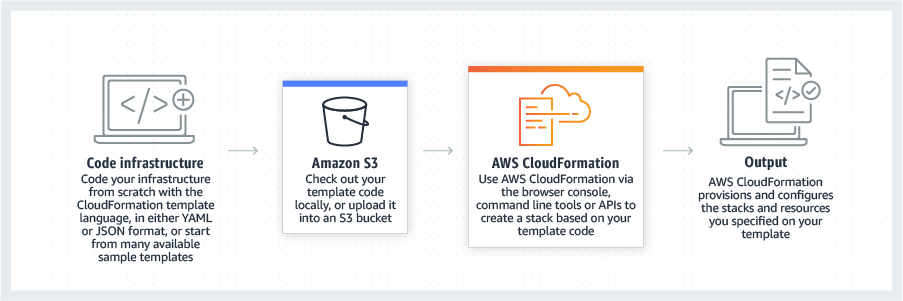
CLOUD FORMATIONS

<https://docs.aws.amazon.com/cloudformation/index.html>

<https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/Welcome.html>



* CloudFormation allows you to model your infrastructure in a text file. You can use [JSON](https://json.org/) or [YAML](https://yaml.org/) to describe what AWS resources you want to create and configure.
* CloudFormation automates the provisioning and updating of your infrastructure in a safe and controlled manner. There are no manual steps or controls that can lead to errors.
* CloudFormation is available at no additional charge. You pay only for the AWS resources needed to run your applications.

Stack

A stack is a deployment of a CloudFormation template. You can create multiple stacks from a single CloudFormation template. A stack contains a collection of AWS resources that you can manage as a single unit. All the resources in a stack are defined by the stack’s AWS CloudFormation template.

AWS CloudFormation will create, update or delete a stack in its entirety:

* If a stack cannot be created or updated in its entirety, AWS CloudFormation will roll it back, and automatically delete any resources that were created.
* If a resource cannot be deleted, any remaining resources are retained until the stack can be successfully deleted.



**Estructura template**

(<https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/template-anatomy.html>)

#### **Format Version**

The AWSTemplateFormatVersion section identifies the capabilities of the template. The latest template format version is 2010-09-09 and is currently the only valid value.

AWSTemplateFormatVersion: '2010-09-09'

#### **Description**

The Description section enables you to include comments about your template.

Description : CFN 101 Workshop - Lab 02 Resources.

#### **Metadata**

You can use the [Metadata section](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/metadata-section-structure.html) to include arbitrary JSON or YAML objects. This section is useful for providing information to other tools that interact with your CloudFormation template. For example, when deploying CloudFormation templates via the AWS console, you can improve the experience of users deploying your templates by specify how to order, label and group parameters. This can be done with the [AWS::CloudFormation::Interface](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cloudformation-interface.html) key.

Metadata:

AWS::CloudFormation::Interface:

ParameterGroups:

- Label:

default: 'Amazon EC2 Configuration'

Parameters:

- InstanceType

ParameterLabels:

InstanceType:

default: 'Type of EC2 Instance'

#### **Parameters**

Parameters enable you to input custom values to your template each time you create or update a stack.

AWS CloudFormation supports the following parameter types:

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| String | A literal string. | “MyUserName” |
| Number | An integer or float. | “123” |
| List<Number> | An array of integers or floats. | “10,20,30” |
| CommaDelimitedList | An array of literal strings. | “test,dev,prod” |
| [AWS-Specific Parameter Types](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html#aws-specific-parameter-types) | AWS values such as Amazon VPC IDs. | AWS::EC2::VPC::Id |
| [SSM Parameter Types](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html#aws-ssm-parameter-types) | Parameters that correspond to existing parameters in Systems Manager Parameter Store. | AWS::SSM::Parameter::Value<AWS::EC2::Image::Id> |

Parameters:

InstanceType:

Type: String

Default: t2.micro

AllowedValues:

- t2.micro

- t2.small

Description: 'Enter t2.micro or t2.small. Default is t2.micro.'

#### **Resources**

The required Resources section declares the AWS resources that you want to include in the stack. Let’s add the EC2 resource to your stack.

Resources:

WebServerInstance:

Type: 'AWS::EC2::Instance'

Properties:

InstanceType: !Ref InstanceType

ImageId: <replace with AMI ID ami-xxxxx>

The only required property of the EC2 resource type is ImageId. Let’s find the AMI ID via AWS console:

1. Open [**AWS EC2 console**](https://console.aws.amazon.com/ec2)
2. Click **Instances** -> **Launch Instance**.
3. Copy the **Amazon Linux 2 AMI** ami-xxxxxxxxx ID.

# **LAB 01: RESOURCES**

(<https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>)

In this Lab, you will learn little bit more about CloudFormation Resources.

