High Performance Computing Introduction

DS730

In this activity, you will be getting familiar with many of the tools we will use in this course. We will be using a preconfigured Linux machine that has all of the software we need on it. In order to do some true high performance computing, we will be using a cloud service. Amazon Web Services offers a free tier for 12 months. You will also sign up for some educational credits since a couple of the things we are doing do not fall in the free tier.

|  |
| --- |
| **Important:**  You are welcome to use newer versions of the software but *you do so at your own risk*. At the time of this writing, the software shown is the most appropriate software available and everything has been tested with this setup. It’s possible a newer version comes out between the modification of this document and when you are taking this course. It’s likely that everything written here will apply to future updates. However, if something goes wrong with newer software, you will be on your own for troubleshooting it. |

If you already have access to a Hortonworks sandbox, you can skip Task 1. The goal of Task 1 is to create a virtual machine with the Hortonworks sandbox installed on it. Using such a sandbox requires a lot of disk space. Instead of purchasing a book for this course, you may need to purchase an extra hard drive. After completing all of the activities and projects in this course, the amount of space that I needed was close to 70GB. Therefore, I recommend having about 120GB of free space available.

## Books & Other Software

Because of the fast paced nature of this area, there is no required book for this course. However, many of the activities are quite long and detailed so one could consider the set of activities the book for this course. The advantage of this kind of “book” is that it can be updated quickly if there is an error or an update. If you come across a portion of text that no longer makes sense or see an outdated image, let your instructor know right away so we can update the document.

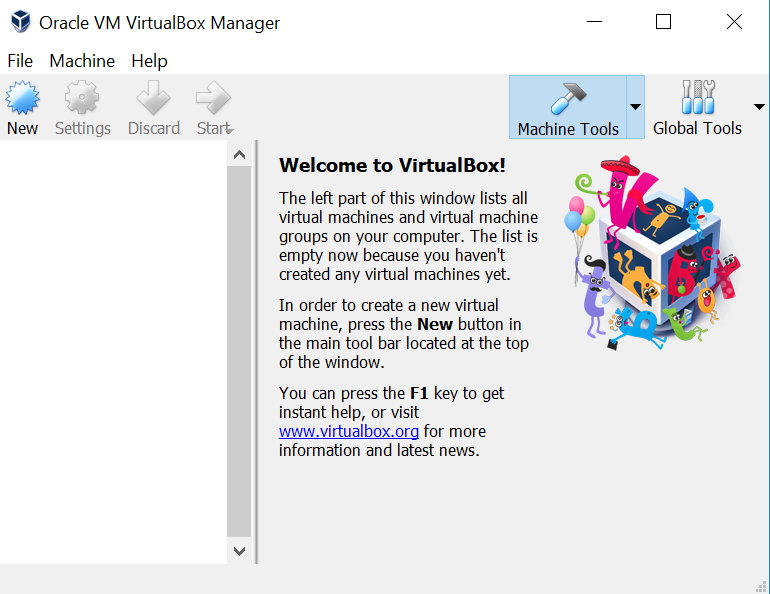
One of the most common questions asked in this course is why we don’t learn tool XYZ where XYZ is the *best* thing we can learn in this course. As the world of high performance computing changes, the goal of this course is to keep up with the newest technologies and present you the newest versions of each software we use. At the same time, we want to teach the core concepts of these technologies. We do not want you to be pigeonholed into using a specific software tool to solve a problem. If you learn the core concepts, then you should be able to pick up whatever the software du jour is when you need it.

# Activity Tasks

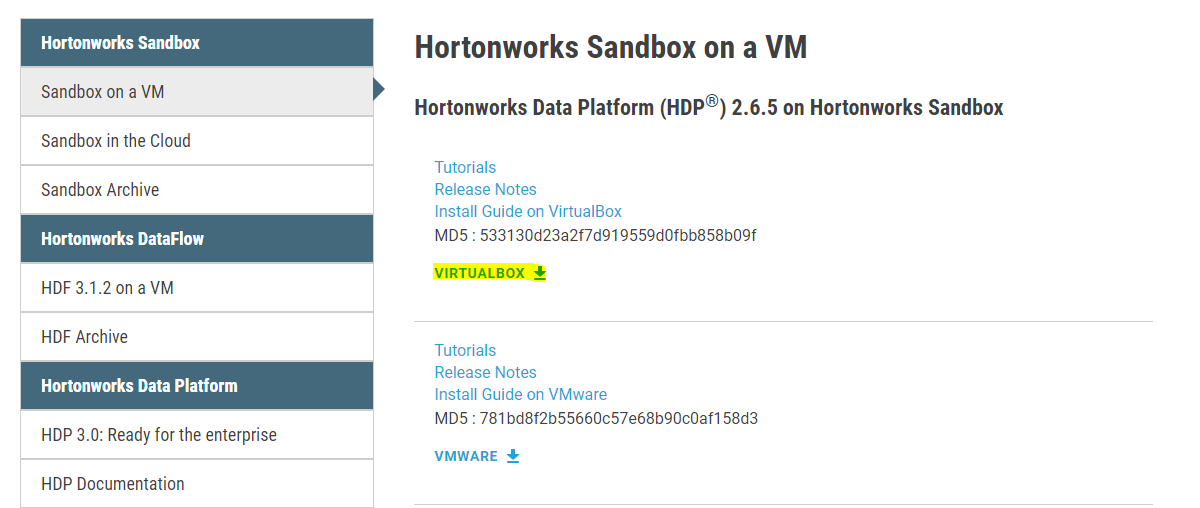
## Task 1: Create a Hortonworks Sandbox

This task creates a sandbox for you and installs all of the software we are using in this course. You will be creating a virtual machine. This task takes a couple of hours depending on your Internet connection speed. However, the majority of it is waiting for everything to download. You will need about 120GB of hard drive space available and a machine that has at least 8GB of memory, preferably more[[1]](#footnote-0). These instructions have worked on a machine running a 64-bit version of Windows 10 and a 64-bit version of Windows 7.

1. **Download VirtualBox**. The current version is 5.2.18. You can find the download here:[www.uwosh.edu/faculty\_staff/krohne/ds730/virtualbo](http://www.uwosh.edu/faculty_staff/krohne/ds730/virtualbox.html)x.html
2. Install VirtualBox on your machine. It is a simple install that requires you to click **Next** and **Install** a few times.
   * If it asks about installing network devices, this is no problem; you can simply click **Install** on those prompts.
3. When the install is complete, click **Finish**.
4. Open VirtualBox up once it is installed. You should see something like this:

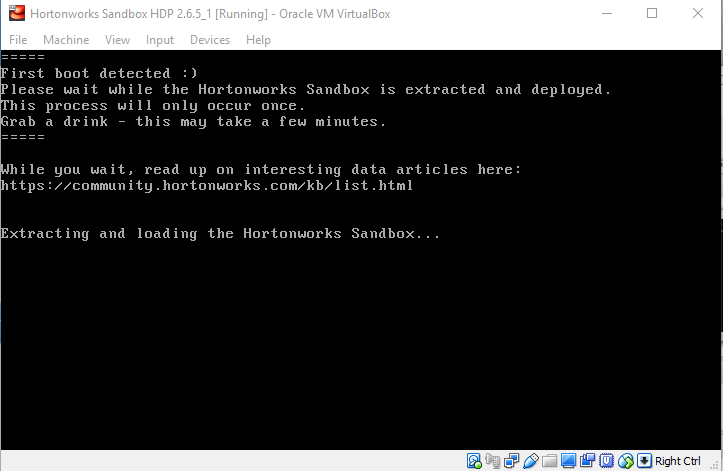


1. **Download Hortonworks Sandbox**. The current version if Hortonworks Data Platform (HDP) 2.6.5. You can download it from here: <http://www.uwosh.edu/faculty_staff/krohne/ds730/hortonworks.html>
   * Be sure to download the one for VirtualBox:

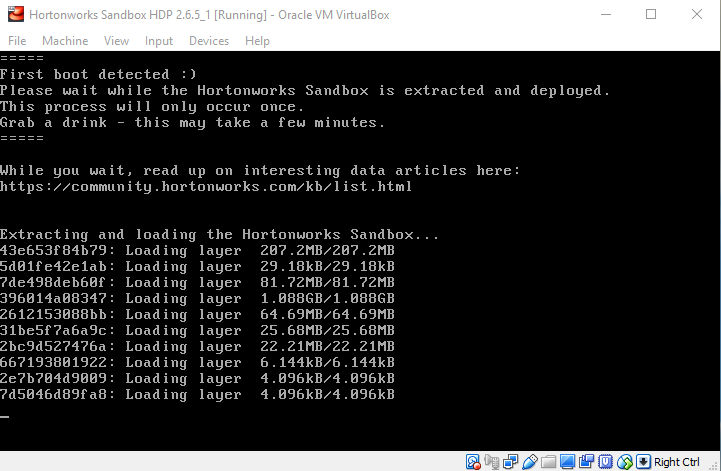


* + It will likely ask you to fill out your name, e-mail address, etc before downloading. Once you submit your information, the download will begin. Note that this is a 15GB download so it may take a while if you have a slower home connection.

