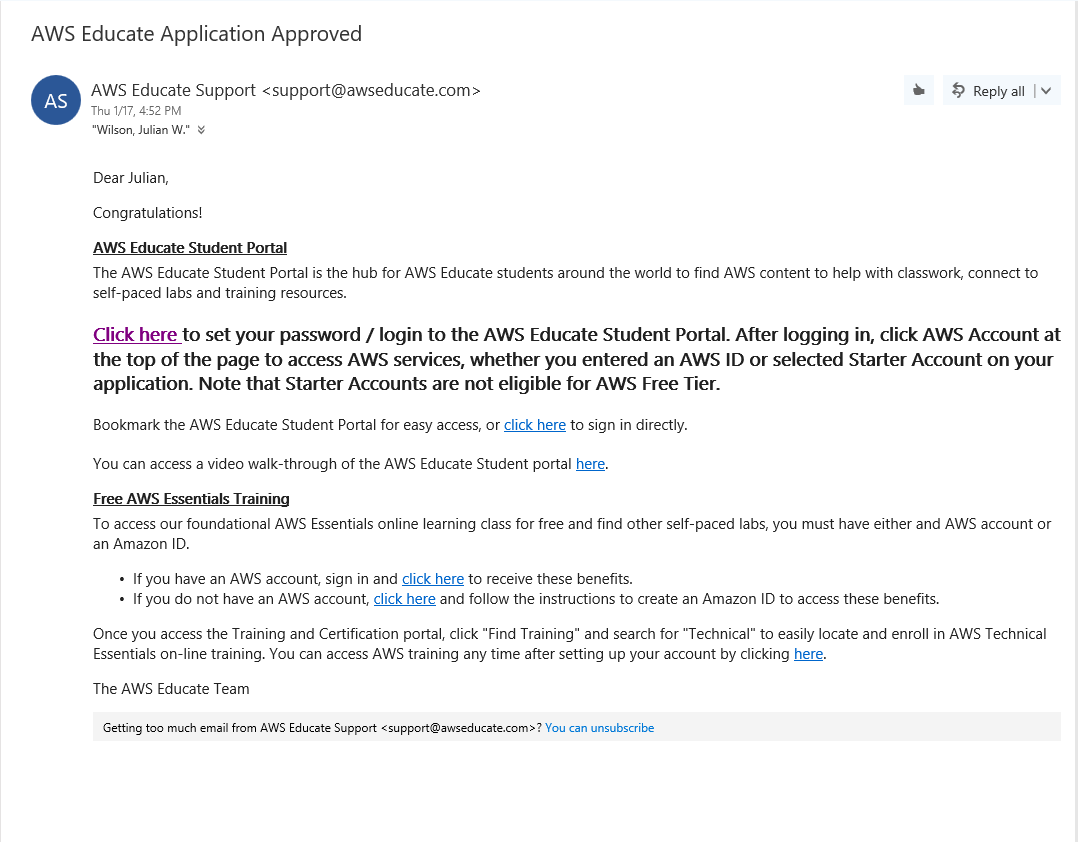
# Virtual Machines (VMs) using Cloud Services

We have created VMs on our own hardware using VMware Workstation Player. Now, we’ll create a Linux VM on Amazon Web Services (AWS). There are many good cloud service providers (see <https://clutch.co/cloud>.) We will use AWS because it is well known and because they offer free services for education. The basic concepts you learn here will apply to any cloud service provider.

# AWS Billing

AWS Educate gives each account $50 in credit, which should be more than enough for our course. Nonetheless, it is prudent to get in the habit of minimizing the expense we occur for our AWS usage. The two major charges for our class will be storage and CPU time. Each instance you create will use a few GB of S3 storage and will be charged. The storage cost is fairly small, so we don’t need to worry about it too much. Just don’t keep lots of images that you don’t need. The charge for CPU time and the IP address a running instance uses is higher. To save money here, shut down your instance when you aren’t using it. When you stop and instance it will release its IP address, and be given a new one when you restart it. This is inconvenient, but it saves money.

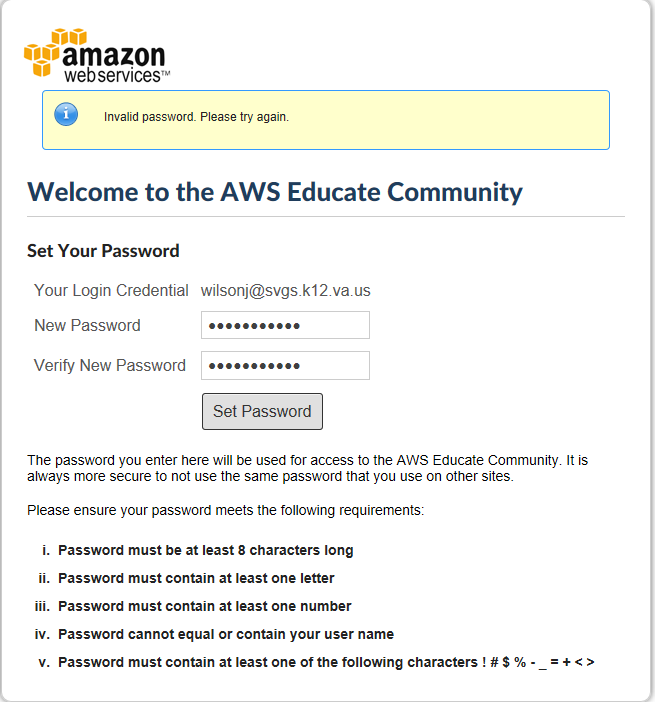
# Access your AWS Account

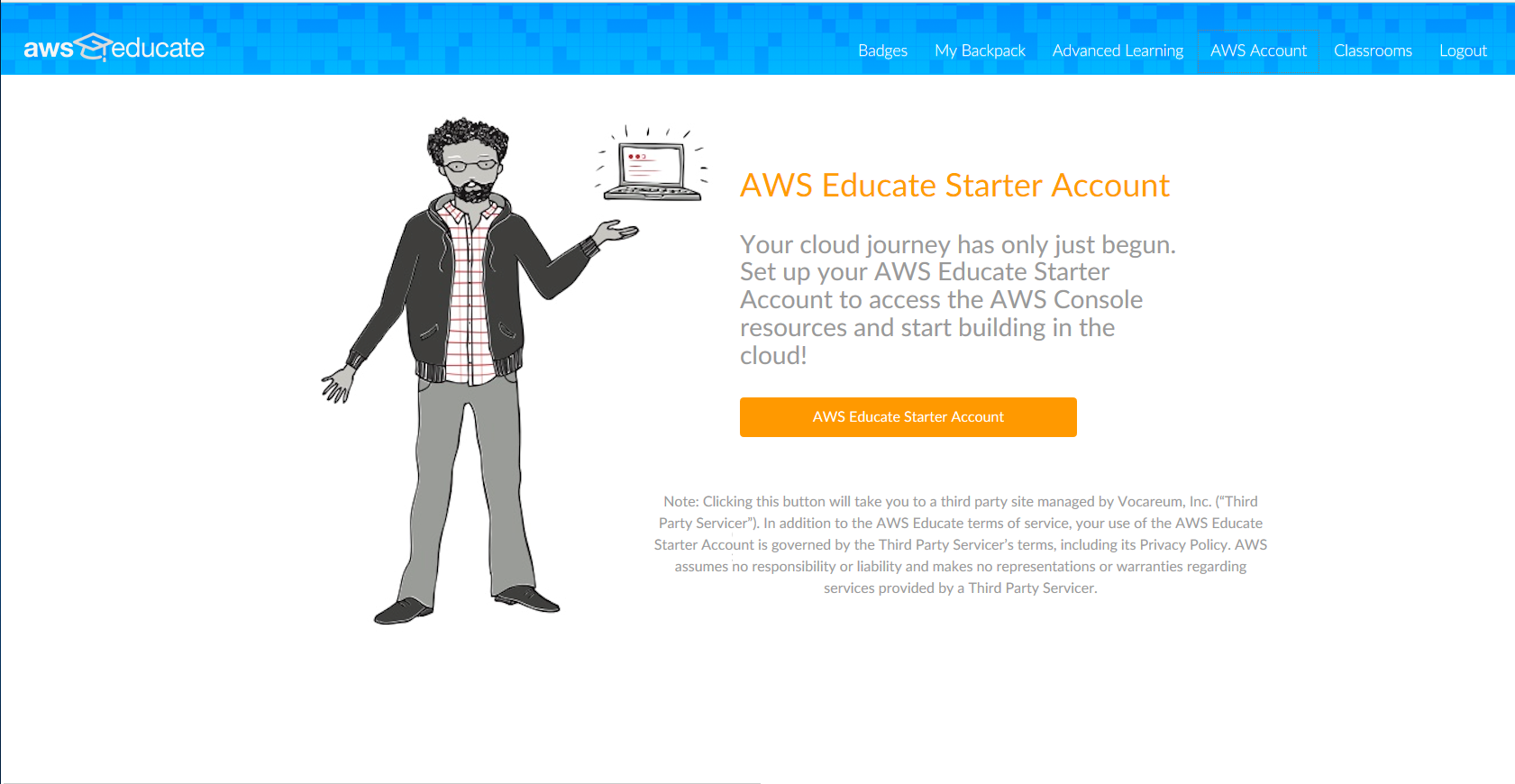
We are using AWS services through AWS Educate. Your instructor will add your SVGS email account to the AWS, and AWS will send you an application email.  


