# CloudFormation Template Anatomy

A template is a JSON- or YAML-formatted text file that describes your AWS infrastructure. The following examples show an AWS CloudFormation template structure and its sections.

## JSON

The following example shows a JSON-formatted template fragment.

{

"AWSTemplateFormatVersion" : "*version date*",

"Description" : "*JSON string*",

"Metadata" : {

*template metadata*

},

"Parameters" : {

*set of parameters*

},

"Mappings" : {

*set of mappings*

},

"Conditions" : {

*set of conditions*

},

"Transform" : {

*set of transforms*

},

"Resources" : {

*set of resources*

},

"Outputs" : {

*set of outputs*

}

}

## YAML

The following example shows a YAML-formatted template fragment.

---

AWSTemplateFormatVersion: "*version date*"

Description:

*String*

Metadata:

*template metadata*

Parameters:

*set of parameters*

Mappings:

*set of mappings*

Conditions:

*set of conditions*

Transform:

*set of transforms*

Resources:

*set of resources*

Outputs:

*set of outputs*

## Template Sections

Templates include several major sections. The Resources section is the only required section. Some sections in a template can be in any order. However, as you build your template, it might be helpful to use the logical ordering of the following list, as values in one section might refer to values from a previous section. The list gives a brief overview of each section.

[**Format Version (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/format-version-structure.html)

The AWS CloudFormation template version that the template conforms to. The template format version is not the same as the API or WSDL version. The template format version can change independently of the API and WSDL versions.

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

**Note**

The template format version is not the same as the API or WSDL version. The template format version can change independently of the API and WSDL versions.

The value for the template format version declaration must be a literal string. You cannot use a parameter or function to specify the template format version. If you don't specify a value, AWS CloudFormation assumes the latest template format version. The following snippet is an example of a valid template format version declaration:

## JSON

"AWSTemplateFormatVersion" : "2010-09-09"

## YAML

AWSTemplateFormatVersion: "2010-09-09"

[**Description (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-description-structure.html)

A text string that describes the template. This section must always follow the template format version section.

The Description section (optional) enables you to include arbitrary comments about your template. The Descriptionmust follow the AWSTemplateFormatVersion section.

The value for the description declaration must be a literal string that is between 0 and 1024 bytes in length. You cannot use a parameter or function to specify the description. The following snippet is an example of a description declaration:

## JSON

"Description" : "Here are some details about the template."

## YAML

Description: >

Here are some

details about

the template.

[**Metadata (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/metadata-section-structure.html)

Objects that provide additional information about the template.

You can use the optional Metadata section to include arbitrary JSON or YAML objects that provide details about the template. For example, you can include template implementation details about specific resources, as shown in the following snippet:

**Important**

During a stack update, you cannot update the Metadata section by itself. You can update it only when you include changes that add, modify, or delete resources.

## JSON

"Metadata" : {

"Instances" : {"Description" : "Information about the instances"},

"Databases" : {"Description" : "Information about the databases"}

}

## YAML

Metadata:

Instances:

Description: "Information about the instances"

Databases:

Description: "Information about the databases"

## Metadata Keys

Some AWS CloudFormation features retrieve settings or configuration information that you define from the Metadatasection. You define this information in the following AWS CloudFormation-specific metadata keys:

AWS::CloudFormation::Init

Defines configuration tasks for the cfn-init helper script. This script is useful for configuring and installing applications on EC2 instances. For more information, see [AWS::CloudFormation::Init](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-init.html).

AWS::CloudFormation::Interface

Defines the grouping and ordering of input parameters when they are displayed in the AWS CloudFormation console. By default, the AWS CloudFormation console alphabetically sorts parameters by their logical ID. For more information, see [AWS::CloudFormation::Interface](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cloudformation-interface.html).

AWS::CloudFormation::Designer

Describes how your resources are laid out in AWS CloudFormation Designer (Designer). Designer automatically adds this information when you use it create and update templates. For more information, see [What Is AWS CloudFormation Designer?](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/working-with-templates-cfn-designer.html).

[**Parameters (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html)

Values to pass to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Use the optional Parameters section to customize your templates. Parameters enable you to input custom values to your template each time you create or update a stack.

## Defining a Parameter in a Template

The following example declares a parameter named InstanceTypeParameter. This parameter lets you specify the Amazon EC2 instance type for the stack to use when you create or update the stack.

Note that InstanceTypeParameter has a default value of t2.micro. This is the value that AWS CloudFormation uses to provision the stack unless another value is provided.

### JSON

"Parameters" : {

"InstanceTypeParameter" : {

"Type" : "String",

"Default" : "t2.micro",

"AllowedValues" : ["t2.micro", "m1.small", "m1.large"],

"Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro."

