```
---
AWSTemplateFormatVersion: '2010-09-09'

Parameters:
  EndpointGroupARN:
    Type: String
    Description: Accelerator Endpoint ARN (not the Accelerator ARN),
make sure it is configured for endpoints in this region; it should
look like 'arn:aws:globalaccelerator::123456789012:accelerator/
c9d8f18d-e6a7-4f28-ae95-261507146530/listener/461df876/endpoint-group/
c3770cbbf005'
  EndpointWeight:
    Type: Number
    Default: 128
    MinValue: 1
    MaxValue: 255
    Description: Endpoint Weight for the EC2 instances in the
Autoscaling group (integer between 1 and 255).
  AutoscalingGroupName:
    Type: String
    Description: Autoscaling group name
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: AWS Global Accelerator parameters
        Parameters:
          - EndpointGroupARN
          - EndpointWeight
      -
        Label:
          default: Autoscaling Group Name
        Parameters:
          - AutoscalingGroupName
Resources:
  LambdaRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
              - 'lambda.amazonaws.com'
          Action:
            - sts:AssumeRole
      Policies:
        -
```

```
          PolicyName: AutoScaling-GlobalAccelerator-Lambda-Policy
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              -
                Effect: Allow
                Action:
                  - autoscaling:CompleteLifecycleAction
                Resource:
                  - '*'
              - Effect: Allow
                Action:
                  - logs:CreateLogGroup
                  - logs:CreateLogStream
                  - logs:PutLogEvents
                Resource:
                  - arn:aws:logs:*:*:*
              - Effect: Allow
                Action:
                  - globalaccelerator:UpdateEndpointGroup
                  - globalaccelerator:DescribeEndpointGroup
                Resource:
                  - !Ref EndpointGroupARN
  LifecycleHook:
      Type: AWS::AutoScaling::LifecycleHook
      Properties:
        AutoScalingGroupName: !Ref AutoscalingGroupName
        LifecycleTransition: autoscaling:EC2_INSTANCE_TERMINATING
        HeartbeatTimeout: 30
  Function:
    Type: AWS::Lambda::Function
    Properties:
      Description: Lambda function to automatically add/remove EC2
Endpoints to/from an AWS Global Accelerator Endpoint Group based on
Autoscaling group events.
      Code:
        ZipFile: |
            '''
            # Copyright 2019 Amazon.com, Inc. or its affiliates. All
Rights Reserved.
            # Licensed under the Apache License, Version 2.0 (the
"License"). A copy of the License is located at http://aws.amazon.com/
apache2.0/
            #
            # Description: This Lambda function updates an AWS Global
Accelerator Endpoint Group based on Auto Scaling group Events.
            '''

            import boto3
            import hashlib
```

```
            import json
            import logging
            import os
            import time

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

            aga_client = boto3.client('globalaccelerator',
region_name='us-west-2')

            EC2_LAUNCHING = 'EC2 Instance Launch Successful'
            EC2_TERMINATING = 'EC2 Instance-terminate Lifecycle
Action'
            ENDPOINT_GROUP_ARN = os.environ['EndpointGroupARN']

            if (os.environ.get('EndpointWeight') != None) and
os.environ['EndpointWeight'].isdigit() and
int(os.environ['EndpointWeight']) < 256:
                ENDPOINT_WEIGHT = int(os.environ['EndpointWeight'])
            else:
                ENDPOINT_WEIGHT = 128

            def check_response(response_json):
                if response_json.get('ResponseMetadata',
{}).get('HTTPStatusCode') == 200:
                    return True
                else:
                    return False

            def list_endpoints():
                response = aga_client.describe_endpoint_group(
                    EndpointGroupArn = ENDPOINT_GROUP_ARN
                )
                return response

            def updated_endpoints_list(detail_type, instance_id):
                endpoints = []
                response = list_endpoints()

                if detail_type == EC2_LAUNCHING:
                    for EndpointID in response['EndpointGroup']
['EndpointDescriptions']:
                        result = {'EndpointId':
EndpointID['EndpointId'],'Weight': EndpointID['Weight']}
                        endpoints.append(result)
                    endpoints.append({'EndpointId':
instance_id,'Weight': ENDPOINT_WEIGHT}) # Add the endpoint

                elif detail_type == EC2_TERMINATING:
```

```
                    for EndpointID in response['EndpointGroup']
['EndpointDescriptions']:
                        if EndpointID['EndpointId'] != instance_id: #
Remove the endpoint
                            result = {'EndpointId':
EndpointID['EndpointId'],'Weight': EndpointID['Weight']}
                            endpoints.append(result)
                return endpoints

            def update_endpoint_group(detail_type, instance_id):
                try:
                    response = aga_client.update_endpoint_group(
                        EndpointGroupArn = ENDPOINT_GROUP_ARN,
                        EndpointConfigurations =
updated_endpoints_list(detail_type, instance_id)
                        )
                    if check_response(response):
                        logger.info("The endpoint group has been
updated: %s", response)
                        return response['EndpointGroup']
['EndpointDescriptions']
                    else:
                        logger.error("Could not update the endpoint
group: %s", response)
                        return None
                except Exception as e:
                    logger.error("Could not update the endpoint group:
%s", str(e))
                    return None

            def lambda_handler(event, context):
                try:
                    logger.info(json.dumps(event))
                    message = event['detail']
                    detail_type = event['detail-type']
                    if 'AutoScalingGroupName' in message:
                        instance_id = message['EC2InstanceId']
                        response = update_endpoint_group(detail_type,
instance_id)
                        if response != None:
                            logging.info("Lambda executed correctly")
                        elif detail_type == EC2_TERMINATING: # Abandon
the lifecycle hook action
                            asg_client = boto3.client('autoscaling')
                            abandon_lifecycle =
asg_client.complete_lifecycle_action(
                                LifecycleHookName =
message['LifecycleHookName'],
                                AutoScalingGroupName =
message['AutoScalingGroupName'],
```

```
                                LifecycleActionResult = 'ABANDON',
                                InstanceId = instance_id
                                )
                            if check_response(abandon_lifecycle):
                                logger.info("Lifecycle hook abandoned
correctly: %s", response)
                            else:
                                logger.error("Lifecycle hook could not
be abandoned: %s", response)
                    else:
                        logging.error("No valid JSON message: %s",
parsed_message)
                except Exception as e:
                    logging.error("Error: %s", str(e))

      Environment:
        Variables:
          EndpointGroupARN: !Ref EndpointGroupARN
          EndpointWeight: !Ref EndpointWeight
      Handler: index.lambda_handler
      ReservedConcurrentExecutions: 1
      Role: !GetAtt LambdaRole.Arn
      Runtime: python3.7
      Timeout: 30
  Permission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: "lambda:InvokeFunction"
      FunctionName: !GetAtt Function.Arn
      Principal: events.amazonaws.com
  Rule:
    Type: AWS::Events::Rule
    Properties:
      EventPattern:
          source:
            - aws.autoscaling
          detail-type:
            - 'EC2 Instance-terminate Lifecycle Action'
            - 'EC2 Instance Launch Successful'
          detail:
            AutoScalingGroupName:
              - !Ref AutoscalingGroupName
      Targets:
      - Arn: !GetAtt Function.Arn
        Id: target
```