**Tutorial: Server Batch Processing**

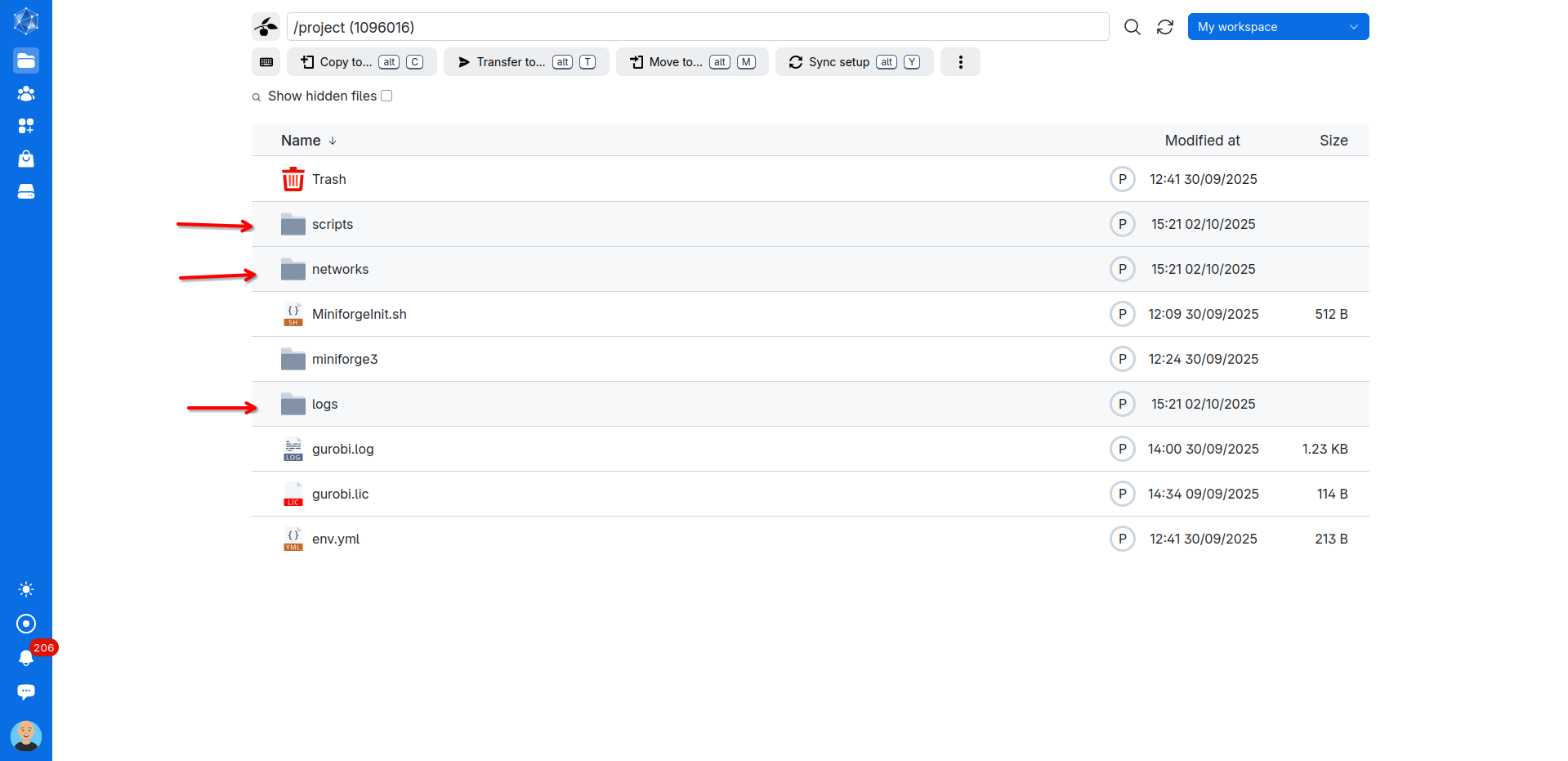
**1. Introduction**

In this tutorial we will run a python script using the batch processing functionality of the server cluster. This allows for the automatic running of multiple scripts, error handling and process termination. Enabling a much smoother and less manual workflow on the cluster.

