DL4MIA 2021: Connection Tutorial

This tutorial will help you to connect to our cluster, claim a compute node that contains a huge GPU, run the tools we need on it, and then connect to it via your browser.

This tutorial might either be totally trivial for you (in case you use these technologies already for your work) or it might be SUPER CONFUSING, in case you never came in touch with all this.

Hint: go through every page, every paragraph, every step… ask us if something is unclear, let us know if something does not work. Take all the time it needs, you will need to do all this over and over again throughout the entire week.

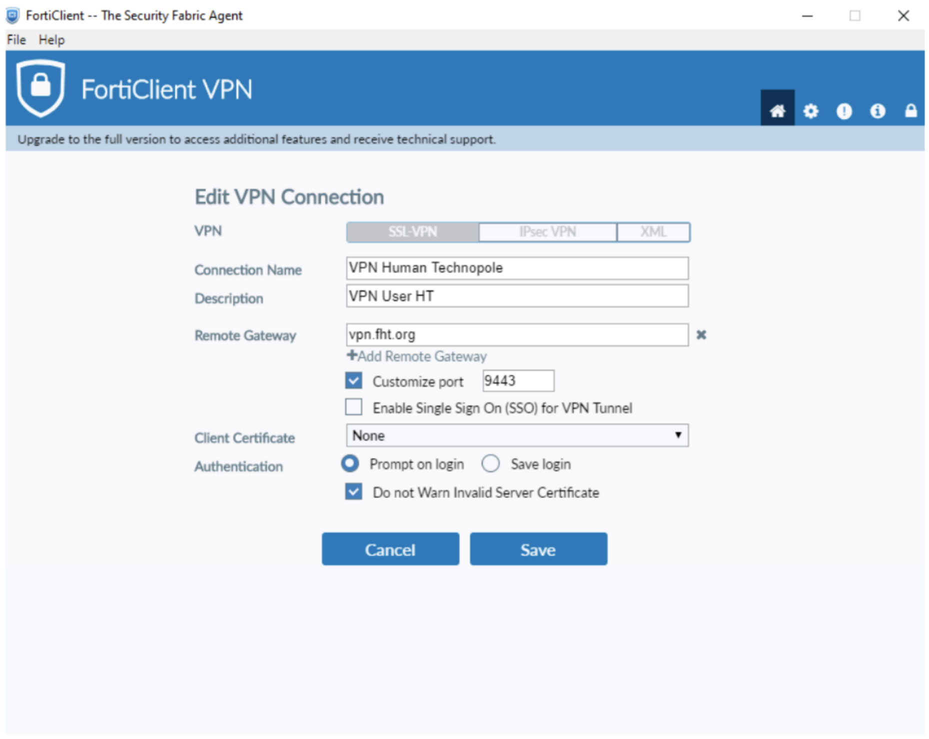
Table of Contents

Exercise #1: Connect to HT by VPN

Note: you have likely already done this exercise last Friday. If you, just connect the VPN and go to Exercise 2 or even Exercise 3 (after connecting via `ssh`)… otherwise… please go on…

Using a virtual private network (VPN) connection, you will be able to access HT's IT infrastructure as if you were on site. The connection to the VPN is established using FortiClient.

1. Download FortiClient for free and for your particular operating system in the "Product Downloads" tab: https://www.fortinet.com/support/product-downloads
2. Create and save a "New VPN Profile" with the following settings:



FortiClient -- The Security Fabric Agent
File Help
FortiClient VPN
Upgrade to the full version to access additional features and receive technical support.

