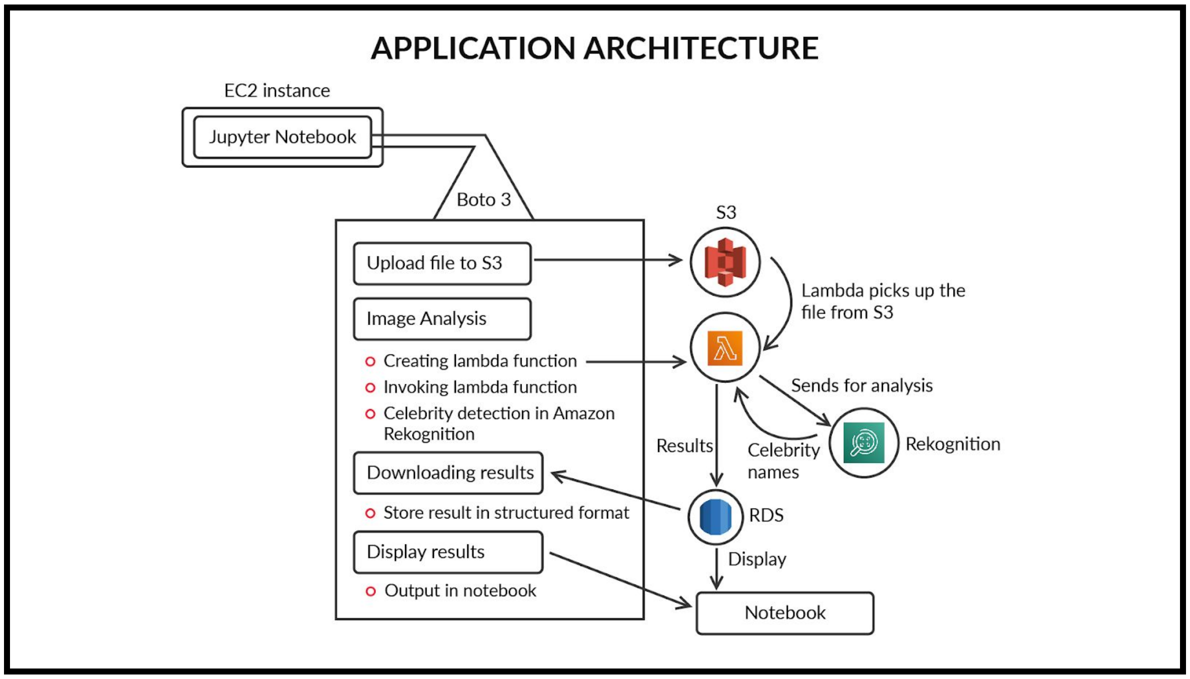
AWS CASE STUDY

You will be working with the following services to build an application to recognise celebrities in an image using AWS services:

Amazon S3: To store images
Amazon EC2: To host the application/Jupyter Notebook
Amazon Rekognition: To detect celebrities
Amazon RDS: To store the results in a database
AWS Lambda: To call AWS services in the notebook



Creating the role for an EC2 instance

1. Viewing the role policy document ***test_role.json***. This will help to attach different policies to the underlying EC2 instance.

```
For Mac users :        vi test_role.json
For windows users :    Text editor (preferably Notepad++)
```

2. Creating a role ***csdrole*** with the role policy document:

```
aws iam create-role  --role-name csdrole --asume-role-policy-document
file://test_role.json
```

3. Attaching the S3 and Lambda full access policies to the created role

```
aws iam attach-role-policy --role-name csdrole --policy-arn
arn:aw:iam::aws:policy/AmazonS3FullAccess

aws iam attach-role-policy --role-name csdrole --policy-arn
arn:aw:iam::aws:policy/AWSLambdaFullAccess
```

4. An instance profile is used to attach a role to the EC2 instance. To create an instance profile, you can run the following command:

```
aws iam create-instance-profile --instance-profile-name csdinstanceprofile
```