First we need a script we want to run. For the sake of this tutorial we will use a script that solves a simple three node network, which can be found in the Github repository. Using this script we will also go through simple file structure on the server and output handling like log files. But this process can be of course used for any script that you want to run on the server. You only need to consider a few things when writing the script:

1. File paths in the script match the ones on the server cluster
2. Environments and packages are installed on the server
3. Files called upon by the script are uploaded to the server

To follow the tutorial create a folder within your working project directory called ***scripts*** and upload the ***test.py*** Python script and the ***sbatch.sh*** bash script to it, like shown below.



In your working project directory also create a ***logs*** folder to store the output log file and a ***networks*** folder within which to upload the ***test.nc*** network file that we will solve using our script.

**2. Setup of the python script**

The ***test.py*** script is set up in a way to allow it to run on the cluster environment. It does three main things:

1. Sets up logging so we can monitor progress and errors.
2. Loads and solves a network file using PyPSA and Gurobi.
3. Exports the solved network to a new file.

import pypsa

import sys

import os

import logging

import gurobi

*# ---------------------------*

*# File paths (relative to project folder)*

*# ---------------------------*

log\_path = f"project/logs/"

network\_path = f"project/networks/test.nc"

output\_path = f"project/networks/test\_solved.nc"

*# Ensure existence of folders*

os.makedirs(log\_path, exist\_ok=True)

*# ---------------------------*

*# Logging setup*

*# ---------------------------*

log\_file\_path = os.path.join(log\_path, "test.log")

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(message)s", handlers=[

logging.FileHandler(log\_file\_path, mode="w"), *# 'w' to overwrite each run*

logging.StreamHandler(sys.stdout) *# Print to console as well*

])

logger = logging.getLogger()

*# ---------------------------*

*# License setup (just to be sure)*

*# ---------------------------*

os.environ["GRB\_LICENSE\_FILE"] = "/work/project/gurobi.lic"

*# ---------------------------*

*# Core workflow*

*# ---------------------------*

*# Loading network file*

logger.info("Loading network...")

n = pypsa.Network(network\_path)

*# solving network*

logger.info("Optimizing network with Gurobi...")

n.optimize(solver\_name='gurobi', keep\_files=False)

*# exporting solved network*

logger.info(f"Exporting solved network to {output\_path}")

n.export\_to\_netcdf(output\_path)

logger.info("Script completed successfully.")

The above shows the complete script annotated and broken down into parts. Key things to note are that you have to match the filepaths to your specific folder structure on the cluster and that the all the logging output is not only stores in the log file but also printed live to the console to make debugging easier.

**3. Setup of the bash script**

To run our script on the cluster, we don’t launch Python directly from the terminal. Instead, we submit it to the scheduler (e.g., SLURM) using a batch script. This script tells the cluster:

* Which environment to use (Python + Conda + packages).
* Where to find licenses or dependencies (like Gurobi).
* How to capture output from the job (logs).
* Which Python script(s) to execute (here we run the test.py).

Below is our batch script (sbatch.sh):

*#!/bin/bash*

*# ===============================*

*# Environment setup*

*# ===============================*

*# Load Conda into the shell session*

source /work/project/miniforge3/etc/profile.d/conda.sh

export PATH=/work/project/miniforge3/bin:$PATH

*# Activate the correct Conda environment*

conda activate test\_env

*# Set Gurobi license location (cluster-specific)*

export GRB\_LICENSE\_FILE=/work/project/gurobi.lic

*# ===============================*

*# Logging setup*

*# ===============================*

*# Define the log directory & log file (job-specific)*

LOG\_DIR="logs"

mkdir -p "$LOG\_DIR" *# Ensure log folder exists*

LOG\_FILE="$LOG\_DIR/full\_run\_%j.log" *# %j = SLURM job ID → unique log per job*

*# Start logging*

echo "===== Job started at $(date) =====" | tee -a "$LOG\_FILE"

echo "Using arguments: $SCENARIO $MOD" | tee -a "$LOG\_FILE"

*# ===============================*

*# Run the Python script*

*# ===============================*

*# Redirect all Python output & errors to the log file*

{

python /work/project/scripts/test.py

} | tee -a "$LOG\_FILE"

