



## TOOLS

# AWS Nightly Shutdown

Jason Meurer  
CE, Public Cloud

## Overview of Tool

This guide will walk the user through creating an IAM Role, Lambda Script, SES approved sender and a CloudWatch event to run across all regions nightly and shutdown all EC2 instances. Instances can be tagged with DoNotStop option to prevent the nightly shutdown.

## Benefits of the Tool

This AWS workflow is intended as a safeguard against exorbitant charges in your AWS account.

## When to Use

It is recommended to use this workflow in any account where systems are started as needed rather than running full time.

# Getting Started

It is assumed that the user as an AWS account with Full Admin privileges as this guide create new configurations across multiple AWS services.

## Before You Begin

Access your [AWS console](#). Open multiple tabs to the AWS console as this will make switching between services easier.

### Procedure 1: **IAM Role Creation**

---

In this procedure, an IAM Role will be created for the Lambda Function to run as. We will follow the concept of least privilege for the Role to have the ability to access EC2 and SES to perform the necessary actions. More information: [Create an IAM policy using the JSON policy editor](#)

- Step 1** Access IAM from the list of AWS Services in the console.
- Step 2** Select Policies from the left-hand menu.
- Step 3** Select Create Policy and Select the JSON tab.
- Step 4** Paste the following JSON block. The raw code can also be accessed here: [IAM Policy](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/lambda/*:*:*",
      "Effect": "Allow"
    }
  ],
}
```

```
{
  "Action": [
    "ec2:DescribeInstances",
    "ec2:DescribeInstanceStatus",
    "ec2:DescribeRegions",
    "ses:SendEmail",
    "ses:SendRawEmail"
  ],
  "Resource": "*",
  "Effect": "Allow"
},
{
  "Action": [
    "ec2:StopInstances",
    "ec2:StartInstances"
  ],
  "Resource": "arn:aws:ec2:*:*:instance/*",
  "Effect": "Allow"
}
]
```

**Step 5** Select the Review Policy Button

**Step 6** Provide a name for your policy on the next screen. Finishing the Policy creation by Hitting the Create Policy Button.

**Step 7** Select Roles from the Left-Hand Menu and then hit the Create Role button.

**Step 8** Ensure that AWS Service is Selected at the top and choose Lambda from the Service list. Click Next:Permissions

**Step 9** Select the Policy Create earlier. Click Next: Tags

**Step 10** Add any desire Tags. Click Next:Review

**Step 11** Provide a Name for your Role and click Create Role.

## Procedure 2: **Simple Email Service Configuration**

This Procedure will allow SES to send emails on behalf of your desired account. The tool will send the configured recipient address an email for each system being shutdown.

**Step 1** Select Simple Email Service from the list of AWS Services. If you are not currently in a support region, you will presented a list to choose from.

✦ It is for this reason I recommend configuring your Lambda Function in a specific region to match where SES is configured. This guide will assume Oregon (us-west-2) for the remaining configuration items.

**Step 2** In the left-hand menu, select Email Addresses.

**Step 3** Select the Verify a New Email Address Button.

**Step 4** Enter the email address that you intend to be the sender of your notification emails.

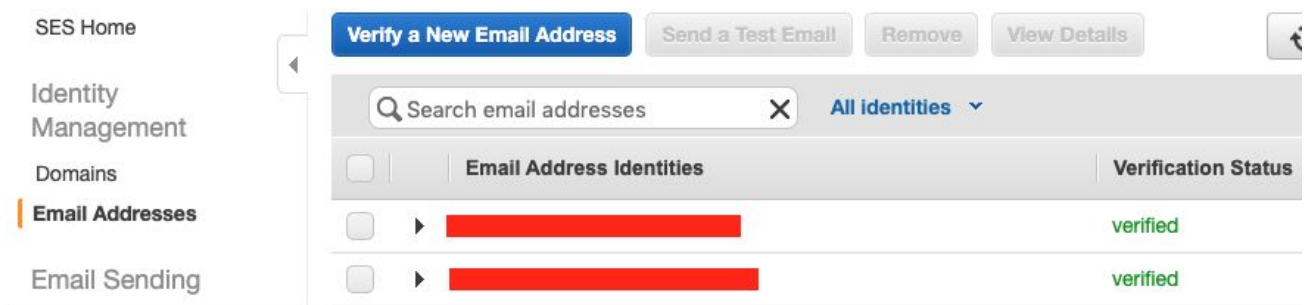
✦ This must be an address that you control as the following steps will require you to verify the email.

**Step 5** Check your Inbox for a Verification Email from AWS.

✦ This email will typically end up in your SPAM folder.

**Step 6** Open the URL within the message to verify the email.

**Step 7** Refresh the SES console to ensure the email address is “verified”.



### Procedure 3: **Lambda Creation**

---

This procedure will create and customize the Lambda function that will shutdown the instances and send the emails.

- Step 1** In the AWS console, select the Lambda Service ensuring you are in your desired region.
- Step 2** Select the Create Function button.
- Step 3** Select “Author from scratch”. Provide a Name for your function and in the Runtime pick the latest version of Python 3.x. At the time of this writing, Python 3.7.
- Step 4** Expand the Choose Execution Role. Select “use an existing role” and select the previously created role.

The screenshot displays the AWS Lambda 'Create function' interface. At the top, the breadcrumb navigation shows 'Lambda > Functions > Create function'. The main heading is 'Create function' with an 'Info' link. Below this, a prompt says 'Choose one of the following options to create your function.' There are two options: 'Author from scratch' (selected with a radio button) and 'Use a blueprint' (unselected). The 'Author from scratch' option includes a description 'Start with a simple Hello World example.' and a gear icon. The 'Use a blueprint' option includes a description 'Build a Lambda application from sample code and configuration presets for common use cases.' and a checklist icon. Below these options is the 'Basic information' section. It