AWS Cloudformation
Configuration Management and Infrastructure as a Code

Objective:

To create a stack with a custom resource, a Lambda function, and an EC2 instance. Custom provisioning logic can be written in templates by using Custom resources.

AWS CloudFormation templates that declare an Amazon Elastic Compute Cloud (Amazon EC2) instance must specify an Amazon Machine Image (AMI) ID(such as an operating system and other software and configuration information) used to launch the instance. AMI IDs can change regularly (when an AMI is updated with software updates) and the correct AMI ID depends on the instance type and region where the stack is launched.

The AMI IDs are mapped to specific instance types and regions. AMI Ids are updated manually in each of the templates. Instead of manually maintaining the mappings we can create a function(by using custom resources and AWS Lambda) that gets the IDs of the latest AMIs for the region and instance type.

This document shows how to create a custom resource and associate a Lambda function with it to look up AMI IDs.

The sample template provided by AWS uses the custom resource type to invoke and send input values to the Lambda function. When the template is used, AWS CloudFormation invokes the function and sends information to it, such as the request type, input data, and a pre-signed Amazon Simple Storage Service (Amazon S3) URL. The function uses that information to look up the AMI ID, and then sends a response to the pre-signed URL.

AWS CloudFormation proceeds with stack creation after it gets a response in the pre-signed URL location. AWS CloudFormation uses the Lambda function's response to specify the instance's AMI ID when it creates instance.

The following list summarizes the process.

1. AWS Identity and Access Management (IAM) permissions to use all the corresponding services, such as Lambda, Amazon EC2, and AWS CloudFormation.
2. Download and save the package in a S3 bucket that's in the same region as the stack.
3. Create Stack: The stack creates an IAM role (execution role), which Lambda uses to make calls to Amazon EC2 and also demonstrates; how to associate the Lambda function with a custom resource and how to use the results from the function to specify AMI ID.
4. Delete the stack in order to clean up all the stack resources and to avoid unnecessary charges.

To download and save the package in Amazon S3

1. Download the sample package from Amazon and save it in S3 using the same filename as the sample `amilookup.zip`.
2. Open the Amazon S3 console and choose or create a bucket that's located in the same region in which AWS CloudFormation stack will be created. Record the bucket name.

Snippets from Stack Template

In order to create the Lambda function, declare the `AWS::Lambda::Function` resource, which requires the function's source code, handler name, runtime environment, and execution role ARN.

Example JSON Syntax

```
"AMIInfoFunction": {

  "Type": "AWS::Lambda::Function",

  "Properties": {

    "Code": {

      "S3Bucket": { "Ref": "S3Bucket" },

      "S3Key": { "Ref": "S3Key" }

    },

    "Handler": { "Fn::Join" : [ "", [{ "Ref": "ModuleName" },".handler"] ]
},

    "Runtime": "nodejs8.10",

    "Timeout": "30",

    "Role": { "Fn::GetAtt" : ["LambdaExecutionRole", "Arn"] }

  }

}
```

The `Code` property specifies the Amazon S3 location (bucket name and file name) where the sample package was uploaded. The sample template uses the following input parameters:

To set the bucket and file names in order to be able to specify the names when the stack is created.

`"Ref": "S3Bucket"` and

`"Ref": "S3Key"`

Similarly, the handler name, which corresponds to the name of the source file (the JavaScript file) in the `.zip` package, also uses an input parameter (`"Ref": "ModuleName"`).

Runtime is specified as `nodejs8.10`.

The execution time for the function exceeds the default value of `3` seconds, so the timeout is set to `30` seconds. If a long timeout is not specified, then Lambda function might cause a timeout before the function can complete, causing stack creation to fail.

The execution role, is specified by using the

`Fn::GetAtt` intrinsic function in the `Role` property.

The execution role grants the Lambda function permission to send logs to AWS and to call the EC2 `DescribeImages` API.

