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

[] using postman, we will Controle the ec2 instances.
[] copy and paste the URL of API in postman, after creating API gateway through lambda trigger, we will get one url.
[]in postman, in POST: paste the URL of API
[] in postman, select body mention ec2 instances ID like
(Ex: {"instanceid":"i-08f54a180aa43a801"}

```
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=10, minute=10, second=0,
microsecond=0)
        stop_time = current_time.replace(hour=10, minute=8, 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)})
        }
```

==========

Using IST

```
import boto3
import json
from datetime import datetime, timedelta

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 and convert it to IST (UTC + 5:30)
        current_time_utc = datetime.utcnow()
        IST = timedelta(hours=5, minutes=30)
        current_time = current_time_utc + IST

        # Define start and stop times in IST
        start_time = current_time.replace(hour=14, minute=40, second=0,
microsecond=0)
        stop_time = current_time.replace(hour=15, minute=00, 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)})
    }
```