XCS224N Azure Guide

This guide will help you set up and use Azure Virtual Machines for Assignments 4 and 5. Before you start, it cannot be stressed enough: **do not leave your machine running when you are not using it**!

Access and Setup

Azure Labs Subscription for this Class

We are using [Azure Lab Services](#) to manage VMs for the XCS224N course. Every student will be allocated <u>65 hours total</u> for completing Assignments 4 and 5 combined. **It's very important for students to manage credit wisely in order to make the most efficient use of it (see next section).**

Credit has been assigned per student and everyone's instances are preconfigured with Linux DSVM (Data Science Virtual Machine) images so you can expect most of the required packages/tools to be installed.

Best Practices for Managing Credit

Azure virtual machines are charged at a flat rate for each minute they are turned on. This is irrespective of:
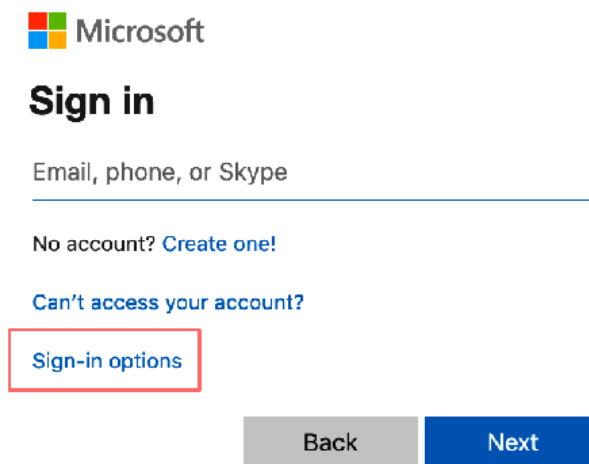- whether you are ssh'd to the machine at that time
- whether you are running any processes on the machine at that time
- the computational intensity of the processes you're running
- whether you're using GPUs

Therefore, the most important thing for managing credit wisely is to **carefully turn your VM on and off only when you need it**.

As you will see described in the Assignment 4 and 5 handout, we advise you to **develop your code on your local machine** (for example your laptop with the CPU version of Pytorch installed) for debugging (i.e., work on your new code until you are able to complete several training iterations without errors), then run your code on your Azure VM when it's time to train on a GPU.

Registration

1. Go to this link: https://labs.azure.com/register/7vutgh4y

2. You'll be presented with a large number of options to register. They are:
   A. Logging in with an existing Microsoft account using the email/phone associated with it
      OR
   B. Logging in with a Skype account
      OR
   C. If you click 'Sign-in Options' you will also be presented with the option to sign in using your GitHub credentials


Once you've done A, B, or C - follow any additional prompt instructions (depends on which way you chose) - and you will be registered for the lab!
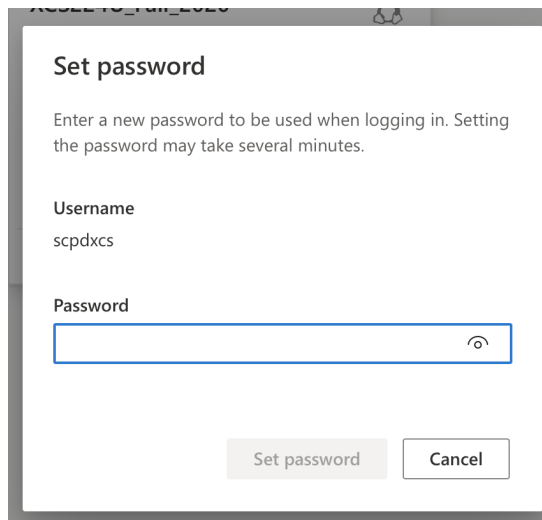
Connecting to a VM

1. After signing in you'll be directed to an Azure Lab Services portal where you can view all your virtual machines. Unless you've used Azure Lab Services before, you'll see only one machine along with your remaining hours.