The following code shows the role and policy that grant the appropriate permission:

Example JSON Syntax

```
"LambdaExecutionRole": {

  "Type": "AWS::IAM::Role",

  "Properties": {

    "AssumeRolePolicyDocument": {
```

```
      "Version": "2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"Service": ["lambda.amazonaws.com"]},
        "Action": ["sts:AssumeRole"]
      }]
    },
    "Path": "/",
    "Policies": [{
      "PolicyName": "root",
      "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Effect": "Allow",
          "Action":
["logs:CreateLogGroup","logs:CreateLogStream","logs:PutLogEvents"],
          "Resource": "arn:aws:logs:*:*:*"
        },
        {
          "Effect": "Allow",
          "Action": ["ec2:DescribeImages"],
```

```
            "Resource": "*"
          }]
        }
      }]
  }
}
```

The custom resource invokes the Lambda function that is associated with it for both Linux and Windows templates. To associate a function with a custom resource, you specify the Amazon Resource Name (ARN) of the function for the `ServiceToken` property, using the `Fn::GetAtt` intrinsic function. AWS CloudFormation sends the additional properties that are included in the custom resource declaration, such as `Region` and `Architecture`, to the Lambda function as inputs. The Lambda function determines the correct names and values for these input properties.

Example JSON Syntax

```
"AMIInfo": {
  "Type": "Custom::AMIInfo",
  "Properties": {
    "ServiceToken": { "Fn::GetAtt" : ["AMIInfoFunction", "Arn"] },
    "Region": { "Ref": "AWS::Region" },
    "Architecture": { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "InstanceType" }, "Arch" ] }
  }
}
```

Example JSON Syntax

```
"AMIInfo": {

  "Type": "Custom::AMIInfo",

  "Properties": {

    "ServiceToken": { "Fn::GetAtt" : ["AMIInfoFunction", "Arn"] },

    "Region": { "Ref": "AWS::Region" },

    "OSName": { "Ref": "WindowsVersion" }

  }

}
```

When AWS CloudFormation invokes the Lambda function, the function calls the EC2 `DescribeImages` API, using the region and instance architecture or the OS name to filter the list of images. Then the function sorts the list of images by date and returns the ID of the latest AMI.

When returning the ID of the latest AMI, the function sends the ID to a pre-signed URL in the `Data` property of the response object.

The data is structured as a name-value pair, as shown in the following example:

```
"Data": {

  "Id": "ami-43795473"

}
```

The following code shows how to get the data from a Lambda function and it uses the `Fn::GetAtt` intrinsic function, providing the name of the custom resource(`AMIInfo`) and the attribute name(`Id`) of the value that you want to get.

Example JSON Syntax

```
"SampleInstance": {

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

  "Properties": {

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

    "ImageId": { "Fn::GetAtt": [ "AMIInfo", "Id" ] }

  }

}
```

Step 2: Creating the Stack

Amazon EC2 stack can be created using a sample template that includes a Lambda function, an IAM execution role, a custom resource that invokes the function, and an EC2 instance that uses the results from the function.

The custom resource invokes the Lambda function and waits until the function sends a response to the pre-signed Amazon S3 URL during stack creation.. The function in its response returns the ID of the latest AMI that corresponds to the EC2 instance type and region in which the instance has been created. The data is stored as an attribute of the custom resource, which is used to specify the AMI ID of the EC2 instance.

To create the stack

1. Open the AWS CloudFormation console, Choose **Create Stack**.
2. In the **Template** section, choose **Specify an Amazon S3 template URL**, and then copy and paste the following URL in the text box:
   Choose **Next**.
3. In the **Stack name** field, type **`amilookup`**.
4. In the **Parameters** section, specify the name of the Amazon S3 bucket that you created, and then choose **Next**.
   The default values for the other parameters are the same names that are used in the sample `.zip` package.
5. Ensure that the template URL and stack name are correct, and then choose **Create**.

Services Resource Groups mohanbrinda Ohio

The previous console is being deprecated in favor of the new console
The previous console is being deprecated in favor of this new AWS CloudFormation Console, that we are continually improving based on customer feedback. There will be no new
added or improvements made to the previous console. We will continue to support bug fixes and patches until 03/31/2020 after which the previous console will no longer be acc

CloudFormation > Stacks

Stacks (0) Delete Update Stack actions Create st

Filter by stack name Active View nested 1

Stack name	Status	Created time	Description

No stacks
No stacks to display

Create stack

View getting started guide

ource Groups mohanbrinda

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

- (●) Template is ready
- ( ) Use a sample template
- ( ) Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

