PowerPortal User Guide

How to use the PowerPortal web interface to submit and monitor Slurm jobs, manage templates, view usage, and access GUI applications — no command line required

- Overview
  - Getting Started with PowerPortal
  - Logging in to PowerPortal
  - Navigating the Interface
- Submitting a Job
  - The Job Submission Form
  - Writing Your Job Script
  - Job Arrays
  - MPI Jobs
- My Jobs
  - Viewing and Filtering Jobs
  - Job States
  - Cancelling a Job
  - Viewing Job Output and Error Files
- Job Templates
  - Using Base Templates
  - Creating and Managing Your Templates
- Usage & Fair-Share

    - Understanding Your Fair-Share Score
- GUI Applications
    - JupyterHub, VS Code, and RStudio
- Account Settings
    - Account Settings and Password

Overview

what PowerPortal is, how to log in, interface tour (navbar, tabs)

Overview

Getting Started with PowerPortal

PowerPortal is a web interface for the TAU HPC cluster. It lets you submit batch jobs to Slurm, monitor running and completed jobs, browse output files, and launch GUI applications — all without using the command line.

What you can do

- **Submit jobs** — fill out a form to configure resources (partition, CPUs, memory, GPUs, time limit) and paste your script. No need to write an sbatch file manually.
- **Monitor jobs** — see live job state, allocated nodes, and exit codes. Poll job output directly in the browser.
- **Save templates** — reuse common job configurations without re-entering every field.
- **View fair-share** — check your group's current usage and priority.
- **Launch GUI apps** — open JupyterHub, VS Code, or RStudio in one click.

What PowerPortal does not do

- It does not replace SSH. For interactive sessions, `salloc`, or file transfers, use the cluster directly.
- It does not manage software modules. Load modules inside your job script with `module load`.

Access

PowerPortal is available at https://powerportal.tau.ac.il. Access requires an active TAU account and membership in the **power** user group. Contact HPC support to request access.

Logging in to PowerPortal

PowerPortal uses your TAU university credentials — the same username and password you use for TAU email and other university systems.

Steps

1. Go to https://powerportal.tau.ac.il.
2. Enter your TAU username (e.g. `jsmith`) and password.
3. Click **Sign in**.

Your session stays active for 4 hours. Activity resets the timer — if you close the browser and return within 4 hours, you will still be logged in.

Forgot your password?

Use the TAU self-service password reset portal: https://pass.tau.ac.il/sspr/public/forgottenpassword. There is also a link to this page on the login screen and in Account Settings.

Access denied?

If your credentials are correct but login fails, your account may not be in the **power** group. Contact HPC support at hpc@tauex.tau.ac.il to request access.

Too many failed attempts

After 10 consecutive failed login attempts from the same IP address, that address is locked out for 1 hour. If you are locked out, wait an hour or contact HPC support to reset the lockout early.

Navigating the Interface

After logging in you land on the **Jobs** page. The navigation bar at the top is present on every page.

Navigation bar

Item	What it does
Jobs	View your running and completed jobs. Staff can also see all jobs on the cluster.
Submit Job	Open the job submission form.
Templates	Manage saved job configurations.
Guide	Link to this documentation.
Usage	View your fair-share score and usage.
GUI Apps	Links to JupyterHub, VS Code, and RStudio.
Username (top right)	Dropdown with **Settings** and **Logout**.

Theme

PowerPortal supports light, dark, and system-default themes. Change it under **Settings** (username dropdown → Settings).

Jobs page layout

The Jobs page has three tabs:

- **My Active Jobs** — jobs submitted under your username, active and recent.
- **Cluster Queue** — all jobs currently on the cluster, visible to staff only.
- **Job Search** — search historical and active jobs by ID, name, user, state, or partition.

Click any job row to open the detail dialog, which shows full resource allocation, the submit host, and buttons to view output or cancel the job.

Submitting a Job

every field in the submit form explained (partition, account, QOS, resources, script, arrays, MPI)

The Job Submission Form

The submit form is the main way to run jobs on the cluster. Open it from **Submit Job** in the navbar. Every field maps directly to an sbatch option — PowerPortal builds the submission for you.