Click where it says “Click here” to start your AWS account. You always click anything that says “click here” don’t you? Seriously though, you should have some reasons to believe the link is safe:

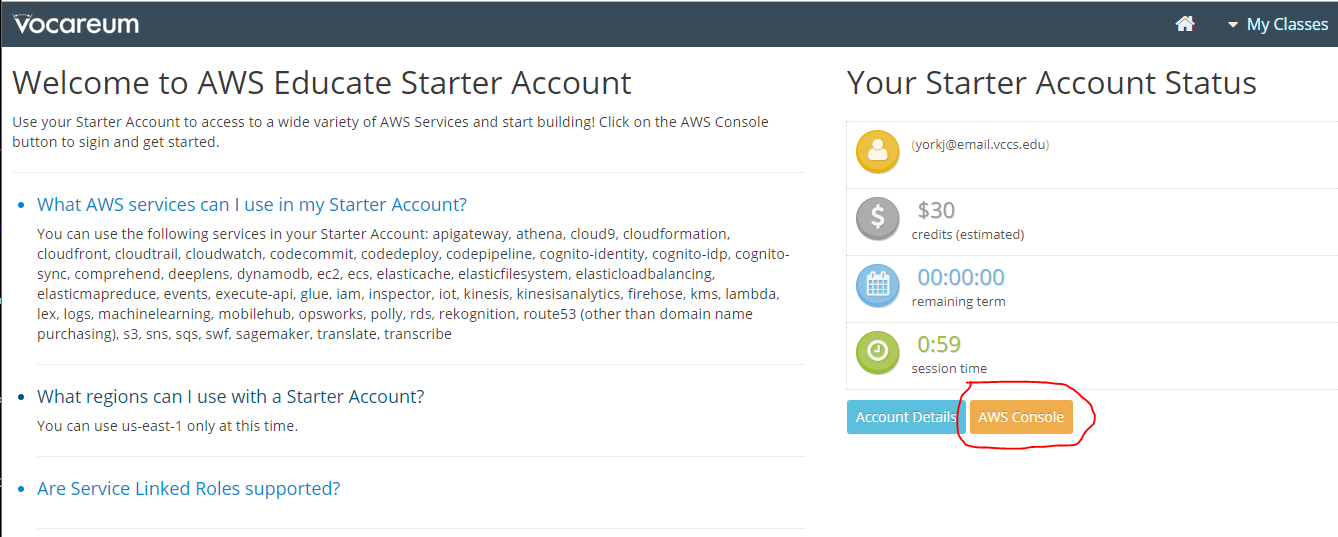
1. Your instructor should tell you ahead of time to expect the email, what it will look like, and what it will do.
2. Hover over the link and examine the URL that it will take you to, to make sure there is no trickery.

If you want to be careful, you could make a copy of one of your VMs and open the link in that VM. If the link turns out to be evil, delete the VM. That assumes you haven’t given the link your credit card number, of course…

AWS Educate will then ask you to set your password.  


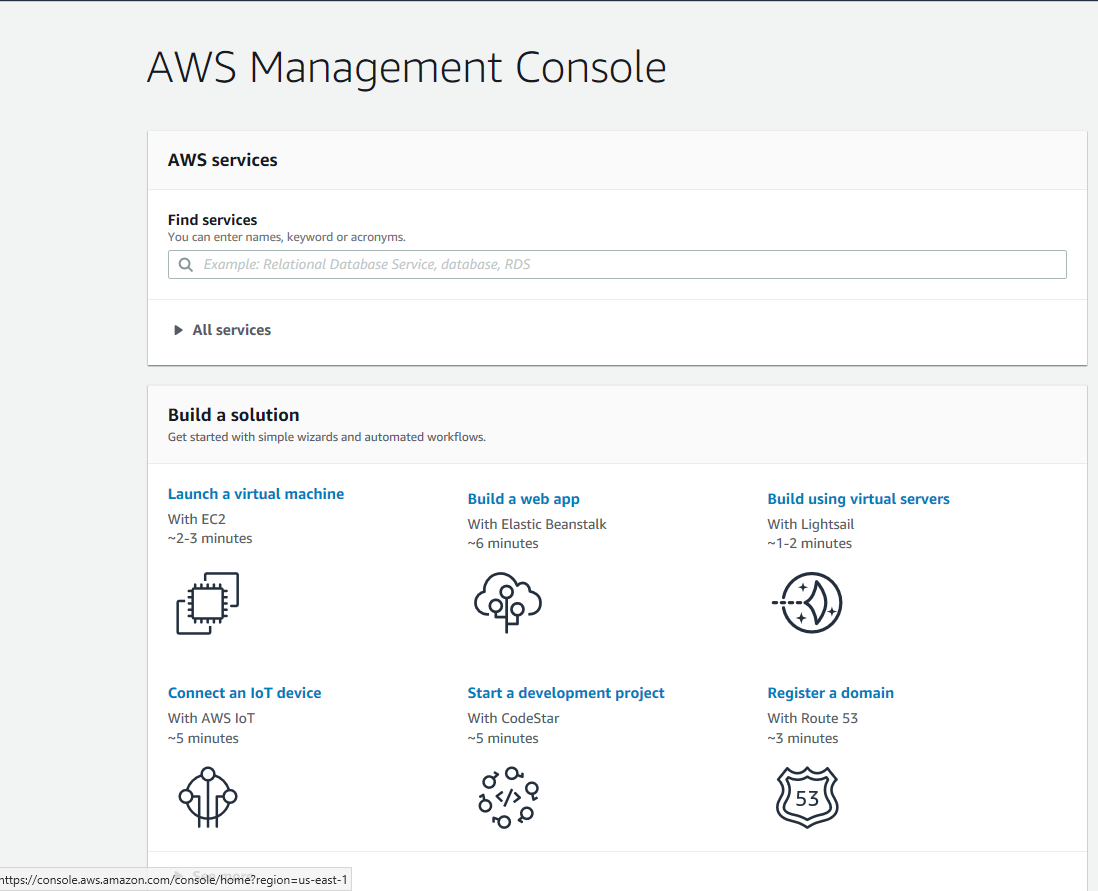
Then you will be able to use your AWS Educate starter account.  


