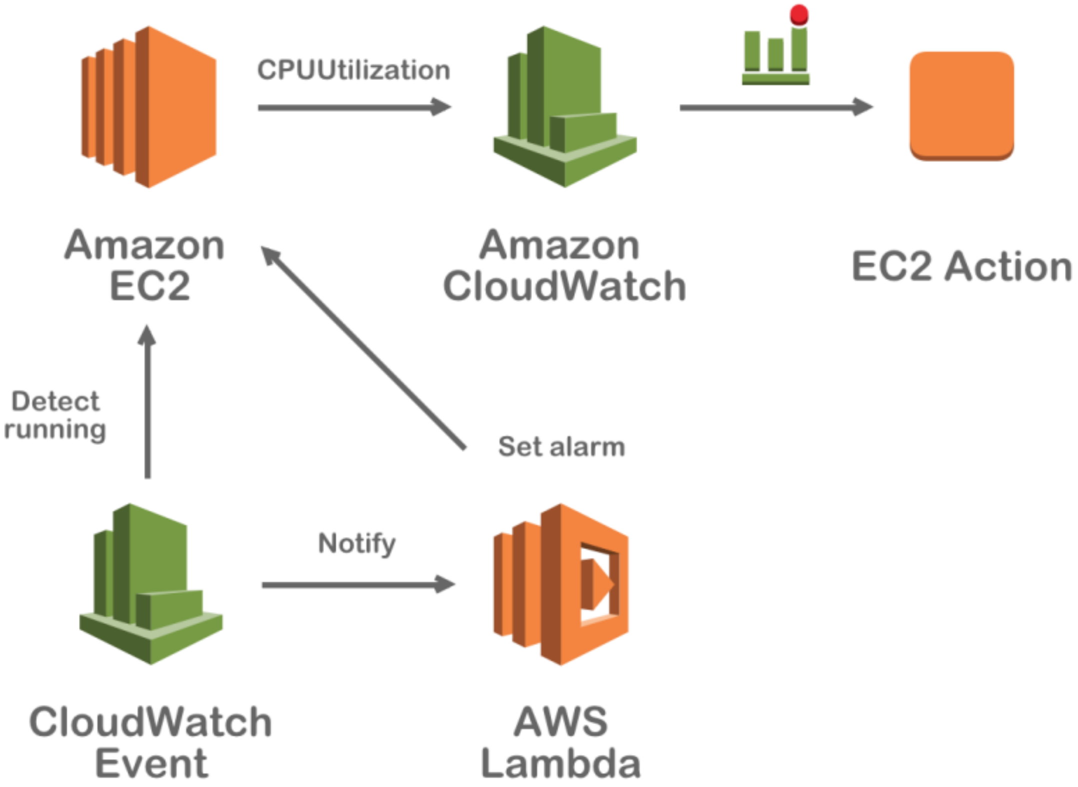
Automatically Stopping Idle Instance Using Lambda and CloudWatch

Save AWS EC2 Cost by Automatically Stopping Idle Instance Using Lambda and CloudWatch
In machine learning, especially deep learning, GPU is often used to train machine learning models. In most cases, GPU instances on the cloud services are selected by engineers because they are easier to prepare. But since the GPU instance is expensive, it is a waste of cost if you leave it running when the state of the instance is idle. In such a case, you can significantly reduce the cost by automatically stopping the instance.
First, we will attach a CloudWatch alarm to an EC2 instance when the state of the instance becomes "running". The alarm monitors some metrics and automatically stops the instance when the alarm is raised. In order to implement these behaviors, we will use AWS Lambda and CloudWatch. Both services are serverless, so we don't have to care about their operation and maintenance. The overview is as follows:



This tutorial is divided into 3 parts; they are:

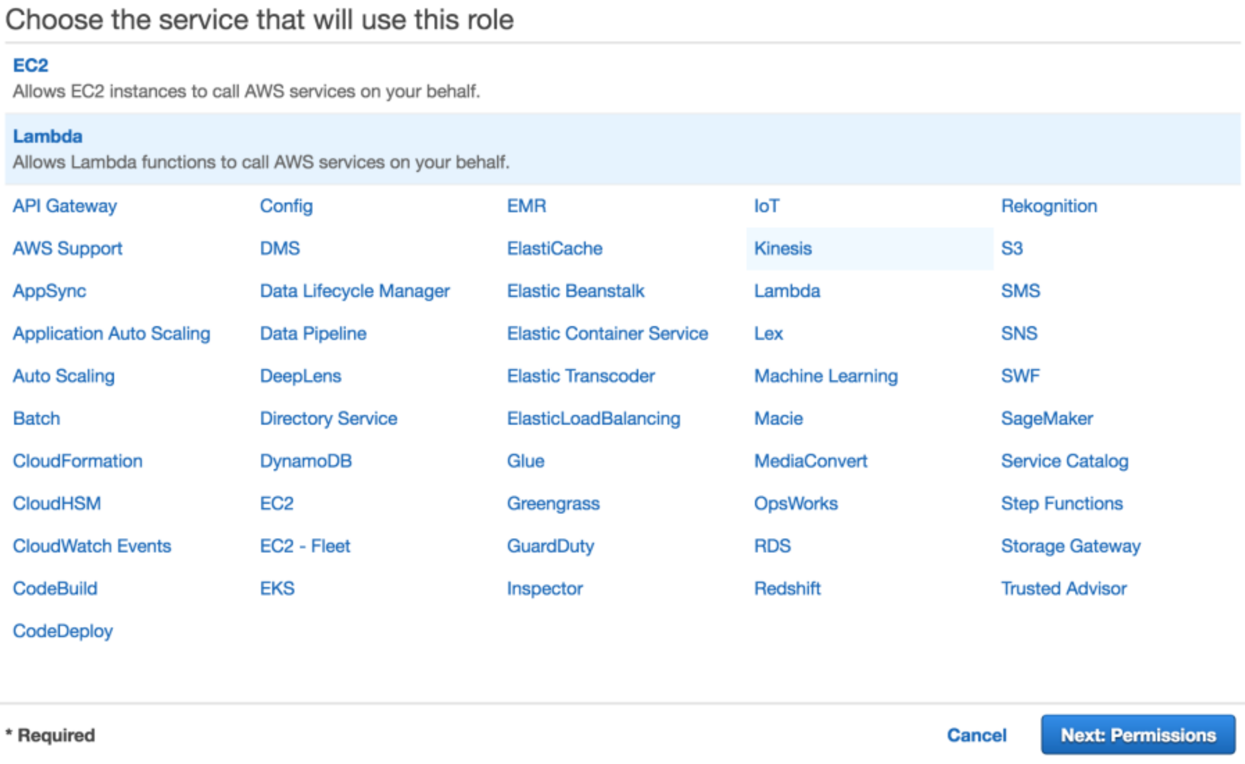
- Create an IAM role
- Create Lambda function
- Create CloudWatch Event

Let's get started.

Create an IAM Role

Let's create a role that can access to EC2 instance from the Lambda function.

Move to the "Create IAM role" page and select Lambda in "Choose the service that will use this role" and click "Next: Permissions" button:



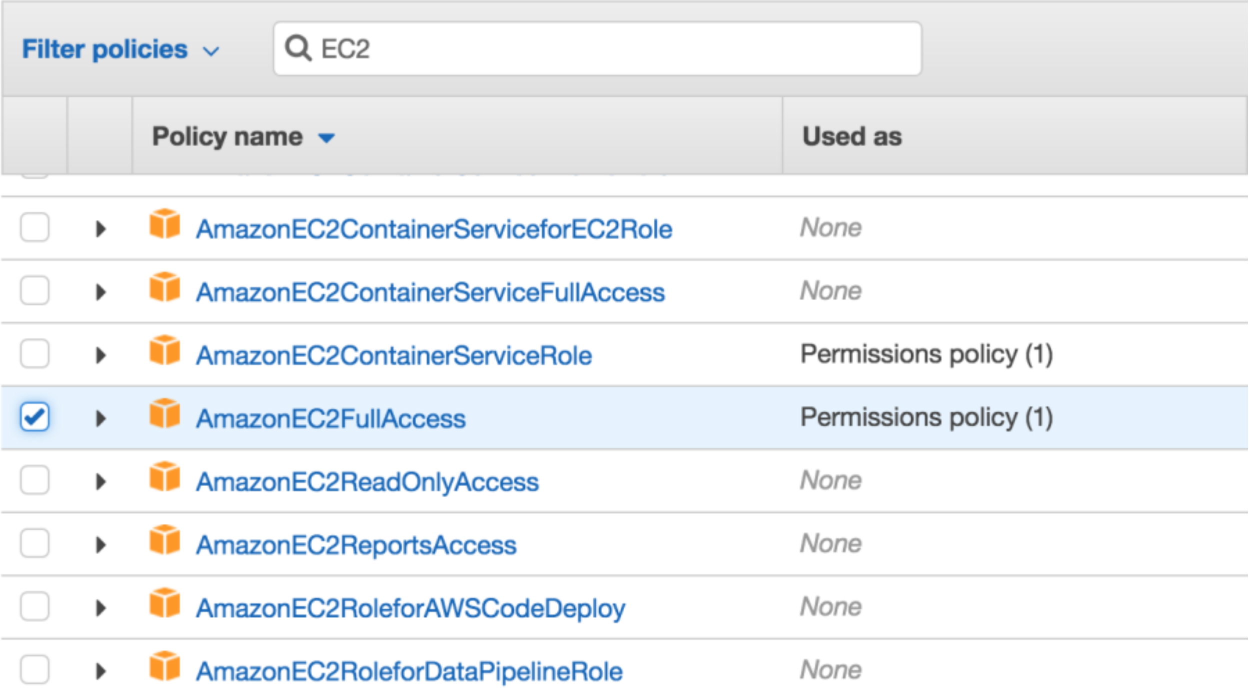
Choose the service that will use this role

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

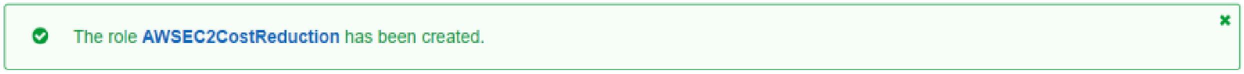
API Gateway
AWS Support
AppSync
Application Auto Scaling
Auto Scaling
Batch
CloudFormation
CloudHSM
CloudWatch Events
CodeBuild
CodeDeploy
Config
DMS
Data Lifecycle Manager
Data Pipeline
DeepLens
Directory Service
DynamoDB
EC2
EC2 - Fleet
EKS
EMR
ElastiCache
Elastic Beanstalk
Elastic Container Service
Elastic Transcoder
ElasticLoadBalancing
Glue
Greengrass
GuardDuty
Inspector
IoT
Kinesis
Lambda
Lex
Machine Learning
Macie
MediaConvert
OpsWorks
RDS
Redshift
Rekognition
S3
SMS
SNS
SWF
SageMaker
Service Catalog
Step Functions
Storage Gateway
Trusted Advisor

Then select the policy to attach to the role. Since I want to access to the EC2 instance from Lambda function, I will attach "Amazon EC2 Full Access" policy. After selecting the policy, click "Next: Review" button. If you want to set only the minimum permission, Create a policy yourself and attach it



		Policy name	Used as
☐	▶	AmazonEC2ContainerServiceforEC2Role	None
☐	▶	AmazonEC2ContainerServiceFullAccess	None
☐	▶	AmazonEC2ContainerServiceRole	Permissions policy (1)
☑	▶	AmazonEC2FullAccess	Permissions policy (1)
☐	▶	AmazonEC2ReadOnlyAccess	None
☐	▶	AmazonEC2ReportsAccess	None
☐	▶	AmazonEC2RoleforAWSCodeDeploy	None
☐	▶	AmazonEC2RoleforDataPipelineRole	None

After input the role name, select "Create role" button to create the role. Here, the role is named "AWSLambdaEC2FullAccess".



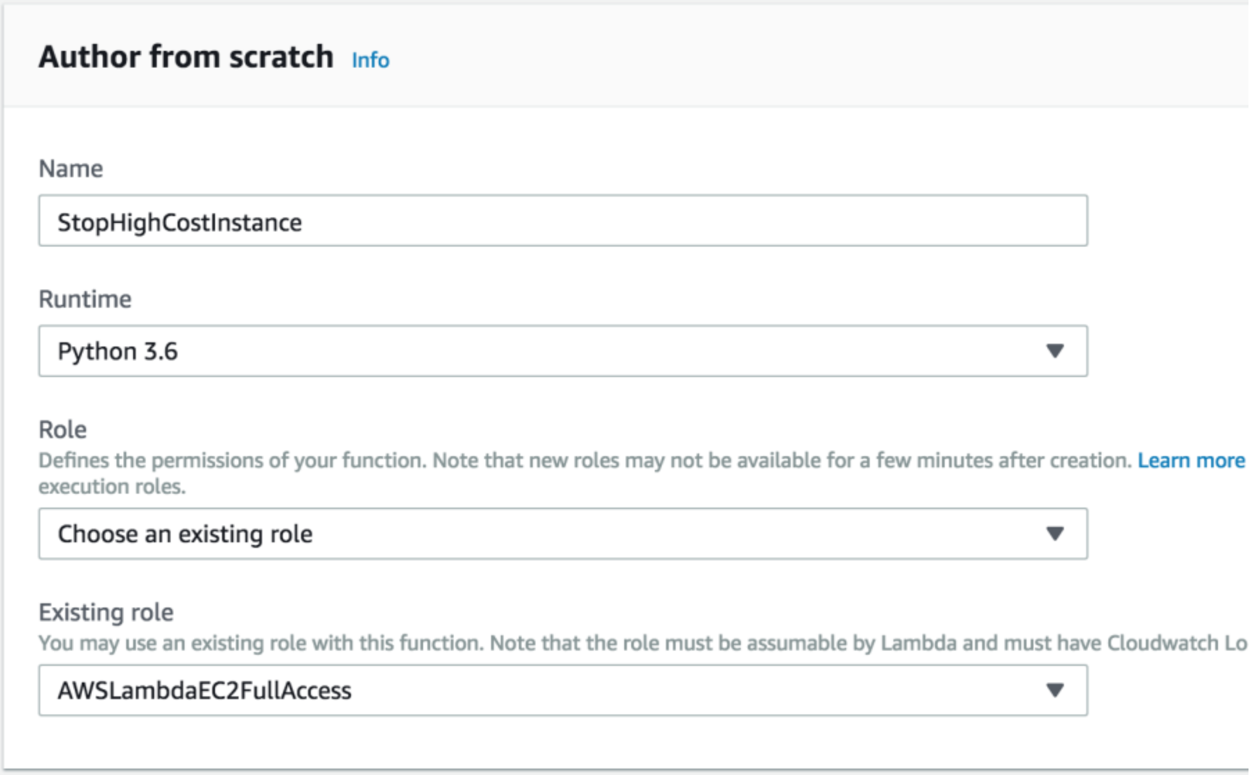
The role **AWSEC2CostReduction** has been created.

Create a Lambda function

Next we will create a Lambda function. In the Lambda function, we do two things:

- Get the EC2 instance ID
- Create an alarm and attach it to the EC2 instance

After moving to the Lambda page, you should set the function setting as follows. Here, you must select the previously created role as existing role:



Author from scratch Info

Name

StopHighCostInstance

Runtime

Python 3.6 ▼

Role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more execution roles.

Choose an existing role ▼

Existing role

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Lo

AWSLambdaEC2FullAccess ▼

After creating the function, we need to write some code in Python. Use boto3 package to get the instance id and create the CloudWatch alarm and attach it to the instance. It can be written as follows.

`

```
import boto3

def put_cpu_alarm(instance_id):
cloudWatch = boto3.client('cloudwatch')
cloudWatch.put_metric_alarm(
AlarmName = f'CPU_ALARM_{instance_id}',
AlarmDescription = 'Alarm when server CPU does not exceed 10%',
AlarmActions = ['arn:aws:automate:us-east-1:ec2:stop'],
MetricName = 'CPUUtilization',
Namespace = 'AWS/EC2' ,
Statistic = 'Average',
Dimensions = [{'Name': 'InstanceId', 'Value': instance_id}],
Period = 300,
EvaluationPeriods = 3,
Threshold = 10,
ComparisonOperator = 'LessThanOrEqualToThreshold',
TreatMissingData = 'notBreaching'
)

def lambda_handler(event, context):
instance_id = event['detail']['instance-id']
ec2 = boto3.resource('ec2')
instance = ec2.Instance(instance_id)
if instance.instance_type.endswith('xlarge'):
put_cpu_alarm(instance_id)
```

`

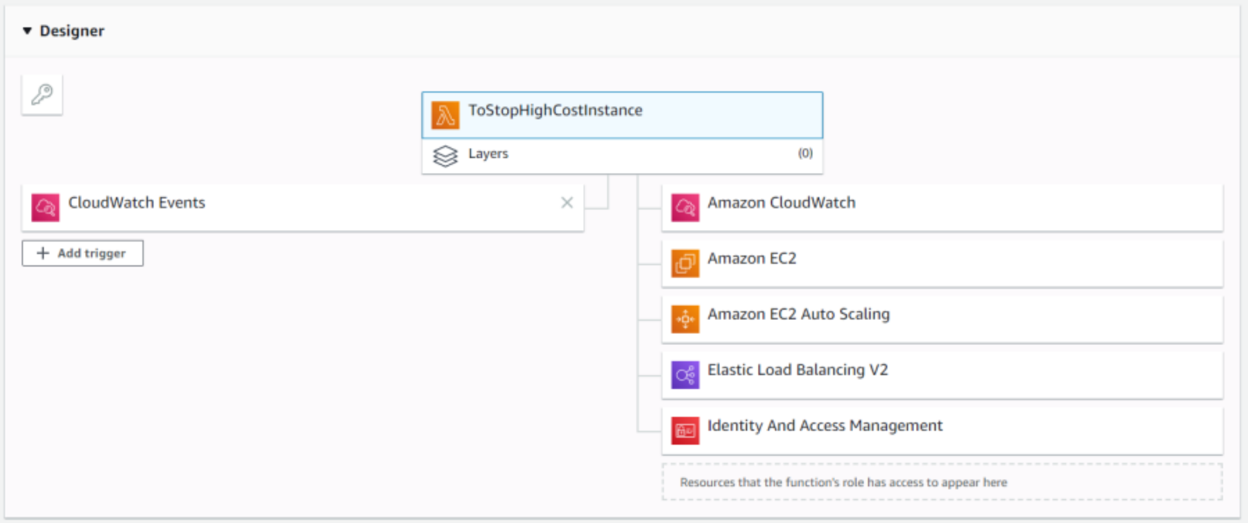
put_cpu_alarm function creates a CloudWatch alarm and attach it to the specified instance. The alarm monitors the CPU utilization on the instance and stops the instance when it reaches the utilization threshold (below 10%) 3 times in a row. Let's change this setting according to your own environment. I think that the materials listed below are useful.

- Create an Amazon CloudWatch alarm
- Creating Alarms in Amazon CloudWatch
- Docs/Available Services/CloudWatch

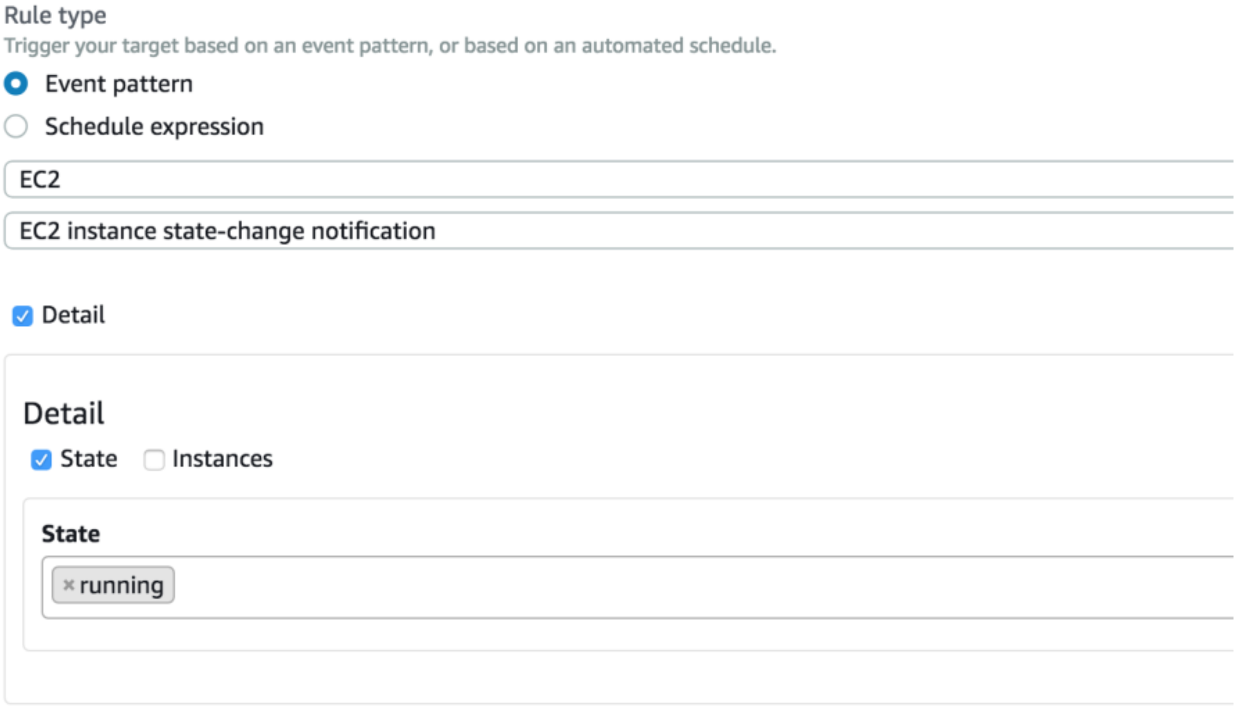
lambda_handler is the entry point to be invoked when it detects the running of the instance. Internally, we get the instance type, and call **put_cpu_alarm** if the instance type name is ends with xlarge. If you want to attach the alarm only to a specific instance type, change the condition in if statement.
In this time, I selected CPU utilization as the monitoring metrics, but you can also monitor GPU utilization by adding custom metrics. The method is detailed in the following documents.

- Monitoring GPU utilization with Amazon CloudWatch

Create CloudWatch Events
Finally, let's set up to call the Lambda function when we detect running the instance. After moving to the Lambda function page, let's select "CloudWatch Events" in "Add triggers".



After adding the trigger, select the CloudWatch Events you added. Then you will see a form for setting the trigger at the bottom of the page. After selecting "Create New Rule", set it as follows. By setting like this, you can call the Lambda function when the instance starts up.



Rule type
Trigger your target based on an event pattern, or based on an automated schedule.

● Event pattern
○ Schedule expression

EC2

EC2 instance state-change notification

☑ Detail

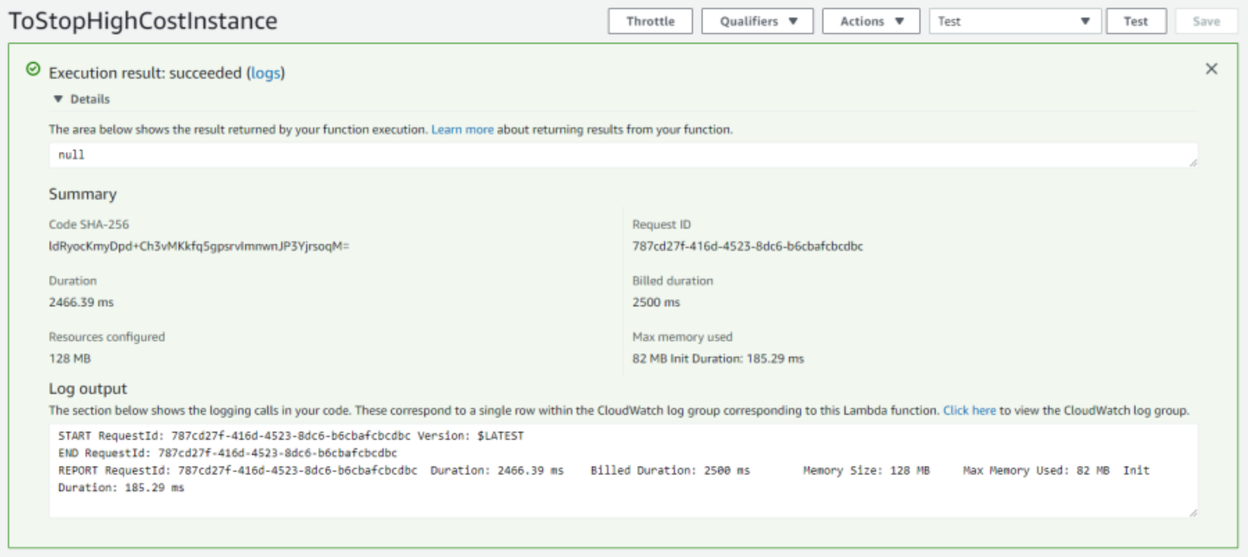
Detail

☑ State □ Instances

State

× running

After executing the lambda function we will get the output as shown in below.



ToStopHighCostInstance

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
null
```

Summary

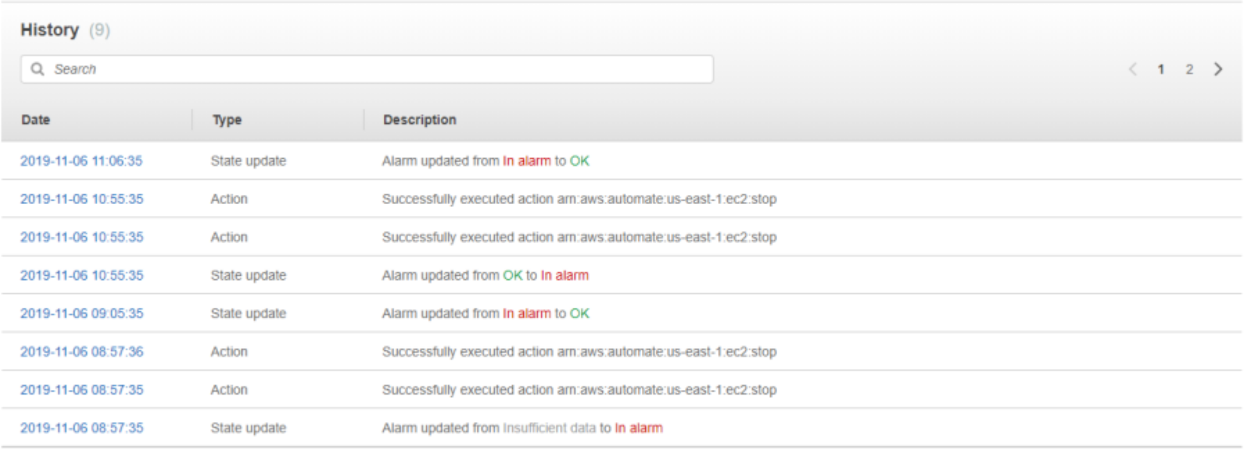
Code SHA-256 ldRyocKmyDpd+Ch3vMKkfq5gpsrvlmmwnJP3YjrsoqM=	Request ID 787cd27f-416d-4523-8dc6-b6cbafcbcdbc
Duration 2466.39 ms	Billed duration 2500 ms
Resources configured 128 MB	Max memory used 82 MB Init Duration: 185.29 ms

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
START RequestId: 787cd27f-416d-4523-8dc6-b6cbafcbcdbc Version: $LATEST
END RequestId: 787cd27f-416d-4523-8dc6-b6cbafcbcdbc
REPORT RequestId: 787cd27f-416d-4523-8dc6-b6cbafcbcdbc  Duration: 2466.39 ms    Billed Duration: 2500 ms       Memory Size: 128 MB     Max Memory Used: 82 MB  Init
Duration: 185.29 ms
```

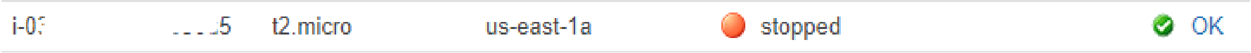
The below output shows the history of EC2 machine alarms how those changed from critical stage to OK state.



History (9)

Date	Type	Description
2019-11-06 11:06:35	State update	Alarm updated from In alarm to OK
2019-11-06 10:55:35	Action	Successfully executed action arn:aws:automate:us-east-1:ec2:stop
2019-11-06 10:55:35	Action	Successfully executed action arn:aws:automate:us-east-1:ec2:stop
2019-11-06 10:55:35	State update	Alarm updated from OK to In alarm
2019-11-06 09:05:35	State update	Alarm updated from In alarm to OK
2019-11-06 08:57:36	Action	Successfully executed action arn:aws:automate:us-east-1:ec2:stop
2019-11-06 08:57:35	Action	Successfully executed action arn:aws:automate:us-east-1:ec2:stop
2019-11-06 08:57:35	State update	Alarm updated from Insufficient data to In alarm

Once the lambda triggered after the given time intervals, EC2 will be stopped.



i-0	5	t2.micro	us-east-1a	● stopped	✓ OK

Summary

In this tutorial, you discovered how to automatically stop an EC2 instance when the state is idle.
Specifically, you learned:

- How to create an IAM role.
- How to create a Lambda function.
- How to create CloudWatch alarm.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

Reference

- Create an Amazon CloudWatch alarm
- Creating Alarms in Amazon CloudWatch

- Docs/Available Services/CloudWatch
- Monitoring GPU utilization with Amazon CloudWatch