

Cloud Computing Fundamentals

A

Project Report on

“AI Resume Parser”

Submitted for Evaluation

By

Hardik Lal - 500120516

Batch: 4 (CCVT)

Purab Sharma - 500121924

Batch: 3 (CCVT)

To

Prof. Abhirup Khanna



SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Table of Contents

Serial No	Content
1	Problem Statement
2	Technologies Used
3	Users of the System
4	Functional Modules
5	Codebase Explanation & Code
6	Challenges & Learnings
7	Concept Used
8	Conclusion

AI Resume Parser – Project Report

1. Problem Statement

Develop a command-line based resume parser using AWS services. The goal was to automatically extract important details from resumes—like name, email, phone number, and skills—and store them in a structured format. This would help automate part of the hiring process and save time.

2. Technologies Used

To bring this idea to life, I used a mix of programming and cloud technologies:

- **Python** – for writing the resume parsing logic and handling file uploads.
- **AWS EC2** – used as a remote server for running and testing the application.
- **AWS S3** – to store uploaded resumes securely.
- **AWS Lambda** – for automating the resume parsing process without having to keep a server running.
- **AWS IAM** – to manage secure access and permissions for different components.
- **Libraries like re, boto3, and pdfminer.six** – for text extraction and cloud interaction.

3. Users of the System

- **Admin:** Manages AWS resources, configures IAM roles, monitors system logs, and ensures secure cloud operations.
- **Recruiter/HR:** Uploads resumes, reviews parsed data, and uses extracted information for candidate shortlisting.
- **Developer:** Develops and maintains parsing logic, updates Lambda functions, and adds new features like file format support or ML integration.

4. Functional Modules

I divided the project into clear, manageable parts:

1. **Resume Uploading** – A CLI command allows the user to upload resumes (PDF files) directly to an S3 bucket.
2. **Automatic Parsing** – As soon as a file lands in the bucket, a Lambda function is triggered to start the parsing process.
3. **Information Extraction** – The Lambda function picks out useful data like the candidate's name, email, phone number, skills, etc.
4. **Secure Access Control** – Using IAM roles to make sure only the right services and users can access specific parts of the system.

5. Codebase Explanation & Code

The Explanation of each concept that we used to power our project:

I.

Connecting AWS user with Command Line

```
C:\Users\Hardik Lal>aws configure
AWS Access Key ID [*****LY46]:
AWS Secret Access Key [*****sB8t]:
Default region name [us-east-1]:
Default output format [json]:
```

II.

Creating role

```
C:\Users\Hardik Lal>aws iam create-role ^
More? --role-name LambdaResumeParserRole ^
More? --assume-role-policy-document file://"C:\Users\Hardik Lal\OneDrive - UPES\Desktop\trust-policy.json"
{
  "Role": {
    "Path": "/",
    "RoleName": "LambdaResumeParserRole",
    "RoleId": "AROAWB63662WMVCI3LYSQ",
    "Arn": "arn:aws:iam::416536262316:role/LambdaResumeParserRole",
    "CreateDate": "2025-05-05T15:09:11+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "lambda.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

III.

Attach policies to it

```
C:\Users\Hardik Lal>
C:\Users\Hardik Lal>aws iam attach-role-policy ^
More? --role-name LambdaResumeParserRole ^
More? --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

C:\Users\Hardik Lal>
C:\Users\Hardik Lal>aws iam attach-role-policy ^
More? --role-name LambdaResumeParserRole ^
More? --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess

C:\Users\Hardik Lal>aws iam list-attached-role-policies --role-name LambdaResumeParserRole
{
  "AttachedPolicies": [
    {
      "PolicyName": "AWSLambdaBasicExecutionRole",
      "PolicyArn": "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
    },
    {
      "PolicyName": "AmazonS3FullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
    }
  ]
}
```

IV.

Create an S3 Bucket

```
C:\Users\Hardik Lal>aws s3api create-bucket ^
More? --bucket resume-parser-hardik-2025 ^
More? --region us-east-1
{
  "Location": "/resume-parser-hardik-2025"
}

C:\Users\Hardik Lal>aws s3 ls
2025-05-05 20:45:51 resume-parser-hardik-2025
```

V.

Attach Additional Policies

```
C:\Users\Hardik Lal>aws iam attach-role-policy --role-name LambdaResumeParserRole --policy-arn arn:aws:iam::aws:policy/AmazonTextractFullAccess
C:\Users\Hardik Lal>aws iam attach-role-policy --role-name LambdaResumeParserRole --policy-arn arn:aws:iam::aws:policy/AmazonComprehendFullAccess
An error occurred (NoSuchEntity) when calling the AttachRolePolicy operation: Policy arn:aws:iam::aws:policy/AmazonComprehendFullAccess does not exist or is not attachable.

