How to get LSV clusters running - Step by Step Guide

Prerequisites:

- OS: any Linux distribution, or alternatively WSL (Windows Subsystem for Linux) if you're currently on Windows (how to WSL: https://learn.microsoft.com/en-us/windows/wsl/install)

  (ssh should also be working out of the box on the Windows command line, but I don't know if there are any intricacies with that and if it will work just as good as on Linux)

- very basic knowledge of how to use the Linux command line (necessary to connect to (using ssh) and work on the cluster nodes, which run on Ubuntu)

  (I will walk you through the basics as good as I can)

- a copy of the project folder for the cluster that I put here (in this repository you are currently looking at):

  https://github.com/alexanderwehner/SoPro_Neural_Networks_25

- LSV account, working and password changed from master to your own password

- ssh installed (should come preinstalled on pretty much any Linux disto, but if not, install via the command 'apt install ssh' (e.g. for Ubuntu)

  or use any other package manager of your choice)

- note: wherever I have put the [yourUsername] placeholder, replace it with just your username, drop the square brackets (and don't put any whitespaces in or something like that)

Disclaimer:

I have explicitly written this guide assuming you don't know how to operate a Linux terminal, to make sure everyone understands what we are doing, even without having any Linux experience.
I tried to go into great detail on each step, explaining the commands you need and how to use them, even for the most basic commands.
If you already are using Linux and have some experience working with the Linux terminal, much of this guide will feel redundant and overly detailed.
I made this decision because I wanted to make sure that everyone can not just reproduce but also understand what they are doing, even if they have no Linux experience.
(And because I remember having asked about operating systems and being told that we have 2 or 3 persons on Windows in the group.)

This guide is the compiled documentation of my work setting up the project files for the cluster and HTCondor infrastructure, as straightforward as possible, as detailed as needed.
The LSV Wiki's guide on HTCondor and the README files in /sopro25 as well as /sopro25/submit-files, which were written by Marius Mosbach, provide some more insight in how to use HTCondor.
Make sure to check that out if you need more information.

In case you have any questions or need help setting up HTCondor yourself, do not hesitate to contact me, I will help you with anything you need as well as I can.

Some tips in advance:

- While working with the terminal / command line, make extensive use of the autocomplete function, by pressing Tab whenever possible (unsure if Windows terminal also has this).
  It will autocomplete fully or partially (if in doubt), when you start typing the name of a file or directory you want to interact with and then press Tab.
  It will make your life much easier.

- If at any point you need to edit a file once it is already on a cluster node, you can use the command 'nano [filename]' to edit it from the terminal (must be in the same directory).

Some essential Linux console commands you will need:

pwd - prints out the current working directory and path (where you are)

cd [directory] - go to the specified directory (you navigate, but forward only)

cd .. - navigate to the parent folder ("Go back 1 step". You can also go back more steps by using ' cd ../.. ' or ' cd ../../.. ' etc.)

cd ~ - navigate to your home directory (' /nethome/[yourUsername] '), from anywhere.

ls - list all contents (subdirectories and files) of the directory you are currently in

mkdir [X] - create a new directory X in the directory you are currently inside of. ("Make new directory [X] here".)

rm [X] - delete a file or directory. Add flag ' -r ' to also recursively delete the contents of a directory. ("Remove [X]".)

nano [file] - edit a file inside the terminal using the nano editor, when you need to edit files that are already on the remote node. (nano > vim)

ssh [destination] - connect to some remote node via SecureShell, the LSV cluster, for example: ' ssh [yourUsername]@contact.lsv.uni-saarland.de '

man [command] - displays a manual page for the specified command, where you can read more on what it does and how to use it

scp -r [sourcepath] [destination:path] - securely copy directories /

files via SecureShell (ssh).

A practical example of me copying a folder (directory) from my device to the LSV cluster node would look like this:

```
                scp -r ~/Desktop/sopro_cluster_setup/projects/sopro25 awehner@contact.lsv.uni-saarland.de:/nethome/awehner/projects

|                               |                                         |
syntax:     [command] [path on my machine to the file I want to copy] [my LSV user account, '@', hostname of the target, ':', destination path]
```

Outline of the file registry structure on the LSV cluster:

(Your account and data in the registries outlined below are available on every node. There are far more directories on each node, but only the following matter to us):

```
/
|________ /data
            |________ /users
                        |________ /[yourUsername]
                                          |________ /logs
                                                        |________ /sopro25

|________ /nethome
                |________ /[yourUsername]
                                |________ /projects
                                                |________ /sopro25 (we will copy this directory here ourselves, details below)
```

This is what the file registry structure should look like. In case something does not work right away, please make sure all paths outlined here exist (if not, please create the relevant missing registries, details on how to do that below).