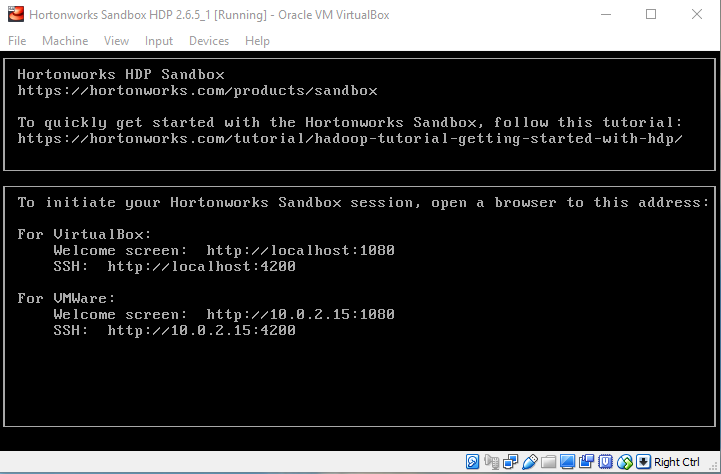
1. Once the file has download, go to VirtualBox and click on **File → Import Appliance**.
2. Click on the folder icon and find the **\*.ova** file you just downloaded and click **Open**. For reference, mine was called HDP\_2.6.5\_virtualbox\_180626.ova. Once you have selected it, click **Next**.
3. All of the settings can remain unchanged. If you wish to store your sandbox on an external drive, scroll down and click on the **Virtual Disk Image** location setting at the bottom and change the location. You may also change the **RAM** setting if you would like to use more/less memory. Do not use less than 4GB of memory or your sandbox may not run well. The default of 8GB is preferred. Click on **Import**.
4. As a reference, my initial *Importing virtual disk image* import took roughly 10 minutes. Once everything is imported, click on the name of the virtual box (probably called Hortonworks Sandbox HDP 2.6.5) and click on **Start**. You should see something like this:



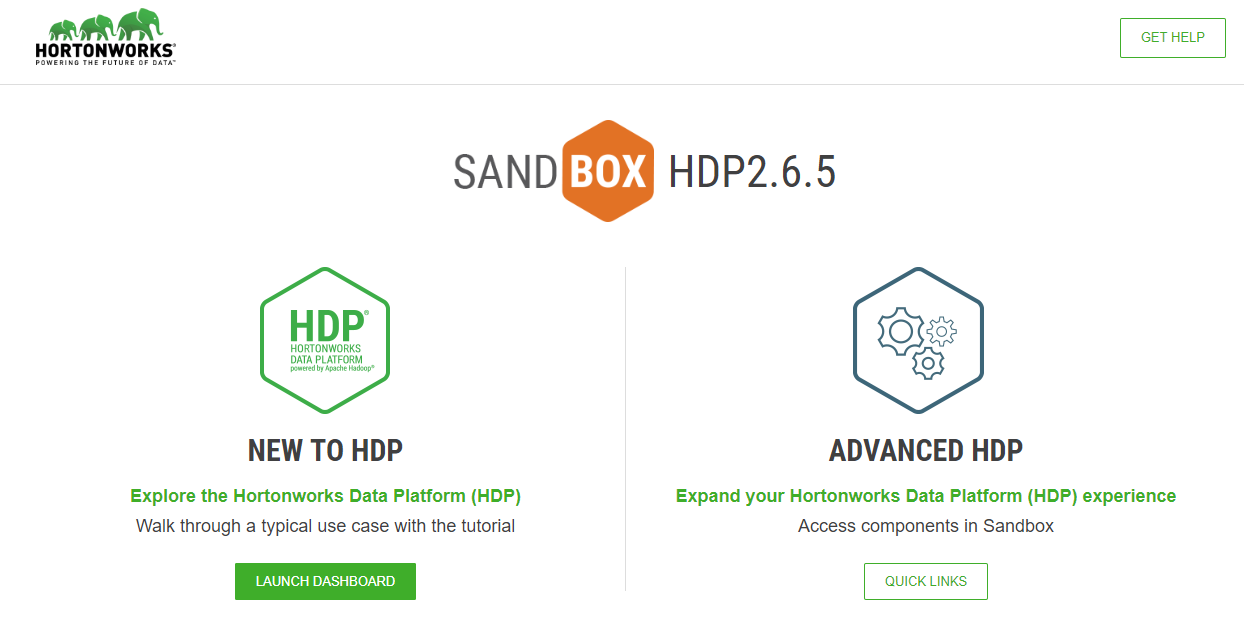
1. This process will take a fair amount of time because you are downloading a lot of software for your sandbox. There is no need to interact with it so just let it run. As a reference, I installed this on my work computer on my very fast 90Mbps school connection. The process took 20 minutes to finish with the *Extracting and loading the Hortonworks Sandbox…* portion before it started downloading a bunch of software:



1. The virtual machine then spent 25 minutes downloading and installing software needed for the sandbox. Starting from step 1 until this point, it took about 75 minutes to reach the following screen.



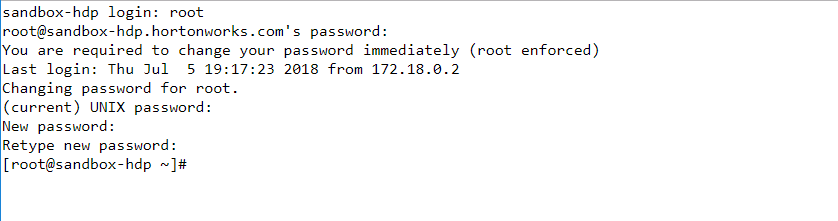
1. Once you see the previous screen, open up your favorite Internet browser and navigate to <http://localhost:1080>. You should see something like this:



### Shell Web Client

The virtual machine that you setup is like any other machine that you have created. There is an operating system, a filesystem, etc. For those of you familiar with the command line, you may want to use the command line to do certain things. Hortonworks provides a nice web client that allows you to connect to your machine.

1. Open up an Internet browser and go to <http://localhost:4200>. You should see something like this:

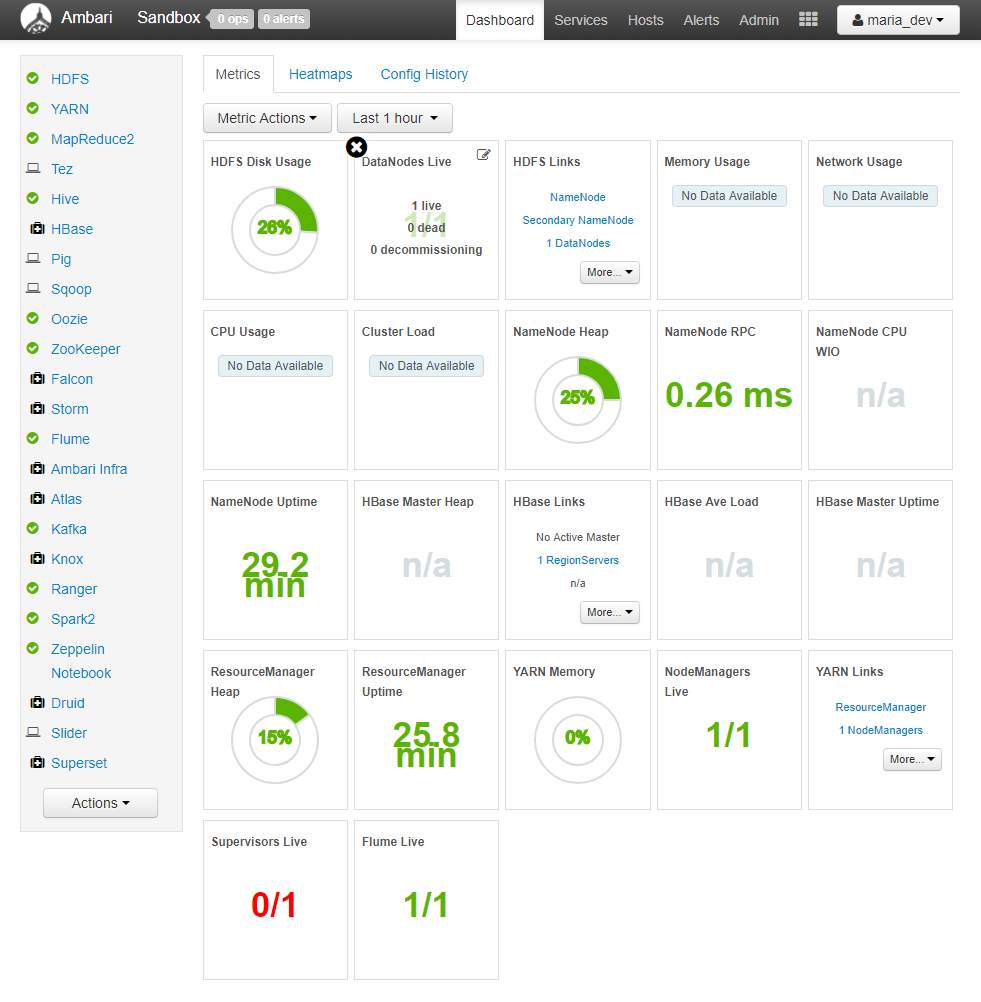


1. Type in **root** for your username and **hadoop** for the password. It will prompt you to change your password (see above). Enter in a new password that is at least 8 characters in length. You can login using root and make any changes that you want to your system.
2. Type in the following command to create an admin account for the Ambari server: **ambari-admin-password-reset**
3. Type in a password that is at least 8 characters long.
4. When you are finished, type in **exit** to close your connection. It will say *Session closed.* and you will see a **Connect** button in the middle of the screen. You can close this window in your browser.

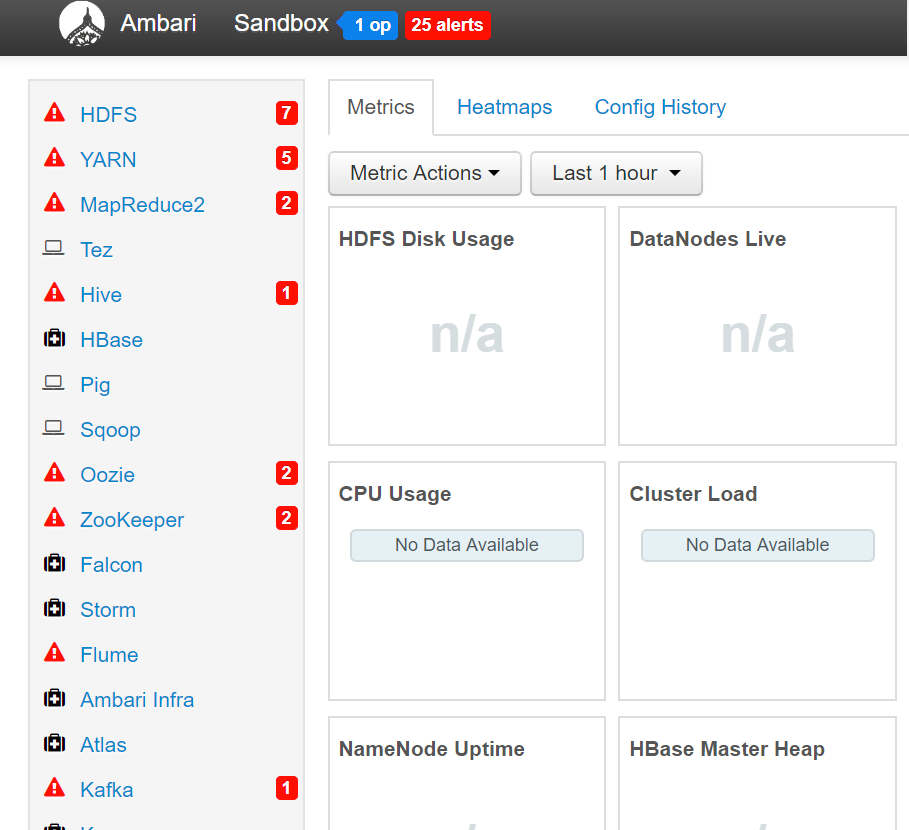
### Explore Ambari

The Ambari dashboard is a nice administrative tool that comes with the sandbox. You will find many applications running but will also find many are stopped. This is fine as we won’t be able to explore everything in the sandbox. Our goal is to view/use some of the main ones.

1. Open up an Internet browser and go to <http://localhost:8080> and type in **admin** for the username and then enter in the password you just created in the previous step to log into Ambari. You will see something like the following image:



1. Click on the **admin** button in the upper right hand corner and then click on **Manage Ambari**. From here you can manage your cluster and do any kind of administrative tasks that your cluster needs. Since we won’t be using this often, you can click on **admin** again and then click on **Sign out**.
2. Log back into the Ambari dashboard by using **maria\_dev** as the username and password. This username will be our main username throughout the semester.
3. Feel free to explore the dashboard to see all of the tools available for you to use.
4. In order to shut down your machine, go back to VirtualBox and click on **File → Close…** Select the **Send the shutdown signal** option and click OK. Your virtual machine will shutdown in a couple of minutes. It may look like nothing is happening but do not force it to close as this might put your virtual machine in a bad state.
5. Once your virtual machine has shutdown, we will open it up again. In order to do this, click on the **Hortonworks Sandbox HDP 2.6.5** and select Start. Your machine will load up in a few minutes. If you see the “welcome screen” in VirtualBox, that means your system is started but it doesn’t mean everything is running. If you quickly go to <http://localhost:8080>, you may get a 502 Bad Gateway error or some other error message. This is normal and it means you opened up a browser too fast. Let the system load up for a minute and try again. If you log in too quickly, you might see something like this:



1. This is normal and there is no concern. Just let the system load itself up. It may take several minutes but eventually the alerts will disappear and everything will be running. If you still have red alerts after 15 minutes, then something is likely wrong. Once everything is loaded up, feel free to poke around the dashboard to see what options you have. There isn’t much to do right now but we will be using many of these software tools in this course. When you are finished exploring, feel free to shutdown the virtual machine.

It is important to understand what is going on inside the sandbox. The sandbox is essentially simulating a cluster of machines that run Hadoop, Pig, Hive and other pieces of software. You do not have to install each piece individually. Rather, you can use the software straight out of the box without having to deal with the administrative part of it. If you want to know more about how to install some of the software, see the optional activities for how to setup a Linux machine, install Hadoop, Pig, Hive, etc.

In this course, we will focus on two parts of the Hadoop environment but there are many other parts and pieces of software you can learn about.

The first part is the Hadoop Distributed File System (HDFS). HDFS allows for scalable and reliable data storage. It is designed to run on many “cheap” computers. HDFS stores data in large chunks and each chunk is stored on multiple machines (in general) to provide reliable access to the data. For example, consider chunk X is stored on machines A, B and C. Assume you start a job that needs to use chunk X. If A and B are busy doing other work, you can run your job on chunk X on machine C. This runs into the classic “space vs time” problem. You can replicate your data as much as you want across numerous machines. If you replicate a lot, your total storage capacity will go down but your processing time will also go down. If you replicate a little, your total storage capacity will go up but your processing time will also go up. We won’t spend much time looking into the innards of HDFS. A general knowledge that HDFS is distributed storage is sufficient for this course.

The main part we will concern ourselves with is how to access and manipulate the data. In this course, we will use MapReduce, Pig, Hive and Spark. There are many other software titles available inside Hortonworks and if there is enough interest, optional activities can be created to learn about Storm, Kafka, Tez or other applications.

## Task 2: Sign Up for Amazon Web Services

It’s important to sign up for Amazon Web Services now so you can apply for the Education Credits and have them in your account before we do start using it.

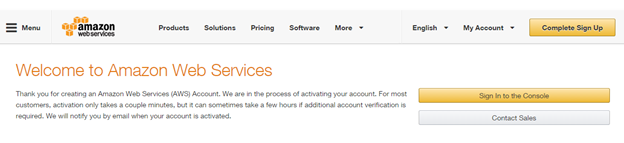
|  |
| --- |
| **Important:**  In order to use AWS, you must enter your credit card information. You need to access a Hadoop setup somehow and it has to be locally (via Hortonworks) or on AWS (preferably both). If you do not sign up, you can use the local Hortonworks setup on VirtualBox but you will not see the true power of Hadoop or high performance computing without AWS. |

If you follow the instructions carefully in future projects and activities, you will not use up all of your Education Credits and you will be charged nothing. However, if you go crazy and allocate 1000 machines and use them for 10 hours, then you’ll have to pay for that. Be careful to only use the processing power you need (what is recommend in the activities/projects) and there will be no surprises.

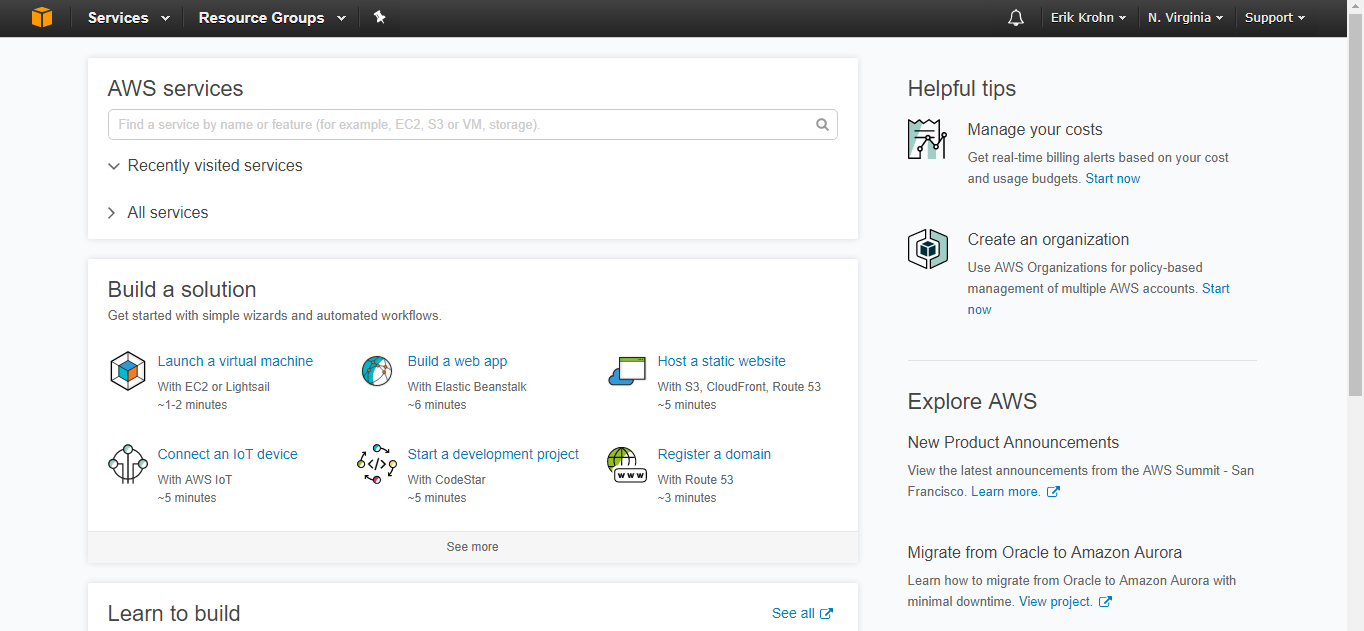
For those of you comfortable with other cloud services like Azure or HDInsight, feel free to use one of those. One of our goals is to see high performance computing with many machines running. Another goal is to learn how to use these cloud computing services. Be warned that if you run into problems with these other services, you are on your own.

The following steps are accurate as of this writing however small things may have changed or might have moved. If something is not accurate, please let an instructor know right away so we can update this document. We will try and keep the instructions generic enough so that any minor modifications won’t matter but specific enough that you know what to do.

1. Go to [http://aws.amazon.com](http://aws.amazon.com/) and click on the **Sign In to the Console** button.
2. You will be asked to sign in or create a new account. Assuming you are a new user, create a new account. Enter your e-mail address and click the **Create a new AWS account** option.
3. You will be asked to supply some login credentials (e-mail address and password). Enter your credentials. Be sure to use your university e-mail account. You will get faster access to AWS credits if you use a \*.edu account. For **AWS account name**, enter in anything you want, e.g. Test Account.
4. Click **Continue**.
5. The next page asks for **Contact Information**. Click the **Professional Account** option as you are using this at an educational institution.
6. Enter your contact information. Enter your university as your **Company Name**.
7. Read the **AWS Customer Agreement** and click the checkbox next to it.
8. Click on **Create Account and Continue**.
9. On the next page, enter your credit card information.
10. The next page asks you to verify your identity. Provide a phone number for AWS to send an automated call.
    * If you are not interested in giving out a cell phone number, I have used a Google Voice number with success.
11. Press the **Secure Submit** button to display a PIN.
12. Answer the call and enter the PIN to verify your identity.
13. Click the button to continue.
14. The next page is to choose your **Support Plan**. Choose the **Basic (Free)** plan and select **Continue**.
15. Assuming everything goes right, confirm you see something like this:



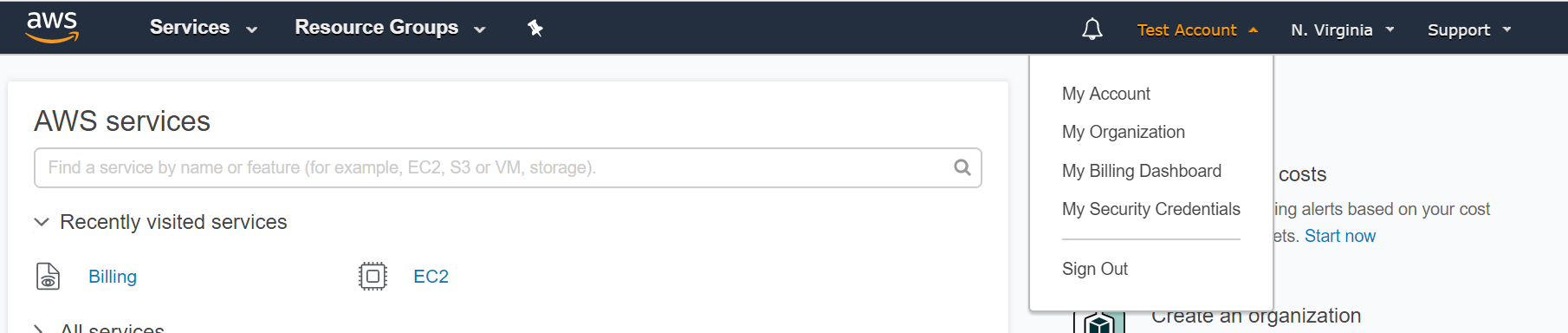
1. Your account should be successfully created. You can click on the Sign In button and should be able to sign into AWS. You should see something like this:



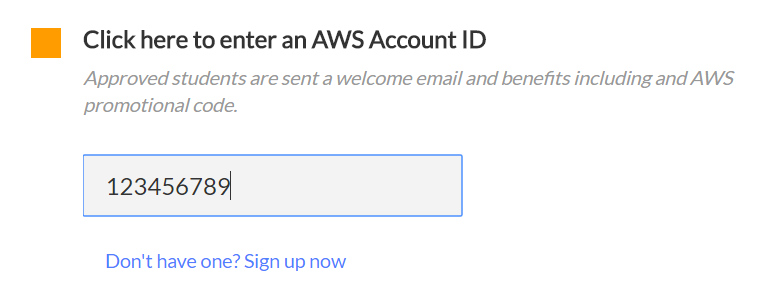
## Task 3: Apply for Educational Credits

Most of the systems we are using are not currently part of the free tier and you always have to pay for whatever processing power you use. Therefore, it’s important to apply for the credits so they are in your account when we do start using AWS.

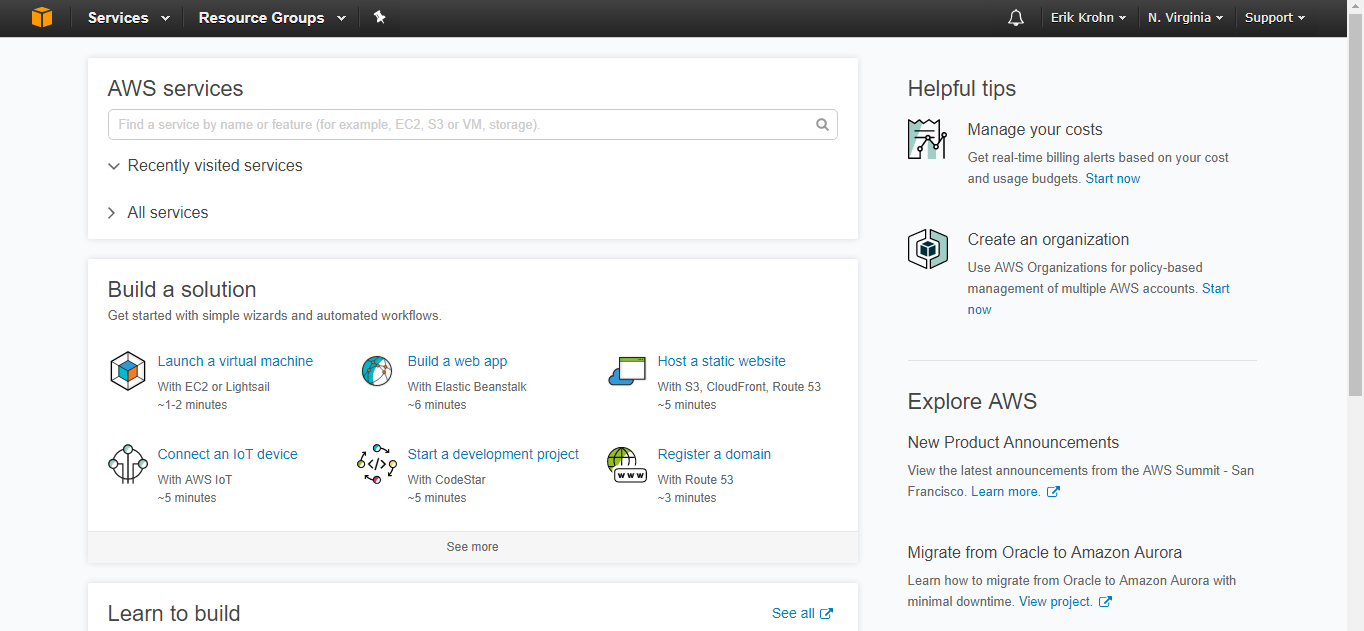
1. To get your credits (renewable annually), go to:  
   <http://www.uwosh.edu/faculty_staff/krohne/ds730/awseducate.html>
2. Click the **Join AWS Educate Today** button.
3. Since you are a student, click the **Student** button**.**
4. Fill out the application on the next page.
   * You should fill it out with your home campus.
   * Enter in your school e-mail address (ending in \*.edu); otherwise it will likely take longer to verify.
5. Hit **Next**.
6. Since you just signed up for AWS in the previous task, you already have an AWS Account ID. Go to your AWS main page, click on the account name you created and then click on **My Account**. See below:



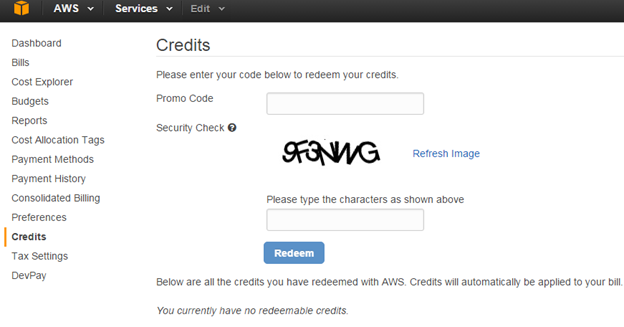
1. Once you have your account id, go back to the AWS Educate signup and enter in your AWS Account number here:



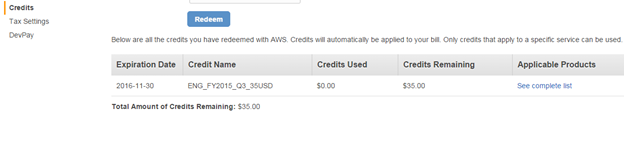
1. The next step is to verify your address. AWS should send an e-mail within seconds of filling out your application. Find this email and click on the link inside of it.
2. Read the terms and conditions and choose accept assuming you accept them. Click the **Submit** button.
3. Wait for and open the approval e-mail.
   * Assuming you give your home campus e-mail address, your application will *likely* be approved instantaneously. However, when I applied for my *Educator* account, it took a few days. Since you are probably doing this at the start of the semester, many other people are also applying for these credits. Therefore, if your account is not immediately approved, it may take several days for your application to be approved. If it takes longer than 5 business days, log onto <http://aws.amazon.com> and use their support to ask them what the issue is. In my experience, their support is phenomenal and you will have a resolution quickly.
4. You will have a credit code inside of the approval e-mail. Copy that credit code.
5. Go back to [http://aws.amazon.com](http://aws.amazon.com/) and log in. You should see the initial screen:



1. In the upper right corner, click your account name.
2. Click **My Billing Dashboard**.
3. On the left side, click the **Credits** link. You should see something like this:



1. Type in your **Promo Code**.
2. Pass the **Security Check**.
3. Click **Redeem**.
4. In a few seconds, confirm you see $40 in credits remaining (the test account credit at the time of this screenshot was $35):



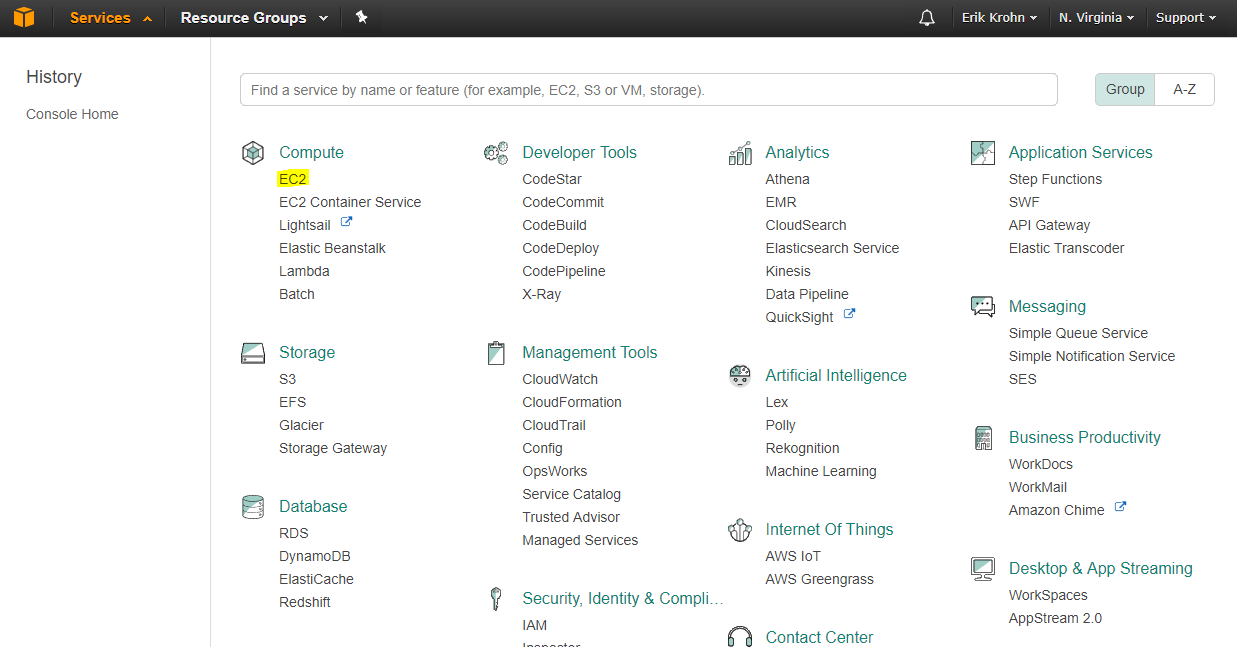
## Task 4: Connect to a Linux Machine on AWS

Connecting to a machine in the cloud is a good start for the cloud based computing we will be using in future activities and projects. We will not connect many times to a remote machine in the cloud but it is something you should know how to do. Whenever you spin up a cluster on AWS or some other cloud service, you should be aware of how to connect to the cluster to make any tweaks you want. The instance you will be creating first will be a *micro* instance which falls under the free tier of usage. Therefore, even if your AWS credits were not instantly approved, this instance will not cost you anything if you create it now.

