Creating and Using a Jupyter Instance on AWS

Authors:

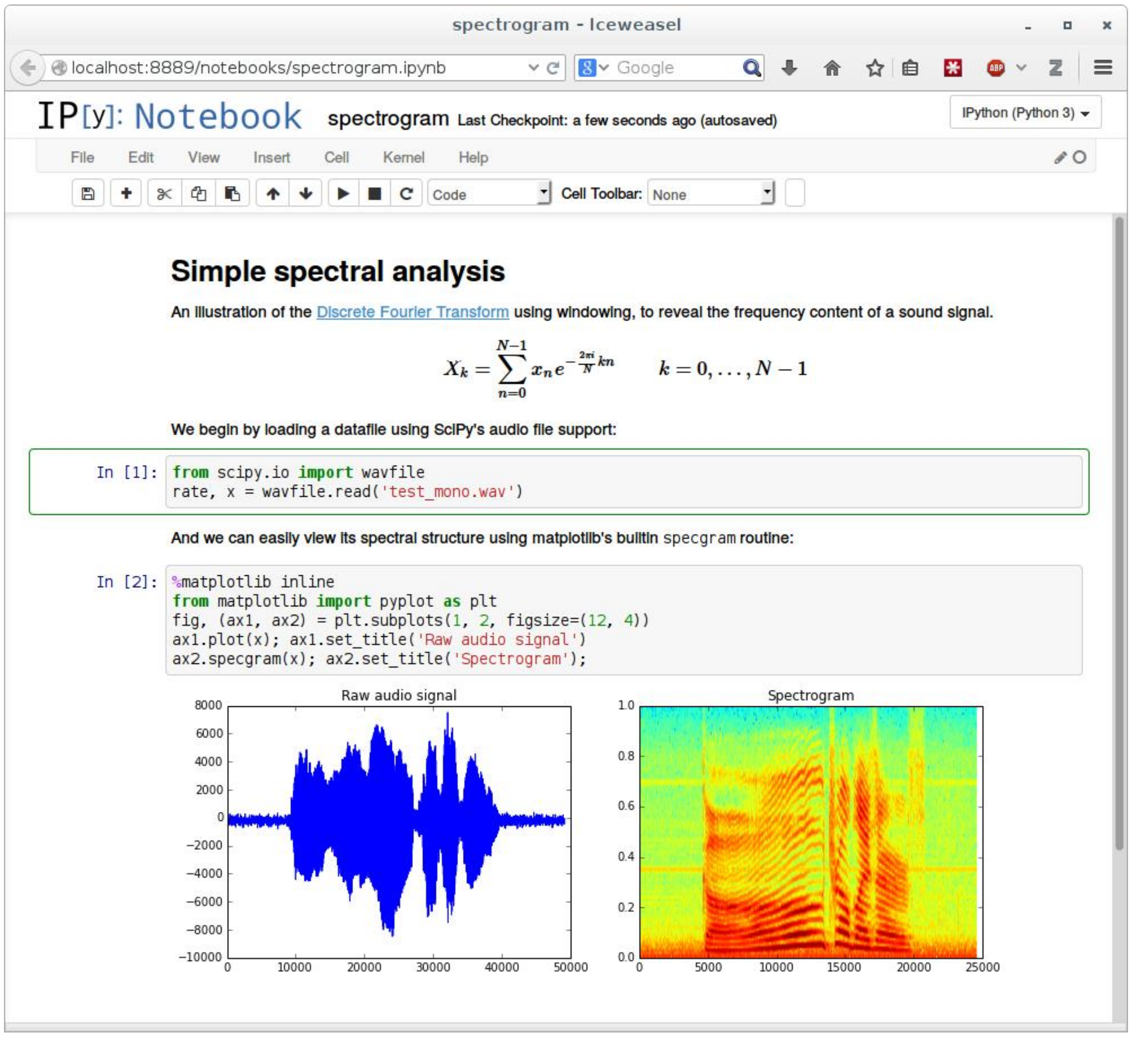
Jeff Layton, AWS Research and Technical Computing Team

