LAB 1.5 GETTING STARTED WITH AWS WINDOWS SERVER 2012 R2 INSTALL LAMP STACK ON AWS UBUNTU

PRAMOD NEPAL

02/08/16

CSCI 5417

EAST TENNESSEE STATE UNIVERSITY

PURPOSE

The purpose of this lab is to launch two instances of Windows Server 2012 on AWS, connect to each using Remote Desktop Protocol (RDP), change the default password, and for the Ubuntu instance that was created in the last lab, connect using ssh and install LAMP stack using tasksel utility. Finally, using a browser verify that the Apache server is running.

MATERIALS

- Lab Instructions
- Web browser
- PuTTY
- AWS console

PROCEDURE

Windows Launch and Server Setup

Initially, we logged into the AWS console, that was setup in first lab. Click "Launch Instance" from the EC2 dashboard. For the base image we select "Microsoft Windows Server 2012 R2 Base" image. Then the choose the VPC we created in last lab (lastnameVPC). i.e.; NepalVPC. During this step the public subnet that was created in last lab should also be selected, because that is the only one that has been created on AWS so far. To automatically get an IP address "Use default setting (Enable)" is chosen. As for the static IP address value of 173.1.1.20 is set.

During the "Add Storage" section default values is selected. We then click on "Next: Tag Instance" button. Here the instance for the first image is selected as lastnameWindowsDC01, which in my case would be NepalWindowsDC01. Next, the security group is configured, in which the security group created in last lab is selected, which is called NepalSG. Since, we will connect to this server using RDP, we need to make sure the security group allows connections from port 3389 (RDP). Since this was added in last lab, no new changes needs to be done. We can add rules (if we need one), by clicking on "Security Groups" from the EC2 console. Then the instance can be launched. For that we press "Review and Launch" and "Launch" button sequentially.

A dialog box showing the private key that we created in last lab is displayed. It should select “4417key” that we created in last lab. We acknowledge and press “Launch Instances”.

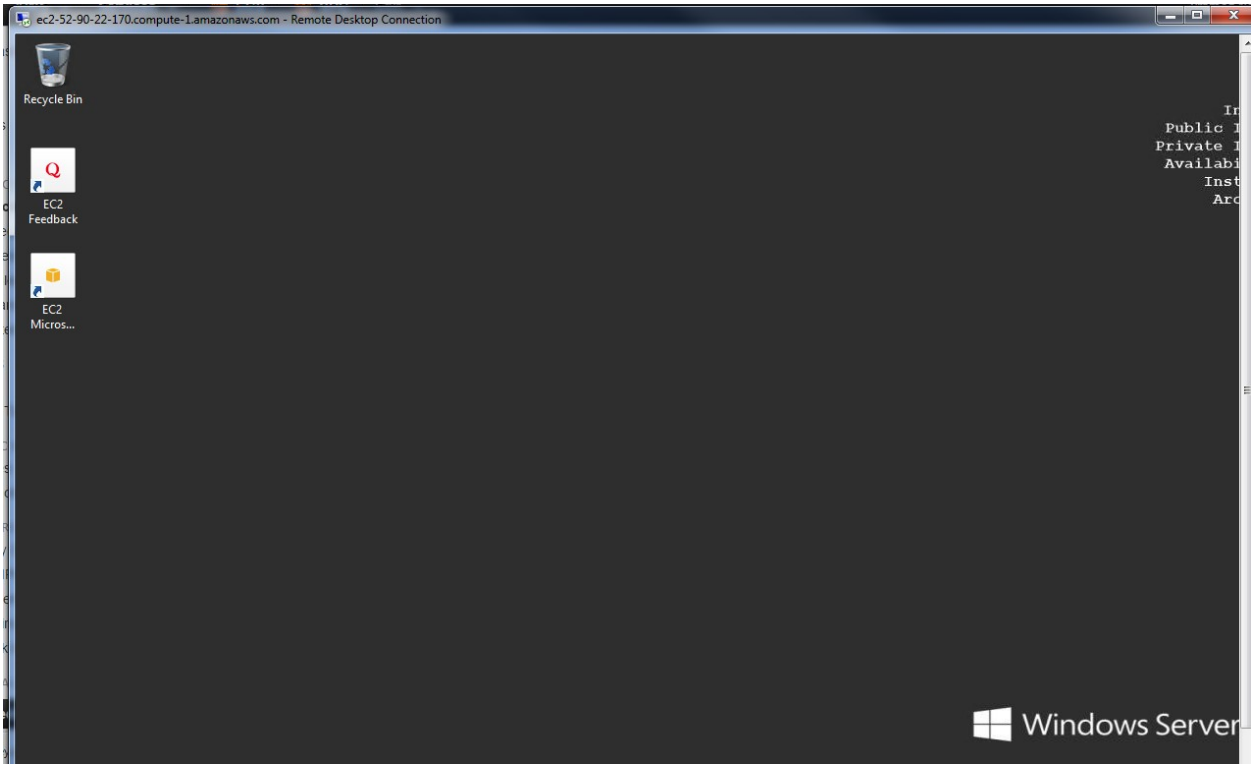
Following all of these steps described above, we created another instance. In my case the instance name is named NepalWindowsMS01 and the IP address is set to 173.1.1.30. We will be using these images in later labs. To launch this instance as well “View Instances” button is pressed. The names we choose represents “Member server” for MS and “Domain controller” for DC suffixes to the instance names. In the view instances page, the instances can be renamed if they didn't follow these naming conventions. Two of my instances are names **NepalWindowsMS01** and **NepalWindowsDC01** as described above.

For easier login access we plan to change password of both windows instances. To change the password, first we go to the EC2 console, right-click on one of the windows instance. E.g.; NepalWindowsDC01 and select “Get Windows Password”. We then browse the 4417key.pem file we saved in the last lab. We then click “Decrypt Password”. We can note the password and Public DNS into a text editor. When connecting to the server, any one of the Public DNS or IP address can be used.

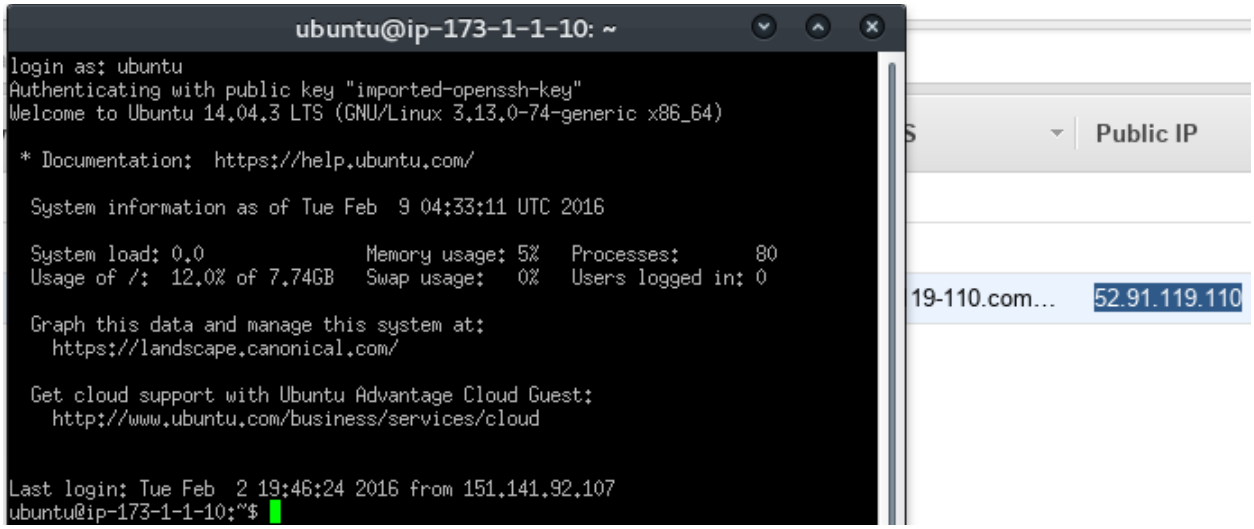
To connect to the windows servers “Remote Desktop Connection” application that comes pre-installed on windows is used. We paste the Public DNS or use the ip address displayed in the View Instances window (once the instance is started). Before connecting we click on “Show Options”, and enter “Administrator” for the “User name” and click “Connect”. On the security warning dialog we selected “Yes” and then type the temporary password that we copied into a text editor. I repeated this process with the NepalWindowsMS01 instance as well.

Once connected to the running instance password can be changed by, pressing “Ctrl-Alt-End” key and selecting “Change a Password”. We type in the temporary password and type a password that we desire to use for the lab. For the purpose of this lab we selected “**Passw0rd!**”. After changing the password the server was shutdown.

Install LAMP stack and test apache on Ubuntu



Using the EC2 dashboard, I changed the name of the Ubuntu instance, we created in previous lab to **NepalUbuntu01**. To start the server, we can right-click on the instance and click **Start** from **Instance State** menu. Once the public IP is displayed IP address is copied, so that we can connect using Putty. We then paste the IP address to the address field.. We then load the 4417key that we created in last lab by clicking on the + sign next to “SSH” pane, clicking on “Auth” tab and selecting “Browse”. To connect to the Linux server we can press “Open” button. We use “ubuntu” as the user name.



Next, step on the lab was to install LAMP stack. For that, we will used a handy utility called tasksel. Once connected to ubuntu server, we type "**sudo tasksel**" into the terminal. We then select the "LAMP Server" checkbox and press "Ok" to install the package group. This process installs Apache, MySQL and PHP. During the installation, we also need to enter MySQL password.

To confirm that the services (Mysql and Apache2) are running we can use "**sudo service –status-all**" command. A + sign indicates that the service is running. The sign next to mysql may show ? instead of +. In that case we can make sure mysql is properly started by once again restarting it with "**sudo service mysql restart**" command. We can also issue '**sudo initctl list | grep "mysql"**' to see if mysql has started. From ubuntu 15.10 onwards the defaul init system (**Upstart**) is replaced is replaced with "**systemd**", which has equivalent commands to start, stop or inquire about a service. E.g., "**sudo systemctl status mysqld.service**".

Next, we intend to test the setup of Apache 2 server. If the service is started, it should display a page at default http port of our server. We can access this page by typing the public IP address of our Ubuntu instance into the address field of a web browser. As it turns out, we don't get to that page. We need to add HTTP service just like we added SSH and RDP in our security group's rule. For that we can go to "Security Groups" in AWS, select the security group that we created in previous lab, go to "Inbound" tab and click on "Edit". We then click "Add Rule" and for "Type", select "HTTP". We also made sure that Source is selected to "Anywhere". After we save, when you revisit the page in the browser, it should

return a page. Finally, we shutdown the instance using "**sudo shutdown now**" from the command line. We could have used **apt-get** command to install MySQL, Apache, and PHP separately, but we made the process easier by using tasksel. Cloud service providers like **Amazon** and **DigitalOcean** offer free images with LAMP pre-installed.

OBSERVATIONS

The first part of the lab demonstrated the setup instance of Windows Server 2012. We created one image for the member server named lastnameWindowsMS01 and another for the domain controller named lastnameWindowsDC01. We renamed the images accordingly using our last names. The image setup process was similar to what we did in last lab. In this lab the IP addresses were different than that used in last lab. We used 173.1.1.20 and 173.1.1.30 for the two window's images. After launching the instances, we noted the temporary password and changed them once the images were loaded using Ctrl-Alt-End keys. We used the "Remote Desktop Connection" application to connect to the server. We could connect to the server using both Public DNS or the IP address by copying them into the address field. As for the user name we used "Administrator". After connecting and changing the password we shutdown the instances both from inside windows and from the view instances page in AWS.

For the second part of the lab we used the ubuntu instance that was created during previous lab. We renamed it to lastnameUbuntu01. We connected through ssh using Putty. We used the key that was created in last lab. After login in using "ubuntu" as the user name, use used **taskel** utility to install LAMP server. LAMP server package installs MySQL, PHP, and Apache. After the installation, we first tried to connect to the server's http port (80). However, since we had not added this port in the security group, we were not able to connect to the server. We then added http in the sercurity group's rule and then we were able to connect to the server using a web browser. We also made sure that the services like MySQL and apache were started after the installation. We learned several commands to administrate the services. Finally, we shutdown the server from command line using "**sudo poweroff now**" command, and then from the AWS console, as we did for other instances. Although we used tasksel to ease the installation process, we could have installed the package individually. If we had to automate the installation process in many servers, that might have been a better option.

As noted, Windows Server 2012 R2 has the functionality to enable and disable the GUI. GUI is useful for point and click activities. Also if the options are easy to find, they are intuitive for most people to get their job done fairly easily. However, GUI is not always the most desirable feature in server admin domain. Many server admins prefer Non-GUI interfaces E.g., Terminal access to the server. Non-GUI interfaces have several advantages. E.g., It saves the bandwidth that could be used for more productive work E.g., Managing more servers than wasting on Graphical display. If a server admin is knowledgeable enough he/she can script several commands to happen simultaneously or concurrently (if tasks are non-dependent). It makes the operation relatively fast (as someone does not have to go through the same point can click chore), and the work flow can be duplicated across several other server's without having to click through all the interfaces in those operating systems.

However, as hinted above not all task are intuitive enough to do through the command line. One has to possibly memorize lots of commands to even write an effective script that does several things. Sometimes it is not worth digging through several manuals just to get few tasks done. In such situations, where task is easy enough to figure out using a graphical interface, firing up a GUI might be more beneficial. Also, windows was not really made up for working through the command line. Therefore, it is much easier to guide one through a GUI in windows than to go through weird commands that many have not seen before.

For the windows forest, I think it will probably be better, if we let Amazon handle the spikes. It is costly to keep up servers just to predict spikes. We could be saving money, when we don't have need for those servers. If there is a spike Amazon will handle that for us. I believe this is the best option in managing the forest.

To install LAMP on 10, 50 or 100 machines, the best route would be to use a pre-installed images provided freely by Amazon or DigitalOcean. If the service provider don't not have such images, they could let the individual clients create custom images. If that is also not an option, the next best thing would be to create a script that connects to all the instances through ssh and install the packages using the good old apt-get command.

As for the course, reviewing some of the Linux command will be helpful. Although, I am fairly comfortable with Linux commands, there are many students that need to use these commands, if they are to select a carrier that involves system administration. Moreover, I will know many new commands that are specific to system administration. Even for a programmer, these days many web programming shop expect the programmers to be knowledgeable enough to use several Linux services and commands.