}

}

### YAML

Parameters:

InstanceTypeParameter:

Type: String

Default: t2.micro

AllowedValues:

- t2.micro

- m1.small

- m1.large

Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.

## Referencing a Parameter within a Template

You use the Ref intrinsic function to reference a parameter, and AWS CloudFormation uses the parameter's value to provision the stack. You can reference parameters from the Resources and Outputs sections of the same template.

In the following example, the InstanceType property of the EC2 instance resource references the InstanceTypeParameter parameter value:

### JSON

"Ec2Instance" : {

"Type" : "AWS::EC2::Instance",

"Properties" : {

"InstanceType" : { "Ref" : "InstanceTypeParameter" },

"ImageId" : "ami-2f726546"

}

}

### YAML

Ec2Instance:

Type: AWS::EC2::Instance

Properties:

InstanceType:

Ref: InstanceTypeParameter

ImageId: ami-2f726546

## General Requirements for Parameters

The following requirements apply when using parameters:

* You can have a maximum of 60 parameters in an AWS CloudFormation template.
* Each parameter must be given a logical name (also called logical ID), which must be alphanumeric and unique among all logical names within the template.
* Each parameter must be assigned a parameter type that is supported by AWS CloudFormation. For more information, see [Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html#parameters-section-structure-properties-type).
* Each parameter must be assigned a value at runtime for AWS CloudFormation to successfully provision the stack. You can optionally specify a default value for AWS CloudFormation to use unless another value is provided.
* Parameters must be declared and referenced from within the same template. You can reference parameters from the Resources and Outputs sections of the template.

[**Mappings (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html)

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the [Fn::FindInMap](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-findinmap.html) intrinsic function in the Resources and Outputs section.

The optional Mappings section matches a key to a corresponding set of named values. For example, if you want to set values based on a region, you can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region. You use the Fn::FindInMap intrinsic function to retrieve values in a map.

You cannot include parameters, pseudo parameters, or intrinsic functions in the Mappings section.

## Syntax

The Mappings section consists of the key name Mappings. The keys in mappings must be literal strings. The values can be String or List types. The following example shows a Mappings section containing a single mapping named Mapping01 (the logical name).

Within a mapping, each map is a key followed by another mapping. The key identifies a map of name-value pairs and must be unique within the mapping. The name can contain only alphanumeric characters (A-Za-z0-9).

### JSON

"Mappings" : {

"Mapping01" : {

"Key01" : {

"Name" : "Value01"

},

"Key02" : {

"Name" : "Value02"

},

"Key03" : {

"Name" : "Value03"

}

}

}

### YAML

Mappings:

Mapping01:

Key01:

Name: Value01

Key02:

Name: Value02

Key03:

Name: Value03

[**Conditions (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html)

Conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

The optional Conditions section includes statements that define when a resource is created or when a property is defined. For example, you can compare whether a value is equal to another value. Based on the result of that condition, you can conditionally create resources. If you have multiple conditions, separate them with commas.

You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment. In your template, you can add an EnvironmentType input parameter, which accepts either **prod** or **test** as inputs. For the production environment, you might include Amazon EC2 instances with certain capabilities; however, for the test environment, you want to use reduced capabilities to save money. With conditions, you can define which resources are created and how they're configured for each environment type.

Conditions are evaluated based on input parameter values that you specify when you create or update a stack. Within each condition, you can reference another condition, a parameter value, or a mapping. After you define all your conditions, you can associate them with resources and resource properties in the Resources and Outputs sections of a template.

At stack creation or stack update, AWS CloudFormation evaluates all the conditions in your template before creating any resources. Any resources that are associated with a true condition are created. Any resources that are associated with a false condition are ignored.

**Important**

During a stack update, you cannot update conditions by themselves. You can update conditions only when you include changes that add, modify, or delete resources.

## How to Use Conditions Overview

To conditionally create resources, you must include statements in at least three different sections of a template:

Parameters section

Define the input values that you want to evaluate in your conditions. Conditions will result in true or false based on values from these input parameters.

Conditions section

Define conditions by using the intrinsic condition functions. These conditions determine when AWS CloudFormation creates the associated resources.

Resources and Outputs sections

Associate conditions with the resources or outputs that you want to conditionally create. AWS CloudFormation creates entities that are associated with a true condition and ignores entities that are associated with a false condition. Use the Condition key and a condition's logical ID to associate it with a resource or output. To conditionally specify a property, use the Fn::If function. For more information, see [Condition Functions](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-conditions.html).

