

## Use case

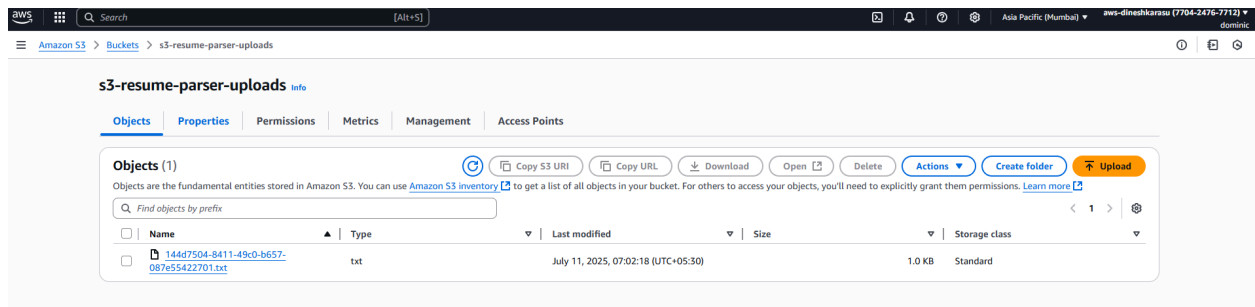
### Mini-project: Resume Parser

## Use case Description

Upload resume file and parse it then store it in db

## Approach :

1. Create S3 bucket to store uploaded resume files
2. Create a dynamo db table to store parsed data
3. Create a Lambda function to process the uploaded file and parse it then store it in db
4. Create API Gateway POST method with CORS enabled
5. Code front end and JS to hit the API from the end user screen
6. Provide Lambda with proper IAM permissions
7. Test the flow



aws [Search] [Alt+S] Asia Pacific (Mumbai) aws-dineshkarasu (7704-2476-7712) dominic

Lambda > Functions > Lambda\_Resume\_Parser

### Lambda\_Resume\_Parser

Throttle Copy ARN Actions

Export to Infrastructure Composer Download

**Function overview** info

Diagram Template

Lambda\_Resume\_Parser

Layers (0)

API Gateway

+ Add trigger

+ Add destination

Description

Last modified 6 minutes ago

Function ARN arn:aws:lambda:ap-south-1:770424767712:function:Lambda\_Resume\_Parser

Function URL info

Code Test Monitor Configuration Aliases Versions

**Code source** info

Upload from

EXPLORER

LAMBDA\_RESUME\_PARSER

lambda\_function.py

```
1 import boto3
2 import uuid
3 import re
4 import os
5 from datetime import datetime
```

aws [Search] [Alt+S] Asia Pacific (Mumbai) aws-dineshkarasu (7704-2476-7712) dominic

Amazon S3 > Buckets > s3-static-website-host-114

Disabled

**Object Lock**

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. [Learn more](#)

Object Lock Disabled

**Requester pays**

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

Requester pays Disabled

**Static website hosting**

Use this bucket to host a website or redirect requests. [Learn more](#)

**We recommend using AWS Amplify Hosting for static website hosting**

Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. [Learn more about Amplify Hosting](#) or [View your existing Amplify apps](#)