# **LAB 02: PArameters - INTRINSIC FUNCTIONS**

# **(**<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html>**)**

parámetros: Valores que se van a pasar a su plantilla en tiempo de ejecución (al crear o actualizar una pila). Puede consultar los parámetros en las secciones Resources y Outputs de la plantilla.

Intrinsic functions are built-in functions that help you manage your stacks. Without them, you will be limited to very basic templates.

* Use the **[Fn::Ref](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-ref.html)** function to dynamically assign parameter values to a resource property.
* Tag an instance with **[Fn::Join](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-join.html)** function.
* Add a tag to the instance using **[Fn::Sub](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-sub.html)** function.

Note:

Intrinsic functions can only be used in certain parts of a template. You can use intrinsic functions in **resource properties, outputs, metadata attributes, and update policy atributes**

#### **Fn::Ref**

The intrinsic function Ref returns the value of the specified parameter or resource.

In the last lab you have “hard coded” an AMI ID directly into the EC2 Resource property. You will now amend this to make your template more flexible. Let’s convert AmiID to variable and pass it to resource property at the runtime.

#### !Ref Parametro

#### **Fn::Join**

The intrinsic function Fn::Join appends a set of values into a single value, separated by the specified delimiter. If a delimiter is the empty string, the set of values are concatenated with no delimiter.

#### [**Fn::Sub**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-sub.html)

The intrinsic function Fn::Sub substitutes variables in an input string with values that you specify. In your templates, you can use this function to construct commands or outputs that include values that aren't available until you create or update a stack

To help you manage your AWS resources, you can optionally assign your own metadata to each resource in the form of **tags**. Each tag is a simple label consisting of a customer-defined key and an optional value that can help you to categorize resources by purpose, owner, environment, or other criteria. Let’s use the intrinsic function **Fn::Join** to name your instance.

Tags:

        - Key: Name

          Value: !Join [ '-', [ !Ref pInstanceType, webserver ] ]

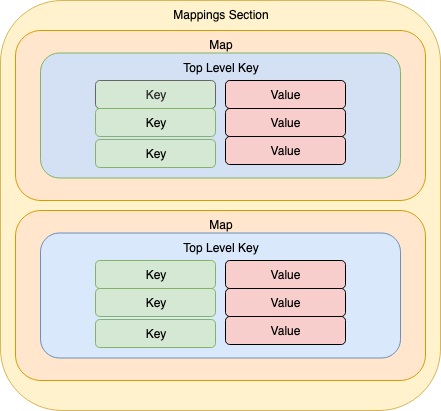
        - Key: InstanceType

          Value: !Sub ${pInstanceType}

# **LAB 03: MAPPING**

A Mappings section is a top level section of a CloudFormation template. It is used to define maps, their keys and values which can be then referenced in your template.

Un mapeo de claves y valores asociados que puede utilizar para especificar valores de parámetros condicionales, similar a una tabla de búsqueda. Puede asociar una clave a un valor correspondiente mediante la función intrínseca [Fn::FindInMap](https://docs.aws.amazon.com/es_es/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-findinmap.html) en las secciones Resources y Outputs



Mappings:

Mapping01:

Key01:

Name: Value01

Key02:

Name: Value02

Key03:

Name: Value03

!FindInMap [ *MapName*, *TopLevelKey*, *SecondLevelKey* ]

## **Parámetros**

MapName

El nombre lógico de un mapeo declarado en la sección Mappings que contiene las claves y valores.

TopLevelKey

El nombre de clave de nivel superior. Su valor es una lista de pares clave-valor.

SecondLevelKey

El nombre de la clave de segundo nivel, que se ha establecido en una de las claves de la lista asignada a TopLevelKey.

## **Valor devuelto:**

El valor que se ha asignado a SecondLevelKey.

# **LAB 04: OUTPUTS**

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html>

*Outputs* enable you to get access to information about resources within a stack. For example, you can output an EC2 instance’s Public DNS name once it is created.

Furthermore, output values can be imported into other stacks. These are known as cross-stack references.

Outputs:

Logical ID:

Description: Information about the value

Value: Value to return

Export:

Name: Value to export

1. Create an AWS::EC2::EIP resource and attach it to your existing EC2 instance.
2. Create a logical ID called ElasticIP and add it to the Outputs section of the template.

The Fn::GetAtt intrinsic function returns the value of an attribute from a resource in the template. For more information about GetAtt return values for a particular resource, refer to the documentation for that resource in the [Resource and property reference](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html).

!GetAtt *logicalNameOfResource*.*attributeName*

To get the PublicDnsName of the instance, you will need to use Fn::GetAtt intrinsic function. Let’s first check the [AWS Documentation](https://docs.aws.amazon.com/en_pv/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#aws-properties-ec2-instance-return-values) for available attributes. You can see that PublicDnsName is valid return value for Fn::GetAtt function