20 Oct: Introduction

20+yrs

**A - AI/Data Science- ML, DL, NN, NLU, NLP, Computer Vision**

B - BigData( Hadoop → Developer,Admin, Spark - PySpark, Databricks - Data Engineers, AutoML)

C - Cloud ( AWS, Azure, GCP)

D - Dev( Python ) + Ops(TF,Ansible, Jenkins, Cloud based CI, Containers, Orchestration)

Python, Java, MS.Net = 17+NC+ GM

Data(BigData)+Platform(Cloud)+Automation(DevOps)+Intelligence(AI)

CloudOps

IaaC - JSON, Python, Powershell, Shell scripting, YAML

Python System

Text Processing Google Search Engine Python

Video Processing Youtube Python

Distributed Computing Bittorrent Python

Cloud Computing AWS,DropBox Python

Game/Enter. Disney, Blender Python

IoT RBPi, Drone, CV, AVR Python

Data Platform Databricks Python(PySpark)

Serverless Microservices Python

Infra(Administrator= Operator) → Automation

Oracle DBA

12c → 15

18c → 05

Win3.1/98/XP → Win10

Cloud/AI-1950

Platform

Baremetal

Virtualization

Cloud - Pay as you go

Dedicated, Shareable, Containerization, Serverless

Containerization

Microservices

Cloud Leaders

1. AWS = 2002/6 → 2021
2. Azure
3. GCP

Python+AWS = CloudOps(Automation with Cloud)

AWS/Azure/GCP

Web Console = Understanding(Beginners)

CLI = Automation(DevOps+Administrator)

SDK = Programming Language(Python), Dev(boto3) (DevOps+Development)

API = Integrations(Architectures)

IaaC Tool = CFT, TF,etc.

Breaks

11:15

1-2

4:30

1. Write Steps
2. Practically
3. Try here
4. Response

What

AWS > Services + Integration

Python > boto3(steps)

AWS Services

S3

EC2,

IAM,

RDS

DynamoDB,

CloudWatch

Lambda,

Step Functions

Kinesis

Data Automation

Files (S3+IAM+Lambda+CloudWatch+RDS)

Stream (Kinesis+DynamoDB)

System Automation

EC2, EBS, Snapshots, AMI

Python

Collection

List Comprehensions

Iterations

Software/Tools

> Python **3**.X # python --version(Done)

> AWS IAM Account+CLI(**2**.0.X) # aws --version

> Git-SCM **2**.28.0 # git --version

> VSCode Editor(Windows, Mac, Linux)-**1**.61.1 # code --version

> No Proxy/VPN

C:\Python390\Scripts\ **and** C:\Python390\

C:\Program Files\Amazon\AWSCLIV2\

C:\Program Files\Microsoft VS Code\bin

1. AWS CLI Configure
   1. # aws s3 ls
   2. # aws configure
      1. Access Key :
      2. Secret AK :
      3. Region Code :
      4. Output Format:

https://862140262984.signin.aws.amazon.com/console User1 Aws@12345 AKIA4RO4QNJEP2I3BTL5 wI/8S7OmnmGbnYcX213TP/bEx4RhwChXyX7BZ/k9

https://862140262984.signin.aws.amazon.com/console User2 Aws@12345 AKIA4RO4QNJEPEAEUSUB wS8cAu6Toq0tV+MGIcRCnICj0dFalIWE2IZyd7H/

https://862140262984.signin.aws.amazon.com/console User3 Aws@12345 AKIA4RO4QNJECKQIKOGW LBgnlmFpHb+dDX83ovu6UDVrlWaz6TNDqGu8Y5dU

https://862140262984.signin.aws.amazon.com/console User4 Aws@12345 AKIA4RO4QNJEAZ5HBWM5 SlkGYl5DJP2eX+XriwbNCvL5NzeOL6durQqpE8Qv

https://862140262984.signin.aws.amazon.com/console User5 Aws@12345 AKIA4RO4QNJELOB6GWGO H3+TT4bR72gttz5LcynatBvlnAf8i52mT7Gf6mS4

https://862140262984.signin.aws.amazon.com/console User6 Aws@12345 AKIA4RO4QNJECSQGYS6L m+etelWNIr/jB1R7+oPylvSn9VbqC7Y5GsNergyc

https://862140262984.signin.aws.amazon.com/console User7 Aws@12345 AKIA4RO4QNJEPHMLKQPE E3HhTMMnaBn3ioNtsoMBEGN41gPTfjk51Tpv/EHx

https://862140262984.signin.aws.amazon.com/console User8 Aws@12345 AKIA4RO4QNJEPKDZ424O 3IKUp67qZkPxEWkDhipjEA9cIWQ3qTlM2KltqD33

https://862140262984.signin.aws.amazon.com/console User9 Aws@12345 AKIA4RO4QNJEIMP5I3OR zQZ8gbCRQZeLGwRpxSRJVVc4bTAJcIx6xWz4+bnF

https://862140262984.signin.aws.amazon.com/console User10 Aws@12345 AKIA4RO4QNJEHYXSV7WT lVZozOg4sL78QHYBIXBf+baE8UnJ1mYCdZpUP24m

https://862140262984.signin.aws.amazon.com/console User11 Aws@12345 AKIA4RO4QNJEBJOSNO44 3WcerXirtvai9zJ7H6Awc9MJj/tHMPHujR8aRw1R

https://862140262984.signin.aws.amazon.com/console User12 Aws@12345 AKIA4RO4QNJEI5YZWUHW 75x/xk2ITWAlAOCegLui/msl23+Ik1+tzkJvDbci

https://862140262984.signin.aws.amazon.com/console User13 Aws@12345 AKIA4RO4QNJELANU4BEX bRcErdBGKQ27SnqIF7K8QmV/JfnZYXD5UTm466cj

https://862140262984.signin.aws.amazon.com/console User14 Aws@12345 AKIA4RO4QNJELA2M3IWE yCDDlMU77E+qDZGeqRiZHJbsi3tJ/lYDhKk+Zziu

https://862140262984.signin.aws.amazon.com/console User15 Aws@12345 AKIA4RO4QNJEKFRJAH5P AX5LCLm7kWT3jXPQUGrG4qf8JHAOtA/KRlHIbiS5

https://862140262984.signin.aws.amazon.com/console User16 Aws@12345 AKIA4RO4QNJEH3B6ZVWK 8U/TCeTCGsdA8OK2M4Z8YQ9ncwoA6cTUwS3nImVX

https://862140262984.signin.aws.amazon.com/console User17 Aws@12345 AKIA4RO4QNJEB6P5JB4E 2loCHo6Esm5Mvh3fd6WFJ0mJE4RmTR3IHBYdxQQq

https://862140262984.signin.aws.amazon.com/console User18 Aws@12345 AKIA4RO4QNJELFU24SNS 5+9l6eHkkuBaiAOo7MhqjLrcbcZoZWZIpfXUAWV6

https://862140262984.signin.aws.amazon.com/console User19 Aws@12345 AKIA4RO4QNJEBNQYQH5P 39NvKROKprESeUrPhtpZW8y7VUvoQvl33zNTxp7c

https://862140262984.signin.aws.amazon.com/console User20 Aws@12345 AKIA4RO4QNJEDMAWZTFC UgxiPkYhKnIDyyfwvWchXfMbIdnx627/W7H6xIfD

> aws configure

> CurrentUserDir/.aws/credentials and config

Boto3

1. AWS Configure
   1. # aws configure
      1. AK:
      2. SAK:
      3. Region Code:
      4. Output Format: json
2. PIP LIST
   1. # pip list
   2. # pip install virtualenv
3. Create Environment
   1. # python -m virtualenv .envBoto3
4. Activate Environment
   1. Windows # .envBoto3/Scripts/activate.bat
5. Check List
   1. # pip list
6. Install Boto3
   1. # pip install boto3
7. Check Boto3
   1. # notepad s3bucketlist.py
      1. >>> import boto3
   2. # python s3bucketlist.py

## 

## Basic Level

Tokens

Statements & Functions, Lambda

IO Statements

print

Input

Conditional

Loop

Functions

Lambda

Collections, List Comprehension

Module & Packages

## Intermediate Level(Services)

Installation

Configuration

Using Boto3

Boto3 Package Objects: Session, Resources,etc

Boto3 with AWS Services: Lambda, CloudWatch, DynamoDB,RDS

X=[[11,22,33]

,[44,55,66]

,[77,88,99]]

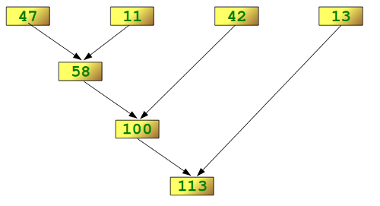
Y=[[11,22,33]

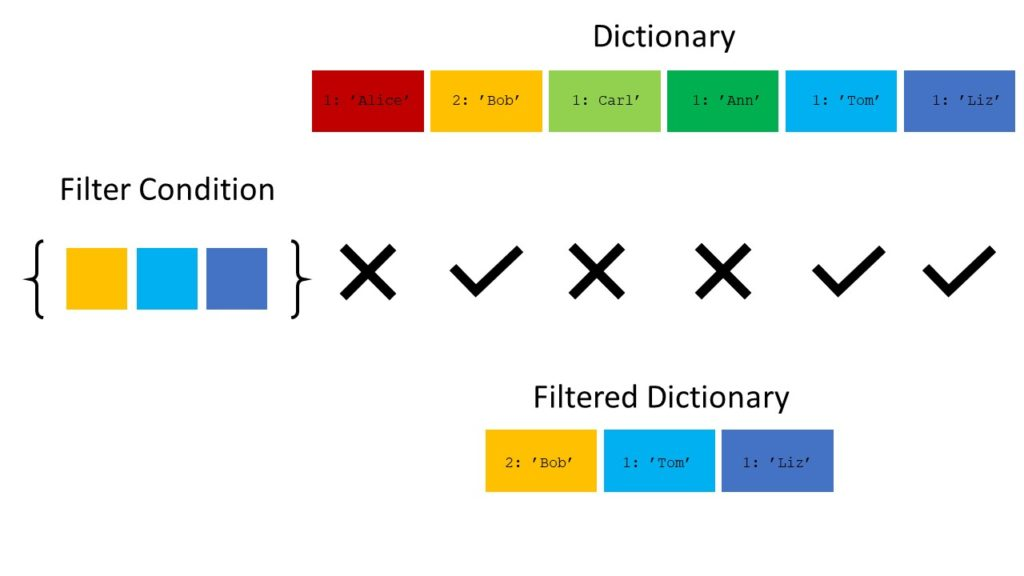
,[44,55,66]

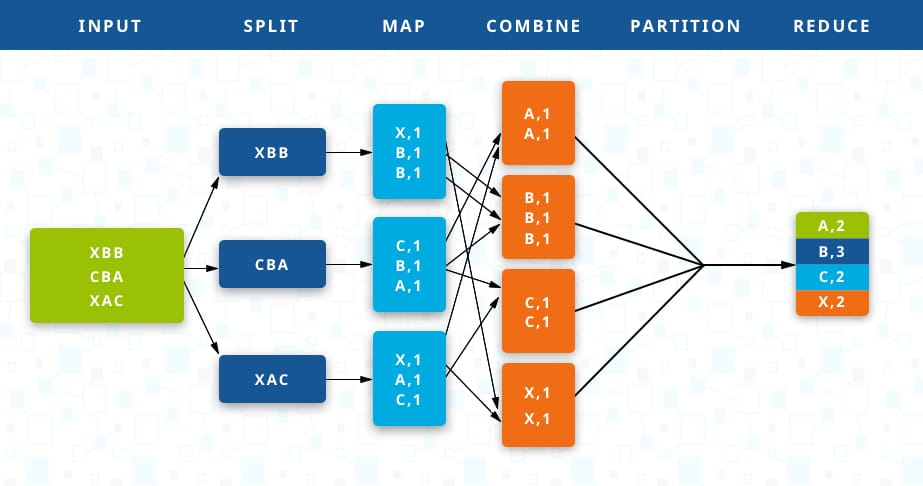
,[77,88,99]]

result = [[X[i][j] + Y[i][j] for j in range(len(X[0]))] for i in range(len(X))]

print(result)







1. Inbuilt Package ( functools)
2. Custom Package( User define ) → pypi.org
3. Installable Package → boto3

Package

Function

Class

Methods(self)

Module(Function, Class, Statements)

AWS

Set of services → Features → Resources

Clients

Session

Resources

Collections

Paginations

Retries

Error Handling

Credentials

S3

1. List Buckets
2. Accept Bucket Name & Create Bucket
3. Write File in Local Computer
4. Upload to S3 Bucket
5. List Objects & Buckets

Credentials

# aws configure( ~/.boto and /etc/boto.cfg)

~/.aws/credentials

aws\_access\_key=Value

role\_arn=arn:aws:iam:feature:resource

web\_identity\_token\_file=/file/path/abc

SSO

AWS CLI V2

## Writing File in Local Machine

f=open("abc.txt","w")

f.write("Welcome to File Handling in Python")

f.close()

# 21 Oct: AWS Services using Boto3

1. Create an Empty Folder
2. Check VirtualEnv Package
   1. # pip show virtualenv
3. Create an Environment
   1. # python -m virtualenv .envBoto3
4. Activate Environment
   1. CMD# .envBoto3\Scripts\activate.bat
   2. PS# .envBoto3\Scripts\activate.ps1
5. Install Boto3
   1. # pip install boto3