Once you click the big orange button, you will be in your account. There are helpful frequently asked questions (FAQs) on the left side of the page. The orange button on the right side of the page will take you into the AWS console where you can create VMs, use storage, or use many of AWS’ 50+ products. The list of AWS products is long and confusing. This link can make it easier to understand. <https://www.expeditedssl.com/aws-in-plain-english>

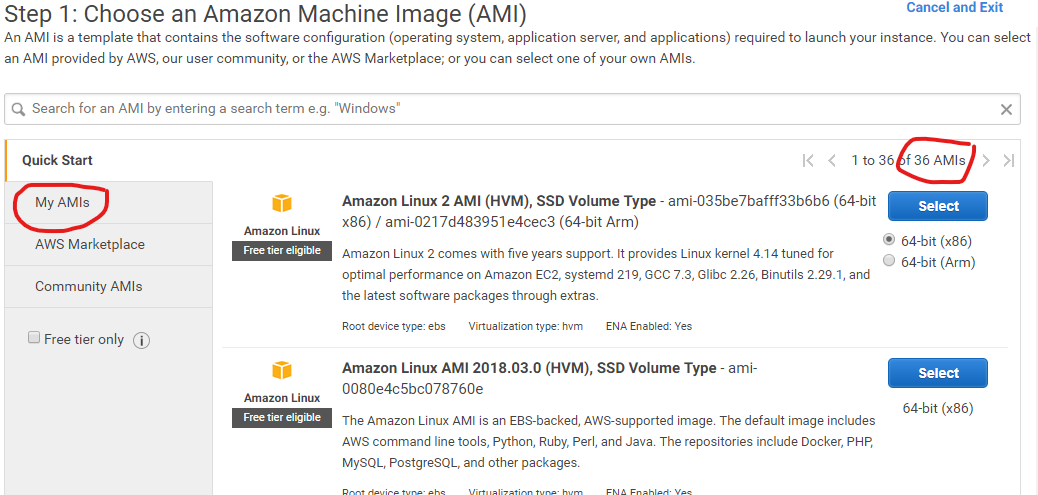
The orange button only works if you allow it past your pop-up blocker. If you click on it (AWS   
Console) and nothing happens, it is probably your pop-pup blocker at work.  


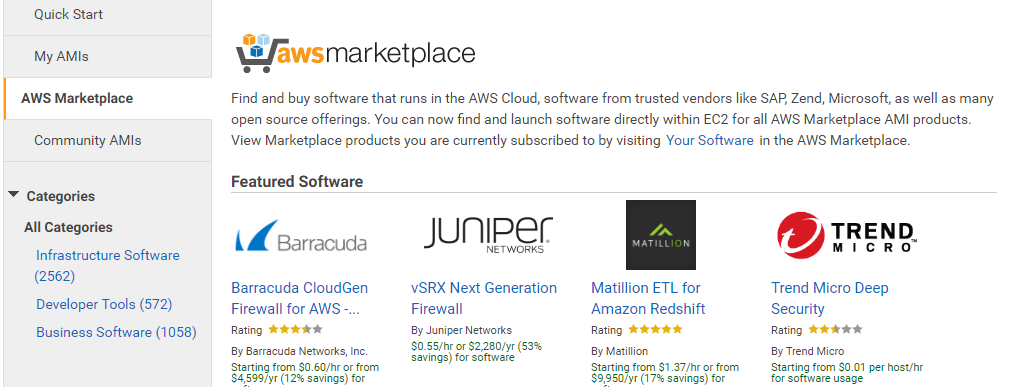
Once you are in the AWS Management Console you can get to work.

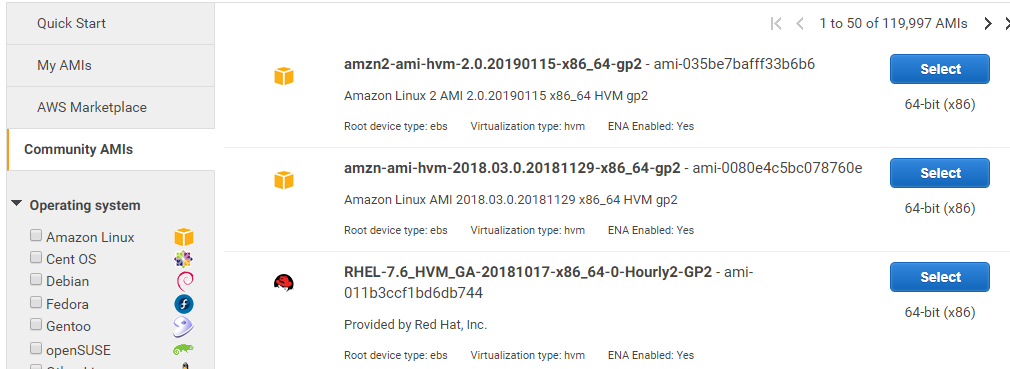
# Create a VM

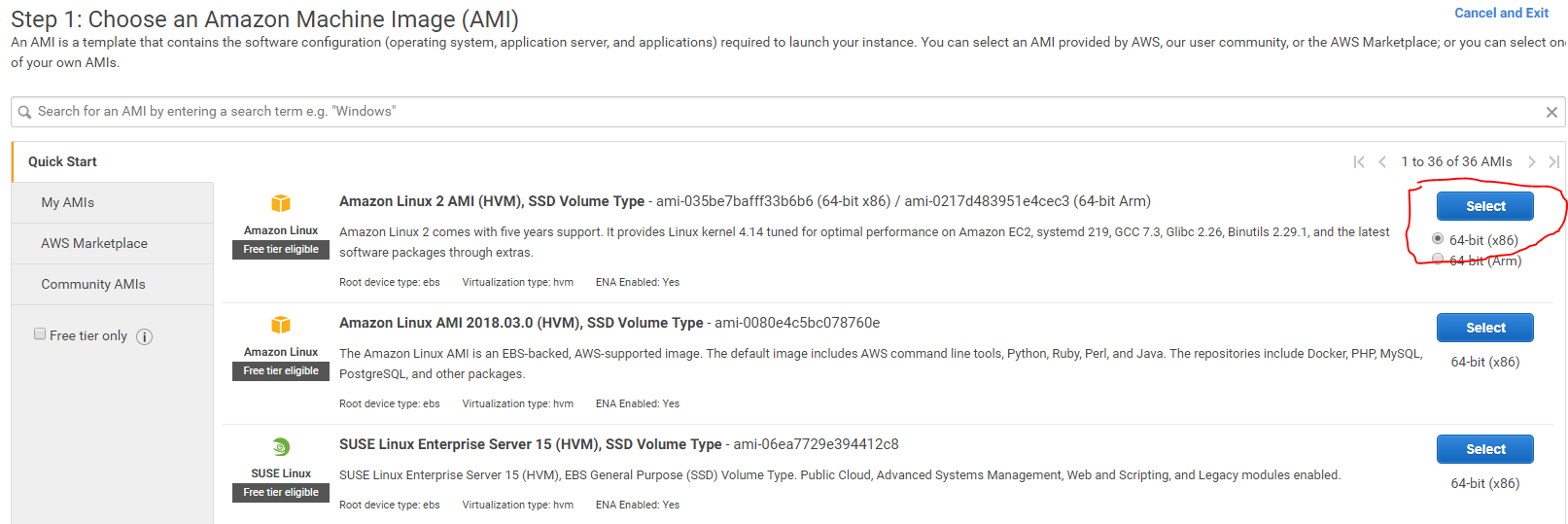
We will use the Elastic Cloud (EC2) service to launch a virtual machine. Click on “Launch a virtual machine” to get started.  


