**Module 9 – Guided Lab: Creating a Highly Available Environment**

**Lab overview and objectives**

Critical business systems should be deployed as *highly available applications*—that is, applications remain operational even when some components fail. To achieve high availability in Amazon Web Services (AWS), we recommend that you *run services across multiple Availability Zones*.

Many AWS services are inherently highly available, such as load balancers. Many AWS services can also be configured for high availability, such as deploying Amazon Elastic Compute Cloud (Amazon EC2) instances in multiple Availability Zones.

In this lab, you will start with an application that runs on a single EC2 instance. You will then make the application highly available.

After completing this lab, you should be able to:

* Inspect a provided virtual private cloud (VPC)
* Create an Application Load Balancer
* Create an Auto Scaling group
* Test the application for *high availability*

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

**Duration**

The lab requires approximately **40 minutes** to complete.

**AWS service restrictions**

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

**Accessing the AWS Management Console**

1. At the top of these instructions, choose Start Lab to launch your lab.

A **Start Lab** panel opens, and it displays the lab status.

**Tip**: If you need more time to complete the lab, restart the timer for the environment by choosing the Start Lab button again.

1. Wait until the **Start Lab** panel displays the message *Lab status: ready*, then close the panel by choosing the **X**.

1. At the top of these instructions, choose AWS.

This action opens the AWS Management Console in a new browser tab. The system automatically logs you in.

**Tip**: If a new browser tab does not open, a banner or icon is usually at the top of your browser with the message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose **Allow pop-ups**.

1. Arrange the **AWS Management Console** tab so that it displays alongside these instructions. Ideally, you will have both browser tabs open at the same time so that you can follow the lab steps more easily.

**Do not change the Region unless specifically instructed to do so**.

**Task 1: Inspecting your VPC**

This lab begins with an environment that is already deployed via AWS CloudFormation. It includes:

* A VPC
* Public and private subnets in two Availability Zones
* An internet gateway (not shown) that is associated with the public subnets
* A Network Address Translation (NAT) gateway in one of the public subnets
* An Amazon Relational Database Service (Amazon RDS) instance in one of the private subnets

In this task, you will review the configuration of the VPC that was created for this lab.

1. On the **AWS Management Console**, on the Services menu, choose **VPC**.

1. In the left navigation pane, under **Filter by VPC**, choose the **Select a VPC** box and select **Lab VPC**.

This setting will limit the console to only show resources that are associated with the *Lab VPC*.

1. In the left navigation pane, choose **Your VPCs**.

Here, you can access information about the **Lab VPC** that was created for you. Choose **Lab VPC**.

In the **Details** tab, notice that the **IPv4 CIDR** field has a value of *10.0.0.0/16*, which means that this VPC includes all IP addresses that start with *10.0.x.x*.

1. In the left navigation pane, choose **Subnets**.

Here, you can access information about **Public Subnet 1**. In the list of subnets view:

* + The **VPC** column for Public Subnet 1 shows that this subnet exists inside of *Lab VPC*.
  + The **IPv4 CIDR** column has a value of *10.0.0.0/24*, which means that this subnet includes the 256 IP addresses between *10.0.0.0* and *10.0.0.255*. Five of these addresses are reserved and unusable.
  + The **Availability Zone** column lists the Availability Zone where this subnet resides.

1. To reveal more details select **Public Subnet 1**.

**Tip:** To adjust the size of the lower window pane, you can drag the divider.

1. In the lower half of the page, choose the **Route table** tab.

This tab includes details about the routing for this subnet:

* + The first entry specifies that traffic destined within the Classless Inter-Domain Routing (CIDR) range for the VPC (*10.0.0.0/16*) will be routed within the VPC (*local*).
  + The second entry specifies that any traffic destined for the internet (*0.0.0.0/0*) is routed to the internet gateway (*igw-*) that exists in Lab VPC. This setting makes the subnet a *public subnet*.

1. Choose the **Network ACL** tab.

This tab has information about the network access control list (network ACL) that is associated with the subnet. The rules are currently set to the default settings. They permit *all traffic* to flow in and out of the subnet, but the rules can be further restricted by using security groups.

1. In the left navigation pane, choose **Internet gateways**.

Notice that an internet gateway with the name **Lab IG** is already attached to **Lab VPC**.

