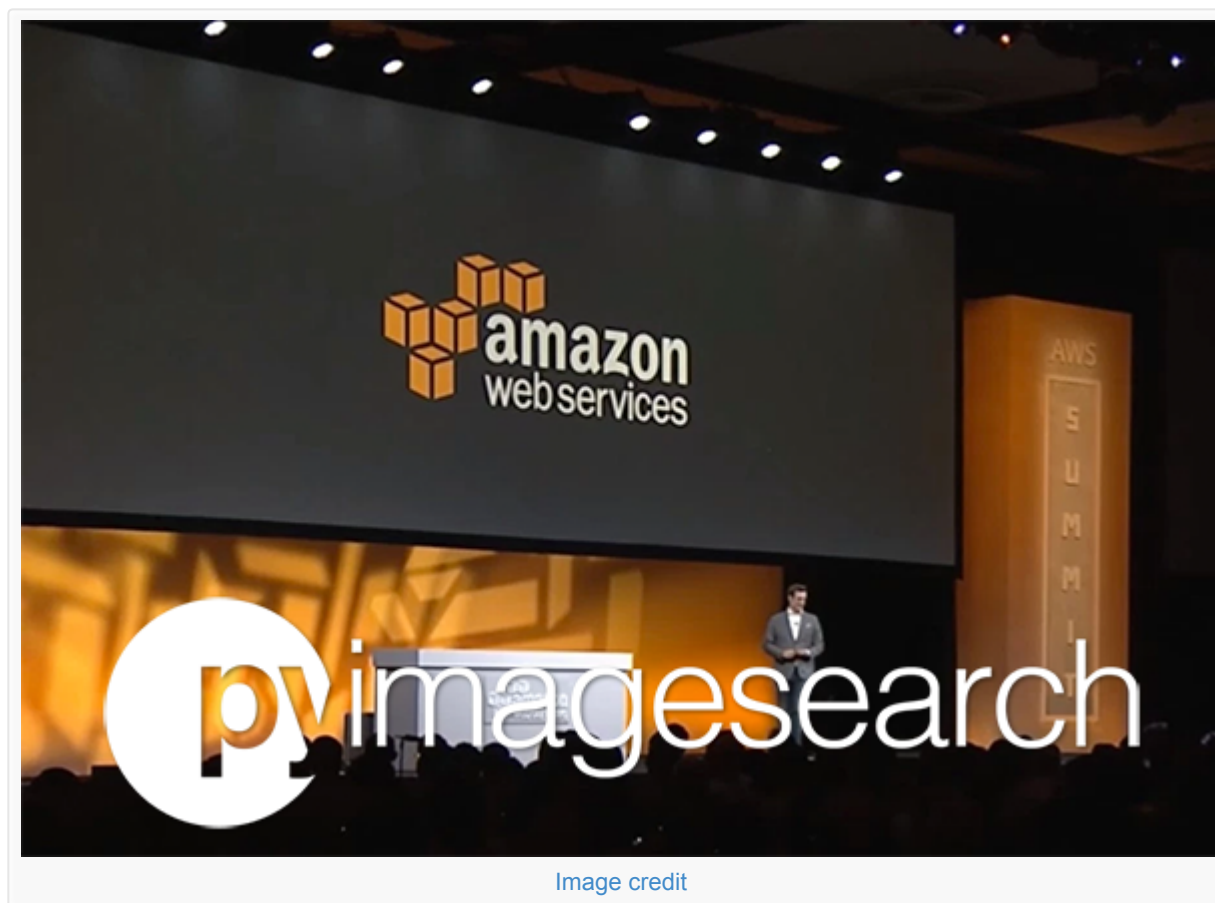


Pre-configured Amazon AWS deep learning AMI with Python

by Adrian Rosebrock on September 20, 2017 in **Deep Learning, DL4CV**



[Image credit](#)

The Ubuntu VirtualBox virtual machine that comes with my book, [Deep Learning for Computer Vision with Python](#), includes all the necessary deep learning and computer vision libraries you need (such as Keras, TensorFlow, scikit-learn, scikit-image, OpenCV, etc.) **pre-installed**.

However, while the deep learning virtual machine is easy to use, it also has a number of drawbacks, including:

- Being significantly slower than executing instructions on your native machine.
- Unable to access your GPU (and other peripherals attached to your host).

What the virtual machine has in *convenience* you end up paying for in *performance* — this makes it a great for readers who are getting their feet wet, but if you want to be able to **dramatically boost speed** while **still maintaining the pre-configured environment**, you should consider using Amazon Web Services (AWS) and my pre-built deep learning Amazon Machine Image (AMI).

Using the steps outlined in this tutorial you'll learn how to login (or create) your AWS account, spin up a new instance (*with* or *without* a GPU), and install my pre-configured deep learning image. This will enable you to enjoy the pre-built deep learning environment *without* sacrificing speed.

To learn how to use my deep learning AMI, *just keep reading*.

Pre-configured Amazon AWS deep learning AMI with Python

In this tutorial I will show you how to:

1. Login/create your AWS account.
2. Launch my pre-configured deep learning AMI.
3. Login to the server and execute your code.
4. Stop the machine when you are done.

