AWS CloudFormation

User Guide

API Version 2010-05-15

Amazon Web Services

AWS CloudFormation: User Guide

Amazon Web Services

Welcome

The *AWS CloudFormation User Guide* explains how to use the AWS CloudFormation service.

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, Amazon Elastic Block Store, Amazon Simple Notification Service, 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).

How Do I...?

How do I...?	Relevant Sections
Decide if AWS CloudFormation is right for my needs?	http://aws.amazon.com/cloudformation/
Get started with AWS CloudFormation quickly?	AWS CloudFormation Getting Started Guide
Find how to do specific tasks?	Modifying AWS CloudFormation Templates (p. 150)
Learn how AWS CloudFormation works?	Introduction (p. 2)
Learn about AWS CloudFormation templates?	Working with AWS CloudFormation Templates (p. 89)
Get starter-template fragments that I can use in my own templates?	Template Snippets (p. 109)
See example templates?	Example Templates (p. 98)
Learn about using and modifying templates?	Working with AWS CloudFormation Templates (p. 89)
Learn the details of the AWS CloudFormation tools?	Command Line Tools Reference (p. 340)
Learn the details of the AWS CloudFormation sample templates?	Template Anatomy (p. 90) and Template Reference (p. 187)

Introduction

Topics

AWS CloudFormation enables you to create and delete related AWS resources together as a unit called a stack. You define the characteristics of a stack parameters, mappings, resource properties, and output values) using a template (a JSON-compliant text file). You can write your template from scratch, or start with one of the example templates we provide. You can use a number of AWS products with AWS CloudFormation, such as Amazon EC2, Amazon RDS and AWS Elastic Beanstalk (for a complete list, see Resource Property Types Reference (p. 292)).

The following sections introduce these concepts in more detail.

Stacks

The stack is a collection of AWS resources. With AWS CloudFormation you can do the following with your stacks:

- Create an AWS CloudFormation stack using `cfn-create-stack`, providing a name, and specifying a template that defines the stack.
- Track the progress of the create operation using `cfn-describe-stack-events`. AWS CloudFormation optimizes the order of member resource creation during stack creation, taking into account resource dependencies, so it's not possible to predict the order in which each resource will be created. The `cfn-describe-stack-events` command enables you to monitor the progress.

- List your running stacks using `cfn-describe-stacks` or `cfn-list-stacks`, filtering by a specific stack name or stack status. Only running stacks and stacks in the process of being created or deleted are listed with `cfn-describe-stacks`. You can use `cfn-list-stacks` to list stacks that have any status (even if they have been deleted within the past 90 days), filtering on the status if you need to.
- Itemize the contents of a stack using `cfn-describe-stack-resources`. You can do this even when a stack is being created or deleted, enabling you to see the state of individual member resources.
- View the history of the events produced by a stack using `cfn-describe-stack-events`, optionally filtering by a specific stack name. You can see events for a deleted stack for up to 90 days.
- Delete a stack using `cfn-delete-stack`. When you delete a stack, each of its member resources is deleted as well. As with the stack creation, AWS CloudFormation optimizes the deletion sequence, so the order isn't predictable. You can track the progress of the deletion using `cfn-describe-stack-events` and list deleted stacks using `cfn-list-stacks`.

AWS CloudFormation makes sure all member resources are created or deleted as appropriate. Because AWS CloudFormation treats the members of a stack as a single unit, they must all be created successfully for the stack to be created. If for any reason a member resource cannot be created, AWS CloudFormation rolls the stack back and automatically deletes the member resources that were created.

Note

You are charged for the stack resources for the time they were operating (even if you deleted the stack right away).

For more information, see Modifying AWS CloudFormation Templates (p. 150).

Templates

You describe your AWS infrastructure requirements in a template. A template is a text file whose format complies with the JSON standard. Because they are just text files, you can edit and manage them in your source control system with the rest of your source code. For more information on the JSON format, see http://www.json.org.

In the template, you can declare six main objects: the template's format version (p. 4), its description (p. 4), and the parameters (p. 5), mappings (p. 6), resources (p. 4), and outputs (p. 5) you need to create your stack. The format version, descriptions, parameters, mappings and outputs are optional. You only need to declare one resource. The following depicts a valid template, which declares just a single resource with no properties.

```
{
    "Resources" : {
        "MyQueue" : {
            "Type" : "AWS::SQS::Queue",
            "Properties" : {
            }
        }
    }
}
```

A resource typically has a Properties section that contains the values needed to create that resource. If a resource does not require any properties to be declared, you can omit the Properties section of that resource.

To check your template file for syntax errors, you can use the `cfn-validate-template` command.

Note

The `cfn-validate-template` command is designed to check only the syntax of your template. It does not ensure that the property values you have specified for a resource are valid for that resource. Nor does it determine the number of resources that will exist when the stack is created.

To check the operational validity, you need to attempt to create the stack. There is no sandbox or test area for AWS CloudFormation stacks, so you are charged for the resources you create during testing.

Format Version

The template format version specifies the AWS CloudFormation template version against which the template was written.

Important

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.

Optional Description

The optional Description property enables you to associate a free valid JSON text string with a template. Descriptions enable you to document a template. The Description can be up to 4000 characters.

Optional Parameters

Optional parameters are listed in the Parameters section. Parameters enable you to pass values to your template at runtime, and can be dereferenced in the Resources and Outputs sections of the template.

Most of the sample templates declare a Parameters section (see Example Templates (p. 98)). Parameters are described more fully in Parameters (p. 5). Also, for technical details on the Parameters section format, see Parameters Declaration (p. 91).

Optional Mappings

The optional Mappings section enables you to declare conditional values. Mappings can be dereferenced in the Resources and Outputs section using the intrinsic function Fn::FindInMap (p. 324).

Two of sample templates declare a Mappings section (see Example Templates (p. 98)). Mappings are described more fully in Mappings (p. 6). Also, for technical details on the Mappings section format, see Mappings Declaration (p. 94).

Resources

The stack's member resources are listed in the Resources section. Each resource is listed separately, and specifies the resource properties necessary for creating that particular resource. Resources can be dereferenced in the Resources and Outputs sections. Their properties can be based on literals, resources, parameters, pseudo parameters, and intrinsic functions. For more information, see the next section.

All of the sample templates declare a Resources section (see Example Templates (p. 98)). Resources are described more fully in Resources (p. 7). Also, for technical details on the Resources section format, see Resource Declaration (p. 96).

Resource Properties

If a resource does not require any properties to be declared, you can omit the Properties section of that resource. Resource properties can be based on literals, resources, parameters, pseudo parameters, and intrinsic functions.

Most of the sample templates declare resources that have one or more properties (see Example Templates (p. 98)). Resource properties are described more fully in Resource Properties (p. 8). Also, for technical details on the Properties section format, see Properties Declaration (p. 97).

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.

All the sample templates declare an Outputs section (see Example Templates (p. 98)). Outputs are described more fully in Outputs (p. 10). Also, for technical details on the Outputs section format, see Outputs Declaration (p. 97).

Parameters

AWS CloudFormation parameters are values that you define in the template Parameters section. A parameter can have a default value. The default value is overridden if you specify a value for the parameter as part of the `cfn-create-stack --`*`parameters`* option. Parameter values you override at runtime are returned as part of the `cfn-describe-stacks` command, unless you suppress that in the parameter declaration by including the NoEcho property with a value of `true`. If you provide the NoEcho property, the parameter value is displayed as asterisks (*****). (Parameter values you do not override are not displayed.)

A parameter can be declared as one of following types: *`String`*, *`Number`*, or *`CommaDelimitedList`*. For a parameter that has a *`String`* or *`Number`* type, you can define constraints that AWS CloudFormation uses to validate the value of the parameter.

For the *`String`* type, you can define the following constraints: MinLength, MaxLength, Default, AllowedValues, and AllowedPattern.

For the *`Number`* type, you can define the following constraints: MinValue, MaxValue, Default, and AllowedValues. A number can be an integer or a float value.

For more information on parameter constraints, see Parameters Declaration (p. 91).

Note that all parameter values are specified as strings in the template JSON. This means Number parameter values must also be surrounded by quotes. For example, the Default value for MyNumber specifies a number and it is surrounded by quotes.

```
"Parameters" : {
    "MyNumber" : {
      "Type" : "Number",
      "Default" : "10",
      "MinValue" : "1"
    }
}
```

Parameters can be dereferenced in the Resources and Outputs section, so you can use any parameter you declare as a value for a resource, resource property, reference, function, or output value.

The following example shows the declaration of the `InstanceType` parameter, a String type that allows only the enumerated values t1.micro, m1.small, and m1.large with a default of *`m1.small`*.

```
"Parameters" : {
    "InstanceType" : {
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : ["t1.micro", "m1.small", "m1.large"],
      "Description" : "Enter t1.micro, m1.small, or m1.large. Default is
t1.micro."
    }
}
```

If you wanted to override this value at the command line, your command might resemble the following:

```
cfn-create-stack TestStack --template-file MyTemplate.template --parameters
"InstanceType=m1.large"
```

If you have more than one parameter, separate the param-value pairs with a semicolon. For example, the assume the MyTemplate.template requires two parameters. You could create a stack based on that template using a command similar to the following:

```
cfn-create-stack TestStack --template-file MyTemplate.template --parameters
"MyName=Joe;MyValue=10"
```

Note that in this case if you mistype the parameter name, AWS CloudFormation will not create the stack. It will report that the template doesn't contain the parameter.

Most of the sample templates declare a Parameters section (see Example Templates (p. 98)). Also, for technical details on the Parameters section format, see Parameters Declaration (p. 91).

Mappings

Mappings enable you to specify conditional parameter values in your template. When used with the intrinsic function Fn::FindInMap (p. 324), it works like a Case statement or lookup table.

In the optional Mappings section, you define one or more *mappings*. Each mapping has a logical name unique within the template, and defines one or more key-attribute pairs. Each attribute must be a literal string or list of literal strings. You cannot base a mapping on a parameter, pseudo parameter, or intrinsic function. Along with declaring as many conditional mapping keys as you need, you can declare which mapping key is the default. The following example shows a Mappings section that declares a map with four selections, in which a region name is mapped to a specific Amazon Machine Image (AMI) name:

```
"Mappings" : {
"RegionMap" : {
    "us-east-1" : {
      "AMI" : "ami-76f0061f"
    },
    "us-west-1" : {
      "AMI" : "ami-655a0a20"
    },
    "eu-west-1" : {
      "AMI" : "ami-7fd4e10b"
    },
    "ap-southeast-1" : {
      "AMI" : "ami-72621c20"
    }
  }
}
```

When you want to assign a mapping attribute value to a resource property or output, you use the `Fn::FindInMap` function, passing it the logical name of the mapping, the mapping key name, and the mapping attribute name you want to retrieve. Specifying a parameter or pseudo parameter as the mapping key name you pass to `Fn::FindInMap` enables you to retrieve the right attribute value for use at runtime.

Two of the sample templates declare a Mappings section (see Example Templates (p. 98)). Also, for technical details on the Mappings section format, see Mappings Declaration (p. 94).

Pseudo Parameters

Pseudo parameters are parameters AWS CloudFormation declares for you. You can use them without having to declare them in your template. AWS CloudFormation declares several pseudo parameters that you can use anywhere you might use a parameter name or logical resource name.

The following pseudo parameters are supported by AWS CloudFormation:

Name	Description
AWS::Region	Returns a string representing the AWS Region in which the encompassing resource is being created.
AWS::StackName	Returns the name of the stack as specified with the `cfn-create-stack` command.

Resources

Resources are created and deleted by AWS CloudFormation. Every template must declare a Resources section with at least one resource. Each resource is declared separately. You can specify more than one resource of the same type in the Resources section; however, a resource declaration creates only one instance of its type.

Each resource declaration includes a logical name unique within the template, and specifies its resource type (legal type names are listed in Resource Property Types Reference (p. 292)).

Resources declared in the Resources section contain a Properties section, in which you declare both required and optional properties for the resource.

The following example declares an image with an id of `myLinuxBundle-2011-12-30`:

```
"Resources" :{
    "MySimpleImage" : {
        "Type" : "AWS::EC2::Image",
        "Properties" : {
            "ImageId" : "myLinuxBundle-2011-12-30",
        }
    }
}
```

The logical name "MySimpleImage" can be used elsewhere in the template as a reference, just like a parameter. Information about resource properties is contained in the next section, Resource Properties (p. 8).

All of the sample templates declare a Resources section (see Example Templates (p. 98)). Also, for technical details on the Resources section format, see Resource Declaration (p. 96).

Resource Properties

Most resources require you to set resource-specific property values before they can be created. If a resource does not require any properties to be declared, you can omit the Properties section of that resource.

The properties declared in a resource's Properties section are specific to the type of resource being created, and are declared according to the owning resource (see AWS Resource Types Reference (p. 187)).

The following example shows the declaration of a resource named "MyVolume", which declares three properties:

```
Resources : {
    "MyVolume" : {
        "Type" : "AWS::EC2::Volume",
        "Properties" : {
            "Size" : "4",
            "SnapshotId" : "snap234",
            "AvailabilityZone" : "us-east-1a"
        }
    }
 }
```

Resource properties can base their value on literals, parameter references, pseudo parameters, and intrinsic functions.

Most of the sample templates declare resources that have one or more properties (see Example Templates (p. 98)). Also, for technical details on the Properties section format, see Properties Declaration (p. 97).

References

With the Ref (p. 326) function, you specify the logical name of any resource to dereference a value for another resource, output, parameter, or intrinsic function.

For example, in the Resources section, you might declare a security group resource with the logical name "HighRestriction". Elsewhere in another resource declaration, you can use `"Ref" : "HighRestriction"` as the value for another resource's property.

In the following example, the parameter "MyURL" is declared with a default String value of "http://aws.amazon.com". Later in the Outputs section, that value is dereferenced as `"Ref" : "MyURL"`.

```
"Parameters" : {
    "MyURL" : {
        "Type" : "String",
        "Default" : "http://aws.amazon.com"
    },

    ...

"Outputs" : {
    "URL" : {
        "Value" : { "Ref" : "MyURL" }
    }
}
```

The value that AWS CloudFormation returns for the dereferenced object depends on the resource type. See Resource Property Types Reference (p. 292) for details on the specific return values for each supported type.

Most of the sample templates make use of the Ref function (see Example Templates (p. 98)). Also, for technical details on the Ref function, see Ref (p. 326).

Intrinsic Functions

AWS CloudFormation provides functions that you can use to pass values that are not available until runtime. You specify a function inline with "Fn::*`function-name`*" supplying whatever arguments it needs inline. The arguments can be literal strings or lists of strings, a parameter reference, a pseudo parameter, or the value returned from another function.

In the following example, the value for the URL output is provided at stack creation time by the `Fn::GetAtt` function, based on the value for *`DNSName`* assigned to the *`MyLoadBalancer`* load balancer:

```
"Outputs" : {
    "URL" : {
        "Value" : { "Fn::GetAtt" : [ "MyLoadBalancer", "DNSName" ] }
    }
}
```

Currently, AWS CloudFormation supports the following functions:

Name	Purpose
Fn::Base64 (p. 323)	The base64 encoding of the argument.
Fn::FindInMap (p. 324)	Returns the value of a key from the specified Mapping.
Fn::GetAtt (p. 324)	Returns the attribute value of the specified resource.
Fn::GetAZs (p. 325)	Get the Availability Zones where you can create AWS CloudFormation stacks.
Fn::Join (p. 326)	Concatenation of the elements of the second argument, separated by the first.

Name	Purpose
Ref (p. 326)	Return a resource or value based on a logical name or parameter.

Many of the sample templates make use of intrinsic functions (see Example Templates (p. 98)). Also, for technical details on the format of intrinsic functions, see Function Declaration (p. 97).

Outputs

You can use the template Output section to declare information to be passed back to the template user. The outputs are returned by the `cfn-describe-stacks` command.

You can use literal values or AWS CloudFormation functions to declare output information.

The information in the Outputs section is returned only by `cfn-describe-stacks` for an existing stack. When the stack fails to create, or when you delete a stack, the values declared in the Output section are not returned.

In the following example, the output named *`URL`* returns the literal value `http://aws.amazon.com/cloudformation`.

```
"Outputs" : {
    "URL" : {
        "Value" : "http://aws.amazon.com/cloudformation"
    }
}
```

Most of the sample templates declare an Outputs section (see Example Templates (p. 98)). Also, for technical details on the format of outputs, see Outputs Declaration (p. 97).

Getting Started with AWS CloudFormation

If you're new to AWS CloudFormation, the guides in this section will help get you started quickly, provide you with fundamental information about using CloudFormation from the AWS Console, and guide you through the process of installing the AWS CloudFormation command line interface (CLI) so you can manage your CloudFormation stacks from your system's command prompt.

Topics

Signing Up for an AWS Account

Before you can use AWS CloudFormation or any Amazon Web Services, you must first sign up for an AWS account.

To sign up for an AWS account

1. Go to http://aws.amazon.com, and then click **Sign Up Now**.
2. Follow the on-screen instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Get Started

With the right template, you can deploy at once all the AWS resources you need for an application. In this section, you'll examine a template that declares the resources for a WordPress blog, create a WordPress blog as a stack, monitor the stack creation process, examine the resources on the stack, and then delete the stack. You'll use the AWS Management Console to complete these tasks.

Step 1: Sign up for the Service

Signing up for AWS CloudFormation also automatically signs you up for other AWS products you need, such as Amazon Elastic Compute Cloud, Amazon Relational Database Service and Amazon Simple Notification Service. You're not charged for any services unless you use them.

Note

AWS CloudFormation is a free service; however, you are charged for the AWS resources you include in your stacks at the current rates for each. For more information about AWS pricing, go to the detail page for each product on http://aws.amazon.com.

To sign up for AWS CloudFormation

1. Go to http://aws.amazon.com/cloudformation, and then click **Sign Up for AWS CloudFormation**.
2. Follow the on-screen instructions.

If you don't already have an AWS account, you'll be prompted to create one when you sign up for AWS CloudFormation.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Step 2: Pick a template

Next, you'll need a template that specifies the resources you want in your stack. For this step, you use a sample template that is already prepared. The sample template creates a basic WordPress blog using an Auto Scaling group, Amazon RDS DB Instance, and Amazon CloudWatch alarms. The template also creates Amazon EC2 and Amazon RDS security groups to control firewall settings for the Amazon EC2 instances in the Auto Scaling group and the DB Instance.

Important

AWS CloudFormation is free, but the AWS resources that AWS CloudFormation creates will be live (and not running in a sandbox). You will incur the standard usage fees for these resources until you terminate them in the last task in this tutorial. The total charges will be minimal. For information on how you might minimize any charges, go to http://aws.amazon.com/free/.

To view the template

- You can download or view the WordPress sample template from https://s3.amazonaws.com/awsdocs/AWSCloudFormation/2010-05-15/templates/WordPress.template. You don't need to download it unless you want to inspect it. You will use the template URL later in this guide.

A template is a JavaScript Object Notation (JSON) text file containing the configuration information about the AWS resources you want to create in the stack. If you look through the example WordPress template, you will see six top-level objects: AWSTemplateFormatVersion, Description, Parameters, Mappings, Resources, and Outputs; however, only the Resources object is required.

The Resources object contains the definitions of the AWS resources you want to create with the template. Each resource is listed separately and specifies the properties necessary for creating that particular resource. The following resource declaration in the template contains the configuration for the Amazon RDS DB Instance, which in this example has the logical name WordPressDB:

```
    "WordPressDB": {
      "Properties": {
        "Engine": "MySQL",
        "DBName": {
          "Ref": "WordPressDBName"
        },
        "Port": { "Ref": "WordPressDBPort" },
        "MultiAZ" : { "Fn::FindInMap" : [ "AWSRegionCapabilities",
                                          { "Ref" : "AWS::Region" },
                                          "RDSMultiAZ"] },
        "MasterUsername": {
          "Ref": "WordPressUser"
        },
        "DBInstanceClass": "db.m1.small",
        "DBSecurityGroups": [
          {
            "Ref": "DBSecurityGroup"
          }
        ],
        "AllocatedStorage": "5",
        "MasterUserPassword": {
          "Ref": "WordPressPwd"
        }
      },
      "Type": "AWS::RDS::DBInstance"
    },
```

If you have created DB Instances before, you'll recognize properties, such as Engine, DBInstanceClass, and AllocatedStorage, that determine the configuration of the DB Instance. Resource declarations are an efficient way to specify all these configuration settings at once. When you put resource declarations in a template, you can create and configure all the declared resources easily by using the template to create a stack. To launch the same configuration of resources, all you have to do is use the same template to create a new stack.

The resource declaration begins with a string that specifies the logical name for the resource. As you'll see, the logical name can be used to refer to resources within the template.

You use the *Parameters* object to declare values that can be passed to the template when you create the stack. A parameter is an effective way to specify sensitive information, such as user names and passwords, that you don't want to store in the template itself. It is also a way to specify information that may be unique to the specific application or configuration you are deploy, for example, a domain name or instance type. When you create the WordPress stack later in this section, you'll see the set of parameters declared in the template appear on the Specify Parameters page of the Create Stack wizard, where you can specify the parameters just before creating the stack.

The following parameters are used in the template to specify values used in properties in the Amazon RDS DB Instance resource:

```
    "WordPressDBName": {
      "Default": "wordpress",
      "Description" : "The WordPress database name",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "64",
      "AllowedPattern" : "[^\\x00\\\\/.]*[^\\x00\\\\/. ]"
    },
    "WordPressUser": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The WordPress database admin account username",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    },
    "WordPressPwd": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The WordPress database admin account password",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*"
    },
    "WordPressDBPort": {
      "Default": "3306",
      "Description" : "TCP/IP port for the WordPress database",
      "Type": "Number",
      "MinValue": "1150",
      "MaxValue": "65535"
    },
```

In the WordPressDB resource declaration, you see the DBName property specified with the WordPressDBName parameter:

```
        "DBName": {
          "Ref": "WordPressDBName"
        },
```

The braces contain a call to the Ref (p. 326) function with WordPressDBName as its input. The Ref function returns the value of the object it refers to. In this case, it's the WordPressDBName parameter, and the Ref function sets the DBName property to the value that was specified for WordPressDBName when the stack was created.

The Ref function can also set a resource's property to the value of another resource. For example, the resource declaration WordPressDB contains the following property declaration:

```
        "DBSecurityGroups": [
          {
            "Ref": "DBSecurityGroup"
          }
        ],
```

The DBSecurityGroups property takes a list of Amazon RDS DB Security Groups. The Ref function has an input of DBSecurityGroup, which is the logical name of a DB security group in the template, and adds the name of DBSecurityGroup to the DBSecurityGroups property.

In the template, you'll also find a *Mappings* object. You use mappings to declare conditional values that are evaluated in a similar manner as a switch statement. The template uses mappings to select the correct Amazon machine image (AMI) for the region and the architecture type for the instance type. *Outputs* define custom values that are returned by the `cfn-describe-stacks` command and in the AWS Management Console's Outputs tab after the stack is created. You can use output values to return information from the resources in the stack, such as the URL for a website created in the template. We'll cover mappings, outputs, and other things about templates in more detail in Learn Template Basics (p. 22).

That's enough about templates for now. Let's start creating a stack.

Step 3: Make sure you have prepared any required items for the stack

Before you create a stack from a template, you must ensure that all dependent resources that the template requires are available. A template can use or refer to both existing AWS resources and resources declared in the template itself. AWS CloudFormation takes care of checking references to resources in the template and also checks references to existing resources to ensure that they exist in the region where you are creating the stack. If your template refers to a dependent resource that does not exist, stack creation will fail.

The example WordPress template contains an input parameter, KeyName, that specifies the key pair used for the EC2 instances created in the Auto Scaling group declared in the template. The template depends on the user who creates a stack from the template to supply a valid key pair for the KeyName parameter. If you supply a valid key pair name, the stack will be created. If you don't supply a valid key pair name, the stack will be rolled back.

Make sure you have a valid EC2 key pair, and make note of the **Key Pair Name**, *before* you create the stack.

Note

If you don't have an EC2 key pair to use, you must create the key pair in the same region where you are creating the stack. For information on creating a key pair, see Getting an SSH Key Pair in the Amazon EC2 User Guide.

Now that you have a valid key pair, let's use the WordPress template to create a stack.

Step 4: Create the stack

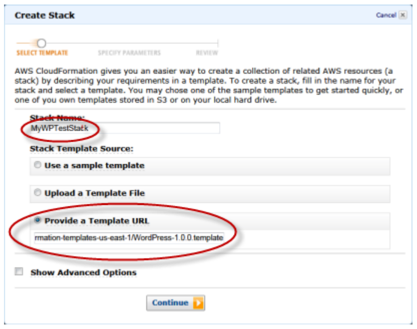
You will create your stack based on the *WordPress-1.0.0* file discussed earlier. The template contains several AWS resources including a LoadBalancer, an Amazon Relational Database Service DB Instance, and an Auto Scaling group.

To create the WordPress stack

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Click **Create New Stack**.

3. In the **Stack Name** box, type a stack name. For this example, use MyWPTestStack. The stack name must not contain spaces.
4. Click **Provide a Template URL**. In the box below, type or paste the URL for the sample WordPress template, and then click **Continue**:

   ```
   https://s3.amazonaws.com/awsdocs/AWSCloudFormation/2010-05-15/templates/WordPress.template
   ```

   Create Stack

   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 chose one of the sample templates to get started quickly, or one of you own templates stored in S3 or on your local hard drive.

   Stack Name: MyWPTestStack

   Stack Template Source:

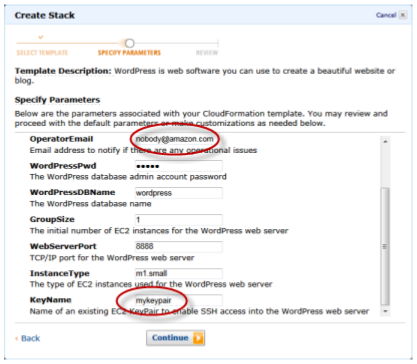
   - Use a sample template
   - Upload a Template File
   - [✓ Provide a Template URL] rmation-templates-us-east-1/WordPress-1.0.0.template

   Show Advanced Options

5. On the **Specify Parameters** page, in the **OperatorEmail** box, enter an email address where notifications can be sent when CloudWatch alarms and Auto Scaling events occur.

   Note

   On the **Specify Parameters** page, you'll recognize the parameters from the Parameters object of the template.
6. In the **KeyName** box, enter the name of a valid Amazon EC2 key pair in the same region you are creating the stack. You may need to scroll down to see **KeyName** on the page.
7. When you have the settings the way you want, click **Continue**.

   Create Stack

   Template Description: WordPress is web software you can use to create a beautiful website or blog.

   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.

OperatorEmail	nobody@amazon.com
Email address to notify if there are any operational issues	
WordPressPwd	•••••
The WordPress database admin account password	
WordPressDBName	wordpress
The WordPress database name	
GroupSize	1
The initial number of EC2 instances for the WordPress web server	
WebServerPort	8888
TCP/IP port for the WordPress web server	
InstanceType	m1.small
The type of EC2 instances used for the WordPress web server	
KeyName	mykeypair
Name of an existing EC2 KeyPair to enable SSH access into the WordPress web server	

8. Review the information for the stack. When you're satisfied with the settings, click **Create Stack**, and then click **Close**.

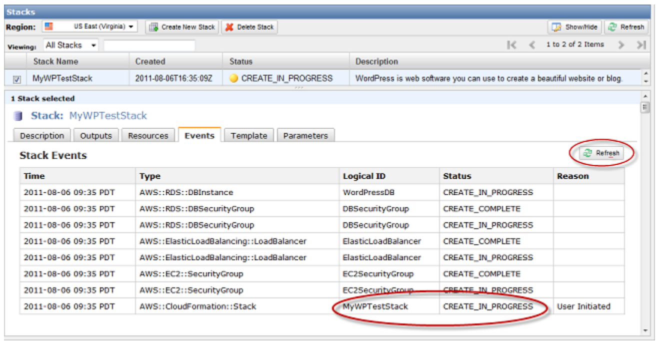
Your stack may take several minutes to create—but you probably don't want to just sit around waiting. If you're like us, you'll want to know how the stack creation is going.

Step 5: Monitor the progress of stack creation

After you complete the Create Stack wizard, AWS CloudFormation begins creating the resources specified in the template. Your new stack, MyWPTestStack, appears in the list at the top portion of the **CloudFormation** console. Its status should be CREATE_IN_PROGRESS. You can see detailed status for a stack by viewing its events.

To view the events for the stack

1. On the AWS CloudFormation console, select the stack MyWPTestStack in the list.
2. In the pane below the list, click the **Events** tab.



	Stack Name	Created	Status	Description
✓	MyWPTestStack	2011-08-06T16:35:09Z	CREATE_IN_PROGRESS	WordPress is web software you can use to create a beautiful website or blog.

1 Stack selected

Stack: MyWPTestStack

Stack Events

Time	Type	Logical ID	Status	Reason
2011-08-06 09:35 PDT	AWS::RDS::DBInstance	WordPressDB	CREATE_IN_PROGRESS	
2011-08-06 09:35 PDT	AWS::RDS::DBSecurityGroup	DBSecurityGroup	CREATE_COMPLETE	
2011-08-06 09:35 PDT	AWS::RDS::DBSecurityGroup	DBSecurityGroup	CREATE_IN_PROGRESS	
2011-08-06 09:35 PDT	AWS::ElasticLoadBalancing::LoadBalancer	ElasticLoadBalancer	CREATE_COMPLETE	
2011-08-06 09:35 PDT	AWS::ElasticLoadBalancing::LoadBalancer	ElasticLoadBalancer	CREATE_IN_PROGRESS	
2011-08-06 09:35 PDT	AWS::EC2::SecurityGroup	EC2SecurityGroup	CREATE_COMPLETE	
2011-08-06 09:35 PDT	AWS::EC2::SecurityGroup	EC2SecurityGroup	CREATE_IN_PROGRESS	
2011-08-06 09:35 PDT	AWS::CloudFormation::Stack	MyWPTestStack	CREATE_IN_PROGRESS	User Initiated

3. To update the event list with the most recent events, you need to click the **Refresh** button in the lower pane.

The **Events** tab displays each major step in the creation of the stack sorted by the time of each event with latest events on top.

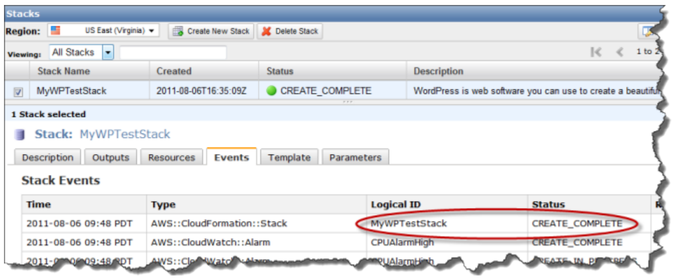
The first event (at the bottom of the event list) is the start of the stack creation process:
`MyWPTestStack AWS::CloudFormation::Stack CREATE_IN_PROGRESS`

Next are events that mark the beginning and completion of the creation of each resource. For example, creation of the EC2SecurityGroup security group would result in the following entries:

```
2011-08-06 09:35 PDT AWS::EC2::SecurityGroup EC2SecurityGroup CREATE_COMPLETE
2011-08-06 09:35 PDT AWS::EC2::SecurityGroup EC2SecurityGroup CREATE_IN_PROGRESS
```

The CREATE_IN_PROGRESS event is logged when AWS CloudFormation reports that it has begun to create the resource. The CREATE_COMPLETE event is logged when the resource is successfully created.

When AWS CloudFormation has successfully created the stack, you will see the following event at the top of the **Events** tab:
`MyWPTestStack AWS::CloudFormation::Stack CREATE_COMPLETE`



	Stack Name	Created	Status	Description
✓	MyWPTestStack	2011-08-06T16:35:09Z	CREATE_COMPLETE	WordPress is web software you can use to create a beautiful

1 Stack selected

Stack: MyWPTestStack

Stack Events

Time	Type	Logical ID	Status
2011-08-06 09:48 PDT	AWS::CloudFormation::Stack	MyWPTestStack	CREATE_COMPLETE
2011-08-06 09:48 PDT	AWS::CloudWatch::Alarm	CPUAlarmHigh	CREATE_COMPLETE
2011-08-06 09:48 PDT	AWS::CloudWatch::Alarm	CPUAlarmHigh	CREATE_IN_PROGRESS

If AWS CloudFormation cannot create a resource, it reports a CREATE_FAILED event and, by default, rolls back the stack. The **Reason** column displays the issue that caused the failure. For example, if you specified a key pair name that did not exist in the sample WordPress stack, you would see the following event for the LaunchConfiguration resource that uses the key pair in its configuration:

```
2011-08-06 10:59 PDT AWS::AutoScaling::LaunchConfiguration LaunchConfig
CREATE_FAILED The key pair 'mykeypair' does not exist
```

Step 6: Use your stack resources

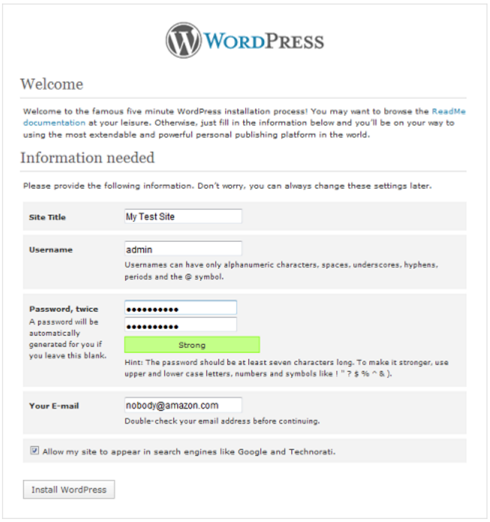
When the stack MyWPTestStack has a status of CREATE_COMPLETE, AWS CloudFormation has finished creating the stack, and you can start using its resources.

The sample WordPress stack creates a WordPress website. You can continue with the WordPress setup by running the WordPress installation script.

To complete the WordPress installation

1. On the the **Outputs** tab, in the **InstallURL** row, in the **Value** column, click the link.

   The InstallURL output value is the URL of the installation script for the WordPress website that you created with the stack.
2. On the web page for the WordPress installation, follow the on-screen instructions to complete the WordPress installation. For more information about installing WordPress, see http://codex.wordpress.org/Installing_WordPress.

   Welcome

   Welcome to the famous five minute WordPress installation process! You may want to browse the ReadMe documentation at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

   Information needed

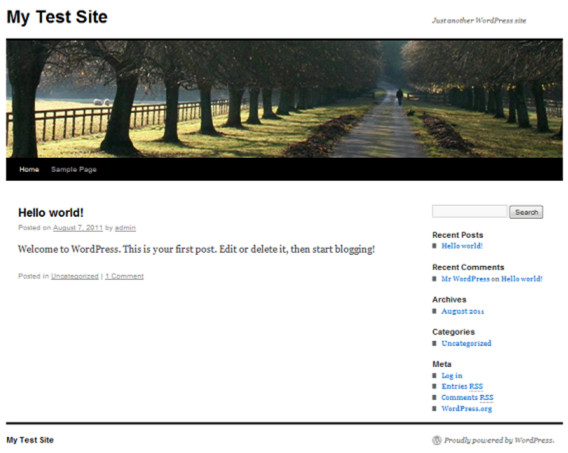
   Please provide the following information. Don't worry, you can always change these settings later.

Site Title	My Test Site
Username	admin Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods and the @ symbol.
Password, twice A password will be automatically generated for you if you leave this blank.	•••••••••• •••••••••• Strong Hint: The password should be at least seven characters long. To make it stronger, use upper and lower case letters, numbers and symbols like ! " ? $ % ^ &).
Your E-mail	nobody@amazon.com Double-check your email address before continuing.

   ☑ Allow my site to appear in search engines like Google and Technorati.

3. Return to the AWS Management Console. On the **Outputs** tab, in the **WebsiteURL** row, in the **Value** column, click the link.

If the web page for the WordPress blog that you created with this stack appears, you have successfully created a WordPress blog using a AWS CloudFormation template.



My Test Site

Just another WordPress site

Home Sample Page

Hello world!

Posted on August 7, 2011 by admin

Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

Posted in Uncategorized | 1 Comment

Search

Recent Posts
- Hello world!

Recent Comments
- Mr WordPress on Hello world!

Archives
- August 2011

Categories
- Uncategorized

Meta
- Log in
- Entries RSS
- Comments RSS
- WordPress.org

My Test Site

Proudly powered by WordPress.

Now that you've created a running application, let's take a look at its resources.

Step 7: View your stack resources

AWS CloudFormation uses the resource declarations in a template to create and configure those resources when you create the stack. After you create the stack, you will probably want to monitor or manage the stack resources. To do so, you'll need to be able to identify the resources associated with the stack. Let's go over how resources are named.

In the template, every resource has a logical name that is used to reference it. When AWS CloudFormation creates the stack, it generates a physical name for the actual resource. The physical name is a combination of a portion of the stack name and the logical name, plus a unique identifier. For example, the following resource declaration for the Amazon CloudWatch alarm uses references to an SNS topic (`{ "Ref": "AlarmTopic" }`)and a LoadBalancer (`{ "Ref": "ElasticLoadBalancer" }`).

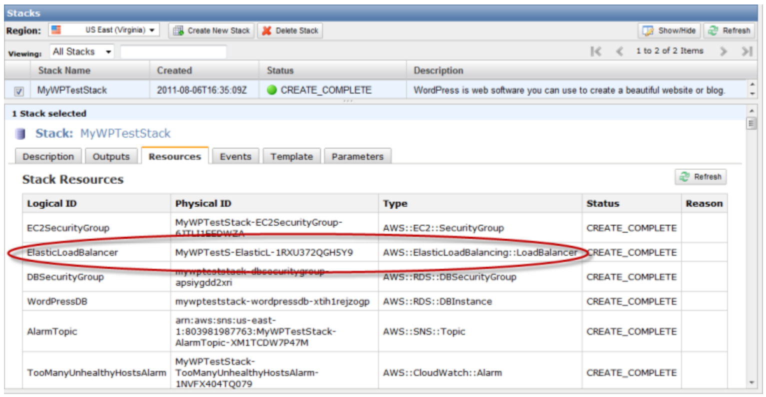
```
    "RequestLatencyAlarmHigh": {
      "Properties": {
        "EvaluationPeriods": "1",
        "Statistic": "Average",
        "Threshold": "1",
       "AlarmDescription": "Alarm if there aren't any requests coming through",

        "Period": "60",
        "AlarmActions": [
          { "Ref": "AlarmTopic" }
        ],
        "Namespace": "AWS/ELB",
        "InsufficientDataActions": [
          { "Ref": "AlarmTopic" }
        ],
        "Dimensions": [
          {
            "Name": "LoadBalancerName",
            "Value": { "Ref": "ElasticLoadBalancer" }
          }
        ],
        "ComparisonOperator": "GreaterThanThreshold",
        "MetricName": "Latency"
      },
      "Type": "AWS::CloudWatch::Alarm"
    },
```

When AWS CloudFormation creates a stack from the template, it will give the SNS topic a physical name like MyWPTestStack1-AlarmTopic-12GY2G508D2NH and the LoadBalancer a name like MyWPTestS-ElasticL-13DBUGHSIDT2W.

Now that you know how resources are named, let's take a look at some of the stack's resources.

To view the resources for a stack

1. In the AWS CloudFormation console, on the **Resources** tab, in the **Stack Resources** list, find the ElasticLoadBalancer resource, and then locate its physical name. Remember the physical name; we'll use it later to find the LoadBalancer on the EC2 console.



	Stack Name	Created	Status	Description
✓	MyWPTestStack	2011-08-06T16:35:09Z	● CREATE_COMPLETE	WordPress is web software you can use to create a beautiful website or blog.

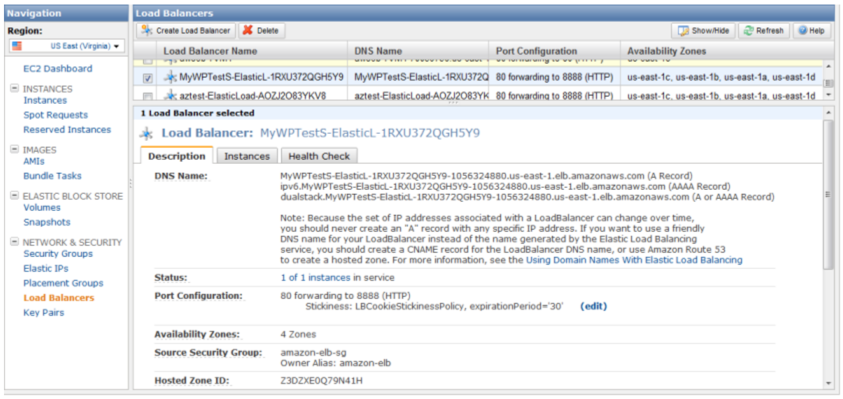
1 Stack selected

Stack: MyWPTestStack

Stack Resources

Logical ID	Physical ID	Type	Status	Reason
EC2SecurityGroup	MyWPTestStack-EC2SecurityGroup-6JTLJ1EEDWZA	AWS::EC2::SecurityGroup	CREATE_COMPLETE	
ElasticLoadBalancer	MyWPTestS-ElasticL-1RXU372QGH5Y9	AWS::ElasticLoadBalancing::LoadBalancer	CREATE_COMPLETE	
DBSecurityGroup	mywpteststack-dbsecuritygroup-apsiygdd2xri	AWS::RDS::DBSecurityGroup	CREATE_COMPLETE	
WordPressDB	mywpteststack-wordpressdb-xtih1rejzogp	AWS::RDS::DBInstance	CREATE_COMPLETE	
AlarmTopic	arn:aws:sns:us-east-1:803981987763:MyWPTestStack-AlarmTopic-XM1TCDW7P47M	AWS::SNS::Topic	CREATE_COMPLETE	
TooManyUnhealthyHostsAlarm	MyWPTestStack-TooManyUnhealthyHostsAlarm-1NVFX404TQ079	AWS::CloudWatch::Alarm	CREATE_COMPLETE	

2. In the **Amazon EC2** console, in the **Navigation** pane, click **Load Balancers**.
3. In the **Load Balancers** pane, in the **Load Balancer Name** column, click the physical name for the LoadBalancer that you got earlier.
4. In the Properties pane, on the **Description** tab, scroll down to the **Port Configuration** and **Availability Zone** sections.



	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
✓	MyWPTestS-ElasticL-1RXU372QGH5Y9	MyWPTestS-ElasticL-1RXU372Q	80 forwarding to 8888 (HTTP)	us-east-1c, us-east-1b, us-east-1a, us-east-1d
	aztest-ElasticLoad-AOZJ2O83YKV8	aztest-ElasticLoad-AOZJ2O83YK	80 forwarding to 8888 (HTTP)	us-east-1c, us-east-1b, us-east-1a, us-east-1d

1 Load Balancer selected

Load Balancer: MyWPTestS-ElasticL-1RXU372QGH5Y9

DNS Name:	MyWPTestS-ElasticL-1RXU372QGH5Y9-1056324880.us-east-1.elb.amazonaws.com (A Record) ipv6.MyWPTestS-ElasticL-1RXU372QGH5Y9-1056324880.us-east-1.elb.amazonaws.com (AAAA Record) dualstack.MyWPTestS-ElasticL-1RXU372QGH5Y9-1056324880.us-east-1.elb.amazonaws.com (A or AAAA Record) Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. If you want to use a friendly DNS name for your LoadBalancer instead of the name generated by the Elastic Load Balancing service, you should create a CNAME record for the LoadBalancer DNS name, or use Amazon Route 53 to create a hosted zone. For more information, see the Using Domain Names With Elastic Load Balancing
Status:	1 of 1 instances in service
Port Configuration:	80 forwarding to 8888 (HTTP) Stickiness: LBCookieStickinessPolicy, expirationPeriod='30' (edit)
Availability Zones:	4 Zones
Source Security Group:	amazon-elb-sg Owner Alias: amazon-elb
Hosted Zone ID:	Z3DZXE0Q79N41H

If you compare the information in the console to the AWS CloudFormation template, you will notice that Port Configuration (80 forwarding to 8888 (HTTP)) matches the properties for the single listener (LoadBalancerPort, InstancePort, and Protocol) defined by the Listeners property in the template's ElasticLoadBalancer resource.

```
    "ElasticLoadBalancer": {
      "Properties": {
        "Listeners": [
          {
            "InstancePort": {
              "Ref": "WebServerPort"
            },
            "PolicyNames": [
              "p1"
            ],
            "Protocol": "HTTP",
            "LoadBalancerPort": "80"
          }
        ],
        "HealthCheck": {
          "HealthyThreshold": "2",
          "Timeout": "5",
          "Interval": "10",
          "UnhealthyThreshold": "5",
          "Target": {
            "Fn::Join": [
              "",
              [
                "HTTP:",
                {
                  "Ref": "WebServerPort"
                },
                "/wp-admin/install.php"
              ]
            ]
          }
        },
        "AvailabilityZones": {
          "Fn::GetAZs": {
            "Ref": "AWS::Region"
          }
        },
        "LBCookieStickinessPolicy": [
          {
            "CookieExpirationPeriod": "30",
            "PolicyName": "p1"
          }
        ]
      },
      "Type": "AWS::ElasticLoadBalancing::LoadBalancer"
    },
```

The InstancePort references the input parameter WebServerPort, which has a default of 8888. Also in the Port Configuration, you will find the stickiness policy defined in the LBCookieStickinessPolicy of the template.

For Availability Zones, you will see the number of zones. This will be equal to the number of availability zones in the region. In the template, AvailabilityZones specifies all availability zones in the region. The `{"Fn::GetAZs": { "Ref": "AWS::Region" }}` function call resolves to the list of all availability zones in the region.

You can also compare the health check settings with the HealthCheck property of the ElasticLoadBalancer resource.

To view health check settings

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the Navigation pane, click **Load Balancers.**
3. Select the check box beside the physical name of your load balancer.
4. In the lower pane, click the **Health Check** tab.

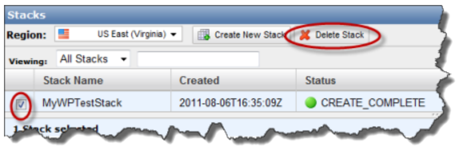
Now you know how to identify the stack's resources and find their configuration information in the AWS Management Console.

Step 8: Clean Up

You have completed the AWS CloudFormation getting started tasks. To make sure you are not charged for any unwanted services, you can clean up by deleting the stack and its resources.

To delete the stack and its resources

1. On the **AWS CloudFormation** console, select the check box beside MyWPTestStack.
2. Click **Delete Stack**.



	Stack Name	Created	Status
✓	MyWPTestStack	2011-08-06T16:35:09Z	● CREATE_COMPLETE

1 Stack selected

3. In the confirmation message that appears, click **Yes, Delete**.

The status for MyWPTestStack changes to DELETE_IN_PROGRESS. In the same way you monitored the creation of the stack, you can monitor its deletion using the Event tab. When AWS CloudFormation completes the deletion of the stack, it removes the stack from the list.

Congratulations! You successfully picked a template, created a stack, viewed and used its resources, and deleted the stack and its resources. Not only that, you were able to set up a WordPress blog using a AWS CloudFormation template. You can find other templates in the AWS CloudFormation Sample Template Library.

Now it's time to learn more about templates so that you can easily modify existing templates or create your own: Learn Template Basics (p. 22).

Learn Template Basics

Topics

- Constructed Values and Output Values (p. 31)
- Next Steps (p. 33)

In Get Started (p. 12), you learned how to use a template to create a stack. You took a brief walk through the resources declared in a template and saw how they map to resources in the stack. We also touched on input parameters and how they enable you to pass in specific values when you create a stack from a template. In this section, we'll go deeper into resources and parameters. We'll also cover the other components of templates so that you'll know how to use these components together to create templates that produce the AWS resources you want.

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 on the JSON format, see http://www.json.org.

In the template, you use a JSON structure 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 (p. 4), Description (p. 4), Parameters (p. 5), Mappings (p. 6), Resources (p. 4), and Outputs (p. 5). 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 (p. 286). For a full list of resource types, see Template Reference (p. 187).

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 HelloBucket.

```
{
    "Resources" : {
        "HelloBucket" : {
            "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 (p. 222) 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 (p. 292).

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" : {
        "HelloBucket" : {
            "Type" : "AWS::S3::Bucket"
            "Properties" : {
               "AccessControl" : "PublicRead"
            }
        }
    }
}
```

Some resources can have multiple properties, and some properties can have one or more subproperties. For example, the AWS::S3::Bucket (p. 286) resource has two properties, AccessControl and WebsiteConfiguration. The WebsiteConfiguration property has two subproperties, IndexDocument and ErrorDocument. The following template shows our original bucket resource with the additional properties.

```
{
    "Resources" : {
        "HelloBucket" : {
            "Type" : "AWS::S3::Bucket"
            "Properties" : {
               "AccessControl" : "PublicRead",
               "WebsiteConfiguration" : {
                    "IndexDocument" : "index.html",
                    "ErrorDocument" : "error.html"
               }
            }
        }
    }
}
```

Note how the sibling properties—AccessControl and WebsiteConfiguration, and IndexDocument and ErrorDocument—are separated with commas. One of the most common syntax errors in a template is a missing comma between sibling property declarations and between resources.

One of the greatest benefits of templates and AWS CloudFormation is the ability to create a set of resources that work together to create an application or solution. As discussed in Step 7: View your stack resources (p. 19), the name used for a resource within the template is a logical name. When AWS CloudFormation creates the resource, it generates a physical name that is based on the combination of the logical name, the stack name, and a unique ID.

You're probably wondering how you set properties on one resource based on the name or property of another resource. For example, you can create a CloudFront distribution backed by an S3 bucket or an EC2 instance that uses EC2 security groups, and all of these resources can be created in the same template. AWS CloudFormation has a number of intrinsic functions that you can use to refer to other resources and their properties. You can use the Ref function (p. 326) to refer to an identifying property of a resource. Frequently, this is the physical name of the resource; however, sometimes it can be an identifier, such as the IP address for an AWS::EC2::EIP (p. 220) resource or an Amazon Resource Name (ARN) for an Amazon SNS topic. For a list of values returned by the Ref function, see Ref function (p. 326). The following template contains an AWS::EC2::Instance (p. 222) resource. The resource's SecurityGroups property calls the Ref function to refer to the AWS::EC2::SecurityGroup resource InstanceSecurityGroup.

```
{
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : "mykey",
        "ImageId" : ""
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  }
}
```

You probably noticed that the Ref function call is expressed like other JSON objects, as a name-value pair separated by a colon and surrounded by braces. The function name is the name, and the input parameter for the function is the value. You'll also notice that the function call is also surrounded by brackets. In JSON, lists are surrounded by brackets. The SecurityGroups property is a list of security groups, and in this example we have only one item in the list. The following template has an additional item in the property list of the SecurityGroup.

```
{
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" }, "MyExisting
SecurityGroup" ],
        "KeyName" : "mykey",
        "ImageId" : "ami-7a11e213"
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  }
}
```

MyExistingSecurityGroup is a string that refers to an existing EC2 security group instead of a security group declared in a template. You use literal strings to refer to existing AWS resources.

In the example above, the KeyName property of the AWS::EC2::Instance (p. 222) is the literal string mykey. This means that a key pair with the name mykey must exist in the region where the stack is being created; otherwise, stack creation will fail because the key pair does not exist. The key pair you use may vary with the region where you are creating the stack, or you may want to share the template with someone else so that they can use it with their AWS account. If so, you can specify the key pair name when the user creates the stack. The Ref function can refer to input parameters that are specified at stack creation time. The following template adds a Parameters object containing the KeyName parameter, which is used to specify the KeyName property for the AWS::EC2::Instance resource.

```
{
  "Parameters" : {
    "KeyName" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "String"
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" }, "MyExisting
SecurityGroup" ],
        "KeyName" : { "Ref" : "KeyName"},
        "ImageId" : "ami-7a11e213"
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  }
}
```

The Ref function is handy if the parameter or the value returned for a resource is exactly what you want; however, you may need other attributes of a resource. For example, if you want to create a CloudFront distribution with an S3 origin, you need to specify the bucket location by using a DNS-style address. A number of resources have additional attributes whose values you can use in your template. To get these attributes, you use the Fn::GetAtt (p. 324) function. The following template creates a CloudFront distribution resource that specifies the DNS name of an S3 bucket resource using Fn::GetAtt function to get the bucket's DomainName attribute.

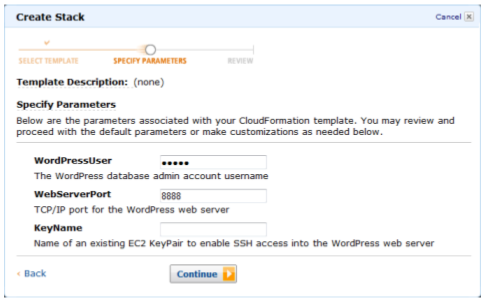
```
{
    "Resources" : {
        "myBucket" : {
            "Type" : "AWS::S3::Bucket"
        },
        "myDistribution" : {
           "Type" : "AWS::CloudFront::Distribution",
           "Properties" : {
              "DistributionConfig" : {
                   "S3Origin" : {
                       "DNSName": {"Fn::GetAtt" : ["myBucket", "DomainName"]}
                   }
               }
            }
        }
    }
}
```

The Fn::GetAtt function takes two parameters, the logical name of the resource and the name of the attribute to be retrieved. For a full list of available attributes for resources, see Fn::GetAtt (p. 324). You'll notice that the Fn::Getatt function lists its two parameters in an array. For functions that take multiple parameters, you use an array to specify their parameters.

Receiving User Input Using Input Parameters

So far, you've learned about resources and a little bit about how to use them together within a template. You've learned how to refer to input parameters, but we haven't gone deeply into how to define the input parameters themselves. Let's take a look at parameter declarations and how you can restrict and validate user input.

You declare parameters in a template's Parameters object. A parameter contains a list of attributes that define its value and constraints against its value. The only required attribute is Type, which can be String, Number, or CommaDelimitedList. You can also add a Description attribute that tells a user more about what kind of value they should specify. The parameter's name and description appear in the Specify Parameters page when a user uses the template in the Create Stack wizard.



Template Description: (none)

Specify Parameters

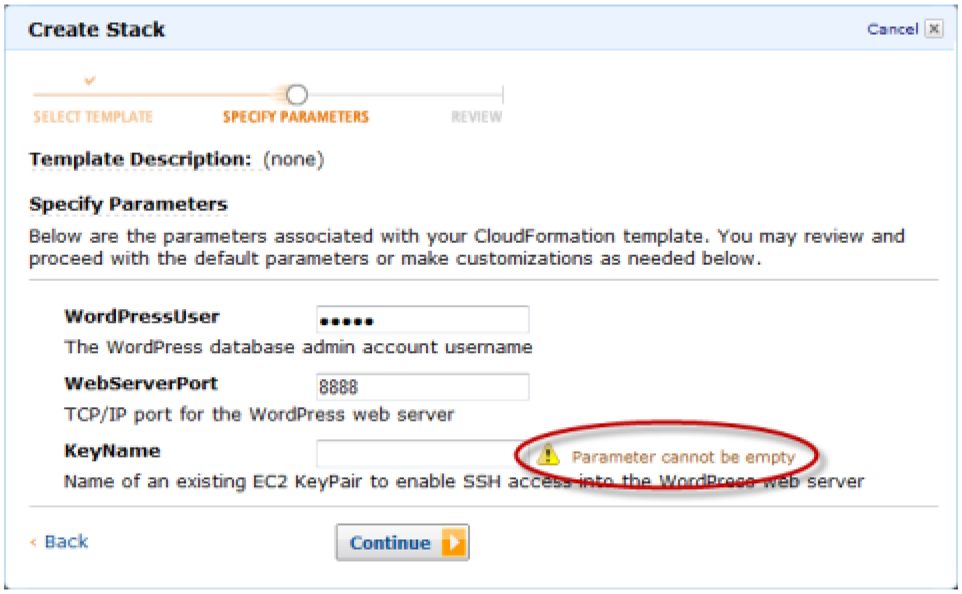
Below are the parameters associated with your CloudFormation template. You may review and proceed with the default parameters or make customizations as needed below.

WordPressUser	•••••
The WordPress database admin account username	
WebServerPort	8888
TCP/IP port for the WordPress web server	
KeyName	
Name of an existing EC2 KeyPair to enable SSH access into the WordPress web server	

The following template fragment is a Parameters object that declares the parameters used in the Specify Parameters page above.

```
  "Parameters": {
    "KeyName": {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access 
into the WordPress web server",
      "Type": "String"
    },
    "WordPressUser": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The WordPress database admin account username",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    },
    "WebServerPort": {
      "Default": "8888",
      "Description" : "TCP/IP port for the WordPress web server",
      "Type": "Number",
      "MinValue": "1",
      "MaxValue": "65535"
    }
  },
```

The KeyName parameter is of type string and has a description. You'll notice that KeyName has no Default attribute and the other parameters do. Because KeyName has no default value, it must be specified at stack creation time: AWS CloudFormation will not create the stack without a value for KeyName. When a user uses the template in the Create Stack wizard, parameters without defaults must have values specified; otherwise, the wizard displays a warning next to the parameter with the missing value and will not allow you to continue until you specify a value.



Template Description: (none)

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.

WordPressUser	•••••
The WordPress database admin account username	
WebServerPort	8888
TCP/IP port for the WordPress web server	
KeyName	Parameter cannot be empty
Name of an existing EC2 KeyPair to enable SSH access into the WordPress web server	

Omitting the Default attribute requires a user to specify a value for a parameter; however, requiring the user to input a value does not ensure that the value is valid. To validate the value of a parameter, you can declare constraints.

For the *`String`* type, you can use the following attributes to declare constraints: MinLength, MaxLength, Default, AllowedValues, and AllowedPattern. In the example above, the WordPressUser parameter has three constraints: the parameter value must be 1 to 16 character long (MinLength, MaxLength) and must begin with a letter followed by any combination of letters and numbers (AllowedPattern).

For the *`Number`* type, you can declare the following constraints: MinValue, MaxValue, Default, and AllowedValues. A number can be an integer or a float value. In the example above, the WebServerPort parameter must be a number between 1 and 65535 inclusive (MinValue, MaxValue).

Earlier in this section, we mentioned that parameters are a good way to specify sensitive or implementation-specific data, such as passwords or user names, that you need to use but do not want to embed in the template itself. For sensitive information, you can use the NoEcho attribute to prevent a parameter value from being displayed in the console, command line tools, or API. If you set the NoEcho attribute to `true`, the parameter value is returned as asterisks (*****). In the example above, the WordPressUser parameter value is not visible to anyone viewing the stack's settings, and its value is returned as asterisks.

Specifying Conditional Values Using Mappings

Parameters are a great way to enable users to specify unique or sensitive values for use in the properties of stack resources; however, there may be settings that are region dependent or are somewhat complex for users to figure out because of other conditions or dependencies. In these cases, you would want to put some logic in the template itself so that users can specify simpler values (or none at all) to get the results that they want. In an earlier example, we hardcoded the AMI ID for the ImageId property of our EC2 instance. This works fine in the US-East region, where it represents the AMI that we want. However, if the user tries to build the stack in a different region he or she will get the wrong AMI or no AMI at all. (AMI IDs are unique to a region, so the same AMI ID in a different region may not represent any AMI or a completely different one.)

To avoid this problem, you need a way to specify the right AMI ID based on a conditional input (in this example, the region where the stack is created). There are two template features that can help, the Mappings object and the AWS::Region pseudo parameter.

The AWS::Region pseudo parameter is a value that AWS CloudFormation resolves as the region where the stack is created. Pseudo parameters are resolved by AWS CloudFormation when you create the stack. Mappings enable you to use an input value as a condition that determines another value. Similar to a switch statement, a mapping associates one set of values with another. Using the AWS::Region parameter together with a mapping, you can ensure that an AMI ID appropriate to the region is specified. The following template contains a Mappings object with a mapping named RegionMap that is used to map an AMI ID to the appropriate region.

```
{
  "Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to
 the instance",
      "Type" : "String"
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {
          "AMI" : "ami-76f0061f"
      },
      "us-west-1" : {
          "AMI" : "ami-655a0a20"
      },
      "eu-west-1" : {
          "AMI" : "ami-7fd4e10b"
      },
      "ap-southeast-1" : {
          "AMI" : "ami-72621c20"
      },
      "ap-northeast-1" : {
          "AMI" : "ami-8e08a38f"
      }
    }
  },

  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region"
 }, "AMI" ]},
        "UserData" : { "Fn::Base64" : "80" }
      }
    }
  }
}
```

In the RegionMap, each region is mapped to a name-value pair. The name-value pair is a label, and the value to map. In the RegionMap, AMI is the label and the AMI ID is the value. To use a map to return a value, you use the Fn::FindInMap (p. 324) function, passing the name of the map, the value used to find the mapped value, and the label of the mapped value you want to return. In the example above, the ImageId property of the resource Ec2Instance uses the Fn::FindInMap function to determine its value by specifying RegionMap as the map to use, AWS::Region as the input value to map from, and AMI as the label to identify the value to map to. For example, if this template were used to create a stack in the us-west-1 region, ImageId would be set to ami-655a0a20.

> **Tip**
>
> The AWS::Region pseudo parameter enables you to get the region where the stack is created. Some resources, such as AWS::EC2::Instance (p. 222), AWS::AutoScaling::AutoScalingGroup (p. 190), and AWS::ElasticLoadBalancing::LoadBalancer (p. 258), have a property that specifies availability zones. You can use the Fn::GetAZs function (p. 325) to get the list of all availability zones in a region.

Constructed Values and Output Values

Parameters and mappings are an excellent way to pass or determine specific values at stack creation time, but there can be situations where a value from a parameter or other resource attribute is only part of the value you need. For example, in the following fragment from the WordPress template, the Fn::Join function constructs the Target subproperty of the HealthCheck property for the ElasticLoadBalancer resource by concatenating the WebServerPort parameter with other literal strings to form the value needed.

```
  "Resources" : {
    "ElasticLoadBalancer" : {
      "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
      "Properties" : {
        "AvailabilityZones" : { "Fn::GetAZs" : "" },
        "Instances" : [ { "Ref" : "Ec2Instance1" },{ "Ref" : "Ec2Instance2" }
],
        "Listeners" : [ {
          "LoadBalancerPort" : "80",
          "InstancePort" : { "Ref" : "WebServerPort" },
          "Protocol" : "HTTP"
        } ],
        "HealthCheck" : {
          "Target" : { "Fn::Join" : [ "", ["HTTP:", { "Ref" : "WebServerPort"
}, "/"]]},
          "HealthyThreshold" : "3",
          "UnhealthyThreshold" : "5",
          "Interval" : "30",
          "Timeout" : "5"
        }
      }
    },
```

The Fn::Join function takes two parameters, a delimiter that separates the values you want to concatenate and an array of values in the order that you want them to appear. In the example above, the Fn::Join function specifies an empty string as the delimiter and HTTP:, the value of the WebServerPort parameter, and a / character as the values to concatenate. If WebServerPort had a value of 8888, the Target property would be set to the following value:

```
HTTP:8888/
```

The Fn::Join function is also useful for declaring output values for the stack. The Outputs object in the template contains declarations for the values that you want to have available after the stack is created.

An output is a convenient way to capture important information about your resources or input parameters. For example, in the WordPress template, we declare the following Outputs object.

```
"Outputs": {
  "InstallURL": {
    "Value": {
      "Fn::Join": [
        "",
        [
          "http://",
          {
            "Fn::GetAtt": [
              "ElasticLoadBalancer",
              "DNSName"
            ]
          },
          "/wp-admin/install.php"
        ]
      ]
    },
    "Description" : "Installation URL of the WordPress website"
  },
  "WebsiteURL": {
    "Value": {
      "Fn::Join": [
        "",
        [
          "http://",
          {
            "Fn::GetAtt": [
              "ElasticLoadBalancer",
              "DNSName" ]
          }
        ]
      ]
    }
  }
}
```

Each output value has a name, a Value attribute that contains declaration of the value returned as the output value, and optionally a description of the value. In the previous example, InstallURL is the string returned by a Fn::Join function call that concatenates http://, the DNS name of the resource ElasticLoadBalancer, and /wp-admin/install.php. The output value would be similar to the following:

```
http://mywptests-elasticl-1gb51l6sl8y5v-206169572.us-east-1.elb.amazonaws.com/wp-
admin/install.php
```

In the Get Started tutorial, we used this link to conveniently go to the installation page for the WordPress blog that we created. AWS CloudFormation generates the output values after it finishes creating the stack. You can view output values in the Outputs tab of the AWS CloudFormation console or by using the cfn-describe-stacks (p. 347) command.

Next Steps

We just walked through the basic parts of a template and how to use them. You learned about the following about templates:

- Declaring resources and their properties
- Referencing other resources with the Ref function and resource attributes using the Fn::GetAtt function
- Using parameters to enable users to specify values at stack creation time and using constraints to validate parameter input
- Using mappings to determine conditional values
- Using the Fn::Join function to construct values based on parameters, resource attributes, and other strings
- Using output values based to capture information about the stack's resources.

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. For more information, see Template Format Version Declaration (p. 91) and Template Description Declaration (p. 91).

Of course, there are more advanced template and stack features. Here is a list of a few important ones that you'll want to learn more about:

Optional attributes that can be used with any resource:

- DependsOn attribute (p. 321) enables you to specify that one resource must be created after another.
- DeletionPolicy attribute (p. 322) enables you to specify how AWS CloudFormation should handle the deletion of a resource.
- Metadata (p. 323) attribute enables you to specify structured data with a resource.

AWS::CloudFormation::Stack (p. 211) enables you to embed another stack as a resource within your template.

Updating AWS CloudFormation Stacks

To update an existing stack, you must submit a template that specifies updates for the properties of resources in the stack. This can be done by using the AWS CloudFormation console, the cfn-update-stack (p. 360) CLI command, or with the UpdateStack API.

When AWS CloudFormation updates a stack, it gets new property settings for the current resources in the stack by using the template that you submit. AWS CloudFormation updates only the resources that have changes specified in the template. AWS CloudFormation does not update resources that have no changes, and those resources will continue to run without disruption during the update process. Updates to resources are handled differently depending on the type of resource and, in some cases, depending on the nature of a particular resource property. AWS CloudFormation uses one of the following techniques to update the resource:

- **Update with no interruption.** AWS CloudFormation updates the resource without disrupting operation of that resource and without changing the resource's physical name. For example, if you update any properties on an AWS::CloudWatch::Alarm (p. 214) resource, AWS CloudFormation updates the alarm's configuration and, during the update, the alarm's operation continues without disruption.

- **Reconfiguration with some interruption.** AWS CloudFormation updates the resource with some interruption. Some resources may experience some interruption during the process of applying property changes to those resources but they will retain their physical names. For example, if you update any properties on an AWS::ElasticLoadBalancing::LoadBalancer (p. 258) resource, there may be some interruption of the LoadBalancer's operation while AWS CloudFormation and Elastic Load Balancing reconfigure the LoadBalancer.
- **Replacement.** AWS CloudFormation updates the resource by recreating the resource. Some resources require creating a new resource with the property changes and generating a new physical name. AWS CloudFormation creates the replacement resource first, changes references from other dependent resources to point to the replacement resource, and then deletes the old resource. For example, if you update the Engine property of an AWS::RDS::DBInstance (p. 274) resource, AWS CloudFormation must create a new resource with the specified property changes and replace the current DBInstance resource with the new one.

Note

Whether or not a change in a resource causes an interruption in service depends on the resource itself, and on the type of change you're making to it. To learn more about updating a particular resource, see the documentation associated with that resource (for example, the EC2 documentation will provide details about what changes will interrupt an EC2 instance). You should be aware of how each resource change will affect your stack before making a change.

Depending on the technique used to modify each updated resource in your stack, you can make good decisions about when it's best to modify resources to reduce the impact of such changes on your application. In particular, you should plan carefully when resources must be *replaced* during an update. For example, if you update the Port property of an AWS::RDS::DBInstance resource, AWS CloudFormation will create a new DB Instance with a changed port setting, as well as a new physical name. To plan for this, you should take a snapshot of the current databases, prepare a strategy for how applications using that DB Instance will handle an interruption while the DB Instance is being replaced, ensure that the applications using that DB Instance take into account the new port setting and any other updates you have made, and use the DB snapshot to restore the databases on the new DB instance. This example is not exhaustive: it's meant to give you an idea of the things to plan for when the update technique for a resource is replacement.

Follow these steps to update a stack and monitor its progress.

Step 1. Get a copy of the template for the stack

The template used to update the stack must contain declarations for all the resources in the existing stack; therefore, you should use the template for the existing stack as a starting point and make your updates to a copy of that template. The first thing you need to do is get a copy of the template for the existing stack.

If you are managing your template in a source control system, use a copy of that template as a starting point. Otherwise, you can get a copy of the template used to create the stack from AWS CloudFormation.

To get the template for a stack from AWS CloudFormation:

AWS Management Console

1. In the AWS CloudFormation console, click the check box next to the stack you want to update and click the **Template** tab to view the template.
2. From the **Template**, copy the template, paste it into a text file, and save the file. You'll specify your changes to the stack's resources in this file.

CLI

1. Use the command `cfn-get-template (p. 362)` to get the template for the stack you want to update.
2. Copy the template, paste it into a text file, and save the file. Make sure that you copy *only* the template. The command encloses the template in double quotes—do not copy the double quotes surrounding the template. The template itself starts with an open brace and ends with the final close brace. You'll specify changes to the stack's resources in this file.

Step 2. Update the template

To specify changes to a stack, modify a copy of the stack's template. Currently, you can specify only the following changes in the update template:

- Add new resources, or remove existing resources.
- Add, modify, or delete properties of existing resources.

  Currently, only the following resources can be updated:
  - AWS::AutoScaling::AutoScalingGroup (p. 190)
  - AWS::AutoScaling::LaunchConfiguration (p. 193)
  - AWS::AutoScaling::ScalingPolicy (p. 197)
  - AWS::CloudWatch::Alarm (p. 214)
  - AWS::EC2::EIPAssociation (p. 221)
  - AWS::EC2::Instance (p. 222)
  - AWS::EC2::SecurityGroup (p. 237)
  - AWS::EC2::SecurityGroupIngress (p. 238)
  - AWS::EC2::SecurityGroupEgress (p. 242)
  - AWS::ElastiCache::CacheCluster (p. 254)
  - AWS::ElasticLoadBalancing::LoadBalancer (p. 258)
  - AWS::IAM::AccessKey (p. 262)
  - AWS::IAM::Group (p. 263)
  - AWS::IAM::Policy (p. 266)
  - AWS::IAM::User (p. 271)
  - AWS::IAM::UserToGroupAddition (p. 273)
  - AWS::RDS::DBInstance (p. 274)
  - AWS::RDS::DBSubnetGroup (p. 278)
  - AWS::RDS::DBSecurityGroup (p. 279)
  - AWS::EC2::SecurityGroupIngress (p. 281)
  - AWS::Route53::RecordSet (p. 282)
  - AWS::Route53::RecordSetGroup (p. 285)

  If you make changes to other resource types in the template, the stack update fails and AWS CloudFormation will roll back the stack.
- Add, modify, or delete attributes for resources (Metadata, DependsOn, and DeletionPolicy)
- Add, modify, or delete parameter declarations. However, you cannot add, modify, or delete a parameter that is used by a resource that does not support updates.
- Add, modify, or delete output value declarations.

Note

Changing the logical name of a resource is currently not supported. If your template includes an unsupported change, AWS CloudFormation returns a message that tells you that the change is not permitted.

You must specify a template containing all the resources from the current stack configuration. Unchanged resources must be specified with the *same values* as their current stack configuration. The updated resources must be one of the supported resources in the list above. In addition, an updated resource must list all the properties that you want to set on that resource and not just the properties that you want to update.

Some resources or properties may have constraints on property values or changes to those values. For example, changes to the AllocatedStorage property of an AWS::RDS::DBInstance (p. 274) resource must be greater than the current setting and if the value specified for the update does not meet those constraints, the update for that resource will fail. For the specific constraints on AllocatedStorage changes, see ModifyDBInstance.