## Syntax

The Conditions section consists of the key name Conditions. Each condition declaration includes a logical ID and intrinsic functions that are evaluated when you create or update a stack. The following pseudo template outlines the Conditions section:

### JSON

"Conditions" : {

"*Logical ID*" : {*Intrinsic function*}

}

### YAML

Conditions:

*Logical ID*:

*Intrinsic function*

#### Condition Intrinsic Functions

You can use the following intrinsic functions to define conditions:

* Fn::And
* Fn::Equals
* Fn::If
* Fn::Not
* Fn::Or

For the syntax and information about each function, see [Condition Functions](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-conditions.html).

**Note**

Fn::If is only supported in the metadata attribute, update policy attribute, and property values in the Resources section and Outputs sections of a template.

[**Transform (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-section-structure.html)

For [serverless applications](http://docs.aws.amazon.com/lambda/latest/dg/deploying-lambda-apps.html" \t "_blank) (also referred to as Lambda-based applications), specifies the version of the [AWS Serverless Application Model (AWS SAM)](https://github.com/awslabs/serverless-application-specification) to use. When you specify a transform, you can use AWS SAM syntax to declare resources in your template. The model defines the syntax that you can use and how it is processed.

You can also use AWS::Include transforms to work with template snippets that are stored separately from the main AWS CloudFormation template. You can store your snippet files in an Amazon S3 bucket and then reuse the functions across multiple templates.

The optional Transform section specifies one or more transforms that AWS CloudFormation uses to process your template. The Transform section builds on the simple, declarative language of AWS CloudFormation with a powerful macro system.

AWS CloudFormation transforms help simplify template authoring by condensing the expression of AWS infrastructure as code and enabling reuse of template components. For example, you can condense a multiple-line resource declaration into a single line in your template.

AWS CloudFormation supports AWS::Serverless and AWS::Include transform types:

* An AWS::Serverless transform specifies the version of the AWS Serverless Application Model (AWS SAM) to use. This model defines the AWS SAM syntax that you can use and how AWS CloudFormation processes it. When you create a change set, AWS CloudFormation resolves all Transform functions. For more information about serverless applications and AWS SAM, see [Deploying Lambda-based Applications](http://docs.aws.amazon.com/lambda/latest/dg/deploying-lambda-apps.html) in the AWS Lambda Developer Guide.
* An AWS::Include transform works with template snippets that are stored separately from the main AWS CloudFormation template. You can insert these snippets into your main template when [Creating a Change Set](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-changesets-create.html) or [Updating Stacks Using Change Sets](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-changesets.html).

You can declare a single transform or multiple transforms within a template. AWS CloudFormation executes transformations in the order that they are specified.

To declare multiple transforms, use a list format and specify one or more AWS::Include transforms and (optionally) an AWS::Serverless transform. The following example declares two AWS::Include transforms.

## JSON

{

"Resources": {

"MyBucket": {

"Type": "AWS::S3::Bucket",

"Properties": {

"Fn::Transform": [

{

"Name": "AWS::Include",

"Parameters": {

"Location": "s3://bucket/myBucketName.json"

}

},

{

"Name": "AWS::Include",

"Parameters": {

"Location": "s3://bucket/myBucketAcl.json"

}

}

]

}

}

}

}

## YAML

Resources:

MyBucket:

Type: 'AWS::S3::Bucket'

Properties:

'Fn::Transform':

- Name: 'AWS::Include'

Parameters:

Location: s3://bucket/myBucketName.yaml

- Name: 'AWS::Include'

Parameters:

Location: s3://bucket/myBucketAcl.yaml

[**Resources (required)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resources-section-structure.html)

Specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

The required Resources section declares the AWS resources that you want to include in the stack, such as an Amazon EC2 instance or an Amazon S3 bucket.

## Syntax

The Resources section consists of the key name Resources. The following pseudo template outlines the Resourcessection:

### JSON

"Resources" : {

"*Logical ID*" : {

"Type" : "*Resource type*",

"Properties" : {

*Set of properties*

}

}

}

### YAML

Resources:

*Logical ID*:

Type: *Resource type*

Properties:

*Set of properties*

### Resource Fields

**Logical ID**

The logical ID must be alphanumeric (A-Za-z0-9) and unique within the template. Use the logical name to reference the resource in other parts of the template. For example, if you want to map an Amazon Elastic Block Store volume to an Amazon EC2 instance, you reference the logical IDs to associate the block stores with the instance.

In addition to the logical ID, certain resources also have a physical ID, which is the actual assigned name for that resource, such as an EC2 instance ID or an S3 bucket name. Use the physical IDs to identify resources outside of AWS CloudFormation templates, but only after the resources have been created. For example, you might give an EC2 instance resource a logical ID of MyEC2Instance; but when AWS CloudFormation creates the instance, AWS CloudFormation automatically generates and assigns a physical ID (such as i-28f9ba55) to the instance. You can use this physical ID to identify the instance and view its properties (such as the DNS name) by using the Amazon EC2 console. For resources that support custom names, you can assign your own names (physical IDs) to help you quickly identify resources. For example, you can name an S3 bucket that stores logs as MyPerformanceLogs. For more information, see [Name Type](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-name.html).

