



# **Launching and Managing a Web Application With AWS CloudFormation**

---

## Table of Contents

Introduction .....	3
What is AWS CloudFormation? .....	3
Start your <i>qwikLAB</i> ™ .....	4
Part 1: Create an S3 bucket using AWS CloudFormation.....	7
Download the AWS CloudFormation Template .....	7
Let's take a look at the file structure. ....	7
What is an AWS CloudFormation Template? .....	7
Resources: Hello Bucket!.....	8
Resource Properties and Using Resources Together .....	9
Optional Outputs .....	9
Select the CloudFormation Service .....	10
Confirm your AWS Region .....	10
Create a CloudFormation Stack .....	11
We will now use CloudFormation to create a stack of AWS resources.....	11
Delete the Stack.....	13
Change the Retention Policy .....	14
Create a New Stack with the Modified Retention Policy.....	15
Conclusion of Part One .....	16
Part 2: Provision a Web Application .....	17
A Simple Application .....	17
Create the Initial Stack.....	19
Change Resources Properties.....	21
Add Resources Properties .....	23
Update IAM Policies.....	26
Change the Stack's Resources.....	26
Update the stack with the full template .....	28
Availability and Impact Considerations .....	30
Conclusion .....	30
End Lab .....	31

Copyright © 2013 Amazon Web Services, Inc. or its affiliates. All rights reserved.  
This work may not be reproduced or redistributed, in whole or in part,  
without prior written permission from Amazon Web Services, Inc.  
Commercial copying, lending, or selling is prohibited.

## Introduction

In this Lab you will learn how to use AWS CloudFormation to provision and update a web application with a number of supporting AWS products and services, including Auto Scaling Groups, Amazon EC2 Instances, Elastic Load Balancers and more.

In the first part we will create a simple Amazon S3 bucket with AWS CloudFormation. We will also look at different retention policies applied when you delete an AWS CloudFormation stack or during a roll-back.

In the second part, we will provision a simple PHP web application using an Amazon Linux instance. We will then see how to re-apply an AWS CloudFormation template to the existing application to change some resource attributes like an Amazon EC2 instance type. We will enhance our application with IAM credentials and a KeyPair. Finally we will add an Elastic Load Balancer (ELB) and an Auto Scaling Group based on an Auto Scaling Configuration.

### What is AWS CloudFormation?

AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. It helps you leverage AWS products such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Block Store (EBS), Amazon Simple Notification Service (SNS), Elastic Load Balancing and Auto Scaling to build highly reliable, highly scalable, cost-effective applications without worrying about creating and configuring the underlying AWS infrastructure. AWS CloudFormation enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).

Additional information can be found at <http://aws.amazon.com/cloudformation/> and in the online documentation: <http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/Welcome.html>.

## Start your *qwikLAB*™

1. Start your *qwikLAB*™

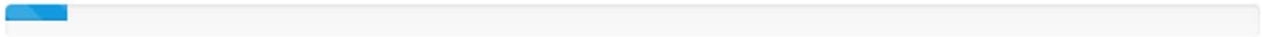
Use the 'Start Lab' button to start your lab.

(Hint: If you are prompted for a token, please use one you've been given or have purchased.)



You will see the lab creation in progress.

*Create in progress...*

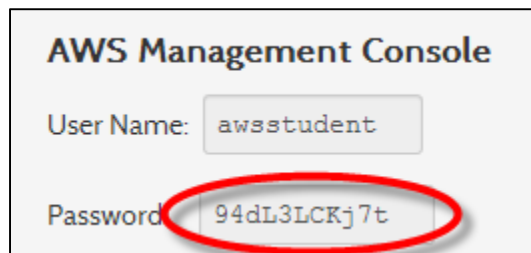


2. Note a few properties of the lab.

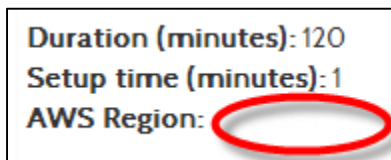
- a. **Duration** - The time the lab will run for before shutting itself down.
- b. **Setup Time** - The estimated lab creation time on starting the lab.
- c. **AWS Region** - The AWS Region the lab resources are being created in.

3. Copy the Password provided.

- d. Hint: selecting the value shown and using Ctrl+C works best



4. Note the AWS Region specified for the lab in *qwikLAB*™



5. Click the 'Open Console' button.



6. Make sure that you are not logged into any other instances of the AWS console (in a student account or your own account), as this may cause conflicts when you open the console and log in below for this lab.
7. Login to the AWS Management Console

Enter the User Name '**awsstudent**' and paste the password you copied from the lab details in *qwikLAB™* into the Password field.

Click on the 'Sign in using our secure server' button.

In this step you logged into the AWS Management Console using login credentials for a user provisioned via AWS Identity Access Management in an AWS account by *qwikLAB™*.

## Amazon Web Services Sign In

Please enter the AWS Identity & Access Management (IAM) User name and password assigned by your system administrator to sign in.

**AWS Account: 832809622232**

**User Name:**

**Password:**

[Sign in using our secure server](#)

Please contact your system administrator if you have forgotten your user credentials.

[Sign in using AWS Account credentials](#)

## Part 1: Create an S3 bucket using AWS CloudFormation

We will start with a very simple AWS CloudFormation template to create a single Amazon S3 bucket with public read rights.

### Download the AWS CloudFormation Template

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

More information here: <http://aws.amazon.com/s3/>

Download the first AWS CloudFormation template here:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-S3-v1.template>

You can have a look into the template with any plain text editor.

You should see the following content:

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Basic S3 CloudFormation template",
  "Resources": {
    "S3BucketForWebsiteContent": {
      "Type": "AWS::S3::Bucket",
      "Properties": {
        "AccessControl": "PublicRead"
      }
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": { "Ref": "S3BucketForWebsiteContent" },
      "Description": "Name of the newly created Amazon S3 Distribution"
    }
  }
}
```

Let's take a look at the file structure.

### What is an AWS CloudFormation Template?

Before we go any further, we should cover the basics of what a template is. A template is a declaration of the AWS resources that make up a stack. The template is stored as a text file whose format complies with the JavaScript Object Notation (JSON) standard. Because they are just text files, you can create and edit them in any text editor and manage them in your source control system with the rest of your source code. For more information about the JSON format, see <http://www.json.org>.

In the template, you use a JSON structure that AWS CloudFormation can interpret to declare the AWS resources you want to create and configure. In the JSON format, an object is declared as a name-value pair or a pairing of a name with a set of child objects enclosed within braces. Multiple sibling objects are separated by commas. An AWS CloudFormation template begins with an open brace and ends with a close brace. Within those braces, you

can declare six top level JSON objects: AWSTemplateFormatVersion, Description, Parameters, Mappings, Resources, and Outputs. Read more about each of these objects here:

<http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/concept-template.html#concept-template-outputs>

The only required top-level object is the Resources object, which must declare at least one resource. Let's start with the most basic template containing only a Resources object, which contains a single resource declaration.

### Resources: Hello Bucket!

The Resources object contains a list of resource objects contained within braces. A resource declaration contains the resource's attributes, which are themselves declared as child objects. A resource must have a *Type* attribute, which defines the kind of AWS resource you want to create. The *Type* attribute has a special format:

```
AWS::ProductIdentifier::ResourceType
```

For example, the resource type for an Amazon S3 bucket is AWS::S3::Bucket. For a full list of resource types, see Template Reference here:

<http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/template-reference.html>

Let's take a look at a very basic template. The following template declares a single resource of type AWS::S3::Bucket: with the name S3BucketForWebsiteContent.

