Introduction to AWS, Part 3

AWS EC2

EC2 = “Elastic Compute Cloud”
EC2 allows for scalable computing capacity in the AWS cloud.

Allows you to launch virtual computing environments, called *instances*, which are either configurable to your needs or you can choose from a set of preconfigured templates.

In these slides, we’ll walk through a workflow to connect to an ec2 instance, train a model, and then retrieve the results back to our local machine.

ssh

To interact with an EC2 instance, you will need to use ssh.

ssh = “Secure Shell”

A network protocol for operating network services securely over an unsecured network.

Authentication is done either through password authentication or through a public-private keypair. We will use the second option to connect to our EC2 instances.

AWS EC2

Before we can connect to our ec2 instance, we will likely need to modify your key to make it not publicly viewable.

This can be done by navigating to the folder containing the file and running

```
$ chmod 400 "<.pem file>"
```

For example, for me, I would run

```
$ chmod 400 ec2.pem
```

AWS EC2

You will have been given a .pem file containing your assigned key pair. These are your credentials allowing you to access your assigned EC2 instance.

To ssh into your assigned instance, you need to run the following command from the folder containing your .pem file:

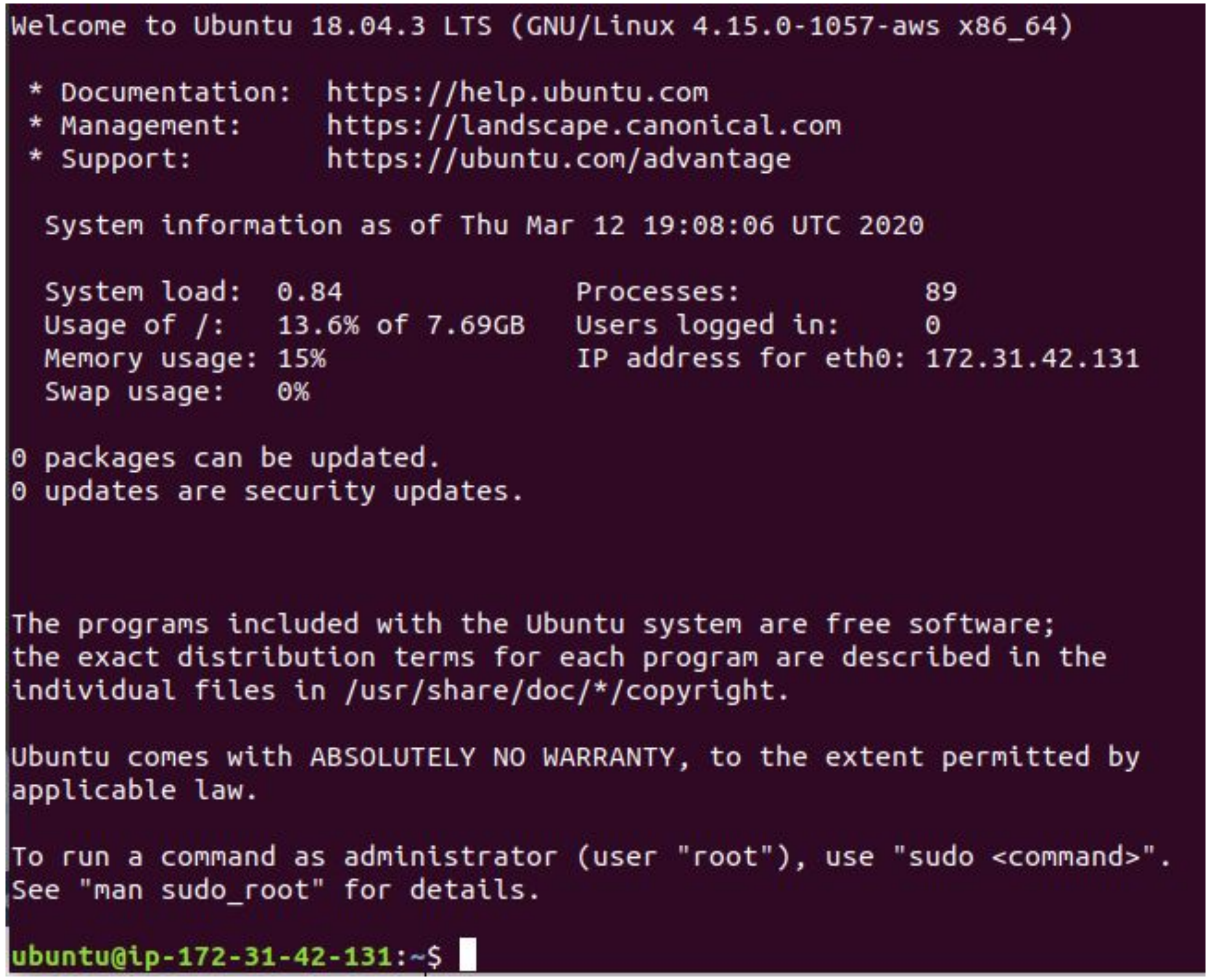
```
$ ssh -i "<.pem file>"  <DNS of EC2 instance>
```

