L3 Exercise 2 - IaC

January 13, 2023

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

1.1 An example of Infrastructure-as-code

```
In [2]: import pandas as pd
        import boto3
        import json
```

2 STEP 0: Make sure you have an AWS secret and access key

- Create a new IAM user in your AWS account
- Give it `AdministratorAccess`, From `Attach existing policies directly` Tab
- Take note of the access key and secret
- Edit the file `dwh.cfg` in the same folder as this notebook and fill [AWS] KEY= YOUR_AWS_KEY SECRET= YOUR_AWS_SECRET

3 Load DWH Params from a file

```
In [3]: import configparser
        config = configparser.ConfigParser()
        config.read_file(open('dwh.cfg'))

        REGION_NAME            = config.get('default', 'REGION_NAME')
        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[3]:

	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

3.1 Create clients for EC2, S3, IAM, and Redshift

In [4]:
```
import boto3

ec2 = boto3.resource('ec2',
                     region_name=REGION_NAME,
                     aws_access_key_id=KEY,
                     aws_secret_access_key=SECRET)

s3 = boto3.resource('s3',
                    region_name=REGION_NAME,
                    aws_access_key_id=KEY,
                    aws_secret_access_key=SECRET)

iam = boto3.client('iam',
                   region_name=REGION_NAME,
                   aws_access_key_id=KEY,
                   aws_secret_access_key=SECRET)

redshift = boto3.client('redshift',
                        region_name=REGION_NAME,
                        aws_access_key_id=KEY,
                        aws_secret_access_key=SECRET)
```

3.2 Check out the sample data sources on S3

In [5]:
```
sampleDbBucket =  s3.Bucket("awssampledbuswest2")
```

```
        # TODO: Iterate over bucket objects starting with "ssbgz" and print
        # for obj in sampleDbBucket.objects.all():
        for obj in sampleDbBucket.objects.filter(Prefix="ssbgz"):
            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')
```

3.3 STEP 1: IAM ROLE

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

```
In [7]: try:
            iam.detach_role_policy(
                RoleName=DWH_IAM_ROLE_NAME,
                PolicyArn='arn:aws:iam::aws:policy/AmazonS3FullAccess'
            )
            iam.delete_role(
                RoleName=DWH_IAM_ROLE_NAME
            )
        except Exception as e:
            print(e)
```

```
In [8]: # TODO: Create the IAM role
        try:
            print('1.1 Creating a new IAM Role')
            dwhRole = iam.create_role(
                Path='/',
                RoleName=DWH_IAM_ROLE_NAME,
                AssumeRolePolicyDocument= json.dumps({
                            "Version": "2012-10-17",
```

```
                                "Statement": [
                                    {
                                        "Effect": "Allow",
                                        "Principal": {
                                            "Service": [
                                                "redshift.amazonaws.com"
                                            ]
                                        },
                                        "Action": [
                                            "sts:AssumeRole"
                                        ]
                                    }
                                ]
                            }),
                    Description='Allows redshift to assume this role and access s3'
                )
            except Exception as e:
                print(e)
```

```
1.1 Creating a new IAM Role
```

```
In [9]: # TODO: Attach Policy
            try:
                print('1.2 Attaching Policy')
                iam.attach_role_policy(
                        RoleName=DWH_IAM_ROLE_NAME,
                        PolicyArn='arn:aws:iam::aws:policy/AmazonS3FullAccess'
                    )
            except Exception as e:
                print(e)
```

```
1.2 Attaching Policy
```

```
In [10]: # TODO: Get and print the IAM role ARN
             try:
                 print('1.3 Get the IAM role ARN')
                 roleArn = dwhRole['Role']['Arn']
                 print(roleArn)
             except Exception as e:
                 print(e)
```

```
1.3 Get the IAM role ARN
arn:aws:iam::996813506119:role/dwhRole
```

3.4 STEP 2: Redshift Cluster

- Create a RedShift Cluster

- For complete arguments to `create_cluster`, see docs

```
In [11]: try:
             response = redshift.create_cluster(
                 # TODO: add parameters for hardware
                 ClusterType=DWH_CLUSTER_TYPE,
                 NodeType=DWH_NODE_TYPE,
                 NumberOfNodes=int(DWH_NUM_NODES),

                 # TODO: add parameters for identifiers & credentials
                 DBName=DWH_DB,
                 MasterUsername=DWH_DB_USER,
                 MasterUserPassword=DWH_DB_PASSWORD,
                 ClusterIdentifier=DWH_CLUSTER_IDENTIFIER,

                 # TODO: add parameter for role (to allow s3 access)
                 IamRoles=[roleArn]
             )
         except Exception as e:
             print(e)
```

3.5 2.1 *Describe* the cluster to see its status

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

```
In [14]: def prettyRedshiftProps(props):
             pd.set_option('display.max_colwidth', -1)
             keysToShow = ["ClusterIdentifier", "NodeType", "ClusterStatus", "MasterUsername", "
             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)['
         prettyRedshiftProps(myClusterProps)
```

Out[14]:

	Key	\
0	ClusterIdentifier	
1	NodeType	
2	ClusterStatus	
3	MasterUsername	
4	DBName	
5	Endpoint	
6	VpcId	
7	NumberOfNodes	

	Valu
0	dwhcluster
1	dc2.large
2	available
3	dwhuser

```
4  dwh
5  {'Address': 'dwhcluster.chkkpnquq5ee.us-east-1.redshift.amazonaws.com', 'Port': 5439
6  vpc-0ea020b3d0daa65e7
7  4
```

2.2 Take note of the cluster endpoint and role ARN
DO NOT RUN THIS unless the cluster status becomes "Available"