Edit VPN Connection

VPN: [✓ SSL-VPN] IPsec VPN XML

Connection Name: VPN Human Technopole

Description: VPN User HT

Remote Gateway: vpn.fht.org

+Add Remote Gateway

☑ Customize port 9443

☐ Enable Single Sign On (SSO) for VPN Tunnel

Client Certificate: None

Authentication: [✓ Prompt on login] Save login

☑ Do not Warn Invalid Server Certificate

3. Connect to the VPN using the credentials and passwords that were assigned to you.

Exercise #2: Connect to the HPC cluster at HT

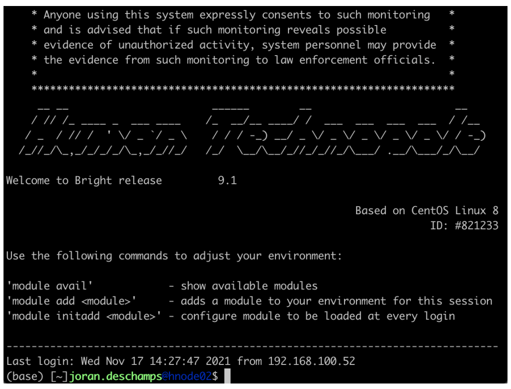
Because deep learning requires lots of computation power, training of neural networks is almost always performed on graphics processing units (GPUs). In order to supply you with this massive processing power, this week we share with you our high performance computing (HPC) infrastructure[1].

In order to connect to the HPC, we will use the so-called Secure Shell (**`ssh`**), which is a secure network protocol, allowing you to connect to and use a remote server.

1. In order to connect via `ssh` to HT's HPC cluster, you will need a suitable terminal or command-line tool. Unix-based systems (Linux and macOS) come with a native terminal. For Windows users, please install Git for Windows: https://gitforwindows.org/ *(other options exist and if you are familiar with them, please feel free to use them)*.
2. Start your terminal application.
3. We will be using many commands throughout the course. The first one allows us to connect to the HPC login node. Type the following (omitting the "$", which only means that this is a command to enter in the terminal):

   ```
   $ ssh <user.name>@hpclogin.fht.org
   ```

   where `<user.name>` is your login. Then, proceed to enter your password. Your terminal should then look something like this:



```
    * Anyone using this system expressly consents to such monitoring   *
    * and is advised that if such monitoring reveals possible          *
    * evidence of unauthorized activity, system personnel may provide  *
    * the evidence from such monitoring to law enforcement officials.  *
    *                                                                  *
    ********************************************************************
     __ __                   ______      __                __
    / // /_ ____ _  ___ ____   /_  __/__ ____/ /  ___  ___  ___  ___  / /__
   / _  / // /  ' \/ _ `/ _ \   / / / -_) __/ _ \/ _ \/ _ \/ _ \/ _ \/ / -_)
  /_//_/\_,_/_/_/_/\_,_/_//_/  /_/  \__/\__/_//_/_//_/\___/ .__/\___/_/\__/

Welcome to Bright release        9.1

                                                        Based on CentOS Linux 8
                                                                    ID: #821233

Use the following commands to adjust your environment:

'module avail'            - show available modules
'module add <module>'     - adds a module to your environment for this session
'module initadd <module>' - configure module to be loaded at every login

-------------------------------------------------------------------------------
Last login: Wed Nov 17 14:27:47 2021 from 192.168.100.52
(base) [~]joran.deschamps@hnode02$
```

1 To illustrate the "high performance part", HT's HPC has 1340 compute cores (CPU), 26TB of RAM memory and 60 NVIDIA V100 SXM2 with 32GB of memory (per GPU).

Exercise #3: Identify HPC node and port

In this course, we will work with jupyter notebooks. They provide an intuitive interface to run code snippets within your browser. Within jupyter notebooks, we will be able to run entire deep learning pipelines, from exploring the data to training a neural network and predicting results on unseen images. We currently have for each participant in the course a jupyter server running on the HPC. In order to connect to the jupyter server and interact with notebooks we will need to have the command-line prompt forward data from the HPC to our local machine. This is achieved through *port forwarding*.

1. If you are not connected to the HPC, go through the previous section again.
2. The jupyter server writes data to a log (within the DL4MIA folder), let's print the last lines of the log to our terminal to confirm that it is indeed running:
   ```
   $ tail DL4MIA/jupyter.log
   ```
3. The output has some important details, for instance you should be able to see lines resembling:
   ```
   [I 06:53:38.716 NotebookApp] Jupyter Notebook 6.4.6 is running at:
   [I 06:53:38.716 NotebookApp] http://gnodeXX:YYYY/
   ```

   Here, ***gnodeXX*** (e.g. gnode02) is the HPC node the jupyter server is running on. The "g" means that this node has access to GPUs. The number **YYYY** (e.g. 8889) following the semicolon (:) is the port where the data from the jupyter server is sent. **Write down both pieces of information, we will need them later on ! We will now refer to the HPC node as <gnode> and the port as <port>.**
4. In order to forward the data from the cluster node to your local machine, we need to connect to the HPC in a slightly different manner. Stop the ssh connection:
   ```
   $ exit
   ```

Exercise #4: Connect to the HPC with port forwarding

We have seen that a jupyter server is running on the HPC under your user name. Additionally, the server is running on a node with access to a GPU and that the data is accessible through a particular port. Let's now forward the information to your machine and connect to the jupyter server.

1. Now we will simply reconnect to the HPC. From now on, the following command will be the only you will use to connect to the HPC.

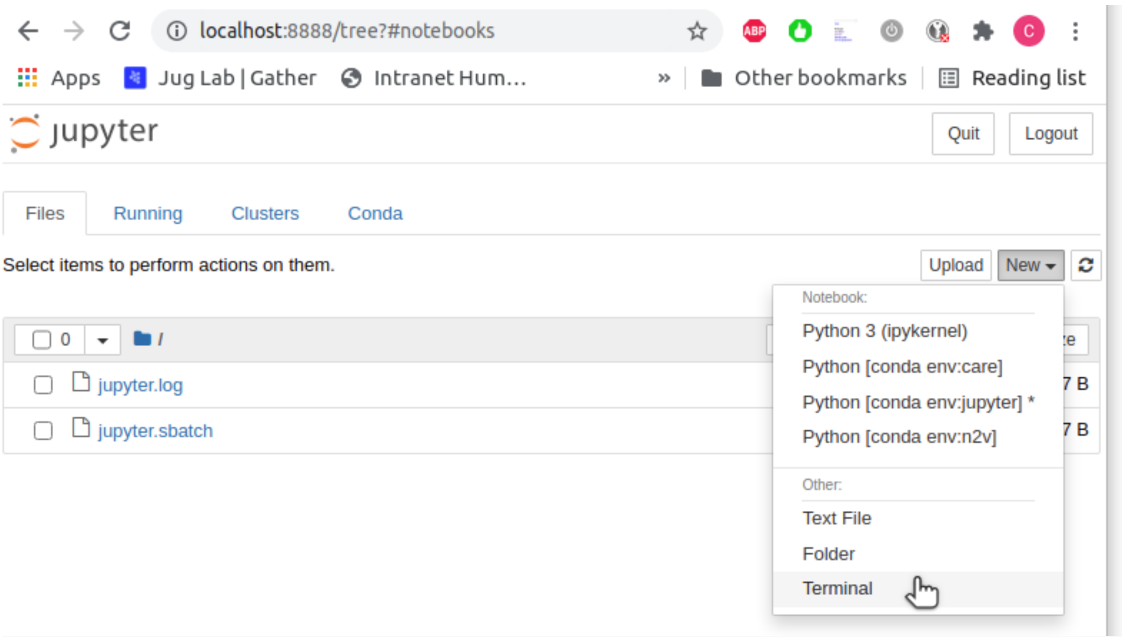
```
$ ssh <user.name>@hpclogin.fht.org -L 8888:<gnode>:<port>
```

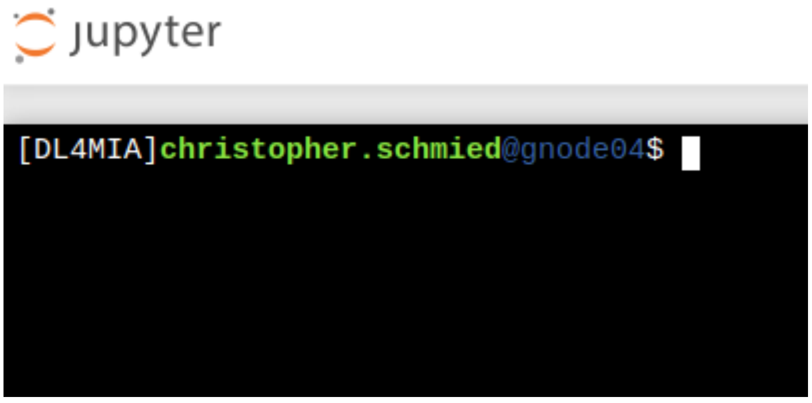
   Notice that we added a -L flag to the command we used previously. This means that you now open a local port with address 888 and that all data from the port number <port> on <gnode> should be forwarded to your local 8888 port.

2. Once connected, nothing special seems to have happened. Let's have a look at the jupyter server log, with a small twist: we will add the -f flag to make it update when new lines are written in the log.

   ```
   $ tail -f DL4MIA/jupyter.log
   ```
3. Now, in your browser, open http://localhost:8888. Enter your password. You then access a page that is mostly empty. Have a look at the output of your terminal. Did anything change?
4. You are now connected to the jupyter server directly on the HPC. But how can we be sure we are on the gnode? In the top right corner of the window, click on "*New*" and then "*Terminal*". Notice that the command line prompt is similar to:

   ```
   [DL4MIA]<user.name>@<gnode>$
   ```





```
[DL4MIA]christopher.schmied@gnode04$
```

Exercise #5: Conda and jupyter

From now on, we will not touch your terminal any more as it allows us to see any error message in the jupyter log. We will instead be using a terminal created directly within the jupyter server, as done in the last point of the previous section. Here we will demonstrate how to install a Conda environment and use it together with a jupyter notebook.

Conda is a powerful tool that allows installing all the packages necessary to run a certain Python pipeline within a boxed environment. This prevents installing incompatible package versions (or even Python versions!) when using different pipelines / projects. In this course, you will learn how to install a Conda environment for a specific project and how to switch between them.

1. In this section, we will use a dummy project that you can find in a Github repository: https://github.com/dl4mia/00_connection

   Often, Python git repositories have a README file with detailed instructions on how to create the Conda environment required to run the project.

2. Run the first command:
   ```
   $ conda create -n 00_connect python==3.8
   ```
   This downloads a version of Python 3.8 with basic packages.
3. As suggested by the Conda, let's activate the conda environment:
   ```
   $ conda activate 00_connect
   ```
4. Did anything change? (tip: look at the command-line prompt)
5. We can now install some packages that will only exist within this specific Conda environment:
   ```
   $ conda install numpy jupyter
   ```
   You might need to enter "y" in the terminal to accept installing the necessary packages.
6. Now we have a conda environment, but no jupyter notebook. Let's use git to download the content of the Github repository:

   ```
   $ git clone https://github.com/dl4mia/00_connection.git
   ```

We will not dive in details in git during this course, but it is a very important tool used for version control and code sharing.

7. Go back to the jupyter server tab in your browser. See the newly created folder? Navigate inside and start the jupyter notebook (*.ipynb).
8. Try to run the notebook following the succinct instructions. **You should get an error**. The problem is that the code uses *numpy* (in the "import" call) but the notebook is running within an environment that does not contain numpy. Yet we installed numpy (see point 5). What is the problem?
   We are currently running the notebook with a Python kernel called "Python [conda env:**jupyter**]" (top right corner). This means that our current Conda environment is called "*jupyter*".
9. In "*Kernel*", go to "*Change Kernel*" and select the Kernel with the conda environment "*00_connect*".
10. Wait for the kernel to restart and run the notebook.

Here are some notes on jupyter:

- The proper way to close a notebook is to do "File > Close and halt".
- If you simply close the window, you can see that in the server page that the notebook is highlighted in green. You can still shut it down by selecting it and clicking on "Shutdown".
- If you click on "Logout", you will stop your connection to the whole jupyter server. Don't panic, just open http://localhost:8888 again.