Architecting on AWS
Lab Guide
Version 5.0

100-ARC-50-EN

Corrections or feedback on the course, please email us at:
aws-course-feedback@amazon.com.
For all other questions, contact us at:
https://aws.amazon.com/contact-us/aws-training/.

Contents

Accessing the Lab Environment 4

Lab 1: Deploying a Web Application on AWS 6

Lab 2: Making Your Environment Highly Available 28

Lab 3: Using Auto Scaling with AWS Lambda 59

Lab 4: Implementing a Serverless Architecture with AWS Managed Services 69

Lab 5: Multi-Region Failover with Route 53 84

Accessing the Lab Environment

The labs for this course are run in the qwikLABS lab environment. QwikLABs offers a complete end-to-end cloud platform for hands-on software training lab creation, management, and consumption. The labs can be delivered anywhere, anytime, on-demand.

1. Open the qwikLABS link provided by your instructor.
2. Log in with your credentials or register for an account.
3. You should have access to the classroom for your course. The classroom will contain links to each of the labs for your class. Follow these steps to launch each of these labs in your course as directed by your instructor.
4. On the left side of the page, select the lab you would like to start.
5. On the right side of the page, click **Start Lab** to launch your lab.

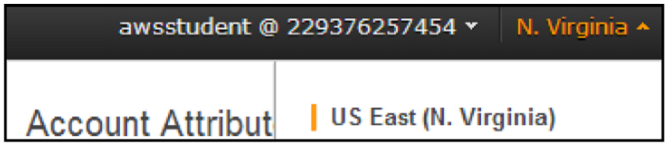
   Note If your lab requires setup time, a status bar shows the progress of the lab environment creation process. The AWS Management Console is accessible during lab creation, but your AWS resources may not be fully available until the process is complete. **You should not start your lab until the lab has finished being created.**

6. On the lab details page, notice the lab properties:
   - **Time Remaining**: The amount of time left to complete the lab.
   - **Setup Time**: The estimated time required to set up the lab environment.
   - **Duration**: The expected completion time for the lab.
   - **Access**: The time the lab will run before automatically shutting down.
7. In the **AWS Management Console** section of the qwikLABS page, copy the **Password** to the clipboard.

8. Click the Open Console button.
9. Log into the AWS Management Console using the following:
   - In the **User Name** field type **awsstudent**
   - In the **Password** field, paste the password copied from the lab details page.
   - Click **Sign in using our secure server**.

Note The AWS account is automatically generated by qwikLABS. Also, the login credentials for the awsstudent account are provisioned by qwikLABS using AWS Identity Access Management.

10. In the AWS Management Console, click **EC2**.
11. Make a note of the **AWS Region** in the AWS Management Console menu bar (beside your User Name).



Lab 1
Deploying a Web Application on AWS

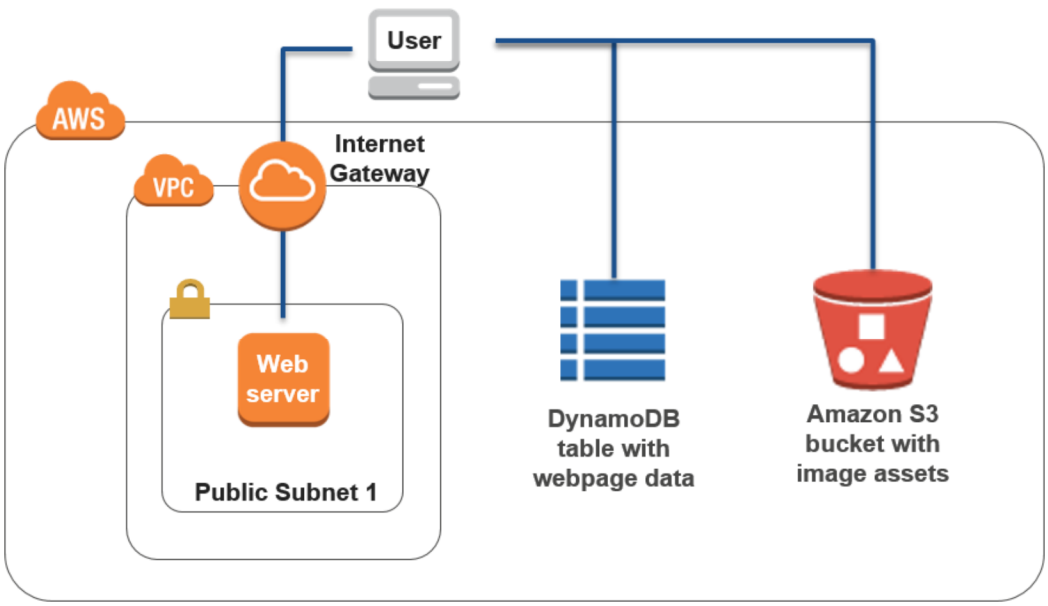
Overview

In this lab you will use IAM, EC2, S3, VPC and DynamoDB to deploy a scalable web application.

Objectives

After completing this lab, you will be able to:

- Create an IAM User and attach a permission policy for controlling access to services
- Create a Virtual Private Cloud (VPC) with an Internet Gateway (IGW) and a Public Subnet
- Create an S3 Bucket and upload static objects into the bucket
- Create a DynamoDB table and populate with items
- Deploy a web application in a cost optimized and scalable manner utilizing EC2, S3, and DynamoDB
- The final product of your lab will be the following application environment:



Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
    - The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: Administrator access to the computer
- An Internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)
- An SSH client such as PuTTY

Duration

This lab will require around **90 minutes** to complete.

Task 1: Create IAM Users and Roles

Overview

In this part of the lab, you will create a new IAM User, assign permissions based on the least privilege principle, and create a role for your EC2 instance which will allow the web application to reach out to multiple AWS services.

Task 1.1: Create IAM User and Role

In this part of the lab, you will create a new IAM User, assign permissions based on the least privilege principle, and create a role for your EC2 instance.

1.1.1 In the **AWS Management Console**, for **Services**, click **IAM**.

1.1.2 In the navigation pane, click **Users**.

1.1.3 Click **Create New Users**.

1.1.4 For the first **Enter User Names**, type **Lab1User**

1.1.5 If it is not already selected, select **Generate an access key for each user**.

1.1.6 Click **Create**.

1.1.7 Click **Download Credentials** to download the user's credentials.

1.1.8 Return to your browser and click **Close** to return to the **Users** menu in the **IAM Management Console**.

1.1.9 Select **Lab1User**.

1.1.10 For **User Actions**, click **Manage Password**.

1.1.11 If it is not already selected, click **Assign an auto-generated password**. Leave the rest as the default.

Click **Apply**.

A new page confirms that IAM has generated the user's password.

1.1.12 Click **Show User Security Credentials**.

1.1.13 Open the **credentials** file you saved earlier and add a **Password** column to the right of the other three columns.

1.1.14 Paste the password generated by IAM for **Lab1User** into the **Password** column of the **Lab1User** entry.

1.1.15 In the **IAM Management Console**, click **Close**.

1.1.16 IAM asks you to confirm the closing of the window because you haven't downloaded the user's password. Because you already copied the password to the **credentials** file, go ahead and click **Close** again to close the window.

Task 1.2: Set Permissions for Lab1User

In this section of the lab, you will grant the appropriate permissions to your Lab1User IAM user.

1.2.1 Return to the **Users** menu in the **IAM Management Console** if you are not already there.

1.2.2 From the list of users, click the name of **Lab1User** to open the **Summary** page.

1.2.3 Click the **Permissions** tab.

1.2.4 Click **Attach Policy**.

Tip To find a policy in the list of policies, you can use the **Search** box at the top of the list to locate any policy more easily. You can use partial names like “S3” to find any policies with “S3” in the name.

1.2.5 Locate the entry marked **AmazonS3FullAccess** under the **Policy Name** column. Select the check box for this entry.

This policy grants the selected user full access to all Amazon S3 functions.

1.2.6 Locate the entry marked **AmazonEC2FullAccess** under the **Policy Name** column. Select the check box for this entry.

This policy grants the selected user full access to all Amazon EC2 functions.

1.2.7 Locate the entry marked **IAMReadOnlyAccess** under the **Policy Name** column. Select the check box for this entry.

This policy grants the selected read-only access to all Amazon IAM resources.

1.2.8 Locate the entry marked **AmazonVPCFullAccess** under the **Policy Name** column. Select the check box for this entry.

This policy grants the selected user full access to all Amazon VPC functions.

1.2.9 Locate the entry marked **AmazonDynamoDBFullAccess** under the **Policy Name** column. Select the check box for this entry. This policy grants the selected user full access to all Amazon DynamoDB functions.

1.2.10 Click **Attach Policy**.

1.2.11 Verify that **AmazonS3FullAccess**, **AmazonEC2FullAccess**, **IAMReadOnlyAccess**, **AmazonVPCFullAccess**, and **AmazonDynamoDBFullAccess** are listed under **Policy Name** within the **Permissions** group.

1.2.12 Next we need to add an Inline Policy to allow the IAM Role to be assumed by a resource. At the bottom of the page, click **Inline Policies**.

1.2.13 Click the **click here** link to create a new policy.

1.2.14 You will use the **Policy Generator** to create a Custom Policy to attach to your IAM Role. Verify that the **Policy Generator** is selected and in the row for Policy Generator, click **Select**.

1.2.15 Configure the following attributes for the policy using the Policy Generator:

Effect	Allow
AWS Service	AWS Identity and Access Management
Actions	PassRole
Amazon Resource Name (ARN)	*

1.2.16 Click **Add Statement** to create the policy.

1.2.17 Click **Next Step**.

1.2.18 Review the **Policy Document** that was generated and click **Apply Policy**.

Task 1.3: Create an IAM Role

In this part of the lab, you will create an **IAM Role** within **AWS IAM**. An IAM Role is an IAM entity that defines a set of permissions for making AWS service requests. IAM Roles are not associated with a specific user or group. Instead, roles are assumed by trusted entities such as IAM users, applications, or AWS services such as Amazon EC2.

You are going to create an **AWS Service Role** in **AWS IAM** and attach two **AWS Managed Policies** to the role. The role will be used to delegate permissions for a **Trusted Entity, Amazon EC2.** You will assign the role to an **Amazon EC2 Instance** so that the applications running on the instance can access **Amazon S3** and **Amazon DynamoDB**. This means that the Amazon EC2 Instance resource can access and manipulate objects in Amazon S3, and create tables and add items to Amazon DynamoDB.

1.3.1 In the navigation pane, click **Roles**.

1.3.2 Click **Create New Role**.

1.3.3 For **Role Name**, type **WebServerRole**

1.3.4 Click **Next Step**.

1.3.5 Under **Select Role Type**, verify that the **AWS Service Roles** option is selected.

1.3.6 In the row for **Amazon EC2**, click **Select**.

1.3.7 Select the **AmazonS3FullAccess** policy from the list.

1.3.8 Select the **AmazonDynamoDBFullAccess** policy from the list.

1.3.9 Click **Next Step**.

1.3.10 Click **Create Role**.

Task 1.4: Switch to your IAM user and create a bucket

In this section of the lab, you will switch over to Lab1User and verify that it has the appropriate permissions by creating the Amazon S3 bucket you will use for this lab.

1.4.1 In the navigation pane, click **Dashboard**.

1.4.2 Copy the entire URL displayed below **IAM users sign-in link**.

Paste the URL into a text file and save the file on your local computer so you can reference it later in the lab.

1.4.3 Open a new browser window or tab and navigate to the AWS Account Alias URL that you copied in step **1.4.2**.

1.4.4 For **User Name**, type **Lab1User**

1.4.5 For **Password**, paste the password for **Lab1User** that you saved in your **credentials** file

1.4.6 Click **Sign In**.

1.4.7 Once the **AWS Management Console** opens, you will be logged out automatically from the first account you logged in with. You can now close your previous window or tab and use your **Lab1User** account for the rest of this lab.

1.4.8 On the **Services** menu, click **S3.**

1.4.9 Click **Create Bucket**.

1.4.10 In the **Create a Bucket – Select a Bucket Name and Region** dialog box:

- For **Bucket Name**, type a unique bucket name (e.g., a name with today's date or your initials inside of it, with no uppercase letters).
- For **Region**, click the region you want to create the bucket in. We recommend leaving it as the default region.

Note Bucket names must:

- Be unique across all existing bucket names in Amazon S3
- Be at least 3 characters and no more than 63 characters long
- Contain only lowercase letters, numbers, and hyphens.

1.4.11 Click **Create**. Verify that the bucket is created. This validates that your permissions for Amazon S3 are set properly.

1.4.12 Click the bucket you just created. Verify that the bucket is currently empty.

Task 1.5: Customize your command reference file

In this part of the lab, you will download your lab's command reference file and edit it so your commands are customized with your environment's region code and s3 bucket name.

1.5.1 Return to your qwikLABS page.

1.5.2 If the green **Connection** menu is open, click **Connection** to retract it.

1.5.3 Right-click the **Lab01-webapp-CommandRef.txt** and save it locally. You may need to open it in a new tab and copy all of the text into a new text editor file.

1.5.4 Open **Lab01-webapp-CommandRef.txt** in a text editor.

In this text file, you will find commands that will be used in later steps. To make sure they are configured correctly, you will need to replace some of the text according to the following steps.

1.5.5 Replace all instances of **<bucket_name>** with the name of the bucket you just created. **Be sure to replace the angle brackets (< >) as well.**

1.5.6 Return to your **AWS Management Console** browser window/tab.

1.5.7 Click the orange box icon in the upper left corner to return to your console home page.

1.5.8 In the upper-right corner, to the right of your username, you will see the name of the region where your lab is running. Use this table to find your region's code:

Region Name	Region Code
N. Virginia	us-east-1
N. California	us-west-1
Oregon	us-west-2
Ireland	eu-west-1
Tokyo	ap-northeast-1
Seoul	ap-northeast-2
Singapore	ap-southeast-1
Sydney	ap-southeast-2
São Paulo	sa-east-1

1.5.9 Return to Lab01-webapp-CommandRef.txt and replace every instance of **<region>** with your region code as specified above. Be sure to replace the angle brackets (< >) as well.

Task 2: Create the Network Layer

In this part of the lab, you will create a basic Virtual Private Cloud (VPC) with a single Public Subnet. You will attach an Internet Gateway (IGW) to the VPC to allow resources in the VPC to communicate across the Internet.

When you first sign in to the AWS Management Console and launch VPC Dashboard, you will notice that there is an existing VPC; this is the default VPC. A default VPC is a logically isolated virtual network in the AWS cloud that is automatically created for your AWS account the first time you provision Amazon EC2 resources. When you launch an instance without specifying a subnet ID, your instance will be launched in your default VPC.

In this part of the lab, you will create a VPC with a public subnet and a user-specified IP address range.

Task 2.1: Create the VPC

In this section of the lab, you will create your VPC.

2.1.1 On the **Services** menu, click **VPC**.

Note You can select your desired region from the drop-down list on the navigation bar. For now, let it remain as the default.

2.1.2 In the navigation pane, click **Your VPCs**.

2.1.3 Click **Create VPC**.

2.1.4 In the **Create VPC** dialog box, enter the following settings:

Name tag: **LabVPC**

CIDR block: **10.200.0.0/16**

Tenancy: **Default**

2.1.5 Click **Yes, Create**.

You should see a new VPC named **LabVPC** with a **VPC ID** assigned to it (e.g., *vpc-530de336*).

Task 2.2: Attach an Internet Gateway (IGW)

In this section of the lab, you will create an Internet gateway and attach it to your VPC so that your VPC will be accessible via the Internet.

2.2.1 In the navigation pane, click **Internet Gateways**.

2.2.2 Click **Create Internet Gateway**.

2.2.3 For **Name tag**, type **LabVPCGateway**

2.2.4 Click **Yes, Create**.

At this point, the newly created **LabVPCGateway** is not attached to your VPC. Note the ID (e.g., *igw-912a31f3*).

2.2.5 If it is not already selected, select the newly created **LabVPCGateway**, and then click **Attach to VPC**.

2.2.6 In the **Attach to VPC** dialog box, for **VPC**, click the **LabVPC** that you created in **Task 2.1.**

2.2.7 Click **Yes, Attach**.

The **State** for the **LabVPCGateway** should change to *attached*, and the VPC ID in the **VPC** column should match the ID for **LabVPC**.

Task 2.3: Create the Public Subnet

You have complete control over your virtual networking environment, including selection of your own IP address range and subnets. A **Subnet** is a segment of a VPCs IP address range where you can place groups of isolated resources.

In this task, you are going to configure your **Amazon VPC** with a single subnet so you can provision resources in the subnet and be able to provide access to the **Internet Gateway**. The direct attachment of a resource that connects to the Internet makes this subnet a **Public Subnet**.

2.3.1 In the navigation pane, click **Subnets**.

2.3.2 Click **Create Subnet**.

2.3.3 In the **Create Subnet** dialog box:

- Name tag: **PublicSubnet1**
- VPC: Click the VPC that includes the name **LabVPC.**
- Availability Zone: Click the first AZ (e.g., *us-west-2a*).
- CIDR block: **10.200.10.0/24**

2.3.4 Click **Yes, Create**.

Task 2.4: Configuring the Route Table

A route table contains a set of rules called routes that are used to determine where network traffic is directed. Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can be associated with only one route table at a time, but you can associate multiple subnets with the same route table.

When you create a VPC, it automatically has a main route table. Initially, the main route table contains only a single route: a local route that enables communication within the VPC. If you don't explicitly associate a subnet with a route table, the subnet is implicitly associated with the main route table.

In this section of the lab, you will create a route table that allows incoming and outgoing traffic through the Internet gateway you created earlier.

2.4.1 In the navigation pane, click **Route Tables**.

2.4.2 Click **Create Route Table**.

2.4.3 In the **Create Route Table** dialog box:

Name tag: **PublicRoute**

VPC: **LabVPC**

2.4.4 Click **Yes, Create**.

2.4.5 If it is not already selected, select the **PublicRoute** route table you just created, and then click the **Routes** tab in the lower pane of the console.

2.4.6 Click **Edit**.

2.4.7 Click **Add another route**.

2.4.8 For **Destination**, type **0.0.0.0/0**

2.4.9 Click in the **Target** box, and then click the **LabVPCGateway** that you created earlier (the ID starts with *igw-*).

2.4.10 Click **Save**.

2.4.11 With **PublicRoute** still selected, click the **Subnet Associations** tab.

2.4.12 Click **Edit**.

2.4.13 Select **PublicSubnet1** (CIDR range of 10.200.10.0/24).

2.4.14 Click **Save**.

Task 3: Build the Compute, Storage and Database Layer

Overview

In this part of the lab, you will create an **Amazon Elastic Compute Cloud (EC2)** Instance that will function as the web server for your web application. This box will communicate with the **S3 Bucket** you built in Task 1 and with an **Amazon DynamoDB** table that will be created in this task. These three services will provide the web application with compute, storage, and database services.

Command Reference File

On your qwikLABS page, you can find a tab which leads to a link to a text file containing any long commands or scripts required in this lab. When instructed to input long commands or scripts, we strongly encourage you to copy and paste them from the provided Command Reference File rather than typing them in manually. Copying from our lab guides has been disabled as doing so frequently leads to errors.

Task 3.1: Create the web server

In this section of the lab, you will create an Amazon EC2 instance in PublicSubnet1.

3.1.1 On the **Services** menu, click **EC2**.

3.1.2 Click **Launch Instance**.

3.1.3 To launch a new instance, you first need to select an Amazon Machine Image (AMI), which is a preconfigured template for an instance in the cloud.

From the **Quick Start** menu, in the row for the first **Amazon Linux AMI**, click **Select**.

3.1.4 On the **Choose an Instance Type** page, you can select the family for your image, which determines how much RAM, storage, and processing speed your instance will have.

To accept the default (**t2.micro**), click **Next: Configure Instance Details**.

3.1.5 On the **Configure Instance Details** page, make these selections

Network: **LabVPC**

Subnet**: PublicSubnet1**

Auto-assign Public IP: **Enable**

IAM role: **WebServerRole** (which you created in Task 1.3)

3.1.6 Click **Advanced Details** to expand it. Copy the contents of the user data script given below from the Command Reference File you have open, and paste the script into the **User data** box.

```
#!/bin/bash
yum remove -y httpd php
yum install -y httpd24 php56
chkconfig httpd on
wget https://d2lrzjb0vjvpn5.cloudfront.net/AWS-100-ARC/v5.0/lab-
1-webapp/scripts/lab1src.zip
unzip lab1src.zip -d /home/ec2-user/
mv /home/ec2-user/lab1src/index.php /var/www/html/index.php
mv /home/ec2-user/lab1src/challenge-me.php
/var/www/html/challenge-me.php
wget https://github.com/aws/aws-sdk-
php/releases/download/3.15.9/aws.zip
unzip aws -d /var/www/html
service httpd start
```

This Linux script installs a specific httpd and PHP package, downloads the source code, images, and scripts for the web application, pulls down the AWS-SDK for PHP and starts the httpd service.

3.1.7 Click **Next: Add Storage**.

You won't be using the storage on this instance, so you are leaving the instance's storage settings as their default.

3.1.8 Click **Next: Tag Instance**.

3.1.9 For **Value**, type **WebServer**

3.1.10 Click **Next: Configure Security Group**.

3.1.11 For **Assign a security group**, verify that the **Create a new security group** option is selected.

Security group name: **WebServerSG**

Description: **Web Server Security Group**

3.1.12 There should be an existing **SSH** rule. Leave that rule as-is.

3.1.13 Click **Add Rule**.

3.1.14 In the new row that appears, specify the following:

For **Type**, click **HTTP**.

For **Source**, click **Anywhere**.

3.1.15 Click **Review and Launch**.

3.1.16 Review the settings and then click **Launch**.

3.1.17 When prompted, accept the *qwikLABS* keypair, select the acknowledgement check box, and then click **Launch Instances**.

3.1.18 Click **View Instances**.

3.1.19 Select the **WebServer** instance you just created.

3.1.20 On the **Description** tab in the lower pane, note the **Public IP** of the instance.

3.1.21 Wait for the **WebServer** to reach **Instance State:** *running* and **Status Checks:** *2/2 checks passed.*

Task 3.2: Connect to WebServer (Windows only)

Note This section is for **Windows** users only. If you are running OSX or Linux, skip to **Task 3-3**.

In this section of the lab, you will download your keypair and use it to connect to your Amazon EC2 instance with PuTTY.

3.2.1 From the qwikLABS page in your browser, in the **Connection Details** section, for **Download PEM/PPK**, click **Download PPK**.

3.2.2 Save the file to your **`\Downloads`** folder or any other easy to access location on your local computer.

3.2.3 Download **PuTTY** from https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

3.2.4 Launch **PuTTY** by running the **putty.exe** file you just downloaded.

3.2.5 For **Host Name**, enter the **Public IP** address from your Web Server instance which you copied into a text editor earlier in the lab.

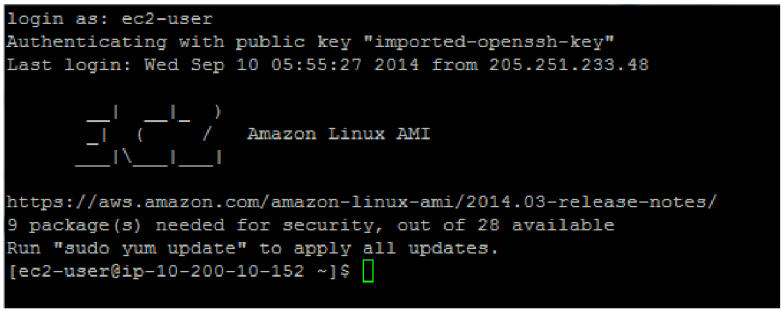
3.2.6 In the **Connection** list, expand **SSH**.

3.2.7 Click **Auth**.

3.2.8 For **Private key file for authentication**, browse to the **.ppk** file that you downloaded in step 3.2.1, then click **Open**.

3.2.9 In the **PuTTY Security Alert** dialog box that opens, click **Yes** to add the key to **PuTTY**'s cache.

3.2.10 For **login as:** type **ec2-user** and press ENTER. You are now logged in to your **Web Server** instance.



Task 3.3: Connect to the WebServer (Linux/OSX only)

Note This section is for **Linux** and **Mac OSX** users only. If you are running **Windows** but have not yet connected to your instance, go back to **Task 3.2**. If you have already connected to your instance, skip ahead to **Task 3.4**.

In this section of the lab, you will download your keypair and use it to connect to your Amazon EC2 instance.

3.3.1 From the qwikLABS page in your browser, in the **Connection Details** section, for **Download PEM/PPK**, click **Download PEM**.

3.3.2 Save the file to your local computer in a place where you can easily access it.

3.3.3 To connect to your EC2 instance, run the following commands in Terminal:

```
chmod 400 <path and name of pem>

ssh -i <path and name of pem> ec2-user@<Public IP>
```

For ***<path and name of pem>***, substitute the path/filename to the .pem file you downloaded.

For ***<Public IP>***, substitute the public IP address for your **Web Server** instance which you copied into a text editor earlier in the lab.

Task 3.4: Configure your server and the DynamoDB table

In this part of the lab, you will use commands to set the bucket and region variables in your site's index.php file, and then create and populate a DynamoDB table which will hold your Items. This table will include the text content for services and acronyms and an object key for the image that is stored in S3.

The images and scripts have been downloaded to the /home/ec2-user/lab1src directory.

The web page (index.php) is located in /var/www/html.

3.4.1 Return to the Command Reference File you edited earlier. You should have already replaced **<region>** and **<bucket_name>** with their specific values in Task 1.5. If you have not made these changes, please return to that task and follow the steps there.

3.4.2 Verify that your S3 bucket is currently empty and that your Instance is able to access the bucket (through the IAM Role) by executing the following command from the AWS CLI:

```
aws s3 ls <bucket_name> --region <region>
```

3.4.3 Verify that you are working out of the /home/ec2-user/lab1src directory by executing the following command:

```
cd /home/ec2-user/lab1src
```

3.4.4 Copy all static website resources (images, JavaScript files, CSS stylesheets) to the S3 bucket by executing the following command:

```
aws s3 cp /home/ec2-user/lab1src/jquery/
s3://<bucket_name>/jquery/ --recursive --acl public-read --
region <region>
```

3.4.5 Copy all static icon images up to the S3 bucket by executing the following command:

```
aws s3 cp /home/ec2-user/lab1src/images/
s3://<bucket_name>/images/ --recursive --acl public-read --
region <region>
```

3.4.6 Verify that all objects were copied up to the S3 bucket by executing the following command:

```
aws s3 ls s3://<bucket_name>/ --region <region> --recursive
```

3.4.7 Execute the following commands one at a time to set the values for the `region` and `bucket` variables in your index.php file.

```
sudo sed -i '2s/%region%/<region>/g' /var/www/html/*.php

sudo sed -i '3s/%bucket%/<bucket_name>/g' /var/www/html/*.php
```


3.4.8 Return to the AWS console and navigate to **DynamoDB** using the **Services** menu at the top of the page.

3.4.9 Click **Create Table**.

3.4.10 Type the following for **Table name**:

- **AWS-Services**

3.4.11 In the text box below **Partition key**, type **Category**

3.4.12 Select **Add sort key**.

3.4.13 In the new text box that appears, type **Name**

3.4.14 Ensure **Use default settings** is selected and click **Create**.

3.4.15 Return to your SSH session. There you will run the commands listed in the next steps in order to execute a bulk write of items to your DynamoDB table.

3.4.16 Populate the DynamoDB table by running the following three commands one at a time:

```
aws dynamodb batch-write-item --request-items file:///home/ec2-
user/lab1src/scripts/services1.json --region <region>

aws dynamodb batch-write-item --request-items file:///home/ec2-
user/lab1src/scripts/services2.json --region <region>

aws dynamodb batch-write-item --request-items file:///home/ec2-
user/lab1src/scripts/services3.json --region <region>
```

After running each command, you should receive the following output:

```
{
      "UnprocessedItems": {}
}
```

3.4.17 Return to the console and refresh the **AWS-Services** table's **Items** page to verify that the items have populated.

TIP You can also scan the table using the command line by executing the following:

```
aws dynamodb scan --table-name AWS-Services --region <region>
```

3.4.18 Return to your web browser and reload the web page using your web server's public IP as the URL. You should see the listing of services from DynamoDB, the service icons from S3, and a formatted page using the javascript and css resources that we also stored in S3.

Challenge

In the top right corner of the webpage you created, click the **Challenge Me** button. You will be redirected to a page that is also hosted by your webserver and uses the resources that you loaded into your S3 bucket and the DynamoDB items you wrote to your table. Take the challenge by dragging the Services listed in the left container and drop them in the correct Service category in the right container. Good luck!

End of Lab

Please return to the qwikLABS page for this lab and click **End** to clean up your lab environment.

Congratulations! You have completed Lab 1.

Lab 2
Making Your Environment Highly Available

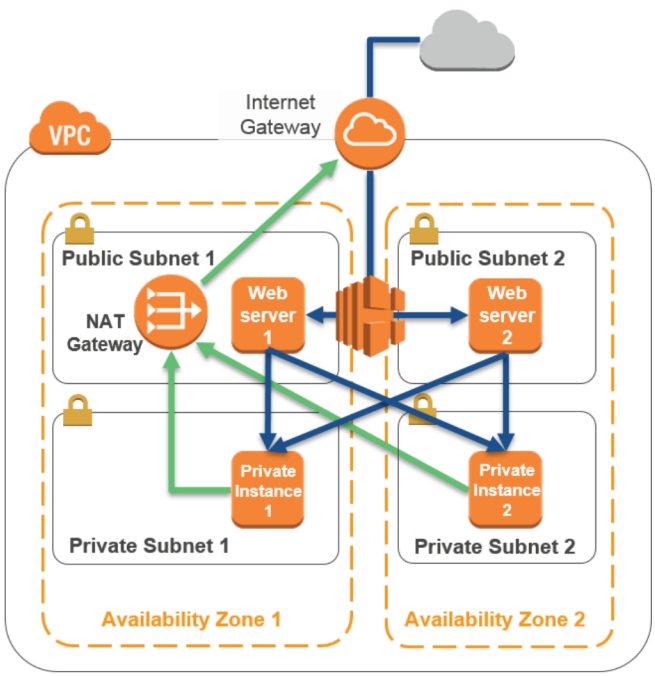
Overview

In this lab, you will start with an Amazon EC2 instance inside of a public subnet (similar to what you created in the last lab), convert it into a basic PHP server, make your environment highly available, and add a highly available private tier that sits behind an AWS NAT Gateway.

Objectives

After completing this lab, you will be able to:

- Create an image of an existing Amazon EC2 instance and use it to launch a new instance.
- Create an Amazon ELB load balancer and attach it to Amazon EC2 instances.
- Create an AWS NAT Gateway.
- Create private subnets and launch Amazon EC2 instances into them.
- Edit private subnet route tables and security groups to intelligently control access.
- Test an AWS NAT Gateway.
- The final product of your lab will be this:



Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
  - The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: Administrator access to the computer
- An Internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)
- An SSH client such as PuTTY

Duration

The lab will require approximately **60 minutes** to complete.

Task 1: Inspect Your Lab Environment

Overview

Unlike the last lab, this lab provides you with an environment already containing a few resources. Via CloudFormation, we've already launched and configured these things for you:

- A VPC (LabVPC) with a public subnet (PublicSubnet1)
- An Internet gateway attached to PublicSubnet1
- An Amazon EC2 instance (Web Server 1) inside of PublicSubnet1

Command Reference File

On your QwikLAB page, you can find a tab which leads to a link to a text file containing any long commands or scripts required in this lab. When instructed to input long commands or scripts, we strongly encourage you to copy and paste them from the provided Command Reference File rather than typing them in manually. Copying from our lab guides has been disabled as doing so frequently leads to errors.

Task 1.1: Inspect Your VPC

Your VPC has been created for you via CloudFormation. In this part of the lab, you will be guided through the VPC to see the various components that have already been set up.

1.1.1 On the **AWS Management Console**, on the **Services** menu, click **VPC**.

1.1.2 In the **VPC dashboard**, you can see what has already been set up for you: 2 VPCs, 4 subnets, 3 Network ACLs, 1 running instance, 2 Internet gateways, 3 route tables, and 3 security groups.

This list includes both the default VPC, subnets, Internet gateway, and route table that comes with any new AWS account, as well as the resources created via CloudFormation for the purposes of this lab.

1.1.3 In the navigation pane, click **Your VPCs**.

1.1.4 Here you can see the VPC that has been created for you via CloudFormation (LabVPC).

In the **CIDR** column, you can see a value of **10.200.0.0/20**, which means this VPC includes 4,096 IPs between 10.200.0.0 to 10.200.15.255 (with some reserved and unusable).

It is also attached to a route table, and includes a Network ACL.

This VPC also has a default tenancy value, which means unless otherwise specified, instances launched into this VPC will use shared tenancy hardware.

You can also see that this lab environment includes a default VPC. We **do not recommend** that you delete your default VPC from your own accounts, as having them restored is not a simple process.

1.1.5 In the navigation pane, click **Subnets**.

1.1.6 Here you can see the subnet created by this lab's CloudFormation template (PublicSubnet1).

In the **VPC** column, you can see that this subnet exists inside of **LabVPC**.

In the **CIDR** column, you can see a value of **10.200.0.0/24**, which means this subnet includes the 256 IPs (5 of which are reserved and unusable) between 10.200.0.0 and 10.200.0.255.

In the **Availability Zone** column, you can see the AZ this subnet was placed inside of. It will vary based on the region in which your instructor launched your lab environments.

1.1.7 Click the check box next to **PublicSubnet1**'s name to reveal more details about it at the bottom of the page.

1.1.8 Click the **Route Table** tab.

1.1.9 Here you can see details about this subnet's routing.

The first entry specifies that traffic destined within the VPC's CIDR (**10.200.0.0/20**) 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 created for your lab. This is what makes this a public subnet rather than a private one.

1.1.10 Click the **Network ACL** tab.

1.1.11 Here you can see the network ACL that has been associated with this subnet.

Even though by default the inbound and outbound rules for your network ACL are open to all traffic, you will be using security groups to control access to your instances.

1.1.12 Click the **Tags** tab. Here you can see the single tag for this subnet that specifies its name as **PublicSubnet1**.

1.1.13 In the navigation pane, click **Internet Gateways**. Here you can inspect the details for the Internet gateway that was created for this lab environment.

1.1.14 In the navigation pane, click **Security Groups**. There are three security groups, two of which were created by default for you by AWS, and one (with the name beginning with **qlstack**) created by CloudFormation.

1.1.15 Click the security group with the group name that begins with *qlstack*. This is the security group that **Web Server 1** belongs to.

1.1.16 Click the **Inbound Rules** tab. Here you can see that this security group only allows traffic via SSH (TCP port 22) and HTTP (TCP port 80).

1.1.17 Click the **Outbound Rules** tab. Here you can see that this security group allows all outbound traffic via TCP and UDP.

Task 1.2: Inspect Your Amazon EC2 Instance

In this part of the lab, you will inspect the Amazon EC2 instance that was already launched for you via CloudFormation.

1.2.1 On the **Services** menu, click **EC2**.

1.2.2 In the **EC2 Dashboard**, you can see you already have 1 running instance, 1 storage volume, 1 key pair, and 3 security groups available (these are the same security groups you inspected in the last subtask).

1.2.3 In the navigation pane, click **Instances**.

1.2.4 Here you can see **Web Server 1** is already running. If it is not already selected, click the box next to its name to select it.

1.2.5 In the **Description** tab, you can inspect the details of this instance, including its public and private IPs and which AZ, VPC, subnet, and security group(s) it has been attached to.

1.2.6 Copy and paste the **Public IP** of this instance into a text editor, such as Notepad.

1.2.7 In the **Actions** menu, click **Instance Settings** > **View/Change User Data**.

Note that no user data appears. This means that while your instance was launched for you, it has not yet been configured to run your server's application. When launching an Amazon EC2 instance, you can specify user data which would do this for you (as was done in the last lab), but for the purposes of this lab we will walk you through this process in the next task.

Task 2: Start your web server's PHP app

Overview

Even though your Amazon EC2 instance has already been launched for you, it is not yet running your web application. To start the web application, you will need to SSH into the instance and run a few basic commands.

Command Reference File

On your QwikLAB page, you can find a tab which leads to a link to a text file containing any long commands or scripts required in this lab. When instructed to input long commands or scripts, we strongly encourage you to copy and paste them from the provided Command Reference File rather than typing them in manually. Copying from our lab guides has been disabled as doing so frequently leads to errors.

Scenario

While in a production environment, you would want to have the application automatically configured upon launch of a new instance, in a testing or development environment there may be circumstances where a more manual or dynamic control over what the instance launches and how it is configured is necessary.

Task 2.1: Connect to Web Server 1 (Windows Only)

Note This section is for Windows users only. If you are running OSX or Linux, skip to Task 2.1.

In this section of the lab, you will download your keypair and use it to connect to your Amazon EC2 instance with PuTTY.

2.1.1 From the qwikLABS page in your browser, in the **CONNECTION** section, click **Download PEM/PPK** > **Download PPK**.

2.1.2 Save the file to your \Downloads folder or any other easy to access location on your local computer.

2.1.3 Launch **PuTTY** by running the putty.exe file you downloaded in the last lab.

2.1.4 For **Host Name**, enter the public IP address from your **Web Server 1** instance which you copied into a text editor earlier in the lab.

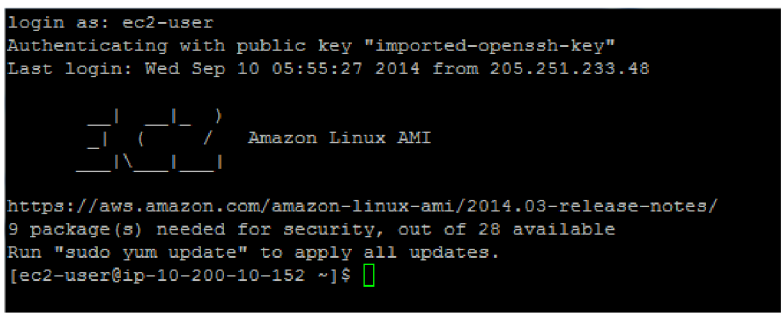
2.1.5 In the **Connection** list, expand **SSH**.

2.1.6 Click **Auth**.

2.1.7 In the **Private key file for authentication** box, browse to the .ppk file that you downloaded earlier, then click Open.

2.1.8 In the **PuTTY Security Alert** dialog box that opens, click **Yes** to add the key to PuTTY's cache.

For **login as:** type **ec2-user** and press **Enter**. You are now logged in to your **Web Server** instance.



Note The actual text you see may differ slightly from the above

2.1.9 Skip the next subtask and proceed to **Task 2.3**.

Task 2.2: Connect to Web Server 1 (Mac OSX/Linux only)

Note This section is for **Linux** and **Mac OSX** users only. If you are running **Windows** but have not yet connected to your instance, go back to **Task 2.1**. If you have already connected to your instance, skip ahead to **Task 2.3**.

In this section of the lab, you will download your keypair and use it to connect to your Amazon EC2 instance.

2.2.1 From the qwikLABS page in your browser, in the **Connection Details** section, click **Download PEM/PPK** > **Download PEM**.

2.2.2 Save the file to your local computer in a place where you can easily access it.

2.2.3 To connect to your EC2 instance, run the following commands in Terminal:

```
chmod 400 <path and name of pem>
ssh -i <path and name of pem> ec2-user@<Public IP>
```

For ***<path and name of pem>***, substitute the path/filename to the .pem file you downloaded.

For ***<Public IP>***, substitute the public IP address for your **Web Server 1** instance which you copied into a text editor earlier in the lab.

Task 2.3: Download, Install, and Launch Your Web Server's PHP Application

In this section of the lab, you will run a series of Linux commands on your instance which will download, install, and launch your server's PHP application. We will step you through each command one at a time so you can understand exactly what you are doing to accomplish this task.

2.3.1 To update your instance, execute the following command. We highly recommend that you copy all of the following commands from the Command Reference File found on your qwikLAB page.

```
sudo yum -y update
```

This will run through a check of what updates are available for your instance, download the updates, and install them.

2.3.2 To install a package that creates a web server, execute the following command:

```
sudo yum -y install httpd php
```

This command installs an Apache web server and the PHP interpreter.

2.3.3 Execute the following command:

```
sudo chkconfig httpd on
```

This configures the Apache web server to automatically start when the instance starts.

2.3.4 Execute the following command

```
wget https://d2lrzjb0vjvpn5.cloudfront.net/AWS-100-ARC/v5.0/lab-2-ha/scripts/phpapp.zip
```

This downloads the sample PHP application into the current directory.

2.3.5 Execute the following command:

```
sudo unzip phpapp.zip -d /var/www/html/
```

This extracts the PHP application into the default Apache web server directory.

2.3.6 Execute the following command:

```
sudo service httpd start
```

This starts the Apache web server.

2.3.7 Open a new web browser or tab, paste the **Public IP** address for your instance in the address bar and hit **Enter**.

The sample PHP application is run and the information specific to your Amazon EC2 instance is displayed.

2.3.8 Close the web browser or window that you opened in the previous step.

2.3.9 Return to your SSH session, type

```
exit
```

and then press **Enter**. This ends your SSH session.

Task 3: Launch a Duplicate of Web Server 1 into a Second Availability Zone

Overview

To increase the availability of your application environment, in this task you will create a second web server in different Availability Zone.

To accomplish this, you will do the following:

1. Create a snapshot of the storage volume for Web Server 1.
2. Create an image (Amazon Machine Image, or AMI) based off of that snapshot.
3. Create a second public subnet (PublicSubnet2) in a second AZ.
4. Launch a new instance (Web Server 2) into PublicSubnet2 using the AMI you created in step 2.

Scenario

Using your own custom AMIs to quickly provision new instances that are already configured for use is a key component to ensuring a consistent experience for users across your environment. It also allows you to easily replace unpatched instances with patched instances, without having to take your server down, patch it, and re-launch it, effectively eliminating downtime for patching and other maintenance.

Task 3.1: Create an Amazon Machine Image of Web Server 1

In this subtask, you will create an Amazon Machine Image (AMI) of Web Server 1, so that it can be used to launch future instances that already have your web application running.

Typically, via the console, you can create an AMI by simple stopping the instance and creating the image from it. But in this circumstance, if your instance is using ephemeral storage and you stop it, you will lose the work you did in Task 2 (installing and running your web server and simple PHP application).

But by instead snapshotting the storage volume of Web Server 1 and creating the AMI off of that, you can launch a new web server without stopping Web Server 1 and without having to run any commands you already ran in Task 2.

Note: When using Windows-based AMIs, snapshotting is a different process from Linux-based AMIs.

3.1.1 On the **Services** menu, click **EC2**.

3.1.2 In the navigation pane, click **Volumes**.

3.1.3 Ensure the storage volume attached to **Web Server 1** is already selected, and click **Actions** > **Create Snapshot**.

3.1.4 In the **Create Snapshot** window that appears, specify the following settings:

- **Name**: WebServer1Snapshot
- **Description**: Snapshot of web server 1 with Simple PHP application running

3.1.5 In the **Create Snapshot** window, click **Create**.

3.1.6 Click the link for your new snapshot next to **View snapshot**.

3.1.7 Within a minute, you should see your snapshot appear in the list. If it doesn't appear right away, periodically refresh the list with the refresh button in the upper right corner of the page until it appears

3.1.8 Periodically refresh the list with the refresh button in the upper right corner of the page until **Status** value for the snapshot is *completed*. It may take a few minutes

3.1.9 With the WebServer1Snapshot selected, click **Actions** > **Create Image**.

3.1.10 In the **Create Image from EBS Snapshot** window that appears, specify the following settings:

- **Name**: WebServerImage
- **Description**: AMI for web server with Simple PHP application running
- **Virtualization type**: Hardware-assisted virtualization

Note If when performing this operation outside of this operation, you're unsure which to use, we recommend using Hardware-assisted virtualization (HVM) rather than paravirtual (PV) machines as HVM provide the best performance in the widest range of scenarios.

3.1.11 In the **Create Image from EBS Snapshot** dialog box, click **Create**.

You've now started the process to create the AMI you will use to launch your second web server. While you wait for that process to complete, in the next task you will create the new public subnet where the new instance will be launched.

Task 3.2: Create PublicSubnet2

In this subtask, you will create PublicSubnet2, which will exist within LabVPC, but in a different Availability Zone from PublicSubnet1, to allow for cross-zone availability. You will then make the subnet publicly available by attaching it to a route table.

3.2.1 On the **Services** menu, click **VPC**.

3.2.2 In the navigation pane, click **Subnets**.

3.2.3 In the row for **PublicSubnet1**, note the value for **Availability Zone**.

3.2.4 Click **Create Subnet**.

3.2.5 Specify the following details:

- Name tag: **PublicSubnet2**
- VPC: **LabVPC**
- Availability Zone: Choose a different Availability Zone from the one where PublicSubnet1 resides.
- CIDR block: **10.200.1.0/24**

This will create a second subnet in a different Availability Zone, but still within **LabVPC**, with an IP range between 10.200.1.0 and 10.200.1.255.

3.2.6 Click **Yes, Create**. In a few seconds, **PublicSubnet2** will appear in your subnet list.

3.2.7 With **PublicSubnet2** selected, click **Subnet Actions** > **Modify Auto-Assign Public IP**.

3.2.8 In the **Modify Auto-Assign Public IP** dialog box that appears, select the check box for **Enable auto-assign Public IP**.

3.2.9 Click **Save**. For instances launched into this subnet, they will now be assigned a public IP by default, unless specified otherwise during launch.

3.2.10 With **PublicSubnet2** still selected, click the **Route Table** tab.

3.2.11 Here you can see that your new subnet has been provided with a default route table, but this route table does not have a connection to your Internet gateway.

3.2.12 Click **Edit**.

3.2.13 In the **Change to:** drop-down list, click the entry that includes **Public** in the listing.

3.2.14 Click **Save**.

PublicSubnet2 is now publicly available.

Task 3.3: Launch Your Second Web Server

In this subtask, you will use your previously created AMI to launch a new instance into PublicSubnet2.

3.3.1 On the **Services** menu, click **EC2**.

3.3.2 In the navigation pane, click **AMIs**.

3.3.3 With **WebServerImage** already selected, click **Launch**.

3.3.4 If it is not already selected, select **t2.micro** as your instance type.

3.3.5 Click **Next: Configure Instance Details**.

3.3.6 Specify the following settings if they are not already specified by default:

- Network: **LabVPC**
- Subnet: **PublicSubnet2**
- Auto-assign Public IP: **Use subnet setting (Enable)**

3.3.7 Leave the remaining settings as their default and click **Next: Add Storage**.

3.3.8 Leave the storage settings as their default and click **Next: Tag Instance**.

3.3.9 In the **Value** column for **Name**, type **Web Server 2**.

3.3.10 Click **Next: Configure Security Group**.

3.3.11 Select **Select an existing security group**.

3.3.12 Select the check box next to the security group whose name starts with *qlstack-labinstance*.

Note This is the same security group that **Web Server 1** resides in.

3.3.13 Click **Review and Launch**.

3.3.14 Review that your settings are as they should be according to the steps above and click **Launch**.

3.3.15 When prompted, ensure that the default **qwikLABS** key provided for this lab environment is selected for **Select a key pair**, and select the acknowledgement check box.

Note You need to use a *different* key pair from the one you downloaded in the first lab.

3.3.16 Click **Launch Instances**.

3.3.17 Click **View Instances**.

3.3.18 Wait until the instance state of **Web Server 2** changes to *running*. This may take a few minutes. You can click the refresh button in the upper right corner to refresh the status of your instances. Once the instance state changes to *running*, proceed to the next task**.**

Task 4: Launch and Attach a Load Balancer

Overview

While you now have two separate web server instances, your environment is not truly highly available as those instances are only accessible via two separate public IPs. In addition to being a confusing experience for users, traffic will not be balanced across the instances automatically, which could result in one instance being overutilized while the other is empty.

Task 4.1: Create and attach a Load Balancer with Amazon ELB

In this subtask, you will use Amazon ELB to create a load balancer and attach it to your two web servers. Then you will test the load balancer to show that it is distributing load between them.

4.1.1 If you are not already in the **EC2 Dashboard**, on the **Services** menu, click **EC2**.

4.1.2 In the navigation pane, click **Load Balancers**.

4.1.3 Click **Create Load Balancer**.

4.1.4 Specify the following settings:

- Load Balancer name: **WebServerLB**
- Create LB Inside: **LabVPC**

4.1.5 When you select **LabVPC**, you should see the **Select Subnets** menu appear at the bottom of the page.

4.1.6 In the **Select Subnets** menu, click the plus symbols in the rows for both subnets (**PublicSubnet1** and **PublicSubnet2**).

4.1.7 With both subnets now listed under **Selected Subnets**, click **Next: Assign Security Groups**.

4.1.8 Select **Create a new security group**.

4.1.9 Specify the following settings:

- Security group name: **WebServerLB**
- Description: **Security group for the load balancer for the web servers**
- Type: **HTTP**
- Source: **Anywhere**

4.1.10 Click **Next: Configure Security Settings**.

4.1.11 Since traffic to this public-facing load balancer does not need to be secure, the load balancer does not need a secure listener. Click **Next: Configure Health Check**.

4.1.12 Specify the following settings:

- Ping Path: **/index.php**

This tells the load balancer to ping index.php when it does a health check on an instance.

- Response Timeout: **5 seconds**

This tells the load balancer to wait 5 seconds for a response to its health check before timing out.

- Health Check Interval: **10 seconds**

This tells the load balancer to wait 10 seconds between health checks.

- Unhealthy Threshhold: **2**

This tells the load balancer that 2 consecutive failed health checks are required to identify the instance as unhealthy and stop sending traffic to it until it is healthy again.

- Healthy Threshhold: **5**

This tells the load balancer that 5 consecutive successful health checks are required to identify the instance as healthy and start sending traffic to it.

4.1.13 Click **Next: Add EC2 instances**.

4.1.14 Select the check boxes next to both of your web server instances (**Web Server 1** and **Web Server 2**).

4.1.15 Leave the remaining settings as their default and click **Next: Add Tags**.

4.1.16 Create a tag with a following settings:

- Key: **Name**
- Value: **WebServerLB**

4.1.17 Click **Review and Create**.

4.1.18 Confirm that your load balancer's settings are as they should be based on the above steps and click **Create**.

4.1.19 Click **Close**.

4.1.20 Your new load balancer has already begun performing health checks on the web servers it is attached to. After a minute has passed, start periodically refreshing the page with the refresh button in the upper right corner of the page until its **Status** is *2 of 2 instances in service*. It should take no more than 2 minutes total.

4.1.21 Copy the value next to **DNS Name**, paste it into a new browser tab or window, and hit **Enter** to navigate to your load balancer's public DNS address.

4.1.22 The landing page for your PHP app should appear. Note the IP address that appears on the page.

4.1.23 Refresh the page. The IP address should change. If it doesn't change, hit refresh again until it does. This shows that your load balancer is successfully balancing traffic between your two instances in your two different subnets.

You have now completed the process of launching your web server into a second AZ and putting your servers behind a load balancer in order to make your application more highly available. In the next Task, you will create a private application environment that sits behind an AWS NAT Gateway.

Task 5: Create a Private Application Environment

Overview

So far, your environments have only used public subnets, however most application environments use private subnets as well, where instances are cut off from direct Internet access. In this task, you will create a simple private application environment that routes traffic from two private instances out through an AWS NAT gateway.

Scenario

Most application environments do not need all of their resources directly accessible via the Internet. Applications are instead configured to control access to certain resources by keeping them in a private layer. This is typical for applications that have sensitive data that should only be accessible via approved applications in approved ways.

Task 5.1: Create Two Private Subnets

In this subtask, you will create two private subnets within LabVPC.

5.1.1 On the **Services** menu, click **VPC**.

5.1.2 In the navigation pane, click **Subnets**.

5.1.3 Click **Create Subnet**.

5.1.4 Specify the following settings:

- Name tag: **PrivateSubnet1**
- VPC: **LabVPC**
- Availability Zone: Select any AZ, but remember which AZ you picked.
- CIDR block: **10.200.2.0/23**

5.1.5 Click **Yes, Create**.

Your first private subnet has now been created.

5.1.6 Click **Create Subnet**.

5.1.7 Specify the following settings:

- Name tag: **PrivateSubnet2**
- VPC: **LabVPC**
- Availability Zone: Select any AZ other than the one you picked for Private Subnet 1
- CIDR block: **10.200.4.0/23**

5.1.8 Click **Yes, Create**.

You now should have two subnets (without routes to an Internet gateway), each in a different Availability Zone.

Task 5.2: Create and attach an AWS NAT Gateway

In this subtask, you will create an AWS NAT gateway and route your subnets through it.

5.2.1 In the navigation pane, click **NAT Gateways**.

5.2.2 Click **Create NAT Gateway**.

5.2.3 Click inside of the text box for **Subnet***. A list of your subnets will appear.

5.2.4 Click on the entry for **PublicSubnet1**.

Note While you are required to place an AWS NAT gateway inside of one subnet, it is accessible by any instances within your VPC, regardless of which subnet they reside in. AWS NAT gateways are also fully managed, which means they are inherently highly available, and do not need to be placed in more than one Availability Zone.

5.2.5 An AWS NAT gateway requires an Elastic IP so that in the event that the underlying hardware fails, the replacement hardware can attach to the same IP without an interruption in service.

Click **Create New EIP**.

5.2.6 Click **Create a NAT Gateway**.

5.2.7 You must now route your private subnets to the NAT gateway for it to work. Click **Edit Route Tables**.

5.2.8 Click **Create Route Table**.

5.2.9 Specify the following settings:

- Name tag: **Private**
- VPC: **LabVPC**

5.2.10 Click **Yes, Create**.

5.2.11 With your **Private** route table selected, click the **Routes** tab.

5.2.12 This table already includes a local route, but it needs a route to the NAT gateway for traffic headed to the Internet.

Click **Edit**.

5.2.13 Click **Add another route**.

5.2.14 Under **Destination**, type **0.0.0.0/0**

5.2.15 Click the text box under **Target**. A list appears. In that list, click the entry that starts with *nat-*.

5.2.16 Click **Save**.

5.2.17 Click the **Subnet Associations** tab.

5.2.18 Click **Edit**.

5.2.19 Select the check boxes for **PrivateSubnet1** and **PrivateSubnet2**.

5.2.20 Click **Save**.

Your private subnets will now route traffic bound for the Internet through your AWS NAT gateway.

Task 5.3: Launch Two Instances into Your Private Subnets

In this subtask, you will launch two Amazon EC2 instances into your private subnets so you can test your AWS NAT gateway.

5.3.1 On the **Services** menu, click **EC2**

5.3.2 Click **Launch Instance**.

5.3.3 Click **Select** next to the entry for **Amazon Linux AMI** at the top of the list.

5.3.4 Click **Next:Configure Instance Details** to use a t2.micro instance type.

5.3.5 Specify the following settings:

- Network: **LabVPC**
- Subnet: **PrivateSubnet1**
- Auto-assign Public IP: **Use subnet setting (Disable)**

5.3.6 Leave the remaining settings as their default and click **Next: Add Storage**.

5.3.7 Leave these settings as their default and click **Next: Tag Instance**.

5.3.8 For **Value**, type **Private Instance 1**.

5.3.9 Click **Next: Configure Security Group**.

5.3.10 Select **Create a new security group**.

5.3.11 Specify the following settings:

- Security group name: **PrivateSG**
- Description: **Security group for private instances**

5.3.12 For **Source**, click **Custom IP**.

5.3.13 In the text box next to **Custom IP** for your SSH rule, type **sg**. A list of your security groups will appear.

5.3.14 Click the security group that includes **WebServerSG** in the name.

Note This means that in order to access instances in your **PrivateSG** security group, you will first need to SSH into one of the instances in your **WebServerSG** security group and then SSH from there into a private instance.

5.3.15 Click **Review and Launch**.

5.3.16 Ensure your instance's settings are correct and click **Launch**.

5.3.17 Ensure that your qwikLABS key pair is selected, select the acknowledgement check box, and click **Launch Instances**.

5.3.18 Click **View Instances**.

5.3.19 Follow steps **5.3.2-5.3.16** to create a second private instance, this time named **Private Instance 2**, and located in PrivateSubnet2. (You do not need to create a new security group, you can just use the same security group you created for Private Instance 1)

5.3.20 You should now have two private instances, each one in a different private subnet.

5.3.21 Click the check box next to **Private Instance 1**.

5.3.22 In the **Description** tab, find the entry for **Private IPs**, copy it, and paste it into the same text file on a new line.

5.3.23 Once **Private Instance 2** is in the *running* state, click the check box next to its name and copy its private IP into the same text file for easy retrieval.

Task 5.4: SSH into Web Server 1 again (Windows Only)

Note This section is for Windows users only. If you are running Mac OSX or Linux skip ahead to Task 5.5.

In this section of the lab, connect to your Amazon EC2 instance via PuTTY again.

5.4.1 If you are not already there, navigate to your list of Amazon EC2 instances in the **EC2 Dashboard**.

5.4.2 Click the check box next to **Web Server 1**.

5.4.3 In the **Description** tab, find the entry for **Public IP**, copy it, and paste it into a text editor, such as Notepad.

5.4.4 Download **Pageant** from:

http://the.earth.li/~sgtatham/putty/latest/x86/pageant.exe

5.4.5 Launch **Pageant**.

If Pageant doesn't appear, find the icon for it in your task bar (a computer terminal with a hat-like object on top) and double-click it to open it.

5.4.6 Click **Add Key**.

5.4.7 Select the .ppk file you downloaded earlier in the lab, click **OK**, and close the Pageant window.

Since you will need to SSH into your private instances from your public instances, This will allow you to forward the key pair necessary for authenticating with your private instances.

5.4.8 Launch **PuTTY** by running the putty.exe file you downloaded previously.

5.4.9 In the Host Name box, enter the public IP address from your **Web Server 1** instance which you copied into a text editor earlier in the lab.

5.4.10 In the **Connection** list, expand **SSH**.

5.4.11 Click **Auth**.

5.4.12 Select **Allow agent forwarding**.

5.4.13 In the **Private key file for authentication box**, browse to the .ppk file that you downloaded earlier, then click **Open**.

5.4.14 For **Log in as**: type **ec2-user** and press Enter.

5.4.15 Skip the next subtask and proceed to **Task 5.6**.

Task 5.5: Connect to Web Server 1 again (Mac OSX/Linux only)

Note This section is for **Linux** and **Mac OSX** users only. If you are running **Windows** but have not yet connected to your instance, go back to **Task 5.4**. If you have already connected to your instance, skip ahead to **Task 5.6**.

In this section of the lab, you will connect to your Amazon EC2 instance.

5.5.1 To connect to **Web Server 1** with SSH agent forwarding enabled, run the following commands in Terminal:

```
chmod 600 <path and name of pem>
```

MacOS only:

```
ssh-add -K <path and name of pem>
```

Linux only:

```
ssh-add -c <path and name of pem>
```

All:

```
ssh -A -i <path and name of pem> ec2-user@<Public IP>
```

For ***<path and name of pem>***, substitute the path/filename to the .pem file you downloaded in step **5.1.1**.

For ***<Public IP>***, substitute the public IP address you copied earlier.

Task 5.6: Test Your NAT Gateway

In this subtask, you will SSH from your Web Service 1 instance into your private instances, where you will test your NAT gateway by pinging a website.

5.6.1 Return to your **EC2 Dashboard**.

5.6.2 Click the check box next to **Private Instance 1** to select it.

5.6.3 In the **Description** tab, find the entry for **Private IPs**, copy it, and paste it into a text file.

5.6.4 Return to your open SSH connection and run the following command:

```
ssh ec2-user@<Private IP>
```

For ***<Private IP>***, substitute the private IP address you copied in the previous step.

5.6.5 When prompted for a response, type **yes** and press **Enter**.

You should be logged in to your private instance now.

5.6.6 To verify that your private instance can connect to the Internet, run the following command:

```
ping ietf.org
```

5.6.7 You should receive a series of continuous responses that look similar to this:

```
PING ietf.org (4.31.198.44) 56(84) bytes of data.
64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=1 ttl=48
time=74.9 ms
```

If you don't receive the expected response, check with your instructor to determine the problem.

Note This test will only work on websites which have ICMP enabled, so websites besides ietf.org may not work.

5.6.8 Once you've received the expected response, press **Ctrl+C** to end the ping command.

5.6.9 To end your SSH session with **Private Instance 1**, type `exit` and press **Enter**.

5.6.10 Return to your **EC2 Dashboard**.

5.6.11 Click the check box next to **Private Instance 2** to select it.

5.6.12 In the **Description** tab, find the entry for **Private IPs**, copy it, and paste it into a text file.

5.6.13 Follow steps **5.6.4-5.6.9** to test the Internet connection for **Private Instance 2**.

End of Lab

Please return to the qwikLABS page for this lab and click **End** to clean up your lab environment.

Congratulations! You have completed Lab 2.

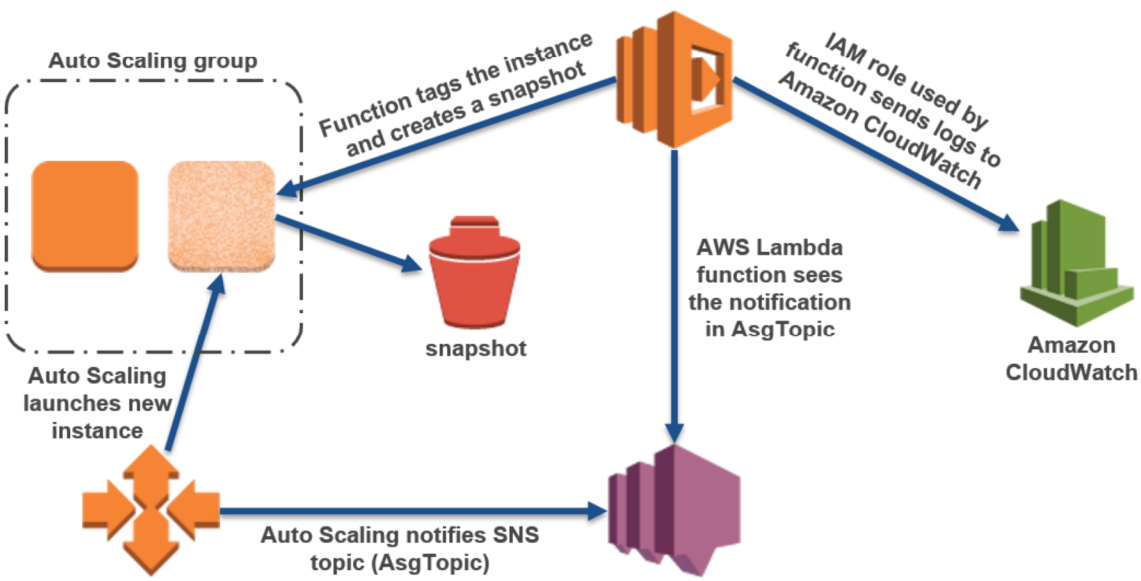
Lab 3
Using Auto-Scaling with AWS Lambda

Overview

In this lab, you will implement Auto Scaling lifecycle hooks to perform a custom action. The custom action, defined as an AWS Lambda function, will tag new Amazon EC2 instances that are launched by Auto Scaling and create .

In this lab, you will:

- Create an Amazon Simple Notification Service (Amazon SNS) topic as a notification target for lifecycle events.
- Set up your Auto Scaling group to send notifications when new EC2 instances are launched.
- Set up the roles and permissions required.
- Create a Lambda function that should be invoked when it receives a message from your Amazon SNS topic that an Auto Scaling event has occurred.
    - The Lambda function will tag the new instance and create a snapshot using API calls.
- Your environment will function this like this when you're done:



Objectives

After completing this lab, you will be able to:

- Manually scale an Auto Scaling group.
- Implement an Auto Scaling lifecycle hook that invokes a Lambda function.

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
  - The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: Administrator access to the computer
- An Internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)

Duration

This lab will require around **60 minutes** to complete.

Task 1: Creating a Notification for an Auto Scaling Event

Overview

In this task, you will configure the lab's Auto Scaling group to send lifecycle event notifications to an SNS topic.

Task 1.1: Create an SNS Topic

In this section, you will create an SNS topic that the Auto Scaling group will use as a notification target. The Auto Scaling group will send notifications to this topic when new EC2 instances are launched.

1.1.1 A pre-configured AWS account has been created for you to complete this lab. Use the instructions in Appendix B of this lab manual to sign in to the AWS Management Console using this student account.

1.1.2 In the **AWS Management Console**, in the **Services** drop-down list, click **SNS**.

1.1.3 Click **Get Started**.

Note If you don't see a **Get Started** button, you can just skip to the next step.

1.1.4 Click **Create Topic**.

1.1.5 In the Create new topic dialog box, enter the following values:

- **Topic name**: **AsgTopic**
- **Display name**: **AsgTopic**

1.1.6 Click **Create topic**.

Task 1.2: Create Notification for Launch Event

In this section, you will configure the Auto Scaling group to send notifications to the SNS topic when new EC2 instances are launched.

1.2.1 In the **AWS Management Console**, on the **Services** menu, click **EC2**.

1.2.2 In the navigation pane, click **Auto Scaling Groups**.

1.2.3 Click **Auto Scaling Group: 1**.

You should now see the Auto Scaling Group that was created for you automatically for this lab.

1.2.4 On the **Notifications** tab, click **Create notification**.

1.2.5 For **Send a notification to**, make sure that **AsgTopic** is selected.

1.2.6 For **Whenever instances**, ensure that only **launch** is selected. All other options should not be selected.

1.2.7 Click **Save**.

Task 2: Handling a Lifecycle Event Notification

Overview

In this task, you will develop a Lambda function that will be invoked by Amazon SNS when a lifecycle event notification is received. The Lambda function will extract the EC2 instance ID from the event notification and add a tag to the EC2 instance.

Command Reference File

At various points, this lab instructs you to enter commands or code into your lab environment. Copy the text of these commands from the lab's associated command reference file, which is available on the Instructions tab of your lab in qwikLABS.

It is not recommended to copy and paste commands from this lab manual, because the manual's rich formatting may inject characters that cannot properly be parsed.

Task 2.1: Create an IAM Role to Grant the Lambda Function Permission to Access AWS Services

In this section, you will create an IAM role that grants permissions to perform operations on EC2 instances, notify Auto Scaling that the custom action associate with the lifecycle hook is complete, and log messages by using Amazon CloudWatch.

You will associate this role with your Lambda function when you create your Lambda function later.

2.1.1 On the **Services** menu, click **IAM**.

2.1.2 In the navigation pane, click **Roles**.

2.1.3 Click **Create New Role**.

2.1.4 For **Role Name**, type **EC2TagSnapLambdaRole,** and then click **Next Step**.

2.1.5 On the **Select Role Type** page, under **AWS Service Roles**, in the row for **AWS Lambda,** click **Select**.

2.1.6 On the **Attach Policy** page, for **Filter** type, **AmazonEC2FullAccess**

2.1.7 Select **AmazonEC2FullAccess**.

2.1.8 Click **Next Step**

2.1.9 Review the role information, and then click **Create Role**.

2.1.10 Click **EC2TagSnapLambdaRole**.

2.1.11 On the **Permissions** tab, expand **Inline Policies** by clicking on it, and click **click here**.

2.1.12 Select **Custom Policy**, and then click **Select**.

2.1.13 For **Policy Name**, type **EC2TagSnapLambdaInlinePolicy**

2.1.14 For **Policy Document**, copy and paste the inline policy from the command reference file. This policy allows the Lambda function to send log messages to Amazon CloudWatch.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
          "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
```



```
            ],
            "Resource": "arn:aws:logs:*:*:*",
            "Effect": "Allow"
        }
    ]
}
```

2.1.15 Click **Apply Policy**.

Task 2.2: Create a Lambda Function

In this section, you will create a Lambda function that will be invoked by Amazon SNS when a lifecycle event notification is received. The Lambda function will extract the EC2 instance ID from the event notification and add a tag to the EC2 instance.

The Lambda function will be developed with Python 2.7 and the Boto 3 SDK.

2.2.1 On the **Services** menu, click **Lambda**.

2.2.2 Click **Get Started Now**.

2.2.3 On the **Step 1: Select blueprint** page, click **Skip**.

2.2.4 For **Name**, type **EC2TagSnapLambdaFunction**

2.2.5 For **Description**, type **Tags EC2 instances with the ManualScaling tag**

2.2.6 For **Runtime**, click **Python 2.7**.

2.2.7 For **Lambda function code**, make sure that **Edit code inline** is selected. Copy and paste the Lambda function code from the command reference file.

2.2.8 For **Role**, click **EC2TagSnapLambdaRole**. This gives AWS Lambda the execution permissions that you have specified for this role.

2.2.9 Leave other values with default settings, and then click **Next**.

2.2.10 Click **Create function**.

2.2.11 On the **Configuration** tab, click **Advanced settings**.

2.2.12 For **Timeout**, change the time to **3 minutes**.

2.2.13 On the **Event sources** tab, click **Add event source**.

2.2.14 In the **Add event source** dialog box, for **Event source type**, click **SNS**.

2.2.15 For **SNS topic**, make sure that **AsgTopic** is selected. Amazon SNS will invoke this Lambda function when AsgTopic receives an Auto Scaling lifecycle event notification.

2.2.16 Make sure that **Enable event source** is selected.

2.2.17 Click **Submit**.

Tip On the **Monitoring** tab, you can view CloudWatch metrics related to your Lambda function. Click **View logs in CloudWatch** to view the messages logged by your Lambda function.

Because the Lambda function has not been invoked yet, the metrics will currently display zero invocation count.

Task 2.3: Scale Out Auto Scaling Group to Trigger Lifecycle Event Hook

In this section, you will increase the desired capacity of the lab's Auto Scaling group from 1 to 2. The Auto Scaling group will launch a new EC2 instance to meet the increased capacity requirement. Auto Scaling will send a lifecycle event notification to AsgTopic (SNS topic). Amazon SNS will invoke EC2TaggerLambdaFunction when it receives the event notification.

2.3.1 On the **Services** menu, click **EC2**.

2.3.2 In the navigation pane, click **Auto Scaling Groups**.

2.3.3 On the **Details** tab, click **Edit**.

2.3.4 For **Desired**, type **2**

2.3.5 Click **Save**.

2.3.6 On the **Activity History** tab, monitor the progress of the new EC2 instance that is being launched. Wait for the status to change to *Successful*.

Note It may take 30 seconds for the second instance to appear in that list. Continue refreshing the page if your new instance doesn't appear.

2.3.7 In the navigation pane on the left side of the page (not the tab at the bottom of the page), click **Instances**.

2.3.8 Click the row for the instance that has the most recent launch time. You might have to scroll to the right to view the **Launch Time** column for your instance.

2.3.9 On the **Tags** tab, you will see a tag with **ManualScaling** as the key, and **Yes** as the value. This tag was added to the EC2 instance by EC2TaggerLambdaFunction.

2.3.10 In the navigation pane, click **Snapshots**.

In the snapshot window, you will see snapshots of the volumes attached to the new instance.

End of Lab

Please return to the qwikLABS page for this lab and click **End** to clean up your lab environment.

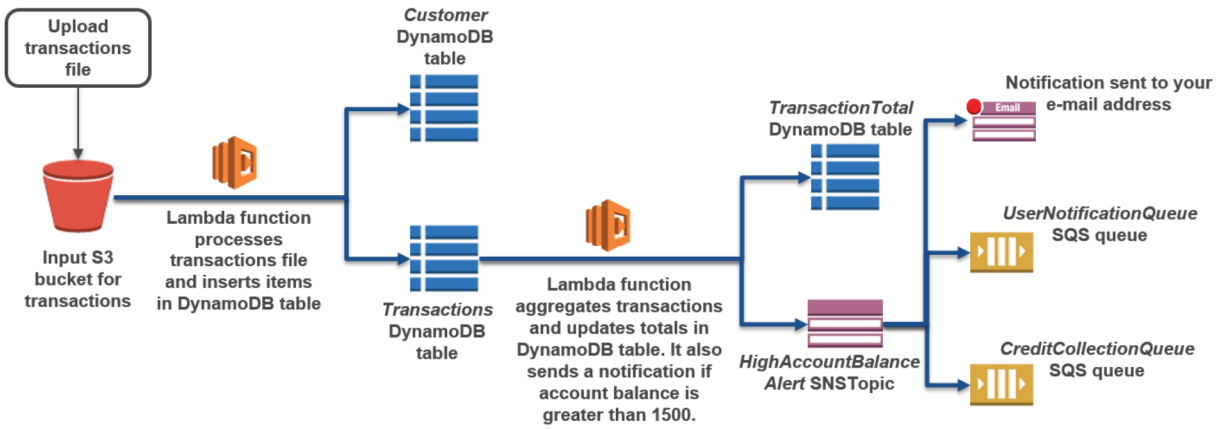
Congratulations! You have completed Lab 3.

Lab 4
Implementing a Serverless Architecture with AWS Managed Services

Overview

In this lab, you will use AWS managed services to implement a serverless architecture.

The following diagram shows the lab scenario.



The lab environment is already setup with the following resources:

- Amazon S3 bucket: You will upload a text file which contains a list of credit card transactions to this bucket.
- Amazon DynamoDB tables
  - *Customer* table: This table will store the customer id and customer address.
  - *Transactions* table: This table will store the transaction id, transaction date, and transaction amount. The streams feature is enabled for this table.
  - *TransactionTotal* table: This table will store the current account balance (aggregate transaction amount) for each customer.
- IAM roles: You will use the IAM roles to assign execution permissions to AWS Lambda.

You will configure the following resources:

- *HighAccountBalanceAlertSNSTopic*: This Amazon SNS topic will be used to send notifications if the account balance for a customer is greater than 1500. The notifications in this topic are fanned out to SQS queues.
- *UserNotificationQueue*: This Amazon SQS queue subscribes to notifications from *HighAccountBalanceAlertSNSTopic*.

- *CreditCollectionQueue*: This Amazon SQS queue subscribes to notifications from *HighAccountBalanceAlertSNSTopic*.
- Lambda functions:
  - *TransactionProcessor*: This Lambda function will process the transactions text file that is uploaded to the S3 bucket and insert items into the *Customer* and *Transactions* DynamoDB tables.
  - *TransactionAggregatorNotifier*: This Lambda function will update the account balance in the *TransactionTotal* table and send a notification when the account balance exceeds 1500.

Objectives

After completing this lab, you will be able to:

- Use AWS managed services to implement a serverless architecture.
- Set up Lambda functions to act as triggers in a DynamoDB table.

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)
  - The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: Administrator access to the computer
- An Internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)

Duration

This lab will require around **60 minutes** to complete.

Task 1: Reviewing Resources in Lab Environment

Overview

In this task, you will review the resources that have already been set up in the lab environment.

Task 1.1: Review Resources

In this section, you will explore the resources that have been set up in the lab environment.

1.1.1 A pre-configured AWS account has been created for you to complete this lab. Use the instructions in Appendix A of this lab manual to sign in to the AWS Management Console using this student account.

1.1.2 In the **AWS Management Console**, on the **Services** menu, click **CloudFormation**.

1.1.3 Select the check box next to the stack name that begins with *qlstack*.

1.1.4 On the **Outputs** tab, review the resources that have been created in the lab environment. You should see they include four entries: **InputS3BucketForTransactionsFilesName**, **CustomerDynamoDBTable**, **TransactionsDynamoDBTable**, and **TransactionTotalDynamoDBTable.**

1.1.5 (Optional) On the **Services** menu, click **S3** to view the S3 bucket created for this lab (it includes **inputs3bucketfortransact** in the name).

1.1.6 (Optional) On the **Services** menu, click **DynamoDB**. In the navigation pane, click **Tables** to view the tables created for this lab. Note that these tables currently contain no items. They will be populated automatically as you complete this lab.

Task 2: Setting Up SNS Topic and SQS Queues

Overview

In this task, you will set up an SNS topic and two SQS queues. You will subscribe the SQS queues to the SNS topic. This setup is known as a fan-out scenario because each SNS notification is distributed to multiple SQS queues.

Task 2.1: Set Up an SNS Topic

In this section, you will set up the SNS topic that will receive notifications about a customer's high account balance. You will also subscribe to the topic with an email address.

2.1.1 In the **AWS Management Console**, on the **Services** menu, click **SNS**, and then click **Get Started** on the main page.

2.1.2 In the navigation pane, click **Topics**, and then click **Create new topic**.

2.1.3 In the **Create new topic** dialog box, enter the following properties:

- **Topic name**: HighAccountBalanceAlertSNSTopic
- **Display name**: HABTopic

2.1.4 Click **Create topic**.

2.1.5 Click the link with the topic ARN for **HighAccountBalanceAlertSNSTopic**.

2.1.6 Copy and paste the **Topic ARN** into a text editor. You will use this value later in the lab.

2.1.7 Click **Create subscription**.

2.1.8 In the **Create Subscription** window that appears, click **Protocol** > **Email**.

2.1.9 For **Endpoint**, type in an email address that you can easily access. This can be either a work or personal email address. This email will receive notifications from the SNS Topic you've created.

2.1.10 Click **Create Subscription**.

2.1.11 Check the email account you just provided for a new email from **HABTopic**. It may take a minute to be delivered.

2.1.12 When you receive the email, open it and click the **Confirm subscription** link contained within.

This SNS topic will now notify your email if it receives a message, however it will also have two more subscribers: the two SQS queues you will create next.

Task 2.2: Set Up Two SQS Queues

In this subtask, you will set up two SQS queues that will subscribe to notifications from the SNS topic you just created.

2.2.1 In the **AWS Management Console**, on the **Services** menu, click **SQS**, and then click **Get Started Now** on the main page.

2.2.2 In the **Create New Queue** dialog box, for **Queue Name**, type **UserNotification**

2.2.3 Leave the remaining settings as their default and click **Create Queue**.

Note In a complete application environment, you could use a Lambda function or other application to read the messages in this queue and notify users of a high balance.

2.2.4 Click **Create New Queue**.

2.2.5 In the **Create New Queue** dialog box, for **Queue Name**, type **CreditCollection**

2.2.6 Leave the remaining settings as their default and click **Create Queue**.

Note In a complete application environment, you could use a Lambda function or other application to read the messages in this queue and notify your credit collection department to monitor this account.

2.2.7 Select the check boxes for both queues.

2.2.8 Click **Queue Actions** > **Subscribe Queues to SNS Topic**.

2.2.9 In the **Subscribe to a Topic** dialog box, for **Choose a Topic**, click **HighAccountBalanceAlertSNSTopic**, and then click **Subscribe**.

2.2.10 In the **Topic Subscription Result** dialog box, click **OK**.

Your two queues are now subscribed to your SNS topic. They will automatically receive any messages pushed to that topic.

Task 3: Configuring Lambda Functions

Overview

In this task, you will set up Lambda functions to:

- Process transactions text file
- Calculate transaction totals and notify about high account balances

Command Reference File

At various points, this lab instructs you to enter commands or code into your lab environment. Copy the text of these commands from the lab's associated command reference file, which is available on the Instructions tab of your lab in qwikLABS.

It is not recommended to copy and paste commands from this lab manual, because the manual's rich formatting may inject characters that cannot properly be parsed.

Task 3.1: Create Lambda Function to Process Transactions Text File

In this section, you will set up a Lambda function to process the transactions text file that you will upload to your S3 bucket, insert customer id and address into the Customer table, and insert transaction information into the Transactions DynamoDB table.

3.1.1 On the **Services** menu, click **Lambda**.

3.1.2 Click **Get Started Now**.

3.1.3 On the **Step 1: Select blueprint** page, click **Skip**.

3.1.4 For **Name**, type **TransactionProcessor**

3.1.5 For **Description**, type **Processes data and sends to DynamoDB tables**

3.1.6 For **Runtime**, click **Python 2.7**.

3.1.7 For **Lambda function code**, make sure that **Edit code inline** is selected. Copy and paste the first Lambda function's code from the command reference file.

3.1.8 For **Role**, click the role that contains the text ***S3LambdaDynamoDBRole***. This gives AWS Lambda the execution permissions required to access Amazon S3 and Amazon DynamoDB.

Note **Do not select the role that contains the text *SNS*.**

3.1.9 Change the **Timeout** value to **0 min 20 sec**.

3.1.10 Leave other values with default settings, and then click **Next**.

Note Since you will not be using any resources that exist inside of a VPC for this architecture, you do not need to specify a VPC for this function.

3.1.11 Click **Create Function**.

3.1.12 On the **Event sources** tab, click **Add event source**.

3.1.13 In the **Add event source** dialog box, for **Event source type**, click **S3**.

3.1.14 For **Bucket**, click the bucket that begins with *qlstack* and contains the text *inputs3bucket*.

3.1.15 For **Event Type**, click **Object Created (All)**.

3.1.16 Make sure that **Enable event source** is selected.

3.1.17 Click **Submit**.

Now whenever a file is uploaded to your input bucket in Amazon S3, this Lambda function will run and sort the data it finds into the **Customer** and **Transactions** tables in **DynamoDB**.

Task 3.2: Create Lambda Function to Calculate Transaction Totals and Notify About High Account Balances

In this section, you will set up a Lambda function to calculate transaction totals and send an SNS notification if an account balances exceeds 1500.

3.2.1 Click **Functions** near the top left corner of the page.

3.2.2 Click **Create a Lambda function**.

3.2.3 On the **Step 1: Select blueprint** page, click **Skip**.

3.2.4 For **Name**, type **TransactionAggregatorNotifier**

3.2.5 For **Description**, type **Updates transaction totals; sends notifications for balance exceeding 1500**

3.2.6 For **Runtime**, click **Python 2.7**.

3.2.7 For **Lambda function code**, make sure that **Edit code inline** is selected. Copy and paste the second Lambda function code from the command reference file.

3.2.8 On **line 9** of the function code, you should see this:

```
snsTopicArn = '<ARN for HighAccountBalanceAlertSNSTopic>'
```

3.2.9 Replace `<ARN for HighAccountBalanceAlertSNSTopic>` with the ARN that you noted earlier. **Replace the text within the single quotes. Do not delete the single quotes.**

3.2.10 For **Role**, under **Use existing role**, click the role that contains the text ***SNSLambdaDynamoDBRole***. This gives AWS Lambda the execution permissions required to access Amazon DynamoDB and Amazon SNS.

Note **Do not select the role that contains the text *S3*.**

3.2.11 Change the **Timeout** value to **0 min 20 sec**.

3.2.12 Leave other values with default settings, and then click **Next**.

3.2.13 Click **Create Function**.

3.2.14 On the **Event sources** tab, click **Add event source**.

3.2.15 In the **Add event source** dialog box, for **Event source type**, click **DynamoDB**.

3.2.16 For **DynamoDB table**, click **Transactions**.

3.2.17 For **Enable event source**, make sure that **Enable now** is selected.

3.2.18 Click **Submit**.

You have now configured the second Lambda function, which starts when the **Transactions** table is updated, and then takes the data it finds there, adds up each customer's transaction amounts, stores that value in the **TransactionTotal** table, and sends a message to your SNS topic if a customer's balance exceeds 1500.

Task 4: Testing Serverless Architecture by Uploading Transactions File

Overview

In this task, you will upload a transactions file to an S3 bucket. You will verify that:

- The *TransactionProcessor* Lambda function has processed the input transaction file and inserted items into the *Customer* and *Transactions* DynamoDB tables.
- The *TransactionAggregatorNotifier* Lambda function has updated transaction totals in the *TransactionTotal* DynamoDB table and has sent SNS notifications about high account balances.
- The *HighAccountBalanceAlertSNSTopic* SNS topic has fanned the notification out to the SQS queues that have subscribed to it.

Task 4.1: Upload Transactions File to S3 Bucket

In this section, you will retrieve the transactions file and upload it to the S3 bucket that has been created for this lab.

4.1.1 Click on the following link to download the transactions.txt file to your local machine. If your web browser displays the file instead of downloading it, right click on the web page, and then save the page as a text file.

https://d2lrzjb0vjvpn5.cloudfront.net/AWS-100-ARC/v5.0/lab-4-serverless/scripts/transactions.txt

4.1.2 On the **Services** menu, click **S3**.

4.1.3 Click the bucket name that begins with the text *qlstack* and contains the text *inputs3bucket*.

4.1.4 Click **Upload**.

4.1.5 Click **Add Files**, and then select the transactions.txt file that you downloaded earlier and click **Open**.

4.1.6 Click **Start Upload**.

Uploading this file to Amazon S3 will immediately trigger the first Lambda function you created, which will immediately start sorting the data it finds in the uploaded text file and storing customer data from it into your **Customers** DynamoDB table.

Task 4.2: Verify Implementation

In this section, you will verify that the transactions file was processed correctly.

4.2.1 On the **Services** menu, click **DynamoDB**.

4.2.2 In the navigation pane, click **Tables**.

4.2.3 Click **Customer**.

4.2.4 In the **Items** tab, verify that there are items with the customer id and address for two customers.

4.2.5 Click **Transactions**.

4.2.6 In the **Items** tab, verify that several transactions exist. You should see 24 items total in the list.

4.2.7 Click **TransactionTotal**.

4.2.8 In the **Items** tab, verify that there are items with the customer id and account balance for two customers. Note the account balance for customer *C2*.

You should have by now received another email from **HABTopic** that includes an alert about customer C2's high account balance. That same message was also sent to your two SQS queues, ready to be picked up by another process (not included in this lab).

To see the messages sent to your queues in the console, you need to start polling for them, which is what you will do next.

4.2.9 On the **Services** menu, click **SQS**.

4.2.10 Click **CreditCollection** to select it.

4.2.11 Click **Queue Actions** > **View/Delete Messages**.

4.2.12 Click **Start Polling for Messages**. Wait for the progress bar at the bottom to complete before proceeding.

4.2.13 Click **More Details** in the message displayed.

4.2.14 Verify that the *Message* attribute displays a warning for customer *C2*, and then click **Close**.

```
{

  "Type" : "Notification",

  "MessageId" : "eb0d030d-5f2d-5695-8f22-4c68d0335c0b",

  "TopicArn" : "arn:aws:sns:us-east-
1:123456789:HighAccountBalanceAlertSNSTopic",
```



```
  "Subject" : "Warning! Account balance is very high",

  "Message" : "{\"customerID\": \"C2\", \"accountBalance\":
\"1750\"}",

...

