**Configuring CloudWatch for EC2 Alarms and Testing with CloudTrail**

### Introduction

In this Lab Step you will configure a more realistic use case: Configuring CloudWatch to isolate stop and terminate EC2 instances from the CloudTrail logs sent to the CloudWatch log group. When captured, it will trigger an alarm that sends an email notification.

### Instructions

1. Navigate to [CloudWatch Log groups](https://us-west-2.console.aws.amazon.com/cloudwatch/home?region=us-west-2#logsV2:log-groups), and select the radio button for the Log Group you just created.

2. Click **Create Metric Filter**.

* For the **Filter Pattern**, enter:  { ( $.eventName = StopInstances ) || ( $.eventName = TerminateInstances ) || ( $.eventName = RunInstances ) }
* Click **Test Pattern**. Although the Results will find 0 matches against the default log data to test against, it will confirm the syntax for your filter is correct and the search was performed.

This pattern will search for stop and terminate instance events.

3. Click the**next** button.

* For the **Filter Name** field enter Stopped, Terminated and Ran Instances
* In the **Metric Details** section, enter CloudTrailMetrics for the **Metric Namespace**
* For the **Metric Name** enter StopTerminateRunInstanceCount
* Click the **Metric Value** field. Enter 1 for the value.

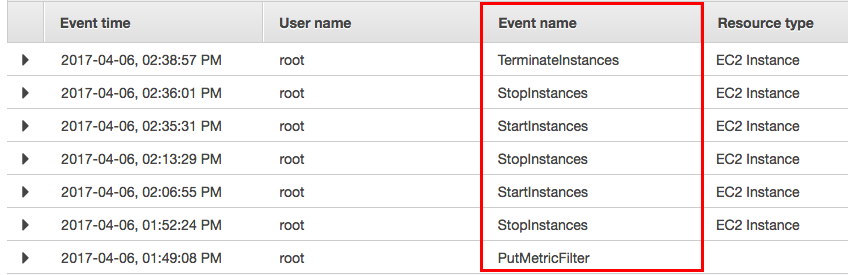
4. Finally, click **Create metric filter**. You will see a confirmation similar to the following:



Now you are ready to test it out. To recap, what will it mean if the metric works?  Essentially, it means that CloudTrail captures events that include Instance state changes. Further, CloudTrail is delivering logs not just to S3 but also to CloudWatch. CloudWatch monitors for both standard and custom metrics. In our case, CloudWatch will look for Instances that transition to either stopped or terminated status. If it catches these the results will be graphed. Next you will generate EC2 related events, confirm capture in CloudTrail, and then setup a metric and alarm for CloudWatch to trigger an email notification.

5. Return to [EC2 instances](https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:), select your running Instance and click **Actions > Instance State > Stop**. Because a t2.micro instance starts and stops pretty quickly, start and stop it again.  Finally, terminate the instance. This will simulate real usage and test the CloudTrail and CloudWatch services cooperating to flag a key metric. (Stopped and terminated instances in our example.)

6. Return to [CloudTrail](https://us-west-2.console.aws.amazon.com/cloudtrail/home?region=us-west-2#/dashboard). For the Trail created earlier, you should see the **PutMetricFilter**. The example below also shows several **StopInstances/StartInstances** events, and finally a **TerminateIntances** event:

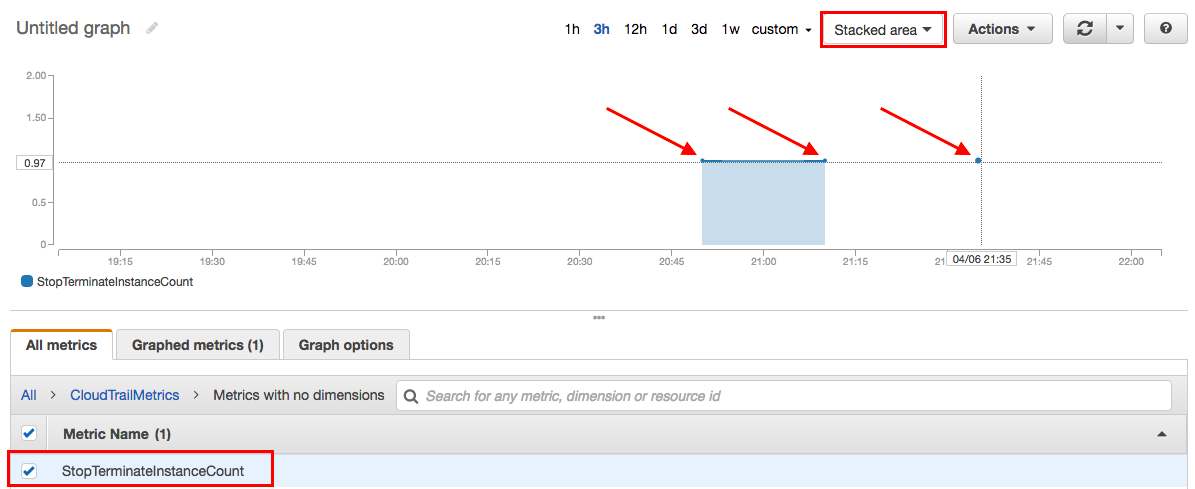


Reminder: There is a delay, and you will not see all events immediately.

Note that if you think of this process as a pipeline, CloudTrail comes before CloudWatch in the pipeline. Therefore, until you see the **PutMetricFilter** and **Stop/Start/TerminateInstances** events in the **API activity history** you will not find them in CloudWatch. (This may be a good time to grab some water, coffee or tea... then continue when you see the Stop/TerminateInstances events.) With the first part of the process flow proven to be working (CloudTrail), you are ready to check the second part of the chain in the process (CloudWatch).

7. Return to [CloudWatch](https://us-west-2.console.aws.amazon.com/cloudwatch/home?region=us-west-2). Click **Metrics** in the left menu. There are many pre-built metrics, but only two custom metrics... the two you created earlier.

8. Click **CloudTrailMetrics** in the **Custom Namespace**. Click **Metrics with no dimensions**, then select the **StopTerminateInstanceCount** metric name. In the drop-down menu at the top of the graph, change **Line** to **Stacked area**:

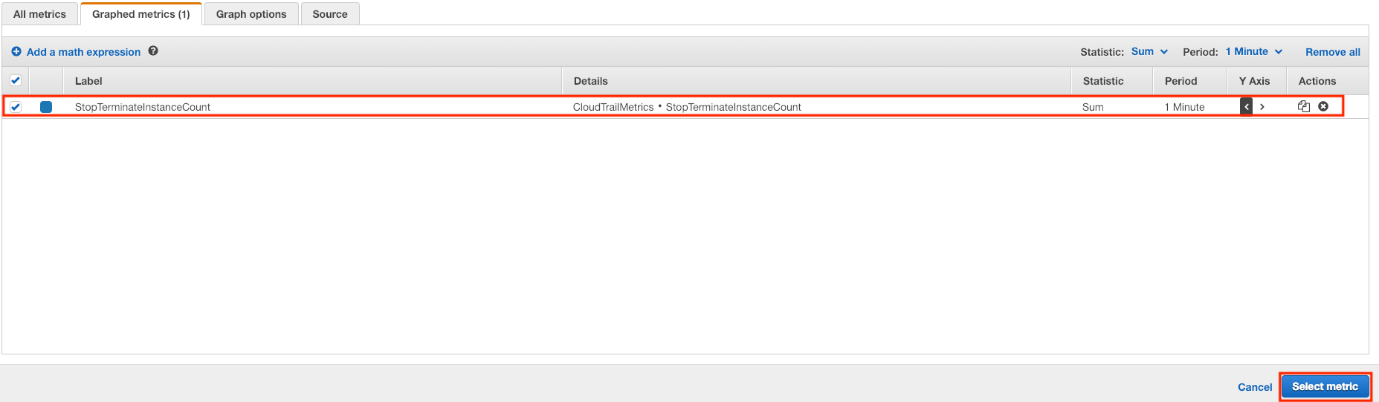


Note: Your graph will look similar but not identical to the example above. Depending on timing, you may just see a single plotted point. For example, coresponding with a **StopInstances** event.