```
{
  "Resources" : {
    "S3BucketForWebsiteContent" : {
      "Type" : "AWS::S3::Bucket"
    }
  }
}
```

The syntactic elements are quoted strings. If you use this template to create a stack, AWS CloudFormation will create an Amazon S3 bucket. Creating a bucket is simple, because AWS CloudFormation can create a bucket with default settings. For other resources, such as a CloudFront distribution, Auto Scaling group, or EC2 instance, AWS CloudFormation requires more information. Resource declarations use a *Properties* attribute to specify the information used to create a resource.

Depending on the resource type, some properties are required, such as the ImageId property for an AWS::EC2::Instance resource, and others are optional. Some properties have default values, such as the AccessControl property of the AWS::S3::Bucket resource, so specifying a value for those properties is optional. Other properties are not required but may add functionality that you want, such as the WebsiteConfiguration property of the AWS::S3::Bucket resource. Specifying a value for such properties is entirely optional and based on your needs. In the example above, because the AWS::S3::Bucket resource has only optional properties and we didn't need any of the optional features, we could accept the defaults and omit the Properties attribute.

To view the properties for each resource type, see the topics in Resource Property Types Reference here:

<http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/aws-product-property-reference.html>



## Resource Properties and Using Resources Together

Usually, a property for a resource is simply a string value. For example, the following template specifies a canned ACL (PublicRead) for the AccessControl property of the bucket.

```
{
  "Resources" : {
    "S3BucketForWebsiteContent" : {
      "Type" : "AWS::S3::Bucket",
      "Properties" : {
        "AccessControl" : "PublicRead"
      }
    }
  }
}
```

## Optional Outputs

In the Outputs section, you can optionally define custom values that are returned in response to the `cfn-describe-stacks` command. These output values can include information based on literals, resources, parameters, pseudo parameters, and intrinsic functions.

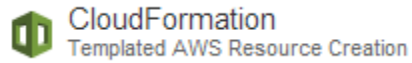
```
"Outputs": {
  "BucketName": {
    "Value": { "Ref": "S3BucketForWebsiteContent" },
    "Description": "Name of the newly created Amazon S3 Distribution"
  }
}
```

We didn't cover two top level objects in a template: `AWSTemplateFormatVersion` and `Description`. `AWSTemplateFormatVersion` is simply the version of the template format—if you don't specify it, AWS CloudFormation will use the latest version. The `Description` is any valid JSON string and this description appears in the Specify Parameters page of the Create Stack wizard.

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "Basic S3 CloudFormation template",
```

## Select the CloudFormation Service

- 1) Select "CloudFormation" from the Console Home.



## Confirm your AWS Region

- 2) Select or confirm that the same AWS Region is already set in the AWS Management Console



Note: You should use the same AWS Region indicated by *qwikLAB™*.

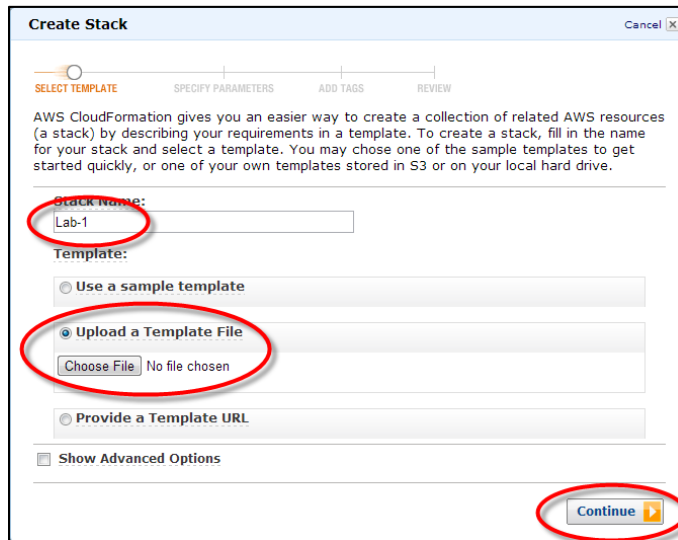
## Create a CloudFormation Stack

We will now use CloudFormation to create a stack of AWS resources.

1. Click on the “Create New Stack” button.



2. On the next screen, enter “Lab-1” in the Stack Name field. Click on “Upload a Template File”
3. Click Choose File and select the AWS CloudFormation template file that you just downloaded
4. Click on “Continue”.

A screenshot of the "Create Stack" dialog box, specifically the "SELECT TEMPLATE" step. The progress bar at the top shows "SELECT TEMPLATE" as the active step. The "Stack Name" field contains "Lab-1". Under the "Template:" section, the "Upload a Template File" radio button is selected. Below it, the "Choose File" button is highlighted with a red circle, and the text "No file chosen" is displayed. At the bottom right, the "Continue" button with a right-pointing arrow is also highlighted with a red circle. A "Cancel" link is in the top right corner.

**Create Stack** Cancel

SELECT TEMPLATE | SPECIFY PARAMETERS | ADD TAGS | REVIEW

AWS CloudFormation gives you an easier way to create a collection of related AWS resources (a stack) by describing your requirements in a template. To create a stack, fill in the name for your stack and select a template. You may choose one of the sample templates to get started quickly, or one of your own templates stored in S3 or on your local hard drive.

Stack Name:  
Lab-1

Template:

☐ Use a sample template

☒ Upload a Template File

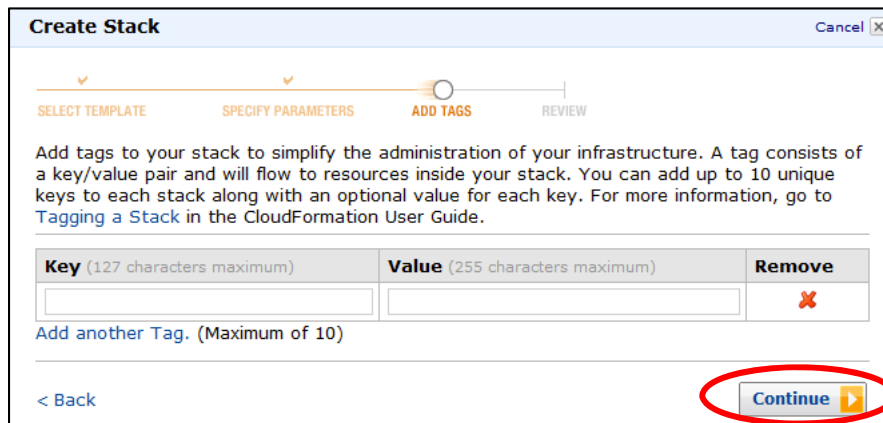
Choose File No file chosen

☐ Provide a Template URL

☐ Show Advanced Options

Continue

5. Click on Continue on the “Add tags” screen

A screenshot of the "Create Stack" dialog box, specifically the "ADD TAGS" step. The progress bar at the top shows "ADD TAGS" as the active step. Below the progress bar, there is explanatory text about tags. A table with three columns: "Key (127 characters maximum)", "Value (255 characters maximum)", and "Remove" is shown. The "Remove" column contains a red 'X' icon. Below the table, there is a link "Add another Tag. (Maximum of 10)". At the bottom left, there is a "< Back" link. At the bottom right, the "Continue" button with a right-pointing arrow is highlighted with a red circle. A "Cancel" link is in the top right corner.

