Alerts

Service: eventbridge

Make a event with **Rule**

Create a custom rule which is

**Code:**

{

"detail": {

"group": ["service:react-docker"],

"lastStatus": ["STOPPED"],

"stoppedReason": [{

"anything-but": {

"prefix": "Scaling activity initiated by (deployment"

}

}]

},

"detail-type": ["ECS Task State Change"],

"source": ["aws.ecs"]

}

**In this code just change your service name.**

Target in with aws lambda function.

Go to lambda function trigger it with aws eventbridge.

Code :

import json

import http.client

from urllib.parse import urlparse

# Create a set to track processed events

processed\_events = set()

def lambda\_handler(event, context):

# Create a unique identifier for the event (you can customize this based on event attributes)

event\_id = event["detail"].get("eventId")

# Check if the event has already been processed

if event\_id in processed\_events:

return {

'statusCode': 200,

'body': json.dumps('Event already processed; no action taken.')

}

# Check if the event matches the failure criteria

if event["detail"]["lastStatus"] == "STOPPED" and \

event["detail"]["group"] == "service:react-docker" and \

"Scaling activity initiated by (deployment" not in event["detail"]["stoppedReason"]:

# Slack Webhook URL

slack\_webhook\_url = "https://hooks.slack.com/services/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"

# Custom message for the failed ECS task

slack\_message = {

"text": "ECS task has failed. Check ECS for more details."

}

# Parse the webhook URL

parsed\_url = urlparse(slack\_webhook\_url)

host = parsed\_url.hostname

path = parsed\_url.path

# Connect to Slack and send the message

connection = http.client.HTTPSConnection(host)

headers = {'Content-Type': 'application/json'}

connection.request("POST", path, body=json.dumps(slack\_message), headers=headers)

response = connection.getresponse()

# Check the response

if response.status == 200:

# Mark the event as processed

processed\_events.add(event\_id)

return {

'statusCode': 200,

'body': json.dumps('Message sent to Slack successfully!')

}

else:

return {

'statusCode': response.status,

'body': json.dumps(f"Failed to send message to Slack: {response.reason}")

}

else:

return {

'statusCode': 200,

'body': json.dumps('Event does not match failure criteria; no Slack message sent.')

}

**In this code change your slack webhook url**

Its prepare if task is failed the message appear on slack.

**For successful deployment**

**Event is :**

{

"source": ["aws.ecs"],

"detail-type": ["ECS Task State Change"],

"detail": {

"lastStatus": ["RUNNING"],

"desiredStatus": ["RUNNING"],

"clusterArn": ["arn:aws:ecs:us-east-1:221082201369:cluster/express-cluster"],

"group": ["service:react-docker"]

}

}**My lambda function is :**

import json

import http.client

from urllib.parse import urlparse

# Replace with your Slack webhook URL

SLACK\_WEBHOOK\_URL = "https://hooks.slack.com/services/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"

def lambda\_handler(event, context):

# Log the event for debugging

print("Received event:", json.dumps(event, indent=2))

# Extract ECS task status and details

detail = event.get('detail', {})

last\_status = detail.get('lastStatus')

desired\_status = detail.get('desiredStatus')

# Only send alert if the task has reached the desired status

if last\_status == 'RUNNING' and last\_status == desired\_status:

# Set message content to a simple success notification

message = "My ECS task is successfully deployed."

# Prepare the Slack payload

slack\_data = json.dumps({"text": message})

# Parse the Slack webhook URL

url = urlparse(SLACK\_WEBHOOK\_URL)

conn = http.client.HTTPSConnection(url.netloc)

# Send the POST request to Slack

conn.request(

"POST", url.path,

body=slack\_data,

headers={'Content-Type': 'application/json'}

)

# Get the response

response = conn.getresponse()

if response.status != 200:

raise ValueError(

f"Request to Slack returned an error {response.status}, the response is:\n{response.read().decode()}"

)

conn.close()

return {

'statusCode': 200,

'body': json.dumps('Alert sent to Slack!')

}