Updated: Feb 13 2019
NOTE: This handout should be used as introductory reference material only. There might be errors and/or improper methods, but it should be enough to get you going. If you have suggestions or improvements, please post them to the Piazza page so your fellow students can gain benefit as well. **END NOTE**

Most modern data science projects are a balance between model computational complexity and computational resources. Many methods are simply infeasible as they do not scale well.

However, even efficient computations can easily overload personal computers, and some operations like matrix multiplication can be sped up on dedicated components such as GPUs and highly parallelized on clusters of many GPUs.

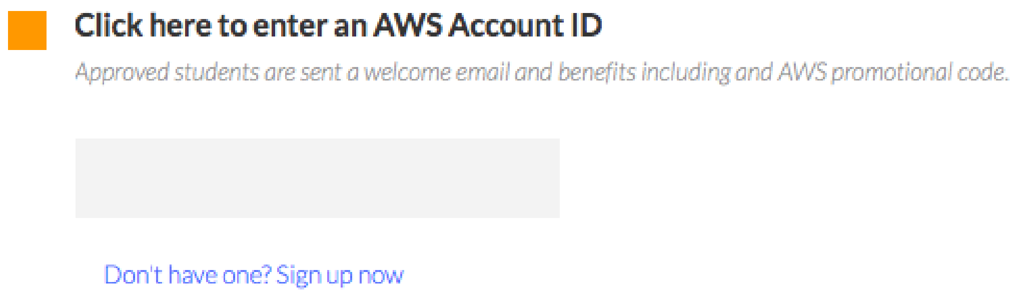
Training a model that could take hours on personal laptops can be reduced to several minutes on appropriate cloud computing platforms.

For COMP 540 we recommend using Amazon Web Services (AWS). They offer student promotions that come with $100 of AWS Credit. Free credits can be registered using rice emails. In addition, Rice alias emails are not recognized as duplicates so you can effectively get 5 student credit promotions!

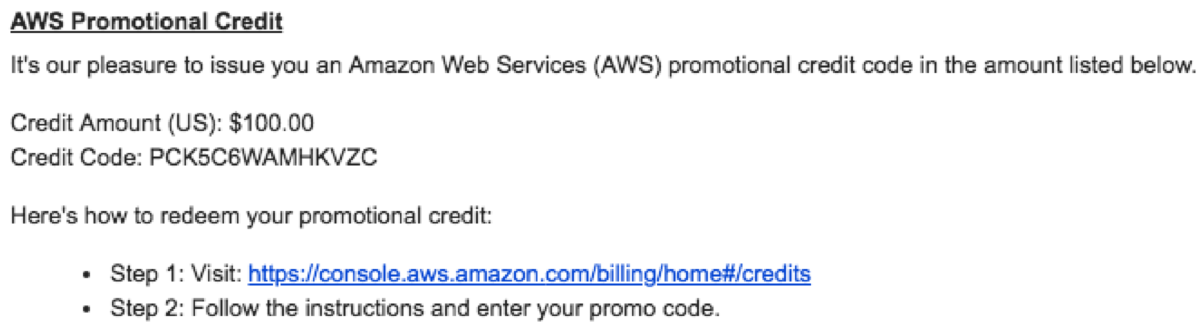
Rice Alias emails:
https://kb.rice.edu/page.php?id=65429

To get Credit there are two paths:
(Recommend) Create a personal account:

1. Create an AWS account (with a non rice email)
    a. https://portal.aws.amazon.com/billing/signup - /start
    b. Get Account Id: #############
2. Register as a student (using your rice email) on https://aws.amazon.com/education/awseducate/:
    a. Click "Click here to enter an AWS account ID"
    b. This will allow you to gain Education Credit on your Personal Account.
    c. This will require your application to be reviewed (Might not be done instantly)
    d. Enter your account ID (made with a non-rice email) in the number below to get $100 credit on your account.


Click here to enter an AWS Account ID
Approved students are sent a welcome email and benefits including and AWS promotional code.
Don't have one? Sign up now

e. You should get an email with a section similar to the figure below
f. Using your promo code you can get AWS Education credit on your perosnal account.



