
HPC Documentation

Release 0.0

HPC group

Oct 07, 2022

QUICK START GUIDE

1	Getting an account	3
2	Accessing the Chalawan	5
3	Connection from Outside NARIT Work	7
4	Transferring Files & Data	9
5	Using Module Environments	11
6	Job Submission	13
7	Compute	15
8	Storage	17
9	Policy & Queue	19
10	Slurm Credit Allocation & Application	21
11	Acknowledgement & Publication	23
12	Slurm Workload Manager	25
13	Running Parallel Jobs	29
14	Job Management	31
15	Slurm Cheatsheet	33
16	Which software is installed on Chalawan	35
17	Installing Software on the Chalawan HPC	37
18	Software Module Scheme & Module environment	39
19	Python (Version, Environment and Packages)	41
20	Basic Linux commands	43
21	Using JupyterLab	45
22	Contact us	51

Chalawan is the high performance computing cluster that powered through the use of parallel programming, increasingly relies on massive datasets and compute-intensive workloads with a wide range of applicability and deployment.

GETTING AN ACCOUNT

To start using Chalawan Cluster, user must first submit an online application to get computing time credit and storage space. The merit of your proposal will also be considered, taking into account amount of requested resources. This will also provide our clusteradmin with a better understanding of what our users need and how best to prepare the environments and applications for you.

1.1 Online application

1.1.1 Sign Up

Sign up for an [Online Application](#) account. If you are eligible, your account should be activated within one working day. After the account is activated, please sign in to the system with the username (email address) and password you provided at the sign up and start filling the online application form.

1.1.2 Submit your application

Fill an [online application](#) form and submit. Once completed, you should receive notification of our decision within a week and further query regarding required software and setup if any.

1.1.3 Account setup notification

Once your account and required setup are ready, our system admin will send you an email regarding your account details, allocated *Slurm credit*, *storage* quota and how to login to Chalawan cluster.

ACCESSING THE CHALAWAN

2.1 The command-line interface

Our operating system is based on GNU/Linux. Thus, a command-line interface or command language interpreter (CLI) is the primary mean of interaction with our HPC. In case you are not familiar with the command-line interface, free-online course at Codecademy is a good place to start.

2.2 Accessing the Chalawan

For Microsoft Windows user, see Connect to the Remote Server from Microsoft Windows.

The Chalawan cluster is an isolated system which resides in the NARIT's internal network. At the present time, we have two systems, Castor and Pollux (hereafter the computing systems).

Castor is the old system which is assigned with the IP address 192.168.5.100. It contains 16 traditional Compute nodes suited for CPU-intensive tasks. Pollux is the newest one assigned with the IP address 192.168.5.105. It contains 3 GPU nodes and 3 traditional Compute node which have been refurbished from Castor. If you are using the internet inside NARIT network you can directly connect to these systems via the Secure shell (ssh) command.

2.3 Connection from outside NARIT network

However, if you are using the internet outside NARIT, you need to log in to the gateway machine, A.K.A. stargate, first. The gateway machine's IP address and other information are given to you once you get the permission to access the Chalawan Cluster.

2.4 Secure shell (ssh) through an intermediate host (the gateway)

This is the easiest method that using the ProxyJump directive. If this method doesn't work for you because you are using the very old version of ssh, please read the next section.

To use ProxyJump, you can simply add the flag `-J` followed by `user@gateway.ip:port`. The example below shows how to connect to Castor (don't forget to replace gateway.ip and port with the given information from the email).

```
[user@local ~]$ ssh -J user@gateway.ip:port user@192.168.5.100
```

asasasa

CONNECTION FROM OUTSIDE NARIT WORK

3.1 topics

wording...

TRANSFERRING FILES & DATA

4.1 rsync & scp (secure copy)

4.1.1 Jump host

4.1.2 Multi-stage

4.2 Cloud Storage

It is possible to mount your cloud storage drive using [Rclone](#).

4.3 JupyterLab Interface

While accessing the cluster via [Chalawan JupyterLab](#), user may use the interface to upload or download local file into or out off the cluster (see [Using Jupyterlab](#)).

Note: The upload filesize limit via JupyterLab interface is set to 100MB. User will be asked to confirm when trying to upload a file with size larger than 15MB. Please consider using alternative method to upload/download many large files as this would negatively affect connections of other users.