contains a 'Function name' field with the value 'JIStopall' and a note 'Enter a name that describes the purpose of your function.' and 'Use only letters, numbers, hyphens, or underscores with no spaces.' The 'Runtime' section shows 'Python 3.7' selected with a note 'Choose the language to use to write your function.' The 'Permissions' section has a note '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.' It includes a dropdown 'Choose or create an execution role' which is expanded to show three options: 'Create a new role with basic Lambda permissions' (unselected), 'Use an existing role' (selected with a radio button), and 'Create a new role from AWS policy templates' (unselected). Below this, the 'Existing role' section shows 'JIRole' selected with a note '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.' and a link 'View the JIRole role on the IAM console.'

- Step 5** Select the Create function button.
- Step 6** Scroll to the Function Code and download this [Python File](#).
- Step 7** Leave the handler set to “lambda\_function.lambda\_handler”
- Step 8** Paste the Function Code and update the following Variables within the code.


- (1) SENDER = This will be the sender address previously verified in SES.
- (2) RECIPIENT = Who will receive the shutdown notifications.
- (3) AWS\_REGION = This the region in which SES was reconfigured.
- (4) (OPTIONAL) for more advanced configuration, if an SES Configuration Set is used, uncomment #CONFIGURATION\_SET and specify the set accordingly.

**Step 9** Scroll down to ensure that the Execution role is properly reflecting the previously created role.

**Step 10** Under the Basic settings section, update the Timeout to 5 minutes.

**Step 11** Save the Function.

**Step 12** At this point in time, you can Test the function without the scheduler.

 THIS WILL SHUTDOWN ALL EC2 INSTANCES IN ALL REGIONS THAT DO NOT HAVE A TAG NAME donotstop. The value of the tag is not used and the tag name is not case sensitive.

**Step 13** Select the Test button at the top of the Function Page.

**Step 14** Key values are irrelevant. Lave the Event Template as Hello World.

### Configure test event

A function can have up to 10 test events. The event is used to test your function with the same events.

☒ Create new test event

☐ Edit saved test events

Event template

Hello World

Event name

JITtest

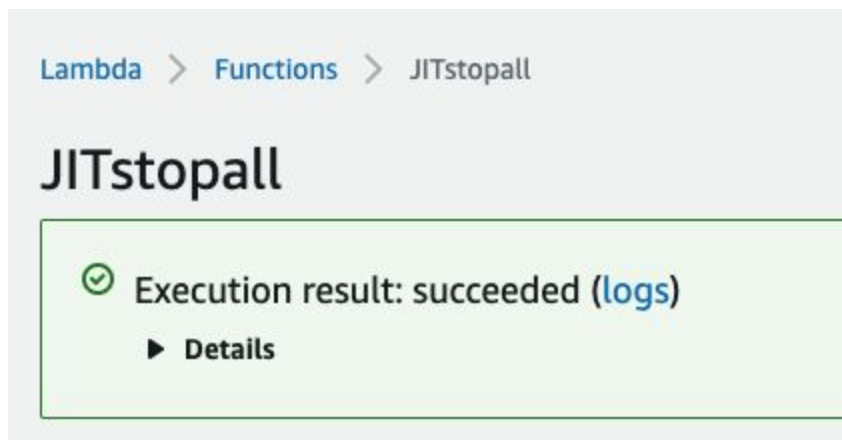
```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 }
```

**Step 15** Select the Create Button.

**Step 16** Select the Test Button again with the newly created test in the drop-down list.

**Step 17** Ensure your EC2 instances are being shutdown from the EC2 console and check both your Inbox and SPAM folder for the corresponding emails.

**Step 18** Additionally, Scroll to the top of the function to access the CloudWatch logs of the run.



**Step 19** Open the Logs link in a new tab to see the corresponding CloudWatch log.

CloudWatch

Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Logs

Log groups

Insights

Metrics

Events

Rules

Event Buses

ServiceLens

Service Map

Traces

Contributor Insights

CloudWatch > Log Groups > /aws/lambda/JITstopall > 2019/12/27/[\$LATEST]328a9cf969874dd88e9ab14af18dd243

Try CloudWatch Logs Insights

CloudWatch Logs Insights allows you to search and analyze your logs using a new, purpose-built query language. Click [here](#) to exper

Filter events

Time (UTC +00:00)	Message
2019-12-27	
No older events found at the moment. <a href="#">Retry.</a>	
18:14:53	START RequestId: fe4d23bc-1ebd-43db-a030-a492103c09bc Version: \$LATEST
18:15:09	stopping instances value ({'StoppingInstances': [{'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'InstanceId': '...
18:15:10	Email sent! Message ID: 0101016f4891c594-3dfc0886-0428-4f95-bf9e-3d7c007dc137-000000
18:15:11	stopping instances value ({'StoppingInstances': [{'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'InstanceId': '...
18:15:11	Email sent! Message ID: 0101016f4891c9f4-508a4f9c-5956-4e35-b006-de6a1ddca24b-000000
18:15:12	END RequestId: fe4d23bc-1ebd-43db-a030-a492103c09bc
18:15:12	REPORT RequestId: fe4d23bc-1ebd-43db-a030-a492103c09bc Duration: 19553.34 ms Billed Duration: 19600
No newer events found at the moment. <a href="#">Retry.</a>	

Page 8 © 2018 Palo Alto Networks



#### Procedure 4: **CloudWatch Scheduler**

---

The last component will be the CloudWatch event scheduler to trigger the function on the desired interval.

- Step 1** Access CloudWatch from the AWS console on a different tab than Lambda. Ensure you are still your desired region.
- Step 2** Select Rules below Events in the left-hand menu.
- Step 3** Select the Create rule button.
- Step 4** Select the Schedule Radio button and specify the desired CRON interval. For more information on schedules: [CloudWatch Events schedules](#)

(1) EXAMPLE Daily @ 6:25PM ET: 25 23 ? \* \* \*

 CloudWatch Event schedules are in UTC, adjust accordingly.

## Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

### Event Source

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

☐ Event Pattern ⓘ ☒ Schedule ⓘ

☐ Fixed rate of

Minutes

☒ Cron expression

Next 10 Trigger Date(s)

1. Fri, 27 Dec 2019 23:25:00 GMT

2. Sat, 28 Dec 2019 23:25:00 GMT

3. Sun, 29 Dec 2019 23:25:00 GMT

4. Mon, 30 Dec 2019 23:25:00 GMT

5. Tue, 31 Dec 2019 23:25:00 GMT

6. Wed, 01 Jan 2020 23:25:00 GMT

7. Thu, 02 Jan 2020 23:25:00 GMT

8. Fri, 03 Jan 2020 23:25:00 GMT

9. Sat, 04 Jan 2020 23:25:00 GMT

10. Sun, 05 Jan 2020 23:25:00 GMT

[Learn more](#) about CloudWatch Events schedules.

▶ Show sample event(s)

**Step 5** On the Right side of the page, select the previously created function from the list.

## Targets

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

Lambda function

Function\*

JITstopall

▸ Configure version/alias

▸ Configure input

Add target\*

**Step 6** Select the Configure details button.

**Step 7** Provide a Name for the Event and select the Create Rule button. The rule should now be displayed with a status of Enabled (triggered by schedule).

## Rules

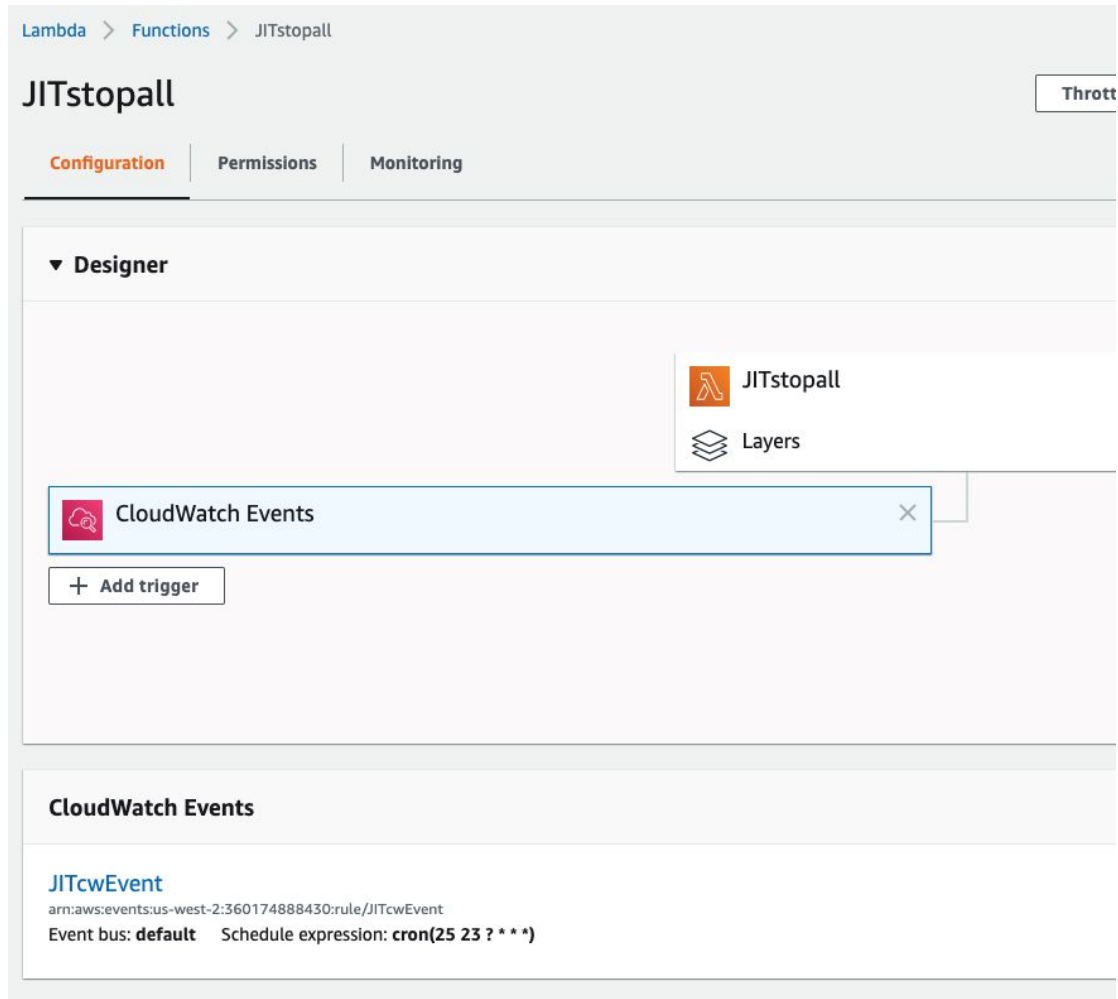
Rules route events from your AWS resources for p

Create rule

Actions ▾

Status	All ▾	Name
	Status	Name
<input type="radio"/>	<input checked="" type="radio"/>	JITcwEvent

**Step 8** Switch Tabs to the Lambda function and refresh the page. The CloudWatch event should now show in the Triggers list.



**Step 9** Your workflow is now Complete. You can adjust your schedule to test your completed product or wait until the desired window to ensure proper functionality.

## Troubleshooting

(Optional) Print statements have been left in the Python code to add diagnostic touchpoints in your CloudWatch logs. You may remove the # comment designator on any print statement to verify the corresponding code in your logs.

## For More Information

<https://github.com/wwce/Scripts/tree/master/AWS-Nightly-Shutdown>