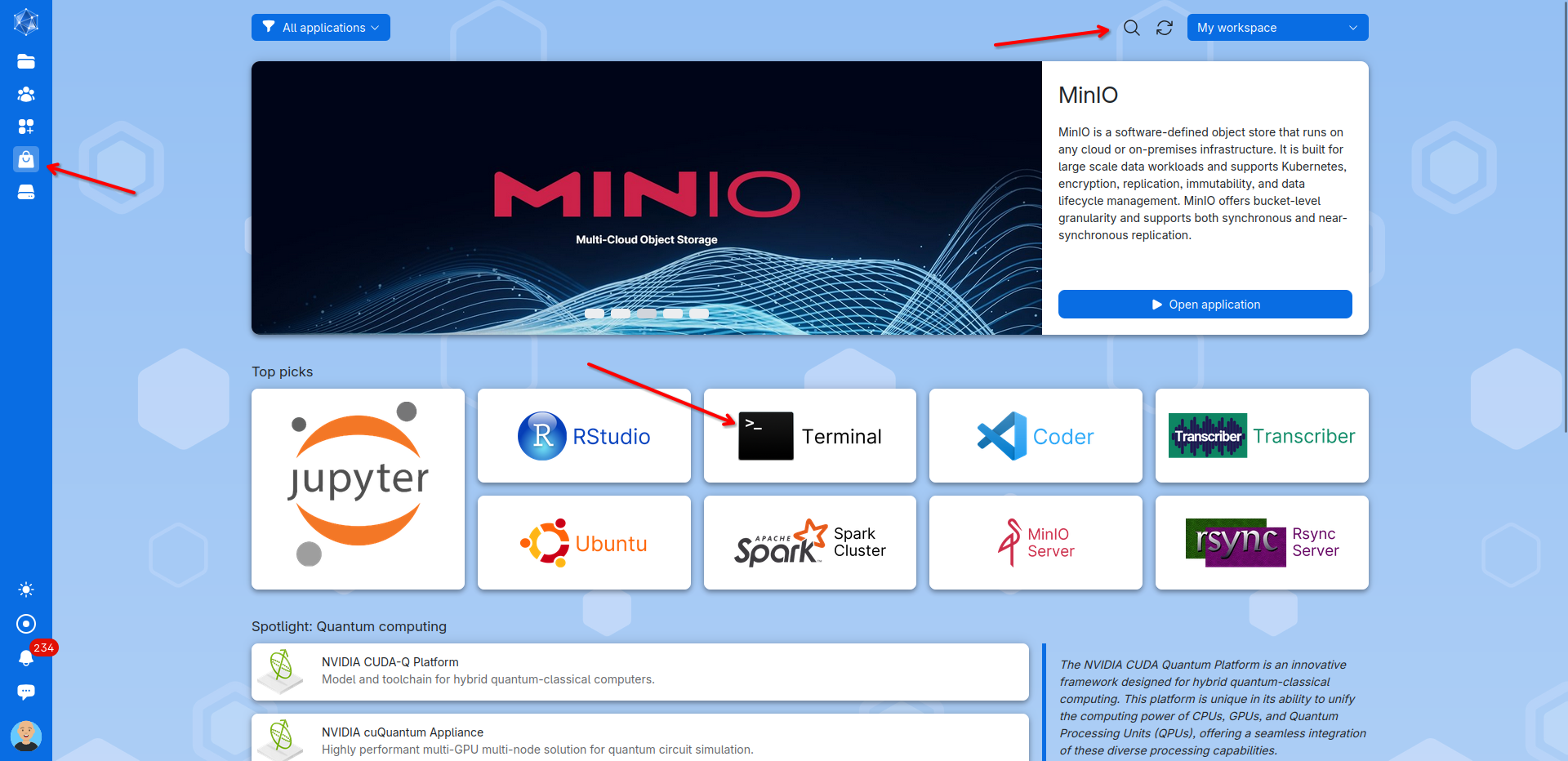
*# Log completion time*

echo "===== Job completed at $(date) =====" | tee -a "$LOG\_FILE"

Even though our Python script already logs internally to logs/test.log, logging in the batch script gives us extra benefits like cluster-level monitoring, capture of any shell-level error and full output history.

