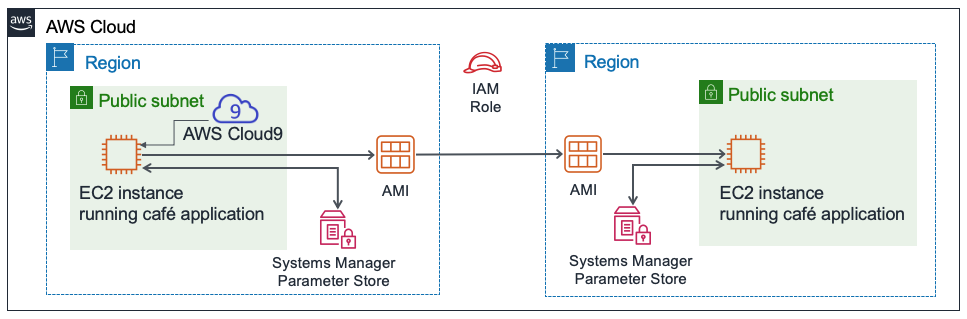
Lab 4:

# Aim: Creating a Dynamic Website for the Café

# Architecture:

When you start the lab, some resources are already created for you in the AWS account:

At the end of this lab, your architecture should look like the following example:



### Task 1: Analyzing the existing EC2 instance

Access the questions in this lab.

* + Choose the **Details** menu, and choose **Show**.
  + Choose the **Access the multiple choice questions** link that appears at the bottom of the page

Task 2: Connecting to the IDE on the EC2 instance

In the search box next to Services, search for and select **Cloud9**, to go to the AWS Cloud9 console.

Choose **Open**.

You are now connected to the AWS Cloud9 IDE that is running on the EC2 instance that you observed earlier.

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

Recall that the objective of this challenge lab is configure an EC2 instance to host the new dynamic website for the café. In this task, you will analyze what is already installed.

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.

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

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

 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

The first command you ran 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.

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.

 Make the website accessible from the internet.

 Task 4: Installing the café application

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-lab-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-lab-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.

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/

 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.

 In the AWS Systems Manager Parameter Store, configure the application parameters.

In the Bash terminal, run these commands:

cd setup

./set-app-parameters.sh

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.

 In the AWS Management Console, from the **Services** menu, choose **Systems Manager**.

 From the panel on the left, choose **Parameter Store**.

Notice how there are now seven parameters stored here.

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.

 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

 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

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;

 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.

 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.

Resolve an issue with the website.

In this step, you will need to figure out how to make the café website function correctly.

Here's a list of what does work:

* + The test page at [http://<public-ip](http://&lt;public-ip/)>/ loads, so you know that the web server works, and is accessible from the internet
  + You also know that the MySQL database is running, and contains tables and data to support the application

### Task 5: Testing the web application

1. 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.
     + Note: you may need to scroll down to find the **Submit Order** button.
   * 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.

 Task 6: Creating an AMI and launching another EC2 instance

1. 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

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

 In the AWS Management Console, browse to the **EC2** service area and select the instance.

Choose **Actions > Images and templates > Create Image**.Back in the AWS Management Console, in the **Create Image** screen, create the new AMI:

* + **Image name**: CafeServer
  + Choose **Create Image**

 From the navigation menu, choose **AMIs** and wait until the image status becomes Available. The process typically takes about 2 minutes. You may need to expand **Images** to find **AMIs**.Create an AMI in another AWS Region

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)

 Copy the **Public DNS** value. You will use it soon.

Task 7: Verifying the new café instance

Copy the IPv4 public IP address, and load it in a web browser.

The Hello from the cafe web server! message should display.

 Load the http://<public-ip>/cafe/ URL in a browser tab.

The entire café website should display.

 Load the **Menu** page.