## Step 1, “Choose an Amazon Machine Image (AMI)”

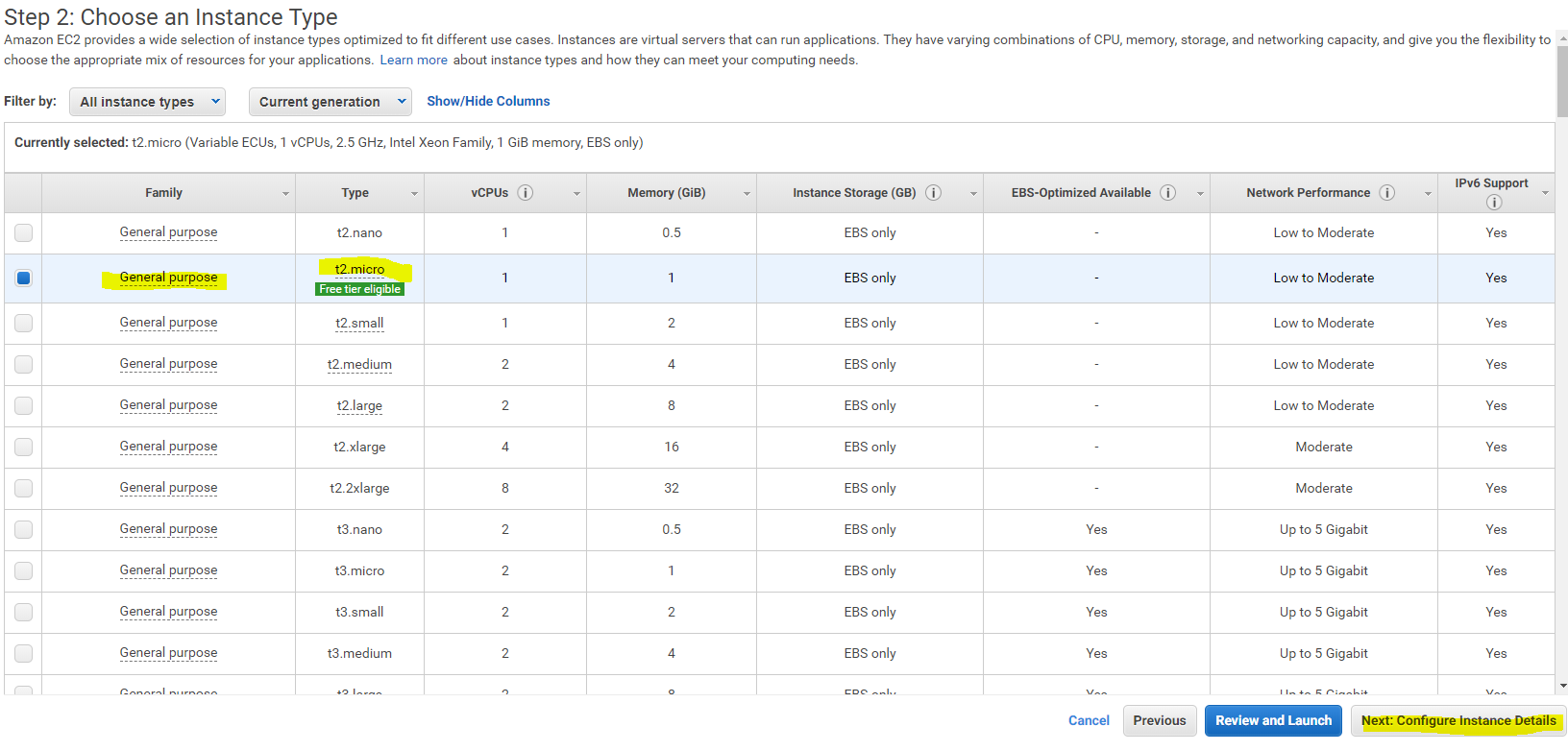
First, look at the images you may choose from. The “My AMIs” category has 36 basic images that are supported by AWS. Major distributions of Linux (Red Hat, Fedora, Ubuntu, SUSE) are available, as well as several versions of Windows server.  


The AWS Marketplace offers over 2500 images that are created (and charged for) by companies large and small.  


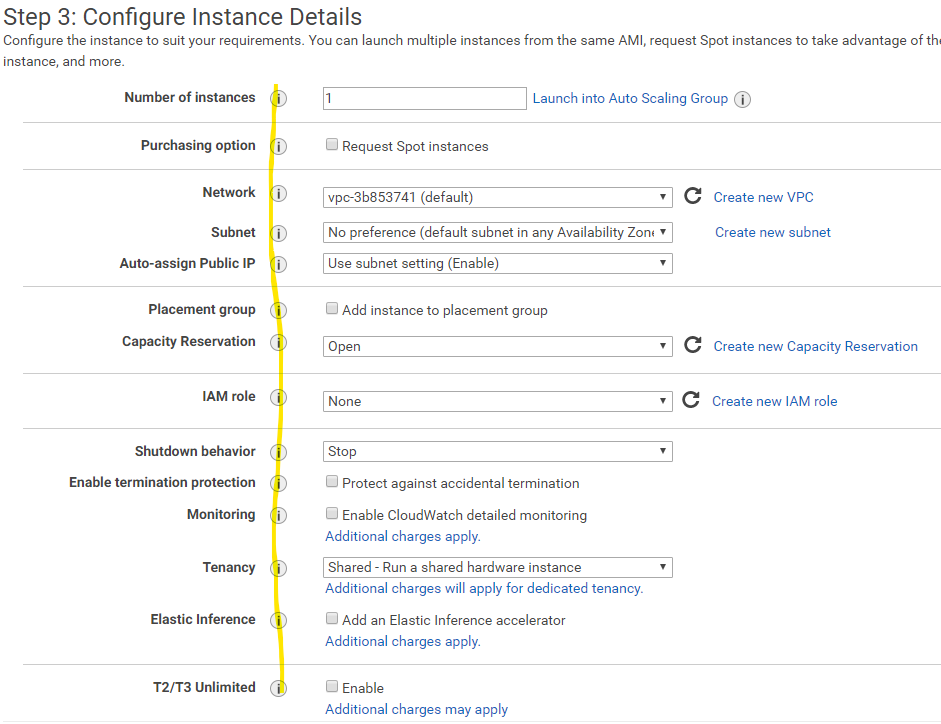
Finally, there are almost 120,000 community images. They follow the open source model and are produced by companies, user groups, and individuals (reputable and unknown alike.)  


We will use the Quick Start Amazon Linux 2 AMI. It is a RedHat based image. The lessons in upcoming lessons on web security and PHP are based on CentOS, which is closely related to RedHat. If you would prefer to use Ubuntu, you should be able to find one on this list.  


## Step 2, “Choose an Instance Type”

Amazon provides an amazing array of combinations of CPU instances, RAM, storage, GPU computing, and networking, mostly for a fee of course. You could easily build a powerful bitcoin miner, but you would probably pay more to AWS than you would gain by finding coins. Scan through the choices that are available. We don’t need much power, RAM, or storage, so the t2-micro choice is best for us.  


## Step 3, “Configure Instance Details

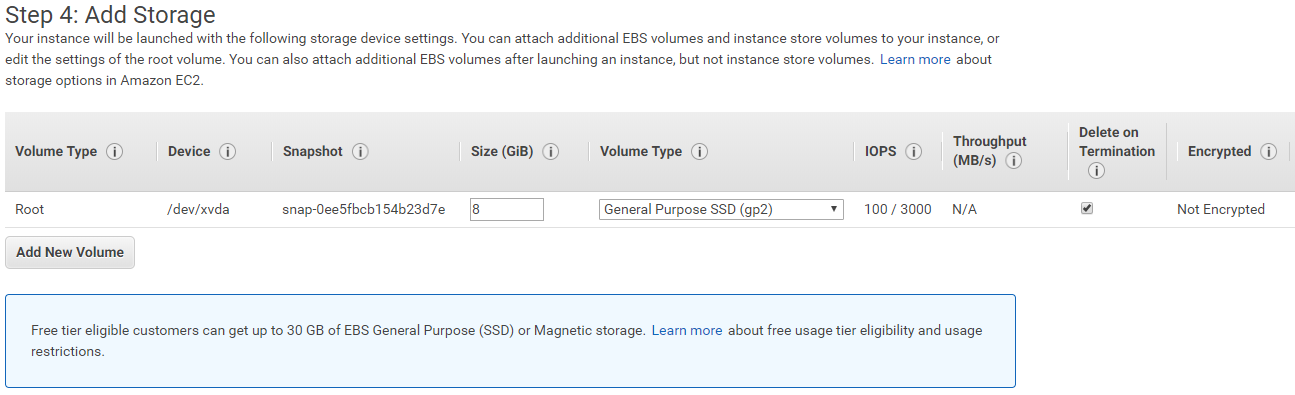
We don’t need to make any changes here, but it is good to get a general idea of the choices that are available. The Number of Instances choice is interesting. If you were building a massive cluster you could launch hundreds of instances; the bill would also be massive. Hover over the information buttons to see what choices are involved. There is nothing you need to change, so click Next: Add Storage.  