For example, for me, I would run

```
$ ssh -i "ec2.pem"  ubuntu@ec2-54-92-131-243.compute-1.amazonaws.com
```

AWS EC2

If all goes well, you should see something like this:



```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1057-aws x86_64)

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

  System information as of Thu Mar 12 19:08:06 UTC 2020

  System load:  0.84               Processes:           89
  Usage of /:   13.6% of 7.69GB   Users logged in:     0
  Memory usage: 15%                IP address for eth0: 172.31.42.131
  Swap usage:   0%

0 packages can be updated.
0 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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-42-131:~$ 
```

AWS EC2

Now that we are successfully connected through ssh, we can interact via the terminal with our instance.

Out of the box, the instance does not have a lot installed on it, so if we want to use, for example, Python or the Anaconda libraries, we need to install them. I have shared a bash script that we will use to install what we need, but first, we need a way to get the files we are going to work with over to our instance.

scp

scp = “Secure file Copy”

A program for copying files between computers which uses the ssh protocol.

We can either copy files to a remote host using:

```
$ scp file host:path
```

Or copy from a remote host using:

```
$ scp host:file path
```

scp

To transfer a whole folder, we'll need to use the -r (for recursive) flag.
Open a new terminal window and navigate to the folder containing the folder you wish to transfer and use:

```
$ scp -i <.pem file> -r <folder to transfer> <host:path>
```

For me, this looks like:

```
$ scp -i ec2.pem -r ec2_example ubuntu@ec2-54-92-131-243.compute-1.amazonaws.com:~/
```

AWS S3

Within the folder that we transferred is a bash script which downloads and installs anaconda.

cd into the folder you just transferred and run

```
$ bash setup_python.sh
```

To test that it worked, run

```
$ python
```

Make sure that it says Anaconda, Inc.

AWS S3

Now that we have Anaconda installed, we can run our model training script.

Note that we are using a Python script here so that we can work exclusively from within the terminal. It is possible to use Jupyter Notebooks on an ec2 instance, but requires some additional setup. For more details, see this blog post: https://chrisalbon.com/aws/basics/run_project_jupyter_on_amazon_ec2/

AWS S3

To build our model, run

```
$ python training_script.py
```

This script will read in the data, train a model, and output a serialized version, *model.joblib*

AWS S3

Now, we need to retrieve our model back to our local machine. Again, scp comes to the rescue.

```
$ scp -i ec2.pem
ubuntu@ec2-54-92-131-243.compute-1.amazonaws.com:~/ec2_example/model.joblib .
```

Warning: make sure that you include the dot at the end, indicating that we want to transfer the file to our current location on the local machine.