USING MODULE ENVIRONMENTS

5.1 Topic

wording...

JOB SUBMISSION

6.1 Topic

wording...

COMPUTE

7.1 Castor & Pollux

7.1.1 Node Configuration

STORAGE

8.1 Lustre

wording...

POLICY & QUEUE

9.1 topic

wording...

SLURM CREDIT ALLOCATION & APPLICATION

10.1 topic

wording...

ACKNOWLEDGEMENT & PUBLICATION

11.1 topic

wording...

SLURM WORKLOAD MANAGER

12.1 Nodes and partitions

Before submitting any job to Pollux, you must learn about available resources. To do that, we use `sinfo` to view information about Compute nodes and partitions. Once it is run, the command will print the information like the output below.

```
[user@pollux]$ sinfo
HOSTNAMES PARTITION      AVAIL CPUS(A/I/O/T) CPU_LOAD ALLOCMEM FREE_MEM GRES      STATE
↪TIMELIMIT
pollux1    chalawan_gpu  up    0/24/0/24    3.68    0        54028    gpu:4    idle
↪infinite
pollux2    chalawan_gpu  up    0/28/0/28    3.71    0        246330   gpu:4    idle
↪infinite
pollux3    chalawan_gpu  up    0/28/0/28    3.60    0        246343   gpu:4    idle
↪infinite
castor1    chalawan_cpu* up    0/16/0/16    0.01    0        55444    (null)   idle
↪infinite
castor2    chalawan_cpu* up    0/16/0/16    0.01    0        55434    (null)   idle
↪infinite
castor3    chalawan_cpu* up    0/16/0/16    0.01    0        55455    (null)   idle
↪infinite
castor4    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor5    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor6    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor7    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor8    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor9    chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor10   chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor11   chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
castor12   chalawan_cpu* up    0/16/0/28    0.01    0        92160    (null)   idle
↪infinite
```

(continues on next page)

(continued from previous page)

castor13	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								
castor14	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								
castor15	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								
castor16	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								
castor17	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								
castor18	chalawan_cpu*	up	0/16/0/28	0.01	0	92160	(null)	idle	↪
	↪infinite								

Here we introduce the new field, PARTITION. Partition is like a specific group of Compute nodes. Note that the suffix “*” identifies the default partition. AVAIL shows a partition’s state: up or down while CPUS(A/I/O/T) shows count of nodes with this particular configuration by node state in the form “available/idle/other/total”.

12.2 Basic job submission

12.2.1 Slurm environment variables

Upon startup, sbatch will read and handle the options set in the environment variables. Note that environment variables will override any options set in a batch script, and command line options will override any environment variables. The full details are on sbatch manual (`man sbatch`), section “INPUT ENVIRONMENT VARIABLES”. For example, we have a script name `task1`.

```
#!/bin/bash

#SBATCH -J task1           # Job name
#SBATCH -t 00:01:00       # Run time (hh:mm:ss)

echo "Hello World!"
```

The default partition is `chalawan_cpu`, but we want to submit a job to `chalawan_gpu` instead, we can do either

```
[user@pollux]$ sbatch -p chalawan_gpu ./task1.slurm
```

or

```
[user@pollux]$ export SBATCH_PARTITION="chalawan_gpu" [user@pollux]$ sbatch ./task1.slurm
```

There are also output environment variables of the batch script which are set by the Slurm controller, e.g., `SLURM_JOB_ID`, `SLURM_CPUS_ON_NODE`. For the full details, see “OUTPUT ENVIRONMENT VARIABLES” on sbatch manual (`man sbatch`). You may combine them with your script for convenience. The example below shows the results when we print out some of these values.

```
[user@pollux]$ cat ./echo.slurm
#!/bin/bash

#SBATCH -J echo           # Job name
```

(continues on next page)

(continued from previous page)

```
#SBATCH -o %x-%j.out           # Name of stdout output file

echo "Job name: $SLURM_JOB_NAME"
echo "Job ID: $SLURM_JOB_ID"
[user@pollux]$ sbatch ./echo.slurm
[user@pollux]$ cat ./echo-130.slurm
Job name: echo
Job ID: 130
```

We use the command `sbatch` followed by a batch script to submit a job to Slurm. `sbatch` then exits immediately after the script is successfully transferred to the Slurm controller assigned a Slurm job ID. The batch script is not necessarily granted resources immediately, it may sit in the queue of pending jobs for some time before its required resources become available.

```
[user@pollux]$ sbatch [OPTIONS...] executable [args...]
```

The batch may contain options preceded with `#SBATCH` before any executable commands in the script. For example, we create a simple batch script to print a string “Hello World!” called `task1.slurm`. Inside the file looks like this

```
#!/bin/bash

#SBATCH -J task1           # Job name
#SBATCH -t 00:01:00       # Run time (hh:mm:ss)

echo "Hello World!"
```

After submission with the command `sbatch task1.slurm`, if there is an empty slot, your task will run and exit instantly. You will find the output file, `slurm-%j.out` at the current working directory where `%j` is replaced with the job allocation number. The words “Hello World!” is appeared inside that output file. By default, both standard output and standard error are directed to the same file.

```
[user@pollux]$ sbatch ./task1.slurm
Submitted batch job 128
[user@pollux]$ cat ./slurm-128.out
Hello World!
```

12.2.2 Batch vs Interactive jobs

We use the command `sbatch` followed by a batch script to submit a job to Slurm. `sbatch` then exits immediately after the script is successfully transferred to the Slurm controller assigned a Slurm job ID. The batch script is not necessarily granted resources immediately, it may sit in the queue of pending jobs for some time before its required resources become available.

```
[user@pollux]$ sbatch [OPTIONS...] executable [args...]
```

The batch may contain options preceded with `#SBATCH` before any executable commands in the script. For example, we create a simple batch script to print a string “Hello World!” called `task1.slurm`. Inside the file looks like this

```
#!/bin/bash

#SBATCH -J task1           # Job name
#SBATCH -t 00:01:00       # Run time (hh:mm:ss)
```

(continues on next page)

(continued from previous page)

```
echo "Hello World!"
```

After submission with the command `sbatch task1.slurm`, if there is an empty slot, your task will run and exit instantly. You will find the output file, `slurm-%j.%j.out` at the current working directory where `%j` is replaced with the job allocation number. The words “Hello World!” is appeared inside that output file. By default, both standard output and standard error are directed to the same file.

```
[user@pollux]$ sbatch ./task1.slurm
Submitted batch job 128
[user@pollux]$ cat ./slurm-128.out
Hello World!
```

12.3 Frequently used sbatch options

There are many options you can add to a script file. The frequently used options are listed below. Each option must be preceded with `#SBATCH`. For other available options, you can learn from the Slurm website or using the command `sbatch -h` or `man sbatch`.

Option	Description
<code>-J, --job-name=<name></code>	name of job
<code>-N, --nodes=<N></code>	number of nodes on which to run ($N = \min[-max]$)
<code>-n<count></code>	number of tasks to run
<code>-c, --cpus-per-task=<ncpus></code>	number of cpus required per task
<code>-e, --error=<err></code>	file for batch script's standard error
<code>-o, --output=<out></code>	file for batch script's standard output
<code>-p, --partition=<partition></code>	partition requested
<code>-t, --time=<minutes></code>	time limit
<code>--mem=<MB></code>	minimum amount of real memory
<code>--gres=<list></code>	required generic resources

RUNNING PARALLEL JOBS

13.1 Shared memory

Shared memory job runs multiple processes which share memory together on one machine. User should write a script to request for running a job on a single Compute node with the following maximum number of threads on each machine:

- 16 on the partition chalawan_cpu
- 24 on the node pollux1
- 28 on the nodes pollux2 and pollux3

It is also recommended for a program is written with [OpenMP](#) directive and C/C++ multi-threading. An example script is displayed here

```
#!/bin/bash

#SBATCH -J shared      # Job name
#SBATCH -N 1           # Total number of nodes requested
#SBATCH -n 16          # Total number of mpi tasks
#SBATCH -t 120:00:00   # Run time (hh:mm:ss)

mpirun -np 16 -ppn 1 [ options ] <program> [ <args> ]
```

