from \_\_future\_\_ import print\_function

from datetime import datetime

import boto3

from dateutil import tz

import time

# ----------Used code---------

#stop\_instance

def stop(ins):

#ins is instance-id

#"ec2" is ec2 object

ec2=boto3.resource('ec2')

#"instance" points to ins

instance=ec2.Instance(ins)

instance.stop()

#start\_instance

def start(ins):

print ("in start")

ec2=boto3.resource('ec2')

instance=ec2.Instance(ins)

instance.start()

#wait till the instance starts

print ("waiting....")

time.sleep(220)

#if even now instance doesn't start.. wait till its state becomes running

while(1):

#chck instance state (instance.state returns dictionary thus check for value corresponding to "Name" ..it returns "stopped" or "pending" or "running")

if(instance.state['Name']=='running'):

print (instance.state['Name'])

#break if its running

break

else:

print (instance.state['Name'])

continue

#update\_service\_task\_count

def update\_inc(id):

#"dynamodb" is dynamod's object

dynamodb = boto3.resource('dynamodb')

#"table" points to our ECSOrchestration table

table = dynamodb.Table('ECSOrchestration')

#"ecs" is ecs object

ecs = boto3.client('ecs')

#id 1 implies 1st instance

if(id==1):

#get row for time=9

response = table.get\_item(Key={'time': '9am'})

item = response['Item']

#item is row,ins is instance id

ins=item['ins\_id']

#call start to start the instance first

start(ins)

time.sleep(10)

#inc task-count to 1 for Final service.. since initially there was no service , count was 0 , we can directly increment count to 1

ecs.update\_service(service='Final',desiredCount=1)

try:

#wait for task to start

time.sleep(15)

#this gets task id of task that we started now

#taskArns has lists of all tasks in given service ,,here "Final" .. We get the first task since there is only task now.

x=ecs.list\_tasks(serviceName="Final")['taskArns'][0].split("/")[1]

except:

print ("in except")

#in case it couldnt get task id (probably coz instance not started yet) wait for sometime ans again try to update count and get the id

time.sleep(20)

ecs.update\_service(service='Final',desiredCount=1)

x=ecs.list\_tasks(serviceName="Final")['taskArns'][0].split("/")[1]

print ("instance\_id=" , ins," Task id = ", x)

#update task\_id in table for 9am row

# x contains task id

table.update\_item( Key={'time':'9am'},UpdateExpression='SET task = :val1',ExpressionAttributeValues={':val1': x})

#update state of instance to on

table.update\_item( Key={'time':'9am'},UpdateExpression='SET stat = :val1',ExpressionAttributeValues={':val1': 'on'})

#id 2 implies 2nd instance

elif (id==2):

#get row for 11am

response = table.get\_item(Key={'time': '11am'})

#item is row,ins is instance id

item = response['Item']

ins=item['ins\_id']

#starts the instance

start(ins)

time.sleep(5)

#inc task-count to 1 for Final service.. since initially there was no service , count was 0 , we can directly increment count to 1

ecs.update\_service(service='Final',desiredCount=2)

time.sleep(15)

try:

#this gets task id of task that we started now

#taskArns has lists of all tasks in given service ,,here "Final" ..

#since we dont which task to take coz there is already a task running.. we chck for previous task id and decide which i current task id.

# To chck we take 2nd task as specified in ['taskArns'][1] n chck if this task for 9am ..

x=ecs.list\_tasks(serviceName="Final")['taskArns'][1].split("/")[1]

#see if this x isnt previous task's id

response = table.get\_item(Key={'time': '9am'})

item = response['Item']

#task\_id is previous task id .. i.e 9am task

task\_id=item['task']

#if they are equal we take ['taskArns'][0] and not ['taskArns'][1]

if(task\_id==x):

x=ecs.list\_tasks(serviceName="Final")['taskArns'][0].split("/")[1]

except:

#try the same thing after sometime if it didnt work initially

print ("in except")

time.sleep(15)

x=ecs.list\_tasks(serviceName="Final")['taskArns'][1].split("/")[1]

#see if this x isnt previous task's id

response = table.get\_item(Key={'time': '9am'})

item = response['Item']

task\_id=item['task']

if(task\_id==x):

x=ecs.list\_tasks(serviceName="Final")['taskArns'][0].split("/")[1]

print ("instance\_id= i",ins,"Task id = ", x)

#update the table with task id

#x is taskid

table.update\_item( Key={'time':'11am'},UpdateExpression='SET task = :val1',ExpressionAttributeValues={':val1': x})

#update state to "on"

table.update\_item( Key={'time':'11am'},UpdateExpression='SET stat = :val1',ExpressionAttributeValues={':val1': 'on'})

def update\_dec(id):

#"dynamodb" is dynamodb's object

dynamodb = boto3.resource('dynamodb')

#table points to ECSOrchestration table

table = dynamodb.Table('ECSOrchestration')

#ecs object

ecs = boto3.client('ecs')

#id=1 implies 1st instance

if(id==1):

#get task\_id for 9am since task started then.

response = table.get\_item(Key={'time': '9am'})

#item is the row

item = response['Item']

#task\_id is taskid

task\_id=item['task']

#ins is instane id

ins=item['ins\_id']

#stop the task

ecs.stop\_task(task=task\_id)

#dec service count from 2 to 1

print ("task\_id " , task\_id)

ecs.update\_service(cluster='default',service="Final",desiredCount=1)

#update task\_id =0 in table and set stat to off

print ("hi")

table.update\_item( Key={'time':'9am'},UpdateExpression='SET task = :val1',ExpressionAttributeValues={':val1': '0'})

table.update\_item( Key={'time':'9am'},UpdateExpression='SET stat = :val1',ExpressionAttributeValues={':val1': 'off'})

#stop the instance with instance id ins

stop(ins)

elif (id==2):

#get task\_id for 11am since task started then.

response = table.get\_item(Key={'time': '11am'})

#item is the row

item = response['Item']

#task\_id is taskid

task\_id=item['task']

#ins is instane id

ins=item['ins\_id']

print ("task\_id " , task\_id)

#stop the task

ecs.stop\_task(task=task\_id)

#dec service count from 1 to 0 since its last instance

ecs.update\_service(service="Final",desiredCount=0)

print ("hi")

#update task\_id =0 in table and set stat to off

table.update\_item( Key={'time':'11am'},UpdateExpression='SET task = :val1',ExpressionAttributeValues={':val1': '0'})

table.update\_item( Key={'time':'11am'},UpdateExpression='SET stat = :val1',ExpressionAttributeValues={':val1': 'off'})

#stop the instance with instance id ins

stop(ins)

#main function

def lambda\_handler(event, context):

#get\_current\_time

now=datetime.now()

#converts time to gmt+5:30 i.e indian time

from\_zone=tz.gettz('UTC')

to\_zone = tz.gettz('GMT+5:30')

now = now.replace(tzinfo=from\_zone)

central = now.astimezone(to\_zone)

#get hour and minute

hour= central.hour

minute= central.minute

print (hour,minute)

#update\_inc for starting instance and service on it

#update\_dec for stopping service and shutting down instance

#accordingly call these functions with id=1 or id=2 for 1st and 2nd instance respectively

#instance-id --> id --> time

#'i-48d7b9cb' --> 1 --> 9 to 6

#'i-1d9b6f80' --> 2 --> 11 to 8

#chck for current hour and call functions

if(hour==9):

update\_inc(1)

elif (hour==11):

update\_inc(2)

elif (hour==18):

update\_dec(1)

elif(hour==20):

update\_dec(2)

#TEST

#uncomment one at a time

#print ("starting 1...")

#update\_inc(1)

#time.sleep(30)

#print ("starting 2..")

#update\_inc(2)

## time.sleep(60)

#print ("stopping 1..")

#update\_dec(1)

#time.sleep(60)

#print ("stopping 2..")

#update\_dec(2)

#-------------Not used for now---------------

"""

def describe():

ec2=boto3.resource('ec2')

ecs = boto3.client('ecs')

#decribe\_cluster

print(ecs.describe\_clusters(clusters=['default',]))

def run\_task():

#to\_run\_task given task\_definition(here Aviate:9)

ec2=boto3.resource('ec2')

ecs = boto3.client('ecs')

x=ecs.run\_task(cluster='default',taskDefinition='Aviate:9')

task\_id= x['tasks'][0]['taskArn'].split('/')[1]

print(ecs.list\_tasks(cluster='default',containerInstance='47465365-edcf-4932-926e-77ba3474d2a6'))

print (task\_id)

return task\_id

def stop\_task(task\_id):

ecs.stop\_task(cluster='default',task=task\_id)

def create():

#create\_service

ec2=boto3.resource('ec2')

ecs = boto3.client('ecs')

x=ecs.create\_service(cluster='default',serviceName='service1',taskDefinition='Aviate:9',loadBalancers=[{'loadBalancerName':'ecs-service-elb','containerName':'aviate','containerPort':8080},],role='ecsServiceRole',desiredCount=1)

print (x)

"""