However, before we get too far I want to mention that:

- The deep learning AMI is **Linux-based** so I would recommend having some basic knowledge of Unix environments, *especially* the command line.
- **AWS is not free and costs an hourly rate.** Exactly how much the hourly rate depends is on which machine you choose to spin up (no GPU, one GPU, eight GPUs, etc.). For less than \$1/hour you can use a machine with a GPU which will dramatically speedup the training of deep neural networks. You pay for only the time the machine is running. You can then shut down your machine when you are done.

Step #1: Setup Amazon Web Services (AWS) account

In order to launch my pre-configured deep learning you first need an Amazon Web Services account.

To start, head to the [Amazon Web Services homepage](#) and click the “Sign In to the Console” link:

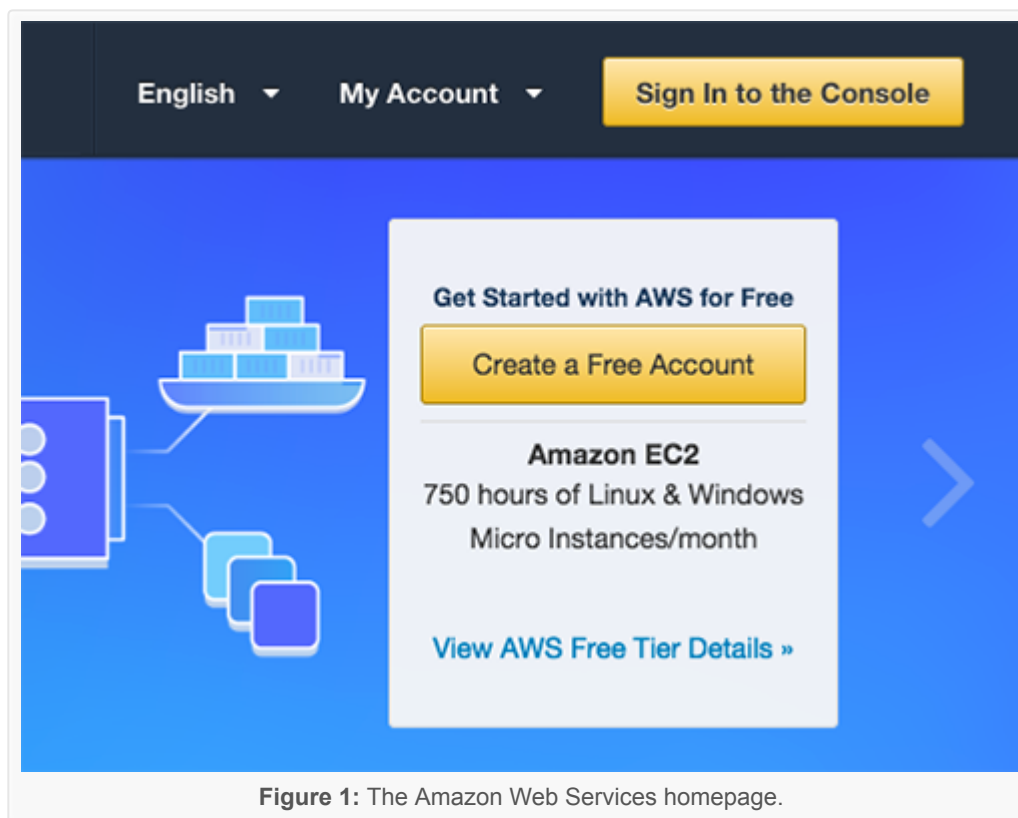
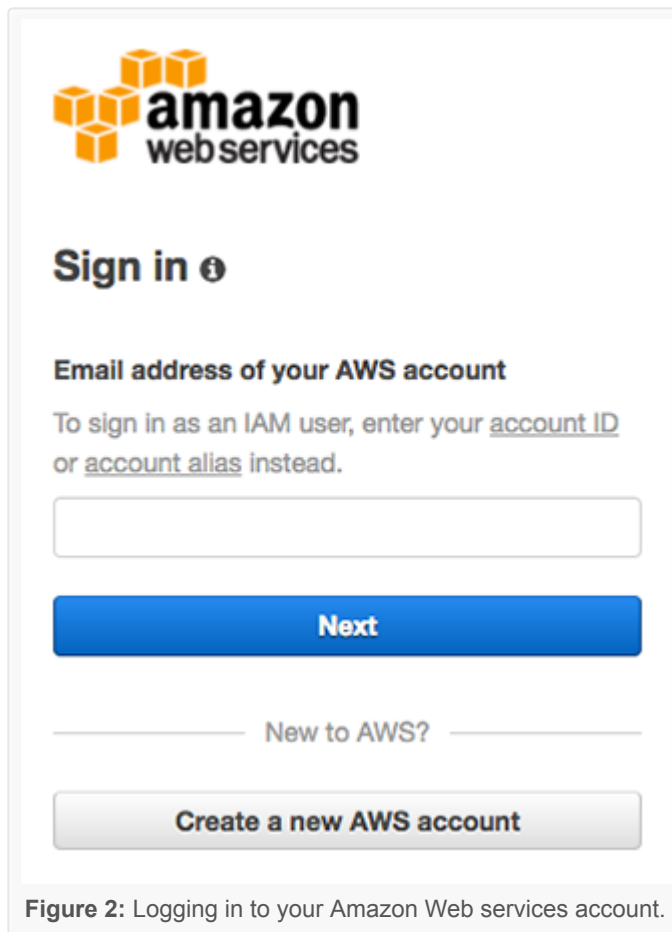


Figure 1: The Amazon Web Services homepage.

