Creating a Dynamic Website for the Cafe

Step 1: Create an instance

1. Create an EC2 instance called *CafeWebServer* with a public IP assigned.

Step 2: Connecting to the IDE on the EC2 instance

2. In the search box next to Services, search for and select **Cloud9**, to go to the AWS Cloud9 console.
   In the Environments page, notice the *CafeWebServer* environment. It indicates that it is of type *EC2 instance*.
3. Choose **Open**.
4. You are now connected to the AWS Cloud9 IDE that is running on the EC2 instance that you created earlier.

Step 3: Analyzing the LAMP stack environment and confirming that the web server is accessible.

5. Observe the OS version. In the AWS Cloud9 bash terminal, run this command:

```
cat /proc/version
```

Notice how the output indicates it is an Amazon Linux instance, roughly analogous to Red Hat 7.

6. Observe the web server, database, and PHP details and server state.
   In the terminal, run these commands:

```
sudo httpd -v
sudo yum -y install httpd php-mbstring
service httpd status
php --version
```

7. The output should show the versions of the web server and the database, and also show that they are not currently running.
8. Start the web server and the database, and also set them to start automatically after any future EC2 instance restart.
   In the terminal, run these commands:

```
sudo chkconfig httpd on
sudo service httpd start
sudo service httpd status
#install database
sudo yum install -y mariadb-server
sudo mariadb --version
sudo systemctl enable mariadb
#
sudo chkconfig mariadb on
sudo service mariadb start
sudo service mariadb status
```

9. Configure the EC2 instance so that you can use the AWS Cloud9 editor to edit web server files.
   Notice that the AWS Cloud9 file browser currently does not display the Apache web server default web directory.
   In the terminal, run these two commands:

```
ln -s /var/www/ /home/ec2-user/environment
sudo chown ec2-user:ec2-user /var/www/html
```

10. The first command created a symlink from the default AWS Cloud9 editor workspace to the /var/www directory that contains your web server files.
    The second command changed ownership of the **html** subdirectory so that the *ec2-user* (which you are logged in as) can edit and create new files in it.
11. . Creating a simple test webpage.
    - In the *file browser*, expand the **CafeWebServer > www** directory, and highlight the **html** directory.
    - Choose **File** > **New File**.
    - In the text editor tab, paste the following line:
    - <html>Hello from the café web server!</html>
    - Choose **File** > **Save**, and save the file in the **html** directory as *index.html*.
12. Make the website accessible from the internet.
    In this step, you will need to verify and update the configurations that make the webpages (which are hosted on the web server) accessible from the internet.

New Business Requirement: Installing a dynamic website application on the EC2 instance.

Task 4: Installing the café application

13. Download and extract the web server application files. In the Bash terminal, run these commands:

```
cd ~/environment
wget
https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-2-91555/04-la
b-mod4-challenge-EC2/s3/setup.zip
unzip setup.zip
wget
https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-2-91555/04-la
b-mod4-challenge-EC2/s3/db.zip
```

```
unzip db.zip
wget
https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-2-91555/04-lab-mod4-challenge-EC2/s3/cafe.zip
```

Notice how the file browser now shows the three .tar.gz files that you downloaded.You also extracted these archive files, which created the cafe, db, and setup directories in your work environment.

14. Copy the café files over to the web server document root. In the Bash terminal, run this command:

```
unzip cafe.zip -d /var/www/html/
```

15. Observe how the application is designed to work.
    - Open the html/cafe/index.php source code in the AWS Cloud9 editor by double-clicking it.
    - Notice that this file has HTML code in it, but it also contains sections that are enclosed in elements. These elements make calls to other systems and resources.
    - For example, on **line 18**, you see that the PHP code references a file named *getAppParameters.php*.
    - Open the **getAppParameters.php** file in the code editor.
    - Notice on **line 3** of this file that the *AWSSDK* is invoked.
    - Also, on **lines 10–33**, the web application creates a client that connects to the *ssm* service, which is AWS Systems Manager. The application then retrieves seven parameters from Systems Manager. Those parameters have not been created in AWS Systems Manager yet, but you will do that next.
16. In the AWS Systems Manager Parameter Store, configure the application parameters. In the Bash terminal, run these commands:

```
cd setup
./set-app-parameters.sh
```

17. The shell script that you just ran issued AWS Command Line Interface (AWS CLI) commands. These commands added the parameters that the application will use to the Parameter Store.
18. In the AWS Management Console, from the **Services** menu, choose **Systems Manager**.
19. From the panel on the left, choose **Parameter Store**. Notice how there are now seven parameters stored here.
20. The café application's PHP code references these values (for example, so that it can retrieve the connection information for the MySQL database).
    Choose the /cafe/dbPassword parameter, and copy the *Value* to your clipboard. You will use this value in a moment.
21. Configure the MySQL database to support the café application. Back in the AWS Cloud9 bash terminal, run the following commands:

```
cd ../db/
./set-root-password.sh
./create-db.sh
```

22. Observe the database tables that were created. In the Bash terminal, run this command to connect the terminal-based MySQL client to the database:

```
mysql -u admin -p
```

23. When you are prompted for the database password, paste the *dbPassword* parameter value that you copied. You should now see a mysql> prompt, which indicates that you are now connected to the MySQL database that runs on this EC2 instance.