AWS Promotional Credit

It's our pleasure to issue you an Amazon Web Services (AWS) promotional credit code in the amount listed below.

Credit Amount (US): $100.00
Credit Code: PCK5C6WAMHKVZC

Here's how to redeem your promotional credit:

- Step 1: Visit: https://console.aws.amazon.com/billing/home#/credits
- Step 2: Follow the instructions and enter your promo code.

(Alternate) Create a student account account:

1. Create an AWS education account on https://aws.amazon.com/education/awseducate/
2. You may have limits on GPU Access but you can request for these limits to be removed (see footnote 2 at **Booting up relevant servers**)

Once you have activated, and funded your account you can begin to unlock the powers of cloud computing!

Before starting if you don't have a Linux based terminal console downloading putty or a similar emulator is highly recommended. All Mac OS, or Linux based computers will already have terminal installed which is sufficient. However, Windows operators will need to take a few minutes to download Putty and become familiar with BASH commands.

Putty:
https://www.putty.org/
BASH Commands:
http://linuxcommand.org/lc3_learning_the_shell.php
The first 8 pages or so of this tutorial should give you most of the commands you will need to know to start.

In most settings for COMP 540, we will simply be using the EC2 instances as short term (a few minutes to several hours) power houses to chug through training of models. It is unlikely (and expensive) that anyone will need to maintain a server for any significant period of time (for a few days or more) so we will be explaining/demonstrating the basic below:

1. AWS Keys
2. Booting up relevant servers
3. Connecting to the server
4. Disconnecting from the Server/Shutting Servers Down
5. Port Forwarding/Jupyter Notebooks
6. Tmux
7. Uploading/Download data
8. Bash Scripts

AWS Keys

AWS Keys are your access code to any AWS server that you launch.[1]

For short term “simple” school projects storing AWS keys in the directory that the SSH shell is launched from will be fine.

If you lose a key, you will not be able to regain access to the files or computation on an AWS server that was launched with that specific key. Your only option would be to shut down that instance and create a new key.

Creating a Key:
On https://aws.amazon.com/ after login in
Navigate: Services > Compute > EC2 > Key Pairs
Create a Key Pair.
A .pem file should be downloaded.
Move this file to a safe location.

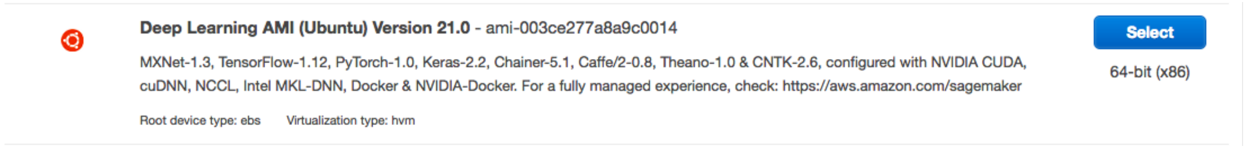
In addition,
Using terminal (or terminal emulator like putty):

```
>>>chmod 400 /path/my-key-pair.pem
```

AWS requires this set of permissions for the *.pem file

Booting up relevant servers

Amazon makes booting up servers very easy.
You can login to your account and navigate to:
Services > Compute > EC2 > Launch Instance
We recommend using the **Deep Learning AMI (Ubuntu) Version 21.0**
This Amazon Machine Image should have a version of anaconda with python2 and python3 preloaded with almost all of the deep learning software you could need.


Deep Learning AMI (Ubuntu) Version 21.0 - ami-003ce277a8a9c0014
MXNet-1.3, TensorFlow-1.12, PyTorch-1.0, Keras-2.2, Chainer-5.1, Caffe/2-0.8, Theano-1.0 & CNTK-2.6, configured with NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. For a fully managed experience, check: https://aws.amazon.com/sagemaker
Root device type: ebs   Virtualization type: hvm

Click select

Here you will be presented with hundreds of different machines to choose between.

