Launching and Managing a Web Application with AWS CloudFormation

**SPL-10 - Version 3.6.26**

© 2024 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. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab overview**

In this lab, you 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 Elastic Compute Cloud (EC2) instances, and Elastic Load Balancing.

In the first part you create a simple resource, an Amazon Simple Storage Service (Amazon S3) bucket, with AWS CloudFormation and you look at different retention policies applied when you delete an AWS CloudFormation stack or during a rollback.

In the second part, you provision a simple PHP web application using an Amazon Linux instance. You then see how to re-apply an AWS CloudFormation template to the existing application to change some resource attributes such as an Amazon EC2 instance type. Finally, you add a load balancer and an Auto Scaling group based on an Auto Scaling configuration.

OBJECTIVES

By the end of this lab, you should be able to do the following:

* Create an Amazon Simple Storage Service (S3) bucket using AWS CloudFormation.
* Provision a simple PHP web application using an Amazon Linux AMI.
* Apply an AWS CloudFormation template to an existing application.
* Modify an existing application using AWS CloudFormation.
* Add IAM roles and Elastic Load Balancing to the application using AWS CloudFormation.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you need to be comfortable editing scripts in a text editor.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
* **Copy edit:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **File contents:** A code block that displays the contents of a script or file you need to run that has been pre-created for you.
* **Learn more:** Where to find more information.
* **Note:** A hint, tip, or important guidance.
* **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
* **Task complete:** A conclusion or summary point in the lab.
* **Warning:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**Warning:** Do not change the **Region** unless instructed.

COMMON SIGN-IN ERRORS

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

SERVICES USED IN THIS LAB

**AWS CloudFormation**

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.

You can use AWS CloudFormation sample templates or create your own templates to describe the AWS resources, and any associated dependencies or runtime parameters, required to run your application. You don’t need to figure out the order for provisioning AWS services or the subtleties of making those dependencies work. AWS CloudFormation takes care of this for you. After the AWS resources are deployed, you can modify and update them in a controlled and predictable way, in effect applying version control to your AWS infrastructure the same way you do with your software.

You can deploy and update a template and its associated collection of resources (called a stack) by using the AWS Management Console, AWS Command Line Interface, or APIs. AWS CloudFormation is available at no additional charge, and you pay only for the AWS resources needed to run your applications.

An *AWS CloudFormation template* is a declaration of the AWS resources that make up a *stack*. The template is stored as a text file in either *JavaScript Object Notation (JSON)* or *YAML* format. 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.

In the templates for this lab, you use the *YAML* structure that AWS CloudFormation can interpret to declare the AWS resources you want to create and configure. In the YAML format, an object is declared as a name-value pair or a pairing of a name. In an AWS CloudFormation template you can declare the following six top-level objects:

* AWSTemplateFormatVersion
* Description
* Parameters
* Mappings
* Resources
* Outputs

The only required top-level object is the **Resources** object, which must declare at least one resource. This lab starts with the most basic template containing only a **Resources** object, which contains a single resource declaration.

**Learn more:** Definitions of each of these objects can be found online in AWS documentation. Refer to *Template sections* in the **Additional resources** section for more information.

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

**Task 1: Create an Amazon S3 bucket using AWS CloudFormation**

In this task, you start with a very simple AWS CloudFormation template to create a single Amazon S3 bucket with public read rights.

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 websites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

1. Examine this CloudFormation template.

**Example:**

AWSTemplateFormatVersion: 2010-09-09

Description: Basic S3 Bucket CloudFormation template

Resources:

S3BucketForWebsiteContent:

Type: AWS::S3::Bucket

Properties:

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

Outputs:

BucketName:

Value: !Ref S3BucketForWebsiteContent

Description: Name of the newly created Amazon S3 Distribution

The **Resources** object contains a list of resource objects.

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.