## Step 4, “Add Storage”

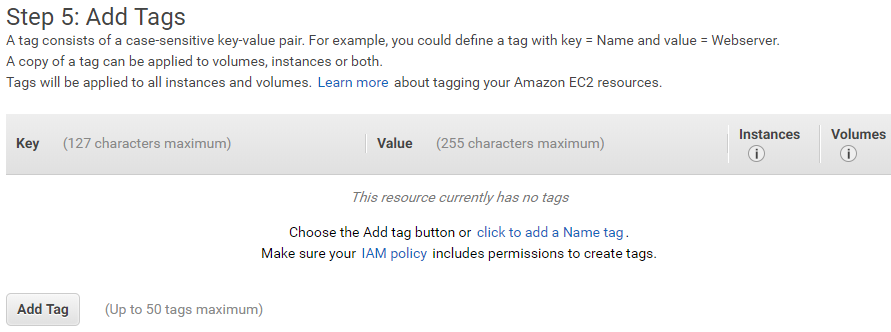
Hover over the information buttons to see explanations of the available storage options. In addition to the normal size and speed options, there are two interesting options. The Snapshot can save the data in your volume at the time you take the snapshot. If your image needed a lot of configuration work by you to get it ready, you can take a snapshot when the image is ready to deploy. Later, if the image is damaged or corrupted, you can quickly revert to the snapshot and be back in business immediately. Most modern storage and hypervisors have this capability, which can save you a lot of pain. In olden times you would have had to restore the data from a tape backup, or just start over.

A second interesting option is encryption. If an attacker compromises your VM, and if you have properly protected your keys, the attacker cannot steal your sensitive data. We spend a lot of time worrying about using HTTPS to encrypt data while it traverses the network (data in motion) but not so much time making sure the data is safe while it sits in the server storage (data at rest.)

We don’t need to make any changes, so click Next.



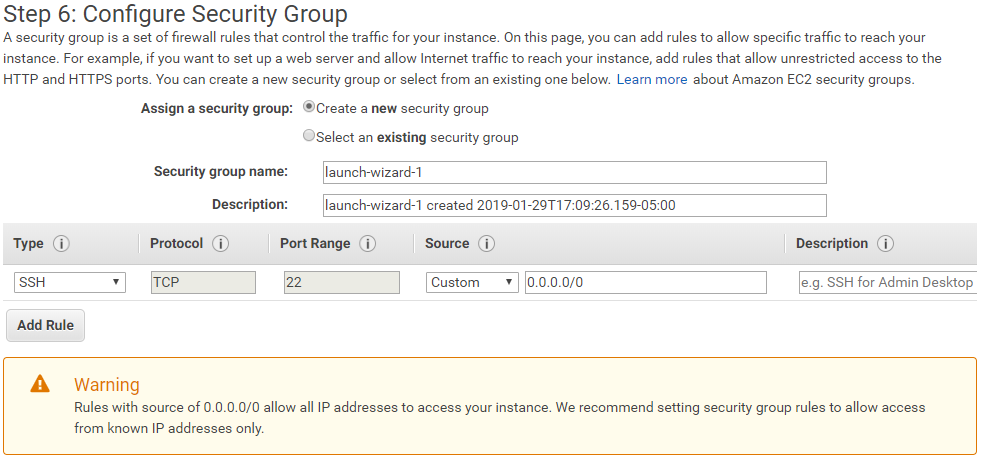
## Step 5, “Add Tags”

Tags are labels or markers that you can use to control large installations. The Identity Access Management (IAM, it was visible in step 3) system can hold many user accounts, and tags can be used to control which users have which privileges on which machines. We just have one user, so we won’t need tags.  


## Step 6, “Configure Security Group”

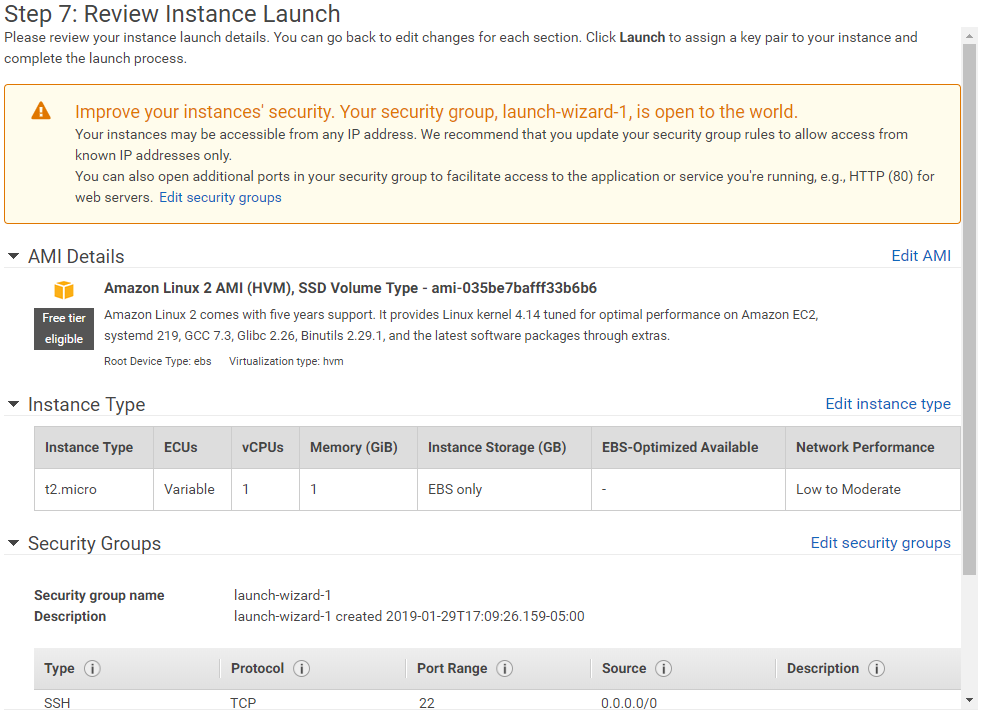
The security group controls the firewall settings for the instance. The default instance shown below allows SSH connections to the instance so that you can manage it. For our simple instance, we can rely on the public/private key authentication that SSH uses to protect our instance. Password authentication is disabled by default, which is good.