**Resource type**

The resource type identifies the type of resource that you are declaring. For example, AWS::EC2::Instancedeclares an EC2 instance. For a list of all of the resource types, see [AWS Resource Types Reference](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html).

**Resource properties**

Resource properties are additional options that you can specify for a resource. For example, for each EC2 instance, you must specify an Amazon Machine Image (AMI) ID for that instance. You declare the AMI ID as a property of the instance, as shown in the following example:

**Example JSON**

"Resources" : {

"MyEC2Instance" : {

"Type" : "AWS::EC2::Instance",

"Properties" : {

"ImageId" : "ami-2f726546"

}

}

}

**Example YAML**

Resources:

MyEC2Instance:

Type: "AWS::EC2::Instance"

Properties:

ImageId: "ami-2f726546"

If a resource doesn't require that properties be declared, omit the properties section of that resource.

Property values can be literal strings, lists of strings, Booleans, parameter references, pseudo references, or the value returned by a function. The following example shows you how to declare different property value types:

**Example JSON**

"Properties" : {

"String" : "one-string-value",

"Number" : 123,

"LiteralList" : [ "first-value", "second-value" ],

"Boolean" : true,

"ReferenceForOneValue" : { "Ref" : "MyLogicalResourceName" } ,

"FunctionResultWithFunctionParams" : {

"Fn::Join" : [ "%", [ "Key=", { "Ref" : "MyParameter" } ] ] }

}

**Example YAML**

Properties:

String: OneStringValue

String: A longer string value

Number: 123

LiteralList:

- "[first]-string-value with a special characters"

- "[second]-string-value with a special characters"

Boolean: true

ReferenceForOneValue:

Ref: MyLogicalResourceName

ReferenceForOneValueShortCut: !Ref MyLogicalResourceName

FunctionResultWithFunctionParams: !Sub |

Key=%${MyParameter}

You can conditionally create a resource by associating a condition with it. You must define the condition in the [Conditions](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html) section of the template.

[**Outputs (optional)**](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html)

Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name.

The optional Outputs section declares output values that you can [import into other stacks](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-importvalue.html) (to [create cross-stack references](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html)), return in response (to describe stack calls), or [view on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-view-stack-data-resources.html). For example, you can output the S3 bucket name for a stack to make the bucket easier to find.

## Syntax

The Outputs section consists of the key name Outputs, followed by a space and a single colon. You can declare a maximum of 60 outputs in a template.

The following example demonstrates the structure of the Outputs section.

### JSON

Use braces to enclose all output declarations. Delimit multiple outputs with commas.

"Outputs" : {

"*Logical ID*" : {

"Description" : "*Information about the value*",

"Value" : "*Value to return*",

"Export" : {

"Name" : "*Value to export*"

}

}

}

### YAML

Outputs:

*Logical ID*:

Description: *Information about the value*

Value: *Value to return*

Export:

Name: *Value to export*

### Output Fields

The Outputs section can include the following fields.

**Logical ID**

An identifier for the current output. The logical ID must be alphanumeric (a-z, A-Z, 0-9) and unique within the template.

**Description (optional)**

A String type that describes the output value. The description can be a maximum of 4 K in length.

**Value (required)**

The value of the property returned by the aws cloudformation describe-stacks command. The value of an output can include literals, parameter references, pseudo-parameters, a mapping value, or intrinsic functions.

**Export (optional)**

The name of the resource output to be exported for a [cross-stack reference](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html).

**Note**

The following restrictions apply to cross-stack references:

* For each AWS account, Export names must be unique within a region.
* You can't create cross-stack references across regions. You can use the intrinsic functionFn::ImportValue to import only values that have been exported within the same region.
* For outputs, the value of the Name property of an Export can't use Ref or GetAtt functions that depend on a resource.

Similarly, the ImportValue function can't include Ref or GetAtt functions that depend on a resource.

* You can't delete a stack if another stack references one of its outputs.
* You can't modify or remove an output value that is referenced by another stack.

You can use intrinsic functions to customize the Name value of an export. The following examples use the Fn::Join function.

JSON

"Export" : {

"Name" : {

"Fn::Join" : [ ":", [ { "Ref" : "AWS::StackName" }, "AccountVPC" ] ]

}

}

YAML

Export:

Name: !Join [ ":", [ !Ref "AWS::StackName", AccountVPC ] ]

To associate a condition with an output, define the condition in the [Conditions](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html) section of the template.