If you already have an account you can login using your email address and password. Otherwise you will need to click the “*Create a new AWS account*” button and create your account:

The image shows the AWS login interface. At the top is the Amazon Web Services logo. Below it is the heading "Sign in" with a help icon. The main section is titled "Email address of your AWS account" and includes a sub-instruction: "To sign in as an IAM user, enter your account ID or account alias instead." There is a text input field below this. A blue "Next" button is positioned below the input field. At the bottom, there is a link "New to AWS?" and a button labeled "Create a new AWS account".

amazon
web services

Sign in ⓘ

Email address of your AWS account

To sign in as an IAM user, enter your account ID or account alias instead.

Next

————— New to AWS? —————

Create a new AWS account

Figure 2: Logging in to your Amazon Web services account.

I would encourage you to use an *existing* Amazon.com login as this will expedite the process.

Step #2: Select and launch your deep learning AWS instance

You are now ready to launch your pre-configured deep learning AWS instance.

First, you should set your region/zone to “*US West (Oregon)*”. I created the deep learning AMI in the Oregon region so you’ll need to be in this region to find it, launch it, and access it:

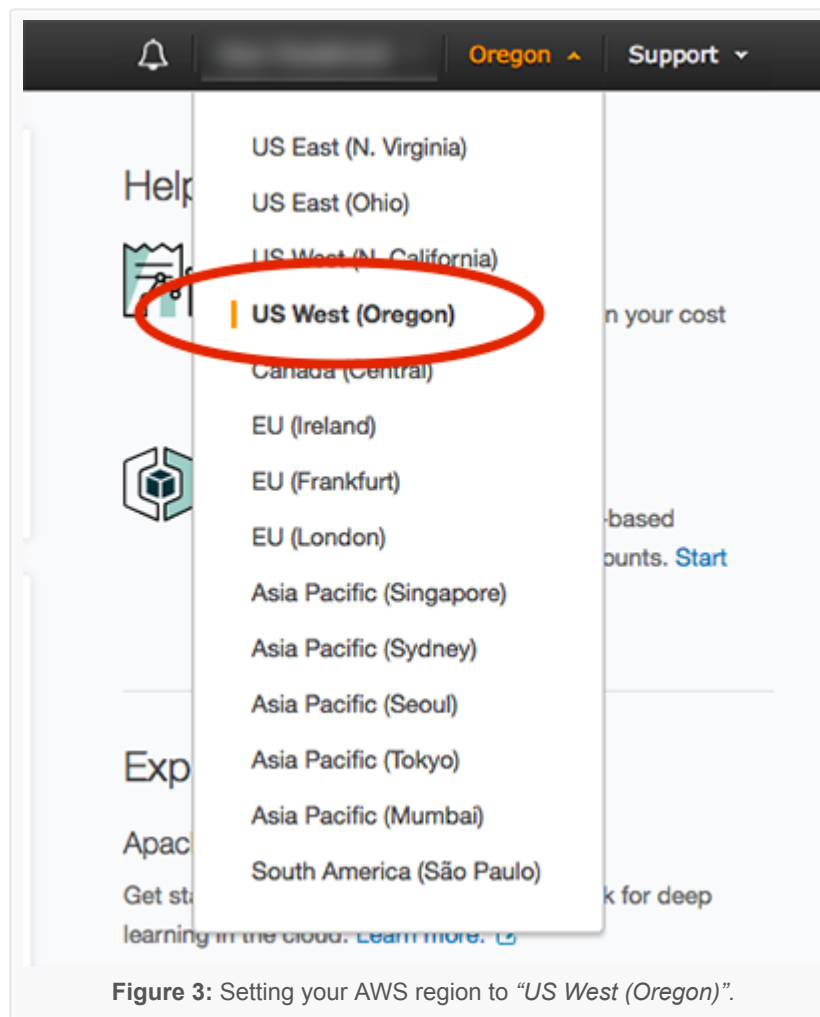


Figure 3: Setting your AWS region to “US West (Oregon)”.