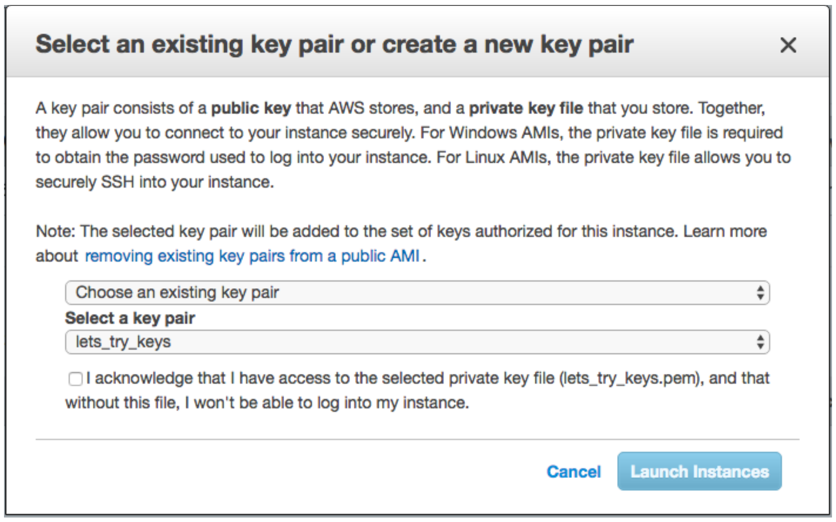
[1] These should be securely stored but this document will not talk about best practices for handling keys. See https://docs.aws.amazon.com/kms/latest/developerguide/overview.html if you want information on proper key management.

I would recommend using p2.xlarge— 4 BIG Nvidia GPUs that work well with TensorFlow and 61 GBs of RAM.[2] Notice, these instances are On-Demand and the pricing can be found here (https://aws.amazon.com/ec2/pricing/). P2.xlarge is $0.90 per hour. So while not expensive they can easily be forgotten about and cost you $20/day.

There are Spot Instances that offer up to 90% discounts in prices, but these require a lot more set up. (https://aws.amazon.com/ec2/spot/)

No matter which instance you select they will always have the same software and layout so moving up or down in size should not affect the code[3], only the performance.

Select your instance and click "Review and Launch"



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI .

Choose an existing key pair

Select a key pair

lets_try_keys

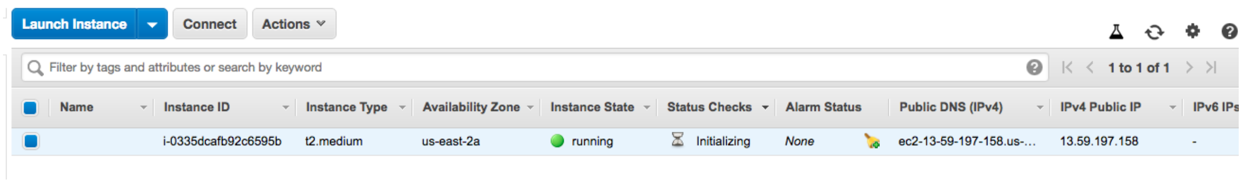
☐ I acknowledge that I have access to the selected private key file (lets_try_keys.pem), and that without this file, I won't be able to log into my instance.

The key pair you select make sure you have access to.

Once your instance is launched you can always find them under Services > Compute > EC2 > EC2 Dashboard > Running Instances.

Here you can monitor the CPU, memory, GPU, and network usage along with other things.