## Create Security Group

import boto3

ec2\_client=boto3.client("ec2")

response=ec2\_client.create\_security\_group(

GroupName="mujahedsg001",

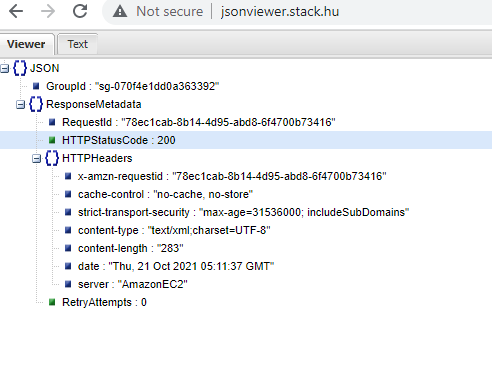
Description="Mujahed Security group here",

VpcId="vpc-fd2df196"

)

print(response[“GroupId”])

{'GroupId': 'sg-070f4e1dd0a363392', 'ResponseMetadata': {'RequestId': '78ec1cab-8b14-4d95-abd8-6f4700b73416', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '78ec1cab-8b14-4d95-abd8-6f4700b73416', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml;charset=UTF-8', 'content-length': '283', 'date': 'Thu, 21 Oct 2021 05:11:37 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}



Egress = Outbound Rule

Ingress = Inbound Rule

import boto3

ec2\_client=boto3.client("ec2")

security\_group\_object=ec2\_client.create\_security\_group(

GroupName="mujahedsg002",

Description="Mujahed Security group here",

VpcId="vpc-fd2df196"

)

print(security\_group\_object)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

response=security\_group\_object.authorize\_security\_group\_ingress( IpProtocol="tcp",ToPort=22,FromPort=22,CidrIp="0.0.0.0/0" )

print(response)

import boto3

ec2\_client=boto3.client("ec2")

groupId="sg-070f4e1dd0a363392"

ip\_perm1=[

{

"FromPort":22,

"ToPort":22,

"IpProtocol":"tcp",

"IpRanges":[

{

"CidrIp":"11.22.33.44/24",

"Description":"SSH for My Friend"

}

]

}

]

response=ec2\_client.authorize\_security\_group\_ingress(

GroupId=groupId,

IpPermissions=ip\_perm1

)

print(response)

import boto3

ec2\_client=boto3.client("ec2")

groupId="sg-070f4e1dd0a363392"

response=ec2\_client.delete\_security\_group(GroupId=groupId)

print(response)

Helpers

SecurityGroupHelper

Service2

Driver

Workflow

import boto3

class SGHelper:

def \_\_init\_\_(self):

self.ec2\_client=boto3.client("ec2")

def createSG(self,GName,VpcId="vpc-fd2df196"):

response=self.ec2\_client.create\_security\_group(

GroupName=GName,

Description="Mujahed Security group here",

VpcId=VpcId

)

return response

def getClientObject(self):

return self.ec2\_client

#DriverClass

sg=SGHelper()

groupName="mujahedclassbasedsg"

response=sg.createSG(GName=groupName)

print(response)

#sg-070f4e1dd0a363392

import string

import random

z=string.ascii\_lowercase[:24]

y=string.ascii\_lowercase[:24:2]

x=random.choice("abcdefgh")

print(x)

import boto3

cloudwatch=boto3.client("cloudwatch")

# response = client.put\_metric\_alarm(

# AlarmName='string',

# AlarmDescription='string',

# ActionsEnabled=True|False,

# OKActions=[

# 'string',

# ],

# AlarmActions=[

# 'string',

# ],

# InsufficientDataActions=[

# 'string',

# ],

# MetricName='string',

# Namespace='string',

# Statistic='SampleCount'|'Average'|'Sum'|'Minimum'|'Maximum',

# ExtendedStatistic='string',

# Dimensions=[

# {

# 'Name': 'string',

# 'Value': 'string'

# },

# ],

# Period=123,

# Unit='Seconds'|'Microseconds'|'Milliseconds'|'Bytes'|'Kilobytes'|'Megabytes'|'Gigabytes'|'Terabytes'|'Bits'|'Kilobits'|'Megabits'|'Gigabits'|'Terabits'|'Percent'|'Count'|'Bytes/Second'|'Kilobytes/Second'|'Megabytes/Second'|'Gigabytes/Second'|'Terabytes/Second'|'Bits/Second'|'Kilobits/Second'|'Megabits/Second'|'Gigabits/Second'|'Terabits/Second'|'Count/Second'|'None',

# EvaluationPeriods=123,

# DatapointsToAlarm=123,

# Threshold=123.0,

# ComparisonOperator='GreaterThanOrEqualToThreshold'|'GreaterThanThreshold'|'LessThanThreshold'|'LessThanOrEqualToThreshold'|'LessThanLowerOrGreaterThanUpperThreshold'|'LessThanLowerThreshold'|'GreaterThanUpperThreshold',

# TreatMissingData='string',

# EvaluateLowSampleCountPercentile='string',

# Metrics=[

# {

# 'Id': 'string',

# 'MetricStat': {

# 'Metric': {

# 'Namespace': 'string',

# 'MetricName': 'string',

# 'Dimensions': [

# {

# 'Name': 'string',

# 'Value': 'string'

# },

# ]

# },

# 'Period': 123,

# 'Stat': 'string',

# 'Unit': 'Seconds'|'Microseconds'|'Milliseconds'|'Bytes'|'Kilobytes'|'Megabytes'|'Gigabytes'|'Terabytes'|'Bits'|'Kilobits'|'Megabits'|'Gigabits'|'Terabits'|'Percent'|'Count'|'Bytes/Second'|'Kilobytes/Second'|'Megabytes/Second'|'Gigabytes/Second'|'Terabytes/Second'|'Bits/Second'|'Kilobits/Second'|'Megabits/Second'|'Gigabits/Second'|'Terabits/Second'|'Count/Second'|'None'

# },

# 'Expression': 'string',

# 'Label': 'string',

# 'ReturnData': True|False,

# 'Period': 123,

# 'AccountId': 'string'

# },

# ],

# Tags=[

# {

# 'Key': 'string',

# 'Value': 'string'

# },

# ],

# ThresholdMetricId='string'

# )

response=cloudwatch.put\_metric\_alarm(

AlarmName="mujahed\_webserver\_cpuutilization",

ComparisonOperator="GreaterThanThreshold",

EvaluationPeriods=1,

MetricName="CPUUtilization",

Namespace="AWS/EC2",

Period=60,

Statistic="Average",

Threshold=70.0,

ActionsEnabled=False,

AlarmDescription="This is Web Server Alarm for CPU Utilization",

Dimensions=[

{

"Name":"InstanceId",

"Value":"INSTANCE\_ID"

}

],

Unit="Seconds"

