AWS Ephemeral Workspaces Documentation
Christopher Alonso

Table of Contents

Overview

This system is designed as the best solution for creating workspaces that do not hold any data on disk after the user log’s off. It is designed as a self serving system in which a workspace is created. And automatically triggers a function that will assign an alarm to that workspace. That alarm (once a user is disconnected) activates a function that will tear down the workspace and rebuild it with a blank slate via a bundle is assigned to that workspace. So once a user is disconnected if they do not reconnect within 10 minutes. The workspace is tore down. The tearing down of the workspace triggers a function that removes the alarm that was assigned to it. Then waits 4 minutes until the workspace is completely deleted to create a new one with the same bundle and user assigned. Which again then triggers the alarm creation which continues the cycle. A diagram is shown below.

Diagram

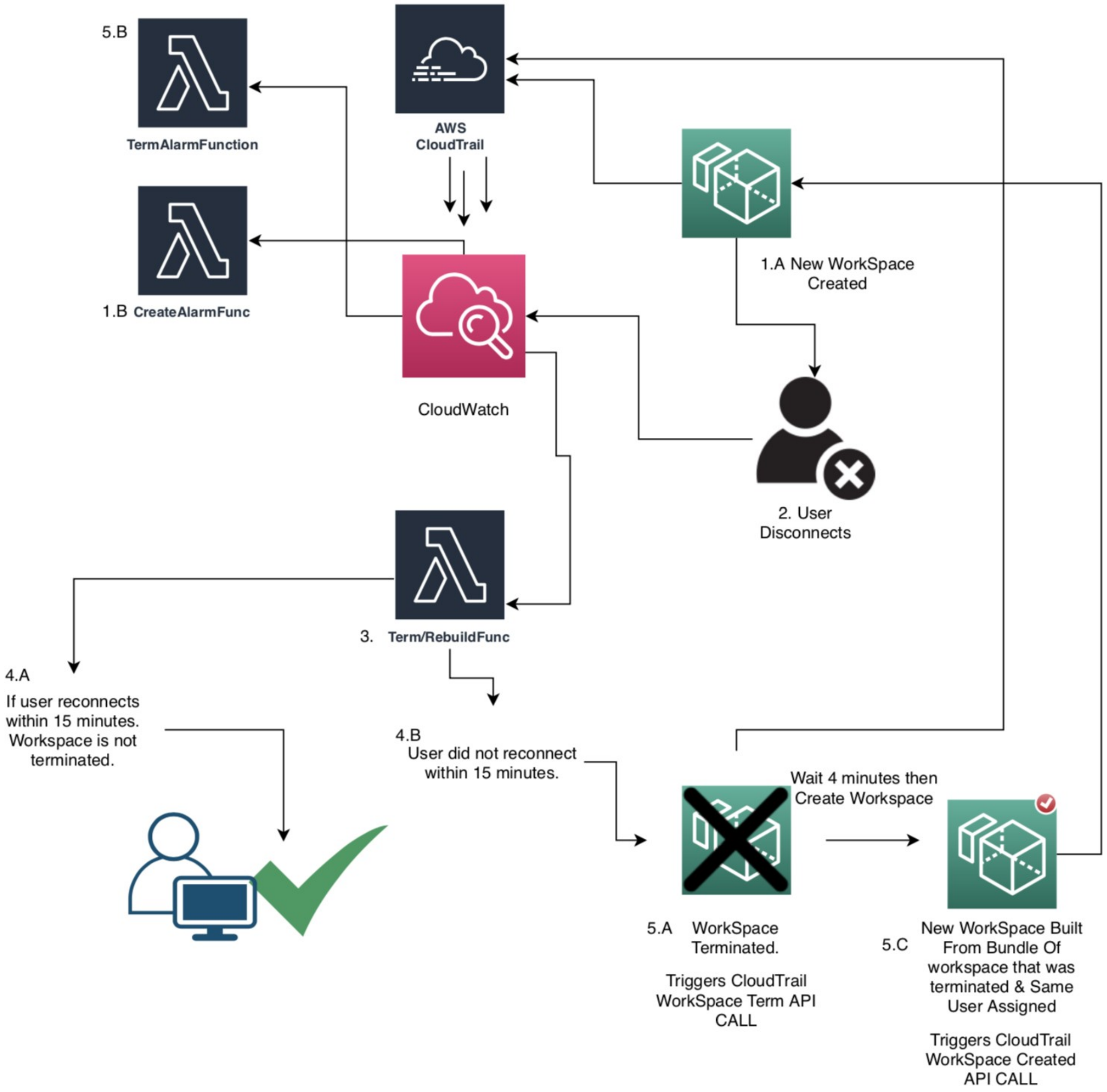
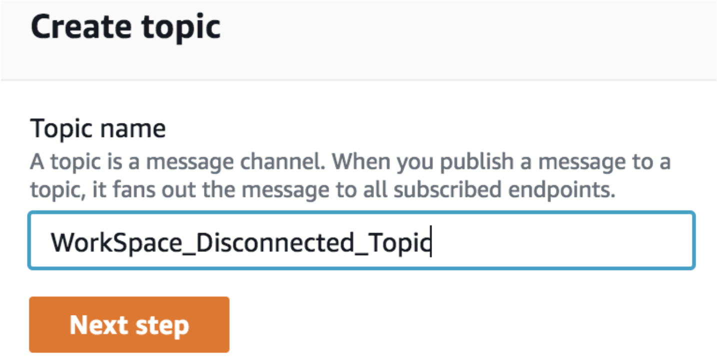

5.B
TermAlarmFunction
AWS
CloudTrail
1.B CreateAlarmFunc
1.A New WorkSpace
Created
CloudWatch
2. User
Disconnects
3. Term/RebuildFunc
4.A
If user reconnects
within 15 minutes.
Workspace is not
terminated.
4.B
User did not reconnect
within 15 minutes.
Wait 4 minutes then
Create Workspace
5.A WorkSpace
Terminated.
Triggers CloudTrail
WorkSpace Term API
CALL
5.C
New WorkSpace Built
From Bundle Of
workspace that was
terminated & Same
User Assigned
Triggers CloudTrail
WorkSpace Created
API CALL

Diagram Breakdown

The diagram is mean to be viewed in order via numbers supplied to icons. Note that multiple outcomes can occur and those are numbered with letters beside them. Keep in mind when a workspace is created or terminated cloudTrail logs those api calls and in turn feeds that to cloudWatch which triggers the rules that are in place. Causing the two functions Create Alarm & Term Alarm to trigger.

Setup

Services → SNS → Create a Topic



Create topic

Topic name

A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

WorkSpace_Disconnected_Topic

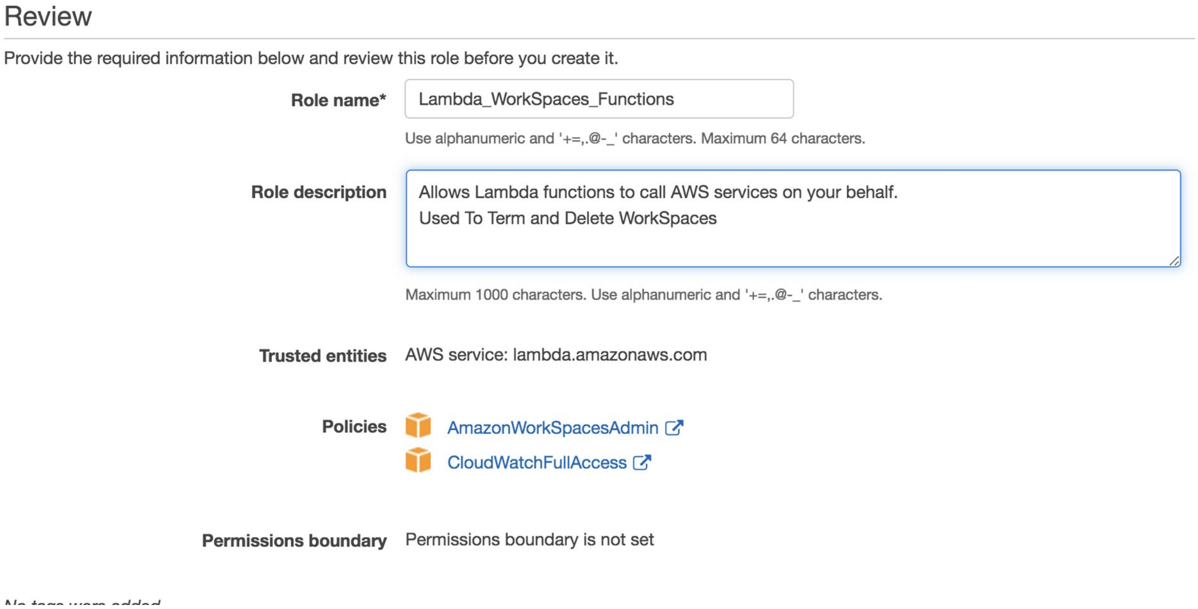
Skip All the optional parameters and create the topic.

Services → IAM → Create Role

Create a role to be used by Lambda and give it the following permissions.

“AmazonWorkSpacesAdmin”
“CloudWatchFullAccess”

This is the end confirmation page.



Review

Provide the required information below and review this role before you create it.

Role name* Lambda_WorkSpaces_Functions

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

Role description Allows Lambda functions to call AWS services on your behalf.
Used To Term and Delete WorkSpaces

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies AmazonWorkSpacesAdmin
CloudWatchFullAccess

Permissions boundary Permissions boundary is not set

No tags were added

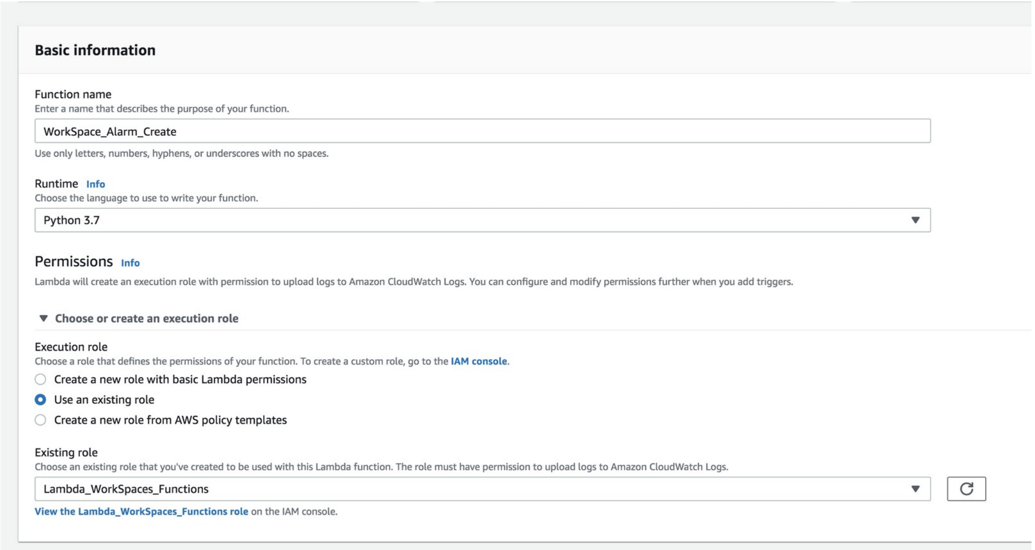
After this is done its time to create the functions and assign them the role we just created.

Services → Lambda → Create Function

Create the following 3 functions.

WorkSpace_Alarm_Create
WorkSpace_Alarm_Del
WorkSpace_Term_Rebuild

Assign the runtime to python 3.7 and add the role we just created, this is the end result.



Basic information

Function name
Enter a name that describes the purpose of your function.
WorkSpace_Alarm_Create
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Python 3.7

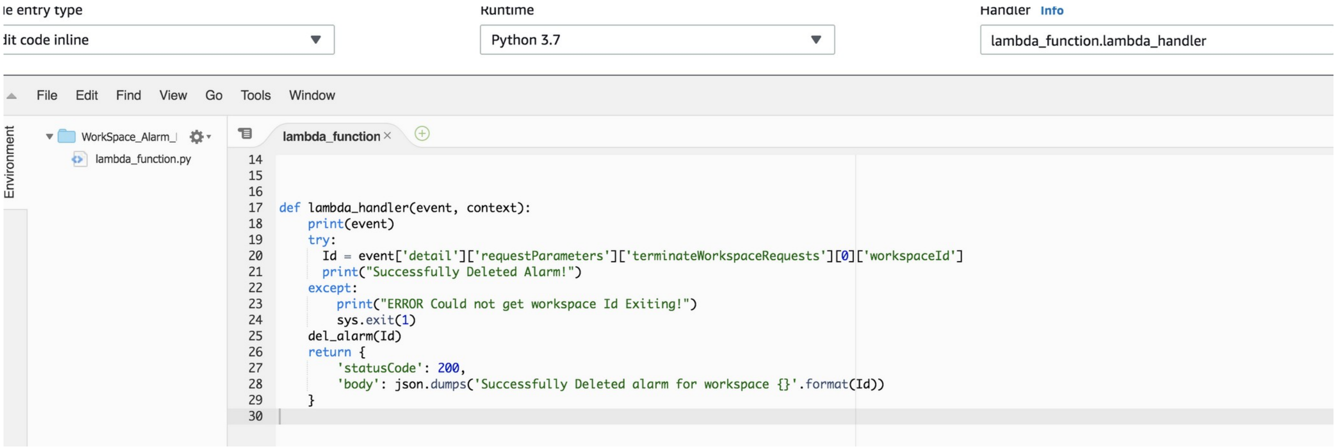
Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
○ Create a new role with basic Lambda permissions
● Use an existing role
○ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
Lambda_WorkSpaces_Functions
View the Lambda_WorkSpaces_Functions role on the IAM console.

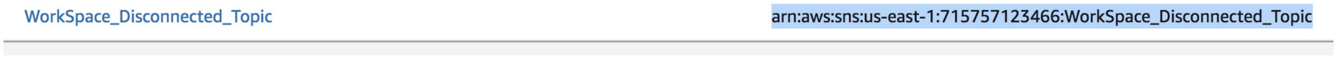
Navigate to each function and paste the source code for each one and save the function. Keep in mind to delete the placeholder code.



ie entry type: dit code inline
Runtime: Python 3.7
Handler Info: lambda_function.lambda_handler

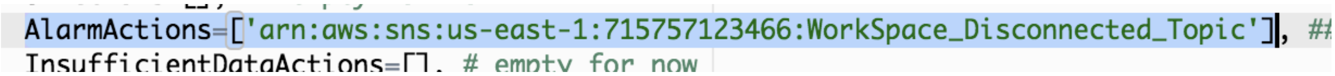
```python



def lambda_handler(event, context):
    print(event)
    try:
      Id = event['detail']['requestParameters']['terminateWorkspaceRequests'][0]['workspaceId']
      print("Successfully Deleted Alarm!")
    except:
        print("ERROR Could not get workspace Id Exiting!")
        sys.exit(1)
    del_alarm(Id)
    return {
        'statusCode': 200,
        'body': json.dumps('Successfully Deleted alarm for workspace {}'.format(Id))
    }

```

Navigate back to the SNS topic that was created and copy the ARN



WorkSpace_Disconnected_Topic arn:aws:sns:us-east-1:715757123466:WorkSpace_Disconnected_Topic

You will need to replace a line of code in the Create Alarm function



```
AlarmActions=['arn:aws:sns:us-east-1:715757123466:WorkSpace_Disconnected_Topic'], ##
InsufficientDataActions=[], # empty for now
```

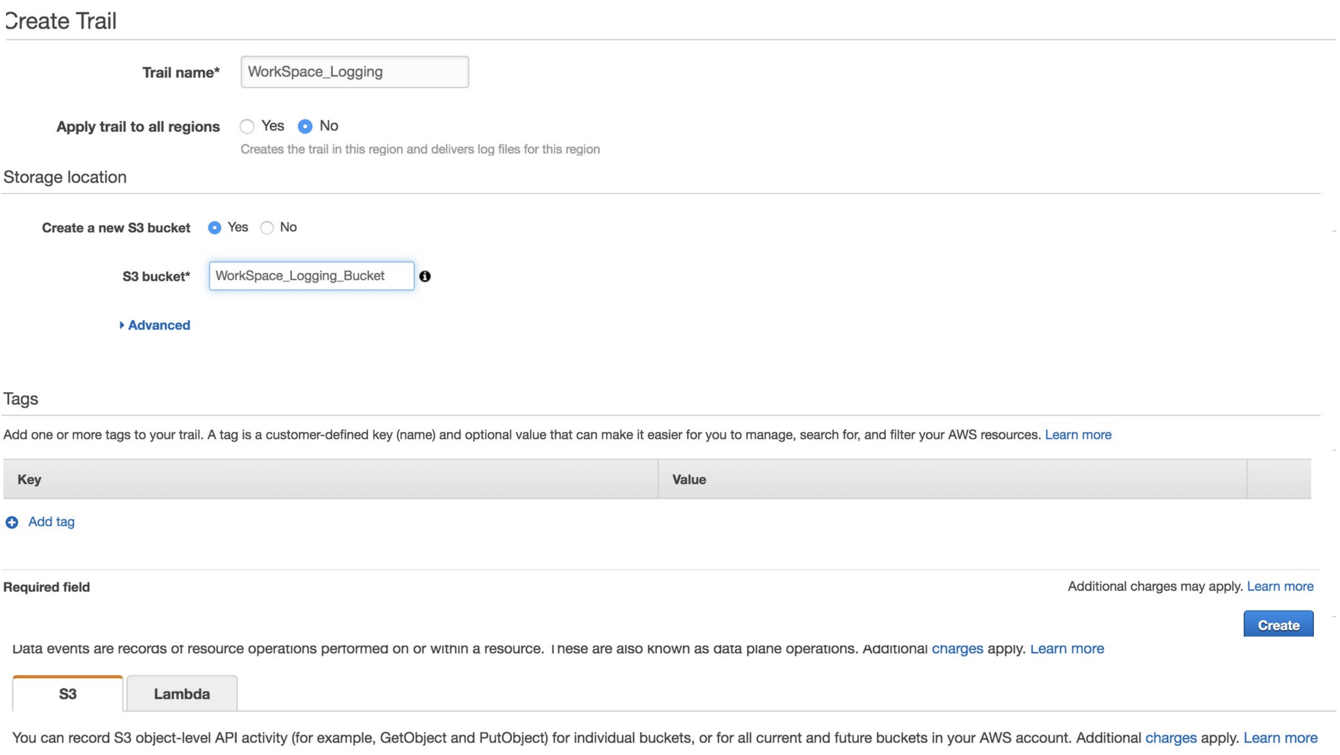
Replace the string with the ARN you just copied and save the changes.

Navigate to the CloudWatch Service.

Services → CloudWatch → Rules → Create Rule

You will need to create two rules, each one triggers a function. One when the Terminate Workspace API is called and one when the Create Workspace API is called. Amazon does everything via API calls each service has a set of api's that are basically commands. Google AWS "Service name here" API Refrence for all the API call's

After clicking create rule, set the service name as workspaces and the event type as AWS API Call via CloudTrail. If Cloud trail is not setup it will ask you to set it up. Follow the link and fill it out as I have below.



Create Trail

Trail name* WorkSpace_Logging

Apply trail to all regions ○ Yes ● No

Creates the trail in this region and delivers log files for this region

Storage location

Create a new S3 bucket ● Yes ○ No

S3 bucket* WorkSpace_Logging_Bucket

Tags

Add one or more tags to your trail. A tag is a customer-defined key (name) and optional value that can make it easier for you to manage, search for, and filter your AWS resources. Learn more

Key	Value

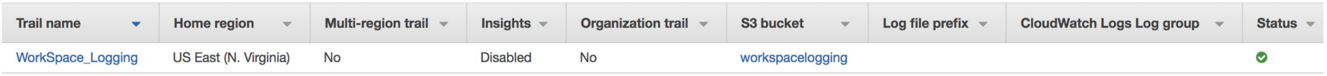
Required field

Additional charges may apply. Learn more

Data events are records of resource operations performed on or within a resource. These are also known as data plane operations. Additional charges apply. Learn more

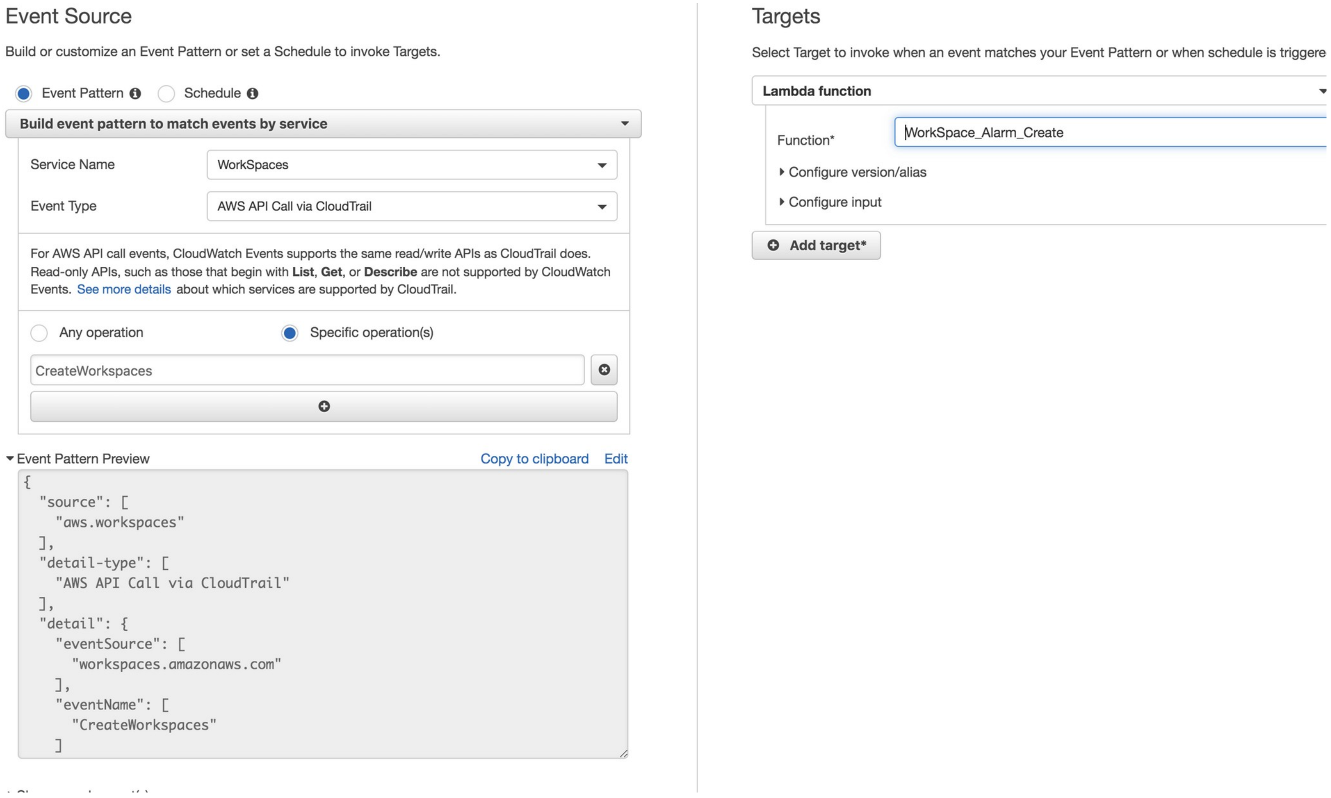
You can record S3 object-level API activity (for example, GetObject and PutObject) for individual buckets, or for all current and future buckets in your AWS account. Additional charges apply. Learn more

You will need to use a bucket for the logging. Or you can create one.



Trail name	Home region	Multi-region trail	Insights	Organization trail	S3 bucket	Log file prefix	CloudWatch Logs Log group	Status
WorkSpace_Logging	US East (N. Virginia)	No	Disabled	No	workspacelogging			✓

Go back to cloudwatch and create the rule for the creating workspaces API. and specify the specific operation as CreateWorkspaces, it should autofill this in the json below and create a target, add the create alarm function. The same will need to be done with the Term Alarm Function. Except the term alarm function has a different API call tge event that will need to be in the json is the following “TerminateWorkspaces”. Here is the final result for the create workspace rule.



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: WorkSpaces

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)

CreateWorkspaces

Event Pattern Preview — Copy to clipboard — Edit

```
{
  "source": [
    "aws.workspaces"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "workspaces.amazonaws.com"
    ],
    "eventName": [
      "CreateWorkspaces"
    ]
```

Targets

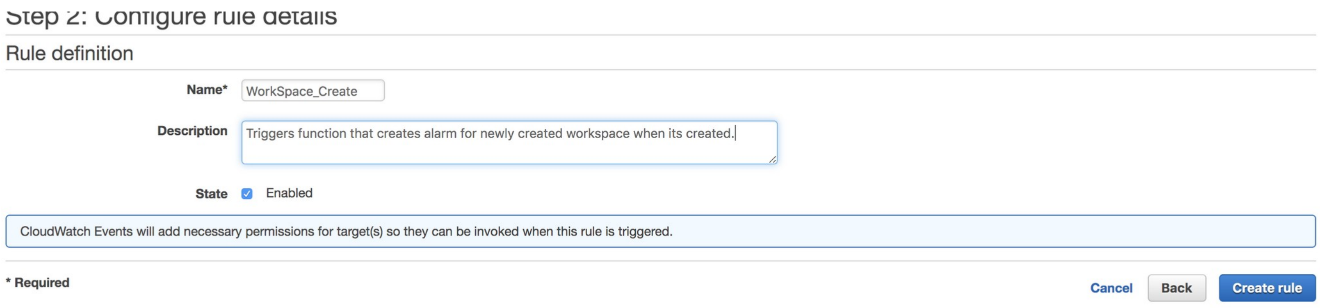
Select Target to invoke when an event matches your Event Pattern or when schedule is triggere

Lambda function

Function*: WorkSpace_Alarm_Create

▸ Configure version/alias

▸ Configure input



Step 2: Configure rule details

Rule definition

Name*: WorkSpace_Create

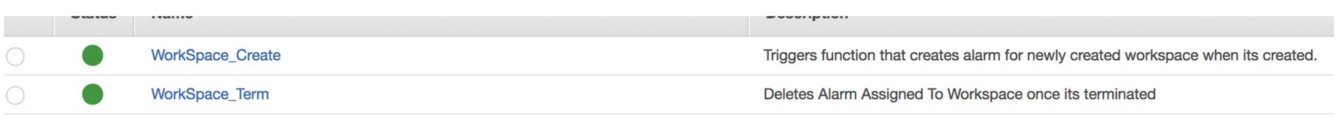
Description: Triggers function that creates alarm for newly created workspace when its created.

State: ☑ Enabled

CloudWatch Events will add necessary permissions for target(s) so they can be invoked when this rule is triggered.

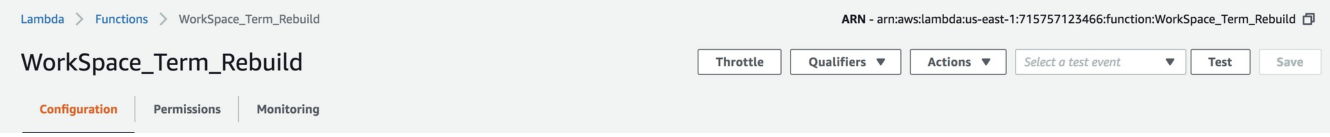
* Required

The end result is two rules.



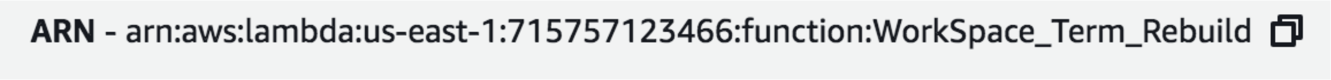
○	●	WorkSpace_Create	Triggers function that creates alarm for newly created workspace when its created.
○	●	WorkSpace_Term	Deletes Alarm Assigned To Workspace once its terminated

Once that is complete go back to lambda and copy the arn for the rebuild and terminate function.



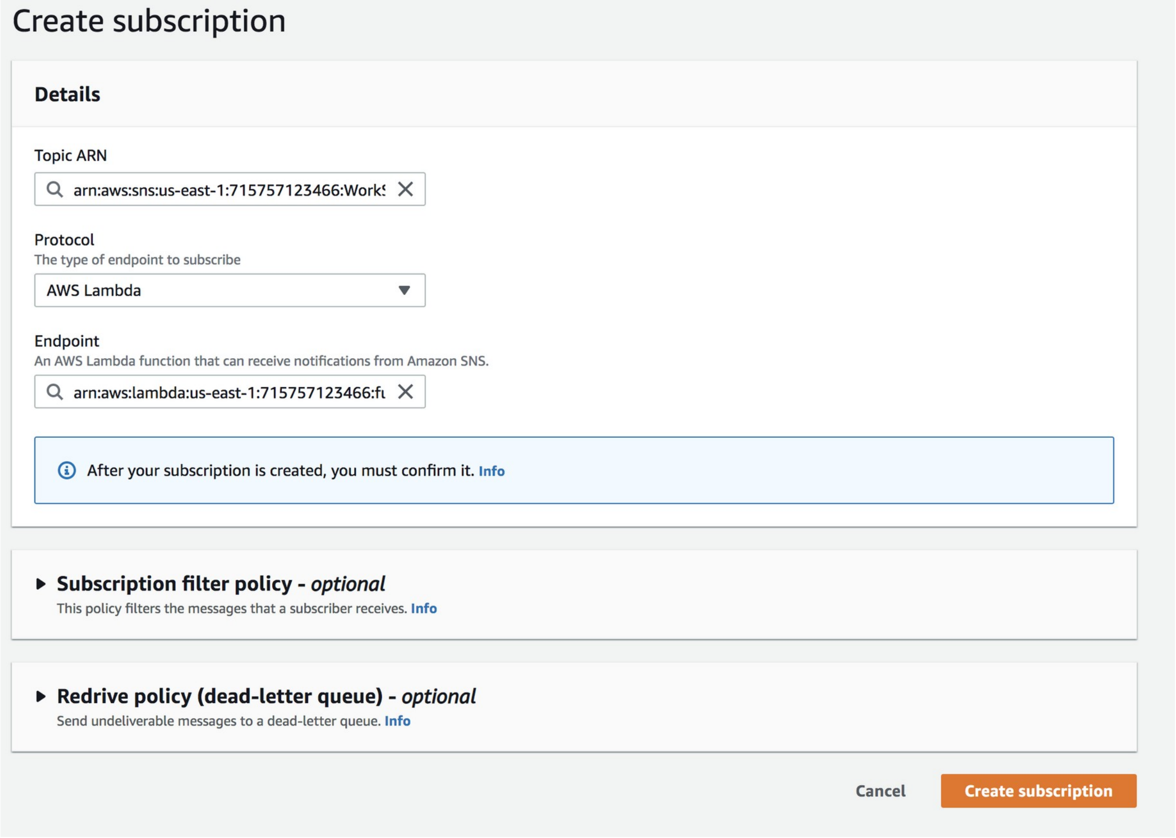
WorkSpace_Term_Rebuild

ARN - arn:aws:lambda:us-east-1:715757123466:function:WorkSpace_Term_Rebuild



ARN - arn:aws:lambda:us-east-1:715757123466:function:WorkSpace_Term_Rebuild

Go to the SNS Service and click subscriptions and create a subscription. Set the topic to be the only topic we created and set the protocol to Lambda. And paste the ARN of your function.



Create subscription

Details

Topic ARN

arn:aws:sns:us-east-1:715757123466:Work!

Protocol

The type of endpoint to subscribe

AWS Lambda

Endpoint

An AWS Lambda function that can receive notifications from Amazon SNS.

arn:aws:lambda:us-east-1:715757123466:fu

After your subscription is created, you must confirm it. Info

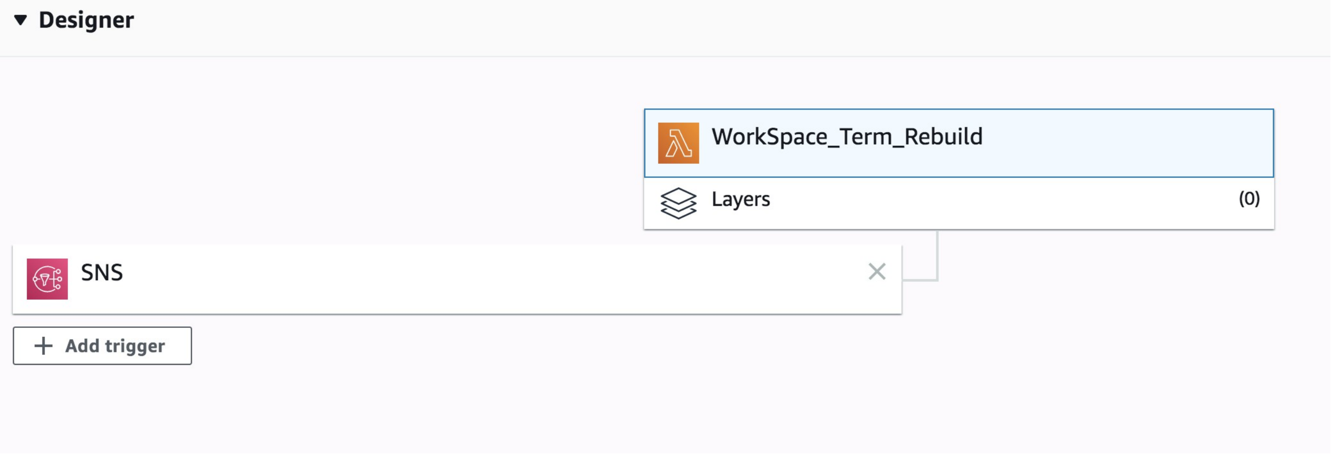
▸ Subscription filter policy - *optional*

This policy filters the messages that a subscriber receives. Info

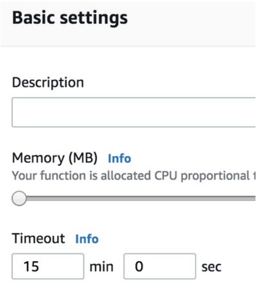
▸ Redrive policy (dead-letter queue) - *optional*

Send undeliverable messages to a dead-letter queue. Info

If you go back to Lambda and look at all your functions. You should have them all being triggered by something in the diagram. Example is below.



The last step is to navigate to the rebuild and term workspace function and make two changes to the settings of that function. Change the retry attempts to 0 and change the timeout time to 15 minutes.

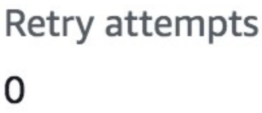


Basic settings

Description

Memory (MB) Info
Your function is allocated CPU proportional

Timeout Info
15 min 0 sec



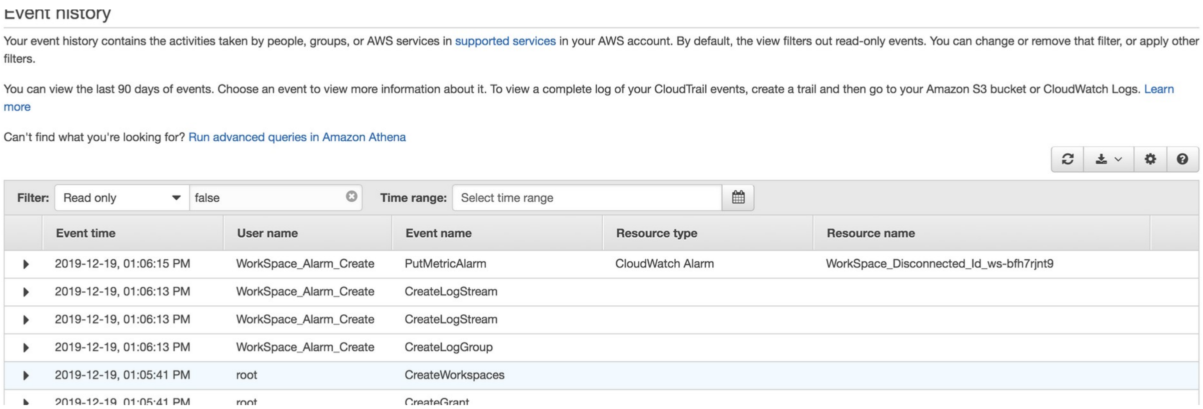
Retry attempts
0

At this point all that is left is to create a workspace and confirm the setup is working properly. All that would need to be done is to launch a workspace from a bundle and check cloudwatch alarms, to see if an alarm is created. Then Login and disconnect/logoff and the workspace should terminate along with the alarm, then create a new instance of itself along with a new alarm.

More Details

CloudTrail

CloudTrail is pretty much a logging system. In which the logs are stored in a bucket. But what its looking out for is API Calls which are how AWS run’s commands in every service. For example if you create a new workspace in the workspaces service by clicking launch workspace. You are actually performing an API call on the backend. Cloud Trails is very important to this system as it allows notifications to go to the cloud watch rules we created which in turn trigger a function. So if a create workspace api is called then through cloud trail logging our cloud watch rule gets triggered which in turn triggers a function to create an alarm or delete one. This is how the system if able to run on its own once setup. Troubleshooting cloud watch is as simple as confirming its actually logging api calls. Below you can see the effect of creating a workspace, it triggers the api which in turn triggers our function which creates an alarm for that workspace that was created in cloud watch.



Event history

Your event history contains the activities taken by people, groups, or AWS services in supported services in your AWS account. By default, the view filters out read-only events. You can change or remove that filter, or apply other filters.

You can view the last 90 days of events. Choose an event to view more information about it. To view a complete log of your CloudTrail events, create a trail and then go to your Amazon S3 bucket or CloudWatch Logs. Learn more

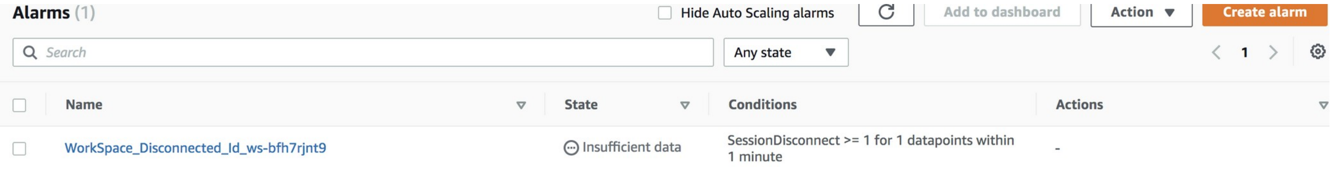
Can't find what you're looking for? Run advanced queries in Amazon Athena

Filter: Read only false Time range: Select time range

	Event time	User name	Event name	Resource type	Resource name
▸	2019-12-19, 01:06:15 PM	WorkSpace_Alarm_Create	PutMetricAlarm	CloudWatch Alarm	WorkSpace_Disconnected_Id_ws-bfh7rjnt9
▸	2019-12-19, 01:06:13 PM	WorkSpace_Alarm_Create	CreateLogStream		
▸	2019-12-19, 01:06:13 PM	WorkSpace_Alarm_Create	CreateLogStream		
▸	2019-12-19, 01:06:13 PM	WorkSpace_Alarm_Create	CreateLogGroup		
▸	2019-12-19, 01:05:41 PM	root	CreateWorkspaces		
▸	2019-12-19, 01:05:41 PM	root	CreateGrant		

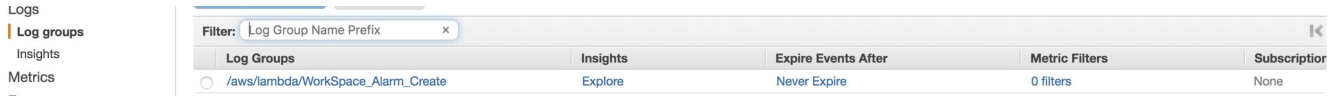
CloudWatch

CloudWatch performs logging for specific metrics, not API calls like Cloud trail. The create alarm and delete alarm functions create/delete alerts in this service. As the alerts they created are based on amazon workspace metrics (Session Disconnect by workspace ID Metric)

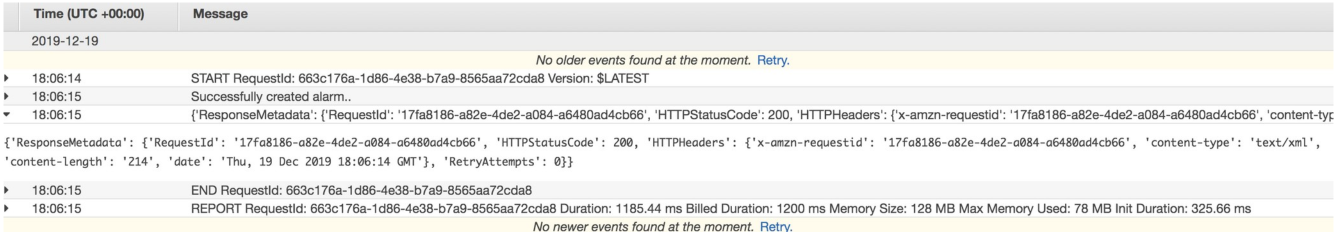


□	Name	State	Conditions	Actions
□	WorkSpace_Disconnected_Id_ws-bfh7rjnt9	Insufficient data	SessionDisconnect >= 1 for 1 datapoints within 1 minute	-

Another import thing to note about Cloud Watch is it performs logging for all lambda functions that have been run atleast once. So you can troubleshoot issues with the lambda functions themselves by looking at the log group for each function.



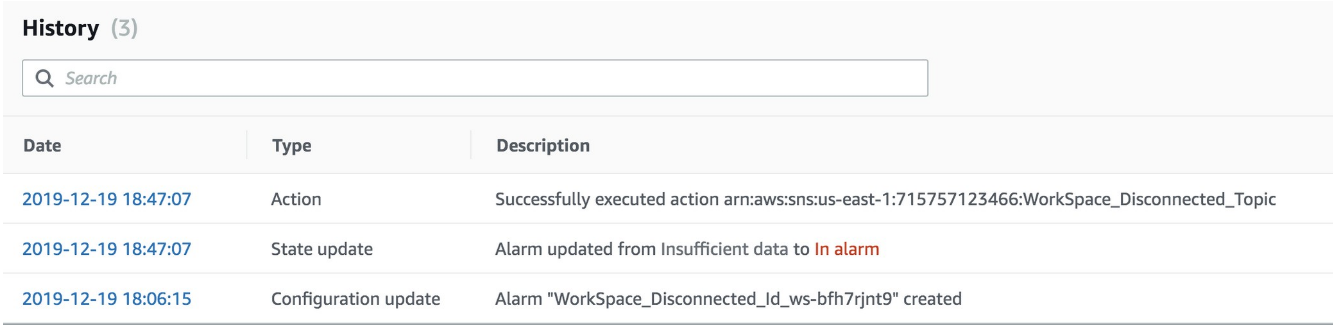
Log Groups	Insights	Expire Events After	Metric Filters	Subscription
/aws/lambda/WorkSpace_Alarm_Create	Explore	Never Expire	0 filters	None



```
Time (UTC +00:00)   Message
2019-12-19
                    No older events found at the moment. Retry.
18:06:14            START RequestId: 663c176a-1d86-4e38-b7a9-8565aa72cda8 Version: $LATEST
18:06:15            Successfully created alarm..
18:06:15            {'ResponseMetadata': {'RequestId': '17fa8186-a82e-4de2-a084-a6480ad4cb66', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '17fa8186-a82e-4de2-a084-a6480ad4cb66', 'content-typ
{'ResponseMetadata': {'RequestId': '17fa8186-a82e-4de2-a084-a6480ad4cb66', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '17fa8186-a82e-4de2-a084-a6480ad4cb66', 'content-type': 'text/xml',
'content-length': '214', 'date': 'Thu, 19 Dec 2019 18:06:14 GMT'}, 'RetryAttempts': 0}}
18:06:15            END RequestId: 663c176a-1d86-4e38-b7a9-8565aa72cda8
18:06:15            REPORT RequestId: 663c176a-1d86-4e38-b7a9-8565aa72cda8 Duration: 1185.44 ms Billed Duration: 1200 ms Memory Size: 128 MB Max Memory Used: 78 MB Init Duration: 325.66 ms
                    No newer events found at the moment. Retry.
```

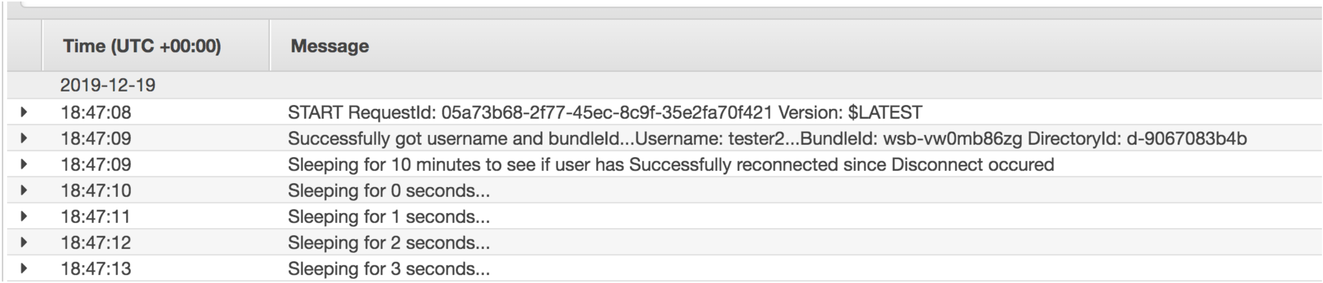
Cloud Watch is the most important aspect of the term and rebuild function that actually terminates then recreates the workspaces. As it checks for reconnection via the Session Reconnect cloudwatch metric and the alarms set in this service trigger the function to be run in the first place.

Example of disconnect triggering alarm.



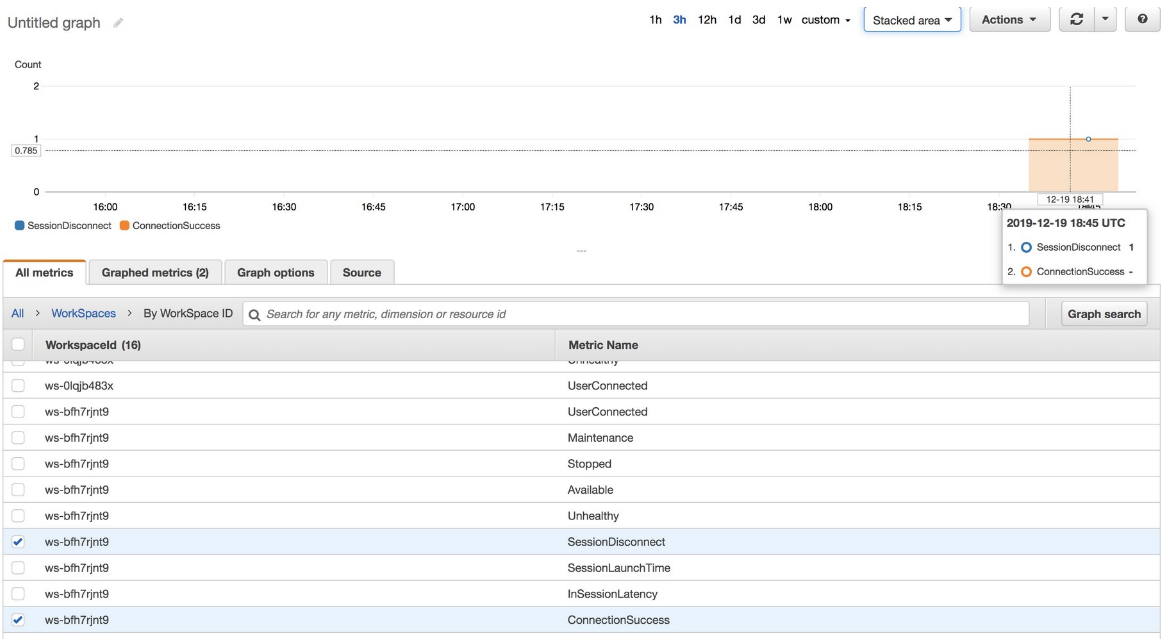
Date	Type	Description
2019-12-19 18:47:07	Action	Successfully executed action arn:aws:sns:us-east-1:715757123466:WorkSpace_Disconnected_Topic
2019-12-19 18:47:07	State update	Alarm updated from Insufficient data to In alarm
2019-12-19 18:06:15	Configuration update	Alarm "WorkSpace_Disconnected_Id_ws-bfh7rjnt9" created

Which then triggers the main term and rebuild function. These are logs from cloudwatch of that function.



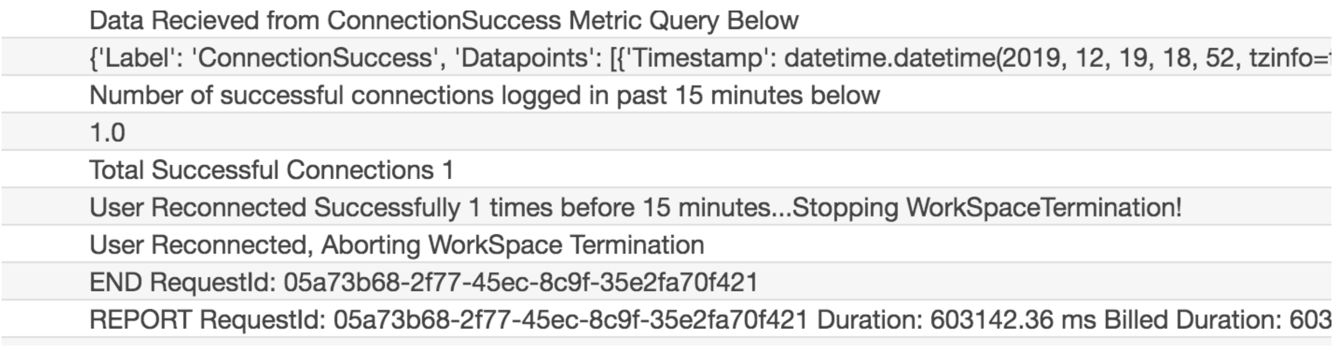
```
Time (UTC +00:00)   Message
2019-12-19
18:47:08            START RequestId: 05a73b68-2f77-45ec-8c9f-35e2fa70f421 Version: $LATEST
18:47:09            Successfully got username and bundleId...Username: tester2...BundleId: wsb-vw0mb86zg DirectoryId: d-9067083b4b
18:47:09            Sleeping for 10 minutes to see if user has Successfully reconnected since Disconnect occured
18:47:10            Sleeping for 0 seconds...
18:47:11            Sleeping for 1 seconds...
18:47:12            Sleeping for 2 seconds...
18:47:13            Sleeping for 3 seconds...
```

Now if users reconnects or doesn’t reconnect you will see it in this logfile. Below are data points of that workspace connecting , disconnecting then reconnecting. This illustrates what Cloud Watch is used for.



□	WorkspaceId (16)	Metric Name
□	ws-0lqjb483x	UserConnected
□	ws-bfh7rjnt9	UserConnected
□	ws-bfh7rjnt9	Maintenance
□	ws-bfh7rjnt9	Stopped
□	ws-bfh7rjnt9	Available
□	ws-bfh7rjnt9	Unhealthy
☑	ws-bfh7rjnt9	SessionDisconnect
□	ws-bfh7rjnt9	SessionLaunchTime
□	ws-bfh7rjnt9	InSessionLatency
☑	ws-bfh7rjnt9	ConnectionSuccess

Since the user reconnected within 15 minutes, the termination was aborted this can be seen via the function logs below.



```
Data Recieved from ConnectionSuccess Metric Query Below
{'Label': 'ConnectionSuccess', 'Datapoints': [{'Timestamp': datetime.datetime(2019, 12, 19, 18, 52, tzinfo=
Number of successful connections logged in past 15 minutes below
1.0
Total Successful Connections 1
User Reconnected Successfully 1 times before 15 minutes...Stopping WorkSpaceTermination!
User Reconnected, Aborting WorkSpace Termination
END RequestId: 05a73b68-2f77-45ec-8c9f-35e2fa70f421
REPORT RequestId: 05a73b68-2f77-45ec-8c9f-35e2fa70f421 Duration: 603142.36 ms Billed Duration: 603
```

SNS

SNS is used as a link between the alarms created by the create alarm function and the lambda function to term and rebuild the workspaces. So when a disconnect metric is seen it triggers a call to the SNS topic we created and our lambda function is subscribed to.

Lambda

The lambda functions are written in python3 and use the boto3 library for interacting with the AWS API. The functions themselves are simple. They all receive a json payload upon being executed and perform api call’s after extracting this information. The biggest function is the function that performs the termination, waits for 10 minutes then checks cloud watch metrics to see if the machine has reconnected if not then it will terminate the workspace. Wait 4 minutes then recreate it from the bundle it was originally created with. To create these functions I referenced the boto3 documentation online which is excellent and very simple to work with.

WorkSpaces

No special configuration for workspace is needed just create images then bundles for each client. When launching the workspace be sure to select the correct bundle. And everything will happen automatically from there. Everything that the functions and alerts do should work independently of where the workspace is located (subnet, vpc ,directory) as long as the region is the same everything should work.