Adrian White, AWS Research and Technical Computing Team

Jupyter Instance

For the scientific researcher, engineer, or technical user, being able to quickly start up a server instance for running applications, writing code, or even post-process data is one of the great things about Amazon Web Services (AWS). One of the most common tools used for developing and maintaining applications is Jupyter (http://jupyter.org/). Jupyter allows interactive data science and scientific computing across 40 different programming languages. It allows researchers to share/exchange live code, data sets, and visualization so that they can collaborate more efficiently. These are called notebooks, and their use is growing.

Below is a screenshot from a github site that introduces Jupyter (which used to be called IPython).



Simple spectral analysis

An illustration of the Discrete Fourier Transform using windowing, to reveal the frequency content of a sound signal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline
        from matplotlib import pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

This illustrates what Jupyter can do. It’s a live interactive notebook that can include the use of JIT’s (Just in time compiler), and GPUs. It can also be used with a very wide variety of languages such as Python, R, and even Julia.

This document is a brief overview of creating Amazon Machine Images (AMI) that contain Jupyter along with Python, R, Julia, and Jupyter. It also explains how to take the base non-GPU (Graphics Processor Unit) AMI and add tools suitable for GPU programming including the CUDA toolkit. These AMIs run on an AWS instance but the user interacts with them using a web browser on their laptop, table, or even their cell phone.

You can think of the AMI as an OS image along with all of the tools and any data you place in there. In this document we describe the creation of two AMIs. One is for non-GPU enabled instances and the other is for GPU enabled instances. These AMIs can be created and shared with users, who can then use a simple AWS Cloud Formation template to start up an AMI on an AWS instance.

Creating the AMIs

Creating the AMIs is not a difficult task. The AMI is the system (OS) image that users can use as a basis for their “Jupyter instance”. This paper will focus on CentOS 7 and Amazon Linux options for the OS to use as a basis for a workstation. If you desire to use a different base OS you can do that but you might have to modify the instructions for the specific platform.

As a starting point, the AMI will contain the common development tools (GNU C, C++, Fortran, Python, Perl), Anaconda Python (freely available from Continuum), Jupyter, R (from Continuum as well), Julia, and a few other tools from Continuum (all freely available). These are all 64-bit applications. There are also instructions for building the AMI for GPUs. It includes the NVidia drivers and the CUDA 7.5 toolkit.

The reason we have chosen the Anaconda distribution of Python from Continuum is that it is one of the fastest and most stable Python distributions available. It also has some excellent extensions available for accelerating Python code and includes a Jupyter compatible distribution of R. Anaconda is also an up-to-date Python environment (latest version of Python and associated tools) relative to what comes standard with various Linux Distributions such as CentOS, or Ubuntu, or Red Hat Enterprise Linux (RHEL). It also means you can choose Python 2 or Python 3.

> *Note: In this document, we are installing Python 2.x. If you want Python 3, anywhere in the document where “Anaconda2” is used, change that to “Anaconda3”. Be sure to check that the versions of tools you install matches the version of Python.*

To start creating the AMI, start up an instance with at least 2 VCPU’s, 4-7 GiB of memory. Be sure to use an OS version with HVM since you will get better performance even though performance is not an issue when creating and AMI. It is recommended that you us an Amazon EBS optimized instance (faster Amazon EBS performance) and one that uses Amazon EBS for the “root” volume. This makes things a little bit easier when creating the AMI.

As a starting point, make the root Amazon EBS volume at least 20GB in size (you likely won’t need all of that space but it’s better to be safe than sorry ☺). You can chose any type of Amazon EBS volume you like but it’s probably a good idea to choose either the standard magnetic volume type or the gp2 volume type since you are just creating the AMI and IO performance isn’t critical.