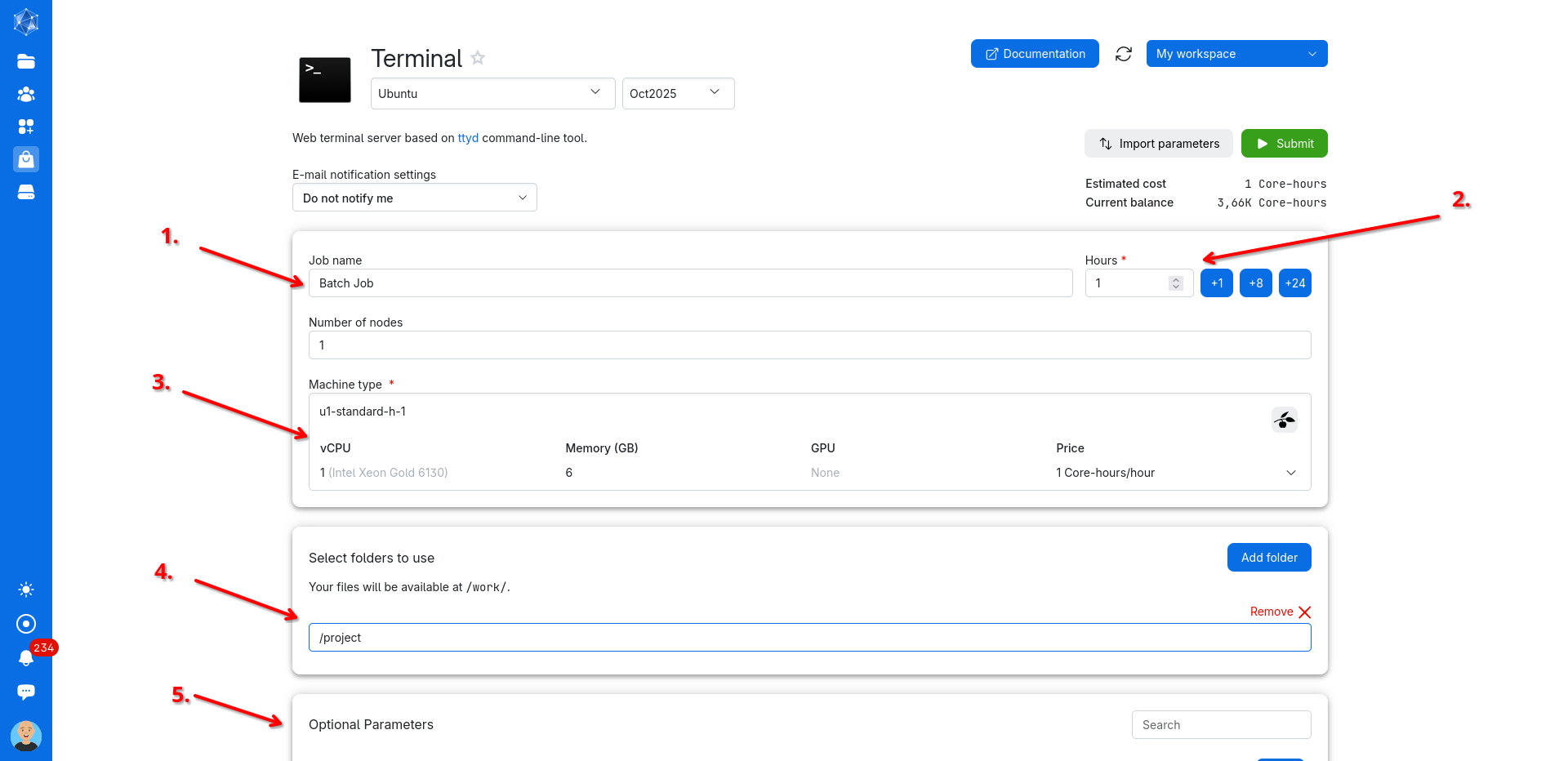
**4. Submitting the Batch Job**

To now run the ***test.py*** python script through batch processing using the ***sbatch.sh*** script we have to do the following. On the server website (<https://cloud.sdu.dk>) navigate to ***Applications*** and then open a ***Terminal*** process either by clicking on the terminal application or searching for it in the search bar.