Create Amplify app

**S3 static website hosting**

Enabled

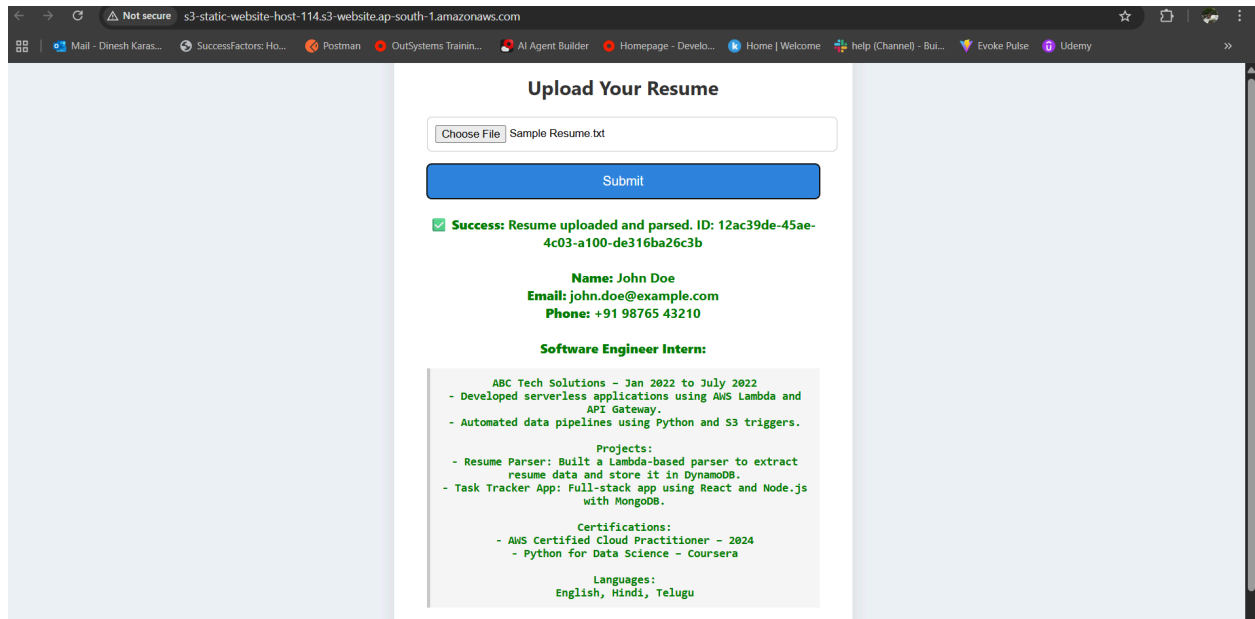
**Hosting type**

Bucket hosting

**Bucket website endpoint**

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://s3-static-website-host-114.s3-website-ap-south-1.amazonaws.com>



**DynamoDB** > **Explore items** > **DC\_Resume\_Uploads**

**DynamoDB**

- Dashboard
- Tables
- Explore items**
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations [New](#)
- Reserved capacity
- Settings

▼ **DAX**

- Clusters
- Subnet groups
- Parameter groups
- Events

**Tables (1)**

Any tag key

Any tag value

Find tables

DC\_Resume\_Uploads

**DC\_Resume\_Uploads**

Autopreview View table details

▼ **Scan or query items**

Scan Query

Select a table or index: Table - DC\_Resume\_Uploads

Select attribute projection: All attributes

► **Filters - optional**

Run Reset

Completed · Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2

**Table: DC\_Resume\_Uploads - Items returned (1)**

Scan started on July 11, 2025, 07:02:22

	ResumeId (String)	email	name	phone	s3_key	uploaded_at
	144d7504-8411-49c0...	john.doe@...	John Doe	+91 98765 ...	144d7504-...	2025-07-11T01:32:17.085541

**API Gateway** > **APIs** > **Resources - Resume\_Parser (22rb9j5mj)**

**API Gateway**

- APIs
- Custom domain names
- Domain name access associations
- VPC links

▼ **API: Resume\_Parser**

- Resources**
- Stages
- Authorizers
- Gateway responses
- Models
- Resource policy
- Documentation
- Dashboard
- API settings

Successfully enabled CORS

**Resources**

Create resource

/

/main

OPTIONS

POST

**Resource details**

Path: /main

Resource ID: jsulku

API actions Deploy API

Delete Update documentation Enable CORS

**Methods (2)**

Method type	Integration type	Authorization	API key
OPTIONS	Mock	None	Not required
POST	Lambda	None	Not required

**CloudWatch** > **Log groups** > **/aws/lambda/Lambda\_Resume\_Parser** > **2025/07/10/[SLATEST]jd0512baec144049d7af88e084b3135**

**CloudWatch**

- Dashboards
- AI Operations [New](#)**
- Alarms** [🔔](#) [🔔](#) [🔔](#) [🔔](#)
- In alarm
- All alarms
- Logs**
- [Log groups](#)
- Log Anomalies
- Live Tail
- Logs Insights
- Contributor Insights
- Metrics**
- All metrics
- Explorer
- Streams
- Application Signals (APM)**
- Network Monitoring**
- Insights**
- Settings
- Telemetry config

**Log events**

You can use the filter bar below to search for and match terms, phrases, or values in your log events. [Learn more about filter patterns](#)

