```
1  #  The goal of this script is to create a VPC, a gateway which is attached to the VPC, subnet for our vpc using specified
2  #  CIDR block, and creating an EC2 Instance. In the EC2 Instance, we will create and populate a database table in DynamoDB,
3  #  create an S3 bucket, and upload local files making a webpage. This webpage will display the names and roles of the AWS
4  #  team stored in our DB table. Best practice is variables instantiated outside of functions to promote scalability and
5  #  mutability. Second Best Practice is to keep each function compartmentalized. This is usually done is seperate scripts,
6  #  but in the interest of time and simplicity, one script runs all functions.
7
8
9  import boto3
10 import array
11 import subprocess
12 #import os
13 import json
14 import decimal
15
16 # Amazon Machine Image launches using Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type
17 ami = 'ami-0b59bfac6be064b78'
18
19 # Region name, in this case, N. Virginia
20 region = 'us-east-1'
21 instance='t2.micro'
22 myKey='For security best practice, key is omitted but would be provided here'
23
24 # Creates EC2 Connection
25 ec2 = boto3.resource('ec2')
26
27 # Defines our Virtual Private Cloud
28 vpc = ec2.create_vpc(CidrBlock='10.0.0.0/24')
29
30 # Establishes our subnet
31 subnet = vpc.create_subnet(CidrBlock='10.0.0.0/25')
32
33 # Creates a gateway
34 gateway = ec2.create_internet_gateway()
35
36 # Attaches our Gateway to our VPC
```

```
37 gateway.attach_to_vpc(VpcId=vpc.vpc_id)
38
39 # Creates a routing table
40 routeTable = vpc.create_route_table()
41
42 # Routing setup
43 ipv4 = routeTable.create_route(DestinationCidrBlock='0.0.0.0/0',
   GatewayId=gateway.internet_gateway_id)
44 ipv6 = routeTable.create_route(DestinationIpv6CidrBlock='::/0',
   GatewayId=gateway.internet_gateway_id)
45
46 routeTable.associate_with_subnet(SubnetId=subnet.subnet_id)
47
48 # Instantiantes our Security Group
49 security = vpc.create_security_group(GroupName="connectrians",
   Description="This is my sample group")
50
51 ipv4range = [{
52     'CidrIp': '0.0.0.0/0'
53 }]
54
55 ipv6range = [{
56     'CidrIpv6': '::/0'
57 }]
58
59 ports = [{
60     'IpProtocol': 'TCP',
61     'FromPort': 80,  #HTTP Port
62     'ToPort': 80,
63     'IpRanges': ipv4range,
64     'Ipv6Ranges': ipv6range
65 }, {
66     'IpProtocol': 'TCP',
67     'FromPort': 443,  #HTTPS Port
68     'ToPort': 443,
69     'IpRanges': ipv4range,
70     'Ipv6Ranges': ipv6range
71 }, {
72     'IpProtocol': 'TCP',
73     'FromPort': 22, #FTP Port
74     'ToPort': 22,
75     'IpRanges': ipv4range,  # Change to supplement use case
76     'Ipv6Ranges': ipv6range  # Change to supplement use case
77 }]
```

```
 78
 79 security.authorize_ingress(IpPermissions=ports)
 80
 81 # Grab ARN for use in EC2 Instance instantiation
 82 client = boto3.client('iam')
 83 arn=client.get_user()['User']['Arn'].split(':')[4]
 84
 85 ec2Name = 'AWS Test instance'
 86
 87 #Launches EC2 instance
 88 ec2.create_instances(ImageId=ami,
 89                      InstanceType=instance,
 90                      MinCount=1, MaxCount=1,
 91                      SecurityGroupIds=[security.GroupName],
 92                      KeyName=myKey,
 93                      IamInstanceProfile={
 94                             'Arn':  arn,
 95                             'Name': ec2Name
 96 }
 97 )
 98
 99 #v=vpc.vpc_id
100 #g=gateway.internet_gateway_id
101 #s=routeTable.subnet_id
102
103 #bash = array.array(v, g, s)
104
105 #print (*bash)#
106
107 # Making bash variables for switch to mini-bash scripts
108 subprocess.run('$SECURITY_GROUP=connectrians')
109 subprocess.run('$REGION=us-east-1')
110 subprocess.run('$BUCKET=connectrianbucket')
111
112 subprocess.run('aws', 's3api', 'create-bucket', '--bucket $BUCKET',
     '--region $REGION')
113
114 # Runs following CLI command: aws s3api create-bucket --bucket
    $BUCKET --region $REGION
115
116 # Create the S3 client
117 s3 = boto3.client('s3')
118
119 # Get Bucket List from S3
```

```
120 query = s3.list_buckets()
121
122 # Pull the name from query
123 buckets = [bucket['Name'] for bucket in query['Buckets']]
124
125 # Assign the bucket name to a local variable
126 myBucket = print(buckets)
127
128 # Upload files to bucket
129 filename = array.array(subprocess.run('ls', '>', 'files.txt', '|',
    'cat', 'files.txt'))
130 #Runs CLI command ls > files.txt | cat files.txt
131
132 # Parses array for upload and pushes files
133 for x in filename:
134     s3.upload_file(filename[x], myBucket, filename[x])
135
136 endPoint="http://localhost:8000"
137
138 # Initialize Dynamodb connection
139 dynamodb = boto3.resource('dynamodb', region_name=region,
    endpoint_url=endPoint)
140
141 # Creation of our demo table
142 table = dynamodb.create_table(
143     TableName='AWS TEAM',
144     KeySchema=[
145         {
146             'AttributeName': 'Name',
147             'KeyType': 'HASH'  #Partition key
148         },
149         {
150             'AttributeName': 'Role',
151             'KeyType': 'RANGE'  #Sort key
152         }
153     ],
154     AttributeDefinitions=[
155         {
156             'AttributeName': 'Name',
157             'AttributeType': 'S'
158         },
159         {
160             'AttributeName': 'Role',
161             'AttributeType': 'S'
```

```
162         },
163
164     ],
165   # Instatiating Table Size Manually due to lack of autoscaling
166     ProvisionedThroughput={
167         'ReadCapacityUnits': 10,
168         'WriteCapacityUnits': 10
169     }
170 )
171
172 # Converts DynameDB items to JSON.
173 class DecimalEncoder(json.JSONEncoder):
174     def default(self, o):
175         if isinstance(o, decimal.Decimal):
176             if abs(o) % 1 > 0:
177                 return float(o)
178             else:
179                 return int(o)
180         return super(DecimalEncoder, self).default(o)
181
182 myTable = dynamodb.Table('AWS Team')
183
184 name = array.array("Bill West", "AJ Mathis", "Ryan Williams", "
    Charlie Brown", "Ryan McCormick",
185                    "Aileen Curtin", "Brandon Franklin")
186 role = array.array("Director of Cloud Services", "Cloud Architect",
     "Systems Engineer",
187                    "Cloud Solutions Architect", "Sr. Systems
    Engineer", "Solution Delivery Analyst",
188                    "AWS Administrator")
189
190 # Puts items into created database. A Multi-Dimensional array could
     also be used but not as practical with just two values.
191
192 for y in name:
193     myTable.put_item(
194         Item={
195             'Name': name[y],
196             'Role': role[y],
197         }
198 )
199
```