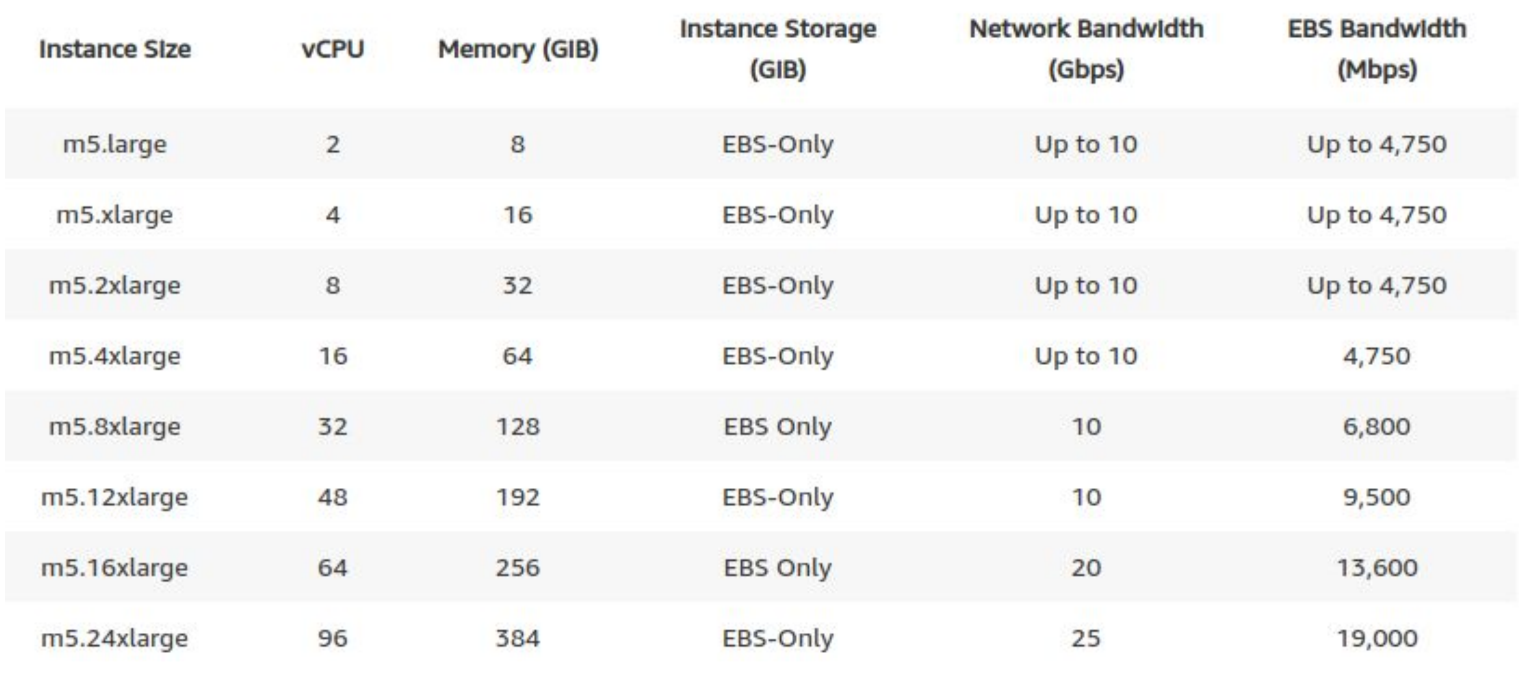
AWS S3

On our local machine, we can deserialize and use our model

```
>>> from sklearn.externals import joblib
>>> model = joblib.load('model.joblib')
```

AWS S3

Note that this was just a toy example using a very small capacity instance, but ec2 offers lots of more powerful instances, or even instances with GPUs if you are training neural network models.



Instance Size	vCPU	Memory (GIB)	Instance Storage (GIB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
m5.large	2	8	EBS-Only	Up to 10	Up to 4,750
m5.xlarge	4	16	EBS-Only	Up to 10	Up to 4,750
m5.2xlarge	8	32	EBS-Only	Up to 10	Up to 4,750
m5.4xlarge	16	64	EBS-Only	Up to 10	4,750
m5.8xlarge	32	128	EBS Only	10	6,800
m5.12xlarge	48	192	EBS-Only	10	9,500
m5.16xlarge	64	256	EBS Only	20	13,600
m5.24xlarge	96	384	EBS-Only	25	19,000