Your home registry (' /~ ') will be /nethome/[yourUsername], so we will put all your project files in the /projects directory there (create the directory if it's not there, details below). In this guide, I will go into detail about how to get code running and submit jobs to the cluster using HTCondor (the system that manages, queues and executes jobs on the cluster).

1. Download the sopro25 project folder from here: LINK HERE

2. Open the following three files in an editor of your choice:

        - run.sh   (in folder 'scripts')

```
        - setup.sh      (in folder 'scripts')
        - run.sub       (in folder 'submit-files')
```

3. In all three files, you will need to edit any occurence of 'awehner' (my LSV username) with your own LSV username, to correctly set up the paths to work on your LSV account on the cluster.

here's where to edit in detail:

```
        - run.sh        line 4
        - setup.sh      lines 4,7,8
        - run.sub       lines 5,6,9,10,11
```

4. Connect via ssh to contact.lsv.uni-saarland.de, using the following command:

```
        ssh [yourUsername]@contact.lsv.uni-saarland.de
```

It will prompt you for your password upon connecting, just enter the password and press enter.

5. Check if the file registry on the remote node matches the outline in the graph, if not, create all missing directories and see that each path exists.

You will start out in /nethome/[yourUsername] , so to get to /data you have to navigate back to / first, using:

```
                cd ../.. (go back 2 directories)
```

Then use this to get to /data/users/[yourUsername]:

```
                cd data/users/[yourUsername]
```

To go back to /nethome/[yourUsername] , navigate to / using cd ../../.. (to go back 3 directories) again and then use:

```
                cd nethome/[yourUsername]
```

or you can use

```
                cd ~ (the "Go back home" - command)
```

from anywhere to get directly back to /nethome/[yourUsername] (your home directory).

First navigate to /nethome[yourUsername] and check if /projects exists with the ls command, if not, create it using the command:

```
mkdir projects
```

Then, use ls again to confirm you have created it there.

Next navigate to /data/users/[yourUsername] and check if /logs exists, if not, create /logs with:

```
mkdir logs
```

You can again check with the ls command if it's now there.

6. Copy over the data from your machine to the cluster node.

Open a fresh terminal (that's not logged in to the cluster node via ssh) and copy over the sopro25 project folder from your machine to /nethome/[yourUsername]/projects on the remote node, using the scp -r command:

For example, from my machine it would be this (first argument: source, second argument: destination, -r flag to recursively copy the folder's content):

```
scp -r ~/Desktop/sopro_cluster_setup/projects/sopro25 awehner@contact.lsv.uni-saarland.de:/nethome/awehner/projects
```

You will be prompted to enter your password again, so just enter it and confirm with Enter.

Once you have finished copying the files, switch back to the other terminal that's connected to the node and navigate to /nethome/[yourUsername]/projects and use the ls command to check if the directory is there.

7. Build and push the docker image.

Now you need to connect (from the contact.lsv.uni-saarland.de node) to workstation 71 of the LSV cluster, which we can use to build docker images and push them to the LSV docker registry.

Use the command:

```
ssh [yourUsername]@ws71lx.lsv.uni-saarland.de
```

to connect to the workstation, enter your password when prompted and confirm with enter.

Here, you can also access your data like you would on the contact node.

Navigate to /projects/sopro25/docker and build the docker image using the following command:

IMPORTANT NOTES:

- You have to edit the command to include your username towards the end where I put the [yourUsername] placeholder.
- When executing the command, include the whitespace and dot at the very end, it is part of the command.

```
docker build -f Dockerfile --build-arg USER_UID=$UID --build-arg USER_NAME=$(id -un) -t docker.lsv.uni-saarland.de/[yourUsername]/dockerimage_sopro25:v1 .
```

Lastly, after you finished building the image, push it to the LSV docker registry with this command (again, you will need to replace the placeholder with your username towards the end):

```
docker push docker.lsv.uni-saarland.de/[yourUsername]/dockerimage_sopro25:v1
```

After successfully pushing the image to the registry, log out of the workstation node by simply typing the command 'logout' into the terminal.

You should now be back to the contact node.

8. Submit a job

From the contact node, connect to the submit node by using the following command:

```
ssh [yourUsername]@submit.lsv.uni-saarland.de
```

Enter your password when prompted, confirm with Enter.

I have prepared a HTCondor submitfile (' run.sub ') that should work out of the box with the paths and struture we have created so far.

Submit the job by using the following command (remember to put in your username instead of the placeholder):

```
condor_submit /nethome/[yourUsername]/projects/sopro25/submit-files/run.sub
```

If it does not immediately throw you any errors, it will first tell you which cluster it has submitted your job to (a five digit number).
Remember this number, as it tells you which of the logfiles correspond to that exact job submission.

