Setup & usage of nam-shub-01 on private laptop

Åsmund Danielsen Kvitvang, Marius Aasan

Updated: December 3, 2020

1 Setup

1. Make sure you have a hidden directory called `.ssh` in your home folder by typing `ls -a`. If not, make one.

2. Next, we need to create private/public rsa key pair. This is done by typing the following (change *username* to your uio username).

   ```
   ssh-keygen -t rsa -b 4096 -C "[username]@uio.no"
   ```

   When terminal asks you

   ```
   Enter file in which to save the key(...):
   ```

   click [ENTER]. When terminal asks you

   ```
   Enter passphrase (empty for no passphrase):
   ```

   choose a password (minimum 5 characters). You also need to repeat the password. By typing `ls` in your terminal you will at this point observe that you have two new items in your directory, namely `id_rsa` and `id_rsa.pub`, which is your private and public key, respectively. By typing `cat id_rsa[.pub]` you may inspect these items.

3. Next, you may create a *config* file for your own convenience. Write `[EDITOR NAME] config` to open such a file (within `.ssh` directory). Write the following into the *config* file and save (change *username* to your uio username).

   ```
   Host *
   ServerAliveInterval 60
   ServerAliveCountMax 1440
   ```

```
Host uio
Hostname login.math.uio.no
user [username]

Host nam-shub-01
Hostname nam-shub-01
ProxyJump [username]@login.math.uio.no
user [username]
```

The first three lines is to make sure that a session don't terminate if user is not active. It tells the host ping a small package every 60 second. This is done a total of 1440 times. Thus, we are made sure that our program runs for at least 24 hours. The next three lines makes our life easier when we log in to the host called `login.math.uio.no`. Instead of typing

```
ssh [username]@login.math.uio.no
```

to log in we now only need to type `ssh uio`. **NB**: Do not run complex computations on this computer, as it only serves as a passage for logging into `nam-shub-01`. The last four lines is for easier login to `nam-shub-01`. The word `ProxyJump` simply means *use this computer as passage to the* `nam-shub-01` *computer*.

4. Lastly, we need to copy the **public** rsa key from our private laptop to the uio computer. Navigate to `.ssh` folder and type the following.

```
ssh-copy-id -i id_rsa.pub uio
```

Note that we now use the alias of the machine, created in the *config* file.

5. Now we are ready for usage of these computers. Simply type

```
ssh nam-shub-01
```

to log into the computer. Enter password if credentials are needed. To exit the computer you may type `CTRL+d`.

2 Useful stuff

2.1 Moving files

You may want to move (relatively small) files back and forth between private laptop and uio computer. To move a file from private laptop to uio computer, type the following.

```
scp [private-location/filename] uio:[uio-location]
```

To move a file from uio computer to private laptop, type the following.

```
scp uio:[uio-location/filename] [private-location]
```

2.2 Module handling

When you are logged into one of UiOs many computers you need to load the modules you are going to use when developing. To see complete list of available modules and their versions, type the following.

```
module avail
```

To load one of the available modules, type the following.

```
module load [module name]
```

Example of *module name* could be `PyTorch`. Then it would load the default version of PyTorch. If specific version for your code is required, this needs to be specified. To list the loaded modules, type the following.

```
module list
```

To remove a loaded module, type the following.

```
module rm [module name]
```

If your are in need of a module which is not listed among the available modules, install it by typing the following.

```
pip3 install --user [module name]
```

The command line argument `--user` is to avoid admin login for installation. It is in this case a module for your use only and will not be available for the outside world.

2.3 Handling GPUs using Python

A neat way of only using e.g. GPU 1 and 2, if one does not want (or need) to use all four at the same time, is by adding the following three lines of code at the beginning of your Python script, i.e. at your programs entry point (BEFORE all other imports are executed).

```
import os
os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="1,2"
```

GPUs 1 and 2 will now be recognized by indices 0 and 1 for the rest of the session, while GPUs 0 and 3 'does not exist'. The following code illustrates this (using both Tensorflow and PyTorch).

```
1 import os
2 os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
3 os.environ["CUDA_VISIBLE_DEVICES"]="1,2"
4
5 import torch
6 import tensorflow as tf
7
8 avail = torch.cuda.is_available()
9 count = torch.cuda.device_count()
10 print("Available:",avail,", Count:",count)
11
12 gpus = tf.config.list_physical_devices('GPU')
13 for gpu in gpus:
14     print("Name:",gpu.name,", Type:",gpu.device_type)
```

