L3 Exercise 2 - IaC - Solution

June 7, 2023

1 Exercise 2: Creating Redshift Cluster using the AWS python SDK

1.1 An example of Infrastructure-as-code

```
In [1]: import pandas as pd
        import boto3
        import json
```

1.2 STEP 0: (Prerequisite) Save the AWS Access key

1.2.1 1. Create a new IAM user

IAM service is a global service, meaning newly created IAM users are not restricted to a specific region by default. - Go to AWS IAM service and click on the "**Add user**" button to create a new IAM user in your AWS account. - Choose a name of your choice. - Select "*Programmatic access*" as the access type. Click Next. - Choose the *Attach existing policies directly* tab, and select the "**AdministratorAccess**". Click Next. - Skip adding any tags. Click Next. - Review and create the user. It will show you a pair of access key ID and secret. - Take note of the pair of access key ID and secret. This pair is collectively known as **Access key**.

Snapshot of a pair of an Access key

1.2.2 2. Save the access key and secret

Edit the file `dwh.cfg` in the same folder as this notebook and save the access key and secret against the following variables:

```
KEY= <YOUR_AWS_KEY>
SECRET= <YOUR_AWS_SECRET>
```

For example:

```
KEY=6JW3ATLQ34PH3AKI
SECRET=wnoBHA+qUBFgwCRHJqgqrLUOi
```

1.2.3 3. Troubleshoot

If your keys are not working, such as getting an `InvalidAccessKeyId` error, then you cannot retrieve them again. You have either of the following two options:

1. **Option 1 - Create a new pair of access keys for the existing user**

- Go to the IAM dashboard and view the details of the existing (Admin) user.
- Select on the **Security credentials** tab, and click the **Create access key** button. It will generate a new pair of access key ID and secret.
- Save the new access key ID and secret in your `dwh.cfg` file

Snapshot of creating a new Access keys for the existing user

2. **Option 2 - Create a new IAM user with Admin access** - Refer to the instructions at the top.

2 Load DWH Params from a file

```
In [2]: import configparser
        config = configparser.ConfigParser()
        config.read_file(open('dwh.cfg'))

        KEY                    = config.get('AWS','KEY')
        SECRET                 = config.get('AWS','SECRET')

        DWH_CLUSTER_TYPE       = config.get("DWH","DWH_CLUSTER_TYPE")
        DWH_NUM_NODES          = config.get("DWH","DWH_NUM_NODES")
        DWH_NODE_TYPE          = config.get("DWH","DWH_NODE_TYPE")

        DWH_CLUSTER_IDENTIFIER = config.get("DWH","DWH_CLUSTER_IDENTIFIER")
        DWH_DB                 = config.get("DWH","DWH_DB")
        DWH_DB_USER            = config.get("DWH","DWH_DB_USER")
        DWH_DB_PASSWORD        = config.get("DWH","DWH_DB_PASSWORD")
        DWH_PORT               = config.get("DWH","DWH_PORT")

        DWH_IAM_ROLE_NAME      = config.get("DWH", "DWH_IAM_ROLE_NAME")

        (DWH_DB_USER, DWH_DB_PASSWORD, DWH_DB)

        pd.DataFrame({"Param":
                          ["DWH_CLUSTER_TYPE", "DWH_NUM_NODES", "DWH_NODE_TYPE", "DWH_CLUSTER_ID
                      "Value":
                          [DWH_CLUSTER_TYPE, DWH_NUM_NODES, DWH_NODE_TYPE, DWH_CLUSTER_IDENTIFIE
                     })
```

Out[2]:

	Param	Value
0	DWH_CLUSTER_TYPE	multi-node
1	DWH_NUM_NODES	4
2	DWH_NODE_TYPE	dc2.large
3	DWH_CLUSTER_IDENTIFIER	dwhCluster
4	DWH_DB	dwh
5	DWH_DB_USER	dwhuser

```
6          DWH_DB_PASSWORD    Passw0rd
7                DWH_PORT        5439
8       DWH_IAM_ROLE_NAME     dwhRole
```

3 Create clients for IAM, EC2, S3 and Redshift

Note: We are creating these resources in the the **us-west-2** region. Choose the same region in the your AWS web console to the see these resources.

```
In [3]: import boto3

        ec2 = boto3.resource('ec2',
                             region_name="us-west-2",
                             aws_access_key_id=KEY,
                             aws_secret_access_key=SECRET
                           )

        s3 = boto3.resource('s3',
                             region_name="us-west-2",
                             aws_access_key_id=KEY,
                             aws_secret_access_key=SECRET
                           )

        iam = boto3.client('iam',aws_access_key_id=KEY,
                             aws_secret_access_key=SECRET,
                             region_name='us-west-2'
                          )

        redshift = boto3.client('redshift',
                             region_name="us-west-2",
                             aws_access_key_id=KEY,
                             aws_secret_access_key=SECRET
                             )
```