Connecting to the server



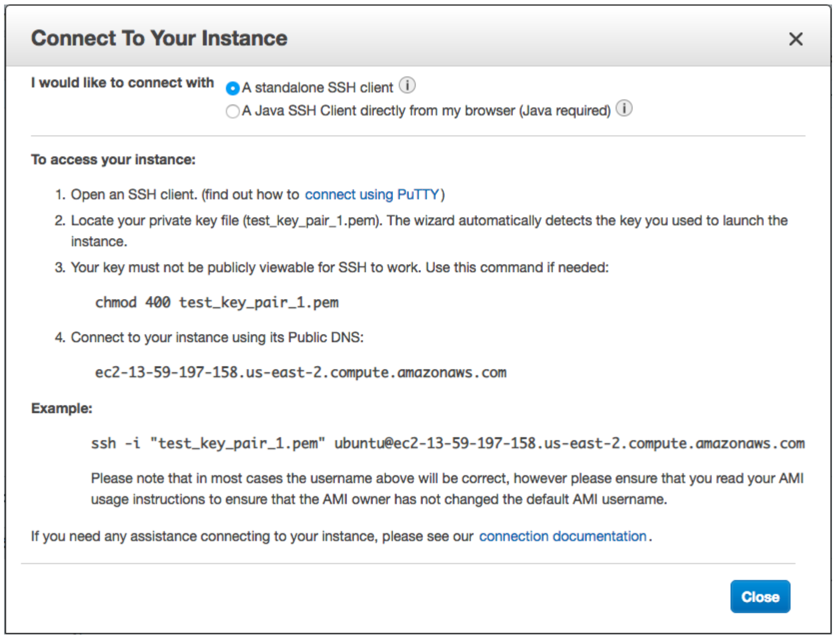
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
	i-0335dcafb92c6595b	t2.medium	us-east-2a	running	Initializing	None	ec2-13-59-197-158.us-...	13.59.197.158	-

Here I am running a t2.medium. If I want to connect, I will select the instance so the box to the left is blue and click connect. You should see something similar to the Figure below. The Public

[2] It is possible you might not have access to this instance. If that is the case, please go to Support > Support Center in the top right corner of the screen and create a "Service limit increase" case. Explain that you are a student doing research for a class and need to use GPU resources to efficiently test your models.

[3] Well Memory Errors might be a thing.

DNS of the AWS EC2 instance will appear along with the .pem file that will be needed to connect to the instance.


Connect To Your Instance

I would like to connect with [✓ A standalone SSH client] A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (test_key_pair_1.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 test_key_pair_1.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-13-59-197-158.us-east-2.compute.amazonaws.com
```

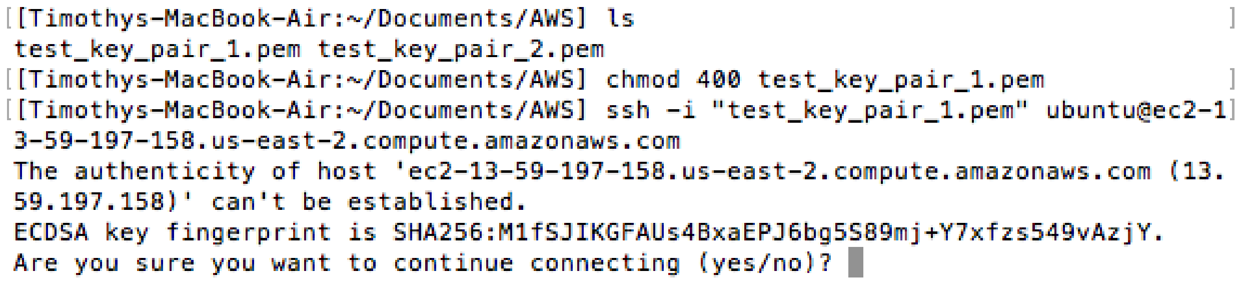
Example:

```
ssh -i "test_key_pair_1.pem" ubuntu@ec2-13-59-197-158.us-east-2.compute.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