**Example:**

AWS::ProductIdentifier::ResourceType

For example, the resource type for an Amazon S3 bucket is:

AWS::S3::Bucket

1. Now take a look at a very basic template. The following template declares a single resource of type

AWS::S3::Bucket

 with the name **S3BucketForWebsiteContent**.

**Example:**

Resources:

S3BucketForWebsiteContent:

Type: AWS::S3::Bucket

The syntactic elements are quoted strings. If you use this template to create a stack, AWS CloudFormation creates 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 Amazon 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 you didn’t need any of the optional features, you could accept the defaults and omit the **Properties** attribute.

**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 for the **PublicAccessBlockConfiguration** property of the bucket.

**Example:**

Resources:

S3BucketForWebsiteContent:

Type: AWS::S3::Bucket

Properties:

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

**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.

**Example:**

Outputs:

BucketName:

Value: !Ref S3BucketForWebsiteContent

Description: Name of the newly created Amazon S3 Distribution

At the top of the template, **AWSTemplateFormatVersion** is simply the version of the template format; if you don’t specify it, AWS CloudFormation uses the latest version. The **Description** is any valid YAML string, and this description appears in the **Specify Parameters** page of the **Create Stack wizard**.

**Example:**

AWSTemplateFormatVersion: 2010-09-09

Description: Basic S3 Bucket CloudFormation template

TASK 1.1: CREATE AN AWS CLOUDFORMATION STACK

Use AWS CloudFormation to create a stack of AWS resources.

1. Download this template to your computer: [lab1.yaml](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/spl-10/v3.6.26.prod-9d4259dc/scripts/lab1.yaml) template.

**File contents:** The template contains the following code snippet.

AWSTemplateFormatVersion: 2010-09-09

Description: Basic S3 Bucket CloudFormation template

Resources:

S3BucketForWebsiteContent:

Type: AWS::S3::Bucket

Properties:

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

Outputs:

BucketName:

Value: !Ref S3BucketForWebsiteContent

Description: Name of the newly created Amazon S3 Distribution

1. At the top of the AWS Management Console, in the search bar, search for and choose

CloudFormation

.

**Note:** One or more stacks may already appear in the list. They were created for you when the lab was started. You create an additional stack.

1. Choose **Create stack** .
2. Choose **With new resources (standard)** .
3. On the **Create stack** page, make the following selections:

* In the **Prerequisite - Prepare template** section, choose **Choose an existing template** .
* In the **Specify template** section, choose **Upload a template file** and then choose **Choose file** .

1. Browse to and select the **lab1.yaml** template you just downloaded.
2. Choose **Next** .
3. On the **Specify stack details** page, locate the **Stack name** textbox and enter

Lab-1

.

1. Choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review and create** page, review the selections made and choose **Submit** .

The new stack status is  CREATE\_IN\_PROGRESS until the Amazon S3 bucket creation is complete.

1. **Refresh:** Choose the  **refresh** button every 30 seconds until the stack status changes to  CREATE\_COMPLETE.
2. Choose the **Outputs** tab to see the name of the new bucket.

You can now view the bucket in the **S3 Management Console**.

TASK 1.2: VIEW THE S3 BUCKET

1. At the top of the AWS Management Console, in the search bar, search for and choose

S3

.

You now see your newly created bucket, starting with

lab-1-s3bucket

. Notice also that AWS CloudFormation automatically created another bucket named *cf-templates-xxx* where it stores the templates you upload.

**Note:** It is normal for other buckets to exist in this list. They are provisioned along with other resources needed for this lab.

**Task complete:** You have successfully created an S3 bucket using CloudFormation. Then you verified the bucket is visible using the S3 Console.

**Task 2: Delete the stack and verify resources are deleted**

In this task, you delete the stack you created and observe how all resources created by this stack are deleted.

TASK 2.1: DELETE THE STACK USING CLOUDFORMATION

1. At the top of the AWS Management Console, in the search bar, search for and choose

CloudFormation

.