Filter events - press enter to search

Clear 1m 30m 1h 12h Custom UTC timezone Display

Timestamp	Message
2025-07-11T01:32:16.380Z	INIT_START Runtime Version: python:3.12.073 Runtime Version ARN: arn:aws:lambda:ap-south-1:runTime320820742c3122a6cc30766022c7c201f98f98406a552404939a1038c05
2025-07-11T01:32:16.030Z	Initializing variables
2025-07-11T01:32:16.030Z	BUCKET_NAME: s3-resume-parser-uploads
2025-07-11T01:32:16.030Z	table: dynamodb.Table(name="DC_resume_uploads")
2025-07-11T01:32:16.034Z	START RequestId: 52263619-64c4-4302-b636-40b9a2e3bd0c Version: SLATEST
2025-07-11T01:32:16.035Z	decoding file from base64
2025-07-11T01:32:16.035Z	body: b'.....web1t0rm0ub0nduryfV4Bt2agVLSAHXUv/vnContent-Disposition: form-data; name="file"; filename="Sample Resume.txt"/v/vnContent-Type: text/plain/v/vnJohn Doe/v(email: john.doe@example.com/v/vnPho...
2025-07-11T01:32:16.035Z	extracting file content from multipart body
2025-07-11T01:32:16.035Z	parsing resume text
2025-07-11T01:32:16.036Z	name: <re.Match object; span=(8, 8), match='John Doe'>
2025-07-11T01:32:16.036Z	email: <re.Match object; span=(17, 37), match='john.doe@example.com'>
2025-07-11T01:32:16.036Z	phone: <re.Match object; span=(46, 62), match='+91 98765 43210/v'>
2025-07-11T01:32:16.025Z	generating unique ID for storage
2025-07-11T01:32:16.025Z	resume_id: 144d7504-8411-49c0-8657-807e55422701
2025-07-11T01:32:16.025Z	filename: 144d7504-8411-49c0-8657-807e55422701.txt
2025-07-11T01:32:17.065Z	uploaded to S3
2025-07-11T01:32:17.065Z	Item: {'ResumeId': '144d7504-8411-49c0-8657-807e55422701', 'name': 'John Doe', 'email': 'john.doe@example.com', 'phone': '+91 98765 43210/v', 'uploaded_at': '2025-07-11T01:32:17.085541', 's3_key': '144d7504-8...
2025-07-11T01:32:17.224Z	saved to DynamoDB
2025-07-11T01:32:17.340Z	END RequestId: 52263619-64c4-4302-b636-40b9a2e3bd0c

Back to top

Since the above approach only works for .txt files but not pdf and docx files as we are using regular expressions, let's go with another approach using python-docx  
We'll keep everything as it is and modify our lambda code. Create a layer then update the lambda code