Use any VPC, Security Group, and subnet you want. Just make sure the VPC allows a public IP to be assigned to the instance and that port 22 and 8080 are open to inbound traffic. To make things easy, use the **default** VPC, SG, and subnet that comes with your account. You will have to open port 8080 in the Security Group manually for inbound TCP traffic (this is for Jupyter). For the default VPC, port 22 should be open (always good to check).

Summary:

- Any instance type (gp2 if you want to add in GPU and CUDA tools).
    - 2 VPCU's (more is better).
    - 4-7 GiB memory (more is better).
    - Amazon EBS root volume – 10GB in size (standard magnetic or gp2 are good).
    - CentOS 7, Amazon Linux (the latest).
        - You can select any Linux you like but these two are the ones documented here.
        - Choose one that supports HVM (Hardware Virtualization).
- Any VPC, Security Group, Subnet.
    - Good idea to use default VPC.
    - Open ports 22 and 8080 for inbound TCP traffic.
- Public DNS for your instance so you can use ssh to connect to it. It is also needed when configuring Jupyter.

Once the instance is created and you can use ssh to connect to it using your ssh key, you are ready to start installing the tools. Appendix A contains the steps for installing everything if you used Amazon Linux. It also includes the steps for installing the NVidia drivers and CUDA & tools. Appendix B is the same thing except if you used CentOS 7.

At this time, select the distribution you want and follow the instructions on creating the AMI or you can do both distributions and make them available to people who want either Linus distro). Once the AMI is created, come back to this section.

Since you have created your own AMIs, you will need to be responsible for keeping them up to date. It is a good idea to periodically start up an instance with the AMIs, run "`sudo yum update`" and then test the functionality to make sure the OS updates have not broken anything.

Checking AMI for functionality

Once you have the software installed on your instance, it's a good idea to make sure that Jupyter is working. There are only a few things that need to be done. The first one is to make sure that port 8080 is open on the Security Group associated with the image. To do this, you go to the EC2 console and on the left hand side, click on "Security Groups". Then look for the Security Group associated with the instance. Look for a tab labeled "inbound rules" and see if there is a TCP rule covering port 8080. If there is not, create a "Custom TCP Rule" that allows TCP traffic on port 8080 from anywhere (0.0.0.0/0).

Next, log into the instance via a different window and run the follow two commands:

- `sudo su`

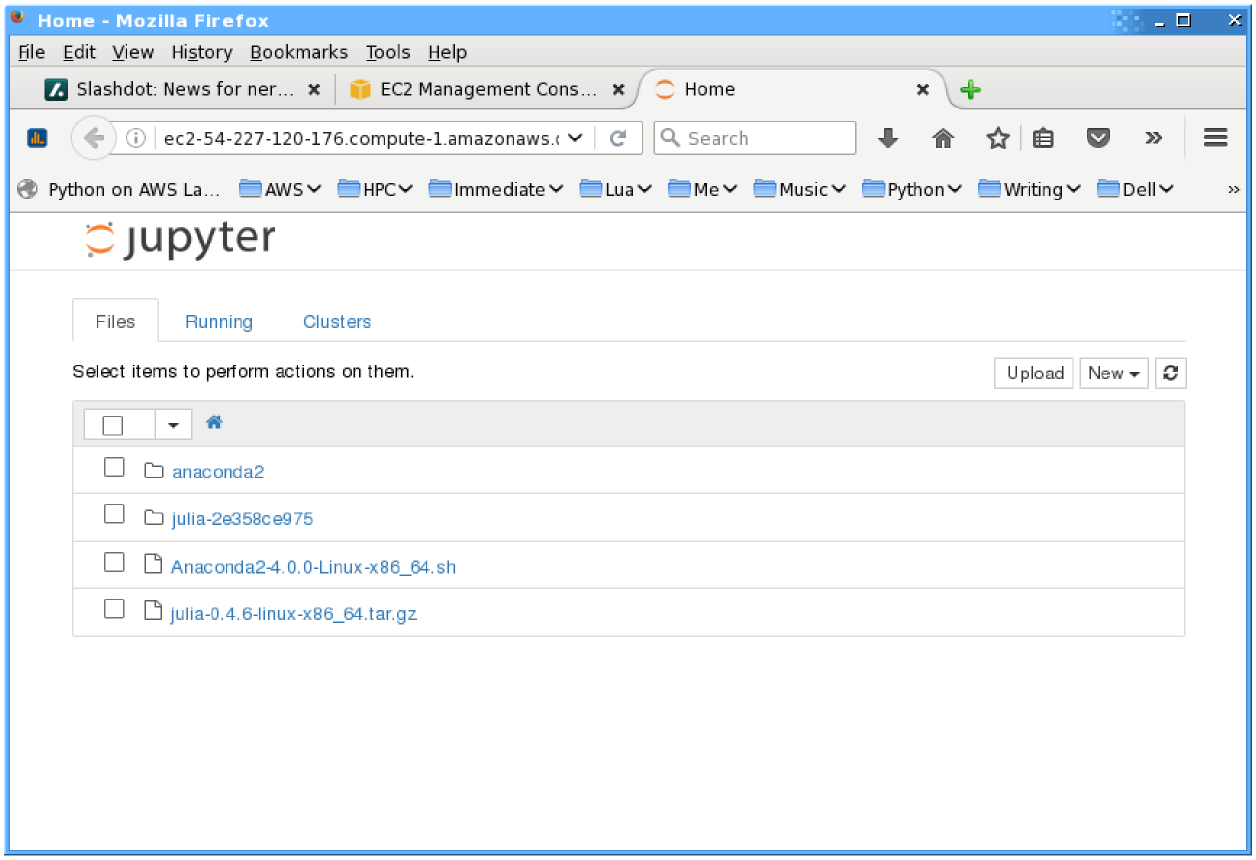
- `su - jupyter -c '~/anaconda2/bin/ipcluster nbextension enable; ~/anaconda2/bin/jupyter notebook --ip `curl http://169.254.169.254/latest/meta-data/public-hostname` --port 8080 -no-browser &'`

This runs Jupyter in the background on the instance. At the bottom, it should give you a URL to use in your browser to access Jupyter on the instance.

Now, go to your web browser and paste in the URL. For example,

- http://ec2-54-227-120-176.compute-1.amazonaws.com:8080/

This bring up a web page that looks like the following:



Select items to perform actions on them.

anaconda2
julia-2e358ce975
Anaconda2-4.0.0-Linux-x86_64.sh
julia-0.4.6-linux-x86_64.tar.gz

You will notice that there are no Jupyter notebooks available. If you have any notebooks, you can copy the to the instance using scp. Be sure you copy them to `/home/jupyter`, which is where the Jupyter, python, R, Julia, etc., were all installed.

CloudFormation Template:

You can find a sample Cloud Formation template in github: https://github.com/scicolabs/jupyter-aws.glt also contains some sample Jupyter notebooks (contained in the directory "labs").

You will likely have to modify the template to work for your AMIs and your situation. There is really only one modification you need to make and that is to the region and the specific AMIs you created. In the template, look for the following lines.

```
"AWSRegionArch2AMI" : {
   "ap-southeast-2" : {"HVM64" : "ami-58facb3b", "HVMG2" : "ami-6cfdcc0f"}
}
```

You will probably have to change the region from "ap-southeast-2" to the region where your AMIs are stored. You will also have to change the names of the specific AMIs. You can find your AMI names on the console under "EC2" and "AMI".

Security

At Amazon, security is extremely important. Remember the shared responsibility model that is used. AWS is responsible for the security from the hypervisor down, and you, the user, is responsible for the security from the OS upward (OS, applications, data, etc.).

