lambda function for automate ec2 instances:

We are going to automate this by using python lambda function and schedule it based on the working hours. By automatically stopping and starting EC2 instances.

[] open lambda.
[] create function.

[] Click on “Author from Scratch” to write our own Lambda Function.

[] Provide a name to the Function.

[] Select “Python 3.11” from the drop-down list of Runtime.

[] Create a new role with basic Lambda Permissions.

[] For this role we must add EC2 full access permission.

[] Add the following code in the function and click on “deploy” button to save the function.

To stop running instances:

```
import boto3

# change the region name below
region = 'us-west-2'
#change the Instance ID
instances = ['i-021fe4d4edef18459',]
ec2 = boto3.client('ec2', region_name=region)
def lambda_handler(event, context):
    ec2.stop_instances(InstanceIds=instances)
    print('stopped your instances: ' + str(instances))
```

[] On the Configuration tab, choose General configuration, and then choose Edit. Set Timeout to 10 seconds, and then choose Save.
[] on the test tab, give Event name and save and test. (this is manual method).

[] for automation. Select “add trigger “and select “eventBridge”.

[] select “create a new rule “and mention rule name whichever.

[] Rule type select “Schedule expression”.

[] for setup trigger time it will follow UTC time format.

[] ex: every 3 min “**cron(*/3 * ? * mon-fri *)”.**

[] every 3 min it will check the instances and if instances are running it will stop the instances.

To start the instances:

```
import boto3
# change the region name below
region = 'us-west-2'
#change the Instance ID
instances = ['i-021fe4d4edef18459',]
ec2 = boto3.client('ec2', region_name=region)
def lambda_handler(event, context):
    ec2.start_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))
```

To stop all running instances:

```
import boto3

# Change the region name below
region = 'us-west-2'

ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    # Get all instances
    response = ec2.describe_instances()

    # Create a list to hold the IDs of running instances
    instances_to_stop = []

    # Loop through the reservations and instances
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            # Check if the instance state is 'running'
```

```
            if instance['State']['Name'] == 'running':
                instances_to_stop.append(instance['InstanceId'])

    # Stop the running instances
    if instances_to_stop:
        ec2.stop_instances(InstanceIds=instances_to_stop)
        print('Stopped your instances: ' + str(instances_to_stop))
    else:
        print('No running instances to stop.')
```

To start all the stopped instances:

```
import boto3

# Change the region name below
region = 'us-west-2'

ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    # Describe all instances
    response = ec2.describe_instances()

    # Collect instance IDs of stopped instances
    instances_to_start = []
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            if instance['State']['Name'] == 'stopped':
                instances_to_start.append(instance['InstanceId'])

    # Start the stopped instances
    if instances_to_start:
        ec2.start_instances(InstanceIds=instances_to_start)
        print('Started your instances: ' + str(instances_to_start))
    else:
        print ('No stopped instances to start.')
```

To stop multiple instances:

```python
import boto3

# change the region name below

region = 'us-west-2'

#change the Instance ID

instances = ['i-0d438b10381112d24','i-08644913a7be25593']

ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):

    ec2.stop_instances(InstanceIds=instances)

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

If instance is running it will stop and if it stopped it will start

```python
import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')
    instance_id = 'i-0b277dfa5b4b1283f'  # Replace with your EC2 instance ID

    # Describe the instance to get its current state
    response = ec2.describe_instances(InstanceIds=[instance_id])
    state = response['Reservations'][0]['Instances'][0]['State']['Name']

    if state == 'running':
        # Stop the instance if it's currently running
        ec2.stop_instances(InstanceIds=[instance_id])
        return {
            'statusCode': 200,
            'body': f'Stopping instance {instance_id}'
```

```
        }
    elif state == 'stopped':
        # Start the instance if it's currently stopped
        ec2.start_instances(InstanceIds=[instance_id])
        return {
            'statusCode': 200,
            'body': f'Starting instance {instance_id}'
        }
    else:
        return {
            'statusCode': 400,
            'body': f'Instance {instance_id} is in state {state}, cannot toggle.'
        }
```

To start the instance for particular time. If that instance is already running at that time, it will ignore and it will stop the instances in particular time. At that time instance is already stopped, it will ignore

[] we have to add Ec2 permission for created lambda role.

[] for this we will add two triggers. One for start and one for stop.

[] for setup trigger time it will follow UTC time format.

[] On the Configuration tab, choose General configuration, and then choose Edit. Set Timeout to 10 seconds, and then choose Save.

[] ex: cron(47 11 * * ? *) means it will start at 5:17 PM.

```
import boto3
import os
from datetime import datetime

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')
    instance_id = 'i-0b277dfa5b4b1283f'  # Replace with your EC2 instance ID
```

```
    # Get the current time in UTC
    current_time = datetime.utcnow()

    # Define start and stop times
    start_time = current_time.replace(hour=11, minute=42, second=0,
microsecond=0)
    stop_time = current_time.replace(hour=11, minute=50, second=0, microsecond=0)

    # Describe the instance to get its current state
    response = ec2.describe_instances(InstanceIds=[instance_id])
    state = response['Reservations'][0]['Instances'][0]['State']['Name']

    if current_time >= start_time and current_time < stop_time:  # Between 10 AM
and 7 PM
        if state == 'stopped':
            ec2.start_instances(InstanceIds=[instance_id])
            return {
                'statusCode': 200,
                'body': f'Starting instance {instance_id}'
            }
        else:
            return {
                'statusCode': 200,
                'body': f'Instance {instance_id} is already running.'
            }
    elif current_time >= stop_time:  # After 7 PM
        if state == 'running':
            ec2.stop_instances(InstanceIds=[instance_id])
            return {
                'statusCode': 200,
                'body': f'Stopping instance {instance_id}'
            }
        else:
            return {
                'statusCode': 200,
                'body': f'Instance {instance_id} is already stopped.'
            }
    else:
        return {
            'statusCode': 200,
            'body': 'Outside of defined time range. No action taken.'
        }
```

==========

using lambda function with Api gateway as trigger Controle the Ec2 instances.

[] create lambda function.
[] On the Configuration tab, choose General configuration, and then choose Edit. Set Timeout to 10 seconds, and then choose Save.
[] add permission to lambda role, i.e. ec2 full access and API gateway
[] create Api gateway for trigger.
[] select add trigger
[] select Api gateway
[] create new API
[] select HTTP API
[]in security. Select open
[] ADD

```
import boto3
import json
from datetime import datetime

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')  # Initialize the EC2 client

    try:
        data = json.loads(event["body"])["instanceid"]
        print(data)

        # Get the current time in UTC
```

```
        current_time = datetime.utcnow()

        # Define start and stop times
        start_time = current_time.replace(hour=11, minute=42, second=0,
microsecond=0)
        stop_time = current_time.replace(hour=11, minute=50, second=0,
microsecond=0)

        # Describe the instance to get its current state
        response = ec2.describe_instances(InstanceIds=[data])
        state = response['Reservations'][0]['Instances'][0]['State']['Name']

        if current_time >= start_time and current_time < stop_time:  # Between
start and stop times
            if state == 'stopped':
                ec2.start_instances(InstanceIds=[data])
                return {
                    'statusCode': 200,
                    'body': f'Starting instance {data}'
                }
            else:
                return {
                    'statusCode': 200,
                    'body': f'Instance {data} is already running.'
                }
        elif current_time >= stop_time:  # After stop time
            if state == 'running':
                ec2.stop_instances(InstanceIds=[data])
                return {
                    'statusCode': 200,
                    'body': f'Stopping instance {data}'
                }

        return {
            'statusCode': 200,
            'body': 'No action taken.'
        }

    except KeyError as e:
        return {
            'statusCode': 400,
            'body': json.dumps({"error": f"Missing key: {str(e)}"})
        }
```

```
except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({"error": str(e)})
    }
```