We could also configure other settings to allow outside access to our instance/VM. If we want it to be accessible to the world as a web server, we could add a rule to allow access to TCP port 80 from 0.0.0.0. Before we did that, it would be wise to restrict access to port 80 to just the IP address of our workstation, until the site was tested and secure. If we wanted to use it for some other purpose (for example as a netcat relay) we could open a different port and restrict it to specific addresses



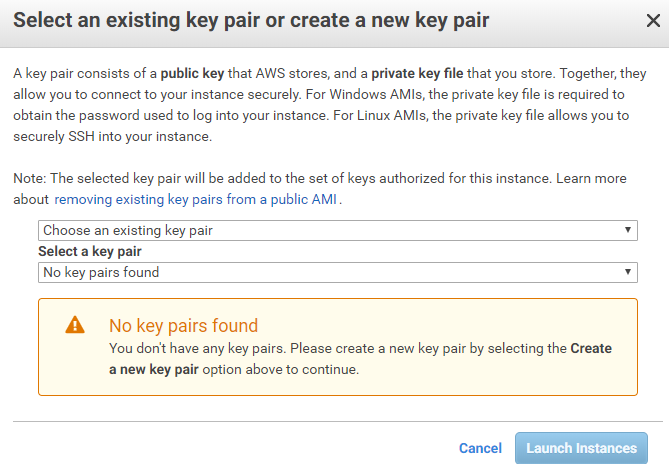
## Step 7, “Review and Launch”

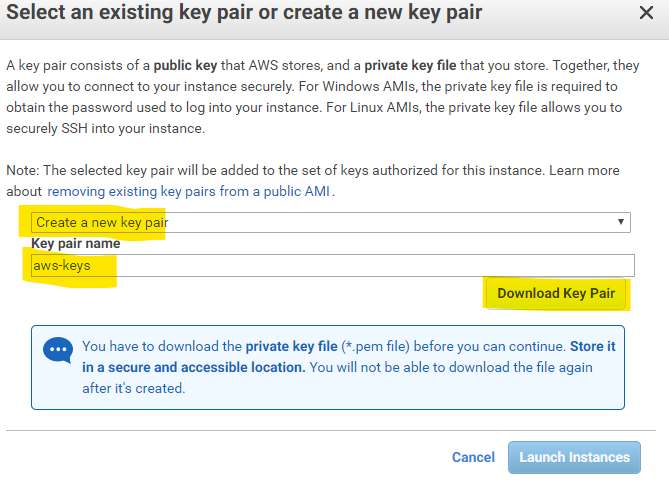
The last step allows you to review all your choices before you commit to launching the instance. We have the security warning again, but for our case we can leave the settings as they are. You could increase your security by discovering the public IP address that the SVGS network uses (not the IP address of your workstation) when it gives you a NAT connection to the outside and replacing the source address 0.0.0.0 with the SVGS public IP. If you wanted to be able to connect from home, you would have to add that address as well. If you want to be able to connect from your favorite McDonalds or Starbucks, you need to leave the address at 0.0.0.0.



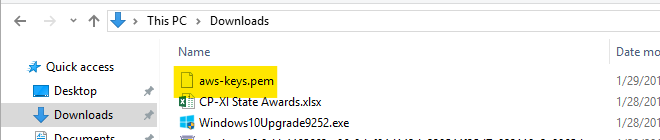
## Step 8, “Select a Key Pair”

This step is critical. If you make a mistake or lose your private key, you will not be able to connect to your instance. If you fail to properly secure your key, anyone who has access to your workstation will be able to use your key and log in as if they were you.

Since you haven’t done anything on AWS yet, you won’t have any key pairs stored on AWS.  


Use the drop-down list to select “Create a new key pair” and give the pair a name. Then click Download Key Pair. You will actually be downloading the private key. The public is available through your AWS console, however.  


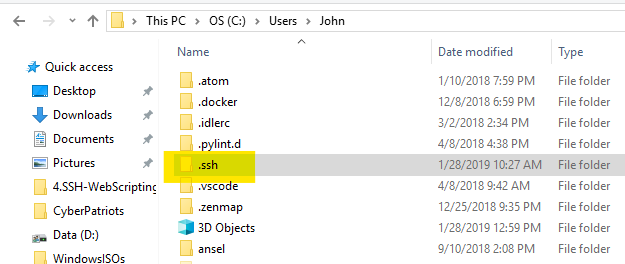
The key should appear in your Downloads folder (or /home/your-user-name/Downloads if you use Linux.)

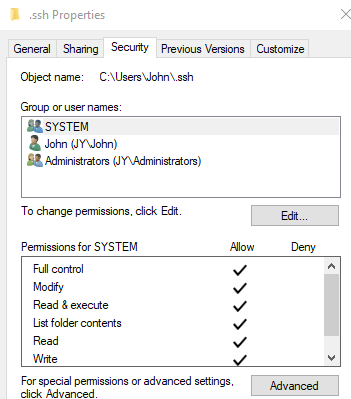


If you open the file in a text editor, you will see that it is an RSA Private Key. It is binary data, encoded with base64 so it can be downloaded as a text file. We will cover keys and encryption in great detail in later lessons.  

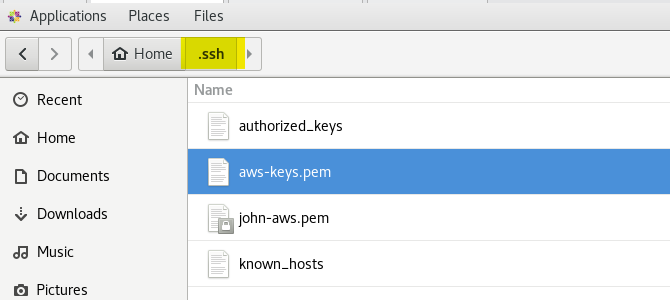

## Windows key Security

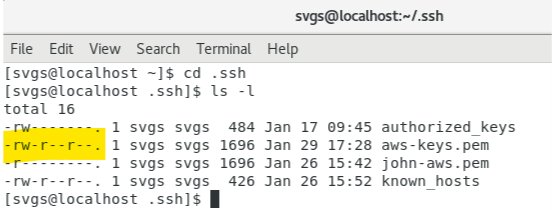
In order to have a common structure between our Windows and Linux hosts, create a directory in your home folder called “.ssh” (or just ssh if Windows complains about the period.) Move (cut/paste) the key file from your Downloads folder to your ssh folder.

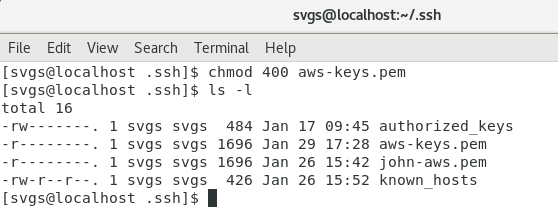


Check both the properties of the ssh folder and of the key file itself to ensure that they are similar to what is shown here. You absolutely do not want the Everyone group, or anyone other than your account, the SYSTEM account, and the Administrators group to be able to read your key file. If they can read your file, they can log in as you.  


## Linux Key Security

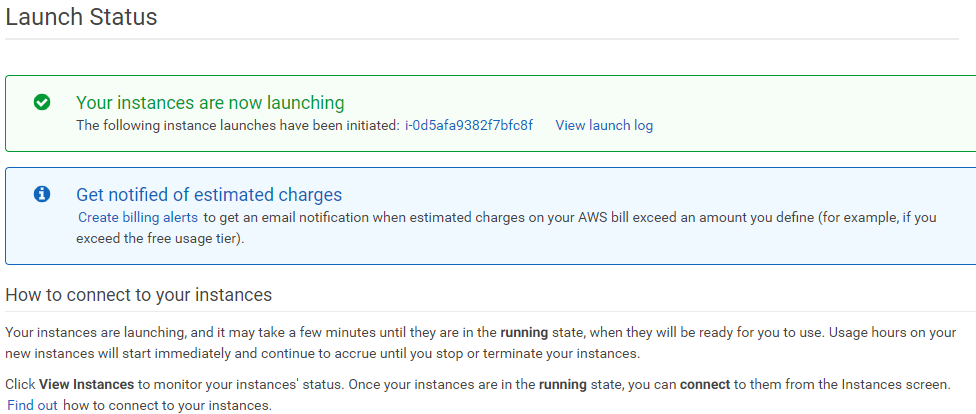
If you downloaded your key in Windows, you can copy and paste it to the .ssh folder in your home directory.  


Once that is done, check the file permissions on the files in your .ssh directory. Any keys should only be readable by your user account (-r-------). Again, if another user can read your private key file, they can use it to log in as you.  