   Click on the 'Stopped' button to start the instance (this will take a few minutes). When it is up and running, it will look like this and say "Running" in the bottom status bar:
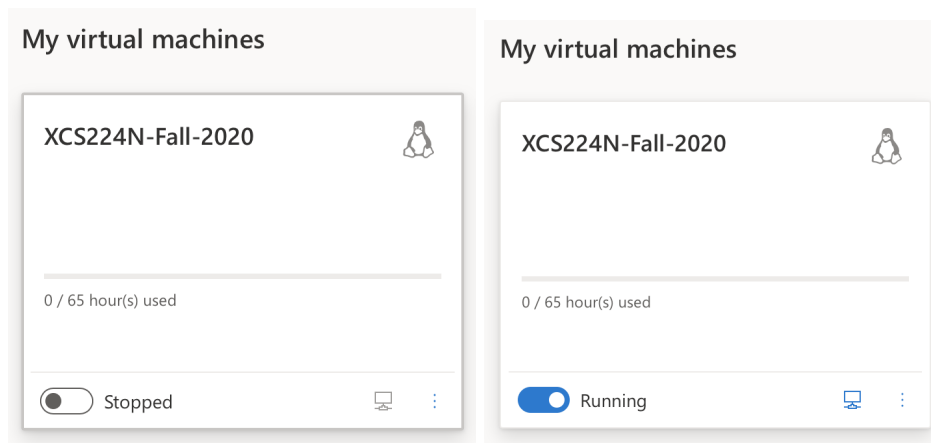
2. Click the monitor icon in the window above and you'll be asked to set the instance password (**make sure you remember/record this password as you will be asked to enter it when logging into your VM via SSH**).
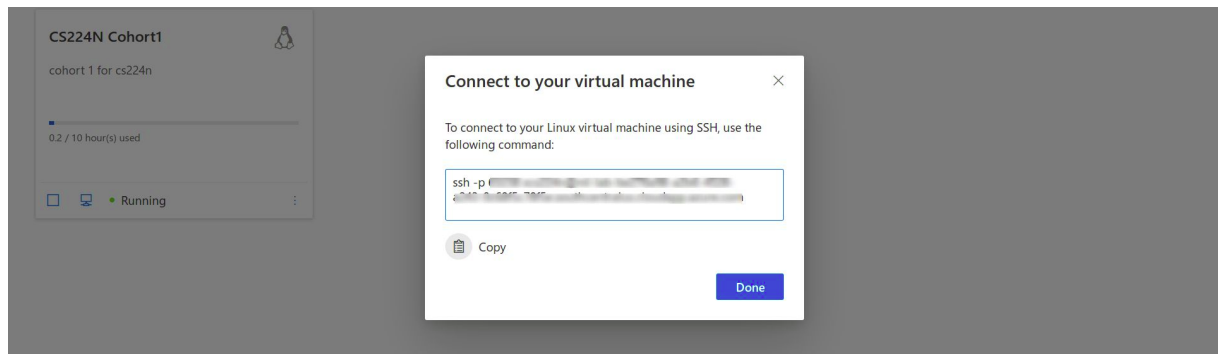
3. Click on the monitor icon and select *'connect via SSH'* to get the SSH link.

Note that you could log in from your development machine to the azure VM without password if you setup the ssh key-based authentication setup, review this link for details.

4. Copy the link and paste it into your terminal (Windows users can use PuTTY)

5. Important, Update the Azure VM (as of 2021)

Due to the recent changes on Azure DSVM and Nvidia driver, which is not compatible with installed GPU. Please run the following commands.

```
sudo apt purge *nvidia* -y
sudo apt autoremove -y
sudo apt update -y
sudo apt-get install nvidia-driver-470 -y
sync
sudo reboot now
```

After rebooting, this is what you should see. Confirm before continuing.

```
scpdxcs@ML-RefVm-795076:~$ nvidia-smi
Fri Feb 11 04:54:09 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000001:00:00.0 Off |                    0 |
| N/A   42C    P0    59W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+

scpdxcs@ML-RefVm-795076:~$ modinfo nvidia | grep version
version:         470.103.01
srcversion:      79522292FECC5F0FF4A61D6
vermagic:        5.4.0-1062-azure SMP mod_unload modversions
```

6. Start by checking that Pytorch can access the GPUs. From the command line, activate one of the preinstalled conda environments by typing, this has been updated from previous conda environment.

conda activate **py38_pytorch**

Open an interactive Python prompt by typing `python` into the command line. Python should greet you by letting you know that it's running Python 3.6.9. *Into the Python prompt*, type each of these lines and then press Enter:

```
import torch
torch.cuda.current_device()
torch.cuda.device(0)
```

```
torch.cuda.device_count()
```
.

