Lab 1.5
Getting Started with AWS
Windows Server 2012 R2
Install LAMP Stack on AWS Ubuntu

East Tennessee State University
CSCI 4417/5417

Jack Ramsey, Lecturer

Purpose

Launch two instances of Windows Server 2012 R2 on our AWS accounts. Connect to each using Remote Desktop Protocol (RDP), and change the default passwords. These servers will be used in future labs.

Rename our UbuntuTest instance. Connect via secure shell (SSH) using PuTTY. Install Linux Apache, Mysql, and PHP (LAMP) stack using **`tasksel`**. Confirm Apache2 server is running by connecting via web browser.

Materials

- Lab Instructions
- PuTTY (already installed on lab computers, available on disk from instructor for personal PCs)

Conventions

- Following best practices (suggested by Amazon) we will prepend our last name to each resource we create. For example, my Windows Server 2012 R2 domain controller will be named RamseyWindowDC01. In the following instructions, '*lastname*' will appear whenever this is the case, e.g., *lastname*WindowsDC01.
- Remote Desktop Protocol (the Windows application that uses this is Remote Desktop Connection) = RDP
- Secure shell = SSH

Procedure

Windows (Launch and name servers)

1. Log in to the Amazon Web Services console (https://aws.amazon.com/)
2. Navigate to the Elastic Cloud Compute (EC2) dashboard
3. Click on 'Launch Instance'
4. Select Microsoft Windows Server 2012 R2 Base – ami-9a0558f0

Figure 1: Select Microsoft Windows Server 2012 R2 Base - ami-9a0558f0

5. Step 2 – Choose an Instance Type. 't2micro' should already be selected. If not, choose it. Click 'Next: Configure Instance Details'
6. Select *lastname*VPC as the Virtual Private Cloud
7. Your public subnet, being the only one we've created so far, should be selected automatically
8. Note that 'Use default setting (Enable)' is selected for Auto-assign public IP
9. Assign an IP address of 173.1.1.10

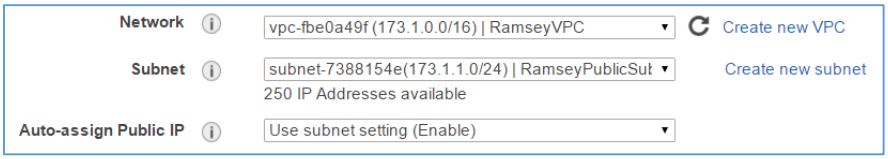

Figure 2: Configuration for instances

10. Click on 'Next: Add Storage'
11. Again, the default value for storage is ok. Click on 'Next: Tag Instance'
12. Name the instance *lastname*WindowsDC01. Click on 'Next: Configure Security Group'
13. Choose 'Select existing security group'
14. Select the security group you created last week, *lastname*SG. Make sure there is a rule allowing connections from Port 3389 (RDP). If not (last week's lab included instructions for adding it), add it now (You'll have to add the rule by clicking on 'Security Groups' from the EC2 console. You can finish launching the instances and modify the security group while they're launching)

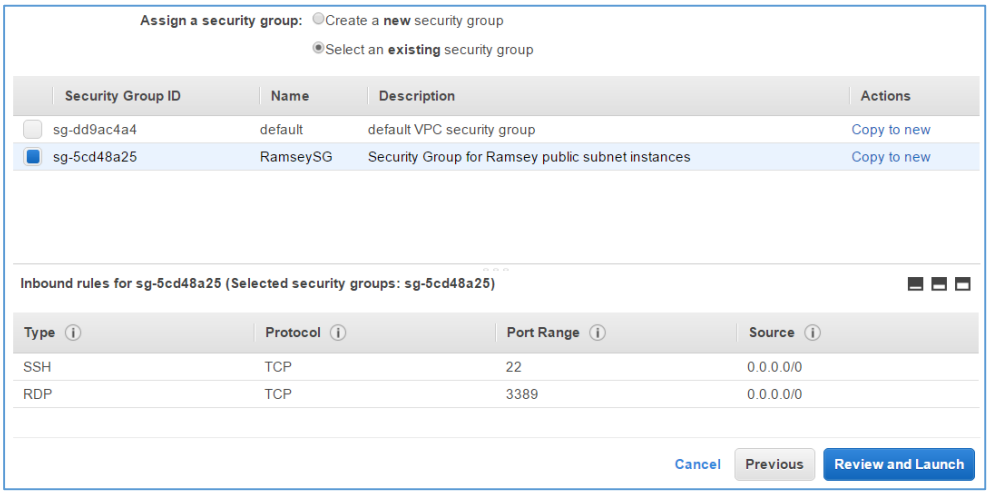

Figure 3: Security group configuration

15. Click 'Review and Launch,' then 'Launch'
16. In the dialog that displays, '4417key' should already be selected as the private key for the instance
17. Click on the acknowledgement and then 'Launch Instances'
18. Repeat Steps 3-17 with the following changes:
    a. IP address: 173.1.1.20
    b. Name: *lastname*WindowsMS01
19. Click on 'View Instances'
20. Hover the mouse cursor over the first instance's 'Name' field. Note that a 'pencil' icon appears at the right. Click on the icon
21. Name your first instance *lastname*WindowsDC01 (Domain controller). Either press Enter or click on the check mark
22. Name your second instance *lastname*WindowsMS01 (Member server).
23. AWS Windows servers take about five minutes to get up and running, so we'll return to them in a bit

Ubuntu (Install LAMP stack and test)

1. On the EC2 dashboard, change the name of your Ubuntu server to *lastname*Ubuntu01
2. Right-click on the instance, hover over to 'Instance State' and click on 'Start'

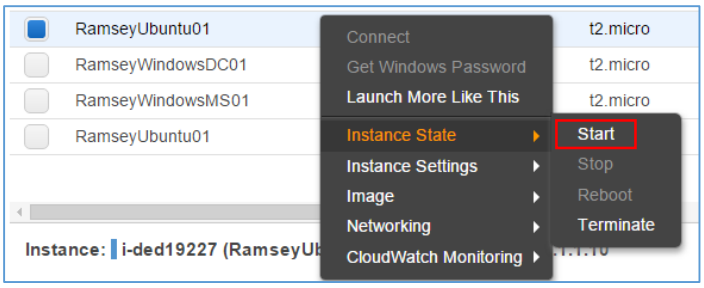

Figure 4: Starting the Ubuntu instance

3. Give it 2-3 minutes to spool up
4. When a public IP address is displayed, copy it
5. Launch PuTTY and copy the IP address into the address field (if you're using your personal computer, load the 4417key saved session first. I forgot when I was writing last week's lab instructions that on lab computers, that any sessions you save get wiped out when you log out)
6. Click on the '+' sign next to 'SSH' in the left pane; then on 'Auth'

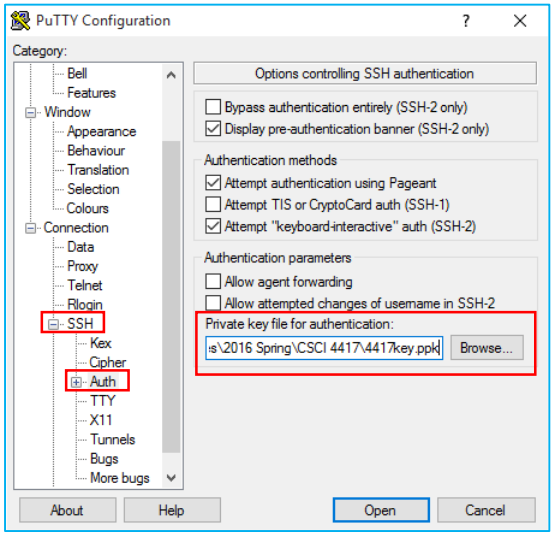

Figure 5: PuTTY SSH - selecting key

7. Click 'Browse' and select your 4417key.ppk file
8. Click 'Open'
9. Click 'Yes' on the PuTTY security warning and log in as user 'ubuntu' (lower-case 'u')
10. Enter the following command:

```
sudo tasksel
```

11. In the display window the launches, once you've wiped the tears from your eyes, press the down arrow until the cursor gets to 'LAMP Server.' Press the space bar to select it. Then press 'Tab' to highlight 'OK' and press 'Enter'

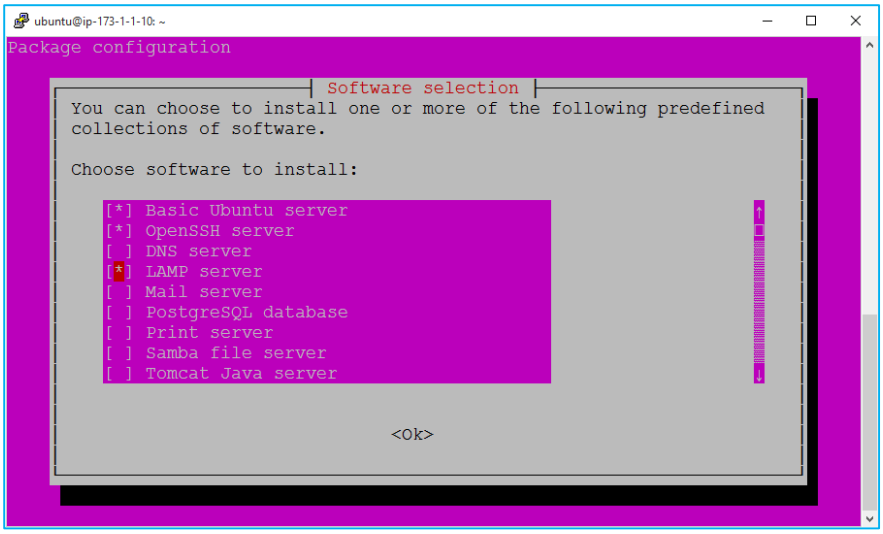

Figure 6: Installing LAMP using tasksel - This looks so 90's

12. Enter a password for the MySQL root user, then, again, 'Tab' and 'Enter'
13. Confirm the password, 'Tab,' 'Enter'
14. Confirm that Apache2 and MySQL are running:
    a. Enter **`sudo service --status-all`**
    b. You may have to scroll up to see the status of 'apache2.' a [+] to the left of the service name indicates that it is running.
    c. You may see [?] next to 'mysql.' If so, you can enter either

`sudo service mysql status`, or
`sudo service mysql restart` (or **`start`**)

(askUbuntu has this to say about that: The question mark indicates that **`service`** was not able to determine the status of the running service since it did not find the status line in the related script in /etc/init.d. I checked and did find a 'status' line in /etc/init.d/mysql, but it still doesn't work with the *--status-all* switch.)

    d. Another command, **`sudo initctl list`**, interestingly, will report that MySQL is running, but has nothing to say about Apache2. Try:

`sudo initctl list | grep "mysql"`
then
`sudo initctl list | grep "apache"`

(You remember **`grep`**, don't you? From CSCI 2200?)

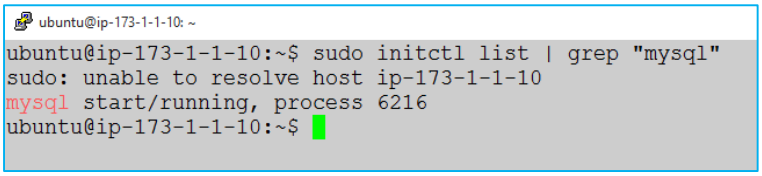

Figure 7: Using initctl to check process status

The **`initctl`** command is packaged with Upstart, the initialization routine that Ubuntu tried to replace the aging System V with, but eventually lost out to Systemd. Ubuntu 14.04 uses Upstart, but transitioned to Systemd with 15.10. It's present on 14.10 and 15.04, but the default was still Upstart. We'll talk about all of this in class, don't worry.

15. Now, copy and paste your Ubuntu's public IP address into the address bar of a web browser to confirm that the Apache server is up, running, and ready to feed web pages out to an unsuspecting public.
16. Oops. What happened? Read on:

17. Remember when I said in lab last week that most times when you have a connectivity error with AWS, the problem can be traced to the security group? Well, that is the case here
18. On the left menu, click on 'Security Groups'
19. Select your security group
20. Click on the 'Inbound' tab
21. You should have two rules: SSH and RDP. We need to add HTTP (remember which port it uses?)
22. Click 'Edit'
23. Click 'Add Rule'
24. For 'Type,' select 'HTTP'
25. Source will probably already be 'Anywhere.' If not, select it

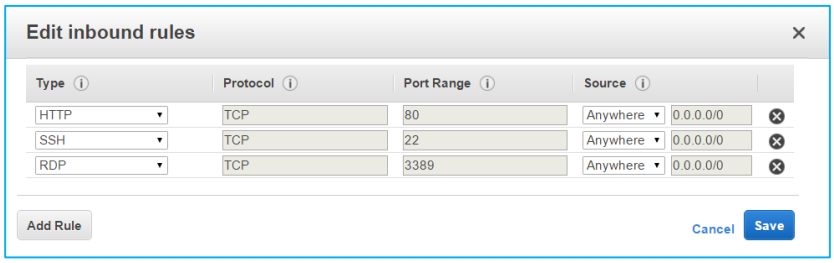

Figure 8: Enabling HTTP

26. Click save and return to your browser. You may have to refresh the page. Then you should see

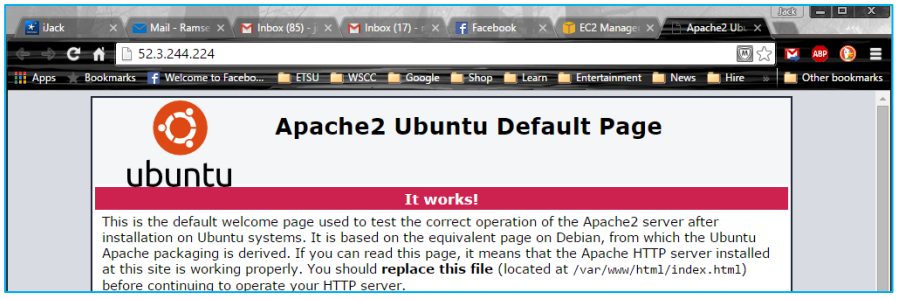

Figure 9: The Apache2 Ubuntu Default Page

27. Close your browser tab and return to PuTTY
28. Shut down your Ubuntu instance (both from PuTTY, first; then from the EC2 console).

```
sudo shutdown now
```

One thing about PuTTY (I think it's PuTTY, not Amazon) is that if you leave your console unattended for too long, when you return to it, you can type, like, two letters and it'll freeze. If this happens, the quickest fix I've found is to right-click on the menu bar and select 'Duplicate Session,' which will let you log back in without having to copy/paste the IP address and load the key again. If you don't close the offending window, which I don't, PuTTY will honk at you after a minute or so. Not closing the window, however, allows you, if your current one also freezes, to switch to it, right-click on the menu bar, and select 'Restart Session,' allowing you to log in again.

Note: We could have `sudo apt-get install` -ed Apache, MySQL, and PHP individually, but I usually use tasksel, which automates things. The two cloud providers I know of, AWS and DigitalOcean, both offer free images with it pre-installed (Aren't you glad I told you that NOW?? ☺)

Windows (Wrapping this beast up)

1. Now, we need to do a little housekeeping work on our Windows instances
2. On the EC2 console, right-click on your *lastname*WindowsDC01 instance.
3. Select 'Get Windows Password'
4. Click 'Browse' and navigate to the location you saved your keys to
5. Select the '4417key.pem' file **(.pem**, not **.ppk**!)
6. Click on 'Decrypt Password'
7. Using either Notepad or Word, copy the Public DNS and temporary password (we'll need to close this window to get the member server's address and temporary password. You can connect to instances via either IP address or Public DNS. I usually use the IP address because it's shorter)

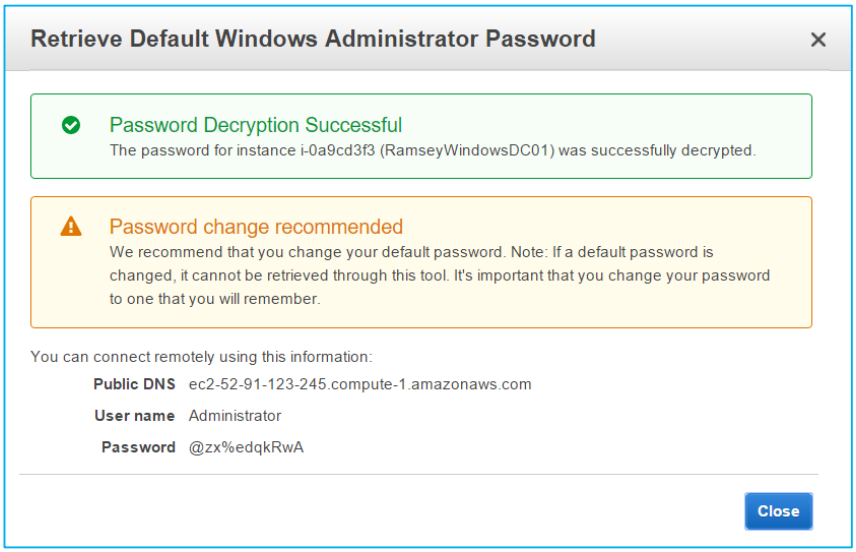

Figure 10: Decripted password & Public DNS

8. Amazon's changing things again, I see: The temporary password used to be shorter by several digits. Great. Makes changing the password that much more tortuous! (I got it wrong three times when I changed my member server's password)
9. Launch Remote Desktop Connection
10. Paste the Public DNS into the address field
11. Click on 'Show Options'
12. Enter 'Administrator' (without quotes) into the 'User name' field
13. Don't worry about saving the credentials or saving the RDP file - The instance's DNS and IP will change after today's lab. Since we're not using Elastic IPs (ironically, 'Elastic IPs' create static IP addresses. Amazon has a thing for catchy name, but that makes no sense to me)
14. Click 'Connect'

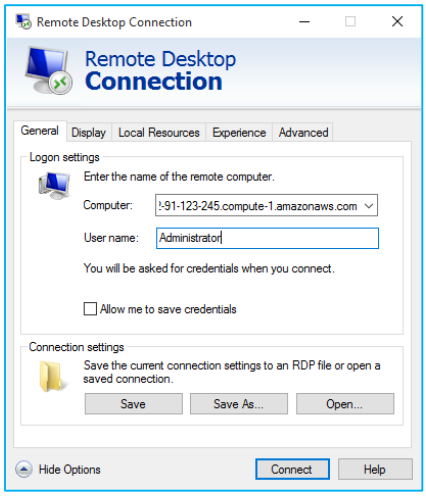

Figure 21: Remote Desktop Connection

15. Click 'Yes' on the security warning dialog
16. Enter the temporary password. Interestingly, I had copied it into Notepad++ on Step 7, above. It let me copy & paste it into the password field
17. It'll take a little while for the instance to initialize, so minimize the RDP window and repeat steps 2-15 with your *lastname*WindowsMS01 instance
18. Return to your domain controller (in the future) instance RDP connection. You should be logged in
19. Type Ctrl-Alt-End (**End**, not **Delete**!)
20. Select 'Change a Password'
21. The instance will not let you copy & paste, so you'll have to type the temporary password in
22. Enter the new password (I **strongly** recommend using **Passw0rd!** Though **definitely not best practice**, for lab purposes it'll be fine. It's easy to remember and satisfies Windows' password strength policy. That's capital 'P', the letters 'assw,' the number zero '0,' the letters 'rd' and exclamation mark, '!')
23. Confirm the new password and click the arrow or type 'Enter'
24. Shut down the domain controller instance: Right-click on the metro button in the lower left corner of the display. The power options can then be accessed similarly to a desktop version of Windows
25. Windows Server 2012 will want to log the reason for shutting down. You can select anything, but I usually go with 'Other (Planned)'
26. Repeat Steps 16-23 with your member server
27. Return to the EC2 console and shut down your Windows instances from there as well

Complete your lab report by the beginning of class time next Tuesday. While it should be structured in accordance with the lab guide, you can consolidate the Windows and Ubuntu portions separately, if you wish, in the Procedure section. Consider the following questions:

1. Knowing what you now know about installing the LAMP stack, what do you think it would be like if you had to install LAMP on 10 machines? 50? 100? What could you do on the front end to ease the pain? (This should be real vanilla for you…there are plenty of hints above)

2. One cool feature of Windows Server 2012 R2 is functionality to enable and disable the GUI. Why might this be desirable? (I can think of two reasons off the top of my head)
3. In your opinion, is it better (more comfortable, more efficient, etc.) to work from a command line or a GUI? (I hesitate to ask this because I think I know how it's going to pan out…your opinion, please, not what you think I 'want' to hear)
4. Let's say you're in charge of a large Windows-based domain (at the top level of the Windows hierarchy, it's actually called a Forest but we'll talk about that later). Based on your experience today, do you think it would be better to let Amazon's infrastructure detect and respond to expected spikes in demand or provision for the spikes proactively (ahead of time)? Why?
5. We did just a little bit with Ubuntu today. But how much Unix/Linux do you remember from CSCI 2200? I'm asking because I've been reviewing a lot of Dr. Pfeiffer's lecture materials for him this semester. I'm seeing a lot of overlap with the material that I have. Answering this will help me determine if I need to change some stuff. If you honestly don't remember a lot of it (it's been two years since 2200 for many of you, after all), that's fine -- less work for me. But I don't want to waste your time, and can find other topics of value.