- ( ) Amazon S3 URL
- (●) Upload a template file

Upload a template file

Choose file *LambdaAMILookupSample.json*

JSON or YAML formatted file

S3 URL: https://s3.us-east-2.amazonaws.com/cf-templates-dnokwwjcd8g0-us-east-2/2019330sax-LambdaAMILookupSample.json

View in Designer

Resource Groups mohanbrinda Ohi

Stack name

amilookup

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

InstanceType
EC2 instance type

t2.micro

ModuleName
The name of the JavaScript file

amilookup

S3Bucket
The name of the bucket that contains your packaged source

mydevops

S3Key
The name of the ZIP package

amilookup.zip

Cancel Previous Next

Tags

You can specify tags (key-value pairs) to apply to resources in your stack. You can add up to 50 unique tags for each stack. Learn more

Key Value Remove

Add tag

Permissions

Choose an IAM role to explicitly define how CloudFormation can create, modify, or delete resources in the stack. If you don't choose a role, CloudFormation uses permissions based on your user credentials. Learn more

IAM role - optional
Choose the IAM role for CloudFormation to use for all operations performed on the stack.

IAM role name Sample-role-name Remove

Advanced options

You can set additional options for your stack, like notification options and a stack policy. Learn more

Stack policy

Defines the resources that you want to protect from unintentional updates during a stack update.

ce Groups mohanbrinda Oh

Template URL

https://s3.us-east-2.amazonaws.com/cf-templates-dnokwwjcd8g0-us-east-2/2019330sax-LambdaAMILookupSample.json

Stack description

AWS CloudFormation AMI Look Up Sample Template: Demonstrates how to dynamically specify an AMI ID. This template provisions an EC2 instance with an AMI ID that is based on the instance's type and region. **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.

Estimate cost not available

Step 2: Specify stack details

Edit

Parameters (4)

Search parameters

Key	Value
InstanceType	t2.micro
ModuleName	amilookup
S3Bucket	mydevops
S3Key	amilookup.zip

Stack creation options

Rollback on failure

Enabled

Timeout

-

Termination protection

Disabled

Quick-create link

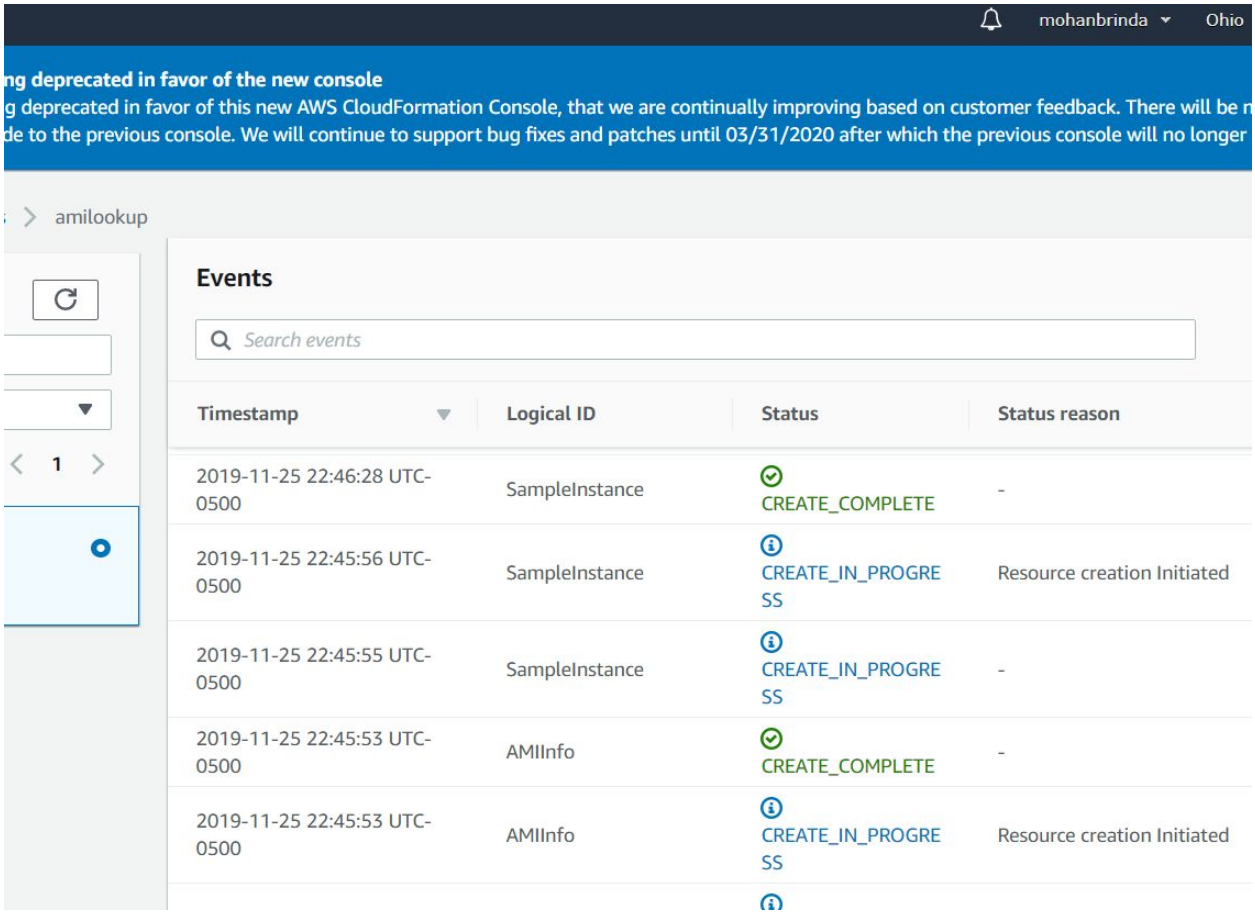
Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. Learn more