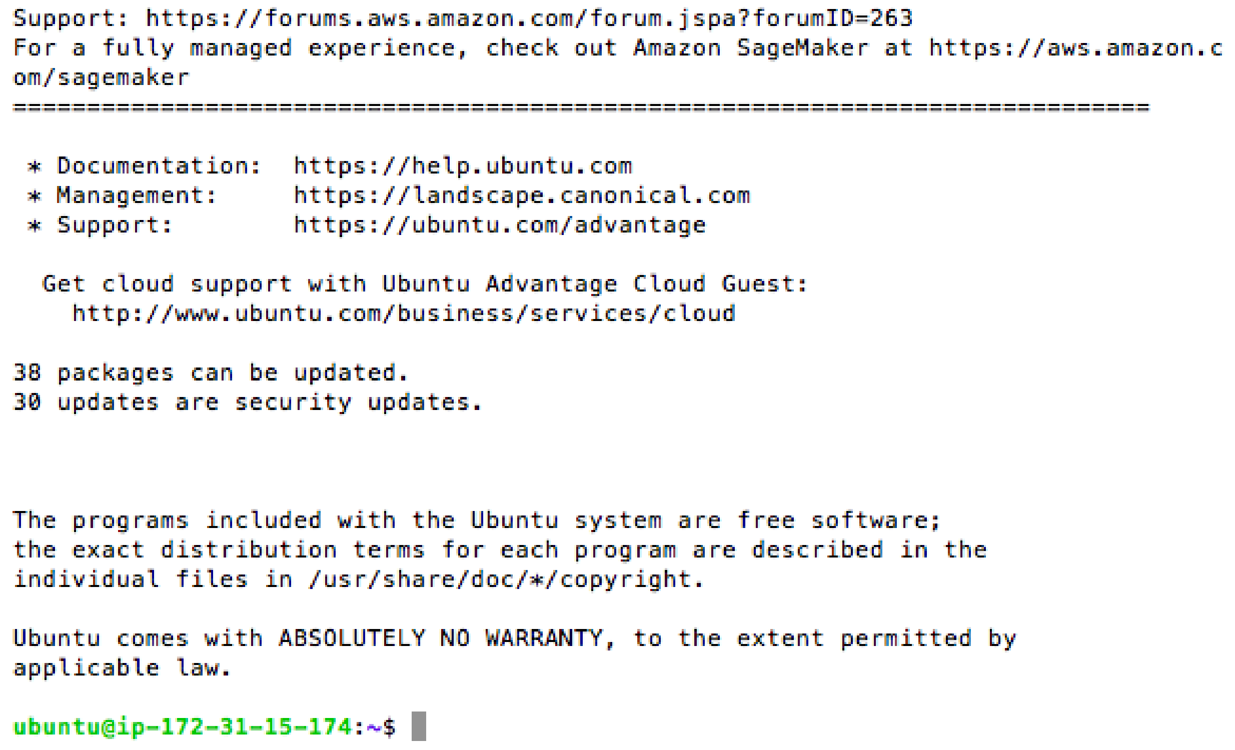
If you need any assistance connecting to your instance, please see our connection documentation.

Now, if one opens a terminal window and navigates to the directory that has the test_key_pair_1.pem one should be able to use the ssh command they give us and connect to the server.


```
[Timothys-MacBook-Air:~/Documents/AWS] ls
test_key_pair_1.pem test_key_pair_2.pem
[Timothys-MacBook-Air:~/Documents/AWS] chmod 400 test_key_pair_1.pem
[Timothys-MacBook-Air:~/Documents/AWS] ssh -i "test_key_pair_1.pem" ubuntu@ec2-1
3-59-197-158.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-13-59-197-158.us-east-2.compute.amazonaws.com (13.
59.197.158)' can't be established.
ECDSA key fingerprint is SHA256:M1fSJIKGFAUs4BxaEPJ6bg5S89mj+Y7xfzs549vAzjY.
Are you sure you want to continue connecting (yes/no)?
```

Here I run through the commands. Note that chmod 400 changes the permissions for a file and only needs to be done once.

Once I type (yes) I will be now connected to the AWS server, in my case, in Ohio. Once connected it should look similar to below.



```
Support: https://forums.aws.amazon.com/forum.jspa?forumID=263
For a fully managed experience, check out Amazon SageMaker at https://aws.amazon.c
om/sagemaker
=============================================================================

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

38 packages can be updated.
30 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-15-174:~$ 
```

Disconnecting from the Server/Shutting Servers Down

At any point you can simply close the terminal window and this will sever the connection. However, this will also stop any code or program that was initialized from that ssh terminal connection, unless you a multiplexer like tmux.