5. Adding role ***csdrole*** to the instance profile ***csdinstanceprofile*** (created above)

```
aws iam add-role-to-instance-profile --role-name csdrole
--instance-profile-name csdinstanceprofile
```

Creating the Security group and Access key pair

6. Creating security group ***csdjupyteracess*** to provide access for SSH and Jupyter

```
aws ec2 create-security-group --group-name csdjupyteraccess --description
“Access for jupyter and ssh”
```

Note: Copy the security group ID as it will be used in the later steps

7. Authorizing the access of ports 22 and 8888 through local IP address for SSH and Jupyter respectively

```
aws ec2 authorize-security-group-ingress --group-id <copied previously>
--protocol tcp --port 22 --cidr <your IP>

aws ec2 authorize-security-group-ingress --group-id <copied previously>
--protocol tcp --port 8888 --cidr <your IP>
```

8. Creating an Access key pair for the account (Skip this step if you have a previously created key pair as only 2 keys are allowed per account.)

```
aws ec2 create-key-pair --key-name csd_pair --query “KeyMaterial” --output
text > csd_pair.pem
```

   You can view the file using VI editor in Linux/Mac or Notepad++ in windows.

Creating the EC2 instance with the above resources

9. Now you are going to create an EC2 instance with the resources - Role, Security group and Keypair. You will be using Amazon Linux OS on a t2.micro instance.

```
aws ec2 run-instances --image-id <ami ID of latest version of amazon linux>
--count 1 --instance-type t2.micro --key-name csd_pair --security-group-ids
<sg id from above> --iam-instance-profile Name=csdinstanceprofile
```

   Note the instance id here, it will be used later

10. You can describe the instance using,

```
aws ec2 describe-instances --instance-id <instance id which was copied>
```

   Note the public DNS and the private IP address which will be used later.

11. Login into the instance

```
MAC users :   ssh -i “csd_pair.pem”  ec2-user@< public DNS of instance>
Windows:      Use PuTTY
```

Creating the RDS instance

12. Creating a MySQL database named ***celebrities*** through CLI. Do not use the web console as the steps may result in some differences than the process followed.

```
aws rds create-db-instance --engine mysql --db-name celebrities
--db-instance-class db.t2.micro --allocated-storage 20 --db-instance-identifier
test-instance --master-username <your_username>  --master-user-password
<your_password>
```

Note the database name, username and password as they will be required later.

13. Checking if the database instance is created

```
aws rds --describe-db-instances
```

Note the endpoint address as you will be using this to connect to the database
Also, note down the port and the Security group ID of the RDS instance

14. Authorising the EC2 and RDS connection by adding EC2 instance IP to the RDS Security group as we will be accessing the SQL database from EC2 instance.

```
aws ec2 authorize-security-group-ingress --group-id <db instance security
group> --protocol tcp --port 3306 --cidr <ec2 instance ip>
```

Creating role for AWS Lambda

15. Creating the role document for AWS Lambda - ***lambda_role.json***

```
{
"Version": "2012-10-17",
"Statement": [
    {
    "Effect": "Allow",
    "Principal": {
        "Service": "lambda.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {}
    }
  ]
}
```

16. Creating a role ***csdlambdarole*** with the role policy document:

```
aws iam create-role  --role-name csdlambdarole --asume-role-policy-document
file://lambda_role.json
```

17. Attaching the S3 and Amazon Rekognition full access policies to the created role

```
aws iam attach-role-policy --role-name csdlambdarole --policy-arn
arn:aw:iam::aws:policy/AmazonS3FullAccess

aws iam attach-role-policy --role-name csdlambdarole --policy-arn
arn:aw:iam::aws:policy/AmazonRekognitionFullAccess
```

18. Fetching the ARN for the created role ***csdlambdarole***

```
aws iam get-role csdlambdarole
```

Note the ARN for the role as it will be used later.

Uploading the file to S3

You must install Jupyter on the instance before proceeding with the next steps.