The graph shows that CloudTrail and CloudWatch are working together correctly. Three points are plotted, in conjunction with the two stop and one terminate instance. (In our example, the first stop instance was not caught in time to be graphed.) Obviously this is a training environment, not a production environment where the volume and key metrics to apply and plot are more interesting. In a test or even small production environment setting up CloudTrail and CloudWatch in a similar manner might be fine. The Operations or Development teams might configure and watch things in a manual fashion similar to this lab exercise. In a more complex production environment, you would likely want a configuration that is more proactive. That is, when a certain metric or threshold is violated, an automated alarm is triggered. AWS provides options for that too.

9. In the [CloudWatch](https://us-west-2.console.aws.amazon.com/cloudwatch/home?region=us-west-2) console, click **Alarms** in the left pane, then click the **Create Alarm** button. Note you can create an Alarm later in the process. In a previous Lab Step you created an Alarm immediately after saving the new Metric Filter.

10. Click **Select metric**, select CloudTrailMetrics under **Custom namespaces**, select Metrics with no dimensions. Here you've to choose the **StopTerminateInstanceCount** metric and select Sum as **Statistic** and 1 minute as **Period**. Now you can click the **Select metric** button:



Now that you have selected the metric, you can define the parameters of the alarm itself.

11. Set the section as described:

* **Threshold type**: **Static**
* **Whenever StopTerminateInstanceCount is...**: **Greater/Equal**
* **than...**: 1

Click **Next**.

12. Set the section as described;

* **Whenever this alarm state is...**: **In Alarm**
* **Select an SNS topic**: **Create new topic**
* **Create a new topic…**: StopTermInstances
* **Email endpoints that will receive the notification…**: your personal email

Click **Create topic** and then **Next**.

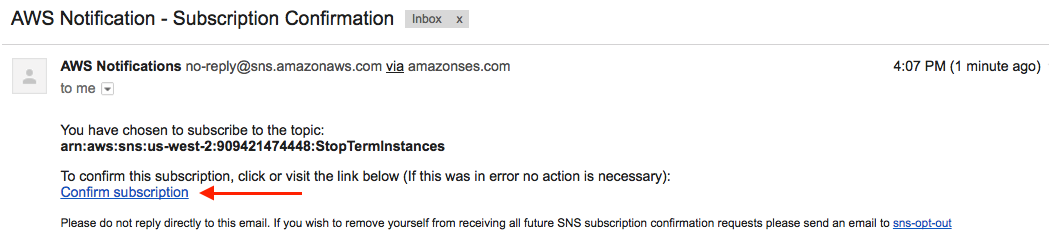
13. Set the section as described:

* **Define a unique name**: StopTerm Changes
* **Alarm description**: Stopped or Terminated EC2 Instance State Changes

