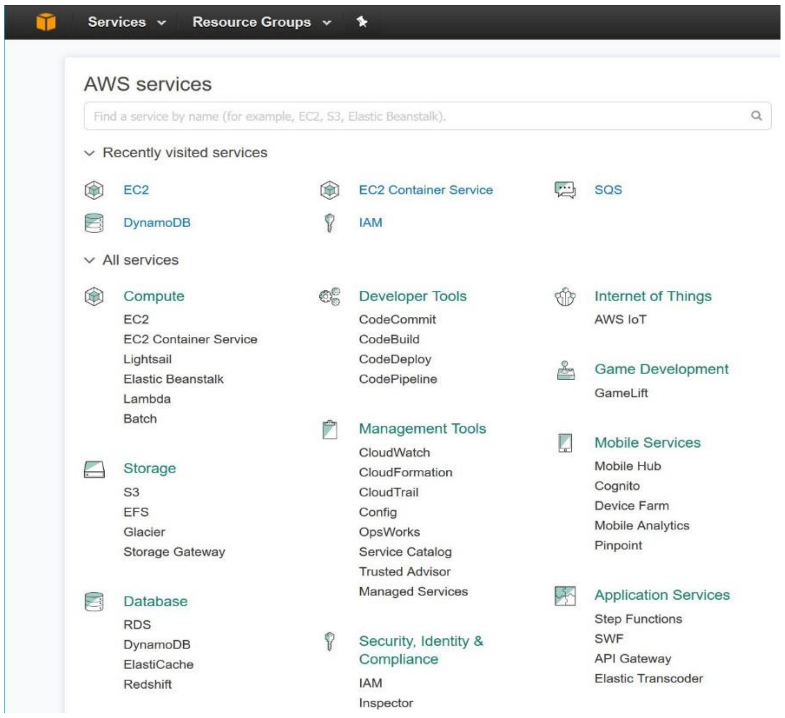
# Homework Supportive Materials

1. Please note that this document aims to shed lights on creating DynamoDB and uploading data to it. However, you are not required to follow the steps that are listed strictly.
2. You will have some issues with running this code that you will have to resolve on your own. As a guideline, you may use Jupyter Notebook to run the code below.

In general, you can access the cloud via two methods:

- Web Portals
  - Dashboard that allow you to see and manage your cloud resources
  - Simple and interactive for small number of tasks
  - E.g. AWS Portal can be accessed from https://console.aws.amazon.com/console/

- Software Development Kits (SDKs)
  - Libraries that give you the tools to manage cloud resources from a program or script.
  - Based on REST web service calls



If you do not already have boto3, the amazon python sdk installed, then uncomment and run the following line

```
In [21]:
#!pip install boto3
```

```
In [1]:
import boto3
```

## Create an s3 instance object

Follow the instructions on the IAM portal to get an access key and a secret key

```
In [2]:
s3 = boto3.resource('s3',
    aws_access_key_id='your access key',
    aws_secret_access_key='your secret key'
)
```

Next let's test this by creating our bucket "datacont-name" in the Oregon data center. The creation function will through an exception if the bucket already exists. (Note that this bucket name may not be acceptable so you will need to choose different bucket name)

```
In [3]:
try:
    s3.create_bucket(Bucket='datacont-name', CreateBucketConfiguration={
        'LocationConstraint': 'us-west-2'})
except Exception as e:
    print (e)
```

Now we will make this bucket publicly readable. We will also need to make each blob in the bucket publicly readable

```
In [4]:
bucket = s3.Bucket("datacont-name")
```

```
In [5]:
bucket.Acl().put(ACL='public-read')
```

```
Out[5]:
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '0',
   'date': 'Thu, 07 Jul 2016 18:37:44 GMT',
   'server': 'AmazonS3',
   'x-amz-id-2': 'RM2zILiBLYtOVnnuVsK0j/7YyEsZFdcGF5PnSQO21HxFGz42U88skmpBXuB
QsQ3/f/+E7tKPuXI=',
   'x-amz-request-id': '2D764B7DE7A58577'},
  'HTTPStatusCode': 200,
  'HostId': 'RM2zILiBLYtOVnnuVsK0j/7YyEsZFdcGF5PnSQO21HxFGz42U88skmpBXuBQsQ3/
f/+E7tKPuXI=',
  'RequestId': '2D764B7DE7A58577'}}
```

Now, let's try to upload a file into the bucket.

In [9]:
```
#upload a new object into the bucket
body = open('path-to-a-file\exp1', 'rb')
```

In [10]:
```
o = s3.Object('datacont-name', 'test').put(Body=body )
```

In [12]:
```
s3.Object('datacont-name', 'test').Acl().put(ACL='public-read')
```

Out[12]:
```
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '0',
   'date': 'Thu, 07 Jul 2016 18:38:33 GMT',
   'server': 'AmazonS3',
   'x-amz-id-2': 'rVO6eBJDldB19+sUQLfv/Zmaq7HBl+UBFhVLpW2AdHFNffUF9LP6koE4XKF
ZXVf5rt19JIG/zSs=',
   'x-amz-request-id': '839011F5955BA066'},
  'HTTPStatusCode': 200,
  'HostId': 'rVO6eBJDldB19+sUQLfv/Zmaq7HBl+UBFhVLpW2AdHFNffUF9LP6koE4XKFZXVf5
rt19JIG/zSs=',
  'RequestId': '839011F5955BA066'}}
```

Next we will create the DynamoDB table. Note that creating the resource does not create the table. The following try-block creates the table. We need to provide a Key schema. One element is hashed to produce a partition that stores a row while the second key is RowKey. The pair (PartitionKey, RowKey) is a unique identifier for the row in the table.

In [13]:
```
dyndb = boto3.resource('dynamodb',
    region_name='us-west-2',
    aws_access_key_id='your AWS access key received from IAM',
    aws_secret_access_key='your AWS secret key received from IAM'
 )
```

In [16]:
```
try:
    table = dyndb.create_table(
        TableName='DataTable',
        KeySchema=[
            {
                'AttributeName': 'PartitionKey',
                'KeyType': 'HASH'
            },
            {
                'AttributeName': 'RowKey',
                'KeyType': 'RANGE'
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'PartitionKey',
                'AttributeType': 'S'
            },
            {
                'AttributeName': 'RowKey',
                'AttributeType': 'S'
            },

        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 5,
            'WriteCapacityUnits': 5
        }
    )
except Exception as e:
    print (e)
    #if there is an exception, the table may already exist.   if so...
    table = dyndb.Table("DataTable")
```

In [17]:
```
#wait for the table to be created
table.meta.client.get_waiter('table_exists').wait(TableName='DataTable')
```

In [18]:
```
print(table.item_count)
```

```
0
```

```
In [19]:
import csv
```

## Reading the csv file, uploading the blobs and creating the table

This code assumes that each row of the csv file looks like: (experiment name, id-number, name-of-ith-file, date, comments).

Note that these column names are different from the column names that you were provided in the Master CSV. We create a URL based on where we know the blobs are stored and we will append that URL to the tuple above and insert the list into the table.

```
In [21]:
with open('c:\users\farag\documents\experiments.csv', 'rb') as csvfile:
    csvf = csv.reader(csvfile, delimiter=',', quotechar='|')
    for item in csvf:
        print item
        body = open('c:\users\farag\documents\datafiles\\'+item[3], 'rb')
        s3.Object('datacont-name', item[3]).put(Body=body )
        md = s3.Object('datacont-name', item[3]).Acl().put(ACL='public-read')

        url = " https://s3-us-west-2.amazonaws.com/datacont-name/"+item[3]
        metadata_item = {'PartitionKey': item[0], 'RowKey': item[1],
                 'description' : item[4], 'date' : item[2], 'url':url}
        try:
            table.put_item(Item=metadata_item)
        except:
            print "item may already be there or another failure"
```

```
['experiment1', '1', '3/15/2002', 'exp1', 'this is the comment']
['experiment1', '2', '3/15/2002', 'exp2', 'this is the comment2']
['experiment2', '3', '3/16/2002', 'exp3', 'this is the comment3']
['experiment3', '4', '3/16/2002', 'exp4', 'this is the comment233']
```

Now let's search for an item'

In [23]:
```
response = table.get_item(
    Key={
        'PartitionKey': 'experiment3',
        'RowKey': '4'
    }
)
item = response['Item']
print(item)
```

```
{u'url': u' https://s3-us-west-2.amazonaws.com/datacont-name/exp4', u'date':
u'3/16/2002', u'PartitionKey': u'experiment3', u'description': u'this is the
comment233', u'RowKey': u'4'}
```

In [24]:
```
response
```

Out[24]:
```
{u'Item': {u'PartitionKey': u'experiment3',
  u'RowKey': u'4',
  u'date': u'3/16/2002',
  u'description': u'this is the comment233',
  u'url': u' https://s3-us-west-2.amazonaws.com/datacont-name/exp4'},
 'ResponseMetadata': {'HTTPHeaders': {'content-length': '198',
   'content-type': 'application/x-amz-json-1.0',
   'date': 'Thu, 07 Jul 2016 18:55:49 GMT',
   'x-amz-crc32': '3835589557',
   'x-amzn-requestid': 'LBV3KQ5GJTK9I2A85EB4MJ2ENVVV4KQNSO5AEMVJF66Q9ASUAAJG'
},
  'HTTPStatusCode': 200,
  'RequestId': 'LBV3KQ5GJTK9I2A85EB4MJ2ENVVV4KQNSO5AEMVJF66Q9ASUAAJG'}}
```