)

print(response)

import boto3

cloudwatch=boto3.client("cloudwatch")

paginator=cloudwatch.get\_paginator("describe\_alarms")

for page in paginator.paginate(StateValue="INSUFFICIENT\_DATA"):

print(page["MetricAlarms"])

import boto3

cw=boto3.client("cloudwatch")

response=cw.delete\_alarms(

AlarmNames=["mujahed\_webserver\_cpuutilization"]

)

print(response)

class DesDemo:

def \_\_init\_\_(self):

print("Welcome")

def \_\_del\_\_(self):

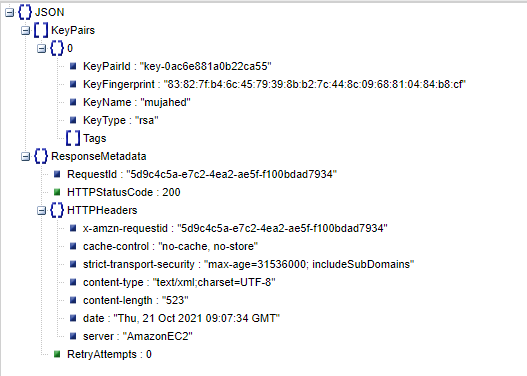
print("Bye")

print("1st")

dd=DesDemo()

print("2nd")

response=ec2.describe\_key\_pairs()



EC2

EBS

Volumes - Cloud Virtual Hard Disk

Size=10

VolumeType=gp2

Snapshot - Cloud Virtual Backup Drive

AMI - Pre Installed Operating System+Tools

import boto3

AWS\_REGION="ap-south-1"

ec2=boto3.client("ec2",region\_name=AWS\_REGION)

newVolume=ec2.create\_volume(

AvailabilityZone=f"{AWS\_REGION}c",

Size=11,

VolumeType="gp2",

TagSpecifications=[

{

"ResourceType":"volume",

"Tags":[

{

"Key":"Name",

"Value":"mujahedvolume"

}

]

}

]

)

print(f'Created Volume with ID: {newVolume["VolumeId"]}')

import boto3

ec2=boto3.client("ec2")

response=ec2.describe\_regions()

print(response["Regions"])

response=ec2.describe\_availability\_zones()

print(response["AvailabilityZones"])

import boto3

ec2=boto3.client("ec2",region\_name="us-west-2")

response=ec2.create\_instances(

BlockDeviceMapping=[

{

"DeviceName":"/dev/xvda",

"Ebs":{

"DeleteOnTermination":True,

"VolumeSize":8,

"VolumeType":"gp2"

}

}

],

ImageId="ami-087107f9778206adb",

InstanceType="t2.micro",

MinCount=1,

MaxCount=2,

Monitoring={

"Enabled":False

},

SecurityGroupIds=[

"sg-abaefd32dc3d23"

]

)

import boto3

ec2=boto3.client("vpc")

vpc=ec2.create\_vpc(CidrBlock="10.0.0.0/24")

subnet=vpc.create\_subnet(CidrBlock="10.0.0.0/25")

gateway=ec2.create\_internet\_gateway()

gateway.attach\_to\_vpc(VpcId=vpc.id)

gateway.detach\_from\_vpc(VpcId=vpc.id)

eip="eip-234nfdqw34ui328"

InstanceId="i-aslndclkacj389"

address=ec2.VpcAddress(eip)

address.associate(InstanceId)

address.association.delete()

#boto3 developer

def f1(\*a):

for i in range(len(a)):

print(i)

#AWS SysOps

data=(1,2,3)

f1(data)

Processing

ServerBase - AWS EC2 (Infras.)

Serverless - AWS Lambda (Submit Job)