Required fields

Field	What it sets
Job name	A label for your job. Appears in the Jobs page and in Slurm output.
Partition	The queue to submit to. Determines available nodes, time limits, and which accounts and QOS you can use. Select this first — other fields update based on your choice.
CPUs per task	Number of CPU cores per task. For single-task jobs this is your total core count.
Memory (GB)	RAM per node in gigabytes.
Time limit	Maximum wall-clock time in `HH:MM:SS` or `D-HH:MM:SS` format. The job is killed when this limit is reached.
Job script	The shell commands to run. You do not need a shebang — PowerPortal adds `#!/bin/bash` and `source ~/.bashrc` automatically so `module load` works.

Optional fields

Field	What it sets
Account	Slurm account to charge. Defaults to your primary account if left blank.
QOS	Quality of service — controls priority and limits. Only QOS allowed for the selected partition are shown.
Nodes	Number of nodes to allocate.
Node list	Request specific nodes by name (comma-separated). Leave blank to let Slurm choose.

Field	What it sets
Exclude nodes	Nodes to avoid. Useful if a node is known to be problematic.
Constraints	Node feature requirements (e.g. a specific CPU generation). Multiple constraints are ANDed.
Tasks	Total number of MPI tasks. Leave blank for non-MPI jobs.
Tasks per node	MPI tasks per node. Leave blank for non-MPI jobs.
Array	Job array specification, e.g. `1-100` or `0-9%5`. See the Job Arrays page.
Output file	Path for stdout. Defaults to `slurm-%j.out` in your working directory.
Error file	Path for stderr. Defaults to `slurm-%j.err` in your working directory (a separate file from stdout, not the same file).
Working directory	Directory the job runs from. Defaults to your home directory. Use absolute paths in your script if your input files are elsewhere.
Dependency	Wait for other jobs before starting, e.g. `afterany:12345`. Separate multiple jobs with `:`, e.g. `afterany:101:102:103`.
Notify email	Email address to notify about job events. Defaults to none.
Notify on	Which events trigger a notification email — Begin, End, Fail, Requeue, Time limit / 90% / 80% / 50% reached, Stage out, Invalid dependency, Array tasks, or All. Requires Notify email to be set.
GPU fields	GPU count and type appear only when you select a GPU-enabled partition.

Submitting

Click **Submit Job**. On success, PowerPortal redirects you to the Jobs page after 2 seconds where your new job appears at the top of the list.

Saving as a template

Click **Save as template** before submitting to store the current configuration for reuse. See the Templates chapter for details.

Writing Your Job Script

The job script is the shell code your job runs on the compute node. Paste it into the **Script / Command** field on the submission form — commands only, nothing else.

What PowerPortal adds automatically

The submit form shows two read-only lines above the editor:

```
#!/bin/bash
source ~/.bashrc
```

These are prepended to every job before submission. You do not need to type them. PowerPortal also sources the module system on the backend, so `module load` works inside your job without any extra setup.

Do not use #SBATCH directives in the script

All resource settings (partition, CPUs, memory, time limit, etc.) are set via the form fields — not via `#SBATCH` lines in the script. If you paste a script that contains `#SBATCH` directives, PowerPortal will strip them automatically and show a warning. Use the form fields instead.

A minimal example

```
module load python/3.11
python my_analysis.py --input data.csv --output results/
```

A more complete example

```
module load openmpi/openmpi-5.0.7-rocky9-slurm
cd $HOME/myproject
python preprocess.py

mpirun -n 16 ./simulate --config config.yaml
```

Working directory

Slurm sets the working directory to the value you enter in the **Working Directory** field. Use absolute paths or `cd` explicitly if your input files are in a different location.

Output and error files

By default, stdout and stderr are written to `slurm-<jobid>.out` in the working directory. Set custom paths in the **Output file** and **Error file** fields. Use `%j` for job ID and `%a` for array task ID in file names.

Tips

- Add `set -e` at the top to stop the script immediately if any command fails.
- Print key variables at the start (`echo "Running on $(hostname)"`) to make debugging easier.
- Test your script interactively with `salloc` before submitting a long job.

Job Arrays

A job array runs the same script multiple times in parallel, each with a different index. Use it to process many inputs with identical resource requirements — for example, running the same analysis on 100 different samples.

Setting up an array

Enter the array specification in the **Array** field on the submission form. Examples:

Value	What it does
`1-100`	100 tasks, indexed 1 to 100.
`0-9`	10 tasks, indexed 0 to 9.
`1-100%10`	100 tasks, at most 10 running simultaneously.
`1,3,7,42`	4 tasks with specific indices.

Using the index in your script

Slurm sets the environment variable `$SLURM_ARRAY_TASK_ID` to the current task's index. Use it to select the right input for each task:

```
module load python/3.11

SAMPLES=(sample_01 sample_02 sample_03 ...)
SAMPLE=${SAMPLES[$SLURM_ARRAY_TASK_ID]}

python analyse.py --input data/${SAMPLE}.csv --output results/${SAMPLE}/
```

Or if your inputs are numbered files:

```
python analyse.py --input data/sample_${SLURM_ARRAY_TASK_ID}.csv
```

Output files

Use `%a` in the output file path to get a separate file per task. For example, setting the output field to `logs/job_%j_%a.out` produces one log file per task.

Monitoring array jobs

Array jobs appear in the Jobs page as a single entry. Each individual task is listed with its own task ID and state. A parent job shows as running until all tasks complete.

Limiting concurrency

Use the `%N` suffix (e.g. `1-500%20`) to cap how many tasks run at once. This prevents a large array from monopolising the partition and avoids hitting per-user job limits.

MPI Jobs

MPI jobs run a single program across multiple processes, potentially spanning several nodes. Use the **Tasks** and **Tasks per node** fields to configure the MPI layout.

Fields to set

Field	sbatch equivalent	When to use
Nodes	`--nodes`	Number of nodes to allocate.
Tasks	`--ntasks`	Total MPI process count across all nodes.
Tasks per node	`--ntasks-per-node`	MPI processes per node. Use instead of Tasks when you want even distribution.
CPUs per task	`--cpus-per-task`	Threads per MPI process. Set to 1 for pure MPI; higher for hybrid MPI+OpenMP.

Example — pure MPI, 4 nodes × 32 tasks

```
module load openmpi/openmpi-5.0.7-rocky9-slurm
cd $HOME/myproject


mpirun ./simulate --config config.yaml
```

Set **Nodes** to `4`, **Tasks per node** to `32`, **CPUs per task** to `1`. Do not pass `-n` to `mpirun` — Slurm passes the process count automatically via PMI.

Example — hybrid MPI + OpenMP

```
module load openmpi/openmpi-5.0.7-rocky9-slurm
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
cd $HOME/myproject

mpirun ./simulate --config config.yaml
```

Set **Tasks per node** to the number of MPI ranks per node, and **CPUs per task** to the number of OpenMP threads. The product of the two should not exceed the node's core count.

Choosing a partition

MPI jobs that span nodes require a partition with inter-node connectivity. Check with HPC support if you are unsure which partition to use for multi-node jobs.

My Jobs

the jobs table, job states, cancelling, output files, auto-refresh

Viewing and Filtering Jobs

The Jobs page shows all jobs associated with your account. Open it by clicking **Jobs** in the navbar.

Tabs

- **My Active Jobs** — jobs submitted under your username, both active and recently completed.
- **Cluster Queue** — every job currently on the cluster, visible to staff only.
- **Job Search** — search historical and active jobs by Job ID, Name, User, State, or Partition. Enter at least one filter and click Search.

Job list columns

Column	What it shows
Job ID	Slurm job ID. Array jobs show as `jobid_taskid`.
Name	The job name you entered on the submission form.
Partition	The queue the job was submitted to.
State	Current job state. See the Job States page for the full list.
Nodes	Number of nodes allocated.
CPUs	Total CPU cores allocated.
Time	Elapsed wall-clock time vs. the time limit.
Submit time	When the job was submitted.

Filtering jobs

On the My Active Jobs and Cluster Queue tabs, use the search bar above the job list to filter by job name, job ID, partition, or state — the filter applies instantly as you type. For deeper historical searches with dedicated fields per criterion, use the Job Search tab instead.

Job detail panel

Click any row to open the detail panel on the right. It shows the full resource allocation, the node the job was submitted from, exit code, and action buttons (output, cancel). The panel polls for state updates every 5 seconds while the job is running.

Job States

Every Slurm job moves through a series of states from submission to completion. The current state is shown in the **State** column on the Jobs page.

Common states

State	Meaning
PENDING	Waiting in the queue. The job has not started yet. See below for common reasons.
RUNNING	Executing on a compute node.
COMPLETING	The job has finished but Slurm is still cleaning up (collecting exit codes, releasing nodes). Transient — usually lasts a few seconds.
COMPLETED	Finished successfully (exit code 0).
FAILED	Finished with a non-zero exit code. Check the output and error files for details.
CANCELLED	Cancelled by the user or an administrator before or during execution.
TIMEOUT	Reached the time limit and was killed by Slurm.
OUT_OF_MEMORY	Exceeded the memory limit and was killed. Resubmit with a higher memory allocation.
NODE_FAIL	The allocated node failed during the job. Resubmit; consider using **Exclude nodes** to avoid the failed node.

Why is my job PENDING?

A pending job is waiting for one of the following reasons:

- **Resources** — not enough free CPUs, memory, or nodes in the partition right now. The job will start automatically when resources become available.
- **Priority** — other jobs have higher fair-share priority. Your priority increases the longer you wait.

- **QOS or account limits** — you have reached a per-user or per-account job or CPU limit set by the QOS. Wait for a running job to finish.
- **Time limit** — your requested time limit exceeds the partition maximum. Lower the time limit or switch partitions.

Terminal states

COMPLETED, FAILED, CANCELLED, TIMEOUT, and OUT_OF_MEMORY are terminal — the job will not change state again. The detail panel stops polling once a terminal state is detected.

Cancelling a Job

You can cancel any of your jobs that are in PENDING or RUNNING state. Cancelling a running job stops it immediately — any output written so far is kept, but the job will not complete.

How to cancel

1. Go to the **Jobs** page.
2. Click the job row to open the detail panel.
3. Click **Cancel Job**.
4. Confirm the action in the dialog.

The job state changes to CANCELLED within a few seconds. The detail panel updates automatically.

Cancelling from the command line

You can also cancel from a terminal on the cluster:

```
scancel
```

To cancel all your pending jobs at once:

```
scancel --user=$USER --state=PENDING
```

Array jobs

Cancelling an array job from the detail panel cancels the entire array — all pending and running tasks are stopped. To cancel a single task within an array, use `scancel` from the command line:

```
scancel _
```

Note on COMPLETING jobs

Jobs in COMPLETING state are already finished and cannot be cancelled. Wait a few seconds for the state to transition to COMPLETED or FAILED.

Viewing Job Output and Error Files

PowerPortal lets you browse your home directory and view job output files directly in the browser — no need to SSH into the cluster.

Opening the output viewer

1. Go to the **Jobs** page and click a job row to open the detail panel.
2. Click **View Output** or **View Error**.
3. The file contents are displayed in a scrollable panel, showing the last 500 lines.

The output and error buttons are only shown for your own jobs. If you open another user's job detail (staff view), the buttons are not available.

File browser

The **Working Directory**, **Output file**, and **Error file** fields on the Submit Job form each have a folder icon next to them. Click it to open a **Select Path** browser that lets you navigate your home directory (folders and files, with file sizes shown) and pick a path instead of typing one manually.

Default output file locations

If you did not set a custom output path in the submission form, Slurm writes output to:

```
slurm-<jobid>.out
```

in the directory where the job was submitted from — typically your home directory. For array jobs the default is:

```
slurm-<jobid>_<taskid>.out
```

File not found?

- The job may not have started yet (PENDING) — output files are created when the job begins running.
- You may have set a custom output path that does not exist. Check the path in the job detail panel.
- The file may be on a path not accessible from the web server. Contact HPC support if this is the case.

Job Templates

saving a job as a template, using base templates, editing/deleting

Using Base Templates

Templates save job configurations so you do not have to re-enter every field each time you submit. There are two kinds: **base templates** provided by HPC staff, and **personal templates** you create yourself.

Base templates

Base templates are created and maintained by HPC staff. They represent tested, recommended configurations for common workloads — for example, a standard GPU job or a multi-node MPI setup. You cannot edit or delete base templates, but you can clone them.

Cloning a base template

1. Go to **Templates** in the navbar.
2. Find the base template you want under **Base Templates**.
3. Load it and save it under a new name (button labels may vary — check the Templates page for the exact action).
4. The template is copied to your personal templates where you can edit it freely.

Cloning is the recommended starting point for new users. Pick the base template closest to your use case and adjust the resource values and script to suit your workload.

What templates store

A template saves all form fields, including the job name: partition, account, QOS, CPUs, memory, time limit, nodes, constraints, array specification, output paths, working directory, notify settings, and the job script. You can still change any field, including the job name, before submitting.

Creating and Managing Your Templates

Personal templates are private to your account. Create them from the submission form or directly from the Templates page.

Creating a template

From the submission form

1. Fill in the submission form as you normally would.
2. Click **Save as template** instead of (or before) submitting.
3. Enter a name for the template and confirm.

This is the quickest way to save a configuration you have just used successfully.

From the Templates page

1. Go to **Templates** in the navbar.
2. Click **New Template** under **My Templates**.
3. Fill in the fields inline and click **Save**.