13.2 Distributed memory

For distributed memory, each process has its own memory and does not share with any others. A distributed memory job can run across multiple Compute nodes. It requires a program that is written with the specific parallel directive, e.g. the Message Passing Interface (MPI). Moreover, it requires an additional set up to scatter the processes over Compute nodes. Suppose we want to run a job with 16 processes which spawn 4 processes on each compute node, we may write:

```
#!/bin/bash

#SBATCH -J distributed # Job name
#SBATCH -N 4           # Total number of nodes requested
#SBATCH -n 16          # Total number of mpi tasks
#SBATCH --ntasks-per-node=4 # Total number of tasks per one node
#SBATCH -t 120:00:00   # Run time (hh:mm:ss)

mpirun -np 16 -ppn 4 [ options ] <program> [ <args> ]
```


JOB MANAGEMENT

14.1 Job status

However, if there is no empty slot, the job will be listed in a pending (**PD**) state. You can view it by using the command `squeue`. The output column **ST** stands for state. A running job is displayed with a state **R**. The Compute node where the job is running on is shown in the last column.

```
[user@pollux]$ squeue
JOBID PARTITION   NAME     USER ST       TIME  NODES NODELIST(REASON)
1587  chalawan_    task2    goku PD        0:00      1 (Priority)
1585  chalawan_    task1    user PD        0:00      1 (Resources)
1584  chalawan_    task0    vegeta R    3:21:49      1 pollux3
```

14.2 Job deletion

To remove a running job or a pending job from the queue, please use the command `scancel` followed by the job id. To cancel all your jobs (running and pending) you can run `scancel -u <username>`.

14.3 Job history

`sacct` displays accounting data for all jobs and job steps in the Slurm job accounting log or Slurm database.

SLURM CHEATSHEET

All Slurm command started with 's' followed by abbreviation of action word. Here we list the basic commands for submitting or deleting a job and query the information from it.

`sacct` is used to report job or job step accounting information about active or completed jobs.

`salloc` is used to allocate resources for a job in real time. Typically this is used to allocate resources and spawn a shell. The shell is then used to execute `srun` commands to launch parallel tasks.

`sbatch` is used to submit a job script for later execution. The script will typically contain one or more `srun` commands to launch parallel tasks.

`scancel` is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

`sinfo` reports the state of partitions and nodes managed by Slurm. It has a wide variety of filtering, sorting, and formatting options.

`squeue` reports the state of jobs or job steps. It has a wide variety of filtering, sorting, and formatting options. By default, it reports the running jobs in priority order and then the pending jobs in priority order.

WHICH SOFTWARE IS INSTALLED ON CHALAWAN

For all users on our system, the Chalawan HPC clusters come with a set of softwares preconfigured. The software on the Chalawan HPC may be divided into two categories:

- General software
- Sharded licence software

Users can manage the preloaded software and compilation tools which are available on the Chalawan HPC system by using the “module” tool.

To list the loaded softwares, type,

```
module list
```

If you find out your program is not available in our HPC system, you can try to install your program by using compilation tools or contact our HPC staff.

INSTALLING SOFTWARE ON THE CHALAWAN HPC

If you find out your desirable software is not installed on our HPC system, you can use these instructions to install it. First thing that you must know before installing the software is which compilation tools or the system requirements are used for installing/compiling the software. Then, you can follow these “general” steps to install your software.

17.1 1. Preparaing a software

You must upload/download your code into your desirable location, for example, you can download the software package via wget by typing,

```
wget -p internet/sourcecode.tar.gz foldername
```

We suggest that you place it in the home folder (e.g., “/home/username/prgogram/”). If the software is archives, you can extract it into the specific folder by typing

```
tar -zxvf sourcecode.tar.gz softwarefolder
```

17.2 2. Installing a software

Before starting to install a software, you must check the system requirements. If you download the source code from the developer, you can find the installation instructions in the package (e.g., README file). You can load the specific compiler version (for example, gnu version 9.3.0) by typing,

```
module load gnu9/9.3.0
```

Please note that, when you loaded the different version of softwares, it might be changed the compiler version accordingly. Therefore, please make sure that all pre-installing modules/compilers are matched.

To install a software, you first move the terminal into the software folder and then starting config the software by typing

```
cd softwarefolder  
./configure
```

Then, we compile all source codes by typing

```
make  
make install
```

If all sources are compiled correctly, you will find the new installed software file in that folder.