I acknowledge that AWS CloudFormation might create IAM resources.

Cancel | Previous | Create change set | Create stack


mohanbrinda
Ohio
ng deprecated in favor of the new console
g deprecated in favor of this new AWS CloudFormation Console, that we are continually improving based on customer feedback. There will be n
de to the previous console. We will continue to support bug fixes and patches until 03/31/2020 after which the previous console will no longer
amilookup
Events
Search events
Timestamp
Logical ID
Status
Status reason
2019-11-25 22:46:28 UTC-0500
SampleInstance
CREATE_COMPLETE
-
2019-11-25 22:45:56 UTC-0500
SampleInstance
CREATE_IN_PROGRESS
Resource creation Initiated
2019-11-25 22:45:55 UTC-0500
SampleInstance
CREATE_IN_PROGRESS
-
2019-11-25 22:45:53 UTC-0500
AMIInfo
CREATE_COMPLETE
-
2019-11-25 22:45:53 UTC-0500
AMIInfo
CREATE_IN_PROGRESS
Resource creation Initiated

Resource Groups | mohanbrinda

Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DN
	i-09f3158ce59fec006	t2.micro	us-east-2b	running	2/2 checks ...	None	ec2-18-21

Security groups	default. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-9ba84ff0
AMI ID	amzn-ami-hvm-2018.03.0.20190826-x86_64-gp2 (ami-0c64dd618a49aeee8)	Subnet ID	subnet-0586c37f
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	-	T2/T3 Unlimited	Disabled
Owner	436050113826	EBS-optimized	False

Resource Groups | mohanbrinda | Ohio | Support

Lambda > Functions

Functions (1)

Actions | Create function

Filter by tags and attributes or search by keyword

1

Function name	Description	Runtime	Code size	Last modified
amilookup-AMIInfoFunction-1M0CR5AI8XGAS		Node.js 8.10	1.5 kB	8 minutes ago

Services Resource Groups

ookup-AMIInfoFunction-1M0CR5AI8XGAS

Throttle | Qualifiers | Actions | Select a test event

Layers (0)

Add trigger

ion code Info

ntry type: ode inline

Runtime: Node.js 8.10

Handler Info: amilookup.handler

File Edit Find View Go Tools Window

amilookup-AMIInfol

amilookup.js

```
/**
* A sample Lambda function that looks up the latest AMI ID for a given region and architecture.
**/

// Map instance architectures to an AMI name pattern
var archToAMINamePattern = {
    "PV64": "amzn-ami-pv*x86_64-ebs",
    "HVM64": "amzn-ami-hvm*x86_64-gp2",
    "HVMG2": "amzn-ami-graphics-hvm*x86_64-ebs*"
```

The previous console is being deprecated in favor of the new console

The previous console is being deprecated in favor of this new AWS CloudFormation Console, that we are continually improving based on customer feedback. There will be no new features added or improvements made to the previous console. We will continue to support bug fixes and patches until 03/31/2020 after which the previous console will no longer be accessible.

