
Using AWS Lambda with Amazon CloudWatch and SNS to Implement a Slack Chat Bot

- Overview
- Start lab
- Task 1: Create Your Slack Account
- Task 2: Configure Incoming WebHooks For Slack
- Task 3: Create and Subscribe to an SNS Topic
- Task 4: Create a Lambda Function
- Task 5: Test your Lambda function
- Task 6: Create a CloudWatch Alarm
- Task 7: Test your Lambda function
- If you have more time...
- Conclusion
- End lab
- Additional Resources

Overview

In this lab you will build a chat bot for Slack, using a Lambda blueprint. Chat bots have the ability to interact with teams and users, respond to commands, and post notifications, giving all conversation participants visibility into team activities. You will build a bot that posts to your Slack channel when it receives CloudWatch alarms.

Topics covered

By the end of this lab, you will be able to:

- Create a Slack chat bot using a Lambda blueprint
- Configure the bot with a Slack webhook to post messages to a Slack channel

Technical knowledge prerequisites

To successfully complete this lab, you should be familiar with AWS Lambda and Amazon CloudWatch. Familiarity with Slack is helpful, though not required.

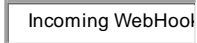
Task 1: Create Your Slack Account

First, you will need to create a Slack account so your bot can interact with it. You can do this prior to starting the lab to save time. If you already have a Slack account that you plan to use for this lab you can skip to the next [Section](#).

3. In a new browser tab, go to <https://slack.com/get-started#/createnew>.
4. Enter your email address, then click **Continue**.
5. Check your email, and use the confirmation code to confirm your account.
6. Choose **Create a Workspace**.
7. For *What's the name of your company or team?* Enter your company name, then click **Next**.

8. If asked, **Let's find the people you work with most**, choose **Skip this step**.
9. For *What's a project your team is working on?*, enter a project name, then click **Next**.
10. For *Who do you email most about ...*, click **skip this step**.
11. Click **Skip Step**.
12. Click your channel on the left.

Task 2: Configure Incoming WebHooks For Slack

13. Sign into your [Slack](#) account if you haven't already.
14. In the navigation menu on the left, click **Browse Slack**.
15. Click **Apps**.
16. In the Search box, type 
17. Under **Incoming WebHooks**, click **Add**.
18. Click **Add to Slack**.

19. Click the **Post to channel** drop-down and select **#general**.
20. Click **Add Incoming WebHooks integration**.
21. Copy the **Webhook URL** to a text editor.

You will need this later.

22. Scroll to the bottom, then click **Save Settings**.
23. Leave this browser tab/window open, you will return to it later.

Task 3: Create and Subscribe to an SNS Topic

In this task, you will create an SNS topic and subscribe to the topic using your email address.

Amazon Simple Notification Service (SNS) is a flexible, fully managed pub/sub messaging and mobile notifications service for coordinating the delivery of messages to subscribing endpoints and clients. With SNS you can fan-out messages to a large number of subscribers, including distributed systems and services, and mobile devices. It is easy to set up, operate, and reliably send notifications to all your endpoints – at any scale. You can get started using SNS in a matter of minutes using the AWS Management Console, AWS Command Line Interface, or using the AWS SDK with just three simple APIs. SNS eliminates the complexity and overhead associated with managing and operating dedicated messaging software and infrastructure.

24. In the **AWS Management Console**, on the **Services** menu, click **Simple Notification Service**.

25. In the left navigation pane, click the three line bars.

26. Click **Topics**.

27. Click **Create topic**

28. In the **Create topic** window, configure:

- **Type:** *Standard*
 - **Name:**
 - **Display name:**
 - Click **Create topic**
29. Click **Create subscription** then configure:
- **Protocol:** *Email*
 - **Endpoint:** *Enter your email address*
 - **Create subscription**
30. Check your email for the **slacknews** confirmation email.

It might take a few minutes for you to receive an email.

31. Click the **Confirm subscription** link in the **slacknews** email to confirm the subscription.

Congratulations! Your SNS topic has been created and you are subscribed to it.

Task 4: Create a Lambda Function

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

In this task, you will create a Lambda function from a blueprint that posts notifications to Slack, based on your SNS topic.

32. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.

33. Click **Create function**

34. Click **Use a blueprint**.

Blueprints are sample configurations of event sources and Lambda functions that do minimal processing for you, then customize it as needed. When you create a new AWS Lambda function, you can use a blueprint that best aligns with your scenario. There are several Slack bot blueprints available, including functions that handle a Slack slash command and echo details back to the user, and an Amazon SNS trigger that sends CloudWatch alarm notifications to Slack.

35. On the **Blueprints** page click the on the drop down box, then

enter

36. Press **Enter**.

37. Select **cloudwatch-alarm-to-slack-python** for Python3.7.

- **Function name:**
 - **Execution role:** *Create a new role with basic Lambda permissions*
39. Scroll down to the **SNS trigger** section, then configure:
- **SNS topic:** *slacknews*
40. In the **Environment variables** section (at the bottom), enter these values:
- **slackChannel:** Enter the channel name you defined earlier, which was:
 - **kmsEncryptedHookUrl:** Enter the Integration Webhook you copied to your text editor earlier in the lab

You can normally encrypt the URL for added security, but it is not necessary for this lab.