Updates to a resource can affect the properties of other resources. If you used the Ref function (p. 326) or the Fn::GetAtt function (p. 324) to specify an attribute from an updated resource as part of a property value in another resource in the template, AWS CloudFormation will also update the resource that contains the reference to the property that has changed. For example, if you updated the MasterUsername property of an AWS::RDS::DBInstance resource and you had an AWS::AutoScaling::LaunchConfiguration resource that had a UserData property that contained a reference to the DB Instance name using the Ref function, AWS CloudFormation would recreate the DB Instance with a new name and also update the LaunchConfiguration resource.

To change a template to update a stack

1. In the template file that you saved in Step 1, modify the template to reflect the changes you want to make to the stack. Make sure you follow the guidelines above.
2. Save the template file. If you want to specify the template as a URL when you update the stack, upload the update template to an Amazon S3 bucket. Make sure that you use a bucket that is in the same region as the stack you are updating.

Step 3. Update the stack

Updating a stack is similar to creating a stack—you specify a template, parameters, and capabilities for the stack—but there are some differences. The stack retains the original settings for notification, rollback, and timeout settings. Currently, these settings cannot be updated. As we discussed in Step 2 (p. 35), an update template has specific requirements that you need to follow for updates.

There are also some differences between creating and updating parameters. When you update the stack, you can change the parameter values that are used for resources that support updates; however, you must keep the existing values in the current stack for parameters that affect resources that do not support updates. Parameters declared with NoEcho must be re-specified.

To update an existing stack

AWS Management Console

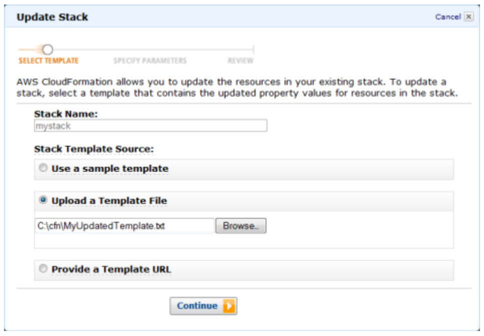
1. In the AWS CloudFormation console, click the check box next to the stack you want to update and click **Update Stack** to start the Update Stack Wizard.
2. Specify the location of the updated template:

   - **For a template stored in a file**

Click **Upload a Template File**. In the box below, enter the location for the template file, or click **Browse** to navigate to the file and select it, and then click **Continue**.

- **For a template stored in an Amazon S3 bucket**

  Click **Provide a Template URL**. In the box below, type or paste the URL for the template, and then click **Continue**.



Update Stack

AWS CloudFormation allows you to update the resources in your existing stack. To update a stack, select a template that contains the updated property values for resources in the stack.

Stack Name: mystack

Stack Template Source:

○ Use a sample template

[✓ Upload a Template File]

C:\cfn\MyUpdatedTemplate.txt

○ Provide a Template URL

3. On the **Specify Parameters** page, enter or modify the parameter values.

   Note

   AWS CloudFormation populates each parameter with the value that is currently set in the stack with the exception of parameters declared with the NoEcho attribute.

   When you have the settings the way you want, click **Continue**.
4. If you have IAM resources in the template, check **I acknowledge that this template may create IAM resources** to specify that you want to go ahead with using IAM resources in the template. For more information about using IAM resources in templates, see Controlling User Access with AWS Identity and Access Management (p. 159).
5. Review the information for the stack. When you're satisfied with the settings, click **Update Stack**, and then click **Close**. Your stack may take several minutes to update.

CLI

- Use the command `cfn-update-stack (p. 360)` to update a stack by specifying the stack to update, updated template, parameter values, and capabilities.

Step 4. Monitor the progress of the stack update

You can monitor the progress of a stack update by viewing the stack's events. The **Events** tab displays each major step in the creation and update of the stack sorted by the time of each event with latest events on top. The start of the stack update process is marked with an UPDATE_IN_PROGRESS event for the stack:

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_IN_PROGRESS
```

Next are events that mark the beginning and completion of the update of each resource that was changed in the update template. For example, updating an AWS::RDS::DBInstance (p. 274) resource named MyDB would result in the following entries:

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
```

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

The UPDATE_IN_PROGRESS event is logged when AWS CloudFormation reports that it has begun to update the resource. The UPDATE_COMPLETE event is logged when the resource is successfully created.

When AWS CloudFormation has successfully updated the stack, you will see the following event:

```
2011-09-30 09:35 PDT AWS::CloudFormation::Stack MyStack UPDATE_COMPLETE
```

If an update of a resource fails, AWS CloudFormation reports an UPDATE_FAILED event that includes a reason for the failure. For example, if your update template specified a property change that is not supported by the resource such as reducing the size of AllocatedStorage for an AWS::RDS::DBInstance (p. 274) resource, you would see events like these:

```
2011-09-30 09:36 PDT AWS::RDS::DBInstance MyDB UPDATE_FAILED Size cannot be
less than current size; requested: 5; current: 10
```

```
2011-09-30 09:35 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

If a resource update fails, AWS CloudFormation rolls back any resources that it has updated during the upgrade to their configurations before the update. Here is an example of the events you would see during an update rollback:

```
2011-09-30 09:38 PDT AWS::CloudFormation::Stack MyStack UPDATE_ROLLBACK_COMPLETE
```

```
2011-09-30 09:38 PDT AWS::RDS::DBInstance MyDB UPDATE_COMPLETE
```

```
2011-09-30 09:37 PDT AWS::RDS::DBInstance MyDB UPDATE_IN_PROGRESS
```

```
2011-09-30 09:37 PDT AWS::CloudFormation::Stack MyStack
UPDATE_ROLLBACK_IN_PROGRESS The following resource(s) failed to update: [MyDB]
```

To view stack events

AWS Management Console

1. In the AWS CloudFormation console, click the check box next to the stack you updated and click **Events** tab to view the stacks events.
2. To update the event list with the most recent events, you need to click the **Refresh** button in the lower pane.

CLI

- Use the command `cfn-describe-stack-events (p. 350)` to view the events for a stack.

Walkthrough: Updating a Stack

With AWS CloudFormation, you can update the properties for resources in your existing stacks. These changes can range from simple configuration changes, such as updating the alarm threshold on a CloudWatch alarm, to more complex changes, such as updating the Amazon Machine Image (AMI) running on an Amazon EC2 instance. Many of the AWS resources in a template can be updated, and we continue to add support for more.

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 (p. 46)—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 (p. 47)—update one of the files in the application and deploy the software using AWS CloudFormation.
3. Update the Instance Type (p. 50)—change the instance type of the underlying Amazon EC2 instance.
4. Update the AMI on an Amazon EC2 instance (p. 53)—change the Amazon Machine Image (AMI) for the Amazon EC2 instance in your stack.
5. Add a Key Pair to an Instance (p. 54)—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 (p. 55)—update the permissions of an IAM user defined in the template.
7. Change the Stack's Resources (p. 56)—add and remove resources from the stack, converting it to an auto-scaled, load-balanced application by updating the template.

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.

```
"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",

        :

        "# Install the simple web page\n",
        "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" },
        "         -r WebServerHost ",
        "         --access-key ", { "Ref" : "WebServerKeys" },
        "         --secret-key ", {"Fn::GetAtt": ["WebServerKeys", "SecretAc
cessKey"]},
        "         --region ", { "Ref" : "AWS::Region" },
        " || error_exit 'Failed to run cfn-init'\n",

        :
      ]]}}
    }
  },
```

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" : {

            :

          "files" : {

            "/etc/cfn/cfn-credentials" : {
              "content" : { "Fn::Join" : ["", [
                "AWSAccessKeyId=", { "Ref" : "WebServerKeys" }, "\n",
              "AWSSecretKey=", {"Fn::GetAtt": ["WebServerKeys", "SecretAccess
Key"]}, "\n"
              ]]},
              "mode"    : "000400",
              "owner"   : "root",
              "group"   : "root"
            },

            "/etc/cfn/cfn-hup.conf" : {
              "content" : { "Fn::Join" : ["", [
                "[main]\n",
                "stack=", { "Ref" : "AWS::StackName" }, "\n",
                "credential-file=/etc/cfn/cfn-credentials\n",
                "region=", { "Ref" : "AWS::Region" }, "\n"
              ]]},
              "mode"    : "000400",
              "owner"   : "root",
              "group"   : "root"
            },

            "/etc/cfn/hooks.d/cfn-auto-reloader.conf" : {
              "content": { "Fn::Join" : ["", [
                "[cfn-auto-reloader-hook]\n",
                "triggers=post.update\n",
                "path=Resources.WebServerHost.Metadata.AWS::CloudForma
tion::Init\n",
                "action=/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName"
 },
                "              -r WebServerHost ",
                "              --credential-file /etc/cfn/cfn-credentials ",
               "              --region     ", { "Ref" : "AWS::Region" }, "\n",

                "runas=root\n"
              ]]}
            }
          },
          :
    },

    "Properties": {

         :

      "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [

        :

        "# Start up the cfn-hup daemon to listen for changes\n",
        "/opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'\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, and a CloudWatch alarm that triggers if the CPU on the instance reaches a threshold. Here's the complete template, which you can also download or reference at
https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part1.template .

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template UpdateEC2 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.",

  "Parameters" : {

    "WebServerInstanceType" : {
      "Description" : "Amazon EC2 instance type for Web Server",
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge",
"m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge", "cc1.4xlarge"
 ],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "us-west-2"      : { "32" : "ami-16fd7026", "64" : "ami-10fd7020" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },
  "Resources" : {

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

    "WebServerKeys" : {
      "Type" : "AWS::IAM::AccessKey",
      "Properties" : {
        "UserName" : {"Ref": "WebServerUser"}
      }
    },

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

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

    "WebServerHost": {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "Comment" : "Install a simple PHP application",
        "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",

                  "?>\n"
                ]]},
                "mode"    : "000644",
                "owner"   : "apache",
                "group"   : "apache"
              },

              "/etc/cfn/cfn-credentials" : {
                "content" : { "Fn::Join" : ["", [
                  "AWSAccessKeyId=", { "Ref" : "WebServerKeys" }, "\n",
                  "AWSSecretKey=", {"Fn::GetAtt": ["WebServerKeys", "SecretAc
cessKey"]},
                  "\n"
                ]]},
                "mode"    : "000400",
                "owner"   : "root",
                "group"   : "root"
              },

              "/etc/cfn/cfn-hup.conf" : {
                "content" : { "Fn::Join" : ["", [
                  "[main]\n",
                  "stack=", { "Ref" : "AWS::StackName" }, "\n",
                  "credential-file=/etc/cfn/cfn-credentials\n",
                  "region=", { "Ref" : "AWS::Region" }, "\n"
                ]]},
                "mode"    : "000400",
                "owner"   : "root",
                "group"   : "root"
              },

              "/etc/cfn/hooks.d/cfn-auto-reloader.conf" : {
                "content": { "Fn::Join" : ["", [
                  "[cfn-auto-reloader-hook]\n",
                  "triggers=post.update\n",
                  "path=Resources.WebServerHost.Metadata.AWS::CloudForma
tion::Init\n",
                "action=/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName"
 },
                  "         -r WebServerHost ",
                  "         --credential-file /etc/cfn/cfn-credentials ",
                  "         --region     ", { "Ref" : "AWS::Region" }, "\n",
                  "runas=root\n"
                ]]}
              }
            },

            "services" : {
              "sysvinit" : {
                "httpd"    : { "enabled" : "true", "ensureRunning" : "true" },
                "sendmail" : { "enabled" : "false", "ensureRunning" : "false"
}
              }
            }
          }
        }
      },

      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                      { "Fn::FindInMap" : [ "AWSInstanceType2Arch",
                      { "Ref" : "WebServerInstanceType" }, "Arch" ] } ] },
        "InstanceType" : { "Ref" : "WebServerInstanceType" },
        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
          "#!/bin/bash\n",
          "yum update -y aws-cfn-bootstrap\n",

          "# Helper function\n",
          "function error_exit\n",
          "{\n",
          "  /opt/aws/bin/cfn-signal -e 1 -r \"$1\" '",
                  { "Ref" : "WebServerWaitHandle" }, "'\n",
          "  exit 1\n",
          "}\n",

          "# Install the simple web page\n",
          "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" },
          "         -r WebServerHost ",
          "         --access-key ", { "Ref" : "WebServerKeys" },
          "         --secret-key ", {"Fn::GetAtt": ["WebServerKeys",
"SecretAccessKey"]},
          "         --region ", { "Ref" : "AWS::Region" },
          " || error_exit 'Failed to run cfn-init'\n",

          "# Start up the cfn-hup daemon to listen for changes\n",
          "/opt/aws/bin/cfn-hup || error_exit 'Failed to start cfn-hup'\n",

          "# All done so signal success\n",
          "/opt/aws/bin/cfn-signal -e 0 -r \"WebServer setup complete\" '",
                  { "Ref" : "WebServerWaitHandle" }, "'\n"
        ]]}}
      }
    },

    "WebServerWaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle"
    },

    "WebServerWaitCondition" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "WebServerHost",
      "Properties" : {
        "Handle" : {"Ref" : "WebServerWaitHandle"},
        "Timeout" : "300"
      }
    }
```

```
  },

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

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. For more information, see Updating Auto Scaling Groups (p. 50).

Create the Initial Stack

For the purposes of this example, we'll use the AWS Management Console to create an initial stack from the sample template.

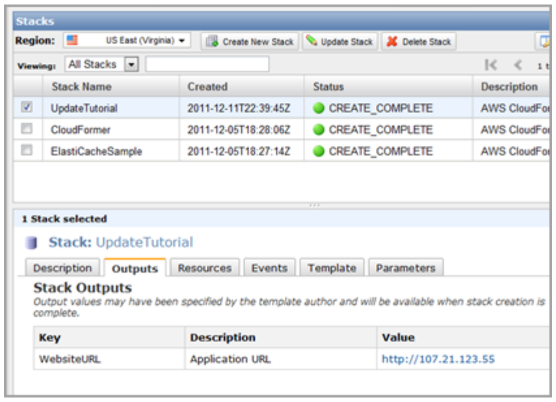
Caution

Completing this procedure will deploy live AWS services. You will be charged the standard usage rates as long as these services are running.

To create the stack from the AWS Management Console

1. Download the template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part1.template and save it in a safe place on your system. Note the location because you'll need to use the file in a subsequent step.
2. Log in to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation .
3. Click **Create New Stack**.
4. In the **Create New Stack** wizard, on the **Select Template** page, type **`UpdateTutorial`** in the **Stack Name** text box. On the same page, click **Upload a Template file** and browse to the file that you downloaded in the first step, and then click **Continue**.
5. On the **Specify Parameters** page, in the **Web Server Instance Type** box, type **`t1.micro`**.
6. Select the **I acknowledge that this template may create IAM resources** check box, and then click **Continue**. 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.
7. On the **Review** page, verify that all the settings are as you want them, and then click **Finish**.

After the status of your stack is CREATE_COMPLETE, the output tab will display the URL of your website:



	Stack Name	Created	Status	Description
✓	UpdateTutorial	2011-12-11T22:39:45Z	● CREATE_COMPLETE	AWS CloudFo
	CloudFormer	2011-12-05T18:28:06Z	● CREATE_COMPLETE	AWS CloudFo
	ElastiCacheSample	2011-12-05T18:27:14Z	● CREATE_COMPLETE	AWS CloudFo

1 Stack selected

Stack: UpdateTutorial

Stack Outputs

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

Key	Description	Value
WebsiteURL	Application URL	http://107.21.123.55

If you click the value of the WebsiteURL output, you will see your new PHP application working.

Update the Application

Now that we have deployed the stack, let's update the application. We'll make a simple change to the text that is printed out by the application. To do so, we’ll add an echo command to the index.php file as shown in this template snippet:

```
"WebServerHost": {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "config" : {
              :

            "files" : {

              "/var/www/html/index.php" : {
                "content" : { "Fn::Join" : ["", [
                  "<?php\n",
                "echo '<h1>AWS CloudFormation sample PHP application</h1>';\n",

                  "echo 'Updated version via UpdateStack';\n ",
                  "?>\n"
                ]]},
                "mode"    : "000644",
                "owner"   : "apache",
                "group"   : "apache"
              },

              :

      }
    },
```

You can manually edit the template you downloaded previously, or download the updated template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part2.template.

Now, we'll update the stack.

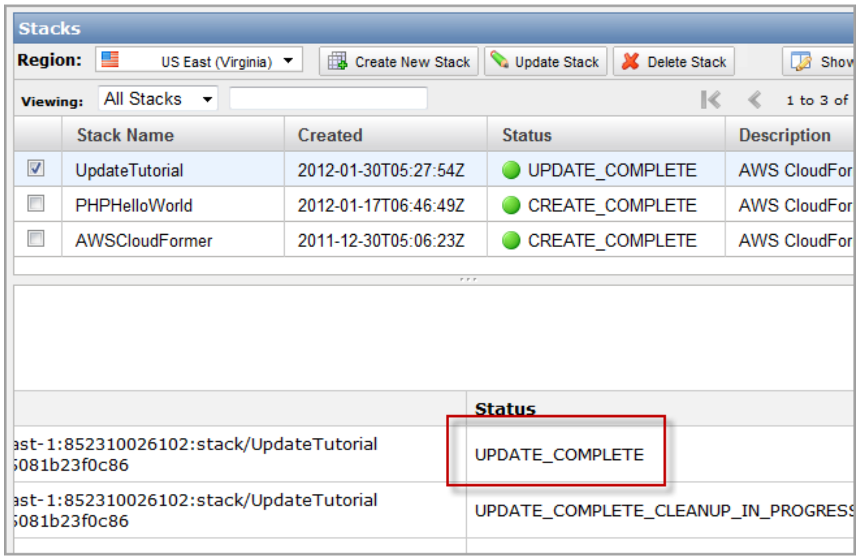
To update the stack from the AWS Management Console

1. Log in to the AWS CloudFormation console, at: https://console.aws.amazon.com/cloudformation.
2. On the AWS CloudFormation dashboard, click the stack you created previously, and then click **Update Stack**.
3. In the **Update Stack** wizard, on the **Select Template** page, click **Upload a Template File**, select the modified template, and then click **Continue**.
4. Select the **I acknowledge that this template may create IAM resources** check box, and then click **Continue**. 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.
5. On the **Review** page, verify that all the settings are as you want them, and then click **Continue**.

If you update the stack from the AWS Management Console, you will notice that the parameters that were used to create the initial stack are prepopulated on the **Parameters** page of the **Update Stack** wizard. If you use the cfn-update-stack command line tool, be sure to type in the same values for the parameters that you used originally to create the stack.

When your stack is in the UPDATE_COMPLETE state, you can click the WebsiteURL output value again to verify that the changes to your application have taken effect. By default, the cfn-hup daemon runs every 15 minutes, so it may take up to 15 minutes for the application to change once the stack has been updated.

To see the set of resources that were updated, go to the AWS CloudFormation console. On the **Events** tab, look at the stack events. In this particular case, the metadata for the Amazon EC2 instance WebServerHost was updated, which caused AWS CloudFormation to also reevaluate the Elastic IP address and the WaitCondition resource to ensure that there were no changes that affected the update. None of the other stack resources were modified. AWS CloudFormation will update only those resources in the stack that are affected by any changes to the stack. Such changes can be direct, such as property or metadata changes, or they can be due to dependencies or data flows through Ref, GetAtt, or other intrinsic template functions.



	Stack Name	Created	Status	Description
☑	UpdateTutorial	2012-01-30T05:27:54Z	● UPDATE_COMPLETE	AWS CloudFor
☐	PHPHelloWorld	2012-01-17T06:46:49Z	● CREATE_COMPLETE	AWS CloudFor
☐	AWSCloudFormer	2011-12-30T05:06:23Z	● CREATE_COMPLETE	AWS CloudFor

	Status
ast-1:852310026102:stack/UpdateTutorial 5081b23f0c86	UPDATE_COMPLETE
ast-1:852310026102:stack/UpdateTutorial 5081b23f0c86	UPDATE_COMPLETE_CLEANUP_IN_PROGRESS

This simple update illustrates the process; however, you can make much more complex changes to the files and packages that are deployed to your Amazon EC2 instances. For example, you may decide that you need to add MySQL to the instance, along with PHP support for MySQL. To do so, simply add the additional packages and files along with any additional services to the configuration and then update the stack to deploy the changes. In the following template snippet, the changes are highlighted in red:

```
    "WebServerHost": {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "Comment" : "Install a simple PHP application",
        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
                "httpd"             : [],
                "php"               : [],
                "php-mysql"         : [],
                "mysql-server"      : [],
                "mysql-devel"       : [],
                "mysql-libs"        : [],
                "mysql"             : []
              }
            },

            :

            "services" : {
              "sysvinit" : {
                "httpd"    : { "enabled" : "true", "ensureRunning" : "true" },

                "mysqld"   : { "enabled" : "true", "ensureRunning" : "true" },

                "sendmail" : { "enabled" : "false", "ensureRunning" : "false"
}
              }
            }
          }
        }
      },

      "Properties": {
          :
      }
    }
```

You can also use UpdateStack, along with the CloudFormation metadata, to update to new versions of the packages used by the application. In the previous examples, the version property for each package is empty, indicating that cfn-init should install the latest version of the package.

```
     "packages" : {
       "yum" : {
         "httpd"             : [],
         "php"               : []
     }
```

You can optionally specify a version string for a package. If you change the version string in subsequent update stack calls, the new version of the package will be deployed. Here's an example of using version numbers for RubyGems packages. Any package that supports versioning can have specific versions.

```
  "packages" : {
    "rubygems" : {
      "mysql"           : [],
      "rubygems-update" : ["1.6.2"],
      "rake"            : ["0.8.7"],
      "rails"           : ["2.3.11"]
    }
    }
```

Updating Auto Scaling Groups

If you are using Auto Scaling groups in your template, as opposed to Amazon EC2 instance resources, updating the application will work in exactly the same way; however, AWS CloudFormation does not provide any synchronization or serialization across the EC2 instances in an Auto Scaling group. The cfn-hup daemon on each host will run independently and update the application on its own schedule. When you use cfn-hup to update the on-instance configuration, each instance will run the cfn-hup hooks on its own schedule; there is no coordination between the instances in the stack. You should consider the following:

- If the cfn-hup changes run on all EC2 instances in the Auto Scaling group at the same time, your service may be unavailable during the update.
- If the cfn-hup changes run at different times, old and new versions of the software may be running at the same.

Changing Resource 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 a t1.micro Amazon EC2 instance. Let's suppose that your newly created website is getting more traffic than a t1.micro instance can handle, and now you want to move to an m1.small 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 t1.micro and m1.small are 32 bit architectures, and they use the same base Amazon Linux AMI.

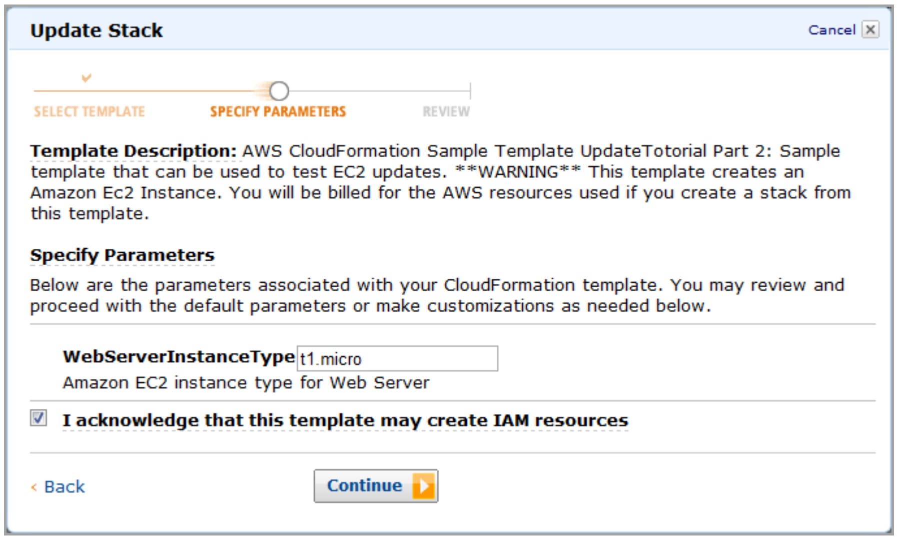
```
  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "us-west-2"      : { "32" : "ami-16fd7026", "64" : "ami-10fd7020" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },
```

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. Log in to the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. On the AWS CloudFormation dashboard, click the stack you created previously, and then click **Update Stack**.
3. In the **Update Stack** wizard, on the **Select Template** page, click **Upload a Template File**, select the modified template, and then click **Continue**.

   The Specify Parameters page appears with the parameters that were used to create the initial stack are pre-populated in the **Specify Parameters** section.



Template Description: AWS CloudFormation Sample Template UpdateTotorial Part 2: 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.

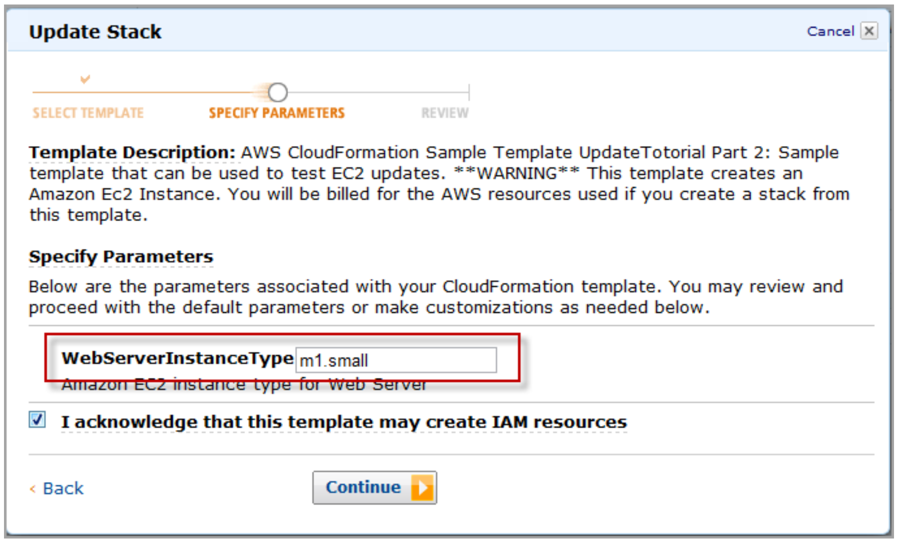
Specify Parameters

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 t1.micro
Amazon EC2 instance type for Web Server

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

4. Change the value of the **WebServerInstanceType** text box from `t1.micro` to `m1.small`.



Update Stack

SELECT TEMPLATE — SPECIFY PARAMETERS — REVIEW

Template Description: AWS CloudFormation Sample Template UpdateTutorial Part 2: 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

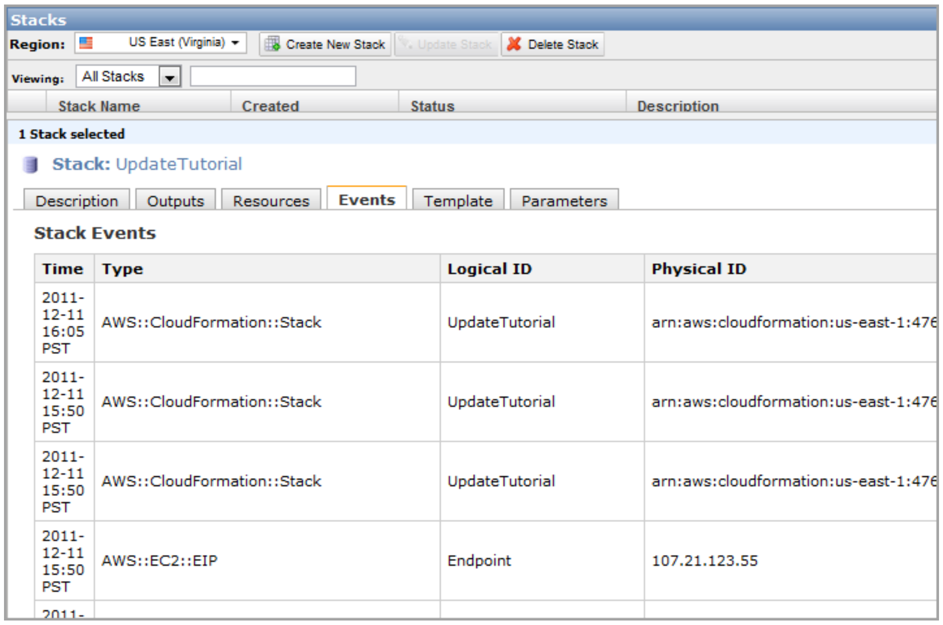
Below are the parameters associated with your CloudFormation template. You may review and proceed with the default parameters or make customizations as needed below.

WebServerInstanceType m1.small
Amazon EC2 instance type for Web Server

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

5. Select the **I acknowledge that this template may create IAM resources** check box, and then click **Continue**. 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.
6. On the **Review** page, verify that all the settings are as you want them, and then click **Continue**.

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.



1 Stack selected

Stack: UpdateTutorial

Stack Events

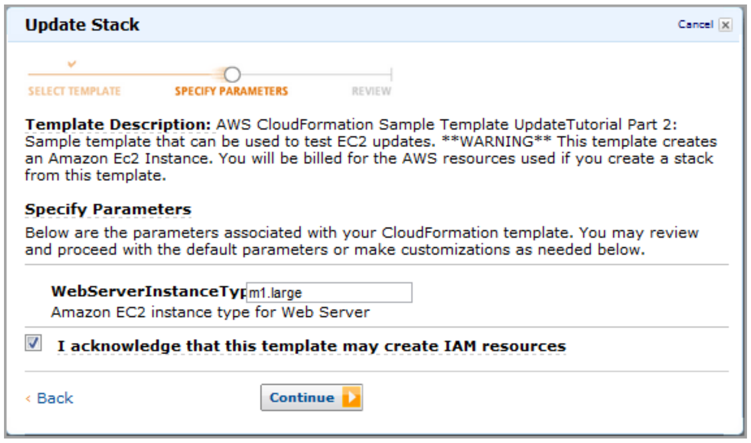
Time	Type	Logical ID	Physical ID
2011-12-11 16:05 PST	AWS::CloudFormation::Stack	UpdateTutorial	arn:aws:cloudformation:us-east-1:476
2011-12-11 15:50 PST	AWS::CloudFormation::Stack	UpdateTutorial	arn:aws:cloudformation:us-east-1:476
2011-12-11 15:50 PST	AWS::CloudFormation::Stack	UpdateTutorial	arn:aws:cloudformation:us-east-1:476
2011-12-11 15:50 PST	AWS::EC2::EIP	Endpoint	107.21.123.55
2011-			

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

Update the AMI on an Amazon EC2 instance

Now let's look at how we might change the Amazon Machine Image (AMI) running on the instance. We will trigger the AMI change by updating the stack to use a new EC2 instance type, such as m1.large, which is a 64-bit instance type.

As in the previous section, we'll use our existing template to change the instance type used by our example stack. In the Stack Update wizard, on the Specify Parameters page, change the value of the Web Server Instance Type:



Template Description: AWS CloudFormation Sample Template UpdateTutorial Part 2: 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.

WebServerInstanceTyp m1.large
Amazon EC2 instance type for Web Server

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

In this case, we cannot simply start and stop the instance to modify the AMI; AWS CloudFormation considers this a change to an immutable property of the resource. In order to make a change to an immutable property, AWS CloudFormation must launch a replacement resource, in this case a new Amazon EC2 instance running the new AMI.

After the new instance is running, AWS CloudFormation updates the other resources in the stack, such as the Elastic IP address, to point to the new resource. When all new resources are created, the old resource is deleted, a process known as UPDATE_CLEANUP. This time, you will notice that the instance ID of the instance in the stack has changed as a result of the update. The events in the Event table contain a description "Requested update has a change to an immutable property and hence creating a new physical resource" to indicate that a resource was replaced.

If you have application code written into the AMI that you want to update, you can use the same stack update mechanism to update the AMI to load your new application.

To update the AMI for an instance on your stack

1. Create your new AMIs containing your application or operating system changes. For more information, go to Creating Your Own AMIs in the *Amazon Elastic Compute Cloud User Guide*.
2. Update your template to incorporate the new AMI IDs.
3. Update the stack, either from the AWS Management Console as explained in Update the Application (p. 47) or through the command line tool cfn-update-stack. For information about cfn-update-stack, see cfn-update-stack (p. 360).

When you update the stack, AWS CloudFormation detects that the AMI ID has changed, and then it triggers a stack update in the same way as we triggered the one above.

Update the Amazon EC2 Launch Configuration for an Auto Scaling Group

If you are using Auto Scaling groups rather than EC2 instances, the process of updating the running instances is a little different. With Auto Scaling resources, the configuration of the EC2 instances, such as the instance type or the AMI Id is encapsulated in the Auto Scaling launch configuration. You can make changes to the launch configuration in the same way as we made changes to the EC2 instance resources in the previous sections. However, changing the launch configuration does not impact any of the running EC2 instances in the Auto Scaling group. An updated launch configuration applies only to new instances that are created after the update.

If you want to propagate the change to your launch configuration across all the instances in your Auto Scaling group, you can use the Auto Scaling as-terminate-instance-in-auto-scaling-group command line tool to replace each instance as follows:

```
as-terminate-instance-in-auto-scaling-group <instance_id> --no-decrement-desired-
capacity
```

For more information about the Auto Scaling command line tools, go to Using the Command Line Tools in the *Auto Scaling Developer Guide*. After the instance is terminated, Auto Scaling will replace it with one that uses the new AMI. Instance replacements are not instantaneous, and it may take some time to register the new instance with Elastic Load Balancing and any other affected services. Take care not to leave your group under capacity during an upgrade.

Adding Resource 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 ac
cess",
      "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"
: …
         {"IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp"
: …
        ]
      }
    },
```

4. Update the stack, either from the AWS Management Console as explained in Update the Application (p. 47) or through the command line tool cfn-update-stack. For information about cfn-update-stack, see cfn-update-stack (p. 360).

You can download or reference the updated template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part3.template.

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:

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

The updated template can be downloaded or referenced from https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part4.template. 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 (p. 54) and convert it to an auto-scaled, load-balanced application by updating the stack.

To begin, you can manually edit the template you downloaded previously, or download the updated template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorial+Part3.template.

This will create 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, I've also changed the name of the resource 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"
}
```

You can download or reference the complete template at:
https://s3.amazonaws.com/cloudformation-templates-us-east-1/UpdateTutorialPart5.template.

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.

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.

Related Resources

For more information on 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
- Integrating AWS CloudFormation with Opscode Chef
- Integrating AWS CloudFormation with Puppet

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.

Installing the AWS CloudFormation Command Line Interface (CLI)

The AWS CloudFormation CLI provides you with a way to interact with your AWS CloudFormation stacks from your system's command-line interface. This section explains how to configure these tools for your system.

The CloudFormation CLI comprises a set of Java command-line tools. These tools will run similarly on every operating system that supports the Java Runtime Environment 1.5. Setting up the tools, however, requires steps that are tailored for the operating system you're running. This section includes specific instructions for configuring the AWS CloudFormation CLI for your Linux, Mac OS X, or Windows system.

Note

For generic (non OS-specific) setup instructions and for more information about installing the AWS CloudFormation CLI, see `README.TXT` contained in `AWSCloudFormation-cli.zip`.

Download the CLI

Before setting anything up on your system, you'll need to download the AWS CloudFormation CLI tools. To download them, go to the AWS CloudFormation Command Line Tools page, and then click **Download**.

The downloaded file is named `AWSCloudFormation-cli.zip`.

Topics

Configuring the AWS CloudFormation CLI Tools for Linux

This guide is based on Ubuntu Linux 11.10. Specific steps may vary for other versions of Linux, but these instructions should apply to most modern Linux distributions.

Topics

The CLI Tools Require Java Runtime Environment 1.5

The AWS AWS CloudFormation CLI requires the Java Runtime Environment (JRE) 1.5 or later. Depending on your Linux distribution, the runtime may already be installed. To see what version of Java is installed on your computer, open a terminal window. At the command prompt, type the following, and then press Enter:

```
$ java -version
```

If this command returns an error or reports a Java version earlier than 1.5, or returns an error, you must update your JRE to at least version 1.5 or install the runtime if it is not already installed. For information about installing the JRE, go to the Java website.

Download the CLI

Before setting anything up on your system, you'll need to download the AWS CloudFormation CLI tools. To download them, go to the AWS CloudFormation Command Line Tools page, and then click **Download**.

The downloaded file is named `AWSCloudFormation-cli.zip`.

Install the CLI

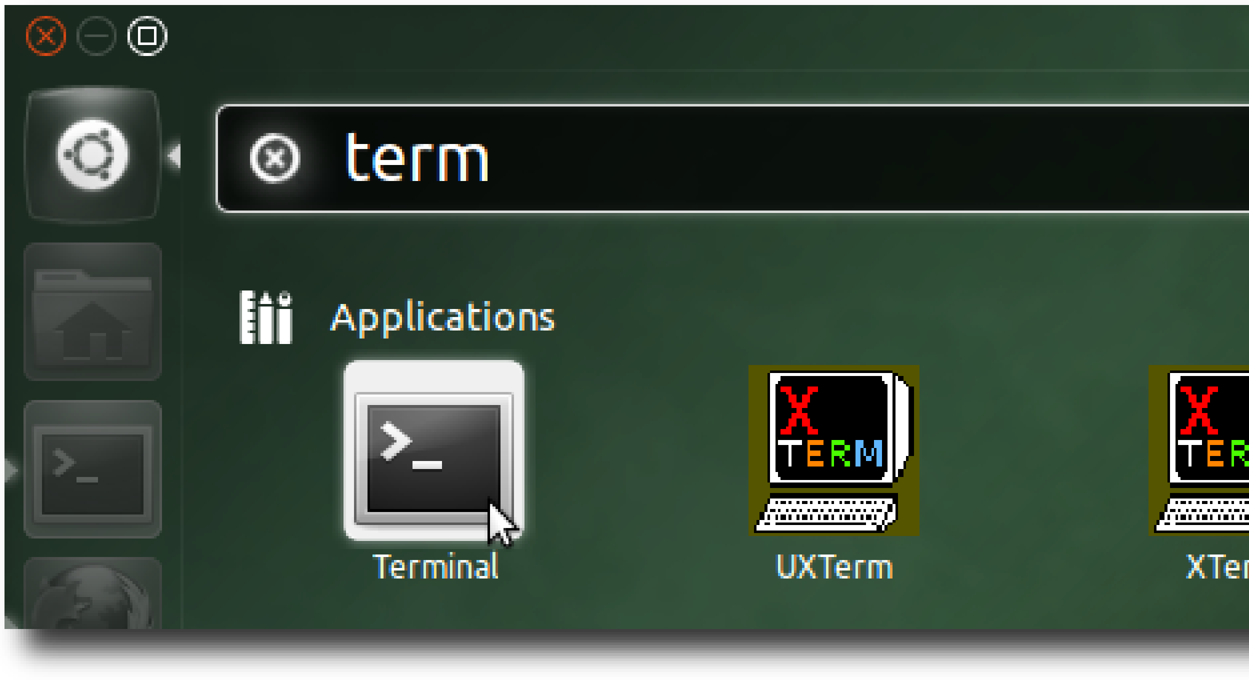
You can install the AWS CloudFormation CLI tools in any directory you choose. For simplicity, though, it's a good idea to install them in a location within your home directory (typically /home/*name*, where *name* is your Linux user name). Once you begin using AWS CloudFormation, you'll probably want to install command-line tools for other AWS services, so we'll install the AWS CloudFormation CLI tools within a directory designed to house all of these toolsets, called `.awstools`.

Note

A directory with an initial period (.) will be hidden by default to the file manager and to the **ls** command. We hide the directory so that it won't clutter the view of your home directory. If you don't want it hidden, name it simply `awstools` or anything else you choose. It won't affect how the tools work as long as you consistently substitute your chosen location for the `.awstools` directory that we'll use throughout this guide.

To prepare the installation directory

1. Open a terminal window. To open a terminal window on Ubuntu Linux:
    1. Open the Ubuntu menu (called **Dash home**).
    2. In the search box, type **`term`**.
    3. Click **Terminal** to open the terminal window.



2. Create the installation directory. At the command prompt, type the following, and then press Enter:

```
$ mkdir ~/.awstools
```

   This command will create a hidden directory (if it doesn't already exist) called `.awstools` in your home directory. You can use `~/.awstools` as the base directory for all AWS toolsets that you want to install. For now, we'll just concern ourselves with installing the AWS CloudFormation CLI tools.

To install the CLI tools

1. Change your working directory to `~/.awstools`. At the command prompt, type the following, and then press Enter:

```
$ cd ~/.awstools
```

2. Unzip `AWSCloudFormation-cli.zip`, which you downloaded earlier, in the installation directory. At the command prompt, type the following, and then press Enter:

```
$ unzip ~/Downloads/AWSCloudFormation-cli.zip
```

   This command will create a subdirectory called `AWSCloudFormation-#.#.#`, where each # represents part of the version number.
3. (Optional) Link to the installation directory. At the command prompt, type the following, replacing #.#.# with the version number of the downloaded tools, and then press Enter.

```
$ ln -s AWSCloudFormation-#.#.# cfn
```

For example, to link to version 1.0.9 of the tools, you would type:

```
$ ln -s AWSCloudFormation-1.0.9 cfn
```

Linking to the installation directory makes it easy to switch to a new version of the tools. It also gives you the ability to easily switch to another version by simply overwriting the link to point to the installation directory of the version you want to use.

Configure the CLI tools

Within the installation directory, you will find a file named `credential-file-path.template`. You can use this file to automatically supply your AWS credentials to the AWS CloudFormation CLI instead of supplying them with each command you issue.

Caution

Adding your credentials to this file can save you time, but it can also present a security risk if your system is used by others, because the file stores your access key ID and secret key as plain text.

To view your AWS access credentials

1. Go to the Amazon Web Services website at http://aws.amazon.com.
2. Click **My Account/Console**, and then click **Security Credentials**.
3. Under **Your Account**, click **Security Credentials**.
4. In the spaces provided, type your user name and password, and then click **Sign in using our secure server**.
5. Under **Access Credentials**, on the **Access Keys** tab, your access key ID is displayed. To view your secret key, under **Secret Access Key**, click **Show**.

Important

Do not share your secret access key with anyone. Anyone who has access to your secret access key will be able to use AWS services in your name and charge usage fees to your account.

If you don't want to keep your credentials in a file on your system, you can skip the next procedure, and instead pass your credentials directly to the AWS AWS CloudFormation CLI with the `--I` *ACCESSKEY* and `--S` *SECRETKEY* arguments, replacing *ACCESSKEY* and *SECRETKEY* with your AWS credentials. You will need to do this every time you issue an AWS command.

To add your AWS credentials to the AWS credential file

1. Open the terminal if you don't have it open already, and change your working directory to the AWS CloudFormation CLI installation directory. For example, if you made a link to your installation directory called `cfn` as suggested previously, At the command prompt, type the following, and then press Enter:

```
$ cd ~/.awstools/cfn
```

2. (Optional) Copy or rename `credential-file-path.template` For this example, we'll use `myAWSCredentials.txt`. At the command prompt, type the following, and then press Enter:

```
$ cp credential-file-path.template myAWSCredentials.txt
```

3. Edit the file with your favorite text editor (both `gedit` and `vi` are installed by default on Ubuntu), enter your AWS credentials as shown, and then save and close the file.

```
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
AWSSecretKey=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Note: These are only example values. Enter your own AWS security credentials in the file.

Finally, you'll need to add a few environment variables to your system. Because you'll be using the AWS CloudFormation CLI from the command-line, you can simply add these variables to your `~/.bashrc` file.

To add the AWS CloudFormation CLI tools environment variables:

1. Edit `~/.bashrc` (or the equivalent terminal startup file for your Linux variant) in your favorite editor. If this file does not exist yet, create it.
2. In `~/.bashrc`, add the following lines, substituting the appropriate directories on your own computer:

```
# AWS CloudFormation CLI tools setup
export JAVA_HOME="/usr/lib/jvm/default-java/jre/"
export AWS_CLOUDFORMATION_HOME="$HOME/.awstools/cfn"
export AWS_CREDENTIAL_FILE="$AWS_CLOUDFORMATION_HOME/myAWSCredentials.txt"
export PATH="$PATH:$AWS_CLOUDFORMATION_HOME/bin"
```

If you did not add your AWS access keys to an external file (such as the `myAWSCredentials.txt` file specified earlier), you should *not* include the `AWS_CREDENTIAL_FILE` environment variable.

Note

On Ubuntu Linux, `default-java` is a link to the default Java installation, which is typically OpenJDK. OpenJDK works fine with the AWS CloudFormation command-line tools. If you prefer a different Java runtime, you can reset `default-java` to point to your preferred runtime location, or you can set the location of your preferred runtime as the value of *`JAVA_HOME`* in `~/.bashrc`.

3. To load your environment variables, either quit and restart the terminal or, at the command prompt, At the command prompt, type the following, and then press Enter:

```
$ source ~/.bashrc
```

4. Test your installation by running **cfn-cmd**. At the command prompt, type the following, and then press Enter:

```
$ cfn-cmd
```

This command will list all of the available AWS CloudFormation CLI commands, their usage, and the available options for **cfn-cmd** itself (there are two options: `help` and `version`).

See Also

- Command Line Tools Reference (p. 340)

Configuring the CloudFormation CLI Tools for Mac OS X

This guide is based on Mac OS X Lion (10.7.4). Other versions of Mac OS X may require adjustments to these instructions.

Topics

The CLI Tools Require Java Runtime 1.5

The CloudFormation CLI tools must be installed on *Mac OS X version 10.4 (Leopard) or greater*, as Java runtime 1.5 is not supported on earlier versions of Mac OS X.

The Java Runtime Environment is installed automatically on Mac OS X, so you don't need to install it yourself. However, you can check it by opening a terminal window and typing:

```
java -version
```

If you're on Mac OS X 1.4 or greater, this should report a Java version of at least 1.5.

Download the CLI

Before setting anything up on your system, you'll need to download the AWS CloudFormation CLI tools. To download them, go to the AWS CloudFormation Command Line Tools page, and then click **Download**.

The downloaded file is named `AWSCloudFormation-cli.zip`.

Install the CLI Tools

You can install the CLI Tools in any directory you choose, though for simplicity, it's good to install them in a location within your home directory (/Users/*name*, where *name* is your Mac OS X user name). Once you begin using CloudFormation, you'll likely want to install command-line tools for other AWS services, so we'll install the CloudFormation CLI tools within a directory designed to house all of these toolsets, called `.awstools`.

Note

A directory with an initial period (.) will be hidden by default to the file manager and to the **ls** command. We hide the directory so that it won't clutter the view of your home directory. If you don't want it hidden, name it simply `awstools` or anything else you choose. It won't affect how the tools work as long as you consistently substitute your chosen location for the `.awstools` directory that we'll use throughout this guide.

To prepare the install location

1. Open a Terminal window by doing one of the following:
   - Open the Launchpad (on Mac OS X Lion), open the **Utilities** folder, and click **Terminal**.
   - Open Finder, navigate to Applications / Utilities, and double-click **Terminal**.

2. In your terminal window, create the installation directory by typing:

```
$ mkdir ~/.awstools
```

   This will create a hidden directory (if it doesn't already exist) called ".awstools" in your home directory.

You can use `~/.awstools` as the base directory for all AWS toolsets you want to install. Here, we'll just concern ourselves with installing the CloudFormation CLI tools.

To install the CLI tools

1. Change your working directory to ~/.awstools. You can do this on the command-line with:

```
$ cd ~/.awstools
```

2. Unzip the downloaded AWSCloudFormation-cli.zip file in the install location, by typing:

```
$ unzip ~/Downloads/AWSCloudFormation-cli.zip
```

   This will create a directory called AWSCloudFormation-#.#.#, where each # represents part of the version number.
3. (Optional) Link to the install directory, by typing:

```
$ ln -s AWSCloudFormation-#.#.# cfn
```

   Again, the three # characters represent part of the version number, so to make the link to version 1.0.9 of the tools, type:

```
$ ln -s AWSCloudFormation-1.0.9 cfn
```

   This enables you to easily install and switch to a new version of the tools, while giving you the freedom to switch back to an older version just as easily. Simply overwrite the link with a new one.

Configure the CLI tools

Within the install directory (which can be accessed by typing `cd ~/.awstools/cfn` on the command-line prompt if you followed the optional step for installing the CLI tools) you will find a file named `credential-file-path.template`. This file can be used to supply your AWS credentials to the tools automatically so you won't need to pass these credentials to the tools on the command-line every time you run them.

Adding your credentials to this file can save you time, but can also present a security risk if your system is used by others, since your security key will be stored as plain text on your system.

If you don't have your AWS credentials handy, you can view them by logging into the AWS console and selecting **Security Credentials** from the **My Account / Console** menu at the top of the screen. On the page, click **Access Credentials** to see your access key under the **Access Key ID** heading. Your secret key can also be viewed here by clicking the **show** button under the **Secret Access Key** heading.

Important

Do not share your secret access key with anyone! With your secret access key, anyone would be able to use AWS services and charge usage fees to your account.

If you don't want to keep your credentials in a file on your system, you can skip the next procedure, and instead pass your credentials directly to the AWS CloudFormation CLI tools with the `--I` *`ACCESSKEY`* and `--S` *`SECRETKEY`* arguments, replacing *ACCESSKEY* and *SECRETKEY* with your AWS credentials. You will need to do this every time you use the CLI tools.

To add your AWS credentials to the AWS credential file

1. Go to the AWS CloudFormation CLI tools install directory from the command-line, using:

   ```
   $ cd ~/.awstools/cfn
   ```

   This assumes that you installed the CLI tools into `~/.awstools` and either linked or renamed the AWSCloudFormation-#.#.# directory to `~/.awstools/cfn`. If you did not, replace this location in the example above with the pathname to the location where you installed the CLI tools.
2. Copy or rename the credential-file-path.template file to a filename of your choosing (you can even use credential-file-path.template as the filename, but I prefer something that better describes its purpose). For the purposes of this article, we'll use "myAWSCredentials.txt":

   ```
   $ cp credential-file-path.template myAWSCredentials.txt
   ```

3. Edit the file with your favorite text editor, enter your AWS credentials, then save and close the file. The file should contain two lines in it, like this:

   ```
   AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
   AWSSecretKey=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```

   Note: These are example values, and won't work for an actual connection to AWS. Instead, use your own AWS security credentials here.

Finally, you'll need to add a few environment variables to your system. Since you'll be using the CloudFormation CLI tools from the command-line, you can simply add these to your `~/.profile` file.

To add the CloudFormation CLI tools environment variables:

1. Edit `~/.profile` in your favorite editor. If it does not exist yet, create it.
2. In `~/.profile`, add the following lines:

```
# CloudFormation CLI tools setup
export JAVA_HOME="/System/Library/Frameworks/JavaVM.framework/Home/"
export AWS_CLOUDFORMATION_HOME="$HOME/.awstools/cfn"
export AWS_CREDENTIAL_FILE="$AWS_CLOUDFORMATION_HOME/myAWSCredentials.txt"
export PATH="$PATH:$AWS_CLOUDFORMATION_HOME/bin"
```

   This assumes that you installed the CLI tools into `~/.awstools` and have either linked or renamed the AWSCloudFormation-#.#.# directory to `~/.awstools/cfn`. If you did not, replace this location (where $HOME represents "~") with the pathname to the location where you installed the CLI tools.

   If you did not add your AWS access keys to an external file (such as the `myAWSCredentials.txt` file specified above), you should *not* assign the `AWS_CREDENTIAL_FILE` environment variable. Instead, remove or comment-out that line.
3. To load your environment variables, either quit and restart Terminal or, at the command-line prompt, type:

```
$ source ~/.profile
```

4. Test your installation by running cfn-cmd:

```
$ cfn-cmd
```

   This will list all of the available CloudFormation CLI tools, their usage, and the available options for `cfn-cmd` itself (there are two options: `help` and `version`).

See Also

- Command Line Tools Reference (p. 340)

Configuring the CloudFormation CLI Tools for Windows

This guide is based on Windows 7, Service Pack 1. Other versions of Windows may require adjustments to these instructions.

Topics

The CLI Tools Require Java Runtime 1.5

The CloudFormation CLI tools must be installed with Java runtime 1.5; Java is not installed by default on Windows 7. To download the latest version of Java, go to http://java.com/download.

To check your Java version, open a Command Prompt and type:

```
> java -version
```

If this returns a Java version of at less than 1.5, or returns an error, you must update your Java VM to at least 1.5.

Download the CLI

Before setting anything up on your system, you'll need to download the AWS CloudFormation CLI tools. To download them, go to the AWS CloudFormation Command Line Tools page, and then click **Download**.

The downloaded file is named `AWSCloudFormation-cli.zip`.

Unzip the Archive

Once you've downloaded the AWS CloudFormation CLI archive, unzip it by following these steps:

1. Open Windows Explorer and navigate to your `Downloads` directory.
2. Right-click the .zip file and click **Extract All** on the context menu.

The extracted directory will be called AWSCloudFormation-#.#.#, where #.#.# represents the current version number (currently 1.0.9).

Install the CLI Tools

You can install the CLI Tools in any directory you choose, though for simplicity, it's good to install them in a location within your home directory (identified on Windows by the `USERPROFILE` environment variable, and usually C:\Users\name, where *name* is your Windows user name). Once you begin using CloudFormation, you'll likely want to install command-line tools for other AWS services, so we'll install the CloudFormation CLI tools within a directory designed to house all of these toolsets, called `awstools`.

Note

You can name the directory anything you want. If you do this, however, remember to consistently substitute the location you've chosen for the `awstools` directory we'll use throughout this guide.

To prepare the install location

1. Open a Terminal window by doing one of the following:

   - Open the Windows Start menu, click **All Programs**, then **Accessories**, and finally **Command Prompt**.

- Open the Windows Start menu and type **`Command`** in the **Search Programs and Files** text box. Click the **Command Prompt** entry.

2. In your command prompt, create the installation directory by typing:

```
> mkdir %USERPROFILE%\awstools
```

   This will create a directory (if it doesn't already exist) called "awstools" in your home directory.

You can use `%USERPROFILE%\awstools` as the base directory for all AWS toolsets you want to install. Here, we'll concern ourselves only with installing the CloudFormation CLI tools.

To install the CLI tools

1. Change your working directory to %USERPROFILE%\awstools. You can do this on the command-line with:

```
> cd %USERPROFILE%\awstools
```

2. Move the unzipped AWSCloudFormation-#.#.# directory to the install location, by typing:

```
> move %USERPROFILE%/Downloads/AWSCloudFormation-1.0.9 .
```

Configure the CLI tools

Within the install directory (which can be accessed by typing `cd %USERPROFILE%\awstools\AWSCloudFormation-1.0.9` on the command-line prompt) you will find a file named `credential-file-path.template`. This file can be used to supply your AWS credentials to the tools automatically so you won't need to pass these credentials to the tools on the command-line every time you run them.

Adding your credentials to this file can save you time, but can also present a security risk if your system is used by others, since your security key will be stored as plain text on your system.

If you don't have your AWS credentials handy, you can view them by logging into the AWS console and selecting **Security Credentials** from the **My Account / Console** menu at the top of the screen. On the page, click **Access Credentials** to see your access key under the **Access Key ID** heading. Your secret key can also be viewed here by clicking the **show** button under the **Secret Access Key** heading.

Important

Do not share your secret access key with anyone! With your secret access key, anyone would be able to use AWS services and charge usage fees to your account.

If you don't want to keep your credentials in a file on your system, you can skip the next procedure, and instead pass your credentials directly to the AWS CloudFormation CLI tools with the `--I` *`ACCESSKEY`* and `--S` *`SECRETKEY`* arguments, replacing *ACCESSKEY* and *SECRETKEY* with your AWS credentials. You will need to do this every time you use the CLI tools.

To add your AWS credentials to the AWS credential file

1. Go to the AWS CloudFormation CLI tools install directory from the command-line, using:

   ```
   > cd %USERPROFILE%\awstools\AWSCloudFormation-1.0.9
   ```

   This assumes that you installed the CLI tools into `%USERPROFILE%\awstools`. If you did not, replace the location in the example above with the pathname to the location where you installed the CLI tools.
2. Copy or rename the credential-file-path.template file to a filename of your choosing (you can even use credential-file-path.template as the filename, but I prefer something that better describes its purpose). For the purposes of this article, we'll use "myAWSCredentials.txt":

   ```
   > copy credential-file-path.template myAWSCredentials.txt
   ```

3. Edit the file with your favorite text editor, enter your AWS credentials, then save and close the file. The file should contain two lines in it, like this:

   ```
   AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
   AWSSecretKey=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```

   Note: These are example values, and won't work for an actual connection to AWS. Instead, use your own AWS security credentials here.

Finally, you'll need to add a few environment variables to your system. Since you'll be using the CloudFormation CLI tools from the command-line, you can simply add these to your `%USERPROFILE%\profile.bat` file.

To add the CloudFormation CLI tools environment variables:

1. Edit `%USERPROFILE%\profile.bat` in your favorite editor. If it does not exist yet, create it.
2. In `%USERPROFILE%\profile.bat`, add the following lines:

   ```
   REM CloudFormation CLI tools setup
   set JAVA_HOME="C:\Program Files (x86)\Java\jre6"
   set AWS_CLOUDFORMATION_HOME="$HOME\awstools\AWSCloudFormation-1.0.9"
   set AWS_CREDENTIAL_FILE="$AWS_CLOUDFORMATION_HOME\myAWSCredentials.txt"
   set PATH="$PATH:$AWS_CLOUDFORMATION_HOME\bin"
   ```

   This assumes that you installed the CLI tools into `%USERPROFILE%\awstools`. If you did not, replace this location (where $HOME represents "%USERPROFILE%") with the pathname to the location where you installed the CLI tools.

   If you did not add your AWS access keys to an external file (such as the `myAWSCredentials.txt` file specified above), you should *not* assign the `AWS_CREDENTIAL_FILE` environment variable. Instead, remove or comment-out that line.
3. Create a shortcut that starts a command-line prompt and then loads your batchfile. To do this:

   1. Right-click your Windows desktop and choose **New**, then **Shortcut** in the context menu.

2. In the **Create Shortcut** dialog, type **`C:\Windows\System32\cmd.exe /k %USERPROFILE%\profile.bat`** and click **Next**.
3. Choose a name for your shortcut, such as **`AWS Command Prompt`**, and click **Finish**. This will create the shortcut on your Windows desktop.

4. To load your environment variables, quit your current command prompt session and then restart it by using the shortcut you just created.
5. Test your installation by running cfn-cmd:

```
> cfn-cmd
```

This will list all of the available CloudFormation CLI tools, their usage, and the available options for `cfn-cmd` itself (there are two options: `help` and `version`).

See Also

• Command Line Tools Reference (p. 340)

Using CloudFormer to Create AWS CloudFormation Templates from Existing AWS Resources

CloudFormer is a tool that enables you to create AWS CloudFormation templates from existing AWS resources in your account. With it, you can provision and configure your application resources the way you want using your existing processes and tools. Once everything is setup and you have the resources provisioned, simply take a "snapshot" of the configuration to create a template, enabling you to launch copies of the application with just a few clicks through the AWS Management Console.

The CloudFormer tool allows you to select any of the AWS resources running in your account for inclusion in the template. If you select resources that have dependent resources (for example, an Amazon EC2 instance is associated with an EC2 security group), the tool will automatically select any dependent resources. You can override any of the pre-selected resources or add other resources as necessary. You have full control of the resources that will be included in your template. Logical names are chosen based on the existing resource names, however, you can edit the names that will be included in the template and you can add output parameters based on the attributes available for any of the resources. The template can be uploaded to your S3 bucket and launched directly via the AWS CloudFormation stack creation wizard.

CloudFormer is a standalone application that you can launch inside your AWS environment. The application is started on a t1.micro Amazon EC2 instance via AWS CloudFormation. No other AWS resources are required to run CloudFormer.

Launch CloudFormer

You can launch CloudFormer in a number of different ways, such as by using the URLs provided on the CloudFormer tool page, on the AWS CloudFormation Templates page, or by using the AWS Management Console. Because the AWS Management Console is a good way to learn about the process of working with AWS resources, we will use the Console to launch CloudFormer.

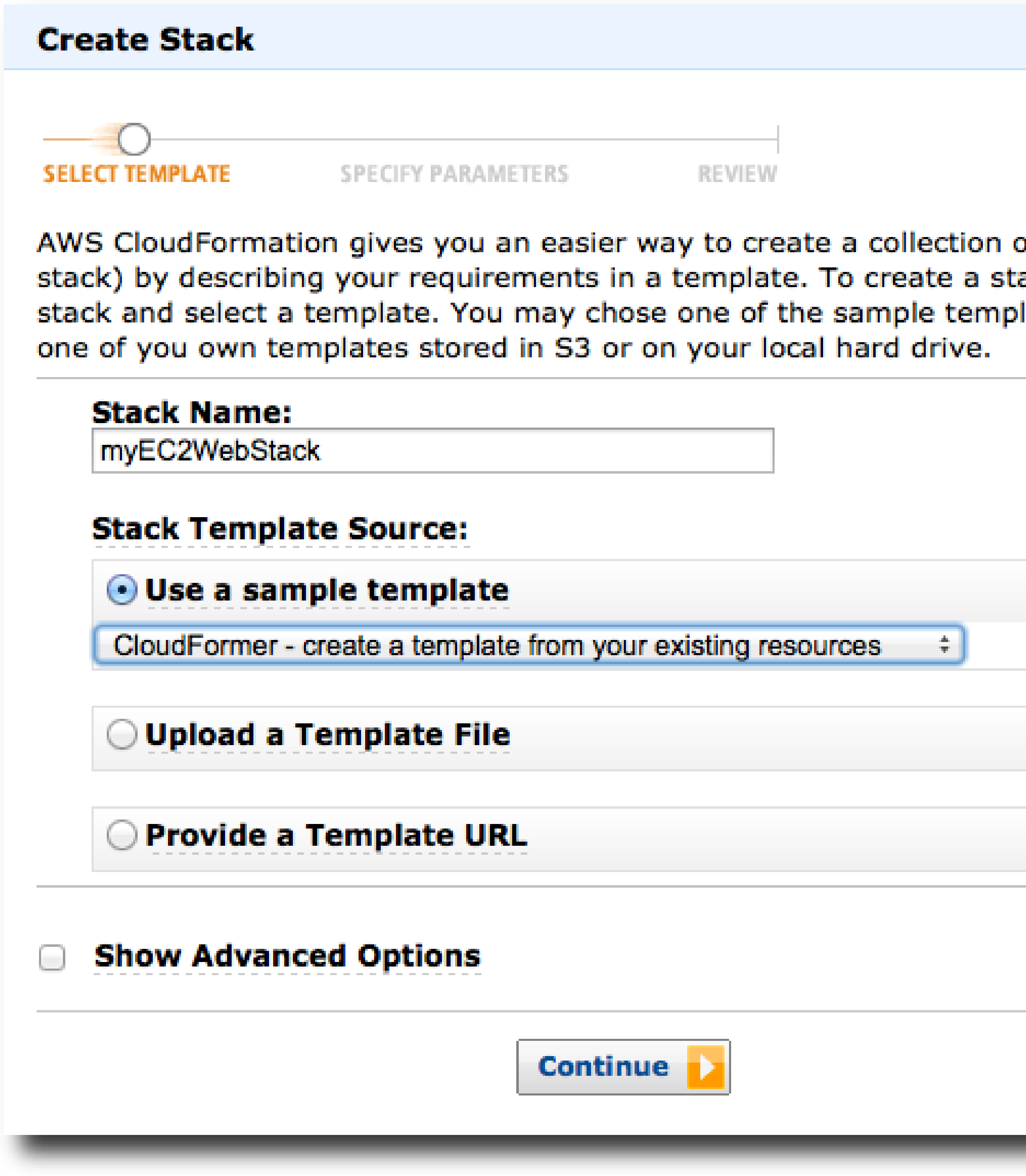
Note

This guide will use a specific scenario, building a simple website on an EC2 instance. Although this example is simple, it will use a number of AWS resources to show you how to work with these in the CloudFormer tool. Keep in mind, however, that this is merely one example of infinite possibilities, since CloudFormer can be used to create a template from any collection of AWS resources that you've created.

To launch CloudFormer with the AWS Management Console

1. Log in to the AWS CloudFormation Console. If you have not done this before, see Logging in to the AWS CloudFormation Console for instructions.
2. Click **Create New Stack**, which will bring up the **Create Stack** dialog.

3. On the **Create Stack** dialog, name your stack, then click **Use a Sample Template** and select **CloudFormer - create a template from your existing resources** in the drop-down list.



Create Stack

AWS CloudFormation gives you an easier way to create a collection o
stack) by describing your requirements in a template. To create a sta
stack and select a template. You may chose one of the sample templ
one of you own templates stored in S3 or on your local hard drive.

Stack Name: myEC2WebStack

Stack Template Source:

(●) Use a sample template
CloudFormer - create a template from your existing resources

( ) Upload a Template File

( ) Provide a Template URL

[ ] Show Advanced Options

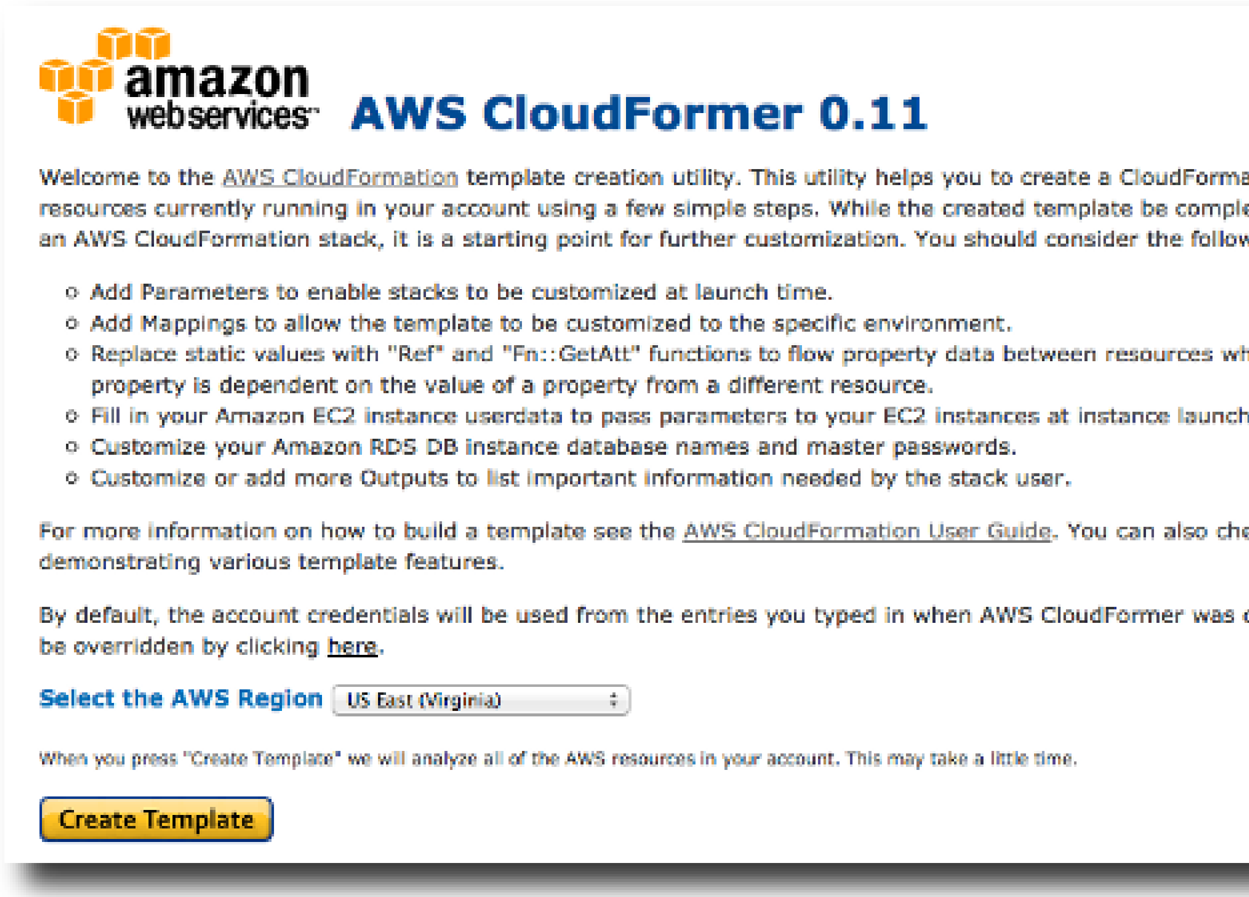
4. Click **Continue** to move to the **Specify Parameters** page.

   The CloudFormer stack runs on a t1.micro instance for lowest cost. There are no parameters for this stack.

5. On the **Specify Parameters** page, check the box labeled "I acknowledge that this template may create IAM resources" and click **Continue** to move to the **Review** page.
6. On the **Review** page, examine the information about the stack that will be created, then click **Continue** to begin creating the CloudFormer stack.

   Note: Since CloudFormer is an AWS CloudFormation stack itself, it will go through the normal stack creation process, from `CREATE_IN_PROGRESS` to `CREATE_COMPLETE`.
7. Once the CloudFormer stack's state is `CREATE_COMPLETE`, click its entry in your AWS CloudFormation Management Console, and select the **Outputs** tab in the stack information pane.
8. In **Outputs**, click the URL to begin using the CloudFormer tool.

   The AWS CloudFormer interface will be displayed.



AWS CloudFormer 0.11

Welcome to the AWS CloudFormation template creation utility. This utility helps you to create a CloudForma
resources currently running in your account using a few simple steps. While the created template be comple
an AWS CloudFormation stack, it is a starting point for further customization. You should consider the follow

- Add Parameters to enable stacks to be customized at launch time.
- Add Mappings to allow the template to be customized to the specific environment.
- Replace static values with "Ref" and "Fn::GetAtt" functions to flow property data between resources wh property is dependent on the value of a property from a different resource.
- Fill in your Amazon EC2 instance userdata to pass parameters to your EC2 instances at instance launch
- Customize your Amazon RDS DB instance database names and master passwords.
- Customize or add more Outputs to list important information needed by the stack user.

For more information on how to build a template see the AWS CloudFormation User Guide. You can also che
demonstrating various template features.

By default, the account credentials will be used from the entries you typed in when AWS CloudFormer was c
be overridden by clicking here.

Select the AWS Region: US East (Virginia)

When you press "Create Template" we will analyze all of the AWS resources in your account. This may take a little time.

Create a Template

The next step is to begin the template creation process. You will likely want to use your own AWS resources, but for the purpose of the tutorial, we will assume that the following resources have been created:

- AWS::EC2::EIP
- AWS::EC2::Instance

- AWS::EC2::SecurityGroup
- AWS::IAM::AccessKey

Note

CloudFormer will automatically resolve dependencies for you when you have AWS Resources that are associated with your EC2 Instance. These will show up as pre-selected options when you are selecting resources to add to your template.

To use CloudFormer to create a template from your AWS resources

1. Select the region in which you'd like to create your template, and click **Create Template**.

   You'll move to the **Intro** page, which allows you to enter a description for your template and select resources with a filter or by selecting all resources in your account. We'll hand-pick the resources, so leave those fields blank and unchecked, respectively.
2. On the **Intro** page, enter a description for your template and click **Continue**.

   This will bring you to the **DNS Names** page. If you have any Route53 DNS records that are used by your instance, you can select them here. We don't have any, so we'll skip this screen.
3. On the **DNS Names** page, click **Continue**.

   This will bring you to the **Network Resources** page. Here, you can select any Amazon EC2 Elastic IP addresses, Elastic Load Balancers or CloudFront distributions. We have an Elastic IP Address, so we'll add it here.
4. Select the Amazon EC2 Elastic IP address that you'd like to include in your template, and click **Continue**.

   This will bring you to the **Compute Resources** page. You can select any Auto Scaling groups or Amazon EC2 Instances to include in your template. If you have already associated the Elastic IP Address with an EC2 instance, that instance will be pre-selected here.
5. Make sure that your EC2 instance is selected, and click **Continue**.