```
In [16]: 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.chkkpnquq5ee.us-east-1.redshift.amazonaws.com
DWH_ROLE_ARN ::  arn:aws:iam::996813506119:role/dwhRole
```

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

```
In [18]: try:
             vpc = ec2.Vpc(id=myClusterProps['VpcId'])
             defaultSg = list(vpc.security_groups.all())[0]
             print(defaultSg)

             defaultSg.authorize_ingress(
                 GroupName= defaultSg.group_name,  # TODO: fill out
                 CidrIp='0.0.0.0/0',  # TODO: fill out
                 IpProtocol='TCP',  # TODO: fill out
                 FromPort=int(DWH_PORT),
                 ToPort=int(DWH_PORT)
             )
         except Exception as e:
             print(e)
```

```
ec2.SecurityGroup(id='sg-07ee8ca5fc814a7d2')
```

3.7 STEP 4: Make sure you can connect to the clusterConnect to the cluster

```
In [19]: %load_ext sql
```

```
In [20]: conn_string="postgresql://{}:{}@{}:{}/{}".format(DWH_DB_USER, DWH_DB_PASSWORD, DWH_ENDP
         print(conn_string)
         %sql $conn_string
```

```
postgresql://dwhuser:Passw0rd@dwhcluster.chkkpnquq5ee.us-east-1.redshift.amazonaws.com:5439/dwh
```

```
Out[20]: 'Connected: dwhuser@dwh'
```

3.8 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 [21]: #### CAREFUL!!
         #-- Uncomment & run to delete the created resources
         redshift.delete_cluster( ClusterIdentifier=DWH_CLUSTER_IDENTIFIER,  SkipFinalClusterSna
         #### CAREFUL!!
```

```
Out[21]: {'Cluster': {'ClusterIdentifier': 'dwhcluster',
           'NodeType': 'dc2.large',
           'ClusterStatus': 'deleting',
           'MasterUsername': 'dwhuser',
           'DBName': 'dwh',
           'Endpoint': {'Address': 'dwhcluster.chkkpnquq5ee.us-east-1.redshift.amazonaws.com',
            'Port': 5439},
           'ClusterCreateTime': datetime.datetime(2023, 1, 13, 18, 43, 51, 673000, tzinfo=tzloca
           'AutomatedSnapshotRetentionPeriod': 1,
           'ClusterSecurityGroups': [],
           'VpcSecurityGroups': [{'VpcSecurityGroupId': 'sg-07ee8ca5fc814a7d2',
             'Status': 'active'}],
           'ClusterParameterGroups': [{'ParameterGroupName': 'default.redshift-1.0',
             'ParameterApplyStatus': 'in-sync'}],
           'ClusterSubnetGroupName': 'default',
           'VpcId': 'vpc-0ea020b3d0daa65e7',
           'AvailabilityZone': 'us-east-1b',
           'PreferredMaintenanceWindow': 'wed:04:00-wed:04:30',
           'PendingModifiedValues': {},
           'ClusterVersion': '1.0',
           'AllowVersionUpgrade': True,
           'NumberOfNodes': 4,
           'PubliclyAccessible': True,
           'Encrypted': False,
           'Tags': [],
           'EnhancedVpcRouting': False,
           'IamRoles': [{'IamRoleArn': 'arn:aws:iam::996813506119:role/dwhRole',
             'ApplyStatus': 'in-sync'}],
           'MaintenanceTrackName': 'current'},
          'ResponseMetadata': {'RequestId': '1a318707-baed-4300-b0eb-cf129c811f08',
           'HTTPStatusCode': 200,
           'HTTPHeaders': {'x-amzn-requestid': '1a318707-baed-4300-b0eb-cf129c811f08',
            'content-type': 'text/xml',
            'content-length': '2631',
            'date': 'Fri, 13 Jan 2023 18:48:25 GMT'},
           'RetryAttempts': 0}}
```

- run this block several times until the cluster really deleted

```
In [27]: myClusterProps = redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER)['
         prettyRedshiftProps(myClusterProps)


         ---------------------------------------------------------------------------

         ClusterNotFoundFault                      Traceback (most recent call last)

         <ipython-input-27-9b3202a2945e> in <module>()
     ----> 1 myClusterProps = redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER
           2 prettyRedshiftProps(myClusterProps)


         /opt/conda/lib/python3.6/site-packages/botocore/client.py in _api_call(self, *args, **kw
         318                     "%s() only accepts keyword arguments." % py_operation_name)
         319             # The "self" in this scope is referring to the BaseClient.
     --> 320             return self._make_api_call(operation_name, kwargs)
         321
         322         _api_call.__name__ = str(py_operation_name)


         /opt/conda/lib/python3.6/site-packages/botocore/client.py in _make_api_call(self, operat
         621             error_code = parsed_response.get("Error", {}).get("Code")
         622             error_class = self.exceptions.from_code(error_code)
     --> 623             raise error_class(parsed_response, operation_name)
         624         else:
         625             return parsed_response


         ClusterNotFoundFault: An error occurred (ClusterNotFound) when calling the DescribeClust


In [29]: #### 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[29]: {'ResponseMetadata': {'RequestId': '9cce5b28-c6f3-498a-9f44-1f77dd2e2aea',
           'HTTPStatusCode': 200,
           'HTTPHeaders': {'x-amzn-requestid': '9cce5b28-c6f3-498a-9f44-1f77dd2e2aea',
            'content-type': 'text/xml',
            'content-length': '200',
            'date': 'Fri, 13 Jan 2023 18:50:56 GMT'},
           'RetryAttempts': 0}}

In [ ]:
```