Cloud Computing – Project 3: Exploring a multi-cloud application

[redacted] [redacted]

Part 1: preparation

1.1:
Code is revised at "**index.html**" a template file to display expected format of greetings.

The <blockquote> tag content within the for loop of greetings is modified as **"Greeting id {{greeting.gid}} saved at {{greeting.date.strftime('%H:%M:%S')}} says: {{greeting.content}}** "

URL: https://[redacted]-guestbook-app.appspot.com

1.2: Code snippet for 1 and 2

1.2.1

```
def create_table(table_name):
    """
    create a table and return the table object
    :param table_name: name of the table
    :return: dynamo db table instance
    """
    dynamodb_resource = boto3.client('dynamodb', aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,region_name="us-east-1")
    # to do
    # check the sample code https://docs.aws.amazon.com/amazondynamodb/latest/develope
rguide/GettingStarted.Python.01.html
    # create the greetings table with attributes (gid, date, content).(1) create a tab
le "greetings" that has the three fields "gid", "date", and "content".
    # The method "create_table" is (2) Make sure you can read/insert/delete greetings.
 (3) Post your code snippets for (1) and (2) here.
    try:
        table=dynamodb_resource.create_table(TableName=table_name,
                           KeySchema=[
                               {
                                   'AttributeName':'gid',
                                   'KeyType':'HASH'
                                }
                               ],
                           AttributeDefinitions=[
                               {
                                   'AttributeName': 'gid',
                                   'AttributeType': 'N'
                               }
                           ],
                           ProvisionedThroughput={
                               'ReadCapacityUnits': 10,
                               'WriteCapacityUnits': 10
                           }
                           )
        table_status=dynamodb_resource.describe_table(TableName=table_name)['Table']['
TableStatus']
        #print("Table status:", table_status)
        while True:
            if table_status=='CREATING':
                time.sleep(10)
                table_status=dynamodb_resource.describe_table(TableName=table_name)['T
able']['TableStatus']
            else:
                break
    except dynamodb_resource.exceptions.ResourceInUseException as e:
        #print('Table already exists')
        pass

    # return the table object
    return get_table(table_name)



def get_table(table_name):
    """
    return the table object, when the table is already created
    :param table_name: name of the table
    :return: dynamo db table instance
    """
    dynamodb_resource = boto3.resource('dynamodb', aws_access_key_id=ACCESS_KEY,
    aws_secret_access_key=SECRET_KEY,region_name="us-east-1")
    table = None
    try:
        table = dynamodb_resource.Table(table_name)
    except:
        print("unable to fetch table", table_name)
    finally:
        return table
```