After you have set your region to Oregon, click the “Services” tab and then select “EC2” (Elastic Cloud Compute):

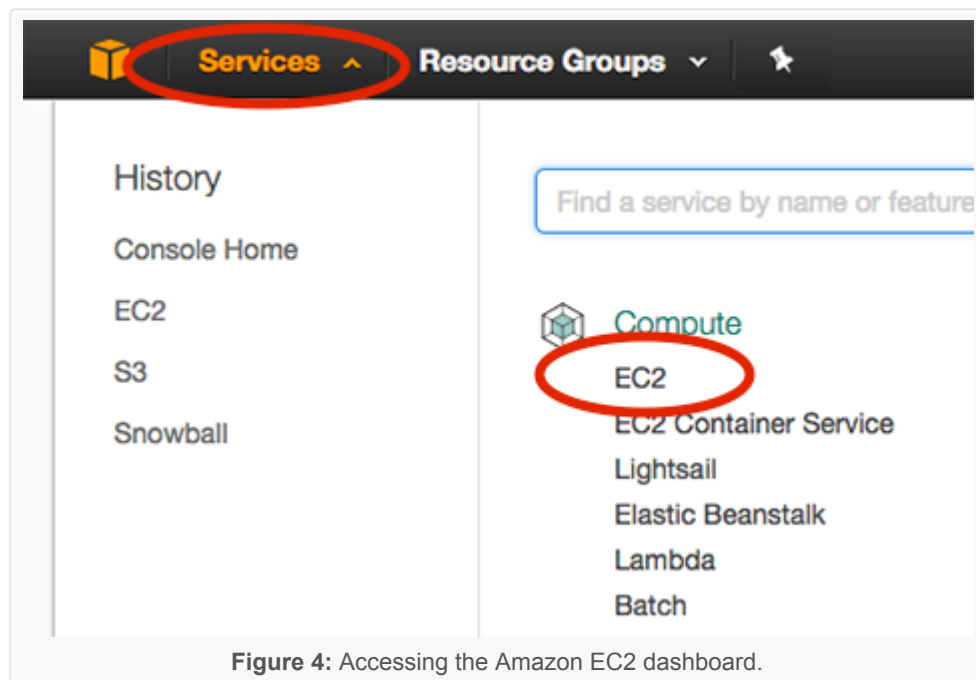


Figure 4: Accessing the Amazon EC2 dashboard.

From there you should click the “Launch Instance” button:

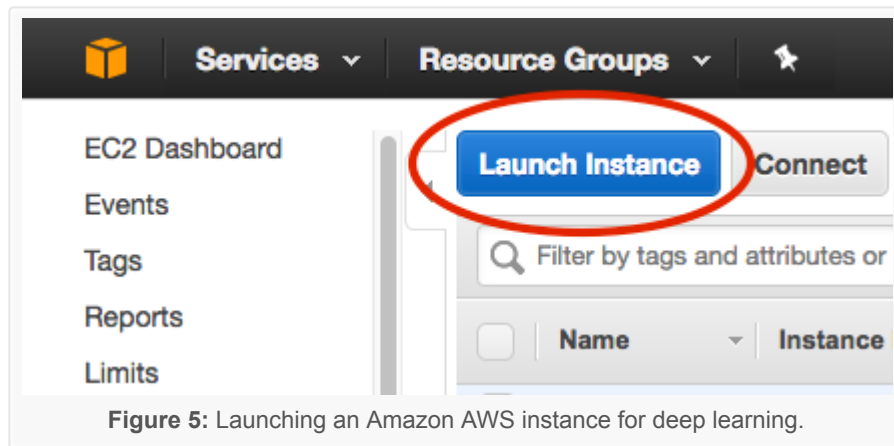


Figure 5: Launching an Amazon AWS instance for deep learning.

Then select the “Community AMIs” and search for either “*deep-learning-for-computer-vision-with-python*” or “*ami-ccba4ab4*”:

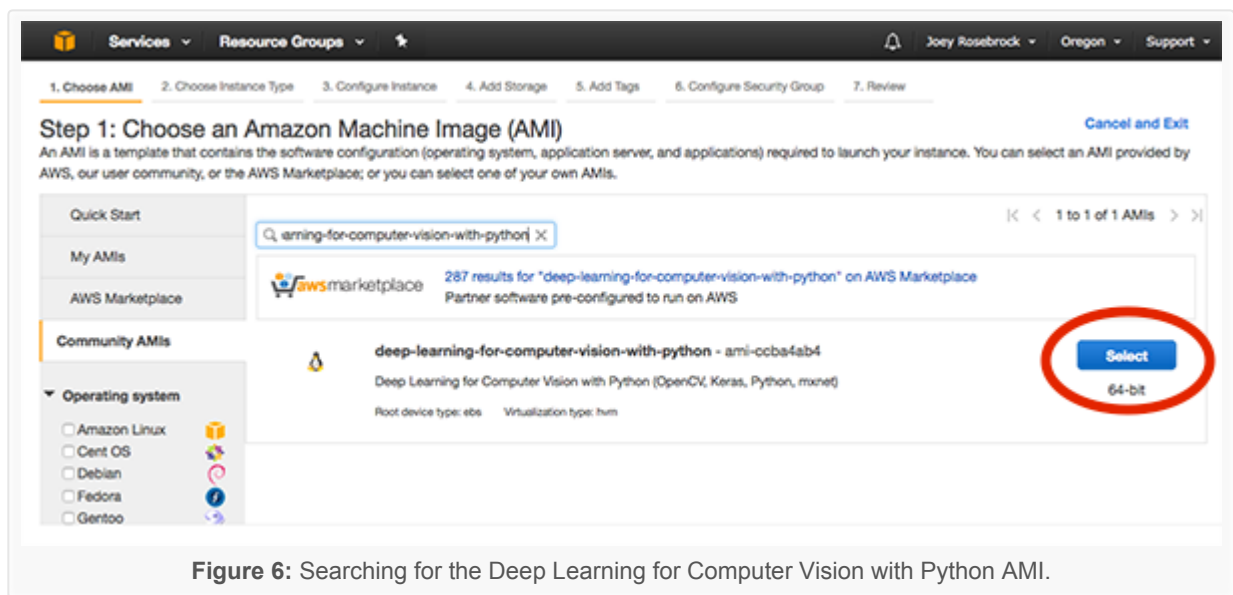


Figure 6: Searching for the Deep Learning for Computer Vision with Python AMI.

Click “Select” next to the AMI.

You are now ready to select your instance type. Amazon provides a *hugenum*ber of virtual servers that are designed to run a wide array of applications. These instances have varying amount of CPU power, storage, network capacity, or GPUs, so you should consider:

1. What type of machine you would like to launch.
2. Your particular budget.

GPU instances tend to cost *much more* than standard CPU instances. However, they can train deep neural networks in a fraction of the time. When you average out the amount of time it takes to train a network on a CPU versus on a GPU you may realize that using the GPU instance will save you money.

For CPU instances I recommend you use the “*Compute optimized*” **c4.***instances. In particular, the **c4.xlarge** instance is a good option to get your feet wet.

If you would like to use a GPU, I would highly recommend the “*GPU compute*” instances. The **p2.xlarge** instance has a single NVIDIA K80 (12GB of memory).

The **p2.8xlarge** sports 8 *GPUs*. While the **p2.16xlarge** has 16 *GPUs*.

I have included the pricing (at the time of this writing) for each of the instances below:

- **c4.xlarge**: \$0.199/hour
- **p2.xlarge**: \$0.90/hour
- **p2.8xlarge**: \$7.20/hour
- **p2.16xlarge**: \$14.40/hour

As you can see, the GPU instances are much more expensive; however, you are able to train networks in a *fraction* of the cost, making them a more economically viable option. Because of this I recommend using the **p2.xlarge** instance if this is your first time using a GPU for deep learning.

In the example screenshot below you can see that I have chosen the p2.xlarge instance:

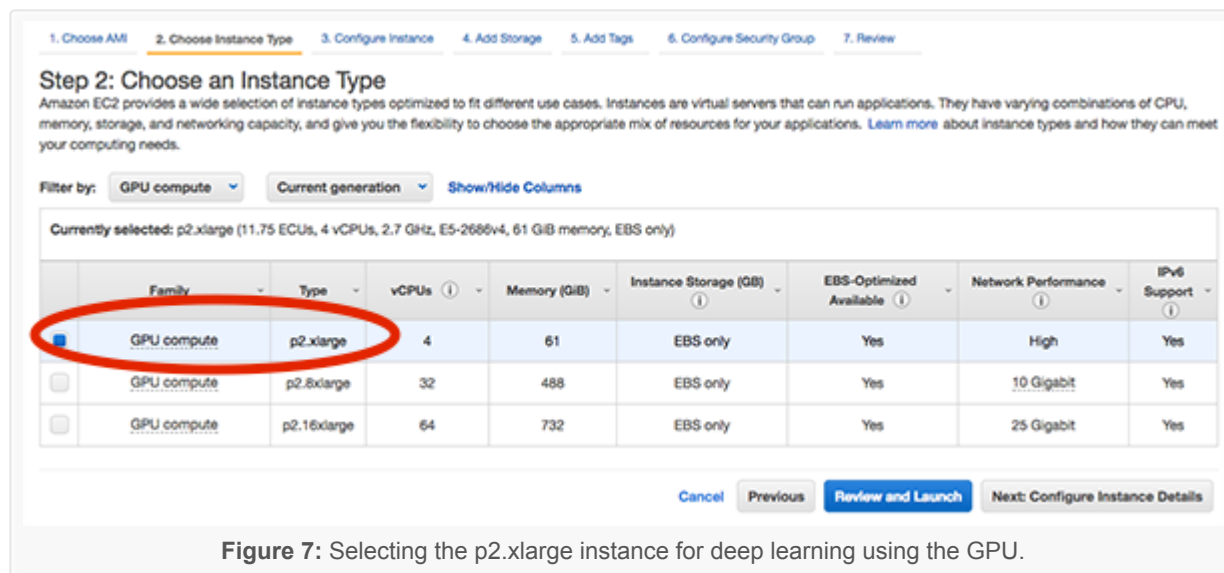
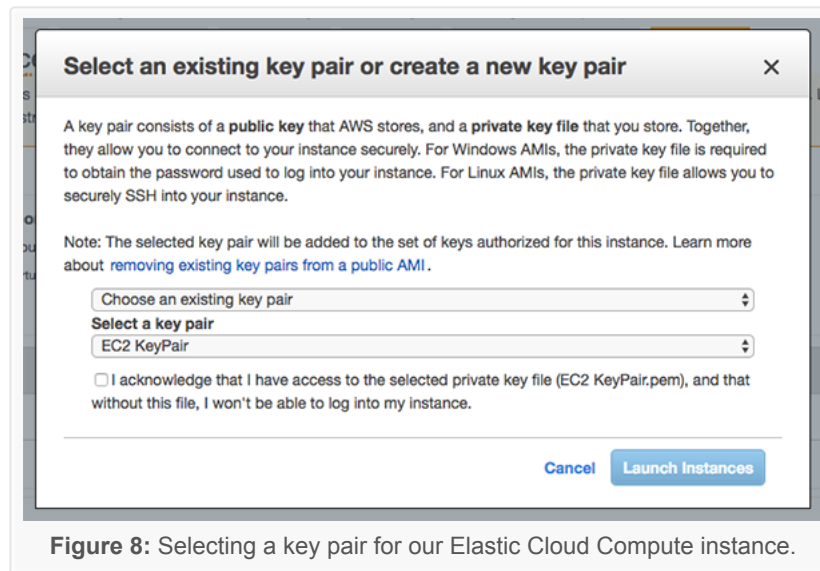