1. In the left navigation pane, choose **Security groups**.

1. Select **Inventory-DB**.

This security group controls incoming traffic to the database.

1. In the lower half of the page, choose the **Inbound rules** tab.

The rule defined in this security group permits inbound MySQL or Aurora traffic (port 3306) from anywhere in the VPC (*10.0.0.0/16*). You will later modify this setting so it only accepts traffic from the application servers.

1. Choose the **Outbound rules** tab.

By default, security groups allow all outbound traffic. However, these settings can be modified as needed.

**Task 2: Creating an Application Load Balancer**

To build a highly available application, it is a best practice to launch resources in *multiple Availability Zones*. Availability Zones are physically separate data centers (or groups of data centers) in the same Region. If you run your applications across multiple Availability Zones, you can provide greater *availability* if a data center experiences a failure.

Because the application runs on multiple application servers, you will need a way to distribute traffic amongst those servers. You can accomplish this goal by using a *load balancer*. This load balancer will also perform health checks on instances and only send requests to healthy instances.

1. On the Services menu, choose **EC2**.

1. In the left navigation pane, choose **Load Balancers** (you might need to scroll down to find it).

1. Choose Create load balancer

Several types of load balancers are displayed. Read the descriptions of each type to understand their capabilities.

1. Under **Application Load Balancer**, choose **Create**.

1. For **Load balancer name**, enter: Inventory-LB

1. Scroll down to the **Network mapping** section, then for **VPC**, select **Lab VPC**.

**Important:** Be sure to choose **Lab VPC**. It is likely not the default selection.

You will now specify which *subnets* the load balancer should use. It will be a public load balancer, so you will select both public subnets.

1. Under **Mappings**, choose the **first** Availability Zone, then choose the **Public Subnet** that displays.

1. Choose the **second** Availability Zone, then choose the **Public Subnet** that displays.

You should now have selected two subnets: **Public Subnet 1** and **Public Subnet 2**. (If not, go back and try the configuration again.)

1. In the **Security groups** section, select the **Create a new security group** hyperlink. This opens a new browser tab. Configure the new security group settings:
   * **Security group name:** Inventory-LB
   * **Description:** Enable web access to load balancer
   * **VPC**: Remove the default VPC by choosing the X to the right of it. Then select **Lab VPC**.

1. Under **Inbound rules**, choose **Add rule** and configure as described:
   * **Type:** *HTTP*
   * **Source:** *Anywhere-IPv4*

1. Still under **Inbound rules**, choose **Add rule** again and configure:
   * **Type:** *HTTPS*
   * **Source:** *Anywhere-IPv4*

1. Choose **Create security group**.

1. Assign the security group to the load balancer:
   * Return to the browser tab where you are still configuring the load balancer.
   * In the **Security groups** area and choose the refresh icon.
   * For **Security groups**, select the **Inventory-LB** security group you just created.
   * Next, below the **Security groups** dropdown menu, select the **X** next to the *default* security group so that Inventory-LB is now the only security group chosen.

1. In the **Listeners and routing** section, choose **Create target group**.

A new browser tab will open.

**Analysis**: *Target groups* define where to *send* traffic that comes into the load balancer. The Application Load Balancer can send traffic to multiple target groups based upon the URL of the incoming request, such as having requests from mobile apps going to a different set of servers. Your web application will use only one target group.

1. Configure the target group as described here:
   * **Choose a target type:** *Instances*
   * **Target group name:** Inventory-App
   * **VPC:** Ensure that *Lab VPC* is chosen.
   * Scroll down and expand **Advanced health check settings**.

**Note**: The Application Load Balancer automatically performs *health checks* on all instances to ensure that they are responding to requests. The default settings are recommended, but you will make them slightly faster for use in this lab.

* + **Healthy threshold:** 2
  + **Interval:** 10 (seconds)

The configurations you have chosen will result in the health check being performed every 10 seconds, and if the instance responds correctly twice in a row, it will be considered healthy.

* + Choose **Next**. The *Register targets* screen appears.

**Note:** *Targets* are the individual instances that will respond to requests from the load balancer.

You do not have any web application instances yet, so you can skip this step.

* + Review the settings and choose **Create target group**.