4 Check out the sample data sources on S3

```
In [4]: sampleDbBucket =  s3.Bucket("awssampledbuswest2")
        for obj in sampleDbBucket.objects.filter(Prefix="ssbgz"):
            print(obj)
        # for obj in sampleDbBucket.objects.all():
        #     print(obj)
```

```
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/customer0002_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/dwdate.tbl.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0000_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0001_part_00.gz')
```

```
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0002_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0003_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0004_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0005_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0006_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/lineorder0007_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/part0000_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/part0001_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/part0002_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/part0003_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/supplier.tbl_0000_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/supplier0001_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/supplier0002_part_00.gz')
s3.ObjectSummary(bucket_name='awssampledbuswest2', key='ssbgz/supplier0003_part_00.gz')
```

5 STEP 1: IAM ROLE

- Create an IAM Role that makes Redshift able to access S3 bucket (ReadOnly)

In [5]:
```
from botocore.exceptions import ClientError

#1.1 Create the role,
try:
    print("1.1 Creating a new IAM Role")
    dwhRole = iam.create_role(
        Path='/',
        RoleName=DWH_IAM_ROLE_NAME,
        Description = "Allows Redshift clusters to call AWS services on your behalf.",
        AssumeRolePolicyDocument=json.dumps(
            {'Statement': [{'Action': 'sts:AssumeRole',
               'Effect': 'Allow',
               'Principal': {'Service': 'redshift.amazonaws.com'}}],
             'Version': '2012-10-17'})
    )
except Exception as e:
    print(e)


print("1.2 Attaching Policy")

iam.attach_role_policy(RoleName=DWH_IAM_ROLE_NAME,
                       PolicyArn="arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
                      )['ResponseMetadata']['HTTPStatusCode']

print("1.3 Get the IAM role ARN")
roleArn = iam.get_role(RoleName=DWH_IAM_ROLE_NAME)['Role']['Arn']
```

```
        print(roleArn)
```

```
1.1 Creating a new IAM Role
An error occurred (EntityAlreadyExists) when calling the CreateRole operation: Role with name dw
1.2 Attaching Policy
1.3 Get the IAM role ARN
arn:aws:iam::918744264023:role/dwhRole
```

6 STEP 2: Redshift Cluster

- Create a RedShift Cluster
- For complete arguments to `create_cluster`, see docs

```
In [6]: try:
            response = redshift.create_cluster(
                #HW
                ClusterType=DWH_CLUSTER_TYPE,
                NodeType=DWH_NODE_TYPE,
                NumberOfNodes=int(DWH_NUM_NODES),

                #Identifiers & Credentials
                DBName=DWH_DB,
                ClusterIdentifier=DWH_CLUSTER_IDENTIFIER,
                MasterUsername=DWH_DB_USER,
                MasterUserPassword=DWH_DB_PASSWORD,

                #Roles (for s3 access)
                IamRoles=[roleArn]
            )
        except Exception as e:
            print(e)
```

6.1 2.1 *Describe* the cluster to see its status

- run this block several times until the cluster status becomes `Available`

```
In [9]: def prettyRedshiftProps(props):
            pd.set_option('display.max_colwidth', -1)
            keysToShow = ["ClusterIdentifier", "NodeType", "ClusterStatus", "MasterUsername", "D
            x = [(k, v) for k,v in props.items() if k in keysToShow]
            return pd.DataFrame(data=x, columns=["Key", "Value"])

        myClusterProps = redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER)['C
        prettyRedshiftProps(myClusterProps)
```

```
Out[9]:                  Key  \
        0  ClusterIdentifier
```

```
1   NodeType
2   ClusterStatus
3   MasterUsername
4   DBName
5   Endpoint
6   VpcId
7   NumberOfNodes

                                                                    Value
0   dwhcluster
1   dc2.large
2   available
3   dwhuser
4   dwh
5   {'Address': 'dwhcluster.c4uipqmqcj1l.us-west-2.redshift.amazonaws.com', 'Port': 5439}
6   vpc-0a4c2bfdb64dcd525
7   4
```

2.2 Take note of the cluster endpoint and role ARN

DO NOT RUN THIS unless the cluster status becomes "Available". Make ure you are checking your Amazon Redshift cluster in the **us-west-2** region.

```
In [10]: DWH_ENDPOINT = myClusterProps['Endpoint']['Address']
         DWH_ROLE_ARN = myClusterProps['IamRoles'][0]['IamRoleArn']
         print("DWH_ENDPOINT :: ", DWH_ENDPOINT)
         print("DWH_ROLE_ARN :: ", DWH_ROLE_ARN)
```

```
DWH_ENDPOINT ::  dwhcluster.c4uipqmqcj1l.us-west-2.redshift.amazonaws.com
DWH_ROLE_ARN ::  arn:aws:iam::918744264023:role/dwhRole
```