Figure 7: Selecting the p2.xlarge instance for deep learning using the GPU.

Next, I can click “*Review and Launch*” followed by “*Launch*” to boot my instance.

After clicking “*Launch*” you’ll be prompted to select your key pair or create a new key pair:

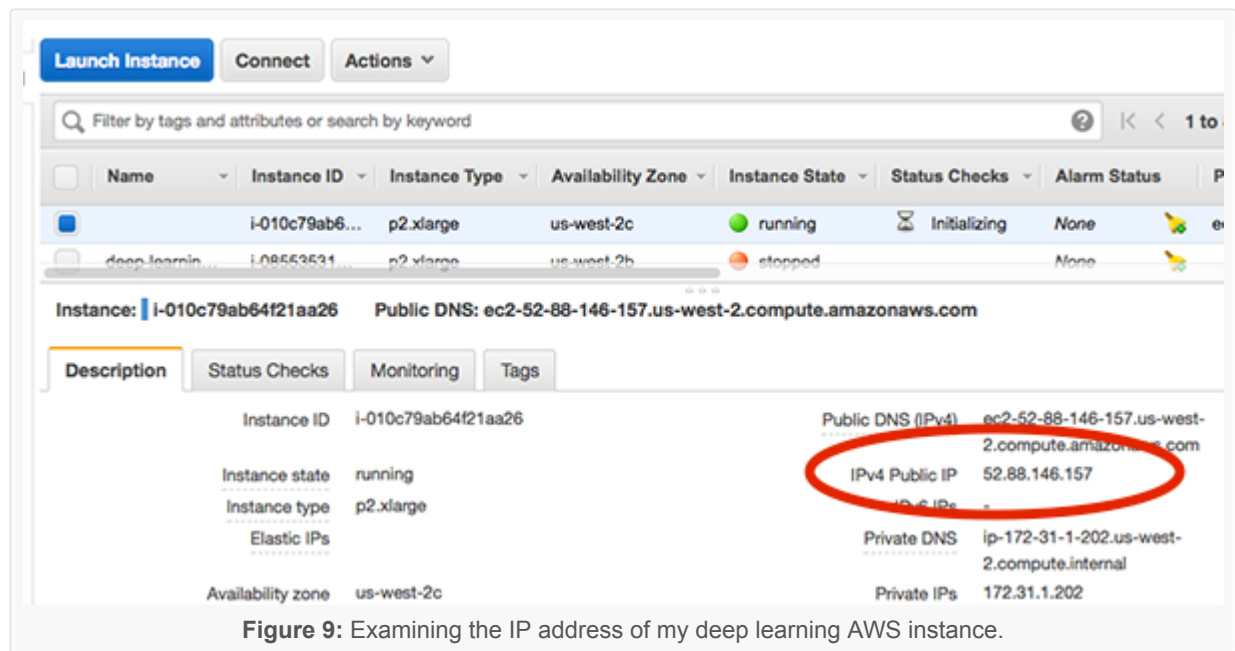


If you have an existing key pair you can select “*Choose an existing key pair*” from the drop down. Otherwise you’ll need to select the “*Create a new key pair*” and then download the pair. The key pair is used to login to your AWS instance.

After acknowledging and accepting login note from Amazon your instance will start to boot. Scroll down to the bottom of the page and click “*View Instances*”. It will take a minute or so for your instance to boot.

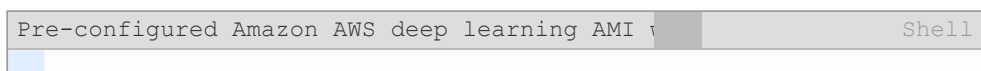
Once the instance is online you’ll see the “*Instance State*” column be changed to “*running*” for the instance.