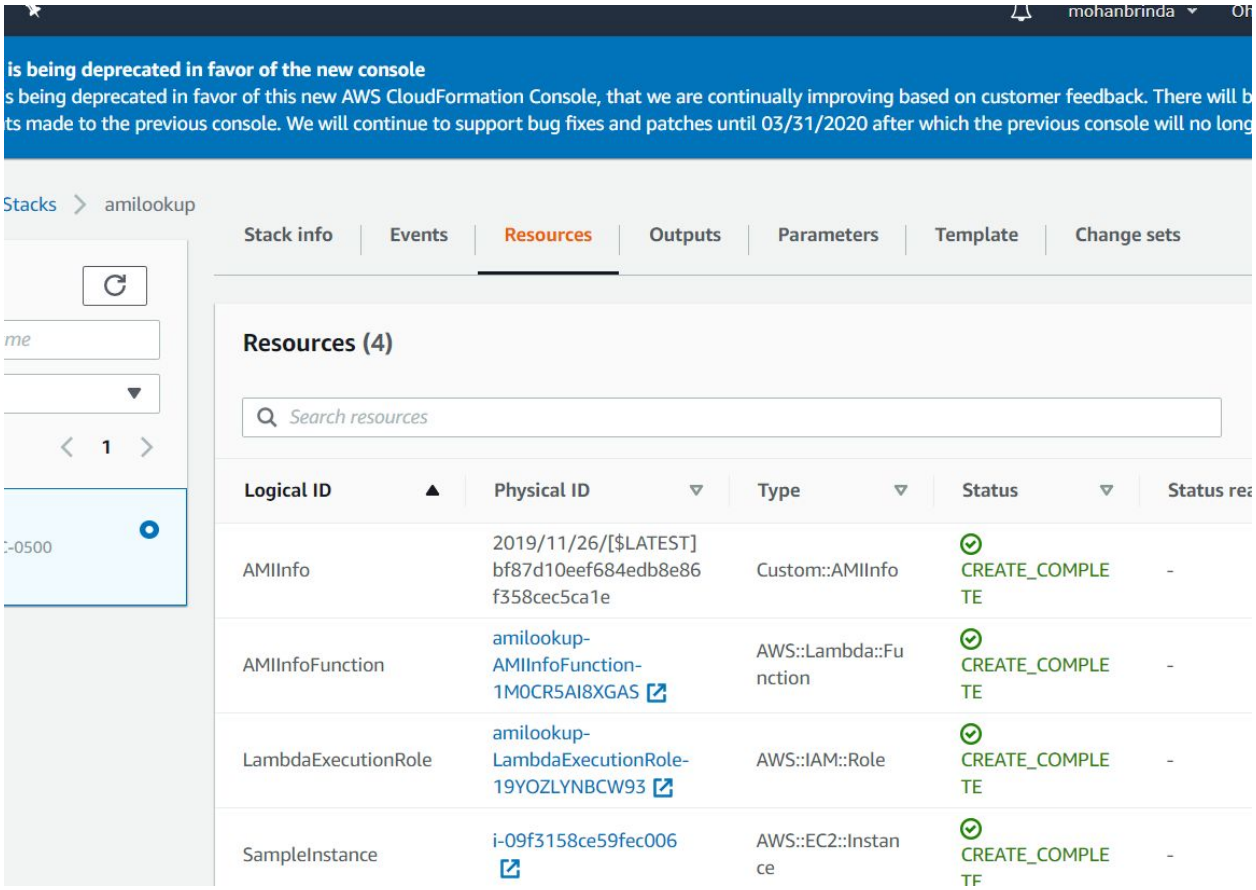
CloudFormation > Stacks

Stacks (1)

Delete | Update | Stack actions | Create stack

Filter by stack name | Active | View nested | 1

Stack name	Status	Created time	Description
amilookup	CREATE_COMPLETE	2019-11-25 22:45:27 UTC-0500	AWS CloudFormation AMI Look Up Sample Te...



AWS CloudFormation takes several minutes to create the stack. If stack creation succeeds, all resources such as the Lambda function, custom resource, and EC2 instance will be created.

