Cloud Configuration - 1.0

On this page:

Summary: This document gives information about the extension providing support for cloud computing frameworks that are using configuration files such as **AWS Cloudformation**, **AWS SAM**, or **Serverless framework**.

Extension ID

com.castsoftware.cloudconfig

What's new

Please see **Cloud Configuration - 1.0 - Release Notes** for more information.

Description

This extension provides support for some cloud computing frameworks that are using configuration files:

- **AWS Cloudformation**
- **AWS SAM**
- **Serverless framework**

In what situation should you install this extension?

If your application uses **AWS Cloudformation**, **AWS SAM**, or **Serverless framework** (for deploying AWS services).

Files analyzed

The files with the following extensions are analyzed: .json, .yaml, .yml, .template

Objects

The following specific objects are displayed in CAST Enlighten:

Icon	Description
	AWS Get API Gateway
	AWS Post API Gateway
	AWS Put API Gateway
	AWS Delete API Gateway
	AWS Any API Gateway
	AWS Post Request
	AWS Lambda Function
	Call to AWS Lambda Function
	AWS Simple Queue Service Receiver
	AWS SNS Subscriber
	AWS Simple Queue Service Publisher
	AWS SMS, AWS Email

AWS support

Cloudformation support

Cloudformation uses configuration files for setting up AWS architecture. The configuration files use yaml or json format. Most examples shown in this page are using the yaml format but json format is also supported.

Supported Resources

Resources are defined within the Resources section of the configuration files. Each resource is defined as follow:

```
Resources:
  NameOfTheResource:
    Type: AWS::ResourceIdentifier
        Properties:
           SomeProperty:
         #...
  NameOfOtherResource:
        #...
```

The resources of the following types are analyzed.

AWS::Lambda::Function

When analysing the following source code

```
Resources:
  lambdaResourceName:
        Type: AWS::Lambda::Function
        Properties:
          FunctionName: lambdaName
      Handler: handler.compute
      Runtime: nodejs8.10
```

A lambda function object named **lambdaName** is created. It has a property for saving the handler path and the runtime.

The linking from the lambda function to the handler function is then carried out by one of the following extensions (depending on the runtime):

runtime	extension
java	com.castsoftware.awsjava
dotnet	com.castsoftware.awsdotnet
python	com.castsoftware.python
nodejs	com.castsoftware.nodejs (when the handler is written in .js) com.castsoftware.typescript (when the handler is written in .ts)

The minimal version required for these extensions to get the link is given in this section.

If the property FunctionName is not provided, the name of the function is by default the name of the resource (lambdaResourceName in this example).

AWS::DynamoDB::Table, AWS::SQS::Queue, and AWS::S3::Bucket

These types of resources are used to set up some services. This extension analyses them to get the name of the services. For instance when analyzing the following:

```
Resources:
  SQSResourceName:
        Type: AWS::SQS::Queue
        Properties:
                QueueName: QueueName
                #...
```

the analyzer will find that any reference to the resource **SQSResourceName** will refer to the SQS Queue named **QueueName**. If no name is provided, the resource name is used.

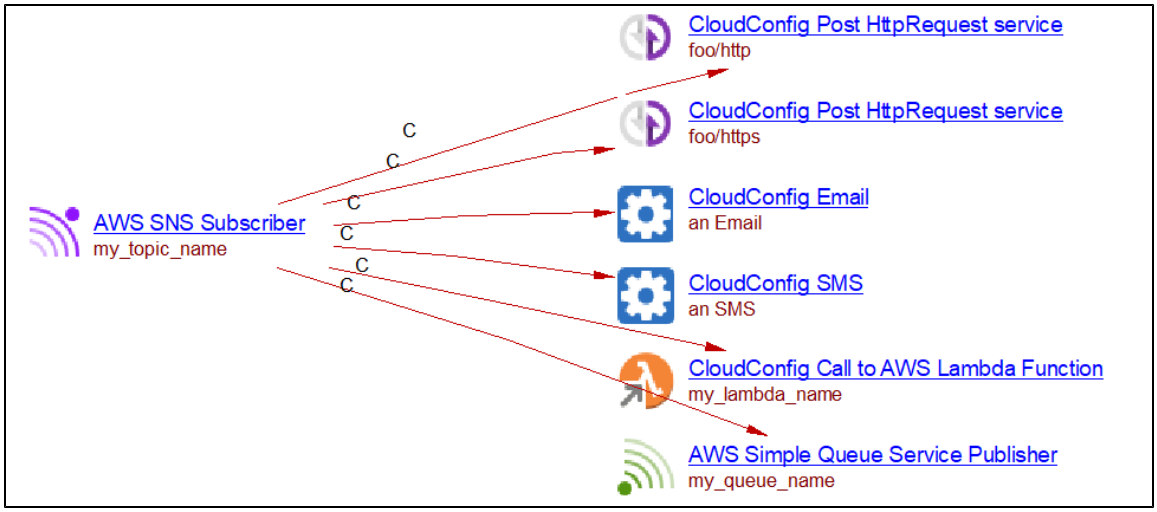
AWS::SNS::Topic

This resource is used to set up an SNS topic with its subscription.

When analyzing the following source code:

```
Resources:
 Queue:
   Type: AWS::SQS::Queue
   Properties:
      QueueName: my_queue_name
 LambdaFunction:
   Type: AWS::Lambda::Function
   Properties:
      FunctionName: my_lambda_name
      Handler: handler.compute
      Runtime: nodejs8.10
 MySNSTopic:
   Type: AWS::SNS::Topic
   Properties:
     Subscription:
       - Endpoint: cloudkatha@gmail.com
         Protocol: email
       - Endpoint: some_number
         Protocol: sms
       - Protocol: https
         Endpoint: foo/https
       - Protocol: http
         Endpoint: foo/http
       - Protocol: lambda
         Endpoint: !GetAtt LambdaFunction.Arn
       - Protocol: sqs
         Endpoint: !GetAtt Queue.Arn
     TopicName: my_topic_name
```

you will get the following result:



A **Nodejs SNS Subscriber** object named **my_topic_name** is created. Then for each supported protocol, an object is created with a callLink from the subscriber to that object. The supported protocols are the following:

protocol	object created	name of the object
email	Nodejs Email	*an Email* (the email addresses are not evaluated)
http/https	Nodejs Post Service	the url (evaluated from the endpoint)
sqs	Nodejs AWS Simple Queue Service Publisher	the name of the queue (evaluated from the endpoint)
lambda	Nodejs Call to AWS Lambda Function	the name of the lambda function (evaluated from the endpoint)
sms	Nodejs SMS	*an SMS* (the SMS numbers are not evaluated)

Whenever the endpoint is an ARN, the ARN may be hardcoded:

```
- Protocol: sqs
  Endpoint: arn:aws:sqs:us-east-2:123456789012:my_queue_name
```

The name of the service (my_queue_name in this example) is extracted.

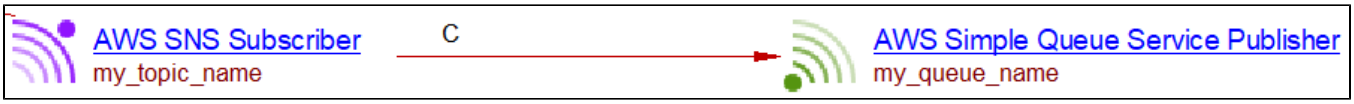
AWS::SNS::Subscription

The subscriptions to an SNS topic can be made inside an AWS::SNS::Subscription resource.

When analyzing the following source code,

```
Resources:
  SnsTopic:
    Type: AWS::SNS::Topic
        Properties:
                TopicName: my_topic_name
  Queue:
    Type: AWS::SQS::Queue
        Properties:
      QueueName: my_queue_name
  SnsSubscription:
    Type: AWS::SNS::Subscription
    Properties:
      Protocol: sqs
      Endpoint: !GetAtt Queue.Arn
      Region: !Ref TopicRegion
      TopicArn: !GetAtt SnsTopic.Arn
```

you will get the following result:



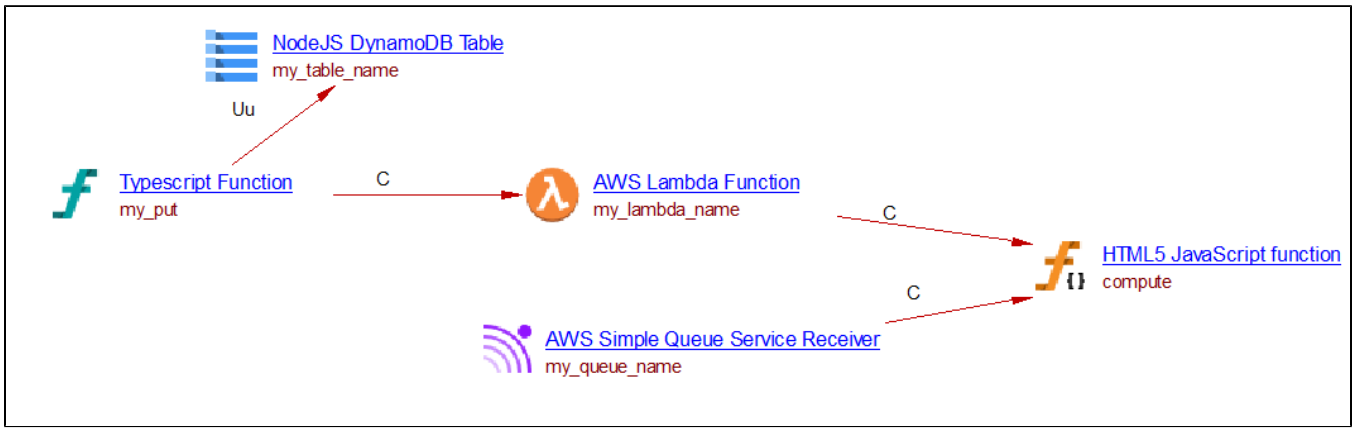
The same endpoints as those for the AWS::SNS::Topic resource are supported.

AWS::Lambda::EventSourceMapping

CloudFormation allows mapping some events to lambda function using AWS::Lambda::EventSourceMapping. When these events occur, the lambda function will be executed. The supported events are DynamoDB and SQS queue. The analysis of the following source code:

```
Resources:
  DataResourceTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: my_table_name
           # ...

  SQSResourceName:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: my_queue_name
          #...

  lambdaResourceName:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: my_lambda_name
      Handler: handler.compute
      Runtime: nodejs8.10

  SQSToLambdaEventSourceMapping:
    Type: AWS::Lambda::EventSourceMapping
    Properties:
      EventSourceArn: !GetAtt SQSResourceName.Arn
      FunctionName: !GetAtt lambdaResourceName.Arn
          #...

  DynamodbToLambdaEventSourceMapping:
    Type: AWS::Lambda::EventSourceMapping
    Properties:
      EventSourceArn: !GetAtt DataResourceTable.StreamArn
      FunctionName: !GetAtt lambdaResourceName.Arn
          #...
```

Lead to the following result (assuming that there is some TypeScript source code with a my_put function with a useUpdateLink to a my_table_name DynamoDB table):



For an SQS event to Lambda, an SQS Receiver object linked to the lambda function is created.

For a DynamoDB event to Lambda, a property is added to the Lambda object with the name of the DynamoDB table triggering the Lambda. Any object with a callLink to a DynamoDB table matching that name is then linked to the Lambda object (see the **my_put** function in the example).

Other features supported

Parameters

Some variables can be defined in the parameter section of the yaml file. When the name of a service is that of a parameter and a default value is given, the extension will create the service with that default value.

For instance when analyzing the following source code, a lambda function named **my_lambda_name** is created.

```
Parameters:

  LambdaName:
    Type: String
    Default: my_lambda_name

Resources:
  lambdaResourceName:
        Type: AWS::Lambda::Function
        Properties:
          FunctionName: !Ref LambdaName
      Handler: handler.compute
      Runtime: nodejs8.10
```

Intrinsic functions

CloudFormation provides some intrinsic functions. Only the **Join**, **GetAtt**, and **Ref** functions are supported.

Serverless Application Model (SAM)

AWS SAM is an extension of AWS CloudFormation. All that is supported for CloudFormation is supported for SAM. It provides other Types for declaring resources.

The following SAM resource is supported

AWS::Serverless::Function

Overview

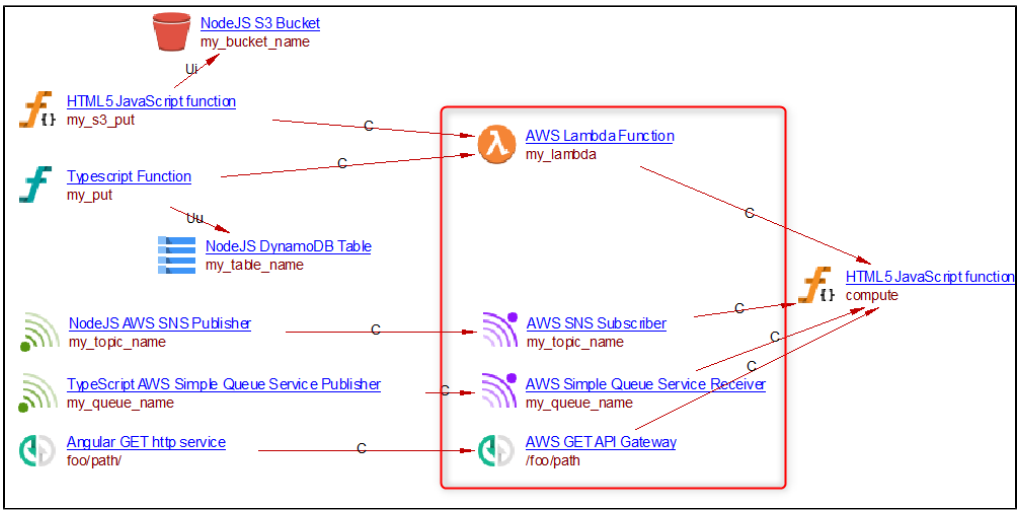
This resource allows defining a lambda function. It has an Events property which allows setting up some triggers for the lambda function.

The supported events types are **S3**, **DynamoDB**, **SNS**, **SQS**, and **HttpApi**.

When analyzing the following source code (in combination with some JavaScript and TypeScript source code which we do not display here):

```
Resources:
  MyLambdaResource:
    Type: 'AWS::Serverless::Function'
    Properties:
      FunctionName: my_lambda
      Handler: handler.compute
      Runtime: nodejs4.3
      Events:
        S3Event:
          Type: S3
          Properties:
            Bucket:
              Ref: S3Bucket
            Events:
              - s3:ObjectCreated:*
        Stream:
          Type: DynamoDB
          Properties:
            Stream: !GetAtt DynamoDBTable.StreamArn
            BatchSize: 100
            StartingPosition: TRIM_HORIZON
        SNSEvent:
          Type: SNS
          Properties:
            Topic:
              Ref: SNSTopic1
        MySQSEvent:
          Type: SQS
          Properties:
            Queue: !GetAtt MySqsQueue.Arn
            BatchSize: 10
        MyHttpEvent:
          Type: HttpApi
          Properties:
            ApiId: !Ref HttpApi
            Path: /foo/path
            Method: GET

  MySqsQueue:
    Type: AWS::SQS::Queue
    Properties:
      QueueName: my_queue_name

  DynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      TableName: my_table_name

  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: my_bucket_name

  SNSTopic1:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'my_topic_name'
```

We get the following result:



Only the objects in the red square are created when analyzing this source code with this extension.

> Some links will be created only when some objects created by other extensions are created with a minimal version for these other extensions as described in this section.

Support for S3 event:

To define an S3 trigger for a lambda function, one needs to define the bucket and the type of events on that bucket that will trigger the Lambda.

The analyzer creates a call link to the lambda function object from all callables linked to the given bucket through a link of matching type.

The following table tells which link type will match which event type.

event_type	matching link types
No event_type	all
ObjectCreated	useInsert
ObjectRemoved	useDelete
ObjectRestore	None
ReducedRedundancyLostObject	None
Replication	None

> In AWS, the event_type can be more specific by adding information after the semicolon. However, the analyzer does not consider this information. For instance, it will make no distinction between ObjectCreated:* and ObjectCreated:Put or ObjectCreated:Post event types.
>
> A Filter field can also be added to trigger the lambda only for some specific files. **This is currently not supported and may lead to wrong links.**

> The links from S3Bucket callers to the lambda object will be created only if the extension which created the S3 bucket is recent enough as described in this section.

Support for DynamoDB event

The analyzer creates a call link to the lambda function object from all callables linked to the DynamoDB table define in the event.

Support for SNS, SQS and HttpApi events

For these triggers an SNS Subscriber (for SNS), an SQS Receiver (for SQS) or an APIGateway (for HttpApi) objects are created. A runtime and a handler property are added to that object. Based on these properties, if a handler function is found that object will be linked to the handler function.

> This links is created only if the extension which analyses the handler object is recent enough as described in this section.

Serverless framework support

Serverless framework can be used to build AWS applications. It provides its own way of defining lambda functions and their events. Other services can be created within the Resources section of the configuration files and using the CloudFormation nomenclature.

The com.castsoftware.cloudformation extension analyzes anything which is inside the Resources section of a serverless framework configuration file as CloudFormation resources and will create objects as described in the Cloudformation support section of this documentation page.

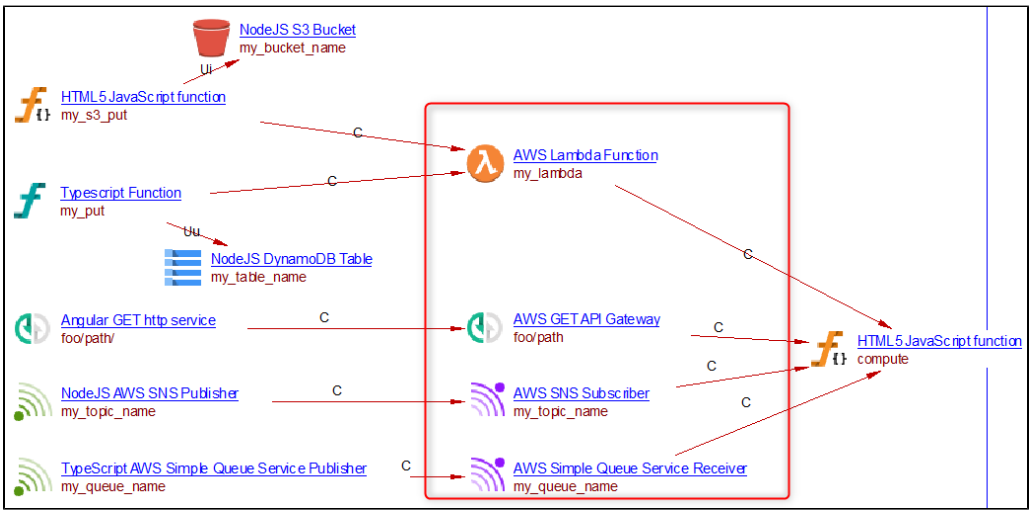
Lambda support

The Lambda support is similar to that for SAM.

When analyzing the following source code (in combination with some JavaScript and TypeScript source code which we do not display here).

```
service: irp-searchCompanies
provider:
  name: aws
  runtime: nodejs4.3
  profile: serverless #user profile used for sls deploy
  versionFunctions: false
  region: us-west-2

functions:
  my_lambda:
    handler: handler.compute
    events:
      - stream: arn:aws:dynamodb:region:XXXXXX:table/my_table_name/stream/1970-01-01T00:00:00.000
      - sns:
         arn:
          Fn::GetAtt:
            - SNSTopic1
            - Arn
      - http: GET foo/path
      - s3:
          bucket: my_bucket_name  #bucket name
          event: s3:ObjectCreated:*
          rules:
            - prefix: logs/ #folder inside a bucket
          existing: true #mark as true if bucket already exists
      - sqs:
          arn:
            Fn::GetAtt:
              - MyQueue
              - Arn

resources:
 Resources:
  SNSTopic1:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'my_topic_name'
  MyQueue:
    Type: "AWS::SQS::Queue"
    Properties:
      QueueName: "my_queue_name"
```

We get the following result:



Only the objects in the red square are created when analyzing this source code with this extension.

> Some links will be created only when some objects created by other extensions are created with a minimal version for these other extensions as described in this section.

Minimal versions for proper linking

Many objects created by this extension may be linked with objects created by other extensions. However for these linkings to work, the minimal version required for these extensions is shown in the following table:

runtime	extension	minimal version for link to handler	minimal version for link from object calling a S3 bucket to lambda function
java	com.castsoftware.awsjava	1.2.0-alpha3	1.2.0-alpha2
dotnet	com.castsoftware.awsdotnet	1.0.0-alpha5	1.0.0-alpha4
python	com.castsoftware.python	1.4.0-beta7	1.4.0-beta4
nodejs	com.castsoftware.nodejs	2.7.0-beta3 (when the handler is written in .js)	2.6.0-funcrel
nodejs	com.castsoftware.typescript	1.9.0-alpha1 (when the handler is written in .ts)	1.8.1-funcrel

Known limitations

- Exports and imports are not supported
- Use of complex mapping key in YAML files is not supported and may lead to missing objects
- Custom resources are not supported in CloudFormation
- Filters in S3 event source for lambda are not supported and may lead to wrong links
- Inline source code for Lambda functions provided in CloudFormation or SAM configuration files is not supported:

```
Code:
  ZipFile: |
    var aws = require('aws-sdk')
    var response = require('cfn-response')
    exports.handler = function(event, context) {
        //...
    }
```