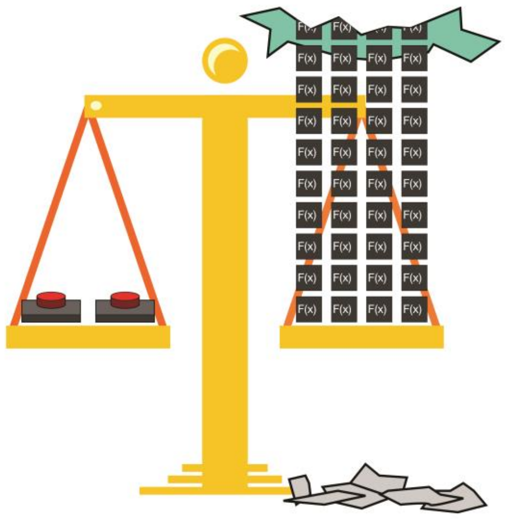


How much does a few minutes cost?

Cisco Dynamic Reverse Proxy

Saving Minutes to Save Millions. A Guide.

Will Rooney William.rooney@ucdenver.edu
Cecil Hutchings Cecil.hutchings@ucdenver.edu
Matthew King Matthew.king@ucdenver.edu

Table of Contents

Summary

The Cisco reverse proxy is secure static web server that serves web applications to Cisco developers. Automation techniques are used to transform the static reverse proxy into a dynamic reverse proxy.

Each web application is hosted on Amazon Web Services (the cloud). Applications are frequently added and removed to save money on cloud resource costs; applications may not be often used and therefore do not need to be hosted on the cloud at all times. The existing static reverse proxy does not update to reflect this. A Cisco developer typically spends a few minutes manually updating the existing proxy every day. Over time this adds up to hours of skilled labor lost to a trivial task. This problem is resolved by automating the tasks a developer must take to update the proxy every time an application is created or deleted.

The Cisco Dynamic Reverse Proxy prototype utilizes Amazon Web Services (AWS) and the python programming language to automate these tasks. The prototype monitors for instances (web applications), as they are added and removed the prototype logs the results to cloud storage. The final product will update the dynamic reverse proxy in fractions of a second.

Transforming a static proxy into a dynamic proxy automates a task that takes five minutes of skilled labor. The resulting operation takes fractions of a second billed at the cost of cloud resources. Over time automation techniques like this save millions of dollars.

Configuring AWS S3 & CloudTrail

This guide assumes general familiarity with AWS S3 and CloudTrail. For more information see: http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-getting-started.html

Tutorial: Invoke Lambda on a State change or API call of an Amazon EC2 Instance Using CloudWatch Events

The following tutorial was taken from Amazons “Log EC2 Instance State” tutorial with slight modifications.
http://docs.aws.amazon.com/AmazonCloudWatch/latest/events/LogEC2InstanceState.html

You can create a AWS Lambda function that logs API calls for an Amazon EC2 instance. You can choose to create a rule that runs the function whenever there is a runInstances or terminateInstances API call. In this tutorial, you log the launch/termination of any new instance using lambda.

Step 1: Create an AWS Lambda Function

Create a Lambda function to log the API call events. You specify this function when you create your rule.