C:\Users\Hardik Lal>aws iam attach-role-policy ^
More? --role-name LambdaResumeParserRole ^
More? --policy-arn arn:aws:iam::aws:policy/ComprehendFullAccess

C:\Users\Hardik Lal>aws iam list-attached-role-policies --role-name LambdaResumeParserRole
{
  "AttachedPolicies": [
    {
      "PolicyName": "ComprehendFullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/ComprehendFullAccess"
    },
    {
      "PolicyName": "AmazonTextractFullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonTextractFullAccess"
    },
    {
      "PolicyName": "AWSLambdaBasicExecutionRole",
      "PolicyArn": "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
    },
    {
      "PolicyName": "AmazonS3FullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
    }
  ]
}
```

VI.

Create a folder for the lambda function

```
C:\Users\Hardik Lal>mkdir C:\lambda\ResumeSkillMatcher
C:\Users\Hardik Lal>cd C:\lambda\ResumeSkillMatcher
```

VII.

Content for lambda function

```
import os
import boto3

s3 = boto3.client('s3')
textract = boto3.client('textract')
comprehend = boto3.client('comprehend')

REQUIRED_SKILLS = os.environ['REQUIRED_SKILLS'].split(',')




def lambda_handler(event, context):
    record = event['Records'][0]['s3']
    bucket = record['bucket']['name']
    key = record['object']['key']

    # 1. Download & extract text
    tmp = '/tmp/input'
    s3.download_file(bucket, key, tmp)
    with open(tmp, 'rb') as f:
        doc = f.read()
    tex_resp = textract.detect_document_text(Document={'Bytes': doc})
    text = ' '.join([item['DetectedText'] for item in tex_resp['Blocks'] if
item['BlockType']=='LINE'])

    # 2. Detect key phrases
    comp = comprehend.detect_key_phrases(Text=text, LanguageCode='en')
    phrases = [p['Text'].lower() for p in comp['KeyPhrases']]
    # 3. Match skills
    matched = [skill for skill in REQUIRED_SKILLS if skill.lower() in phrases]
    if matched:
        print(f"{key} matched skills: {matched}")
        # e.g., tag the object or copy to a folder
        s3.put_object_tagging(
            Bucket=bucket,
            Key=key,
            Tagging={'TagSet': [{'Key': 'Matched', 'Value': ','.join(matched)}}} )
    else:
        print(f"{key} had no matches.")
```

VIII.

Zip the Code

 ResumeSkillMatcher	06-05-2025 04:35 AM	File folder	
 SortedResumes	06-05-2025 03:46 AM	File folder	
 resume_parser.zip	05-05-2025 09:12 PM	Compressed (zipped)...	1 KB

IX.

Create the Lambda Function in AWS with our AWS account ID

```
C:\lambda\ResumeSkillMatcher>aws lambda create-function --function-name ResumeSkillMatcher --runtime python3.9 --role arn:aws:iam::416536262316:role/LambdaResumeParserRole --handler lambda_function.lambda_handler --zip-file fileb://C:/lambda/resume_parser.zip --environment "{\"Variables\":{\"REQUIRED_SKILLS\":\"Python,AWS,Machine Learning\"}}"
```

```
{
  "FunctionName": "ResumeSkillMatcher",
  "FunctionArn": "arn:aws:lambda:us-east-1:416536262316:function:ResumeSkillMatcher",
  "Runtime": "python3.9",
  "Role": "arn:aws:iam::416536262316:role/LambdaResumeParserRole",
  "Handler": "lambda_function.lambda_handler",
  "CodeSize": 858,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2025-05-05T15:59:14.748+0000",
  "CodeSha256": "HQQ5JWtQeG9v1WLaMun3jpN1aVG8w9tt9Cnx6wpHHEA=",
  "Version": "$LATEST",
  "Environment": {
    "Variables": {
      "REQUIRED_SKILLS": "Python,AWS,Machine Learning"
    }
  },
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "462ceaab-7711-4165-9367-9146e1bd232b",
  "State": "Pending",
  "StateReason": "The function is being created.",
  "StateReasonCode": "Creating",
  "PackageType": "Zip",
  "Architecture": "x86_64",
  "EphemeralStorage": {
    "Size": 512
  },
  "SnapStart": {
    "ApplyOn": "None",
    "OptimizationStatus": "Off"
  },
  "RuntimeVersionConfig": {
    "RuntimeVersionArn": "arn:aws:lambda:us-east-1::runtime:c8c62727548b4b718883e395881b633842dceea97a5c4914a1258f41c91d3b7"
  },
  "LoggingConfig": {
    "LogFormat": "Text",
    "LogGroup": "/aws/lambda/ResumeSkillMatcher"
  }
}
```

X.

Setup our S3 bucket to trigger the Lambda Function when a new file is uploaded

```
C:\lambda\ResumeSkillMatcher>aws lambda add-permission ^
More? --function-name ResumeSkillMatcher ^
More? --statement-id S3InvokePermission ^
More? --action lambda:InvokeFunction ^
More? --principal s3.amazonaws.com ^
More? --source-arn arn:aws:s3:::resume-parser-hardik-2025
```