1.2.2

```python
def read_table_item(table, pk_name, pk_value):
    """
    table is the object returned by get_table
    Return item read by primary key.
    """
    response = table.get_item(Key={pk_name: pk_value})

    return response

def read_table(table,filter_name,filter_value,pe,ean):
    #response meta data is returned which is read by primary key
    pe = pe
    ean = ean
    #read all values from dynamodb"
    #since date is a reserved keyword ,using projection expressions such that Expressi
on attribute names doesnot throw reserve keyword exception
    if filter_name=="None":
        response = table.scan( \
            ProjectionExpression=pe, \
            ExpressionAttributeNames=ean
        )
    else:
        fe = Key(filter_name).eq(filter_value)

        # read values from dynamodb with filter
        response = table.scan( FilterExpression=fe,
                               ProjectionExpression=pe, \
                               ExpressionAttributeNames=ean
    )

    return response


def add_item(table, col_dict):
    """
    Add one item (row) to table. col_dict is a dictionary {col_name: value}.
    """
    response = table.put_item(Item=col_dict)

    return response

def update_item(table, col_dict,table_key):
    """
    Add one item (row) to table. col_dict is a dictionary {col_name: value}.
    """
    for items in col_dict:
        if items==table_key:
            pk_name=table_key
            pk_value=col_dict[items]

    response = table.update_item(Key={pk_name:pk_value},\
                                 UpdateExpression="set #date = :new_date, content= :co
ntent",
                                 ExpressionAttributeValues={':new_date':col_dict['date
'],
                                                            ':content':col_dict['conte
nt']} , \
                                 ExpressionAttributeNames={'#date':'date'},\
                                 ReturnValues='UPDATED_NEW')

    return response

def delete_item(table, pk_name, pk_value):
    """
    Delete an item (row) in table from its primary key.
    """
    response = table.delete_item(Key={pk_name: pk_value})

    return response
```

```
if __name__=="__main__":
        tableobj=create_table('Greetings')

        #date=time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime())
        greetingmsg=[{'gid':1,'date':time.strftime("%Y-%m-
%d %H:%M:%S", time.gmtime()),'content':'greeting 1'},\
                     {'gid': 2, 'date': time.strftime("%Y-%m-
%d %H:%M:%S", time.gmtime()), 'content': 'greeting 2'},\
                     {'gid': 3, 'date': time.strftime("%Y-%m-
%d %H:%M:%S", time.gmtime()), 'content': 'greeting 3'}, \
                     {'gid': 4, 'date': time.strftime("%Y-%m-
%d %H:%M:%S", time.gmtime()), 'content': 'greeting 4'}]

        #adding msgs
        for msgs in greetingmsg:
            write_response=add_item(tableobj,msgs)
            print(write_response['ResponseMetadata']['HTTPStatusCode'])

        #read messages with keys
        read_id=[1,3,4]
        for gid in read_id:
            read_reponse=read_table_item(tableobj,'gid',gid)
            print(read_reponse['Item'])
        #delete messages using key
        delete_id=[1,4]

        for gid in delete_id:
            delete_response=delete_item(tableobj,'gid',gid)
            print(delete_response)

        read_reponse = read_table_item(tableobj, 'gid', 3)
        print("rad_table_item",read_reponse['Item'])

        #update the values in dynamo db"
        update={'gid': 3, 'date': time.strftime("%Y-%m-
%d %H:%M:%S", time.gmtime()), 'content': 'greeting upated'}
        update_item(tableobj,update,'gid')

        pe = " gid,#date,content"
        ean = {"#date": "date"}
        #Read all contents from a table
        read_table_response=read_table(table=tableobj,filter_name="None",filter_value=
"None",pe=pe,ean=ean)
```

Part 2: A guestbook application working with multiple cloud storages

2.1:

Note: Deployed the microservice on aws by creating an EC2 instance and copied "dynamodb.py" and "microservices.py" file to ubuntu instance by ssh into the instance.

EC2 instance Endpoint URL:

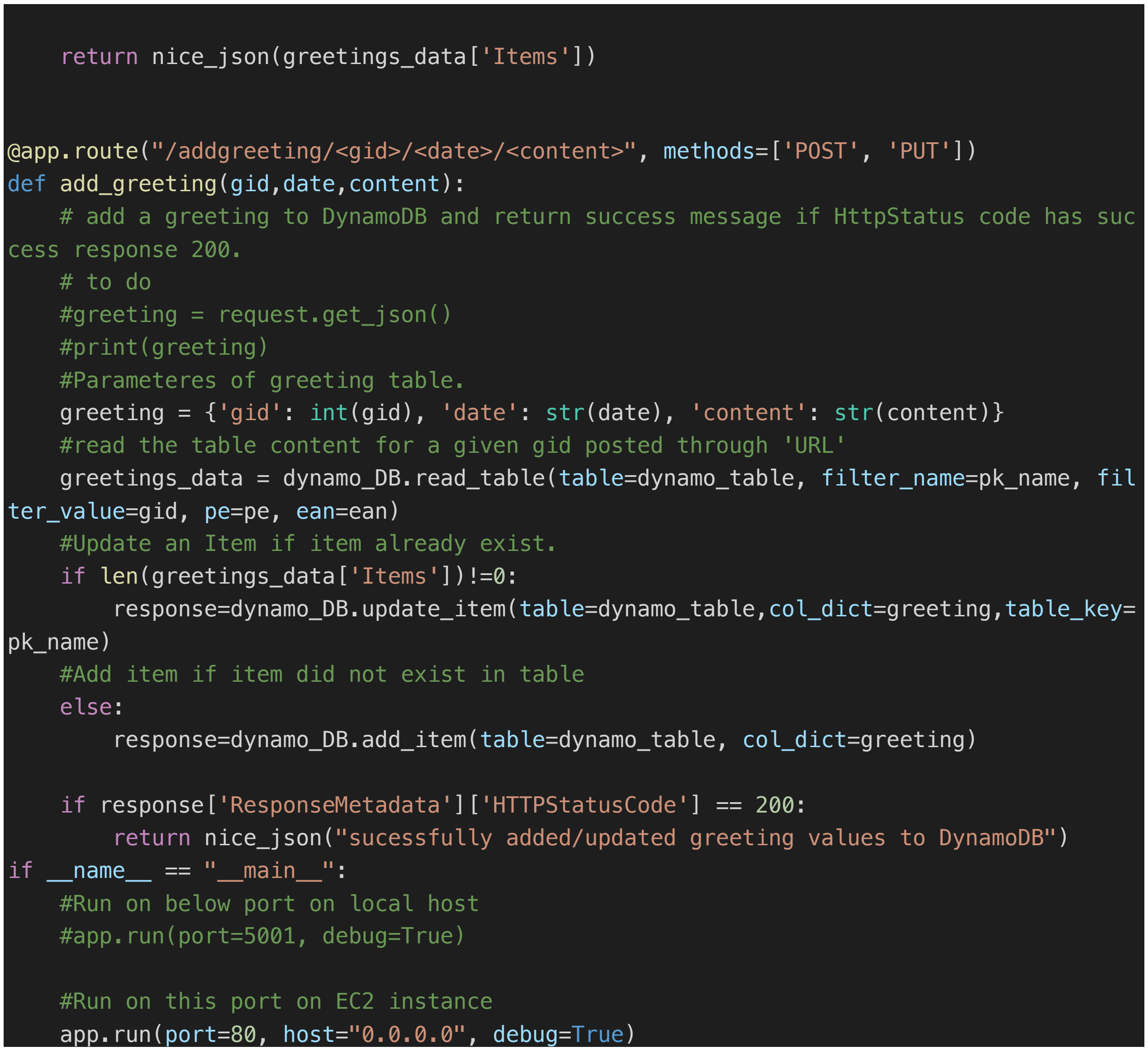
Retrieve Greeting:http://ec2-54-91-119-236.compute-1.amazonaws.com/greetings,

Store Greeting:http://ec2-54-91-119-236.compute-1.amazonaws.com/addgreeting/gid/date/content
Eg: http://ec2-54-91-119-236.compute-1.amazonaws.com/addgreeting/57845/2019-11-19 2004:18:40.077565/postrequestcontent)

Microservice code:

```
from flask import Flask,redirect,url_for,request
from werkzeug.exceptions import NotFound
from flask import make_response, request
import json
import os
import time
import decimal
import dynamo_DB # the code you finished for Part I


app = Flask(__name__)
```

```python
# code here to open the DynamoDB table. If the table is not there, create it
dynamo_table=dynamo_DB.create_table('Greetings')

#variables related to specific Greetings table in dynamo DB where pe=ProjectionExpression and ean=ExpressionAttributeNames
pe = " gid,#date,content"
ean = {"#date": "date"}
pk_name='gid'


def root_dir():
    """ Returns root director for this project """
    return os.path.dirname(os.path.realpath(__file__ + '/..'))

#helper class to convert dynamoDB items to json
class DecimalEncoder(json.JSONEncoder):
    def default(self, o):
        if isinstance(o, decimal.Decimal):
            if o % 1 > 0:
                return float(o)
            else:
                return int(o)
        return super(DecimalEncoder, self).default(o)

def nice_json(arg):
    response = make_response(json.dumps(arg,sort_keys = True, indent=4,cls=DecimalEncoder))
    response.headers['Content-type'] = "application/json"
    return response


@app.route("/", methods=['GET'])
def hello():
    return nice_json({
        "uri": "/",
        "subresource_uris": {
            "greetings": "/greetings",
            "add_greeting": "/addgreeting/<id>/<date>/<content>",
        }
    })

@app.route("/greetings", methods=['GET'])
def greetings():
    # return all greetings records in json format
    # to do
    greetings_data = dynamo_DB.read_table(table=dynamo_table,filter_name="None",filter_value="None",pe=pe,ean=ean)
```



```
    return nice_json(greetings_data['Items'])


@app.route("/addgreeting/<gid>/<date>/<content>", methods=['POST', 'PUT'])
def add_greeting(gid,date,content):
    # add a greeting to DynamoDB and return success message if HttpStatus code has suc
cess response 200.
    # to do
    #greeting = request.get_json()
    #print(greeting)
    #Parameteres of greeting table.
    greeting = {'gid': int(gid), 'date': str(date), 'content': str(content)}
    #read the table content for a given gid posted through 'URL'
    greetings_data = dynamo_DB.read_table(table=dynamo_table, filter_name=pk_name, fil
ter_value=gid, pe=pe, ean=ean)
    #Update an Item if item already exist.
    if len(greetings_data['Items'])!=0:
        response=dynamo_DB.update_item(table=dynamo_table,col_dict=greeting,table_key=
pk_name)
    #Add item if item did not exist in table
    else:
        response=dynamo_DB.add_item(table=dynamo_table, col_dict=greeting)

    if response['ResponseMetadata']['HTTPStatusCode'] == 200:
        return nice_json("sucessfully added/updated greeting values to DynamoDB")
if __name__ == "__main__":
    #Run on below port on local host
    #app.run(port=5001, debug=True)

    #Run on this port on EC2 instance
    app.run(port=80, host="0.0.0.0", debug=True)
```

2.1 How to Test micro Service:

>Tested micro service by running the app in debug mode and observing the final outputs at respective End Points in browser and in postman.
Locally host :“http://127.0.0.1:5001/”,”http://127.0.0.1:5001/greetings”
EC2 instance: “http://ec2-54-91-119-236.compute-1.amazonaws.com/greetings,”

> Post request (Localhost:http://127.0.0.1:5001/addgreeting/<id>/<date>/<content>,
EC2 URL:http://ec2-54-91-119-236.compute-1.amazonaws.com/addgreeting/gid/date/content)
response is validated to be success when “HTTP Status” code is 200 with response body message as “successfully added/updated greeting values to DynamoDB” as expected. Since browser doesn’t support “POST” method URL content, so verified the status code of each API service URL

using postman. Finally Navigated to "http://127.0.0.1:5001/greetings" or "EC2 instance greetings end point" to check if the content passed through post request is added into the table and displayed in browser with get request.

2.2:

>GAE **main program needs to be changed**. In guestbook.py change is made in "Mainpages" class where get () method is redefined using urlftech as get () method contains code logic to retrieve values from GAE datastore and "Guestbook" class redefined with post () method as it contains logic to add content to the GAE datastore.

>Redefined get () method will have **urlfetch.fetch(url=url,method=urlfetch.GET**) where **url="http://ec2-54-91-119-236.compute-1.amazonaws.com/greetings"** to access microservices to retrieve greetings from dynamo DB

>Redefined post () method will have **urlfetch.fetch(url, method=urlfetch.POST)** where **url="http://ec2-54-91-119-236.compute 1.amazonaws.com/addgreeting/"+str(gid)+"/"+urllib.pathname2url(date)+"/"+urllib.pathname2url(content)** to store greetings to dynamo DB with given input values.

2.3:

Datamodel.py code snippet:

```
import abc
import random
import json
import urllib

from google.appengine.api import urlfetch
from google.appengine.ext import ndb

from greeting import Greeting

"""
command to execute:python2 /Users/vaishnaviv/CC_GCP/google-cloud-
sdk/platform/google_appengine/dev_appserver.py app.yaml
"""

DEFAULT_GUESTBOOK_NAME = 'mydefault-guestbook'

Dynamo_DB_name="Greetings"
#HOST="http://127.0.0.1:5001"
HOST = "http://ec2-54-91-119-236.compute-1.amazonaws.com"

def guestbook_key(guestbook_name=DEFAULT_GUESTBOOK_NAME):
```

```python
    """Constructs a Datastore key for a Guestbook entity.

    We use guestbook_name as the key.
    """
    return ndb.Key('Guestbook', guestbook_name)

# the base class
class GreetingModel:
    __metaclass__ = abc.ABCMeta

    @abc.abstractmethod
    def getGreetings(self):
        pass
    @abc.abstractmethod
    def addGreeting(self, gid, date, content):
        pass



class GAEGreeting(GreetingModel):
    def __init__(self, guestbook_name):
        # constructor, initialize anything you need
        # Initialize the guestbook_name which is specific to each type Datastore model
(GAE datastore ndb name or Dynamodb table name)
        self.guestbook_name=guestbook_name
        pass

    def getGreetings(self):
        # Fetch the greetings from GAE guestbook entry and return greetings and guestb
ook name
        greetings_query = Greeting.query(
        ancestor=guestbook_key(self.guestbook_name)).order(-Greeting.date)
        greetings = greetings_query.fetch(10)

        return greetings,self.guestbook_name

    def addGreeting(self, gid, date, content):
        # to do
        greeting = Greeting(parent=guestbook_key(self.guestbook_name))
        greeting.gid = gid
        greeting.content = content
        #greeting.date=date
        greeting.put()
        return greeting.date

class DynamoGreeting(GreetingModel):
    def __init__(self, guestbook_name):
        # to do
        self.guestbook_name=guestbook_name

    def getGreetings(self):
        # to do
            #fetch reponse from microservice and returns guetbookname and content as g
reeting items from dynamo DB

            #url="http://127.0.0.1:5001/greetings"
            #url="http://ec2-54-165-158-186.compute-1.amazonaws.com/greetings"
            url = HOST+"/greetings"
            get_response=urlfetch.fetch(url=url,method=urlfetch.GET)
            #print(get_response)
            if get_response.status_code==200:
                    greetings=json.loads(get_response.content)
            else:
                greetings="Error"+str(get_response.status_code)

            return greetings,self.guestbook_name

    def addGreeting(self, gid, date, content):
        # Content entered in guestbook form is passed into the post request and status
 of request is returned.

            #url="http://127.0.0.1:5001/addgreeting/"+str(gid)+"/"+urllib.pathname2url
(date)+"/"+urllib.pathname2url(content)
            #url="http://ec2-54-165-158-186.compute-
1.amazonaws.com/addgreeting/"+str(gid)+"/"+urllib.pathname2url(date)+"/"+urllib.pathna
me2url(content)
            url = HOST+"/addgreeting/"+str(gid)+"/"+urllib.pathname2url(date)+"/"+urll
ib.pathname2url(content)
            post_response=urlfetch.fetch(url, method="POST")
            if post_response.status_code==200:
                    greetings=json.loads(post_response.content)
                    #self.response.headers['Content-type'] = "application/json"
                    #self.response.out.write(dynamo_post_reponse.content)
            else:
                greetings="Error"+str(post_response.status_code)

            #here greetings is a  status of the request which is used for test purpose
.
            return greetings

class UnifiedGreeting(GreetingModel):
    def __init__(self, guestbook_name):
        # create both GAE and Dynamo Models
        # the UnifiedGreeting model will be used by the GAE main program
        # to do
        self.guestbook_name=guestbook_name
    def getGreetings(self):
        # pick one model to get all greetings
        # to do
        #return greeting items to GAE main program(or where function is called based o
n type of model(DynamoDb or GAE datamodel) the data is requested from.)
        # if guestbook_name is "Greetings" then dynamo DB will be picked
        if self.guestbook_name==Dynamo_DB_name:
            Dynamo_obj=DynamoGreeting(Dynamo_DB_name)
            greeting_data=Dynamo_obj.getGreetings()

        #if guestbook_name is  "mydefault-guestbook"  GAE DB data will be picked
        if self.guestbook_name==DEFAULT_GUESTBOOK_NAME:
            GAE_obj=GAEGreeting(DEFAULT_GUESTBOOK_NAME)
            greeting_data=GAE_obj.getGreetings()

        return greeting_data

    def addGreeting(self, gid, date, content):
        # append the new record to both models
        # Adds record to both Models any exceptions during these operations is handled
 in GAE "Guestbook.py"

        GAE_obj=GAEGreeting(DEFAULT_GUESTBOOK_NAME)
        gae_date=GAE_obj.addGreeting(gid,date,content)

        Dynamo_obj=DynamoGreeting(Dynamo_DB_name)
        greetings_response = Dynamo_obj.addGreeting(gid,str(gae_date),content)

        return greetings_response
```

Modifying GAE Main program:" Guestbook.py"

Changes are made in get () and post () methods in "MainPage class" and "Guestbook class" of "guestbook.py"
@ MainPage class get () method, Unified Greetings class (from datamodel.py) is instantiated with parameter "guestbookname" and call to getGreetings () method of Unified Greetings class is made. Based on Guestbook name passed during instantiation, greetings are fetched from GAE datastore or DynamoDB

>greetings returned from getGreetings () is stored in template values dictionary at guestbook.py and finally template(index.html) is rendered where all greetings are displayed in browser with 'Sign Guestbook form".

@Guestbook class post () method, use **request.get('content')** to fetch content posted used "Sign Guestbook form" which is passed to addGreetings(gid, date,content) method of Unified Greetings class.

>Once the request is successful, values posted will be stored in both GAE datastore and DynamoDB and we can see the same "Sign Guestbook form" page with new content greeting being added.

Note: gid is randomly generated and date is current time of the post.