git   openldap  slurm/ai-ku-slurm/23.02.6
cm-bios-tools              dot      ipmitool/1.8.19  module-info  python3   use.own
----- /cm/shared/modulefiles -----
anaconda3/24.1.2            default-environment  hwloc/1.11.13                micromamba/1.5.6            openmpi/gcc/64/4.1.5
apptainer/apptainer        gdb/13.1             hwloc/2.8.0                  mvapich2/gcc/64/2.3.7      openmpi4/gcc/4.1.5
cm-pm1x3/3.1.7             hdf5_18/1.8.21      jupyter-eg-kernel-wlm-py39/3.0.2  ncbi-blast/2.15.0+        singularity/4.1.1
cm-pm1x4/4.1.3            hpl/2.3              jupyter/15.1.2                openblas/dynamic/0.3.18    ucx/1.10.1
(base) wpeeranon@dgx-01:~$ module
module
module-assistant modulecmd
(base) wpeeranon@dgx-01:~$ module load singularity
(base) wpeeranon@dgx-01:~$ singularity shell --nv /cm/shared/sif/pytorch_23.12-py3.sif
Singularity> nvidia-smi
Fri Mar 1 14:54:02 2024
+-----+
| NVIDIA-SMI 535.154.05                Driver Version: 535.154.05   CUDA Version: 12.3         |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                   Persistence-M | Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf              Pwr:Usage/Cap |      |      | GPU-Util  Compute M. |
|====+=====+====+=====+=====+=====+=====+=====+
| 0    NVIDIA A100-SXM4-80GB     On          | 00000000:07:00:0  Off |          | Default  |
| N/A  29C    P0                 59W / 400W   |      |      | 0%        Disabled |
+-----+-----+-----+-----+-----+-----+
+-----+
| Processes: |
| GPU   GI   CI          PID    Type   Process name                      GPU Memory |
| ID   ID   ID              |              |           | Usage              |
+-----+-----+-----+-----+-----+
| No running processes found |
+-----+
```

จากนั้น run คำสั่งเพื่อ start JupyterLab ในกรณีที่ต้องการใช้ GUI

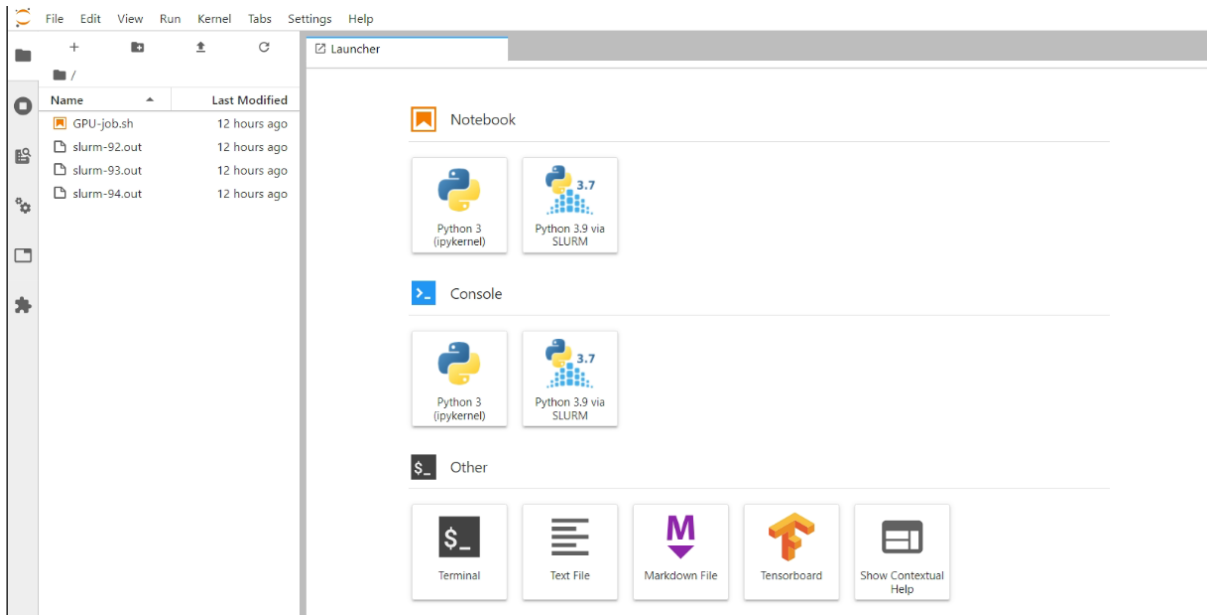
```
Singularity> jupyter-lab --no-browser --ip 0.0.0.0
[I 14:55:24.967 LabApp] Writing notebook server cookie secret to /home/wpeeranon/.local/share/jupyter/runtime/notebook_cookie_secret
[I 14:55:26.552 LabApp] jupyter_tensorboard extension loaded.
[I 14:55:26.558 LabApp] JupyterLab extension loaded from /usr/local/lib/python3.10/dist-packages/jupyterlab
[I 14:55:26.558 LabApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[I 14:55:26.561 LabApp] [Jupyter Server Extension] NotebookApp.contents_manager_class is (a subclass of) jupyter.TextFileContentsManager already - OK
[I 14:55:26.567 LabApp] Serving notebooks from local directory: /home/wpeeranon
[I 14:55:26.567 LabApp] Jupyter Notebook 6.4.10 is running at:
[I 14:55:26.567 LabApp] http://hostname:8888/?token=9c4e96351b8310f26b001bbe1139a8caa151c9d9f0936ec4
[I 14:55:26.567 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:55:26.597 LabApp]

To access the notebook, open this file in a browser:
file:///home/wpeeranon/.local/share/jupyter/runtime/nbserver-229566-open.html
Or copy and paste this URL:
http://hostname:8888/?token=9c4e96351b8310f26b001bbe1139a8caa151c9d9f0936ec4
```

ทำการ copy URL ด้านบน แล้วเปลี่ยนชื่อ hostname เป็น IP ของเครื่องให้ถูกต้อง



จะได้หน้าจอ JupyterLab สำหรับเขียน code



เมื่อต้องการเลิกใช้งาน ไปที่ File เมนูแล้วเลือก “Shut Down” เพื่อปิด JupyterLab