1. Return to the browser tab where you already started defining the load balancer.

1. In the **Listeners and routing** section, choose the refresh icon.

1. For the **Listener HTTP:80** row, set the **Default action** to forward to the **Inventory-App** target group you just created.

1. Scroll to the bottom of the page, and choose **Create load balancer**.
   * The load balancer is successfully created.
   * Choose **View load balancer**.

**Task 3: Creating an Auto Scaling group**

*Amazon EC2 Auto Scaling* is a service designed to *launch* or *terminate* Amazon EC2 instances automatically based on user-defined policies, schedules, and health checks. It also *automatically distributes instances across multiple Availability Zones* to make applications *highly available*.

In this task, you will create an Auto Scaling group that deploys EC2 instances across your *private subnets*, which is a security best practice for application deployment. Instances in a private subnet cannot be accessed from the internet. Instead, users send requests to the load balancer, which forwards the requests to EC2 instances in the private subnets.

**Create an AMI for Auto Scaling**

You will create an Amazon Machine Image (AMI) from the existing *Web Server 1*. This will save the contents of the root volume of the web server so that new instances can be launched with an identically configure guest operating system.

1. In the **AWS Management Console**, on the Services menu, choose **EC2**.

1. In the left navigation pane, choose **Instances**.

First, you will confirm that the instance created for you in this lab is running.

1. Wait until the **Status check** for **Web Server 1** displays *2/2 checks passed*. Choose refresh to update.

You will now create an AMI based upon this instance.

1. Select **Web Server 1**.

1. In the Actions menu, choose **Image and templates** > **Create image**, then configure:
   * **Image name:** Web Server AMI
   * **Image description:** Lab AMI for Web Server

1. Choose Create image

A banner at the top of the screen displays the **AMI ID** for your new AMI.

You will use this AMI when launching the Auto Scaling group later in the lab.

**Create a Launch Template and an Auto Scaling Group**

You will first create a *launch template*. A launch template is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch template, you specify information for the instances such as the AMI, the instance type, a key pair, and security group.

1. In the left navigation pane, choose **Launch Templates**.

1. Choose Create launch template

1. Configure the launch template settings and create it:
   * **Launch template name:** Inventory-LT
   * Under **Auto Scaling guidance**, select *Provide guidance to help me set up a template that I can use with EC2 Auto Scaling*
   * In the Application and OS Images (Amazon Machine Image) area, choose *My AMIs*.
   * **Amazon Machine Image (AMI)**: choose *Web Server AMI*
   * **Instance type:** choose *t2.micro*
   * **Key pair name**: choose *vockey*
   * **Firewall (security groups)**: choose *Select existing security group*
   * **Security groups**: choose *Inventory-App*
   * Scroll down to the **Advanced details** area and expand it.
     + **IAM instance profile**: choose *Inventory-App-Role*
     + Scroll down to the **Detailed CloudWatch monitoring** setting. Select *Enable* Note: This will allow Auto Scaling to react quickly to changing utilization.
     + Under **User data**, paste in the script below:

#!/bin/bash

# Install Apache Web Server and PHP

yum install -y httpd mysql

amazon-linux-extras install -y php7.2

# Download Lab files

wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-ACACAD-20-EN/mod9-guided/scripts/inventory-app.zip

unzip inventory-app.zip -d /var/www/html/

# Download and install the AWS SDK for PHP

wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip

unzip aws -d /var/www/html

# Turn on web server

chkconfig httpd on

service httpd start

* + Choose Create launch template

Next, you will create an Auto Scaling group that uses this launch template. The *Auto Scaling group* defines *where to launch* the EC2 instances.

1. In the Success dialog, choose the **Inventory-LT** launch template.

1. From the Actions menu, choose *Create Auto Scaling group*

1. Configure the details in Step 1 (Choose launch template or configuration):
   * **Auto Scaling group name:** Inventory-ASG (*ASG* stands for *Auto Scaling group*)
   * **Launch template**: confirm that the *Inventory-LT* template you just created is selected.
   * Choose Next

1. Configure the details in Step 2 (Choose instance launch options):
   * **VPC**: choose *Lab VPC*
   * **Availability Zones and subnets**: Choose *Private Subnet 1* and then choose *Private Subnet 2*. This will launch EC2 instances in private subnets across both Availability Zones.
   * Choose Next

1. Configure the details in Step 3 (Configure advanced options):
   * In the **Load balancing** panel:
     + Choose **Attach to an existing load balancer**
     + **Existing load balancer target groups**: select *Inventory-App*.
   * In the **Health checks** panel:
     + **Health check grace period**: 90 seconds
   * In the **Additional settings** panel:
     + Select **Enable group metrics collection within CloudWatch**

This will capture metrics at 1-minute intervals, which allows Auto Scaling to react quickly to changing usage patterns.