Troubleshooting tips (skip this one step if you didn't encounter any issues with loading the Menu page).

ssh -i ~/.ssh/id\_rsa ec2-user@<public-ip-of-ProdCafeServer>

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

To confirm that you want to end the lab, at the top of this page, choose **End Lab**, and then choose **Yes**.

A panel should appear with this message: DELETE has been initiated... You may close this message box now.

To close the panel, choose the **X** in the top-right corner.

# Lab 6

# Aim: Creating a VPC Networking Environment for the Cafe

# Architecture:

### Task 1: Creating a public subnet

### Open the **Amazon VPC console**.

### Note that a VPC called Lab VPC was created for you.

### Create a public subnet that meets the following criteria:

* + **Name tag**: Public Subnet
  + **VPC**: Lab VPC
  + **Availability Zone**: Choose Availability Zone **a** of your Region (for example, if your Region is us-east-1, then select **us-east-1a**)
  + **IPv4 CIDR block**: 10.0.0.0/24

Create a new internet gateway and attach it to the Lab VPC.

Edit the route table that was created in your VPC. Add the route 0.0.0.0/0. For the target, select the internet gateway that you created in the previous step

### Task 2: Creating a bastion host

From the **Amazon EC2 console**, create an EC2 instance in the Public Subnet of the Lab VPC that meets the following criteria:

* + **Amazon Machine Image (AMI)**: Amazon Linux 2023 AMI (HVM)
  + **Instance type**: t2.micro
  + **Auto-assign Public IP**: This setting should be disabled
  + **Name**: Bastion Host
  + Security group called Bastion Host SG that only allows the following traffic:
    - **Type**: SSH
    - **Port**: 22
    - **Source**: Your IP address
  + Uses the **vockey** key pair

Task 3: Allocating an Elastic IP address for the bastion host

Allocate an Elastic IP address, and make it reachable from the internet over IPv4 by associating it with your bastion host.

Task 4: Testing the connection to the bastion host

In this task, you will use the SSH key (.pem file or .ppk file) to test the SSH connection to your bastion host. This key was created for you.

In the top-right area of these instructions, select **Details**.

Next to **AWS**, choose **Show**.

Download the SSH key. Note the file will be named **labuser.\***.

* + **Microsoft Windows PuTTY users**: Download PPK
  + **macOS or Linux users**: Download PEM

To close the window, choose **X**.

Connect to your bastion host by using SSH.

After you have tested your connection to the bastion host, you can close the terminal or PuTTY.

Task 5: Creating a private subnet

In this task, you will create a private subnet in the Lab VPC.

In the console, create a private subnet that meets the following criteria:

* + **Name tag**: Private Subnet
  + **Availability Zone**: Same as Public Subnet
  + **IPv4 CIDR block**: 10.0.1.0/24

### Task 6: Creating a NAT gateway

In this task, you will create a NAT gateway, which enables resources in the Private Subnet to connect to the internet.

Create a NAT gateway that meets the following criteria:

* + **Name**: Lab NAT Gateway
  + **Subnet**: Public Subnet

Create a new route table that meets the following criteria:

* + **Name tag**: Private Route Table
  + **Destination**: 0.0.0.0/0
  + **Target**: NAT Gateway

1. Attach this route table to the Private Subnet, which you created earlier.

### Task 7: Creating an EC2 instance in the private subnet

In this task, you will create an EC2 instance in the Private Subnet, and you will configure it to allow SSH traffic from the bastion host. You will also create a new key pair to access this instance.

1. Create a new key pair named vockey2, and download the appropriate .ppk (Microsoft Windows) or .pem (macOS or Linux).
2. Create an EC2 instance in the Private Subnet of the Lab VPC that meets the following criteria.
   * **AMI**: Amazon Linux 2023 AMI (HVM)
   * **Instance type**: t2.micro
   * **Name**: Private Instance
   * Only allows the following traffic:
     + **Type**: SSH
     + **Port**: 22
     + **Source**: Bastion host security group (**Hint**: Refer to the [AWS Documentation](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html)
   * Uses the **vockey2** key pair that you created earlier

Task 8: Configuring your SSH client for SSH passthrough

Download and install Pageant, which is available from the PuTTY [download page](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

After you install Pageant, open it. Pageant runs as a Windows service.