17.3 3. Loading a software

In order to easily use your installed software, we can load or make an alias by setting it in the bash script.

First, you open the bash script (.bashrc) located in your home folder “/home/username/.bashrc”.

Then, you load your software and make an alias to call your software by adding this line into the “.bashrc” file,

```
alias myprogram='softwarefolder/sotwarename'
```

where “myprogram” is the software alias that you can change and “softwarefolder/sotwarename” is the location of your software. Note that, before setting an alias, you must check that this alias is available first (ex., typing this alias in terminal).

Finally, you reload the bash script by typing (one time),

```
source .bashrc
```

After you finished all theses steps, you can call your software via the alias by typing,

```
myprogram
```

SOFTWARE MODULE SCHEME & MODULE ENVIRONMENT

The Chalawan HPC is already pre-installed software packages for all users. To find out a list of software package that is available, type:

```
module av
```

To check a list of software packages is currently loaded, type:

```
module list
```

18.1 Load/Unload module

If you would like to change/swap the module, you can load /unload the module by typing:

```
module load modulename  
module unload modulename
```


PYTHON (VERSION, ENVIRONMENT AND PACKAGES)

19.1 Default Python version on the Chalawan HPC

Python is preinstalled on the Chalawan HPC clusters. You can call the Python program by typing,

- Python version 2.x

```
python
```

- Python version 3.x

```
python3
```

It is the default program, and pre-installed some packages, for example, astropy, matplotlib, numpy, pandas, scipy and etc. You can easily run your code by typing,

```
python myprogram.py  
python3 myprogram.py
```

19.2 Create your environment

Users may be needed to manage the packages which is not allowed with the default. Therefore, if you want to manage the installed packaged, you should create your own environment (packages). Note that, due to the limitation of free space in your home folder, you should recheck the freespace and cache files when you installed the new packages.

You can use the following steps to crate your own environment;

To load the Anaconda module to manage the environment, type,

```
module load anaconda
```

To create your own environment, for example, the envirotnment name “myenv”, type,

```
conda create -n myenv
```

If you want to set a specific version of Python, type:

```
conda create -n myenv python=3.7
```

When you needed to load your environment for coding or running your program, type,

```
module load anaconda
conda activate myenv
```

19.3 Managing Packages

We can use Anaconda to manage the Python packages.

To install the new package, type,

```
conda install packagename
```

Or, via the “pip” module,

```
pip install packagename
```

To uninstall the package, type,

```
pip uninstall packagename
```

Packages that you installed may be upgradable for fixing some bugs and improving performance.

To update all packages of your environment, type,

```
conda update -n myenv --update-all
```

Or, with a specific package, type,

```
pip install packagename --upgrade
```

BASIC LINUX COMMANDS

In this page, you can learn the basic commands of linux systems. It helps to work effectively in Linux.

20.1 File Management

20.1.1 pwd

Find out what directory you are currently in.

20.1.2 cd

Move to another working directory

20.1.3 ls

List all files and directories.