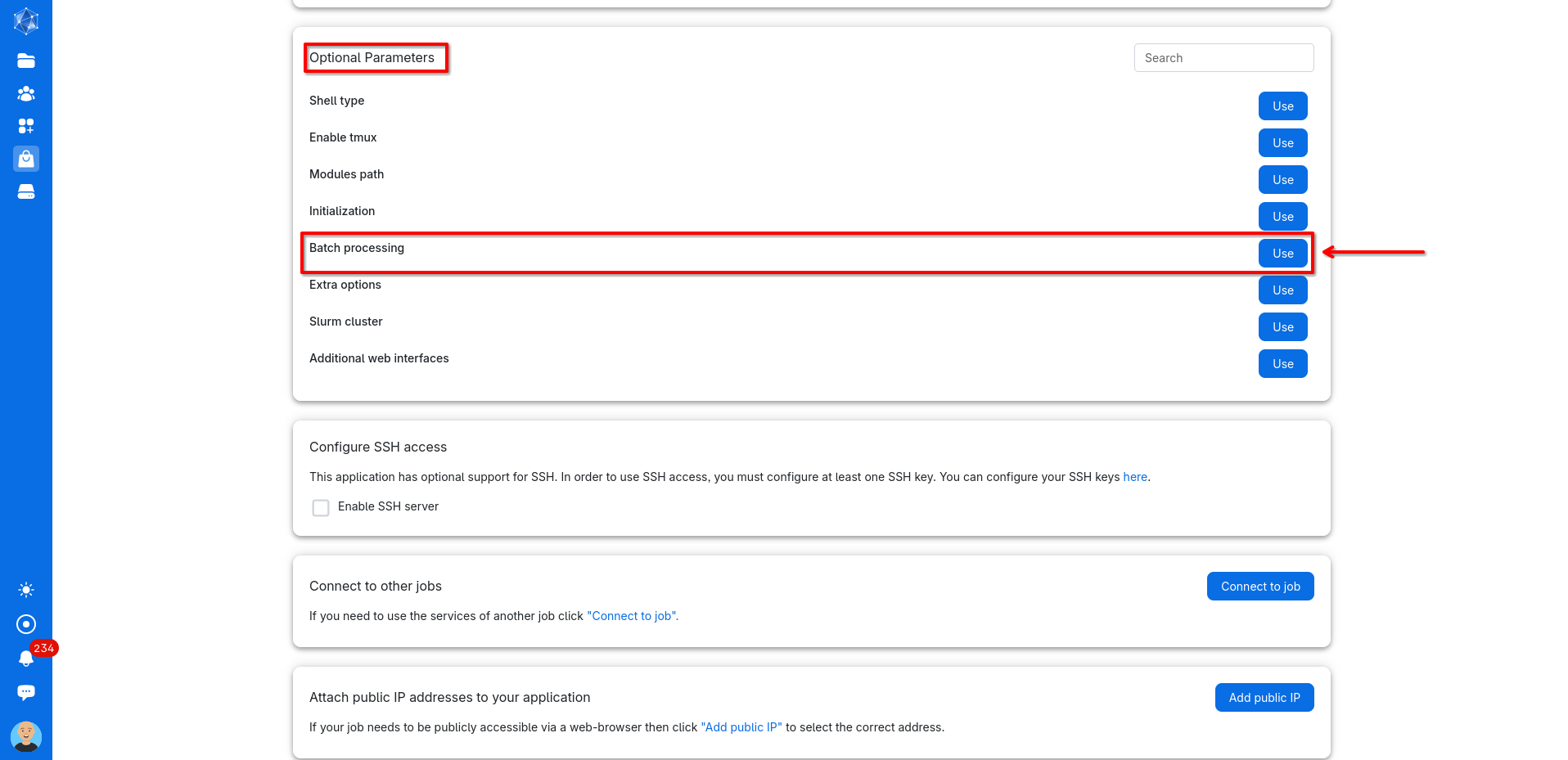


In the setup window for the terminal server process, you’ll need to configure the following options:

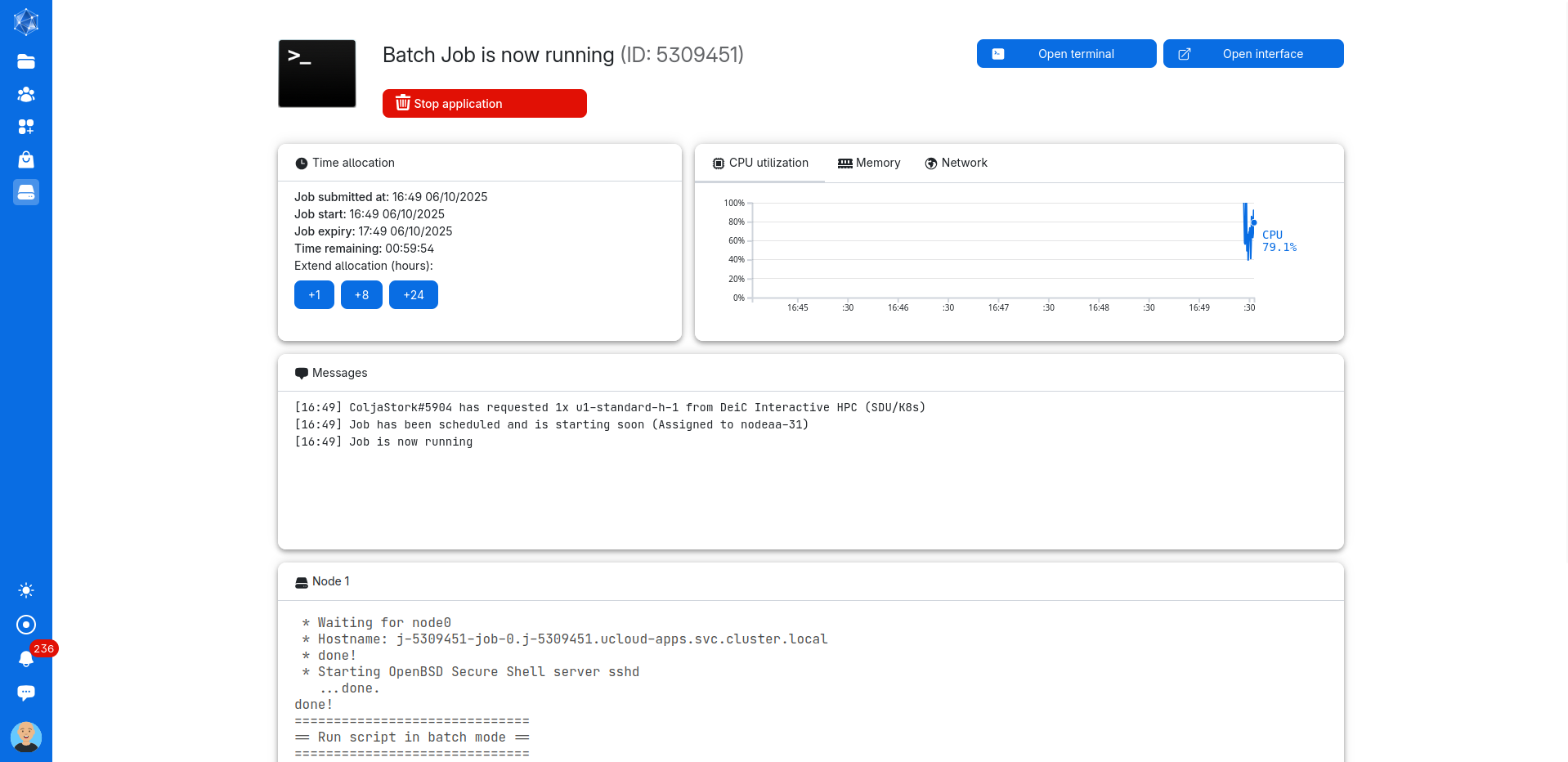
1. **Process name**: Choose a name for your process.
2. **Machine type:** Select a fitting machine, the smallest should be sufficient for this script.
3. **Runtime duration:** Set how long the process should run. One hour is more than enough. *(Note: Any unused time after the process ends will not be deducted from your balance.)*
4. **Working folder:** Select your project directory as the working folder.
5. **Optional parameters**: Scroll down to Optional Parameters.



Under Optional Parameters find the ***Batch Processing*** and click on ***Use***.



Navigate to and choose your batch processing file, which for this tutorial is placed in the scripts folder. Afterwards you can scroll back up in the setup window and press ***submit*** to run your batch job. The process window will start and the process will run the python script by itself and terminate the process when it is finished or fails. The newly opened process window should look something like this:



The outputted solved network file will be in the networks folder and the logs files of the process in the logs folder.