   This will bring you to the **System Configuration** page. Here, you can select any AWS AutoScaling Launch Configurations to add them your template.
6. On the **System Configuration** page, click **Continue**.

   This brings you to the **Storage** page, which allows you to add Amazon Elastic Block Storage (EBS) volumes, Amazon RDS Database Instances and S3 Buckets to your template.
7. On the **Storage Page**, click **Continue**.

   This will bring you to the **Security Groups** page, where you can select any Amazon EC2 or RDS security groups to add to your template. If you have associated any security groups with your EC2 instance, they will be pre-selected here.
8. Make sure that your EC2 security group is selected, and click **Continue**.

   This takes you to the **Application Services** page (called "Other" on the CloudFormer timeline), where you can select any SQS Queues or Amazon SimpleDB Domains to add to your template.
9. On the **Application Services** page, click **Continue**.

   This brings you to the **Operational Resources** page, which allows you to add any auto-scaling triggers that you have set up. We have none, so we'll skip this step.
10. on the **Operational Resources** page, click **Continue**.

    This brings you to the **Summary** page, which serves a number of purposes:

- It is a chance to review the resources you've added to your template, and to go back and re-select them if you want.
- It gives you an opportunity to change the auto-generated logical resource names for your resources.
- It gives you an opportunity to specify outputs that will provide necessary information, such as your site's IP address or URL, the Availability Zone, or other bits of information that will be useful for configuring and using your stack.

11. On the **Summary** page, examine the resources you've selected. You should have one Amazon Elastic IP Address, one Amazon EC2 instance, and one Amazon EC2 Security Group.

    To rename a resource with a logical name you specify, type its name in the **Logical Name** box. To choose an **Output** field to expose to the **Outputs** tab of the stack information pane, click the outputs here.
12. On the **Summary** page, verify your selections, change the logical names of your resources, and add Outputs. When you are satisfied with the changes you've made, click **Continue**.

    The next page shows you the template that was generated for your resources. Using this template will create a stack that contains all of the resources you selected. You can use this template to deploy your resources as a combined set with AWS CloudFormation, or you can use it as a base template for further modification.

    You can also save your stack directly to an S3 bucket by clicking the **Save Stack** button, so you can launch the stack from the URL by entering it in the **Provide a Template URL** field when creating a new stack.
13. Click **Save** to save the stack to S3. This action also gives you the option to launch an instance of the stack immediately. If you do this, the S3 URL will be automatically selected to create the stack.

Note

Once your template has been created, you can (and should) delete your CloudFormer stack to minimize any charges to your AWS account.

Using the AWS CloudFormation Console

The AWS CloudFormation console allows you to create, monitor, update and delete stacks directly from your web browser. This section contains guidance on using the AWS CloudFormation console to perform common actions.

Topics

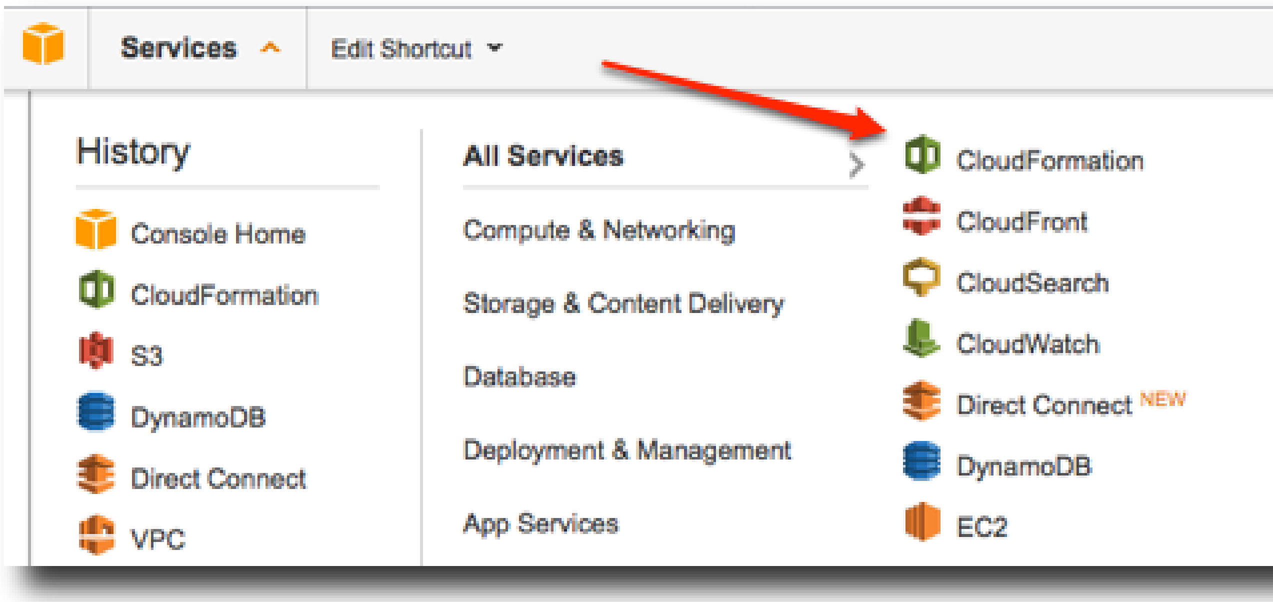
Logging In to the AWS CloudFormation Console

The AWS CloudFormation console allows you to create, monitor, update and delete your CloudFormation stacks with a web-based interface. It is part of the AWS Management Console.

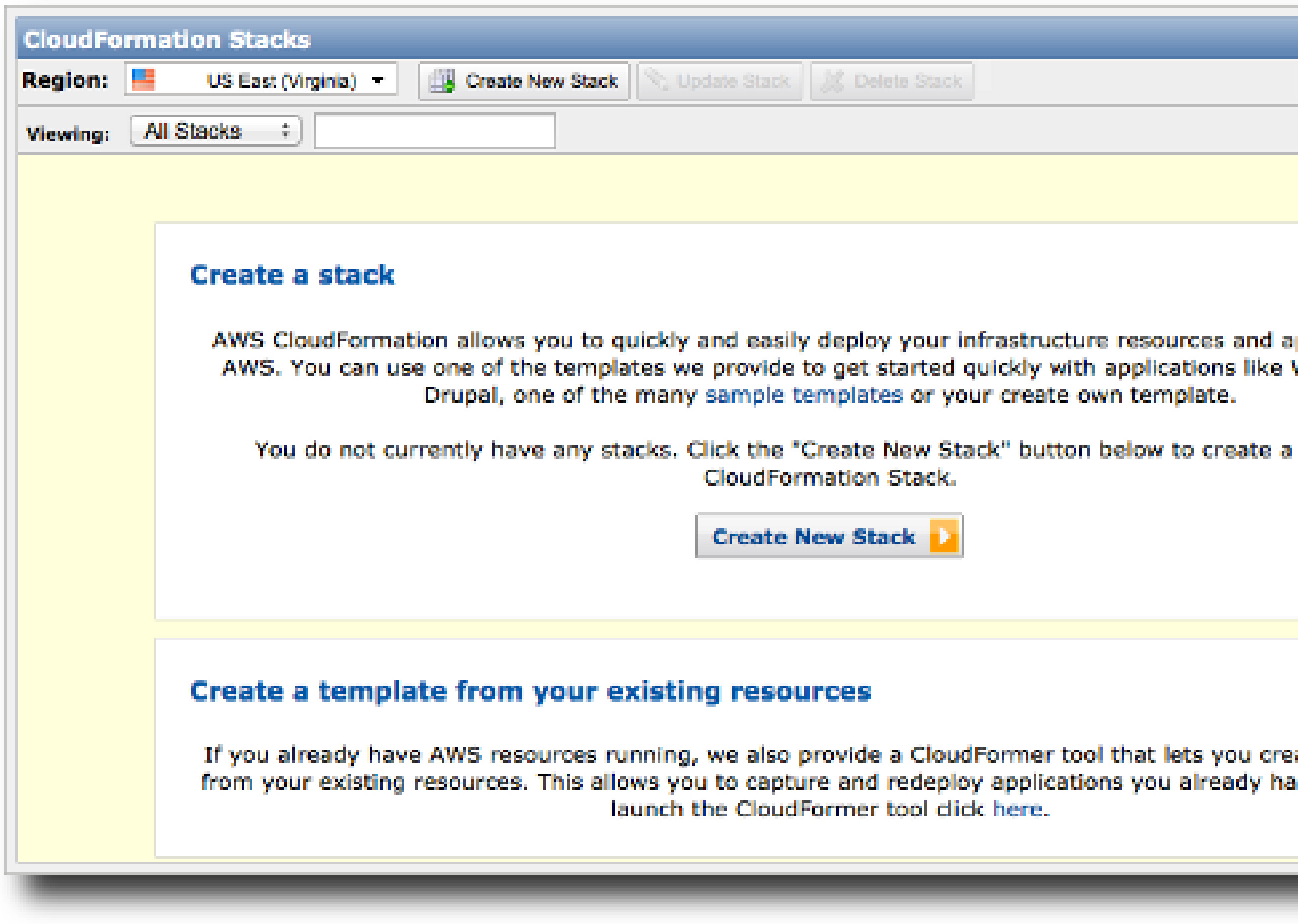
You can access the AWS CloudFormation Console in a number of ways:

- Open the AWS CloudFormation Console directly with the URL https://console.aws.amazon.com/cloudformation/. If you are not logged in to the AWS Management Console yet, you will need to log in before using the AWS CloudFormation Console.

- If you are logged into and using the AWS Management Console, you can access the AWS CloudFormation console by opening the **Services** menu and click **AWS CloudFormation** from the **All Services** submenu or the **Deployment and Management** submenu.



If you don't have any AWS CloudFormation stacks running, you will be presented with the option to **Create a stack**. Otherwise, you will see a list of your currently-running stacks.



Create a stack

AWS CloudFormation allows you to quickly and easily deploy your infrastructure resources and a
AWS. You can use one of the templates we provide to get started quickly with applications like
Drupal, one of the many sample templates or your create own template.

You do not currently have any stacks. Click the "Create New Stack" button below to create a
CloudFormation Stack.

Create New Stack

Create a template from your existing resources

If you already have AWS resources running, we also provide a CloudFormer tool that lets you crea
from your existing resources. This allows you to capture and redeploy applications you already ha
launch the CloudFormer tool click here.

Estimating the Cost of Your AWS CloudFormation Stack

There is no additional charge for AWS CloudFormation. You pay for AWS resources (e.g. Amazon EC2 instances, Elastic Load Balancing load balancers and so on) created using AWS CloudFormation as if you created them by hand.

We recommend that you use the Amazon Web Services Simple Monthly Calculator at http://calculator.s3.amazonaws.com/calc5.html to estimate the cost of your stacks beforehand. You can also estimate the cost of your stacks while creating or updating a stack from the AWS CloudFormation console.

To estimate the cost of your stack

1. On the **Review** page of the **Create Stack** or **Update Stack** dialog, click the **Cost** link.

   This will open the **AWS Simple Monthly Calculator** in a new browser page (or tab, depending on how your browser is set up).

> **Note**
>
> Because you launched the calculator from the AWS CloudFormation console, it will be pre-populated with your template configuration and parameter values. There are many additional configurable values that can provide you with a better estimate if you have an idea of how much data transfer you expect to your Amazon EC2 instance. For more information about what factors contribute to the total cost of your AWS resources, refer to the AWS Pricing Overview PDF, which can also be accessed by clicking the help link at the top of the calculator page.

2. Click the **Estimate of your Monthly Bill** tab for a monthly estimate of running your stack, along with a categorized display of what factors contributed to the estimate.

Creating an EC2 Key Pair

The use of some AWS CloudFormation resources and templates will require you to specify an EC2 key pair for authentication, such as when you are configuring SSH access to your instances.

Amazon EC2 key pairs can be created with the AWS Management Console by using the following procedure.

To create an EC2 key pair

1. In the AWS Management Console, switch from the AWS CloudFormation console to the EC2 console by clicking the Services button in the top-left corner of the screen, and select **EC2**.

   The console display will now show the Amazon EC2 Console Dashboard.
2. In the EC2 Console, in the **Navigation** pane, click **Key Pairs**.

   You will see the **Key Pairs** page, displaying your EC2 key pairs. If you haven't created any yet, the list will be empty, and will, instead, show a button called **Create Key Pair**.
3. Click the **Create Key Pair** button.
4. Type a key pair name, and click **Create**. It doesn't matter what you name it, but make it something you can easily remember.

   The key pair will be created, and the download of your private key will begin. It will be called *`name`*.pem, where *name* represents the name you gave to your key pair.
5. Download the key pair, and set the permissions to 400 (on a Linux or Mac OS system).

Tagging an AWS CloudFormation Stack

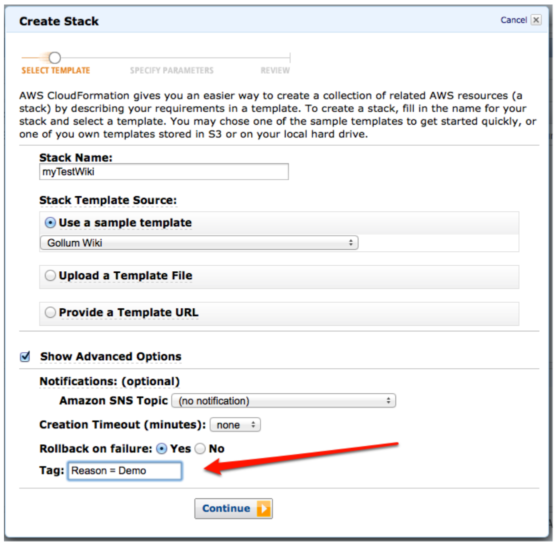
When creating a stack, you can add arbitrary key/value tags to identify it for purposes such as cost allocation. For complete information about what tags are and how they can be used, see Tagging Your Resources in the *Amazon EC2 User Guide*.

Adding a Tag

This procedure will demonstrate how you can add an arbitrary key/value pair to tag a stack, using the AWS CloudFormation console.

To add a tag to an AWS CloudFormation stack

1. On the **Create Stack** dialog, click **Show Advanced Options**.



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 chose one of the sample templates to get started quickly, or one of you own templates stored in S3 or on your local hard drive.

Stack Name: myTestWiki

Stack Template Source:
[✓ Use a sample template] Gollum Wiki
Upload a Template File
Provide a Template URL

✓ Show Advanced Options

Notifications: (optional)
Amazon SNS Topic (no notification)
Creation Timeout (minutes): none
Rollback on failure: [✓ Yes] No
Tag: Reason = Demo

2. Add the tag for your stack in the **Tag** text box, using the following syntax, where `myTagKey` and `myTagValue` are replaced by an arbitrary key name and value that you specify:

   `myTagKey` = *`myTagValue`*

   Note

   Although you are free to choose your own key name and value, key names and values prefixed with **aws:** are reserved for Amazon Web Services. You cannot create tags prefixed with **aws:**

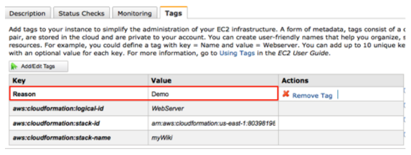
3. Click **Continue** to proceed with creating your AWS CloudFormation stack.

Once your stack been created with tags, you can view your tags with the EC2 console.

Viewing Your Tags

To view tags for your AWS CloudFormation stack

1. Click on your stack in the AWS CloudFormation console. This will show you the stack detail pane.
2. Click the **Resources** tab to view your stack's resources. Find the AWS::EC2::Instance resource and make note of its physical ID.
3. In the **Services** menu, click **EC2** to open the EC2 Console Dashboard.
4. In the **Navigation** pane, click **Instances**.
5. Find the EC2 instance with the physical ID that matches your stack's EC2 instance, and click it to show the EC2 instance detail pane.
6. In the detail pane, click the **Tags** tab to view the tags associated with your instance. The tag you added to your stack should be among them.



Add tags to your instance to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a c
pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, s
resources. For example, you could define a tag with key = Name and value = Webserver. You can add up to 10 unique ke
with an optional value for each key. For more information, go to Using Tags in the *EC2 User Guide*.

Key	Value	Actions
Reason	Demo	Remove Tag
aws:cloudformation:logical-id	*WebServer*	
aws:cloudformation:stack-id	*arn:aws:cloudformation:us-east-1:80398198*	
aws:cloudformation:stack-name	*myWiki*	

Note

You might also notice some **aws:**-prefixed tags. These were automatically added to your instance by AWS, and cannot be deleted or modified.

Related Topics

- Using the AWS CloudFormation Command Line Interface (p. 83)

Using the AWS CloudFormation Command Line Interface

The AWS CloudFormation command line interface (CLI) allows you to create, monitor, update and delete stacks from your system's terminal. This section contains guidance on using the CLI to perform actions related to managing your AWS CloudFormation stacks.

Important

Before you can use the AWS CloudFormation CLI, you must first download and install it. If you have not done this already, refer to Installing the AWS CloudFormation Command Line Interface (CLI) (p. 59) for instructions.

In This Section

Describing and Listing Your Stacks

AWS CloudFormation provides two commands, `cfn-describe-stacks` and `cfn-list-stacks`, that enable you to get information about your stacks.

cfn-describe-stacks

The `cfn-describe-stacks` command provides information on your running stacks, and provides the option of filtering on the stack name. It returns information about the stack, including the name, stack identifier, and status. For example, suppose you have created three separate stacks on the same template. After they are created, the `cfn-describe-stacks` command will return output similar to the following:

```
PROMPT> cfn-describe-stacks

STACK  my-stack-test-01  CREATE_COMPLETE
IPAddress=50.17.250.145;InstanceId=i-13b3207f  2011-02-04T00:39:06Z

STACK  my-stack-test-02  CREATE_COMPLETE
IPAddress=50.17.250.160;InstanceId=i-1fb22173  2011-02-04T00:40:53Z

STACK  my-stack-test-03  CREATE_COMPLETE
IPAddress=50.17.250.161;InstanceId=i-fbb22197  2011-02-04T00:41:05Z
```

If you know the stack name, you can use `--stack-name` to limit the output to one stack. Otherwise, information on all your running stacks is returned.

cfn-list-stacks

The `cfn-list-stacks` command enables you to get a list of any of the stacks you have created (even those which have been deleted up to 90 days), and provides you the option to filter the results by stack status. The `cfn-list-stacks` command returns summary information about any of your running or deleted stacks, including the name, stack identifier, template, and status.

Note

The cfn-list-stacks command returns information on deleted stacks for 90 days after they have been deleted.

The `cfn-list-stacks` command returns output similar to the following:

```
cfn-list-stacks --stack-status CREATE_COMPLETE
STACK  STACK_ID
NAME           CREATION_TIME         STATUS

STACK
arn:aws:cloudformation:us-east-1:123456789012:stack/mut2/83327aa0-863f-11e0
mut2           2011-05-24T19:53:36Z  CREATE_COMPLETE

STACK  arn:aws:cloudformation:us-east-1:123456789012:stack/mut1/5d569960-8635-
11e0
mut1           2011-05-24T18:40:57Z  CREATE_COMPLETE
```

Viewing a Stack's Event History

It takes time to create and delete a stack's member resources. The amount of time depends on the complexity of your stack. You can track the status of the resources AWS CloudFormation is creating and deleting with the `cfn-describe-stack-events` command.

In the following example, a sample stack is created. The events are checked several times, showing the change in state of the stack's member resources as they are created.

First, the stack creation begins with the `cfn-create-stack` command:

```
PROMPT> cfn-create-stack MyCustomStack02
-f MyCustomAlarms.template
--parameters "Email=valid-email-address"
```

```
arn:aws:cfn:us-east-1:165024647323:stack/d835bc30-2ff8-11e0-a09f-
5017c2aa8c86/MyCustomStack02
```

The `cfn-describe-stack-events` command is run within 10 seconds of the response from `cfn-create-stack`.

```
PROMPT> cfn-describe-stack-events MyCustomStack02

STACK_EVENT  MyCustomStack02  MyInstance        AWS::EC2::Instance
2011-02-04T00:51:07Z  CREATE_IN_PROGRESS

STACK_EVENT  MyCustomStack02  MyEventStack02  AWS::CloudFormation::Stack
2011-02-04T00:51:04Z  CREATE_IN_PROGRESS  User Initiated
```

You can opt for a full listing with the *`--show-long`* option:

```
PROMPT> cfn-describe-stack-events MyEventStack02 --show-long

STACK_EVENT,
db20a720-2ff8-11e0-9db9-5081c39d0e19,
MyEventStack02,arn:aws:aws21:us-east-1:165024647323:stack/d835bc30-2ff8-11e0-
a09f-5017c2aa8c86/MyEventStack02,
MyInstance,
MyEventStack02-MyInstance-ZUJ095B3FP66,
"{"AvailabilityZone":"us-east-1a","ImageId":"ami-20b65349"}",
AWS::EC2::Instance,2011-02-04T00:51:07Z,CREATE_IN_PROGRESS,(nil)

STACK_EVENT,
d847bd90-2ff8-11e0-a09f-5017c2aa8c86,
MyEventStack02,arn:aws:aws21:us-east-1:165024647323:stack/
   d835bc30-2ff8-11e0-a09f-5017c2aa8c86/MyEventStack02,
MyEventStack02,arn:aws:aws21:us-east-1:165024647323:stack/
   d835bc30-2ff8-11e0-a09f-5017c2aa8c86/MyEventStack02,
{"Email":"tom@fictitious.corp"},
AWS::CloudFormation::Stack,2011-02-04T00:51:04Z,CREATE_IN_PROGRESS,User Initiated
```

The long report lists all the available history data on the two events. The most recent events are reported first. The following data points are returned in the long report:

Field	Description
`EVENT_ID`	Event identifier
`STACK_NAME`	Name of the stack that the event corresponds to
`STACK_ID`	Identifier of the stack that the event corresponds to
`LOGICAL_ID`	Logical identifier of the resource
`PHYSICAL_ID`	Physical identifier of the resource
`RESOURCE_PROPERTIES`	Properties of the resource

Field	Description
RESOURCE_TYPE	Type of the resource
EVENT_TIME	Time when the event occurred
RESOURCE_STATUS	Resource status; one of *IN_PROGRESS* | *CREATE_FAILED* | *CREATE_COMPLETE* | *DELETE_IN_PROGRESS* | *DELETE_FAILED* | *DELETE_COMPLETE*.
RESOURCE_STATUS_REASON	More information on the status

Listing Member Resources

Immediately after you run the `cfn-create-stack` command, you can list its resources using cfn-list-stack-resources (p. 356). This command lists a summary of each resource in the stack you specify with the `--stack-name` parameter. The report includes a summary of the stack, including the creation or deletion status.

For example, suppose you have sample stack with the name *MyEventStack02*. You list the resources for the stack as follows:

```
PROMPT> cfn-list-stack-resources --stack-name MyEventStack02

STACK_RESOURCE  MyInstance    i-67a6050b
AWS::EC2::Instance      2011-01-11T06:24:21Z  CREATE_COMPLETE

STACK_RESOURCE  MyAlarmTopic
arn:aws:sns:us-east-1:165024647323:MyEventStack02-MyAlarmTopic-8RFXAXGNW520/My
EventStack02
AWS::SNS::Topic         2011-01-11T06:24:37Z  CREATE_COMPLETE

STACK_RESOURCE  CPUAlarm      MyEventStack02-CPUAlarm-PKOIPR1BE050
AWS::CloudWatch::Alarm  2011-07-11T06:24:56Z  CREATE_COMPLETE
```

AWS CloudFormation will report resource details on any running or deleted stack. If you specify the name of a stack whose status is *CREATE_IN_PROCESS*, AWS CloudFormation will report only those resources whose status is *CREATE_COMPLETE*.

Note

The cfn-describe-stack-resources command returns information on deleted stacks for 90 days after they have been deleted.

Retrieving a Template

AWS CloudFormation stores the template you use to create your stack as part of the stack. You can download the template from AWS CloudFormation using the `cfn-get-template` command. Even if a stack has been deleted, you can download its template.

Note

The command returns the deleted stacks templates for up to 90 days after the stack has been deleted.

For example, suppose you have created a stack named *`MySQSTemplate`* which creates a simple SQS queue. The following shows what could be returned by `cfn-get-template`:

```
PROMPT> cfn-get-template MySQSTemplate

TEMPLATE "{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "MyInstance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "UserData" : {
                    "Fn::Base64" : {
                      "Fn::Join" : [ "", [ "Queue=", { "Ref" : "MyQueue" }]]
                    }
                },
                "AvailabilityZone" : "us-east-1a",
                "ImageId" : "ami-20b65349"
            }
        },
        "MyQueue" : {
            "Type" : "AWS::SQS::Queue",
            "Properties" : {
            }
        }
    },
    "Outputs" : {
        "QueueName" : {
            "Value" : { "Ref" : "MyQueue" }
        }
    }
}
"
```

The output contains the entire template body, enclosed in quotation marks.

Validating a Template

To check your template file for syntax errors, you can use the `cfn-validate-template` command.

Note

The `cfn-validate-template` command is designed to check only the syntax of your template. It does not ensure that the property values you have specified for a resource are valid for that resource. Nor does it determine the number of resources that will exist when the stack is created.

To check the operational validity, you need to attempt to create the stack. There is no sandbox or test area for AWS CloudFormation stacks, so you are charged for the resources you create during testing.

You can validate templates locally (using the *`--template-file`* parameter), or with the *`--template-url`* parameter. For example, the following command validates a sample template, from its remote location:

```
PROMPT> cfn-validate-template --template-url
https://www.fictitious.us/templates/sample.template
--region US-West-1
```

```
PARAMETERS  PARAMETER_NAME    DEFAULT_VALUE  NOECHO
PARAMETERS  Email                            false
PARAMETERS  AccessKey                        true
PARAMETERS  VolumeSize        1              false
PARAMETERS  SnapshotId        snap-91f774fb  false
PARAMETERS  SecretKey                        true
PARAMETERS  AvailabilityZone  us-east-1a     false
PARAMETERS  ServerPort        9418           false
PARAMETERS  KeyName                          false
PARAMETERS  InstanceType      m1.small       false
```

The expected result is no error message, with information about all parameters listed.

Suppose you were editing a local copy of the sample, and you accidentally removed the comma after the *`ServerPort`* parameter. Trying to validate it again would result in the following output:

```
cfn-validate-template:
Malformed input-Template format error: JSON not well-formed.
(line 11, column 6)
```

Related Topics

- Using the AWS CloudFormation Console (p. 77)

Working with AWS CloudFormation Templates

Topics

The key to getting the most out of AWS CloudFormation is a thorough understanding of templates.

To get you started quickly on modifying and authoring templates, this section provides template anatomy details, example templates and template snippets. This section also discusses how to modify and validate templates.

- In Template Anatomy (p. 90), we provide the technical details for coding each of the template objects.
- In Template Snippets (p. 109), we provide a number of template sections that demonstrate how to write the JSON code for a particular section of a template. In this section you'll find starter snippets for Amazon EC2 instances, Amazon S3 domains, AWS CloudFormation mappings, and more. The snippets are selected to cover a range of resources and properties you are likely to include often in your templates. They are grouped by the resources they would be used to declare, with general-purpose AWS CloudFormation snippets in AWS CloudFormation Template Snippets (p. 145)).
- The section Example Templates (p. 98) contains a number of sample templates that will create stacks with little or no modification. The samples range in complexity, and highlight the use of AWS CloudFormation template features in the context of a complete application. Some of the templates require you to specify values in the command's `--parameters` option.

For details about the supported resources, type names, intrinsic functions, and pseudo parameters you can use in your templates, see the Template Reference (p. 187) section.

Template Anatomy

Topics

This section describes the parts of a template in detail.

Note

After you have a template written, you can validate its syntactic correctness using `cfn-validate-template`. For more information, see Validating a Template (p. 87).

Template Declaration

Templates have six major sections, each separated by a comma. The *`Resources`* section is required. The rest are optional. There is no required ordering of the sections within the template.

The first character in the template must be an open brace, "{". The last character must be a closed brace, "}". The sections are delimited by a comma.

The following template fragment shows the template structure and sections. Again, these sections may appear in the template in any order.

```
{

    "AWSTemplateFormatVersion" : "version date",

    "Description" : "Valid JSON strings up to 4K",

    "Parameters" : {
        set of parameters
    },

    "Mappings" : {
            set of mappings
            },

    "Resources" : {
        set of resources
     },

    "Outputs" : {
        set of outputs
    }
}
```

Further information about each of the sections is as follows.

Template Format Version Declaration

The template format version is an optional declaration that identifies the capabilities of the template format. Because the AWS CloudFormation template format has not changed, there is currently only one value supported: .

The value for the template format version declaration must be a literal string. You cannot base its value on a parameter or function. If not present, AWS CloudFormation assumes the latest template format version. Here is an example of a valid template format version declaration:

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

Note

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

Template Description Declaration

The template Description section is optional. When it is present, it *must* follow the AWSTemplateFormatVersion section.

The template Description enables you to provide arbitrary comments about your template. It is a literal string between 0 and 4K in length. It is composed of a double-quoted key name (*`Description`*) and a double-quoted string value. The key and value are separated by a single colon. The following depicts a valid Description section declaration:

```
"Description" : "Contains an autoscaling group and a load balancer.",
```

The value for the Description section must be a literal string. It must be a valid JSON string. You cannot base its value on a parameter or function.

Parameters Declaration

The optional Parameters section enables you to pass values into your template at stack creation time. Parameters allow you to create templates that can be customized for each stack deployment. When a user creates a stack from a template containing parameters, the user can specify values for those parameters. Within the template, you can use the "Ref" intrinsic function to specify those parameter values in properties values for resources. For example, you can define a string parameter with the following Parameters section:

```
"Parameters" : {
    "URL" : {
      "Type" : "String"
    }
    }
```

At runtime, you can use the `--parameters` option of `cfn-create-stack to set the URL parameter to a specific value:`

```
cfn-create-stack MyStack --template-file My.template --parameters "URL=127.0.0.1"
```

Note that multiple parameters assignments are separated with a semicolon.

Parameters can have rules, called constraints, that determine valid values for a parameter. These constraint properties allow you to validate the template user's input before any resources are created. For example, you can have a constraint that validates that a string contains only alphanumeric characters or that a numeric parameter is between 1 and 10.

Parameter Syntax and Properties

The Parameters section is composed of the key name *`Parameters`*, followed by a single colon. Braces enclose all parameter declarations. Parameters declared within the Parameters section are delimited with a comma.

Each parameter must declare a double-quoted name, followed by a colon, and its properties, enclosed within a single set of braces, and delimited with a comma. Parameter names are required to be alphanumeric and must be unique among all logical names within a template.

A parameter can be declared as one of following types: *`String`*, *`Number`*, or *`CommaDelimitedList`*. For a parameter that has a *`String`* or *`Number`* type, you can define constraints that AWS CloudFormation uses to validate the value of the parameter.

Parameters have the following properties:

Property	Required	Description
Type	Yes	String, Number, or CommaDelimitedList. A parameter of type *`String`* is simply a literal string. A parameter of type *`Number`* can be an integer or float. Note that AWS CloudFormation validates the parameter as a number but uses the parameter value within the template as a string. A parameter of type *`CommaDelimitedList`* is an array of literal strings separated by commas. The member strings are space trimmed and there is one more string than there are commas in the specified value.
Default	No	A value of the appropriate type for the template to use if no value is specified at stack creation. If the parameter has constraints defined, this value must adhere to those constraints.
NoEcho	No	If *`TRUE`*, the value of the parameter is masked with asterisks (*****) with `cfn-describe-stacks`.
AllowedValues	No	An array containing the list of values allowed for the parameter.
AllowedPattern	No	*`String`* constraint. A regular expression that represents the patterns allowed in the parameter's string value.
MaxLength	No	*`String`* constraint. A integer value that determines the maximum number of characters in the parameter's string value.
MinLength	No	*`String`* constraint. A integer value that determines the minimum number of characters in the parameter's string value.

Property	Required	Description
MaxValue	No	*Number* constraint. A numeric value that determines the largest numeric value allowed for the parameter.
MinValue	No	*Number* constraint. A numeric value that determines the smallest numeric value allowed for the parameter.
Description	No	A String type up to 4000 characters describing the parameter.
ConstraintDescription	No	A String type that describes the constraint requirements when the user specifies a parameter value that does not match the constraints defined for the parameter. For example, a parameter that has an AllowedPattern of "[A-Za-z0-9]+" would display this error message when the user specified an invalid value: `cfn-create-stack:  Malformed input-Parameter MyParameter must match pattern [A-Za-z0-9]+` By adding a ConstraintDescription with a value "must only contain upper- and lowercase letters, and numbers", you can display a customized error message: `cfn-create-stack:  Malformed input-Parameter MyParameter must only contain upper and lower case letters and numbers`

Parameter Examples

The following example *Parameters* section declares two parameters. The DBPort parameter is of type `Number` with a default of 3306 and a minimum value of 1150 and maximum value 65535. The DBPwd parameter is of type `String` with no default value, minimum length of 1, maximum length of 41, and pattern that allows lowercase and uppercase alphabetic characters and numerals. The UserRoles parameter is a `CommaDelimitedList` with a default list containing the string values guest and newhire.

```
"Parameters" : {
    "DBPort": {
      "Default": "3306",
      "Description" : "TCP/IP port for the database",
      "Type": "Number",
      "MinValue": "1150",
      "MaxValue": "65535"
    },
    "DBPwd": {
      "NoEcho": "true",
      "Description" : "The database admin account password",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*"
    },

    "UserRoles" : {
      "Type" : "CommaDelimitedList",
      "Default" : "guest,newhire",
      "NoEcho" : "TRUE"
    }
}
```

Mappings Declaration

Mappings match 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.

Mappings are declared in the *`Mappings`* section, with each mapping separated by a comma. The keys and values in mappings must be literal strings. Each mapping is formatted as a double-quoted key, a single colon, and braces enclosing the sets of values to map. The following example shows a *`Mappings`* section containing a single mapping named *`Mapping01`*.

```
"Mappings" : {
    "Mapping01" : {
        "Key01" : {
            "Value" : "Value01"
        },
        "Key02" : {
            "Value" : "Value02"
        },
        "Key03" : {
            "Value" : "Value03"
        }
    }
}
```

Within a mapping, each map is a key followed by a single colon and a set of name-value pairs surrounded by braces. The key identifies each map, and it must be unique within the mapping. Within the braces, you can declare multiple name-value pairs. Each pair is formatted as a double-quoted name, a single colon, and a value or an array of values.

The following example shows a Mappings section with a map *`RegionMap`*, which contains five keys that map to name-value pairs containing single string values. The keys are region names. Each name-value pair is the AMI ID for the 32-bit AMI in the region represented by the key.

```
"Mappings" : {
      "RegionMap" : {
        "us-east-1" : { "32" : "ami-6411e20d"},
        "us-west-1" : { "32" : "ami-c9c7978c"},
        "eu-west-1" : { "32" : "ami-37c2f643"},
        "ap-southeast-1" : { "32" : "ami-66f28c34"},
        "ap-northeast-1" : { "32" : "ami-9c03a89d"}
      }
}
```

The name-value pairs have a name (32 in the example) and a value. By naming the values, you can map more than one set of values to a key. The following example has region keys that are mapped to two sets of values: one named 32 and the other 64.

```
"RegionMap" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" },
      "ap-southeast-1" : { "32" : "ami-66f28c34", "64" : "ami-60f28c32" },
      "ap-northeast-1" : { "32" : "ami-9c03a89d", "64" : "ami-a003a8a1" }
}
```

You can use the Fn::FindInMap (p. 324) function to return a named value based on a specified key. The following example template contains an AWS::EC2::Instance (p. 222) resource whose ImageId property is assigned by the *FindInMap* function. The *FindInMap* function specifies key as the region where the stack is created (using the AWS::Region pseudo parameter (p. 329)) and 32 as the name of the value to map to.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" },
      "ap-southeast-1" : { "32" : "ami-66f28c34", "64" : "ami-60f28c32" },
      "ap-northeast-1" : { "32" : "ami-9c03a89d", "64" : "ami-a003a8a1" }
    }
  },

  "Resources" : {
     "myEC2Instance" : {
        "Type" : "AWS::EC2::Instance",
        "Properties" : {
         "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region"
 }, "32"]},
           "InstanceType" : "m1.small"
        }
     }
 }
}
```

The following example shows a Mappings section with a mapping that contains three keys that map to arrays that contain multiple string values. The keys represent three regions, and the mapped values are the list of availability zones used in each region. The AWS::ElasticLoadBalancing::LoadBalancer (p. 258) resource uses the *FindInMap* function and the *Region2AZ* map to specify the *AvailabilityZones* property.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",

    "Mappings" : {
      "Region2AZ" : {
        "us-west-1" : { "AZ" : ["us-west-1a", "us-west-1b"] },
        "us-east-1" : { "AZ" : ["us-east-1a", "us-east-1b", "us-east-1c"] },
        "eu-west-1" : { "AZ" : ["eu-west-1a", "eu-west-1b"] }
      }
    },
    "Resources" : {
      "MyELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
          "AvailabilityZones" : { "Fn::FindInMap" : [ "Region2AZ", { "Ref" :
"AWS::Region" }, "AZ" ] },
          "Listeners" : [ {
            "LoadBalancerPort" : "8888" ,
            "InstancePort" : "8888" ,
            "Protocol" : "HTTP"
          } ],
          "HealthCheck" : {
           "Target" : { "Fn::Join" : [ "", ["HTTP:", "8888", "/"]]},
            "HealthyThreshold" : "5",
            "UnhealthyThreshold" : "2",
            "Interval" : "10",
            "Timeout" : "8"
          }
        }
      }
    }
}
```

Resource Declaration

In the Resources section you declare the AWS resources you want as part of your stack. Resources are separated by a comma.

Each resource must have a logical name unique within the template. This is the name you use elsewhere in the template as a dereference argument.

You must specify the type for each resource. Type names are fixed according to those listed in Resource Property Types Reference (p. 292).

If a resource does not require any properties to be declared, you can omit the Properties section of that resource. For information on Property declaration, see (p. 97).

The following example shows a typical Resource declaration. It defines two resources. The *`MyInstance`* resource includes the *`MyQueue`* resource as part of its *`UserData`* property:

```
  "Resources" : {
    "MyInstance" : {
        "Type" : "AWS::EC2::Instance",
        "Properties" : {
            "UserData" : {
                "Fn::Base64" : {
                    "Fn::Join" : [ "", [ "Queue=", { "Ref" : "MyQueue" } ] ]
                 } },
            "AvailabilityZone" : "us-east-1a",
            "ImageId" : "ami-20b65349"
        }
    },

    "MyQueue" : {
        "Type" : "AWS::SQS::Queue",
        "Properties" : {
        }
    }
}
```

Properties Declaration

Properties are declared in the Properties section of a resource. In addition, outputs declared in the Outputs section follow the rules for a property.

A Properties section is declared for each resource immediately after the resource Type declaration. Multiple properties are separated by a comma. Each property is declared with a double-quoted name, a single colon, and the value for the property.

Property values can be literal strings, lists of strings, parameter references, pseudo references, or the value returned by a function. When a property value is a literal string, the value is enclosed in double quotes. If a value is the result of a list of any kind, it is enclosed in brackets ("[ ]"). If a value is the result of an intrinsic function or reference, it is enclosed in braces ("{ }"). These rules apply when you combine literals, lists, references, and functions to obtain a value.

The following example shows several ways to declare a property.

```
"Properties" : {
    "MyString" : "one-string-value",
    "MyLiteralList" : [ "first-value", "second-value" ],
    "MyReferenceForOneValue" :  { "Ref" : "MyLogicalResourceName" } ,
    "MyFunctionResultWithFunctionParams" : {
        "Fn::Join" : [ "%", [ "Key=", { "Ref" : "MyParameter" } ] ] }
}
```

Function Declaration

AWS CloudFormation intrinsic functions are special actions you use in your template to assign values to properties not avalible until runtime. Each function is declared with a double-quoted name, a single colon, and its arguments. When an argument is a literal string, it is enclosed in double quotes. If an argument is the result of a list of any kind, it is enclosed in brackets ("[ ]"). If an argument is the result of an intrinsic function or reference, it is enclosed in braces ("{ }").

The following example shows the function "Fn::GetAtt" being used to assign a value to the *`MyLBDNSName`*, which it does by retrieving the value of the attribute *`DNSName`* from the Elastic Load Balancing load balancer named *`MyLoadBalancer`*.

```
"Properties" : {
    "MyMyLBDNSName" : {
        "Fn::GetAtt" : [ "MyLoadBalancer", "DNSName" ]
    }
}
```

For more information on intrinsic functions, see Intrinsic Function Reference (p. 323).

Outputs Declaration

The template Outputs section enables you to return one or more values to the user in response to the `cfn-describe-stacks` and `cfn-describe-stack-resources` commands. If present, it must declare at least one output value.

To declare the Outputs section, the double-quoted key name *Outputs* is followed by a single colon. All outputs declared in the Outputs section are contained within a single set of braces, and are delimited with a comma.

Each output is composed of a double-quoted key name, a single colon, and one or more properties.

There are two Output properties:

- *Value* (required). The value of the property that is returned by `cfn-describe-stacks`.
- *Description* (optional). A String type up to 4K in length describing the output value.

Output properties are declared like any other property. In the following example, the output named *LoadBalancer* returns the information for the resource with the logical name *BackupLoadBalancer*.

```
"Outputs" : {
    "LoadBalancer" : {
        "Value" : { Ref : "BackupLoadBalancer" }
    }
}
```

Example Templates

The example AWS CloudFormation templates are written to show the features of AWS CloudFormation, and to serve as a starting point for you to create custom stacks. We provide the two stack applications below. In the following sections we describe the template, its parts, and detail any special features it may have. A link to the latest source code for the template is also included.

Topics

More sample templates are available at http://aws.amazon.com/cloudformation/aws-cloudformation-templates/. In addition, we add new sample templates regularly to provide examples for newly supported features. Please check the AWS CloudFormation Discussion Forum for announcements. Also, other AWS CloudFormation users may have developed templates to provide custom solutions, and may post their AWS CloudFormation solutions to the forum as well.

Auto Scaling Group with LoadBalancer, Auto Scaling Policies, and CloudWatch Alarms

Topics

This template creates a sample web site that uses Auto Scaling and Elastic Load Balancing and is configured to use multiple availability zones. The template also contains Amazon CloudWatch alarms that execute Auto Scaling policies to add or remove instances from the Auto Scaling group when the defined thresholds are exceeded.

Important

This template creates one or more Amazon EC2 instances. You will be billed for the AWS resources used if you create a stack from this template.

To create the stack using this sample template, use the following command:

```
cfn-create-stack StackName --template-url https://s3.amazonaws.com/cloudforma
tion-templates-us-east-1/AutoScalingMultiAZSample.template --parameters "Key
Name=key-pair-name"
```

Or click the following the link to create the stack in the US-East Region:

https://console.aws.amazon.com/cloudformation/home?region=us-east-1#cstack=sn~AutoScalingMultiAZSample|turl~https://s3.amazonaws.com/cloudformation-templates-us-east-1/AutoScalingMultiAZSample.template

You can get the latest version of this sample template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/AutoScalingMultiAZSample.template.

Auto Scaling Multi-AZ Template

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "Create a multi-az, load balanced, Auto Scaled sample web site.
 The Auto Scaling trigger is based on the CPU utilization of the web servers. The
 AMI is chosen based on the region in which the stack is run. This example creates
 a web service running across all availability zones in a region. The instances
are load balanced with a simple health check. The web site is available on port
80, however, the instances can be configured to listen on any port (8888 by de
fault). **WARNING** This template creates one or more Amazon EC2 instances. You
will be billed for the AWS resources used if you create a stack from this tem
plate.",

  "Parameters" : {
    "InstanceType" : {
      "Description" : "Type of EC2 instance to launch",
      "Type" : "String",
      "Default" : "m1.small"
    },
    "WebServerPort" : {
      "Description" : "The TCP port for the Web Server",
      "Type" : "String",
      "Default" : "8888"
    },
    "KeyName" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instances",
      "Type" : "String"
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "64" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" },
      "ap-southeast-1" : { "32" : "ami-66f28c34", "64" : "ami-60f28c32" },
      "ap-northeast-1" : { "32" : "ami-9c03a89d", "64" : "ami-a003a8a1" }
    }
  },

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

    "LaunchConfig" : {
      "Type" : "AWS::AutoScaling::LaunchConfiguration",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                                          { "Fn::FindInMap" : [ "AWSInstance
Type2Arch", { "Ref" : "InstanceType" },
                                          "Arch" ] } ] },
        "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }},
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "InstanceType" : { "Ref" : "InstanceType" }
      }
    },

    "WebServerScaleUpPolicy" : {
      "Type" : "AWS::AutoScaling::ScalingPolicy",
      "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "WebServerGroup" },
        "Cooldown" : "60",
        "ScalingAdjustment" : "1"
      }
    },
    "WebServerScaleDownPolicy" : {
      "Type" : "AWS::AutoScaling::ScalingPolicy",
      "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "WebServerGroup" },
        "Cooldown" : "60",
        "ScalingAdjustment" : "-1"
      }
    },

    "CPUAlarmHigh": {
     "Type": "AWS::CloudWatch::Alarm",
     "Properties": {
        "AlarmDescription": "Scale-up if CPU > 90% for 10 minutes",
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/EC2",
        "Statistic": "Average",
        "Period": "300",
        "EvaluationPeriods": "2",
        "Threshold": "90",
        "AlarmActions": [ { "Ref": "WebServerScaleUpPolicy" } ],
        "Dimensions": [
          {
            "Name": "AutoScalingGroupName",
            "Value": { "Ref": "WebServerGroup" }
          }
        ],
        "ComparisonOperator": "GreaterThanThreshold"
      }
    },
    "CPUAlarmLow": {
     "Type": "AWS::CloudWatch::Alarm",
     "Properties": {
        "AlarmDescription": "Scale-down if CPU < 70% for 10 minutes",
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/EC2",
        "Statistic": "Average",
        "Period": "300",
        "EvaluationPeriods": "2",
        "Threshold": "70",
        "AlarmActions": [ { "Ref": "WebServerScaleDownPolicy" } ],
        "Dimensions": [
          {
            "Name": "AutoScalingGroupName",
            "Value": { "Ref": "WebServerGroup" }
          }
        ],
        "ComparisonOperator": "LessThanThreshold"
      }
    },

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

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access and HTTP access on the inbound 
port",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        },
        {
          "IpProtocol" : "tcp",
          "FromPort" : { "Ref" : "WebServerPort" },
          "ToPort" : { "Ref" : "WebServerPort" },
          "SourceSecurityGroupOwnerId" : {"Fn::GetAtt" : ["ElasticLoadBalancer", 
"SourceSecurityGroup.OwnerAlias"]},
          "SourceSecurityGroupName" : {"Fn::GetAtt" : ["ElasticLoadBalancer", 
"SourceSecurityGroup.GroupName"]}
        } ]
      }
    }
  },

  "Outputs" : {
    "URL" : {
      "Description" : "The URL of the website",
      "Value" :  { "Fn::Join" : [ "", [ "http://", { "Fn::GetAtt" : [ "ElasticLoad
Balancer", "DNSName" ]}]]}
    }
  }
}
```

Template Walkthrough

The example template contains an Auto Scaling group with a LoadBalancer, a security group that defines ingress rules, Amazon CloudWatch alarms, and Auto Scaling policies.

The template has three input parameters: InstanceType is the type of EC2 instance to use for the Auto Scaling group and has a default of m1.small; WebServerPort is the TCP port for the web server and has a default of 8888; KeyName is the name of an EC2 key pair to be used for the Auto Scaling group. KeyName must be specified at stack creation (parameters with no default value must be specified at stack creation).

The AWS::AutoScaling::AutoScalingGroup (p. 190) resource WebServerGroup declares the following Auto Scaling group configuration:

- *AvailabilityZones* specifies the availability zones where the auto scaling group's EC2 instances will be created. The Fn::GetAZs (p. 325) function call `{ "Fn::GetAZs" : "" }` specifies all availability zones for the region in which the stack is created.

- *MinSize* and *MaxSize* set the minimum and maximum number of EC2 instances in the Auto Scaling group.
- *LoadBalancerNames* lists the LoadBalancers used to route traffic to the Auto Scaling group. The LoadBalancer for this group is the ElasticLoadBalancer resource.

The AWS::AutoScaling::LaunchConfiguration (p. 193) resource LaunchConfig declares the following configurations to use for the EC2 instances in the WebServerGroup Auto Scaling group:

- *KeyName* takes the value of the KeyName input parameter as the EC2 key pair to use.
- *UserData* is the Base64 encoded value of the WebServerPort parameter, which is passed to an application .
- *SecurityGroups* is a list of EC2 security groups that contain the firewall ingress rules for EC2 instances in the Auto Scaling group. In this example, there is only one security group and it is declared as a AWS::EC2::SecurityGroup (p. 237) resource: InstanceSecurityGroup. This security group contains two ingress rules: 1) a TCP ingress rule that allows access from all IP addresses ("CidrIp" : "0.0.0.0/0") for port 22 (for SSH access) and 2) a TCP ingress rule that allows access from the ElasticLoadBalancer resource for the WebServerPort port by specifying the LoadBalancer's source security group. The GetAtt (p. 324) function is used to get the SourceSecurityGroup.OwnerAlias and SourceSecurityGroup.GroupName properties from the ElasticLoadBalancer resource. For more information on the Elastic Load Balancing security groups, see Using Security Groups With Elastic Load Balancing.
- *ImageId* is the evaluated value of a set of nested maps. We added the maps so that the template contained the logic for choosing the right image ID. That logic is based on the instance type that was specified with the InstanceType parameter (AWSInstanceType2Arch maps the instance type to an architecture 32 or 64) and the region where the stack is created (AWSRegionArch2AMI maps the region and architecture to a image ID):

```
{ "Fn::FindInMap" : [ "AWSRegionArch2AMI",
   { "Ref" : "AWS::Region" },
   { "Fn::FindInMap" : [ "AWSInstanceType2Arch",
      { "Ref" : "InstanceType" },
      "Arch" ]
   }
   ]}
```

  For example, if you use this template to create a stack in the us-east-1 region and specify m1.small as InstanceType, AWS CloudFormation would evaluate the inner map for AWSInstanceType2Arch as the following:

```
{ "Fn::FindInMap" : [ "AWSInstanceType2Arch", "m1.small", "Arch" ] }
```

  In the AWSInstanceType2Arch mapping, the Arch value for the m1.small key maps to 32, which is used as the value for the outer map. The key is the evaluated result of the AWS::Region pseudo parameter which is the region where the stack is being created. For this example, AWS::Region is us-east-1; therefore, the outer map is evaluated as follows:

```
Fn::FindInMap" : [ "AWSRegionArch2AMI", "us-east-1", "32"]
```

  In the AWSRegionArch2AMI mapping, the value 32 for the key us-east-1 maps to ami-6411e20d. This means that ImageId would be ami-6411e20d.

The AWS::ElasticLoadBalancing::LoadBalancer (p. 258) resource ElasticLoadBalancer declares the following LoadBalancer configuration:

- *AvailabilityZones* is a list of availability zones where the LoadBalancer will distribute traffic. In this example, the Fn::GetAZs function call `{ "Fn::GetAZs" : "" }` specifies all availability zones for the region in which the stack is created.
- *Listeners* is a list of load balancing routing configurations that specify the port that the LoadBalancer accepts requests, the port on the registered EC2 instances where the LoadBalancer forwards requests, and the protocol used to route requests.
- *HealthCheck* is the configuration that Elastic Load Balancing uses to check the health of the EC2 instances that the LoadBalancer routes traffic to. In this example, the HealthCheck targets the root address of the EC2 instances using the port specified by WebServerPort over the HTTP protocol. If the WebServerPort is 8888, the `{ "Fn::Join" : [ "", ["HTTP:", { "Ref" : "WebServerPort" }, "/"]]}` function call is evaluated as the string `HTTP:8888/`. It also specifies that the EC2 instances have an interval of 30 seconds between health checks (Interval). The Timeout is defined as the length of time Elastic Load Balancing waits for a response from the health check target (5 seconds in this example). After the Timeout period lapses, Elastic Load Balancing marks that EC2 instance's health check as unhealthy. When an EC2 instance fails 5 consecutive health checks (UnhealthyThreshold), Elastic Load Balancing stops routing traffic to that EC2 instance until that instance has 3 consecutive healthy health checks at which point Elastic Load Balancing considers the EC2 instance healthy and begins routing traffic to that instance again.

The AWS::AutoScaling::ScalingPolicy (p. 197) resource WebServerScaleUpPolicy is an Auto Scaling policy that scales up the Auto Scaling group WebServerGroup. The *`AdjustmentType`* property is set to ChangeInCapacity. This means that the *`ScalingAdjustment`* represents the number of instances to add (if *`ScalingAdjustment`* is positive, instances are added; if negative, instances are deleted). In this example, *`ScalingAdjustment`* is 1; therefore, the policy increments the number of EC2 instances in the group by 1 when the policy is executed. The Cooldown property specifies that Auto Scaling waits 60 seconds before starting any other policy or trigger related actions.

The AWS::CloudWatch::Alarm (p. 214) resource CPUAlarmHigh specifies the scaling policy WebServerScaleUpPolicy as the action to execute when the alarm is in an *`ALARM`* state (AlarmActions). The alarm monitors the EC2 instances in the WebServerGroup Auto Scaling group (Dimensions). The alarm measures the average (Statistic) EC2 instance CPU utilization (Namespace and MetricName) of the instances in the WebServerGroup (Dimensions) over a 300 second interval (Period). When this value (average CPU utilization over 300 seconds) remains greater than 90 percent (ComparisonOperator and Threshold) for 2 consecutive periods (EvaluationPeriod), the alarm will go into an *`ALARM`* state and Amazon CloudWatch will execute the WebServerScaleUpPolicy policy (AlarmActions) described above scale up the WebServerGroup.

The CPUAlarmLow alarm measures the same metrics but has an alarm that triggers when CPU utilization is less than 75 percent (ComparisonOperator and Threshold) and executes the WebServerScaleDownPolicy policy to remove 1 EC2 instance from the Auto Scaling group WebServerGroup.

Amazon EC2 Running an Amazon Linux 32-bit AMI

This template declares one parameter and four mappings. Resources include an Amazon EC2 instance and a security group. The mapping uses the AWS::Region pseudo parameter to select the appropriate AMI. The Outputs section prints the instance ID of the instance, the Availability Zone in which it is created, and its public IP address.

To create the stack using this sample template, use the following command:

```
cfn-create-stack StackName --template-file EC2InstanceWithSecurityGroupSample-
1.0.0.template --parameters "KeyName=key-pair-name"
```

You can get the latest version of this sample template at
https://s3.amazonaws.com/cloudformation-templates-us-east-1/EC2InstanceWithSecurityGroupSample-1.0.0.template.

Amazon Linux 32-bit AMI Sample Template

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI. The
 AMI is chosen based on the region in which the stack is run. This example creates
 an EC2 security group for the instance to give you SSH access. **WARNING** This
template creates one or more Amazon EC2 instances. You will be billed for the AWS
 resources used if you create a stack from this template.",

  "Parameters" : {
    "KeyName" : {
      "Description" : "Name of and existing EC2 KeyPair to enable SSH access to
the instance",
      "Type" : "String"
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {
          "AMI" : "ami-76f0061f"
      },
      "us-west-1" : {
          "AMI" : "ami-655a0a20"
      },
      "eu-west-1" : {
          "AMI" : "ami-7fd4e10b"
      },
      "ap-southeast-1" : {
          "AMI" : "ami-72621c20"
      },
      "ap-northeast-1" : {
          "AMI" : "ami-8e08a38f"
      }
    }
  },

  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" },
 "AMI" ]}
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  },

  "Outputs" : {
    "InstanceId" : {
      "Description" : "InstanceId of the newly created EC2 instance",
      "Value" : { "Ref" : "Ec2Instance" }
    },
    "AZ" : {
      "Description" : "Availability Zone of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "AvailabilityZone" ] }
    },
    "PublicIP" : {
      "Description" : "Public IP address of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicIp" ] }
    }
  }
}
```

Create a Load-Balanced Apache Website

This template declares two parameters and four mappings. Resources include an Elastic Load Balancing load balancer with listeners and health check, two Amazon EC2 instances, and a security group. The Outputs section prints the URL of the load balancer.

To create the stack using this sample template, use the following command:

```
cfn-create-stack StackName --template-file ELBSample-1.0.0.template --parameters
 "KeyName=key-pair-name"
```

You can get the latest version of this sample template at https://s3.amazonaws.com/cloudformation-templates-us-east-1/ELBSample-1.0.0.template.

Load-Balanced Apache Website Sample Template

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "Create a load balanced sample web site. The AMI is chosen based
 on the region in which the stack is run. This example creates 2 EC2 instances
behind a load balancer with a simple health check. The instances may be created
in one or more AZs. The web site is available on port 80, however, the instances
can be configured to listen on any port (8888 by default). **WARNING** This template
 creates one or more Amazon EC2 instances. You will be billed for the AWS resources
 used if you create a stack from this template.",

  "Parameters" : {
    "InstanceType" : {
      "Description" : "Type of EC2 instance to launch",
      "Type" : "String",
      "Default" : "m1.small"
    },
    "WebServerPort" : {
      "Description" : "TCP/IP port of the web server",
      "Type" : "String",
      "Default" : "8888"
    },
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the
 instances",
      "Type" : "String"
    }
  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "64" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" },
      "ap-southeast-1" : { "32" : "ami-66f28c34", "64" : "ami-60f28c32" },
      "ap-northeast-1" : { "32" : "ami-9c03a89d", "64" : "ami-a003a8a1" }
    }
  },

  "Resources" : {
    "ElasticLoadBalancer" : {
      "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
      "Properties" : {
        "AvailabilityZones" : { "Fn::GetAZs" : "" },
        "Instances" : [ { "Ref" : "Ec2Instance1" },{ "Ref" : "Ec2Instance2" } ],
        "Listeners" : [ {
          "LoadBalancerPort" : "80",
          "InstancePort" : { "Ref" : "WebServerPort" },
          "Protocol" : "HTTP"
        } ],
        "HealthCheck" : {
          "Target" : { "Fn::Join" : [ "", ["HTTP:", { "Ref" : "WebServerPort" },
"/"]]},
          "HealthyThreshold" : "3",
          "UnhealthyThreshold" : "5",
          "Interval" : "30",
          "Timeout" : "5"
        }
      }
    },

    "Ec2Instance1" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                                          { "Fn::FindInMap" : [ "AWSInstance
Type2Arch", { "Ref" : "InstanceType" },
                                          "Arch" ] } ] },
        "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }}
      }
    },

    "Ec2Instance2" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                                          { "Fn::FindInMap" : [ "AWSInstance
Type2Arch", { "Ref" : "InstanceType" },
                                          "Arch" ] } ] },
        "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }}
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access and HTTP access on the inbound
port",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        },
        {
          "IpProtocol" : "tcp",
          "FromPort" : { "Ref" : "WebServerPort" },
          "ToPort" : { "Ref" : "WebServerPort" },
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  },

  "Outputs" : {
    "URL" : {
      "Description" : "URL of the sample website",
     "Value" :  { "Fn::Join" : [ "", [ "http://", { "Fn::GetAtt" : [ "ElasticLoad
Balancer", "DNSName" ]}]]}
    }
  }
}
```

Template Snippets

This section provides a number of example scenarios that you can use to understand how to declare various AWS CloudFormation template parts. You can also use the snippets as a starting point for sections of your custom templates.

Note

Because AWS CloudFormation templates must be JSON compliant, there is no provision for a line continuation character. The wrapping of the snippets in this document may be random if the line is longer that 80 characters.

Auto Scaling Snippets

Topics

Auto Scaling Launch Configuration Resource

This example shows an Auto Scaling AWS::AutoScaling::LaunchConfiguration resource. The SecurityGroups property specifies both an AWS::EC2::SecurityGroup resource named myEC2SecurityGroup and an existing EC2 security group named myExistingEC2SecurityGroup. The BlockDeviceMappings property lists two devices: a 50 gigabyte EBS volume mapped to /dev/sdk and a virtual device ephemeral0 mapped to /dev/sdc.

```
"SimpleConfig" : {
    "Type" : "AWS::AutoScaling::LaunchConfiguration",
    "Properties" : {
        "ImageId" : "ami-6411e20d",
       "SecurityGroups" : [ { "Ref" : "myEC2SecurityGroup" }, "myExistingEC2Se
curityGroup" ],
        "InstanceType" : "m1.small",
        "BlockDeviceMappings" : [
           {
              "DeviceName" : "/dev/sdk",
              "Ebs" : {"VolumeSize" : "50"}
           },
           {
              "DeviceName" : "/dev/sdc",
              "VirtualName" : "ephemeral0"
           }
        ]
    }
},
```

Auto Scaling Group Resource

This example shows an Auto Scaling AWS::AutoScaling::AutoScalingGroup (p. 190) resource. The AvailabilityZones property specifies the availability zones where the Auto Scaling group's EC2 instances will be created. In this example, the Fn::GetAZs (p. 325) function call `{ "Fn::GetAZs" : "" }` specifies all availability zones for the region in which the stack is created. The LoadBalancerNames property lists the LoadBalancers used to route traffic to the Auto Scaling group. In this example, one LoadBalancer is specified, the AWS::ElasticLoadBalancing::LoadBalancer (p. 258) resource LB.

```
"MyServerGroup" : {
       "Type" : "AWS::AutoScaling::AutoScalingGroup",
       "Properties" : {
            "AvailabilityZones" : { "Fn::GetAZs" : ""},
            "LaunchConfigurationName" : { "Ref" : "SimpleConfig" },
            "MinSize" : "1",
            "MaxSize" : "3",
            "LoadBalancerNames" : [ { "Ref" : "LB" } ]
       }
},
```

Auto Scaling Policy Triggered by Amazon CloudWatch Alarm

This example shows an AWS::AutoScaling::ScalingPolicy (p. 197) resource that scales up the Auto Scaling group asGroup. The *`AdjustmentType`* property specifies ChangeInCapacity, which means that the *`ScalingAdjustment`* represents the number of instances to add (if *`ScalingAdjustment`* is positive) or delete (if it is negative). In this example, *`ScalingAdjustment`* is 1; therefore, the policy increments the number of EC2 instances in the group by 1 when the policy is executed.

The AWS::CloudWatch::Alarm (p. 214) resource CPUAlarmHigh specifies the scaling policy ScaleUpPolicy as the action to execute when the alarm is in an *`ALARM`* state (*`AlarmActions`*).

```
    "ScaleUpPolicy" : {
      "Type" : "AWS::AutoScaling::ScalingPolicy",
      "Properties" : {
        "AdjustmentType" : "ChangeInCapacity",
        "AutoScalingGroupName" : { "Ref" : "asGroup" },
        "Cooldown" : "1",
        "ScalingAdjustment" : "1"
      }
    },
    "CPUAlarmHigh": {
     "Type": "AWS::CloudWatch::Alarm",
     "Properties": {
        "EvaluationPeriods": "1",
        "Statistic": "Average",
        "Threshold": "10",
        "AlarmDescription": "Alarm if CPU too high or metric disappears indic
ating instance is down",
        "Period": "60",
        "AlarmActions": [
          {
            "Ref": "ScaleUpPolicy"
          }
        ],
        "Namespace": "AWS/EC2",
        "Dimensions": [
          {
            "Name": "AutoScalingGroupName",
            "Value": {
              "Ref": "asGroup"
            }
          }
        ],
        "ComparisonOperator": "GreaterThanThreshold",
        "MetricName": "CPUUtilization"
      }
    },
```

Auto Scaling Group with Notifications

This example shows an AWS::AutoScaling::AutoScalingGroup (p. 190) resource that sends Amazon SNS notifications when the specified events take place. The *`NotificationConfiguration`* property specifies the SNS topic where AWS CloudFormation sends a notification and the events that will cause AWS CloudFormation to send notifications. When the events specified by *`NotificationTypes`* occur, AWS CloudFormation will send a notification to the SNS topic specified by *`TopicARN`*. In this example, AWS CloudFormation sends a notification to the SNS topic topic1 when the *`autoscaling:EC2_INSTANCE_LAUNCH`* and *`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`* events occur.

```
"MyAsGroupWithNotification" : {
             "Type" : "AWS::AutoScaling::AutoScalingGroup",
              "Properties" : {
                  "AvailabilityZones" : { "Ref" : "azList" },
                  "LaunchConfigurationName" : { "Ref" : "myLCOne" },
                  "MinSize" : "0",
                  "MaxSize" : "2",
                  "DesiredCapacity" : "1",
                  "NotificationConfiguration" : {
                      "TopicARN" : { "Ref" : "topic1" },
                      "NotificationTypes" : [ "autoscaling:EC2_IN
STANCE_LAUNCH","autoscaling:EC2_INSTANCE_LAUNCH_ERROR","autoscaling:EC2_IN
STANCE_TERMINATE", "autoscaling:EC2_INSTANCE_TERMINATE_ERROR"]
                  }
              }
          }
```

Auto Scaling Trigger Resource

This example shows an Auto Scaling trigger for the MyServerGroup AWS::AutoScaling::AutoScalingGroup resource.

```
"MyTrigger" : {
    "Type" : "AWS::AutoScaling::Trigger",
    "Properties" : {
        "MetricName" : "CPUUtilization",
        "Namespace" : "AWS/EC2",
        "Statistic" : "Average",
        "Period" : "300",
        "UpperBreachScaleIncrement" : "1",
        "LowerBreachScaleIncrement" : "-1",
        "AutoScalingGroupName" : { "Ref" : "MyServerGroup" },
        "BreachDuration" : "600",
        "UpperThreshold" : "90",
        "LowerThreshold" : "75",
        "Dimensions" : [ {
            "Name" : "AutoScalingGroupName",
            "Value" : { "Ref" : "MyServerGroup" }
        } ]
    }
}
```

Amazon EC2 Snippets

Topics

Assigning an Amazon EC2 Elastic IP Using AWS::EC2::EIP Snippet

This example shows how to allocate an Amazon EC2 Elastic IP address and assign it to an Amazon EC2 instance using a AWS::EC2::EIP resource (p. 220).

```
"MyEIP" : {
    "Type" : "AWS::EC2::EIP",
    "Properties" : {
       "InstanceId" : { "Ref" : "logical name of an AWS::EC2::Instance resource"
 }
    }
}
```

Assigning an Existing Elastic IP to an Amazon EC2 instance using AWS::EC2::EIPAssociation Snippet

This example shows how to assign an existing Amazon EC2 Elastic IP address to an Amazon EC2 instance using an AWS::EC2::EIPAssociation resource (p. 221).

```
"IPAssoc" : {
            "Type" : "AWS::EC2::EIPAssociation",
            "Properties" : {
               "InstanceId" : { "Ref" : "logical name of an AWS::EC2::Instance
 resource" },
                "EIP" : "existing Elastic IP address"
            }
        }
```

Assigning an Existing VPC Elastic IP to an Amazon EC2 instance using AWS::EC2::EIPAssociation Snippet

This example shows how to assign an existing VPC Elastic IP address to an Amazon EC2 instance using an AWS::EC2::EIPAssociation resource (p. 221).