Running the code WITH lines 1-3 yields the following output.

```
Available: True , Count: 4
Name: /physical_device:GPU:0 , Type: GPU
Name: /physical_device:GPU:1 , Type: GPU
Name: /physical_device:GPU:2 , Type: GPU
Name: /physical_device:GPU:3 , Type: GPU
```

Running the code WITHOUT lines 1-3 yields the following output.

```
Available: True , Count: 2
Name: /physical_device:GPU:0 , Type: GPU
Name: /physical_device:GPU:1 , Type: GPU
```

2.4 Jupyter notebook/lab

The following is a description on how to use Jupyter notebook/lab on UiOs computers.

1. Open two terminals, hereby called *term1* and *term2*.
2. In *term1*, log into `nam-shub-01` by typing the following.

```
ssh -Y nam-shub-01
```

The flag `-Y` enables trusted X11 forwarding.

3. Still being in *term1*, start Jupyter notebook by typing the following.

   ```
   jupyter notebook --no-browser --port=8081
   ```

   Jupyter lab may be started in similar fashion. The port chosen (=8081) to transfer data is somewhat arbitrarily chosen, however we need to ensure that we choose a port which is not used for anything else.

4. In *term2* we now want to start forwarding the ssh signal. Do this by typing the following.

   ```
   ssh -N -L 8081:localhost:8081 nam-shub-01
   ```

   The flag `-N` states not to execute a remote command. This is useful for just forwarding ports (protocol version 2 only). The flag `-L` specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side.

5. Now that you have obtained connection through given port, go back to *term1* where you can find some links which can be copied to your browser, to be able to view the notebook.

2.5 Anaconda support

Many users will be familiar with Anaconda as a Python virtual environment and package manager. The server has Anaconda support, and you can create your own personal environment as opposed to using the base environment on the server. To create your own environment, first load the Anaconda module.

```
module load Anaconda3
```

You then need to initialize the Anaconda environment.

```
module load Anaconda3
```

We now have an active base Anaconda environment. However, to fully utilize Anaconda, we would often like to create our own environment where we can add packages to serve our specific purposes. This environment will be stored in your `home` folder on the UiO servers. Anaconda has a specific command for adding the relevant executables to your `.bashrc` file.

```
conda init bash
```

This will append the necessary lines. However, the `.bashrc` file does not run automatically when you login to the server. Thus you will have to manually run these lines, by typing in

```
. .bashrc
```

After this has been completed, you can create a new Anaconda environment by typing

```
conda create --name [name_of_environment]
```

You can then activate your environment by typing

```
conda activate [name_of_environment]
```

You can now install any relevant packages you might need in your very own virtual Python environment. Remember that every time you login to the server, you will have to run the commands

```
module load Anaconda3
. .bashrc
conda activate [name_of_environment]
```

to load your personal conda environment. To simplify this, you can optionally create a bash script to do this for you. The bash script will look something like the three lines above, but with an added

```
#!/bin/sh
```

as the first line of the script, i.e. at the top. After you have created your bash script, and called it `[name_of_script].sh`, make sure you make it executable by running

```
chmod +x [name_of_script].sh
```

Now you can run your script by typing

```
. [name_of_script].sh
```

and your Anaconda environment is ready to go.

2.6 VSCode support

VSCode has ssh support for working in your home environment on the UiO servers. By using VSCode, you can easily browse, edit, and generally manage the files on your home environment.

To initialize VSCode ssh support, you start by simply opening VSCode on your client / computer. In the bottom right of the VSCode window there should be a green button which you can click. If you have set up your `.ssh` folder according to this guide, VSCode should find your ssh aliases and prompt you with a menu where you can select either the `nam-shub-01` alias or the general `uio` environment.

If this doesn't show up, you will need to set these manually. This guide (which can be found at `https://code.visualstudio.com/docs/remote/ssh`) can provide more detailed information on how to go about setting up ssh support with VSCode.

2.7 Process handling

To get a list of active processes on CPU, type `htop`. To see status on GPUs, type `nvidia-smi`.

2.8 Additional

If you want to repeat a command in the terminal every N second, type the following.

```
watch -n [N] [command]
```