Dockerized REST API that ingests an in-force illustration PDF, auto-extracts key fields (carrier-agnostic), PII scrubs content, computes a confidence score, generates proof snips, and returns decision-ready JSON plus a redacted PDF. If confidence >= 0.80 mark Final; otherwise mark Provisional - Needs Review but still return full output. No user interaction beyond upload

What you need before starting

1. **Your AWS login** (the normal web console you use).
2. **Your OpenAI account** (we'll create an API key later).
3. **The project ZIP** you downloaded from me: `1035_extractor_scaffold_ascii.zip` (it's on your computer).

Part A — Quick test in your browser (AWS Cloud9)

This lets you run everything without installing anything on your computer.

1. **Log in to AWS Console.**
2. In the top search bar, type **Cloud9**, open it.
3. Click **Create environment**.
    - Name: `extractor-workbench`
    - Leave defaults (t3.small is fine), click **Create**. It takes ~1–2 minutes.
4. You'll land in a browser IDE (file tree + terminal at bottom).
5. **Upload the ZIP** (no command line yet):

- In the left file tree, right-click the top folder, choose **Upload Local Files…**
- Pick `1035_extractor_scaffold_ascii.zip` from your computer.

6. **Unzip it** (1 command):

In the bottom terminal (it's already open), paste:

```
unzip 1035_extractor_scaffold_ascii.zip
cd 1035_extractor
```

-

Build and run the Docker app (2 commands):

```
docker build -t extractor:latest .
docker run --rm -p 8000:8000 extractor:latest
```

7. Leave this window running—it starts the API inside Cloud9.
8. **Test it right there**:
   - In Cloud9 menu, click **Preview** → **Preview Running Application**.
   - In the URL bar that opens, add `/docs` at the end and press Enter.
   - You'll see the API page. Click **POST /analyze** → **Try it out** → **Choose File**, upload a sample in-force PDF, then **Execute**.
   - You should get JSON back (it includes `confidence_overall`, series, and a `redacted_pdf_b64`).

> If that worked, your extractor is good. Next, we'll put it on your own server so you can hit it from anywhere.

Part B — Put it on your AWS server (EC2 + Docker)

This is the simplest way to make it reachable from the internet.

B1) Launch the server

1. In AWS search, open **EC2** → **Instances** → **Launch instances**.
2. Name: `extractor-server`.
3. Image: **Amazon Linux 2023** (default).
4. Instance type: **t3.small** (okay for testing).
5. Key pair: create or select one (you won't need it if you use the browser connect, but it's fine).
6. **Network settings** → **Edit**:
    - Allow **SSH (22)** *from My IP* (so you can connect).
    - Add **Custom TCP** port **8000** from **Anywhere** (temporary so you can test from your computer).
    - (We'll lock this down later or put HTTPS in front.)
7. Click **Launch instance**. Wait ~1–2 minutes until status is "Running".

B2) Connect and install Docker

1. Back in **Instances**, select `extractor-server` → **Connect** → **EC2 Instance Connect** → **Connect** (opens a browser terminal).

Paste these commands (install & start Docker, and Git & unzip):

```
sudo yum install -y docker git unzip
sudo service docker start
```

2. (No need to log out/in; we'll just use `sudo` when running Docker.)

B3) Get your project onto the server

Use the **same ZIP** you have on your computer:

Option 1 (easiest): upload via GitHub (web only, no coding)

1. Go to github.com → **New repository** (public or private).
2. Click **Add file** → **Upload files** and upload `1035_extractor_scaffold_ascii.zip`.
3. Click **Commit** (save).

Back on the EC2 terminal (browser), run:

```
git clone https://github.com/<your-username>/<your-repo>.git
cd <your-repo>
unzip 1035_extractor_scaffold_ascii.zip
cd 1035_extractor
```

4.

(If you prefer not to use GitHub, we can instead use S3; just ask.)

B4) Build and run the app

Build the image:

```
sudo docker build -t extractor:latest .
```

1.

Run it in the background:

```
sudo docker run -d --name extractor -p 8000:8000 extractor:latest
```

2.
3. Find your server's **Public IPv4 address** in the EC2 console.

On your own computer, open a browser to:

```
http://<YOUR_PUBLIC_IP>:8000/docs
```

4. Click **POST /analyze** → **Try it out** and upload an in-force PDF. You should get a JSON response.

At this point, your extractor is live. For production you'll want HTTPS (we can add a Load Balancer + certificate later).

Part C — (Optional) Add HTTPS with a Load Balancer (later)

When you're ready to secure it:

1. Request a free **certificate** in AWS Certificate Manager (ACM) for your domain (e.g., `api.yourdomain.com`).

2. Create an **Application Load Balancer (ALB)** in EC2:

    - Listener **443 (HTTPS)** with the ACM certificate, target group → your EC2 instance on port **8000**.

3. Update your DNS (Route 53 or your registrar) to point `api.yourdomain.com` to the ALB.
   (If you want, I can write these clicks out step-by-step too.)

Part D — Hook up OpenAI (the decision & report part)

Your extractor returns clean JSON. Now we'll ask OpenAI to (1) give a Yes/No/Needs-Review decision, and (2) write the client-friendly text for your PDF.

1. **Create an OpenAI API key** (in your OpenAI account → API Keys → Create new secret key). Copy it.

2. In your app that orchestrates the flow (Zapier/Make, or a tiny backend), send the **JSON** from `/analyze` to OpenAI. Here's the idea in plain words:

- System message: “You are deciding 1035 exchanges. Only use provided JSON. If confidence < 0.80, return Needs-Review.”
- User message: paste the JSON from `/analyze`.
- Model returns: `{ decision: Yes/No/Needs-Review, reasons: [...], fields_used: [...] }`.
- Then call it again (or same call) to write the **report narrative** using those fields.

3. Use a document tool (Documint, PDFMonkey, or similar) to merge the narrative + your **proof snips** into a PDF template.

(If you want, I’ll give you an exact prompt and a Zapier recipe to drop in.)

Common “gotchas” (and the quick fix)

- **Can’t reach `:8000` from your computer?** Make sure the EC2 **Security Group** allows inbound **TCP 8000** from your IP (you set that when launching).
- **Docker says “permission denied”?** Add `sudo` in front of the docker commands on EC2.

Upload limit in `/docs` test? The Swagger UI is fine for typical PDFs; for very large files, use `curl`:

```
curl -F "file=@/full/path/to/your.pdf"
http://<YOUR_PUBLIC_IP>:8000/analyze
```

-
- **OCR is slow on huge scans?** That’s normal. Start with a clearer, text-based PDF when testing.

What you just accomplished

- Ran the extractor safely in a browser (Cloud9).
- Deployed it on your own AWS server (EC2 + Docker).
- Confirmed the `/analyze` endpoint works with a real PDF.
- You're ready to plug the JSON into OpenAI for the Yes/No/Needs-Review decision and the report text.

If you tell me which **final target** you want next—keep it on EC2 with HTTPS, or move to a fully managed service (ECS Fargate)—I'll give you the exact clicks and commands just for that choice.