To import the PuTTY-formatted key into Pageant, follow these steps.

In the Windows system tray, double-click the **Pageant** icon.

Choose **Add Key**.

Select the .ppk file that you downloaded when you created the vockey2 key pair.

Add the first vockey that you downloaded earlier. The filename was **labuser.\***.

In PuTTY, under **Connection > SSH > Auth**, select **Allow agent forwarding**. Expand **Auth** and choose **Credentials**. Under **Private key file for authentication** choose **Browse**. Browse to the labsuser.ppk file that you downloaded, select it, and choose **Open**. Choose **Accept**. After you have completed this step, continue on to Task 9, step 32. Proceed to connect to the bastion host using PuTTY as you normally would, but don't open a .ppk file.

Task 9: Testing the SSH connection from the bastion host

In this task, you will test the SSH connection from your bastion host to the EC2 instance that is running in the Private Subnet.

Connect to the bastion host instance by using SSH.

Connect to the private instance by using SSH and the IP address for the private instance.

ssh ec2-user@<private-ip-address-of-instance-in-private-subnet>

Now that you are connected to the EC2 instance in the Private Subnet, test its connection to the internet.

ping 8.8.8.8

Task 10: Creating a network ACL

Go to the **Amazon VPC console**, and inspect the default network ACL of Lab VPC.

**Note 1**: The subnets that you created are automatically associated with the default network ACL.  
**Note 2**: The inbound and outbound rules of the default network ACL allow all traffic.

Create a custom network ACL called Lab Network ACL for the Lab VPC.

**Note**: The default inbound and outbound rules of the custom network ACL deny all traffic.

Configure your custom network ACL to allow ALL traffic that goes into and out of the Private Subnet.

### Task 11: Testing your custom network ACL

### Create an EC2 instance in the Public Subnet of the Lab VPC. It should meet the following criteria.

* + AMI: Amazon Linux 2023 AMI (HVM)
  + Instance type: t2.micro
  + Name: Test Instance
  + Allows All ICMP – IPv4 inbound traffic to the instance through the security group

Note the private IP address of the Test Instance.

Test that you can reach the private IP address of the Test Instance from the Private Instance. From the Private Instance terminal window, run the following ping command:

ping <private-ip-address-of-test-instance>

Leave the ping utility running.

Modify your custom network ACL to deny All ICMP – IPv4 traffic to the <private-ip-address-of-test-instance>/32