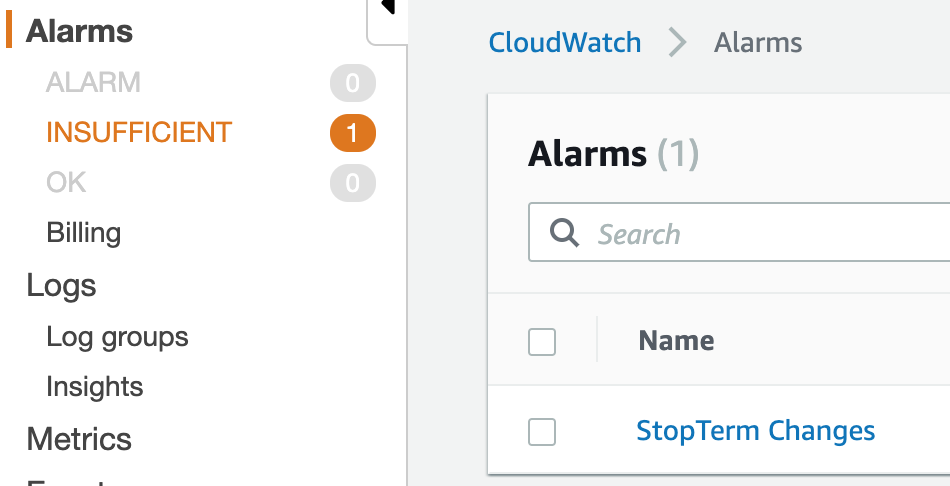
Click **Next**.

14. Click **Create Alarm** when ready.

15. Check for an email from **AWS Notifications**. Open up the email and click the **Confirm subscription** link:



16. Click the **View Alarm** button in the email confirmation dialog.  Note: It is common to see **INSUFFICIENT\_DATA** alarms when first configured:



This typically means one of three things:

* The alarm was just started
* The metric is unavailable
* There is not enough data yet to calculate if an alarm should be triggered

Normally the **INSUFFICIENT\_DATA** state goes away pretty quickly.

Now that everything is configured, it's time to test the alarm by emulating the alarm condition you just configured.

17. Return to the [EC2 service](https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Home:) and launch another AWS t2.micro Linux Instance. Click **Next** and skim each page of the Wizard. Finally, click **Review and Launch**. Select **Proceed without a key pair**, then check the acknowledge check box. (You will not need to connect to the instance later.) Make sure the instance changes to a **Running** state.

Tip: When initially configuring and testing, sometimes it is helpful to extend the parameters or the way you emulate an alarm. For example, in our case, you might spin up several instances so you can terminate and/or stop more than one instance to test the alarm trigger (please note creating 5 or more computing instances at a time is not permitted). Or similarly, you might create (then delete) more than one S3 bucket, assuming that is the service and metric you are concerned with.

18. Finally, stop the instance again. Wait for the instance state to change accordingly. Now that CloudTrail and CloudWatch are both configured, and there is an alarm looking for Instances that get stopped or terminated, it should be caught.

19. Return to CloudTrail and verify the run and stop instance events were captured. Recall this can take a few minutes. It is not necessary to check this all the time, but is strongly recommended when setting things up for the first time. By checking this you are simply confirming CloudTrail is still up, working and capturing events. Without CloudTrail capturing such events, the CloudWatch metric and associated alarm will not catch the stopped (or terminated) instances. The following is an example of both a stop and terminate instance being captured:



Now is the moment of truth... time to check if the CloudWatch Alarm was automatically triggered by the instance state change metric configured earlier.

20. Return to [CloudWatch Alarms](https://us-west-2.console.aws.amazon.com/cloudwatch/home?region=us-west-2#alarmsV2:). You should see an ALARM. If so, select it to see additional information.

21. Check your email to make sure the Alarm generated an email notification from **AWS Notifications** (which uses the AWS SNS service).

### Summary

Congratulations! That was a fair amount of configuration across multiple AWS services, traffic generation and verification that communications work correctly! As you become more familiar with CloudTrail and CloudWatch the amount of time spent configuring, testing and verifying the services will become more streamlined.

After confirming the logs were being delivered successfully, and EC2 events captured instance states changing, you configured a metric filter to look for Instances transitioning to either a stopped or terminated state. You verified the communication and logic was working by bouncing Instances and making sure the transitions were captured and graphed by CloudWatch. Finally, you configured an alarm so that an notification was proactively sent to your email when the specific instance state transitions were detected.