The screenshot displays the AWS Lambda console interface. On the left, the navigation pane shows the 'Layers' section under 'Additional resources'. The main content area shows the details for a newly created layer named 'pdfplumber\_docx'. The 'Version details' section indicates that version 1 was created 8 minutes ago, is compatible with arm64 and x86\_64 architectures, and uses the python3.12 runtime. Below this, the 'All versions (1)' section shows a single version with its ARN: 'arn:aws:lambda:ap-south-1:770424767712:layer:pdfplumber\_docx:1'. At the bottom, a code editor shows the updated lambda function code in 'lambda\_function.py'. The code imports 'pdfplumber' and 'docx' libraries, sets up boto3 clients for S3 and DynamoDB, and defines functions to parse resume sections and extract fields like name, email, and phone from a document.

```
1 import boto3
2 import uuid
3 import re
4 import os
5 import json
6 import traceback
7 import tempfile
8 from datetime import datetime
9 from base64 import b64decode
10
11 import pdfplumber
12 import docx
13
14 s3 = boto3.client('s3')
15 dynamodb = boto3.resource('dynamodb')
16 table = dynamodb.Table('DC_Resume_Uploads')
17
18 BUCKET_NAME = "s3-resume-parser-uploads"
19
20 def is_section_heading(line):
21     return bool(re.match(r'^[A-Za-z]{2,30}$', line.strip())) and line.strip().istitle()
22
23 def extract_fields(text):
24     lines = text.splitlines()
25     sections = {}
26     current_section = None
27     buffer = []
28
29     for line in lines:
30         line = line.strip()
31         if is_section_heading(line):
32             if current_section and buffer:
33                 sections[current_section] = '\n'.join(buffer).strip()
34                 buffer = []
35             current_section = line
36         elif current_section:
37             buffer.append(line)
38     if current_section and buffer:
39         sections[current_section] = '\n'.join(buffer).strip()
40
41 name = re.search(r'(?i)[A-Z][a-z]{1,30}[A-Z][a-z]{1,}', text)
42 email = re.search(r'[a-zA-Z0-9]{1,30}@[a-zA-Z0-9]{1,30}\.com', text)
43 phone = re.search(r'\d{10}', text)
```

```
lambda_function.py X
lambda_function.py
23 def extract_fields(text):
42     email = re.search(r'[\w\.-]+@[\w\.-]+\.', text)
43     phone = re.search(r'(\+?)(\d{3,14})', text)
44
45     return {
46         'name': name.group() if name else 'N/A',
47         'email': email.group() if email else 'N/A',
48         'phone': phone.group() if phone else 'N/A',
49         'sections': sections
50     }
51
52 def extract_text_from_pdf(file_path):
53     with pdfplumber.open(file_path) as pdf:
54         return "\n".join([page.extract_text() or "" for page in pdf.pages])
55
56 def extract_text_from_docx(file_path):
57     doc = docx.Document(file_path)
58     return "\n".join([para.text for para in doc.paragraphs])
59
60 def lambda_handler(event, context):
61     try:
62         body = event['body']
63         is_base64_encoded = event.get('isBase64Encoded', False)
64         if is_base64_encoded:
65             body = base64decode(body)
66
67         # Extract file content from multipart body
68         file_binary = body.split(b'\r\n\r\n', 1)[1].rsplit(b'\r\n-----', 1)[0]
69
70         # Extract filename from headers
71         header_line = body.split(b'\r\n', 2)[1]
72         filename_match = re.search(b'filename="(.*?)"', header_line)
73         filename = filename_match.group(1).decode() if filename_match else f"{uuid.uuid4()}.txt"
74         ext = filename.lower().split('.')[-1]
75
76         # Save to temp file for parsing
77         with tempfile.NamedTemporaryFile(delete=False, suffix=f".{ext}") as tmp:
78             tmp.write(file_binary)
79             tmp_path = tmp.name
80
81         # Parse file content
82         if ext == 'pdf':
83             text = extract_text_from_pdf(tmp_path)
```

Ln 29, Col 23: Spaces: 4 UTF-8 LF Python Lambda Layout US

```
lambda_function.py X
lambda_function.py
60 def lambda_handler(event, context):
61     try:
94         item = {
95             'ResumeId': resume_id,
96             'name': parsed['name'],
97             'email': parsed['email'],
98             'phone': parsed['phone'],
99             'uploaded_at': datetime.utcnow().isoformat(),
100             's3_key': s3_key,
101             'sections': parsed['sections']
102         }
103     table.put_item(Item=item)
104
105     cleaned_sections = {}
106     main_fields = (parsed['name'].lower(), parsed['email'].lower(), parsed['phone'].replace(" ", ""))
107     for section_title, section_text in parsed['sections'].items():
108         lower_text = section_text.lower().replace(" ", "")
109         if not any(field in lower_text for field in main_fields):
110             cleaned_sections[section_title] = section_text
111     parsed['sections'] = cleaned_sections
112
113     return {
114         'statusCode': 200,
115         'headers': {
116             'Access-Control-Allow-Origin': '*',
117             'Access-Control-Allow-Methods': 'POST, OPTIONS',
118             'Access-Control-Allow-Headers': 'Content-Type'
119         },
120     },
121     'body': json.dumps({
122         'message': f'Resume uploaded and parsed. ID: {resume_id}',
123         'parsed': {
124             'name': parsed['name'],
125             'email': parsed['email'],
126             'phone': parsed['phone'],
127             'sections': parsed['sections']
128         }
129     })
130 }
131
132 except Exception as e:
133     traceback.print_exc()
134     return {
```

Ln 28, Col 21: Spaces: 4 UTF-8 LF Python Lambda Layout US