```
"VpcIPAssoc" : {
            "Type" : "AWS::EC2::EIPAssociation",
            "Properties" : {
               "InstanceId" : { "Ref" : "logical name of an AWS::EC2::Instance
 resource" },
                "AllocationId" : "existing VPC Elastic IP allocation ID"
            }
        }
```

Elastic Network Interface (ENI) Template Snippets

VPC_EC2_Instance_With_ENI

Sample template showing how to create an instance with 2 network interfaces, one for Web assess and one for SSH access. The default ENI for the instance is used for Web traffic and a second ENI is created for control port traffic. It assumes you have already created a VPC. **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.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template VPC_EC2_Instance_With_ENI:
 Sample template showing how to create an instance with 2 network interfaces,
one for Web assess and one for SSH access. The default ENI for the instance is
 used for Web traffic and a second ENI is created for control port traffic. It
 assumes you have already created a VPC. **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.",
  "Parameters" : {

    "KeyName" : {
      "Description" : "Name of and existing EC2 KeyPair to enable SSH access
to the instance",
      "Type" : "String"
    },

    "VpcId" : {
      "Type" : "String",
      "Description" : "VpcId of your existing Virtual Private Cloud (VPC)"
    },

    "SubnetId" : {
      "Type" : "String",
      "Description" : "SubnetId of an existing subnet in your Virtual Private
Cloud (VPC)"
    },

    "WebServerPort" : {
      "Description" : "TCP/IP port of the web server",
      "Type" : "String",
      "Default" : "80"
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1"      : { "AMI" : "ami-aba768c2" },
      "us-west-1"      : { "AMI" : "ami-458fd300" },
      "us-west-2"      : { "AMI" : "ami-fcff72cc" },
      "eu-west-1"      : { "AMI" : "ami-018bb975" },
      "sa-east-1"      : { "AMI" : "ami-a039e6bd" },
      "ap-southeast-1" : { "AMI" : "ami-425a2010" },
      "ap-northeast-1" : { "AMI" : "ami-7871c579" }
    }
  },

  "Resources" : {

    "ControlPortAddress" : {
      "Type" : "AWS::EC2::EIP",
      "Properties" : {
        "Domain" : "vpc"
      }
    },

    "AssociateControlPort" : {
      "Type" : "AWS::EC2::EIPAssociation",
      "Properties" : {
       "AllocationId" : { "Fn::GetAtt" : [ "ControlPortAddress", "AllocationId"
 ]},
        "NetworkInterfaceId" : { "Ref" : "controlXface" }
      }
    },

    "SSHSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "VpcId" : { "Ref" : "VpcId" },
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ { "IpProtocol" : "tcp", "FromPort" : "22",
"ToPort" : "22", "CidrIp" : "0.0.0.0/0" } ]
      }
    },

    "WebSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "VpcId" : { "Ref" : "VpcId" },
        "GroupDescription" : "Enable HTTP access via user defined port",
        "SecurityGroupIngress" : [ { "IpProtocol" : "tcp", "FromPort" : { "Ref"
 : "WebServerPort" }, "ToPort" : { "Ref" : "WebServerPort" }, "CidrIp" :
"0.0.0.0/0" } ]
      }
    },

    "controlXface" : {
      "Type" : "AWS::EC2::NetworkInterface",
      "Properties" : {
        "SubnetId" : { "Ref" : "SubnetId" },
        "Description" :"Interface for control traffic such as SSH",
        "GroupSet" : [ {"Ref" : "SSHSecurityGroup"} ],
        "SourceDestCheck" : "true",
        "Tags" : [ {"Key" : "Network", "Value" : "Control"}]
      }
    },

    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region"
 }, "AMI" ]},
        "KeyName" : { "Ref" : "KeyName" },
        "NetworkInterfaces" : [ { "NetworkInterfaceId" : {"Ref" : "con
trolXface"}, "DeviceIndex" : "0" } ],
        "Tags" : [ {"Key" : "Role", "Value" : "Test Instance"}],
        "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }}
      }
    }
  },

  "Outputs" : {
    "InstanceId" : {
      "Value" : { "Ref" : "Ec2Instance" },
      "Description" : "Instance Id of newly created instance"
    },

    "ControlPortPublicAddress" : {
      "Value" : { "Ref" : "ControlPortAddress" },
      "Description" : "Control port public IP address of instance for SSH"
    }
  }
}
```

Amazon EC2 Instance Resource

This snippet shows a simple AWS::EC2::Instance resource.

```
"MyInstance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
        "AvailabilityZone" : "us-east-1a",
        "ImageId" : "ami-20b65349"
    }
}
```

Amazon EC2 Instance Resource with UserData, Volume and Tag

This snippet shows an AWS::EC2::Instance resource with one Amazon EC2 Volume, one Tag, and Base64-encoded UserData. An AWS::EC2::SecurityGroup resource, an AWS::SNS::Topic resource, and an AWS::ETC::Volume resource all must be defined in the same template. Also, the references *KeyName*, *AccessKey*, and *SecretKey* are parameters must be defined in the Parameters section of the template.

```
"MyInstance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "SecurityGroups" : [ {
            "Ref" : "logical name of AWS::EC2::SecurityGroup resource"
        } ],
        "UserData" : {
            "Fn::Base64" : {
                "Fn::Join" : [ ":", [
                    "PORT=80",
                    "TOPIC=", {
                        "Ref" : "logical name of an AWS::SNS::Topic resource"
                    },
                    "ACCESS_KEY=", { "Ref" : "AccessKey" },
                    "SECRET_KEY=", { "Ref" : "SecretKey" } ]
                ]
            }
         },
        "InstanceType" : "m1.small",
        "AvailabilityZone" : "us-east-1a",
        "ImageId" : "ami-1e817677",
        "Volumes" : [
           { "VolumeId" : {
                "Ref" : "logical name of AWS::EC2::Volume resource"
           },
           "Device" : "/dev/sdk" }
        ],

        "Tags" : [ {
            "Key" : "Name",
            "Value" : "MyTag"
        } ]
    }
}
```

Amazon EC2 Instance Resource with an Amazon SimpleDB Domain

This snippet shows an AWS::EC2::Instance resource with an Amazon SimpleDB domain specified in the UserData.

```
"MyInstance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
        "UserData" : {
            "Fn::Base64" : {
                "Fn::Join" : [ "",
                    [ "Domain=", {
                        "Ref" : "logical name of an AWS::SDB::Domain resource"

                    } ]
                ]
            }
         },
        "AvailabilityZone" : "us-east-1a",
        "ImageId" : "ami-20b65349"
    }
}
```

Amazon EC2 Security Group Resource with Two CIDR Range Ingress Rules

This snippet shows an AWS::EC2::SecurityGroup resource that describes two ingress rules giving access to a specified CIDR range for the TCP protocol on the specified ports.

```
"ServerSecurityGroup" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
        "GroupDescription" : "allow connections from specified CIDR ranges",
        "SecurityGroupIngress" : [
            {
                "IpProtocol" : "tcp",
                "FromPort" : "80",
                "ToPort" : "80",
                "CidrIp" : "0.0.0.0/0"
            },{
                "IpProtocol" : "tcp",
                "FromPort" : "22",
                "ToPort" : "22",
                "CidrIp" : "192.168.1.1/32"
            }
        ]
    }
}
```

Amazon EC2 Security Group Resource with Two Security Group Ingress Rules

This snippet shows an AWS::EC2::SecurityGroup resource that describes two security group ingress rules. The first ingress rule grants access to the existing security group myadminsecuritygroup, which is owned by the 1234-5678-9012 AWS account, for the TCP protocol on port 22. The second ingress rule grants access to the security group mysecuritygroupcreatedincfn for TCP on port 80. This ingress rule uses the Ref intrinsic function to refer to a security group (whose logical name is mysecuritygroupcreatedincfn) created in the same template. You must declare a value for both the `SourceSecurityGroupName` and `SourceSecurityGroupOwnerId` properties.

```
"ServerSecurityGroupBySG" : {
    "Type" : "AWS::EC2::SecurityGroup",
    "Properties" : {
        "GroupDescription" : "allow connections from specified source security
 group",
        "SecurityGroupIngress" : [
            {
               "IpProtocol" : "tcp",
               "FromPort" : "22",
               "ToPort" : "22",
               "SourceSecurityGroupName" : "myadminsecuritygroup",
               "SourceSecurityGroupOwnerId" : "123456789012"
            },
            {
               "IpProtocol" : "tcp",
               "FromPort" : "80",
               "ToPort" : "80",
               "SourceSecurityGroupName" : {"Ref" : "mysecuritygroupcre
atedincfn"}
            }
        ]
    }
}
```

Amazon EC2 Security Group Resource with LoadBalancer Ingress Rule

This snippet shows an AWS::EC2::SecurityGroup resource that contains a security group ingress rule that grants access to the LoadBalancer myELB for TCP on port 80. Note that the rule uses the *`SourceSecurityGroup.OwnerAlias`* and *`SourceSecurityGroup.GroupName`* properties of the myELB resource to specify the source security group of the LoadBalancer.

```
        "myELB" : {
                    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
                    "Properties" : {
                        "AvailabilityZones" : [ "us-east-1a" ],
                        "Listeners" : [ {
                            "LoadBalancerPort" : "80",
                            "InstancePort" : "80",
                            "Protocol" : "HTTP"
                        } ]
                    }
                },
        "ELBIngressGroup" : {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties" : {
                "GroupDescription" : "ELB ingress group",
                "SecurityGroupIngress" : [
                    {
                       "IpProtocol" : "tcp",
                       "FromPort" : "80",
                       "ToPort" : "80",
                      "SourceSecurityGroupOwnerId" : {"Fn::GetAtt" : ["myELB",
 "SourceSecurityGroup.OwnerAlias"]},
                       "SourceSecurityGroupName" : {"Fn::GetAtt" : ["myELB",
"SourceSecurityGroup.GroupName"]}
                    }
                ]
            }
```

Using AWS::EC2::SecurityGroupIngress to Create Mutually Referencing Amazon EC2 Security Group Resources

This snippet shows two AWS::EC2::SecurityGroupIngress resources that add mutual ingress rules to the EC2 security groups SGroup1 and SGroup2. The SGroup1Ingress resource enables ingress from SGroup2 through TCP/IP port 80 to SGroup1. The SGroup2Ingress resource enables ingress from SGroup1 through TCP/IP port 80 to SGroup2.

```
        "SGroup1" : {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties" : {
                "GroupDescription" : "EC2 Instance access"
            }
        },
        "SGroup2" : {
            "Type" : "AWS::EC2::SecurityGroup",
            "Properties" : {
                "GroupDescription" : "EC2 Instance access"
            }
        },
        "SGroup1Ingress" : {
            "Type" : "AWS::EC2::SecurityGroupIngress",
            "Properties" : {
                "GroupName" : { "Ref" : "SGroup1" },
                "IpProtocol" : "tcp",
                "ToPort" : "80",
                "FromPort" : "80",
                "SourceSecurityGroupName" : { "Ref" : "SGroup2" }
            }
        },
        "SGroup2Ingress" : {
            "Type" : "AWS::EC2::SecurityGroupIngress",
            "Properties" : {
                "GroupName" : { "Ref" : "SGroup2" },
                "IpProtocol" : "tcp",
                "ToPort" : "80",
                "FromPort" : "80",
                "SourceSecurityGroupName" : { "Ref" : "SGroup1" }
            }
        }
```

Amazon EC2 Volume Resource

This snippet shows a simple Amazon EC2 volume resource with a DeletionPolicy attribute set to Snapshot. With the Snapshot DeletionPolicy set, AWS CloudFormation will take a snapshot of this volume before deleting it during stack deletion. Make sure you specify a value for `SnapShotId`, or a value for `Size`, but not both. Remove the one you don't need.

```
"MyEBSVolume" : {
    "Type" : "AWS::EC2::Volume",
    "Properties" : {
        "Size" : "specify a size if no SnapShotId",
        "SnapshotId" : "specify a SnapShotId if no Size",
        "AvailabilityZone" : { "Ref" : "AvailabilityZone" }
    },
    "DeletionPolicy" : "Snapshot"
}
```

Amazon EC2 VolumeAttachment Resource

This snippet shows the following resources: an Amazon EC2 instance using an Amazon Linux AMI from the US-East (Northern Virginia) Region, an EC2 security group that allows SSH access to IP addresses, a new Amazon EBS volume sized at 100 GB and in the same Availability Zone as the EC2 instance, and a volume attachment that attaches the new volume to the EC2 instance.

```
"Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "ImageId" : "ami-76f0061f"
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    },

    "NewVolume" : {
      "Type" : "AWS::EC2::Volume",
      "Properties" : {
        "Size" : "100",
       "AvailabilityZone" : { "Fn::GetAtt" : [ "Ec2Instance", "AvailabilityZone"
 ]},
      }
    },

    "MountPoint" : {
      "Type" : "AWS::EC2::VolumeAttachment",
      "Properties" : {
        "InstanceId" : { "Ref" : "Ec2Instance" },
        "VolumeId"  : { "Ref" : "NewVolume" },
        "Device" : "/dev/sdh"
      }
    }
}
```

Elastic Load Balancing Snippets

Topics

Elastic Load Balancing Load Balancer Resource

This example shows an Elastic Load Balancing load balancer with a single listener, and no instances.

```
"MyLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "us-east-1a" ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
            "InstancePort" : "80",
            "Protocol" : "HTTP"
        } ]
    }
}
```

Elastic Load Balancing Load Balancer Resource with Health Check

This example shows an Elastic Load Balancing load balancer with two Amazon EC2 instances, a single listener and a health check.

```
"MyLoadBalancer" : {
    "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
    "Properties" : {
        "AvailabilityZones" : [ "us-east-1a" ],
        "Instances" : [
            { "Ref" : "logical name of AWS::EC2::Instance resource 1" },
            { "Ref" : "logical name of AWS::EC2::Instance resource 2" }
        ],
        "Listeners" : [ {
            "LoadBalancerPort" : "80",
            "InstancePort" : "80",
            "Protocol" : "HTTP"
        } ],

        "HealthCheck" : {
            "Target" : "HTTP:80/",
            "HealthyThreshold" : "3",
            "UnhealthyThreshold" : "5",
            "Interval" : "30",
            "Timeout" : "5"
        }
    }
}
```

Identity and Access Management (IAM) Template Snippets

This section contains AWS Identity and Access Management template snippets.

Topics

Important

When creating or updating a stack using a template containing IAM resources, you must acknowledge the use of IAM capabilities. For more information about using IAM resources in templates, see Controlling User Access with AWS Identity and Access Management (p. 159).

Declaring an IAM User Resource

This snippet shows how to declare an AWS::IAM::User (p. 271) resource to create an IAM user. The user is declared with the path "/" and a login profile with the password myP@ssW0rd.

The policy document named `giveaccesstoqueueonly` gives the user permission to perform all SQS actions on the SQS queue resource myqueue, and denies access to all other SQS queue resources. The Fn::GetAtt (p. 324) function gets the Arn attribute of the AWS::SQS::Queue (p. 288) resource myqueue.

The policy document named `giveaccesstotopiconly` is added to the user to give the user permission to perform all SNS actions on the SNS topic resource mytopic and to deny access to all other SNS resources. The Ref function (p. 326) gets the ARN of the AWS::SNS::Topic (p. 288) resource mytopic.

```
"myuser" : {
   "Type" : "AWS::IAM::User",
   "Properties" : {
      "Path" : "/",
      "LoginProfile" : {
         "Password" : "myP@ssW0rd"
      },
      "Policies" : [ {
         "PolicyName" : "giveaccesstoqueueonly",
         "PolicyDocument" : {
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sqs:*" ],
               "Resource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sqs:*" ],
               "NotResource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            }
         ] }
      }, {
         "PolicyName" : "giveaccesstotopiconly",
         "PolicyDocument" : {
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sns:*" ],
               "Resource" : [ { "Ref" : "mytopic" } ]
            }, {
               "Effect" : "Deny",
               "Action" : [ "sns:*" ],
               "NotResource" : [ { "Ref" : "mytopic" } ]
            } ]
         }
      } ]
   }
}
```

Declaring an IAM Access Key Resource

This snippet shows an AWS::IAM::AccessKey (p. 262) resource. The myaccesskey resource creates an access key and assigns it to an IAM user that is declared as an AWS::IAM::User (p. 271) resource in the template.

```
"myaccesskey" : {
   "Type" : "AWS::IAM::AccessKey",
   "Properties" : {
      "UserName" : { "Ref" : "myuser" }
   }
}
```

You can get the secret key for an AWS::IAM::AccessKey resource using the Fn::GetAtt (p. 324) function. The only time that you can get the secret key for an AWS access key is when it is created. One way to retrieve the secret key is by putting it into an output value. You can get the access key using the Ref function. The following output value declarations get the access key and secret key for myaccesskey.

```
"AccessKeyformyaccesskey" : {
   "Value" : { "Ref" : "myaccesskey" }
},
"SecretKeyformyaccesskey" : {
   "Value" : {
      "Fn::GetAtt" : [ "myaccesskey", "SecretAccessKey" ]
   }
}
```

You can also pass the AWS access key and secret key to an EC2 instance or Auto Scaling group defined in the template. The following AWS::EC2::Instance (p. 222) declaration uses the UserData property to pass the access key and secret key for the myaccesskey resource.

```
"myinstance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "AvailabilityZone" : "us-east-1a",
      "ImageId" : "ami-20b65349",
      "UserData" : {
         "Fn::Base64" : {
            "Fn::Join" : [
               "", [
                  "ACCESS_KEY=", {
                     "Ref" : "myaccesskey"
                  },
                  "&",
                  "SECRET_KEY=",
                  {
                     "Fn::GetAtt" : [
                        "myaccesskey",
                        "SecretAccessKey"
                     ]
                  }
               ]
            ]
         }
      }
   }
}
```

Declaring an IAM Group Resource

This snippet shows an AWS::IAM::Group (p. 263) resource. The group has a path ("/myapplication/"). The policy document named myapppolicy is added to the group to allow the group's users to perform all SQS actions on the SQS queue resource myqueue and deny access to all other SQS resources except myqueue.

To assign a policy to a resource, IAM requires the Amazon Resource Name (ARN) for the resource. In the snippet, the Fn::GetAtt (p. 324) function gets the ARN of the AWS::SQS::Queue (p. 288) resource queue.

```
"mygroup" : {
   "Type" : "AWS::IAM::Group",
   "Properties" : {
      "Path" : "/myapplication/",
      "Policies" : [ {
         "PolicyName" : "myapppolicy",
         "PolicyDocument" : {
            "Statement" : [ {
               "Effect" : "Allow",
               "Action" : [ "sqs:*" ],
               "Resource" : [ {
                  "Fn::GetAtt" : [ "myqueue", "Arn" ]
               } ]
            },
            {
               "Effect" : "Deny",
               "Action" : [ "sqs:*" ],
               "NotResource" : [ { "Fn::GetAtt" : [ "myqueue", "Arn" ] } ]
            }
         ] }
      } ]
   }
}
```

Adding Users to a Group

The AWS::IAM::UserToGroupAddition (p. 273) resource adds users to a group. In the following snippet, the addUserToGroup resource adds the following users to an existing group named myexistinggroup2: an existing user existinguser1 and a user myuser that is declared as an AWS::IAM::User (p. 271) resource in the template.

```
"addUserToGroup" : {
   "Type" : "AWS::IAM::UserToGroupAddition",
   "Properties" : {
      "GroupName" : "myexistinggroup2",
      "Users" : [ "existinguser1", { "Ref" : "myuser" } ]
   }
}
```

Declaring an IAM Policy

This snippet shows how to create a policy and apply it to multiple groups using an AWS::IAM::Policy (p. 266) resource named mypolicy. The mypolicy resource contains a PolicyDocument property that allows GetObject, PutObject, and PutObjectAcl actions on the objects in the S3 bucket represented by the ARN arn:aws:s3:::myAWSBucket. The mypolicy resource applies the policy to an existing group named myexistinggroup1 and a group mygroup that is declared in the template as an AWS::IAM::Group (p. 263) resource. This example shows how apply a policy to a group using the Groups property; however, you can alternatively use the Users property to add a policy document to a list of users.

```
"mypolicy" : {
   "Type" : "AWS::IAM::Policy",
   "Properties" : {
      "PolicyName" : "mygrouppolicy",
      "PolicyDocument" : {
         "Statement" : [ {
            "Effect" : "Allow",
            "Action" : [
               "s3:GetObject" , "s3:PutObject" , "s3:PutObjectAcl" ],
            "Resource" : "arn:aws:s3:::myAWSBucket/*"
         } ]
      },
      "Groups" : [ "myexistinggroup1", { "Ref" : "mygroup" } ]
   }
}
```

Declaring an Amazon S3 Bucket Policy

This snippet shows how to create a policy and apply it to an Amazon S3 bucket using the AWS::S3::BucketPolicy (p. 287) resource. The mybucketpolicy resource declares a policy document that allows the group mygroup to perform the GetObject action on all objects in the S3 bucket to which this policy is applied. In the snippet, the Fn::GetAtt (p. 324) function gets the ARN of the mygroup resource. The mybucketpolicy resource applies the policy to the AWS::S3::Bucket (p. 286) resource mybucket. The Ref function (p. 326) gets the bucket name of the mybucket resource.

```
"mybucketpolicy" : {
   "Type" : "AWS::S3::BucketPolicy",
   "Properties" : {
      "PolicyDocument" : {
         "Id" : "MyPolicy",
         "Statement" : [ {
            "Sid" : "ReadAccess",
            "Action" : [ "s3:GetObject" ],
            "Effect" : "Allow",
            "Resource" : { "Fn::Join" : [
                  "", [ "arn:aws:s3:::", { "Ref" : "mybucket" } , "/*" ]
               ] },
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "mygroup", "Arn" ] }
            }
         } ]
      },
      "Bucket" : { "Ref" : "mybucket" }
      }
   }
}
```

Declaring an Amazon SNS Topic Policy

This snippet shows how to create a policy and apply it to an Amazon SNS topic using the AWS::SNS::TopicPolicy (p. 288) resource. The mysnspolicy resource contains a PolicyDocument property that allows an AWS::IAM::User (p. 271) resource myuser to perform the publish action on an AWS::SNS::Topic (p. 288) resource mytopic. In the snippet, the Fn::GetAtt (p. 324) function gets the ARN for the myuser resource and the Ref (p. 326) function gets the ARN for the mytopic resource.

```
"mysnspolicy" : {
   "Type" : "AWS::SNS::TopicPolicy",
   "Properties" : {
      "PolicyDocument" :  {
         "Version" : "2008-10-17",
         "Id" : "MyTopicPolicy",
         "Statement" : [ {
            "Sid" : "My-statement-id",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : { "Fn::GetAtt" : [ "myuser", "Arn" ] }
            },
            "Action" : "sns:Publish",
            "Resource" : "*"
         } ]
      },
      "Topics" : [ { "Ref" : "mytopic" } ]
   }
}
```

Declaring an Amazon SQS Policy

This snippet shows how to create a policy and apply it to an Amazon SQS queue using the AWS::SQS::QueuePolicy (p. 292) resource. The PolicyDocument property allows an existing user myapp (specified by its ARN) to perform the send message action on an existing queue, which is specified by its URL, and an AWS::SQS::Queue (p. 288) resource myqueue. The Ref (p. 326) function gets the URL for the myqueue resource.

```
"mysqspolicy" : {
   "Type" : "AWS::SQS::QueuePolicy",
   "Properties" : {
      "PolicyDocument" : {
         "Version" : "2008-10-17",
         "Id" : "MyQueuePolicy",
         "Statement" : [ {
            "Sid" : "Allow-User-SendMessage",
            "Effect" : "Allow",
            "Principal" : {
               "AWS" : "arn:aws:iam::123456789012:user/myapp"
            },
            "Action" : [ "sqs:SendMessage" ],
            "Resource" : "*"
         } ]
      },
      "Queues" : [
         "https://sqs.us-east-1.amazonaws.com/123456789012/myexistingqueue",
         { "Ref" : "myqueue" }
      ]
   }
}
```

IAM Role Template Examples

This section provides CloudFormation template examples for IAM Roles for EC2 Instances.

For detailed information about IAM Roles, see Working with Roles in the *AWS Identity and Access Management User Guide*.

IAM Role with EC2

Example IAM Role with External Policy and Instance Profiles wired to an EC2 Instance

In this example, the Instance Profile is referenced by the IamInstanceProfile property of the EC2 Instance. Both the Instance Policy and Role Policy reference the AWS::IAM::Role (p. 268).

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myEC2Instance": {
         "Type": "AWS::EC2::Instance",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-205fba49",
            "InstanceType": "m1.small",
            "Monitoring": "true",
            "DisableApiTermination": "false",
            "IamInstanceProfile": {
               "Ref": "RootInstanceProfile"
            }
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
```

```
    }
}
```

IAM Role with AutoScaling Group

Example IAM Roles With External Policy And Instance Profiles Wired to an AutoScaling Group

In this example, the Instance Profile is referenced by the IamInstanceProfile property of an AutoScaling Group Launch Configuration.

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "myLCOne": {
         "Type": "AWS::AutoScaling::LaunchConfiguration",
         "Version": "2009-05-15",
         "Properties": {
            "ImageId": "ami-205fba49",
            "InstanceType": "m1.small",
            "InstanceMonitoring": "true",
            "IamInstanceProfile": { "Ref": "RootInstanceProfile" }
         }
      },
      "myASGrpOne": {
         "Type": "AWS::AutoScaling::AutoScalingGroup",
         "Version": "2009-05-15",
         "Properties": {
            "AvailabilityZones": [ "us-east-1a" ],
            "LaunchConfigurationName": { "Ref": "myLCOne" },
            "MinSize": "0",
            "MaxSize": "0",
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": "120"
         }
      },
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ { "Ref": "RootRole" } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "RootRole" } ]
         }
      }
   }
}
```

Amazon RDS Template Snippets

Topics

Amazon RDS DB Instance Resource

This example shows an Amazon RDS DB Instance resource. Because the optional EngineVersion property is not specified, the default engine version is used for this DB Instance. For details on the default engine version and other default settings, see CreateDBInstance. The DBSecurityGroups property authorizes network ingress to the AWS::RDS::DBSecurityGroup resources named MyDbSecurityByEC2SecurityGroup and MyDbSecurityByCIDRIPGroup. For details, see AWS::RDS::DBInstance (p. 274). The DB Instance resource also has a DeletionPolicy attribute set to Snapshot. With the Snapshot DeletionPolicy set, AWS CloudFormation will take a snapshot of this DB Instance before deleting it during stack deletion.

```
"MyDB" : {
    "Type" : "AWS::RDS::DBInstance",
    "Properties" : {
        "DBSecurityGroups" : [
          {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" : "MyDbSecurityBy
CIDRIPGroup"} ],
        "AllocatedStorage" : "5",
        "DBInstanceClass" : "db.m1.small",
        "Engine" : "MySQL",
        "MasterUsername" : "MyName",
        "MasterUserPassword" : "MyPassword"
    },
    "DeletionPolicy" : "Snapshot"
}
```

Amazon RDS Oracle Database DB Instance Resource

This example creates an Oracle Database DB Instance resource by specifying the Engine as oracle-ee with a license model of bring-your-own-license. For details on the settings for Oracle Database DB instances, see CreateDBInstance. The DBSecurityGroups property authorizes network ingress to the AWS::RDS::DBSecurityGroup resources named MyDbSecurityByEC2SecurityGroup and

MyDbSecurityByCIDRIPGroup. For details, see AWS::RDS::DBInstance (p. 274). The DB Instance resource also has a DeletionPolicy attribute set to Snapshot. With the Snapshot DeletionPolicy set, AWS CloudFormation will take a snapshot of this DB Instance before deleting it during stack deletion.

```
"MyDB" : {
    "Type" : "AWS::RDS::DBInstance",
    "Properties" : {
        "DBSecurityGroups" : [
          {"Ref" : "MyDbSecurityByEC2SecurityGroup"}, {"Ref" : "MyDbSecurityBy
CIDRIPGroup"} ],
        "AllocatedStorage" : "5",
        "DBInstanceClass" : "db.m1.small",
        "Engine" : "oracle-ee",
        "LicenseModel" : "bring-your-own-license",
        "MasterUsername" : "master",
        "MasterUserPassword" : "SecretPassword01"
    },
    "DeletionPolicy" : "Snapshot"
}
```

Amazon RDS DBSecurityGroup Resource for CIDR Range

This example shows an Amazon RDS DBSecurityGroup resource with ingress authorization for the specified CIDR range in the format ddd.ddd.ddd.ddd/dd. For details, see AWS::RDS::DBSecurityGroup (p. 279) and RDS Security Group Rule (p. 319).

```
"MyDbSecurityByCIDRIPGroup" : {
    "Type" : "AWS::RDS::DBSecurityGroup",
    "Properties" : {
        "GroupDescription" : "Ingress for CIDRIP",
        "DBSecurityGroupIngress" : {
            "CIDRIP" : "192.168.0.0/32"
        }
    }
}
```

Amazon RDS DBSecurityGroup Resource for Amazon EC2 security group

This example shows an AWS::RDS::DBSecurityGroup (p. 279) resource with ingress authorization for the Amazon EC2 security group referenced by MyEc2SecurityGroup.

```
"MyDbSecurityByEC2SecurityGroup" : {
    "Type" : "AWS::RDS::DBSecurityGroup",
    "Properties" : {
        "GroupDescription" : "Ingress for Amazon EC2 security group",
        "DBSecurityGroupIngress" : {
           "EC2SecurityGroupName" : {"Ref" : "MyEc2SecurityGroup"},
           "EC2SecurityGroupOwnerId" : "123456789012"
        }
    }
}
```

Multiple VPC security groups

This example shows an AWS::RDS::DBSecurityGroup (p. 279) resource with ingress authorization for multiple Amazon EC2 VPC security groups with the EC2SecurityGroupId property of AWS::RDS::DBSecurityGroupIngress (p. 281).

```
{
   "Resources" : {
      "DBinstance" : {
         "Type" : "AWS::RDS::DBInstance",
         "Properties" : {
            "DBSecurityGroups" : [ {"Ref" : "DbSecurityByEC2SecurityGroup"} ],

            "AllocatedStorage" : "5",
            "DBInstanceClass" : "db.m1.small",
            "Engine" : "MySQL",
            "MasterUsername" : "YourName",
            "MasterUserPassword" : "YourPassword"
         },
         "DeletionPolicy" : "Snapshot"
      },
      "DbSecurityByEC2SecurityGroup" : {
         "Type" : "AWS::RDS::DBSecurityGroup",
         "Properties" : {
            "GroupDescription" : "Ingress for Amazon EC2 security group",
            "DBSecurityGroupIngress" : [ {
                  "EC2SecurityGroupId" : "sg-b0ff1111",
                  "EC2SecurityGroupOwnerId" : "111122223333"
               }, {
                  "EC2SecurityGroupId" : "sg-ffd722222",
                  "EC2SecurityGroupOwnerId" : "111122223333"
               } ]
         }
      }
   }
}
```

Amazon SimpleDB Snippets

Amazon SimpleDB Domain Resource

This example shows an Amazon SimpleDB domain resource.

```
"MySDBDomain" : {
    "Type" : "AWS::SDB::Domain",
    "Properties" : {
       "Description" : "Other than this AWS CloudFormation Description property,
 SDB Domains have no properties."
    }
}
```

Amazon SNS Snippets

Topics

Amazon SNS Topic Resource

This example shows an Amazon SNS topic resource. It requires a valid email address.

```
"MySNSTopic" : {
    "Type" : "AWS::SNS::Topic",
    "Properties" : {
        "Subscription" : [ {
            "Endpoint" : "add valid email address",
            "Protocol" : "email"
        } ]
    }
}
```

Amazon SQS Queue Snippet

This example shows an Amazon SQS queue.

```
"MyQueue" : {
    "Type" : "AWS::SQS::Queue",
    "Properties" : {
        "VisibilityTimeout" : "value"
    }
}
```

Amazon CloudFront Template Snippets

Topics

Amazon CloudFront Distribution Resource with Amazon S3 Origin

This example shows an Amazon CloudFront distribution using an Amazon S3 origin.

```

"myDistribution" : {
   "Type" : "AWS::CloudFront::Distribution",
   "Properties" : {
      "DistributionConfig" : {
         "S3Origin" : {
            "DNSName": "mybucket.s3.amazonaws.com",
           "OriginAccessIdentity" : "origin-access-identity/cloudfront/E127EX
AMPLE51Z"
         },
         "Enabled" : "true",
         "Comment" : "Some comment",
         "DefaultRootObject" : "index.html",
         "Logging" : {
            "Bucket" : "mylogs.s3.amazonaws.com",
            "Prefix" : "myprefix"
         },
         "CNAMEs" : [ "mysite.example.com", "yoursite.example.com" ],
         "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
         "RequiredProtocols" : [ "https" ]
      }
   }
}
```

Amazon CloudFront Distribution Resource with Custom Origin

This example shows an Amazon CloudFront distribution using a custom origin.

```
"myDistribution" : {
   "Type" : "AWS::CloudFront::Distribution",
   "Properties" : {
      "DistributionConfig" : {
         "CustomOrigin" : {
            "DNSName": "www.example.com",
            "HTTPPort" : "80",
            "HTTPSPort" : "443",
            "OriginProtocolPolicy" : "http-only"
         },
         "Enabled" : "true",
         "Comment" : "Some comment",
         "DefaultRootObject" : "index.html",
         "Logging" : {
            "Bucket" : "mylogs.s3.amazonaws.com",
            "Prefix" : "myprefix"
         },
         "CNAMEs" : [ "mysite.example.com", "yoursite.example.com" ],
         "TrustedSigners" : [ "1234567890EX", "1234567891EX" ],
         "RequiredProtocols" : [ "https" ]
      }
   }
}
```

Amazon CloudFront Distribution with Multi-origin Support.

This template snippet shows how to declare a CloudFront distribution with multi-origin support. In the DistributionConfig, a list of origins is provided and a DefaultCacheBehavior is set.

```
"myDistribution" : {
   "Type" : "AWS::CloudFront::Distribution",
   "Properties" : {
      "DistributionConfig" : {
         "Origins" : [ {
            "Id" : "Origin 1",
            "DomainName" : "cloudformation-examples.s3.amazonaws.com",
            "S3OriginConfig" : {
               "OriginAccessIdentity" : ""
            }
         } ],
         "DefaultCacheBehavior" : {
            "TargetOriginId" : "Origin 1",
            "ForwardedValues" : {
               "QueryString" : "false"
            },
            "ViewerProtocolPolicy" : "allow-all"
         },
         "Comment" : "Some comment",
         "Enabled" : "true"
      }
   }
}
```

Amazon Route 53 Template Snippets

Topics

Amazon Route 53 Resource Record Set Using Hosted Zone Name or ID

When you create an Amazon Route 53 resource record set, you must specify the hosted zone where you want to add it. AWS CloudFormation provides two ways to do this. You can explicitly specify the hosted zone using the HostedZoneId property or have AWS CloudFormation find the hosted zone using the HostedZoneName property. If you use the HostedZoneName property and there are multiple hosted zones with the same domain name, AWS CloudFormation doesn't create the stack.

Adding RecordSet using HostedZoneId

This example adds an Amazon Route 53 resource record set containing a CNAME record for the domain name "mysite.example.com" using the HostedZoneId property to specify the hosted zone.

```
"myDNSRecord" : {
            "Type" : "AWS::Route53::RecordSet",
            "Properties" : {
                "HostedZoneId" : "/hostedzone/Z3DG6IL3SJCGPX",
                "Comment" : "CNAME for my frontends.",
                "Name" : "mysite.example.com.",
                "Type" : "CNAME",
                "TTL" : "900",
                "ResourceRecords" : [
                    {"Fn::GetAtt":["myLB","DNSName"]}
                ]
            }
        }
```

Adding RecordSet using HostedZoneName

This example adds an Amazon Route 53 resource record set containing A records for the domain name "mysite.example.com" using the HostedZoneName property to specify the hosted zone.

```
"myDNSRecord2" : {
            "Type" : "AWS::Route53::RecordSet",
            "Properties" : {
                "HostedZoneName" : "example.com.",
                "Comment" : "A records for my frontends.",
                "Name" : "mysite.example.com.",
                "Type" : "A",
                "TTL" : "900",
                "ResourceRecords" : [
                    "192.168.0.1",
                    "192.168.0.2"
                ]
            }
        }
```

Using RecordSetGroup to Set Up Weighted Resource Record Sets

This example uses an AWS::Route53::RecordSetGroup (p. 285) to set up two CNAME records for the "example.com." hosted zone. The *RecordSets* property contains the CNAME record sets for the "mysite.example.com" DNS name. Each record set contains an identifier (SetIdentifier) and weight (Weight). The weighting for Frontend One is 40% (4 of 10) and Frontend Two is 60% (6 of 10). For more information on weighted resource record sets, see Setting Up Weighted Resource Record Sets in Route 53 Developer Guide.

```
        "myDNSOne" : {
            "Type" : "AWS::Route53::RecordSetGroup",
            "Properties" : {
                "HostedZoneName" : "example.com.",
                "Comment" : "Weighted RR for my frontends.",
                "RecordSets" : [
                  {
                    "Name" : "mysite.example.com.",
                    "Type" : "CNAME",
                    "TTL" : "900",
                    "SetIdentifier" : "Frontend One",
                    "Weight" : "4",
                    "ResourceRecords" : ["example-ec2.amazonaws.com"]
                  },
                  {
                    "Name" : "mysite.example.com.",
                    "Type" : "CNAME",
                    "TTL" : "900",
                    "SetIdentifier" : "Frontend Two",
                    "Weight" : "6",
                    "ResourceRecords" : ["example-ec2-larger.amazonaws.com"]
                  }
                  ]
            }
        }
```

Using RecordSetGroup to Set Up an Alias Resource Record Set

This example uses an AWS::Route53::RecordSetGroup (p. 285) to set up an alias resource record set for the "example.com." hosted zone. The `RecordSets` property contains the A record for the zone apex "example.com." The AliasTarget (p. 320) property specifies the hosted zone ID and DNS name for the myELB LoadBalancer by using the GetAtt (p. 324) intrinsic function to retrieve the CanonicalHostedZoneNameID and CanonicalHostedZoneName properties of myELB resource. For more information on alias resource record sets, see Creating Alias Resource Record Sets in the *Route 53 Developer Guide*.

```
      "myELB" : {
        "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
        "Properties" : {
            "AvailabilityZones" : [ "us-east-1a" ],
            "Listeners" : [ {
                "LoadBalancerPort" : "80",
                "InstancePort" : "80",
                "Protocol" : "HTTP"
            } ]
        }
      },
      "myDNS" : {
        "Type" : "AWS::Route53::RecordSetGroup",
        "Properties" : {
          "HostedZoneName" : "example.com.",
          "Comment" : "Zone apex alias targeted to myELB LoadBalancer.",
          "RecordSets" : [
            {
              "Name" : "example.com.",
              "Type" : "A",
              "AliasTarget" : {
                 "HostedZoneId" : { "Fn::GetAtt" : ["myELB", "CanonicalHosted
ZoneNameID"] },
                  "DNSName" : { "Fn::GetAtt" : ["myELB","CanonicalHostedZone
Name"] }
              }
            }
          ]
        }
    }
```

Amazon S3 Template Snippets

Topics

Creating an Amazon S3 Bucket with Defaults

This example uses a AWS::S3::Bucket (p. 286) to create a bucket with default settings.

```
"myS3Bucket" : {
      "Type" : "AWS::S3::Bucket"
      }
```

Creating an Amazon S3 Bucket for Website Hosting and with a DeletionPolicy

This example creates a bucket as a website. The AccessControl property is set to the canned ACL PublicRead (public read permissions are required for buckets set up for website hosting). Because this bucket resource has a DeletionPolicy attribute (p. 322) set to *Retain*, AWS CloudFormation will not delete this bucket when it deletes the stack. The Output section uses *Fn::GetAtt* to retrieve the WebsiteURL attribute and DomainName attribute of the S3Bucket resource.

```
"{
      "AWSTemplateFormatVersion" : "2010-09-09",
      "Resources" : {
          "S3Bucket" : {
             "Type" : "AWS::S3::Bucket",
             "Properties" : {
                 "AccessControl" : "PublicRead",
                 "WebsiteConfiguration" : {
                     "IndexDocument" : "index.html",
                     "ErrorDocument" : "error.html"
                 }
             },
             "DeletionPolicy" : "Retain"
          }
      },
      "Outputs" : {
          "WebsiteURL" : {
                "Value" : { "Fn::GetAtt" : [ "S3Bucket", "WebsiteURL" ] },
                "Description" : "URL for website hosted on S3"
          },
          "S3BucketSecureURL" : {
                "Value" : { "Fn::Join" : [ "", [ "https://", { "Fn::GetAtt" :
[ "S3Bucket", "DomainName" ] } ] ] },
                "Description" : "Name of S3 bucket to hold website content"
          }
      }
}
```

Stack Resource Snippets

Topics

Embedding a Stack in a Template

This example template contains an embedded stack resource called myStack. When AWS CloudFormation creates a stack from the template, it creates the myStack, whose template is specified in the `TemplateURL` property. The output value StackRef returns the stack ID for myStack and the value OutputFromEmbeddedStack returns the output value BucketName from within the myStack resource. The Outputs.*embeddedstackoutputname* format is reserved for specifying output values from embedded stacks and can be used anywhere within the containing template.

For more information, see AWS::CloudFormation::Stack (p. 211).

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStack" : {
        "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
           "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-
us-east-1/S3_Bucket.template",
              "TimeoutInMinutes" : "60"
        }
        }
    },
    "Outputs": {
       "StackRef": {"Value": { "Ref" : "myStack"}},
       "OutputFromEmbeddedStack" : {
             "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
       }
    }
}
```

Embedding a Stack with Input Parameters in a Template

This example template contains a stack resource that specifies input parameters. When AWS CloudFormation creates a stack from this template, it uses the value pairs declared within the Parameters property as the input parameters for the template used to create the myStackWithParams stack. In this example, the InstanceType and KeyName parameters are specified.

For more information, see AWS::CloudFormation::Stack (p. 211).

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myStackWithParams" : {
          "Type" : "AWS::CloudFormation::Stack",
        "Properties" : {
           "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-
us-east-1/EC2ChooseAMI.template",
            "Parameters" : {
                "InstanceType" : "t1.micro",
                "KeyName" : "mykey"
            }
          }
        }
    }
}
```

Wait Condition Template Snippets

Topics

Using a Wait Condition with an Amazon EC2 Instance

This example template declares an Amazon EC2 instance with a wait condition. The wait condition myWaitCondition uses myWaitConditionHandle for signaling, uses the DependsOn attribute (p. 321) to specify that the wait condition will trigger after the Ec2Instance resource has been created, and uses the Timeout property to specify a duration of 4500 seconds for the wait condition. In addition, the presigned URL that signals the wait condition is passed to the EC2 instance with the UserData property of the Ec2Instance resource, thus enabling an application or script running on that EC2 instance to retrieve the pre-signed URL and employ it to signal a success or failure to the wait condition. Note that you need to create the application or script that signals the wait condition. The output value ApplicationData contains the data passed back from the wait condition signal.

For more information, see Creating Wait Conditions in a Template (p. 155), AWS::CloudFormation::WaitCondition (p. 212), and AWS::CloudFormation::WaitConditionHandle (p. 213).

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : {
                "AMI" : "ami-76f0061f"
            },
            "us-west-1" : {
                "AMI" : "ami-655a0a20"
            },
            "eu-west-1" : {
                "AMI" : "ami-7fd4e10b"
            },
            "ap-northeast-1" : {
                "AMI" : "ami-8e08a38f"
            },
            "ap-southeast-1" : {
                "AMI" : "ami-72621c20"
            }
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "UserData" : { "Fn::Base64" : {"Ref" : "myWaitHandle"}},
                "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" :
"AWS::Region" }, "AMI" ]}
            }
        },
        "myWaitHandle" : {
            "Type" : "AWS::CloudFormation::WaitConditionHandle",
            "Properties" : {
            }
        },
        "myWaitCondition" : {
            "Type" : "AWS::CloudFormation::WaitCondition",
            "DependsOn" : "Ec2Instance",
            "Properties" : {
                "Handle" : { "Ref" : "myWaitHandle" },
                "Timeout" : "4500"
            }
        }
    },
    "Outputs" : {
        "ApplicationData" : {
            "Value" : { "Fn::GetAtt" : [ "myWaitCondition", "Data" ]},
            "Description" : "The data passed back as part of signalling the
WaitCondition."
        }
    }
}
```

Using Curl to signal a Wait Condition

This example shows a Curl command line that signals success to a wait conditon.

```
curl -T /tmp/a https://cloudformation-waitcondition-test.s3.amazon
aws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack%2Fstack-
gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-11e0-
81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccessKeyId=AKI
AIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo%3D
```

where the file /tmp/a contains the following JSON structure:

```
{
  "Status" : "SUCCESS",
  "Reason" : "Configuration Complete",
  "UniqueId" : "ID1234",
  "Data" : "Application has completed configuration."
}
```

This example shows a Curl command line that sends the same success signal except it sends the JSON as a parameter on the command line.

```
curl -X PUT -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason" :
 "Configuration Complete","UniqueId" : "ID1234","Data" : "Application has com
pleted configuration."}'  https://cloudformation-waitcondition-test.s3.amazon
aws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack%2Fstack-
gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-11e0-
81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccessKeyId=AKI
AIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo%3D
```

AWS CloudFormation Template Snippets

Topics

Base64 Encoded UserData Property

This example shows the assembly of a UserData property using the Fn::Base64 and Fn::Join functions. The references *`MyValue`* and *`MyName`* are parameters that must be defined in the Parameters section of the template. The literal string *`Hello World`* is just another value this example passes in as part of the *`UserData`*.

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ ",", [
            { "Ref" : "MyValue" },
            { "Ref" : "MyName" },
            "Hello World" ] ]
    }
}
```

Base64 Encoded UserData Property with AccessKey and SecretKey

This example shows the assembly of a UserData property using the Fn::Base64 and Fn::Join functions. It includes the `AccessKey` and `SecretKey` information. The references *`AccessKey`* and *`SecretKey`* are parameters that must be defined in the Parameters section of the template.

```
"UserData" : {
    "Fn::Base64" : {
        "Fn::Join" : [ "", [
            "ACCESS_KEY=", { "Ref" : "AccessKey" },
            "SECRET_KEY=", { "Ref" : "SecretKey" } ]
        ]
    }
}
```

Parameters Section with One Literal String Parameter

The following example depicts a valid Parameters section declaration in which a single String type parameter is declared.

```
"Parameters" : {
    "UserName" : {
        "Type" : "String",
        "Default" : "nonadmin",
       "Description" : "Assume a vanilla user if no command-line spec provided"

    }
}
```

Parameters Section with String Parameter with Regular Expression Constraint

The following example depicts a valid Parameters section declaration in which a single String type parameter is declared. The AdminUserAccount parameter has a default of admin. The parameter value must have a minimum length of 1, a maximum length of 16, and contains alphabetic characters and numbers but must begin with an alphabetic character.

```
"Parameters" : {
    "AdminUserAccount": {
      "Default": "admin",
      "NoEcho": "true",
      "Description" : "The admin account username",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*"
    }
}
```

Parameters Section with Number Parameter with MinValue and MaxValue Constraints

The following example depicts a valid Parameters section declaration in which a single Number type parameter is declared. The WebServerPort parameter has a default of 80 and a minimum value 1 and maximum value 65535.

```
"Parameters" : {
    "WebServerPort": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "MinValue": "1",
      "MaxValue": "65535"
    }
}
```

Parameters Section with Number Parameter with AllowedValues Constraint

The following example depicts a valid Parameters section declaration in which a single Number type parameter is declared. The WebServerPort parameter has a default of 80 and allows only values of 80 and 8888.

```
"Parameters" : {
    "WebServerPortLimited": {
      "Default": "80",
      "Description" : "TCP/IP port for the web server",
      "Type": "Number",
      "AllowedValues" : ["80", "8888"]
    }
}
```

Parameters Section with One Literal CommaDelimitedList Parameter

The following example depicts a valid Parameters section declaration in which a single `CommaDelimitedList` type parameter is declared. The NoEcho property is set to `TRUE`, which will mask its value with asterisks (*****) in the `cfn-describe-stacks` output.

```
"Parameters" : {
    "UserRoles" : {
        "Type" : "CommaDelimitedList",
        "Default" : "guest,newhire",
        "NoEcho" : "TRUE"
    }
}
```

Parameters Section with Parameter Value Based on Pseudo Parameter

This example shows a parameter assignment based on the value returned from the psuedo parameter, "AWS::StackName".

```
"Parameters" : {
    "StackName" : {
        "Type" : "String",
        "Default" : { "Ref" : "AWS::StackName"}
    }
},
```

Mapping Section with Three Mappings

The following example depicts a valid Mapping section declaration that contains three mappings. The map, when matched with a mapping key of *`Stop`*, *`SlowDown`*, or *`Go`*, provides the RGB values assigned to the corresponding *`RGBColor`* attribute.

```
"Mappings" : {
    "LightColor" : {
        "Stop" : {
            "Description" : "red",
            "RGBColor" : "RED 255 GREEN 0 BLUE 0"
        },
        "SlowDown" : {
            "Description" : "yellow",
            "RGBColor" : "RED 255 GREEN 255 BLUE 0"
        },
        "Go" : {
            "Description" : "green",
            "RGBColor" : "RED 0 GREEN 128 BLUE 0"
        }
    }
},
```

Description Based on Literal String

The following example depicts a valid Description section declaration where the value is based on a literal string. This snippet can be for templates, parameters, resources, properties, or outputs.

```
"Description" : "Replace this value"
```

Outputs Section with One Literal String Output

This example shows a output assignment based on a literal string.

```
"Outputs" : {
    "MyPhone" : {
        "Value" : "Please call 555-5555",
        "Description" : "A random message for cfn-describe-stacks"
    }
}
```

Outputs Section with One Resource Reference and One Pseudo Reference Output

This example shows an Outputs section with two output assignments. One is based on a resource, and the other is based on a pseudo reference.

```
"Outputs" : {
    "SNSTopic" : {
        "Value" : "{ "Ref" : "MyNotificationTopic" }
    },

    "StackName" : {
        "Value" : { "Ref" : "AWS::StackName" }
    }
}
```

Outputs Section with an Output Based on a Function, a Literal String, a Reference, and a Pseudo Parameter

This example shows an Outputs section with one output assignment. The Join function is used to concatenate the value, using a percent sign as the delimiter.

```
"Outputs" : {
    "MyOutput" : {
        "Value" : { "Fn::Join" :
            [ "%", [ "A-string", {"Ref" : "AWS::StackName" } ] ]
        }
    }
}
```

Template Format Version

The following snippet depicts a valid Template Format Version section declaration.

```
"AWSTemplateFormatVersion" : "2010-09-09"
```

AWS Tag Property

This example shows an AWS Tag property. You would specify this property within the Properties section of a resource. When the resource is created, it will be tagged with the tags you declare.

```
"Tags" : [
      {
        "Key" : "keyname1",
        "Value" : "value1"
      },
      {
        "Key" : "keyname2",
        "Value" : "value2"
      }
    ]
},
```

Modifying AWS CloudFormation Templates

Topics

Adding Input Parameters to Your Template

You can configure your templates to require input parameters by adding them to the Parameters section. Each parameter you add must contain a value at runtime. You can specify a default value for each parameter to make the parameter an option. If you do not specify a default value, you must provide a value for that parameter when you create the stack.

A parameter can be declared as a *String*, *Number*, or *CommaDelimitedList* type. The *String* and *Number* types can have constraints that AWS CloudFormation uses to validate the value of the parameter. For more information on parameter constraints, see Parameters Declaration (p. 91).

The following sample configures a single parameter, *Email*:

```
"Parameters" : {
    "Email" : {
        "Type" : "String"
    }
}
```

The parameter has no default, so you must provide a value to create the stack. After you create the CloudWatch Alarms stack with a value for *Email*, the `cfn-describe-stacks` command returns the following:

```
STACK  myAlarms
arn:aws:cfn:us-east-1:165024647323:stack/f5b4cbb0-24d7-11e0-93a-
508be05d086/myAlarms
Email=Joe@Joe.com  2011-01-20T20:57:57Z  CREATE_COMPLETE
User Initiated  false  Instance=i-0723826b
```

You can configure the parameter to not display with the *NoEcho* parameter:

```
"Parameters" : {
    "Email" : {
        "Type" : "String",
        "NoEcho" : "TRUE"
    }
}
```

Here's the output from a stack created with the same template, but with the *NoEcho* set to *TRUE*:

```
STACK  myAlarms2
arn:aws:cfn:us-east-1:165024647323:stack/ff6ff540-24db-11e0-94f8-
5081b017c4b/myAlarms2
Email=******  2011-01-20T21:26:52Z  CREATE_COMPLETE  User Initiated
false  Instance=i-f734959b
```

The value for *Email* is masked with asterisks.

To supply the values for parameters, you include the `--`*`parameters`* option to the `cfn-create-stack` command.

For example, the following command adds a value for the *UserName* and *Password* parameters:

```
PROMPT> cfn-create-stack MyStack -f My.Template --parameters "UserName=Joe;Pass
word=JoesPw"
```

Parameters are separated with semicolons. Note that parameter names are case sensitive. If you mistype the parameter name when you run `cfn-create-stack`, AWS CloudFormation will not create the stack, and will report that the template doesn't contain the parameter.

Adding Conditional Values to Your Template with Mappings

You use *`Parameters`*, *`Mappings`*, and the `Fn::FindInMap` function to enable conditional parameter functionality in your template:

1. Add one parameter to your Parameters section for every mapping you want to include. The parameter is how you pass in the desired mapping key.
2. Create the mappings that contain the key options and key values.
3. Use the `Fn::FindInMap` function as the value for the resource property or output you want to assign conditionally.

Note

You can set a default value for both parameters and mappings. You should set a default for at least one of these. If `Fn::FindInMap` cannot resolve a key and value at runtime, an error will prevent the stack from being created.

Consider this example. Suppose you want the `cfn-describe-stacks` command to print the AMI name of the AMI you want to run based on a particular region. You could do this with the following:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "TemplateName - ShortMapExample.template",

  "Parameters" : {
    "Region" : {
      "Default" : "us-east-1",
     "Description" : " 'us-east-1' | 'us-west-1' | 'eu-west-1' | 'ap-southeast-
1' "
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {
          "AMI" : "ami-76f0061f"
      },
      "us-west-1" : {
          "AMI" : "ami-655a0a20"
      },
      "eu-west-1" : {
          "AMI" : "ami-7fd4e10b"
      },
      "ap-southeast-1" : {
          "AMI" : "ami-72621c20"
      }
    }
  },

  "Resources" : {

   ...other resources...

    },

  "Outputs" : {
    "OutVal" : {
      "Description" : "Return the name of the AMI matching the RegionMap key",

      "Value" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "Region" }, "AMI"
 ]}
    }
  }

}
```

The parameter *Region* accepts a string value, ideally one of the region identifiers in the template. The Mappings section declares the *RegionMap* mapping. Each mapping key assigns a value to the *AMI* attribute. The Outputs section declares the *OutVal* output, which gets its value based on the value returned from `Fn:FindInMap`.

The following shows the value assigned to *OutVal* based on the listed command:

Command Lines	**Value Assigned to OutVal**

`cfn-create-stack MyTestStack -f ShortRegionExample.template --parameters "Region=us-west-1"`
`...`
`cfn-describe-stacks MyTestStack`
	ami-655a0a20

`cfn-create-stack MyTestStack -f ShortRegionExample.template --parameters "Region=eu-west-1"`
`...`
`cfn-describe-stacks MyTestStack`
	ami-7fd4e10b

`cfn-create-stack MyTestStack -f ShortMapExample.template`
`...`
`cfn-describe-stacks MyTestStack`
	ami-76f0061f

In the first two cases, the value specified as part of the *--parameters* option determines the value of *OutVal*. In the third example, the mapping key is *us-east-1*, which is declared as the default value for the Region parameter.

Tagging Your Member Resources

AWS CloudFormation automatically tags your resources with the stack name that you can filter on when viewing those resources in the AWS Management Console.

In addition to the stack name tags that AWS CloudFormation adds for you, you can add custom tags to the resources that support tagging.

Note

Tags you add to a member resource do not appear in the output from `cfn-describe-stack-resources`. However, they do appear in the AWS Management Console on the tab for the tagged resource.

Suppose you wanted to customize a template to include the tag *`Stage`* for deployment stage, and *`QA`* for its value. You could write the definition for the *`MyInstance`* resource as follows:

```
"MyInstance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
        "SecurityGroups" : [ { "Ref" : "MySecurityGroup" } ],
        "AvailabilityZone" : "us-east-1a",
        "ImageId" : "ami-20b65349",
        "Volumes" : [
            { "VolumeId" : { "Ref" : "MyEBS" },
                        "Device" : "/dev/sdk" }
        ],
        "Tags" : [
            {
                "Key" : "Stage",
                "Value" : "QA"
            }
       ]
    }
}
```

After you created the stack, you could then filter on the *`Stage`* tag in the AWS Management Console.

Specifying Return Values with Outputs

You can use the template Outputs section to specify custom values that are included in the values returned by `cfn-describe-stacks` command. You specify each custom value according to template property rules (Resource Properties (p. 8)), so you can base their value on literals, parameter references, pseudo parameters, mapping value, and intrinsic functions.

For a simple example, a sample template declares two outputs, *`IPAddress`* and *`InstanceId`*:

```
"Outputs" : {
    "IPAddress" : {
        "Value" : { "Ref" : "MyIp" }
    },

    "InstanceId" : {
        "Value" : { "Ref" : "MyInstance" }
    }
}
```

Both values are based on logical names declared within the template. *IPAddress* refers to the AWS::EC2::EIP type with the logical name *MyIp*, and *InstanceId* refers to the AWS::EC2::Instance type with the logical name *MyInstance*.

After the stack is created, and `cfn-describe-stacks` reports its status as being `CREATE_COMPLETE`, it also reports the following:

```
PROMPT> cfn-describe-stack-resources StackName

STACK  MyIOStack
arn:aws:cfn:us-east-1:165024647323:stack/1fe2a4f0-24c1-11e0-94f8-
5081b017bc4b/MyIOStack
2011-01-20T18:14:30Z  CREATE_COMPLETE  User Initiated  false
IPAddress=184.72.229.56;InstanceId=i-47ab0a2b
```

The custom output values `IPAddress` and `InstanceId` are present at the end of the report.

Creating Wait Conditions in a Template

Using the AWS::CloudFormation::WaitCondition (p. 212) and AWS::CloudFormation::WaitConditionHandle (p. 213) resources, 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.

You can use a wait condition to coordinate the creating stack resources with other configuration actions external to the stack creation. For example, you might want to download and configure applications on an Amazon EC2 instance before considering the creation of that Amazon EC2 instance complete.

This is how a wait condition works:

- AWS CloudFormation creates a wait condition just like any other resource. When AWS CloudFormation creates a wait condition, it reports the wait condition's status as CREATE_IN_PROGRESS and waits until it receives the requisite number of success signals or the wait condition's timeout period has expired. If AWS CloudFormation receives the requisite number of success signals before the time out period expires, it continues creating the stack; otherwise, it sets the wait condition's status to CREATE_FAILED and rolls the stack back.
- The Timeout property determines how long a wait condition lasts. When the wait condition is triggered, AWS CloudFormation waits for a signal for the length of the defined timeout period.
- Typically, you want a wait condition to begin immediately after the creation of a specific resource, such as an Amazon EC2 instance, RDS DB instance, or Auto Scaling group. You do this by adding the DependsOn attribute (p. 321) to a wait condition. When you add a DependsOn attribute to a wait condition, you specify that the wait condition is created only after the creation of a particular resource has completed. When the wait condition is created, AWS CloudFormation begins the timeout period and waits for success signals.
- You can also use the DependsOn attribute on other resources. For example, you may want an RDS DB instance to be created and a database configured on that DB instance first before creating the EC2 instances that use that database. In this case, you create a wait condition that has a DependsOn attribute that specifies the DB instance, and you create EC2 instance resources that have DependsOn attributes that specify the wait condition. This would ensure that the EC2 instances would only be created directly after the DB instance and the wait condition were completed.
- AWS CloudFormation must receive a specified number of success signals for a wait condition before setting that wait condition's status to CREATE_COMPLETE continuing the creation of the stack. The wait condition's Count property specifies the number of success signals. If none is set, the default is 1.
- A wait condition requires a wait condition handle to set up a presigned URL that is used as the signaling mechanism. The presigned URL enables you to send a signal without having to supply your AWS

credentials. You use that presigned URL to signal success or failure, which is encapsulated in a JSON statement. For the format of that JSON statement, see the Wait Condition Signal JSON Format (p. 158). For an example cUrl command that sends a JSON statement to a presigned URL, see Wait Condition Template Snippets (p. 144).

- If a wait condition receives the requisite number of success signals (as defined in the Count property) before the timeout period expires, AWS CloudFormation marks the wait condition as CREATE_COMPLETE and continues creating the stack. Otherwise, AWS CloudFormation fails the wait condition and rolls the stack back (for example, if the timeout period expires without requisite success signals or if a failure signal is received).

To use a wait condition in a stack:

1. Declare an AWS::CloudFormation::WaitConditionHandle resource in the stack's template. A wait condition handle has no properties; however, a reference to a WaitConditionHandle resource resolves to a pre-signed URL that you can use to signal success or failure to the WaitCondition. For example:

```
"myWaitHandle" : {
     "Type" : "AWS::CloudFormation::WaitConditionHandle",
     "Properties" : {
     }
}
```

2. Declare an AWS::CloudFormation::WaitCondition resource in the stack's template. A WaitCondition resource has two required properties: Handle is a reference to a WaitConditionHandle declared in the template and Timeout is the number seconds for AWS CloudFormation to wait. You can optionally set the Count property, which determines the number of success signals that the wait condition must receive before AWS CloudFormation can resume creating the stack.

   To control when the wait condition is triggered, you set a DependsOn attribute on the wait condition. A DependsOn clause associates a resource with the wait condition. After AWS CloudFormation creates the DependsOn resource, it blocks further stack resource creation until one of the following events occur: a) the timeout period expires b) The requisite number of success signals are received c) A failure signal is received.

   Here is an example of a wait condition that begins after the successful creation of the Ec2Instance resource, uses the myWaitHandle resource as the WaitConditionHandle, has a timeout of 4500 seconds, and has the default Count of 1 (since no Count property is specified):

```
"myWaitCondition" : {
    "Type" : "AWS::CloudFormation::WaitCondition",
    "DependsOn" : "Ec2Instance",
    "Properties" : {
        "Handle" : { "Ref" : "myWaitHandle" },
         "Timeout" : "4500"
    }
}
```

3. Get the presigned URL to use for signaling.

   In the template, the presigned URL can be retrieved by passing the logical name of the AWS::CloudFormation::WaitConditionHandle resource to the Ref intrinsic function. For example, you can use the UserData property on AWS::EC2::Instance resources to pass the presigned URL to the Amazon EC2 instances so that scripts or applications running on those instances can signal success or failure to AWS CloudFormation:

```
"UserData" : {
   "Fn::Base64" : {
       "Fn::Join" : [ "", ["SignalURL=", { "Ref" : "myWaitHandle" } ] ]
   }
}
```

Note: In the AWS Management Console or the AWS CloudFormation command line tools, the presigned URL is displayed as the physical ID of the wait condition handle resource.

4. Select a method for detecting when the stack enters the wait condition.

   If you create the stack with notifications enabled, AWS CloudFormation publishes a notification for every stack event to the specified topic. If you or your application subscribe to that topic, you can monitor the notifications for the wait condition handle creation event and retrieve the presigned URL from the notification message.

   You can also monitor the stack's events using the AWS Management Console, the AWS CloudFormation command line tools, or the AWS CloudFormation API.

5. Use the presigned URL to signal success or failure.

   To send a signal, you send an HTTP request message using the presigned URL. The request method must be PUT and the Content-Type header must be an empty string or omitted. The request message must be a JSON structure of the form specified in Wait Condition Signal JSON Format (p. 158).

   You need to send the number of success signals specified by the Count property in order for AWS CloudFormation to continue stack creation. If you have a Count that is greater than 1, the UniqueId value for each signal must be unique across all signals sent to a particular wait condition.

   A Curl command is one way to send a signal. The following example shows a Curl command line that signals success to a wait condition.

```
curl -T /tmp/a https://cloudformation-waitcondition-test.s3.amazon
aws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A034017226601%3Astack%2Fstack-
gosar-20110427004224-test-stack-with-WaitCondition--VEYW%2Fe498ce60-70a1-
11e0-81a7-5081d0136786%2FmyWaitConditionHandle?Expires=1303976584&AWSAccess
KeyId=AKIAIOSFODNN7EXAMPLE&Signature=ik1twT6hpS4cgNAw7wyOoRejVoo%3D
```

   where the file /tmp/a contains the following JSON structure:

```
{
  "Status" : "SUCCESS",
  "Reason" : "Configuration Complete",
  "UniqueId" : "ID1234",
  "Data" : "Application has completed configuration."
}
```

   This example shows a Curl command line that sends the same success signal except it sends the JSON structure as a parameter on the command line.

```
curl -X PUT -H 'Content-Type:' --data-binary '{"Status" : "SUCCESS","Reason"
 : "Configuration Complete","UniqueId" : "ID1234","Data" : "Application has
 completed configuration."}'  https://cloudformation-waitcondition-
test.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-
1%3A034017226601%3Astack%2Fstack-gosar-20110427004224-test-stack-with-Wait
Condition--VEYW%2Fe498ce60-70a1-11e0-81a7-5081d0136786%2FmyWaitCondition
Handle?Expires=1303976584&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signa
ture=ik1twT6hpS4cgNAw7wyOoRejVoo%3D
```

Wait Condition Signal JSON Format

When you signal a wait condition, you must use the following JSON format:

```
{
  "Status" : "StatusValue",
  "UniqueId" : "Some UniqueId",
  "Data" : "Some Data",
  "Reason" : "Some Reason"
  }
```

Where:

StatusValue must be one of the following values:

- *SUCCESS* indicates a success signal.
- *FAILURE* indicates a failure signal and triggers a failed wait condition and a stack rollback.

UniqueId identifies the signal to AWS CloudFormation. If the Count property of the wait condition is greater than 1, the UniqueId value must be unique across all signals sent for a particular wait condition; otherwise, AWS CloudFormation will consider the signal a retransmission of the previously sent signal with the same UniqueId, and it will ignore the signal.

Data is any information that you want to send back with the signal. The Data value can be accessed by calling the Fn::GetAtt function (p. 324) within the the template. For example, if you create the following output value for the wait condition mywaitcondition, you can use the cfn-describe-stacks command (p. 347), DescribeStacks action, or Outputs tab of the CloudFormation console to view the Data sent by valid signals sent to AWS CloudFormation:

```

        "WaitConditionData" : {
            "Value" : { "Fn::GetAtt" : [ "mywaitcondition", "Data" ]},
            "Description" : "The data passed back as part of signalling the 
WaitCondition"
        },

```

The Fn::GetAtt function returns the UniqueId and Data as a name/value pair within a JSON structure. The following is an example of the Data attribute returned by the WaitConditionData output value defined above:

```
{"Signal1":"Application has completed configuration."}
```

Reason is a string with no other restrictions on its content besides JSON compliance.

AWS CloudFormation Endpoints

To reduce data latency in your applications, most Amazon Web Services products allow you to select a regional endpoint to make your requests. An endpoint is a URL that is the entry point for a web service.

The AWS CloudFormation endpoints are:

Region	Endpoint
US East (Northern Virginia) Region	cloudformation.us-east-1.amazonaws.com
US West (Oregon) Region	cloudformation.us-west-2.amazonaws.com
US West (Northern California) Region	cloudformation.us-west-1.amazonaws.com
EU (Ireland) Region	cloudformation.eu-west-1.amazonaws.com
Asia Pacific (Singapore) Region	cloudformation.ap-southeast-1.amazonaws.com
Asia Pacific (Tokyo) Region	cloudformation.ap-northeast-1.amazonaws.com
South America (Sao Paulo) Region	cloudformation.sa-east-1.amazonaws.com

Note

All AWS CloudFormation endpoints use the HTTPS protocol for access.

For more information about regions and endpoints for AWS CloudFormation and other services, go to Regions and Endpoints in the *Amazon Web Services General Reference*.

Using Regular Expressions in AWS CloudFormation Templates

Regular expressions (commonly known as regexes) can be specified in a number of places within an AWS CloudFormation template, such as for the AllowedPattern property when creating a template parameter (p. 91).

Regular expressions in AWS CloudFormation conform to the Java regular expression syntax. A full description of this syntax and its constructs can be viewed in the Java documentation, here: java.util.regex.Pattern.

Important

Since AWS CloudFormation templates use the JSON syntax for specifying objects and data, you will need to add an additional backslash to any backslash characters in your regular expression, or JSON will interpret these as escape characters.

For example, if you include a `\d` in your regular expression to match a digit character, you will need to write it as `\\d` in your template.

Controlling User Access with AWS Identity and Access Management

AWS CloudFormation integrates with AWS Identity and Access Management (IAM), a service that lets you do the following:

- Create users and groups under your organization's AWS account
- Easily share your AWS account resources between the users in the account
- Assign unique security credentials to each user

- Granularly control users access to services and resources
- Get a single AWS bill for all users in the AWS account
- Assign a role to an EC2 instance for secure access to your AWS Services from an application.

During the validation of your template, AWS CloudFormation also checks your template for capabilities that you should be aware of and acknowledge before creating the stack. Currently, AWS CloudFormation checks only for IAM resource capabilities. If your template contains IAM resources, AWS CloudFormation will fail to create the stack if you do not acknowledge the template's creation of IAM resources.

Before you create a stack from such a template, you should be sure that you trust the source of the template since IAM resources can create or modify users, groups, and policies that control access to your AWS resources. If you decide to create or update a stack from a template containing IAM resources, you must acknowledge that capability during that operation. You can acknowledge that capability using the AWS CloudFormation console, command line, or API.

- In the AWS CloudFormation console, check **I acknowledge that this template may create IAM resources** on the **Specify Parameters** page of the Create Stack or Update Stack wizards.
- With the cfn-create-stack (p. 342) and cfn-update-stack (p. 360) commands, you must specify the *`CAPABILITY_IAM`* value for the *`--capabilities`* parameter.
- With the CreateStack and UpdateStack APIs, you must specify the parameter *`Capabilities.member.1=CAPABILITY_IAM`*.

For example, you can use IAM with Amazon EC2 to control which users in your AWS account can create AMIs or launch instances.

Example 1: Allow only the group `RegisteredDevelopers` to access AWS CloudFormation APIs.

In this example, we create the following policy to attach to the `RegisteredDevelopers` group:

```
{
    "Statement":[ {
        "Effect":"Allow",
        "Action":[
            "cloudformation:CreateStack",
            "cloudformation:DeleteStack",
            "cloudformation:DescribeStacks",
            "cloudformation:DescribeStackEvents",
            "cloudformation:DescribeStackResources",
            "cloudformation:GetTemplate",
            "cloudformation:VerifyTemplate"
        ],
        "Resource":"*",
    } ]
}
```

Only the developers in the `RegisteredDevelopers` group will be able to call the APIs listed.

In addition to the permissions required to access the AWS CloudFormation APIs itemized in the policy, the developers in the `RegisteredDevelopers` group will also have to have permission to create and describe the individual resources declared in the template. For example, if the template declares an Amazon SQS Queue, the developer creating a stack based on that template will need to have corresponding IAM permission for Amazon SQS.

AWS CloudFormation supports the following IAM resources:

- AWS::IAM::AccessKey (p. 262)
- AWS::IAM::Group (p. 263)
- AWS::IAM::Policy (p. 266)
- AWS::IAM::User (p. 271)
- AWS::IAM::UserToGroupAddition (p. 273)
- AWS::IAM::Role (p. 268)
- AWS::IAM::InstanceProfile (p. 264)

For more information about IAM, go to:

- Identity and Access Management (IAM)
- AWS Identity and Access Management Getting Started Guide
- Using AWS Identity and Access Management

Automating Application Installation Using AWS CloudFormation and Cloud-Init

By Chris Whitaker, May 2011

This section shows how the Amazon Linux AMI can be used along with AWS CloudFormation to start up and configure an application dynamically at boot time. The example uses the new WaitCondition resource supported by AWS CloudFormation to wait for a Ruby on Rails application to be configured and launched before the stack is considered to be successfully created. The example also takes advantage of the Amazon Linux AMI support for Cloud-init, an open source application built by Canonical. Cloud-init enables you to use the Amazon Elastic Compute Cloud (Amazon EC2) *UserData* parameter to specify actions to run on your instance at boot time. The example uses this mechanism to specify the application configuration commands and the command to signal success to the wait condition so that stack creation can continue.

Note

Alternatively, you can use AWS CloudFormation helper scripts (cfn-init and cfn-signal) to automate the installation of your application. For more information, see Deploying Applications with AWS CloudFormation (p. 167).

First, let’s create a simple Rails application. Start by typing the command:

```
$ rails new <path to application>
```

Before you can execute this command on an Amazon Linux instance, you must install several packages and RubyGems. Here is the full set of commands you need to create a simple application:

```
#!/bin/bash -ex
yum -y install gcc-c++ make
yum -y install mysql-devel sqlite-devel
yum -y install ruby-rdoc rubygems ruby-mysql ruby-devel
gem install --no-ri --no-rdoc rails
gem install --no-ri --no-rdoc mysql
gem install --no-ri --no-rdoc sqlite3
rails new myapp
```

```
cd myapp
rails server -d
```

Typically, you can start a new Amazon EC2 instance running the Amazon Linux AMI from the AWS Management Console. In the console you can SSH to the instance, and type the commands listed above to set up and run a Rails application.

By using AWS CloudFormation to create your application, you can gain several advantages:

1. To use the application you created, you need to allow access to various TCP/IP ports on your Amazon EC2 instance. In particular, you must open port 3000 so you can connect to the Rails application. You might need to open port 22 so you can SSH to your instance to manage the instance when it is up and running. To enable access to these ports, you need to create an Amazon EC2 instance, plus you need to have an appropriately configured EC2 Security Group. AWS CloudFormation lets you define an Amazon EC2 security group alongside your instance; this lets you keep your application's entire AWS resource configuration in one place.
2. You can use the AWS CloudFormation template to create multiple instances of your application. Each instance is guaranteed to be the same as the others. All of your application configuration and installation scripts are kept in a single place. By taking advantage of various facilities in the template, you can use the same template to create instances of your application in different Amazon EC2 Regions. For example, you can create an instance in the US-East (Northern Virginia) Region and one in the EU (Ireland) Region, safe in the knowledge that the applications are configured identically.
3. By using the new WaitCondition resource in the AWS CloudFormation template, you know exactly when the application is ready to accept traffic.

The following code shows the full template for creating and configuring a sample Rails application in any Amazon EC2 Region.