1. Go here: [www.uwosh.edu/faculty\_staff/krohne/ds730/putty.html](http://www.uwosh.edu/faculty_staff/krohne/ds730/putty.html)
2. Download these two files:
   * **putty.exe**
   * **puttygen.exe**

|  |
| --- |
| **Note:**  You can download the Windows installer to install more than you need. However, downloading what is specified above is sufficient. |

1. Sign into your AWS console by going to<http://aws.amazon.com>.
2. Click **Services**.
3. In the **Compute** section, click **EC2**:



1. Click the **Launch Instance** button.

|  |
| --- |
| **Note:**  As a comment for the future, if you are looking to save some money, you should look into Spot Requests (Instances). Everything explained in this course is with *On Demand* instances. Basically, when you start up an *On Demand* server, it will run continuously without being interrupted. |

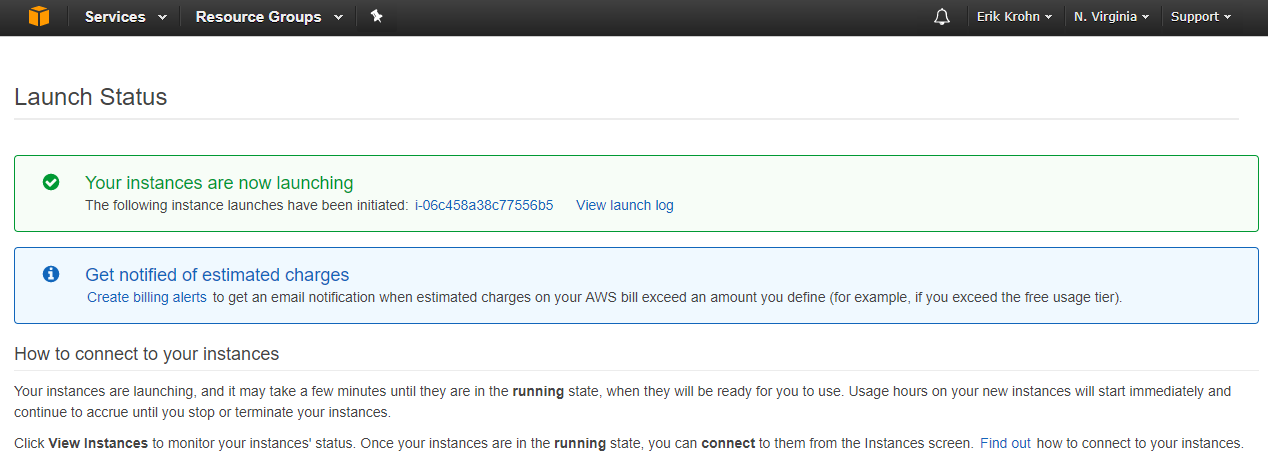
1. Select the **Ubuntu Server…** option.
   * As of now, it is version 16.04.
2. Choose **General Purpose t2.micro** to stay in the free tier. If you choose a larger option (e.g. t2.small, t2.medium), all computing you do using EC2 for the course will likely be free because you will be covered by your educational credits (the cost is currently about 2.5 cents an hour for t2.small). The one caveat is that you don’t go crazy with usage. Make sure to stop your instance when you are done using it and stay within the recommended bounds as described in future documents. Another thing to note is that there are only 1 CPU available and 1GB of memory on the t2.micro machine. If you need more processing power or more memory, adjust accordingly.

|  |
| --- |
| **Note:**  The call is yours to make on how big your instance is. Once we get to Java, you will need at least t2.medium. If you need to adjust your instance type, look at step 84 and that will show you how to change the type. You can find all of the pricing details here:<https://aws.amazon.com/ec2/pricing/> It's unlikely you'll even come close to your credit amount if you choose to go with t2.medium right away. |

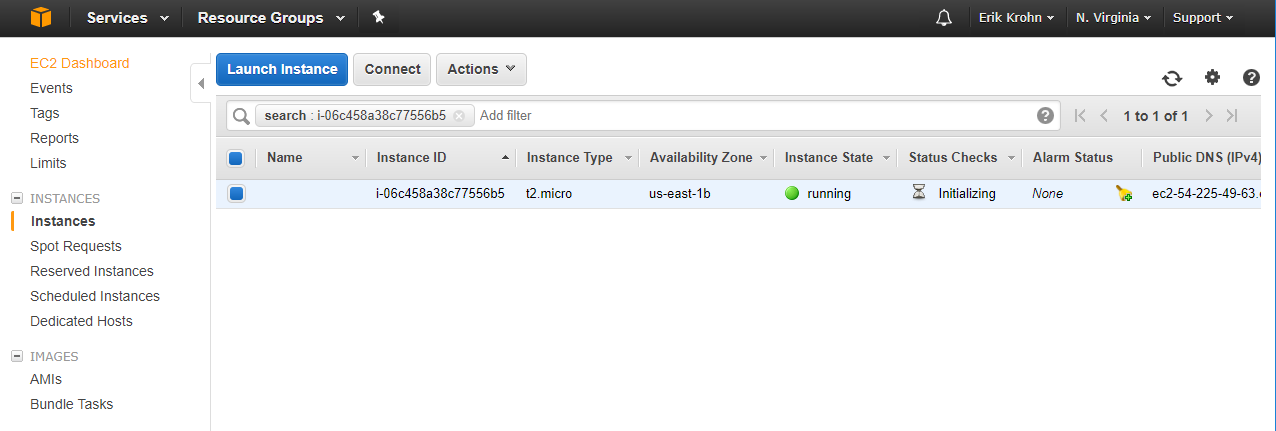
1. Once you've made your decision, click **Next: Configure Instance Details**.
2. All of the defaults are good here. click **Next: Add Storage**.
3. Change the GB size from 8GB to 30GB.
4. Click **Next: Add Tags**.
5. The default values on this page can be kept as they are. Click **Next: Configure Security Group**.
6. Make sure **Create a new security** group is checked.
7. Change the **Source** from **Custom** to **Anywhere**.
   * This assumes you will be connecting to the instance from any machine. You should change it at **My IP** if you will only be connecting from 1 machine and are hyper concerned about security.
8. The type should be **SSH**.
9. The Protocol should be **TCP**.
10. The Port Range should be **22**.
11. Click **Review and Launch**.
12. Click the **Launch** button.
13. Click **Create a new key pair**.
14. Enter a **Key pair name** of whatever you want.
    * I called mine **UbuntuAWS**
15. Click **Download Key Pair**.
16. Save that file somewhere to your hard drive and do not lose it!

|  |
| --- |
| **Important:**  If you lose your key, you will not be able to access the server. |

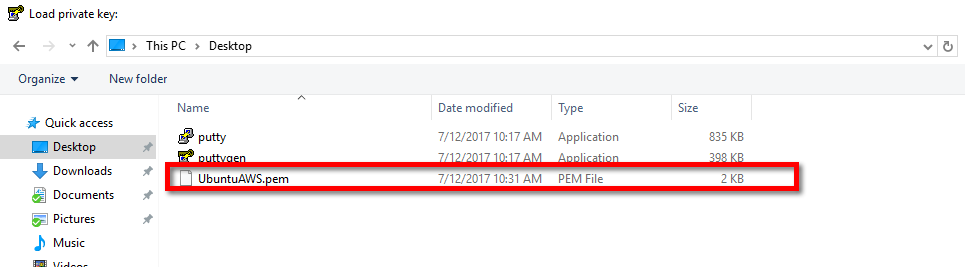
1. Click **Launch Instances**.
2. Confirm you see something like this:



1. Click the link that appears after **The following instance launches have been initiated**.
   * In the previous screen shot, the link appears as **i-06c458a38c77556b5**. Yours will have a different name.
2. Once you click that link, confirm you see something similar to this:



1. In the **Instance State** column, notice it says **running**. If it is still initializing, pending or waiting, then wait for the instance state to be in running mode before continuing.
2. Scroll to the right and find your **Public DNS (IPv4)**. Write this address down.
   * For reference, mine was called: **ec2-54-225-49-63.compute-1.amazonaws.com**
3. The .pem file downloaded in Step 24 is not compatible with PuTTY. Because of this, we need to create a key file that PuTTY can read. Open up the **puttygen.exe** file that you downloaded in Step 2.
4. Click the **Load** button.
5. By default, PuTTYgen only looks for files with extensions of ppk so you must change it to look for all files. Find your .pem file that you downloaded in Step 24, which will look something like this:



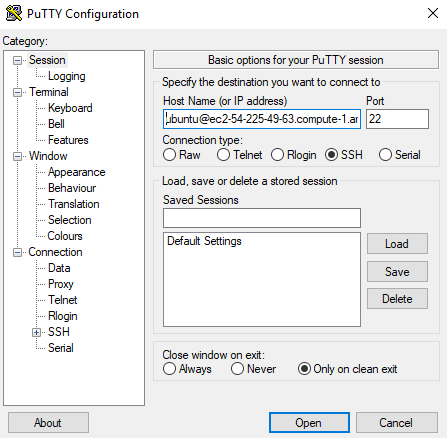
1. Click **open**.
2. PuTTYgen will give you some message about successfully importing the foreign key. Simply click **OK**.
3. In order to save the key that PuTTY can use, click the **Save private key** button.
4. The system will ask you if you want to save it without a passphrase. Click **yes**.
5. Save this **ppk** file somewhere safe.

|  |
| --- |
| **Important:**  If you lose this file, you will not be able to access your server. |

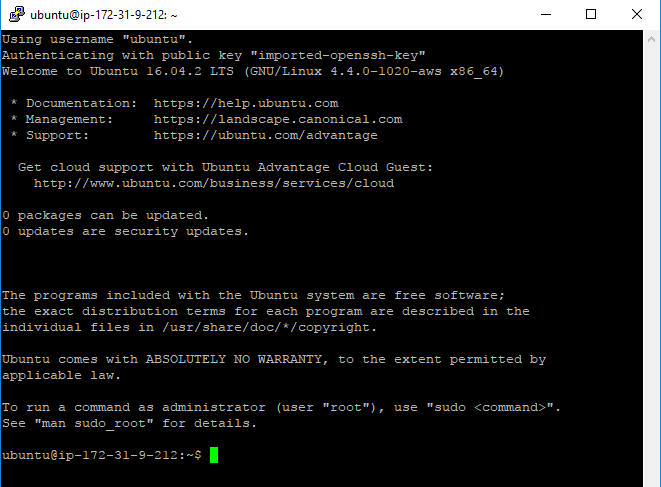
1. Close out of PuTTYgen.
2. We are now ready to connect to our server. Open **putty.exe**.
3. In the **Host Name** section, enter **ubuntu@** followed by the IP address you wrote down in Step 31.
   * My **Host Name** was:  
     [ubuntu@ec2-54-225-49-63.compute-1.amazonaws](mailto:ubuntu@ec2-54-225-49-63.compute-1.amazonaws.com).com

|  |
| --- |
| **Important:**  Be aware that every time you restart your server, you will likely get a new IP address. |

1. Enter in a **Port** of 22. You should have something similar to this:



1. On the left side, click the plus symbol next to **SSH**.
2. Click the **Auth** option.
3. Click the **Browse** button for the **Private key file for authentication**
4. Find your ppk file that you saved in Step 39.
5. Click the **Open** button.
6. When the system asks you about accepting an RSA fingerprint, indicate **Yes**.
7. Assuming everything went correctly, confirm you see something like this:



1. We will install Java as we will need it later in the course. Before we can install it, we need to update our package list. In the PuTTy terminal window, type in:  
   **sudo apt-get update**
2. In order to install Java, enter this command:  
   **sudo apt-get -y install openjdk-8-jdk-headless**
3. Enter **javac -version** 
   * The system will say it is 1.8.0\_XXX.
4. Make sure your software is up to date by entering:  
   **sudo apt-get upgrade**
5. If necessary, enter ‘**Y**' to the upgrade.
6. If you get messages about updating the /boot/grub/menu.lst file, choose the default option of **keep the local version currently installed**.

## Using VIM

We will have to edit files on our Linux filesystem from time to time. If you have a favorite editor, feel free to use that. A simple one explained here is called Vim. There is a presentation that covers Linux and how to install a GUI if you want. Please view that presentation to learn how to get a graphical interface instead of the command line.

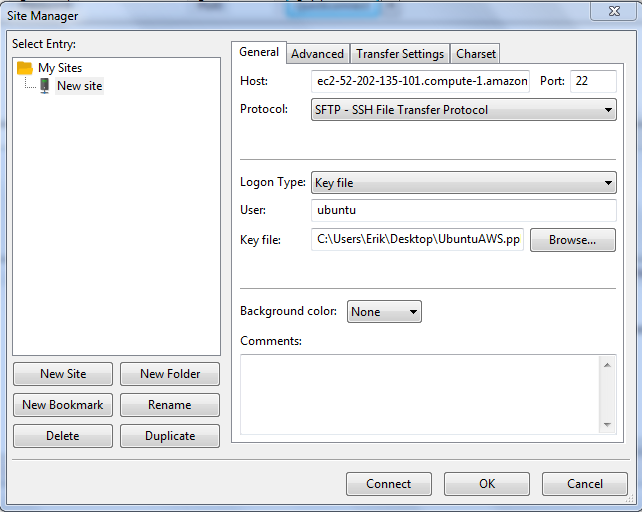
1. To edit a file, enter  
   **vim nameOfFile**   
   This will open up a file called nameOfFile in a program called Vim, which you can think of as Notepad.
2. Press the letter **i** to enter Insert mode.   
   Notice the word **-- INSERT --** on the bottom. Once you are in insert mode, you can type like you normally would.
3. Add the following text to the file:  
   **Hello, this is a test.**
4. After you have added that text to your file, Press the ESC key.   
   You should notice the -- INSERT -- disappear from the bottom.
5. Enter the following exactly as it’s written:  
   **:wq**  
   That’s a colon, then w, then q, then the <enter> or <return> key. This will save your file and take you back to the command prompt.
6. To ensure the file was created successfully, use the following command to display it on the screen: **cat nameOfFile**.

|  |
| --- |
| **Note:**  If you are struggling with Vim, search online for Vim tutorials such as <http://www.openvim.com/> and read up on the commands. Once you’ve learned a few of the commands, it becomes very easy to use. Another common editor is called **nano** and instructions for how to use it are in the Linux presentation. |

### Connect to EC2 with FileZilla

In order to connect to your EC2 instance using FileZilla to transfer files, follow these steps:

1. Download FileZilla from <http://www.uwosh.edu/faculty_staff/krohne/ds730/filezilla.html> and install it. The install is straightforward.
2. Open FileZilla.
3. Go up to **File > Site Manager**.
4. Click the **New Site** button.
5. Enter in your IP address in the **Host**.
6. For the **Port,** enter **22**
7. Change the **Protocol** to SFTP.
8. Change the **Logon Type** to Key File.
9. For the **User,** enter **ubuntu**
10. Click the **Browse** button.
11. Find the .ppk file that you created using PuTTYgen previously.
12. Confirm that your screen looks something like this:



1. Click **OK**.
2. Directly below File, find this icon: . Click the dropdown box and select **New Site**.
3. If the system asks you to trust the key, simply indicate **OK**. You should be connected.

|  |
| --- |
| **Note:**  Whenever you need to reconnect to your EC2 instance, go to **File > Site Manager** and update the IP address (i.e. **Host**) as needed. |

### Test Connection and Stop Instance

1. To test what you just did, type **python3** and the python interpreter should load. To quit out of the Python interpreter, type **exit**.
2. When you are finished, enter **exit** to quit your session.

|  |
| --- |
| **Important:**  You now need to go back to the AWS console page and stop your instance. This is especially important if you chose the t2.small instance or above. You don't want to pay for idle time if you are not connected to your machine. |

1. In order to stop your instance, first click your instance.
2. Go up to the **Actions** button and go down to **Instance State**.
3. Click the **Stop** option.
4. You might see a box that says something about losing ephemeral storage. This is fine; click **Yes**. Wait on this page until the instance is stopped.
5. This is not a step that you need to complete right now but is being provided for the future. If you want to change your instance type, this is the webpage where you do it (when the instance is stopped). You need to use t2.medium or above to see the powerful of multithreading near the end of the semester. It's a CPU issue; smaller instances do not provide multiple CPUs. In order to change your instance type, ensure that your instance is stopped. Right click on the instance you want to change. Choose **Instance Settings** and choose **Change Instance Type**. Choose the instance type you want and hit **Apply**.
6. At the end of the semester, make sure you terminate your instance. An instance that is not running will not cost anything in processing time. However, you will still be using 30GB of storage with the instance. This storage is not free after the 12 month free tier is over. Be sure to terminate your instance(s) at the end of the semester or you will be charged (once your credits are depleted) for this storage.

|  |
| --- |
| **Important:**  During the semester, I recommended *stopping* as you will be able to restart your instance as it were without having to reconfigure anything. If you *terminate* it each time throughout the semester, you will lose everything and have to redo everything.  Be aware that when you stop your instance, you lose your IP address. In order to restart your instance:   1. Come back to this page. 2. Click your instance. 3. Click **Actions**, **Instance state** and then **Start**.   Every time you start your instance, you'll likely have a new IP address. |

## Task 5: Testing Java

We will be testing Java in next week’s activity so we will wait until then to officially test it. However, if you are interested in testing it now, the steps are below.