Use the chmod command to give your user account read permissions, and no permissions at all to group and other (everyone).  
chmod 400 your-private-key.pem  


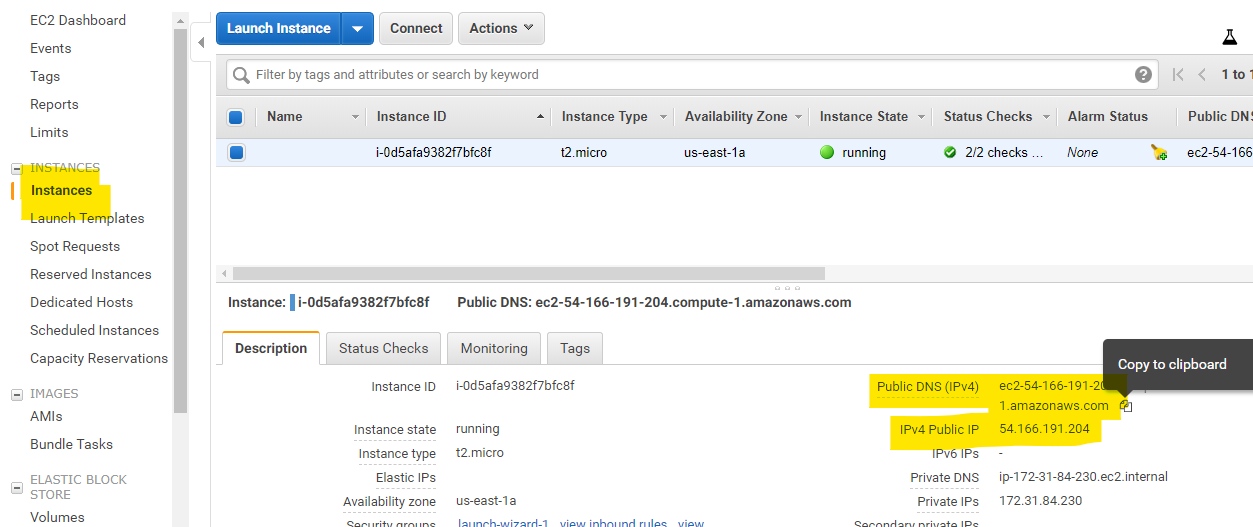
As a reminder, the authorized\_keys file holds the public keys of any users that are allowed to connect to your Linux machine with SSH with your account. If other users can write to this file, they can add their public keys and then use SSH to connect as you. The known\_hosts file contains the public keys of SSH servers that you can connect to.

## Launch the instance

Click the Launch button, and away you go.

## Check the details of your instance

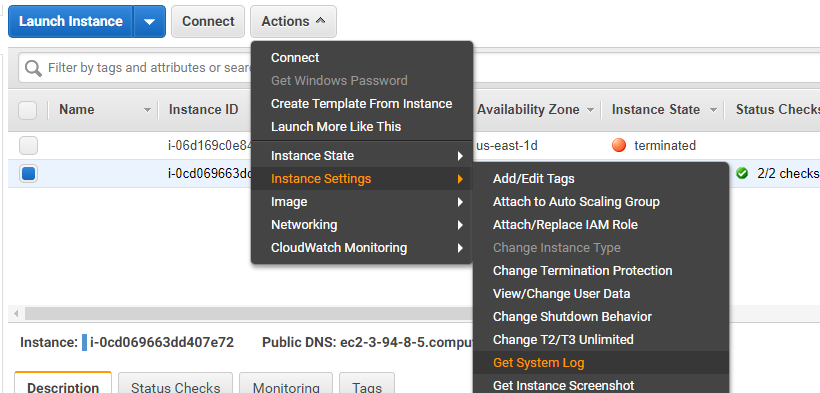
The Instances legend in the navigation tree on the left of the screen shows you the details of your instances. One of the important details we will need to connect to the instance is the DNS address. Since the address is long, AWS kindly gives you a copy button to put the address on the clipboard.

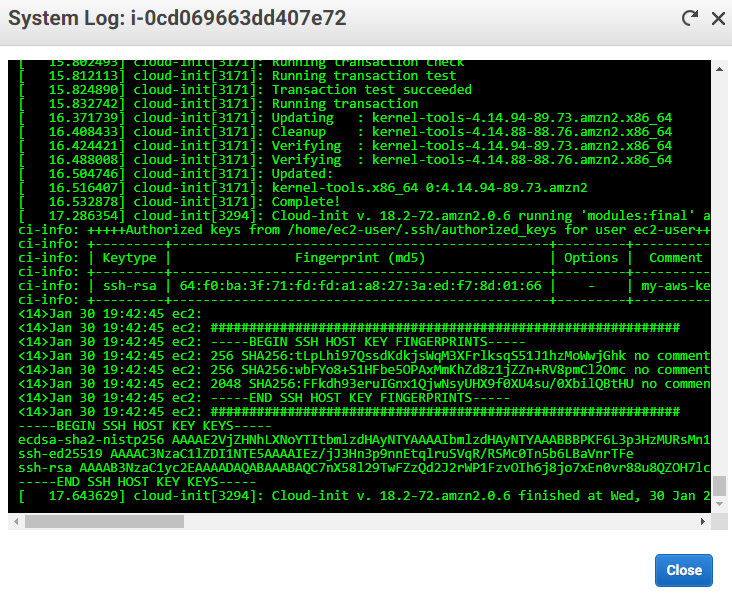


## Get the fingerprint of the server’s key

The key pair that AWS generated above is used to verify our identity to the instance. The instance also generates a second key pair that it uses to verify its identity to us. When our SSH client connects to the instance, it receives the public key of the instance. Normally the client checks the key against those saved in ~/.ssh/known\_hosts to verify that we are connecting to the correct host, and not to an imposter from a Man in the Middle (MitM) attack. The first time we connect, the key is not stored in the known\_hosts file, so the SSH client shows us the fingerprint of the key and asks us to verify that it is correct. One method is to hope that we are connecting to our instance and answer yes. Another method is to get the fingerprint of the key from the AWS Console and verify that we are connected to the correct host.

This procedure only works the first time the instance is run, so that is why we are doing it now. (That’s also why you see a terminated instance in the screenshot; I had to start over.) Select your instance, and then Actions > Instance Settings > Get System Log.



You should see a window like the one shown below. If you scroll to the bottom of the log, you will see the fingerprints of the keys that the server uses to identify itself. It would be good to save a copy of them.  


<14>Jan 30 19:42:45 ec2: #############################################################

<14>Jan 30 19:42:45 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----

<14>Jan 30 19:42:45 ec2: 256 SHA256:tLpLhi97QssdKdkjsWqM3XFrlksqS51J1hzMoWwjGhk no comment (ECDSA)

<14>Jan 30 19:42:45 ec2: 256 SHA256:wbFYo8+S1HFbe5OPAxMmKhZd8z1jZZn+RV8pmCl2Omc no comment (ED25519)

<14>Jan 30 19:42:45 ec2: 2048 SHA256:FFkdh93eruIGnx1QjwNsyUHX9f0XU4su/0XbilQBtHU no comment (RSA)

<14>Jan 30 19:42:45 ec2: -----END SSH HOST KEY FINGERPRINTS-----

<14>Jan 30 19:42:45 ec2: #############################################################  
  
There are other ways to verify the identity of the instance. This link is based on WinSCP, which we won’t use, but the information is good. <https://winscp.net/eng/docs/guide_amazon_ec2>

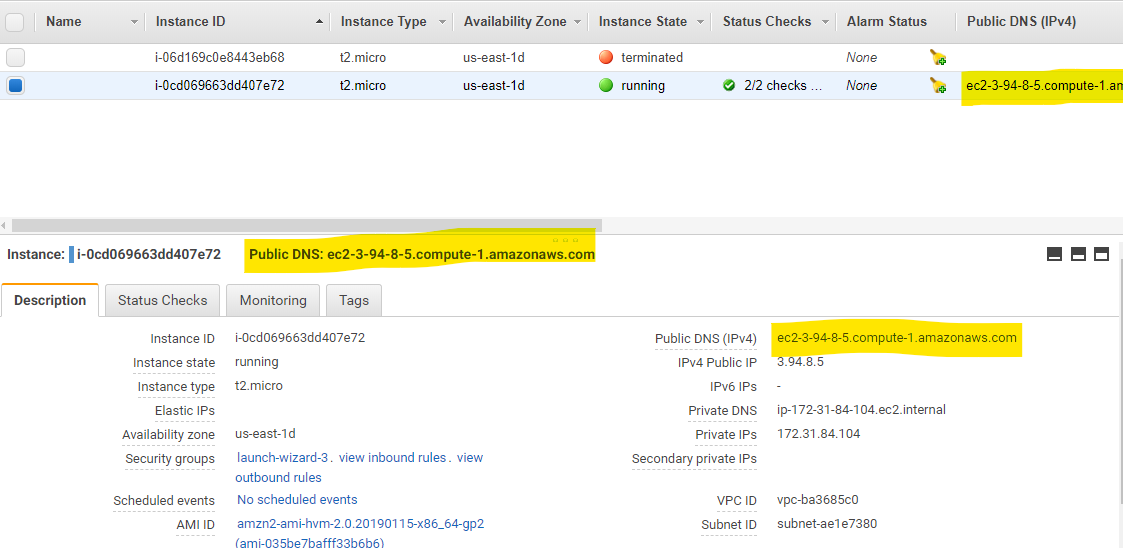
## Connect from your VM or host

To connect to an AWS instance, you need three things.