```
{
 "AWSTemplateFormatVersion" : "2010-09-09",
 "Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to
 the instance",
      "Type" : "String"
    }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {"AMI" : "ami-8c1fece5"},
      "us-west-1" : {"AMI" : "ami-3bc9997e"},
      "eu-west-1" : {"AMI" : "ami-47cefa33"},
      "ap-southeast-1" : {"AMI" : "ami-6af08e38"},
      "ap-northeast-1" : {"AMI" : "ami-300ca731"}
    }
  },

  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region"
 }, "AMI" ]},
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["",[
            "#!/bin/bash -ex","\n",
            "yum -y install gcc-c++ make","\n",
            "yum -y install mysql-devel sqlite-devel","\n",
            "yum -y install ruby-rdoc rubygems ruby-mysql ruby-devel","\n",
            "gem install --no-ri --no-rdoc rails","\n",
            "gem install --no-ri --no-rdoc mysql","\n",
            "gem install --no-ri --no-rdoc sqlite3","\n",
            "rails new myapp","\n",
            "cd myapp","\n",
            "rails server -d","\n",
            "curl -X PUT -H 'Content-Type:' --data-binary '{\"Status\" :
\"SUCCESS\",",
                                                           "\"Reason\" : \"The
 application myapp is ready\",",
                                                           "\"UniqueId\" :
\"myapp\",",
                                                        "\"Data\" : \"Done\"}'
 ",
                  "\"", {"Ref" : "WaitForInstanceWaitHandle"},"\"\n" ]]}}
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable Access to Rails application via port 3000
 and SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        }, {
          "IpProtocol" : "tcp",
          "FromPort" : "3000",
          "ToPort" : "3000",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    },

    "WaitForInstanceWaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle",
      "Properties" : {
      }
    },

    "WaitForInstance" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "Ec2Instance",
      "Properties" : {
        "Handle" : {"Ref" : "WaitForInstanceWaitHandle"},
        "Timeout" : "600"
      }
    }
  },
  "Outputs" : {
    "WebsiteURL" : {
      "Description" : "The URL for the newly created Rails application",
      "Value" : { "Fn::Join" : ["", [ "http://", { "Fn::GetAtt" : [ "Ec2In
stance", "PublicIp" ] }, ":3000" ]]}
    }
  }
}
```

The template assumes that you want to have SSH access to the running Amazon EC2 instance. It requires you to enter the name of an existing Amazon EC2 key pair from your account as an input parameter when you create the stack:

```
"Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to
 the instance",
      "Type" : "String"
    }
},
```

The `KeyName` you type in is referenced in the Amazon EC2 instance resource definition:

```
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
...
```

If you do not already have a key pair, you can create a new one by going to the AWS Management Console and opening the Amazon EC2 console. Remember to save the key file that you create so you can use it to SSH to the instance later. For more information on creating and using Amazon EC2 key pairs, see Getting an SSH Key Pair in the *Amazon EC2 User Guide*.

AMI IDs are Region-specific, so the actual AMI to launch depends on the Region in which the stack is created. This template uses the “Mappings” feature to select the correct AMI based on the Region:

```
    "Mappings" : {
    "RegionMap" : {
      "us-east-1" : {"AMI" : "ami-8c1fece5"},
      "us-west-1" : {"AMI" : "ami-3bc9997e"},
      "eu-west-1" : {"AMI" : "ami-47cefa33"},
      "ap-southeast-1" : {"AMI" : "ami-6af08e38"},
      "ap-northeast-1" : {"AMI" : "ami-300ca731"}
    }
  },
```

The actual AMI to use is defined in the Amazon EC2 instance resource definition using the *`FindInMap`* intrinsic function along with the pseudo parameter *`AWS::Region`* which returns a string representing the Region in which the stack is being built:

```
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
         ...
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region"
 }, "AMI" ]},
         ...
      }
    },
```

To open the ports for SSH access (TCP/IP port 22) and to allow access to the newly created Rails application (TCP/IP port 3000), the template defines a new security group:

```
    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable Access to Rails application via port 3000
 and SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        }, {
          "IpProtocol" : "tcp",
          "FromPort" : "3000",
          "ToPort" : "3000",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    },
```

This is referenced in the Amazon EC2 instance resource definition:

```
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
          ...
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
          ...
      }
    },
```

Finally, to complete the application configuration, you need to configure Cloud-init to install the required packages and RubyGems and start the application. Cloud-init uses the Amazon EC2 *`UserData`* field to pass configuration information. If the *`UserData`* field starts with `#!`, the contents of the *`UserData`* are assumed to contain a script to execute on startup. All of the text in a CloudFormation template must conform to a JSON structure; therefore, it is necessary to escape some of the script. The sample template uses the *`Fn::Base64`* function to base64 encode the user data and to allow parameters and references from the template to be substituted in the script at runtime (in this case a reference to the WaitConditionHandle). The example uses the *`Fn::Join`* function to concatenate the various pieces of the script.

```


        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["",[
            "#!/bin/bash -ex","\n",
            "yum -y install gcc-c++ make","\n",
            "yum -y install mysql-devel sqlite-devel","\n",
            "yum -y install ruby-rdoc rubygems ruby-mysql ruby-devel","\n",
            "gem install --no-ri --no-rdoc rails","\n",
            "gem install --no-ri --no-rdoc mysql","\n",
            "gem install --no-ri --no-rdoc sqlite3","\n",
            "rails new myapp","\n",
            "cd myapp","\n",
            "rails server -d","\n",
            "curl -X PUT -H 'Content-Type:' --data-binary '{\"Status\" :
\"SUCCESS\",",
                                                                  "\"Reason\" : \"The
 application myapp is ready\",",
                                                                  "\"UniqueId\" :
\"myapp\",",
                                                               "\"Data\" : \"Done\"}'
 ",
                  "\"", {"Ref" : "WaitForInstanceWaitHandle"},"\"\n" ]]}}
```

So that the stack does not indicate *CREATE_COMPLETE* until the packages have been installed and the application is running, we are using the new WaitCondition resource. In the Amazon EC2 instance resource definition above, you can see that the last line in the *UserData* script is a *CURL* command that signals the WaitCondition using the WaitConditionHandle resource called WaitForInstanceWaitHandle.

The WaitCondition itself is defined as follows:

```
    "WaitForInstance" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "Ec2Instance",
      "Properties" : {
        "Handle" : {"Ref" : "WaitForInstanceWaitHandle"},
        "Timeout" : "600"
      }
    }
```

The WaitCondition definition uses the *DependsOn* construct. This ensures that the WaitForInstance WaitCondition resource is only created directly after the EC2 instance resource is created. Why is this important? The Timeout value specified in the WaitCondition (in this case 600 seconds) starts ticking when the WaitCondition object is put into the *CREATE_IN_PROGRESS* state. In this template, we want to give the Ruby application some time to start, but not too much time (in case something bad happened with the instance). By making the WaitCondition dependent on the Amazon EC2 instance, the WaitCondition resource will only be created after the EC2 instance enters the EC2 running state and the Cloud-init script starts. Using *DependsOn* ensures that the configuration script has 600 seconds to run. The stack creation will fail when the WaitCondition timeout triggers if the script does not signal via the *CURL* command.

NOTE: The WaitCondition resource can be used to synchronize creation of other resources in the template, not just stack creation. For example, you might chose not to associate the instance with an Elastic IP address until the application is running. By adding a DependsOn clause in other resources in the template that refer to the WaitCondition, you ensure that the resources that depend on the WaitCondition are not created until the WaitCondition is signaled. Watch for more articles about passing data back from the application in the template, as well as using WaitCondition objects to wait for multiple instances to be up and running before the application is considered healthy.

If you want to download, modify, or try out the Rails sample template, it is available as part of the AWS CloudFormation sample templates (US-East). To launch a new Rails application in the US-East (Northern Virginia) Region, you can simply click the following link: Launch the Rails sample in the AWS Management Console.

Related Resources

- Amazon EC2 AMI Basics in the *Amazon Elastic Compute Cloud User Guide*
- Amazon Linux AMI Basics in the *Amazon Elastic Compute Cloud User Guide*
- Deploying Applications with AWS CloudFormation (p. 167)
- Ubuntu CloudInit documentation
- Using Amazon's CloudFormation, cloud-init, chef and fog to automate infrastructure

Deploying Applications with AWS CloudFormation

Topics

AWS CloudFormation Deploying Applications Overview

You can use AWS CloudFormation to automatically install, configure, and start up your applications. Doing so can save you a lot of time and effort. For example, there are a number of ways you could deploy an Amazon instance to run a LAMP (Linux, Apache, MySQL, and PHP) web server:

- Manual installation. A manual installation is reliable, but it is also time-consuming. You can reduce the effort by choosing an AMI that already contains all the necessary software, but you will still need to connect to the resulting Amazon EC2 instance to complete the configuration process and start your applications.
- Cloud-init. The latest Linux AMIs contain cloud-init, an open source program that automatically configures and starts your applications according to a script that you provide. If you want to change the configuration or install new applications or updates, you must connect to the instance and make the changes manually.

Using AWS CloudFormation to automate deployment of your application provides two advantages. First, it's easier to duplicate a deployment when all of the installation, configuration, and startup commands are included in the CloudFormation template. Second, you can update an existing installation without connecting directly to the instance.

AWS CloudFormation includes a set of helper applications (cfn-init, cfn-signal, cfn-get-metadata, and cfn-hup) that are based on cloud-init. These helper applications not only provide functionality similar to cloud-init, but also allow you to update your metadata after your instance and applications are up and running. You can update the metadata after deployment because AWS CloudFormation stores the metadata. This added flexibility does require some additional setup—you need to create security credentials for the instance so that the instance can call the CloudFormation API to retrieve the updated metadata.

The following sections describe how to create a template that describes a LAMP stack and uses cfn-init to install, configure and start Apache, MySQL, and PHP. We'll start with a simple template that sets up a basic Amazon EC2 instance running Linux, and then we'll continue adding to the template until it describes a full LAMP stack with deployment instructions.

Create a Basic Amazon EC2 Instance Using CloudFormation

We start with a basic template that defines a single Amazon EC2 instance with a security group that allows SSH traffic on port 22 and HTTP traffic on port 80. The template consists of five top-level JSON objects:

- **Description—**EC2_Single_Instance

  EC2_Single_Instance specifies a single EC2 instance and a corresponding security group that allows SSH and HTTP access.
- **Parameters—**KeyName and InstanceType

  The KeyName parameter specifies the EC2 keypair to use for SSH access, and the InstanceType parameter specifies the type of EC2 instance.
- **Mappings—**AWSInstanceType2Arch and AWSRegionArch2AMI

  AWSInstanceType2Arch maps the proper architecture to the instance size so that the template user need specify only the instance size. AWSRegionArch2AMI maps AMI IDs to their specific region so that template users do not need to research which AMI IDs are available in each region.
- **Resources—**WebServer and WebServerSecurityGroup

  The WebServer resource defines the Amazon EC2 instance and the WebServerSecurityGroup defines the security group that allows incoming traffic on ports 22 (SSH) and 80 (HTTP).
- **Outputs—**WebsiteURL

  The WebsiteURL returns the public URL for the newly created website.

Here is the template:

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template EC2_Single_Instance:
Create a single EC2 instance. **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.",

  "Parameters" : {

    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to
 the instances",
      "Type" : "String",
      "MinLength": "1",
      "MaxLength": "64",
      "AllowedPattern" : "[-_ a-zA-Z0-9]*",
      "ConstraintDescription" : "can contain only alphanumeric characters,
spaces, dashes and underscores."
    },

    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "m1.small",
      "AllowedValues" : [ "t1.micro", "m1.small", "m1.large", "m1.xlarge",
"m2.xlarge", "m2.2xlarge", "m2.4xlarge", "c1.medium", "c1.xlarge", "cc1.4xlarge"
 ],
      "ConstraintDescription" : "must be a valid EC2 instance type."
    }

  },

  "Mappings" : {
    "AWSInstanceType2Arch" : {
      "t1.micro"    : { "Arch" : "32" },
      "m1.small"    : { "Arch" : "32" },
      "m1.large"    : { "Arch" : "64" },
      "m1.xlarge"   : { "Arch" : "64" },
      "m2.xlarge"   : { "Arch" : "64" },
      "m2.2xlarge"  : { "Arch" : "64" },
      "m2.4xlarge"  : { "Arch" : "64" },
      "c1.medium"   : { "Arch" : "32" },
      "c1.xlarge"   : { "Arch" : "64" },
      "cc1.4xlarge" : { "Arch" : "64" }
    },
    "AWSRegionArch2AMI" : {
      "us-east-1"      : { "32" : "ami-7f418316", "64" : "ami-7341831a" },
      "us-west-1"      : { "32" : "ami-951945d0", "64" : "ami-971945d2" },
      "us-west-2"      : { "32" : "ami-16fd7026", "64" : "ami-10fd7020" },
      "eu-west-1"      : { "32" : "ami-24506250", "64" : "ami-20506254" },
      "sa-east-1"      : { "32" : "ami-3e3be423", "64" : "ami-3c3be421" },
      "ap-southeast-1" : { "32" : "ami-74dda626", "64" : "ami-7edda62c" },
      "ap-northeast-1" : { "32" : "ami-dcfa4edd", "64" : "ami-e8fa4ee9" }
    }
  },

  "Resources" : {

    "WebServer": {
      "Type": "AWS::EC2::Instance",

      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref"
 : "InstanceType" }, "Arch" ] } ] },
        "InstanceType"   : { "Ref" : "InstanceType" },
        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],
        "KeyName"        : { "Ref" : "KeyName" }

      }
    },


    "WebServerSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable HTTP access via port 80",
        "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"}
        ]
      }
    }
  },

  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["http://", { "Fn::GetAtt" : [ "WebServer",
 "PublicDnsName" ]}]] },
      "Description" : "URL for newly created EC2 Instance"
    }
  }
}
```

Automate LAMP Installation Using CloudFormation

In this section, we'll build on the basic Amazon EC2 template to create a LAMP template that automatically installs Apache, MySQL, and PHP. In later sections, we'll configure and start the applications. Before we add the necessary metadata and shell commands to the template, we need to set up an IAM user with permissions to read the AWS CloudFormation metadata.

In order to read the metadata from an Amazon EC2 instance created in the template, we recommend creating an IAM user in the template and passing the AWS access key ID and secret access key to the instance as metadata. Give the IAM user permission to call only the DescribeStackResource action. Creating an IAM user locks down the account so that it can only retrieve metadata, but it can perform no other action. By using the IAM features in the template, the existence of the user is tied to the lifetime of the stack. Each new stack will have a separate and unique user. For more information about using IAM in AWS CloudFormation, see Controlling User Access with AWS Identity and Access Management (p. 159).

The updated template includes several new sections. The stack created by this template installs Apache, MySQL, and PHP, but it does not configure or start any of them. The new template sections are as follows:

- **IAM Resources—**`CfnUser` and `HostKeys`

  These resources define an IAM user with limited access rights and a life span that is tied to the lifetime of the stack.
- **Metadata Key—**`AWS::CloudFormation::Init`

  Cfn-init reads this metadata key and installs the packages listed in this key (e.g., httpd, mysql, and php). Cfn-init also retrieves and expands files listed as sources.
- **Properties Resource—**`UserData`

  The UserData key allows you to execute shell commands. This template issues two shell commands: the first command installs the AWS CloudFormation helper scripts; the second executes the cfn-init script.

The following example template creates a LAMP template that automatically installs Apache, MySQL, and PHP. Sections marked with an ellipsis (...) are omitted for brevity. Additions to the template are shown in red italic text.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template LAMP_Install_Only: ...",


  "Parameters" : {

    "KeyName" : { ... },

    "InstanceType" : { ...  },

  "Mappings" : { ... },

  "Resources" : {

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

    "HostKeys" : {
      "Type" : "AWS::IAM::AccessKey",
      "Properties" : {
        "UserName" : {"Ref": "CfnUser"}
      }
    },

    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : {
        "Comment1" : "Configure the bootstrap helpers to install the Apache Web
 Server and PHP",
       "Comment2" : "The website content is downloaded from the CloudFormation
PHPSample.zip file",

        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
                "mysql"        : [],
                "mysql-server" : [],
                "mysql-libs"   : [],
                "httpd"        : [],
                "php"          : [],
                "php-mysql"    : []
              }
            },
            "sources" : {
              "/var/www/html" : "https://s3.amazonaws.com/cloudformation-ex
amples/CloudFormationPHPSample.zip"
            }
          }
        }
      },
      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref"
 : "InstanceType" }, "Arch" ] } ] },
        "InstanceType"   : { "Ref" : "InstanceType" },
        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],
        "KeyName"        : { "Ref" : "KeyName" },
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
          "#!/bin/bash -v\n",
          "yum update -y aws-cfn-bootstrap\n",

          "# Install LAMP packages\n",
          "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" }, " -r Web
Server ",
          "    --access-key ",  { "Ref" : "HostKeys" },
         "    --secret-key ", {"Fn::GetAtt": ["HostKeys", "SecretAccessKey"]},

          "    --region ", { "Ref" : "AWS::Region" }, " || error_exit 'Failed
to run cfn-init'\n"
    ]]}}
      }
    },


    "WebServerSecurityGroup" : { ...  }
  },

  "Outputs" : { ... }
}
```

Automate LAMP Configuration Using CloudFormation

In this section, we'll use AWS CloudFormation to automatically configure and start Apache, MySQL, and PHP.

- **Parameters—**`DBName`, `DBUsername`, `DBPassword`, and `DBRootPassword`

  These parameters allow the template user to specify database names and passwords. The parameters also contain constraints that catch incorrectly formatted values before stack creation begins.
- **Metadata—**`files` and `services`

  The `files` metadata key creates a MySQL setup file. The services metadata key ensures that the httpd and mysqld services are not only running when cfn-init finishes (ensureRunning is set to true); but that they are also restarted upon reboot (enabled is set to true).
- **Properties Resource—**`UserData`

The `UserData` key contains the MySQL shell commands that create a database and a user now follow the call to cfn-init. Additionally, shell commands to configure PHP follow the MySQL shell commands.

The following example template creates a LAMP stack, automatically installs Apache, MySQL, and PHP, and then configures and starts each service. Sections marked with an ellipsis (...) are omitted for brevity. Additions to the template are shown in red italic text.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template LAMP_Single_Instance:
Create a LAMP stack using a single EC2 instance and a local MySQL database for
 storage. This template demonstrates using the AWS CloudFormation bootstrap
scripts to install the packages and files necessary to deploy the Apache web
server, PHP and MySQL at instance launch time. **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.",

  "Parameters" : {

    "KeyName" : { ...  },

    "DBName": {
      "Default": "MyDatabase",
      "Description" : "MySQL database name",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "64",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
      "ConstraintDescription" : "must begin with a letter and contain only al
phanumeric characters."
    },

    "DBUsername": {
      "NoEcho": "true",
      "Description" : "Username for MySQL database access",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "16",
      "AllowedPattern" : "[a-zA-Z][a-zA-Z0-9]*",
      "ConstraintDescription" : "must begin with a letter and contain only al
phanumeric characters."
    },

    "DBPassword": {
      "NoEcho": "true",
      "Description" : "Password for MySQL database access",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*",
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },

    "DBRootPassword": {
      "NoEcho": "true",
      "Description" : "Root password for MySQL",
      "Type": "String",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern" : "[a-zA-Z0-9]*",
      "ConstraintDescription" : "must contain only alphanumeric characters."
    },

    "InstanceType" : { ...  }

  },

  "Mappings" : {
  ...
  },

  "Resources" : {

    "CfnUser" : { ... },

    "HostKeys" : { ... },

    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : {
        "Comment1" : "Configure the bootstrap helpers to install the Apache Web
 Server and PHP",
        "Comment2" : "The website content is downloaded from the CloudFormation
PHPSample.zip file",

        "AWS::CloudFormation::Init" : {
          "config" : {
            "packages" : {
              "yum" : {
                "mysql"        : [],
                "mysql-server" : [],
                "mysql-libs"   : [],
                "httpd"        : [],
                "php"          : [],
                "php-mysql"    : []
              }
            },

            "sources" : {
              "/var/www/html" : "https://s3.amazonaws.com/cloudformation-ex
amples/CloudFormationPHPSample.zip"
            },

            "files" : {
              "/tmp/setup.mysql" : {
                "content" : { "Fn::Join" : ["", [
                  "CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",
                  "GRANT ALL ON ", { "Ref" : "DBName" }, ".* TO '", { "Ref" :
"DBUsername" }, "'@localhost IDENTIFIED BY '", { "Ref" : "DBPassword" }, "';\n"

                  ]]},
                "mode"  : "000644",
                "owner" : "root",
                "group" : "root"
              }
            },

            "services" : {
              "sysvinit" : {
                "mysqld" : {
                  "enabled"       : "true",
                  "ensureRunning" : "true"
                },
                "httpd" : {
                  "enabled"       : "true",
                  "ensureRunning" : "true"
                }
              }

            }
          }
        }
      },
      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref"
 : "InstanceType" }, "Arch" ] } ] },
        "InstanceType"   : { "Ref" : "InstanceType" },
        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],
        "KeyName"        : { "Ref" : "KeyName" },
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
          "#!/bin/bash -v\n",
          "yum update -y aws-cfn-bootstrap\n",

          "# Install LAMP packages\n",
          "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" }, " -r Web
Server ",
          "    --access-key ",  { "Ref" : "HostKeys" },
         "    --secret-key ", {"Fn::GetAtt": ["HostKeys", "SecretAccessKey"]},

          "    --region ", { "Ref" : "AWS::Region" }, "\n",

          "# Setup MySQL, create a user and a database\n",
          "mysqladmin -u root password '", { "Ref" : "DBRootPassword" }, "\n",

          "mysql -u root --password='", { "Ref" : "DBRootPassword" }, "\n",

          "# Configure the PHP application - in this case, fixup the page with
 the right references to the database\n",
          "sed -i \"s/REPLACE_WITH_DATABASE/localhost/g\" /var/www/html/in
dex.php\n",
          "sed -i \"s/REPLACE_WITH_DBUSER/", { "Ref" : "DBUsername" }, "/g\"
/var/www/html/index.php\n",
         "sed -i \"s/REPLACE_WITH_DBPASSWORD/", { "Ref" : "DBPassword" }, "/g\"
 /var/www/html/index.php\n"

        ]]}}
      }
    },

    "WebServerSecurityGroup" : { ... }
```

```
  },

  "Outputs" : { ... }
}
```

Use AWS CloudFormation Wait Conditions

The template so far will create the stack resources and attempt to start all of the specified services. At that point, it will mark the status of the stack as CREATE_COMPLETE, even if one or more services failed to start. To prevent the status from changing to CREATE_COMPLETE until all the services have successfully started, AWS CloudFormation supports a WaitCondition resource that you can use to ensure that either all of your stack resources are created or none of them are. The WaitCondition resource pauses execution of the template until a specified condition is met or a timeout period is exceeded.

To wait for the application to be ready, we can extend the previous template to create a WaitCondition and a WaitConditionHandle and use the cfn-signal helper script to signal that the application is installed. The WaitConditionHandle resource is a signaling mechanism that notifies the WaitCondition resource of status changes that determine if deployment can continue or must be rolled back. For more information about wait conditions, see Creating Wait Conditions in a Template (p. 155).

To add the WaitCondition and WaitConditionHandle resources, we'll add the following to the template:

- **Resources > WebServer > Properties—**`UserData error_exit` helper function

  The error_exit function calls cfn-signal with an exit code that signals an error. The UserData script invokes error_exit if cfn-init fails or if the attempts to set up MySQL fail.
- **Resources > WebServer > Properties—**`UserData cfn-signal` on success

  If all the services are configured and started successfully, the UserData script calls cfn-signal with an exit code that signals success.
- **Resources > WaitHandle—**`AWS::CloudFormation::WaitConditionHandle`

  The `AWS::CloudFormation::WaitConditionHandle` key creates a pre-signed URL that is used as the signaling mechanism.
- **Resources > WaitCondition—**`AWS::CloudFormation::WaitCondition`

  The `AWS::CloudFormation::WaitCondition` key creates a special resource that waits for a specified period of time for a success signal. If the time period elapses without a success signal, the wait condition's status is set to CREATE_FAILED and the entire stack is rolled back.

The template now looks as follows. Additions to the template are shown in red italic text.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template LAMP_Single_Instance:
...",

  "Parameters" : { ... },

  "Mappings" : { ... },

  "Resources" : {

    "CfnUser" : { ... },
    "HostKeys" : { ... },

    "WebServer": {
      "Type": "AWS::EC2::Instance",
      "Metadata" : { ...
      },
      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" :
"AWS::Region" },
                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref"
 : "InstanceType" }, "Arch" ] } ] },
        "InstanceType"    : { "Ref" : "InstanceType" },
        "SecurityGroups" : [ {"Ref" : "WebServerSecurityGroup"} ],
        "KeyName"        : { "Ref" : "KeyName" },
        "UserData"       : { "Fn::Base64" : { "Fn::Join" : ["", [
          "#!/bin/bash -v\n",
          "yum update -y aws-cfn-bootstrap\n",

          "# Helper function\n",
          "function error_exit\n",
          "{\n",
          "  /opt/aws/bin/cfn-signal -e 1 -r \"$1\" '", { "Ref" : "WaitHandle"
 }, "'\n",
          "  exit 1\n",
          "}\n",

          "# Install LAMP packages\n",
          "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" }, " -r Web
Server ",
          "    --access-key ",  { "Ref" : "HostKeys" },
         "    --secret-key ", {"Fn::GetAtt": ["HostKeys", "SecretAccessKey"]},

          "    --region ", { "Ref" : "AWS::Region" },
          " || error_exit 'Failed to run cfn-init'\n",

          "# Setup MySQL, create a user and a database\n",
          "mysqladmin -u root password '", { "Ref" : "DBRootPassword" },
          "' || error_exit 'Failed to initialize root password'\n",
          "mysql -u root --password='", { "Ref" : "DBRootPassword" },
         "' < /tmp/setup.mysql || error_exit 'Failed to initialize database'\n",


          "# Configure the PHP application - in this case, fixup the page with
 the right references to the database\n",
          "sed -i \"s/REPLACE_WITH_DATABASE/localhost/g\" /var/www/html/in
dex.php\n",
          "sed -i \"s/REPLACE_WITH_DBUSER/", { "Ref" : "DBUsername" }, "/g\"
/var/www/html/index.php\n",
         "sed -i \"s/REPLACE_WITH_DBPASSWORD/", { "Ref" : "DBPassword" }, "/g\"
 /var/www/html/index.php\n",

          "# All is well so signal success\n",
          "/opt/aws/bin/cfn-signal -e 0 -r \"LAMP Stack setup complete\" '", {
 "Ref" : "WaitHandle" }, "'\n"
        ]]}}
      }
    },
    "WaitHandle" : {
      "Type" : "AWS::CloudFormation::WaitConditionHandle"
    },

    "WaitCondition" : {
      "Type" : "AWS::CloudFormation::WaitCondition",
      "DependsOn" : "WebServer",
      "Properties" : {
        "Handle" : {"Ref" : "WaitHandle"},
        "Timeout" : "300"
      }
    },

    "WebServerSecurityGroup" : { ...
    }
  },

  "Outputs" : {
    "WebsiteURL" : { ...
    }
  }
}
```

The full template is available at the following location:

https://s3.amazonaws.com/cloudformation-templates-us-east-1/LAMP_Single_Instance.template

Working with Microsoft Windows Stacks on AWS CloudFormation

AWS CloudFormation allows you to create Microsoft Windows stacks based on Amazon EC2 Windows Amazon Machine Images (AMIs) and provides you with the ability to install software, to use remote desktop to access your stack, and to update and configure your stack.

The topics in this section are designed to demonstrate how common tasks related to creation and management of Windows instances are accomplished with AWS CloudFormation.

In This Section

Microsoft Windows Amazon Machine Images (AMIs) and AWS CloudFormation Templates

With AWS CloudFormation, you can create Microsoft Windows stacks for running Windows server instances. A number of pre-configured templates are available to launch directly from the AWS CloudFormation Sample Templates page, such as the following templates:

- Windows_Single_Server_SharePoint_Foundation.template - SharePoint® Foundation 2010 running on Microsoft Windows Server® 2008 R2
- Windows_Single_Server_Active_Directory.template - Create a single server installation of Active Directory running on Microsoft Windows Server® 2008 R2.
- Windows_Roles_And_Features.template - Create a single server specifying server roles running on Microsoft Windows Server® 2008 R2.
- ElasticBeanstalk_Windows_Sample.template - Launch an AWS Elastic Beanstalk sample application on Windows Server 2008 R2 running IIS 7.5.

Note

Microsoft, Windows Server, and SharePoint are trademarks of the Microsoft group of companies.

Although these stacks are already configured, you can use any EC2 Windows AMI as the basis of an AWS CloudFormation Windows stack. For a complete list of Amazon EC2 AMIs available, go to the Amazon Machine Images (AMIs) page, and click **Windows** in the left sidebar under **Platforms**.

The main difference between the preconfigured AWS CloudFormation Windows templates and the Amazon EC2 Windows AMIs are that all of the AWS CloudFormation machine images come with AWS CloudFormation bootstrap tools already installed. For a base-level AMI with the bootstrap tools pre-installed and with the ability to run Windows Powershell commands directly from the User Data sections of your templates, see the Amazon EBS-Backed Windows Server 2008 R2 English 64-bit - Base for CloudFormation image.

Bootstrapping AWS CloudFormation Windows Stacks

The Amazon EBS-Backed Windows Server 2008 R2 English 64-bit - Base for CloudFormation AMI comes supplied with the AWS CloudFormation helper scripts pre-installed in the `C:\Program Files (x86)\Amazon\cfn-bootstrap` directory. This path is added to the system PATH environment variable for the instance, so you can call the scripts without specifying the full pathname.

Note

For complete information about the AWS CloudFormation helper scripts, see CloudFormation Helper Scripts Reference (p. 329).

If you will be creating your own Windows image for use with CloudFormation, see the information at Configuring a Windows Instance Using EC2ConfigService in the *Amazon EC2 Microsoft Windows Guide* for instructions. You must set up a Windows instance with EC2ConfigService for it to work with the AWS CloudFormation bootstrapping tools.

Example of Bootstrapping a Windows Stack

For the purposes of illustration, we'll examine the AWS CloudFormation single-instance Sharepoint server template, which can be viewed, in its entirety, at the following URL:

- https://s3.amazonaws.com/cloudformation-templates-us-east-1/Windows_Single_Server_SharePoint_Foundation.template

This example demonstrates how to:

- Create an IAM User and Security Group for access to the instance
- Configure initialization files: `cfn-credentials`, `cfn-hup.conf`, and `cfn-auto-reloader.conf`
- Download and install a package such as Sharepoint Foundation 2010 on the server instance.
- Use a WaitCondition to ensure resources are ready
- Retrieve an IP for the instance with Amazon Elastic IP (EIP).

The AWS CloudFormation helper script `cfn-init` is used to perform each of these actions, based on information in the AWS::CloudFormation::Init (p. 203) resource in the Windows Single Server Sharepoint Foundation template.

The AWS::CloudFormation::Init section is named "SharePointFoundation", and begins with a standard declaration:

```
"SharePointFoundation": {
   "Type" : "AWS::EC2::Instance",
   "Metadata" : {
     "AWS::CloudFormation::Init" : {
       "config" : {
```

After this, the **files** section of AWS::CloudFormation::Init is declared:

```
"files" : {
  "c:\\cfn\\cfn-credentials" : {
    "content" : { "Fn::Join" : ["", [
      "AWSAccessKeyId=", { "Ref" : "IAMUserAccessKey" }, "\n",
     "AWSSecretKey=", {"Fn::GetAtt": ["IAMUserAccessKey", "SecretAccessKey"]},
 "\n"
      ]]}
    },
  "c:\\cfn\\cfn-hup.conf" : {
    "content" : { "Fn::Join" : ["", [
      "[main]\n",
      "stack=", { "Ref" : "AWS::StackName" }, "\n",
      "credential-file=c:\\cfn\\cfn-credentials\n",
      "region=", { "Ref" : "AWS::Region" }, "\n"
      ]]}
  },
  "c:\\cfn\\hooks.d\\cfn-auto-reloader.conf" : {
    "content": { "Fn::Join" : ["", [
      "[cfn-auto-reloader-hook]\n",
      "triggers=post.update\n",
     "path=Resources.SharePointFoundation.Metadata.AWS::CloudFormation::Init\n",

      "action=cfn-init.exe -v -s ", { "Ref" : "AWS::StackName" },
                                     " -r SharePointFoundation",
                                     " --credential-file c:\\cfn\\cfn-creden
tials",
                                     " --region ", { "Ref" : "AWS::Region" },
"\n"
    ]]}
  },
  "C:\\SharePoint\\SharePointFoundation2010.exe" : {
    "source" : "http://d3adzpja92utk0.cloudfront.net/SharePointFoundation.exe"

  }
},
```

Three files are created here and placed in the `C:\cfn` directory on the server instance. They are:

- `cfn-credentials`, used to store the IAM User credentials.
- `cfn-hup.conf`, the configuration file for cfn-hup.
- `cfn-auto-reloader.conf`, the configuration file for the hook used by cfn-hup to initiate an update (calling cfn-init) when the metadata in AWS::CloudFormation::Init changes.

There is also a file that is downloaded to the server: `SharePointFoundation.exe`. This file is used to install SharePoint on the server instance.

Important

Since paths on Windows use a backslash ('\') character, you must always remember to properly escape all backslashes by prepending another backslash whenever you refer to a Windows path in the AWS CloudFormation template.

Next is the **commands** section, which looks like this:

```
"commands" : {
  "1-extract" : {
    "command" : "C:\\SharePoint\\SharePointFoundation2010.exe /extract:C:\\Share
Point\\SPF2010 /quiet /log:C:\\SharePoint\\SharePointFoundation2010-extract.log"

  },
  "2-prereq" : {
    "command" : "C:\\SharePoint\\SPF2010\\PrerequisiteInstaller.exe /unattended"

  },
  "3-install" : {
     "command" : "C:\\SharePoint\\SPF2010\\setup.exe /config C:\\Share
Point\\SPF2010\\Files\\SetupSilent\\config.xml"
  }
```

Because commands in the instance are processed in *alphabetical order by name*, each command has been prepended with a number indicating its desired execution order. Thus, we can make sure that the installation package is first extracted, all prerequisites are then installed, and finally, installation of SharePoint is started.

Next is the **Properties** section:

```
"Properties": {
  "InstanceType" : { "Ref" : "InstanceType" },
  "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region"
 },
               { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "Instan
ceType" }, "Arch" ] } ] },
  "SecurityGroups" : [ {"Ref" : "SharePointFoundationSecurityGroup"} ],
  "KeyName" : { "Ref" : "KeyPairName" },
  "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
    "<script>\n",

    "cfn-init.exe -v -s ", { "Ref" : "AWS::StackName" },
    " -r SharePointFoundation",
    " --access-key ", { "Ref" : "IAMUserAccessKey" },
    " --secret-key ", {"Fn::GetAtt": ["IAMUserAccessKey", "SecretAccessKey"]},

    " --region ", { "Ref" : "AWS::Region" }, "\n",

    "cfn-signal.exe -e %ERRORLEVEL% ", { "Fn::Base64" : { "Ref" : "SharePoint
FoundationWaitHandle" }}, "\n",

    "SCHTASKS /Create /SC MINUTE /MO 10 /TN cfn-hup /RU SYSTEM /TR \"cfn-hup.exe
 -v --no-daemon\"", "\n",

    "</script>"
    ]]}}
  }
```

Note that in this section, the UserData property contains a `cmd.exe` script that will be executed by `cfn-init`, surrounded by <script> tags. You can use a Windows Powershell script here instead by surrounding your script with <powershell> tags.

SharePointFoundationWaitHandle is referenced here and run with `cfn-signal`. The **WaitConditionHandle** and associated **WaitCondition** are declared next in the template:

```
"SharePointFoundationWaitHandle" : {
   "Type" : "AWS::CloudFormation::WaitConditionHandle"
},

"SharePointFoundationWaitCondition" : {
   "Type" : "AWS::CloudFormation::WaitCondition",
   "DependsOn" : "SharePointFoundation",
   "Properties" : {
     "Handle" : {"Ref" : "SharePointFoundationWaitHandle"},
     "Timeout" : "3600"
   }
}
```

Since executing all of the steps and installing SharePoint might take a while, but not an entire hour, the WaitCondition waits an hour (3600 seconds) before timing out.

If all goes well, an Elastic IP is used to provide access to the SharePoint instance:

```
"Outputs" : {
 "SharePointFoundationURL" : {
   "Value" : { "Fn::Join" : ["", ["http://", { "Ref" : "SharePointFoundationEIP"
 } ]] },
   "Description" : "SharePoint Team Site URL. Please retrieve Administrator
password of the instance and use it to access the URL"
 }
```

Once stack creation is complete, the IP address supplied by EIP will be displayed in the **Outputs** tab of the AWS CloudFormation console. However, before you can access the instance you will need to retreive the auto-generated temporary Administrator password for the instance. Instructions about how to do this are provided in the Accessing AWS CloudFormation Windows Instances (p. 184) topic.

How to Troubleshoot Stack Creation Issues

If your stack fails during creation, the default behavior is to Rollback on failure. While this is normally a good default because it avoids unnecessary charges, it makes it difficult to debug why your stack creation is failing.

To turn this behavior off, click **Show Advanced Options** when creating your stack with the AWS CloudFormation console, and click the **No** selector next to **Rollback on failure**. This will allow you to log into your instance and view the logfiles to pinpoint issues encountered when running your startup scripts.

Important logs to look at are:

- The EC2 configuration log at `C:\Program Files\Amazon\Ec2ConfigService\Logs\Ec2ConfigLog.txt`
- The **cfn-init** log at `C:\cfn\log\cfn-init.log`

Accessing AWS CloudFormation Windows Instances

Once you've successfully created a Microsoft Windows stack on AWS CloudFormation, you can log in to your instance with Remote Desktop to configure it manually. There are a number of steps involved:

1. Find the physical id of your Windows instance.
2. Use the physical id to retrieve the login credentials from Amazon EC2.
3. Use the login credentials to access your instance with Remote Desktop.

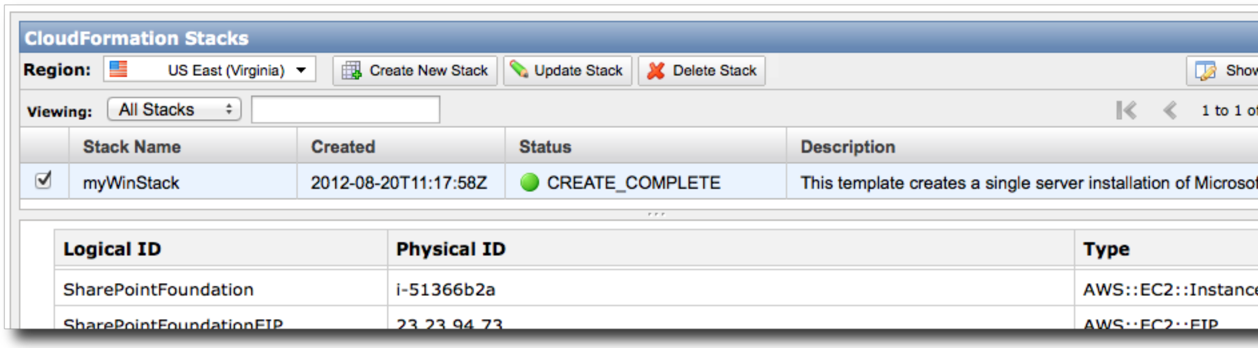
Note

Before starting, you'll need to have an AWS CloudFormation Windows stack running, and you'll also need the private key of the key pair you used when creating the instance. For information about generating Amazon EC2 key pairs, see Creating an EC2 Key Pair (p. 80).

To retrieve the physical ID of your AWS CloudFormation Windows instance:

1. From the AWS CloudFormation console, click on your Windows-based stack. You will see your stack information appear in the lower pane of the window.
2. Click the **Resources** tab, and find the **Physical ID** of the AWS::EC2::Instance (p. 222). It will look something like this: `i-51366b2a`.

   If you have many instances running, you will probably want to remember the physical ID of your instance, or write it down. You'll need it to recover the Administrator password to log in to your instance.



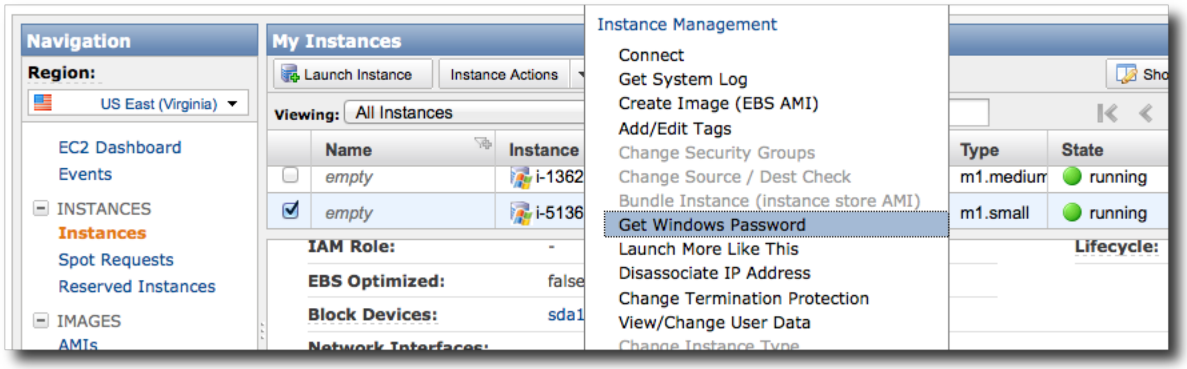
	Stack Name	Created	Status	Description
✓	myWinStack	2012-08-20T11:17:58Z	● CREATE_COMPLETE	This template creates a single server installation of Microsof

Logical ID	Physical ID	Type
SharePointFoundation	i-51366b2a	AWS::EC2::Instance
SharePointFoundationEIP	23.23.94.73	AWS::EC2::EIP

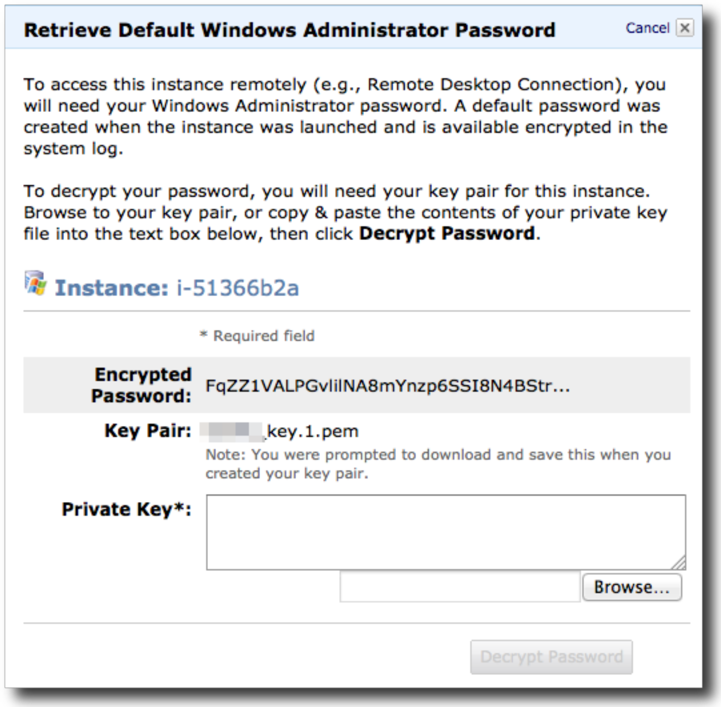
Once you have the physical ID of your instance, you can use this to retrieve the Administrator password.

To retrieve the Administrator password for your Windows instance:

1. At the top left corner of the AWS CloudFormation console, click **Services** and then **EC2**. This will bring you to the **Amazon EC2 Console Dashboard**.
2. On the **Navigation Bar**, click **Instances**. This will bring up a list titled **My Instances**.
3. In the list, find your instance by its physical ID. Once you find it, right-click its entry on the list. This will display the **Instance Management** context menu.



4. On the context menu, click **Get Windows Password**. A dialog will appear, called **Retrieve Default Windows Administrator Password**. On this dialog, an encrypted password will be shown, as well as the Amazon EC2 key pair that you used when creating the AWS CloudFormation Windows stack.


Retrieve Default Windows Administrator Password Cancel

To access this instance remotely (e.g., Remote Desktop Connection), you will need your Windows Administrator password. A default password was created when the instance was launched and is available encrypted in the system log.

To decrypt your password, you will need your key pair for this instance. Browse to your key pair, or copy & paste the contents of your private key file into the text box below, then click **Decrypt Password**.

Instance: i-51366b2a

* Required field

Encrypted Password:	FqZZ1VALPGvlilNA8mYnzp6SSI8N4BStr...
Key Pair:	[redacted]key.1.pem Note: You were prompted to download and save this when you created your key pair.
Private Key*:	

Browse...

Decrypt Password

5. Do *one* of the following (they are equivalent):

   • Locate the private key file you downloaded that corresponds to the key pair shown, copy its contents to the clipboard, and then paste it into the **Private Key** box on the dialog.

- Click the **Browse** button to browse for the private key file on your system. When you select it, the contents of the file will appear in the **Private Key** box.

6. Click **Decrypt Password**. The connection information for your instance will be shown, consisting of:
   - the IP address of your remote instance.
   - The username to use when logging in.
   - The decrypted password.

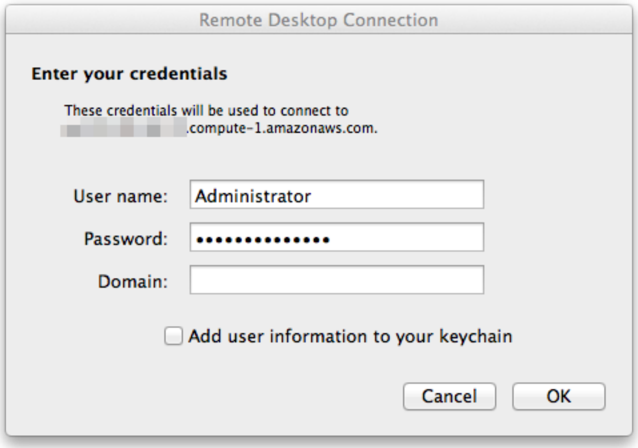
Note

This password is meant to be temporary. Once you log in to your instance, you should change it to one of your own choice.

These user credentials can be used to log in to your Windows instance with Remote Desktop.

To log in to your AWS CloudFormation Windows stack:

1. Start your Remote Desktop client.
2. When prompted for the **Server**, enter the server name that you retrieved for your instance from EC2.



Remote Desktop Connection

Enter your credentials

These credentials will be used to connect to [redacted].compute-1.amazonaws.com.

User name: Administrator

Password: ••••••••••••••

Domain:

☐ Add user information to your keychain

3. Enter the **User name** ("Administrator") and the **Password** that you retrieved from EC2.
4. If you are prompted for a **Domain**, leave the field blank.
5. Click **OK** to finish connecting.

Once you're logged in to your server, you can configure it how you like. You can also use this credential information to log in to any secure outputs that your stack created, such as a Sharepoint site. It's your Windows instance, do what you want with it!

Template Reference

This section details the supported resources, type names, intrinsic functions and pseudo parameters used in AWS CloudFormation templates.

Topics

AWS Resource Types Reference

The following table lists the AWS resources and API versions that AWS CloudFormation supports.

As we add support for more resources, resource type identifiers will always take the following form:

```
AWS::aws-product-name::data-type-name
```

AWS Resource	AWS CloudFormation Resource Type Identifier	Supported API Version
Auto Scaling AutoScalingGroup	AWS::AutoScaling::AutoScalingGroup (p. 190)	2009-05-15
Auto Scaling LaunchConfiguration	AWS::AutoScaling::LaunchConfiguration (p. 193)	2009-05-15
Auto Scaling Policy	AWS::AutoScaling::ScalingPolicy (p. 197)	2009-05-15
Auto Scaling Trigger	AWS::AutoScaling::Trigger (p. 198)	2009-05-15
Amazon CloudFormation Authentication	AWS::CloudFormation::Authentication (p. 200)	2010-05-15

AWS Resource	AWS CloudFormation Resource Type Identifier	Supported API Version
Amazon CloudFormation Init	AWS::CloudFormation::Init (p. 203)	2010-05-15
Amazon CloudFormation Stack	AWS::CloudFormation::Stack (p. 211)	2010-05-15
Amazon CloudFormation WaitCondition	AWS::CloudFormation::WaitCondition (p. 212)	2010-05-15
Amazon CloudFormation WaitConditionHandle	AWS::CloudFormation::WaitConditionHandle (p. 213)	2010-05-15
Amazon CloudFront Distribution	AWS::CloudFront::Distribution (p. 213)	2010-11-01
Amazon CloudWatch	AWS::CloudWatch::Alarm (p. 214)	2010-05-15
Amazon DynamoDB Table	AWS::DynamoDB::Table (p. 216)	2010-05-15
Amazon EBS Volume	AWS::EC2::Volume (p. 247)	2009-11-30
Amazon EBS Volume Attachment	AWS::EC2::VolumeAttachment (p. 247)	2009-11-30
Amazon EC2 Customer Gateway	AWS::EC2::CustomerGateway (p. 217)	2011-12-15
Amazon EC2 DHCP Options	AWS::EC2::DHCPOptions (p. 218)	2011-12-15
Amazon EC2 Elastic IP Address	AWS::EC2::EIP (p. 220)	2009-11-30
Amazon EC2 Elastic IP Address Association	AWS::EC2::EIPAssociation (p. 221)	2009-11-30
Amazon EC2 Instance	AWS::EC2::Instance (p. 222)	2009-11-30
Amazon EC2 Internet Gateway	AWS::EC2::InternetGateway (p. 227)	2011-12-15
Amazon EC2 Network ACL	AWS::EC2::NetworkAcl (p. 228)	2011-12-15
Amazon EC2 Network ACL Entry	AWS::EC2::NetworkAclEntry (p. 229)	2011-12-15
Amazon EC2 Route	AWS::EC2::Route (p. 233)	2011-12-15
Amazon EC2 Route Table	AWS::EC2::RouteTable (p. 236)	2011-12-15
Amazon EC2 Security Group	AWS::EC2::SecurityGroup (p. 237)	2009-11-30
Amazon EC2 Security Group Egress	AWS::EC2::SecurityGroupEgress (p. 242)	2011-12-15

AWS Resource	AWS CloudFormation Resource Type Identifier	Supported API Version
Amazon EC2 Security Group Ingress	AWS::EC2::SecurityGroupIngress (p. 238)	2009-11-30
Amazon EC2 Subnet	AWS::EC2::Subnet (p. 243)	2011-12-15
Amazon EC2 Subnet Network ACL Association	AWS::EC2::SubnetNetworkAclAssociation (p. 245)	2011-12-15
Amazon EC2 Subnet Route Table Association	AWS::EC2::SubnetRouteTableAssociation (p. 246)	2011-12-15
Amazon EC2 VPC	AWS::EC2::VPC (p. 248)	2011-12-15
Amazon EC2 VPC Dhcp Options Association	AWS::EC2::VPCDHCPOptionsAssociation (p. 249)	2011-12-15
Amazon EC2 VPC Gateway Attachment	AWS::EC2::VPCGatewayAttachment (p. 250)	2011-12-15
Amazon EC2 VPN Connection	AWS::EC2::VPNConnection (p. 251)	2011-12-15
Amazon EC2 VPN Gateway	AWS::EC2::VPNGateway (p. 252)	2011-12-15
Amazon ElastiCache Cache Cluster	AWS::ElastiCache::CacheCluster (p. 254)	2011-07-15
Amazon ElastiCache Parameter Group	AWS::ElastiCache::ParameterGroup (p. 256)	2011-07-15
Amazon ElastiCache Security Group	AWS::ElastiCache::SecurityGroup (p. 256)	2011-07-15
Amazon ElastiCache Security Group Ingress	AWS::ElastiCache::SecurityGroupIngress (p. 257)	2011-07-15
AWS Elastic Beanstalk Application	AWS::ElasticBeanstalk::Application (p. 257)	2010-12-01
AWS Elastic Beanstalk Environment	AWS::ElasticBeanstalk::Environment (p. 257)	2010-12-01
Elastic Load Balancing LoadBalancer	AWS::ElasticLoadBalancing::LoadBalancer (p. 258)	2009-05-15
IAM Access Key	AWS::IAM::AccessKey (p. 262)	2010-05-08
IAM Group	AWS::IAM::Group (p. 263)	2010-05-08
IAM Instance Profile	AWS::IAM::InstanceProfile (p. 264)	2010-05-08
IAM Policy	AWS::IAM::Policy (p. 266)	2010-05-08
IAM Role	AWS::IAM::Role (p. 268)	2010-05-08
IAM Add User to Group	AWS::IAM::UserToGroupAddition (p. 273)	2010-05-08
IAM User	AWS::IAM::User (p. 271)	2010-05-08

AWS Resource	AWS CloudFormation Resource Type Identifier	Supported API Version
Amazon RDS DBInstance	AWS::RDS::DBInstance (p. 274)	2009-10-16
Amazon RDS DBSecurityGroup	AWS::RDS::DBSecurityGroup (p. 279)	2009-10-16
Amazon RDS Security Group Ingress	AWS::RDS::SecurityGroupIngress (p. 242)	2012-01-15
Amazon RDS DB Subnet Group	AWS::RDS::DBSubnetGroup (p. 278)	2012-01-15
Amazon Route 53 Resource Record Set	AWS::Route53::RecordSet (p. 282)	2010-10-01
Amazon Route 53 Resource Record Set Group	AWS::Route53::RecordSetGroup (p. 285)	2010-10-01
Amazon S3 Bucket	AWS::S3::Bucket (p. 286)	2006-03-01
Amazon S3 Bucket Policy	AWS::S3::BucketPolicy (p. 287)	2006-03-01
Amazon SimpleDB Domain	AWS::SDB::Domain (p. 287)	2009-04-15
SNS Topic Policy	AWS::SNS::TopicPolicy (p. 288)	2009-04-15
Amazon SNS Subscription	AWS::SNS::Subscription (p. 321)	2010-03-31
Amazon SNS Topic	AWS::SNS::Topic (p. 288)	2010-03-31
Amazon SQS Queue Policy	AWS::SQS::QueuePolicy (p. 292)	2009-02-01
Amazon SQS Queue	AWS::SQS::Queue (p. 288)	2009-02-01

AWS::AutoScaling::AutoScalingGroup

The AWS::AutoScaling::AutoScalingGroup type creates an Auto Scaling group.

This type supports updates. For more information about updating this resource, see UpdateAutoScalingGroup. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Caution

When you update the *`LoadBalancerNames`* property, AWS CloudFormation first creates a replacement auto scaling group resource with a new physical name, changes references from other dependent resources to point to the replacement resource, and then deletes the old resource.

Syntax

```
{
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "AvailabilityZones (p. 191)" : [ AvailabilityZone1, ... ],
      "Cooldown (p. 191)" : String,
      "DesiredCapacity (p. 191)" : String,
      "HealthCheckGracePeriod (p. 191)" : Integer,
      "HealthCheckType (p. 191)" : String,
      "LaunchConfigurationName (p. 191)" : String,
      "LoadBalancerNames (p. 192) : [ String1, ... ]
      "MaxSize (p. 192)" : String,
      "MinSize (p. 192)" : String,
      "NotificationConfiguration (p. 192)" : String,
      "Tags (p. 192)" : [ Tag1, ..., ]
      "VPCZoneIdentifier (p. 192)" : String
   }
}
```

Properties

AvailabilityZones

Contains a list of Availability Zones for the group.

Required: Yes

Type: list of Strings

Cooldown

The number of seconds after a scaling activity completes before any further scaling activities can start.

Required: No

Type: String

DesiredCapacity

Specifies the desired capacity for the auto scaling group.

Required: No

Type: String

Note

If the `SpotPrice` property is set in the launch config for this autoscaling group, then `DesiredCapacity` is ignored.

HealthCheckGracePeriod

Length of time in seconds after a new EC2 instance comes into service that Auto Scaling starts checking its health.

Required: No

Type: Integer

HealthCheckType

The service you want the health status from, Amazon EC2 or Elastic Load Balancer. Valid values are "EC2" or "ELB."

Required: No

Type: String

LaunchConfigurationName

Specifies the name of the associated AWS::AutoScaling::LaunchConfiguration (p. 193).

Required: Yes

Type: String

Update requires: replacement (p. 34)

LoadBalancerNames

A list of load balancers associated with this auto scaling group.

Required: Yes

Type: list of Strings

Update requires: replacement (p. 34)

MaxSize

The maximum size of the auto scaling group.

Required: Yes

Type: String

MinSize

The minimum size of the auto scaling group.

Required: Yes

Type: String

NotificationConfiguration

An embedded property that configures an autoscaling group to send notifications when specified events take place.

Required: No

Type: NotificationConfiguration (p. 294) type

Tags

The tags you want to attach to this resource.

For more information about tags, go to Tagging Auto Scaling Groups and Amazon EC2 Instances in the *Auto Scaling Developer Guide*.

Required: Yes

Type: list of Auto Scaling Tags (p. 294)

Update requires: no interruption (p. 33)

VPCZoneIdentifier

A comma-separated list of subnet identifiers of Amazon Virtual Private Clouds (Amazon VPCs).

The subnets that you specify for *`VPCZoneIdentifier`* must reside in the availability zones that you specify with the *`AvailabilityZones`* parameter.

For more information, go to Using EC2 Dedicated Instances Within Your VPC in the *Auto Scaling Developer Guide*.

Update requires: replacement (p. 34)

Required: No

Type: List of Strings

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyASGroup" }
```

For an autoscaling group with the logical ID "MyASGroup", `Ref` will return:

arn:aws:cloudformation:us-east-1:803981987763:stack /mystack-myembeddedstack-sggfrhxhum7w/
f449b250-b969-11e0-a185-5081d0136786

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

To view AutoScalingGroup snippets, see Auto Scaling Group Resource (p. 110).

AWS::AutoScaling::LaunchConfiguration

The AWS::AutoScaling::LaunchConfiguration type creates an Auto Scaling launch configuration that can be used by an Auto Scaling group to configure EC2 instances in the Auto Scaling group.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Important

When you update a LaunchConfiguration resource, AWS CloudFormation will delete that resource and create a new one with the updated properties and a new name. This update action does not deploy any change across the running EC2 instances in the auto scaling group. In other words, an update simply replaces the LaunchConfiguration so that when the auto scaling group launches new instances, they will get the updated configuration but existing instances will continue to run with the configuration that they were originally launched with. This works the same way as if you made similar changes manually to an auto scaling group.

Syntax

```
{
   "Type" : "AWS::AutoScaling::LaunchConfiguration",
   "Properties" : {
      "BlockDeviceMappings (p. 193)" : [ BlockDeviceMapping1, ... ],
      "IamInstanceProfile (p. 194)" : String,
      "ImageId (p. 194)" : String,
      "InstanceMonitoring (p. 194)" : Boolean,
      "InstanceType (p. 194)" : String,
      "KernelId (p. 194)" : String,
      "KeyName (p. 194)" : String,
      "RamDiskId (p. 194) : String,
      "SecurityGroups (p. 194)" : [ SecurityGroup1, ... ],
      "SpotPrice (p. 194)" : String,
      "UserData (p. 195)" : String
   }
}
```

Properties

BlockDeviceMappings

Specifies how block devices are exposed to the instance. You can specify virtual devices and EBS volumes.

Required: No

Type: list of BlockDeviceMapping (p. 293)s.

IamInstanceProfile

Provides the name or the Amazon Resource Name (ARN) of the instance profile associated with the IAM role for the instance. The instance profile contains the IAM role.

Required: No

Type: String (1–1600 chars)

ImageId

Provides the unique ID of the Amazon Machine Image (AMI) that was assigned during registration.

Required: Yes

Type: String

InstanceMonitoring

Indicates whether or not instance monitoring should be enabled for this autoscaling group. This is enabled by default. To turn it off, set `InstanceMonitoring` to "false".

Required: No. Default value is "true".

Type: Boolean

InstanceType

Specifies the instance type of the EC2 instance.

Required: Yes

Type: String

KernelId

Provides the ID of the kernel associated with the EC2 AMI.

Required: No

Type: String

KeyName

Provides the name of the EC2 key pair.

Required: No

Type: String

RamDiskId

The ID of the RAM disk to select. Some kernels require additional drivers at launch. Check the kernel requirements for information on whether you need to specify a RAM disk. To find kernel requirements, refer to the AWS Resource Center and search for the kernel ID.

Required: No

Type: String

SecurityGroups

A list containing the EC2 security groups to assign to the Amazon EC2 instances in the Auto Scaling group. The list can contain the name of existing EC2 security groups, references to AWS::EC2::SecurityGroup resources created in the template, or both.

Required: No

Type: list of EC2 security groups

SpotPrice

The spot price for this autoscaling group. If a spot price is set, then the autoscaling group will launch when the current spot price is less than the amount specified in the template.

When you have specified a spot price for an autoscaling group, the group will launch when the spot price has been met, regardless of the setting in the autoscaling group's *`DesiredCapacity`*.

For more information about configuring a spot price for an autoscaling group, see Using Auto Scaling to Launch Spot Instances in the *AutoScaling Developer Guide*.

Required: No

Type: String

Update requires: replacement (p. 34)

Note

When you change your bid price by creating a new launch configuration, running instances will continue to run as long as the bid price for those running instances is higher than the current Spot price.

UserData

The user data available to the launched EC2 instances.

Required: No

Type: String

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyAutoScalingGroup" }
```

For the resource with the logical ID "MyAutoScalingGroup", `Ref` will return the AutoScaling launch config name, such as: `mystack-mylaunchconfig-1DDYF1E3B3I`.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example LaunchConfig with Spot Price in Autoscaling Group

This example shows a launch configuration that features a spot price in the AutoScaling group. This launch configuration will only be active if the current spot price is less than the amount in the template specification (0.05).

```
"LaunchConfig" : {
   "Type" : "AWS::AutoScaling::LaunchConfiguration",
   "Properties" : {
      "KeyName" : { "Ref" : "KeyName" },
      "ImageId" : {
         "Fn::FindInMap" : [
            "AWSRegionArch2AMI",
            { "Ref" : "AWS::Region" },
            {
               "Fn::FindInMap" : [
                  "AWSInstanceType2Arch", { "Ref" : "InstanceType" }, "Arch"
               ]
            }
         ]
      },
      "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
      "SpotPrice" :  "0.05",
      "InstanceType" : { "Ref" : "InstanceType" }
   }
}
```

Example LaunchConfig with IAM Instance Profile

Here's a launch configuration using the IamInstanceProfile (p. 194) property.

Only the AWS::AutoScaling::LaunchConfiguration specification is shown. For the full template, including the definition of, and further references from, the AWS::IAM::InstanceProfile (p. 264) object referenced here as "RootInstanceProfile" see: auto_scaling_with_instance_profile.template.

```
"myLCOne": {
   "Type": "AWS::AutoScaling::LaunchConfiguration",
   "Properties": {
      "ImageId": {
         "Fn::FindInMap": [
            "AWSRegionArch2AMI",
            { "Ref": "AWS::Region" },
            {
               "Fn::FindInMap": [
                  "AWSInstanceType2Arch", { "Ref": "InstanceType" }, "Arch"
               ]
            }
         ]
      },
      "InstanceType": { "Ref": "InstanceType" },
      "IamInstanceProfile": { "Ref": "RootInstanceProfile" }
   }
}
```

To view more LaunchConfiguration snippets, see Auto Scaling Launch Configuration Resource (p. 109).

AWS::AutoScaling::ScalingPolicy

The AWS::AutoScaling::ScalingPolicy resource adds a scaling policy to an auto scaling group. A scaling policy specifies whether to scale the auto scaling group up or down, and by how much. For more information on scaling policies, see Scaling by Policy in the Auto Scaling Developer Guide.

You can use a scaling policy together with an Amazon CloudWatch alarm. An Amazon CloudWatch alarm can automatically initiate actions on your behalf, based on parameters you specify. A scaling policy is one type of action that an alarm can initiate. For a snippet showing how to create an Auto Scaling policy that is triggered by an Amazon CloudWatch alarm, see Auto Scaling Policy Triggered by Amazon CloudWatch Alarm (p. 110).

This type supports updates. For more information about updating this resource, see PutScalingPolicy. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type" : "AWS::AutoScaling::ScalingPolicy",
   "Properties" : {
      "AdjustmentType (p. ?)" : String,
      "AutoScalingGroupName (p. ?)" : String,
      "Cooldown (p. ?)" : String,
      "ScalingAdjustment (p. ?)" : String,
   }
}
```

Properties

AdjustmentType

Specifies whether the *`ScalingAdjustment`* is an absolute number or a percentage of the current capacity. Valid values are *`ChangeInCapacity`*, *`ExactCapacity`*, and *`PercentChangeInCapacity`*.

Required: Yes

Type: String

AutoScalingGroupName

The name or Amazon Resource Name (ARN) of the Auto Scaling Group that you want to attach the policy to.

Required: Yes

Type: String

Cooldown

The amount of time, in seconds, after a scaling activity completes before any further trigger-related scaling activities can start.

Required: No

Type: String

ScalingAdjustment

The number of instances by which to scale. AdjustmentType determines the interpretation of this number (e.g., as an absolute number or as a percentage of the existing Auto Scaling group size). A positive increment adds to the current capacity and a negative value removes from the current capacity.

Required: Yes

Type: String

Return Value

When you specify an AWS::AutoScaling::ScalingPolicy type as an argument to the `Ref` function, AWS CloudFormation returns the policy name.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::AutoScaling::Trigger

Syntax

```
{
   "Type" : "AWS::AutoScaling::Trigger",
   "Properties" : {
      "Dimensions (p. 199)" : [ CloudWatch Dimension1, ... ],
      "MetricName (p. 199)" : String,
      "Namespace (p. 199)" : String,
      "Period (p. 199)" : String,
      "Statistic (p. 199)" : String,
      "Unit (p. 199)" : String,
      "UpperBreachScaleIncrement (p. 199)" : String,
      "LowerBreachScaleIncrement (p. 199)" : String,
      "AutoScalingGroupName (p. 199)" : String,
      "BreachDuration (p. 200)" : String,
      "UpperThreshold (p. 200)" : String,
      "LowerThreshold (p. 200)" : String,
   }
}
```

Properties

Dimensions
: The dimensions used to retrieve metric statistics that the trigger uses to determine when to fire.

  Required: Yes

  Type: List of CloudWatch Dimensions

MetricName
: The metric name used by the trigger to determine when to fire.

  Required: Yes

  Type: String

Namespace
: The namespace used by the trigger to determine when to fire.

  Required: Yes

  Type: String

Period
: The period used in retrieving metric statistics used by the trigger to determine when to fire.

  Required: Yes

  Type: String

Statistic
: The statistic used by the trigger to determine which metric statistics to examine.

  Required: Yes

  Type: String

Unit
: The standard unit associated with a measure, used by the trigger when fetching the metric statistics it uses to determine when to activate.

  Required: No

  Type: String

UpperBreachScaleIncrement
: The incremental amount to use when performing scaling activities when the upper threshold has been breached.

  Must be a negative or positive integer, or integer percentage value.

  Required: Conditional

  Type: String

LowerBreachScaleIncrement
: The incremental amount to use when performing scaling activities when the lower threshold has been breached.

  Must be a negative or positive integer, or integer percentage value.

  Required: Conditional

  Type: String

AutoScalingGroupName
: Name of the auto scaling group.

  Required: Yes

  Type: String

BreachDuration
The amount of time to wait while the trigger is firing before performing scaling activities in response to the breach.

Required: Yes

Type: String

UpperThreshold
The upper limit of the metric used. The trigger fires if all data points in the last BreachDuration seconds exceed the upper threshold or fall below the lower threshold.

Required: Yes

Type: String

LowerThreshold
The lower limit for the metric. The trigger fires if all data points in the last BreachDuration seconds exceed the upper threshold or fall below the lower threshold.

Required: Yes

Type: String

Return Value

When you specify an AWS::AutoScaling::Trigger type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`TriggerName`*.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::CloudFormation::Authentication

Use the AWS::CloudFormation::Authentication type to specify authentication credentials for files or sources that you specify with the AWS::CloudFormation::Init (p. 203) type.

To include authentication information for a file or source that you specify with AWS::CloudFormation::Init, use the `uris` property if the source is a URI or the buckets property if the source is an Amazon S3 bucket. For more information about files, see Files (p. 206). For more information about sources, see Sources (p. 210).

You can also specify authentication information for files directly in the AWS::CloudFormation::Init resource. The files key of the resource contains a property named `authentication`. You can use the `authentication` property to associate authentication information defined in an AWS::CloudFormation::Authentication resource directly with a file.

For files, AWS CloudFormation looks for authentication information in the following order:

1. The `authentication` property of the AWS::CloudFormation::Init `files` key.
2. The `uris` or `buckets` property of the AWS::CloudFormation::Authentication resource.

For sources, AWS CloudFormation looks for authentication information in the `uris` or `buckets` property of the AWS::CloudFormation::Authentication resource.

Key	Description
type	Required. Specifies whether the authentication scheme uses a username and password (basic) or an access key ID and secret key (S3). • If you specify `basic`, you must also specify the `username`, `password`, and `uris` properties. • If you specify `S3`, you must also specify the `accessKeyId`, `secretKey`, and `buckets` properties. Valid values: `basic | S3`
username	Conditional. Specifies the username for basic authentication. Condition: Can be specified only if the type property is `basic`.
password	Conditional. Specifies the password for basic authentication. Condition: Can be specified only if the type property is `basic`.
uris	Conditional. A comma-delimited list of URIs to be associated with the basic authentication credentials. The authorization applies to the specified URIs and any more specific URI. For example, if you specify `http://www.example.com`, the authorization will also apply to `http://www.example.com/test`. Condition: Can be specified only if the type property is `basic`.
accessKeyId	Conditional. Specifies the access key ID for S3 authentication. Condition: Can be specified only if the type property is `S3`.
secretKey	Conditional. Specifies the secret key for S3 authentication. Condition: Can be specified only if the type property is `S3`.
buckets	Conditional. A comma-delimited list of Amazon S3 buckets to be associated with the S3 authentication credentials. Condition: Can be specified only if the type property is `S3`.

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Examples

The following example template snippet includes both types of authentication.

```
"AWS::CloudFormation::Authentication" : {
    "testBasic" : {
                    "type" : "basic",
                    "username" : "myuser",
                    "password" : "mypassword",
                    "uris" : ["http://www.example.com/test"]
    },
    "testS3" : {
                    "type" : "S3",
                    "accessKeyId" : "<Your Access Key ID>",
                    "secretKey" : "<Your Secret Key>",
                    "buckets" : ["myawsbucket"]
    }
}
```

The following example template snippet defines an authentication resource that references an Amazon S3 bucket defined elsewhere in the template.

```
        "AWS::CloudFormation::Authentication" : {
          "S3AccessCreds" : {
            "type" : "S3",
            "accessKeyId" : { "Ref" : "CfnKeys" },
            "secretKey" : {"Fn::GetAtt": ["CfnKeys", "SecretAccessKey"]},
            "buckets" : [ { "Ref" : "BucketName" } ]
          }
        }
```

In the following example template snippet, the AWS::CloudFormation::Init resource files key uses the authentication property to reference the AWS::CloudFormation::Authentication resource.

```
            "files" : {
              "/var/www/html/index.html" : {
                "source" : { "Fn::Join" : ["", ["http://s3.amazonaws.com/",
                           { "Ref" : "BucketName" }, "/index.html"]]},
                "mode"   : "000400",
                "owner"  : "apache",
                "group"  : "apache",
                "authentication" : "S3AccessCreds"
              }
            },

            :
          }
        },


        "AWS::CloudFormation::Authentication" : {
          "S3AccessCreds" : {
            "type" : "S3",
            "accessKeyId" : { "Ref" : "CfnKeys" },
            "secretKey" : {"Fn::GetAtt": ["CfnKeys", "SecretAccessKey"]}
          }
        }
```

AWS::CloudFormation::Init

Topics

Use the AWS::CloudFormation::Init type to include metadata on an Amazon EC2 instance for the cfn-init helper script. If your template calls the cfn-init script, the script looks for resource metadata rooted in the AWS::CloudFormation::Init metadata key. For more information about cfn-init, see cfn-init (p. 330).

The metadata is organized into config keys, which you can group into configsets. You can specify a configset when you call cfn-init in your template. If you don't specify a configset, cfn-init looks for a single config key named *config*.

The configuration is separated into sections. The following template snippet shows how you can attach metadata for cfn-init to an Amazon EC2 instance resource within the template:

```
"Resources": {
  "MyInstance": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
      "AWS::CloudFormation::Init" : {
        "config" : {
          "packages" : {
            :
          },
          "sources" : {
            :
          },
          "commands" : {
            :
          },
          "files" : {
            :
          },
          "services" : {
            :
          },
          "users" : {
            :
          },
          "groups" : {
            :
          }
        }
      }
    },
    "Properties": {
      :
    }
  }
}
```

For an example of using AWS::CloudFormation::Init and the cfn-init helper script, see Deploying Applications with AWS CloudFormation (p. 167).

Configsets

If you want to create more than one config key and to have cfn-init process them in a specific order, create a configset that contains the config keys in the desired order. For example, the following template snippet creates configsets named `ascending` and `descending` that each contain two config keys.

```
"AWS::CloudFormation::Init" : {
    "configSets" : {
        "ascending" : [ "config1" , "config2" ],
        "descending" : [ "config2" , "config1" ]
    },
    "config1" : {
        "commands" : {
            "test" : {
                "command" : "echo \"$CFNTEST\" > test.txt",
                "env" : { "CFNTEST" : "I come from config1." },
                "cwd" : "~"
            }
        }
    },
    "config2" : {
        "commands" : {
            "test" : {
                "command" : "echo \"$CFNTEST\" > test.txt",
                "env" : { "CFNTEST" : "I come from config2" },
                "cwd" : "~"
            }
        }
    }
}
```

The following example calls to cfn-init refer to the preceding example configsets. The example calls are abbreviated for clarity, see cfn-init (p. 330) for the complete syntax.