If you examine the cloud formation template, you will see that the only two ports that are opened in the security group are 22 (ssh) and 8080 (used by Jupyter). Also, the user has the option of specifying a specific IP address range for instances that may be accessing the instance.

User instructions

Once the AMIs are configured and made available to users, all the user has to do is go to the AWS console, go to CloudFormation and upload the template. Cloud Formation will take care of starting up everything for the user. Once it's done, the user can find the IP address in the console (look under EC2/instances).

Using the public IP address the user can upload Jupyter notebooks to the instance (be sure to upload them to `/home/jupyter`). Then the user can then point their web browser to the Jupyter instance. For example, let's assume the public IP address is

http://ec2-54-227-120-176.compute-1.amazonaws.com

The URL for the web browser would be:

http://ec2-54-227-120-176.compute-1.amazonaws.com:8080/

The user is now off to the races!

Before a user terminates the instance, be sure they copy any data from the AWS instance to their laptop or tablet!!!!!

When the user is done and wants to terminate the instance, they go back to the AWS console and go to the Cloud Formation service. They select the cloud formation instance they want to terminate and then from the drop down terminate the instance. However, be careful – when the instance is terminated, the volume will be erased so save your work before you terminate the instance!!!

Appendix A - Amazon Linux

Amazon Linux is developed and supported by AWS and is only available on AWS. It usually has a later kernel than CentOS or Red Hat Enterprise Linux. Let's start with the basic non-GPU installation which is also the basis for the GPU version.

- Update the OS:
  - `sudo yum update -y`
- Install the Development Tools:
  - `sudo yum groupinstall 'Development Tools' -y`
- Create "jupyter" user
  - `sudo useradd jupyter`
- Login as "jupyter" user
  - `sudo su`
  - `su - jupyter`
  - Install Anaconda Python as the "jupyter" user. This is a manual process. Note this command install Python 2.x. If you want Python 3. Change the name to "Anaconda3-4.0.0-Linux-x86_64.sh".
  - `wget` [`http://repo.continuum.io/archive/Anaconda2-4.0.0-Linux-x86_64.sh`](http://repo.continuum.io/archive/Anaconda2-4.0.0-Linux-x86_64.sh)
  - Manual install as "jupyter" user (installs into home directory)
    - `bash Anaconda2-4.0.0-Linux-x86_64.sh`
      - Let the installer modify the .bashrc file.
- "source" your .bashrc file (only for testing – you don't need to do this when you log in)
  - `source ~/.bashrc`
- Use "conda", the Anaconda install tool, to install Jupyter and the dependencies
  - `conda install ipyparallel notebook jupyter -y`
- To install R support for Jupyter (as the Jupyter user)
  - `conda install -c r r-essentials`

    This will make R available to the Jupyter install.
- Install numbapro (https://docs.continuum.io/numapro)
  - `conda install numbapro`
  - `conda update numbapro`
- To install Julia support for Jupyter (again do this as the Jupyter user)
  - `wget` [`https://julialang.s3.amazonaws.com/bin/linux/x64/0.4/julia-0.4.6-linux-x86_64.tar.gz`](https://julialang.s3.amazonaws.com/bin/linux/x64/0.4/julia-0.4.6-linux-x86_64.tar.gz)

    ```
    tar zxvpf julia-0.4.6-linux-x86_64.tar.gz

    ./julia-2e358ce975/bin/julia
    ```
  - Then from the Julia console, type

    ```
    Pkg.add("IJulia")
    ```

This will install IJulia and make it available to the Jupyter install.

At this point, the AMI with no GPU support is built. In the console, under EC2, there is an option on the left hand side that says "AMI". Select the volume you want to save and select "Create AMI" from the drop down. Be sure to give the AMI a good name but don't make it too long. For example, you could name it,

```
ALinux_Jupyter_NoGPU
```

In the "description" field you can put a better description of the AMI. This can include versions that are used, various tools, etc. Just remember that this description field is limited in size.

It might take a few minutes for the AMI to save. Once it's done, you can terminate the instance if you like. It's a good idea to back to the main part of this document and test out the instance to make sure it works correctly.

GPU installation

Note that Amazon Linux does not have binaries for X since it runs entirely in the cloud and an interactive desktop is usually not needed. However, it has very up to date packages and a very modern kernel. Plus it is designed for the cloud from the beginning.

- Start up a GPU instance using the Amazon Linux AMI that you saved in the previous section (non-GPU). You will have to search your AMI list ("My AMIs") when you launch the instance.
- After the instance starts up, the first thing that needs to be done is to make sure that the kernel source matches the running kernel.
    - `sudo yum erase kernel-devel –y`
    - `sudo yum install kernel-devel-$(uname –r)`
- Install NVidia drivers. This is a manual installation so you will have to answer questions from the NVidia Installer.
    - `wget http://us.download.nvidia.com/XFree86/Linux-x86_64/367.44/NVIDIA-Linux-x86_64-367.44.run`
    - `sudo /bin/bash ./NVIDIA-Linux-x86_64-367.44.run`
        - Note: This is an interactive install using the terminal (i.e. ncurses). Accepting the defaults is usually good enough but be sure to accept the license terms.
- Install Accelerate (https://docs.continuum.io/accelerate/).
    - `sudo su`
    - `su – jupyter`
    - `conda install accelerate`
        - Note: This adds the cudatoolkit version 7.5, mkl libraries, numpa, llvmlite, openblas, scipy, scikit-learn, and a few other tools).

The GPU enabled AMI is now ready at this point. You can follow the steps for saving it as an AMI. Be sure you name it something that indicates that it is GPU enabled such as, `ALinux_Jupyter_GPU`.

Appendix B - CentOS 7

You begin by starting a fairly simple instance type with a base CentOS 7 AMI. For example, use a c4.xlarge instance (4 VCPU's, 7.5GB memory). Select a CentOS 7 AMI with HVM from the marketplace (HVM allows better performance). Just choose "Marketplace" on the left hand side of the console for Launching an Instance. Then search for "Centos 7 HVM" and the AMI should be the first one to appear in the list. For building the AMI and testing it, allow the instance to have a public DNS (this will be useful for installing tools and for Jupyter). Make sure ports 22 and 8080 is open for inbound TCP traffic (for Jupyter).

After instance is booted login as user "centos". Some of these steps are manual, as indicated.

- Install wget (not installed by default).
  - `sudo yum install wget -y.`
- Install EPEL repo (needed for other packages).
  - `wget http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-8.noarch.rpm`
  - `sudo rpm -ivh epel-release-7-8.noarch.rpm`
- Install python-pip so other Python packages can be installed.
  - `sudo yum install python-pip -y`
- Install python-devel (needed to install other python packages).
  - `sudo yum install python-devel -y`
- Install development tools.
  - `sudo yum groupinstall 'Development Tools' -y`
- Update everything.
  - `sudo yum update -y`
- Create "jupyter" user (manual process).
  - `sudo useradd jupyter`
- Log in as "jupyter" user.
  - `sudo su`
  - `su - jupyter`
- Install Anaconda Python as the "jupyter" user. This is a manual process. Note this command install Python 2.x. If you want Python 3. Change the name to "Anaconda3-4.0.0-Linux-x86_64.sh".
  - `wget http://repo.continuum.io/archive/Anaconda2-4.0.0-Linux-x86_64.sh`
  - `bash Anaconda2-4.0.0-Linux-x86_64.sh` (answer questions)
    - Note: Let the Anaconda installer update the .bashrc file
- "source" your .bashrc file (only for testing – you don't need to do this when you log in).
  - `source ~/.bashrc`
- Use "conda", the Anaconda install tool, to install Jupyter and the dependencies.
  - `conda install ipyparallel notebook jupyter -y`
- To install R support for Jupyter (as the Jupyter user).
  - `conda install -c r r-essentials`

  This will make R available to the Jupyter install.

- Install numbapro (https://docs.continuum.io/numapro).
  - `conda install numbapro`
  - `conda update numbapro`
- To install Julia support for Jupyter (again as the Jupyter user).
  - `wget https://julialang.s3.amazonaws.com/bin/linux/x64/0.4/julia-0.4.6-linux-x86_64.tar.gz`

    `tar zxvpf julia-0.4.6-linux-x86_64.tar.gz`

    `./julia-2e358ce975/bin/julia`

  - Then from the Julia console, type

    `Pkg.add("IJulia")`

    This will install IJulia and make it available to the Jupyter install.
- Log out of instance and log back in.
- Go to “root”.
  - `sudo su`
  - Create a file (e.g. “runit”) with the following command:
    - `su - jupyter –c "~/anaconda2/bin/ipcluster nbextension enable; ~/anaconda2/bin/jupyter notebook --ip `curl http://169.254.169.254/latest/meta-data/public-hostname` --port 8080 –-no-browser &"`
    - Make sure the file is executable (e.g. chmod 770)
    - Execute the file:
      - `./runit`

At this point, the AMI with no GPU support is built. In the console, under EC2, there is an option on the left hand side that says “AMI”. Select the volume you want to save and select “Create AMI” from the drop down. Be sure to give the AMI a good name but don’t make it too long. For example, you could name it,

```
CentOS7_Jupyter_NoGPU
```

In the “description” field you can put a better description of the AMI. This can include versions that are used, various tools, etc. Just remember that this description field is limited in size.

It might take a few minutes for the AMI to save. Once it’s done, you can terminate the instance if you like. It’s a good idea to back to the main part of this document and test out the instance to make sure it works correctly.

GPU installation:

- Start up a GPU instance using the CentOS 7 AMI that you saved in the previous section (non-GPU). You will have to search your AMI list ("My AMIs")
- CentOS 7 comes with the Noveau driver for NVidia cards so we first need to disable this before we install the NVidia drivers. You can follow along from the link http://linuxconfig.org.nvidia-geforce-driver-installation-centos-7-linux-64-bit.
    - Log into the instance
    - `sudo su`
    - `echo 'blacklist noveau' >> /etc/modprobe.d/blacklist.conf`
    - `dracut /boot/initramfs-$(uname -r).img $(uname -r) --force`
    - Reboot the instance (only reboot!)
        - ***NOTE – you may have to perform this operation 2 or more times before the NVidia drivers stop complaining about the "Noveau" driver being installed.***
- Install NVidia drivers. This is a manual installation so you will have to answer questions from the NVidia Installer.
    - `wget` [`http://us.download.nvidia.com/XFree86/Linux-x86_64/367.44/NVIDIA-Linux-x86_64-367.44.run`](http://us.download.nvidia.com/XFree86/Linux-x86_64/367.44/NVIDIA-Linux-x86_64-367.44.run)
    - `sudo /bin/bash ./NVIDIA-Linux-x86_64-367.44.run`
        - Note: This is an interactive install
        - If you reboot and try to install the Nvidia drivers and it fails complaining about the "noveau" driver already being installed, reboot the instance again. This should fix the issue.
    - Install Accelerate (https://docs.continuum.io/accelerate/)
        - `sudo su`
        - `su - jupyter`
        - `conda install accelerate`
            - Note: This adds the cudatoolkit version 7.5, mkl libraries, numpa, llvmlite, openblas, scipy, scikit-learn, and a few other tools).

At this point, the GPU enabled AMI is now ready. You can follow the steps for saving it as an AMI. Be sure you name it something that indicates that it is GPU enabled such as, `CentOS7_Jupyter_GPU`.