```
{
  "Statement": "{ \"Sid\": \"S3InvokePermission\", \"Effect\": \"Allow\", \"Principal\": { \"Service\": \"s3.amazonaws.com\" }, \"Action\": \"lambda:InvokeFunction\", \"Resource\": \"arn:aws:lambda:us-east-1:416536262316:function:ResumeSkillMatcher\", \"Condition\": { \"ArnLike\": { \"AWS:SourceArn\": \"arn:aws:s3:::resume-parser-hardik-2025\" } } }"
```


XI.

Configure the S3 bucket to trigger the Lambda Function

```
C:\lambda\ResumeSkillMatcher>aws s3api put-bucket-notification-configuration ^
More? --bucket resume-parser-hardik-2025 ^
More? --notification-configuration ^
More? "[{"LambdaFunctionConfigurations":[{"LambdaFunctionArn":"arn:aws:lambda:us-east-1:416536262316:function:ResumeSkillMatcher"},"Events":["s3:ObjectCreated:*"]}]]"
```

XII.

Upload the Resume to test

```
C:\lambda\ResumeSkillMatcher>aws s3 cp "C:\Users\Hardik Lal\OneDrive - UPES\Desktop\Hardik_Lal_2024_Resume.pdf" s3://resume-parser-hardik-2025/
upload: ..\..\Users\Hardik Lal\OneDrive - UPES\Desktop\Hardik_Lal_2024_Resume.pdf to s3://resume-parser-hardik-2025/Hardik_Lal_2024_Resume.pdf

C:\lambda\ResumeSkillMatcher>aws s3 cp "C:\Users\Hardik Lal\OneDrive - UPES\Desktop\HarshilMittal_2024_Resume.pdf" s3://resume-parser-hardik-2025/
upload: ..\..\Users\Hardik Lal\OneDrive - UPES\Desktop\HarshilMittal_2024_Resume.pdf to s3://resume-parser-hardik-2025/HarshilMittal_2024_Resume.pdf
```

```
PS C:\lambda\ResumeSkillMatcher> aws s3 cp "C:\Users\Hardik Lal\OneDrive - UPES\Desktop\PurabSharma_2024_Resume (2).pdf" s3://resume-parser-hardik-2025/
upload: ..\..\Users\Hardik Lal\OneDrive - UPES\Desktop\PurabSharma_2024_Resume (2).pdf to s3://resume-parser-hardik-2025/PurabSharma_2024_Resume (2).pdf
```

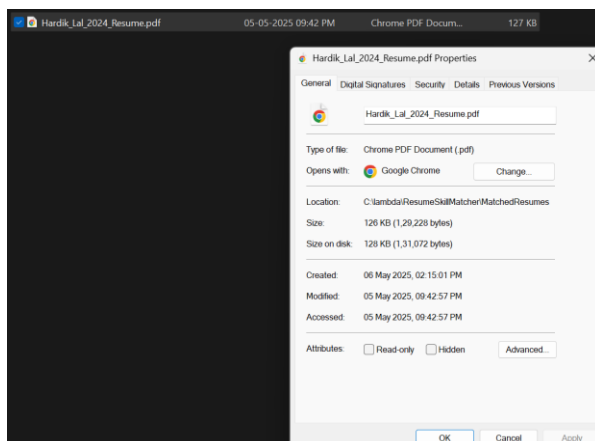
XIII.

Test with a skill in Windows PowerShell

```
PS C:\lambda\ResumeSkillMatcher> .\List-MatchedResumes.ps1 -Skill Python
Searching for resumes with skill: 'Python'
Match found: Hardik_Lal_2024_Resume.pdf
Matching resumes downloaded to folder: MatchedResumes
Done.
PS C:\lambda\ResumeSkillMatcher> |
```

XIV.

The Matched Resume that has the desired input (We took python as a skill) will automatically save in a folder named Matched Resume)



6. Challenges & Learnings

Challenges I Faced:

- Understanding how AWS Lambda connects with S3 and IAM took some time.
- Regex extraction wasn't always accurate due to the different resume formats and writing styles.
- Testing Lambda functions locally was a bit tricky, especially with permissions.

What I Learned:

- How to work with multiple AWS services together in a real-world use case.
- Practical experience in setting up IAM roles and understanding security best practices.
- Improved my Python skills, especially with libraries for file handling and text extraction.

7. Concepts Used

- **Regex (Regular Expressions)** – To extract text patterns like emails, phone numbers and Skills of a person.
- **AWS Cloud Services** – Used S3, EC2, IAM, and Lambda for cloud-based automation.
- **CLI Development** – Got experience building command-line tools in Python.
- **Automation** – Triggering tasks based on events (file upload) using cloud functions.

8. Conclusion

Building this AI Resume Parser project was a hands-on experience in combining cloud technology with automation. I was able to apply core programming concepts along with cloud services to solve a real-world problem. The system currently works well for parsing resumes in PDF format and can be extended in many ways—like supporting more file types, storing data in databases, or even adding machine learning to score resumes. It helped me understand both the power and responsibility that comes with cloud computing and automation.

Submitted By:

Name: Hardik Lal

SAP ID: 500120516

Enrolment No: R2142230372

Batch: 4 (CCVT)

Name: Purab Sharma

SAP ID: 500121924

Enrolment No: R2142230329

Batch: 3 (CCVT)