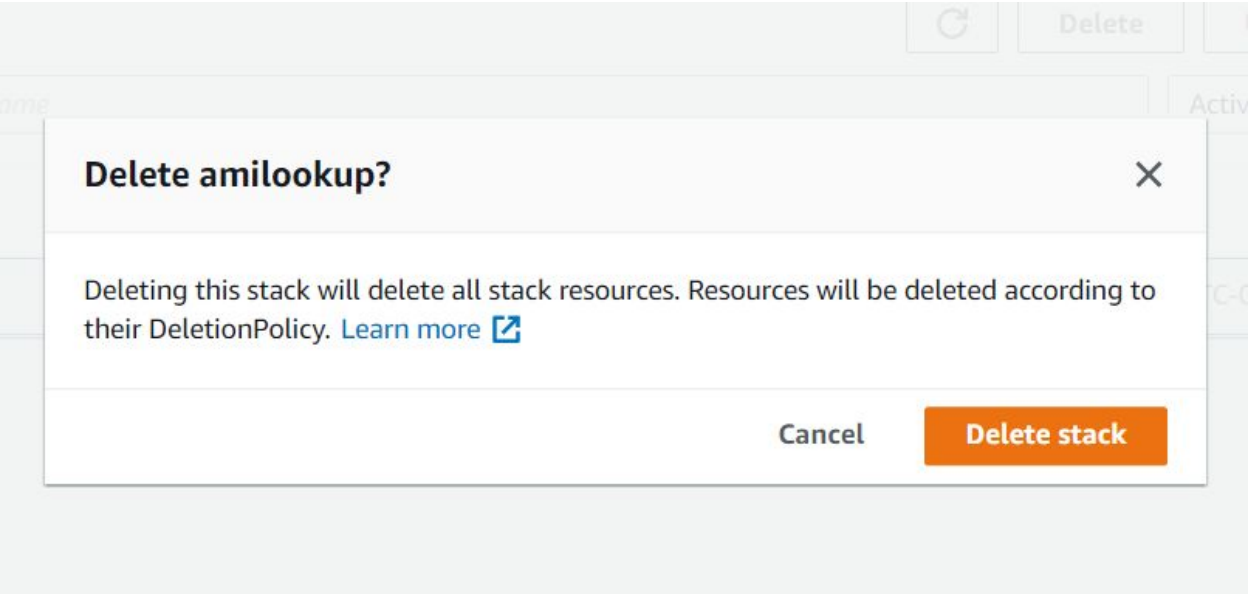
View the stack outputs, in order to see which AMI ID AWS CloudFormation was used to create the EC2 instance.The function's logs can be viewed in the Amazon CloudWatch Logs console incase if the Lambda function returns error.

Step 3: Clean Up Resources

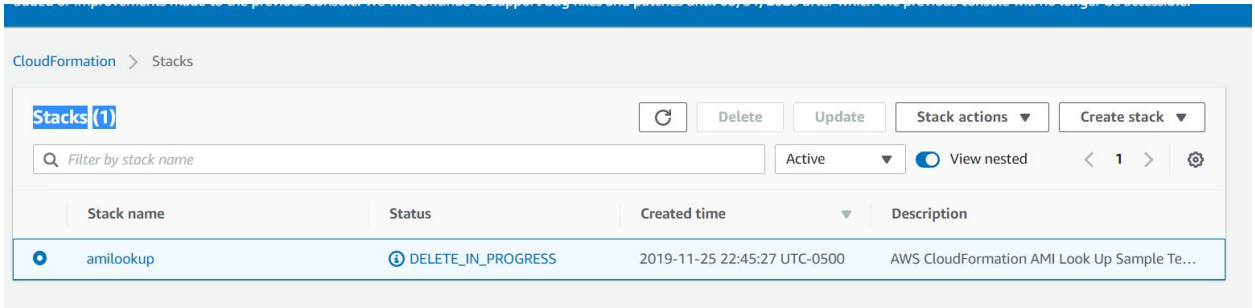
Delete the stack in order to avoid being charged for unwanted services.

To delete the stack

1. From the AWS CloudFormation console, choose the **SampleEC2Instance** stack.
2. Choose **Actions** and then **Delete Stack**.



3. In the confirmation message, choose **Yes, Delete**.



All the resources that were created are deleted.

References:

http://aws.amazon.com.