For example, if you simply ssh into your instance and run code in class and close your laptop when walking to the next classroom, your computer will most likely loss internet connection, sever the ssh connection to the EC2 instance, and the code will stop and most likely will be lost.

We will discuss how to get around this shortly.

Whenever you no longer interested in paying for you EC2 instance you can proceed back to the EC2 DashBoard and click on “Actions” > “Instance State” > “Stop”.

This will temporary move any data on this Instance to storage (that is quite cheap to pay for a few dollars a month). The instance will be stopped and you will no longer have access to it as it is turned off and wiped for the next customer. Then, you can reactivate this Instance with “Actions” > “Instance State” > “Start”. However, it is recommended that you saved/downloaded any important data to your local laptop before you do this!

Whenever you no longer interested in paying for you EC2 instance you can proceed back to the EC2 DashBoard and click on “Actions” > “Instance State” > “Terminate”.

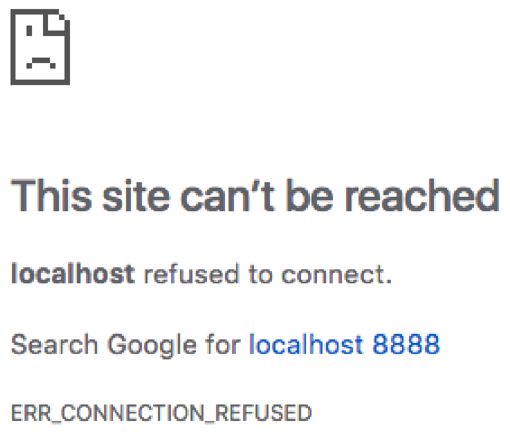
This will permanently delete any data on this Instance, and not save it in storage, as it is turned off and wiped for the next customer. Make sure you have saved/downloaded any important data to your local laptop before you do this!

Port Forwarding/Jupyter Notebook

Deep Learning AMI Instance will have Jupyter notebook already installed. Thus from the ssh connected terminal you can run:

```
>>>jupyter notebook
```

However, if you try to use the URL given to you once the notebook is launched, you will get an error. This is because the notebook is only hosted locally (In this case in Ohio on the EC2 AWS Instance). Your browser, on your laptop, doesn’t know how to “find” that notebook server. If you try to connect you will get:



This site can’t be reached

localhost refused to connect.

Search Google for localhost 8888

ERR_CONNECTION_REFUSED

You can connect slightly differently with an ssh command so that you can use a browser on your local computer to connect to the EC2 instance.

Here you will add a flag to the port “–L 8157:127.0.0.1:8888”, where 8157 is the local port that will be used with jupyter notebooks.

Without port forwarding:

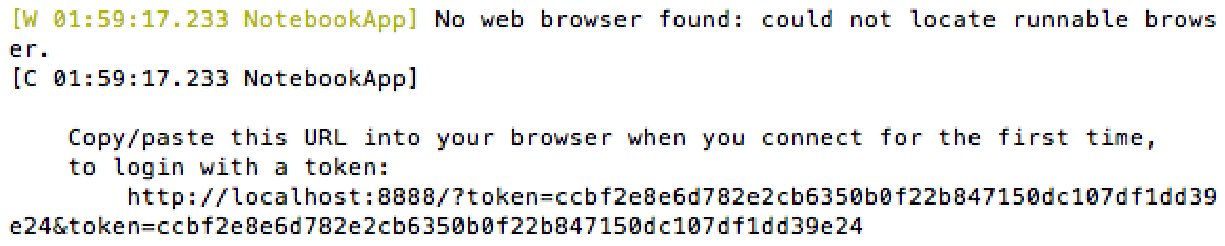
```
>>>ssh -i "test_key_pair_1.pem"
ubuntu@ec2-13-59-197-158.us-east-2.compute.amazonaws.com
```