* + Choose Next

1. Configure the details in Step 4 (Configure group size and scaling policies - optional):
   * Under **Group size**, configure:
     + **Desired capacity:** 2
     + **Minimum capacity:** 2
     + **Maximum capacity:** 2

This will allow Auto Scaling to automatically add/remove instances, always keeping between 2 and 6 instances running.

* + Under **Scaling policies**, choose *None*

For this lab, you will *maintain two instances at all times* to ensure *high availability*. If the application is expected to receive varying loads of traffic, you can also create *scaling policies* that define when to launch or terminate instances. However, you do not need to create scaling policies for the Inventory application in this lab.

* + Choose Next

1. Configure the details in Step 5 (Add notifications - optional):

You do not need to configure any of these settings.

* + Choose Next

1. Configure the details in Step 6 (Add tags - optional):
   * Choose Add tag and Configure the following:
     + **Key:** Name
     + **Value:** Inventory-App

These settings will *tag* the Auto Scaling group with a *Name*, which will also appear on the EC2 instances that are launched by the Auto Scaling group. You can use tags to identify which Amazon EC2 instances are associated with which application. You could also add tags such as *Cost Center* to assign application costs in the billing files.

* + Choose Next

1. Configure the details in Step 6 (Review):
   * Review the details of your Auto Scaling group
   * At the bottom of the page, choose Create Auto Scaling group

Your Auto Scaling group will initially show an instance count of zero, but new instances will be launched to reach the **Desired** count of 2 instances.

The *Inventory-ASG* will appear in the console

The review shows that:

* + - The group currently has *no instances*, but the **Status** column indicates *Updating capacity*.
    - The **Desired** quantity is *2 instances*. Amazon EC2 Auto Scaling will attempt to launch two instances to reach the desired quantity.
    - The **Min** and **Max** are also set to *2 instances*. Amazon EC2 Auto Scaling will try to always provide two instances, even if failure occurs.

Your application will soon run across two Availability Zones. Amazon EC2 Auto Scaling will maintain that configuration even if an instance or Availability Zone fails.

* + After a minute, choose the refresh icon to update the display. It should show that *2 instances* are running.

**Task 4: Updating security groups**

The application you deployed is a *three-tier architecture*. You will now configure the security groups to enforce these tiers:

**Load balancer security group**

You already configured the *load balancer security group* when you created the load balancer. It accepts all incoming *HTTP* and *HTTPS* traffic.

The load balancer has been configured to forward incoming requests to a *target group*. When Auto Scaling launches new instances, it will automatically add those instances to the target group.

**Application security group**

The *application security group* was provided as part of the lab setup. You will now configure it to only accept incoming traffic from the load balancer.

1. In the left navigation pane, choose **Security Groups**.

1. Select **Inventory-App**.

1. In the lower half of the page, choose the **Inbound rules** tab.

The security group is currently empty. You will now add a rule to accept incoming *HTTP* traffic from the load balancer. You do not need to configure *HTTPS* traffic because the load balancer was configured to forward HTTPS requests via HTTP. This practice offloads security to the load balancer, reducing the amount of work that is required by the individual application servers.

1. Choose Edit inbound rules.

1. On the **Edit inbound rules** page, choose Add rule and configure these settings:
   * **Type:** *HTTP*
     + **Source:**
       - Choose the search box to the right of **Custom**
       - Delete the current contents
       - Enter sg
       - From the list that appears, select **Inventory-LB**
   * **Description:** Traffic from load balancer
   * Choose Save rules

The application servers can now receive traffic from the load balancer. This includes *health checks* that the load balancer performs automatically.

**Database security group**

You will now configure the *database security group* to only accept incoming traffic from the application servers.

1. In the **Security groups** list, choose **Inventory-DB** (and make sure that no other security groups are selected).

The existing rule permits traffic on port 3306 (used by MySQL) from any IP address within the VPC. This is a good rule, but security can be restricted further.

1. In the **Inbound rules** tab, choose Edit inbound rules and configure these settings:
   * **Delete** the existing rule.
   * Choose **Add rule**.
   * For **Type**, choose **MYSQL/Aurora**
   * Choose the search box to the right of **Custom**
   * Enter sg
   * From the list that appears, select **Inventory-App**
   * **Description:** Traffic from application servers
   * Choose Save rules

You have now configured *three-tier security*. Each element in the tier only accepts traffic from the tier above.

In addition, the use of private subnets means that you have two security barriers between the internet and your application resources. This architecture follows the best practice of applying multiple layers of security.

**Task 5: Testing the application**

Your application is now ready for testing.

In this task, you will confirm that your web application is running. You will also test that it is highly available.

1. In the left navigation pane, choose **Target Groups**.

1. Select **Inventory-App**.

1. In the lower half of the page, choose the **Targets** tab.

This tab should show two *registered targets*. The **Health status** column shows the results of the load balancer health check that is performed against the instances.

1. In the **Registered targets** area, occasionally choose the refresh icon until the **Status** for both instances appears as *healthy*.

If the status does not eventually change to *healthy*, ask your educator for help with diagnosing the configuration.

You will test the application by connecting to the load balancer, which will then send your request to one of the EC2 instances. You will first need to retrieve the Domain Name System (DNS) name of the load balancer.

1. In the left navigation pane, choose **Load Balancers** and then choose **Inventory-LB**.

1. In the **Details** tab in the lower half of the window, copy the **DNS name** to your clipboard.

It should be similar to: *inventory-LB-xxxx.elb.amazonaws.com*

1. Open a new web browser tab, paste the DNS name from your clipboard and press ENTER.

The load balancer forwarded your request to one of the EC2 instances. The instance ID and Availability Zone are shown at the bottom of the webpage.

1. Reload the page in your web browser. You should notice that the instance ID and Availability Zone sometimes toggles between the two instances.

When this web application displays, the flow of data over the network is:

* + You sent the request to the *load balancer*, which resides in the *public subnets* that are connected to the internet.
  + The load balancer chose one of the *EC2 instances* that reside in the *private subnets* and forwarded the request to it.
  + The EC2 instance then returned the webpage content to the load balancer, which returned it to your web browser.

**Task 6: Testing high availability**

Your application was configured to be highly available. You can prove the application's high availability by terminating one of the EC2 instances.

1. Return to the **EC2 console** tab in your web browser (but do not close the web application tab—you will return to it soon).

1. In the left navigation pane, choose **Instances**.

You will now terminate one of the web application instances to simulate a failure.

1. Select one of the **Inventory-App** instances (it does not matter which one you select).

1. Choose **Instance State > Terminate instance**.

1. Choose **Terminate**.

In a short time, the load balancer health checks will notice that the instance is not responding. The load balancer will automatically route all requests to the remaining instance.

1. Return to the web application tab in your web browser and reload the page several times.

You should notice that the *Availability Zone* that is shown at the bottom of the page stays the same. Though an instance failed, your application remains available.

After a few minutes, Amazon EC2 Auto Scaling will also notice the instance failure. It was configured to keep two instances running, so Amazon EC2 Auto Scaling will *automatically launch a replacement instance*.

1. Return to the **EC2 console** tab where you have the instances list displayed. In the top-right area, choose the refresh icon every 30 seconds or so until a new EC2 instance appears.

After a few minutes, the health check for the new instance should become healthy. The load balancer will resume sending traffic between the two Availability Zones. You can reload your web application tab to see this happen.

This task demonstrates that your application is now *highly available*.

**Optional task 1: Making the database highly available**

*This task is* ***optional****. You can work on this task if you have remaining lab time.*

The application architecture is now highly available. However, the Amazon RDS database operates from only one database instance.

In this optional task, you will make the database highly available by configuring it to run across multiple Availability Zones (that is, in a *Multi-AZ deployment*).

1. On the Services menu, choose **RDS**.

1. In the left navigation pane, choose **Databases**.

1. Choose the link for the name of the **inventory-db** instance.

Feel free to explore the information about the database.

1. Choose Modify

1. Scroll down to the **Availability & durability** section. For **Multi-AZ deployment**, select **Create a standby instance**.

*Analysis*: You only need to reconfigure this one setting to convert the database to run across multiple data centers (Availability Zones).

This option does not mean that the database is *distributed* across multiple instances. Instead, one instance is the *primary* instance, which handles all requests. Another instance will be launched as the *standby* instance, which takes over if the primary instance fails. Your application continues to use the same DNS name for the database. However, the connections will automatically redirect to the currently active database server.

You can scale an EC2 instance by changing attributes, and you can also scale an RDS database this way. You will now scale up the database.

1. Scroll back up and for **DB instance class**, select **db.t3.small**.

This action doubles the size of the instance.

1. For **Allocated storage**, enter: 10

This action doubles the amount of space that is allocated to the database.

Feel free to explore the other options on the page, but do not change any other settings.

1. At the bottom of the page, choose Continue

Database performance will be impacted by these changes. Therefore, these changes can be scheduled during a defined maintenance window, or they can be run immediately.

1. Under **Schedule modifications**, select **Apply immediately**.

1. Choose Modify DB instance

The database enters a *modifying* state while it applies the changes. You do not need to wait for it to complete.

**Optional task 2: Configuring a highly available NAT gateway**

*This task is* ***optional****. You can work on this task if you have remaining lab time.*

The application servers run in a private subnet. If the servers must access the internet (for example, to download data), the requests must be redirected through a *Network Address Translation (NAT) gateway*. (The NAT gateway must be located in a public subnet).

The current architecture has only one NAT gateway in *Public Subnet 1*. Thus, if Availability Zone 1 fails, the application servers will not be able to communicate with the internet.

In this optional task, you will make the NAT gateway highly available by launching another NAT gateway in the other Availability Zone. The resulting architecture will be highly available:

1. On the Services menu, choose **VPC**.

1. In the left navigation pane, choose **NAT gateways**.

The existing NAT gateway displays. You will now create a NAT gateway for the other Availability Zone.

1. Choose Create NAT gateway and configure these settings:
   * **Subnet:** *Public Subnet 2*
   * Choose Allocate Elastic IP
   * Choose Create NAT gateway

You will now create a new route table for *Private Subnet 2*. This route table will redirect traffic to the new NAT gateway.

1. In the left navigation pane, choose **Route tables**.
2. Choose Create route table and configure these settings:
   * **Name:** Private Route Table 2
   * **VPC:** *Lab VPC*
   * Choose **Create route table**.

1. Observe the settings in the **Routes** tab.

Currently, one route directs all traffic *locally*. You will now add a route to send internet-bound traffic through the new NAT gateway.

1. Choose Edit routes and then configure these settings:
   * Choose Add route
   * **Destination:** 0.0.0.0/0
   * **Target:** Select *NAT Gateway*, then select the *nat-* entry that is *not* the entry for \_NATGateway1

**Tip**: To discover which nat- entry is *NOT* the one to select, choose the **Details** button above these instructions and then select **Show**. the name of NATGateway1 is displayed. Return to the VPC console and select the Target that is *NOT* NATGateway1.

* + Choose **Save changes**

1. Choose the **Subnet associations** tab.

1. Choose Edit subnet associations

1. Select **Private Subnet 2**.

1. Choose Save associations

This action now sends internet-bound traffic from Private Subnet 2 to the NAT gateway that is in the same Availability Zone.

Your NAT gateways are now highly available. A failure in one Availability Zone will not impact traffic in the other Availability Zone.

**Submitting your work**

1. At the top of these instructions, choose Submit to record your progress and when prompted, choose **Yes**.

1. If the results don't display after a couple of minutes, return to the top of these instructions and choose Grades

**Tip**: You can submit your work multiple times. After you change your work, choose **Submit** again. Your last submission is what will be recorded for this lab.

1. To find detailed feedback on your work, choose Details followed by **View Submission Report**.

**Lab complete**

Congratulations! You have completed the lab.

1. Choose End Lab at the top of this page, and then select Yes to confirm that you want to end the lab.

A panel indicates that *DELETE has been initiated... You may close this message box now.*

1. Select the **X** in the top right corner to close the panel.

*©2023 Amazon Web Services, Inc. and its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited.*