To observe the contents of the database (specifically, the tables that support the café web application), enter the following commands:

```
show databases;

use cafe_db;

show tables;

select * from product;

exit;
```

24. Update the *timezone* configuration in PHP.In the Bash terminal, run the following commands:

```
sudo sed -i "2i date.timezone = \"America/New_York\" " /etc/php.ini
sudo service httpd restart
```

The first command that you ran configured the time zone in the PHP software.
The second command that you ran restarted the web server so that the web server notices the configuration update.

25. Test whether the café website is working and can be accessed from the internet. In a new browser tab, try to load the application at http://<public-ip>/cafe where <*public-ip*> is the IPv4 public IP address of the EC2 instance. You will see that *only the title banner* of the website loads. The rest of the webpage is not loading correctly.
26. Resolve an issue with the website. In this step, you will need to figure out how to make the café website function correctly by fixing the permissions.
27. The website should load.

Step 5: Testing the web application

27. Test by placing an order.
    - In the browser tab where you have the http://<public-ip>/cafe page open, choose **Menu**.
    - Submit an order for at least one of the menu items displayed.
    - Return to the menu page and place another order, then go to the **Order History** page to see the order details for all the orders that you placed.

New business Requirement: Creating development and production websites in different AWS Regions.

Task 6: Creating an AMI and launching another EC2 instance

28. Set a static internal hostname and create a new key pair on the EC2 instance.
    In the bash terminal, run the following commands:

```
sudo hostname cafeserver
ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

29. For the two times that you are prompted for a passphrase, press the ENTER key.
    To make the new key available to the SSH utilities, in the Bash terminal, run the following command:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

30. In the AWS Management Console, browse to the **EC2** service area and select the instance.
31. Choose **Actions > Images and templates > Create Image**.
    - **Image name**: CafeServer
    - Choose **Create Image**
34. From the navigation menu, choose **AMIs** and wait until the image status becomes *Available*. The process typically takes about 2 minutes.
35. Create an AMI in another AWS Region
    In this step, your objective is to create a new EC2 instance from the AMI that you just captured. However, you must create the new instance in the Oregon (us-west-2) AWS Region.
36. Create the new café instance from your AMI. The new instance that you create must match the following criteria.
    - **Region**: Oregon
    - **Instance Size**: t2.small
    - **Network**: Lab VPC Region 2, Public Subnet
    - **IAM Role**: CafeRole
    - Tag:
        - **Key**: Name
        - **Value**: ProdCafeServer

- Security Group:
  - Create a new one named **cafeSG**, with TCP port **22** open to anywhere
  - Set TCP port **80** so that it's open to anywhere as well
- **Proceed without a key pair** (the key pair that you created earlier in this lab should work to connect to it, if necessary)

37. Wait for the new instance to have a *Public DNS* value assigned to it, even if the status of the instance is still not *Available*.
38. Copy the **Public DNS** value. You will use it soon.
39. To create the needed *AWS Systems Manager parameters* in the new AWS Region, complete these steps.
    - Return to the AWS Cloud9 IDE in the **N. Virginia (us-east-1)** Region.
    - Open the CafeWebServer/setup/**set-app-parameters.sh** file in the text editor.
    - Edit **line 12** of the file to match this setting:
    - region="us-west-2"
    - Edit **line 18** to match this setting (where <*public-dns-of-ProdCafeServer-instance*> is the actual DNS of the ProdCafeServer instance):
    - publicDNS="<public-dns-of-ProdCafeServer-instance>"

Note: The line should still contain the quotation marks, but it should *not* contain the angle brackets (< >).

This example shows what line 12 should look like and how line 18 should be formatted. However, the value of your public DNS will be different.

- **File > Save** the change.
- To run this script, go to the top of the IDE and choose the **Run** button.

In the Bash terminal below the text editor, you should see output that's formatted in JavaScript Object Notation (JSON). This output indicates that the parameters script ran successfully.

By changing the AWS Region details and running this script again, you created the same parameters that you created earlier in the us-east-1 Region of the AWS Systems Manager Parameter Store. However, this time, you created these parameters in the Oregon Region.

Step 7: Verifying the new café instance

40. Return to the EC2 Console in the **Oregon** Region, and verify that the new **ProdCafeServer** instance is running.
41. Copy the IPv4 public IP address, and load it in a web browser.
    The *Hello from the cafe web server!* message should display.
42. Load the http://<public-ip>/cafe/ URL in a browser tab.
    The entire café website should display.
43. Load the **Menu** page.
    The full *Menu* page should load, and the order-placing functionality should work.
44. Place an order to verify that the website is working as intended.

45. Troubleshooting tips (skip this one step if you didn't encounter any issues with loading the *Menu* page).
    The grading script can provide additional tips for parts of the lab that you didn't complete successfully. You can submit your work as many times as you like—only the score that you achieve on the last submission will be retained.
    Also, if you want to connect to the new EC2 instance in Oregon (us-west-2) to do some troubleshooting, run this command from the AWS Cloud9 IDE in us-east-1:

```
ssh -i ~/.ssh/id_rsa ec2-user@<public-ip-of-ProdCafeServer>
```

46. Note that <*public-ip-of-ProdCafeServer*> is the actual public IP address of the ProdCafeServer instance.