**ECS Alerts with Slack Integration for failure**

**1. Service Setup**

**Event Bridge Configuration**

1. **Navigate to Amazon EventBridge**:
   * Open the [AWS Management Console](https://aws.amazon.com/console/) and go to **EventBridge**.
2. **Create a Rule**:
   * Name the rule appropriately (e.g., ECS-Failure-Alert).
   * Select **Event Source** as **AWS Services** and choose **ECS**.
3. **Add Custom Event Pattern**: Copy and paste the following JSON into the **Event Pattern** section, updating the service name (e.g., service:react-docker):

**Code for event bridge**

{

"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"]

}

1. **Target Configuration**:
   * Choose **AWS Lambda Function** as the target.
   * Select or create a Lambda function to handle the alert.

**2. Lambda Function for Failure Alert**

1. **Set Up the Lambda Function**:
   * Go to the **AWS Lambda Console** and create a new function.
   * Name it appropriately (e.g., ECSFailureAlert).
   * Use the following code, ensuring to update the Slack Webhook URL:

**Code for ecs failure alert**

import json

import http.client

from urllib.parse import urlparse

processed\_events = set()

def lambda\_handler(event, context):

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

if event\_id in processed\_events:

return {'statusCode': 200, 'body': 'Event already processed.'}

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 = "YOUR\_SLACK\_WEBHOOK\_URL"

slack\_message = {"text": "ECS task has failed. Check ECS for details."}

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

host, path = parsed\_url.hostname, parsed\_url.path

connection = http.client.HTTPSConnection(host)

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

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

response = connection.getresponse()

if response.status == 200:

processed\_events.add(event\_id)

return {'statusCode': 200, 'body': 'Message sent to Slack!'}

else:

return {'statusCode': response.status, 'body': f"Failed to send message: {response.reason}"}

else:

return {'statusCode': 200, 'body': 'No action required.'}

* + Replace "YOUR\_SLACK\_WEBHOOK\_URL" with your Slack Webhook URL.

1. **Attach EventBridge Trigger**:
   * In the Lambda console, go to **Configuration** > **Triggers**.
   * Add EventBridge as a trigger and link it to the rule created earlier.

**Expected Results**

* **Failure Scenario**: If an ECS task fails, a message will be sent to Slack, notifying the team