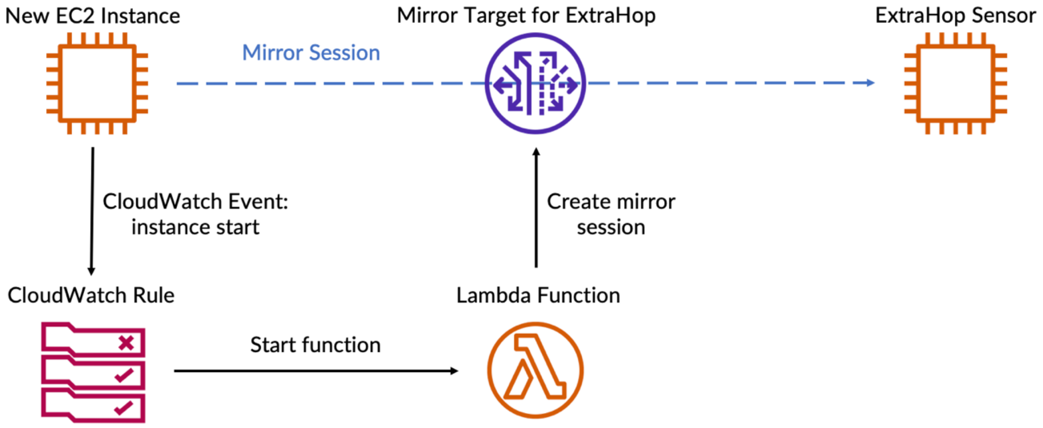
Automate traffic mirroring with AWS Lambda

Published: 2021-08-31

You can configure a Lambda function to automatically mirror traffic from EC2 instances to your ExtraHop sensors deployed in AWS. We recommend that you configure some form of automation to ensure that all of your EC2 instances are monitored by the ExtraHop system.

This guide provides instructions for configuring and installing an example Lambda function that is available on the ExtraHop GitHub repository. Here's how the function works:



The following steps outline the individual processes described in the above diagram:

1. Every time an EC2 instance starts running, a CloudWatch rule runs the Lambda function.
2. The function checks to see if a mirror session exists for the new EC2 instance.
3. If there is no mirror session for the instance, the function selects which ExtraHop sensor it will mirror traffic to.
    a. First, the function searches for sensors that are in the same Availability Zone as the traffic mirror.

        Note: If the function is unable to find any sensors in the same Availability Zone, the LOCAL_ZONE_ONLY variable determines whether the function will select sensors outside of the Availability Zone. Mirroring traffic across Availability Zones carries an additional charge per GB. See the AWS documentation for more information.

    b. Next, the function filters out sensors with security groups that block traffic from the EC2 instance.
    c. Then the function filters out sensors that are on VPCs with ACLs that block traffic from the EC2 instance.
    d. After the function has a list of valid sensors, the function searches for the sensor with the lowest number of mirror sessions to ensure that mirror sessions are evenly distributed.
4. Finally, the function creates a mirror session that forwards traffic from the EC2 instance to the selected sensor.

Before you begin

- Create traffic mirror targets for each of your ExtraHop sensors. Note the IDs of the targets; you will need to add the IDs to the script.
- Create a traffic mirror filter that determines what traffic will be mirrored to your sensors. Note the ID of the mirror filter; you will need to add the ID to an environment variable in the Lambda function.

Retrieve and install the example script

1. Go to the ExtraHop code-examples GitHub repository and click **lambda_traffic_mirror**.
2. Copy the `lambda_traffic_mirror.py` file to your local machine.

3. Add the `lambda_traffic_mirror.py` file to a zip file with the netaddr Python module.

   The script imports the `netaddr` Python module, which is not available to Lambda functions by default. For information about creating a zip file to import third-party libraries into Lambda, see the AWS documentation.
4. In AWS, create a Lambda function.

   For more information about creating Lambda functions, see the AWS documentation.
5. On the Lambda function page, click **Actions** and select **Upload a .zip** file.
6. Select the zip file you created.

Configure the Lambda function

Before you can run the example Lambda function, you must assign the necessary permissions to the function and configure the function to reference information from your AWS environment. Finally, you can configure a CloudWatch rule to run the function automatically.

1. Assign the following permissions to the example Lambda function:

   - CreateTags
   - CreateTrafficMirrorSession
   - DescribeInstances
   - DescribeNetworkInterfaces
   - DescribeTrafficMirrorSessions
   - DescribeTrafficMirrorTargets
   - DescribeSecurityGroups
   - DescribeNetworkAcls

   For information about configuring Lambda permissions, see the AWS tutorial here.
2. In the `lambda_function.py` file, replace the `targets` environment variable with the IDs of the traffic mirror targets for your ExtraHop sensors.
3. Add the ID of the mirror filter you created as a Lambda environment variable named `filter_id`.

   For more information about Lambda environment variables, see the AWS documentation.
4. Configure a CloudWatch rule to start the Lambda function whenever an EC2 instance starts running.

   The CloudWatch rule must run according to the following event pattern:

   ```
   {
     "source": [
       "aws.ec2"
     ],
     "detail-type": [
       "EC2 Instance State-change Notification"
     ],
     "detail": {
       "state": [
         "running"
       ]
     }
   }
   ```

   For more information about starting Lambda functions with CloudWatch rules, see the AWS documentation.