1. Connect to the Server

Command

```bash

ssh -p 22 root@62.72.29.253

```

Expected Output

Welcome to your server!

It will ask for a password.

Password

` india@2012AWS#2008 `

---

2. Update the System

Updating the system ensures all packages are up to date and secure.

Commands

```bash

sudo apt update && sudo apt upgrade -y

```

3. Install Docker

Docker allows you to containerize and deploy your application.

Commands

1. Install prerequisites:

```bash

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

```

2. Add Docker's GPG key and repository:

```bash

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

```

3. Install Docker:

```bash

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io

```

4. Start and enable Docker:

```bash

sudo systemctl start docker

sudo systemctl enable docker

```

Verify Docker Installation

```bash

docker --version

```

Output Example

`Docker version 24.0.7, build afdd53b`

4. Install and Configure NGINX

NGINX acts as a reverse proxy to forward requests to your FastAPI application.

Commands

1. Install NGINX:

```bash

sudo apt install -y nginx

```

2. Start and enable NGINX:

```bash

sudo systemctl start nginx

sudo systemctl enable nginx

```

3. Verify NGINX is running:

```bash

sudo systemctl status nginx

```

---

5. Configure NGINX for Your Application

Commands

1. Create a configuration file:

```bash

sudo nano /etc/nginx/sites-available/myapp

```

2. Paste the Configuration:

```nginx

server {   
listen 80;   
server\_name stage.indiahousepathlab.com;   
location /public/ocr { proxy\_pass http://127.0.0.1:3000/; # Ensure FastAPI port is correct proxy\_set\_header Host $host;   
proxy\_set\_header X-Real-IP $remote\_addr;   
proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;   
proxy\_set\_header X-Forwarded-Proto $scheme;   
}   
}   
server {   
listen 443 ssl;   
server\_name stage.indiahousepathlab.com;   
ssl\_certificate   
/etc/letsencrypt/live/stage.indiahousepathlab.com/fullchain.pem;  
ssl\_certificate\_key  
/etc/letsencrypt/live/stage.indiahousepathlab.com/privkey.pem;   
ssl\_protocols TLSv1.2 TLSv1.3;   
ssl\_ciphers 'ECDHE-ECDSA-...'; # Update with your preferred ciphers   
location /public/ocr {   
proxy\_pass http://127.0.0.1:3000/;  
proxy\_set\_header Host $host;   
proxy\_set\_header X-Real-IP $remote\_addr;   
proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;   
proxy\_set\_header X-Forwarded-Proto $scheme;   
}   
}  
```

3. Activate the configuration:

```bash

sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/

```

4. Test NGINX configuration:

```bash

sudo nginx -t

```

Expected Output

```plaintext

nginx: configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful

```

5. Reload NGINX:

```bash

sudo systemctl reload nginx

```

---

## \*\*6. Set Up HTTPS with Certbot\*\*

Certbot will install SSL certificates and configure NGINX for HTTPS.

### \*\*Commands\*\*

1. Install Certbot:

```bash

sudo apt install -y certbot python3-certbot-nginx

```

2. Generate SSL Certificates:

```bash

sudo certbot --nginx -d domainname.com -d www.domainname.com

```

3. \*\*Follow Certbot Prompts\*\*:

- Enter your email.

- Agree to terms.

- Certificates will be installed automatically.

4. Verify certificates:

```bash

sudo certbot certificates

```

---

7. Deploy Your Application Using Docker

Commands

1. Pull your Docker image:

```bash

docker pull ayush230/aspiring\_noapi:v2

```

2. Run the container:

```bash

docker run -d --restart unless-stopped -p 3000:80 ayush230/aspiring\_noapi:v2

```

3. Check the running container:

```bash

docker ps -a

```

Expected Output:

```

CONTAINER ID IMAGE STATUS PORTS

xxxxxxxxxx ayush230/aspiring\_noapi:v2 Up 5 secs 0.0.0.0:3000->80/tcp

```

---

8. Verify Your Application

1. Visit your domain in a browser:

```

https://domainname.com

```

2. Confirm:

- Your app loads successfully.

- A padlock icon appears (SSL is active).