To create a Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. If you are new to Lambda, you see a welcome page; choose **Get Started Now**; otherwise, choose **Create a Lambda function**.
3. On the **Select blueprint** page, type `hello` for the filter, and then choose the **hello-world** blueprint.
4. On the **Configure triggers** page, choose **Next**.
5. On the **Configure function** page, do the following:
    1. Type a name and description for the Lambda function. (For example, name the function "LogEC2InstanceAPICall".)
    2. Edit the sample code for the Lambda function. For example (Python 3.6):

```
import boto3
import time
import json

ec2 = boto3.resource('ec2')

def lambda_handler(event, context):
    """
    Handle EC2 Run and Terminate Instance API Calls.
    Log Event data to S3 and POST event data to a reverse proxy.
    """
    #print("Received event: " + json.dumps(event, indent=2))

    # Create a timestamp to ensure unique filename for log written to S3
    timestamp = time.strftime("%Y%m%d-%H%M%S")

    # upload the event data to S3
    bucket = 'my-bucket'
    fileName = timestamp

    # Event Dict
    eventOutput = {}
    eventOutput['Events'] = event

    try:
        if event['detail-type'] == 'AWS API Call via CloudTrail':
            # API Call to RunInstances or TerminateInstances
            eventName = event['detail']['eventName']
            if eventName == 'RunInstances':
                # RunInstances API Call
            elif eventName == 'TerminateInstances':
                # TerminateInstances API Call
            else:
                raise Exception('Invalid event: ' + eventName)
        else:
            raise Exception('Invalid detail-type: '+event['detail-type'])
    except Exception as e:
        # Error Parsing Event - Log Error details
```

```
            print(e)
            errorDetails = {'Exception': str(e), 'EventInput': event}
            eventOutput['Events'].append({"Error": errorDetails})

        # Log eventOutput to S3
        path = '/tmp/' + fileName
        with open(path, 'w') as f: json.dump(eventOutput, f, indent=2)
        s3.meta.client.upload_file(path, bucket, fileName)
```

*A further detailed example is included with this guide: "cmw_instance_cloudwatch_lambda.py"

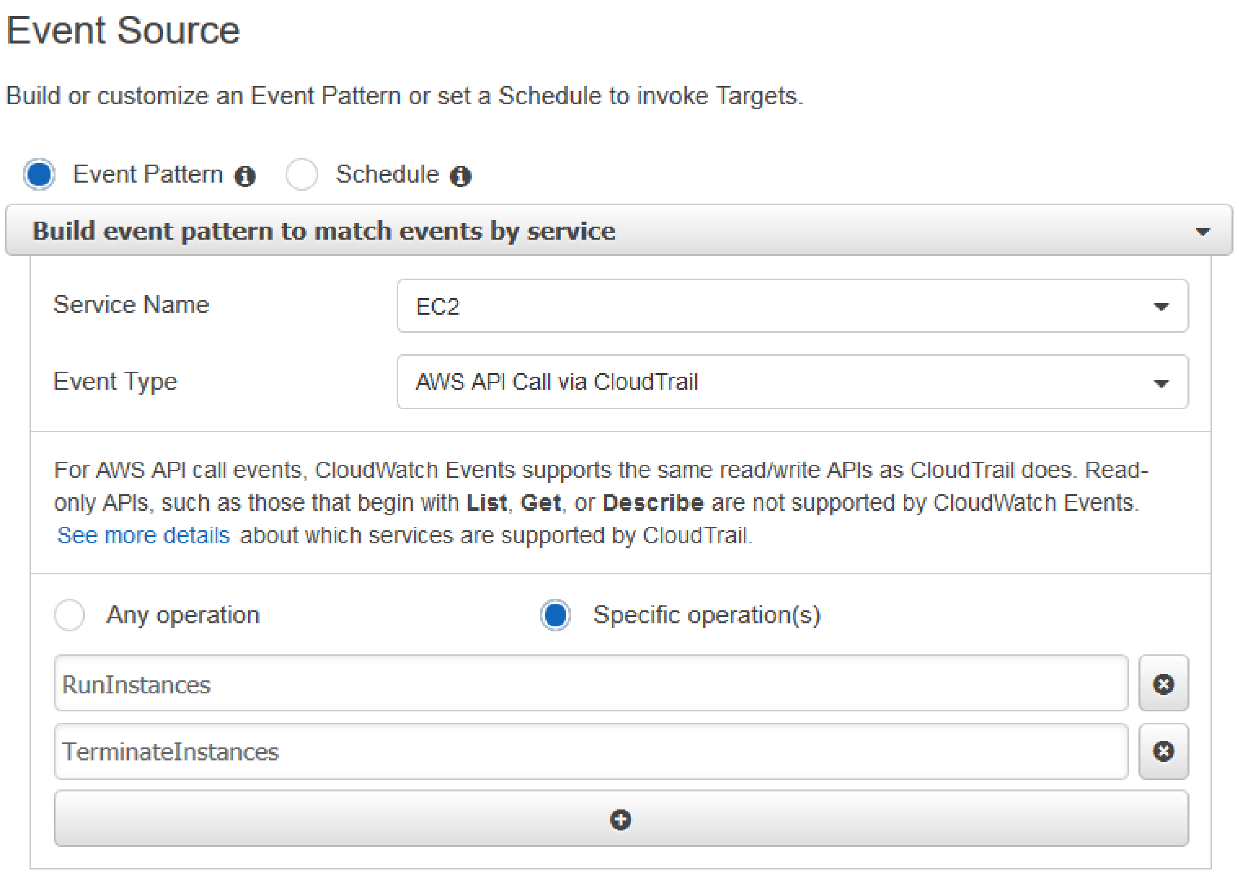
    3. For **Role**, choose **Choose an existing role** and then choose your basic execution role from **Existing role**. Otherwise, create a new basic execution role.
    4. Choose **Next**.
6. On the **Review** page, choose **Create function**.

Step 2: Create a Rule

Create a rule to run your Lambda function whenever you launch/terminate an Amazon EC2 instance.

To create a CloudWatch Events rule

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Events**, **Create rule**.
3. For **Event source**, do the following:
    1. Choose **Event Pattern**.
    2. Choose **Build event pattern to match events by service**.
    3. Choose **EC2** and then choose **AWS API Call via CloudTrail**.
    4. Choose **Specific operations(s)** and then add two operations:
        1. **RunInstances**
        2. **TerminateInstances**


Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

[✓ Event Pattern] Schedule

Build event pattern to match events by service

Service Name: EC2

Event Type: AWS API Call via CloudTrail

For AWS API call events, CloudWatch Events supports the same read/write APIs as CloudTrail does. Read-only APIs, such as those that begin with **List**, **Get**, or **Describe** are not supported by CloudWatch Events. See more details about which services are supported by CloudTrail.

Any operation [✓ Specific operation(s)]

RunInstances

TerminateInstances

4. For **Targets**, choose **Add target** and then choose **Lambda function**.
5. For **Function**, select the Lambda function that you created.
6. Choose **Configure details**.
7. For **Rule definition**, type a name and description for the rule.
8. Choose **Create rule**.

Step 3: Test the Rule

To test your rule, launch an Amazon EC2 instance. After waiting a few minutes for the instance to launch and initialize, you can verify that your Lambda function was invoked.

To test your rule by launching an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Launch an instance. For more information, see Launch Your Instance in the *Amazon EC2 User Guide for Linux Instances*.
3. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
4. In the navigation pane, choose **Events**, **Rules**, select the name of the rule that you created, and choose **Show metrics for the rule**.
5. To view the output from your Lambda function, do the following:
    1. In the navigation pane, choose **Logs**.
    2. Choose the name of the log group for your Lambda function (/aws/lambda/*function-name*).

    3. Choose the name of log stream to view the data provided by the function for the instance that you launched.

6. (Optional) When you are finished, you can open the Amazon EC2 console and stop or terminate the instance that you launched. For more information, see Terminate Your Instance in the *Amazon EC2 User Guide for Linux Instances*.

Appendix

Links

- https://d0.awsstatic.com/whitepapers/compliance/AWS_Security_at_Scale_Logging_in_AWS_Whitepaper.pdf
- http://docs.aws.amazon.com/lambda/latest/dg/with-cloudtrail-example-prepare.html
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-create-a-trail-using-the-console-first-time.html#advanced-settings-for-your-trail
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-create-and-update-a-trail.html
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-create-and-update-a-trail.html
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/use-the-cloudtrail-processing-library.html
- https://boto3.readthedocs.io/en/latest/reference/services/cloudwatch.html
- https://aws.amazon.com/cloudwatch/pricing/
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-event-reference-record-contents.html
- http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudwatch-alarms-for-cloudtrail.html#cloudwatch-alarms-for-cloudtrail-ec2-instance-changes
- http://docs.aws.amazon.com/AmazonCloudWatch/latest/events/LogEC2InstanceState.html
- https://boto3.readthedocs.io/en/latest/reference/services/ec2.html#instance