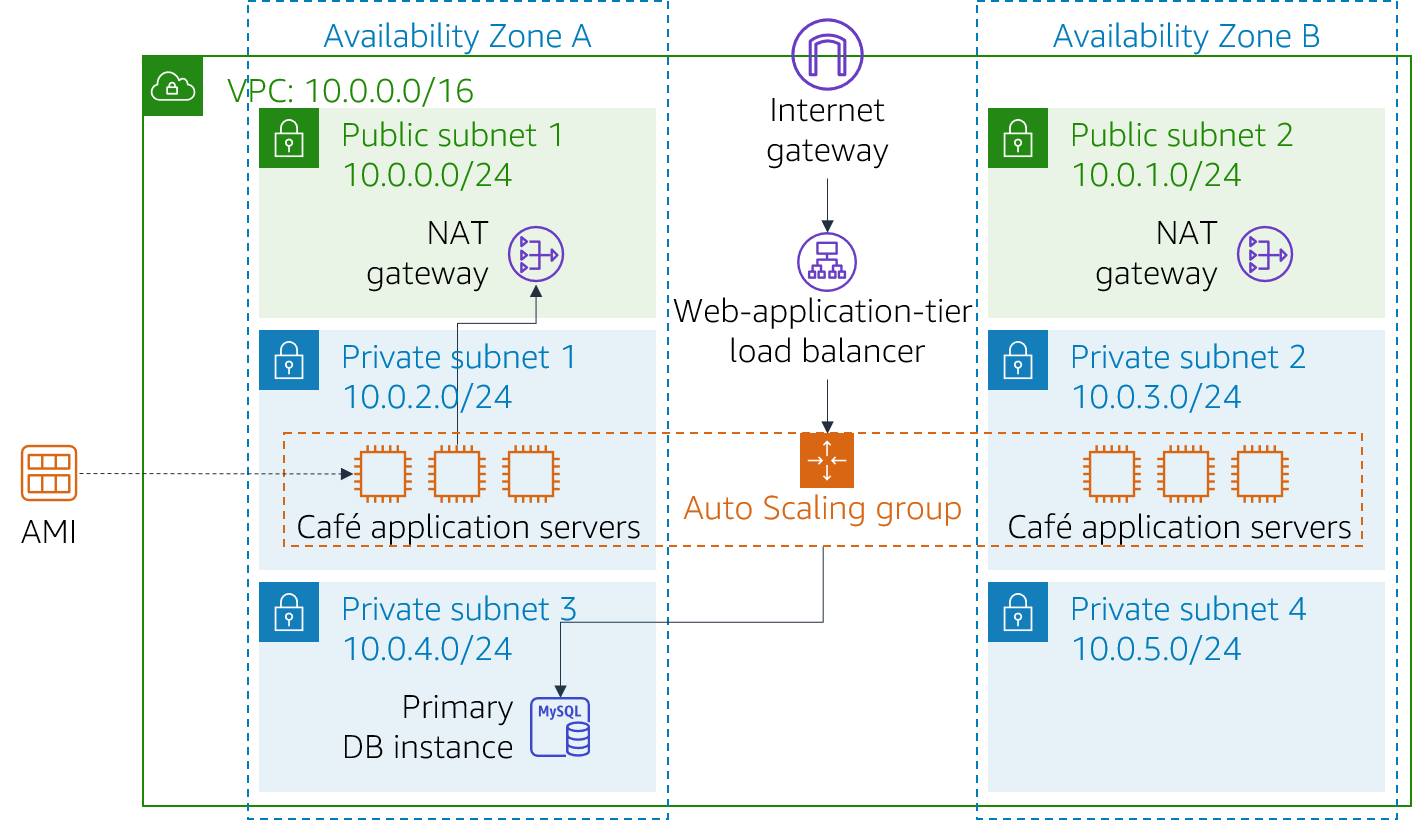
* + Make sure to add /32 to the end of the private IP address.
  + Make sure that this rule is evaluated **first**.

In the Private Instance terminal window, the ping command should stop responding. The traffic to the Test Instance has been blocked.

# To confirm that you want to end the lab, at the top of this page, choose **End Lab**, and then choose **Yes**

Lab 9

# Aim: Creating a Scalable and Highly Available Environment for the Cafe

Architecture:

### Task 1: Inspecting your environment

In this task, you will evaluate the current state of your lab environment.

Explore the lab environment, including how the network is set up.

### Task 2: Creating a NAT gateway for the second Availability Zone

To achieve high availability, the architecture must span at least two Availability Zones. However, before you launch Amazon Elastic Compute Cloud (Amazon EC2) instances for your web application servers in the second Availability Zone, you must create a NAT gateway for them. A NAT gateway will allow instances that do not have a public IP address to access the internet.

Create a NAT gateway in the Public Subnet in the second Availability Zone.

Configure the network to send internet-bound traffic from instances in Private Subnet 2 to the NAT gateway you just created.

Task 3: Creating a bastion host instance in a public subnet

In this task, you will create a bastion host in a public subnet. In later tasks, you will create an EC2 instance in a private subnet and connect to it from this bastion host.