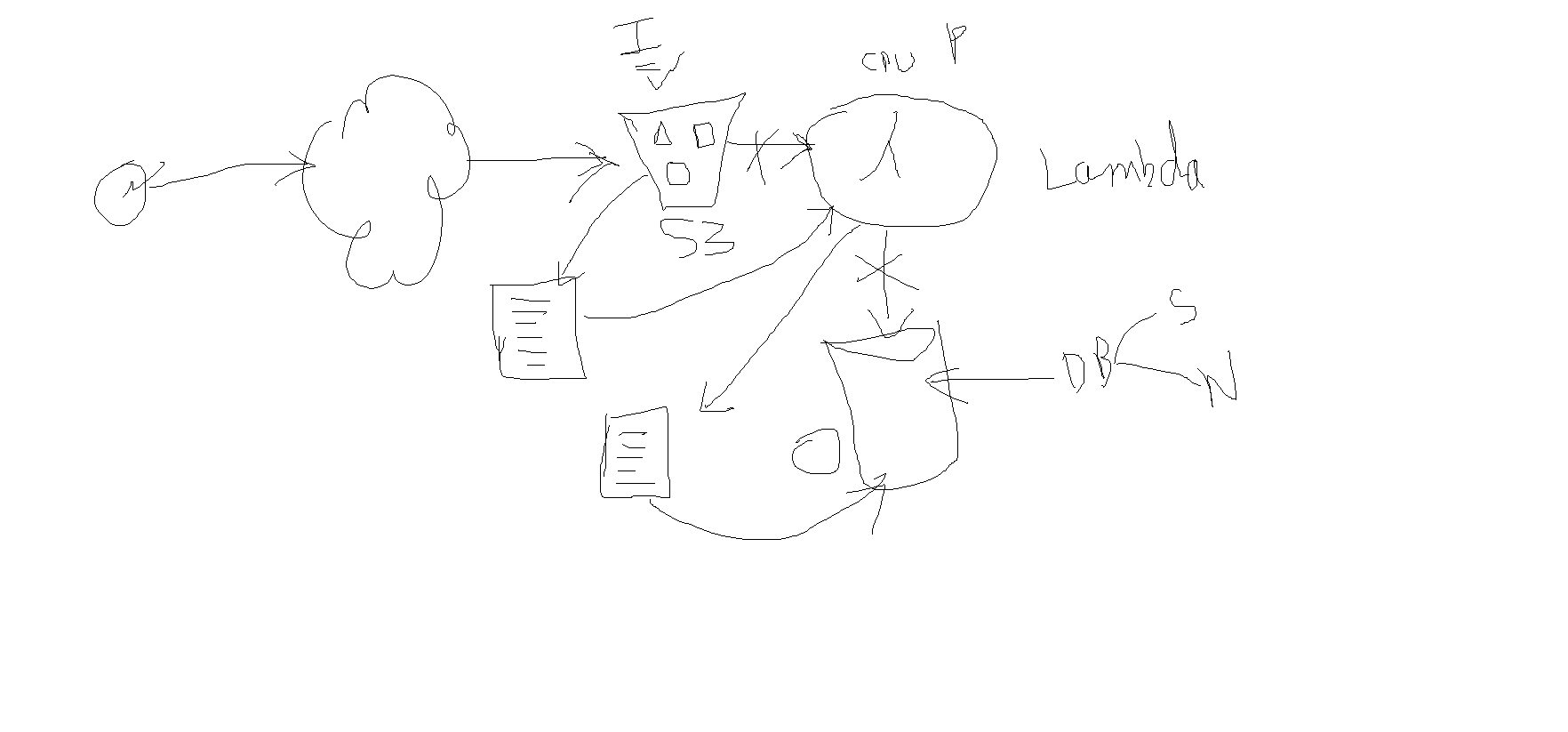
Event Driven Programing

S3 = ReArchitecture Pattern

Live = Streaming

Dead = Files(Structured, Unstructured, Semi.), Trans, OLAP, OLTP, IoT

Sys1→ 1 Hr 10 GB → S3 → SQS(Buffer) → Sys2(Lambda+AI/ML = 1GB - 100Min) → SNS



Event Notification

Notify = SNS( SES, Tele Cloud, SMS, Voice, WhatsApp API)

Processing = Lambda(On the fly processing) = Video Analytics

Who is Using What they are using

Trusted Entity Policy

S3 → IAM(Role) → Lambda

Service Service

S3 → IAM(Policy) → User1

Service User/Group

1. Create IAM Role(Service to Service)
   1. Trusted Entity: Lambda
   2. Policy:
      1. CloudWatch
      2. RDS
      3. S3
      4. DynamoDB
2. Create Lambda
   1. Template Type
   2. Select Runtime
   3. Select Role
3. Create S3
   1. Create Bucket
      1. Event Notification → Lambda

IaaC

JSON → S3(Bucket)

→ Event Notification → IAM(Role)

→ Lambda(Python- Understand JSON → Table→ RDS)

CloudWatchLogs → RDS

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html>

# 22 Oct: System Integration(AWS+Python)

Python

AWS

Boto3

Workflow/Requirements

> Storage/Data- Traditional Data- Redshift, Big Data- EMR, New Data- IoT

> Automation - S3 → Role → Lambda → NoSQL

> AI/ML - Loading Library(sklearn,tensorflow,keras) , Selecting Services(AWS AIML)

Data+Platform = Intelligence

Traditional Data

> Relational → RDBMS(SQL) → **RDS**

> Warehouse → BI

Big Data

NoSQL

Hadoop

Spark

DataBricks

EMR

New Data

NoSQL

**DynamoDB**

**EBS**

Snapshot, Volume

EBS

Stored in Blocks

## Practical 1: Setting up Project Environment

1. Create requirements.txt
   1. # pip freeze > requirements.txt
2. Create Empty Environment
   1. # python -m virtualenv .envBoto3
3. Activate Environment
   1. # .envBoto3\Scripts\activate.bat
4. Install Library
   1. # pip list
   2. # pip install -r requirements.txt
   3. # pip list

## Practical 02: Create Volume

import boto3

AWS\_REGION="us-east-2"

ec2\_client=boto3.client("ec2",region\_name=AWS\_REGION)

new\_volume=ec2\_client.create\_volume(

AvailabilityZone=f'{AWS\_REGION}c',

Size=10,

VolumeType='gp2',

TagSpecifications=[

{

'ResourceType': 'volume',

'Tags': [

{

'Key': 'Name',

'Value': 'mujahedvolume1'

}

]

}

]

)

print(f"Created volume ID: {new\_volume['VolumeId']}")

#vol-024820347f43c53c1

## Practical 03: Listing Volume

import boto3

AWS\_REGION="us-east-2"

ec2\_resource=boto3.resource("ec2",region\_name=AWS\_REGION)

volumeCollection=ec2\_resource.volumes.all()

for volume in volumeCollection:

print(volume)

## 

## Practical 04: Listing all Regions & AZ

import boto3

ec2=boto3.client("ec2")

response=ec2.describe\_regions()

print(response["Regions"])

response=ec2.describe\_availability\_zones()

print(response["AvailabilityZones"])

## Practical 05: Add DCVCS in Project

1. Git Initialization
   1. # git init
2. Add All files
   1. # git add .
3. Configure Git Repo
   1. # git config user.name "githubUserName"
   2. # git config user.email "githubEmailName"
4. Commit
   1. # git commit -m "Integration & Project"
5. Add Remote
   1. # git remote add origin <https://github.com/NubeEra-Samples/AWSBoto3Python.git>
6. Add Branch
   1. # git branch -M main
7. Push to Remote Repo
   1. # git push -u origin main

## Practical 06: Search Volume using Filter

import boto3

AWS\_REGION = "us-east-2"

ec2\_resource = boto3.resource('ec2', region\_name=AWS\_REGION)

for volume in ec2\_resource.volumes.filter(

VolumeIds=[

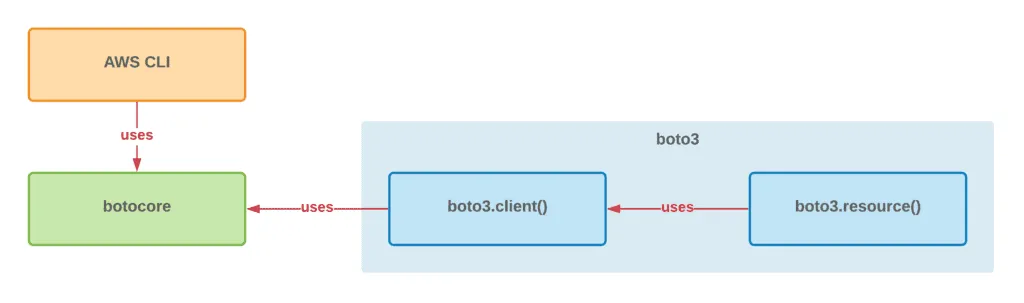
'vol-0075f8ef1cf617e12',

'vol-0c49d1090104becca'

],

):

print(f'Volume {volume.id} ({volume.size} GiB) -> {volume.state}')



## Practical 07: JSON Parsing

import json

import pathlib

BASE\_DIR=pathlib.Path(\_\_file\_\_).parent.resolve()

#FILE\_DIR=f"{BASE\_DIR}\\files"

data={

"Id":1,

"UserName":"Apple",

"Password":"Orange"

}

with open(f"{BASE\_DIR}\\data.json","w") as fp:

json.dump(data,fp)

fp.write("\n")

RDS

Features(Instances → DBEngine → Database → Userdetails, Snapshots)

## Practical 08: Create RDS Database Instance

import boto3

client=boto3.client("rds")

response=client.create\_db\_instance(

AllocatedStorage=5,

DBInstanceClass="db.t2.micro",

DBInstanceIdentifier="mujahed-db-instance-01",

Engine="MySQL",

MasterUsername="admin01",

MasterUserPassword="test123pwd0021"