You should see something like this:

If you receive error messages or find that this isn't working, post to Slack and/or reach out to your Course Facilitator.

7. Return to the Azure Lab Services portal, click on the "Running" button. After a few minutes, the status should update to "Stopped." **Be sure to do this whenever you are not actively running code on the VM.**

Setting up a github ssh key could be helpful to pull the code through github, see the details on **Practical Guide for Using the VM** section.

As you're working on Assignments 4 and 5, instead of activating a preinstalled environment by typing `conda activate azureml_py36_pytorch`, after you coy all of the code and files from the repository to your Azure instance, you'll create a new conda environment by typing, and the environment.yml file is in the git repository, such as this link.

NOTE that this has been updated from the previous command.
```
$ conda env create -f ./environment_gpu.yml
```

This will ensure all of the correct versions of the libraries you'll need for the assignment are installed and create a new environment that you can activate by typing:

NOTE that this environment has been updated.
```
$ conda activate XCS224N_GPU
```

Troubleshooting guide

Here are a few troubleshooting tips we have noticed during the environment setup.
If the issue persists after trying the following steps, please reach out to the teaching team.

1. Even after you have created and activated the conda setup, you will see the module missing.

In this case, reboot the VM and see if this issue persists. It is likely due to the conda environment and installation issue.

2. When you run `nvidia-smi` command and see an error message instead of GPU status report.

In this case, reboot the VM and see if the issue persists. This is likely due to installation of CUDA environment during conda setup.

Practical Guide for Using the VM

Managing Processes on a VM

In developing your deep learning models, you will likely have to leave certain processes, such as Tensorboard and your training script, running for multiple hours. If you leave a script running on a VM and log-off, your process will likely be disrupted. Furthermore, it is often quite nice to be able to have multiple terminal windows open with different processes all visible at the same time, without having to SSH into the same machine multiple different times.

TMUX or "Terminal Multiplexer" is a very simple solution to all the problems above.

Essentially, TMUX makes it such that in a single SSH session, you can virtually have multiple terminal windows open, all doing completely separate things. Also, you can actually tile these windows such that you have multiple terminal sessions all visible in the same window.

The basic commands are below. Terminal commands are prefaced with a " $ " otherwise the command is a keyboard shortcut.

TMUX Cheatsheet

1. Start a new session with the default name (an integer) `$ tmux`

2. Start a new session with a user-specified name `$ tmux new -s [name]`
3. Attach to a new session `$ tmux a -t [name]`
4. Switch to a session `$ tmux switch -t [name]`
5. Detach from a session `$ tmux detach OR ctrl - b - d`
6. List sessions `$ tmux list-sessions`
7. Kill a session `ctrl - b - x`
8. Split a pane horizontally `ctrl - b - "`
9. Split a pane vertically `ctrl - b - %`
10. Move to pane `ctrl - b - [arrow_key]`

Managing Code Deployment to a VM

There are multiple options to transfer files between your VM and your local computer.

SCP (secure cp)

One option is to use a tool called `scp`, which stands for "secure copy". `scp` uses a similar command to `ssh` for transferring files to and from your VM. Let's say you can access your VM with the following `ssh` command:

```
ssh -p 54003 scpdxcs@ml-lab-XXXXXXXXXXXXX.southcentralus.cloudapp.azure.com
```

To transfer files to the VM from your local machine, use the following command (differences highlighted):

```
scp -r -P 54003 path/to/local/file scpdxcs@ml-lab-XXXXXXXXXXXXX.southcentralus.cloudapp.azure.com:path/to/remote/destination
```

To transfer files from the VM to your local machine, use the following command:

```
scp -r -P 54003 scpdxcs@ml-lab-XXXXXXXXXXXXX.southcentralus.cloudapp.azure.com:path/to/remote/file path/to/local/destination
```

The `-r` option indicates that a recursive copy should be performed, meaning that you can transfer an entire directory structure with just this one command! (note that the `-p` (lowercase) is now a `-P` (uppercase))

GIT

A different solution is to use a version control system, such as **Git**. This way, you can easily keep track of the code you have deployed, what state it's in and even create multiple branches on a VM or locally and keep them sync'd.

The simplest way to accomplish this is as follows.

1. Create a Git repo on Github, Bitbucket or whatever hosted service you prefer.
2. Create an SSH key on your VM. (see the link below)

3. Add this SSH key to your Github/service profile.
4. Clone the repo via SSH on your laptop and your VM.
5. When the project is over, delete the VM SSH key from your Github/service account.

Resources:
- [Github SSH key tutorial](#)
- [Codecademy Git tutorial](#) (great for Git beginners to get started)

*Note: If you use Github to manage your code, <mark>you must keep the repository **private** until the class is over.</mark>*

rsync (remote sync)

`scp` commands copies files all the time regardless of the file changes from the source to the destination; however, `rsync` only copies files when the file is updated on the source location. So if you have a large file (such as a model), then `rsync` could be better in terms of time and data transfer point of view. [Here](#) is a tutorial on how to use it. <mark>Just remember trailing slash (/) in the source path</mark> is important as in the tutorial.

Here is an example of *rsync* command with specific port that can be found from Azure web:

```
rsync  -arvz -e  'ssh -p PORT_NO --progress /Users/name/SCPD/XCS224N/A4/ scpdxcs@ml-lab-xxx:/home/scpdxcs/SCPD/XCS224N/A4
```

Managing Memory, CPU and GPU Usage on a VM

If your processes are suddenly stopping or being killed after you start a new process, it's probably because you're running out of memory (either on the GPU or just normal RAM).

First of all, it's important to check that you are not running multiple memory hungry processes that may have slipped into the background (or a stray TMUX session).

You can **see/modify which processes you are running** by using the following commands.

1. View all processes `$ ps au`
2. To search among processes for those containing the a query, use
   `$ ps -fA | grep [query]`.
   For example, to see all python processes run `ps -fA | grep python.`
3. Kill a process `$ kill -9 [PID]`

You can find the PID (or Process ID) from the output of (1) and (2).

To **monitor your normal RAM and CPU usage**, you can use the following command: `$ htop` (Hit `q` on your keyboard to quit.)

To **monitor your GPU memory usage**, you can use the `$ nvidia-smi` command. If training is running very slowly, it can be useful to see whether you are actually using your GPU fully. (In most cases, when using the GPU for any major task, utilization will be close to 100%, so that number itself doesn't indicate an Out of Memory (OOM) problem.)

However, it may be that **your GPU is running out of memory simply because your model is too large** (i.e. requires too much memory for a single forward and backward pass) to fit on the GPU. In that case, you need to either:

1. Train using multiple GPUs (this is troublesome to implement, and costs much more on Azure)
2. Reduce the size of your model to fit on one GPU. This means reducing e.g. the number of layers, the size of the hidden layers, or the maximum length of your sequences (if you're training a model that takes sequences as input).
3. Lower the batch size used for the model. Note, however, that this will have other effects as well (as we have discussed previously in class).

Tunneling on Azure VM for Jupyter

(Thanks Luis Valerio Hernandez for sharing this method! Here is the link we use.)

You may need to run the notebook located in the server from your own computer. In that case, you will need to establish a tunnel between your local computer and the remote computer. Below are the necessary steps to establish it.

[Reference: documentation for using jupyter with multiple options.](#)

Step 1: Setup SSH Tunnel

Access your remote machine with a regular ssh command with an additional part that establishes a tunnel between a `local port` and `target port` as exemplified below.

```
PS C:\Users\Admin> ssh -p 63616 xcs224u_student@***********.southcentralus.cloudapp.azu
re.com -L 8080:localhost:8888
xcs224u_student@ml-lab-d96b8b7c-aabd-428a-874a-c14ab55dff7a.southcentralus.cloudapp.azure.com's
password:

(Some welcome messages here)

Last login: Wed Mar 11 08:43:55 2020 from 73.158.65.76
```

```
xcs224u_student@ML-EnvVm-00047:~$
```

Step 2: Run Jupyter

Simply run the jupyter as usual with an ip parameter [and an optional --port=#### parameter].

```
xcs224u_student@ML-EnvVm-00047:~/Desktop/cs224u$ jupyter notebook --no-browser --ip='0.0.0.0'
--port=8888
[I 08:51:55.080 NotebookApp] JupyterLab extension loaded from
/data/anaconda/envs/py35/lib/python3.5/site-packages/jupy
terlab
[I 08:51:55.080 NotebookApp] JupyterLab application directory is
/data/anaconda/envs/py35/share/jupyter/lab
[I 08:51:56.273 NotebookApp] sparkmagic extension enabled!
[I 08:51:56.273 NotebookApp] Serving notebooks from local directory:
/data/home/xcs224u_student/Desktop/cs224u
[I 08:51:56.273 NotebookApp] The Jupyter Notebook is running at:
[I 08:51:56.273 NotebookApp] http://(ML-EnvVm-00047 or
127.0.0.1):8888/?token=990b86c0ee61f2ba4f3808baa24bc6696c1b407c5
da63cba
[I 08:51:56.273 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip
confirmation).
[C 08:51:56.274 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://(ML-EnvVm-00047 or 127.0.0.1):8888/?token=990b86c0ee61f2ba4f3808baa24bc6696c1b407c5da63cba
```
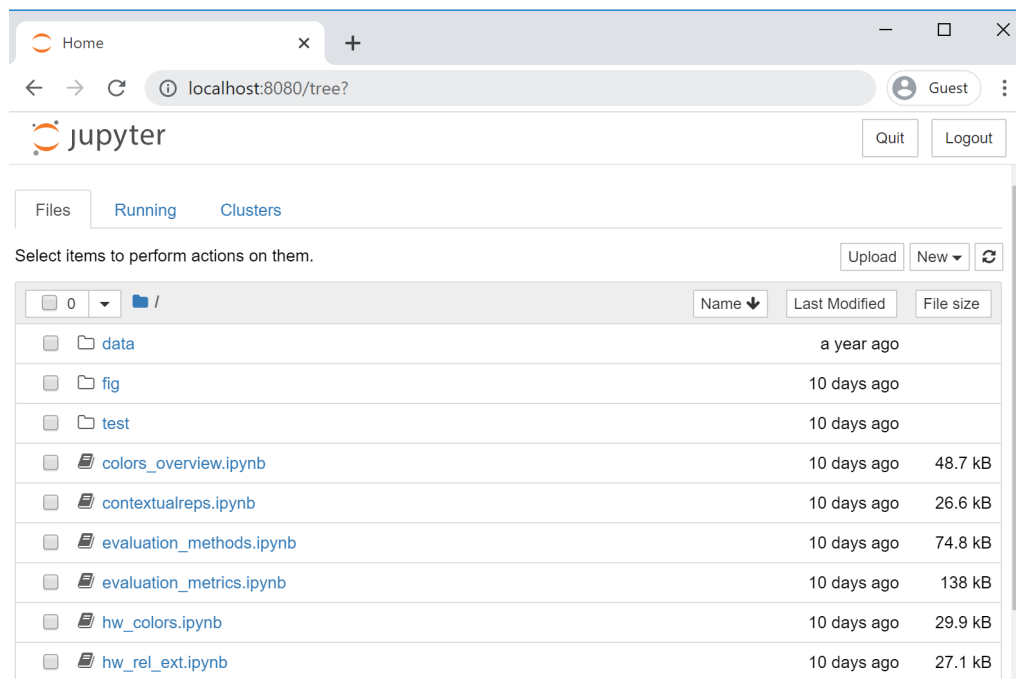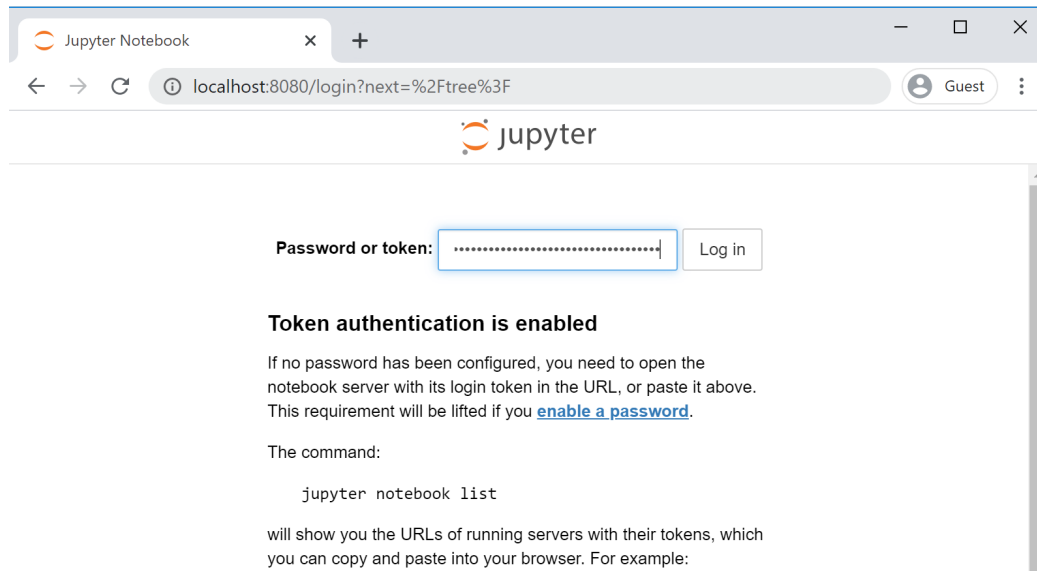
Step 3: Access Jupyter

Go to the link `http://localhost:8080` in the browser of your local machine and enter the `token`

As of 9/26/2021, Chrome browser on Mac did warn about the certificate, but didn't give me an option to go ahead and connect.

Safari (version 15) on Mac warned, but gave an option to continue. Once you accept that you will continue, a connection will be established.
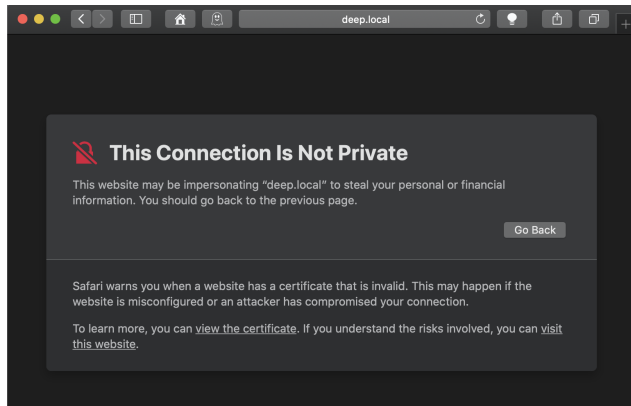
Step 4: Enjoy :)

This is the most important step! Enjoy experimenting with Jupyter on Azure VM :)

Troubleshooting

Browser says "This connection is not private".

Depending on the web browser it may flag a security alarm, but you can choose to select 'visit' the site as shown below examples from safari and chrome respectively.
You can choose to visit the site, so that you can connect to a jupyter notebook.

One might try to check these links (use with cautions) to find more about this case
- https://medium.com/analytics-vidhya/get-rid-of-ssl-errors-with-jupyter-notebooks-1a80dd509988