1. Connect to your EC2 instance in the cloud using putty.
2. Create a folder called **JavaExamples**. To do this, type in **mkdir JavaExamples**. To enter that directory, type in **cd JavaExamples**.
3. Create a file called **Test.java** in that folder (see VIM from earlier).
4. Enter the following code into that file[[2]](#footnote-1):

|  |
| --- |
| **public class Test{  public static void main(String args[]){  System.out.println("It works!");  } }** |

1. When you are back at the terminal window, compile your program by entering   
   **javac \*.java**
2. In order to run the file, enter **java Test**
   * You should see **It works!** printed to the screen.

## Task 6: Test Python

It is assumed that you have some generic programming skills before starting this course. You should be familiar with the following topics before starting this course:

Selection Statements (e.g. if)  
 Repetition Statements (e.g. while, for)  
 Variables  
 Lists/Arrays  
 Calling Functions/Methods  
 Creating Functions/Methods

If you do not know what one of those topics is or do not have a good grasp on how to use each one, you might want to reconsider whether you are prepared for this course. We will write a few short Python programs to test out our Python installation and also to assess your programming ability. You must test your code on your EC2 instance using the command line. If you find yourself taking many hours to solve these tasks, then it might be best to take a refresher programming course before continuing with this course.

|  |
| --- |
| **Important:**   * You are not allowed to use any external libraries (e.g. numpy, sympy, etc.) for these problems as they make the problems trivial. * Make sure to follow the directions exactly as my tester code will not work if you don't. For example, in the first problem, if you do not call your file **first.py** or create a function called **fact** instead of **factorial**, my automated tests will fail. The final Python problem addresses the importance of creating appropriate output. |

### Run Python program from terminal

Many of you may be unfamiliar with the terminal (command line) and we will be using it for a portion of this course. The following short instructions describe how to create a Python file using only the terminal and how to run your Python code from the terminal.

1. Connect to your EC2 instance in the cloud using putty.
2. Create your Python programs using vim as described before. You may also create your primes.py file locally and transfer it over to your EC2 Linux machine using Filezilla.
3. Once you have created your **first.py** program, go back to the terminal so that you can test your code.
4. In order to run your program, enter:  
   **python3 first.py**

### Create a Factorial Calculation Program

You will be creating a program that calculates the factorial of a number.

1. Create a file called **first.py**.
2. Inside that python file, create a function called **factorial** that accepts a number as an argument and returns the factorial of that number. Do not print out anything in the factorial function. Your factorial function should work as expected no matter what you write in future steps. If the factorial function is called with a negative number, a -1 is returned from the function.
3. Make sure your function definition looks like this:  
   **def factorial(val):**
4. Inside your python file (but outside of the factorial function), prompt the user to enter in an integer.
   * You can assume that the value being entered is an integer but you should not assume that it is greater than or equal to 0. If the number entered is less than 0, prompt the user again until the number entered is greater than or equal to 0.
5. Once you have a number that is greater than or equal to 0, call the factorial function and print out the answer returned by the factorial function.

### Output Average of Integers

Create a program that reads in integers and outputs the average of all of the numbers. Your program will be called using the following command:  
**python3 second.py < someInput**  
Some things to note:

* Only integers will appear in the input file.
* All of the integers in the input file will be separated by a space.
* Create a separate file called second.py to solve this problem.
* Do not read in from a specific file.

### Output Prime Numbers

The goal of this task is to create a Python program that prompts the user to enter 2 numbers. Your Python code must be stored in a file called **primes.py**. Your program then prints out all of the prime numbers **strictly** between those 2 numbers in increasing order. If there are no prime numbers strictly between those 2 numbers, then a **No Primes** message is printed out. The order that the numbers were entered does not matter.

**Output:** If there are no primes between the two numbers, your code outputs exactly:

**No Primes**

If there is one prime number, only that 1 number prints out.

If there are multiple prime numbers, the following pattern is used:

**firstNum:secondNum!thirdNum,fourthNum:fifthNum!sixthNum,seventhNum**

In other words, you separate the first number from the second number with a colon. You separate the second number from the third number with an exclamation point. You separate the third number from the fourth number with a comma. This pattern of separation is repeated in subsequent numbers (colon, exclamation point, comma) until there are no more numbers left to be printed. Note there is no delimiter before the first element nor is there a delimiter after the last element.

|  |
| --- |
| **Example:**  For example, if **5** and **24** were entered in that order, then **7:11!13,17:19!23** would print out.  If **24** and **5** were entered in that order, then **7:11!13,17:19!23** would print out. |

### Interpret Dirty Data

Throughout this course, your output will be tested using automated testers. Therefore, it is important that your code is contained in the files we specify and produces the exact output we expect. Having incorrect delimiters, incorrect spacing, etc will cause our automated tests to fail. Be sure you read the output specifications closely so that you are producing the correct output. The output of the previous problem is quite arbitrary (although it assesses good problem solving skills) but this is done to ensure you are prepared for future output requirements. Future output requirements will be much more natural and obvious.

To show you why it is helpful to have all output be the same, consider the following problem. You are a manager for a global weather company and you have asked your national offices to send you information in the following comma separated format:

Year,Month,Day,TimeCST,TemperatureF,WindMPH

All offices put a header row in each file to explain the data. A key thing is that the temperature should have been in Fahrenheit and located in the second to last column. The wind should have been located in the last column and should have been in miles per hour. None of the offices sent the data exactly how you asked for it. Your job is to figure out what each of them did and then answer the following 2 questions:

1. Take the average of all of the wind speeds for March, 2006. Which city had the closest wind speed to 8.30 mph?
2. Take the average of all of the temperatures for 2006. Which city had the closest temperature in fahrenheit to 49.65.

You can download the 4 files from <http://www.uwosh.edu/faculty_staff/krohne/ds730/a1Weather.zip>.

The 4 cities are ABC, KLM, PQR and XYZ (see filenames). The way you clean the data and obtain the answers is entirely up to you. However, you must document the steps you took to clean the data to obtain an answer. Your steps must be included in a file called **dirty.txt**. If you wrote code to solve this problem, upload any code that you created. Also create a text file called **answers.txt** that contains the answer to each of the above questions.

## Task 7: Submitting your Work

We want to transfer the files to your local machine and zip them up for submitting. Once you have finished writing your Python code, you want to transfer them to your local machine. This can be done with Filezilla. In order to transfer files:

1. Open FileZilla.
2. Click **File > Site Manager**.
3. Click **New Site**.
4. Enter in your host name. As a reminder, mine was **ec2-54-225-49-63.compute-1.amazonaws.com**. Yours will be different.
5. Set the **Port** to **22**.
6. Change the protocol to **SFTP**.
7. Change the **Logon Type** to be: **Key file**.
8. For the User, enter **ubuntu**
9. Click **Browse**.
10. Find your .ppk file.
11. Click the **Connect** button.
12. There may be an option to save the password. You can choose whatever you want here.

You will see your local machine’s files on the left side and your EC2 instance files on the right side. You can transfer files now. You should have a total of 3 Python files: **first.py, second.py,** and **primes.py**. You should also have a **dirty.txt** file and an **answers.txt** file. Zip up these 5 files into 1 zip file called **a1.zip** and submit **a1.zip** to the dropbox on d2l.

## Task 8: Introduce Yourself

Lastly, go to the online discussion board on Piazza (use the Introductions thread that is pinned at the top left of the page) and enter in some information about yourself. Since this is not a face-to-face course, I will likely never meet most of you. However, I would still like to know something about you. You can put whatever you want in that post. Tell me who you are, why you are interested in our data science degree, any interests or hobbies, what you hope to learn from this course, whether or not you signed up for AWS, etc. You can write as much or as little as you want. If you have questions for me, feel free to add those and I’ll answer them directly. I know you’ve probably had to do something similar in other courses so feel free to copy and paste that “intro document” so I can get to know you a little bit.

1. I had success running the sandbox while allocating only 4GB of RAM to the system and it has worked fairly well. It was a little slow but it was acceptable. I tried running this with 2GB of RAM allocated and it didn’t work. [↑](#footnote-ref-0)
2. In general, copy and paste is safe with the notes unless there are quotations. [↑](#footnote-ref-1)