20.1.4 cp

Copy file

20.1.5 scp

Copy file across systems

20.1.6 mv

Move file

20.1.7 rm

Remove file

20.1.8 tar

Extract and zip files

20.1.9 wget

Download a file via the internet

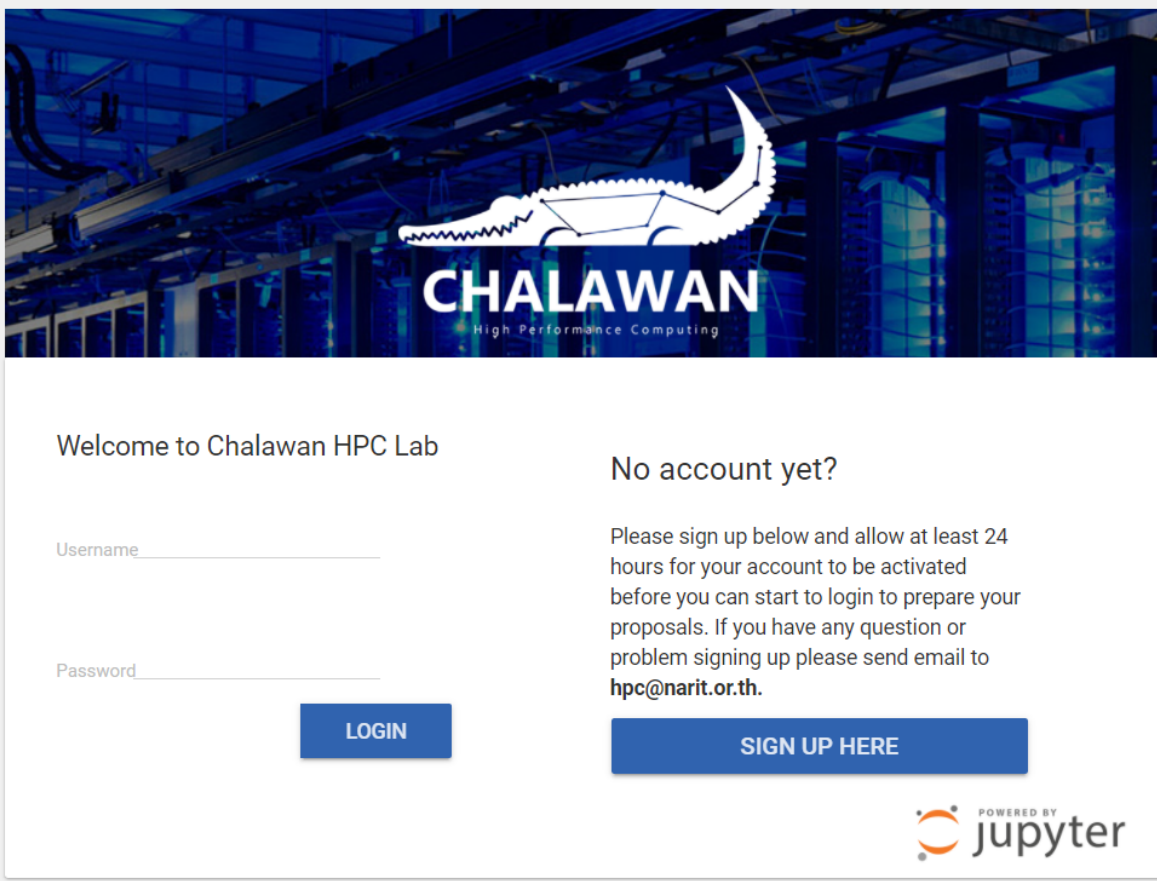
USING JUPYTERLAB

21.1 Getting Started with Chalawan HPC Lab

Chalawan HPC Lab is a next-generation web-based user interface powered by Jupyterhub . It enables you to work with documents and activities such as terminals, Jupyter notebooks, text editors and custom components in a flexible, integrated, and extensible manner. This allow you to access the Pollux from the internet outside NARIT so you don't have to access through stargate or VPN anymore.

21.1.1 Login to Chalawan HPC Lab

- 1.1 Go to the Chalawan HPC Lab website <https://lab.narit.or.th>
- 1.2 Enter your username and password (these are the same as the one you normally use to login to the Chalawan headnode).
- 1.3 Click login.




The image shows a web interface for the Chalawan HPC Lab. At the top, there is a banner with a blue background showing server racks and a white silhouette of a crocodile with the text "CHALAWAN High Performance Computing". Below the banner, the page is divided into two main sections. The left section is titled "Welcome to Chalawan HPC Lab" and contains a login form with fields for "Username" and "Password", and a blue "LOGIN" button. The right section is titled "No account yet?" and contains a sign-up instruction: "Please sign up below and allow at least 24 hours for your account to be activated before you can start to login to prepare your proposals. If you have any question or problem signing up please send email to hpc@narit.or.th." Below this text is a blue "SIGN UP HERE" button. At the bottom right of the page, there is a logo for "POWERED BY jupyter" with the Jupyter logo icon.

21.1.2 Start Jupyter server

Once logged in, if you do not have any currently running Jupyter server, you will be asked to spawn a new server. Spawners will control how Jupyterhub starts the individual server for each user. In order to work with the Jupyterlab, you should specify the resources for your job; CPU core(s), RAM, Time limit and etc. Please note that the Jupyterlab uses the time credit from your HPC account, thus, please stop the server when you finished your jobs. If you have a running Jupyter server, this step will be skipped and you will be brought to the Jupyterlab interface.

21.1.3 Running on Headnode (pollux)

For developing, testing, submitting Slurm jobs or just to access your files, you may spawn the Jupyter server on the Chalawan headnode. But please bear in mind that any heavy usage or long running jobs on the headnode are not allowed and you must either submit the jobs using Slurm "sbatch" (using the "Terminal" from the "Launcher") for a batch job. For interactive jobs that required heavy usage, you must spawn a Jupyter server on compute node(s).


[Home](#)
[Token](#)
anirut
[Logout](#)

Home

Named Servers

In addition to your default server, you may have additional 5 server(s) with names. This allows you to have more than one server running at the same time.


Server name	URL	Last activity	Actions
<input type="text" value="Name your server"/>	Add New Server		

Default Server

STOP SERVER
MY SERVER

21.1.4 Running on Compute node(s)

For heavy usage, you must select other spawn options than the “Chalawan headnode”. A few template profiles are provided. These options will allocated computing resources as listed in the options via Slurm job scheduler. For more flexibility, please use the “Advance Slurm job config” option. If the requested resources are available, your Jupyter server and Jupyterlab (see section 3) will be launched. If the remaining resources on the cluster does not meet your request, the spawner will wait in the queue for 1 minute before it aborts and give you a fail message.


[Home](#)
[Token](#)
[Admin](#)
utane
[Logout](#)

Home / Spawn

Spawner Options

Select job profile



- Chalawan headnode
- Chalawan_CPU (castor**) - 1 cores,32 GB, 8 hours
- Chalawan_GPU (pollux**) - 1 GPU, 1 core, 64 GB, 4 hours
- ✓ Chalawan_GPU (pollux**) - 4 GPU, 4 CPU core, 64GB, 4 hours
- Advanced Slurm Job config

Slurm Account

Slurm Reservation

START

21.1.5 “Advance Slurm Job config”

 **CHALAWAN** [Home](#) [Token](#) [Admin](#)  [utane](#) [Logout](#)

[Home](#) / [Spawn](#)

Spawner Options

Select a job profile:

Advanced Slurm Job config

Choose Node type	CPU node (castor**)	Parallel job type (for a serial job use "Shared memory")	Shared memory (single node)
Number of CPU cores (tasks)	1	Required Memory per node (in GBytes; leave empty for the default 2GB/CPU core)	
Job duration (hh:mm:ss)	hh:mm:ss		
Other Slurm options	if any, leave empty otherwise.		
Slurm Account	leave empty for default.	Slurm Reservation	if any, leave empty otherwise

START

21.1.6 Managing your Jupyter servers

Each user is allowed to simultaneously run 5 extra servers apart from the main “Default Server”. The server management page can be accessed via the “Home” tab or “File > Hub Control Panel” menu. You may use this interface to start, stop or add a server.