- If a call to cfn-init specifies the `ascending` configset:

  ```
  cfn-init -c ascending
  ```

  the script processes `config1` and then processes `config2` and the test.txt file would contain the text `I come from config2`.
- If a call to cfn-init specifies the `descending` configset:

  ```
  cfn-init -c descending
  ```

the script processes `config2` and then processes `config1` and the test.txt file would contain the text `I come from config1`.

You can create multiple configsets, and call a series of them using your cfn-init script. Each configset can contain a list of config keys or references to other configsets. For example, the following template snippet creates three configsets. The first configset, `test1`, contains one config key named `1`. The second configset, `test2`, contains a reference to the `test1` configset and one config key named `2`. The third configset, default, contains a reference to the configset `test2`.

```
"AWS::CloudFormation::Init" : {
    "configSets" : {
        "test1" : [ "1" ],
        "test2" : [ { "ConfigSet" : "test1" }, "2" ],
        "default" : [ { "ConfigSet" : "test2" } ]
    },
    "1" : {
        "commands" : {
            "test" : {
                "command" : "echo \"$MAGIC\" > test.txt",
                "env" : { "MAGIC" : "I come from the environment!" },
                "cwd" : "~"
            }
        }
    },
    "2" : {
        "commands" : {
            "test" : {
                "command" : "echo \"$MAGIC\" >> test.txt",
                "env" : { "MAGIC" : "I am test 2!" },
                "cwd" : "~"
            }
        }
    }
}
```

The following calls to cfn-init refer to the configSets declared in the preceding template snippet. The example calls are abbreviated for clarity, see cfn-init (p. 330) for the complete syntax.

- If you specify `test1` only:

  ```
  cfn-init -c test1
  ```

  cfn-init processes config key `1` only.
- If you specify `test2` only:

  ```
  cfn-init -c test2
  ```

  cfn-init processes config key `1` and then processes config key `2`.
- If you specify the `default` configset (or no configsets at all):

  ```
  cfn-init -c default
  ```

  you get the same behavior that you would if you specify configset `test2`.

Commands

You can use the commands key to execute commands on the EC2 instance. The commands are processed in alphabetical order by name.

Key	Description
command	Required. Either an array or a string specifying the command to run. If you use an array, you do not need to escape space characters or enclose command parameters in quotes.
env	Optional. Sets environment variables for the command. This property overwrites, rather than appends, the existing environment.
cwd	Optional. The working directory
test	Optional. A command that must return the value `true` in order for cfn-init to process the command contained in the command key.
ignoreErrors	Optional. A boolean value that determines whether cfn-init continues to run if the command in contained in the command key fails (returns a non-zero value). Set to `true` if you want cfn-init to continue running even if the command fails. Set to `false` if you want cfn-init to stop running if the command fails. The default value is `false`.

The following example snippet calls the echo command.

```
"commands" : {
    "test" : {
        "command" : "echo \"$MAGIC\" > test.txt",
        "env" : { "MAGIC" : "I come from the environment!" },
        "cwd" : "~",
        "test" : "test ! -e ~/test.txt",
        "ignoreErrors" : "false"
    }
}
```

Files

You can use the files key to create files on the EC2 instance. The content can be either inline in the template or the content can be pulled from a URL. The files are written to disk in lexicographic order. The following table lists the supported keys.

Key	Description
content	Either a string or a properly formatted JSON object. If you use a JSON object as your content, the JSON will be written to a file on disk. Any intrinsic functions such as Fn::GetAtt or Ref are evaluated before the JSON object is written to disk.
source	A URL to load the file from. This option cannot be specified with the content key.

Key	Description
encoding	The encoding format. Only used if the content is a string. Encoding is not applied if you are using a source. Valid values: `plain` | `base64`
group	The name of the owning group for this file.
owner	The name of the owning user for this file.
mode	A six-digit octal value representing the mode for this file.
authentication	The name of an authentication method to use. This overrides any default authentication. You can use this property to select an authentication method you define with the AWS::CloudFormation::Authentication (p. 200) resource.

The following example snippet creates a file named setup.mysql as part of a larger installation.

```
"files" : {
  "/tmp/setup.mysql" : {
    "content" : { "Fn::Join" : ["", [
      "CREATE DATABASE ", { "Ref" : "DBName" }, ";\n",
      "CREATE USER '", { "Ref" : "DBUsername" }, "'@'localhost' IDENTIFIED BY
'",
                        { "Ref" : "DBPassword" }, "';\n",
      "GRANT ALL ON ", { "Ref" : "DBName" }, ".* TO '", { "Ref" : "DBUsername"
 },
                        "'@'localhost';\n",
      "FLUSH PRIVILEGES;\n"
      ]]},
    "mode"  : "000644",
    "owner" : "root",
    "group" : "root"
  }
},
```

The full template is available at:
https://s3.amazonaws.com/cloudformation-templates-us-east-1/Drupal_Single_Instance.template

Groups

You can use the groups key to create Linux/UNIX groups and to assign group IDs. To create a group, add a new key-value pair that maps a new group name to an optional group ID. The groups key can contain one or more group names. The following table lists the available keys.

Key	Description
`gid`	A group ID number. If a group ID is specified, and the group already exists by name, the group creation will fail. If another group has the specified group ID, the OS may reject the group creation. Example: `{ "gid" : "23" }`

Example snippet

The following snippet specifies a group named `groupOne` without assigning a group ID and a group named `groupTwo` that specified a group ID value of `45`.

```
"groups" : {
    "groupOne" : {},
    "groupTwo" : { "gid" : "45" }
}
```

Packages

You can use the packages key to download and install pre-packaged applications and components.

Supported package formats

Cfn-init currently supports the following package formats: apt, yum, rubygems, python and rpm. Packages are processed in the following order: rpm, yum/apt, and then rubygems and python. There is no ordering between rubygems and python, and packages within each package manager are not guaranteed to be installed in any order.

Specifying versions

Within each package manager, each package is specified as a package name and a list of versions. The version can be a string, a list of versions, or an empty string or list. An empty string or list indicates that you want the latest version. For rpm manager, the version is specified as a path to a file on disk or a URL.

If you specify a version of a package, cfn-init will attempt to install that version even if a newer version of the package is already installed on the instance. Some package managers support multiple versions, but others may not. Please check the documentation for your package manager for more information. If you do not specify a version and a version of the package is already installed, the cfn-init script will not install a new version—it will assume that you want to keep and use the existing version.

Example snippet

The following snippet specifies a version URL for rpm, requests the latest versions from yum, and version 1.10.2 of chef from rubygems.

```
"rpm" : {
  "epel" : "http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-
4.noarch.rpm"
},
"yum" : {
  "httpd" : [],
  "php" : [],
  "wordpress" : []
},
"rubygems" : {
  "chef" : [ "0.10.2" ]
}
```

Services

You can use the services key to define which services should be enabled or disabled when the instance is launched.

The services key also allows you to specify dependencies on sources, packages and files so that if a restart is needed due to files being installed, cfn-init will take care of the service restart. For example, if you download the Apache HTTP Server package, the package installation will automatically start the Apache HTTP Server during the stack creation process. However, if the Apache HTTP Server configuration is updated later in the stack creation process, the update won't take effect unless the Apache server is restarted. You can use the services key to ensure that the Apache HTTP service is restarted.

The following table lists the supported keys.

Key	Description
ensureRunning	Set to true to ensure that the service is running after cfn-init finishes. Set to false to ensure that the service is not running after cfn-init finishes. Omit this key to make no changes to the service state.
enabled	Set to true to ensure that the service will be started automatically upon boot. Set to false to ensure that the service will not be started automatically upon boot. Omit this key to make no changes to this property.
files	A list of files. If cfn-init changes one directly via the files block, this service will be restarted
sources	A list of directories. If cfn-init expands an archive into one of these directories, this service will be restarted.
packages	A map of package manager to list of package names. If cfn-init installs or updates one of these packages, this service will be restarted.
commands	A list of command names. If cfn-init runs the specified command, this service will be restarted.

The following example snippet configures the services as follows:

- The nginx service will be restarted if either /etc/nginx/nginx.conf or /var/www/html are modified by cfn-init.
- The php-fastcgi service will be restarted if cfn-init installs or updates php or spawn-fcgi using yum.
- The sendmail service will be stopped and disabled.

```
"services" : {
  "sysvinit" : {
    "nginx" : {
      "enabled" : "true",
      "ensureRunning" : "true",
      "files" : ["/etc/nginx/nginx.conf"],
      "sources" : ["/var/www/html"]
    },
    "php-fastcgi" : {
      "enabled" : "true",
      "ensureRunning" : "true",
      "packages" : { "yum" : ["php", "spawn-fcgi"] }
    },
    "sendmail" : {
      "enabled" : "false",
      "ensureRunning" : "false"
    }
  }
}
```

Sources

You can use the sources key to download an archive file and unpack it in a target directory on the EC2 instance.

Supported formats

Supported formats are tar, tar+gzip, tar+bz2 and zip.

GitHub

If you use GitHub as a source control system, you can use cfn-init and the sources package mechanism to pull a specific version of your application. GitHub allows you to create a zip or a tar from a specific version via a URL as follows:

```
https://github.com/<your directory>/(zipball|tarball)/<version>
```

For example, the following snippet pulls down version *master* as a tarfile.

```
"sources" : {
  "/etc/puppet" : https://github.com/user1/cfn-demo/tarball/master
  }
```

Example

The following example downloads a zip file from an Amazon S3 bucket and unpacking it into /etc/myapp:

```
"sources" : {
  "/etc/myapp" : "https://s3.amazonaws.com/mybucket/myapp.tar.gz"
  }
```

You can include authentication credentials for a source by using the AWS::CloudFormation::Authentication (p. 200) resource.

Users

You can use the users key to create Linux/UNIX users on the EC2 instance. The following table lists the supported keys.

Key	Description
uid	A user ID. The creation process fails if the user name exists with a different user ID. If the user ID is already assigned to an existing user the operating system may reject the creation request.
groups	A list of group names. The user will be added to each group in the list.
homeDir	The user's home directory.

Users are created as non-interactive system users with a shell of /sbin/nologin. This is by design and cannot be modified.

```
"users" : {
    "myUser" : {
        "groups" : ["groupOne", "groupTwo"],
        "uid" : "50",
        "homeDir" : "/tmp"
    }
}
```

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::CloudFormation::Stack

The AWS::CloudFormation::Stack type embeds a stack as a resource in a template.

You can add output values from an embedded stack within the containing template. You use the GetAtt (p. 324) function with the embedded stack's logical name and the name of the output value in the embedded stack in the format Outputs.*`EmbeddedStackOutputName`*.

AWS::CloudFormation::Stack Snippets: Stack Resource Snippets (p. 142).

Properties

Property	Type	Required	Notes
TemplateURL	String	Yes	The URL of a template that specifies the stack that you want to create as a resource.
TimeoutInMinutes	String	No	The length of time, in minutes, that AWS CloudFormation waits for the embedded stack to reach the CREATE_COMPLETE state. The default is no timeout. When AWS CloudFormation detects that the embedded stack has reached the CREATE_COMPLETE state, it marks the embedded stack resource as CREATE_COMPLETE in the parent stack and resumes creating the parent stack. If the timeout period expires before the embedded stack reaches CREATE_COMPLETE, AWS CloudFormation marks the embedded stack as failed and rolls back both the embedded stack and parent stack.
Parameters	Parameters type (p. 296)	Conditional	The set of parameters passed to AWS CloudFormation when this embedded stack is created.

Return Values

Ref

For AWS::CloudFormation::Stack, `Ref` returns the Stack ID. For example:

```
arn:aws:cloudformation:us-east-1:803981987763:stack/mystack-myembeddedstack-
sggfrhxhum7w/f449b250-b969-11e0-a185-5081d0136786
```

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

Outputs.*`EmbeddedStackOutputName`*
Returns: The output value from the specified embedded stack where *`EmbeddedStackOutputName`* is the name of the output value.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

AWS::CloudFormation::WaitCondition

The following properties are available with the AWS::CloudFormation::WaitCondition type.

For information about how to use wait conditions, see Creating Wait Conditions in a Template (p. 155).

AWS::CloudFormation::WaitCondition Snippets: Wait Condition Template Snippets (p. 144)

Properties

Property	Type	Required	Notes
Handle	Reference	Yes	A reference to the wait condition handle used to signal this wait condition. Use the `Ref` intrinsic function to specify an AWS::CloudFormation::WaitConditionHandle resource (p. 213).
Timeout	String	Yes	The length of time (in seconds) to wait for the number of signals that the Count property specifies.
Count	String	No	The number of success signals that AWS CloudFormation must receive before it continues the stack creation process. When the wait condition receives requisite the number of success signals, AWS CloudFormation resumes creating the stack. If the wait condition does not receive the specified number of success signals before the Timeout period expires, AWS CloudFormation assumes the wait condition has failed and rolls the stack back.

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Data

Returns: A JSON object containing the *`UniqueId`* and *`Data`* values from the wait condition signal(s) for the specified wait condition. For more information about wait condition signals, see Wait Condition Signal JSON Format (p. 158).

Example return value for wait condition with 2 signals:

```
{ "Signal1" : "Step 1 complete." , "Signal2" : "Step 2 complete." }
```

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

AWS::CloudFormation::WaitConditionHandle

The AWS::CloudFormation::WaitConditionHandle type has no properties.

The AWS::CloudFormation::WaitConditionHandle resource is used for signaling an associated AWS::CloudFormation::WaitCondition (p. 212) resource.

When you specify an AWS::CloudFormation::WaitConditionHandle type as an argument to the `Ref` function, AWS CloudFormation returns the the pre-signed URL you use to deliver a signal to the AWS::CloudFormation::WaitCondition resource associated with that wait condition handle.

For information about how to use wait conditions, see Creating Wait Conditions in a Template (p. 155).

AWS::CloudFormation::WaitCondition Snippets: Wait Condition Template Snippets (p. 144)

AWS::CloudFront::Distribution

Creates an Amazon CloudFront download distribution. For general information about CloudFront distributions, see the Introduction to Amazon CloudFront in the *Amazon CloudFront Developer Guide*. For specific information about creating CloudFront download distributions, see POST Distribution in the *Amazon CloudFront API Reference*.

Syntax

```
{
   "Type" : "AWS::CloudFront::Distribution",
   "Properties" : {
      "DistributionConfig (p. 214)" : DistributionConfig
   }
}
```

Properties

DistributionConfig

The distribution's configuration information.

Required: Yes

Type: DistributionConfig (p. 300) type

Update requires: replacement (p. 34)

Return Values

Ref

Returns: The CloudFront distribution ID. For example: `E27LVI50CSW06W`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

DomainName

Returns: The domain name of the resource. For example: `d2fadu0nynjpfn.cloudfront.net`.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

AWS::CloudWatch::Alarm

The AWS::CloudWatch::Alarm type creates an Amazon CloudWatch alarm.

This type supports updates. For more information about updating this resource, see PutMetricAlarm. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

When you specify an AWS::CloudWatch::Alarm type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`AlarmName`*.

Properties

Property	Type	Required	Notes
ActionsEnabled	String	No	Indicates whether or not actions should be executed during any changes to the alarm's state. Either *`true`* or *`false`*.

Property	Type	Required	Notes
AlarmActions	List of String	No	The list of actions to execute when this alarm transitions into an ALARM state from any other state. Each action is specified as an Amazon Resource Number (ARN). Currently the only action supported is publishing to an Amazon SNS topic or an Amazon Auto Scaling policy.
AlarmDescription	String	No	The description for the alarm.
ComparisonOperator	String	Yes	The arithmetic operation to use when comparing the specified Statistic and Threshold. The specified Statistic value is used as the first operand. Valid Values: `GreaterThanOrEqualToThreshold` | `GreaterThanThreshold` | `LessThanThreshold` | `LessThanOrEqualToThreshold`
Dimensions	List of Metric Dimension (p. 304) type	No	The dimensions for the alarm's associated metric.
EvaluationPeriods	String	Yes	The number of periods over which data is compared to the specified threshold.
InsufficientDataActions	List of String	No	The list of actions to execute when this alarm transitions into an INSUFFICIENT_DATA state from any other state. Each action is specified as an Amazon Resource Number (ARN). Currently the only action supported is publishing to an Amazon SNS topic or an Amazon Auto Scaling policy.
MetricName	String	Yes	The name for the alarm's associated metric.
Namespace	String	Yes	The namespace for the alarm's associated metric.
OKActions	List of String	No	The list of actions to execute when this alarm transitions into an OK state from any other state. Each action is specified as an Amazon Resource Number (ARN). Currently the only action supported is publishing to an Amazon SNS topic or an Amazon Auto Scaling policy.
Period	String	Yes	The period in seconds over which the specified statistic is applied.
Statistic	String	Yes	The statistic to apply to the alarm's associated metric. Valid Values: `SampleCount` | `Average` | `Sum` | `Minimum` | `Maximum`
Threshold	String	Yes	The value against which the specified statistic is compared.

Property	Type	Required	Notes
Unit	String	No	The unit for the alarm's associated metric. Valid Values: Seconds | Microseconds | Milliseconds | Bytes | Kilobytes | Megabytes | Gigabytes | Terabytes | Bits | Kilobits | Megabits | Gigabits | Terabits | Percent | Count | Bytes/Second | Kilobytes/Second | Megabytes/Second | Gigabytes/Second | Terabytes/Second | Bits/Second | Kilobits/Second | Megabits/Second | Gigabits/Second | Terabits/Second | Count/Second | None

Return Values

Function	Type	Description
Ref (p. 326)	Name	mystack-myalarm-3AOHFRGOXR5T

AWS::DynamoDB::Table

Creates an Amazon DynamoDB table.

Syntax

```
{
   "Type" : "AWS::DynamoDB::Table",
   "Properties" : {
      "KeySchema (p. 216)" : { DynamoDB Primary Key },
      "ProvisionedThroughput (p. 216)" : { DynamoDB Provisioned Throughput }
   }
}
```

Properties

KeySchema

The primary key structure for the table, consisting of a required *HashKeyElement* and an optional *RangeKeyElement*, required only for *composite* primary keys. For more information about primary keys, see DynamoDB Primary Key (p. 304).

Required: Yes

Type: DynamoDB Primary Key (p. 304)

Update requires: replacement (p. 34)

ProvisionedThroughput

New throughput for the specified table, consisting of values for ReadCapacityUnits and WriteCapacityUnits. For more information about the contents of a Provisioned Throughput structure, see DynamoDB Provisioned Throughput (p. 305).

Required: Yes

Type: DynamoDB Provisioned Throughput (p. 305)

Update requires: replacement (p. 34)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example Simple DynamoDB Template

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myDynamoDBTable" : {
         "Type" : "AWS::DynamoDB::Table",
         "Properties" : {
            "KeySchema" : {
               "HashKeyElement": {
                  "AttributeName" : "AttributeName1",
                  "AttributeType" : "S"
               },
               "RangeKeyElement" : {
                  "AttributeName" : "AttributeName2",
                  "AttributeType" : "N"
               }
            },
            "ProvisionedThroughput" : {
               "ReadCapacityUnits" : "5",
               "WriteCapacityUnits" : "10"
            }
         }
      }
   }
}
```

AWS::EC2::CustomerGateway

Provides information to AWS about your VPN customer gateway device.

For more information, go to CreateCustomerGateway in the *Amazon Elastic Compute Cloud API Reference*.

The following properties are available with the AWS::EC2::CustomerGateway type.

Property	Type	Required	Notes
Type	String	Yes	The type of VPN connection this customer gateway supports. Example: ipsec.1
IpAddress	String	Yes	The Internet-routable IP address for the customer gateway's outside interface. The address must be static.
BgpAsn	Integer	Yes	The customer gateway's Border Gateway Protocol (BGP) Autonomous System Number (ASN).

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myCustomerGateway" : {
         "Type" : "AWS::EC2::CustomerGateway",
         "Properties" : {
            "Type" : "ipsec.1",
            "BgpAsn" : "64000",
            "IpAddress" : "1.1.1.1"
         }
      }
   }
}
```

AWS::EC2::DHCPOptions

Creates a set of DHCP options for your VPC.

For more information, go to CreateDhcpOptions in the *Amazon Elastic Compute Cloud API Reference*.

The following properties are available with the AWS::EC2::DHCPOptions type.

Property	Type	Required	Notes
DomainName	String	Conditional	A domain name of your choice. Condition: At least one DHCP option must be specified Example: example.com
DomainNameServers	String list	Conditional	The IP address of a domain name server. You can specify up to four addresses. Condition: At least one DHCP option must be specified Valid values: IPv4 addresses Example: 10.0.0.1
NTPServers	String	Conditional	The IP address of a Network Time Protocol (NTP) server. You can specify up to four addresses. Condition: At least one DHCP option must be specified Valid values: IPv4 addresses Example: 10.0.0.1
NetbiosNameServers	String	Conditional	The IP address of a NetBIOS name server. You can specify up to four addresses. Condition: At least one DHCP option must be specified Valid values: IPv4 addresses Example: 10.0.0.1
NetbiosNodeType	Number list	Conditional	Value indicating the NetBIOS node type (1, 2, 4, or 8). For more information about the values, go to RFC 2132. We recommend you only use 2 at this time (broadcast and multicast are currently not supported). Condition: At least one DHCP option must be specified Valid values: `1` | `2` | `4` | `8`
Tags	List of EC2 Tags (p. 310) type	No	The tags you want to attach to this resource. For more information about tags, go to Using Tags in the *Amazon Elastic Compute Cloud User Guide*. *Update requires*: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myDhcpOptions" : {
         "Type" : "AWS::EC2::DHCPOptions",
         "Properties" : {
            "DomainName" : "example.com",
            "DomainNameServers" : ["AmazonProvidedDNS"],
            "NtpServers" : ["10.2.5.1"],
            "NetbiosNameServers" : ["10.2.5.1"],
            "NetbiosNodeType" : "2",
            "Tags" : [{"Key" : "foo", "Value" : "bar"}],
            }
         }
      }
   }
}
```

AWS::EC2::EIP

The AWS::EC2::EIP resource allocates an Elastic IP (EIP) address and can, optionally, associate it with an Amazon EC2 instance.

Syntax

```
{
   "Type" : "AWS::EC2::EIP",
   "Properties" : {
      "InstanceId (p. 220)" : String,
      "Domain (p. 220)" : String
   }
}
```

Properties

InstanceId

The Instance ID of the Amazon EC2 instance that you want to associate with this Elastic IP address.

Required: No

Type: String

Domain

Set to "vpc" to allocate the address to your Virtual Private Cloud (VPC). No other values are supported.

For more information, see AllocateAddress in the *Amazon Elastic Compute Cloud API Reference*. For more information about Elastic IP Addresses in VPC, go to Elastic IP Addresses in the *Amazon VPC User Guide*.

Required: Conditional. Required when allocating an address to a VPC

Type: String

Return Values

Ref

When you specify the logical ID of an AWS::EC2::EIP object as an argument to the `Ref` function, AWS CloudFormation returns the value of the associated instance's *`PublicIp`*.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

AllocationId
The ID that AWS assigns to represent the allocation of the address for use with Amazon VPC. THis is returned only for VPC elastic IP addresses. Example return value: `eipalloc-5723d13e`

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Examples

To view AWS::EC2::EIP snippets, see Assigning an Amazon EC2 Elastic IP Using AWS::EC2::EIP Snippet (p. 112).

AWS::EC2::EIPAssociation

The AWS::EC2::EIPAssociation resource associates an Elastic IP address with an Amazon EC2 instance. The Elastic IP address can be an existing Elastic IP address or an Elastic IP address allocated through an AWS::EC2::EIP resource (p. 220).

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::SQS::Queue",
   "Properties": {
      "AllocationId (p. 221)": String
      "EIP (p. 222)": String
      "InstanceId (p. 222)": String
      "NetworkInterfaceId (p. 222)": String
   }
}
```

Properties

AllocationId
Allocation ID for the VPC Elastic IP address you want to associate with an Amazon EC2 instance in your VPC.

Required: Conditional. This property must not be specified if the EIP property is specified.

Type: String
Update requires: replacement (p. 34)

EIP
Elastic IP address that you want to associate with the Amazon EC2 instance specified by the InstanceId property. You can specify an existing Elastic IP address or a reference to an Elastic IP address allocated with a AWS::EC2::EIP resource (p. 220).

Required: Conditional. This property must not be specified if the AllocationId property is specified.

Type: String
Update requires: replacement (p. 34)

InstanceId
Instance ID of the Amazon EC2 instance that you want to associate with the Elastic IP address specified by the EIP property.

Required: No

Type: String

Update requires: replacement (p. 34)

NetworkInterfaceId
The ID of the network interface to associate with the Elastic IP address (VPC only).

Required: No

Type: String

Update requires: replacement (p. 34)

Return Values

Ref

When you specify an AWS::EC2::EIPAssociation type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`PublicIp`*.

Examples

For AWS::EC2::EIPAssociation snippets, see Assigning an Amazon EC2 Elastic IP Using AWS::EC2::EIP Snippet (p. 112).

AWS::EC2::Instance

The AWS::EC2::Instance type creates an Amazon EC2 instance.

If an Elastic IP address is attached to your instance, AWS CloudFormation will reattach the Elastic IP address after it updates the instance. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "AvailabilityZone (p. 223)" : String,
      "DisableApiTermination (p. 223)" : Boolean,
      "IamInstanceProfile (p. 223)" : String,
      "ImageId (p. 223)" : String,
      "InstanceType (p. 224)" : String,
      "KernelId (p. 224)" : String,
      "KeyName (p. 224)" : String,
      "Monitoring (p. 224)" : Boolean,
      "PlacementGroupName (p. 224)" : String,
      "PrivateIpAddress (p. 224)" : String,
      "RamdiskId (p. 225)" : String,
      "SecurityGroups (p. 225)" : [ String, ... ],
      "SecurityGroupIds (p. 225)" : [ String, ... ],
      "SourceDestCheck (p. 225)" : Boolean,
      "SubnetId (p. 225)" : String,
      "Tags (p. 225)" : [ EC2 Tag (p. 310), ... ],
      "Tenancy (p. 226)" : String,
      "UserData (p. 226)" : String,
      "Volumes (p. 226)" : [ EC2 MountPoint (p. 306), ... ]
   }
}
```

Properties

AvailabilityZone

Specifies the name of the availability zone the instance is located in. If not specified, the default availability zone for the region will be used.

Required: No

Type: String

Update requires: replacement (p. 34)

DisableApiTermination

Indicates whether the instance can be terminated through the API (specify "true"), or not (specify "false").

Required: No

Type: Boolean

Update requires: no interruption (p. 33)

IamInstanceProfile

A reference to an AWS::IAM::InstanceProfile type.

For detailed information about IAM Roles, see Working with Roles in the *AWS Identity and Access Management User Guide*.

Required: No

Type: String

ImageId

Provides the unique ID of the Amazon Machine Image (AMI) that was assigned during registration.

Required: Yes

Type: String

Update requires: replacement (p. 34)

InstanceType

The instance type (for example, "m1.small").

Required: Yes

Type: String

Update requires:

- *Update requires*: some interruptions (p. 34) (EBS-backed AMIs)
- *Update requires*: replacement (p. 34) (instance-store backed AMIs)

KernelId

The kernel ID

Required: No

Type: String

Update requires:

- *Update requires*: some interruptions (p. 34) (EBS-backed AMIs)
- *Update requires*: replacement (p. 34) (instance-store backed AMIs)

KeyName

Provides the name of the EC2 key pair.

Required: No

Type: String

Update requires: replacement (p. 34)

Monitoring

Whether monitoring is enabled for the instance.

Required: No

Type: Boolean

Update requires: no interruption (p. 33)

PlacementGroupName

The name of an existing placement group you want to launch the intstance into (for cluster instances).

Required: No

Type: String

Update requires: replacement (p. 34)

PrivateIpAddress

The private IP address for this instance.

If you're using an Amazon Virtual Private Cloud (VPC), you can optionally use this parameter to assign the instance a specific available IP address from the subnet (e.g., 10.0.0.25). By default, Amazon VPC selects an IP address from the subnet for the instance.

Required: No

Type: String

Update requires: replacement (p. 34)

RamdiskId

The ID of the RAM disk to select. Some kernels require additional drivers at launch. Check the kernel requirements for information on whether you need to specify a RAM disk. To find kernel requirements, refer to the AWS Resource Center and search for the kernel ID.

Required: No

Type: String

Update requires: some interruptions (p. 34) (EBS-backed AMIs).

Update requires: replacement (p. 34) (instance-store backed AMIs).

SecurityGroups

Valid only for EC2 security groups. A list containing the EC2 security groups to assign to the Amazon EC2 instance. The list can contain both the name of existing EC2 security groups or references to AWS::EC2::SecurityGroup resources created in the template.

Required: No

Type: list of Strings

Update requires: replacement (p. 34).

SecurityGroupIds

A list containing the security group IDs for VPC security groups to assign to the Amazon EC2 instance.

Required: Conditional. Required for VPC security groups.

Type: list of Strings

Update requires: no interruption (p. 33)

SourceDestCheck

Controls whether source/destination checking is enabled on the instance. Also determines if an instance in a VPC to perform network address translation (NAT).

A value of "true" means checking is enabled, and "false" means checking is disabled. The value *must* be "false" for the instance to perform NAT. For more information, go to NAT Instances in the Amazon Virtual Private Cloud User Guide.

Required: No

Type: Boolean

Update requires: no interruption (p. 33)

SubnetId

If you're using Amazon Virtual Private Cloud, this specifies the ID of the subnet you want to launch the instance into.

Required: No

Type: String

Update requires: replacement (p. 34)

Tags

The tags you want to attach to the instance.

Required: No

Type: List of EC2 Tags (p. 310)

Update requires: no interruption (p. 33)

Tenancy

The tenancy of the instance you want to launch. Can be either "default" or "dedicated". An instance with a *`tenancy`* value of `"dedicated"` runs on single-tenant hardware and can be launched only into a VPC. For more information, see Using EC2 Dedicated Instances Within Your VPC in the *Amazon Virtual Private Cloud User Guide*.

Required: No

Type: String

Update requires: replacement (p. 34)

UserData

Base64-encoded MIME user data made available to the instances.

Required: No

Type: String

Update requires:

- *Update requires*: some interruptions (p. 34) (EBS-backed AMIs)
- *Update requires*: replacement (p. 34) (instance-store backed AMIs)

Volumes

The EBS volumes to attach to the instance.

Required: No

Type: List of EC2 MountPoints (p. 306).

Update requires: You must unmount a filesystem that you want to update.

Return Values

Ref

When you pass the logical ID of an AWS::EC2::Instance object to the intrinsic `Ref` function, the object's InstanceId is returned. For example: `i-636be302`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

AvailabilityZone

The availability zone where the specified instance is launched. For example: `us-east-1b`.

You can retrieve a list of all availability zones for a region with the Fn::GetAZs (p. 325) intrinsic function.

PrivateDnsName

The private DNS name of the specified instance. For example: `ip-10-24-34-0.ec2.internal`

PublicDnsName

Public DNS name of the specified instance. For example: `ec2-107-20-50-45.compute-1.amazonaws.com`

PrivateIp

The private IP address of the specified instance. For example: `10.24.34.0`

PublicIp

The public IP address of the specified instance. For example: `192.0.2.0`

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Examples

Templates can be downloaded that demonstrate using AWS::EC2::Instance to create a Virtual Private Cloud (VPC):

- Single instance in a single subnet
- Multiple subnets with ELB and Auto Scaling group

For an example of an AWS::EC2::Instance with an IAM instance profile, see: Create an EC2 instance with an associated instance profile.

For more EC2 template examples, see: Amazon EC2 Snippets (p. 112).

AWS::EC2::InternetGateway

Creates a new Internet gateway in your AWS account. After creating the Internet gateway, you then attach it to a VPC.

For more information, go to CreateInternetGateway in the *Amazon Elastic Compute Cloud API Reference*.

The following properties are available with the AWS::EC2::InternetGateway type.

Property	Type	Required	Notes
Tags	List of EC2 Tags (p. 310) type	No	The tags you want to attach to this resource. For more information about tags, go to Using Tags in the *Amazon Elastic Compute Cloud User Guide*. *Update requires*: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myInternetGateway" : {
             "Type" : "AWS::EC2::InternetGateway",
              "Properties" : {
                     "Tags" : [ {"Key" : "foo", "Value" : "bar"}]
                 }
             }
        }
    }
}
```

AWS::EC2::NetworkAcl

Creates a new network ACL in a VPC.

For more information, go to CreateNetworkAcl in the *Amazon Elastic Compute Cloud API Reference*. For more information about network ACLs, go to Network ACLs in the *Amazon Virtual Private Cloud User Guide*.

Syntax

```
{
   "Type" : "AWS::EC2::NetworkAcl",
   "Properties" : {
      "VpcId (p. ?)" : String,
      "Tags (p. ?)" : [ EC2 Tags (p. 310) ]
   }
}
```

Properties

VpcId
: The ID of the VPC where the network ACL will be created.

  Required: Yes

  Type: String

Tags
: The tags you want to attach to this resource.

  For more information about tags, go to Using Tags in the *Amazon Elastic Compute Cloud User Guide*.
  Update requires: no interruption (p. 33)

  Required: No

  Type: List of EC2 Tags (p. 310)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myNetworkAcl" : {
         "Type" : "AWS::EC2::NetworkAcl",
         "Properties" : {
            "VpcId" : { "Ref" : "myVPC" },
            "Tags" : [ { "Key" : "foo", "Value" : "bar" } ]
         }
      }
   }
}
```

AWS::EC2::NetworkAclEntry

Creates an entry (i.e., rule) in a network ACL with a rule number you specify. Each network ACL has a set of numbered ingress rules and a separate set of numbered egress rules.

For more information, go to NetworkAclEntry in the *Amazon Elastic Compute Cloud API Reference*. For more information about network ACLs, go to Network ACLs in the *Amazon Virtual Private Cloud User Guide*.

Syntax

```
{
   "Type" : "AWS::EC2::NetworkAclEntry",
   "Properties" : {
      "NetworkAclId (p. 229)" : String,
      "RuleNumber (p. 230)" : Integer,
      "Protocol (p. 230)" : Integer,
      "RuleAction (p. 230)" : String,
      "Egress (p. 230)" : Boolean,
      "CidrBlock (p. 230)" : String,
      "Icmp (p. 230)" : EC2 ICMP Property Type (p. 306),
      "PortRange (p. 230)" : EC2 PortRange Property Type (p. 308),
}
```

Properties

NetworkAclId
  ID of the ACL where the entry will be created.
  Required: Yes

Type: String

RuleNumber
Rule number to assign to the entry (e.g., 100). This must be a postive integer from 1 to 32766.
Required: Yes

Type: Integer

Protocol
IP protocol the rule applies to. You can use -1 to mean all protocols. This must be -1 or a protocol number (go to Protocol Numbers at iana.org).
Required: Yes

Type: Integer

RuleAction
Whether to allow or deny traffic that matches the rule; valid values are "allow" or "deny".
Required: Yes

Type: String

Egress
Whether this rule applies to egress traffic from the subnet ("true") or ingress traffic to the subnet ("false").
Required: Yes

Type: Boolean

CidrBlock
The CIDR range to allow or deny, in CIDR notation (e.g., 172.16.0.0/24).
Required: Yes

Type: String

Icmp
The Internet Control Message Protocol (ICMP) code and type.
Required: Conditional, required if specifying 1 (ICMP) for the protocol parameter.

Type: EC2 ICMP Property Type (p. 306)

PortRange
The range of port numbers for the UDP/TCP protocol.
Required: Conditional, required if specifying 6 (TCP) or 17 (UDP) for the protocol parameter.

Type: EC2 PortRange Property Type (p. 308)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myNetworkAclEntry" : {
         "Type" : "AWS::EC2::NetworkAclEntry",
         "Properties" : {
            "NetworkAclId" : { "Ref" : "myNetworkAcl" },
            "RuleNumber" : "100",
            "Protocol" : "-1",
            "RuleAction" : "allow",
            "Egress" : "true",
            "CidrBlock" : "172.16.0.0/24",
            "Icmp" : { "Code" : "-1", "Type" : "-1" },
            "PortRange" : { "From" : "53", "To" : "53" },
         }
      }
   }
}
```

AWS::EC2::NetworkInterface

Describes a network interface in an Elastic Compute Cloud (EC2) instance for AWS CloudFormation. This is provided in a list in the NetworkInterfaces property of AWS::EC2::Instance (p. 222).

Syntax

```
{
   "Type" : "AWS::EC2::NetworkInterface",
   "Properties" : {
      "Description (p. 231)": String,
      "GroupSet (p. 231)": [String],
      "PrivateIpAddress (p. 232)": String,
      "SourceDestCheck (p. 232)": Boolean,
      "SubnetId (p. 232)": String,
      "Tags (p. 232)": [EC2 Tag (p. 310), ...],
   }
}
```

Properties

Description

The description of this network interface.

Required: No

Type: String

Update requires: no interruption (p. 33)

GroupSet

A list of security group IDs associated with this network interface.

Required: No

Type: String.

Update requires: no interruption (p. 33)

PrivateIpAddress

IP address of the interface within the subnet.

Required: No

Type: String

Update requires: replacement (p. 34)

SourceDestCheck

Flag indicating whether traffic to or from the instance is validated.

Required: No

Type: Boolean

Update requires: no interruption (p. 33)

SubnetId

The ID of the subnet to associate with the network interface.

Required: Yes

Type: String

Update requires: replacement (p. 34)

Tags

A list of tags associated with this network interface.

Required: No

Type: List of EC2 Tag (p. 310)s.

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Tip

For more NetworkInterface template examples, see Elastic Network Interface (ENI) Template Snippets (p. 113).

Simple Standalone ENI

This is a simple stand-alone Elastic Network Interface (ENI), using all of the available properties.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Description" : "Simple Standalone ENI",
   "Resources" : {
      "myENI" : {
         "Type" : "AWS::EC2::NetworkInterface",
         "Properties" : {
            "Tags": [{"Key":"foo","Value":"bar"}],
            "Description": "A nice description.",
            "SourceDestCheck": "false",
            "GroupSet": ["sg-75zzz219"],
            "SubnetId": "subnet-3z648z53",
            "PrivateIpAddress": "10.0.0.16"
         }
      }
   }
}
```

ENI on an EC2 instance

This is an example of an ENI on an EC2 instance. In this example, one ENI is added to the instance, but since NetworkInterfaces is a list, you can add more than one.

```
"Ec2Instance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
      "KeyName" : { "Ref" : "KeyName" },
      "SecurityGroupIds" : [{ "Ref" : "WebSecurityGroup" }],
      "SubnetId" : { "Ref" : "SubnetId" },
      "NetworkInterfaces" : [ {
         "NetworkInterfaceId" : {"Ref" : "controlXface"}, "DeviceIndex" : "1" } ],
      "Tags" : [ {"Key" : "Role", "Value" : "Test Instance"}],
      "UserData" : { "Fn::Base64" : { "Ref" : "WebServerPort" }}
   }
}
```

AWS::EC2::Route

Creates a new route in a route table within a VPC. The route's target can be either a gateway attached to the VPC or a NAT instance in the VPC.

For more information, go to CreateRoute in the *Amazon Elastic Compute Cloud API Reference*. For more information about route tables, see Route Tables in the *Amazon Virtual Private Cloud User Guide*.

Syntax

```
{
   "Type" : "AWS::EC2::Route",
   "Properties" : {
      "DestinationCidrBlock (p. 234)" : String,
      "GatewayId (p. 234)" : String,
      "InstanceId (p. 234)" : String,
      "NetworkInterfaceId (p. 234)" : String,
      "RouteTableId (p. 234)" : String,
      "Tags (p. 234)" : [ EC2 Tag, ... ]
}
```

Properties

DestinationCidrBlock
The CIDR address block used for the destination match. For example, `"0.0.0.0/0"`. Routing decisions are based on the most specific match.

Required: Yes

Type: String

GatewayId
The ID of a gateway attached to your VPC. For example: `"igw-eaad4883"`.

Required: Conditional. You must provide *only one* of the following: a *`GatewayID`*, *`InstanceID`*, or *`NetworkInterfaceId`*.

Type: String

InstanceId
The ID of a NAT instance in your VPC. For example, `"i-1a2b3c4d"`.

Required: Conditional. You must provide *only one* of the following: a *`GatewayID`*, *`InstanceID`*, or *`NetworkInterfaceId`*.

Type: String

NetworkInterfaceId
Allows the routing of network interface IDs.

Required: Conditional. You must provide *only one* of the following: a *`GatewayID`*, *`InstanceID`*, or *`NetworkInterfaceId`*.

Type: String

RouteTableId
The ID of the route table where the route will be added.

Required: Yes

Type: String

Tags
The tags you want to attach to this resource.

For more information about tags, go to Using Tags in the *Amazon Elastic Compute Cloud User Guide*.

Required: No

Type: List of EC2 Tags (p. 310).

Update requires: no interruption (p. 33)

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name.

For more information about using the `Ref` function, see Ref (p. 326).

Examples

Example Route with Gateway ID

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myRoute" : {
         "Type" : "AWS::EC2::Route",
         "Properties" : {
            "RouteTableId" : { "Ref" : "myRouteTable" },
            "DestinationCidrBlock" : "0.0.0.0/0",
            "GatewayId" : { "Ref" : "myInternetGateway" },
            "Tags" : [ { "Key" : "foo", "Value" : "bar" } ]
         }
      }
   }
}
```

Example Route with Instance ID

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myRoute" : {
         "Type" : "AWS::EC2::Route",
         "Properties" : {
            "RouteTableId" : { "Ref" : "myRouteTable" },
            "DestinationCidrBlock" : "0.0.0.0/0",
            "InstanceId" : { "Ref" : "myInstance" },
            "Tags" : [ { "Key" : "foo", "Value" : "bar" } ]
         }
      }
   }
}
```

Example Route with Network Interface ID.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myRoute" : {
         "Type" : "AWS::EC2::Route",
         "Properties" : {
            "RouteTableId" : { "Ref" : "myRouteTable" },
            "DestinationCidrBlock" : "0.0.0.0/0",
            "NetworkInterfaceId" : { "Ref" : "eni-1a2b3c4d" },
            "Tags" : [ { "Key" : "foo", "Value" : "bar" } ]
         }
      }
   }
}
```

AWS::EC2::RouteTable

Creates a new route table within a VPC. After you create a new route table, you can add routes and associate the table with a subnet.

For more information, go to CreateRouteTable in the *Amazon Elastic Compute Cloud API Reference* or Route Tables in the *Amazon Virtual Private Cloud User Guide*.

The following properties are available with the AWS::EC2::RouteTable type.

Property	Type	Required	Notes
VpcId	String	Yes	The ID of the VPC where the route table will be created. Example: vpc-11ad4878
Tags	List of EC2 Tags (p. 310) type	No	The tags you want to attach to this resource. For more information about tags, go to Using Tags in the *Amazon Elastic Compute Cloud User Guide*. *Update requires*: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

The following example snippet uses the VPC ID from a VPC named *myVPC* that was declared elsewhere in the same template.

```
    {
        "AWSTemplateFormatVersion" : "2010-09-09",
        "Resources" : {
            "myRouteTable" : {
                 "Type" : "AWS::EC2::RouteTable",
                  "Properties" : {
                      "VpcId" : {"Ref" : "myVPC"},

                      "Tags" : [ {"Key" : "foo", "Value" : "bar"}]
                  }
            }
        }
    }
```

AWS::EC2::SecurityGroup

When you specify an AWS::EC2::SecurityGroup type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`GroupName`*.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For information about Amazon EC2 security groups, go to Using Security Groups in the *Amazon Elastic Compute Cloud User Guide*.

To create a VPC security group, use the VpcId property.

The following properties are available with the AWS::EC2::SecurityGroup type.

Syntax

```
{
   "Type" : "AWS::EC2::SecurityGroup",
   "Properties" :
   {
      "GroupDescription" : String,
      "VpcId" : {"Ref" : "myVPC"},
      "SecurityGroupIngress" : [EC2 Security Group],
      "SecurityGroupEgress" : [EC2 Security Group]
   }
}
```

Properties

GroupDescription
Description of the security group.

Type: String

Required: Yes

Update requires: no interruption (p. 33)

VpcId

ID of the VPC.

Type: VPC security group ID

Required: Yes, for VPC security groups

Update requires: no interruption (p. 33)

> **Note**
>
> For more information about VPC security groups, go to Security Groups in the *Amazon Virtual Private Cloud User Guide*.

SecurityGroupIngress

A list of EC2 security group ingress rules.

Type: EC2 Security Group Rule Property Type (p. 309)

Required: No

Update requires: no interruption (p. 33)

SecurityGroupEgress

A list of EC2 security group egress rules.

Type: EC2 Security Group Rule Property Type (p. 309)

Required: No

Update requires: no interruption (p. 33)

Example

AWS::EC2::SecurityGroup exists as a top-level element inside an AWS CloudFormation template. Here's an example:

```
"InstanceSecurityGroup" : {
   "Type" : "AWS::EC2::SecurityGroup",
   "Properties" : {
      "GroupDescription" : "Allow http to client host",
      "VpcId" : {"Ref" : "myVPC"},
      "SecurityGroupIngress" : [{
            "IpProtocol" : "tcp",
            "FromPort" : "80",
            "ToPort" : "80",
            "CidrIp" : "0.0.0.0/0"
         }],
      "SecurityGroupEgress" : [{
         "IpProtocol" : "tcp",
         "FromPort" : "80",
         "ToPort" : "80",
         "CidrIp" : "0.0.0.0/0"
      }]
   }
}
```

AWS::EC2::SecurityGroupIngress

The AWS::EC2::SecurityGroupIngress type adds an ingress rule to an Amazon EC2 or VPC security group.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For more information about adding ingress rules to Amazon EC2 or VPC security groups, go to AuthorizeSecurityGroupIngress in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "GroupName" : String
   "GroupId" : String
   "IpProtocol" : String
   "CidrIp" : String
   "SourceSecurityGroupName" : String
   "SourceSecurityGroupId" : String
   "SourceSecurityGroupOwnerId" : String
   "FromPort" : Number
   "ToPort" : Number
}
```

Properties

GroupName
Name of the EC2 security group to modify. This value can be a reference to an AWS::EC2::SecurityGroup (p. 237) resource or the name of an existing EC2 security group.

Type: String

Required: Can be used *instead of* GroupId for EC2 security groups.

Update requires: no interruption (p. 33)

GroupId
ID of the EC2 or VPC security group to modify. The group must belong to your account.

Type: String

Required: Yes, for VPC security groups; can be used instead of GroupName for EC2 security groups

Update requires: no interruption (p. 33)

IpProtocol
IP protocol name or number. For valid values, see the IpProtocol parameter in AuthorizeSecurityGroupIngress

Type: String

Required: Yes

Update requires: no interruption (p. 33)

CidrIp
Specifies a CIDR range.

For an overview of CIDR ranges, go to the Wikipedia Tutorial.

Condition: If you specify SourceSecurityGroupName, do not specify CidrIp.

Type: String

Required: Conditional—if you specify SourceSecurityGroupName, do not specify CidrIp.

Update requires: no interruption (p. 33)

SourceSecurityGroupName
Specifies the name of the Amazon EC2 Security Group to allow access or uses the Ref intrinsic function to refer to the logical name of a security group defined in the same template.

Type: String

Required: Conditional—if you specify CidrIp, do not specify SourceSecurityGroupName.

Update requires: no interruption (p. 33)

SourceSecurityGroupId

Specifies the ID of the source Security Group or uses the Ref intrinsic function to refer to the logical ID of a security group defined in the same template.

Condition: If you specify CidrIp, do not specify SourceSecurityGroupId.

Type: String

Required: Conditional—if you specify CidrIp, do not specify SourceSecurityGroupId.

Update requires: no interruption (p. 33)

SourceSecurityGroupOwnerId

Specifies the AWS Account ID of the owner of the Amazon EC2 Security Group specified in the SourceSecurityGroupName property.

Type: String

Required: Conditional—if you specify SourceSecurityGroupName and that security group is owned by a different account than the account creating the stack, you must specify the SourceSecurityGroupOwnerId; otherwise, this property is optional.

Update requires: no interruption (p. 33)

FromPort

Start of port range for the TCP and UDP protocols, or an ICMP type number. An ICMP type number of -1 indicates a wildcard (i.e., any ICMP type number).

Type: String

Required: Yes, for ICMP and any protocol that uses ports.

Update requires: no interruption (p. 33)

ToPort

End of port range for the TCP and UDP protocols, or an ICMP code. An ICMP code of -1 indicates a wildcard (i.e., any ICMP code).

Type: String

Required: Yes, for ICMP and any protocol that uses ports.

Update requires: no interruption (p. 33)

Examples

EC2 Security Group and Ingress Rule

To create an Amazon EC2 (non-VPC) security group and an ingress rule, use the SourceSecurityGroupName property in the ingress rule.

The following template snippet creates an EC2 security group with an ingress rule that allows incoming traffic on port 80 from any other host in the security group. The snippet uses the intrinsic function Ref (p. 326) to specify the value for SourceSecurityGroupName.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "SGBase": {
            "Type": "AWS::EC2::SecurityGroup",
            "Properties": {
                "GroupDescription": "Base Security Group",
                "SecurityGroupIngress": [
                    {
                        "IpProtocol": "tcp",
                        "CidrIp": "0.0.0.0/0",
                        "FromPort": "22",
                        "ToPort": "22"
                    }
                ]
            }
        },
        "SGBaseIngress": {
            "Type": "AWS::EC2::SecurityGroupIngress",
            "Properties": {
                "GroupName": { "Ref": "SGBase" },
                "IpProtocol": "tcp",
                "FromPort": "80",
                "ToPort": "80",
                "SourceSecurityGroupName": { "Ref": "SGBase" }
            }
        }
    }
}
```

VPC Security Group and Ingress Rule

To create a Amazon VPC security group and an ingress rule:

- Specify the VpcId property in the security group.
- Use the SourceSecurityGroupId property in the ingress rule.

The following template snippet creates a VPC security group with an ingress rule that allows incoming traffic on port 80 from any other host in the security group. The snippet uses the intrinsic function Ref (p. 326) to specify the value for SourceSecurityGroupId.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "SGBase": {
      "Type": "AWS::EC2::SecurityGroup",
      "Properties": {
        "VpcId" : "vpc-12345678",
        "GroupDescription": "Base Security Group",
        "SecurityGroupIngress": [
          {
            "IpProtocol": "tcp",
            "CidrIp": "0.0.0.0/0",
            "FromPort": "22",
            "ToPort": "22"
          }
        ]
      }
    },
    "SGBaseIngress": {
      "Type": "AWS::EC2::SecurityGroupIngress",
      "Properties": {
        "GroupId": { "Ref": "SGBase" },
        "IpProtocol": "tcp",
        "FromPort": "80",
        "ToPort": "80",
        "SourceSecurityGroupId": { "Ref": "SGBase" }
      }
    }
  }
}
```

AWS::EC2::SecurityGroupEgress

The AWS::EC2::SecurityGroupEgress type adds an egress rule to an Amazon VPC security group.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For more information about adding egress rules to Amazon VPC security groups, go to AuthorizeSecurityGroupEgress in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "GroupId" : String,
   "IpProtocol" : String,
   "CidrIp" : String,
   "DestinationSecurityGroupId" : String,
   "FromPort" : Number,
   "ToPort" : Number
}
```

Properties

GroupId

ID of the Amazon VPC security group to modify. This value can be a reference to an AWS::EC2::SecurityGroup (p. 237) resource that has a valid VpcId property or the ID of an existing Amazon VPC security group.

Type: String

Required: Yes

Update requires: no interruption (p. 33)

IpProtocol

IP protocol name or number. For valid values, see the IpProtocol parameter in AuthorizeSecurityGroupIngress

Type: String

Required: Yes

Update requires: no interruption (p. 33)

CidrIp

CIDR range.

Type: String

Required: Conditional—cannot be used when specifying a destination security group.

Update requires: no interruption (p. 33)

DestinationSecurityGroupId

Specifies the GroupId of the destination Amazon VPC security group.

Type: String

Required: Conditional—cannot be used when specifying a CIDR IP address.

Update requires: no interruption (p. 33)

FromPort

Start of port range for the TCP and UDP protocols, or an ICMP type number. An ICMP type number of -1 indicates a wildcard (i.e., any ICMP type number).

Type: String

Required: Yes

Update requires: no interruption (p. 33)

ToPort

End of port range for the TCP and UDP protocols, or an ICMP code. An ICMP code of -1 indicates a wildcard (i.e., any ICMP code).

Type: String

Required: Yes

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```

"SecurityGroupEgress" : [{
   "IpProtocol" : "tcp",
   "FromPort" : "80",
   "ToPort" : "80",
   "CidrIp" : "0.0.0.0/0"
}]
```

AWS::EC2::Subnet

Creates a subnet in an existing VPC.

For more information, go to CreateSubnet in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "Type" : "AWS::EC2::Subnet",
   "Properties" : {
      "VpcId" : { "Ref" : String },
      "CidrBlock" : String,
      "AvailabilityZone" : String,
      "Tags" : [ EC2 Tag, ... ]
   }
}
```

Properties

VpcId

A Ref structure containing the ID of the VPC where you want to create the subnet. The VPC ID is provided as the value of the "Ref" property, as: { "Ref": "VPCID" }.

Required: Yes

Type: Ref ID

CidrBlock

The CIDR block you want the subnet to cover (for example, `"10.0.0.0/24"`).

Required: Yes

Type: String

AvailabilityZone

The Availability Zone you want the subnet in. Default: AWS selects a zone for you (recommended)

Required: No

Type: String

Tags

The tags you want to attach to this resource.

For more information about using tags in EC2, see: Using Tags in the *Amazon Elastic Compute Cloud User Guide*.

Required: No

Type: List of EC2 Tags (p. 310).

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

The following example snippet uses the VPC ID from a VPC named *myVPC* that was declared elsewhere in the same template.

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "mySubnet" : {
         "Type" : "AWS::EC2::Subnet",
         "Properties" : {
            "VpcId" : { "Ref" : "myVPC" },
            "CidrBlock" : "10.0.0.0/24",
            "AvailabilityZone" : "us-east-1a",
            "Tags" : [ { "Key" : "foo", "Value" : "bar" } ]
         }
      }
   }
}
```

AWS::EC2::SubnetNetworkAclAssociation

Associates a subnet with a network ACL.

For more information, go to ReplaceNetworkAclAssociation in the *Amazon Elastic Compute Cloud API Reference*.

Note

The EC2 API Reference refers to the *SubnetId* parameter as the *AssociationId*.

Syntax

```
"Type" : "AWS::EC2::SubnetNetworkAclAssociation",
"Properties" : {
   "SubnetId (p. 245)" : { String }
   "NetworkAclId (p. 245)" : { String }
}
```

Properties

SubnetId
The ID representing the current association between the original network ACL and the subnet.
Required: Yes
Type: String

NetworkAclId
The ID of the new ACL to associate with the subnet.
Required: Yes
Type: String

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

AssociationId
Returns the value of this object's SubnetId (p. 245) property.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "mySubnetNetworkAclAssociation" : {
         "Type" : "AWS::EC2::SubnetNetworkAclAssociation",
         "Properties" : {
            "SubnetId" : { "Ref" : "mySubnet" },
            "NetworkAclId" : { "Ref" : "myNetworkAcl" },
         }
      }
   }
}
```

AWS::EC2::SubnetRouteTableAssociation

Associates a subnet with a route table.

For more information, go to AssociateRouteTable in the *Amazon Elastic Compute Cloud API Reference*.

The following properties are available with the AWS::EC2::SubnetRouteTableAssociation type.

Property	Type	Required	Notes
SubnetId	String	Yes	The ID of the subnet.
RouteTableId	String	Yes	The ID of the route table.

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "mySubnetRouteTableAssociation" : {
             "Type" : "AWS::EC2::SubnetRouteTableAssociation",
              "Properties" : {
                  "SubnetId" : {"Ref" : "mySubnet"}
                  "RouteTableId" : {"Ref" : "myRouteTable"}
              }
        }
    }
}
```

AWS::EC2::Volume

The AWS::EC2::Volume type creates a new Elastic Block Store volume.

You can set a deletion policy for your volume to control how AWS CloudFormation handles the volume when the stack is deleted. For Elastic Block Store volumes, you can choose to retain the volume, to delete the volume, or to create a snapshot of the volume. For more information, see DeletionPolicy Attribute (p. 322).

Note

If you set a deletion policy that creates a snapshot, all tags on the volume are included in the snapshot.

When you specify an AWS::EC2::Volume type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`VolumeId`*.

The following properties are available with the AWS::EC2::Volume type.

Property	Type	Required	Notes
AvailabilityZone	String	Yes	The Availability Zone in which to create the new volume.
Size	String	Conditional	The size of the volume, in GiBs. If you specify Size, do not specify SnapshotId.
SnapshotId	String	Conditional	The snapshot from which to create the new volume. If you specify SnapshotId, do not specify Size.
Tags	List of EC2 Tags (p. 310) type	No	The tags you want to attach to the volume.

AWS::EC2::VolumeAttachment

The following properties are available with the AWS::EC2::VolumeAttachment type.

Property	Type	Required	Notes
InstanceId	String	Yes	The ID of the instance to which the volume attaches. This value can be a reference to an AWS::EC2::Instance (p. 222) resource, or it can be the instance ID of an existing EC2 instance.
VolumeId	String	Yes	The ID of the Amazon EBS volume. The volume and instance must be within the same Availability Zone. This value can be a reference to an AWS::EC2::Volume (p. 247) resource, or it can be the volume ID of an existing Amazon EBS volume.
Device	String	Yes	How the device is exposed to the instance (e.g., /dev/sdh, or xvdh).

AWS::EC2::VPC

Creates a Virtual Private Cloud (VPC) with the CIDR block that you specify.

For more information about VPCs on EC2 instances, go to CreateVpc in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "Type" : "AWS::EC2::VPC",
   "Properties" : {
      "CidrBlock (p. 248)" : String,
      "InstanceTenancy (p. 248)" : String,
      "Tags (p. 248)" : [{EC2 Tag}, ...]
   }
}
```

Properties

CidrBlock

The CIDR block you want the VPC to cover. For example: `"10.0.0.0/16"`.

Type: String

Required: Yes

InstanceTenancy

The allowed tenancy of instances launched into the VPC. A value of `"default"` means instances can be launched with any tenancy; a value of `"dedicated"` means instances must be launched with tenancy as dedicated.

Type: String

Required: No

Valid values: `"default"` or `"dedicated"`

Tags

The tags you want to attach to the instance.

Type: List of EC2 Tags (p. 310).

Required: No

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Example

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myVPC" : {
         "Type" : "AWS::EC2::VPC",
         "Properties" : {
            "CidrBlock" : "10.0.0.0/16",
            "InstanceTenancy" :"dedicated",
            "Tags" : [ {"Key" : "foo", "Value" : "bar"}]
         }
      }
   }
}
```

AWS::EC2::VPCDHCPOptionsAssociation

Associates a set of DHCP options (that you've previously created) with the specified VPC.

For more information, go to AssociateDhcpOptions in the *Amazon Elastic Compute Cloud API Reference*.

The following properties are available with the AWS::EC2::VPCDHCPOptionsAssociation type.

Property	Type	Required	Notes
VpcId	String	Yes	The ID of the VPC to associate with this DHCP options set.
DhcpOptionsId	String	Yes	The ID of the DHCP options you want to associate with the VPC, or "default" if you want the VPC to use no DHCP options.

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example Example Template Snippet

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myVPCDHCPOptionsAssociation" : {
         "Type" : "AWS::EC2::VPCDHCPOptionsAssociation",
         "Properties" : {
             "VpcId" : {"Ref" : "myNetworkAcl"},
             "DhcpOptionsId" : {"Ref" : "myDhcpOption"},
         }
      }
   }
}
```

AWS::EC2::VPCGatewayAttachment

Attaches a gateway to a VPC.

For more information, go to AttachVpnGateway in the *Amazon Elastic Compute Cloud API Reference*.

Properties

VpcId
The ID of the VPC to associate with this gateway.

Required: Yes

Type: String

InternetGatewayId
The ID of the Internet gateway.

Required: Conditional. You must specify either InternetGatewayId or VpnGatewayId, but not both.

Type: String

VpnGatewayId
The ID of the virtual private networ (VPN) gateway to attach to the VPC.

Required: Conditional. You must specify either InternetGatewayId or VpnGatewayId, but not both.

Type: String

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Examples

Example Attaching both an Internet gateway and a VPN gateway to a VPC

To attach both an Internet gateway and a VPN gateway to a VPC, you must specify two separate AWS::EC2::VPCGatewayAttachment resources:

```
"AttachGateway" : {
   "Type" : "AWS::EC2::VPCGatewayAttachment",
   "Properties" : {
      "VpcId" : { "Ref" : "VPC" },
      "InternetGatewayId" : { "Ref" : "myInternetGateway" }
   }
},

"AttachVpnGateway" : {
   "Type" : "AWS::EC2::VPCGatewayAttachment",
   "Properties" : {
      "VpcId" : { "Ref" : "VPC" },
      "VpnGatewayId" : { "Ref" : "myVPNGateway" }
   }
},
```

AWS::EC2::VPNConnection

Creates a new VPN connection between an existing virtual private gateway and a VPN customer gateway.

For more information, go to CreateVpnConnection in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "Type" : "AWS::EC2::VPNConnection",
   "Properties" : {
      "Type (p. 251)" : String,
      "CustomerGatewayId (p. 251)" : GatewayID,
      "VpnGatewayId (p. 252)" : GatewayID
   }
}
```

Properties

Type

The type of VPN connection this virtual private gateway supports.

Example: "ipsec.1"

Required: Yes

Type: String

CustomerGatewayId

The ID of the customer gateway. This can either be an embedded JSON object or a reference to a Gateway ID.

Required: Yes

Type: String

VpnGatewayId
The ID of the virtual private gateway. This can either be an embedded JSON object or a reference to a Gateway ID.

Required: Yes

Type: String

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyVPNConnection" }
```

For the VPNConnection with the logical ID "MyVPNConnection", `Ref` will return the VPN connection's resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example VPNConnection

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myVPNConnection" : {
         "Type" : "AWS::EC2::VPNConnection",
         "Properties" : {
            "Type" : "ipsec.1",
            "CustomerGatewayId" : {"Ref" : "myCustomerGateway"},
            "VpnGatewayId" : {"Ref" : "myVPNGateway"}
         }
      }
   }
}
```

AWS::EC2::VPNGateway

Creates a new virtual private gateway. A virtual private gateway is the VPC-side endpoint for your VPN connection.

For more information, go to CreateVpnGateway in the *Amazon Elastic Compute Cloud API Reference*.

Syntax

```
{
   "Type" : "AWS::EC2::VPNGateway",
   "Properties" : {
      "Type (p. 253)" : String
      "AvailabilityZone (p. 253)" : String
      "Tags (p. 253)" : [ Tag1, ... ]
   }
}
```

Properties

Type

The type of VPN connection this virtual private gateway supports.

Example: ipsec.1

Required: Yes

Type: String

AvailabilityZone

The availability zone where you want the virtual private gateway.

Required: No

Type: String

Tags

The tags you want to attach to the instance.

Required: No

Type: List of EC2 Tags (p. 310)
Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyVPNGateway" }
```

For the VPN gateway with the logical ID "MyVPNGateway", `Ref` will return the gateway's resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

Example Create a VPNGateway in AvailabilityZone us-east-1a

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myVPNGateway" : {
         "Type" : "AWS::EC2::VPNGateway",
         "Properties" : {
            "Type" : "ipsec.1",
            "AvailabilityZone" : "us-east-1a"
         }
      }
   }
}
```

AWS::ElastiCache::CacheCluster

The AWS::ElastiCache::CacheCluster type creates an Amazon ElastiCache cache cluster.

For details on the settings for cache clusters, see CreateCacheCluster.

This type supports updates. The update behavior for each property is described in the following table.

Caution

When updates are made to the following properties, AWS CloudFormation creates a replacement cache cluster resource first, changes references from other dependent resources to point to the replacement resource, and then deletes the old resource:

- CacheNodeType
- Engine
- Port
- PreferredAvailabilityZone

For more information about updating other properties on this resource, go to ModifyCacheCluster in the *Amazon ElastiCache API Reference Guide*. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Properties

Property	Type	Required	Notes
AutoMinorVersionUpgrade	Boolean	No	Indicates that minor engine upgrades will be applied automatically to the cache cluster during the maintenance window. Default: `true` *Update requires*: replacement (p. 34)

Property	Type	Required	Notes
CacheNodeType	String	Yes	The compute and memory capacity of nodes in a cache cluster. *Update requires*: replacement (p. 34)
CacheParameterGroupName	String	No	The name of the cache parameter group associated with this cache cluster.
CacheSecurityGroupNames	List of Strings	Yes	A list of cache security group names associated with this cache cluster. *Update requires*: no interruption (p. 33)
Engine	String	Yes	The name of the cache engine to be used for this cache cluster. *Update requires*: replacement (p. 34)
EngineVersion	String	No	The version of the cache engine to be used for this cluster. *Update requires*: no interruption (p. 33)
NotificationTopicArn	String	No	The Amazon Resource Name (ARN) of the Amazon Simple Notification Service (SNS) topic to which notifications will be sent. *Update requires*: no interruption (p. 33)
NumCacheNodes	Integer	Yes	The number of cache nodes the cache cluster should have. *Update requires*: some interruptions (p. 34)
Port	Integer	No	The port number on which each of the cache nodes will accept connections. *Update requires*: no interruption (p. 33)
PreferredAvailabilityZone	String	No	The EC2 Availability Zone that the cache cluster will be created in. *Update requires*: replacement (p. 34)
PreferredMaintenanceWindow	String	No	The weekly time range (in UTC) during which system maintenance can occur. *Update requires*: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::ElastiCache::ParameterGroup

The AWS::ElastiCache::ParameterGroup type creates a new cache parameter group. Cache parameter groups control the parameters for a cache cluster.

For more information on the settings for Cache Parameter Groups, go to CreateCacheParameterGroup in the *Amazon ElastiCache API Reference Guide*.

For more information on updating other properties on this resource, go to ModifyCacheParameterGroup in the *Amazon ElastiCache API Reference Guide*. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Properties

Property	Type	Required	Notes
CacheParameterGroupFamily	String	Yes	The name of the cache parameter group family that the cache parameter group can be used with.
Description	String	Yes	The description for the Cache Parameter Group.
Properties	CommaDelimitedList	No	A comma-delimited list of parameter name/value pairs. For more information, go to ModifyCacheParameterGroup in the *Amazon ElastiCache API Reference Guide*.

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::ElastiCache::SecurityGroup

The AWS::ElastiCache::SecurityGroup type creates a cache security group. For more information about cache security groups, go to Cache Security Groups in the *Amazon ElastiCache User Guide* or go to CreateCacheSecurityGroup in the *Amazon ElastiCache API Reference Guide*.

When you specify an AWS::ElastiCache::SecurityGroup type as an argument to the `Ref` function, AWS CloudFormation returns the *`CacheSecurityGroupName`* property of the new cache security group.

Property	Type	Required	Notes
Description	String	Yes	The description property of the new cache security group.

AWS::ElastiCache::SecurityGroupIngress

The AWS::ElastiCache::SecurityGroupIngress type authorizes ingress to a cache security group from hosts in specified EC2 security groups. For more information about ElastiCache security group ingress, go to AuthorizeCacheSecurityGroupIngress in the *Amazon ElastiCache API Reference Guide*.

When you specify an AWS::ElastiCache::SecurityGroup type as an argument to the `Ref` function, AWS CloudFormation returns the value of the `CacheSecurityGroupName` property.

The following properties are available with the ElastiCache Security Group Ingress type.

Property	Type	Required	Notes
CacheSecurityGroupName	String	Yes	The name of the Cache Security Group to authorize.
EC2SecurityGroupName	String	Yes	Name of the EC2 Security Group to include in the authorization.
EC2SecurityGroupOwnerId	String	No	Specifies the AWS Account ID of the owner of the EC2 security group specified in the EC2SecurityGroupName property. The AWS Access Key ID is not an acceptable value.

AWS::ElasticBeanstalk::Application

When you specify an AWS::ElasticBeanstalk::Application type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`ApplicationName`*.

The following properties are available with the AWS::ElasticBeanstalk::Application type.

Property	Type	Required	Notes
ApplicationVersions	List of ApplicationVersion (p. 310) types	Yes	Application versions associated with this application. An application version is a specific, labeled iteration of deployable code.
ConfigurationTemplates	List of ConfigurationTemplate (p. 311)	No	Configuration templates associated with this application. You can use templates to deploy different versions of an application using the configuration settings defined in the template.

AWS::ElasticBeanstalk::Environment

Properties

Property	Type	Required	Notes
ApplicationName	String	Yes	The name of the application associated with this environment.
CNAMEPrefix	String	No	The URL to the CNAME for this environment.

Property	Type	Required	Notes
OptionSettings	An array of OptionSettings (p. 312).	No	The option settings to add.
OptionsToRemove	An array of OptionSettings (p. 312).	No	The option settings to remove.
SolutionStackName	String	No	Stack name associated with the environment.
TemplateName	String	No	Name of the template to use with the environment.
VersionLabel	String	No	Version to associate with the environment

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Return Values

Function	Type	Description
Ref (p. 326)	Name	mystack-myenv-LKGNQSFHO1DB
Fn::GetAtt (p. 324)	EndpointURL	The URL to the LoadBalancer for this environment. Example: awseb-myst-myen-132MQC4KRLAMD-1371280482.us-east-1.elb.amazonaws.com

AWS::ElasticLoadBalancing::LoadBalancer

The AWS::ElasticLoadBalancing::LoadBalancer type creates a LoadBalancer.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Important

If you update the property values for a listener specified by the Listeners (p. 260) property, AWS CloudFormation will delete the existing listener and create a new one with the updated properties. During the time that AWS CloudFormation is performing this action, clients will not be able to connect to the load balancer.

Syntax

```
{
   "Type": "AWS::ElasticLoadBalancing::LoadBalancer",
   "Properties": {
      "AppCookieStickinessPolicy (p. 259)" : [ AppCookieStickinessPolicy, ...
],
      "AvailabilityZones (p. 259)" : [ String, ... ],
      "HealthCheck (p. 259)" : HealthCheck,
      "Instances (p. 259)" : [ String, ... ],
      "LBCookieStickinessPolicy (p. 259)" : [ LBCookieStickinessPolicy, ... ],
      "Listeners (p. 260)" : [ Listener, ... ],
      "Scheme (p. 260)" : String,
      "SecurityGroups (p. 260)" : [ Security Group, ... ],
      "Subnets (p. 260)" : [ String, ... ]
   }
}
```

Properties

AppCookieStickinessPolicy

Generates one or more stickiness policies with sticky session lifetimes that follow that of an application-generated cookie. These policies can be associated only with HTTP/HTTPS listeners.

Required: No

Type: A list of AppCookieStickinessPolicy (p. 313) objects.

AvailabilityZones

The Availability Zones in which to create the load balancer. You can specify *either* AvailabilityZones or Subnets, but not both.

Required: No

Type: List of Strings

HealthCheck

When specified, declares an application health check for the instances.

Required: No

Type: ElasticLoadBalancing HealthCheck Type (p. 314).

Instances

Provides a list of EC2 instance IDs for the LoadBalancer.

Required: No

Type: List of Strings

LBCookieStickinessPolicy

Generates a stickiness policy with sticky session lifetimes controlled by the lifetime of the browser (user-agent), or by a specified expiration period. This policy can be associated only with HTTP/HTTPS listeners.

Required: No

Type: A list of LBCookieStickinessPolicy (p. 315) objects.

Listeners

One or more listeners for this load balancer. Each listener must be registered for a specific port, and you can not have more than one listener for a given port.

Required: No

Type: A list of ElasticLoadBalancing Listener Property Type (p. 316) objects.

Scheme

For LoadBalancers attached to an Apazon VPC, this parameter can be used to specify the type of LoadBalancer to use. Specify **`"internal"`** to create an internal LoadBalancer with a DNS name that resolves to private IP addresses.

Required: No

Type: String

SecurityGroups

Required: No

Type: A list of security groups assigned to your load balancer within your virtual private cloud (VPC).

Update requires: no interruption (p. 33)

Subnets

A list of subnet IDs in your virtual private cloud (VPC) to attach to your load balancer. You can specify *either* AvailabilityZones or Subnets, but not both.

Required: No

Type: List of Strings

For more information about using Elastic Load Balancing in a VPC, see How Do I Use Elastic Load Balancing in Amazon VPC in the *Elastic Load Balancing Developer Guide*.

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example, `mystack-myelb-1WQN7BJGDB5YQ`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

CanonicalHostedZoneName

The name of the Route 53 hosted zone that is associated with the LoadBalancer.

Example: `mystack-myelb-15HMABG9ZCN57-1013119603.us-east-1.elb.amazonaws.com`

CanonicalHostedZoneNameID

The ID of the Route 53 hosted zone name that is associated with the LoadBalancer.

Example: `Z3DZXE0Q79N41H`

DNSName

The DNS name for the LoadBalancer.

Example: `mystack-myelb-15HMABG9ZCN57-1013119603.us-east-1.elb.amazonaws.com`

SourceSecurityGroup.GroupName
The security group that you can use as part of your inbound rules for your LoadBalancer's back-end Amazon EC2 application instances.

Example: `amazon-elb`

SourceSecurityGroup.OwnerAlias
Owner of the source security group.

Example: `amazon-elb-sg`

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Examples

Simple Load Balancer with Health Check

```
"ElasticLoadBalancer" : {
   "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
   "Properties" : {
      "AvailabilityZones" : { "Fn::GetAZs" : "" },
      "Instances" : [ { "Ref" : "Ec2Instance1" },{ "Ref" : "Ec2Instance2" } ],

      "Listeners" : [ {
         "LoadBalancerPort" : "80",
         "InstancePort" : { "Ref" : "WebServerPort" },
         "Protocol" : "HTTP"
      } ],
      "HealthCheck" : {
         "Target" : {
            "Fn::Join" : [ "", [ "HTTP:", { "Ref" : "WebServerPort" }, "/" ] ]

         },
         "HealthyThreshold" : "3",
         "UnhealthyThreshold" : "5",
         "Interval" : "30",
         "Timeout" : "5"
      }
   }
}
```

More examples

Examples of AWS CloudFormation templates can be viewed and downloaded from the AWS CloudFormation Sample Templates. These include:

- ELBSample.template: Elastic Load Balancer with a health check
- ELBStickinessSample.template: Elastic Load Balancer example configured with cookie-based stickiness
- ELBWithLockedDownEC2Instances.template: Elastic Load Balancer with instances that are locked down to only receive traffic from the load balancer
- ELBWithLockedDownAutoScaledInstances.template: Elastic Load Balancer with and Auto Scaling group that is locked down to only receive traffic from the load balancer
- ELBZoneApex.template: Map an Elastic Load Balancer to a DNS Zone apex

See Also

- CreateLoadBalancer in the *Elastic Load Balancing API Reference*

AWS::IAM::AccessKey

The AWS::IAM::AccessKey resource type generates a secret access key and assigns it to an IAM user or AWS account.

This type supports updates. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::IAM::AccessKey",
   "Properties": {
      "Serial (p. 262)": Integer,
      "Status (p. 262)": String,
      "UserName (p. 262)": String
   }
}
```

Properties

Serial

This value is specific to AWS CloudFormation and can only be *incremented*. Incrementing this value notifies AWS CloudFormation that you want to rotate your access key. When you update your stack, AWS CloudFormation will replace the existing access key with a new key.

Required: No

Type: Integer

Update requires: replacement (p. 34)

Status

The status of the access key.

Required: Yes

Type: String

Valid values: `"Active" or "Inactive"`

Update requires: no interruption (p. 33)

UserName

THe name of the user that the new key will belong to.

Required: Yes

Type: String

Update requires: replacement (p. 34)

Return Values

Ref

Specifying this resource ID to the intrinsic `Ref` function will return the *`AccessKeyId`*. For example: `AKIAIOSFODNN7EXAMPLE`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

SecretAccessKey
Returns the secret access key for the specified AWS::IAM::AccessKey resource. For example: `wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY`.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

To view AWS::IAM::AccessKey snippets, see Declaring an IAM Access Key Resource (p. 123).

AWS::IAM::Group

The AWS::IAM::Group type creates an Identity and Access Management (IAM) group.

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::IAM::Group",
   "Properties": {
      "Path (p. 262)": String,
      "Policies (p. 262)": [ Policy1, ... ]
   }
}
```

Properties

Path
The path to the group. For more information about paths, see Identifiers for IAM Entities in *Using IAM*.

Required: No

Type: String

Update requires: no interruption (p. 33)

Policies
The policies to be added to the group. For information about policies, see Overview of Policies in *Using IAM*.

Required: No

Type: List of AWS::IAM::Policy (p. 266) types

Update requires: no interruption (p. 33)

Return Values

Ref

Specifying this resource ID to the intrinsic `Ref` function will return the GroupName. For example: `mystack-mygroup-1DZETITOWEKVO`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Arn

Returns the Amazon Resource Name (ARN) for the AWS::IAM::Group resource. For example: `arn:aws:iam::123456789012:group/mystack-mygroup-1DZETITOWEKVO`.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

To view AWS::IAM::Group snippets, see Declaring an IAM Group Resource (p. 124)

AWS::IAM::InstanceProfile

Creates an AWS Identity and Access Management (IAM) Instance Profile that can be used with IAM Roles for EC2 Instances.

For detailed information about IAM Roles, see Working with Roles in the *AWS Identity and Access Management User Guide*.

Syntax

```
{
   "Type": "AWS::IAM::InstanceProfile",
   "Properties": {
      "Path (p. 264)": String,
      "Roles (p. 265)": [ IAM Roles ]
   }
}
```

Properties

Path

The path associated with this IAM instance profile. For information about IAM paths, see Friendly Names and Paths in the *AWS Identity and Access Management User Guide*.

Required: Yes

Type: String

Update requires: replacement (p. 34)

Roles

The roles associated with this IAM instance profile.

Required: Yes

Type: List of references to AWS::IAM::Roles. Currently, a maximum of one role can be assigned to an instance profile.

Update requires: no interruption (p. 33)

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyProfile" }
```

For the IAM::InstanceProfile with the logical ID "MyProfile", `Ref` will return the resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Arn

Returns the Amazon Resource Name (ARN) for the instance profile. For example:

```
{"Fn::GetAtt" : ["MyProfile", "Arn"] }
```

This will return a value such as `“arn:aws:iam::1234567890:instance-profile/MyProfile-ASDNSDLKJ”`.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

Example IAM Role with Embedded Policy and Instance Profiles

This example shows an embedded Policy in the IAM::Role. The policy is specified inline in the IAM::Role Policies property.

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/",
            "Policies": [ {
               "PolicyName": "root",
               "PolicyDocument": {
                  "Statement": [ {
                     "Effect": "Allow",
                     "Action": "*",
                     "Resource": "*"
                  } ]
               }
               } ]
            }
         },
         "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "RootRole"
            } ]
         }
      }
   }
}
```

AWS::IAM::Policy

The AWS::IAM::Policy type applies an Identity and Access Management (IAM) policy to users or groups. For more information about IAM policies, see Overview of Policies" in the *AWS Identity and Access Management User Guide*.

This type supports updates. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::IAM::Policy",
   "Properties": {
      "PolicyName (p. 267)": String,
      "PolicyDocument (p. 267)": JSON,
      "Groups (p. 267)": [ Group1, ... ],
      "Users (p. 267)": [ User1, ... ],
   }
}
```

Properties

PolicyName

The name of the policy.

Required: Yes

Type: String

Update requires: no interruption (p. 33)

PolicyDocument

A policy document containing permissions to add to the specified users or groups.

Required: Yes

Type: JSON

Update requires: no interruption (p. 33)

Groups

The names of groups to which you want to add the policy.

Required: Conditional

Type: List of groups

Update requires: no interruption (p. 33)

Users

The names of users for whom you want to add the policy.

Required: Conditional

Type: List of users

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

To view AWS::IAM::Policy snippets, see Declaring an IAM Policy (p. 125).

AWS::IAM::Role

Creates an AWS Identity and Access Management (IAM) Role for EC2 Instances. An IAM Role can be used to enable applications running on an Amazon EC2 instance to securely access your AWS resources.

For detailed information about IAM Roles, see Working with Roles in the *AWS Identity and Access Management User Guide*.

Syntax

```
{
   "Type": "AWS::IAM::Role",
   "Properties": {
      "AssumeRolePolicyDocument (p. 268)": { JSON },
      "Path (p. 268)": String,
      "Policies (p. 268)": [ Embedded IAM Policies ]
   }
}
```

Properties

AssumeRolePolicyDocument
The IAM Assume Role Policy Document associated with this role.

Required: Yes

Type: A JSON Policy Document.

Update requires: no interruption (p. 33)

Note

There can be only one Assume Role Policy document associated with a role, and currently the only policy supported allows an EC2 instance to assume the role. For an example of the allowed policy document, see the Template Examples (p. 269) section.

Path
The path associated with this role. For information about IAM paths, see Friendly Names and Paths in the *AWS Identity and Access Management User Guide*.

Required: Yes

Type: String

Update requires: replacement (p. 34)

Policies
A list of embedded policies to associate with this role. Policies can also be specified externally. For sample templates demonstrating both embedded and external policies, see Template Examples (p. 269).

Required: No

Type: A list of embedded IAM Policies.

Update requires: no interruption (p. 33)

Notes on policies for IAM Roles

For general information about IAM Policies and Policy documents, see How to Write a Policy in the *AWS Identity and Access Management User Guide*.

Return Values

Ref

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "RootRole" }
```

For the IAM::Role with the logical ID "RootRole", `Ref` will return the resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Arn

Returns the Amazon Resource Name (ARN) for the instance profile. For example:

```
{"Fn::GetAtt" : ["MyRole", "Arn"] }
```

This will return a value such as “`arn:aws:iam::1234567890:role/MyRole-AJJHDSKSDF`”.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

Note

- More extensive template examples for IAM Roles can be found in Identity and Access Management (IAM) Template Snippets (p. 122).
- The template snippets in this section also use IAM *instance profiles*. For more information, see . (p. 264)

Example IAM Role with Embedded Policy and Instance Profiles

This example shows an embedded Policy in the IAM::Role. The policy is specified inline in the IAM::Role Policies property.

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/",
            "Policies": [ {
               "PolicyName": "root",
               "PolicyDocument": {
                  "Statement": [ {
                     "Effect": "Allow",
                     "Action": "*",
                     "Resource": "*"
                  } ]
               }
               } ]
            }
         },
         "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "RootRole"
            } ]
         }
      }
   }
}
```

Example IAM Role with External Policy and Instance Profiles

In this example, the Policy and InstanceProfile resources are specified externally to the IAM Role. They refer to the role by specifying its name, "RootRole", in their respective Roles properties.

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Resources": {
      "RootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "root",
            "PolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
               } ]
            },
            "Roles": [ {
               "Ref": "RootRole"
            } ]
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "RootRole"
            } ]
         }
      }
   }
}
```

AWS::IAM::User

The AWS::IAM::User type creates a user.

This type supports updates. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::IAM::User",
   "Properties": {
      "Path (p. 272)": String,
      "Groups (p. 272)": [ Group, ... ],
      "LoginProfile (p. 272)": { "Password" : String },
      "Policies (p. 272)": [ Embedded IAM Policies ]
   }
}
```

Properties

Path

The path for the user name. For more information about paths, see Identifiers for IAM Entities in Using AWS Identity and Access Management.

Required: No

Type: String

Update requires: no interruption (p. 33)

Groups

The names of groups to which you want to add the user.

Required: No

Type: List of Groups

Update requires: no interruption (p. 33)

LoginProfile

Creates a login profile for the user so the user can access AWS services such as the AWS Management Console.

The LoginProfile type is an embedded property in the AWS::IAM::User type. The LoginProfile property contains a single field: *`Password`*, which takes a string as its value. For example:

```
"LoginProfile": { "Password": "myP@ssW0rd" }
```

Required: No

Type: LoginProfile type

Update requires: no interruption (p. 33)

Policies

Applies specified policies to the user. For information about policies, see Overview of Policies in [Using IAM].

Required: No

Type: List of AWS::IAM::Policy (p. 266) types

Update requires: no interruption (p. 33)

Return Values

Ref

Specifying this resource ID to the intrinsic Ref function will return the UserName. For example: `mystack-myuser-1CCXAFG2H2U4D`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Arn
Returns the Amazon Resource Name (ARN) for the specified AWS::IAM::User resource. For example: `arn:aws:iam::123456789012:user/mystack-myuser-1CCXAFG2H2U4D`.

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

To view AWS::IAM::User snippets, see: Declaring an IAM User Resource (p. 122)

AWS::IAM::UserToGroupAddition

The AWS::IAM::UserToGroupAddition type adds Identity and Access Management (IAM) users to a group.

This type supports updates. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
{
   "Type": "AWS::IAM::UserToGroupAddition",
   "Properties": {
      "GroupName (p. 273)": String,
      "Users (p. 273)": [ User1, ... ]
   }
}
```

Properties

GroupName
The name of group to add users to.

Required: Yes

Type: String

Update requires: no interruption (p. 33)

Users
Required: Yes

Type: List of users

Update requires: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyUserToGroupAddition" }
```

For the AWS::IAM::UserToGroupAddition with the logical ID "MyUserToGroupAddition", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

To view AWS::IAM::UserToGroupAddition snippets, see Adding Users to a Group (p. 125).

AWS::RDS::DBInstance

The AWS::RDS::DBInstance type creates an Amazon RDS database instance. For detailed information about configuring RDS DB instances, see CreateDBInstance.

Syntax

```
{
   "Type" : "AWS::RDS::DBInstance",
   "Properties" :
   {
      "DBSnapshotIdentifier (p. 275)" : String,
      "AllocatedStorage (p. 275)" : String,
      "AvailabilityZone (p. 275)" : String,
      "BackupRetentionPeriod (p. 275)" : String,
      "DBInstanceClass (p. 275)" : String,
      "DBName (p. 275)" : String,
      "DBParameterGroupName (p. 276)" : String,
      "DBSecurityGroups (p. 276)" : [ String or Reference, ... ],
      "DBSubnetGroupName (p. 276)" : String,
      "Engine (p. 276)" : String,
      "EngineVersion (p. 276)" : String,
      "LicenseModel (p. 276)" : String,
      "MasterUsername (p. 276)" : String,
      "MasterUserPassword (p. 277)" : String,
      "Port (p. 277)" : String,
      "PreferredBackupWindow (p. 277)" : String,
      "PreferredMaintenanceWindow (p. 277)" : String,
      "MultiAZ (p. 277)" : Boolean
   }
}
```

Properties

DBSnapshotIdentifier
The identifier for the DB snapshot to restore from.

By specifying this property, you can create a DB instance from the specified DB snapshot. If the DBSnapshotIdentifier property is an empty string or the AWS::RDS::DBInstance declaration has no DBSnapshotIdentifier property, the database is created as a new database. If the property contains a value (other than empty string), AWS CloudFormation creates a database from the specified snapshot. If no snapshot exists with the specified name, the database creation fails and the stack rolls back.

Required: No

Type: String

Update requires: no interruption (p. 33)

AllocatedStorage
The allocated storage size specified in gigabytes.

Required: Yes

Type: String

Update requires: no interruption (p. 33)

AvailabilityZone
The name of the Availability Zone where the DB instance is. You cannot set the `AvailabilityZone` parameter if the `MultiAZ` parameter is set to true.

Required: No

Type: String

Update requires: replacement (p. 34)

BackupRetentionPeriod
The number of days for which automatic DB snapshots are retained.

Required: No

Type: String

Update requires: no interruption (p. 33)

DBInstanceClass
The name of the compute and memory capacity class of the DB instance.

Required: Yes

Type: String

Update requires: no interruption (p. 33)

DBName
The name of the initial database of this instance that was provided at create time, if one was specified when the DB instance was created. This same name is returned for the life of the DB instance.

Required: No

Type: String

Update requires: replacement (p. 34)

DBParameterGroupName
The name of the DB parameter group.

Required: No

Type: String

Update requires: no interruption (p. 33)

DBSecurityGroups
A list containing the DB security groups to assign to the Amazon RDS instance. The list can contain both the name of existing DB security groups or references to AWS::RDS::DBSecurityGroup resources created in the template.

Required: No

Type: list of Strings or References

Update requires: no interruption (p. 33)

DBSubnetGroupName
A DB Subnet Group to associate with this DB instance.

If there is no DB Subnet Group, then it is a non-VPC DB instance.

For more information about using Amazon RDS in a VPC, go to Using Amazon RDS with Amazon Virtual Private Cloud (VPC) in the *Amazon Relational Database Service Developer Guide*.

Required: No

Type: String

Update requires: replacement (p. 34)

Engine
The name of the database engine to be used for this DB instance.

Required: Yes

Type: String

Update requires: replacement (p. 34)

EngineVersion
The version number of the database engine to use.

Required: No

Type: String

Update requires: no interruption (p. 33)

LicenseModel
The license model information for this DB instance.

Required: No

Type: String

Update requires: replacement (p. 34)

MasterUsername
The master username for the DB instance.

Required: Yes

Type: String

Update requires: replacement (p. 34)

MasterUserPassword

The master password for the DB instance.

Required: Yes

Type: String

Update requires: no interruption (p. 33)

Port

The port for the instance.

Required: No

Type: String

Update requires: replacement (p. 34)

PreferredBackupWindow

The daily time range during which automated backups are created if automated backups are enabled, as determined by the BackupRetentionPeriod.

Required: No

Type: String

Update requires: no interruption (p. 33)

PreferredMaintenanceWindow

The weekly time range (in UTC) during which system maintenance can occur.

Required: No

Type: String

Update requires: no interruption (p. 33)

MultiAZ

Specifies if the DB instance is a multiple availability-zone deployment. You cannot set the *`AvailabilityZone`* parameter if the *`MultiAZ`* parameter is set to true.

Required: No

Type: Boolean

Update requires: no interruption (p. 33)

Updating and Deleting AWS:RDS::DBInstances

Caution

When updates are made to properties labeled "*Update requires*: replacement (p. 34)", AWS CloudFormation first creates a replacement DB Instance resource, then changes references from other dependent resources to point to the replacement resource and, finally, deletes the old resource.

If you do not take a snapshot of the database before updating the stack, you will lose the data when your DB instance is replaced. To preserve your data, you should take the following precautions:

1. Quiesce any applications that are using the DB instance so that there is no activity against the DB instance.
2. Create a snapshot of the DB instance. For more information about creating DB snapshots, see Creating a DB snapshot.
3. If you want to restore your instance using a DB snapshot, modify the update template with your DB instance changes and add the DBSnapshotIdentifier property with the ID of the DB snapshot you want to use.
4. Update the stack.

For more information about updating other properties on this resource, see ModifyDBInstance. For more information about updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

You can set a deletion policy for your DB instance to control how AWS CloudFormation handles the instance when the stack is deleted. For Amazon RDS DB instances, you can choose to *retain* the instance, to *delete* the instance, or to *create a snapshot* of the instance. For more information, see DeletionPolicy Attribute (p. 322).

Return Values

Ref

When you provide the RDS DB instance's logical name to the `Ref` intrinsic function, `Ref` will return the DBInstanceIdentifier. For example: `mystack-mydb-ea5ugmfvuaxg`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

- **Endpoint.Address**

  The connection endpoint for the database. For example: `mystack-mydb-1apw1j4phylrk.cg034hpkmmjt.us-east-1.rds.amazonaws.com`.
- **Endpoint.Port**

  The port number on which the database accepts connections. For example: `3306`

For more information about using `Fn:GetAtt`, see Fn::GetAtt (p. 324).

Template Examples

Example Amazon RDS DB instance snippets

To view AWS::RDS::DBInstance template snippets, see Amazon RDS Template Snippets (p. 134).

AWS::RDS::DBSubnetGroup

The AWS::RDS::DBSubnetGroup type creates DB subnet group. DB subnet groups must contain at least one subnet in each AZ in the region.

For details on the settings for DB Instances, see CreateDBSubnetGroup.

This type supports updates.

For more information about updating other properties on this resource, see ModifyDBSubnetGroup. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

Properties

Property	Type	Required	Notes
DBSubnetGroupDescription	String	Yes	The description for the DB Subnet Group. *Update requires*: no interruption (p. 33)
SubnetIds	String list	Yes	The EC2 Subnet IDs for the DB Subnet Group. *Update requires*: no interruption (p. 33)

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Example

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myDBSubnetGroup" : {
             "Type" : "AWS::RDS::DBSubnetGroup",
              "Properties" : {
                  "DBSubnetGroupDescription" : "description",
                  "SubnetIds" :["subnet-7b5b4112", "subnet-7b5b4115"]
              }
        }
    }
}
```

AWS::RDS::DBSecurityGroup

The AWS::RDS::DBSecurityGroup type is used to create or update an Amazon RDS DB Security Group. For more information about DB Security Groups, see Working with DB Security Groups in the *Amazon Relational Database Service Developer Guide*.

For details on the settings for DB security groups, see CreateDBSecurityGroup.

When you specify an AWS::RDS::DBSecurityGroup as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`DBSecurityGroupName`*.

Syntax

```
{
   "Type" : "AWS::RDS::DBSecurityGroup",
   "Properties" :
   {
      "Ec2VpcId (p. 280)" : { "Ref" : "myVPC" },
     "DBSecurityGroupIngress (p. 280)" : [ RDS Security Group Rule (p. 319) object
 1, ... ],
      "GroupDescription (p. 280)" : String,
   }
}
```

Properties

EC2VpcId

The Id of VPC. Indicates which VPC this DB Security Group should belong to.

Type: String

Required: Conditional. Must be specified to create a DB Security Group for a VPC; may not be specified otherwise.

Update requires: replacement (p. 34)

DBSecurityGroupIngress

Network ingress authorization for an Amazon EC2 security group or an IP address range.

Type: List of RDS Security Group Rules (p. 319).

Required: Yes

Update requires: no interruption (p. 33)

GroupDescription

Description of the security group.

Type: String

Required: Yes

Update requires: replacement (p. 34)

Template Examples

Tip

For more RDS template examples, see Amazon RDS Template Snippets (p. 134).

Single VPC security group

This template snippet creates/updates a single VPC security group, referred to by EC2SecurityGroupName.

```
"DBSecurityGroup": {
   "Type": "AWS::RDS::DBSecurityGroup",
   "Properties": {
      "EC2VpcId" : { "Ref" : "VpcId" },
      "DBSecurityGroupIngress": {
         ["EC2SecurityGroupName": { "Ref": "WebServerSecurityGroup"}]
      },
      "GroupDescription": "Frontend Access"
   }
},
```

Multiple VPC security groups

This template snippet creates/updates multiple VPC security groups.

```
{
   "Resources" : {
      "DBinstance" : {
         "Type" : "AWS::RDS::DBInstance",
         "Properties" : {
            "DBSecurityGroups" : [ {"Ref" : "DbSecurityByEC2SecurityGroup"} ],

            "AllocatedStorage" : "5",
            "DBInstanceClass" : "db.m1.small",
            "Engine" : "MySQL",
            "MasterUsername" : "YourName",
            "MasterUserPassword" : "YourPassword"
         },
         "DeletionPolicy" : "Snapshot"
      },
      "DbSecurityByEC2SecurityGroup" : {
         "Type" : "AWS::RDS::DBSecurityGroup",
         "Properties" : {
            "GroupDescription" : "Ingress for Amazon EC2 security group",
            "DBSecurityGroupIngress" : [ {
                  "EC2SecurityGroupId" : "sg-b0ff1111",
                  "EC2SecurityGroupOwnerId" : "111122223333"
               }, {
                  "EC2SecurityGroupId" : "sg-ffd722222",
                  "EC2SecurityGroupOwnerId" : "111122223333"
               } ]
         }
      }
   }
}
```

AWS::RDS::DBSecurityGroupIngress

The AWS::RDS::DBSecurityGroupIngress type enables ingress to a DBSecurityGroup using one of two forms of authorization. First, EC2 or VPC Security Groups can be added to the DBSecurityGroup if the application using the database is running on EC2 or VPC instances. Second, IP ranges are available if the application accessing your database is running on the Internet. For more information about DB Security Groups, see Working with DB Security Groups

This type supports updates. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For details on the settings for DB security group ingress, see AuthorizeDBSecurityGroupIngress.

Syntax

```
{
   "CIDRIP (p. 282)": String,
   "EC2SecurityGroupId (p. 282)": String,
   "EC2SecurityGroupName (p. 282)": String,
   "EC2SecurityGroupOwnerId (p. 282)": String
}
```

Properties

CIDRIP

The IP range to authorize.

For an overview of CIDR ranges, go to the Wikipedia Tutorial.

Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupId

Id of the VPC or EC2 Security Group to authorize.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.

Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupName

Name of the EC2 Security Group to authorize.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.

Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupOwnerId

AWS Account Number of the owner of the EC2 Security Group specified in the EC2SecurityGroupName parameter. The AWS Access Key ID is not an acceptable value.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.

Type: String

Required: No

Update requires: replacement (p. 34)

AWS::Route53::RecordSet

The AWS::Route53::RecordSet type can be used as a standalone resource or as an embedded property in the AWS::Route53::RecordSetGroup (p. 285) type. Note that some AWS::Route53::RecordSet properties are valid only when used within AWS::Route53::RecordSetGroup.

Note that before you use AWS CloudFormation to add a recordset to hosted zone, that hosted zone must already be created in Amazon Route 53. AWS CloudFormation does not create new hosted zones.

This type supports updates. For details and constraints on updates for resource record sets, see Changing Your Resource Record Sets. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For more information about constraints and values for each property, see POST CreateHostedZone for hosted zones and POST ChangeResourceRecordSet for resource record sets.

Syntax

```
{
   "Type" : "AWS::Route53::RecordSet",
   "Properties" : {
      "HostedZoneId (p. 283)" : String,
      "HostedZoneName (p. 283)" : String,
      "Name (p. 283)" : String,
      "Type (p. 283)" : String,
      "TTL (p. 284)" : String,
      "ResourceRecords (p. 284)" : [ String ],
      "Weight (p. 284)" : Integer,
      "SetIdentifier (p. 284)" : String,
      "AliasTarget (p. 284)" : AliasTarget (p. 320),
      "Comment (p. 284)" : String,
   }
}
```

Properties

HostedZoneId

The ID of the hosted zone.

Required: Conditional. You must specify either the `HostedZoneName` or `HostedZoneId`, but you cannot specify both.

Type: String

HostedZoneName

The name of the domain for the hosted zone where you want to add the record set.

When you create a stack using an AWS::Route53::RecordSet that specifies `HostedZoneName`, AWS CloudFormation attempts to find a hosted zone whose name matches the HostedZoneName. If AWS CloudFormation cannot find a hosted zone with a matching domain name, or if there is more than one hosted zone with the specified domain name, AWS CloudFormation will not create the stack.

If you have multiple hosted zones with the same domain name, you must explicitly specify the hosted zone using `HostedZoneId`.

Required: Conditional. You must specify either the `HostedZoneName` or `HostedZoneId`, but you cannot specify both.

Type: String

Name

The name of the domain. This must be a fully-specified domain, ending with a period as the last label indication. If you omit the final period, Amazon Route 53 assumes the domain is relative to the root.

Required: Yes

Type: String

Type

The type of records to add.

Required: Yes

Type: String

Valid Values: A | AAAA | CNAME | MX | NS | PTR | SOA | SPF | SRV | TXT

TTL

The resource record cache time to live (TTL), in seconds.

If `TTL` is specified, `ResourceRecords` is also required.

Required: No

Type: String

ResourceRecords

List of resource records to add. Each record should be in the format appropriate for the record type specified by the `Type` property. For information about different record types and their record formats, see Appendix: Domain Name Format in the *Route 53 Developer Guide*.

Required: Conditional. Required if TTL is specified.

Type: list of Strings

Weight

Weighted resource record sets only: Among resource record sets that have the same combination of DNS name and type, a value that determines what portion of traffic for the current resource record set is routed to the associated location.

For more information on weighted resource record sets, see Setting Up Weighted Resource Record Sets in the *Route 53 Developer Guide*.

Required: Conditional. Required if you are creating a weighted resource record set.

Type: Integer

SetIdentifier

A unique identifier that differentiates among multiple resource record sets that have the same combination of DNS name and type.

Required: Conditional. Required if you are creating a weighted resource record set.

For more information on weighted resource record sets, see Setting Up Weighted Resource Record Sets in the *Route 53 Developer Guide*.

Type: String

AliasTarget

Alias resource record sets only: Information about the domain to which you are redirecting traffic.

For more information on alias resource record sets, see Creating Alias Resource Record Sets in the *Route 53 Developer Guide*.

> **Note**
>
> Currently, Amazon Route 53 supports aliases *only* for Elastic Load Balancing.

Required: Conditional. Required if you are creating an alias resource record set.

Type: AliasTarget (p. 320)

Comment

Any comments you want to include about the hosted zone.

Required: No

Type: String

Return Value

When you specify an AWS::Route53::RecordSet type as an argument to the `Ref` function, AWS CloudFormation returns the value of the domain name of the record set.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

For AWS::Route53::RecordSet snippets, see Amazon Route 53 Template Snippets (p. 139) .

AWS::Route53::RecordSetGroup

Note

Before you use AWS CloudFormation to add a record set to a hosted zone, that hosted zone must already be created in Amazon Route 53. AWS CloudFormation does not create new hosted zones.

This type supports updates. For details and constraints on updates for resource record sets, see Changing Your Resource Record Sets. For more information on updating stacks, see Updating AWS CloudFormation Stacks (p. 33).

For more information about constraints and values for each property, see POST CreateHostedZone for hosted zones and POST ChangeResourceRecordSet for resource record sets.

Syntax

```
{
   "Type" : "AWS::Route53::RecordSetGroup",
   "Properties" : {
      "HostedZoneId (p. 285)" : String,
      "HostedZoneName (p. 285)" : String,
      "RecordSets (p. 285)" : [ RecordSet1, ... ],
      "Comment (p. 286)" : String,
   }
}
```

Properties

HostedZoneId

The ID of the hosted zone.

Required: Conditional: You must specify either the `HostedZoneName` or `HostedZoneId`, but you cannot specify both.

Type: String

HostedZoneName

The name of the domain for the hosted zone where you want to add the record set.

When you create a stack using an AWS::Route53::RecordSet that specifies `HostedZoneName`, AWS CloudFormation attempts to find a hosted zone whose name matches the HostedZoneName. If AWS CloudFormation cannot find a hosted zone with a matching domain name, or if there is more than one hosted zone with the specified domain name, AWS CloudFormation will not create the stack.

If you have multiple hosted zones with the same domain name, you must explicitly specify the hosted zone using `HostedZoneId`.

Required: Conditional. You must specify either the `HostedZoneName` or `HostedZoneId`, but you cannot specify both.

Type: String

RecordSets

List of resource record sets to add.

Required: Yes

Type: list of AWS::Route53::RecordSet (p. 282)

Comment

Any comments you want to include about the hosted zone.

Required: No

Type: String

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyRecordSetGroup" }
```

For the resource with the logical ID "MyRecordSetGroup", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Template Examples

For AWS::Route53::RecordSetGroup snippets, see Amazon Route 53 Template Snippets (p. 139).

AWS::S3::Bucket

The AWS::S3::Bucket type creates an Amazon S3 bucket.

You can set a deletion policy for your bucket to control how AWS CloudFormation handles the bucket when the stack is deleted. For S3 buckets, you can choose to retain the bucket or to delete the bucket. For more information, see DeletionPolicy Attribute (p. 322).

AWS::S3::Bucket Snippets: Amazon S3 Template Snippets (p. 141).

Properties

Property	Type	Required	Notes
AccessControl	String	No	A canned ACL that grants predefined permissions on the bucket. Default is Private. For more information about canned ACLs, see Canned ACLs in the Amazon S3 documentation. Valid values for AccessControl: Private | PublicRead | PublicReadWrite | AuthenticatedRead | BucketOwnerRead | BucketOwnerFullControl
WebsiteConfiguration	Website Configuration Type (p. 320)	No	Information used to configure the bucket as a static website. For more information, see Hosting Websites on Amazon S3.

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

Return Values

Function	Type	Description
Ref (p. 326)	Bucket name	Example: mystack-mybucket-kdwwxmddtr2g
Fn::GetAtt (p. 324)	DomainName	The DNS name of the specified bucket. Example: mystack-mybucket-kdwwxmddtr2g.s3.amazonaws.com
Fn::GetAtt (p. 324)	WebsiteURL	Amazon S3 website endpoint for the specified bucket. Example: http://mystack-mybucket-kdwwxmddtr2g.s3-website-us-east-1.amazonaws.com/

Return Value

When this resource's logical ID is provided to the `Ref` intrinsic function, it will return the resource's name. For example:

```
{ "Ref": "MyResource" }
```

For the resource with the logical ID "MyResource", `Ref` will return the AWS resource name.

For more information about using the `Ref` function, see Ref (p. 326).

AWS::S3::BucketPolicy

The AWS::S3::BucketPolicy type applies a policy to an Amazon S3 bucket.

AWS::S3::BucketPolicy Snippet: Declaring an Amazon S3 Bucket Policy (p. 126)

Property	Type	Required	Notes
PolicyDocument	JSON	Yes	A policy document containing permissions to add to the specified bucket.
Bucket	String	Yes	Name of the bucket to which you want to add the policy.

AWS::SDB::Domain

The AWS::SDB::Domain type does not have any properties

When you specify an AWS::SDB::Domain type as an argument to the `Ref` function, AWS CloudFormation returns the value of the *`DomainName`*.

AWS::SNS::TopicPolicy

The AWS::SNS::TopicPolicy type applies a policy to SNS topics.

AWS::SNS::TopicPolicy Snippet: Declaring an Amazon SNS Topic Policy (p. 126)

Property	Type	Required	Notes
PolicyDocument	JSON	Yes	A policy document containing permissions to add to the specified SNS topics.
Topics	Array of SNS topic ARNs	Yes	The Amazon Resource Names (ARN) of the topics to which you want to add the policy. You can use the Ref function (p. 326) to specify an AWS::SNS::Topic (p. 288) resource.

AWS::SNS::Topic

The AWS::SNS::Topic type creates an Amazon SNS topic.

When you specify an AWS::SNS::Topic type as an argument to the `Ref` function, AWS CloudFormation returns the ARN of the topic.

Property	Type	Required	Notes
Subscription	Array	No	Array of SNS Subscription (p. 321) type.

AWS::SQS::Queue

The AWS::SQS::Queue type creates an Amazon SQS queue.

Syntax

```
{
   "Type": "AWS::SQS::Queue",
   "Properties": {
      "VisibilityTimeout (p. 288)": Integer
   }
}
```

Properties

VisibilityTimeout
The length of time during which the queue will be unavailable once a message is delivered from the queue. This blocks other components from receiving the same message and gives the initial component time to process and delete the message from the queue.

Values must be from 0 to 43200 seconds (12 hours). If no value is specified, the default value of 30 seconds will be used.

For more information about SQS Queue visibility timeouts, see Visibility Timeout in the *Amazon Simple Queue Service Developer Guide*.

Required: No

Type: Integer

Return Values

Ref

Returns: The queue URL. For example:
`https://sqs.us-east-1.amazonaws.com/803981987763/aa4-MyQueue-Z5NOSZO2PZE9`.

For more information about using the `Ref` function, see Ref (p. 326).

Fn::GetAtt

`Fn::GetAtt` returns a value for a specified attribute of this type. This section lists the available attributes and corresponding return values.

Arn
Returns the Amazon Resource Name (ARN) of the queue. For example:
`arn:aws:sqs:us-east-1:123456789012:mystack-myqueue-15PG5C2FC1CW8`

QueueName
Returns the queue name. For example:

mystack-myqueue-1VF9BKQH5BJVI

Examples

SQS Queue with Cloudwatch Alarms

```
{

  "AWSTemplateFormatVersion" : "2010-09-09",



  "Description" : "AWS CloudFormation Sample Template SQS_With_CloudWatch_Alarms:
 Sample template showing how to create an SQS queue with AWS CloudWatch alarms
 on queue depth. **WARNING** This template creates an Amazon SQS Queue and one
 or more Amazon CloudWatch alarms. You will be billed for the AWS resources
used if you create a stack from this template.",



  "Parameters" : {

    "AlarmEmail": {
      "Default": "nobody@amazon.com",

      "Description": "Email address to notify if there are any operational is
sues",

      "Type": "String"

    }

  },


  "Resources" : {

    "MyQueue" : {

      "Type" : "AWS::SQS::Queue",

      "Properties" : {

      }

    },

    "AlarmTopic": {

      "Type": "AWS::SNS::Topic",

      "Properties": {

        "Subscription": [{

          "Endpoint": { "Ref": "AlarmEmail" },

          "Protocol": "email"

        }]

      }

    },

    "QueueDepthAlarm": {

      "Type": "AWS::CloudWatch::Alarm",

      "Properties": {

        "AlarmDescription": "Alarm if queue depth grows beyond 10 messages",

        "Namespace": "AWS/SQS",

        "MetricName": "ApproximateNumberOfMessagesVisible",

        "Dimensions": [{

          "Name": "QueueName",
          "Value" : { "Fn::GetAtt" : ["MyQueue", "QueueName"] }
        }],
        "Statistic": "Sum",
        "Period": "300",
        "EvaluationPeriods": "1",
        "Threshold": "10",
        "ComparisonOperator": "GreaterThanThreshold",
        "AlarmActions": [{
          "Ref": "AlarmTopic"
        }],
        "InsufficientDataActions": [{
          "Ref": "AlarmTopic"
        }]
      }
    }
  },
  "Outputs" : {
    "QueueURL" : {
      "Description" : "URL of newly created SQS Queue",
      "Value" : { "Ref" : "MyQueue" }
    },
    "QueueARN" : {
      "Description" : "ARN of newly created SQS Queue",
      "Value" : { "Fn::GetAtt" : ["MyQueue", "Arn"]}
    },
    "QueueName" : {
      "Description" : "Name newly created SQS Queue",
      "Value" : { "Fn::GetAtt" : ["MyQueue", "QueueName"]}
    }
  }
}
```

See Also

- CreateQueue, in the *Amazon Web Services Amazon Simple Queue Service API Reference*
- What is Amazon Simple Queue Service?, in the *Amazon Simple Queue Service Developer Guide*.

AWS::SQS::QueuePolicy

The AWS::SQS::QueuePolicy type applies a policy to SQS queues.

AWS::SQS::QueuePolicy Snippet: Declaring an Amazon SQS Policy (p. 127)

Property	Type	Required	Notes
PolicyDocument	JSON	Yes	A policy document containing permissions to add to the specified SQS queues.
Queues	List of queue URLs	Yes	The URLs of the queues to which you want to add the policy. You can use the Ref function (p. 326) to specify an AWS::SQS::Queue (p. 288) resource.

Resource Property Types Reference

This section details the resource-specific properties for the resources supported by AWS CloudFormation.

Topics

- DynamoDB Provisioned Throughput (p. 305)
- EC2 ICMP Property Type (p. 306)
- EC2 MountPoint Property Type (p. 306)
- EC2 Network Interface Association (p. 307)
- EC2 Network Interface Attachment (p. 307)
- EC2 Network Interface Group Item (p. 308)
- EC2 PortRange Property Type (p. 308)
- EC2 Security Group Rule Property Type (p. 309)
- EC2 Tag (p. 310)
- ElasticBeanstalk ApplicationVersion Property Type (p. 310)
- ElasticBeanstalk ConfigurationTemplate Property Type (p. 311)
- ElasticBeanstalk OptionSettings Property Type (p. 312)
- ElasticBeanstalk SourceBundle Property Type (p. 313)
- ElasticLoadBalancing AppCookieStickinessPolicy Type (p. 313)
- ElasticLoadBalancing HealthCheck Type (p. 314)
- ElasticLoadBalancing LBCookieStickinessPolicy Type (p. 315)
- ElasticLoadBalancing Listener Property Type (p. 316)
- ElasticLoadBalancing Policy Type (p. 317)
- RDS Security Group Rule (p. 319)
- Route 53 AliasTarget Property Type (p. 320)
- Amazon S3 Website Configuration Property Type (p. 320)
- SNS Subscription Property Type (p. 321)

AutoScaling BlockDeviceMapping Property Type

The BlockDeviceMapping type is an embedded property of the Auto Scaling LaunchConfiguration (p. 193) type.

The following properties are available with the AutoScaling BlockDeviceMapping type.

Property	Type	Required	Notes
DeviceName	String	Yes	The name of the device within Amazon EC2.
VirtualName	String	Conditional	The name of the virtual device. The name must be in the form `ephemeral`*X* where *X* is a number starting from 0, for example, ephemeral0. If you specify the VirtualName property, do not specify the Ebs property.
Ebs	Block Device Template (p. 293) type	Conditional	The information for the Elastic Block Store volume. If you specify the Ebs property, do not specify the VirtualName property.

AutoScaling BlockDeviceTemplate Property Type

The BlockDeviceTemplate type is an embedded property of the Auto Scaling Block Device Mapping (p. 293) type.

For BlockDeviceTemplate snippets, see Auto Scaling Launch Configuration Resource (p. 109).

The following properties are available with the AutoScaling BlockDeviceTemplate type.

Property	Type	Required	Notes
SnapshotId	String	Conditional	The Snapshot ID of the volume to use. If you specify SnapshotId, do not specify VolumeSize.
VolumeSize	String	Conditional	The volume size, in GigiBytes. If you specify VolumeSize, do not specify SnapshotId.

Auto Scaling NotificationConfiguration Property Type

The NotificationConfiguration property configures an Auto Scaling group to send notifications when specified events take place.

The NotificationConfiguration is an embedded property of the AWS::AutoScaling::AutoScalingGroup (p. 190) type.

For NotificationConfiguration snippets, see Auto Scaling Group with Notifications (p. 111).

Property	Type	Required	Notes
TopicARN	String	Yes	The Amazon Resource Name (ARN) of the Amazon Simple Notification Service (SNS) topic.
NotificationTypes	String list	Yes	The type of events that will trigger the notification. A list of events that will trigger the notification, which can include any or all of the following: autoscaling: EC2_INSTANCE_LAUNCH, autoscaling:EC2_INSTANCE_LAUNCH_ERROR, autoscaling:EC2_INSTANCE_TERMINATE, autoscaling:EC2_INSTANCE_TERMINATE_ERROR, and autoscaling:TEST_NOTIFICATION. For more information on event types, see DescribeAutoScalingNotificationTypes.

Auto Scaling Tags Property Type

The Auto Scaling Tags property is an embedded property of the AWS::AutoScaling::AutoScalingGroup (p. 190) type. For more information about tags, go to Tagging Auto Scaling Groups and Amazon EC2 Instances in the *Auto Scaling Developer Guide*.

AWS CloudFormation adds the following tags to all Auto Scaling groups and associated instances:

- aws:cloudformation:stack-name
- aws:cloudformation:stack-id
- aws:cloudformation:logical-id

Syntax

```
{
   "Key (p. 295)" : String,
   "Value (p. 295)" : String,
   "PropagateAtLaunch (p. 295)" : Boolean
}
```

Properties

Key

The key name of the tag.

Required: Yes

Type: String

Value

The value for the tag.

Required: Yes

Type: String

PropagateAtLaunch

Set to `true` if you want AWS CloudFormation to copy the tag to EC2 instances that are launched as part of the auto scaling group. Set to `false` if you want the tag attached only to the auto scaling group and not copied to any instances launched as part of the auto scaling group.

Required: Yes

Type: Boolean

Example

The following example template snippet creates two Auto Scaling tags. The first tag, `MyTag1`, is attached to an Auto Scaling group named `WebServerGroup` and is copied to any EC2 instances launched as part of the Auto Scaling group. The second tag, `MyTag2`, is attached only to the Auto Scaling group named `WebServerGroup`.

```
"WebServerGroup" : {
   "Type" : "AWS::AutoScaling::AutoScalingGroup",
   "Properties" : {
      "AvailabilityZones" : { "Fn::GetAZs" : "" },
      "LaunchConfigurationName" : { "Ref" : "LaunchConfig" },
      "MinSize" : "1",
      "MaxSize" : "2",
      "LoadBalancerNames" : [ { "Ref" : "ElasticLoadBalancer" } ],
      "Tags" : [ {
         "Key" : "MyTag1",
         "Value" : "Hello World 1",
         "PropagateAtLaunch" : "true"
      }, {
         "Key" : "MyTag2",
         "Value" : "Hello World 2",
         "PropagateAtLaunch" : "false"
      } ]
   }
}
```

CloudFormation Stack Parameters Property Type

The Parameters type is an embedded property of the AWS::CloudFormation::Stack (p. 211) type.

The Parameters type contains a set of value pairs that represent the parameters that will be passed to the template used to create an AWS::CloudFormation::Stack resource. Each parameter has a name corresponding to a parameter defined in the embedded template and a value respresenting the value that you want to set for the parameter. For example, the sample template EC2ChooseAMI.template contains the following Parameters section:

```
"Parameters" : {
   "InstanceType" : {
      "Type" : "String",
      "Default" : "m1.small",
      "Description" : "EC2 instance type, e.g. m1.small, m1.large, etc."
   },
   "WebServerPort" : {
      "Type" : "String",
      "Default" : "80",
      "Description" : "TCP/IP port of the web server"
   },
   "KeyName" : {
      "Type" : "String",
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to
 the web server"
   }
}
```

You could use the following template to embed a stack (myStackWithParams) using the EC2ChooseAMI.template and use the Parameters property in the AWS::CloudFormation::Stack resource to specify m1.large as the InstanceType and mykey as the KeyName:

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "myStackWithParams" : {
         "Type" : "AWS::CloudFormation::Stack",
         "Properties" : {
           "TemplateURL" : "https://s3.amazonaws.com/cloudformation-templates-
us-east-1/EC2ChooseAMI.template",
            "Parameters" : {
               "InstanceType" : "t1.micro",
               "KeyName" : "mykey"
            }
         }
      }
   }
}
```

CloudFront CacheBehavior Type

Describes the Amazon CloudFront cache behavior when the requested URL matches a pattern. This is an embedded property of the DistributionConfig (p. 300) type.

Syntax

```
{
   "TargetOriginId (p. 297)" : String,
   "ForwardedValues (p. 297)" : ForwardedValues,
   "TrustedSigners (p. 297)" : [ String, ... ],
   "ViewerProtocolPolicy (p. 297)" : String,
   "MinTTL (p. 297)" : String
   "PathPattern (p. 297)" : String
}
```

Properties

TargetOriginId
The ID value of the origin to which you want CloudFront to route requests when a request matches the PathPattern.

Required: Yes

Type: String

ForwardedValues
Specifies how CloudFront handles query strings.

Required: Yes

Type: ForwardedValues type

TrustedSigners
A list of AWS accounts that you want to allow to create signed URLs for private content.

Required: No

Type: list of Strings

ViewerProtocolPolicy
Specifies the protocol that users can use to access the files in the origin specified by TargetOriginId when a request matches the PathPattern.

Required: Yes

Type: String

MinTTL
The minimum amount of time that you want objects to stay in the cache before CloudFront queries your origin to see whether the object has been updated.

Required: No

Type: String

PathPattern
The pattern for which you want this cache behavior to apply (ex. "images/*.jpg").

When CloudFront receives an end-user request, the requested path is compared with path patterns in the order in which cache behaviors are listed in the stack specification for the distribution.

Required: Yes

Type: String

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront ForwardedValues Type

Specifies how Amazon CloudFront handles query strings for a cache behavior. This is an embedded property of the DefaultCacheBehavior (p. 299) and CacheBehavior (p. 297) types.

Syntax

```
{
   "QueryString (p. 298)" : Boolean
}
```

Properties

QueryString
Indicates whether you want CloudFront to forward query strings to the origin that is associated with this cache behavior. If so, specify "true"; if not, specify "false".

Required: Yes

Type: Boolean

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront CustomOrigin Type

CustomOrigin is an embedded property of the CloudFront Origin (p. 302) type.

Syntax

```
{
   "HTTPPort (p. 299)" : String
   "HTTPSPort (p. 299)" : String
   "OriginProtocolPolicy (p. 299)" : String
}
```

Properties

Note

For more information about constraints and values for each property, see DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

HTTPPort
The HTTP port the custom origin listens on.

Required: No

Type: String

HTTPSPort
The HTTPS port the custom origin listens on.

Required: No

Type: String

OriginProtocolPolicy
The origin protocol policy to apply to your origin.

Required: Yes

Type: String

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront DefaultCacheBehavior Type

Describes the default cache behavior for an Amazon CloudFront distribution. This is an embedded property of the DistributionConfig (p. 300) type.

Syntax

```
{
   "TargetOriginId (p. 299)" : String,
   "ForwardedValues (p. 300)" : ForwardedValues,
   "TrustedSigners (p. 300)" : [ String, ... ],
   "ViewerProtocolPolicy (p. 300)" : String,
   "MinTTL (p. 300)" : String
}
```

Properties

TargetOriginId
The value of ID for the origin that you want CloudFront to route requests to when the default cache behavior is applicable to a request.

Required: Yes

Type: String

ForwardedValues
Specifies how CloudFront handles query strings.

Required: Yes

Type: ForwardedValues type

TrustedSigners
A list of AWS accounts that you want to allow to create signed URLs for private content.

Required: No

Type: list of Strings

ViewerProtocolPolicy
Specifies the protocol that users can use to access the files in the origin specified by TargetOriginId when the default cache behavior is applicable to a request.

Required: Yes

Type: String

MinTTL
The minimum amount of time that you want objects to stay in the cache before CloudFront queries your origin to see whether the object has been updated.

Required: No

Type: String

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront DistributionConfig Type

DistributionConfig is an embedded property of the AWS::CloudFront::Distribution (p. 213) type.

Syntax

```
{
   "Aliases (p. 301)" : [ String, ... ],
   "DefaultRootObject (p. 301)" : String,
   "Origins (p. 301)" : [ Origin, ... ]
   "DefaultCacheBehavior (p. 301)" : DefaultCacheBehavior,
   "CacheBehaviors (p. 301)" : [ CacheBehavior, ... ],
   "Comment (p. 301)" : String,
   "Logging (p. 301)" : Logging,
   "Enabled (p. 301)" : Boolean,
}
```

Properties

Note

For more information about constraints and values for each property, see DistributionConfig Complex Type in the *Amazon CloudFront API Reference*.

Aliases
CNAMEs (alternate domain names), if any, for the distribution.

Required: No

Type: List of Strings

DefaultRootObject
The object (ex. "index.html") that you want CloudFront to request from your origin when the root URL for your distribution (ex. "http://example.com/") is requested.

Note

Specifying a default root object avoids exposing the contents of your distribution.

Required: No

Type: String

Origins
A list of origins for this CloudFront distribution. For each origin, you can specify whether it is an *S3* or *custom* origin.

Required: Yes

Type: list of Origin (p. 302)s.

DefaultCacheBehavior
The default cache behavior that is triggered if you do not specify a CacheBehavior element, or if files don't match any of the values of PathPattern in CacheBehavior elements.

Required: Yes

Type: DefaultCacheBehavior type (p. 299)

Update requires: no interruption (p. 33)

CacheBehaviors
A list of CacheBehaviors for the distribution.

Required: No

Type: String

Update requires: no interruption (p. 33)

Comment
Any comments you want to include about the distribution.

Required: No

Type: String

Update requires: no interruption (p. 33)

Logging
Controls whether access logs are written for the distribution. To turn on access logs, include this property.

Required: No

Type: Logging (p. 302) type

Enabled
Controls whether the distribution is enabled to accept end user requests for content.

Required: Yes

Type: Boolean

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront Logging Type

Logging is an embedded property of the DistributionConfig (p. 300) type.

Syntax

```
{
   "Bucket (p. 302)" : String,
   "Prefix (p. 302)" : String
}
```

Properties

Note

For more information about constraints and values for each property, see DistributionConfig Complex Type.

Bucket

The Amazon S3 bucket to store the access logs in.

Required: Yes

Type: String

Prefix

An optional string that, if defined, will be used as a prefix for the access log file names for this distribution.

Required: No

Type: String

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront Origin Type

Describes an Amazon CloudFront distribution origin. This is an embedded property of the DistributionConfig (p. 300) type.

Syntax

```
{
   "DomainName (p. 303)" : String,
   "Id (p. 303)" : String,
   "S3OriginConfig (p. 303)" : S3 Origin,
   "CustomOriginConfig (p. 303)" : CustomOrigin,
}
```

Properties

DomainName
The DNS name of the Amazon S3 bucket or the HTTP server from which you want CloudFront to get objects for this origin.

Required: Yes

Type: String

Id
An identifier for the origin. The value of Id must be unique within the distribution.

Required: Yes

Type: String

S3OriginConfig
Origin information to specify an Amazon S3 origin.

Required: Conditional. You cannot use S3Origin and CustomOrigin in the same distribution, but you *must* specify one or the other.

Type: S3Origin (p. 303) type

CustomOriginConfig
Origin information to specify a custom origin.

Required: Conditional. You cannot use CustomOrigin and S3 Origin in the same distribution, but you *must* specify one or the other.

Type: CustomOrigin (p. 298) type

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudFront S3Origin Type

S3Origin is an embedded property of the Origin (p. 302) type.

Syntax

```
{
   "OriginAccessIdentity (p. 304)" : String
}
```

Properties

Note

For more information about constraints and values for OriginAccessIdentity, see DistributionConfig Complex Type in the *Auto Scaling API Reference*.

OriginAccessIdentity
The CloudFront origin access identity to associate with the origin. This is used to configure the origin so that end users can only access objects in an Amazon S3 bucket through CloudFront.

Required: No

Type: String

Template Examples

To view AWS::CloudFront::Distribution snippets, see Amazon CloudFront Template Snippets (p. 137).

CloudWatch Metric Dimension Property Type

The following properties are available with Amazon CloudWatch Metric Dimension.

The Metric Dimension is an embedded property of the AWS::CloudWatch::Alarm (p. 214) type.

Property	Type	Required	Notes
Name	String	Yes	Name of the dimension.
Value	String	Yes	The value representing the dimension measurement.

DynamoDB Primary Key

Describes an Amazon DynamoDB primary key for an AWS::DynamoDB::Table (p. 216) resource.

There are two types of primary key types: *hash* type primary keys, and *hash and range* type primary keys:

- A hash type primary key creates a table that has an unordered hash index based on the `AttributeName` of the HashKeyElement.
- A hash and range type primary key creates a table that has both an unordered hash index based on the `AttributeName` of the HashKeyElement, and an ordered hash index based on the `AttributeName` of the RangeKeyElement.

For a complete discussion of Amazon DynamoDB primary keys, see Primary Key in the *Amazon DynamoDB Developer Guide*.

Syntax

```
{
   "HashKeyElement": {
      "AttributeName (p. 305)" : String,
      "AttributeType (p. 305)" : String
   },
   "RangeKeyElement" : {
      "AttributeName (p. 305)" : String,
      "AttributeType (p. 305)" : String
   }
}
```

Parameters

Each element type has an AttributeName and AttributeType, defined as follows:

AttributeName
The name of the attribute that will serve as the primary key for this table. Primary key element names can be 1 – 255 characters long and have no character restrictions.

Required: Yes

Type: String

AttributeType
The type of this attribute. This must be either "S" for string data, or "N" for numeric data.

Required: Yes

Type: String

Note

For detailed information about the limits of primary key values in Amazon DynamoDB, see Limits in Amazon DynamoDB in the *Amazon DynamoDB Developer Guide*.

Examples

For an example of a declared primary key, see AWS::DynamoDB::Table (p. 216).

DynamoDB Provisioned Throughput

Describes a set of provisioned throughput values for an AWS::DynamoDB::Table (p. 216) resource. Amazon DynamoDB uses these capacity units to allocate sufficient resources to provide the requested throughput.

For a complete discussion of Amazon DynamoDB provisioned throughput values, see Specifying Read and Write Requirements (Provisioned Throughput) in the *Amazon DynamoDB Developer Guide*.

Syntax

```
{
   "ReadCapacityUnits (p. 306)" : Number,
  "WriteCapacityUnits (p. 306)" : Number
}
```

Parameters

ReadCapacityUnits
Sets the desired minimum number of consistent reads of items (of up to 1KB in size) per second for the specified table before Amazon DynamoDB balances the load.

Required: Yes

Type: Number

WriteCapacityUnits
Sets the desired minimum number of consistent writes of items (of up to 1KB in size) per second for the specified table before Amazon DynamoDB balances the load.

Note

For detailed information about the limits of provisioned throughput values in Amazon DynamoDB, see Limits in Amazon DynamoDB in the *Amazon DynamoDB Developer Guide*.

Examples

For an example of declared provisioned throughput values, see AWS::DynamoDB::Table (p. 216).

EC2 ICMP Property Type

The EC2 ICMP property is an embedded property of the AWS::EC2::NetworkAclEntry (p. 229) type.

The following properties are available with the EC2 ICMP type.

Property	Type	Required	Notes
Code	Integer	Conditional	The Internet Control Message Protocol (ICMP) code. You can use -1 to specify all ICMP codes for the given ICMP type. Condition: Required if specifying 1 (ICMP) for the CreateNetworkAclEntry protocol parameter.
Type	Integer	Conditional	The Internet Control Message Protocol (ICMP) type. You can use -1 to specify all ICMP types. Condition: Required if specifying 1 (ICMP) for the CreateNetworkAclEntry protocol parameter.

EC2 MountPoint Property Type

The EC2 MountPoint property is an embedded property of the AWS::EC2::Instance (p. 258) type.

The following properties are available with the EC2 MountPoint type.

Property	Type	Required	Notes
Device	String	Yes	How the device is exposed to the instance (such as /dev/sdh, or xvdh).
VolumeId	String	Yes	The ID of the Amazon EBS volume. The volume and instance must be within the same Availability Zone and the instance must be running.

EC2 Network Interface Association

Describes a network interface association for an Elastic Network Interface (ENI). AWS::EC2::NetworkInterface (p. 231) takes an object of this type in its Association property.

Syntax

```
{
   "AttachmentID" : String,
   "InstanceID" : String,
   "PublicIp" : String,
   "IpOwnerId" : String
}
```

Properties

AttachmentID
The ID of the network interface attachment.

Required: Yes

Type: String

InstanceID
The ID of the instance attached to the network interface.

Required: Yes

Type: String

PublicIp
The address of the Elastic IP address bound to the network interface.

Required: Yes

Type: String

IpOwnerId
The ID of the Elastic IP address owner.

Required: Yes

Type: String

EC2 Network Interface Attachment

Describes a network interface attachment for an Elastic Network Interface (ENI). AWS::EC2::NetworkInterface (p. 231) takes an object of this type in its Attachment property.

Syntax

```
{
   "AttachmentID" : String,
   "InstanceID" : String
}
```

Properties

AttachmentID
: The ID of the network interface attachment.

  Required: Yes

  Type: String

InstanceID
: The ID of the instance attached to the network interface.

  Required: Yes

  Type: String

EC2 Network Interface Group Item

Refers to an individual EC2 security group by ID or name in a group set.
AWS::EC2::NetworkInterface (p. 231) takes a list of objects of this type in its GroupSet property.

Syntax

```
{
   "GroupId" : String,
   "GroupName" : String
}
```

Properties

Key
: ID of the security group.

  Required: Yes

  Type: String

Value
: Name of the security group.

  Required: Yes

  Type: String

EC2 PortRange Property Type

The EC2 PortRange property is an embedded property of the AWS::EC2::NetworkAclEntry (p. 229) type.

The following properties are available with the EC2 PortRange type.

Property	Type	Required	Notes
From	Integer	Conditional	The first port in the range. Condition: Required if specifying 6 (TCP) or 17 (UDP) for the CreateNetworkAclEntry protocol parameter.
To	Integer	Conditional	The last port in the range. Condition: Required if specifying 6 (TCP) or 17 (UDP) for the CreateNetworkAclEntry protocol parameter.

EC2 Security Group Rule Property Type

The following properties are available with the EC2 Security Group Rule type.

The EC2 Security Group Rule is an embedded property of the AWS::EC2::SecurityGroup (p. 237) type.

Property	Type	Required	Notes
IpProtocol	String	Required	IP protocol name or number. For valid values, go to the IpProtocol parameter in AuthorizeSecurityGroupIngress
CidrIp	String	Conditional	If you specify SourceSecurityGroupName, do not specify CidrIp.
SourceSecurityGroupName	String	Conditional	Specifies the name of the Amazon EC2 Security Group to allow access or uses the Ref intrinsic function to refer to the logical name of a security group defined in the same template. If you specify CidrIp, do not specify SourceSecurityGroupName.
SourceSecurityGroupId	String	Conditional	For VPC security groups only. Specifies the ID of the Amazon EC2 Security Group to allow access or uses the Ref intrinsic function to refer to the logical ID of a security group defined in the same template. Condition: If you specify CidrIp, do not specify SourceSecurityGroupName.
SourceSecurityGroupOwnerId	String	Conditional	Specifies the AWS Account ID of the owner of the Amazon EC2 Security Group specified in the SourceSecurityGroupName property. If you specify SourceSecurityGroupName and that security group is owned by a different account than the account creating the stack, you must specify the SourceSecurityGroupOwnerId; otherwise, this property is optional.
FromPort	String	Yes	Start of port range for the TCP and UDP protocols, or an ICMP type number. An ICMP type number of -1 indicates a wildcard (i.e., any ICMP type number).

Property	Type	Required	Notes
ToPort	String	Yes	End of port range for the TCP and UDP protocols, or an ICMP code. An ICMP code of -1 indicates a wildcard (i.e., any ICMP code).

EC2 Tag

EC2 Tags are embedded properties of the AWS::EC2::Instance (p. 222), AWS::EC2::Volume (p. 247), and AWS::EC2::NetworkInterface (p. 231) types.

EC2 Tags are typically provided in a list:

```
"Tags": [{"Key": "Role", "Value": "Test Instance"}, ...]
```

Syntax

```
{
   "Key" : String,
   "Value" : String
}
```

Properties

Key
The key term for this item.
Required: Yes
Type: String

Value
A value associated with the key term.
Required: Yes
Type: String

ElasticBeanstalk ApplicationVersion Property Type

The ApplicationVersion property is an embedded property of the `AWS::ElasticBeanstalk::Application` type.

The following properties are available with ApplicationVersion Property.

Property	Type	Required	Notes
VersionLabel	String	Yes	A label uniquely identifying the version for the associated application.
SourceBundle	Source Bundle (p. 313)	No	The location where the source bundle is located for this version.

ElasticBeanstalk ConfigurationTemplate Property Type

`ConfigurationTemplate` is an embedded property of the AWS::ElasticBeanstalk::Application (p. 257) type, which contains an array of these objects in its ConfigurationTemplates property.

Syntax

```
{
   "TemplateName" : String,
   "Description" : String
   "OptionSettings" : [OptionSetting1,...],
   "SolutionStackName" : String
}
```

Members

TemplateName
The name of the configuration template.

Type: String

Required: Yes

Description
An optional description for this configuration.

Type: String

Required: No

OptionSettings
An array of OptionSettings (p. 312) for this Elastic Beanstalk configuration. For a complete list of Elastic Beanstalk configuration options, see: Option Values, in the *AWS Elastic Beanstalk Developer Guide*.

Type: [OptionSettings (p. 312)]

Required: No

SolutionStackName
The name of an existing Elastic Beanstalk solution stack used by this configuration. A solution stack specifies the operating system, architecture, and application server for a configuration template. It also defines configuration options, their possible and default values. If SolutionStackName is not specified, the default Elastic Beanstalk solution stack will be used.

Type: String

Required: No

Example

This example of an ElasticBeanstalk `ConfigurationTemplate` is found in the AWS CloudFormation sample template: ElasticBeanstalkSample.template, which also provides an example of its use within an `AWS::ElasticBeanstalk::Application`.

```
{
   "TemplateName" : "DefaultConfiguration",
   "Description" : "Default Configuration Version 1.0 - with SSH access",
   "OptionSettings" : [{
      "Namespace" : "aws:autoscaling:launchconfiguration",
      "OptionName" : "EC2KeyName",
      "Value" : { "Ref" : "KeyName" }
   }]
}
```

ElasticBeanstalk OptionSettings Property Type

`OptionSettings` is an embedded property of the AWS::ElasticBeanstalk::Application (p. 257) type. It is used by ConfigurationTemplate (p. 311) to specify an array of options for the Elastic Beanstalk configuration described by the template.

Note

You can get the set of valid settings for an Elastic Beanstalk configuration by using the `elastic-beanstalk-describe-configuration-settings` command. For more information, see the AWS Elastic Beanstalk User Guide.

Syntax

```
{
   "NameSpace" : String,
   "OptionName" : String],
   "Value" : String
}
```

Members

NameSpace
A unique namespace identifying the option's associated AWS resource.

Type: String

Required: Yes

OptionName
The name of the configuration option.

Type: String

Required: Yes

Value
The value of the setting.

Type: String

Required: Yes

Example

This example of using `OptionSettings` is found in the AWS CloudFormation sample template: ElasticBeanstalkSample.template, which also provides an example of its use within an `AWS::ElasticBeanstalk::Application`.

```
"OptionSettings" : [{
   "Namespace" : "aws:autoscaling:launchconfiguration",
   "OptionName" : "EC2KeyName",
   "Value" : { "Ref" : "KeyName" }
}]
```

ElasticBeanstalk SourceBundle Property Type

The SourceBundle property is an embedded property of the `AWS::ElasticBeanstalk::Application` type.

The following properties are available with the SourceBundle property.

Property	Type	Required	Notes
S3Bucket	String	Yes	The Amazon S3 bucket where the data is located.
S3Key	Source Bundle	No	The Amazon S3 key where the data is located.

ElasticLoadBalancing AppCookieStickinessPolicy Type

The AppCookieStickinessPolicy type is an embedded property of the AWS::ElasticLoadBalancing::LoadBalancer (p. 258) type.

Syntax

```
{
   "CookieName (p. 313)" : String,
   "PolicyName (p. 313)" : String
}
```

Properties

CookieName

Name of the application cookie used for stickiness.

Required: Yes

Type: String

PolicyName

The name of the policy being created. The name must be unique within the set of policies for this Load Balancer.

Required: Yes

Type: String

See Also

- AWS::ElasticLoadBalancing::LoadBalancer (p. 258)
- ElasticLoadBalancing Policy Type (p. 317)
- ElasticLoadBalancing LBCookieStickinessPolicy Type (p. 315)
- CreateAppCookieStickinessPolicyin the *Elastic Load Balancing API Reference*

ElasticLoadBalancing HealthCheck Type

The ElasticLoadBalancing HealthCheck is an embedded property of the AWS::ElasticLoadBalancing::LoadBalancer (p. 258) type.

Syntax

```
{
   "HealthyThreshold (p. 314)" : String,
   "Interval (p. 314)" : String,
   "Target (p. 314)" : String,
   "Timeout (p. 315)" : String,
   "UnhealthyThreshold (p. 315)" : String
}
```

Properties

HealthyThreshold
Specifies the number of consecutive health probe successes required before moving the instance to the Healthy state.

Required: Yes

Type: String

Interval
Specifies the approximate interval, in seconds, between health checks of an individual instance.

Required: Yes

Type: String

Target
Specifies the instance being checked. The protocol is either TCP or HTTP. The range of valid ports is 1 through 65535.

Required: Yes

Type: String

Note

TCP is the default, specified as a TCP: port pair—for example, "TCP:5000". In this case a healthcheck simply attempts to open a TCP connection to the instance on the specified port. Failure to connect within the configured timeout is considered unhealthy.

For HTTP, the situation is different. HTTP is specified as a HTTP:port;/;PathToPing; grouping—for example, "HTTP:80/weather/us/wa/seattle". In this case, a HTTP GET request is issued to the instance on the given port and path. Any answer other than "200 OK" within the timeout period is considered unhealthy.

The total length of the HTTP ping target needs to be 1024 16-bit Unicode characters or fewer.

Timeout
Specifies the amount of time, in seconds, during which no response means a failed health probe. This value must be less than the value for *Interval*.

Required: Yes

Type: String

UnhealthyThreshold
Specifies the number of consecutive health probe failures required before moving the instance to the Unhealthy state.

Required: Yes

Type: String

ElasticLoadBalancing LBCookieStickinessPolicy Type

The LBCookieStickinessPolicy type is an embedded property of the AWS::ElasticLoadBalancing::LoadBalancer (p. 258) type.

Syntax

```
{
   "CookieExpirationPeriod (p. 315)" : String,
   "PolicyName (p. 315)" : String
}
```

Properties

CookieExpirationPeriod
The time period, in seconds, after which the cookie should be considered stale. If this parameter isn't specified, the sticky session will last for the duration of the browser session.

Required: No

Type: String

PolicyName
The name of the policy being created. The name must be unique within the set of policies for this load balancer.

See Also

- AWS::ElasticLoadBalancing::LoadBalancer (p. 258)
- ElasticLoadBalancing Policy Type (p. 317)
- ElasticLoadBalancing AppCookieStickinessPolicy Type (p. 313)

- CreateLBCookieStickinessPolicyin the *Elastic Load Balancing API Reference*

ElasticLoadBalancing Listener Property Type

The Listener property is an embedded property of the AWS::ElasticLoadBalancing::LoadBalancer (p. 258) type.

Syntax

```
{
   "InstancePort (p. 316)" : String,
   "InstanceProtocol (p. 316)" : String,
   "LoadBalancerPort (p. 316)" : String,
   "PolicyNames (p. 316)" :  [ String, ... ],
   "Protocol (p. 317)" : String,
   "SSLCertificateId (p. 317)" : String
}
```

Properties

InstancePort

Specifies the TCP port on which the instance server is listening. This property cannot be modified for the life of the LoadBalancer.

Required: Yes

Type: String

InstanceProtocol

Specifies the protocol to use for routing traffic to back-end instances—HTTP, HTTPS, TCP, or SSL. This property cannot be modified for the life of the load balancer.

Required: Yes

Type: String

Note

- If the front-end protocol is HTTP or HTTPS, `InstanceProtocol` has to be at the same protocol layer, i.e., HTTP or HTTPS. Likewise, if the front-end protocol is TCP or SSL, `InstanceProtocol` has to be TCP or SSL.
- If there is another listener with the same InstancePort whose `InstanceProtocol` is secure, i.e., HTTPS or SSL, the listener's `InstanceProtocol` has to be secure, i.e., HTTPS or SSL. If there is another listener with the same InstancePort whose `InstanceProtocol` is HTTP or TCP, the listener's `InstanceProtocol` must be either HTTP or TCP.

LoadBalancerPort

Specifies the external LoadBalancer port number. This property cannot be modified for the life of the LoadBalancer.

Required: Yes

Type: String

PolicyNames

List of policies to be associated with the listener.

Required: No

Type: List of Policy (p. 317), AppCookieStickinessPolicy (p. 313), or LBCookieStickinessPolicy (p. 315) objects.

Protocol

Specifies the LoadBalancer transport protocol to use for routing — TCP or HTTP. This property cannot be modified for the life of the LoadBalancer.

Required: Yes

Type: String

SSLCertificateId

The ARN of the SSL certificate to use. For more information on SSL certificates, see Managing Server Certificates in the AWS Identity and Access Management documentation.

Required: No

Type: String

ElasticLoadBalancing Policy Type

The ElasticLoadBalancing Policy type is an embedded property of the AWS::ElasticLoadBalancing::Listener (p. 316) type. It is used to describe a policy to include in the listener's `PolicyNames` field.

Syntax

```
{
   "Attributes (p. 317)" : [ { "Name", String, "Value", String }, ... ],
   "InstancePorts (p. 317)" : [ String, ... ],
   "LoadBalancerPorts (p. 317)" : [ String, ... ],
   "PolicyName (p. 318)" : String,
   "PolicyType (p. 318)" : String
}
```

Properties

Attributes

A list of arbitrary attributes for this policy.

Required: No

Type: List of JSON name-value pairs.

InstancePorts

A list of instance ports for the policy. These are the ports associated with the back-end server.

Required: No

Type: String

LoadBalancerPorts

A list of external load balancer ports for the policy.

Required: Yes

Type: String

PolicyName

A name for this policy that is unique to the load balancer.

Required: Yes

Type: String

PolicyType

The name of the policy type for this policy. This must be one of the types reported by the Elastic Load Balancing DescribeLoadBalancerPolicyTypes action.

Required: Yes

Type: String

Example

This example shows a snippet of the Policies section of an ELB listener.

```
"Policies" : [
   {
      "PolicyName" : "MySSLNegotiationPolicy",
      "PolicyType" : "SSLNegotiationPolicyType",
      "Attributes" : [
         { "Name" : "Protocol-TLSv1", "Value" : "true" },
         { "Name" : "Protocol-SSLv2", "Value" : "true" },
         { "Name" : "Protocol-SSLv3", "Value" : "false" },
         { "Name" : "DHE-RSA-AES256-SHA", "Value" : "true" } ]
   }, {
      "PolicyName" : "MyAppCookieStickinessPolicy",
      "PolicyType" : "AppCookieStickinessPolicyType",
      "Attributes" : [
         { "Name" : "CookieName", "Value" : "MyCookie"} ]
   }, {
      "PolicyName" : "MyPublicKeyPolicy",
      "PolicyType" : "PublicKeyPolicyType",
      "Attributes" : [ {
         "Name" : "PublicKey",
         "Value" : { "Fn::Join" : [
            "\n", [
               "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDh/51Aohx5Vrpm
lfGHZCzciMBa",
               "fkHve+MQYYJcxmNUKMdsWnz9WtVfKxxWUU7Cfor4lorYmENGCG8FWqCoLD
MFs7pN",
               "yGEtpsrlKhzZWtgY1d7eGrUrBil03bI90E2KW0j4qAwGYAC8xix
OkNClicojeEz4",
               "f4rr3sUf+ZBSsuMEuwIDAQAB" ]
         ] }
      } ]
   }, {
      "PolicyName" : "MyBackendServerAuthenticationPolicy",
      "PolicyType" : "BackendServerAuthenticationPolicyType",
      "Attributes" : [
         { "Name" : "PublicKeyPolicyName", "Value" : "MyPublicKeyPolicy" } ],
      "InstancePorts" : [ "8443" ]
   }
]
```

See Also

- AWS::ElasticLoadBalancing::LoadBalancer (p. 258)
- ElasticLoadBalancing AppCookieStickinessPolicy Type (p. 313)
- ElasticLoadBalancing LBCookieStickinessPolicy Type (p. 315)

RDS Security Group Rule

The RDS Security Group Rule is an embedded property of the AWS::RDS::DBSecurityGroup (p. 279) type.

Syntax

```
{
   "CIDRIP (p. 319)": String,
   "EC2SecurityGroupId (p. 319)": String,
   "EC2SecurityGroupName (p. 319)": String,
   "EC2SecurityGroupOwnerId (p. 320)": String
}
```

Properties

CIDRIP

The IP range to authorize.

For an overview of CIDR ranges, go to the Wikipedia Tutorial.
Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupId

Id of the VPC or EC2 Security Group to authorize.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.
Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupName

Name of the EC2 Security Group to authorize.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.
Type: String

Required: No

Update requires: replacement (p. 34)

EC2SecurityGroupOwnerId

AWS Account Number of the owner of the EC2 Security Group specified in the EC2SecurityGroupName parameter. The AWS Access Key ID is not an acceptable value.

For VPC DB Security Groups, use EC2SecurityGroupId. For EC2 Security Groups, use EC2SecurityGroupOwnerId and either EC2SecurityGroupName or EC2SecurityGroupId.
Type: String

Required: No

Update requires: replacement (p. 34)

Route 53 AliasTarget Property Type

The AliasTarget is an embedded property of the AWS::Route53::RecordSet (p. 282) type.

For more information on alias resource record sets, see Creating Alias Resource Record Sets in the *Route 53 Developer Guide*.

The following properties are available with the Route 53 AliasTarget property.

Property	Type	Required	Notes
HostedZoneId	String	Yes	The hosted zone name ID of the Load Balancer that is the target of the alias. You can specify this property using the GetAtt (p. 324) intrinsic function to get the CanonicalHostedZoneNameID property for the LoadBalancer resource you want.
DNSName	String	Yes	The DNS name of the Load Balancer that is the target of the alias. You can specify this property using the GetAtt (p. 324) intrinsic function to get the CanonicalHostedZoneName property for the LoadBalancer resource you want.

Amazon S3 Website Configuration Property Type

Website Configuration is an embedded property of the AWS::S3::Bucket (p. 286) type.

The following properties are available with the Website Configuration type.

Property	Type	Required	Notes
IndexDocument	String	No	The name of the index document. For more information, see Index Document Support.
ErrorDocument	String	No	The name of the error document. For more information, see Custom Error Document Support.

SNS Subscription Property Type

The SNS Subscription is an embedded property of the AWS::SNS::Topic (p. 288) type.

The following properties are available with the Amazon SNS subscription type.

Property	Type	Required	Notes
Endpoint	String	Yes	The subscription's endpoint (format depends on the protocol).
Protocol	String	Yes	The subscription's protocol.

Resource Attribute Reference

This section details the attributes that you can add to a resource to control additional behaviors and relationships.

DependsOn Attribute

The `DependsOn` attribute enables you to specify that the creation of a specific resource follows another. When you add a DependsOn attribute to a resource, you specify that that resource is created only after the creation of the resource specified in the DependsOn attribute. When you use wait conditions, you typically use the DependsOn attribute on another resource to determine when the wait condition goes into effect. For more information, see Creating Wait Conditions in a Template (p. 155). However, you can use the DependsOn attribute on any resource.

For example, the following template contains an AWS::EC2::Instance resource with a DependsOn attribute that specifies myDB, an AWS::RDS::DBInstance. When AWS CloudFormation creates this stack, it first creates myDB, then creates Ec2Instance.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Mappings" : {
        "RegionMap" : {
            "us-east-1" : { "AMI" : "ami-76f0061f" },
            "us-west-1" : { "AMI" : "ami-655a0a20" },
            "eu-west-1" : { "AMI" : "ami-7fd4e10b" },
            "ap-northeast-1" : { "AMI" : "ami-8e08a38f" },
            "ap-southeast-1" : { "AMI" : "ami-72621c20" }
        }
    },
    "Resources" : {
        "Ec2Instance" : {
            "Type" : "AWS::EC2::Instance",
            "Properties" : {
                "ImageId" : {
                   "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" },
 "AMI" ]
                }
            },
            "DependsOn" : "myDB"
        },
        "myDB" : {
            "Type" : "AWS::RDS::DBInstance",
            "Properties" : {
               "AllocatedStorage" : "5",
               "DBInstanceClass" : "db.m1.small",
               "Engine" : "MySQL",
               "EngineVersion" : "5.5",
               "MasterUsername" : "MyName",
               "MasterUserPassword" : "MyPassword"
            }
        }
    }
}
```

DeletionPolicy Attribute

The DeletionPolicy attribute enables you to specify how AWS CloudFormation handles the resource deletion. By adding a DeletionPolicy attribute to a resource, you can control how AWS CloudFormation handles the resource when its stack is deleted. By default, AWS CloudFormation deletes the resource if it has no DeletionPolicy attribute. You can specify *Retain* for AWS CloudFormation to leave a resource without deleting it. For resources that support snapshots, such as AWS::RDS::DBInstance and AWS::EC2::Volume, you can specify *Snapshot* for AWS CloudFormation to create a snapshot before deleting the resource.

For example, the following template contains an Amazon S3 bucket resource with a *Retain* deletion policy. When this stack is deleted, AWS CloudFormation will leave the bucket without deleting it.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
        "myS3Bucket" : {
            "Type" : "AWS::S3::Bucket",
            "DeletionPolicy" : "Retain"
        }
    }
}
```

The following table describes the options for the DeletionPolicy attribute.

Value	Description
Delete	Default. This policy directs AWS CloudFormation to delete the resource and all its content if applicable during stack deletion. You can add this deletion policy to any resource type.
Retain	This policy directs AWS CloudFormation to keep the resource without deleting the resource or its content--as applicable, during stack deletion. You can add this deletion policy to any resource type. Note that when AWS CloudFormation completes the stack deletion, the stack will be in Delete_Complete state; however, resources with a Retain policy will continue to exist and will continue to incur applicable charges until you delete those resources.

Value	Description
Snapshot	This policy is allowed only for resources that support snapshots: AWS::RDS::DBInstance and AWS::EC2::Volume. This policy directs AWS CloudFormation to create a snapshot for the resource before deleting the resource. Note that when AWS CloudFormation completes the stack deletion, the stack will be in the Delete_Complete state; however, the snapshots created with this policy will continue to exist and continue to incur applicable charges until you delete those snapshots.

Metadata Attribute

The Metadata attribute enables you to associate structured data with a resource. By adding a Metadata attribute to a resource, you can add data in JSON format to the resource declaration. In addition, you can use intrinsic functions (such as GetAtt (p. 324) and Ref (p. 326)), parameters, and pseudo parameters within the Metadata attribute to add those interpreted values. AWS CloudFormation does not validate the JSON in the Metadata attribute.

You can retrieve this data using the cfn-describe-stack-resource (p. 352) command or the DescribeStackResource action.

The following template contains an Amazon S3 bucket resource with a Metadata attribute.

```
{
    "AWSTemplateFormatVersion" : "2010-09-09",
    "Resources" : {
       "MyS3Bucket" : {
         "Type" : "AWS::S3::Bucket",
         "Metadata" : { "Object1" : "Location1",  "Object2" : "Location2" }
       }
    }
}
```

Intrinsic Function Reference

Topics

AWS CloudFormation provides several built-in functions that help you manage your stacks.

Fn::Base64

The intrinsic function `Fn::Base64` returns the Base64 representation of the input string. This function is typically used to pass encoded data to Amazon EC2 instances by way of the UserData property.

Declaration

"Fn::Base64" : {*valueToEncode*}

Parameters

valueToEncode
The string value you want to convert to Base64.

Return Value:

The original string, in Base64 representation.

Example

```
"Fn::Base64" : { "AWS CloudFormation" }
```

Fn::FindInMap

The intrinsic function `Fn::FindInMap` returns the value of a key from a mapping declared in the Mappings section.

Declaration

"Fn::FindInMap" : ["*MapName*", "*Key*", "*Value*"]

Parameters

MapName
The logical name of the mapping declared in the Mappings section that contains the key-value pair.

Key
The name of the mapping key whose value you want.

Value
The value for the named mapping key.

Return Value:

The map value. The previous example returns a string containing the value of MapValue from the MapKey in the specified mapping.

Example

```
"Fn::FindInMap" : [ "MyMap" , "MapKey", "MapValue"] }
```

Fn::GetAtt

The intrinsic function `Fn::GetAtt` returns the value of an attribute from a resource in the template.

Declaration

"Fn::GetAtt" : ["*`logicalNameOfResource`*", "*`attributeName`*"]

Parameters

logicalNameOfResource
: The logical name of the resource that contains the attribute you want.

attributeName
: The name of the resource-specific attribute whose value you want. See the resource's reference page for details about the attributes available for that resource type.

Return Value

The attribute value.

Example

This example returns a string containing the DNS name of the LoadBalancer with the logical name *`MyLB`*.

```
"Fn::GetAtt" : [ "MyLB" , "DNSName" ]
```

See Also

For more information about the values returned by Fn::GetAtt for a particular resource, see the **Return Values** section of the resource's reference page.

Fn::GetAZs

The intrinsic function `Fn::GetAZs` returns an array that lists all Availability Zones for the specified region.

Because customers have access to different Availability Zones, the intrinsic function `Fn::GetAZs` enables template authors to write templates that adapt to the calling user's access. This frees you from having to hard-code a full list of Availability Zones for a specified region.

Declaration

"Fn::GetAZs" : "*`region`*"

Parameters

region
: The name of the region for which you want to get the Availability Zones.
: You can use the *`AWS::Region`* pseudo parameter to specify the region in which the stack is created. Specifying an empty string is equivalent to specifying *`AWS::Region`*.

Return Value

The list of Availability Zones for the region.

Examples

```
{ "Fn::GetAZs" : "" }
```

```
{ "Fn::GetAZs" : "us-east-1" }
```

For both of the previous examples, AWS CloudFormation evaluates Fn::GetAZs to the following array—assuming that the user has created the stack in the us-east-1 region:

```
[ "us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d" ]
```

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.

Declaration

"Fn::Join" : ["*`delimiter`*", [*`comma-delimited list of values`*]]

Parameters

delimiter
The value you want to occur between fragments. The delimiter will occur between fragments only. It will not terminate the final value.

listOfValues
The list of values you want combined.

Return Value

The combined string.

Example

```
"Fn::Join" : [ ":", [ "a", "b", "c" ] ]
```

This example returns: "a:b:c".

Ref

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

- When you specify a parameter's logical name, it returns the value of the parameter.
- When you specify a resource's logical name, it returns a value that you can typically use to refer to that resource.

When you are declaring a resource in a template and you need to specify another template resource by name, you can use the `Ref` to refer to that other resource. In general, `Ref` returns the name of the resource. For example, a reference to an AWS::AutoScaling::AutoScalingGroup (p. 190) returns the name of that Auto Scaling group resource.

For some resources, an identifier is returned that has another significant meaning in the context of the resource. An AWS::EC2::EIP (p. 220) resource, for instance, returns the IP address, and an AWS::EC2::Instance (p. 222) returns the instance ID.

At the bottom of this topic, there is a table that lists the values returned for many common resource types. More information about `Ref` return values for a particular resource or property can be found in the documentation for that resource or property.

Tip

You can also use `Ref` to add values to Output messages.

Declaration

"Ref" : "*logicalName*"

Parameters

logicalName
The logical name of the resource or parameter you want to dereference.

Return Value

The value of the *MyInputParameter* parameter.

Example

The following resource declaration for an Elastic IP address needs the instance ID of an EC2 instance and uses the `Ref` function to specify the instance ID of the MyEC2Instance resource:

```
"MyEIP" : {
   "Type" : "AWS::EC2::EIP",
   "Properties" : {
      "InstanceId" : { "Ref" : "MyEC2Instance" }
   }
}
```

Resource Return Examples

This section lists sample values returned by `Ref` for particular AWS CloudFormation resources. For more information about `Ref` return values for a particular resource or property, refer to the documentation for that resource or property.

Resource Type	Reference Value	Example
AWS::AutoScaling::AutoScalingGroup (p. 190)	Name	mystack-myasgroup-NT5EUXTNTXXD
AWS::AutoScaling::LaunchConfiguration (p. 193)	Name	mystack-mylaunchconfig-1DDYF1E3B3I
AWS::AutoScaling::ScalingPolicy (p. 197)	Name	mystack-myaspolicy-1DDYF1E3B3I

Resource Type	Reference Value	Example
AWS::AutoScaling::Trigger (p. 198)	Name	mystack-mytrigger-7GQPDS5YHI5B
AWS::CloudFormation::Stack (p. 211)	Stack ID	[illegible]
AWS::CloudFormation::WaitCondition (p. 212)	Name	[illegible]
AWS::CloudFormation::WaitConditionHandle (p. 213)	Wait Condition Signal URL	[illegible]
AWS::CloudFront::Distribution (p. 213)	Distribution ID	E27LVI50CSW06W
AWS::CloudWatch::Alarm (p. 214)	Name	mystack-myalarm-3AOHFRGOXR5T
AWS::EC2::Volume (p. 247)	Volume ID	vol-3cdd3f56
AWS::EC2::VolumeAttachment (p. 247)	Name	mystack-myvola-ERXHJITXMRLT
AWS::EC2::EIP (p. 220)	Elastic IP Address	192.0.2.0
AWS::EC2::EIPAssociation (p. 221)	Name	mystack-myeipa-1NU3IL8LJ313N
AWS::EC2::Instance (p. 222)	Instance ID	i-636be302
AWS::EC2::SecurityGroup (p. 237)	Name	mystack-mysecuritygroup-QQB406M8FISX
AWS::EC2::SecurityGroupIngress (p. 238)	Name	mysecuritygroupingress
AWS::ElasticLoadBalancing::LoadBalancer (p. 258)	Name	mystack-myelb-1WQN7BJGDB5YQ
AWS::ElasticBeanstalk::Application (p. 257)	Name	mystack-myapplication-FM6BIXY7U8PK
AWS::ElasticBeanstalk::Environment (p. 257)	Name	mystack-myenv-LKGNQSFHO1DB
AWS::IAM::AccessKey (p. 262)	AccessKeyId	AKIAIOSFODNN7EXAMPLE
AWS::IAM::Group (p. 263)	GroupName	mystack-mygroup-1DZETITOWEKVO
AWS::IAM::User (p. 271)	UserName	mystack-myuser-1CCXAFG2H2U4D
AWS::RDS::DBInstance (p. 274)	Name	mystack-mydb-ea5ugmfvuaxg
AWS::RDS::DBSecurityGroup (p. 279)	Name	mystack-mydbsecuritygroup-1k5u5dxjb0nxs
AWS::S3::Bucket (p. 286)	Name	mystack-mys3bucket-1hbsmonr9mytq
AWS::SDB::Domain (p. 287)	Name	mystack-mysdbdomain-IVNAOZTDFVXL
AWS::SNS::Topic (p. 288)	Topic ARN	arn:aws:sns:us-east-1:803981987763:mystack-mytopic-NZJ5JSMVGFE
AWS::SQS::Queue (p. 288)	Queue URL	https://sqs.us-east-1.amazonaws.com/803981987763/aa4-MyQueue-Z5NOSZO2PZE9
Pseudo Parameter (p. 329)	AWS::Region	us-east-1
Pseudo Parameter (p. 329)	AWS::StackName	MyStack

Pseudo Parameters Reference

Pseudo Parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them the same as you do a parameter, as the argument for the `Ref` function. For example, the following fragment assigns the Value of the *`AWS::Region`* pseudo parameter:

```
"Outputs" {
    "MyStacksRegion" : {
        "Value" : { "Ref" : "AWS::Region" }
    }
}
```

The following pseudo parameters are supported by AWS CloudFormation:

Name	Description
AWS::Region	Returns a string representing the AWS Region in which the encompassing resource is being created.
AWS::StackName	Returns the name of the stack as specified with the `cfn-create-stack` command.

CloudFormation Helper Scripts Reference

Topics

AWS CloudFormation provides a set of Python helper scripts that you can use to install software and start services on an Amazon EC2 instance that you create as part of your stack. You can call the helper scripts directly from your template. The scripts work in conjunction with resource metadata that you define in the same template. The helper scripts run on the Amazon EC2 instance as part of the stack creation process.

The helper scripts are pre-installed on the latest versions of the Amazon Linux AMI. The helper scripts are also available from the Amazon Linux yum repository for use with other UNIX/Linux AMIs.

Currently, AWS CloudFormation provides the following helpers:

- cfn-init (p. 330): Used to retrieve and interpret the resource metadata, installing packages, creating files and starting services.
- cfn-signal (p. 333): A simple wrapper to signal a CloudFormation WaitCondition allowing you to synchronize other resources in the stack with the application being ready.
- cfn-get-metadata (p. 335): A wrapper script making it easy to retrieve either all metadata defined for a resource or path to a specific key or subtree of the resource metadata.
- cfn-hup (p. 336): A daemon to check for updates to metadata and execute custom hooks when the changes are detected.

These scripts are installed by default on the latest Amazon Linux AMI in /opt/aws/bin. They are also available in the Amazon Linux AMI yum repository for previous versions of the Amazon Linux AMI as well as via RPM for other Linux/Unix distributions. The scripts can also be installed on Microsoft Windows using Python for Windows.

The scripts are not executed by default. You must include calls to execute specific helper scripts.

The AWS CloudFormation helper scripts are available from the following locations:

- The latest version of the Amazon Linux AMI has the AWS CloudFormation helper scripts installed by default in /opt/aws/bin.
- The AWS helper scripts are available in the Amazon Linux AMI yum repository (the package name is aws-cfn-bootstrap) for previous versions of the Amazon Linux AMI.
- The helpers are also available in other formats:
  - https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.amzn1.noarch.rpm
  - https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz to install the helper scripts via the Python easy-install tools.
  - https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.zip
  - https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.msi for installation on Microsoft Windows.
- The source for the scripts is available at https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.src.rpm, which can be used for Linux distributions other than the Amazon Linux AMI.

Note

A complete list of available helper scripts and information regarding their use can be found on the AWS CloudFormation tools page: Bootstrapping Applications using AWS CloudFormation.

cfn-init

Description

The cfn-init helper script reads template metadata from the AWS::CloudFormation::Init key and acts accordingly to:

- Fetch and parse metadata from CloudFormation
- Install packages
- Write files to disk
- Enable/disable and start/stop services

For information about the template metadata that cfn-init uses, see AWS::CloudFormation::Init (p. 203).

Syntax

```
cfn-init --stack|-s stack.name.or.id \
         --resource|-r logical.resource.id \
         --region region
         --access-key access.key \
         --secret-key secret.key \
         --credential-file|-f credential.file \
         --configsets|-c config.sets \
         --url|-u service.url \
         -v
```

Options

Name	Description	Required
`-s, --stack`	Name of the Stack. Type: String Default: None Example: `-s { "Ref" : "AWS::StackName" },`	Yes
`-r, --resource`	The logical resource ID of the resource that contains the metadata. Type: String Example: `-r WebServerHost`	Yes
`--region`	The region to derive the CloudFormation URL from. Type: String Default: None Example: `--region ", { "Ref" : "AWS::Region" },`	No
`--access-key`	AWS Access Key for an account with permission to call DescribeStackResource on CloudFormation. Type: String Exmaple: `--access-key ", { "Ref" : "WebServerKeys" },` Condition: The credential file parameter supercedes this parameter.	Conditional
`--secret-key`	AWS Secret Key that corresponds to the specified AWS Access Key. Type: String Example: `--secret-key ", {"Fn::GetAtt": ["WebServerKeys", "SecretAccessKey"]},` Condition: The credential file parameter supercedes this parameter.	Conditional

Name	Description	Required
`-f, --credential-file`	A file that contains both a secret key and an access key. Type: String Condition: The credential file parameter supercedes the --access-key and --secret-key parameters.	Conditional
`-c, --configsets`	A comma-separated list of configsets to run (in order). Type: String Default: `default`	No
`-u, --url`	The CloudFormation service URL to hit. Type: String Condition: The credential file parameter supercedes the --access-key and --secret-key parameters.	No
`-v`	Verbose output. This is useful for debugging cases where cfn-init is failing to initialize. **Note** To debug initialization events, you should turn DisableRollback on. You can do this by using the CloudFormation console, selecting *Show Advanced Options*, and then setting "Rollback on failure" to "No". You can then SSH into the console and read the logs at /var/log/cfn-init.log.	No

Examples

The following snippet is associated with a resource named WebServer.

```
"/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" },
"    -r WebServer ",
"    --access-key ",  { "Ref" : "HostKeys" },
"    --secret-key ", {"Fn::GetAtt": ["HostKeys", "SecretAccessKey"]},
"    --region ", { "Ref" : "AWS::Region" }, \n",
```

Several AWS CloudFormation sample templates use cfn-init, including the following templates.

- LAMP: Single EC2 Instance with local MySQL database
- WordPress: Single EC2 Instance with local MySQL database
- Drupal: Single EC2 Instance with local MySQL database

cfn-signal

Description

You can use the cfn-signal helper script to pause the stack creation process. Use the cfn-signal script in conjunction with two AWS CloudFormation resources:

- AWS::CloudFormation::WaitCondition (p. 212)
- AWS::CloudFormation::WaitConditionHandle (p. 213)

For more information about how to use wait conditions in your template, see Creating Wait Conditions in a Template (p. 155).

Syntax

```
cfn-signal --success|-s signal.to.send \
        --reason|-r resource.status.reason \
        --data|-d data \
        --id|-i unique.id \
        --exit-code|-e exit.code \
        waitconditionhandle.url
```

Options

Name	Description	Required
`-s, --success`	if true, signal SUCCESS, else FAILURE. Type: Boolean Default: `true`	No
`-r, --reason`	A status reason for the resource event (currently only used on failure) - defaults to 'Configuration failed' if success is false. Type: String	No
`-d, --data`	Data to send back with the waitConditionHandle. Defaults to blank. Type: String Default: blank	No
`-i, --id`	The unique id to send back with the WaitConditionHandle. Type: String Default: The machine's Fully Qualified Domain Name (FQDN).	No

Name	Description	Required
`-e, --exit-code`	The error code from a process that can be used to determine success or failure. If specified, the --success parameter is ignored. Type: String Example: `--secret-key ", {"Fn::GetAtt": ["WebServerKeys", "SecretAccessKey"]},`	No
`waitconditionhandle.url`	A pre-signed URL that you can use to signal success or failure to an associated WaitCondition Type: String	Yes

Examples

Example 1

A common usage pattern is to use cfn-init and cfn-signal together. The cfn-signal call uses the return status of the call to cfn-init (using the $? shell construct). If the application fails to install, the WaitCondition will fail to create and the stack will rollback.

```
"MyInstance": {
  "Type": "AWS::EC2::Instance",
  "Metadata": {
    :
  },
  "Properties": {
    "ImageId" : "ami-12345678",
    "UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [
      "#!/bin/bash\n",
      "/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackName" },
      "         -r MyInstance ",
      "         --region ", { "Ref" : "AWS::Region" },
      "         --access-key ", { "Ref" : "MyKeys" },
      "         --secret-key ", {"Fn::GetAtt": ["MyKeys", "SecretAccessKey"]},

      "\n",
      "/opt/aws/bin/cfn-signal $? '", { "Ref" : "MyHandle" }, "'\n",
    ]]}}
  }
},
```

Example 2

The following snippet shows another common usage pattern in which a function calls cfn-signal. Wrapping cfn-signal in a function makes it easier to call cfn-signal from multiple places.

```
"# Helper function\n",
          "function error_exit\n",
          "{\n",
          "  /opt/aws/bin/cfn-signal -e 1 -r \"$1\" '", { "Ref" : "WaitHandle"
 }, "'\n",
          "  exit 1\n",
          "}\n",
```

The following snippet shows the error_exit function called in two commands.

```
"# Setup MySQL, create a user and a database\n",
"mysqladmin -u root password '", { "Ref" : "DBRootPassword" },
"' || error_exit 'Failed to initialize root password'\n",

"mysql -u root --password='", { "Ref" : "DBRootPassword" },
"' < /tmp/setup.mysql || error_exit 'Failed to initialize database'\n",
```

Examples in Sample Templates

Several AWS CloudFormation sample templates use cfn-signal, including the following templates.

- LAMP: Single EC2 Instance with local MySQL database
- WordPress: Single EC2 Instance with local MySQL database
- Drupal: Single EC2 Instance with local MySQL database

cfn-get-metadata

Description

You can use the cfn-get-metadata helper script to fetch a metadata block from CloudFormation and print it to standard out. You can also print a sub-tree of the metadata block if the you specify a key. However, only top-level keys are supported.

Syntax

```
cfn-get-metadata --access-key access.key \
                 --secret-key secret.key \
                 --credential-file|f credential.file \
                 --key|k key \
                 --stack|-s stack.name.or.id \
                 --resource|-r logical.resource.id \
                 --url|-u service.url \
                 --region region
```

Options

Name	Description	Required
`-s, --stack`	Name of the Stack. Type: String Default: None Example: `-s { "Ref" : "AWS::StackName" },`	Yes

Name	Description	Required
`-r, --resource`	The logical resource ID of the resource that contains the metadata. Type: String Example: `-r WebServerHost`	Yes
`--region`	The region to derive the CloudFormation URL from. Type: String Default: None Example: `--region ", { "Ref" : "AWS::Region" },`	No
`--access-key`	AWS Access Key for an account with permission to call DescribeStackResource on CloudFormation. Type: String Exmaple: `--access-key ", { "Ref" : "WebServerKeys" },` Condition: The credential file parameter supercedes this parameter.	Conditional
`--secret-key`	AWS Secret Key that corresponds to the specified AWS Access Key. Type: String Example: `--secret-key ", {"Fn::GetAtt": ["WebServerKeys", "SecretAccessKey"]},` Condition: The credential file parameter supercedes this parameter.	Conditional
`-f, --credential-file`	A file that contains both a secret key and an access key. Type: String Condition: The credential file parameter supercedes the --access-key and --secret-key parameters.	Conditional

cfn-hup

Description

The cfn-hup helper is a daemon that detects changes in resource metadata and runs user-specified actions when a change is detected. This allows you to make configuration updates on your running Amazon EC2 instances through the UpdateStack API action.

Syntax

```
cfn-hup --config|-c config.dir \
        --no-daemon \
        --verbose|-v
```

Options

Name	Description	Required
`--no-daemon`	If `true`, run cfn-hup once and exit. Type: Boolean Default: `false`	Yes
`-v, --verbose`	If `true`, use verbose mode. Type: Boolean Default: `false`	No

cfn-hup.conf Configuration File

The cfn-hup.conf file stores the name of the stack and the AWS credentials that the cfn-hup daemon targets. The cfn-hup.conf file uses the following format:

```
[main]
stack=<stack-name-or-id>
credential-file=<credential-file>
```

Name	Description	Required
`stack`	A stack name or ID. Type: String	Yes

`credential-file`	An owner-only credential file, in the same format used for the command line tools.
	
	Example:
	
	`AWSAccessKeyId=<your access key Id>`
	`AWSSecretKey=<your secret key>`
		Yes

`region`	The name of the AWS region containing the stack. Example: `us-east-1`	No

Name	Description	Required
`interval`	interval used to check for changes to the resource metadata in minutes Type: Number Default: `10`	No

hooks.conf Configuration File

The user actions that the cfn-hup daemon calls periodically are defined in the hooks.conf configuration file. The hooks.conf file uses the following format:

```
[hookname]
triggers=post.add|post.update|post.remove
path=Resources.<logicalResourceId> (.Metadata|PhysicalResourceId)(.optional
Metadatapath)
action=<arbitrary shell command>
runas=<runas user>
```

When the action is run, it is run in a copy of the current environment (that cfn-hup is in), with CFN_OLD_METADATA set to the previous value of path, and CFN_NEW_METADATA set to the current value.

The hooks configuration file is loaded at cfn-hup daemon startup only, so new hooks will require the daemon to be restarted. A cache of previous metadata values is stored at /var/lib/cfn-hup/data/metadata_db (not human readable)—you can delete this cache to force cfn-hup to run all post.add actions again.

Name	Description	Required
`hookname`	A unique name for this hook Type: String	Yes
`triggers`	A comma-delimited list of conditions to detect. Valid Values: `post.add	post.update	post.remove`  Example: `post.add, post.update`	Yes

Name	Description	Required
`path`	The path to the metadata object. Supports an arbitrarily deep path within the Metadata block. **Path format options** • Resources.*`<LogicalResourceId>`*—monitor the last updated time of the resource, triggering on any change to the resource. • Resources.*`<LogicalResourceId>`*.PhysicalResourceId—monitor the physical ID of the resource, triggering only when the associated resource identity changes (such as a new EC2 instance). • Resources.*`<LogicalResourceId>`*.Metadata(*`.optional path`*)—monitor the metadata of a resource for changes (a metadata subpath may be specified to an arbitrarily deep level to monitor specific values).	Yes
`action`	An arbitrary shell command that is run as given.	Yes
`runas`	A user to run the commands as. Cfn-hup uses the su command to switch to the user.	Yes

hooks.d Directory

To support composition of several applications deploying change notification hooks, cfn-hup supports a directory named hooks.d that is located in the hooks configuration directory. You can place one or more additional hooks configuration files in the hooks.d directory. The additional hooks files must use the same layout as the hooks.conf file.

The cfn-hup daemon parses and loads each file in this directory. If any hooks in the hooks.d directory have the same name as a hook in hooks.conf, the hooks will be merged (meaning hooks.d will overwrite hooks.conf for any values that both files specify).

Command Line Tools Reference

AWS CloudFormation provides a number of tools that enable you to manage your CloudFormation stacks directly from your computer's command-line interface (CLI). This section describes each tool in detail.

Note

The CloudFormation *CLI tools*, which are run from your local system, are distinct from the *helper scripts*, which are installed on Amazon EC2 instances that you create with CloudFormation. To learn more about the AWS CloudFormation helper scripts, see the CloudFormation Helper Scripts Reference (p. 329).

For information about downloading and installing the AWS CloudFormation CLI, see Installing the AWS CloudFormation Command Line Interface (CLI) (p. 59).

Topics

Common Options for CloudFormation CLI Tools

Many CLI tools described in this section accept optional parameters described in this topic. For details, see the reference page for the tool you are using, or type `toolname --help` on the command-line, where *toolname* is the name of the command (such as cfn-describe-stacks).

--aws-credential-file *VALUE*
: Location of the file with your AWS credentials. This value can be set automatically by using the environment variable 'AWS_CREDENTIAL_FILE'.

--connection-timeout *VALUE*
: Specify a connection timeout *VALUE* (in seconds). The default value is '30'.

--delimiter *VALUE*
: Specify the delimiter to use when displaying delimited (long) results.

--headers
: If you are displaying tabular or delimited results, it includes the column headers. If you are showing xml results, it returns the HTTP headers from the service request, if applicable. This setting is off by default.

-I, --access-key-id *VALUE*
: Specify *VALUE* as the AWS Access ID to use.

-K, --ec2-private-key-file-path *VALUE*
: The private key file path. This value can be set automatically by using the environment variable 'EC2_PRIVATE_KEY'.

--region *VALUE*
: Specify region *VALUE* as the web service region to use. This value can be set by using the environment variable 'EC2_REGION'.

-S, --secret-key *VALUE*
: Specify *VALUE* as the AWS Secret Key to use.

--show-empty-fields
: Show empty fields and rows, using a "(nil)" value. The default is to suppress empty fields or columns.

--show-request
: Displays the URL the tools used to call the AWS Service. The default value is 'false'.

--show-table, --show-long, --show-xml, --quiet
: Specify how the results are displayed: tabular, delimited (long), xml, or no output (quiet). Tabular shows a subset of the data in fixed column-width form, while long shows all of the returned values delimited by a character. The xml is the raw return from the service, while quiet suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url *VALUE*
: This option will override the URL for the service call with *VALUE*. This value can be set by using the environment variable 'AWS_CLOUDFORMATION_URL'.

cfn-create-stack

Description

Create a new stack from a template stored in a local file or in an Amazon S3 bucket. If any resource defined in the template cannot be created, the stack will be rolled back, deleting all resources created up to that point. This behavior can be disabled by using the --disable-rollback option.

Syntax

`cfn-create-stack` ***`StackName`*** `common-options (p. 340)` ***`cfn-create-stack-options`***

The *`cfn-create-stack-options`* that you can specify are described in the **Options** section of this topic.

Options

Name	Description	Required
`StackName`	Name of the Stack. You can also set this value using "--stack-name". Type: String Default: None Example: websrv	Yes
`-d, --disable-rollback`	Flag to disable rollback of created resources when failures are encountered during stack creation. The default value is `false`. Type: String Default: None	No
`--template-file` *`VALUE`*	Path to the file that contains the template. Type: String Default: None	No
`-n, --notification-arns` *`VALUE1, VALUE2, VALUE3...`*	SNS ARNs to receive notification about the stack. Type: String Default: None	No
`-p, --parameters` *`"key1=value1; key2=value2 ..."`*	Parameter values used to create the stack. Type: String Default: None	No

Name	Description	Required
`-t, --timeout` *`VALUE`*	Stack creation timeout in minutes. Type: String Default: None	No
`--tag` *`"Key=value; Value=value"`*	A set of user-defined Tags to associate with this stack, represented by key/value pairs. This option can be repeated to specify a number of such pairs. Type: String Example: `--tag "Key=Purpose; Value=Testing" --tag "Key=endpoint; Value=us-east-1"`	No
`-u, --template-url` *`VALUE`*	Path of the URL that contains the template. This must be a reference to a template in an Amazon S3 bucket in the same region that the stack will be created in. Type: String Default: None	No
`-c, --capabilities` *`VALUE`*	The list of capabilities that you want to allow in the stack. If your template contains IAM resources, you must specify the CAPABILITY_IAM value for this parameter; otherwise, this action returns an InsufficientCapabilities error. IAM resources are the following: AWS::IAM::AccessKey (p. 262), AWS::IAM::Group (p. 263), AWS::IAM::Policy (p. 266), AWS::IAM::UserToGroupAddition (p. 273), and AWS::IAM::User (p. 271). For more information about using IAM resources in templates, see Controlling User Access with AWS Identity and Access Management (p. 159). Type: String Valid Values: CAPABILITY_IAM Default: None	No

Output

This command returns a table that contains the following:

- STACK_ID

  Unique Identifier for the Stack

AWS CloudFormation displays errors on stderr.

Examples

Example Request

Create a Stack named `example-stack` using template located in file `example-template-file` and specifying template parameters *`param1`* and *`param2`*.

```
PROMPT> cfn-create-stack example-stack --template-file example-template-file -
-parameters "param1=foo;param2=bar"

arn:aws:aws21:us-east-1:123456789:stack/aaf549a0-a413-11df-adb3-
5081b3858e83/example-stack
```

cfn-delete-stack

Description

Delete an existing stack

Syntax

`cfn-delete-stack` ***`StackName`*** `common-options (p. 340)`

Options

Name	Description	Required
`StackName`	The name of the stack you want to delete. You can also set this value using *`--stack-name`*. Type: String Default: None	Yes
`--force`	Delete the stack without a confirmation prompt. Type: String Default: `false` Valid value: `true` | `false`	No

Output

The command returns no output. Use cfn-describe-stacks to discover the stack delete status.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example deletes a stack named `example-stack`.

```
PROMPT> cfn-delete-stack example-stack

Warning: Deleting a stack will lead to deallocation of all of the stack's
                        resources. Are you sure you want to delete this stack?
 [Ny] y
```

Related Operations

- cfn-describe-stacks (p. 347)
- cfn-describe-stack-events (p. 350)

- cfn-list-stacks (p. 358)

cfn-describe-stacks

Describe one or more running stacks.

Note

cfn-describe-stacks will only list stacks that are running, or are in the process of being created or deleted. If you want to list stacks that have already been deleted, use **cfn-list-stacks**.

Syntax

`cfn-describe-stacks` [`options`]

Options

`stackname`
: The name of the stack you'd like information about. If this option isn't specified, cfn-describe-stacks will list all stacks for the account.

--aws-credential-file *VALUE*
: Location of the file with your AWS credentials. This value can be set automatically by using the environment variable 'AWS_CREDENTIAL_FILE'.

--connection-timeout *VALUE*
: Specify a connection timeout *VALUE* (in seconds). The default value is '30'.

--delimiter *VALUE*
: Specify the delimiter to use when displaying delimited (long) results.

--headers
: If you are displaying tabular or delimited results, it includes the column headers. If you are showing xml results, it returns the HTTP headers from the service request, if applicable. This setting is off by default.

-I, --access-key-id *VALUE*
: Specify *VALUE* as the AWS Access ID to use.

-K, --ec2-private-key-file-path *VALUE*
: The private key file path. This value can be set automatically by using the environment variable 'EC2_PRIVATE_KEY'.

--region *VALUE*
: Specify region *VALUE* as the web service region to use. This value can be set by using the environment variable 'EC2_REGION'.

-S, --secret-key *VALUE*
: Specify *VALUE* as the AWS Secret Key to use.

--show-empty-fields
: Show empty fields and rows, using a "(nil)" value. The default is to suppress empty fields or columns.

--show-request
: Displays the URL the tools used to call the AWS Service. The default value is 'false'.

--show-table, --show-long, --show-xml, --quiet
: Specify how the results are displayed: tabular, delimited (long), xml, or no output (quiet). Tabular shows a subset of the data in fixed column-width form, while long shows all of the returned values delimited by a character. The xml is the raw return from the service, while quiet suppresses all standard output. The default is tabular, or 'show-table'.

-U, --url *VALUE*
: This option will override the URL for the service call with *VALUE*. This value can be set by using the environment variable 'AWS_CLOUDFORMATION_URL'.

Output

The command returns a table with the following columns:

NAME
Identifies the entry as a stack

STACK_ID
Unique identifier for the stack. This column appears only in the `--show-long` view.

STATUS
The current status of the stack.

One of: *`IN_PROGRESS`* | *`CREATE_FAILED`* | *`CREATE_COMPLETE`* | *`DELETE_IN_PROGRESS`* | *`DELETE_FAILED`* | *`DELETE_COMPLETE`*.

STATUS_REASON
Status reason. This column appears only in the `--show-long` view.

DESCRIPTION
Description from template used to create stack.

PARAMETERS
Parameters provided during stack creation. This column appears only in the `--show-long` view.

OUTPUTS
Outputs returned from stack creation.

CREATED_TIME
Time when the stack was created.

LAST_UPDATED_TIME
Time when the stack was last updated.

DISABLE_ROLLBACK
Disable rollback of created resources if the stack creation fails. Normally, when a stack fails, all its resources are deleted. This column indicates whether a resource will not be deleted if the stack fails. This column appears only in the `--show-long` view.

TIMEOUT_IN_MINUTES
Stack creation timeout. This column appears only in the `--show-long` view.

NOTIFICATION_ARNS
SNS ARNs to receive notification about the stack. This column appears only in the `--show-long` view.

TAGS
A set of user-defined Tags associated with this stack. This column appears only in the `--show-long` or `--show-xml` views.

Examples

List All Running Stacks

To list all running stacks, simply type **cfn-list-stacks** with no arguments:

```
$ cfn-describe-stacks

STACK  myGollumWiki       CREATE_COMPLETE     AWS CloudFormation Sample Template
 Gol... you create a stack from this template.  WebsiteURL=http://myurl.amazon
aws.com  2012-07-07T18:11:58Z
STACK  myMultiAzWordPress  CREATE_IN_PROGRESS  AWS CloudFormation Sample Template
```

```
Wor... you create a stack from this template.
                            2012-07-07T18:19:36Z
```

List Information About a Particular Running Stack

By supplying the stack ID to **cfn-list-stacks**, you can get information specific to that stack:

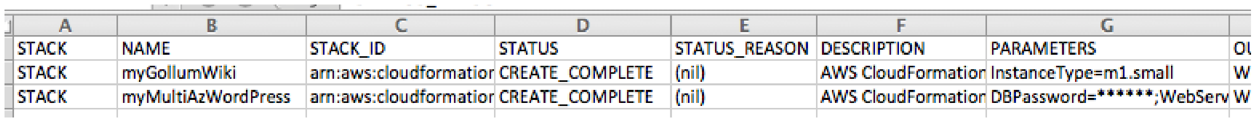
```
$ cfn-describe-stacks myGollumWiki

STACK  myGollumWiki       CREATE_COMPLETE    AWS CloudFormation Sample Template
 Gol... you create a stack from this template.  WebsiteURL=http://myurl.amazon
aws.com  2012-07-07T18:11:58Z
```

Get a CSV File Suitable for Importing to a Spreadsheet

By supplying the --headers and --show-long arguments, you can create output suitable for importing to a spreadsheet:

```
$ cfn-describe-stacks --headers --show-long >output.csv
```

The resulting file, `output.csv`, will contain a comma-separated-value listing of the output columns and corresponding data for all running stacks. For example:



A	B	C	D	E	F	G	
STACK	NAME	STACK_ID	STATUS	STATUS_REASON	DESCRIPTION	PARAMETERS	OU
STACK	myGollumWiki	arn:aws:cloudformation	CREATE_COMPLETE	(nil)	AWS CloudFormation	InstanceType=m1.small	W
STACK	myMultiAzWordPress	arn:aws:cloudformation	CREATE_COMPLETE	(nil)	AWS CloudFormation	DBPassword=******;WebServ	W

Related Operations

- cfn-create-stack (p. 342)
- cfn-describe-stack-events (p. 350)
- cfn-describe-stack-resources (p. 354)
- cfn-delete-stack (p. 345)
- cfn-list-stacks (p. 358)

cfn-describe-stack-events

Description

Describe the events for one or more stacks.

Syntax

`cfn-describe-stack-events` ***`StackName`*** `common-options (p. 340)`

Options

Name	Description	Required
`StackName`	The name of the stack you want to describe. You can also set this value using `--`*`stack-name`*. If you do not specify this value, AWS CloudFormation returns event information on all your running stacks. Type: String Default: None	No

Output

The command returns a table that contains the following information:

- `EVENT_ID`

  Unique identifier for the event.
- `STACK_NAME`

  The name of the stack the event correspond to
- `STACK_ID`

  The identifier of the stack the event correspond to.
- `LOGICAL_ID`

  Logical identifier of the resource.
- `PHYSICAL_ID`

  Physical identifier for the resource.
- `RESOURCE_PROPERTIES`

  Properties of the resources.
- `RESOURCE_TYPE`

  AWS type of the resource.
- `EVENT_TIME`

  Time when the event occurred.
- `RESOURCE_STATUS`

The status of the resource associated with the event.

- `RESOURCE_STATUS_REASON`

  More information about the status of the resource associated with the event.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example returns the event information for the stack named MyExampleStack

```
PROMPT> cfn-describe-stack-events MyExampleStack

STACK_EVENT     EVENT_ID    STACK_NAME      STACK_ID        LOGICAL_ID
PHYSICAL_ID                        RESOURCE_TYPE     RESOURCE_PROPERTIES
EVENT_TIME          RESOURCE_STATUS
STACK_EVENT     7163741d-e920-4dac-b4a4-a0ac0e35a20a     2     f9501d75-5402-
4bf6-b5d8-84e838feb5f5     myEC2SGrpOne    2-myEC2SGrpOne-496118474
{"GroupDescription":"instance access for load balancer"}    AWS::SecurityGroup
    2010-06-22T22:05:27Z     CREATE_IN_PROGRESS
```

Related Operations

- cfn-create-stack (p. 342)
- cfn-describe-stack-resources (p. 354)
- cfn-delete-stack (p. 345)

cfn-describe-stack-resource

Description

Returns the description for the specified resource in the specified stack.

Note

The cfn-describe-stack-resource command returns information on deleted stacks for 90 days after they have been deleted.

Syntax

```
cfn-describe-stack-resource StackName common-options (p. 340)
```

Options

Name	Description	Required
StackName	The name of the stack containing the resource whose description you want to view. You can also set this value using `-s` or `--stack-name`. Type: String Default: None	Yes
LogicalResourceId	The logical resource ID for the resource whose description you want to view. You can also set this value using `-l` or `--logical-resource-id`. Type: String Default: None	Yes

Output

The command returns a table that contains the following information:

- LOGICAL_ID

  Logical identifier for the resource.
- PHYSICAL_ID

  Physical identifier for the resource.
- TYPE

  Type of the resource.
- LAST_UPDATED_TIMESTAMP

Time when the resource's status was last updated.
- STATUS

  The current status of the resource.
- STATUS_REASON

  More information about the resource status. To display this value, you must specify --show-long.
- METADATA

  The JSON format content of the `Metadata` attribute declared for the resource. To display this value, you must specify --show-long or --show-xml. For more information, see Metadata Attribute (p. 323).

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example returns the example-lb1 resource in the example-1 stack.

```
PROMPT> cfn-describe-stack-resource example-1 example-lb1

STACK_RESOURCE     LOGICAL_ID      PHYSICAL_ID                        TYPE
                      LAST_UPDATED_TIMESTAMP      STATUS
STACK_RESOURCE     example-lb1     stack-1-example-lb1-2052137016
AWS::LoadBalancer              2011-07-11T15:39:56Z    CREATE_COMPLETE
```

Related Operations

- cfn-create-stack (p. 342)
- cfn-describe-stack-events (p. 350)
- cfn-list-stack-resources (p. 356)

cfn-describe-stack-resources

Description

Describe one or more member resources for one, or all of your running stacks.

If you do not provide either a stack or resource id, information for all running stacks and resources will be returned, up to a limit of 100 records.

Note

To list more than 100 resources use cfn-list-stack-resources (p. 356) instead.

Syntax

`cfn-describe-stack-resources` ***`StackName`*** `common-options (p. 340)`

Options

Name	Description	Required
-l, --logical-resource-id **VALUE**	The logical resource ID for a particular resource to describe. If you do not specify either a logical or physical resource ID, then all resources will be returned. Type: String Default: None	No
-p, --physical-resource-id **VALUE**	The physical resource ID for a particular resource to describe. If you do not specify either a logical or physical resource ID, then all resources will be returned. Type: String Default: None	No
-s, --stack-name	The name of the stack whose resources you want to describe. If you do not specify a stack, then resources for all of your running stacks and up to 90 days of your deleted stacks will be returned, up to a total of 100 resources. Type: String Default: None	No

Output

The command returns a table that contains the following information:

- LOGICAL_ID

  Logical identifier for the resource
- PHYSICAL_ID

  Physical identifier for the resource
- TYPE

  Type of the resource
- CREATED_TIME

  Time when the resource's status changed to `CREATE_COMPLETE`
- STATUS

  The current status of the resource.
- STATUS_REASON

  More information about the resource status.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example describes the resources in the example-1 stack.

```
PROMPT> cfn-describe-stack-resources example-1

STACK_RESOURCE     LOGICAL_ID      PHYSICAL_ID                          TYPE
                      CREATED_TIME        STATUS
STACK_RESOURCE     example-lb1     stack-1-example-lb1-2052137016
AWS::LoadBalancer              2009-04-15:39:56Z   CREATE_COMPLETE
STACK_RESOURCE     example-lc1     stack-1-example-lc1-2052137016
AWS::LaunchConfiguration       2009-04-16:45:00Z   CREATE_COMPLETE
```

Related Operations

- cfn-create-stack (p. 342)
- cfn-describe-stack-events (p. 350)
- cfn-describe-stack-resources (p. 354)

cfn-list-stack-resources

Description

Returns descriptions for all resources of the specified stack.

Note

The cfn-list-stack-resources command returns information on deleted stacks for 90 days after they have been deleted.

Syntax

`cfn-list-stack-resources` ***`StackName`*** `common-options (p. 340)`

Options

Name	Description	Required
`StackName`	The name of the stack. Alternatively, you can specify the stack ID. You can also set this value using *`-s`* or *`--stack-name`*. Type: String Default: None	Yes

Output

The command returns a table that contains the following information:

- LOGICAL_ID

  The logical name of the resource specified in the template.
- PHYSICAL_ID

  The name or unique identifier that corresponds to the physical instance ID of the resource.
- TYPE

  Type of the resource.
- LAST_UPDATED_TIMESTAMP

  Time when the resource's status was last updated.
- STATUS

  The current status of the resource.
- STATUS_REASON

  More information about the resource status. To display this value, you must specify --show-long or --show-xml.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example lists the resources in the example-1 stack.

```
PROMPT> cfn-list-stack-resources example-1

STACK_RESOURCE     LOGICAL_ID      PHYSICAL_ID                       TYPE
                      LAST_UPDATED_TIMESTAMP      STATUS
STACK_RESOURCE     example-lb1     stack-1-example-lb1-2052137016
AWS::LoadBalancer              2011-07-11T15:39:56Z    CREATE_COMPLETE
STACK_RESOURCE     example-lc1     stack-1-example-lc1-2052137016
AWS::LaunchConfiguration       2011-07-11T15:45:00Z        CREATE_COMPLETE
```

Related Operations

- cfn-create-stack (p. 342)
- cfn-describe-stack-events (p. 350)
- cfn-describe-stack-resource (p. 352)

cfn-list-stacks

Description

List one or more running or deleted stacks filtered by status.

Note

The cfn-list-stacks command returns information on deleted stacks for 90 days after they have been deleted.

Syntax

`cfn-list-stacks` ***`stack-status`*** **`common-options (p. 340)`**

Options

This command has the following options.

Name	Description	Required
`stack-status`	Filters for the ending status on the deleted stacks you want to list. If you do not specify this value, AWS CloudFormation only returns information stacks that ended with the `DELETE_COMPLETE` status. You can specify multiple status values for `--stack-status` if you separate them by a comma. Type: String Default: `DELETE_COMPLETE` valid Values: *`CREATE_IN_PROGRESS`* | *`CREATE_COMPLETE`* | *`CREATE_FAILED`* | *`DELETE_IN_PROGRESS`* | *`DELETE_COMPLETE`* | *`DELETE_FAILED`* | *`ROLLBACK_IN_PROGRESS`* | *`ROLLBACK_COMPLETE`* | *`ROLLBACK_FAILED`*	No

Output

The command returns a table that contains the following information:

- `STACK_ID`

  Unique identifier for the stack.
- `NAME`

  The stack name.
- `CREATED_TIME`

  Time when the stack was created.
- `STATUS`

The status of the stack when it was deleted.

One of *CREATE_IN_PROGRESS* | *CREATE_COMPLETE* | *CREATE_FAILED* | *DELETE_IN_PROGRESS* | *DELETE_COMPLETE* | *DELETE_FAILED* | *ROLLBACK_IN_PROGRESS* | *ROLLBACK_COMPLETE* | *ROLLBACK_FAILED*.

- TEMPLATE_DESCRIPTION

  Description of the template used to create the stack.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example returns information on stacks that have the CREATE_COMPLETE status, and stacks which were deleted (within the past 90 days) with the status DELETE_COMPLETE.

```
PROMPT> cfn-list-stacks --stack-status CREATE_COMPLETE,DELETE_COMPLETE
STACK  STACK_ID
NAME           CREATION_TIME         STATUS     TEMPLATE_DESCRIPTION
STACK  arn:aws:cloudformation:us-east-1:123456789012:stack/mut2/83327aa0-863
mut2           2011-05-24T19:53:36Z  CREATE_COMPLETE   TestTemplate556
STACK  arn:aws:cloudformation:us-east-1:897345613465:stack/mut4/45627ax0-980
mut2           2011-05-24T19:53:36Z  DELETE_COMPLETE   TestTemplate559
```

Related Operations

- cfn-describe-stacks (p. 347)
- cfn-delete-stack (p. 345)

cfn-update-stack

Description

Update an existing stack from a template stored in a local file or in an Amazon S3 bucket. If any resource defined in the updated template cannot be updated, the stack will be rolled back to the stack configuration of the previous template.

For detailed information on creating an update template, updating a stack, and monitoring its progress, see Updating AWS CloudFormation Stacks (p. 33).

Syntax

```
cfn-update-stack StackName common-options (p. 340)
```

Options

Name	Description	Required
StackName	Name or stack ID of the stack to update. You can also set this value using "--stack-name". Type: String Default: None Example: websrv	Yes
`--template-file` *VALUE*	Path to the file that contains the template with the updated stack information. Type: String Default: None	No
`-u, --template-url` *VALUE*	Path of the URL that contains the template with the updated stack information. This must be a reference to a template in an Amazon S3 bucket in the same region that the stack will be created in. Type: String Default: None	No
`-p, --parameters` *"key1=value1; key2=value2 ..."*	Parameter values used to update the stack. Type: String Default: None	No

Name	Description	Required
`-c, --capabilities` *`VALUE`*	The list of capabilities that you want to allow in the stack. If your stack contains IAM resources, you must specify the CAPABILITY_IAM value for this parameter; otherwise, this action returns an InsufficientCapabilities error. IAM resources are the following: AWS::IAM::AccessKey (p. 262), AWS::IAM::Group (p. 263), AWS::IAM::Policy (p. 266), UserToGroupAddition (p. 273), and AWS::IAM::User (p. 271). For more information about using IAM resources in templates, see Controlling User Access with AWS Identity and Access Management (p. 159). Type: String Valid Values: CAPABILITY_IAM Default: None	No

Output

This command returns a table that contains the following:

- STACK_ID

  Unique Identifier for the stack

AWS CloudFormation displays errors on stderr.

Examples

Example Request

Update a Stack named `example-stack` using template located in file `example-template-file` and specifying template parameters *`param1`* and *`param2`*.

```
PROMPT> cfn-update-stack example-stack --template-file example-template-file -
-parameters "param1=foo;param2=bar"

arn:aws:aws21:us-east-1:123456789:stack/aaf549a0-a413-11df-adb3-
5081b3858e83/example-stack
```

cfn-get-template

Description

Get the template used to create a running or deleted stack.

This command will return the text of the template used to create a stack.

Note

The cfn-get-template command returns template information from a deleted stack for 90 days after the stack has been deleted.

Syntax

`cfn-get-template` ***`StackName`*** `common-options (p. 340)`

Options

Name	Description	Required
`StackName`	The name of the stack whose template you want to get. You can also set this value using `--`*`stack-name`*. Type: String Default: None	No

Output

The command returns the template body, enclosed in double quotes.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This example gets the template used with the example-template stack.

```
PROMPT> cfn-get-template example-template

"
  {
    "AWSTemplateFormatVersion" : "2010-04-01",
    "Resources" : {
      "myEC2SGrpOne" : {
        "Type" : "AWS::SecurityGroup","Version" : "2009-11-30","Properties" :
{
          "GroupDescription" : "foo"
        }
```

```
          }
        }
    }
"
```

Related Operations

- cfn-create-stack (p. 342)

cfn-validate-template

Description

Do a static evaluation of a template.

Note

This command checks the syntatic correctness of the template. It does not do a deep check of resources you have declared. For example, if you have specified an invalid region name, cfn-validate-template will not detect that. Declaration errors of that type can be discovered only creating a stack with the template.

Syntax

`cfn-validate-template {-f` ***`VALUE`*** **`| -u`** ***`VALUE`*** **`}`** **`common-options (p. 340)`**

Options

Name	Description	Required
`-f, --template-file`	The local path and filename to the template to validate Type: String Default: None One of `-f` or `-u` must be specified, but not both.	Conditional
`-u, --template-url`	The URL of the template to validate Type: String Default: None One of `-u` or `-f` must be specified, but not both.	Conditional

Output

If the template is valid, the command returns information about capabilties and parameters defined in the template.

If the template contains capabilities (currently, only IAM capabilities are detected), it returns the following capability information:

```
CAPABILITIES_REASON  REASON
CAPABILITIES_REASON  Template contains IAM resources
CAPABILITIES  CAPABILITY_NAME
CAPABILITIES  CAPABILITY_IAM
```

If the template contains IAM resource and you use that template with the cfn-create-stack (p. 342) or cfn-update-stack (p. 360) commands, you must specify the CAPABILITY_IAM value for the --capabilities parameter; otherwise, this command returns an InsufficientCapabilities error.

If the template contains parameters, the command returns a table that contains the following parameter information. (If a valid template has no parameters, no output is returned.)

- PARAMETER_NAME

  Parameter name
- DEFAULT_VALUE

  If the parameter has a default value, that value is reported.
- NOECHO

  True if the parameter is declared with the `NOECHO` property.
- DESCRIPTION

  Value of the Description property for the parameter.

AWS CloudFormation displays errors on stderr.

Examples

Example Request

This reports the validity for the local file `example-template-file.template`.

```
PROMPT> cfn-validate-template --template-file example-template-file.template

PARAMETERS  PARAMETER_NAME     DEFAULT_VALUE          NOECHO
PARAMETERS  WordPressUser      admin                  false
PARAMETERS  WordPressDBName    wordpressdb            false
PARAMETERS  WordPressPwd       password               true
PARAMETERS  AvailabilityZones  us-east-1a,us-east-1b  false
PARAMETERS  GroupSize          2                      false
PARAMETERS  InstanceType       m1.small               false
```

Related Operations

- cfn-get-template (p. 362)

Glossary

chargeable resources	AWS CloudFormation is a free AWS product. However, in a AWS CloudFormation stack, AWS resources which have been created incur charges. The amount charged depends on the usage load. We recommend you use the Amazon Web Services Simple Monthly Calculator at http://calculator.s3.amazonaws.com/calc5.html to estimate your cost prior to creating your stacks.
CommaSeparatedList, comma separated list, comma delimited list	Some AWS CloudFormation properties and intrinsic functions take a CommanSeparatedList as an argument. The following shows the `MyValue` parameter being assigned a CommaSeparatedList: `"MyValue" : "'one','two','three'"`
creating	When you create a AWS CloudFormation stack, you put it and all its member resources into operation. When a resource has been created, it begins to incur charges.
creation sequence	The member resources of each AWS CloudFormation stack are created and deleted in an order determined by the AWS CloudFormation service. Due to the dependent nature of some of the AWS resource, you cannot predict the order in which member resources are created or deleted. You use the `cfn-describe-stack-events` command to get information on the resources which have been created or deleted.
conditional parameter	See mapping.
AWS Resources	See resources.
deleting	When you delete a AWS CloudFormation stack, you take it and all its member resources out of operation. When a resource has been deleted, it no longer incurs charges.
deletion sequence	See creation sequence.
describing stacks, describing resources	To get information about your running stacks, you decribe them using the `cfn-describe-stacks` and `cfn-describe-stack-resources` commands.
Description property	AWS CloudFormation adds a Description property to parameters, resources, resource properties, mappings, and outputs, to enable you to document your template elements.

endpoint	An endpoint is the DNS name or IP address of the server that receives an HTTP request. There are a number of AWS CloudFormation endpoints providing service to particular regions. For more information, see Regions and Endpoints in the *Amazon Web Services General Reference*
events	As AWS CloudFormation creates and deletes stacks and resources, it reports the progress of those operations with `cfn-describe-stack-events`. Each entry in the report is an event.
format version	See template format version.
function	See intrinsic function.
intrinsic function	AWS CloudFormation defines a number of intrinsic functions you can use within your templates, such as `Fn::GetAtt`. Arguments for intrinsic functions can be parameters, pseudo parameters, and the output of other intrinsic functions.
JSON	JSON (JavaScript Object Notation) is a lightweight data-interchange format. In order to be valid, all AWS CloudFormation templates must be compliant to the JSON standard. The `cfn-validate-template` command will check that your templates are JSON-compliant. For information on JSON, see http://www.json.org/.
logical name	A case-sensitive unique string within a template which identifies a resource, mapping, parameter, or output. In a AWS CloudFormation template, each parameter, resource, property, mapping, and output must be declared with a unique logical name. You use the logical name when dereferencing these items using the `Ref` function.
mapping	In conjunction with the intrinsic function `FN::FindInMap`, AWS CloudFormation mappings enable you to provide conditional values to your templates at runtime.
mapping attribute	When you use the intrinsic function `FN::FindInMap`, it returns the value of the mapping attribute for the matching or default map.
member resources	See resources.
NoEcho	AWS CloudFormation will report the names and values of your template parameters unless you declare the *`NoEcho`* property for the parameter. Declaring the *`NoEcho`* property causes the parameter value to be masked with asterisks in the report by the `cfn-describe-stacks` command.
object	See template object.
output	A AWS CloudFormation template provides a section where you can define values which are reported with the `cfn-describe-stack-resources` command. Outputs can be based on parameters, resources, literal strings, pseudo parameters, mappings, and intrinsic functions.
parameter	A AWS CloudFormation parameter provides a way to pass runtime values to your template. Parameter values can be assigned to resource properties and outputs.
physical name	When you create a stack, AWS CloudFormation assigns each resource a unique physical name. Some AWS CloudFormation commands will accept the physical name as a value with the `--physical-name` parameter.
properties	See resource properties.
property rules	In a AWS CloudFormation template, you declare resource properties, mappings, and output values according to a few JSON-compliant rules for markup.

pseudo parameter	AWS CloudFormation provides some predefined parameters, such as `AWS:StackName` that you can use in your templates. You can use pseudo parameters anywhere you can use a regular parameter.
rollback	If a AWS CloudFormation stack fails to create, it is rolled back. All resources the failed stack created are deleted during the rollback, unless you specify the `--disable-rollback` option.
resources	Each AWS CloudFormation stack contains at least one resource, such as an Auto Scaling LaunchConfiguration. All resources in a stack must be created successfully for the stack to be created.
resource properties	Each AWS CloudFormation resource may have one or more properties associated with it. For example, an `AWS::EC2::Instance` resource may have a `UserData` property. In a AWS CloudFormation template, resources must declare a properties section, even if the resource has no properties.
stack	A stack is a collection of AWS resources you create and delete as a single unit.
section	See template object.
template	AWS CloudFormation uses a JSON compliant file called a template to create a stack.
template format version	The features available in a AWS CloudFormation template depend on the version you declare in the `AWSTemplateFormatVersion` section. If you omit the `AWSTemplateFormatVersion` section from your template, AWS CloudFormation assumes the most recent format version.
template object	Also template section. Each AWS CloudFormation template contains several sections in which you declare the objects your stack will need, such as resources, parameters, and so on.
template validation	The code in a AWS CloudFormation template must be valid JSON code. You can validate any AWS CloudFormation template using the `cfn-validate-template` command.
validation	See template validation.
AWS Type	AWS CloudFormation supports a subset of all AWS resources. Each supported resource is assigned an AWS type, such as `AWS::EC2::Instance`. You specify the AWS type in the template when you declare a resource.
type	See AWS Type.
resource events	See events.
resource status	See status.
stack events	See events.
stack status	See status.
status	AWS CloudFormation reports the status of your stack and its resources when you use the cfn-create-stack, cfn-delete-stack, , and cfn-describe-stack-resources commands. One of *`IN_PROGRESS`*|*`CREATE_FAILED`*|*`CREATE_COMPLETE`* | *`DELETE_IN_PROGRESS`* | *`DELETE_FAILED`* | *`DELETE_COMPLETE`*.

tag	Metadata (consisting of up to 10 key/value pairs) that you can define and assign to EC2 resources. Along with your tas, AWS CloudFormation adds the stack name as a tag to the member resources of your stacks.
UserData	You can pass information to a Amazon EC2 instance when you run it. With AWS CloudFormation, you can specify this information for an instance by declaring the UserData in the template.

Document History

The following table describes the important changes to the AWS CloudFormation documentation. This documentation is associated with the 2010-05-15 release of AWS CloudFormation. This guide was last updated on August 27, 2012.

Change	Description	Release Date
New and Updated Documentation	Reorganization of topics to more clearly provide specific information about using the AWS Management Console (p. 77) and using the AWS CloudFormation command-line interface (CLI) (p. 83). Information about tagging AWS CloudFormation stacks has been added to the documentation, including new guides and updated reference topics: • New topic in Using the Console: Tagging an AWS CloudFormation Stack (p. 80). • Updates to CLI commands: cfn-create-stack (p. 342), which now contains a **--tag** argument, and cfn-describe-stacks (p. 347), which now returns information about tags. • New information about tags in the *AWS CloudFormation API reference*: CreateStack, Stack, and Tag. New information about working with Windows Stacks (p. 179): • Microsoft Windows Amazon Machine Images (AMIs) and AWS CloudFormation Templates (p. 179) • Bootstrapping AWS CloudFormation Windows Stacks (p. 180) • Accessing AWS CloudFormation Windows Instances (p. 184) New topic: Using Regular Expressions in AWS CloudFormation Templates (p. 159). New section: Installing the AWS CloudFormation Command Line Interface (CLI) (p. 59), with tailored guides for Linux (p. 59), Mac OS X (p. 64) and Windows (p. 67).	August 27, 2012

Change	Description	Release Date
New Feature	AWS CloudFormation now provides full support for Virtual Private Cloud (VPC) security with Amazon EC2. You can now create and populate an entire VPC with every type of VPC resource (subnets, gateways, network ACLs, route tables, and so forth) using a single AWS CloudFormation template. Templates can be downloaded that demonstrate new VPC features: Single instance in a single subnet Multiple subnets with Elastic Load Balancing (ELB) and an auto scaling group Documentation for the following resource types has been updated: AWS::EC2::SecurityGroup (p. 237) AWS::EC2::SecurityGroupIngress (p. 238) AWS::EC2::SecurityGroupEgress (p. 242) AWS::EC2::Instance (p. 222) AWS::AutoScaling::AutoScalingGroup (p. 190) AWS::EC2::EIP (p. 220) AWS::EC2::EIPAssociation (p. 221) AWS::ElasticLoadBalancing::LoadBalancer (p. 258) New resource types have been added to the documentation: AWS::EC2::VPC (p. 248) AWS::EC2::InternetGateway (p. 227) AWS::EC2::DHCPOptions (p. 218) AWS::EC2::DHCPOptions (p. 236) AWS::EC2::RouteTable (p. 233) AWS::EC2::NetworkAcl (p. 228) AWS::EC2::NetworkAclEntry (p. 229) AWS::EC2::Subnet (p. 243) AWS::EC2::VPNGateway (p. 252) AWS::EC2::CustomerGateway (p. 217)	April 25, 2012
New Feature	AWS CloudFormation now allows you to add or remove elements from a stack when updating it. Updating AWS CloudFormation Stacks (p. 33) has been updated, and a new section has been added to the walkthrough: Change the Stack's Resources (p. 56), which describes how to add and remove resources when updating the stack. The reference topics: ElasticBeanstalk ConfigurationTemplate Property Type (p. 311), and ElasticBeanstalk OptionSettings Property Type (p. 312), have been significantly updated in response to developer requests for more information.	April 13, 2012

Change	Description	Release Date
New Feature	AWS CloudFormation now provides support for resources in an existing Amazon Virtual Private Cloud (VPC). With this release, you can: • Launch an EC2 Dedicated Instance into an existing VPC. For more information, see AWS::EC2::Instance (p. 222). • Set the SourceDestCheck attribute of an Amazon EC2 instance that resides in an existing VPC. For more information, see AWS::EC2::Instance (p. 222) • Create Amazon Elastic IP Addresses in an existing VPC. For more information, see AWS::EC2::EIP (p. 220) • Use CloudFormation to create VPC security groups and ingress/egress rules in an existing VPC. For more information, see AWS::EC2::SecurityGroup (p. 237). • Associate an Auto Scaling Group with an existing Amazon VPC by setting the VPCZoneIdentifier property of your AWS::AutoScaling::AutoScalingGroup resource. For more information, see AWS::AutoScaling::AutoScalingGroup (p. 190). • Attach an Elastic Load Balancing LoadBalancer to a VPC subnet and create security groups for the LoadBalancer. For more information, see AWS::ElasticLoadBalancing::LoadBalancer (p. 258). • Create an RDS instance in an existing VPC. For more information, see AWS::RDS::DBInstance (p. 274).	February 2, 2012
New Feature	You can now update properties for the following resources in an existing stack: • AWS::EC2::SecurityGroupIngress (p. 238) • AWS::EC2::SecurityGroupEgress (p. 242) • AWS::EC2::EIPAssociation (p. 221) • AWS::RDS::DBSubnetGroup (p. 278) • AWS::RDS::DBSecurityGroup (p. 279) • AWS::RDS::DBSecurityGroupIngress (p. 281) • AWS::Route53::RecordSetGroup (p. 285) For the full list of updateable resources and details about things to consider when updating a stack, see Updating AWS CloudFormation Stacks (p. 33).	February 2, 2012
Restructured Guide	Reorganized existing sections into new sections: Working with AWS CloudFormation Templates (p. 89) and **Managing Stacks**. Moved Template Reference (p. 187) to the top level of the Table of Contents. Moved Estimating the Cost of Your AWS CloudFormation Stack (p. 79) to the Getting Started section.	February 2, 2012

Change	Description	Release Date
New Content	Added three new sections: • Walkthrough: Updating a Stack (p. 38) is a tutorial that walks through the process of updating a LAMP stack. • Deploying Applications with AWS CloudFormation (p. 167) describes how to use AWS CloudFormation helper scripts to deploy applications using metadata stored in your template. • CloudFormation Helper Scripts Reference (p. 329) provides reference material for the AWS CloudFormation helper scripts (cfn-init, cfn-get-metadata, cfn-signal, and cfn-hup).	February 2, 2012
New Feature	AWS CloudFormation now provides the cfn-list-stacks command, which enables you to list stacks filtered by stack status. Deleted stacks can be listed for up to 90 days after they have been deleted. For more information, see Describing and Listing Your Stacks (p. 83).	May 26, 2011
New Features	The cfn-describe-stack-resources and cfn-get-template commands now enable you to get information from stacks which have been deleted for 90 days after they have been deleted. For more information, see Listing Member Resources (p. 86) and Retrieving a Template (p. 86).	May 26, 2011
New Link	AWS CloudFormation endpoint information is now located in the Amazon Web Services General Reference. For more information, go to Regions and Endpoints in Amazon Web Services General Reference.	March 1, 2011
Initial Release	This is the initial public release of AWS CloudFormation.	February 25, 2011