6.2 STEP 3: Open an incoming TCP port to access the cluster ednpoint

```
In [11]: try:
             vpc = ec2.Vpc(id=myClusterProps['VpcId'])
             defaultSg = list(vpc.security_groups.all())[0]
             print(defaultSg)
             defaultSg.authorize_ingress(
                 GroupName=defaultSg.group_name,
                 CidrIp='0.0.0.0/0',
                 IpProtocol='TCP',
                 FromPort=int(DWH_PORT),
                 ToPort=int(DWH_PORT)
             )
         except Exception as e:
             print(e)
```

```
ec2.SecurityGroup(id='sg-0b21a18c369a6a006')
An error occurred (InvalidPermission.Duplicate) when calling the AuthorizeSecurityGroupIngress o
```

7 STEP 4: Make sure you can connect to the cluster

```
In [12]: %load_ext sql

In [13]: conn_string="postgresql://{}:{}@{}:{}/{}".format(DWH_DB_USER, DWH_DB_PASSWORD, DWH_ENDP
         print(conn_string)
         %sql $conn_string

postgresql://dwhuser:Passw0rd@dwhcluster.c4uipqmqcj1l.us-west-2.redshift.amazonaws.com:5439/dwh


Out[13]: 'Connected: dwhuser@dwh'
```

8 STEP 5: Clean up your resources

DO NOT RUN THIS UNLESS YOU ARE SURE We will be using these resources in the next exercises

```
In [85]: #### CAREFUL!!
         #-- Uncomment & run to delete the created resources
         redshift.delete_cluster( ClusterIdentifier=DWH_CLUSTER_IDENTIFIER,  SkipFinalClusterSna
         #### CAREFUL!!

Out[85]: {'Cluster': {'AllowVersionUpgrade': True,
           'AutomatedSnapshotRetentionPeriod': 1,
           'AvailabilityZone': 'us-west-2b',
           'ClusterCreateTime': datetime.datetime(2019, 2, 16, 6, 21, 30, 630000, tzinfo=tzutc()
           'ClusterIdentifier': 'dwhcluster',
           'ClusterParameterGroups': [{'ParameterApplyStatus': 'in-sync',
             'ParameterGroupName': 'default.redshift-1.0'}],
           'ClusterSecurityGroups': [],
           'ClusterStatus': 'deleting',
           'ClusterSubnetGroupName': 'default',
           'ClusterVersion': '1.0',
           'DBName': 'dwh',
           'Encrypted': False,
           'Endpoint': {'Address': 'dwhcluster.csmamz5zxmle.us-west-2.redshift.amazonaws.com',
            'Port': 5439},
           'EnhancedVpcRouting': False,
           'IamRoles': [{'ApplyStatus': 'in-sync',
             'IamRoleArn': 'arn:aws:iam::988332130976:role/dwhRole'}],
           'MasterUsername': 'dwhuser',
           'NodeType': 'dc2.large',
           'NumberOfNodes': 4,
           'PendingModifiedValues': {},
           'PreferredMaintenanceWindow': 'fri:10:30-fri:11:00',
           'PubliclyAccessible': True,
           'Tags': [],
```

```
 'VpcId': 'vpc-54d40a2c',
 'VpcSecurityGroups': []},
'ResponseMetadata': {'HTTPHeaders': {'content-length': '2041',
  'content-type': 'text/xml',
  'date': 'Sat, 16 Feb 2019 07:13:32 GMT',
  'x-amzn-requestid': '5e58b2d8-31ba-11e9-b19b-0945d449b0a9'},
 'HTTPStatusCode': 200,
 'RequestId': '5e58b2d8-31ba-11e9-b19b-0945d449b0a9',
 'RetryAttempts': 0}}
```

- run this block several times until the cluster really deleted

In [86]:
```
myClusterProps = redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER)['
prettyRedshiftProps(myClusterProps)
```

Out[86]:

	Key	Valu
0	ClusterIdentifier	dwhcluster
1	NodeType	dc2.large
2	ClusterStatus	deleting
3	MasterUsername	dwhuser
4	DBName	dwh
5	Endpoint	{'Address': 'dwhcluster.csmamz5zxmle.us-west-2.redshift.amazonaws.com', 'Port': 5439
6	VpcId	vpc-54d40a2c
7	NumberOfNodes	4

In [87]:
```
#### CAREFUL!!
#-- Uncomment & run to delete the created resources
iam.detach_role_policy(RoleName=DWH_IAM_ROLE_NAME, PolicyArn="arn:aws:iam::aws:policy/A
iam.delete_role(RoleName=DWH_IAM_ROLE_NAME)
#### CAREFUL!!
```

Out[87]:
```
{'ResponseMetadata': {'HTTPHeaders': {'content-length': '200',
   'content-type': 'text/xml',
   'date': 'Sat, 16 Feb 2019 07:13:50 GMT',
   'x-amzn-requestid': '694f8d91-31ba-11e9-9438-d3ce9c613ef8'},
  'HTTPStatusCode': 200,
  'RequestId': '694f8d91-31ba-11e9-9438-d3ce9c613ef8',
  'RetryAttempts': 0}}
```

In [ ]: