Cluster quick start

This section is a quick start for someone who is somewhat familiar with Linux/Unix and is looking at how to start using the University of Tartu UTHPC clusters quickly.

If you're not familiar with UTHPC and Linux in general, many beginner tutorials are available online. UTHPC team is also actively working on providing a better guides.

Request an account

In order to open an account with UTHPC, please fill out the form here to ensure that UTHPC team receives all the necessary information to quickly create the account.

Alternatively you can email your request to support@hpc.ut.ee. If you've already got a UT account, please provide your username with the email. If you are a student, you must also CC your supervisor in the email.

Galaxy users

To request an access to Galaxy Tartu Ülikool, please fill out the form here to assure a quick response. Alternatively you can send an email to galaxy@hpc.ut.ee but please include also your UT account username.

> **Info**
>
> For more information about Galaxy Tartu Ülikool go to our galaxy.hpc.ut.ee docs

Cluster login

Shell access (SSH)

You can access UTHPC cluster from anywhere, but for security reasons please use either of the following:

1. Be physically in a university building.
2. Connect from a remote location utilizing UT VPN.

To connect from a Unix-like system like Linux, macOS, WSL, use a Secure Shell Protocol called SSH to log in to `rocket.hpc.ut.ee` with your UT credentials:

```
ssh <username>@rocket.hpc.ut.ee
```

To connect from a Windows system, please follow guide for PuTTY or use Windows Subsystem for Linux (WSL). WSL is highly recommended.

For the ETAIS users, please follow this guideline to order a Rocket cluster account and to access it.

Open Ondemand

Open Ondemand is a portal that provides a single point of access to the UTHPC cluster. For full instructions, see the Open Ondemand guide.

Your home directory

The home directory, which makes all files and directories available on all cluster nodes, resides on a shared file system called `GPFS`.

Quotas manage the Disk space consumption. There are two types of quotas - directory size and file count. By default, a user has 2 TB of `$HOME` space and a maximum file count of 1 million files.

> **Tip**
>
> Please keep your home directory clean by regularly cleaning old data.

To see your quota, you can use the `myquota` command to see your maximum limits and current usage.

Copy data

There are multiple ways to transfer files between a local machine and the cluster, mainly depending on your local operating system. For a Unix-like OS, you can use `scp`, `rsync`, `sftp` commands on the command line. If you are on Windows or prefer a Graphical User Interface, FileZilla is one of a tools that you can use.

> **Info**
>
> More comprehensive guides are available here: File Transfer to/out of the cluster

To copy data to the cluster from your local machine, use the secure copy command `scp`:

```
scp /path/to/file <username>@rocket.hpc.ut.ee:/path/to/target_dir/
```

To retrieve data from the cluster to your local machine:

```
scp <username>@rocket.hpc.ut.ee:/path/to/file /path/to/target_dir/
```

Using software

You can make use of already pre-installed software, or you can compile and install software on your own. UTHPC uses an environment module system to make software and specific versions available to users:

For example on searching and loading '**python**' software.

1. Check the available '**python**' versions on cluster:

   ```
   module av python
   ```

2. Load the desired version of '**python**':

   ```
   module load python/3.8.6
   ```

3. List loaded modules:

   ```
   module list
   ```

Usually you would have a good idea about what software you need beforehand. For this guide we will use a module called py-openvino-diffusion. This is a CPU based AI image generator which we will use to generate some images. We will load it with the following:

```
[user@login2 ~]$ module av py-openvino-diffusion

------------------------ /gpfs/space/software/cluster_software/modules/spack/linux-
centos7-x86_64/Core ------------------------
   py-openvino-diffusion/master

[user@login2 ~]$ module load py-openvino-diffusion/master
```

```
[user@login2 ~]$ module list
Currently Loaded Modules:
  1) py-pip/21.3.1   2) python/3.9.12   3) py-openvino-diffusion/master
```

You now have the module loaded. Modules make multiple changes to your current environment, most notably the $PATH variable. In this case the `diffuse` command is now available:

```
[user@login2 ~]$ diffuse --help
usage: diffuse [-h] [--model MODEL] [--device DEVICE] [--seed SEED] [--beta-start
BETA_START] [--beta-end BETA_END] [--beta-schedule BETA_SCHEDULE] [--num-inference-
steps NUM_INFERENCE_STEPS] [--guidance-scale GUIDANCE_SCALE] [--eta ETA]
               [--tokenizer TOKENIZER] [--prompt PROMPT] [--params-from PARAMS_FROM]
[--init-image INIT_IMAGE] [--strength STRENGTH] [--mask MASK] [--output OUTPUT]

optional arguments:
  -h, --help            show this help message and exit
  --model MODEL         model name
  --device DEVICE       inference device [CPU]
  --seed SEED           random seed for generating consistent images per prompt
  --beta-start BETA_START
                        LMSDiscreteScheduler::beta_start
...<output omited>...
```

Warning

Loaded software is only for operating in the current terminal session. If you open a new session, it's a blank slate. Therefore, it's advisable to specify and load the needed modules in your job script.

Info

For a more thorough guide on modules, please go to Modules guide

Testing your job before submission

If you want to check whether your software starts executing before committing time and resources to it, you have an option to interactively test it.

You can open an interactive shell that is running inside a Slurm job. To achieve this, run the following command:

```
srun -A <accountname> --partition=testing -t 10:00 --cpus-per-task=4 --mem=16G --pty
bash
```

The command does the following:

- starts a Slurm process
- with the account (this is required for ETAIS users)
- in the partition `testing`
- with a Timelimit of 10 minutes
- with 4 cpus
- with 16G of memory
- the command will be executed in a `pseudoterminal`
- the command to execute will be `bash`

This command allocates resources on a compute node and once done, you will have an interactive terminal session where you can run the commands you need - like the `module load`, and run your software. When you continue with the guide to the `First job` section, everything you do in the terminal can be applied to the `#your code goes here` section in the job script.

For more information please check out the interactive jobs guide

First job

The cluster utilizes a scheduler called **Slurm** to control job execution and distribute running jobs across available physical resources like memory and CPU cores.

The following is an example of how to run your first job. A job script (sbatch file) consists of two main parts - instructions for the scheduler and the actual commands to run for the job, which operate your choice of software. Start with the scheduler instructions:

UT account

```
#!/bin/bash
#SBATCH -J hello_world
#SBATCH --partition=testing
#SBATCH -t 1:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=16GB

# your code goes below
```

ETAIS account

```
#!/bin/bash
#SBATCH -J hello_world
#SBATCH --partition=testing
#SBATCH --account="ealloc_905b0_something"
#SBATCH -t 1:00:00
```

```
#SBATCH --cpus-per-task=4
#SBATCH --mem=16GB

# your code goes below
```

Note

ETAIS users can only submit jobs when they use the proper allocation account. The allocation specifies which ETAIS organization and project is billed for the job.

You can get the information about which allocation name to use from the https://minu.etais.ee website, by going to the appropriate organization's UTHPC resource, where's written how to submit a job.

Then add the part for loading software and running a command:

```
module load py-openvino-diffusion

diffuse --prompt "An HPC user submitting their first job script"
```

The finalized job script looks like this and you should save it into a file, for example `generate_image.sh`:

```
#!/bin/bash
#SBATCH -J hello_world
#SBATCH --partition=testing
#SBATCH -t 1:00:00
#SBATCH --cpus-per-task=4
#SBATCH --mem=16G

# your code goes below
module load py-openvino-diffusion

diffuse --prompt "An HPC user submitting their first job script"
```

This job will run in the same directory that the job script is in and will generate the image in `output.png`. To download and view the image, refer to the data uploading section above.

As you can see, the script is basically a bash script which means that you can do a lot of tricks in it.

You can play around with the `--cpus-per-task` flag to see how it affects your job run time later. Please be aware that it has a direct effect on your accounting and comes with a bigger price.

Submit your job

Once you have a job definition script, you can submit your job script to the scheduler. Scheduler allocates the requested resources for your job and give you a job id. If the requested resources are available, your job start immediately. Otherwise, the job stays in queue until sufficient resources are available. To submit your job to **Slurm**, use the `sbatch` command:

```
sbatch generate_image.sh
```

```
Submitted batch job 15304092
```

If the job submission was successful, the command provides you with a job-ID, which you can refer to later.

The job will run for a couple of minutes. During the run time you can continue with the guide to see how to monitor it.

> **Warning**
>
> Running jobs directly on the cluster, without the queue system, is strictly **forbidden** and the jobs are **killed!**

There are various options for different kinds of jobs in cluster. Please review the following sections for more information Submitting Jobs, GPU Computing, Interactive Jobs for more information.

Monitor your job

You can inspect the status of your running jobs with the `squeue` command:

```
squeue -j 15304092
```

```
JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
15304092   testing generat    test_user  R       0:10      1 stage43
```

Here you can see the job '**hello_world**' with job-ID '**15304092**' is in state '**RUNNING**' (`R`). The job runs on the '**testing**' partition on the node '**stage43**' for 10 seconds.

Be aware, that if the requested resources aren't available, the job status is '**PENDING**' (`PD`). The job is in the queue, and starts as soon as the requested resources are available.

You can also see all active submitted jobs with `squeue` :

```
squeue -u <test_user>
```

Cancel your job

You can cancel your job via the `scancel` command by passing the job ID as an argument.

```
scancel 15304092
```

2025-09-05 2021-02-22