From the **Amazon EC2 console**, create an EC2 instance in one of the public subnets of the Lab VPC. It must meet the following criteria:

* + Name**:** Bastion Host
  + **Amazon Machine Image (AMI)**: Amazon Linux 2023 AMI
  + **Instance type**: t2.micro
  + Uses the **vockey** key pair
  + **Auto-assign Public IP**: This setting should be enabled
  + Only allows the following traffic:
    - **Type**: SSH
    - **Port**: 22
    - **Source**: Your IP address

Task 4: Creating a launch template

During the lab setup, an Amazon Machine Image (AMI) was created from the CafeWebAppServer instance. In this task, you will create a launch template by using this AMI.

Create a launch template by using the AMI that was created during lab setup. It must meet the following criteria.

* + **AMI**: Cafe WebServer Image   
    **Tip**: To locate the AMI, go to the the **AMI** dropdown menu and enter: Cafe
  + **Instance type**: t2.micro  
    **Tip**: To locate the instance type, go to the **Instance Type** dropdown menu and enter: t2
  + **Key pair (login)**: Uses a new key pair **Tip**: Create a new key pair and select it. Make sure that you download the key pair to your local computer.
  + **Security groups**: CafeSG  
    **Tip**: To locate the security group, go to the **Security groups** dropdown menu and enter: CafeSG
  + **Resource tags**:
    - **Key**: Name
    - **Value**: webserver
    - **Resource types**: Instances
  + **IAM Instance Profile**: CafeRole  
    **Tip**: Look in **Advanced Details** for this setting.

Task 5: Creating an Auto Scaling group

Create a new Auto Scaling Group that meets the following criteria:

* + **Launch template**: Uses the launch template that you created in the previous task
  + **VPC**: Uses the VPC that was configured for this lab
  + **Subnets**: Uses Private Subnet 1 and Private Subnet 2
  + Skips all the advanced options
  + Has a **Group size** configured as:
    - **Desired capacity**: 2
    - **Minimum capacity**: 2
    - **Maximum capacity**: 6
  + Enables the **Target tracking scaling policy** configured as:
    - **Metric type**: Average CPU utilization
    - **Target Value**: 25
    - **Instances need**: 60

### Task 6: Creating a load balancer

Create an HTTP Application Load Balancer that meets the following criteria:

* + **VPC**: Uses the VPC configured for this lab
  + **Subnets**: Uses the two public subnets
  + Skips the HTTPS security configuration settings
  + **Security group**: Creates a new security group that allows HTTP traffic from anywhere
  + **Target group**: Creates a new target group
  + Skips registering targets

**Note**: Wait until the load balancer is active.

Modify the Auto Scaling group that you created in the previous task by adding this new load balancer.

### Task 7: Testing the web application

In this task, you will test the café web application.

To test the café web application, visit the Domain Name System (DNS) name of your load balancer and append /cafe to the URL.

The café application should load.

If it does not, go back through the lab tasks and check your work. Pay attention to the following resources:

* Network configuration: Did you add the NAT gateway correctly?
* Route tables: Did you update the route tables with the NAT gateway?
* Launch template: Does the instance specify an IAM role?
* Load balancer: Is the load balancer in the public subnets?
* Instances: Are the instances deployed from the Auto Scaling group that is in the correct subnets?
* Security groups: Do the security groups allow HTTP traffic from the internet?

Task 8: Testing automatic scaling under load

In this task, you will test whether the café application scales out automatically.

By using Secure Shell (SSH) passthrough through the bastion host instance, use SSH to connect to one of the running web server instances.

From the web server instance, use the following commands to start a stress test. This test increases the load on the web server CPU:

sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

sudo yum install stress -y

stress --cpu 1 --timeout 600Verify that the Auto Scaling group deploys new instances.

* + Continue to observe the Amazon EC2 console.
  + During the test, you should observe that more web server instances are deployed.

To confirm that you want to end the lab, at the top of this page, choose **End Lab**, and then choose **Yes**