Conditional Resources

In this lesson, you'll learn how to handle conditional resources by using CloudFormation template and SAM CLI!

Email subscriptions are not the right solution for production deployments. Emailing after each individual event would most likely cause too much noise, and people will just start ignoring it. It would be much better to set up error logging in a different way completely. You can make the email subscription optional by adding a condition to your resource.

CloudFormation can activate or deactivate resources based on certain conditions, which you need to set up in a separate template section titled `Conditions`. I usually put this between parameters and resources. The following lines are added to your template as a top-level section, for example above `Resources`.

```
Conditions:
  ContactEmailSet: !Not [ !Equals ['', !Ref ContactEmailAddress]]
```

Line 29 to Line 30 of code/ch9/template-with-dlq.yaml

The previous listing creates a condition to check whether the parameter `ContactEmailAddress` has a value, or more precisely to check that it does not equal a blank string. For functions with multiple parameters, or to specify the order of execution, CloudFormation uses square brackets (`[]`). `!Not` and `!Equals` are built-in functions with fairly obvious purposes. For more information on other functions you can use, check out the AWS CloudFormation *Condition Functions* reference page.

You can use conditions to include or exclude resources from a deployed application by adding a `Condition` field below the resource header (at same indentation level as `Type`). For example, you can add the following code to the resources section of your template to create a conditional email subscription (at the same indentation level as the SNS topic, for example).

```
AlarmNotifyOpsSubscription:
  Type: AWS::SNS::Subscription
  Condition: ContactEmailSet
  Properties:
    Endpoint: !Ref ContactEmailAddress
    Protocol: email
  TopicArn: !Ref NotifyAdmins
```

Line 140 to Line 146 of code/ch9/template-with-dlq.yaml

Now build, package, and deploy the stack without setting the email, and CloudFormation will not set up the subscription at all. You can confirm that by listing the stack resources from the command line, or by inspecting the stack from the AWS CloudFormation Web Console.

Then deploy again, but provide an email (no need to package and build the stack again; you're using the same template). You can set the email using `--parameter-overrides`:

```
sam deploy --template-file output.yaml --stack-name sam-test-1 --capabilit
ies CAPABILITY_IAM --parameter-overrides ContactEmailAddress=YOUR_EMAIL
```

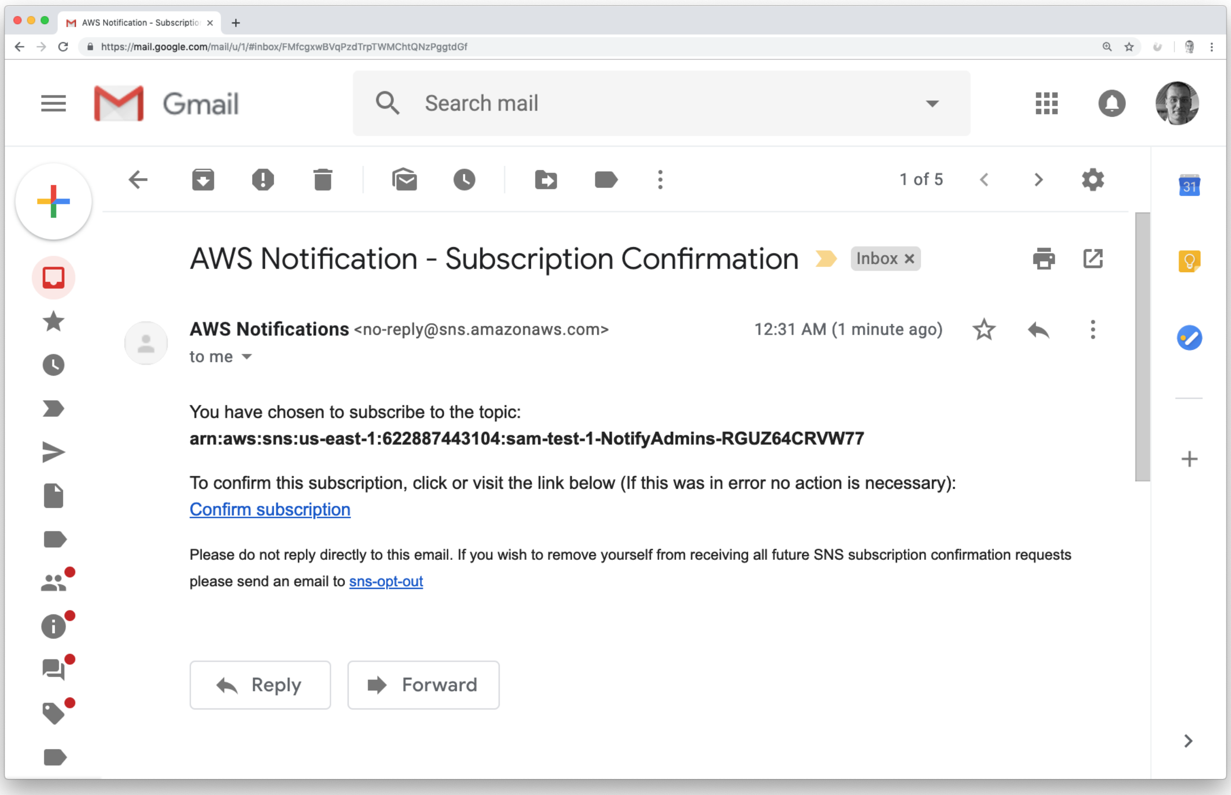
Environment Variables

Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...
AWS_REGION	Not Specified...

```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",

    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,
      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"
  }
}
```

Check the stack resources after the deployment and you’ll see the subscription object there. Email subscriptions are not automatically active, to prevent spam, but if you check your email, you’ll see a notification from AWS asking you to confirm the subscription (see figure below).


AWS Notification - Subscription Confirmation

AWS Notifications <no-reply@sns.amazonaws.com> 12:31 AM (1 minute ago)
to me

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:622887443104:sam-test-1-NotifyAdmins-RGUZ64CRVW77

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

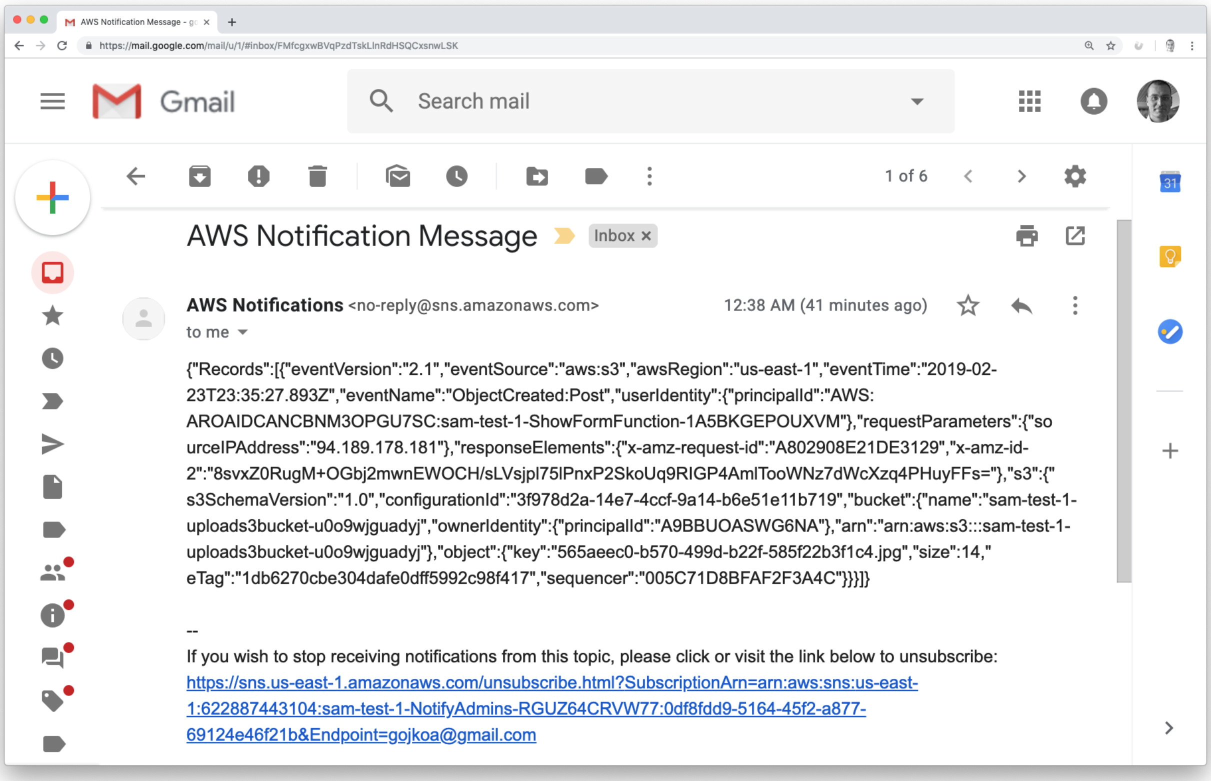
https://mail.google.com/mail/u/1/#inbox/FMfcgxwBVqPzdTrpTWMChtQNzPggtdGf
Gmail
Search mail
1 of 5
Inbox ×
Reply
Forward

SNS sends a request to confirm a new email subscription to the dead letter queue.

Click the confirmation link in the email to set up the subscription. After that, try uploading a file that is not an image to the conversion application. You will see two retries in the logs, and, after that, SNS will send you the failed event (see figure below).

This is a great way to catch unexpected issues during testing, so you can then learn about them and handle such errors better in the code. Dead letter queues will also catch timeouts, so you'll know that some function needs to be reconfigured so it can run longer, or alternatively be optimised to run faster.

You can use this same dead letter queue for as many functions as you like now. There's not much of a point in setting this up for events triggered by API Gateway, because it uses synchronous invocations, so Lambda sends errors directly to the caller instead. At the time this was written, dead letters and Lambda destinations worked only for asynchronous calls.



AWS Notification Message

AWS Notifications <no-reply@sns.amazonaws.com> 12:38 AM (41 minutes ago)

to me

{"Records":[{"eventVersion":"2.1","eventSource":"aws:s3","awsRegion":"us-east-1","eventTime":"2019-02-23T23:35:27.893Z","eventName":"ObjectCreated:Post","userIdentity":{"principalId":"AWS:AROAIDCANCBNM3OPGU7SC:sam-test-1-ShowFormFunction-1A5BKGEPOUXVM"},"requestParameters":{"sourceIPAddress":"94.189.178.181"},"responseElements":{"x-amz-request-id":"A802908E21DE3129","x-amz-id-2":"8svxZ0RugM+OGbj2mwnEWOCH/sLVsjpl75lPnxP2SkoUq9RIGP4AmlTooWNz7dWcXzq4PHuyFFs="},"s3":{"s3SchemaVersion":"1.0","configurationId":"3f978d2a-14e7-4ccf-9a14-b6e51e11b719","bucket":{"name":"sam-test-1-uploads3bucket-u0o9wjguadyj","ownerIdentity":{"principalId":"A9BBUOASWG6NA"},"arn":"arn:aws:s3:::sam-test-1-uploads3bucket-u0o9wjguadyj"},"object":{"key":"565aeec0-b570-499d-b22f-585f22b3f1c4.jpg","size":14,"eTag":"1db6270cbe304dafe0dff5992c98f417","sequencer":"005C71D8BFAF2F3A4C"}}}]}

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:622887443104:sam-test-1-NotifyAdmins-RGUZ64CRVW77:0df8fdd9-5164-45f2-a877-69124e46f21b&Endpoint=gojkoa@gmail.com

Failed events are no longer lost, but sent by email to developers.

Now, this chapter is at an end and you can move to the interesting experiments section!