Assignment 1

Colin Hansen, Bryan Steitz, Pan Wang
September 21, 2017

In this assignment, we setup a LAMP stack on two virtual machines. The first machine, a webserver, has a floating IP address and is open to external traffic. The database server is located within the Horizon environment and can only communicate with other instances within the environment. The LAMP setup is divided into six steps: create unique security group, create webserver VM using GUI, create database server using CLI, setup webserver, setup database server, and populate the database.

Create Unique Security Group

The following steps are required to create and configure a unique security group.

1. Navigate to Horizon GUI and select Access & Security > Security Groups
2. Select *Create Security Group* and give the group a name *teamCBP_group*
3. Select *Manage Rules* for this group
4. Add rule for Ingress SSH (port 22)
5. Add rule for Ingress HTTP (port 80)
6. Add rule for Ingres MYSQL (port 3306)

Webserver Setup

The following steps are required to initialize and configure the webserver.

Create Webserver VM Using GUI

To create the webserver virtual machine, we use the OpenStack GUI.

1. Navigate to Horizon GUI and select Launch Instance in the Compute > Instances tab
2. Select *Ubuntu 16.04* image source and *m1.small* flavor
3. Select internal cloud class subnet, *internal_network*
4. Assign group key pair and launch instance
5. When instance has started, assign floating IP address to access the VM from an external network

Setup Webserver

Now that the webserver has been initialized in Horizon and has been assigned a floating IP address, we can begin

configuration.

1. SSH into webserver using group key, providing the path to the key, ssh username, and IP of webserver `ssh -i ~/.ssh/class-key.pem ubuntu@129.59.107.197`
2. Set the sudo password: `sudo passwd`.
3. Update packages on the webserver: `sudo apt-get update`
4. Upon installing packages, we noticed an error with sudo requiring the local machine to be added to the list of hosts. We open: `sudo vim /etc/hosts/` and add `129.59.107.197 cbp-web` as the second line in the file.
5. Now that errors had been resolved and packages are up to date, we installed apache2 as `sudo apt-get install apache2`
6. To ensure the installation was successful, we run `sudo service apache2 start` and `sudo service apache2 stop` to start and stop the apache service.
7. Since apache was working as expected, we can install php `sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql` and git `sudo apt-get install git`.
8. Configure `etc/apache2/mods-enabled/dir.conf` to prioritize php files. We moved index.php to the first file position, immediately after *DirectoryIndex*.
9. Restart apache: `sudo service apache2 restart`
10. We then navigate to `/var/www/html/` and clone the git repository `sudo git clone https://github.com/doc-vu/phpMySQLapp`.
11. Finally, we change the following credentials in `phpMySQLapp/movies/includes/movieDatabase.php` and `phpMYSQLapp/books/includes/bookDatabase.php`.

```
$servername = "10.10.19.138"
$username = "root"
$password = "password"
```

Database Server Setup

The following steps are required to initialize and configure the database server.

Create Database Server Using CLI

1. Install the Openstack client via *pip*: `sudo -h pip install python-openstackclient`
2. Download the Class-openrc.sh from Openstack

3. Source the class-openrc.sh file: `source class-openrc.sh`.
4. Create the instance using the command line:

```
server create --wait --image “ubuntu-16.04_20170605” --flavor m1.small --key-name testkey
--security-group teamCBP_group --nic net-id b16b0244-e1b5-4d36-90ff-83a0d87d8682 CBP_dp
```

Setup Database Server

1. Move the keypair for the database server into the webserver: `scp -i ~/.ssh/class-key.pem ~/.ssh/testkey.pem ubuntu@129.59.107.197:~/.ssh/`.
2. SSH into the webserver: `ssh -i ~/.ssh/class-key.pem ~/.ssh/testkey.pem ubuntu@129.59.107.197`.
3. SSH into the database server: `ssh -i ~/.ssh/testkey.pem 10.10.19.138`.
4. Set sudo password with `sudo passwd`.
5. We encouter the same hosts error as we did with the webserver. We similarly add `127.0.1.1 cbp-db` (machine IP address and machine name) to `/etc/hosts`.
6. Update packages on server: `sudo apt-get update`
7. Install git: `sudo apt-get install git`.
8. Clone the phpMySQL repository: `Git clone https://github.com/doc-vu/phpMySQLapp`.
9. Install MySQL: `Sudo apt-get install mysql-server`.
10. Navigate to `Sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf` to allow for remote access to MySQL
11. Once in the *mysqld.cnf* file, change `bind-address = 127.0.0.1` to `bind-address = 0.0.0.0`.
12. Login to MySQL: `mysql -u root -p`.
13. Create bookstore: `create database bookstore` and movie `create database moviedb` databases
14. Give access permissions for webserver to access the database server via root user: `GRANT ALL ON *.* TO root@10.10.19.134 IDENTIFIED BY PASSWORD ‘password’ WITH GRANT OPTION;` `FLUSH PRIVILEGES;` `exit;`
15. Restart MySQL: `sudo service mysql restart`

Populate MySQL Database

The following steps assume that you are already in the database server

1. Populate bookstore database `Mysql -u root -p bookstore < ~/phpMYSQLapp/mySqlDB/bookDB.sql`
2. Populate movieDB database `Mysql -u root -p bookstore < ~/phpMYSQLapp/mySqlDB/movieDB.sql`

End of Assignment Survey: Evaluating the Deployment Approach

Scale: 1 to 10; increasing in difficulty

Ease of deployment: Scale [1-10]

How much time and effort you required to deploy the application (in minutes), including internet searches you have to do to find the steps?

- To install Apache: *10 minutes*
- To install and configure PHP: *5 minutes*
- To install and configure MySQL: *60 minutes*
- Overall time to deploy application: *75 minutes*

Document all commands you executed to deploy the application

(See above sections)

Challenges you faced for this simple 3-tier application deployment

The main obstacle was configuring MySQL on the database server to correctly allow for remote access. The members of our group had previous experience configuring and deploying LAMP servers, which helped to reduce the overall difficulty of the 3-tier deployment.

Ideas to alleviate the manual effort and avoid mistakes you may have faced during manual installation and configuration of the software packages for the application deployment.

- Use automated method to deploy and configure servers remotely. This would allow a user to list all commands sequentially and allow each VM to run the list of commands from a single script.
- Use automated method such as *Ghostcast* or other image deployment software to configure a single machine that can be deployed multiple times. This method would require individual configuration for SQL and Apache but would significantly improve the package installation process.