Editing a template

1. Go to **Templates** and find the template under **My Templates**.
2. Click **Edit**.
3. Update any fields and click **Save**.

The same partition-aware behaviour applies as on the submission form — changing the partition refreshes the available accounts, QOS, constraints, and node list.

Submitting from a template

1. Go to **Templates** and find the template you want to use.
2. Click **Use** — the submission form opens pre-filled with the template's values.
3. Enter a job name, make any last-minute adjustments, and click **Submit Job**.

Deleting a template

Click **Delete** next to the template and confirm. Deletion is permanent.

Usage & Fair-Share

reading your fair-share score, what it means for scheduling

Understanding Your Fair-Share Score

Fair-share is Slurm's mechanism for balancing cluster access across all users and groups. Your fair-share score determines your job priority in the queue relative to other users.

How fair-share works

Every user and account is allocated a **target share** of the cluster — a fraction of the total resources they are expected to use over time. Slurm tracks actual usage and compares it to the target:

- If you have used **less** than your share, your priority is **higher** — your jobs start sooner.
- If you have used **more** than your share, your priority is **lower** — other users' jobs take precedence.

Usage decays over time, so a period of heavy use does not permanently lower your priority. The decay window on this cluster is set by HPC staff.

Viewing your fair-share

1. Click **Usage** in the navbar.
2. The page shows your current fair-share score, your account's usage, and how you compare to other users in your group.

Improving your priority

- **Wait** — priority increases naturally as past usage decays.
- **Use less** — avoid requesting more CPUs, memory, or time than your job actually needs. Over-requesting counts as usage even if the resources sit idle.

- **Request accurate time limits** — a job that requests 24 hours but finishes in 2 hours still accrues 2 hours of usage, but an accurate time limit helps the scheduler backfill other jobs around yours, which benefits everyone.

Checking from the command line

```
sshare -u $USER
```

The `FairShare` column shows your current score. A value close to 1.0 means you are under your target share (high priority); close to 0.0 means you are over it (low priority).

GUI Applications

JupyterHub, VS Code, RStudio via PowerIDE (links + brief description)

JupyterHub, VS Code, and RStudio

PowerPortal provides quick links to browser-based GUI applications running on the cluster. These applications run on **poweride.tau.ac.il** and are accessible without SSH or a VPN.

Available applications

Application	What it is
JupyterHub	Jupyter notebooks and JupyterLab. Supports Python, R, and other kernels available on the cluster.
VS Code	Visual Studio Code running in the browser, connected to the cluster filesystem. Full terminal access included.
RStudio	RStudio Server for interactive R development and data analysis.

Launching an application

1. Click **GUI Apps** in the navbar.
2. Select the application you want.
3. The application opens in a new browser tab.

You log in with the same TAU credentials you use for PowerPortal.

Important notes

- GUI applications run on a shared login node, not on a dedicated compute node. They are intended for interactive development and light work — not for heavy computation or long-running jobs. For intensive workloads, submit a batch job through PowerPortal instead.
- Sessions are not persistent across server restarts. Save your work frequently.

- Files are stored on the same NFS home directory as the rest of the cluster — anything you create or edit in a GUI app is immediately accessible from compute nodes and vice versa.

Further reading

For full documentation on poweride — including how to request a session, configure kernels, and use VS Code extensions — see the PowerIDE User Guide in the HPC documentation.

Account Settings

theme, your LDAP profile, changing your password

Account Settings and Password

The Settings page shows your account information and lets you customise your PowerPortal experience. Open it from the username dropdown in the top-right corner → **Settings**.

Account information

The following fields are read from the TAU LDAP directory and cannot be changed in PowerPortal:

Field	Source
Username	Your TAU login name.
Full name	First and last name from the TAU directory.
Email	Your TAU email address.
UID	Your POSIX user ID on the cluster filesystem.

To update your name or email, contact the TAU registrar or IT helpdesk.

Groups

The Settings page also lists the POSIX groups your account belongs to, which affect partition and account access. If the list is long, use the show more/less control to expand or collapse it.

Theme

Choose between **Light**, **Dark**, or **System** (follows your operating system or browser preference). The setting is saved to your account and persists across sessions and devices.

Changing your password

Passwords are managed centrally by TAU — they cannot be changed inside PowerPortal. Use the TAU self-service password reset portal:

https://pass.tau.ac.il/sspr/public/forgottenpassword

The same link is available on the login page if you are locked out.