Select it and you’ll be able to view information on the instance, including the IP address:



Here you can see that my IP address is `52.88.146.157`. **Your IP address will be different.**

Fire up a terminal and you can SSH into your AWS instance:




```
1 $ ssh -i EC2KeyPair.pem ubuntu@52.88.146.157
```

You'll want to update the command above to:

1. Use the filename you created for the key pair.
2. Use the IP address of your instance.

Step #3: (GPU only) Re-install NVIDIA deep learning driver

If you selected a GPU instance you will need to:

1. Reboot your AMI via the command line
2. Reinstall the NVIDIA driver

The reason for these two steps is because instances launched from a *pre-configured* AMI can potentially restart with a slightly different kernel, therefore causing the Nouveau (default) driver to be loaded instead of the NVIDIA driver.

To avoid this situation you can either:

1. Reboot your system *now*, essentially “locking in” the current kernel and then reinstalling the NVIDIA driver *once*.
2. Reinstall the NVIDIA driver *each time* you launch/reboot your instance from the AWS admin.

Both methods have their pros and cons, but I would recommend the first one.

To start, reboot your instance via the command line:

```
Pre-configured Amazon AWS deep learning AMI v Shell
1 $ sudo reboot
```

Your SSH connection will terminate during the reboot process.

Once the instance has rebooted, re-SSH into the instance, and reinstall the NVIDIA kernel drivers. Luckily this is easy as I have included the driver file in the home directory of the instance.

If you list the contents of the `installers` directory you'll see three files:

```
Pre-configured Amazon AWS deep learning A Shell
1 $ ls -l installers/
2 total 1435300
3 -rwxr-xr-x 1 root root 1292835953 Sep  6 14:03 cuda-linux64-rel-8
4 -rwxr-xr-x 1 root root 101033340 Sep  6 14:03 cuda-samples-linux
5 -rwxr-xr-x 1 root root 75869960 Sep  6 14:03 NVIDIA-Linux-x86_64
```

Change directory into `installers` and then execute the following command:

```
Pre-configured Amazon AWS deep learning AMI v Shell
1 $ cd installers
2 $ sudo ./NVIDIA-Linux-x86_64-375.26.run --silent
```


Follow the prompts on screen (including overwriting any existing NVIDIA driver files) and your NVIDIA deep learning driver will be installed.

You can validate the NVIDIA driver installed successfully by running the `nvidia-smi` command:

Pre-configured Amazon AWS deep learning AMI v										Shell
1	\$ nvidia-smi									
2	Wed Sep 13 12:51:43 2017									
3	+-----+-----+									
4	NVIDIA-SMI 375.26					Driver Version: 375.26				
5	+-----+-----+									
6	GPU Name		Persistence-M		Bus-Id		Disp.A		Volatil	
7	Fan Temp Perf		Pwr:Usage/Cap				Memory-Usage		GPU-Uti	
8	+=====+=====+									
9	0 Tesla K80		Off		0000:00:1E.0		Off			
10	N/A 43C P0		59W / 149W		0MiB / 11439MiB				97%	
11	+-----+-----+									
12										
13	+-----+-----+									
14	Processes:									
15	GPU		PID		Type		Process name			
16	+=====+=====+									
17	No running processes found									
18	+-----+-----+									

Step #4: Access deep learning Python virtual environment on AWS

You can access our deep learning and computer vision libraries by using the `workon dl4cv` command to access the Python virtual environment:

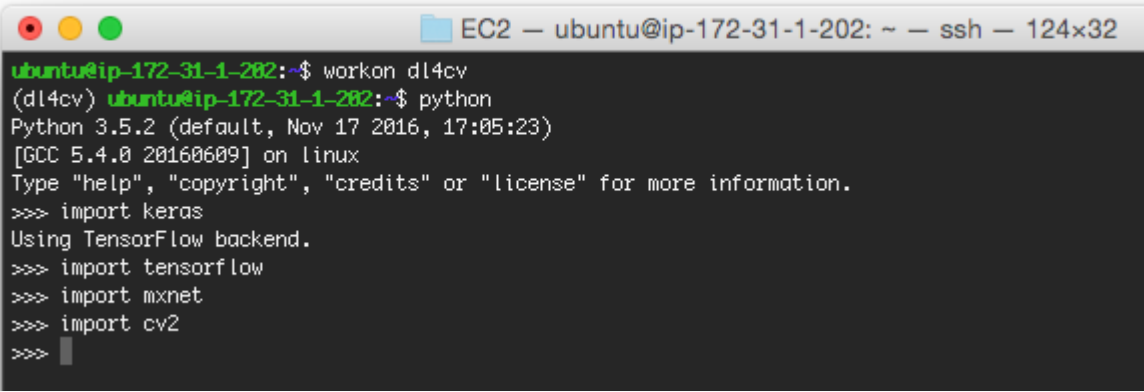


Figure 10: Accessing the dl4cv Python virtual environment for deep learning.

Notice that my prompt now has the text `(dl4cv)` preceding it, implying that I am inside the `dl4cv` Python virtual environment.

You can run `pip freeze` to see all the Python libraries installed.

I have included a screenshot below demonstrating how to import Keras, TensorFlow, mxnet, and OpenCV from a Python shell:



```

ubuntu@ip-172-31-1-202:~$ workon dl4cv
(dl4cv) ubuntu@ip-172-31-1-202:~$ python
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import keras
Using TensorFlow backend.
>>> import tensorflow
>>> import mxnet
>>> import cv2
>>>

```

Figure 11: Importing Keras, TensorFlow, mxnet, and OpenCV into our deep learning Python virtual environment.