}
```

4.2.15 Click **Close** again to return to your list of queues.

4.2.16 Clear the check box for **Credit Collection** and select the check box for **UserNotification**.

4.2.17 Click **Queue Actions** > **View/Delete Messages**.

4.2.18 Click **Start Polling for Messages**. Wait for the progress bar at the bottom to complete before proceeding.

4.2.19 Click **More Details** in the message displayed.

4.2.20 Verify that the *Message* attribute displays a warning for customer *C2*, and then click **Close**.

4.2.21 Click **Close** again to return to your list of queues.

4.2.22 On the **Services** menu, click **Lambda**.

4.2.23 Click **TransactionProcessor**, and then click the **Monitoring** tab to view CloudWatch metrics for the Lambda function. You can also click **View logs in CloudWatch** to see the logs for your function.

4.2.24 Click **Functions** near the top of the page.

4.2.25 Click **TransactionAggregatorNotifier**, and then click the **Monitoring** tab to view CloudWatch metrics for the Lambda function and view logs in Amazon CloudWatch.

End of Lab

Please return to the qwikLABS page for this lab and click **End** to clean up your lab environment.

Congratulations! You have completed Lab 4.

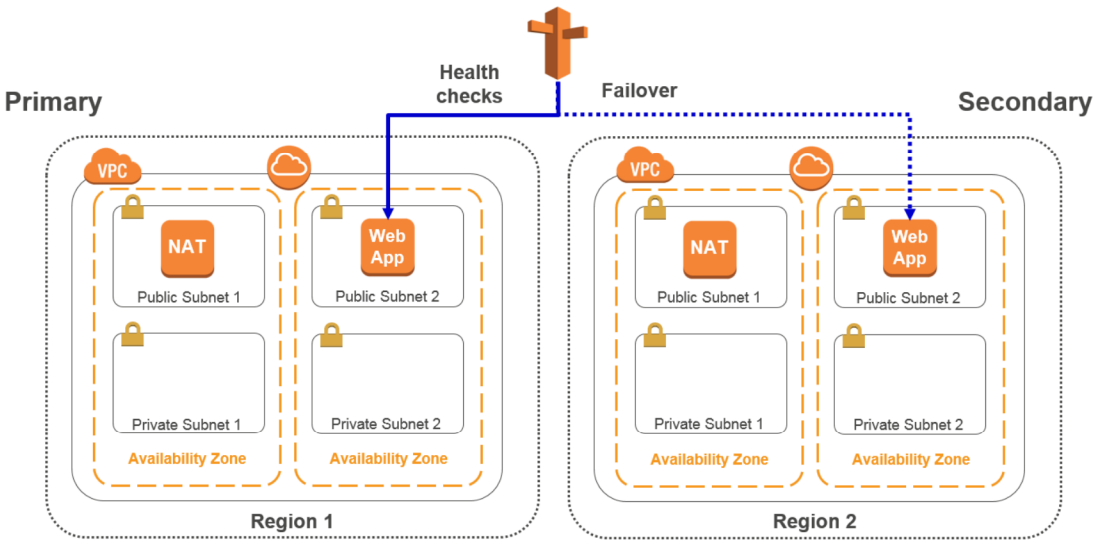
Lab 5
Multi-Region Failover with Amazon Route 53

Overview

In this lab, you will configure and test a cross-region disaster recovery scenario.

This lab comes with a basic application environment already created, duplicated across two separate regions. You will set up your domain so that if the web application's resources in its primary region become unavailable, Route 53 automatically fails the application's traffic over to its secondary region.

Here is what your environment will look like when you're done:



Objectives

After completing this lab, you will be able to:

- Use Route 53 to configure cross-region failover of a web application.
- Use Route 53 health checks to determine the health of a resource.

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat)

    - The qwikLABS lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: Administrator access to the computer
- An Internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)

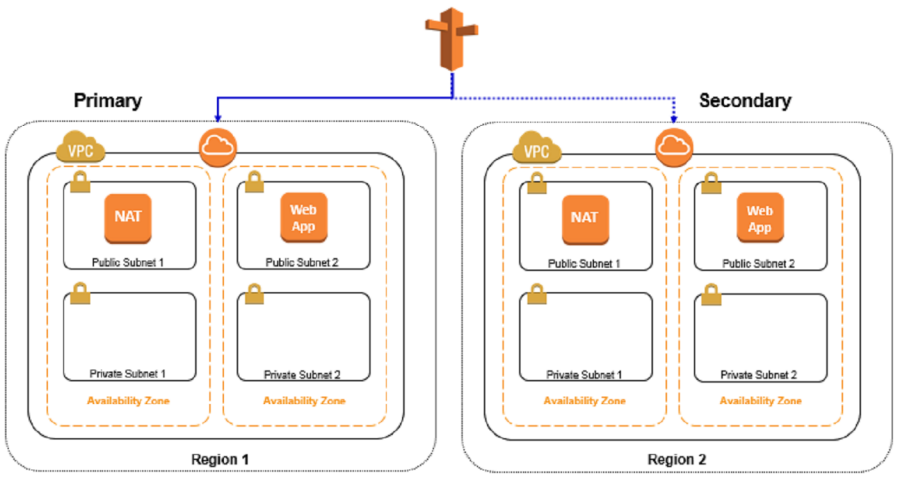
Duration

This lab will require around **30 minutes** to complete.

Task 1: Inspect Your Environment

Overview

Below is a diagram of the environment that was created for you using CloudFormation when you started this lab:



In addition, a randomly generated domain was created for you and is listed in Route 53. In this part of the lab, you will configure your domain so that if your primary resources become unavailable, Route 53 automatically fails over new requests to your secondary resources instead.

Scenario

Region-wide events, such as natural disasters, can disrupt the availability of a region for an extended length of time, making cross-region availability a critical component to ensuring that an application is as highly available as possible. This part of the lab demonstrates that, if resources in one region become inaccessible, Route 53 can help keep your web application available with a minimal amount of downtime.

Task 1.1: Examine the Primary Region

In this task, you will assess the VPC and Amazon EC2 resources in your primary region, which have been automatically created for you by CloudFormation.

1.1.1 In the **AWS Management Console**, on the **Services** menu, click **VPC**.

1.1.2 In the navigation pane, click **Your VPCs**.

LabVPC should be listed.

1.1.3 In the navigation pane, click **Subnets**.

There should be two sets of public and private subnets, with one of each in separate Availability Zones, like in Lab 2. Other subnets are listed, but these are attached to the Default VPC for your account and can be ignored.

1.1.4 On the **Services** menu, click **EC2**.

1.1.5 In the navigation pane, click **Instances**.

1.1.6 You should see a **NAT** instance and a web application instance named **Web-Application-1**.

Task 1.2: Examine the Secondary Region

In this task, you will assess the VPC and Amazon EC2 resources in your secondary region, which have been created for you with CloudFormation.

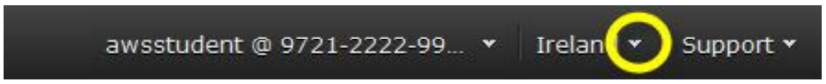
1.2.1 Return to your the qwikLABS page.

1.2.2 Click the **ADDL INFO** tab to see where your secondary region is.

Use the chart below to ensure that your environment was set up correctly:

Primary Region	Secondary Region
US East (N. Virginia), **US West (Oregon or N. California)** **South America (São Paulo)**	**EU** (Ireland)
EU (Ireland or Frankfurt)	**US East** (N. Virginia)
Asia Pacific (Singapore, Tokyo, or Sydney)	**US West** (N. California)

1.2.3 Return to the AWS Management Console, and click your current region to expand the region drop-down list, and then click your secondary region.



1.2.4 On the **Services** menu, click **VPC**.

1.2.5 In the **VPC Dashboard**, click **Your VPCs**.

LabVPC should also be listed in this region.

1.2.6 Verify that you have the same VPC configuration as the first region, including two public subnets and two private subnets, one of each type in each of the two Availability Zones.

1.2.7 On the **Services** menu, click **EC2**.

1.2.8 In the navigation pane, click **Instances**.

You should find a **NAT** instance and a web application instance named **Web-Application-2**.

Task 2: Configure Your Primary Resources to Failover to Your Secondary Resources

Overview

In this task, you will take a domain which has already been registered for you, add a health check and two record sets to it (one for each region's instance), and configure the record sets so that if an error is returned by the primary region, any new requests to the domain will fail over to the secondary region instead.

The objective of this lab is to demonstrate how to configure Route 53 to fail over from one region to another.

Note In the lab environment, you are not using a registered domain. The domain name that you see in the Route 53 Dashboard is an auto-generated, unique, un-registered, non-existing domain.

Task 2.1: Configure Route 53 Failover and Health Check

In this part of the lab, you will create a health check for your primary server and then configure your domain to failover to your secondary region in the event that the first region registers as unhealthy.

2.1.1 On the **Services** menu, click **Route 53**.

If you see an error message, you can safely ignore it because this is due to the IAM restrictions placed on these lab accounts by the CloudFormation template.

2.1.2 In the navigation pane, click **Health checks**.

2.1.3 Click **Create health check**, and then enter the following:

- **Name**: healthchk-1
- **IP address**: ***<Web Application 1's public IP>***
- **Path**: info.php

2.1.4 Expand **Advanced configuration** and enter the following:

- **Request interval**: Fast (10 seconds)
- **Failure threshold**: 2

2.1.5 Leave all other settings as the default. Click **Next**.

2.1.6 Click **Create health check**.

Route 53 will now check the health of your site by periodically requesting the IP address and path combination you provided and verifying that it returns a successful response (to be more specific, it's checking independently from multiple locations around the world, with each location requesting the page every 10 seconds).

You can view the CloudWatch metrics for healthchk-1 on the **Monitoring** tab on the **Health Checks** page.

2.1.7 In the navigation pane, click **Hosted zones**.

The domain name **qwiklabs-<xxxx-xxxxx>.training** has already been created for you to use.

In a real-world scenario, you need to create a hosted zone for your own domain name; however, this step would require you to modify the DNS settings for a real domain that you own, so this unregistered domain was generated for you.

2.1.8 Select **qwiklabs-<XXXX-XXXXX>.training**.

Note All lab participants will have a unique domain name.

2.1.9 Click **Go to Record Sets**.

2.1.10 Click **Create Record Set**.

2.1.11 In **Create Record Set**, configure the following:

- **Name**: www
- **Type**: A – IPv4 address
- **TTL (Seconds)**: Click **1m** to set the TTL to 60 seconds.
- **Value**: ***<Web Application 1's public IP>***
- **Routing Policy**: Failover
- **Failover Record Type**: Primary
- **Associate with Health Check**: Yes
- **Health Check to Associate**: healthchk-1

2.1.12 Leave all other settings as the default and click **Create**.

An A type record set should be listed. If the newly created record does not immediately appear in the table, periodically click the refresh icon to update the table until it appears.

2.1.13 Click **Create Record Set** again.

2.1.14 In **Create Record Set**, configure the following:

- **Name**: www
- **Type**: A – IPv4 address
- **TTL (Seconds)**: Click **1m** to set the TTL to 60 seconds.
- **Value**: ***<Web Application 2's public IP>***
- **Routing Policy**: Failover
- **Failover Record Type**: Secondary
- **Associate with Health Check**: No

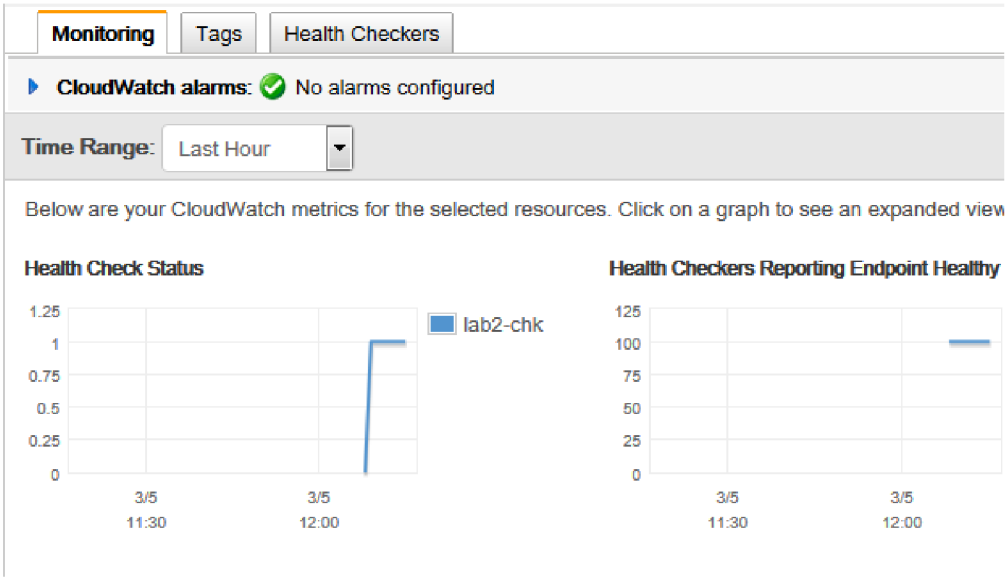
2.1.15 Leave all other settings as the default and click **Create**.

2.1.16 In the navigation pane, click **Health checks**.

The status for **healthchk-1** should be *Healthy*.

2.1.17 Select **healthchk-1**.

The **Monitoring** tab in the lower pane displays more detail about the selected health check's status.



You have now configured your web application to fail over across two regions.

Task 2.2: Check the DNS resolution

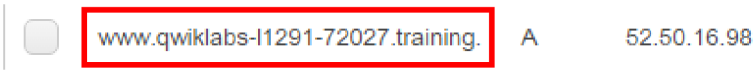
In this task, you will query DNS to obtain the IP address mapping in order to verify that Route 53 is pointing correctly to your primary web application server.

Because the domain you are using is un-registered, you will not be able to check your domain's DNS resolution directly via a web browser. Instead, you will use a website that performs an nslookup command for you.

2.2.1 In the navigation pane, click **Hosted zones**, and then select the check box for your domain name.

2.2.2 Click **Go to Record Sets**.

2.2.3 Copy your **A** record name to a text editor, *but do not copy the ending period*.



2.2.4 In a new browser window or tab, open this URL:

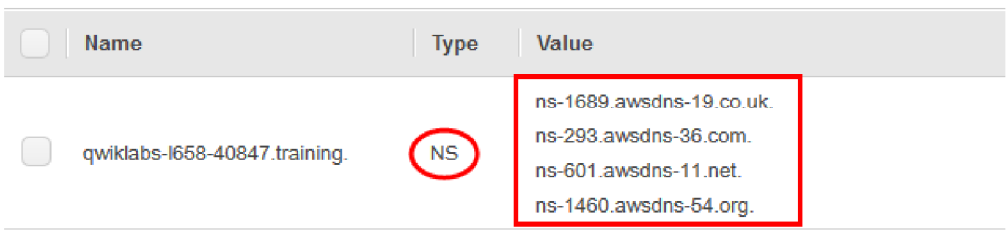
http://networking.ringofsaturn.com/Tools/nslookup.php

Note If this website does not work properly for you, there are many similar sites that can be found with a quick search of "nslookup."

2.2.5 For **Hostname/Domain Name**, paste the **A** record name you copied in step **2.2.3**.

2.2.6 Return to your previous browser window or tab, where the record sets for your hosted zone are displayed.

2.2.7 Copy one of the **NS** record type values to a text editor, *but do not copy the ending period*.



You will use one of these name server record values in the following step.

2.2.8 Return to your browser or tab with the *nslookup* tool.

2.2.9 For **Server**, paste the **NS** record name you just copied.

2.2.10 Click **Submit**.

In a few seconds, you should receive results at the bottom of the page. The address under your **A** record name should be the same IP as the server in your primary region.

This DNS resolution check confirms that requests to your domain will be routed to the server in your primary region.

2.2.11 Keep this nslookup page open, as you will use it again later to test your multi-region failover in the next task.

Task 3: Test Your Failover

Overview

In this task, you will try to verify that Route 53 correctly fails over to your secondary region if your primary region fails. For the purposes of this demonstration, you will simulate a region failure by manually stopping the instance in your primary region.

Because the domain you are using is un-registered, you will not be able to test the failover through the web browser. Instead, you will use the nslookup web application used in the last task.

Task 3.1: Stop the Primary Web Application Server

To test your application's cross-region failover ability, you will stop the primary server, **Web-Application-1**.

3.1.1 Return to the AWS Management Console. On the **Services** menu, click **EC2**.

3.1.2 Select your primary region from the region drop-down list.

3.1.3 In the navigation pane, click **Instances**.

3.1.4 Right-click **Web-Application-1**, click **Instance State**, and click **Stop**.

3.1.5 In the **Stop Instances** dialog box, click **Yes, Stop**.

Wait until the instance state is *stopped*.

3.1.6 On the Services menu, click **Route 53**.

3.1.7 In the navigation pane, click **Health checks**.

3.1.8 Select **healthchk-1**, and click the **Health checkers** tab in the lower pane. It should start reporting failed health checks.

3.1.9 Wait until the status of healthchk-1 is *Unhealthy*. If necessary, periodically click the refresh icon.

3.1.10 Return to your web browser or tab with the nslookup web application and click **Submit** again.

3.1.11 This time, the query results should return the IP address of the web application server in your **secondary** region instead.

If you don't get the correct results, re-confirm that healthchk-1 has registered as *Unhealthy* and then try again.

You've now successfully confirmed that your application environment can fail over from its primary region to its secondary region if the server in the primary region fails.

End of Lab

Please return to the qwikLABS page for this lab and click **End** to clean up your lab environment.

Congratulations! You have completed Lab 5.