41. Click **Create function** (at the bottom of the page).

42. In the **Function code** section below, select **lambda_fuction.py** and delete all of the code in the code window.

43. Copy the Python code below and paste it into the Lambda code window.

```
import boto3
import json
import logging
import os

from base64 import b64decode
from urllib.request import Request, urlopen
from urllib.error import URLError, HTTPError

HOOK_URL = os.environ['kmsEncryptedHookUrl']
SLACK_CHANNEL = os.environ['slackChannel']

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    logger.info("Event: " + str(event))
    message = event['Records'][0]['Sns']['Message']
    logger.info("Message: " + str(message))

    alarm_name = message['AlarmName']
    new_state = message['NewStateValue']
    reason = message['NewStateReason']

    slack_message = {
        'channel': SLACK_CHANNEL,
        'text': "%s state is now %s: %s" % (alarm_name, new_state, reason)
    }

    req = Request(HOOK_URL, json.dumps(slack_message).encode('utf-8'))
    try:
        response = urlopen(req)
        response.read()
        logger.info("Message posted to %s", slack_message['channel'])
    except HTTPError as e:
        logger.error("Request failed: %d %s", e.code, e.reason)
    except URLError as e:
        logger.error("Server connection failed: %s", e.reason)
```

Examine the code. It is doing the following:

- Retrieving information about the CloudWatch Alarm

- Sending a message to the Slack channel with the state of the alarm

44. Click **Deploy** at the top of the page.

45. Choose **Configuration** tab, and click **General configuration**, select Edit:

- **Timeout:** *10 sec*

46. Click **Save**

Congratulations! You have created and configured your Lambda function.

Task 5: Test your Lambda function

47. Choose the **Test** tab, then configure:

48. Choose **Test** and then configure:

- **Event name:**
- **Template:** Search for and select:

49. Delete all of the code in the window.

50. Copy the JSON code snippet below into the code window.


```

{
  "Records": [
    {
      "EventVersion": "1.0",
      "EventSubscriptionArn": "arn:aws:sns:EXAMPLE",
      "EventSource": "aws:sns",
      "Sns": {
        "SignatureVersion": "1",
        "Timestamp": "1970-01-01T00:00:00.000Z",
        "Signature": "EXAMPLE",
        "SigningCertUrl": "EXAMPLE",
        "MessageId": "95df01b4-ee98-5cb9-9903-4c221d41eb5e",
        "Message": {
          "AlarmName": "SlackAlarm",
          "NewStateValue": "OK",
          "NewStateReason": "Threshold Crossed: 1 datapoint (0.0) was not
greater than or equal to the threshold (1.0).",
        },
        "MessageAttributes": {
          "Test": {
            "Type": "String",
            "Value": "TestString"
          },
          "TestBinary": {
            "Type": "Binary",
            "Value": "TestBinary"
          }
        },
        "Type": "Notification",
        "UnsubscribeUrl": "EXAMPLE",
        "TopicArn": "arn:aws:sns:EXAMPLE",
        "Subject": "TestInvoke"
      }
    }
  ]
}

```

51. Look at the code that you just pasted into the window.

You can see that “AlarmName” in the JSON code is “SlackAlarm”. You will create this alarm in the next part of the lab.

52. Click

53. Click

You should see **Execution results: succeeded (log)**

Task 6: Create a CloudWatch Alarm

In this task you will create a CloudWatch alarm to notify your SNS topic when the alarm is triggered.

Amazon CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, set alarms, and automatically react to changes in your AWS resources. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate. You can use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health. You can use these insights to react and keep your application running smoothly.

54. In the **AWS Management Console**, on the **Services** menu, click **CloudWatch**.

55. In left navigation pane, click **Alarms** and then **All Alarms**.

56. Select **Create alarm**.

57. Choose **Select metric**

58. Under **Metrics**, select **Lambda**.

59. Select **Across All Functions**.

60. In the search box, enter **Errors** and select it.

61. Choose **Select metric**

62. Under **Specify metric and conditions**, configure:

- **Statistic:** *Sum*
- **Period:** *1 Minute*

63. Scroll down to the **Conditions** section

- **Whenever Error is...** select **Greater/Equal**

- **than:** enter 1
64. Click **Next**
 65. For **Notification**, configure:
 - **Whenever this alarm state is...** select **OK**
 - **Select an SNS Topic:** Select an existing SNS topic
 - **Send a notification to...** in the text box, select *slacknews*
 66. Click **Next**
 67. **Alarm name** enter
 68. **Alarm Description** enter
 69. Click **Next**
 70. Click **Create alarm**

The alarm state will change from Insufficient to OK. This may take a few minutes.

Task 7: Test your Lambda function

71. In the **AWS Management Console**, on the **Services** menu, click **Lambda**.
72. Click **Slackfunction**.
73. Click **Test**
74. View the log messages by:
 - clicking on (**logs**)

This should navigate you to CloudWatch -> Log groups -> /aws/lambda/Slackfunction

An SNS notification has been sent to your email address and your chat bot has posted the notification in your Slack Channel.

75. Return to the web browser tab/window with Slack.

Load

`https://<your-team-domain>.slack.com/messages`

the

n select your channel to see the notifications.

Congratulations!

You have created your very own Slack bot using AWS Lambda. You can check your Slack feed on your mobile device to see the messages there as well.

Conclusion

Congratulations! You now have successfully:

- Created a Slack chat bot using a Lambda blueprint
- Configured the bot with a Slack webhook and successfully posted messages to a Slack channel
- Tested the chat bot and verified that it posts CloudWatch alarms to your Slack channel