);

print(response)

# Practical 09: Connectivity From RDS to Local Machine

1. DB Instance
   1. Check Public Access
   2. Check Security Group if not add
2. Command :
   1. EndPoint
   2. # mysql -h **mujahed-db-instance-01.cnfs3veczxfo.ap-south-1.rds.amazonaws.com** -u admin01 -p
   3. Enter Password: test123pwd0021

## Practical 11: Describe DB Instance using Filter

response=client.describe\_db\_instances(

DBInstanceIdentifier="",

Filters=[

{

"Name":"engine",

"Values":["MySQL"]

}

]

)

print(response)

## Practical 12: Manage Instance State

import boto3

client=boto3.client("rds")

response = client.stop\_db\_instance(

DBInstanceIdentifier='mujahed-db-instance-01',

DBSnapshotIdentifier='stop-snapshot001'

)

response = client.start\_db\_instance(

DBInstanceIdentifier='mujahed-db-instance-01'

)

response = client.modify\_db\_instance(

DBInstanceIdentifier='mujahed-db-instance-01',

MasterUserPassword="NEW PASSWORD"

)

response = client.create\_db\_instance\_read\_replica(

DBInstanceIdentifier='mujahed-db-instance-01',

PublicyAccessible=True,

DBInstanceClass="db.t2.micro",

SourceDBInstanceIdentifier="mujahed-db-instance-01-read-replica",

StorageType="gp2",

Tags=[

{

"Key":"ReadreplicaNumber",

"Value":"readreplica001"

}

]

)

## Practical 13: Deleting DB Instance

import boto3

client = boto3.client('rds')

response = client.delete\_db\_instance(

DBInstanceIdentifier='mujahed-db-instance-01',

SkipFinalSnapshot=True

)

print(response)

## NoSQL

Key-Value+Document ⇒ DynamoDB (NoSQLaaS - DaaS)

Graph

Columnar

<https://youtu.be/fiP2e-g-r4g>

<https://youtu.be/_KNrRdWD25M>

Dynamic Schema

Practical 11: Create Table

## Practical 12: Batch Write item

import boto3

dynamodb=boto3.resource("dynamodb")

table =dynamodb.Table("MujahedEmployees")

with table.batch\_writer() as batch:

batch.put\_item(Item={"Name":"B1","Email":"b1@nubeera.com"})

batch.put\_item(Item={"Name":"B2","Email":"b2@nubeera.com"})

print(batch)

IAM

Features(Roles, Policy)

Policy

JSON

Statement

[

{

...S3FullAccess..

}

]

## Practical 13: Create IAM Role

import boto3

import json

iam=boto3.client("iam")

# policyLambdaProject={

# "Version": "2012-10-17",

# "Statement": [

# {

# "Sid": "",

# "Effect": "Allow",

# "Principal":{

# "Service":"lambda.amazonaws.com"

# },

# "Action": "sts:assumeRole"

# }

# ]

# }

policyLambdaProject={

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Principal":{

"Service":"lambda.amazonaws.com"

},

"Action": "dynamodb:\*"

}

]

}

response=iam.create\_role(

RoleName="rlmujahedv1",

AssumeRolePolicyDocument=json.dumps(policyLambdaProject)

)

## Practical 14: Paginator in Policies

import boto3

iam=boto3.client("iam")

response=iam.list\_policies(

Scope="Local"

)

paginator=iam.get\_paginator("list\_policies")

for page in paginator.paginate():

for p in page["Policies"]:

print(f"- { p['PolicyName'] }")

## Practical 15: Attach Policy on Role

import boto3

ROLE\_NAME="rl\_serverless\_mujahed"

iam=boto3.resource("iam")

role=iam.Role(ROLE\_NAME)

role.attach\_policy(

PolicyArn="arn:aws:iam::aws:policy/AmazonS3FullAccess"

)

print("Policy Attached...")

## Practical 16: create lambda (handler.py)

import json

def lambda\_handler(event,context):

return {

"statusCode":200,

"body":json.dumps("Welcome to Python Lambda Boto3")

}

> Compression Required

# zip lambda.zip handler.py

## Practical 17: Create Zip using Python

import shutil

import pathlib

dir\_name="."

dir\_name=pathlib.Path(\_\_file\_\_).parent.resolve()

output\_file="lambda"

shutil.make\_archive(output\_file,"zip",dir\_name)

## Practical 18: Lambda Upload Create\_function

import boto3

iam\_client=boto3.client("iam")

lambda\_client=boto3.client("lambda")

print("Client Object Prepared")

zipped\_code=None

with open("lambda.zip","rb") as f:

zipped\_code=f.read()

print("ZippedCode Object Prepared")

iam\_role=iam\_client.get\_role(RoleName="rl\_serverless\_mujahed")

print("IAM Role Selected")

response=lambda\_client.create\_function(

FunctionName="fnmujahedlambda",

Runtime="python3.9",

Role=iam\_role["Role"]["Arn"],

Handler="handler.lambda\_handler",

Code=dict(ZipFile=zipped\_code),

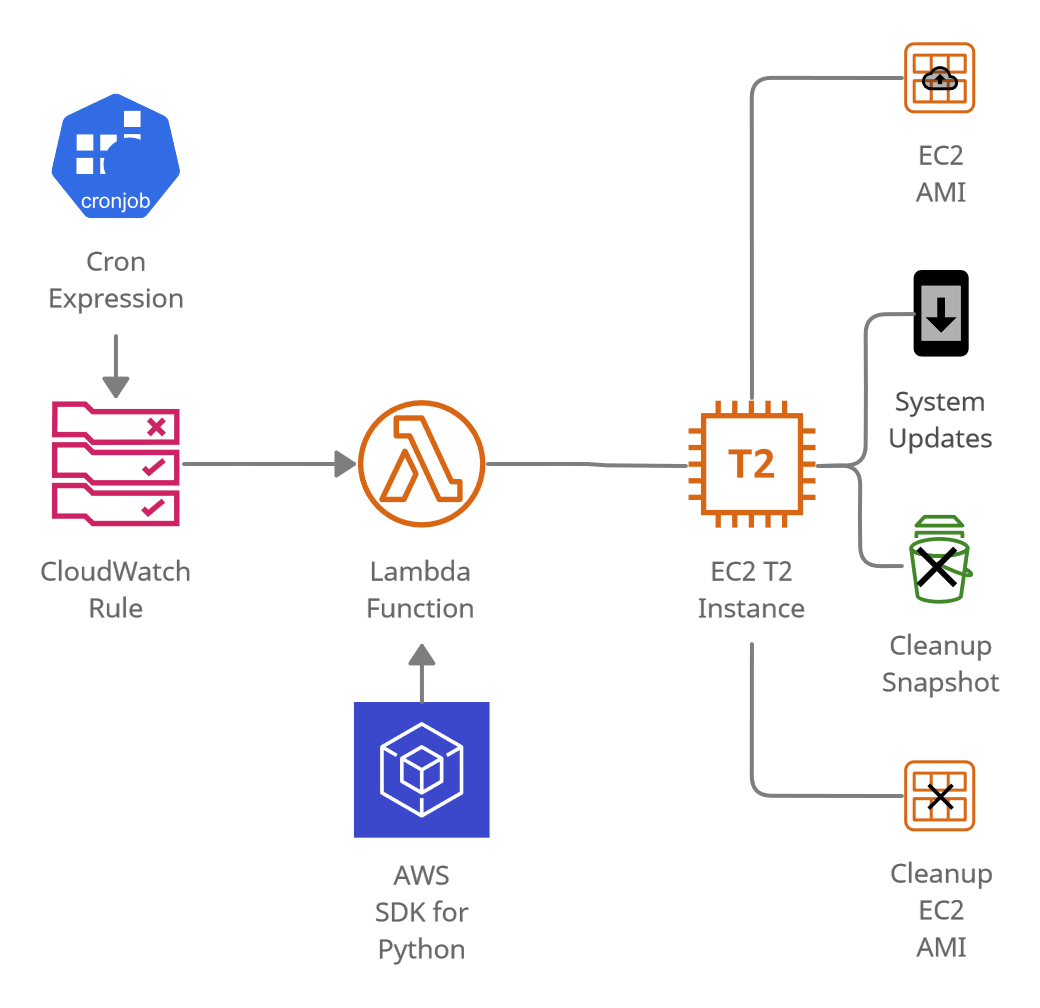
Timeout=300

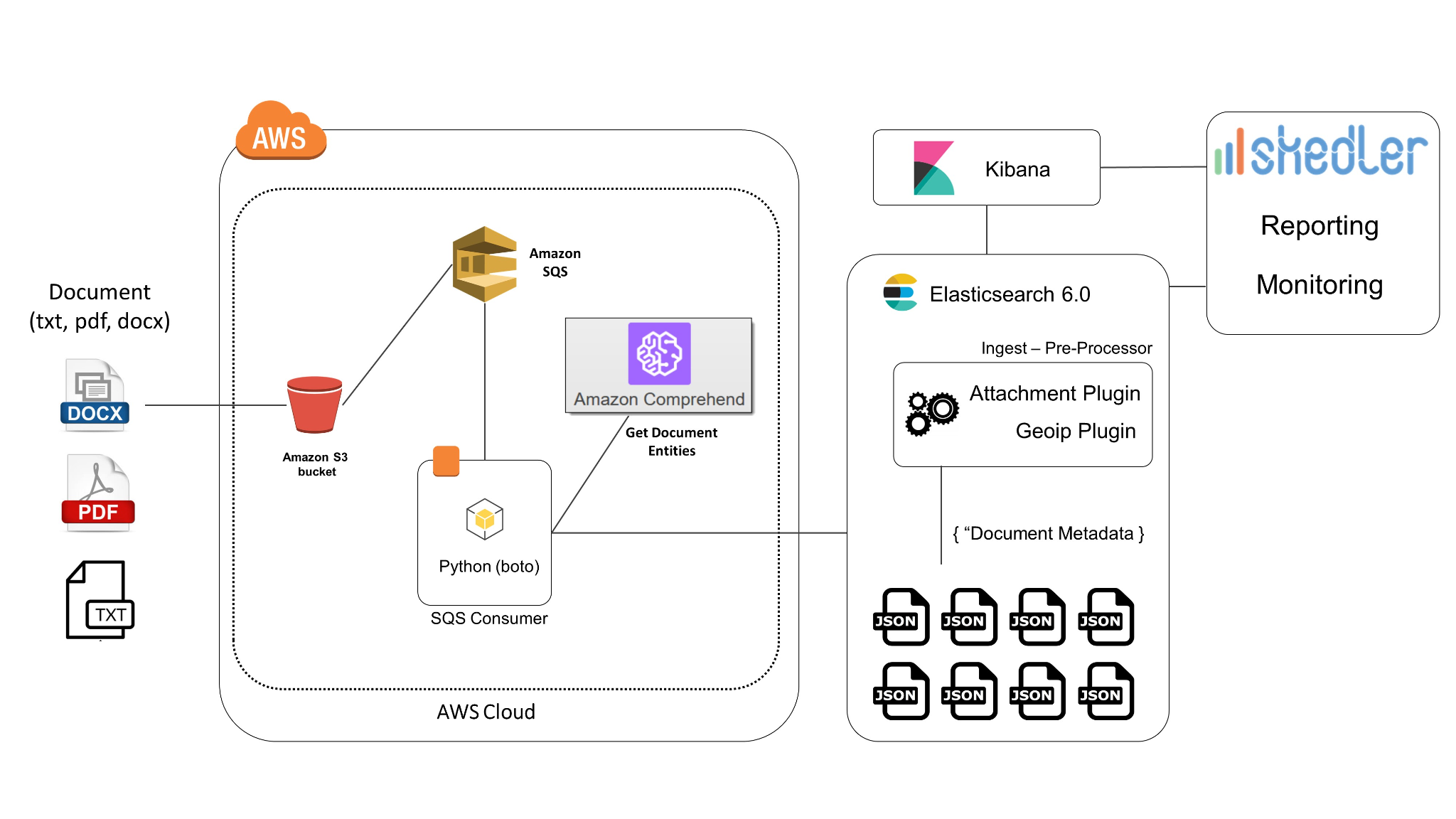
)

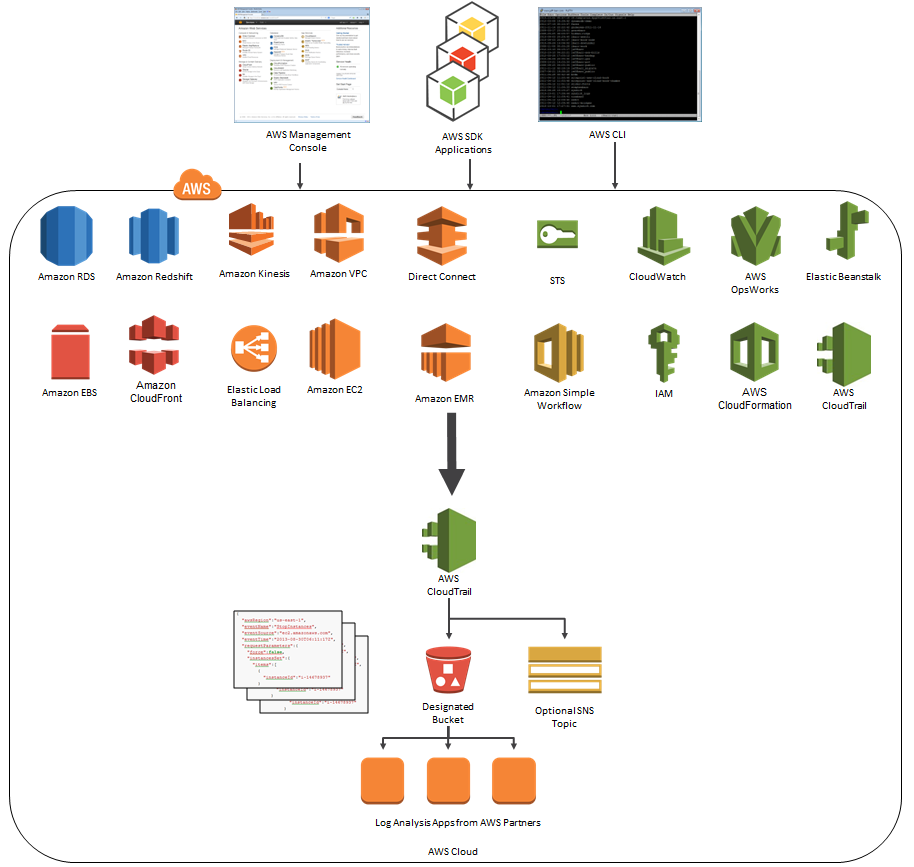
print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