With port forwarding:

```
>>> ssh -i "test_key_pair_1.pem" -L 8157:127.0.0.1:8888
```

ubuntu@ec2-13-59-197-158.us-east-2.compute.amazonaws.com

Now in a ssh shell with port forwarding if you launch a jupyter notebook and are give the URL and token



```
[W 01:59:17.233 NotebookApp] No web browser found: could not locate runnable brows
er.
[C 01:59:17.233 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=ccbf2e8e6d782e2cb6350b0f22b847150dc107df1dd39
e24&token=ccbf2e8e6d782e2cb6350b0f22b847150dc107df1dd39e24
```

You can change 8888 to 8157 and use the URL link on your local machine to connect to the AWS notebook.

Thus,
http://localhost:8888/?token=ccbf2e8e6d782e2cb6350b0f22b847150dc107df1dd39e24
changes to
http://localhost:8157/?token=ccbf2e8e6d782e2cb6350b0f22b847150dc107df1dd39e24

The later will allow you to connect through your browser to the notebook hosted on AWS!

However, we still have the problem that if you lose the ssh connection, this notebook will shut down and lose any work that wasn't saved to disk.

Tmux

tmux is a multiplexer that is already loaded onto EC2 instances. It allows for the EC2 instance (or any computer using it) to host multiple command line (terminal) sessions at once and maintain them even when connection to the command line is severed.

\>>>tmux
will create a terminal session hosted on the EC2 instance that will persist even after your ssh connection is lost.

For example:
\>>>ssh ... (with port forwarding)
\>>>tmux (launches terminal session on EC2)
\>>>jupyter notebook (launches jupyter notebook inside the tmux session)
Connect to notebook using port forwarding
….
Get code running
….
>>> control+b[4] (prompts tmux terminal session to accept a non-terminal input)

[4] These commands can be found at https://gist.github.com/MohamedAlaa/2961058.

```
>>>d (non-terminal input short for detach. Detaches from tmux)
>>>exit (ends ssh connection)
```

After this sequence your computer will no longer be connected and you will not have access to the notebooks, but they will be running as the terminal session hosted in tmux is still running. You can even check the EC2 Dashboard form before to see CPU usage.

You can then reconnect

```
>>>ssh …
>>>tmux attach (you will be brought into the terminal session launched previously)
```

You will again be able to connect using your browser to the notebook.

NOTE

Your dedicated EC2 Instance will maintain any files on it after you disconnect from an SSH connection.
Your EC2 instance will not maintain any code or commands running from a terminal window that has been terminated.
Your EC2 instance will maintain terminal sessions that are run through tmux even after you disconnect from an SSH connection.
Your EC2 instance will not maintain any code/data/information after you terminate the instance from the EC2 Dashboard.

END NOTE

Copy over data

Since you will most likely have data, code, or directories that are on your local machine that you will need on the EC2 instance to run your models you will need to move the data.

I recommend using FileZilla as it lets you see the directories and files in a non-terminal way and easily facilities downloading and uploading files.
https://filezilla-project.org/

Once downloaded you can go to "File" > "Site Manager" > "New Site"

Protocol: SFTP – SSH File Transfer Protocol
Host: ec2-##-##-###-###.us-east-2.compute.amazonaws.com
    (Find these numbers from EC2 Dashboard > "Connect"
Logon Type: Key File
User: ubuntu
Key File: (Permission File you used to launch EC2 Instance)

Click connect!
Your local files should be on the left and the EC2 instance should be on the right.

Tmux has a lot of awesome features

Right click on any file or directory or upload/download them.

NOTE
It is better to transfer a large zip file then many smaller files
Files can be unzip in the terminal window with
```
>>> unzip “file_name”
```
or
```
>>> unzip *.zip
```
END NOTE

BASH Scripts

All if not most of these operations can be combined into a BASH script to automate the launch process. AWS has an API specifically for this purpose. However, this material is for understanding the process, not for automating it. If you want to automate it or create a BASH script resource can be found online.