---

Code Files

1. `Dockerfile

# Use the official Python image as a base

FROM python:3.10-slim-buster

# Install system dependencies including Tesseract

RUN apt-get update && \

apt-get -qq -y install tesseract-ocr && \

apt-get -qq -y install libtesseract-dev

# Set the working directory in the container

WORKDIR /app

# Copy the requirements file

COPY requirements.txt .

# Install Python dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Copy your application code

COPY . .

# Command to run your application

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]

```

---

2. `requirements.txt

```altair==5.4.1

annotated-types==0.7.0

anyio==4.6.2.post1

attrs==24.2.0

blinker==1.8.2

cachetools==5.5.0

certifi==2024.8.30

cffi==1.17.1

charset-normalizer==3.3.2

click==8.1.7

colorama==0.4.6

cryptography==43.0.1

easyocr==1.7.2

fastapi==0.115.2

filelock==3.16.1

fsspec==2024.10.0

gitdb==4.0.11

GitPython==3.1.43

h11==0.14.0

idna==3.10

imageio==2.36.0

Jinja2==3.1.4

jsonschema==4.23.0

jsonschema-specifications==2023.12.1

lazy\_loader==0.4

markdown-it-py==3.0.0

MarkupSafe==2.1.5

mdurl==0.1.2

mpmath==1.3.0

narwhals==1.9.1

networkx==3.4.2

ninja==1.11.1.1

numpy==2.1.2

opencv-python-headless==4.10.0.84

packaging==24.1

pandas==2.2.3

pdf2image==1.17.0

pdfminer.six==20231228

pdfplumber==0.11.4

pillow==10.4.0

protobuf==5.28.2

pyarrow==17.0.0

pyclipper==1.3.0.post6

pycparser==2.22

pydantic==2.9.2

pydantic\_core==2.23.4

pydeck==0.9.1

Pygments==2.18.0

PyMuPDF==1.24.11

pypdf==5.0.1

pypdfium2==4.30.0

pytesseract==0.3.13

python-bidi==0.6.3

python-dateutil==2.9.0.post0

python-dotenv==1.0.1

python-multipart==0.0.12

pytz==2024.2

PyYAML==6.0.2

pyzbar==0.1.9

referencing==0.35.1

requests==2.32.3

rich==13.9.2

rpds-py==0.20.0

scikit-image==0.24.0

scipy==1.14.1

setuptools==75.2.0

shapely==2.0.6

six==1.16.0

smmap==5.0.1

sniffio==1.3.1

starlette==0.40.0

streamlit==1.39.0

sympy==1.13.1

tenacity==9.0.0

tifffile==2024.9.20

toml==0.10.2

torch==2.5.0

torchvision==0.20.0

tornado==6.4.1

typing\_extensions==4.12.2

tzdata==2024.2

urllib3==2.2.3

uvicorn==0.32.0

watchdog==5.0.3

```

3. `.env` File

```

API\_KEY = LoUCOXUU8Ge5R9HeyobVUcY0KToFE2x3n1Sl8JdNe1JVgD35FiD9EX4tcmBhXjoq, RuBUBg0yZmkh3DhqTD3CiGB1MFe4sRHBms9PPa1jkwGQM5RzbDSHrIg7dDaL9J50, nHCxBIJanF2xqRmL0icVP2J6NOuHLFSzXktD9DplE3gocRrzzD3WG93GLKfBXTZl

```

---

Docker Commands for Building and Pushing to Hub

1. Build the Docker Image

```bash

docker build -t ayush230/aspiring\_noapi:v2 .

```

2. Log in to Docker Hub

```bash

docker login

```

3. Push the Image

```bash

docker push ayush230/aspiring\_noapi:v2

```

---

Main.py  
```

import re

import fitz

from fastapi.responses import JSONResponse

import os

import requests

from dotenv import load\_dotenv

from PIL import Image

import pytesseract

import requests

from fastapi import FastAPI, UploadFile, File, Request, Depends, HTTPException

import uvicorn

from fastapi.middleware.cors import CORSMiddleware

import cv2

import numpy as np

from PIL import Image

app = FastAPI()

load\_dotenv()

# middleware setup

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=["\*"],

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

# Setting up Tesseract

pytesseract.pytesseract.tesseract\_cmd = '/usr/bin/tesseract'

# pytesseract.pytesseract.tesseract\_cmd = os.getenv('Tesseract')

# Define a function to extract text from a single page of the PDF

def extract\_text\_from\_page(page):

    # Extract text from a specific page

    return page.get\_text("text")

def get\_api\_key(request: Request):

    api\_key = request.headers.get('X-API-Key') or request.query\_params.get('api\_key')

    # print(f"Received API key: {api\_key}")

    correct\_api\_keys = os.getenv("API\_KEY").split(',')

    # print(correct\_api\_keys)

    if api\_key not in correct\_api\_keys:

        raise HTTPException(status\_code=403, detail="Forbidden: Invalid API Key")

    return api\_key

def preprocess\_image\_for\_ocr(image):

    # Step 1: Increase Image Resolution (upscale)

    image = image.resize((int(image.width \* 1.5), int(image.height \* 1.5)), Image.Resampling.LANCZOS)

    # Convert to a format compatible with OpenCV for further processing

    open\_cv\_image = np.array(image)

    open\_cv\_image = cv2.cvtColor(open\_cv\_image, cv2.COLOR\_RGB2BGR)

    processed\_image = Image.fromarray(open\_cv\_image)

    return processed\_image

# Define a function to extract codes based on the fixed pattern

def extract\_codes\_from\_text(text):

    # Regex pattern for codes like AC1-21-02-15-3, AC1-21-02-15-12, etc.

    # pattern = r'[A-Z]{2}\d-\d{2}-\d{2}-\d{1,2}-\d{1,2}'

    pattern = r'[A-Z]{2,5}\d{1,9}-\d{1,5}-\d{1,5}-\d{1,5}-\d{1,3}'

    codes = re.findall(pattern, text)

    return codes

# extracting codes using ocr

def extract\_codes\_from\_image(page):

    pix = page.get\_pixmap()

    img = Image.frombytes("RGB", [pix.width, pix.height], pix.samples)

    preprocessed\_img = preprocess\_image\_for\_ocr(img)

    text = pytesseract.image\_to\_string(preprocessed\_img)

    # print(text)

    return text

# function to check if the pdf has images or text

def check\_pdf (doc):

    for page\_num in range(len(doc)):

        page = doc.load\_page(page\_num)

        if page.get\_text("text").strip():

            return False

    return True

# API route

@app.post('/ocr')

async def extract(url: str = None, api\_key: str = Depends(get\_api\_key)):

    temp\_pdf = 'temp.pdf'

    try:

        response = requests.get(url, stream=True)

        response.raise\_for\_status()  # Check for HTTP request errors

        with open(temp\_pdf, "wb") as f:

            f.write(response.content)  # Write the downloaded content to a file

    except Exception as e:

        return JSONResponse(content={"error": f"Failed to download PDF from URL. Error: {str(e)}"})

    # Open the PDF document

    doc = fitz.open(temp\_pdf)

    # Empty list to collect all the codes along with page nos

    codes\_with\_page\_no = []

    contains\_image = check\_pdf(doc)

    # Loop through each page in the PDF

    for page\_num in range(doc.page\_count):

        page = doc.load\_page(page\_num)

        if contains\_image:

            text = extract\_codes\_from\_image(page)

        else:

        # Extract text from the current page

            text = extract\_text\_from\_page(page)

        # Extract codes from the current page's text

        codes = extract\_codes\_from\_text(text)

        code = codes[0] if codes else None

        # Print the result for each page

        if code:

            parts = code.split('-')

            certificate\_id = parts[0]

            document\_id = parts[1]

            sequence = parts[-3]

            page\_number = parts[-1]

            total\_pages = parts[-2]

            codes\_with\_page\_no.append({

                "certificate\_id": certificate\_id,

                "document\_id": document\_id,

                "sequence": sequence,

                "total\_pages": total\_pages,

                "page\_number": page\_number

                })

        else:

            # Pages with no codes found

            codes\_with\_page\_no.append({

                "certificate\_id": None,

                "document\_id": None,

                "sequence": None,

                "total\_pages": None,

                "page\_number": None

            })

    # closing the pdf

    doc.close()

    # Clean up the temporary file

    os.remove(temp\_pdf)

    if codes\_with\_page\_no:

        return JSONResponse(content={"extracted\_data": codes\_with\_page\_no})

    else:

        return {'message' : 'NO CODES FOUND'}

if \_\_name\_\_=='\_\_main\_\_' :

    uvicorn.run(app, host = '127.0.0.1', port = 80)

```