

# How To Start & Stop EC2 Instances On AWS

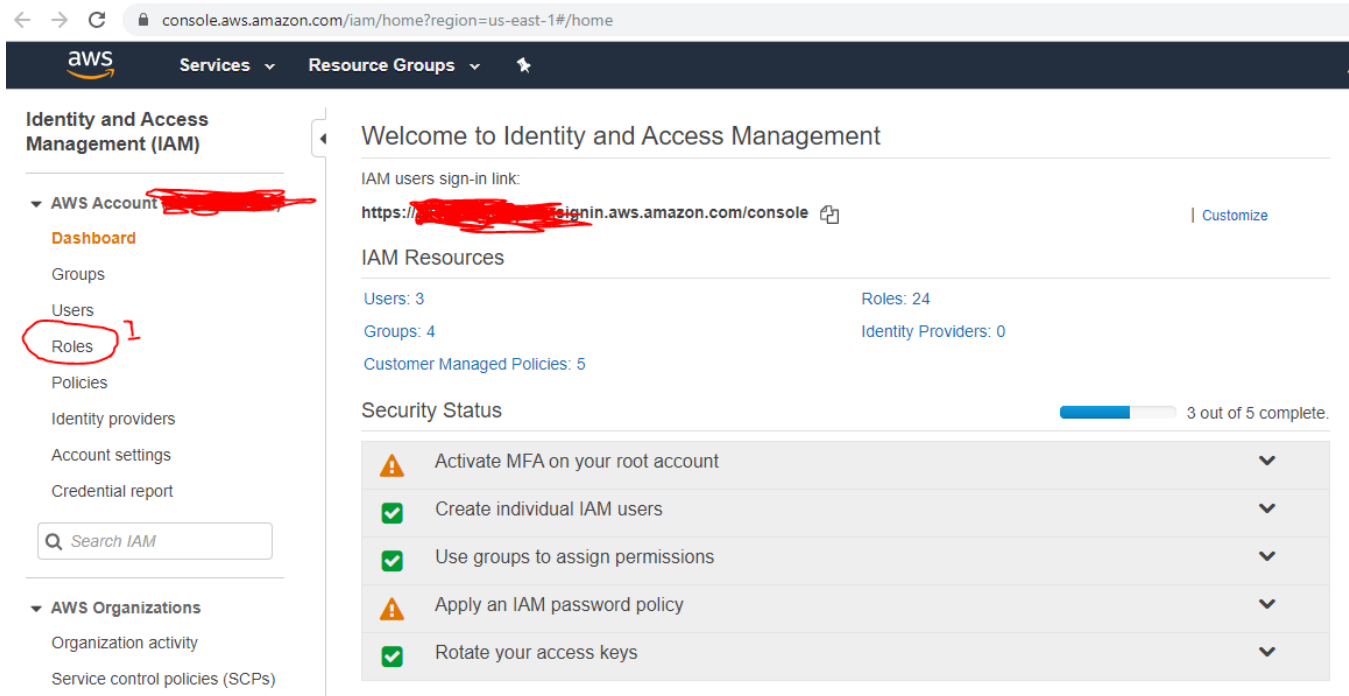
Having your EC2 instances start and stop so that they only run between work hours and not on weekends can be a bit of a puzzle. When you first look around within the AWS EC2 section you won't find any auto start-stop feature. Instead you need to use some of the other features in conjunction with EC2 to accomplish this. At first you'll be introduced to some new technologies that are beyond the scope of this article but don't be shy, it's very easy to get it up and running.

We will use the below technologies :

- IAM
- CloudWatch
- Lambda
- EC2

## Create IAM Role :


Navigate to AWS IAM service & Roles section, and click on Roles.





Create a role to allow the Lambda functions to call AWS services.


# Create role

## Select type of trusted entity

**AWS service**  
EC2, Lambda and others

**Another AWS account**  
Belonging to you or 3rd party

**Web identity**  
Cognito or any OpenID provider

**SAML 2.0 federation**  
Your corporate directory

Allows AWS services to perform actions on your behalf. [Learn more](#)

## Choose the service that will use this role

### EC2

Allows EC2 instances to call AWS services on your behalf.

