# 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 SANS recently published excellent instructions on how to create a VM on AWS (<https://pen-testing.sans.org/blog/2017/12/10/putting-my-zero-cents-in-using-the-free-tier-on-amazon-web-services-ec2>.) The basic concepts you learn here will apply to any cloud service provider.

## Create an Account

Create an AWS account following the instructions in the Pen Test Blog, link given above. The account we will use is free for the first year, but AWS (and most providers) require a credit card to reduce malicious use of their services. Your instructor has pre-paid credit cards for your use. In filling out the application, use your information and school email address, until you reach the credit card information. Use the information your instructor gives you for the credit card section.

## Create a VM

When you reach Step 1, “Choose an Amazon Machine Image (AMI),” choose the basic Amazon Linux AMI at the top of the list. The blog post uses an Ubuntu image, but the Amazon Linux AMI is very similar to the CentOS installation we’ve been using in class.

Save the private key that AWS gives you. Don’t lose it, as you will need it to connect to your VM. Practice connecting to the AWS VM using both your Windows host, and your CentOS VM--it is good to be able to use either method.

## Connect from your CentOS VM

Once you have your private key, move it to the .ssh directory in your home directory. If you registered with a web browser on your Windows host, you can copy/paste the file to the file browser on your CentOS VM. 

Check the permissions on your keys. They should not be readable by any other user, as they would be able to log in to AWS (or whatever the key is for) as you, with all your permissions.

 

The permissions for JohnInstance.pem are bad and need to be fixed. 

The known\_hosts file stores the public keys of the servers you have connected to. The SSH client will give you a warning if the public key of a server you connect to changes or is not in the file. Read access for other users is not a problem.

Since we now have more than one key, we will have to tell the SSH client which key to use when we connect to AWS. The -i option allows us to tell the client which identity file (key) we wish to use. If you used the Amazon Linux AMI, the user name is ec2-user. The public DNS name of our instance is found in several places on our AWS console. 

ssh -i .ssh/john-svgs.pem ec2-user@ec2-35-135-154.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.



We answer yes, and then login. It seems that the authentication fails the first time after saying yes to continue connecting.



## Connect from Windows/PuTTY

Follow the directions in the Pen Test Blog, with just two exceptions. First, the user name for the Amazon Linux AMI we are using is ec2-user, not Ubuntu. Second, to be consistent with Linux (and to be ready when OpenSSH in Windows 10 is out of Beta), create a directory in you home directory called “.ssh” and store your keys there. Best practice in Linux is to set permissions so that no other user can read your private keys. We need to do the same for Windows.

Select properties of the .ssh folder, then click the Advanced button. Check the access that is allowed to different users in the Advanced Security Settings and make note of any that should be removed. SYSTEM, Administrators, and your user should have access, and no others. Click the Disable Inheritance button to prevent other users from gaining read permissions through inheritance. Click “Convert inherited permissions into explicit permissions…” and click Apply. If there were users that should not have access, go back into the security settings and delete them now.



## Windows and OpenSSH

In December 2017, Microsoft added OpenSSH to Windows 10, in Beta. OpenSSH is the primary SSH client on most Linux distributions, so now the same interface is available on both OSs.

Note: Right now, MS Beta OpenSSH only supports ED25519 keys, and AWS generates RSA keys. To use the MS OpenSSH, you would have to generate keys in MS OpenSSH and upload them to the AWS instance. For the time being, we will not use MS OpenSSH.

 To install OpenSSH, follow the directions given here <https://blogs.msdn.microsoft.com/powershell/2017/12/15/using-the-openssh-beta-in-windows-10-fall-creators-update-and-windows-server-1709/>.

If you have installed your keys in a .ssh directory in your home, you should be able to use the ssh command just as you would on Linux.