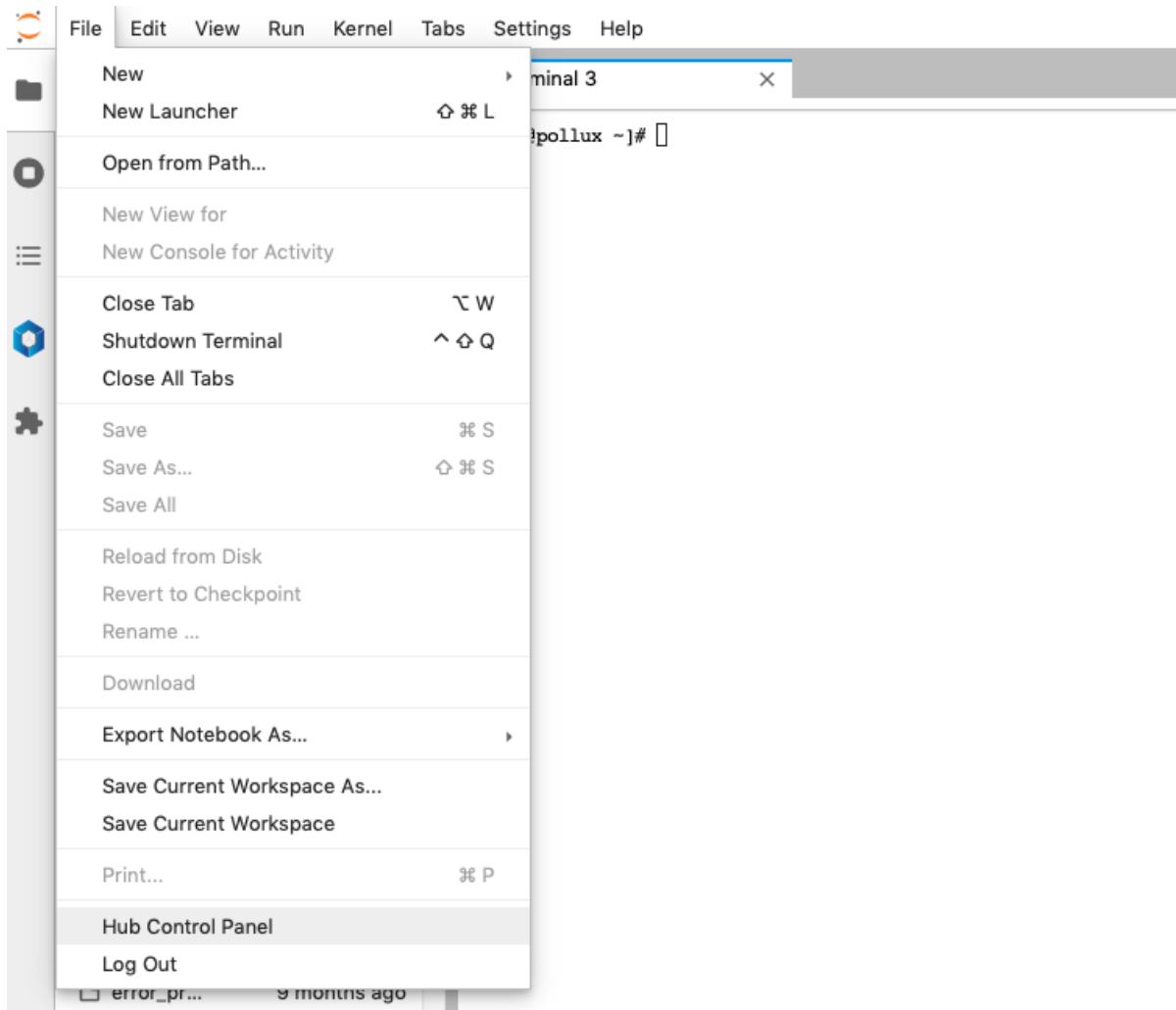
The screenshot shows the CHALAWAN web interface. The top navigation bar is blue with the CHALAWAN logo, links for Home, Token, and Admin, and a user profile for 'utane' with a Logout button. Below the navigation bar, a 'Home' breadcrumb is visible. The main section is titled 'Named Servers' and includes a descriptive paragraph: 'In addition to your default server, you may have additional 5 server(s) with names. This allows you to have more than one server running at the same time.' Below this is a table with columns: Server name, URL, Last activity, and Actions.

Server name	URL	Last activity	Actions
<input type="text" value="Name your server"/> Add New Server			
gpu1	/user/utane/gpu1	a day ago	<button>STOP</button>
test1	/user/utane/test1	2 minutes ago	<button>STOP</button>
test2		Never	<div> <div><button>START</button></div> <div><button>DELETE</button></div> </div>

Below the table, there is a 'Default Server' section with two buttons: STOP SERVER and MY SERVER.

21.1.7 Stop and manage your Jupyterlab server

Go to the HUB Homepage <https://lab.narit.or.th/hub/home> or you can click from the file menu.



This webpage you will see your default server and your additional server(s). You can add up to 5 additional servers by field a name of your server and click “Add New Server”.

To stop your server, click “Stop Server”.

CONTACT US

E-mail: hpc@narit.or.th



: Join Slack Chalawan Workspace for support

FREQUENTLY ASKED QUESTIONS

23.1 normal

23.1.1 I can not log-in to Pollux system, what can I do ?

23.1.2 I can not submit jobs to slurm queue in the Pollux system, what can I do ?