print(response)

1. Requirements
   1. Small Task(Cleaning, Handling, Orchestration)
2. Best Solutions
   1. Cost
      1. Run → %temp% temp prefetch
   2. Performance
   3. Scalability







Function As A Service

Lambda

Boto3 S3, IAM Roles

EC2, EBS, Volumes

RDS DBInstance, Snapshot, Backups, List, Create,

DynamoDB, CloudWatch

AWS Functions

Architecture Implementations

S3 → DynamoDB, Lambda, CloudWatch, IAM Role, Policy

Event Source Mapping

Invocation

Components of Lambda

## Project 20: Creating Project for Event Driven Programming

### File1: handler.py

import json

import os

import logging

import boto3

LOGGER=logging.getLogger()

LOGGER.setLevel(logging.INFO)

AWS\_REGION=os.environ.get("AWS\_REGION","ap-south-1")

DB\_TABLE\_NAME=os.environ.get("DB\_TABLE\_NAME","MujahedEmployees")

S3\_CLIENT=boto3.client("s3")

DYNAMODB\_CLIENT=boto3.resource("dynamodb",region\_name=AWS\_REGION)

DYNAMODB\_TABLE=DYNAMODB\_CLIENT.Table(DB\_TABLE\_NAME)

def get\_data\_from\_file(bucketName,keyName):

response=S3\_CLIENT.get\_object(Bucket=bucketName,Key=keyName)

content=response["Body"]

data=json.loads(content.read())

LOGGER.info("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

LOGGER.info("%s/%s file content: %s",bucketName,keyName,data)

return data

def save\_data\_to\_db(item):

result=DYNAMODB\_TABLE.put\_item(Item=item)

return result

def lambda\_handler(event,context):

for r in event["Records"]:

s3\_bucket\_name=r["s3"]["bucket"]["name"]

s3\_file=r["s3"]["object"]["key"]

data=get\_data\_from\_file(s3\_bucket\_name,s3\_file)

for i in data:

save\_data\_to\_db(i)

return {

"statusCode":200,

"body":json.dumps("We took Uploaded Object From s3 and Inserted into DynamoDB")

}

return {

"statusCode":200,

"body":json.dumps("Welcome to Python Lambda Boto3")

}

### File 2: createZip.py

import shutil

import pathlib

dir\_name=pathlib.Path(\_\_file\_\_).parent.resolve()

output\_file="lambda"

shutil.make\_archive(output\_file,"zip",dir\_name)

print("Created lambda.ZIP based on %s Folder ",dir\_name)

### File 3: Upload **lambda.zip** file into newly created lambda function

import boto3

iam\_client=boto3.client("iam")

lambda\_client=boto3.client("lambda")

**print("Client Object Prepared")**

zipped\_code=None

with open("lambda.zip","rb") as f:

zipped\_code=f.read()

**print("ZippedCode Object Prepared")**

iam\_role=iam\_client.get\_role(RoleName="rl\_serverless\_mujahed")

**print("IAM Role Selected")**

response=lambda\_client.create\_function(

FunctionName="fnmujahedlambdav2",

Runtime="python3.9",

Role=iam\_role["Role"]["Arn"],

Handler="handler.lambda\_handler",

Code=dict(ZipFile=zipped\_code),

Timeout=300

)

**print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")**

print(response)

mujahed.trainer@gmail.com

### File 04: Invoke Lambda

import boto3,json

lambda\_client=boto3.client("lambda")

test\_event=dict()

FuncName="fnmujahedv0"

response=lambda\_client.invoke(

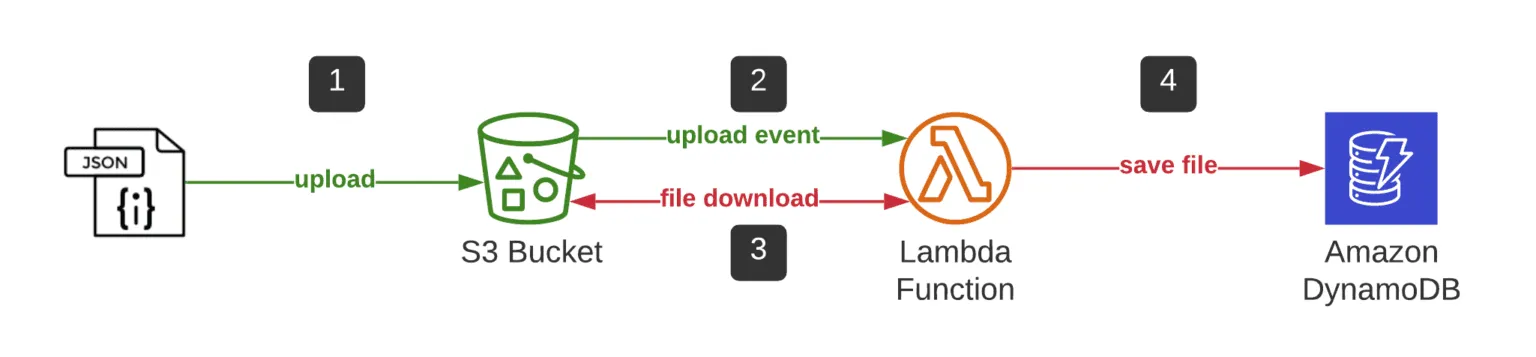
FunctionName=FuncName,

Payload=json.dumps(test\_event)

)

print(response["Payload"])

print(response["Payload"].read().decode("utf-8"))



Live = Streaming(Kinesis) → Lambda(Enc, Raw → Info)→ S3

EDA