19. Running Jupyter Notebook on the EC2 instance - Refer to the documentation provided before.

20. Download the code file '***celebrity_recognition***' and the image from the upGrad platform and upload it on Jupyter. You will be running the commands in the Jupyter Notebook now.

21. Install Boto3 (if not done already)

```
!pip3 install boto3 --user
```

22. Importing the Boto3 library

```
# Importing boto3 library to access AWS services in Python
import boto3
```

23. Listing the S3 buckets:

```
# Initiating the S3 client
s3 = boto3.client('s3')

# Printing the S3 buckets present in the AWS account
response = s3.list_bucktes()
for bucket in response[‘Buckets’]:
        print(bucket[‘Name’])
```

24. Uploading the image

```
# Specifying the attributes for file upload
file_name = 'bezos_demo_image.jpg'
bucket = '<bucket_name>'
object_name = 'bezos_demo_image.jpg'

# Initiating the S3 client
s3 = boto3.client('s3')

# Uploading the file on S3
s3.upload_file(file_name, bucket, object_name)

# Objects in the selected bucket
response = s3.list_objects(Bucket = ‘<bucket_name>')

# Printing the bucket names
for objects in response['Contents']:
        print(objects[‘Key’])
```

Image analysis - Watch each step in the video carefully

25. Under the Jupyter instance, create and store ***handler.py*** with the following code in the ***lambda*** folder:

```
import boto3
import json
```

```
def lambda_handler(event, context) :
        client = boto3.client('rekognition’)
        response = client.recognize_celebrities(
                Image={
                                ‘S3Object’ : {
                                        ‘Bucket’ =  event [‘bucket’],
                                        ‘Name’: event[‘object name']
                                }
                        }
        )
        return (response[ ‘CelebrityFaces’][0][‘Name’])
```

26. Install boto3 in the target folder ***lambda***

```
!pip3 install --target ./lambda/boto3 boto3
```

27. Zip the files using the command given below:

```
%cd lambda
!zip -r9 ../celebrity_detector.zip .
```

After zipping the files, return to the main folder:

```
%cd ..
```

28. Creating the function ***celebrity_detector***

```
# Creating the function
!aws lambda create-function \
--function-name celebrity_detector \
--runtime python3.6 \
--zip-file fileb://celebrity_detector.zip \
--handler handler.lambda_handler \
--role <ARN for csdlambdarole>
--region us-east-1
```

29. Invoking the function ***celebrity_detector*** and storing the results in a variable ***celebrity_name***

```
# Invoking the function
lam = boto3.client('lambda')

response = lam.invoke(
FunctionName = 'celebrity_detector',
InvocationType = 'RequestResponse',
LogType =' Tail',
Payload = json.dumps({"bucket": "<bucket_name>", "object_name":
"bezos_demo_image.jpg"})
)

# Storing and printing the results in a variable
celebrity_name = response['Payload'].read().decode('utf8')
print(celebrity_name)
```

Storing Results

30. Installing and importing the MySQL connector library to connect with the database through the notebook

```
# Installing the connector
!pip install mysql-connector

# Importing the library
import mysql.connector
```

31. Extracting and storing the obtained results in the RDS database named ***celebrities*** that was created previously

```
# Connector object
dbc = mysql.connector.connect(
        host = "<endpoint>",
        user = "<username>",
        passwd = "<password>",
        port = 3306,
        database = "<database name>"
)
```

```
# Defining the cursor object
cur = dbc.cursor()

# Creating the table
cur.execute("CREATE TABLE celebrities (image VARCHAR(255), celebrity
VARCHAR(255))")

# Inserting the values into columns
sql = "INSERT INTO celebrities(image, celebrity) VALUES (%s, %s)"
val = (file_name, celebrity_name)
cur.execute(sql, val)

# Read the stored values from RDS
cur.execute("SELECT image, celebrity FROM celebrities")
result = cur.fetchall()

# Displaying the results
print(result)
```