### Lambda ✓ 2

Allows Lambda functions to call AWS services on your behalf.

API Gateway	CodeBuild	EKS	Kinesis	S3
AWS Backup	CodeDeploy	EMR	Lambda	SMS
AWS Chatbot	CodeStar Notifications	ElastiCache	Lex	SNS
AWS Support	Comprehend	Elastic Beanstalk	License Manager	SWF
Amplify	Config	Elastic Container Service	Machine Learning	SageMaker
AppStream 2.0	Connect	Elastic Transcoder	Macie	Security Hub
AppSync	DMS	ElasticLoadBalancing	MediaConvert	Service Catalog
Application Auto Scaling	Data Lifecycle Manager	Forecast	Migration Hub	Step Functions
Application Discovery	Data Pipeline	Global Accelerator	OpsWorks	Storage Gateway

\* Required

Cancel

Next: Permissions

Click on Next button to save the permissions and to attach the roles which are required.

# Create role

## ▼ Attach permissions policies

Choose one or more policies to attach to your new role.


Create policy

↺

Filter policies ▼

Q amazonec2full

Showing 1 result

	Policy name ▼	Used as
4 ✓ <input checked="" type="checkbox"/>	▶  AmazonEC2FullAccess	Permissions policy (1)

## ▶ Set permissions boundary

\* Required

Create a Tag for filter purpose.

## Create role

1 2 3 4

### Add tags (optional)

IAM tags are key-value pairs you can add to your role. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this role. [Learn more](#)

Key	Value (optional)	Remove
<div>6 ✓ Name</div>	<div>AWSEC2CostReduction</div>	<div>✕</div>
<div>Add new key</div>		

You can add 49 more tags.

Cancel

Previous

✓ 7  
Next: Review

Review the role and save it by specifying the Role name & Role description.

# Create role

1 2 3 4

## Review

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



**Role name\***

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

**Role description**

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

**Trusted entities** AWS service: lambda.amazonaws.com

**Policies**  [AmazonEC2FullAccess](#) 

**Permissions boundary** Permissions boundary is not set

The new role will receive the following tag

Key	Value
Name	AWSEC2CostReduction

\* Required

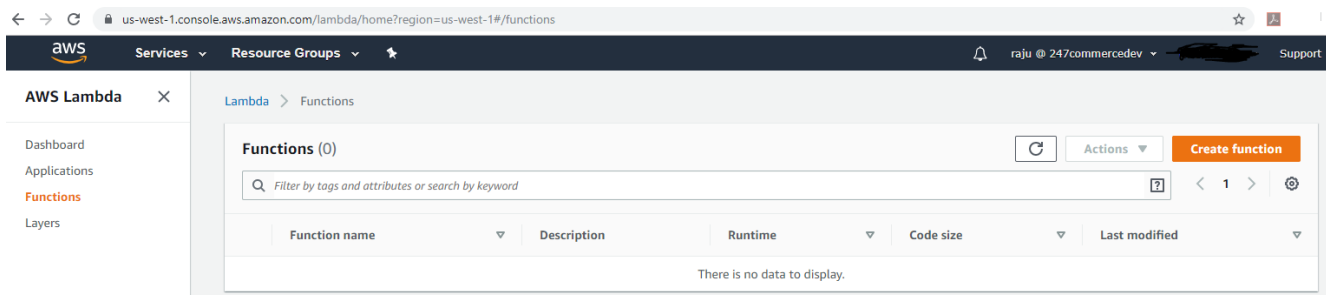
[Cancel](#)

[Previous](#)

[Create role](#)

## Create Lambda Function :

To create a lambda function select the Lambda service and navigate to Functions, click on 'Create function' button to create it.




Lambda > Functions > Create function

## Create function [Info](#)

Choose one of the following options to create your function.


**Author from scratch**

Start with a simple Hello World example.




**Use a blueprint**

Build a Lambda application from sample code and configuration presets for common use cases.



**Browse serverless app repository**

Deploy a sample Lambda application from the AWS Serverless Application Repository.



---

### Basic information

**Function name**  
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function.

**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](#)

[Cancel](#)
[Create function](#)

Paste the code to start instances, for stop function replace the start word with stop in the code and repeat the same steps.

#### For Start :

```
import boto3
instances = ['i-XXXXXX1', 'i-XXXXXX2']
def lambda_handler(event, context):
    ec2 = boto3.client('ec2', region_name='us-east-1')
    ec2.start_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))
```

#### For Stop :

```
import boto3
instances = ['i-XXXXXX1', 'i-XXXXXX2']
def lambda_handler(event, context):
    ec2 = boto3.client('ec2', region_name='us-east-1')
    ec2.stop_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))
```

### Function code [Info](#)

Code entry type:

Runtime:

Handler:

Environment

- EC2Start\_Dev\_Instance
  - lambda\_function.py

lambda\_function.py

```
1 import boto3
2 instances = ['i-0...', 'i-0...', 'i-0...']
3 def lambda_handler(event, context):
4     ec2 = boto3.client('ec2', region_name='us-east-1')
5     ec2.start_instances(InstanceIds=instances)
6     print('started your instances: ' + str(instances))
```

6:55 Python Spaces: 4

Execution Result X

Execution results

No execution results yet

### Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

► Encryption configuration

### Tags

You can use tags to group and filter your functions. A tag consists of a case-sensitive key-value pair. [Learn more](#)

### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

#### 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.

[View the AWSEC2CostReduction role](#) on the IAM console.

### Basic settings

#### Description

#### Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

#### Timeout [Info](#)

min  sec

### Network

#### Virtual Private Cloud (VPC) [Info](#)

Choose a VPC for your function to access.

### AWS X-Ray [Info](#)

Enable active tracing to record timing and error information for a subset of invocations.

☐ Active tracing

### Concurrency

Unreserved account concurrency **1000**

☒ Use unreserved account concurrency

☐ Reserve concurrency

### Error handling

#### DLQ resource [Info](#)

Choose the AWS service to send the event payload to after maximum retries are exceeded.

### Auditing and compliance

AWS CloudTrail can log this function's invocations for operational and risk auditing, governance, and compliance. [Get started](#) on the CloudTrail console.

☑ Successfully updated the function EC2Start\_Dev\_Instance.

[Lambda](#) > [Functions](#) > EC2Start\_Dev\_Instance

ARN - arn:aws:lambda:us-east-1:123456789012:function:EC2Start\_Dev\_Instance

EC2Start\_Dev\_Instance

▼

▼

▼

## Configure test event



A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☐ Create new test event

☒ Edit saved test events

Saved test event

Start



```
1 {  
2   "action": "start"  
3 }
```

Delete

Cancel

Save

For stop function, create the event with stop name.

### Create CloudWatch Rule :

To create the CloudWatch rule, select the CloudWatch service and navigate to Rules section, and click on Create rule button.



← → ↻ us-west-1.console.aws.amazon.com/cloudwatch/home?region=us-west-1#rules:

Services ▾

Resource Groups ▾

🌟

CloudWatch

Dashboards

Alarms

ALARM 0

INSUFFICIENT 0

OK 0

Billing

Logs

Log groups

Insights

Metrics

Events

**Rules**

Event Buses

Settings **NEW**

Favorites

[+ Add a dashboard](#)

# Rules

Rules route events from your AWS resources for processing by selected targets. Y

Create rule

Actions ▾

Status All ▾

Name

Status

Select Cron expression under Schedule option, specify the time at what time you need to start/stop the instances. In the target section select lambda function respective to the CloudWatch rule.

## 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

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

▶ Show sample event(s)

### Targets

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

Lambda function ▾

Function\*  ▾

▶ Configure version/alias

▶ Configure input

Add target\*

\* Required

Cancel

Configure details

## Step 2: Configure rule details

### Rule definition

Name\*

EC2Start\_Dev\_Instance

Description

To Start EC2 Dev Instances

State

☐ Enabled

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

\* Required

Cancel

Back

Update rule

Specify the rule name and description and update the rule.  
Repeat the same process for Stop rule also.