How do I stop and start Amazon EC2 instances at regular intervals using Lambda?

*Last updated: 2021-02-15*

I want to reduce my Amazon Elastic Compute Cloud (Amazon EC2) usage by stopping and starting my EC2 instances automatically. How do I use AWS Lambda and Amazon CloudWatch Events to do that?

Short description

Note: This example setup is a simple solution. For a more robust solution, [use the AWS Instance Scheduler](https://docs.aws.amazon.com/solutions/latest/instance-scheduler/welcome.html). For more information, see [How do I stop and start my instances using the AWS Instance Scheduler?](https://aws.amazon.com/premiumsupport/knowledge-center/stop-start-instance-scheduler/)

To stop and start EC2 instances at regular intervals using Lambda, do the following:

1.    Create a custom AWS Identity and Access Management (IAM) policy and execution role for your Lambda function.

2.    Create Lambda functions that stop and start your EC2 instances.

3.    Test your Lambda functions.

4.    [Create CloudWatch Events rules that trigger your function on a schedule](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Create-CloudWatch-Events-Scheduled-Rule.html).  
Note: You can also [create rules that trigger on an event that takes place in your AWS account](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Create-CloudWatch-Events-Rule.html).

Or, you can use the AWS CloudFormation template provided at the end of this article to automate the procedure.

Resolution

Note: If you receive a Client error on launch error after completing the following procedure, follow the steps outlined in this [Client error on launch troubleshooting article](https://aws.amazon.com/premiumsupport/knowledge-center/encrypted-volumes-stops-immediately/).

[Get the IDs of the EC2 instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connection-prereqs.html#connection-prereqs-get-info-about-instance) that you want to stop and start. Then, do the following.

Create an IAM policy and execution role for your Lambda function

1.    [Create an IAM policy using the JSON policy editor](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html#access_policies_create-json-editor). Copy and paste the following JSON policy document into the policy editor:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:Start\*",

"ec2:Stop\*"

],

"Resource": "\*"

}

]

}

2.    [Create an IAM role](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html#roles-creatingrole-service-console) for Lambda.

Important: When attaching a permissions policy to Lambda, make sure that you choose the IAM policy you just created.

Create Lambda functions that stop and start your EC2 instances

1.    In the [AWS Lambda console](https://console.aws.amazon.com/lambda/), choose Create function.

2.    Choose Author from scratch.

3.    Under Basic information, add the following:  
For Function name, enter a name that identifies it as the function used to stop your EC2 instances. For example, "StopEC2Instances".  
For Runtime, choose Python 3.8.  
Under Permissions, expand Choose or create an execution role.  
Under Execution role, choose Use an existing role.  
Under Existing role, choose the IAM role that you created.

4.    Choose Create function.

5.    Under Function code, copy and paste the following code into the [editor pane in the code editor](https://docs.aws.amazon.com/lambda/latest/dg/code-editor.html#code-editor-code) (lambda\_function). This code stops the EC2 instances that you identify.

Example function code—stopping EC2 instances

import boto3

region = 'us-west-1'

instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

ec2.stop\_instances(InstanceIds=instances)

print('stopped your instances: ' + str(instances))

Important: For region, replace "us-west-1" with the AWS Region that your instances are in. For instances, replace the example EC2 instance IDs with the IDs of the specific instances that you want to stop and start.

6.    Under Basic settings, set Timeout to 10 seconds.

Note: [Configure the Lambda function settings](https://docs.aws.amazon.com/lambda/latest/dg/resource-model.html) as needed for your use case. For example, if you want to stop and start multiple instances, you might need a different value for Timeout and Memory.

7.    Choose Deploy.

8.    Repeat steps 1-7 to create another function. Do the following differently so that this function starts your EC2 instances:  
In step 3, enter a different Function name than the one you used before. For example, "StartEC2Instances".  
In step 5, copy and paste the following code into the editor pane in the code editor (lambda\_function):

Example function code—starting EC2 instances

import boto3

region = 'us-west-1'

instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

ec2.start\_instances(InstanceIds=instances)

print('started your instances: ' + str(instances))

Important: For region and instances , use the same values that you used for the code to stop your EC2 instances.

Test your Lambda functions

1.    In the [AWS Lambda console](https://console.aws.amazon.com/lambda/home), choose Functions.

2.    Choose one of the functions that you created.

3.    Choose Test.

4.    In the Configure test event dialog box, choose Create new test event.

5.    Enter an Event name. Then, choose Create.

Note: You don't need to change the JSON code for the test event—the function doesn't use it.

6.    Choose Test to run the function.

7.    Repeat steps 1-6 for the other function that you created.

Tip: You can [check the status of your EC2 instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-system-instance-status-check.html#viewing_status) before and after testing to confirm that your functions work as expected.

Create CloudWatch Events rules that trigger your Lambda functions

1.    Open the [Amazon CloudWatch console](https://console.aws.amazon.com/cloudwatch/).

2.    In the left navigation pane, under Events, choose Rules.

3.    Choose Create rule.

4.    Under Event Source, choose Schedule.

5.    Do either of the following:  
For Fixed rate of, enter an interval of time in minutes, hours, or days.  
For Cron expression, enter an expression that tells Lambda when to stop your instances. For information on expression syntax, see [Schedule expressions for rules](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html).  
Note: Cron expressions are evaluated in UTC. Make sure that you adjust the expression for your preferred time zone.

6.    Under Targets, choose Add target.

7.    Choose Lambda function.

8.    For Function, choose the function that stops your EC2 instances.

9.    Choose Configure details.

10.    Under Rule definition, do the following:  
For Name, enter a name to identify the rule, such as "StopEC2Instances".  
(Optional) For Description, describe your rule. For example, "Stops EC2 instances every night at 10 PM."  
For State, choose the Enabled check box.

11.    Choose Create rule.

12.    Repeat steps 1-11 to create a rule to start your EC2 instances. Do the following differently:  
In step 5, for Cron expression, enter an expression that tells Lambda when to start your instances.  
In step 8, for Function, choose the function that starts your EC2 instances.  
In step 10, under Rule definition, enter a Name, such as "StartEC2Instances".  
(Optional) Enter a Description, such as "Starts EC2 instances every morning at 7 AM."

(Optional) Use an AWS CloudFormation template to automate the procedure

1.    Save the following AWS CloudFormation template as a YAML file.

2.    In the YAML file, enter the required variables.

3.    [Deploy the AWS CloudFormation template in Visual Studio](https://docs.aws.amazon.com/toolkit-for-visual-studio/latest/user-guide/tkv-cfn-editor-deploy-template.html) or the [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/cli/latest/reference/cloudformation/deploy/index.html).

Note: If you receive errors when running AWS CLI commands, [make sure that you’re using the most recent version of the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-troubleshooting.html).

For more information on how to use the template, see [Working with AWS CloudFormation templates](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-guide.html).

AWS CloudFormation template for stopping and starting EC2 instances at regular intervals using Lambda

AWSTemplateFormatVersion: '2010-09-09'

Description: Lambda function with cfn-response.

Parameters:

instances:

Default: i-12345678

Description: Instance ID's seperated by commers

Type: String

Region:

Description: region only 1 region supported

Type: String

StopScheduled:

Default: cron(0 18 ? \* MON-FRI \*)

Description: enter an Schedule expression example cron(0 18 ? \* MON-FRI \*) see https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

Type: String

StartScheduled:

Default: cron(0 7 ? \* MON-FRI \*)

Description: enter an Schedule expression example cron(0 7 ? \* MON-FRI \* ) see https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

Type: String

Resources:

StopEC2Instances:

Type: AWS::Lambda::Function

Properties:

Runtime: python3.8

Role: !GetAtt Role.Arn

Handler: index.lambda\_handler

Timeout: 60

Environment:

Variables:

instances: !Ref instances

Region: !Ref Region

Code:

ZipFile: |

import json

import re

import os

import boto3

def lambda\_handler(event, context):

# TODO implement

instances\_str = os.environ['instances']

region = os.environ['Region']

ec2 = boto3.client('ec2', region\_name=region)

instances= re.findall(r"i-[0-9a-z]{17}|i-[0-9a-z]{8}", instances\_str)

print('stopped your instances: ' + str(instances) + "in Region "+ region)

ec2.stop\_instances(InstanceIds=instances)

return {

'statusCode': 200,

'body': json.dumps('stopped your instances: ' + str(instances))

}

Description: Function that stops instances

permissionForEventsToInvokeStopEC2Instances:

Type: AWS::Lambda::Permission

Properties:

FunctionName: !GetAtt StopEC2Instances.Arn

Action: lambda:InvokeFunction

Principal: events.amazonaws.com

"SourceArn" : !GetAtt StopScheduledRule.Arn

StartEC2Instances:

Type: AWS::Lambda::Function

Properties:

Runtime: python3.8

Role: !GetAtt Role.Arn

Handler: index.lambda\_handler

Timeout: 60

Environment:

Variables:

instances: !Ref instances

Region: !Ref Region

Code:

ZipFile: |

import json

import re

import os

import boto3

def lambda\_handler(event, context):

# TODO implement

instances\_str = os.environ['instances']

region = os.environ['Region']

ec2 = boto3.client('ec2', region\_name=region)

instances= re.findall(r"i-[0-9a-z]{17}|i-[0-9a-z]{8}", instances\_str)

print('started your instances: ' + str(instances)+ "in Region "+ region)

ec2.start\_instances(InstanceIds=instances)

return {

'statusCode': 200,

'body': json.dumps('started your instances: ' + str(instances))

}

Description: Function that started instances

permissionForEventsToInvokeStartEC2Instances:

Type: AWS::Lambda::Permission

Properties:

FunctionName: !GetAtt StartEC2Instances.Arn

Action: lambda:InvokeFunction

Principal: events.amazonaws.com

"SourceArn" : !GetAtt StartScheduledRule.Arn

Role:

Type: 'AWS::IAM::Role'

Properties:

AssumeRolePolicyDocument:

Version: 2012-10-17

Statement:

- Effect: Allow

Principal:

Service:

- lambda.amazonaws.com

Action:

- 'sts:AssumeRole'

Path: /

ManagedPolicyArns:

- "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"

Policies:

- PolicyName: Ec2permissions

PolicyDocument:

Version: 2012-10-17

Statement:

- Effect: Allow

Action:

- "ec2:StartInstances"

- "ec2:StopInstances"

Resource: '\*'

StopScheduledRule:

Type: AWS::Events::Rule

Properties:

Description: "ScheduledRule"

ScheduleExpression: !Ref StopScheduled

State: "ENABLED"

Targets:

-

Arn: !GetAtt StopEC2Instances.Arn

Id: "TargetFunctionV1"

StartScheduledRule:

Type: AWS::Events::Rule

Properties:

Description: "ScheduledRule"

ScheduleExpression: !Ref StartScheduled

State: "ENABLED"

Targets:

-

Arn: !GetAtt StartEC2Instances.Arn

Id: "TargetFunctionV1"