You can then monitor your job with the command:

```
watch condor_q [yourUsername]
```

There you can see the status of all your jobs and check if they are still running, on hold, rejected or finished.

Wait to see when the job finishes. It usually takes up to a minute to complete, depending on availability of the compute node we are running on.

Once the job finishes, exit this monitor by pressing CTRL + C once, to interrupt the monitor process.

9. Check if logfiles were produced

When the job is finished running, navigate back to / and from there to data/users/[yourUsername]/logs/sopro25/logfiles and check if there is any files in that directory.
If everything worked correctly, there should be three files (one .log file, one .out file and one .err file), each named something like this:

```
run.sh.(XXXXX).2025_month_day_somemorenumbers.log/.out/.err
```

The first number in parentheses (XXXXX) will be the cluster/job number from before, the rest will be today's date and then some more numbers.

You can check these out using the cat command, like this for example:

```
cat run.sh.39139.2025_07_28_1753713314.out
```

(This was one of my output files, 39139 is the job number, the rest is the date and probably timestamps.)

(Also, remember to make use of the autocomplete function of your terminal by pressing Tab a lot.)

If you start typing 'cat run.sh' and then press Tab, it will autocomplete using whats in the folder until a point at which it is unsure of what file exactly you mean.
The first time will be until the file extension, as you now have three files with the same name but different file extensions in your directory.
Press Tab and you just have to type the extension, to access either file you want to see.
If you do this again later, when you have some older and more recent logfiles in the directory, it will autocomplete up to some point in the job number.
You will then have to type in the job number to specify, and you can hit Tab again so that you don't have to type the whole thing out.

10. Read all three logfiles

If all paths are working correctly as intended, each job submission will put out three files:

- a log file, containing information about the job on the cluster
- an error file, which in case something goes wrong during the job will contain info about why it failed (file empty means the job finished as intended)
- an output file, which will contain the output of your actual project, the code you are trying to run

I have put a small example task in the project files (you can check out the code in /sopro25/src/mnist.py), to make the cluster actually run something.

In the output file, you should see some output at the very bottom (way below the long list of packages and CUDA info).
It should be some loss and accuracy metrics of two tiny neural networks I have trained and evaluated in the code.

If you can see that, the cluster setup works.

Troubleshooting in case of errors:

If something goes wrong at any point, it is most likely a path issue.

Make sure all relevant paths used in the files exist in your LSV account registry, or create them if necessary.

If you encounter any errors you don't know how to fix or need help in general, send me a message and I will do my best to help you get things running.

Whats next?

- Managing dependencies

Installing new dependencies requires building and pushing a new dockerimage.
In the Dockerfile, lines 32 and below contain the dependencies that are currently in the image we're using.
Just add whatever you need here in the same format as the ones already there, and docker will install and build them into the image.
In the case that you choose to use a new image name, the submission file 'run.sub' needs to be updated with the correct docker image name (line 5, parameter 'docker_image')

- Getting the actual project code into the cluster

Whatever code we want to run on the cluster needs to be put in the /sopro25/src directory .
Following that, the file 'run.sh' (in /sopro25/scripts) will need to be updated with the new path and file name of the code to specify what code to run.
(line 14, under #run code; currently this specifies my example task 'mnist.py')

- Adapting the submitfile to request code execution on similar hardware as the one outlined in the Averitec shared task documents

At the moment, the submitfile is set up in a way that requests a specific cluster compute node and some specific hardware specs.
We will need to change this to more closely match the hardware outlined in the

shared task (or something comparable).

Lastly, depending on your willingness to deal with this system or not, I can always take on the job of running code and managing dependencies for the project, as I have spent some time getting into it now and have set up everything on my machine and on my cluster account.

This guide should just serve as a springboard into the workflow of HTCondor and the LSV cluster for you, if you want to get into it yourself.

If you don't want the hassle, just tell me when to pull your code from the repo and run it on the cluster, if you want me to test it for you.

I have uploaded the files for this to a GitHub repository for now (link above), so that you can clone/pull or download it more easily to test it, but I will integrate it into the GitLab repo soon(ish).

Also, if you have the time and nerves to spare, you should check out the other README files in the sopro25 project folder. They come from the LSV HTCondor documentation, written by Marius Mosbach, which goes a bit more into detail regarding HTCondor itself, what commands to use and how to handle it in use.

If you find any errors in this document or something does not work as expected, please let me know as soon as possible.

Alex

PS: Also please do me the favour of keeping a separate list of all dependencies you use, so that I can keep the docker image up to date more easily. Thanks!