1. Your private key
2. User name
3. Address of the instance
4. The host key fingerprint, if you want to verify the instance’s key

Since you probably have more than one key, you will have to tell the SSH client which key to use when you connect to AWS. The -i option allows you to tell the client which identity file (key) to use.

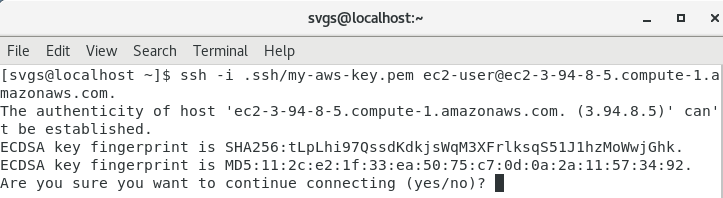
If you used the Amazon Linux AMI, the user name is ec2-user. For other images, Google  
AWS default username or see the WinSCP link above.

The public DNS name of our instance is found in several places on our AWS console.   


## Connect from Linux

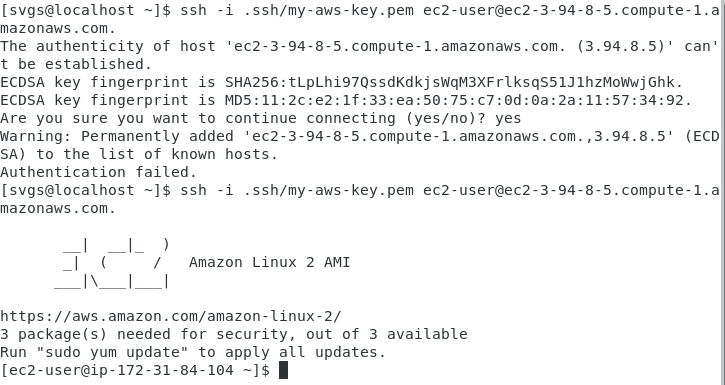
Use the following command to connect to your AWS instance. Of course, the key name and the DNS name of you instance will be different from what is shown below.

ssh -i .ssh/my-aws-key.pem ec2-user@ec2-3-94-8-5.compute-1.amazonaws.com

Since this is the first connection to this server, the public key for the server is not in the .ssh/known\_hosts file and the client warns us.   


Rather than hope we are not experiencing a MitM attack let’s check the fingerprints. From the AWS logs above, we have this.  
256 SHA256:tLpLhi97QssdKdkjsWqM3XFrlksqS51J1hzMoWwjGhk no comment (ECDSA)

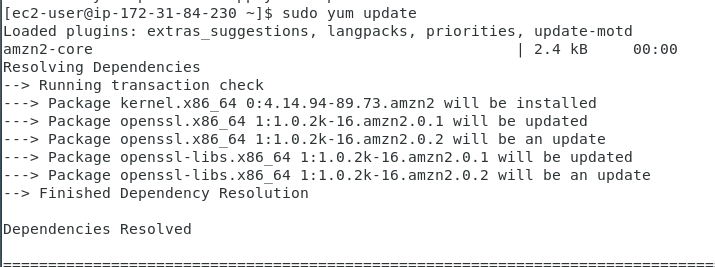
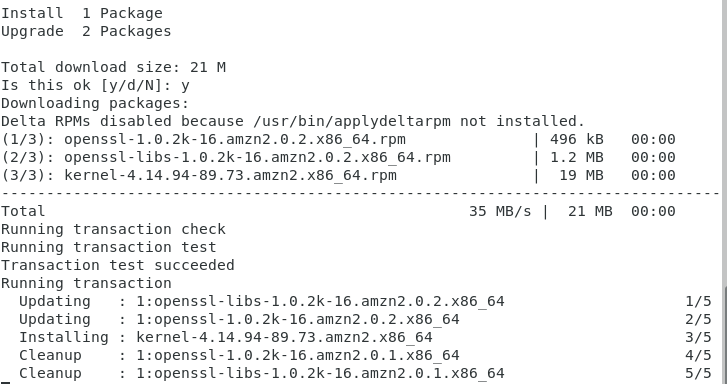
From the SSH client, we have this.  
ECDSA key fingerprint is SHA256:tLpLhi97QssdKdkjsWqM3XFrlksqS51J1hzMoWwjGhk.  
  
Match! We can continue.



(I don’t know why it failed the first time. Up arrow and enter worked.

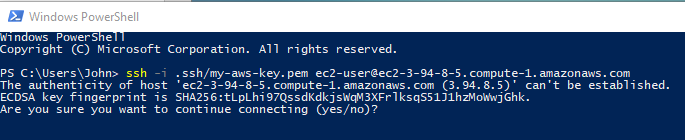
The server’s public key is now stored in the .ssh/known\_hosts file. If a subsequent login attempts to use a different key for the same address, SSH will warn us of an MITM attack and refuse to connect.

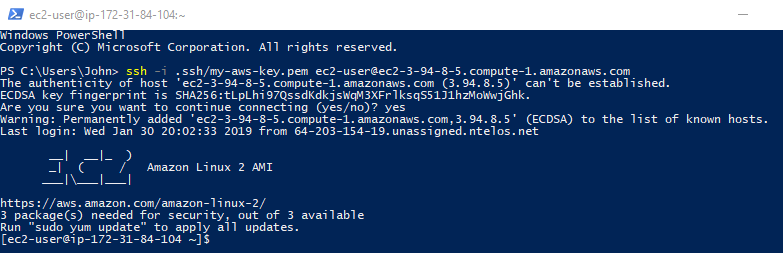
It looks like the VM/instance could use some updates, so we can do that. We have been using Ubuntu, and like most Debian-based distributions it uses apt or apt-get to do updates. The Amazon AMI instance we have is based on RedHat, which used the yum update manager.

  
<snip>  


## Connect from Windows

We used to use third-party tools like PuTTY and WinSCP to do SSH-related tasks in Windows. However, Windows 10 incorporated OpenSSH, the same SSH client and server that Linux uses, into the April feature update, version 1803, and above. Most of the tools we need, ssh, scp, ssh-keygen, and others are now available.

Assuming you have saved a copy of your private key in your home directory /.ssh or /ssh, you can easily connect to your instance.  


We have the same fingerprint as we did in Linux, so we’re good.  


# Hand In

There are a couple of things that are interesting enough for me to make you look at the configuration files in the instance.

1. If you updated the instance with sudo yum update, you may have noticed that it did not require you to enter a password. Why did it not require a password? (Hint: Look at the files /etc/sudoers and /etc/group.
2. Does the SSH server on your instance allow password authentication? What line allows the use of public key authentication? (This is kind of a trick question. Commented out lines in a configuration file are often there to show you what the defaults are, but do not change configuration.) (Hint for both questions: Look at /etc/ssh/sshd\_config.