If you run into an error importing mxnet, simply recompile it:

```

Pre-configured Amazon AWS deep learning AMI v Shell
1 $ cd ~/mxnet
2 $ make -j4 USE_OPENCV=1 USE_BLAS=openblas USE_CUDA=1 \
3   USE_CUDA_PATH=/usr/local/cuda USE_CUDNN=1

```

This due to the NVIDIA kernel driver issue I mentioned in **Step #3**. You only need to recompile mxnet *once* and *only if* you receive an error at import.

The code + datasets to [Deep Learning for Computer Vision with Python](#) are **not** included on the pre-configured AMI by default (as the AMI is publicly available and can be used for tasks *other than* reading through *Deep Learning for Computer Vision with Python*).

To upload the code from the book on your local system to the AMI I would recommend using the `scp` command:

```

Pre-configured Amazon AWS deep learning AMI v Shell
1 $ scp -i EC2KeyPair.pem ~/Desktop/sb_code.zip ubuntu@52.88.146.15

```

Here I am specifying:

- The path to the `.zip` file of the *Deep Learning for Computer Vision with Python* code + datasets.
- The IP address of my Amazon instance.

From there the `.zip` file is uploaded to my home directory.

You can then unzip the archive and execute the code:

```

Pre-configured Amazon AWS deep learning A Shell
1 $ unzip sb_code.zip
2 $ cd sb_code/chapter12-first_cnn/
3 $ workon dl4cv
4 $ python shallownet_animals.py --dataset ../datasets/animals
5 Using TensorFlow backend.
6 [INFO] loading images...
7 ...
8 Epoch 100/100

```

9	2250/2250 [=====] - 0s - loss: 0.3429 -
10	[INFO] evaluating network...
11	precision recall f1-score support
12	
13	cat 0.67 0.52 0.58 262
14	dog 0.59 0.64 0.62 249
15	panda 0.75 0.87 0.81 239
16	
17	avg / total 0.67 0.67 0.67 750

Step #5: Stop your deep learning AWS instance

Once you are finished working with your AMI head back to the “*Instances*” menu item on your EC2 dashboard and select your instance.

With your instance selected click “*Actions => Instance State => Stop*”:

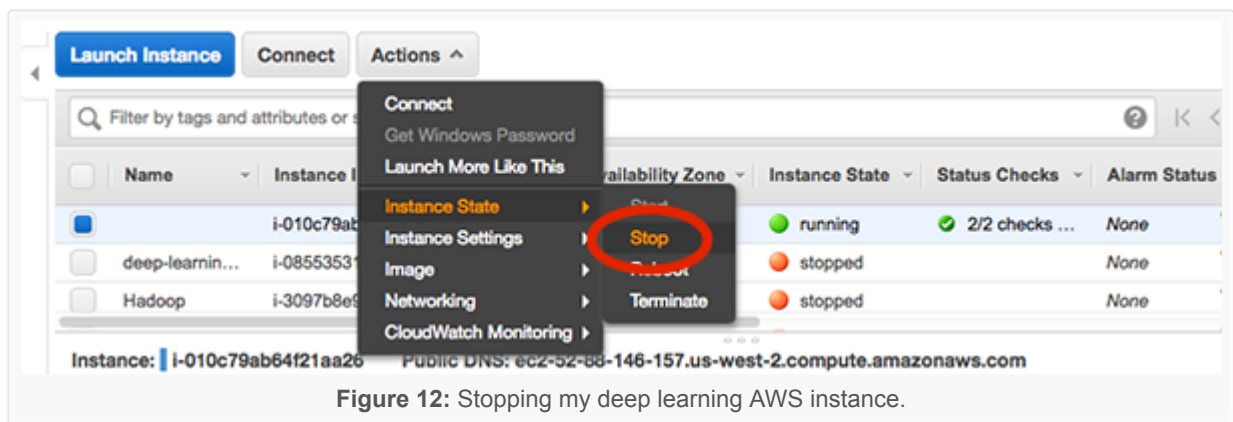


Figure 12: Stopping my deep learning AWS instance.

This process will shutdown your deep learning instance (and you will no longer be billed hourly for it).

If you wanted to instead delete the instance you would select “*Terminate*”.

Troubleshooting and FAQ

In this section I detail answers to frequently asked questions and problems regarding the pre-configured deep learning AMI.

How do I execute code from *Deep Learning for Computer Vision with Python* from the deep learning AMI?

Please see the “*Access deep learning Python virtual environment on AWS*” section above. The gist is that you will upload a `.zip` of the code to your AMI via the `scp` command. An example command can be seen below:

```
Pre-configured Amazon AWS deep learning AMI v Shell
1 $ scp -i EC2KeyPair.pem path/to/code.zip ubuntu@your_aws_ip_addre
```

Can I use a GUI/window manager with my deep learning AMI?

No, the AMI is terminal only. I would suggest using the deep learning AMI if you are: