Assignment#1 – AWS Storage, AWS RDS and AWS DynamoDB

**Due Date:** Midnight of October 13 (Saturday)

**Purpose:** The purpose of this assignment is to help you:

1. Understand AWS S3, AWS RDS and AWS DynamoDB
2. Become familiar with IAM
3. Able to set up development environment

**Instructions**: Be sure to read the following general instructions carefully:

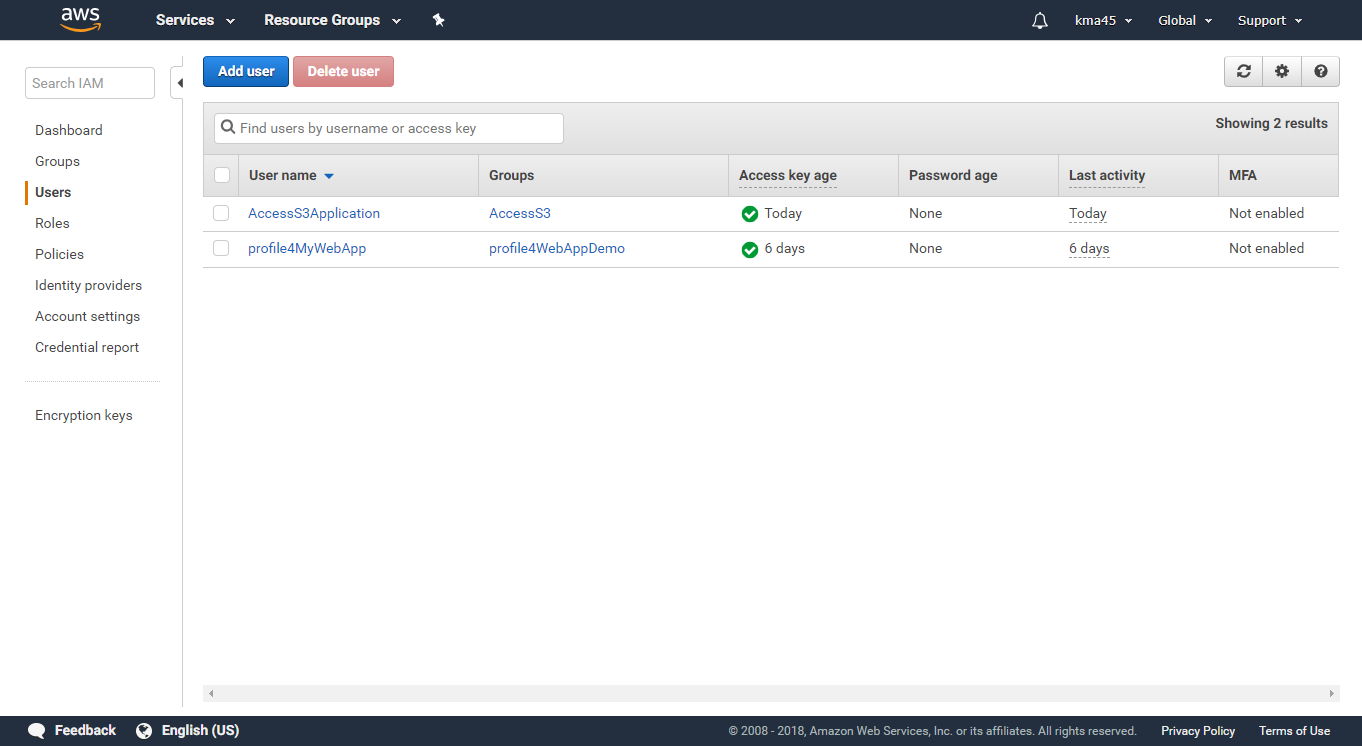
This assignment should be completed individually by all students. You need to demonstrate your solution during lab session and submit your solution **through the dropbox**. Your submission must be a word document or a pdf document which contains necessary screenshot and be named following the pattern of **studentID(yourlastname)\_ASSnumber.pdf**. e.g., 300123456(**smith)\_ASS#1**.pdf

**Question 1 [10 marks]**

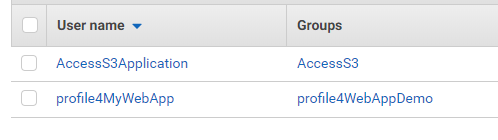
* 1. [6 marks] Use example to explain elements of IAM, such as *user, group, role, access key, secret key* and *policy*.

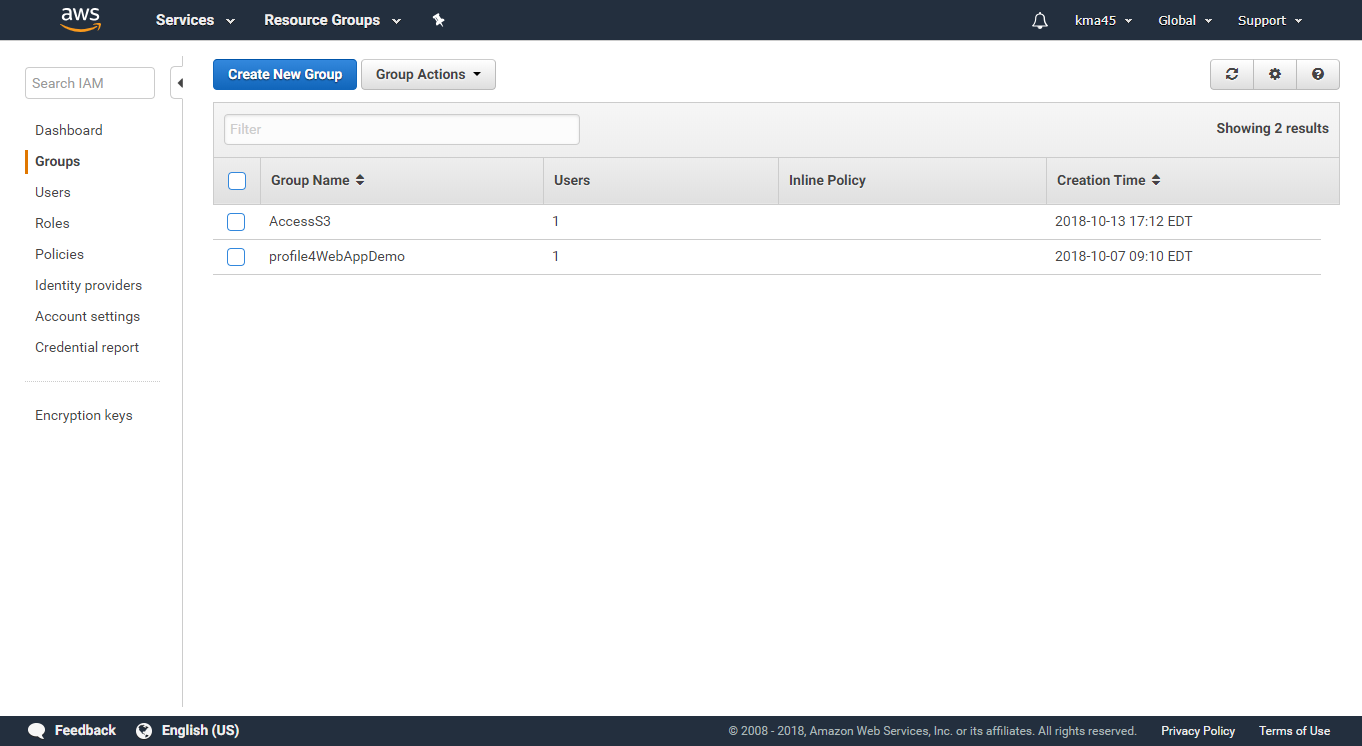
**IAM** stands for **Identity and access Management** and it is a web service provided by AWS for securely controlling access to the other services offered by AWS. It allows for the management of users, security credentials such as access keys, and permissions that control which AWS resources users and applications have the ability access. There are some key concepts one should know about before working with IAM. They are as follows:

* *User*
  + A person or an application that requires access to AWS resources to perform various tasks
  + E.g. I have created two users under my IAM service on AWS in order to access my S3 and my other AWS resources.



* *Group*
  + A collection of IAM users that can be given certain permissions through policies
  + E.g. the two users I created are assigned to different groups. However, each group can have multiple users within it as well.



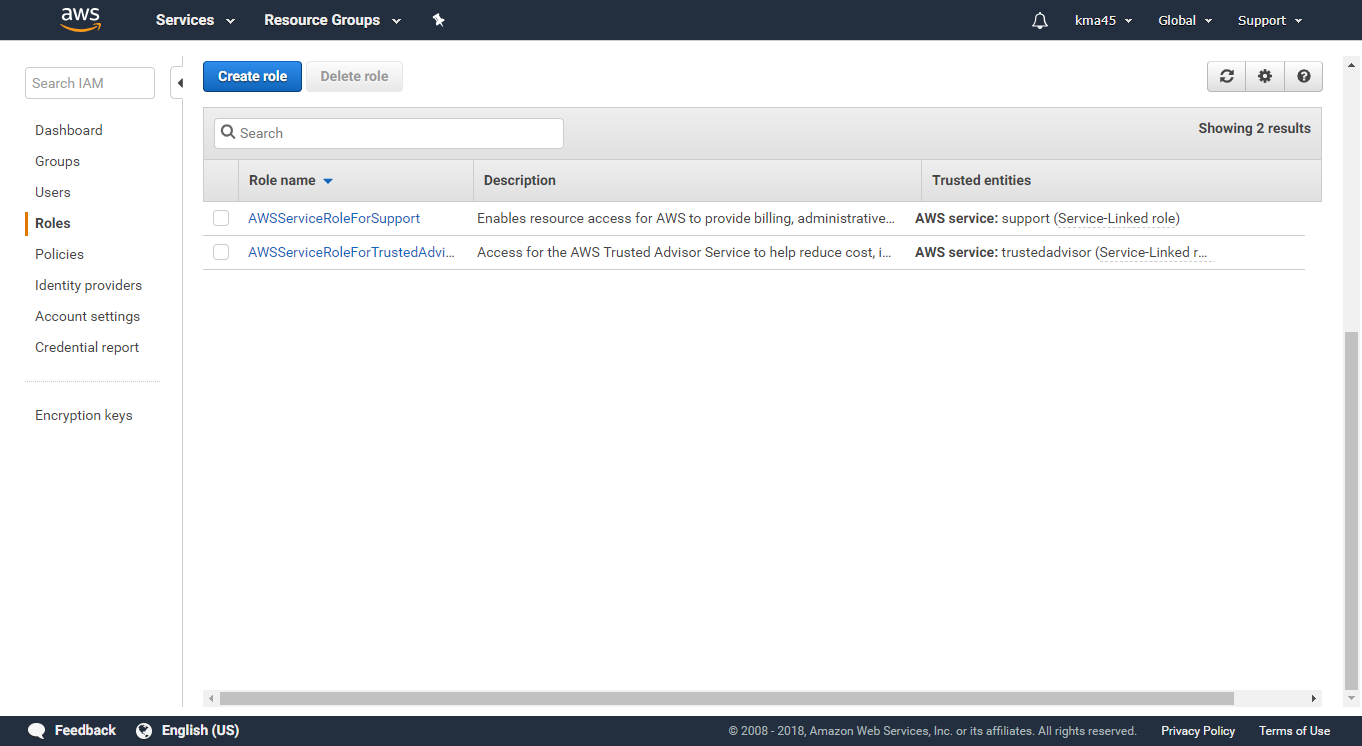


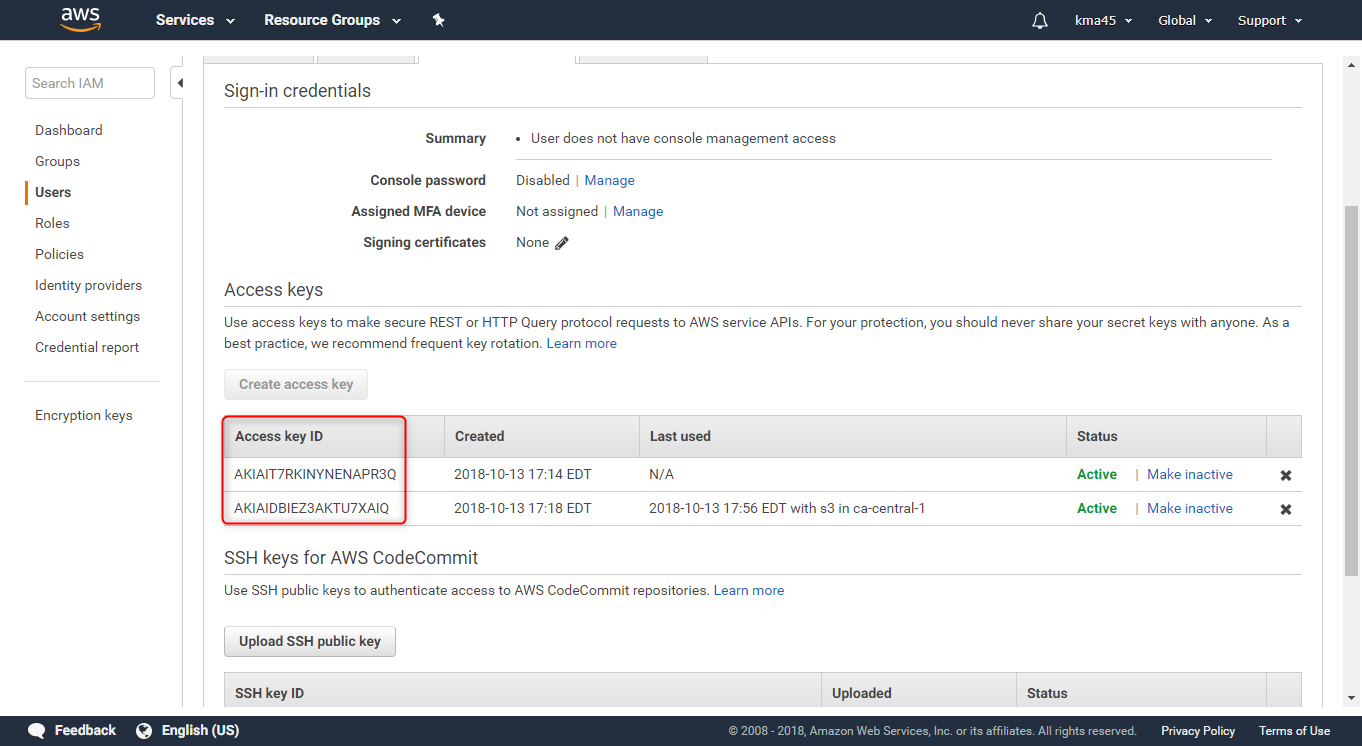
* + In the AccessS3 group, I gave it full access (CRUD) to my S3 instances by assigning the policy **AmazonS3FullAccess** to the group

A screenshot of a computer

Description generated with very high confidence

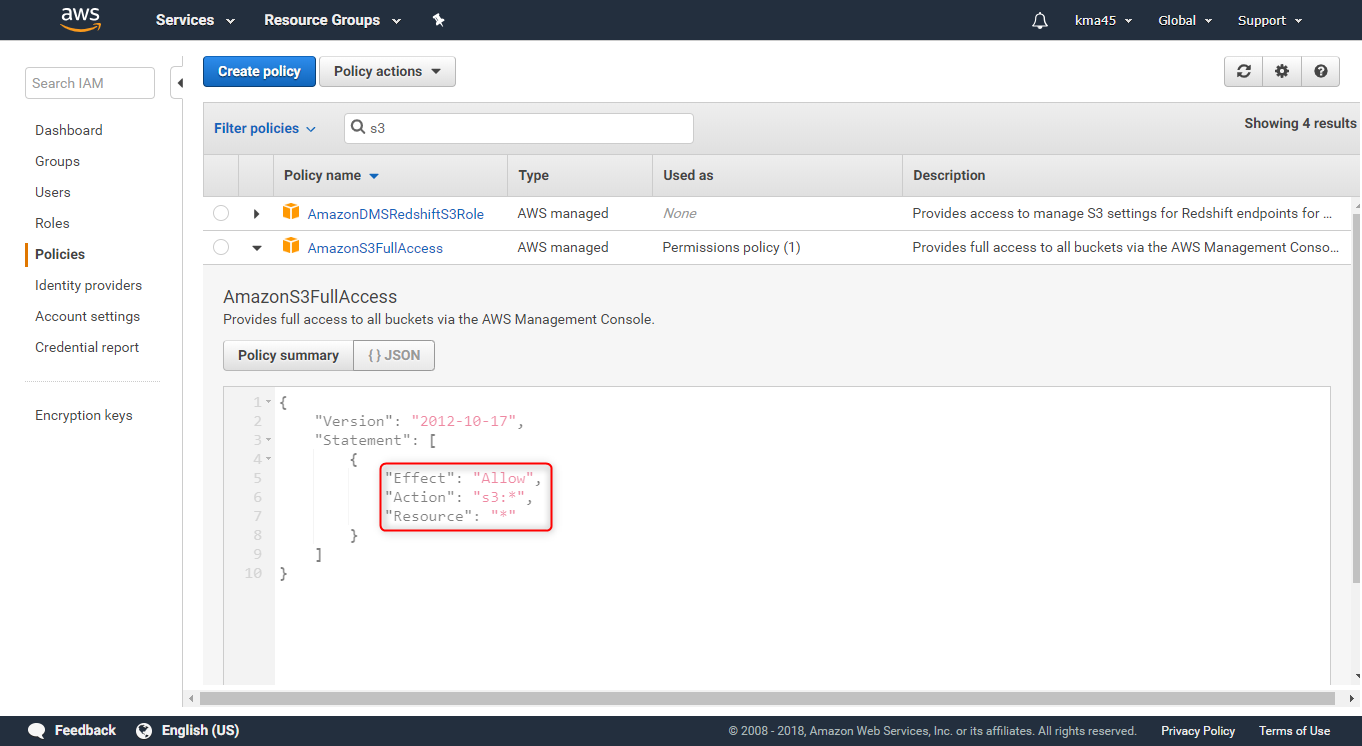
* *Role*
  + An IAM entity that is made up of one or more IAM policies which define resource permissions. These roles may grant permissions to entities that you trust such as:
    1. IAM user in another account
    2. Application code running on an EC2 instance that needs to perform actions on AWS resources
    3. An AWS service that needs to act on resources in your account to provide its features
    4. Users from a corporate directory who use identity federation with SAML
  + Also, IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.
  + E.g. There are some default roles created for **AWS service** to perform actions on AWS resources



* *Access Key*
  + A 20-character alphanumeric key that acts as a user ID
  + E.g. When we select a user, we can see the access keys for that user or create new ones.
  + 
  + It is used to identify the user when using the AWS user.
  + A screenshot of a cell phone

    Description generated with very high confidence
* *Secret Key*
  + A 40-character alphanumeric key that acts as a password or secret key
  + E.g. A secret key is generated when we create a secret key.
  + A screenshot of a computer screen

    Description generated with very high confidence
  + Instead of using the 40-char key as the password for accessing a AWS user access key, we can also use a .csv file in its place
  + A screenshot of a cell phone

    Description generated with very high confidence
* *Policy*
  + A document written in JSON format, as per the IAM policy standards, that states one or more permissions
  + E.g. The **AmazonS3FullAccess** policy gives all permissions for AWS S3 Resources in the following JSON file
  + 
  1. [4 marks] Finish the example ***Wk2\_Access\_S3\_Programmatically***, make sure it can upload object(s) to your S3.

Please see the completed example submitted along with this word document:



Please see the screenshots detailing the example and its output in the PowerPoint submitted with this word document:



**Question2 [10 marks]**

* 1. Create a MySQL database instance, create a table called **Student** to hold all information about student such as ID, first name, last name, email address, contact number; insert your own information into table Student [4 marks]

Please see the screenshots detailing the process I took to create a MySQL database instance and the Student table in the PowerPoint submitted with this word document:



* 1. Explain the partition key and sort key of AWS DynamoDB [2 marks]

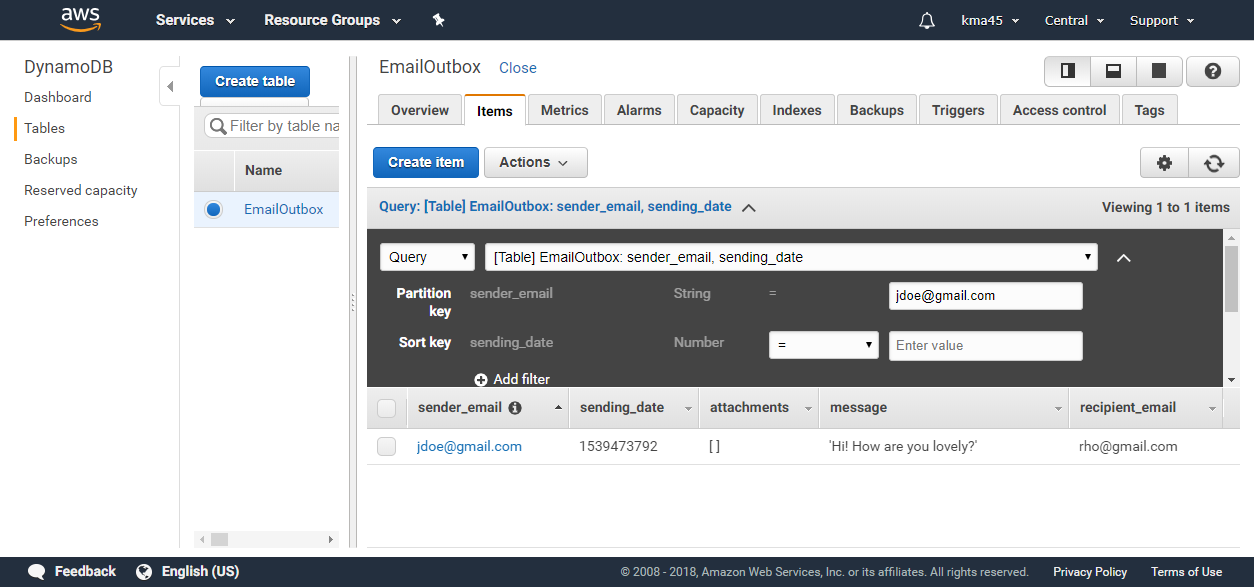
DynamoDB is a schema-less database that only requires a table name and primary key. The table’s primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

* *Partition key*
  + This is a *simple primary key*, composed of one attribute known as the partition key. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.
* *Sort key*
  + In DynamoDB, we construct *composite primary keys* by composing it with two attributes. The first attribute is the *partition key* which we mentioned above. The second attribute is the *sort key*. The sort key helps to sort the data within each partition and uniquely identify items within the table.
  1. Create a DynamoDB table to record emails for a mail server, the table should include sending date, sender, recipient, subject, message, etc. Explain the rationale of your design [4 marks]

Please see the screenshots detailing the process I took to create a DynamoDB table and insert item into the table in the PowerPoint submitted with this word document:



Shown below is the design of my table:



I chose to use a composite primary key for this table. I chose to have **sender\_email** as the partition key and then have **sending\_date** as the sort key.

This is because since we want to track emails sent (i.e. acting as a mail server outbox) we would want to be able to partition the data by the sender. Thus, I chose the partition key to be the sender\_email, which would serve as a simple primary key. I chose the sort key to be sending\_date. This is because although we can identify emails sent by the sender\_email field, there would be many emails sent by the sender. Thus, we needed a way to uniquely identify each email with the timestamp or sending\_date.