Connecting to the source

Anton Zhitomirsky

December 26, 2023

This documents the process to connect to the resources provided by the department in order to access files.

Basic SSH into the labmachine

Setting up the public key to not type in password

Copy the public key on your machine over to the authorized hosts on the other machine to avoid typing in your password every time you ssh. [source]

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub <usrname>@<host>
```

Setting up the tunnel from shell1 – another machine

Create an alias on your home machine for the shell you'll use [source1] [source2]

```
Match host shell*
    Hostname %h.doc.ic.ac.uk
    User az620
```

To begin tunnelling through with ssh [source] and find your favourite [workstation] and set up tunnel with [source3]

```
Match host texel*
    Hostname %h.doc.ic.ac.uk
    User az620
    ProxyJump shell1
```

Working with Slurm [reference]

The department has a pool of GPUs for deep learning tasks. To reduce complexity of scheduling, slurm is a system that schedules tasks into these resources.

SSH into the gpu cluster

In the `~/.ssh/config` file, add the line below and ssh into it to schedule jobs. This should only be used for scheduling jobs into the slurm scheduler.

```
# should be either 2 or 3
Match host gpucluster*
    Hostname %h.doc.ic.ac.uk
    User az620
    ProxyJump shell1
```

Then submit a pre-existing script using the sbatch command. The output will be stored, by default, in the root of your `~/` directory, with the filename `slurm20-{xyz}.out`.

Where to store data

Data is stored in the `/vol/bitbucket/${USER}`. It can be created with `mkdir -p ...` if it doesnt exist. You should create personal folders here. Otherwise, if you store data in the home directory you risk going over the quota limit [source][1]. `/vol/bitbucket` should only be used for temporary storage of material that cna be regenerated (downloads or compilations). To see where you are storing disk space use `/vol/linux/bin/usage` (disk space) and `/vol/linux/bin/nfiles` (number of files and directories) [source].

Why do you need a python virtual environment?

Python isn't good at dependency management. `pip` places all external packages into `site-packages/` in the base Python installation. By creating a virtual environment you avoid system pollution (since these packages mix with system-relevant packages causing unexpected side effects)

Creating a Python Virtual Environment in /vol/bitbucket

☞ Use a lab PC to prepare the Python environment; don't use the gpucluster machine for this. You may encounter 'out of space' errors.

Its advised that you create Python virtual environments on `/vol/bitbucket`. Steps taken from [source]

1. create `/vol/bitbucket/${USER}`

2. create virtual environment

```
export PENV=/vol/bitbucket/${USER}/myenv
python3 -m virtualenv $PENV
ls -al $PENV
```

3. activate your VE:

```
source $PENV/bin/activate
which pip
[should say: /vol/bitbucket/${USER}/myenv/bin/pip]
which python
[should say: /vol/bitbucket/${USER}/myenv/bin/python]
which python3
[should say: /vol/bitbucket/${USER}/myenv/bin/python3]
```

[1]you can check disk quota with `quota -Q`

4. install packages and verify with `which <module name>` that it is saved under `/vol/bitbucket/$USER/myenv/bin/`
5. Or prepare a package requirements file `requirements.txt` which speicifes a list of packages (optinally with specific version constriants) and install with `pip install -r requirements.txt`
6. once you're done working with the virtual environment you can deactivate it with `deactivate`
7. each time you login to a specific machine redo the activate stage with `source /vol/bitbucket/$USER/myenv/bin/activate` (this can be appended to the end of the `~./bash_profile` in home dir.)

Why activate and deactivate?

Activating it gives us a new path for the pyhton executable because, in an active environment, the $PATH environment variable is slightly modified.

Using CUDA

Many jobs will make use of the Nvidia CUDA toolkit, multiple versions of which are at `/vol/cuda`. If there ever appears a need to use this toolkit then in the submission bash script add:

```
. /vol/cuda/12.0.0/setup.sh # if you're using version 12.0.0
```

Care that you choose a version with which your tensorflow or pytorch are compatible with.

Template Example bash submission script

☞ This example assumes you have followed the previous steps and installed a python environment (using virtualenv, extra lines may be needed using minconda, check the example script further below) as directed. Please adjust paths if you have an existing python environment, or if you already load your environment in /.bashrc (note: sbatch does not load /.bashrc, source it as per example script) . Do not uncomment #SBATCH lines, keep them as below, make sure the #SBATCH directives are directly after `#!/bin/bash`

Remember to make the sumbission script[2]. executable with `chmod +x <script_name>.sh`

```
#!/bin/bash
#SBATCH --gres=gpu:1
#SBATCH --mail-type=ALL # required to send email notifcations
#SBATCH --mail-user=<your_username> # required to send email notifcations
export PATH=/vol/bitbucket/${USER}/myvenv/bin/:$PATH
# the above path could also point to a miniconda install
# if using miniconda, uncomment the below line
# source ~/.bashrc
source activate
source /vol/cuda/12.0.0/setup.sh
/usr/bin/nvidia-smi
uptime
```

[2]example can be found at `/vol/bitbucket/shared/slurmseg.sh`

Scheduling job

- `ssh gpucluster2`
- `cd /vol/bitbucket/$USER`
- `sbatch <path to executable>.sh`
- `less slurm-XYZ.out` saves the output.
- `squeue -l` for queue of running jobs
- `scancel <job ID>` to delete slurm job