**Create Stack** Cancel

SELECT TEMPLATE | SPECIFY PARAMETERS | ADD TAGS | REVIEW

Add tags to your stack to simplify the administration of your infrastructure. A tag consists of a key/value pair and will flow to resources inside your stack. You can add up to 10 unique keys to each stack along with an optional value for each key. For more information, go to [Tagging a Stack](#) in the CloudFormation User Guide.

Key (127 characters maximum)	Value (255 characters maximum)	Remove
		X

[Add another Tag. \(Maximum of 10\)](#)

< Back

Continue

## 6. Review the configuration and click on “Continue”

The screenshot shows the 'Create Stack' wizard in the AWS Management Console, specifically the 'REVIEW' step. The progress bar at the top indicates the steps: SELECT TEMPLATE, SPECIFY PARAMETERS, ADD TAGS, and REVIEW (which is the current step). Below the progress bar, a message says: 'Please review the information below, then click Continue to create the stack.' The 'Stack Information' section includes: Stack Name: Lab-1, Stack Description: Basic S3 Bucket CloudFormation template, Template: https://cf-templates-1vkimx4x52pz8-us-west-2.s3.amazonaws.com/201311SiOI-CloudFormation-S3-v1.template.txt, IAM Acknowledgement: false, and Estimated Cost: Cost. The 'Notification' section includes: Notification: none, Creation Timeout (minutes): none, and Rollback on Failure: true. At the bottom right, the 'Continue' button is highlighted with a red circle.

## 7. Then click on “Close”

- You will for see the new stack in status “CREATE\_IN\_PROGRESS” until the S3 bucket creation is complete. It should not take more than 1 or 2 minutes. Click on the “Refresh” button every 30 seconds till you see the stack go to status “CREATE\_COMPLETE”.

CloudFormation Stacks ( Showing 1 of 1 )				
<div>  Create Stack                      Update Stack                      Delete Stack                     Viewing: Active                      Show/Hide                      Refresh                 </div>				
	Name	Created	Status	Description
<input checked="" type="checkbox"/>	Lab-1	2013-04-25 13:19:59 UTC-7	CREATE_IN_PROGRESS	Basic S3 Buck...

<b>Stack: Lab-1</b>	
<b>Description</b>	<div> <div>Outputs</div> <div>Resources</div> <div>Events</div> <div>Template</div> <div>Parameters</div> <div>Tags</div> </div>
<b>Stack Name:</b>	Lab-1
<b>Stack ID:</b>	arn:aws:cloudformation:us-west-2:647214902741:stack/Lab-1/80a41850-ade5-11e2-ac80-507bfc8736d2
<b>Status:</b>	CREATE_IN_PROGRESS
<b>Status (Reason):</b>	User Initiated
<b>Created:</b>	2013-04-25 13:19:59 UTC-7
<b>Description:</b>	Basic S3 Bucket CloudFormation template

- After 1 or 2 minutes, you should see the status change to "CREATE\_COMPLETE". Navigate between the different tabs. The "Outputs" tabs will give you the output values specified by the template. In this example, it will give you the name of the newly created S3 bucket (your display will vary).

	Name	Created	Status	Description
<input checked="" type="checkbox"/>	Lab-1	2013-04-25 13:19:59 UTC-7	CREATE_COMPLETE	Basic S3 Buck...

**Stack: Lab-1**
Description
**Outputs**
Resources
Events
Template
Parameters
Tags
Refresh

**Stack Outputs**

Output values may have been specified by the template author and will be available when stack creation is complete.

Key	Value	Description
BucketName	lab-1-s3bucketforwebsitecontent-1fuz8xujyq9j7	Name of the newly created Amazon S3 Distribution

- Navigate to the S3 service in the AWS Console.



- You will see your newly created bucket. Please notice also that CloudFormation automatically created another bucket named "cf-template-<xxxx>" where it will store all the templates you have uploaded.

All Buckets	
	Name
	cf-templates-1vkimx4x52pz8-us-west-2
	lab-1-s3bucketforwebsitecontent-1fuz8xujyq9j7

## Delete the Stack


We will now delete the stack we have just created and see how all resources created by this stack will be deleted too.

- Navigate back to the AWS CloudFormation Service on the Console.
- Select the "Lab-1" stack and click on "Delete Stack"

CloudFormation Stacks ( Showing 1 of 1 )				
			viewing: Active	
	Name	Created	Status	Description
<input checked="" type="checkbox"/>	Lab-1	2013-04-25 13:19:59 UTC-7	CREATE_COMPLETE	Basic S3 Buck...

- Confirm the stack deletion. The stack should now be in "DELETE\_IN\_PROGRESS" status for 1-2 minutes. Click on the Refresh button every minute or so.
- After 1-2 minutes, the stack should now be gone and the list empty

5. Navigate to the S3 service. The S3 bucket previously created has been deleted. Please note that the bucket created by CloudFormation for template storage is still available.

All Buckets	
	Name
	cf-templates-1vkimx4x52pz8-us-west-2

### Change the Retention Policy

A common use case is to store application assets in S3. These assets can be results of heavy data processing, files uploaded by users or any other valuable data. Automatically deleting an S3 bucket when deleting an AWS CloudFormation stack is usually not what we would want to happen. We will see now how to configure the CloudFormation template and specify that it should not delete some of the resources when deleting the stack.

Let's edit our previous downloaded template as follows:

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Basic S3 CloudFormation template",
  "Resources": {
    "S3BucketForWebsiteContent": {
      "Type": "AWS::S3::Bucket",
      "DeletionPolicy": "Retain",
      "Properties": {
        "AccessControl": "PublicRead"
      }
    }
  },
  "Outputs": {
    "BucketName": {
      "Value": { "Ref": "S3BucketForWebsiteContent" },
      "Description": "Name of the newly created Amazon S3 Distribution"
    }
  }
}
```

We just add the following line in the S3 bucket resource definition:

```
"DeletionPolicy" : "Retain",
```

This line indicates that the resource should not be deleted once created, either in case of user initiated stack deletion or in case of roll-back.

Please edit the file and save it with the following name: "10-CF-S3-2.template.txt".

For your convenience, the completed modified template can be downloaded here:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-S3-v2.template>

## Create a New Stack with the Modified Retention Policy

We will now create a new stack from our modified template, then delete it and verify that the newly created S3 bucket is still available.

1. Navigate to the AWS CloudFormation Service, and create a new stack named “Lab-2” with the modified template.
2. Wait for 1-2 minutes for the stack to complete, select it and look at the “Outputs” tab to check the name of the S3 bucket

Name	Created	Status	Description
<input checked="" type="checkbox"/> Lab-2	2013-04-25 13:45:21 UTC-7	CREATE_COMPLETE	Re:Invent lab ...

**Stack: Lab-2**

Description Outputs Resources Events Template Parameters Tags

**Stack Outputs**

Output values may have been specified by the template author and will be available when stack creation is complete.

Key	Value	Description
BucketName	lab-2-s3bucketforwebsitecontent-r7of4si8ve7x	Name of the newly created Amazon S3 Distribution

3. Navigate to the Amazon S3 service and locate the new bucket
4. Navigate back to the AWS CloudFormation service, select the “Lab-2” stack and click on “Delete Stack”. Wait for 1-2 minutes for the stack deletion to complete.

Name	Created	Status	Description
<input checked="" type="checkbox"/> Lab-2	2013-04-25 13:45:21 UTC-7	DELETE_IN_PROGRESS	Re:Invent lab ...

**Stack: Lab-2**

Description Outputs Resources Events Template Parameters Tags

**Stack Name:** Lab-2

**Stack ID:** arn:aws:cloudformation:us-west-2:647214902741:stack/Lab-2/0bd882a0-ade9-11e2-aa4b-50ba1b9b0aa6

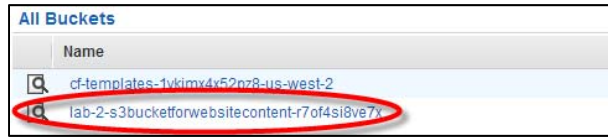
**Status:** DELETE\_IN\_PROGRESS

**Status (Reason):** User Initiated

**Created:** 2013-04-25 13:45:21 UTC-7

**Description:** Re:Invent lab - basic S3 CloudFormation template

5. Navigate to the Amazon S3 service and verify that the S3 bucket has not been deleted.



## Conclusion of Part One

In this first part we have gone through AWS CloudFormation basics, the structure of the JSON file template and simple resources creation. We have learnt about DeletionPolicy and how resources are deleted.

In the next part we will review more advanced features of AWS CloudFormation: using cfn-init, receiving inputs, using mappings and updating stacks.



## Part 2: Provision a Web Application

This section walks through a simple progression of updates of a running stack. It shows how the use of templates makes it possible to use a version control system for the configuration of your AWS infrastructure, just as you use version control for the software you are running. We will walk through the following steps:

1. Create the Initial Stack—create a stack using a base Amazon Linux AMI, installing the Apache Web Server and a simple PHP application using the AWS CloudFormation helper scripts.
2. Update the Application—update one of the files in the application and deploy the software using AWS CloudFormation.
3. Update the Instance Type—change the instance type of the underlying Amazon EC2 instance.
4. Update the AMI on an Amazon EC2 instance—change the Amazon Machine Image (AMI) for the Amazon EC2 instance in your stack.
5. Add a Key Pair to an Instance—add an Amazon EC2 key pair to the instance, and then update the security group to allow SSH access to the instance.
6. Update IAM Policies—update the permissions of an IAM user defined in the template.
7. Change the Stack's Resources—add and remove resources from the stack, converting it to an auto-scaled, load-balanced application by updating the template.

This section is heavily based on the online documentation:

<http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/updating.stacks.walkthrough.html>

### A Simple Application

We'll begin by creating a stack that we can use throughout the rest of this section. We have provided a simple template that launches a single instance PHP web application hosted on the Apache Web Server and running on an Amazon Linux AMI.

The Apache Web Server, PHP, and the simple PHP application are all installed by the AWS CloudFormation helper scripts that are installed by default on the Amazon Linux AMI. The following template snippet shows the metadata that describes the packages and files to install, in this case the Apache Web Server and the PHP infrastructure from the Yum repository for the Amazon Linux AMI. The snippet also shows the Services section, which ensures that the Apache Web Server is running. In the Properties section of the Amazon EC2 instance definition, the UserData property contains the CloudInit script that calls cfn-init to install the packages and files.

The application itself is a very simple two-line "Hello, World" example that is entirely defined within the template. For a real-world application, the files may be stored on Amazon S3, GitHub, or another repository and referenced from the template. AWS CloudFormation can download packages (such as RPMs or RubyGems), as well as reference individual files and expand .zip and .tar files to create the application artifacts on the Amazon EC2 instance.

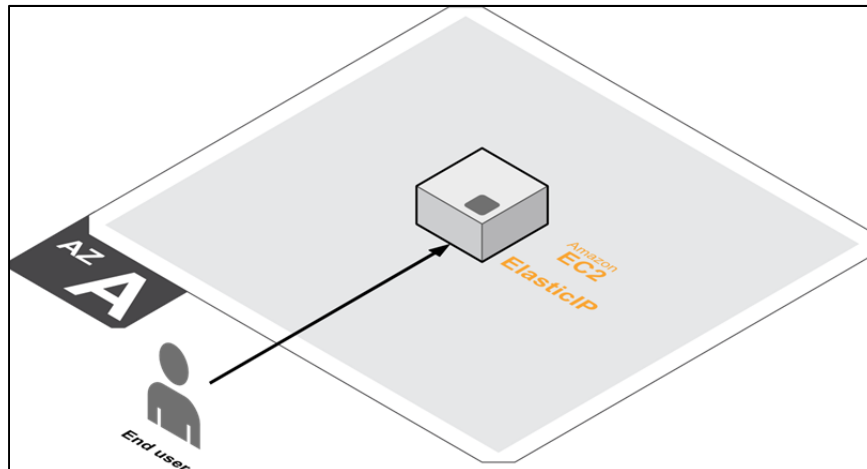
The template enables and configures the cfn-hup daemon to listen for changes to the configuration defined in the metadata for the Amazon EC2 instance. By using the cfn-hup daemon, you can update application software, such

as the version of Apache or PHP, or you can update the PHP application file itself from AWS CloudFormation. The following snippet from the same EC2 resource in the template shows the pieces necessary to configure cfn-hup to call cfn-init to update the software if any changes to the metadata are detected:

```
"WebServerHost": {
  "Type": "AWS::EC2::Instance",
  "Metadata": {
    "AWS::CloudFormation::Init": {
      "config": {
        "packages": {
          "yum": {
            "httpd": [],
            "php": []
          }
        },
        "files": {
          "/var/www/html/index.php": {
            "content": {
              "Fn::Join": [
                "", [
                  "<?php\n",
                  "echo '<h1>AWS CloudFormation sample PHP application</h1>';\n",
                  "echo '<p>', { "Ref": "WelcomeMessage" }, "</p>";\n",
                  "?>\n"
                ]
              ]
            },
            "mode": "000644",
            "owner": "apache",
            "group": "apache"
          }
        }
      }
    },
    "services": {
      "sysvinit": {
        "httpd": {
          "enabled": "true",
          "ensureRunning": "true"
        },
        "sendmail": {
          "enabled": "false",
          "ensureRunning": "false"
        }
      }
    }
  },
  "Properties": {
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [
          "", [
            "#!/bin/bash\n",
            "yum update -y aws-cfn-bootstrap\n",
            "\n",
            "# Install the simple web page\n",
            "/opt/aws/bin/cfn-init -s \" { \"Ref\": \"AWS::StackName\" },\n",
            "  -r WebServerHost\n",
            "  --access-key \" { \"Ref\": \"WebServerKeys\" },\n",
            "  --secret-key \" { \"Fn::GetAtt\": [\"WebServerKeys\", \"SecretAccessKey\"] },\n",
            "  --region \" { \"Ref\": \"AWS::Region\" },\n",
            " || error exit 'Failed to run cfn-init'\n",
            ""
          ]
        ]
      }
    }
  }
}
```

To complete the stack, the template creates an Amazon EC2 security group, an elastic IP so that we have a consistent IP address to reference the application. Here's the complete template, which you can also download or reference at

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v1.template>



This example uses a single EC2 instance and Elastic IP address, but you can use the same mechanisms on more complex solutions that make use of Elastic Load Balancers and Auto Scaling groups to manage a collection of application servers. There are, however, some special considerations for Auto Scaling groups.

### Create the Initial Stack

Navigate to the AWS CloudFormation service on the Console.

1. Click on “Create New Stack”



2. Enter “Lab-3” as a Stack Name. Click on “Provide a Template URL” and paste the template URL:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v1.template>

A screenshot of the AWS CloudFormation 'Create Stack' console window. The window has a title bar 'Create Stack' and a 'Cancel' button. Below the title bar is a progress bar with four steps: 'SELECT TEMPLATE' (active), 'SPECIFY PARAMETERS', 'ADD TAGS', and 'REVIEW'. The main content area contains instructions: 'AWS CloudFormation gives you an easier way to create a collection of related AWS resources (a stack) by describing your requirements in a template. To create a stack, fill in the name for your stack and select a template. You may choose one of the sample templates to get started quickly, or one of your own templates stored in S3 or on your local hard drive.' Below this are three input fields: 'Stack Name:' with the value 'Lab-3', 'Template:', and a radio button selection for 'Provide a Template URL' (which is selected). The 'Provide a Template URL' field contains the URL 'https://us-east-1-aws-training.s3.amazonaws.com/'. There is also a checkbox for 'Show Advanced Options'. At the bottom right is a 'Continue' button with a right-pointing arrow. Red circles highlight the 'Stack Name' field, the 'Provide a Template URL' radio button and its corresponding field, and the 'Continue' button.

- On the “Specify parameters” page, click on “I acknowledge that this template may create IAM resources”, otherwise you may see the following error:

✖ CreateStack failed: Requires capabilities : [CAPABILITY\_IAM]

This step is necessary because the template creates an IAM user that is locked down to only allow access to the API actions necessary for cfn-init and cfn-hup. The credentials for the IAM user are stored on the newly created EC2 instance.

**Create Stack** [Cancel]

SELECT TEMPLATE | **SPECIFY PARAMETERS** | ADD TAGS | REVIEW

**Stack Description:** AWS CloudFormation Sample Template UpdateTutorial Part 1: Sample template that can be used to test EC2 updates. **\*\*WARNING\*\*** This template creates an Amazon EC2 Instance. You will be billed for the AWS resources used if you create a stack from this template.

**Specify Parameters**  
Below are the parameters associated with your CloudFormation template. You may review and proceed with the default parameters or make customizations as needed below.

**WebServerInstanceType** m1.small  
WebServer EC2 instance type

☒ **I acknowledge that this template may create IAM resources**

< Back **Continue** ▶

- Click “Continue” on the next screen “Add Tags”
- Review the parameters and click on “Continue”
- The template takes about 5 minutes to complete. After the status of your stack is CREATE\_COMPLETE,
  - the output tab will display the URL of your website:

	Name	Created	Status	Description
<input checked="" type="checkbox"/>	Lab-3	2013-04-25 13:55:36 UTC-7	<span style="color: green;">●</span> CREATE_COMPLETE	AWS CloudFor...

Stack: Lab-3

Description

**Outputs**

Resources

Events

Template

Parameters

Tags

**Stack Outputs** Refresh

*Output values may have been specified by the template author and will be available when stack creation is complete.*

Key	Value	Description
WebsiteURL	<span style="border: 1px solid red; border-radius: 50%; padding: 2px;">http://54.214.20.30</span>	Application URL

7. Connect the web site, you should see a simple page with the following message:



## Change Resources Properties

With AWS CloudFormation, you can change the properties of an existing resource in the stack. The following sections describe various updates that solve specific problems; however, any property of any resource that supports updating in the stack can be modified as necessary.

### Update the Instance Type

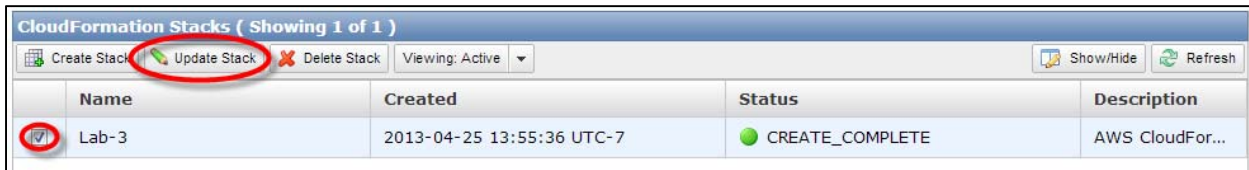
The stack we have built so far uses an m1.small Amazon EC2 instance. Let's suppose that your newly created website is getting more traffic than an m1.small instance can handle, and now you want to move to an m1.medium EC2 instance type. If the architecture of the instance type changes from 32 bit to 64 bit, the instance will be created with a different AMI. If you check out the mappings in the template, you will see that both the m1.small and m1.medium are 32 bit architectures, and they use the same base Amazon Linux AMI.

```
{
  "AWSInstanceType2Arch": {
    "c1.medium": { "Arch": "64" },
    "c1.xlarge": { "Arch": "64" },
    "cc1.4xlarge": { "Arch": "64" },
    "m1.small": { "Arch": "64" },
    "m1.medium": { "Arch": "64" },
    "m1.large": { "Arch": "64" },
    "m1.xlarge": { "Arch": "64" },
    "m2.xlarge": { "Arch": "64" },
    "m2.4xlarge": { "Arch": "64" },
    "m2.xlarge": { "Arch": "64" },
    "t1.micro": { "Arch": "64" }
  },
  "AWSRegionArch2AMI": {
    "us-east-1": { "32": "ami-5675ee3f", "64": "ami-3275ee5b" },
    "us-west-2": { "32": "ami-d0be2ae0", "64": "ami-ecbe2adc" },
    "us-west-1": { "32": "ami-d8d1fc9d", "64": "ami-66d1fc23" },
    "eu-west-1": { "32": "ami-6893991c", "64": "ami-44939930" },
    "ap-southeast-1": { "32": "ami-a29ed2f0", "64": "ami-aa9ed2f8" },
    "ap-southeast-2": { "32": "ami-383eaf02", "64": "ami-363eaf0c" },
    "ap-northeast-1": { "32": "ami-0f3fbf0e", "64": "ami-173fbf16" },
    "sa-east-1": { "32": "ami-a56bb0b8", "64": "ami-dd6bb0c0" }
  }
}
```

Let's use the template that we modified in the previous section to change the instance type. Because InstanceType was an input parameter to the template, we don't need to modify the template; we can simply change the value of the parameter in the Stack Update wizard, on the Specify Parameters page.

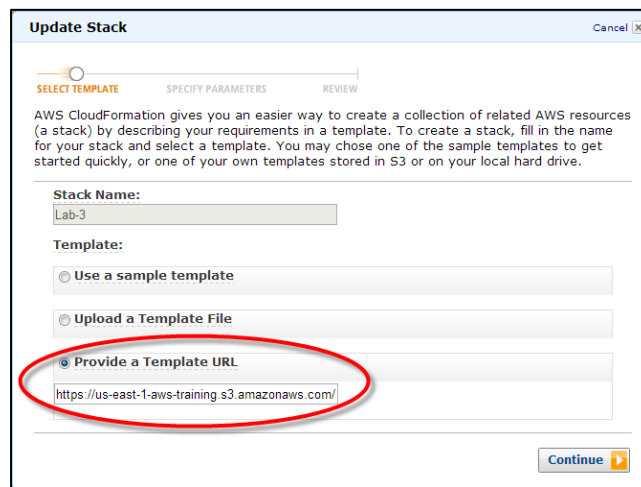
### To update the stack from the AWS Management Console

1. In the AWS CloudFormation console, select the “Lab-3” stack. Click on “Update Stack”

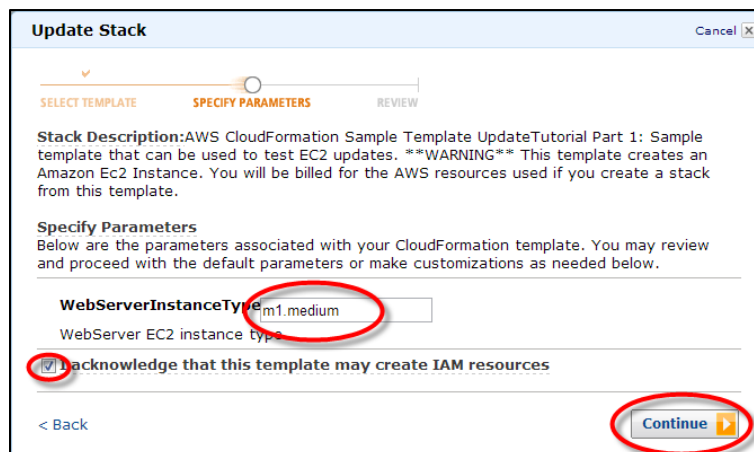


2. In the Update Stack wizard, on the Select Template page, Click on “Provide a Template URL” and paste the template URL:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v1.template>



3. The Specify Parameters page appears with the parameters that were used to create the initial stack are pre-populated in the Specify Parameters section. Enter “m1.medium” in the WebServerInstanceType field, and check the “I acknowledge that this template may create IAM resources”



4. On the **Review** page, verify that all the settings are as you want them, and then click Continue.
5. Wait for about 5 minutes for the CloudFormation update to complete.

You can dynamically change the instance type of an EBS-backed Amazon EC2 instance by starting and stopping the instance. AWS CloudFormation tries to optimize the change by updating the instance type and restarting the instance, so the instance ID does not change. When the instance is restarted, however, the public IP address of the instance does change. To ensure that the Elastic IP address is bound correctly after the change, AWS CloudFormation will also update the Elastic IP address. You can see the changes in the AWS CloudFormation console on the Events tab.

Here we have changed more than just the Instance Type, hence the first instance is terminated and a new one is launched.

To check the instance type from the AWS Management Console, open the Amazon EC2 console, and locate your instance there.

 empty	 i-9000c5a5	ami-30fe7300	ebs	m1.medium	 pending	 initializing...
---	--	--------------	-----	-----------	---	---

## Add Resources Properties

So far, we've looked at changing existing properties of a resource in a template. You can also add properties that were not originally specified in the template. To illustrate that, we'll add an Amazon EC2 Key Pair to an existing EC2 instance and then open up port 22 in the Amazon EC2 Security Group so that you can use Secure Shell (SSH) to access the instance.

### Add a Key Pair to an Instance

To add SSH access to an existing Amazon EC2 instance

1. Add an additional parameter to the template to pass in the name of an existing EC2 key pair.

```
"Parameters" : {
  "WebServerKeyName" : {
    "Description" : "Name of an existing Amazon EC2 key pair for SSH access",
    "Type" : "String"
  },
},
```

2. Add the KeyName property to the Amazon EC2 instance.

```
"WebServerHost": {
  "Type" : "AWS::EC2::Instance",
  "Properties": {
    "KeyName" : { "Ref" : "WebServerKeyName" }
  }
},
```

3. Add port 22 to the ingress rules for the Amazon EC2 security group.

```
"WebServerSecurityGroup" : {
  "Type" : "AWS::EC2::SecurityGroup",
  "Properties" : {
    "GroupDescription" : "Enable HTTP and SSH",
    "SecurityGroupIngress" : [
      { "IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : "0.0.0.0/0" },
      { "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0" }
    ]
  }
},
```

You can download or reference the completed updated template at

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v2.template>

- Go to the EC2 console and then the Key Pairs and copy the Key Pair Name to your clipboard so you use it as the value for the WebServerKeyName parameter of your template when you update your stack.

	Key Pair Name	Fingerprint
<input checked="" type="checkbox"/>	qwiklab-I22-6470	2e:e7:0a:8f:9f:b5:d2:b8:89:36:eb:0d:e9:cb:10:62:32:fc:4b:93

- Click on "Update Stack". Select "Upload a Template File" and specify your edited or downloaded template. Click on Continue.

**Update Stack** Cancel X

SELECT TEMPLATE SPECIFY PARAMETERS REVIEW

AWS CloudFormation gives you an easier way to create a collection of related AWS resources (a stack) by describing your requirements in a template. To create a stack, fill in the name for your stack and select a template. You may choose one of the sample templates to get started quickly, or one of your own templates stored in S3 or on your local hard drive.

Stack Name:  
Lab-3

Template:

☐ Use a sample template

☒ Upload a Template File

Choose File CloudFormat...mplate.txt

☐ Provide a Template URL

Continue

- On the next screen you will notice a new blank "WebServerKeyName". Paste the Key Pair Name you copied from the EC2 console from your clipboard (select the name and use your key board short cut for Copy) and check the "I acknowledge that this template may create IAM resources".



## Launching and Managing a Web Application With AWS CloudFormation

**Update Stack** [Cancel]

SELECT TEMPLATE | **SPECIFY PARAMETERS** | REVIEW

**Stack Description:** AWS CloudFormation Sample Template UpdateTutorial Part 3: Sample template that can be used to test EC2 updates. **\*\*WARNING\*\*** This template creates an Amazon EC2 Instance. You will be billed for the AWS resources used if you create a stack from this template.

**Specify Parameters**  
Below are the parameters associated with your CloudFormation template. You may review and proceed with the default parameters or make customizations as needed below.

**WebServerInstanceType** m1.medium  
WebServer EC2 instance type

**WebServerKeyName** qwiklab-l22-6470  
Name of an existing Amazon EC2 KeyPair for SSH access

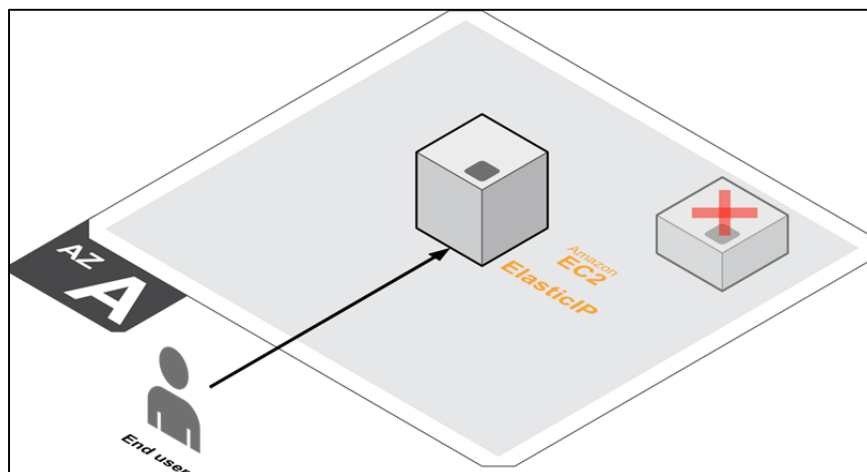
☒ I acknowledge that this template may create IAM resources

< Back Continue

7. Review the configuration and launch the AWS CloudFormation stack update. It should take around 4 minutes to complete.
8. Connect to your web site, you should see the following message showing that the stack has been successfully updated.



The application architecture is very similar to the previous one, but we have switched from an m1.small machine to a bigger m1.medium.



## Update IAM Policies

Next, we'll update the IAM policy associated with the IAM user that is passed to and used by code running on the Amazon EC2 instance. Suppose that a new version of the application requires access to the Amazon EC2 API from the instance. To enable access, update the IAM policy in the template as follows:

**You actually don't need to edit the JSON file. Please review the modifications and download the edited file at the end of the section.**

```
"WebServerUser" : {
  "Type" : "AWS::IAM::User",
  "Properties" : {
    "Path": "/",
    "Policies": [{
      "PolicyName": "root",
      "PolicyDocument": {
        "Statement": [{
          "Effect": "Allow",
          "Action": [
            "cloudformation:DescribeStackResource",
            "ec2:*"
          ],
          "Resource": "*"
        }]
      }
    ]
  }
},
```

This change will modify the policy for the user; it does not require other changes in the stack. When the stack is updated, the user credentials on the Amazon EC2 instance will have access to the Amazon EC2 API.

## Change the Stack's Resources

Since application needs can change over time, AWS CloudFormation allows you to change the set of resources that make up the stack. To demonstrate, we'll take the single instance application from Adding Resource Properties and convert it to an auto-scaled, load-balanced application by updating the stack.

Editing the JSON file can be error-prone. Please review the modification below and download the completed file here:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v3.template>

**Find below the changes that were made between the v2 and the v3 template.**

The previous template creates a simple, single instance PHP application using an Elastic IP address. We'll now turn the application into a highly available, auto-scaled, load balanced application by changing its resources during an update.

1. Remove the Elastic IP address resource from the template.

```
"Endpoint" : {
  "Type" : "AWS::EC2::EIP",
  "Properties" : {
    "InstanceId" : {
      "Ref" : "WebServerHost"
    }
  }
},
```

2. Add an Elastic Load Balancer resource.

```
"ElasticLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "AvailabilityZones" : { "Fn::GetAZs" : "" },
    "Listeners" : [ {
      "LoadBalancerPort" : "80",
      "InstancePort" : "80",
      "Protocol" : "HTTP"
    } ],
    "HealthCheck" : {
      "Target" : "HTTP:80/",
      "HealthyThreshold" : "3",
      "UnhealthyThreshold" : "5",
      "Interval" : "30",
      "Timeout" : "5"
    }
  }
},
  }
```

3. Convert the EC2 instance in the template into an Auto Scaling Launch Configuration. The properties are identical, so we only need to change the type name from:

```
"WebServerHost": {
  • "Type" : "AWS::EC2::Instance",
```

To:

```
"WebServerConfig": {
  "Type" : "AWS::AutoScaling::LaunchConfiguration",
```

For clarity in the template, the name of the resource has been changed from *WebServerHost* to *WebServerConfig*, so you'll need to update the resource name referenced by *cfn-init* and *cfn-hup* (just search for *WebServerHost* and replace it with *WebServerConfig*).

## 4. Add an Auto Scaling Group resource.

```
"WebServerGroup" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "AvailabilityZones" : { "Fn::GetAZs" : "" },
    "LaunchConfigurationName" : { "Ref" : "WebServerConfig" },
    "MinSize" : "1",
    "MaxSize" : "3",
    "LoadBalancerNames" : [ { "Ref" : "ElasticLoadBalancer" } ]
  }
},
```

## 5. Update the Security Group definition to lock down the traffic to the instances from the load balancer.

```
"WebServerSecurityGroup" : {
  "Type" : "AWS::EC2::SecurityGroup",
  "Properties" : {
    "GroupDescription" : "Enable SSH access and HTTP from the load balancer only",
    "SecurityGroupIngress" : [{
      "IpProtocol" : "tcp",
      "FromPort" : "22",
      "ToPort" : "22",
      "CidrIp" : "0.0.0.0/0"
    }, {
      "IpProtocol" : "tcp",
      "FromPort" : "80",
      "ToPort" : "80",
      "SourceSecurityGroupOwnerId" : { "Fn::GetAtt" :
        ["ElasticLoadBalancer", "SourceSecurityGroup.OwnerAlias"] },
      "SourceSecurityGroupName" : { "Fn::GetAtt" :
        ["ElasticLoadBalancer", "SourceSecurityGroup.GroupName"] }
    }
  ]
},
```

## 6. Update the Outputs to return the DNS Name of the Elastic Load Balancer as the location of the application from:

```
WebsiteURL" : {
  "Value" : { "Fn::Join" : [ "", [ "http://", { "Ref" : "Endpoint" } ] ] },
  "Description" : "Application URL"
}
```

To:

```
"WebsiteURL" : {
  "Value" : { "Fn::Join" : [ "", [ "http://",
    { "Fn::GetAtt" : [ "ElasticLoadBalancer", "DNSName" ] } ] ] },
  "Description" : "Application URL"
}
```

If you use this template to update the stack, you will convert your simple, single instance application into a highly available, multi-AZ, auto-scaled and load balanced application. Only the resources that need to be updated will be altered, so had there been any data stores for this application, the data would have remained intact. Now, you can use AWS CloudFormation to grow or enhance your stacks as your requirements change.

## Update the stack with the full template

Download the completed template here:

<https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-10/CloudFormation-WebApp-v3.template>

1. Navigate to the AWS CloudFormation service in the AWS Console
2. Locate your “Lab-3” stack, and click on “Update Stack”
3. Review and validate all the parameters. Don’t forget to click on “I acknowledge that this template may create IAM resources”. Launch the update process.
4. After about 5 minutes, the updates should be complete.
5. **Wait another 3 minutes after AWS CloudFormation update is finished, for Auto Scaling / ELB to launch and configure all resources.**
6. Locate the new web server address. This is not an IP address anymore but a DNS name of the Elastic Load Balancer (ELB) of your newly high availability application.

	Name	Created	Status	Description
<input checked="" type="checkbox"/>	Lab-3	2013-04-25 14:51:03 UTC-7	UPDATE_COMPLETE	AWS CloudFor...

**Stack: Lab-3**

[Description](#)
[Outputs](#)
[Resources](#)
[Events](#)
[Template](#)
[Parameters](#)
[Tags](#)

**Stack Outputs** [Refresh](#)

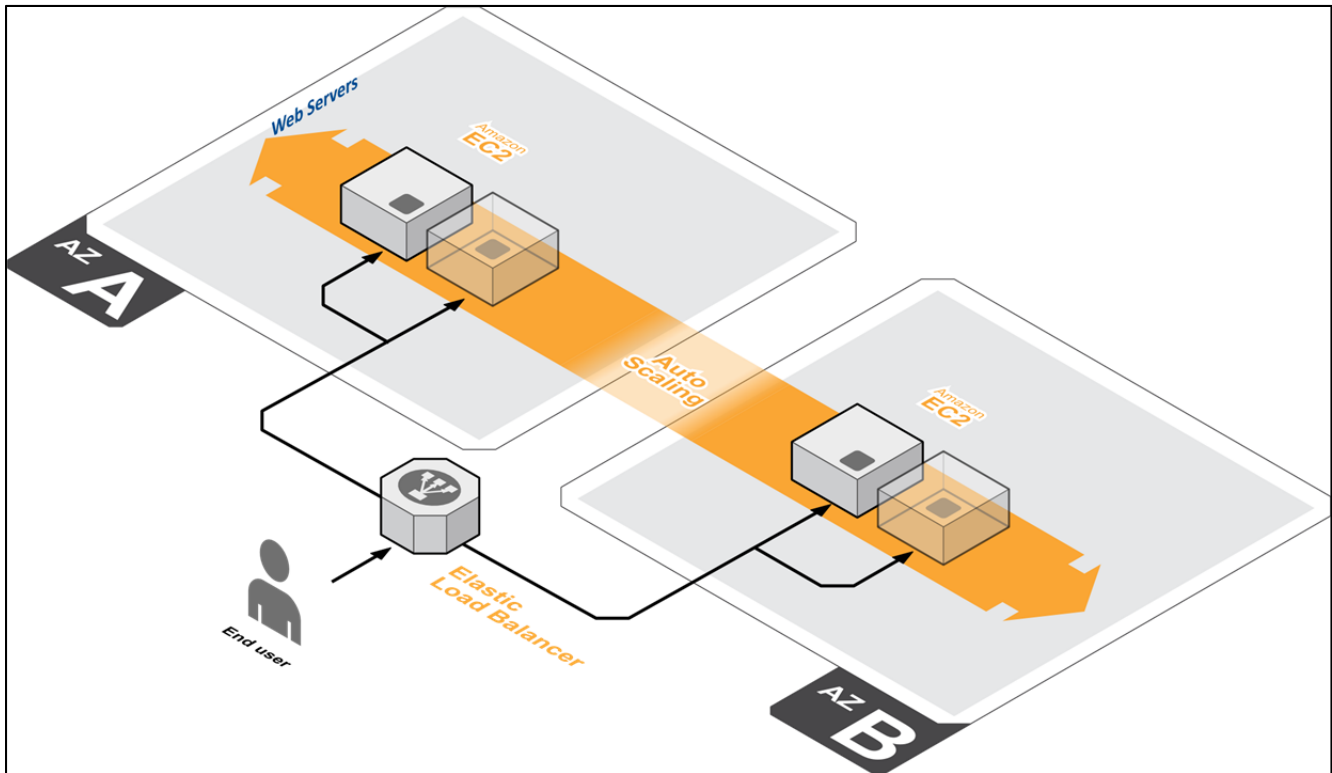
*Output values may have been specified by the template author and will be available when stack creation is complete.*

Key	Value	Description
WebsiteURL	<a href="http://Lab-3-ElasticLoadB-1101UBDS896ZP-162983195.us-west-2.elb.amazonaws.com">http://Lab-3-ElasticLoadB-1101UBDS896ZP-162983195.us-west-2.elb.amazonaws.com</a>	Application URL

7. Connect to your new application URL, you should see the following:



8. Click on the “Resources” tab of the AWS CloudFormation stack to identify the AWS resources created by the template. You should see the following list of resources types:
  - IAM User + AccessKey
  - Amazon EC2 SecurityGroup
  - ELB LoadBalancer
  - Auto Scaling Group + Auto Scaling Launch Configuration
  - AWS CloudFormation WaitCondition (you can place a wait condition in a template to make AWS CloudFormation pause the creation of the stack and wait for a signal before it continues to create the stack)



## Availability and Impact Considerations

Different properties have different impacts on the resources in the stack. You can use AWS CloudFormation to update any property; however, before you make any changes, you should consider these questions:

1. How does the update affect the resource itself? For example, updating an alarm threshold will render the alarm inactive during the update. As we have seen, changing the instance type requires that the instance be stopped and restarted. AWS CloudFormation uses the Update or Modify actions for the underlying resources to make changes to resources. To understand the impact of updates, you should check the documentation for the specific resources.
2. Is the change mutable or immutable? Some changes to resource properties, such as changing the AMI on an Amazon EC2 instance, are not supported by the underlying services. In the case of mutable changes, AWS CloudFormation will use the Update or Modify type APIs for the underlying resources. For immutable property changes, AWS CloudFormation will create new resources with the updated properties and then link them to the stack before deleting the old resources. Although AWS CloudFormation tries to reduce the down time of the stack resources, replacing a resource is a multistep process, and it will take time. During stack reconfiguration, your application will not be fully operational. For example, it may not be able to serve requests or access a database.

## Conclusion

In this Lab you learned how to use AWS CloudFormation to provision and update a web application with a number of supporting AWS products and services, including Auto Scaling Groups, Amazon EC2 Instances, Elastic Load Balancers and more.

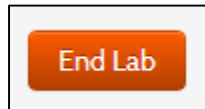
For more information about using AWS CloudFormation to start applications and on integrating with other configuration and deployment services such as Puppet and Opscode Chef, see the following whitepapers:

1. Bootstrapping Applications via AWS CloudFormation  
<https://s3.amazonaws.com/cloudformation-examples/BoostrappingApplicationsWithAWSCloudFormation.pdf>
2. Integrating AWS CloudFormation with Opscode Chef  
<https://s3.amazonaws.com/cloudformation-examples/IntegratingAWSCloudFormationWithOpscodeChef.pdf>
3. Integrating AWS CloudFormation with Puppet  
<https://s3.amazonaws.com/cloudformation-examples/IntegratingAWSCloudFormationWithPuppet.pdf>

The template used throughout this section is a "Hello, World" PHP application. The template library also has an Amazon ElastiCache sample template that shows how to integrate a PHP application with ElastiCache using `cfn-hup` and `cfn-init` to respond to changes in the Amazon ElastiCache Cache Cluster configuration, all of which can be performed by Update Stack.

## End Lab

1. Sign-out of the AWS Management Console.
2. Click the End Lab button in *qwikLAB*™.



3. Give the lab a thumbs-up/down, or enter a comment and click Submit

A feedback form interface. At the top, there are three icons: a thumbs-up, a thumbs-down, and a square box, all enclosed in a red oval. Below these icons is a text input field labeled "Comment". At the bottom right of the form, there is a "Submit" button, also enclosed in a red oval.

You can report any errors in these lab instructions to [aws-course-feedback@amazon.com](mailto:aws-course-feedback@amazon.com)