Agenda

1. Boto3 Introduction
2. Boto3 Installation
3. Boto3 Environment setup to work with AWS Services
4. Configure Credentials using AWS CLI Commands
5. Boto3 Core Concepts - Session, resource, Client, Meta, Collections, Waiters and Paginators
6. Session
7. Resource & Client

Sample Excercises

```
1. List IAM Users with resource object
2. List IAM users with Client Object
3. EC2 Instance Create, Stop, Terminate and List Instance ID with state
4. List the S3 bucket
5. S3 Bucket creation, file upload and deletion
6. Cloud watch
7. SNS
8. SQS
```

Boto3

1. Python SDK/API/Library/Module for AWS
2. To ineract with aws services -without AWS Management console window- To maintain or view all session for entire account
3. It allows us to directly create, update,& delete AWS services from our python scripts
4. botocore module - Base module - Low level core functional - exception, session, resource

Boto3 Installation

1. python --version
2. pip3 install boto3 --user
3. python -m pip install --upgrade pip --user - 20.0.1
4. pip3 --version

Boto3 Environment setup to work with AWS Services

1. IAM User Creation - ec2,s3 developer
2. Programmatic access keys - use either root user or any IAM Users
3. Copy and Save ID and Secret access Key for root and IAM Users
4. root user

```
         Access Key ID:
     Secret Access Key:
```

5. IAM User - ec2_Developer

1. IAM User - S3_Developer

Configure Credentials of your AWS account - awscli commands

1. pip3 install awscli --user
2. aws configure - cd .aws , type config, type credentials
3. Configure root / IAM user access keys / Crdentials using aws configure --profile root aws configure --profile non_prod
4. echo %HOMEPATH%
5. cd .aws
6. type config
7. type credentials

In [ ]:
```
#session
AWS Management Console
stores the configuration info
create service clients & resources
```

In [18]:
```
#create a session
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
dir(aws_mag_con_root)
print(aws_mag_con_root.get_available_resources())
```

```
['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opswor
ks', 's3', 'sns', 'sqs']
```

In [ ]:
```
#resource & Client

Resource - higher level object oriented service access & AWS some of the servi
ces
client - low level service access
         o/p dict format
         IAM Operations
```

In [23]:
```
#list IAM users with resource object
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
iam_con_re=aws_mag_con_root.resource(service_name='iam',region_name='us-east-
2')
for each_user in iam_con_re.users.all():
    print(each_user.name)
dir(each_user)
```

```
ec2_developer
rds_developer
s3_developer
```

```
Out[23]: ['AccessKey',
 'LoginProfile',
 'MfaDevice',
 'Policy',
 'SigningCertificate',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_name',
 'access_keys',
 'add_group',
 'arn',
 'attach_policy',
 'attached_policies',
 'create',
 'create_access_key_pair',
 'create_date',
 'create_login_profile',
 'create_policy',
 'delete',
 'detach_policy',
 'enable_mfa',
 'get_available_subresources',
 'groups',
 'load',
 'meta',
 'mfa_devices',
 'name',
 'password_last_used',
 'path',
 'permissions_boundary',
 'policies',
 'reload',
 'remove_group',
 'signing_certificates',
 'tags',
 'update',
 'user_id',
 'user_name']
```

In [28]:
```
#List IAM users with resource object
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
iam_con_cli=aws_mag_con_root.client(service_name='iam',region_name='us-east-2'
)
print(iam_con_cli)
print(iam_con_cli.list_users()['Users'])
```

```
<botocore.client.IAM object at 0x000001CF1F5CB448>
[{'Path': '/', 'UserName': 'ec2_developer', 'UserId': 'AIDA4EETYBK6STPRRKLH
B', 'Arn': 'arn:aws:iam::833533512381:user/ec2_developer', 'CreateDate': date
time.datetime(2020, 6, 9, 5, 22, 45, tzinfo=tzutc())}, {'Path': '/', 'UserNam
e': 'rds_developer', 'UserId': 'AIDA4EETYBK63WCF2P3A6', 'Arn': 'arn:aws:iam::
833533512381:user/rds_developer', 'CreateDate': datetime.datetime(2020, 6, 9,
7, 0, 46, tzinfo=tzutc())}, {'Path': '/', 'UserName': 's3_developer', 'UserI
d': 'AIDA4EETYBK6WAWPEVHUL', 'Arn': 'arn:aws:iam::833533512381:user/s3_develo
per', 'CreateDate': datetime.datetime(2020, 6, 9, 5, 23, 36, tzinfo=tzutc
())}]
s3_developer
s3_developer
s3_developer
```

In [29]:
```
for eachuser in iam_con_cli.list_users()['Users']:
    print(each['UserName'])
```

```
s3_developer
s3_developer
s3_developer
```

In [30]:
```
#EC2
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
ec2 = aws_mag_con_root.client(service_name='ec2',region_name='us-east-2')
response = ec2.describe_key_pairs()
print(response)
```

```
{'KeyPairs': [{'KeyPairId': 'key-018cf6422fc6d916a', 'KeyFingerprint': '54:d
1:47:37:3f:75:aa:67:f1:f8:c4:64:8e:26:f7:d3:10:de:c9:58', 'KeyName': 'cloudwa
tch', 'Tags': []}, {'KeyPairId': 'key-0ccc1da3c42f7dbcb', 'KeyFingerprint':
'4b:e9:00:99:0c:95:44:f7:1a:22:56:e2:49:d8:58:b4:28:98:cd:f3', 'KeyName': 'Jo
hn_Key', 'Tags': []}], 'ResponseMetadata': {'RequestId': 'bc620177-166e-413e-
81d0-c87dfa4009cc', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requesti
d': 'bc620177-166e-413e-81d0-c87dfa4009cc', 'content-type': 'text/xml;charset
=UTF-8', 'content-length': '746', 'date': 'Tue, 09 Jun 2020 07:49:29 GMT', 's
erver': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

In [34]:
```
#create ec2 instance
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
ec2 = aws_mag_con_root.resource(service_name='ec2',region_name='us-east-2')
instance = ec2.create_instances(
 ImageId='ami-07c1207a9d40bc3bd',
 MinCount=1,
 MaxCount=1,
 InstanceType='t2.micro')
```

In [37]:
```
print(instance[0].id)
```

```
i-0f568ed09370315fe
```

In [38]:
```
#Terminating instances
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
ec2 = aws_mag_con_root.resource('ec2')
instance = ec2.Instance("i-0f568ed09370315fe")
response = instance.terminate()
print(response)
```

```
{'TerminatingInstances': [{'CurrentState': {'Code': 32, 'Name': 'shutting-dow
n'}, 'InstanceId': 'i-0f568ed09370315fe', 'PreviousState': {'Code': 16, 'Nam
e': 'running'}}], 'ResponseMetadata': {'RequestId': '121255f6-57c3-4146-aac7-
cbb7cbaed6cf', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '12
1255f6-57c3-4146-aac7-cbb7cbaed6cf', 'content-type': 'text/xml;charset=UTF-
8', 'transfer-encoding': 'chunked', 'vary': 'accept-encoding', 'date': 'Tue,
09 Jun 2020 07:57:34 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

In [39]:
```
#list all instances with state
import boto3
aws_mag_con_root=boto3.session.Session(profile_name="root")
ec2 = aws_mag_con_root.resource('ec2')
for instance in ec2.instances.all():
    print(instance.id, instance.state)
```

```
i-0f568ed09370315fe {'Code': 48, 'Name': 'terminated'}
i-0c0156ac572534a1d {'Code': 16, 'Name': 'running'}
```

In [ ]:
```

```