1. From the list of stacks, select  **Lab-1**.
2. Choose **Delete** .
3. Choose **Delete** .

**Note:** The stack status should now be  DELETE\_IN\_PROGRESS for a few minutes. The list updates, and you notice the stack is gone.

1. **Refresh:** Wait for the status the stack to disappear from the list (choose the  refresh button as necessary).

TASK 2.2: VERIFY THAT THE S3 BUCKET HAS BEEN DELETED

In this task, you verify the S3 bucket has been deleted.

1. At the top of the AWS Management Console, in the search bar, search for and choose

S3

.

1. Verify you no longer find the bucket starting with **lab-1-s3bucket**.

**Task complete:** You have deleted an S3 bucket that was originally created with CloudFormation using CloudFormation **Delete stack** action. You also verified in the S3 Console that the S3 bucket no longer exists.

**Task 3: Change the Retention Policy**

A common use-case is to store application assets in Amazon S3. They can be results of heavy data processing, files uploaded by users, or any other valuable data. Automatically deleting an Amazon S3 bucket when deleting an AWS CloudFormation stack is usually an unwanted behavior. You configure the AWS CloudFormation template and specify that it should not delete some of the resources when deleting the stack.

TASK 3.1: DOWNLOAD AND VIEW THE TEMPLATE

In this task, you download the updated CloudFormation template with the **DeletionPolicy** attribute.

1. Download the [lab2.yaml](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/spl-10/v3.6.26.prod-9d4259dc/scripts/lab2.yaml) template and examine it.

**File contents:** The template file has been updated and should now look like the code snippet below.

AWSTemplateFormatVersion: 2010-09-09

Description: Basic S3 Bucket CloudFormation template

Resources:

S3BucketForWebsiteContent:

Type: AWS::S3::Bucket

DeletionPolicy: Retain

Properties:

PublicAccessBlockConfiguration:

BlockPublicAcls: false

BlockPublicPolicy: false

IgnorePublicAcls: false

RestrictPublicBuckets: false

Outputs:

BucketName:

Value: !Ref S3BucketForWebsiteContent

Description: Name of the newly created Amazon S3 Distribution

The following line of code indicates that the resource should not be deleted once created, either in case of user-initiated stack deletion or in case of a rollback scenario.

**Example:**

DeletionPolicy: Retain

TASK 3.2: CREATE A NEW STACK USING THE TEMPLATE

In this task, you create a new stack from the modified template, then delete it, and verify that the newly created Amazon S3 bucket is still available.

1. At the top of the AWS Management Console, in the search bar, search for and choose

CloudFormation

.

1. Choose **Create stack** .
2. Choose **With new resources (standard)** .
3. On the **Create stack** page, make the following selections:

* In the **Prerequisite - Prepare template** section, choose **Choose an existing template** .
* In the **Specify template** section, choose **Upload a template file** and then choose **Choose file** .

1. Browse to and select the **lab2.yaml** template you just downloaded.
2. Choose **Next** .
3. On the **Specify stack details** page, locate the **Stack name** textbox and enter

Lab-2

.

1. Choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review and create** page, review the selections made and choose **Submit** .

The new stack status is  CREATE\_IN\_PROGRESS until the Amazon S3 bucket creation is complete.

1. **Refresh:** Choose the  **refresh** button every 30 seconds until the stack status changes to  CREATE\_COMPLETE.
2. Choose the **Outputs** tab.
3. Verify the new bucket is listed.

**Note:** The bucket value starts with **lab-2-s3bucket**.

TASK 3.3: VERIFY THE BUCKET HAS BEEN CREATED

In this task, use the **S3 Management Console** to verify the S3 bucket was created.

1. From the **CloudFormation Management Console**, for stack **Lab-2**, choose the **Resources** tab. Under the **Physical ID** column choose link to the S3 bucket with lab-2-s3bucket in the name.
2. You are redirected to a new browser tab opened to the **S3 Management Console**. This is confirmation that the new **lab-2-s3bucket** bucket was created.

TASK 3.4: DELETE THE STACK

In this task, you delete the stack using the **CloudFormation Management Console**.

1. Switch to the browser tab opened to the **CloudFormation Management Console**.
2. Select  **Lab-2**.
3. Choose **Delete** .
4. Choose **Delete** .
5. Wait for the stack deletion to be completed.

TASK 3.5: CHECK TO SEE IF THE S3 BUCKET EXISTS

In this task, you verify if the **S3 bucket** still exists or if it was deleted.

1. Switch back to the browser tab opened to the **S3 Management Console** which is opened to the S3 bucket.
2. Choose Buckets from the top-left navigation pane.
3. Verify that the bucket prefixed with lab-2-s3bucket has **not** been deleted.

**Task complete:** You have created a CloudFormation stack that creates an Amazon S3 bucket with a *DeletionPolicy* attribute. Then you deleted the stack to verify that the Amazon S3 bucket was not deleted due to the *DeletionPolicy* attribute.

**Consider:**

* In the first half of this lab, you learned the basics of **AWS CloudFormation**, the structure of the YAML file template, and how simple resources are created. You have learned about the **DeletionPolicy** attribute and how resources are deleted.
* In the second half of the lab, you review more advanced features of AWS CloudFormation including using **cfn-init**, receiving inputs using **parameters**, using **mappings**, and **updating** existing stacks.

**Task 4: Provision a Web Application**

This section of the lab 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. You walk through the following steps:

* **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.
* **Update the application:** Update one of the files in the application and deploy the software using AWS CloudFormation.
* **Update the instance type:** Change the instance type of the underlying Amazon EC2 instance.
* **Update IAM role:** Change the role assigned to the instance to another role providing additional permissions.
* **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.

**Learn more:** The steps in this lab are based on an example in the AWS CloudFormation documentation. Refer to *Walkthrough: Updating a stack* in the **Additional resources** section for more information.

**A Simple Application**

You begin by creating a stack that you can use throughout the rest of this section. The lab has provided a simple template that launches a single instance PHP web application hosted on the Apache Web Server 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**.

TASK 4.1: DOWNLOAD AND REVIEW THE NEW TEMPLATE

1. Download and examine the [lab3.yaml](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/spl-10/v3.6.26.prod-9d4259dc/scripts/lab3.yaml) template.

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 Amazon 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.

**Example:**

WebServerHost:

CreationPolicy:

ResourceSignal:

Timeout: PT5M

Type: AWS::EC2::Instance

Metadata:

Comment: Install a simple PHP application

AWS::CloudFormation::Init:

config:

...

files:

...

/etc/cfn/hooks.d/cfn-auto-reloader.conf:

content: !Sub |

[cfn-auto-reloader-hook]

triggers=post.update

path=Resources.WebServerHost.Metadata.AWS::CloudFormation::Init

action=/opt/aws/bin/cfn-init -s ${AWS::StackName} -r WebServerHost --region ${AWS::Region}

runas=root

mode: 000644

owner: apache

group: apache

/etc/cfn/cfn-hup.conf:

content: !Sub |

[main]

stack=${AWS::StackId}

region=${AWS::Region}

interval=1

mode: 000400

owner: root

group: root

services:

sysvinit:

httpd:

enabled: true

ensureRunning: true

cfn-hup:

enabled: true

ensureRunning: true

files:

- /etc/cfn/cfn-hup.conf

- /etc/cfn/hooks.d/cfn-auto-reloader.conf

Properties:

...

UserData:

...

To complete the stack, the template creates an Amazon EC2 security group and an elastic IP so that you have a consistent IP address to reference the application.

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

TASK 4.2: CREATE THE INITIAL STACK

In this task, you deploy the template you just downloaded using the **CloudFormation Management Console**.

1. Switch to the browser tab opened to the **AWS CloudFormation Console**.
2. Choose **Create stack** .
3. Choose **With new resources (standard)** .
4. On the **Create stack** page, make the following selections:

* In the **Prerequisite - Prepare template** section, choose **Choose an existing template** .
* In the **Specify template** section, choose **Upload a template file** and then choose **Choose file** .

1. Browse to and select the **lab3.yaml** template you just downloaded.
2. Choose **Next** .
3. On the **Specify stack details** page, make the following selections:

* In the **Stack name** section, locate the **Stack name** textbox and enter

Lab-3

.

* In the **Parameters** section, configure the following:
  + **ENV:** Select **dev** from the dropdown menu.
  + **VPCId:** Copy and paste the value of *VPCId* from the left of these instructions.
  + **VPCPublicSubnetId:** Copy and paste the value of *VPCPublicSubnetId* from the left of these instructions.

1. Choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review and create** page, review the selections made, scroll down to the **Capabilities** section and select the checkbox next to **I acknowledge that AWS CloudFormation might create IAM resources**.
3. Choose **Submit** .
4. **Refresh:** Choose the  **refresh** button every 30 seconds to see the updated events until the status of your stack displays  CREATE\_COMPLETE.

**Note:** The stack takes about five minutes to create. While you are waiting, examine the **Events** and **Resources** tabs that show the stack creation progress.

1. Choose the **Outputs** tab.

**Note:** The **Outputs** tab displays the **URL** of your website.

1. **Copy edit:** Copy and paste the **WebsiteURL** into a new browser tab.
2. Press **Enter** to navigate to the website.

**Note:** You are returned with a simple page with the following message:



*Image description: The preceding diagram depicts a simple web page with the message: AWS CloudFormation sample PHP application.*

1. Switch back to the browser tab opened to the **CloudFormation Management Console**.
2. Choose the **Resources** tab.
3. Under the **Physical ID** column choose the **instance ID link** with the **Logical ID** of

WebServerHost

.

1. This opens a new browser tab opened to the **EC2 Management Console** and filters the results based on the **instance ID**.
2. Notice that the **Instance type** is set to

t2.micro

.

**Task complete:** You have reviewed the latest CloudFormation template that deploys a web application, and verified the website is up and accessible. You also confirmed that the instance type is set to a **t2.micro**.

**Task 5: 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.

The stack you have built uses a **t2.micro** Amazon EC2 instance type. Pretend that your newly created website is getting more traffic than what a **t2.micro** instance can handle and you want to move to an **t2.small** Amazon EC2 instance type to handle the traffic.

Within the **mappings** section of the template, you can see that there is a mapping for the **EC2 instance type**. If you choose the **prod** environment, a **t2.small** instance is provisioned for you and if you choose the **dev** environment, a **t2.micro** instance is provisioned for you.

**Example:**

Mappings:

EC2TypeConfig:

prod:

InstanceType: t2.small

dev:

InstanceType: t2.micro

It is easy to change your instance type because **ENV** was an input parameter to your template. Therefore, you do not need to modify your template; you can simply change the value of the parameter in the **Stack Update wizard**, on the **Specify Parameters** page.

**Example:**

ENV:

Type: String

Default: dev

AllowedValues:

- dev

- prod

TASK 5.1: UPDATE A RESOURCE IN AN EXISTING CLOUDFORMATION STACK

1. Switch back to the browser tab opened to the **CloudFormation Management Console** and select  **Lab-3** from the list of stacks if not already selected.
2. Choose **Update** .
3. On the **Update stack** page, choose **Use existing template** .
4. Choose **Next** .
5. On the **Specify stack details** page, make the following selections:

* In the **Parameters** section, configure the following:
  + **ENV:** Select **prod** from the dropdown menu.

1. Keep default values for other parameters and choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review** page, review the selections made and view the list of the changes to be deployed in the **Change set preview** section.
3. Scroll down to the **Capabilities** section and select the checkbox next to **I acknowledge that AWS CloudFormation might create IAM resources**.
4. Choose **Submit** .

The instance type of an Amazon EC2 instance can be changed 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 also updates the Elastic IP address. You can see the changes in the AWS CloudFormation console on the Events tab. Here you have changed more than just the Instance Type, hence the first instance is terminated and a new one is launched.

1. **Refresh:** Choose the  **refresh** button every 30 seconds to see the updated events until the stack status displays  UPDATE\_COMPLETE.
2. At the top of the AWS Management Console, in the search bar, search for and choose

EC2

.

1. In the left navigation pane, choose **Instances**.
2. Confirm that the instance type has changed from **t2.micro** to **t2.small**.

**Task complete:** You have successfully updated an existing CloudFormation stack and changed the EC2 instance type for the **Web Application server** from **t2.micro** to **t2.small** by changing the value of stack parameter **ENV** from **dev** to **prod**.

**Task 6: Add Tags to an Existing Resource**

So far, you’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 this, you update your EC2 instance so that it is tagged with a name and you also open port 22.

TASK 6.1: DOWNLOAD AND REVIEW A NEW CLOUDFORMATION TEMPLATE

1. Download and examine the [lab4.yaml](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/spl-10/v3.6.26.prod-9d4259dc/scripts/lab4.yaml) template.

**File contents:** The code below adds a **name tag** to your EC2 instance.

Properties:

Tags:

- Key: Name

Value: Web server - port 80 and 22

**File contents:** The following code adds port **22** to the **ingress rules** for the Amazon EC2 security group.

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable HTTP

VpcId: !Ref VPCId

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

- IpProtocol: tcp

FromPort: 22

ToPort: 22

CidrIp: 0.0.0.0/0

TASK 6.2: ADD TAGS TO AN EC2 INSTANCE

In this task, you update the existing stack with an updated template that adds a **tag** with a specific **key** and **value** to an EC2 instance.

1. At the top of the AWS Management Console, in the search bar, search for and choose

CloudFormation

.

1. Select  **Lab-3** from the list of stacks, if not selected already.
2. Choose **Update** .
3. On the **Update stack** page, choose **Replace existing template** .
4. In the **Specify template** section, choose **Upload a template file** and then choose **Choose file** .
5. Browse to and select the **lab4.yaml** template you just downloaded.
6. Choose **Next** .
7. On the **Specify stack details** page, choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review** page, review the selections made and view the list of the changes to be deployed in the **Change set preview** section.

The **Change set preview** section shows how resources are be updated.

1. Scroll down to the **Capabilities** section and select the checkbox next to **I acknowledge that AWS CloudFormation might create IAM resources**.
2. Choose **Submit** .
3. **Refresh:** Choose the  **refresh** button every 30 seconds to see the updated events until the stack status displays  UPDATE\_COMPLETE.
4. Choose the **Outputs** tab.

**Note:** The **Outputs** tab displays the URL of your website.

1. **Copy edit:** Copy and paste the **WebsiteURL** into a new browser tab.
2. Press **Enter** to navigate to the website.

**Note:** A web page returns with the following message showing that the stack has been successfully updated:



*Image description: The preceding diagram depicts a simple web page with the message: AWS CloudFormation sample PHP application Updated version via UpdateStack.*

1. Switch back to the browser tab opened to the **CloudFormation Management Console**.
2. Choose the **Resources** tab.
3. Choose **instance ID link** under the **Physical ID** column for the instance with a **Logical ID** of **WebServerHost**.
4. This opens the **EC2 Management Console** and filter the results based on the instance ID. Verify you now see the **Name** tag with a value set to **Web server - port 80 and 22**.

**Task complete:** You have successfully updated your CloudFormation stack with an updated template that adds a tag to your EC2 instance and updates your web application.

**Task 7: Create an Application with Auto Scaling**

Next, you change the IAM role that is assigned to the Amazon EC2 instance, which eventually helps the code running on it to make **AWS API** calls. Suppose that a new version of the application requires access to the **Amazon EC2 read-only APIs** from the instance.

TASK 7.1: DOWNLOAD AND REVIEW AN UPDATED TEMPLATE

1. Download and examine the updated template: [lab5.yaml](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/spl-10/v3.6.26.prod-9d4259dc/scripts/lab5.yaml) template.

This code is providing the name of another role that was created as part of the lab setup. The role **ec2-role-2** was created as part of the lab setup and provides permissions to call **EC2 read-only APIs**.

**Example:**

WebServerInstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles:

- ec2-role-2

This change updates the role assigned to the instance; it does not require other changes in the stack. When the stack is updated, the Amazon EC2 instance has access to the Amazon EC2 read-only APIs.

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

The template removes the **Elastic IP address** resource from the template.

**Before:**

Endpoint:

Type: AWS::EC2::EIP

Properties:

InstanceId: !Ref WebServerHost

The template adds an **Elastic Load Balancer** resource.

**After:**

ElasticLoadBalancer:

Type: AWS::ElasticLoadBalancing::LoadBalancer

Properties:

Subnets:

- !Ref VPCPublicSubnetId

SecurityGroups:

- !Ref WebServerSecurityGroup

Listeners:

- LoadBalancerPort: 80

InstancePort: 80

Protocol: HTTP

HealthCheck:

Target: HTTP:80/

HealthyThreshold: 3

UnhealthyThreshold: 5

Interval: 30

Timeout: 5

The template also converts the Amazon EC2 instance in the template into an **Auto Scaling Launch Configuration**. The properties are identical, it only needs to change the type name.

**Before:**

WebServerHost:

Type: AWS::EC2::Instance

**After:**

WebServerConfig:

Type: AWS::AutoScaling::LaunchConfiguration

For clarity in the template, the name of the resource has also been changed from **WebServerHost** to **WebServerConfig**.

The template includes an **Auto Scaling group resource**.

**Example:**

WebServerGroup:

Type: AWS::AutoScaling::AutoScalingGroup

Properties:

AvailabilityZones:

- !Ref VPCPublicSubnetAZ

LaunchConfigurationName: !Ref WebServerConfig

MinSize: 2

MaxSize: 3

LoadBalancerNames:

- !Ref ElasticLoadBalancer

VPCZoneIdentifier:

- !Ref VPCPublicSubnetId

The **security group** definition allows traffic to all IPs. However, in a production environment, it is recommended that you restrict the traffic to the **ELB security group**.

**Example:**

WebServerSecurityGroup:

Type: AWS::EC2::SecurityGroup

Properties:

GroupDescription: Enable HTTP

VpcId: !Ref VPCId

SecurityGroupIngress:

- IpProtocol: tcp

FromPort: 80

ToPort: 80

CidrIp: 0.0.0.0/0

Finally, the **Output** section now returns the **DNS Name** of the **load balancer** as the location of the application.

**Example:**

Outputs:

WebsiteURL:

Value: !Sub 'http://${ElasticLoadBalancer.DNSName}'

Description: Application URL

When you use this template to update the stack, you 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 are altered, so if there were 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.

TASK 7.2: UPDATE THE STACK WITH AN APPLICATION USING AWS AUTO SCALING

1. Switch to the browser tab already opened to the **CloudFormation Management Console**.
2. Select  **Lab-3** from the list of stacks, if not selected already.
3. Choose **Update** .
4. On the **Update stack** page, choose **Replace existing template** .
5. In the **Specify template** section, choose **Upload a template file** and then choose **Choose file** .
6. Browse to and select the **lab5.yaml** template you just downloaded.
7. Choose **Next** .
8. On the **Specify stack details** page, configure the following parameter:

* **VPCPublicSubnetAZ:** Copy and paste the value of *VPCPublicSubnetAZ* from the left of these instructions.

1. Choose **Next** .

The **Configure stack options** page allows you to specify **Tags**, **Permissions**, **Stack failure options**, and **Advanced options**.

1. Under **Stack failure options** section, choose  **Preserve successfully provisioned resources**.

**Note:** This allows the successfully created resource to remain and rolls back the unsuccessful resource. You can use this to troubleshoot resources like EC2 instances and so on.

1. Keeping default values for all other options, navigate to the bottom of the screen and choose **Next** .
2. On the **Review** page, review the selections made and view the list of the changes to be deployed in the **Change set preview** section.
3. You can see the changes to be implemented in the **Change set preview** section. Some resources are **added**, **removed**, and **modified**.
4. Scroll down to the **Capabilities** section and select the checkbox next to **I acknowledge that AWS CloudFormation might create IAM resources**.
5. Navigate to the bottom of the page and choose **Submit** .
6. **Refresh:** Choose the  **refresh** button every 30 seconds to see the updated events until the stack status displays  UPDATE\_COMPLETE.
7. Wait **three minutes** after the AWS CloudFormation update is finished.

**Note:** This gives time for **Auto Scaling/Elastic Load Balancing** to launch and configure all resources.

1. Choose the **Outputs** tab.
2. **Copy edit:** Copy the new **WebSiteURL** into a new browser tab.

This is **not** an **IP address** anymore, but instead is the **DNS name** of the **load balancer** of your newly high-availability application.

1. Press **Enter** to navigate to the website.



*Image description: The preceding diagram depicts a simple web page with the message: AWS CloudFormation sample PHP application Highly available, load balanced stack updated version via UpdateStack.*

1. Switch to the browser tab already opened to the **CloudFormation Management Console**.
2. Choose the **Resources** tab of the **Lab3** stack.

The following list of resources types were created:

* IAM Instance Profile
* EC2 SecurityGroup
* ELB LoadBalancer
* AutoScaling Group + AutoScaling Launch Configuration

**Task complete:** You have updated your stack with a CloudFormation template that updates a single-instance web application to a web application using AWS Auto Scaling.

**Availability and impact considerations**

**Consider:** 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, consider these questions:

* **How does the update affect the resource itself?** For example, updating an alarm threshold renders the alarm inactive during the update. As you 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, check the documentation for the specific resources.
* **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 uses the Update or Modify type APIs for the underlying resources. For immutable property changes, AWS CloudFormation creates 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 downtime of the stack resources, replacing a resource is a multi-step process, and takes time. During stack reconfiguration, your application is not fully operational. For example, it may not be able to serve requests or access a database.

**Learn 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:

* [Bootstrapping Applications via AWS CloudFormation](https://s3.amazonaws.com/cloudformation-examples/BoostrappingApplicationsWithAWSCloudFormation.pdf)
* [Using Chef with AWS CloudFormation](https://s3.amazonaws.com/cloudformation-examples/IntegratingAWSCloudFormationWithOpscodeChef.pdf)
* [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 the **Update Stack** action.

**Conclusion**

You have successfully done the following:

* Created an Amazon Simple Storage Service (S3) bucket using AWS CloudFormation.
* Provisioned a simple PHP web application using an Amazon Linux AMI.
* Applied an AWS CloudFormation template to an existing application.
* Modified an existing application using AWS CloudFormation.
* Added IAM roles and Elastic Load Balancing to the application using AWS CloudFormation.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional resources**

* [Template sections](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/concept-template.html#concept-template-description)
* [Walkthrough: Updating a stack](http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/updating.stacks.walkthrough.html)
* [AWS CloudFormation](http://aws.amazon.com/cloudformation/)
* [AWS Training and Certification](http://aws.amazon.com/training/)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).