

CS LAB 3



How to Auto Stop Virtual Instances in AWS Cloud to Save the cost during non-working hours

To **auto-stop virtual instances (EC2 instances)** in AWS during non-working hours and save costs, you can use **AWS Instance Scheduler** or create your own solution using **AWS Lambda with CloudWatch Events (EventBridge)**.

✓ Option 1: Using AWS Instance Scheduler (Recommended)

AWS Instance Scheduler is a solution provided by AWS to automatically start and stop EC2 and RDS instances based on custom schedules.

🔧 Steps:

1. Launch the AWS Instance Scheduler solution:

- Go to the AWS [Solutions Library](#)
- Click "**Launch Solution**"
- This uses a **CloudFormation template** to deploy.

2. Define a Schedule:

- During deployment, you'll define schedules like:

```
makefile
WorkingHours: Mon-Fri 09:00-18:00
```

- Instances will stop outside this range.

3. Tag Your EC2 Instances:

- Add tags to the instances you want to control:

```
makefile
Key: Schedule
Value: WorkingHours
```

- 4. **Done** – The scheduler will now **automatically start and stop** instances based on the defined time.

✓ Option 2: Manual Setup with Lambda + EventBridge (CloudWatch Events)

If you want a **custom or lightweight solution**, this is the way.

Printed using [ChatGPT to PDF](#), powered by PDFCrowd [HTML to PDF API](#).

Steps:

Step 1: Create Lambda Function

Create two Lambda functions:

- **Start EC2 instances**
- **Stop EC2 instances**

Example Python code for stopping:

```
python

import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')
    # Replace with your instance IDs
    instance_ids = ['i-0123456789abcdef0']
    ec2.stop_instances(InstanceIds=instance_ids)
    print('Stopped instances: ' + str(instance_ids))
```

And similar for starting (`start_instances` instead of `stop_instances`).

Step 2: Create EventBridge Rules (CloudWatch Events)

- Go to **Amazon EventBridge > Rules > Create Rule**
- Set the **Schedule** (e.g., cron expression like `cron(0 18 ? * MON-FRI *)` for 6 PM on weekdays)
- Set the target as the appropriate **Lambda function**

Step 3: Give Lambda Permissions

Make sure your Lambda has an IAM Role with `ec2:StartInstances` and `ec2:StopInstances` permissions.

Example Schedule

Action	Time (IST)	CRON (UTC)
Stop	6:00 PM (IST)	`cron(30 12 ? * MON-FRI *)`
Start	9:00 AM (IST)	`cron(3 3 ? * MON-FRI *)`

Tips to Save More Cost:

- Use **Spot Instances** if possible.
- Use **Auto Scaling Groups** with scheduled actions.
- Use **Savings Plans** or **Reserved Instances** if the instance is always required during certain hours.

Would you like me to help you create the Lambda + EventBridge setup step-by-step?



How to Auto Stop Virtual Instances in AWS Cloud to Save the cost during non-working hours

Select a Region. For example North Virginia US -East

Step 1 Create a EC2 instance

Step 2 Create a IAM Role with Lambda services along with EC2Full access Permission

Step 3 Copy the Instance ID (Virtual Machine ID) from EC2 Instance

Step 4 Create a Lambda function along with the Code and replace the instance ID and region in the code.

```
import boto3
import botocore
def lambda_handler(event, context):
    ec2 = boto3.client('ec2', region_name='us-east-1') # Corrected region name
    instance_id = "i-0cb771a7775f3909f" # Replace with your EC2 Instance ID
    try:
        response = ec2.stop_instances(InstanceIds=[instance_id])
        return {
            'statusCode': 200,
            'body': f"Stopping instance {instance_id}. Response: {response}"
        }
    except botocore.exceptions.ClientError as e:
        return {
            'statusCode': 400,
            'body': f"Error stopping instance {instance_id}: {str(e)}"
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': f"An unexpected error occurred: {str(e)}"
        }
```

Step 5 Click on the Deploy button to save the code

Step 6 Go to the configuration option in Lambda function and change the execution time from 3 seconds to 1

min

Step 7 Click on the Test option in Lambda function and create a Test event

Step 8 Click on the Test

Step 9 Go to the EC2 console and check your EC2 instance is stopped or not

Here's a **complete step-by-step guide** to **Auto Stop EC2 (Virtual Machine) Instances in AWS** during non-working hours (e.g., after 6 PM) using **AWS Lambda** in **North Virginia (us-east-1)** region:

✓ Objective:

Automatically stop an EC2 instance during non-working hours to save cost.

🌐 Region:

North Virginia (US East - `us-east-1`)

Steps:

✓ Step 1: Create an EC2 Instance

1. Go to **EC2 Dashboard**.
 2. Launch an instance:
 - AMI: Amazon Linux 2 (free tier eligible)
 - Instance type: t2.micro (free tier eligible)
 3. Complete setup and launch.
 4. **Copy the Instance ID** (e.g., `i-0cb771a7775f3909f`).
-

✓ Step 2: Create IAM Role for Lambda

1. Go to **IAM > Roles > Create Role**
 2. **Select "AWS service" > Lambda**
 3. Attach policies:
 - **AmazonEC2FullAccess**
 4. Name the role: `LambdaEC2StopRole`
 5. Create role.
-

✓ Step 3: Create a Lambda Function

1. Go to **Lambda > Create Function**
 2. Name: `StopEC2InstanceFunction`
 3. Runtime: **Python 3.12**
 4. Choose **existing role**: `LambdaEC2StopRole`
 5. Click **Create Function**
-

✓ Step 4: Add the Python Code

Paste this code in the **Code source** editor:

```
python

import boto3
import botocore

def lambda_handler(event, context):
    ec2 = boto3.client('ec2', region_name='us-east-1') # North Virginia region
    instance_id = "i-0cb771a7775f3909f" # Replace with your EC2 Instance ID

    try:
        response = ec2.stop_instances(InstanceIds=[instance_id])
```

```

    return {
        'statusCode': 200,
        'body': f"Stopping instance {instance_id}. Response: {response}"
    }
except botocore.exceptions.ClientError as e:
    return {
        'statusCode': 400,
        'body': f"Error stopping instance {instance_id}: {str(e)}"
    }
except Exception as e:
    return {
        'statusCode': 500,
        'body': f"An unexpected error occurred: {str(e)}"
    }

```

✓ Step 5: Deploy the Code

- Click the **Deploy** button to save your changes.

✓ Step 6: Update Timeout Settings

1. Go to the **Configuration tab > General Configuration**
2. Edit **timeout** → Set it to **1 minute**
3. Click **Save**

✓ Step 7: Create a Test Event

1. Click **Test**
2. Create a test event with any name (e.g., ``StopTestEvent``)
3. Use this default JSON:

```

json

{}

```

✓ Step 8: Run the Test

1. Click the **Test** button again
2. ✓ Output will show if the EC2 instance was successfully stopped

✓ Step 9: Confirm in EC2 Console

1. Go to **EC2 > Instances**
2. Your instance should now be in **stopped** state

Bonus (Optional): Automate Using EventBridge

To **automate daily stopping** (e.g., at 6 PM), follow this:

Step A: Go to EventBridge > Create Rule

- Rule name: ``StopEC2EveryEvening``
- Schedule pattern: ``cron(0 22 ? * MON-FRI *)`` → 6 PM IST = 12:30 PM UTC

Step B: Add Target

- Target: Your Lambda function

Now the instance will **automatically stop every weekday at 6 PM IST**.

Would you also like help creating a similar **"Start EC2" function** for mornings?



```
"statusCode": 400,
  "body": "Error stopping instance i-0709588f172a00e4f: An error occurred (UnauthorizedOperation)
when calling the StopInstances operation: You are not authorized to perform this operation. User:
arn:aws:sts::651706743518:assumed-role/toss-role-27trkws9/toss is not authorized to perform:
ec2:StopInstances on resource: arn:aws:ec2:us-east-1:651706743518:instance/i-0709588f172a00e4f
because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure
message:
b0BYPuOqDp1OsQAeAE7KnQuUQbl9ByodNXMEMVT1h9BGCNLZP88VJzkm3TjU6tIjb6KgEZHqfzMVy9tVS.
KHeXo8UO3PF8hI2mRh0IaJFKAyumjBXRfKMtxYONe-
ux8RoaRv7XARfP1Y8EpW1tWvLcYqfpmLHnmGxeHfQaSayVDhO0EFcFAZIsnuw4boL88Td5IFimljBKSLXMql
YTaG969NmCy5_xSGuNkm0srn-
uCdRsbhTpuYVvXk4rQ8IPSgApZTLzLXjxSHL6UbduMmNWbqX51nXB6U1iRMUbhO-
Mb6nrXeLVPxqg_fE2W-7ZIYyoatf5Zbhaq8l39EJGTFXilJH8k6sPZYU7y4Puo1o26D8ajwj-
t2djjNqd0MkHRvrQ3344H_fa43MjrXE0FIHh7b6lQq16XObP7qQ-
jBLE4TkpyDK2LulqQ1z32XpcCKUIAAS2AMrcYb8ZmtBjdKuCchaM6zwM-DqmGwc0-
RrOJ1Kj_Z02w6XKGtOdoWIEQLjruMT1_SI9Vvk4666KN4IrKPi6Osb36STZ3ILGRsrYovRrk0xD3vf-q-
8kntQkS2PbWb5HSDsC8G80dTmtRoeVZzbQ0dyR6IXUJ1_4XUQe-
_PP2qPAZ8rT8p2iSpPF0beXUibqmm3GDMUX-k_80C_621jdJ0uHtHulvitBFtT9rOkvCy4fj-
NcIBoG9RVrJhCc"
}
```

The error you're seeing:

```
arduino
```

```
"Error stopping instance... because no identity-based policy allows the ec2:StopInstances action."
```

means that **the IAM role used by your Lambda function (``toss-role-27trkws9``) does not have permission** to stop EC2 instances.

How to Fix This:


You need to **attach the correct permissions** to the IAM role (``toss-role-27trkws9``) used by your Lambda function.

Step-by-Step Fix:

Step 1: Go to the IAM Console

- Open the AWS Console.
- Navigate to **IAM > Roles**.
- Search for the role: ``toss-role-27trkws9``.

Step 2: Attach Policy for EC2 Access

- Click on the role name.
- Under **Permissions**, click "Add permissions" > "Attach policies".
- Search for: ``AmazonEC2FullAccess``
-  Select it and click **Attach policy**.

OR create a **custom minimal policy**:

```
json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StopInstances",
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    }
  ]
}
```

Attach this instead if you want to **limit access to just stopping instances**.

Step 3: Retry Lambda

- Go back to your **Lambda function**.
- Click **Test** again.
- Now it should work without the unauthorized error.

Optional (Recommended for Better Security)

If you want to restrict it to a **specific instance only**, modify the custom policy's ``Resource``:

```
json
```

```
"Resource": "arn:aws:ec2:us-east-1:<account-id>:instance/i-0709588f172a00e4f"
```

Replace `<account-id>` with your AWS Account ID.

Let me know if you'd like help creating the custom policy or attaching it through the console.



```
ode": 200,
  "body": "Stopping instance i-0709588f172a00e4f. Response: {'StoppingInstances': [{'InstanceId': 'i-0709588f172a00e4f', 'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'PreviousState': {'Code': 16, 'Name': 'running'}}], 'ResponseMetadata': {'RequestId': '81759fea-85d0-402f-b9f3-aea9ed113144', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '81759fea-85d0-402f-b9f3-aea9ed113144', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml; charset=UTF-8', 'content-length': '411', 'date': 'Tue, 13 May 2025 14:45:51 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}"
```

✅ **Success!** Your Lambda function has **successfully initiated the stop process** for the EC2 instance:

↺ Current EC2 State:

- **Instance ID:** `i-0709588f172a00e4f`
- **Previous State:** `running`
- **Current State:** `stopping`

This means AWS is now stopping your virtual machine to **save costs**.

↺ What to Do Next

✅ Verify in EC2 Console

1. Go to **EC2 Dashboard**.
2. Find instance `i-0709588f172a00e4f`.
3. Confirm its status is:
 - `stopping` → will soon become `stopped`.

🕒 Optional Next Step: Automate Daily Shutdown

If you want this to **happen automatically every day** (e.g., 6 PM IST):

+ Use Amazon EventBridge (CloudWatch Events)

1. Go to **Amazon EventBridge > Rules > Create Rule**

2. Set a schedule (e.g., for 6 PM IST):

SCSS

```
cron(30 12 ? * MON-FRI *) # 12:30 PM UTC = 6:00 PM IST
```

3. Add **Target**:

- **Target type**: Lambda Function
- Choose your `StopEC2InstanceFunction``

4. Click **Create Rule**

Would you like help setting up a similar **"Start EC2" function in the morning** (e.g., at 9 AM)?



```
"statusCode": 200,
"body": "Stopping instance i-0709588f172a00e4f. Response: {'StoppingInstances': [{'InstanceId': 'i-0709588f172a00e4f', 'CurrentState': {'Code': 80, 'Name': 'stopped'}, 'PreviousState': {'Code': 80, 'Name': 'stopped'}}], 'ResponseMetadata': {'RequestId': '5625261d-abb0-4538-84c3-9da2b0164348', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '5625261d-abb0-4538-84c3-9da2b0164348', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml; charset=UTF-8', 'content-length': '410', 'date': 'Tue, 13 May 2025 14:48:20 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}"
} i am get this error
```

This is **not an error** — it's actually a **successful response** from AWS!

Here's what it means:

✓ Lambda Response Summary

```
json

{"statusCode": 200
"body": "Stopping instance i-0709588f172a00e4f. Response: {
  'StoppingInstances': [{
    'InstanceId': 'i-0709588f172a00e4f',
    'CurrentState': {'Code': 80, 'Name': 'stopped'},
    'PreviousState': {'Code': 80, 'Name': 'stopped'}
  }]
}"
```

💡 Explanation:

- `CurrentState: stopped`` → The EC2 instance **is already stopped**.
- `PreviousState: stopped`` → It was already in the **stopped** state before this Lambda function was triggered.

✓ Meaning:

Your Lambda function **executed successfully**, but **did not stop anything new** because the instance was **already stopped**.

✓ Next Steps:

If you're done with this Lambda (for stopping), you can now create a **second Lambda function to start the instance during working hours**.

Would you like a step-by-step guide and code for the "Start EC2" Lambda function?