ATS Web App Using Python, AI & Streamlit

"This document is a professional, end-to-end guide for deploying, configuring, securing, and operating the ATS (Applicant Tracking System) web application built with Python and Streamlit and integrated with a Generative AI model."

Contents:--

1. Overview
2. Reference Architecture
3. Prerequisites
4. Detailed Steps

1. Overview

The ATS web application enables resume screening and analysis via a Streamlit-based user interface. It leverages a Generative AI model (via the Google Generative AI Python SDK) to extract insights from uploaded resumes and return structured feedback or scores to the user. The app supports standard PDF resumes and also handles image-based resumes by using Poppler utilities to process PDFs/images before sending content to the AI layer.

Key Features:

- Simple web UI built with Streamlit.
- Resume parsing & analysis powered by a Generative AI model.
- Support for PDF and image-based resumes using Poppler utilities.
- Self-hosted on a single Amazon EC2 instance for quick deployment.

2. Reference Architecture

A minimal single-EC2 design is used:

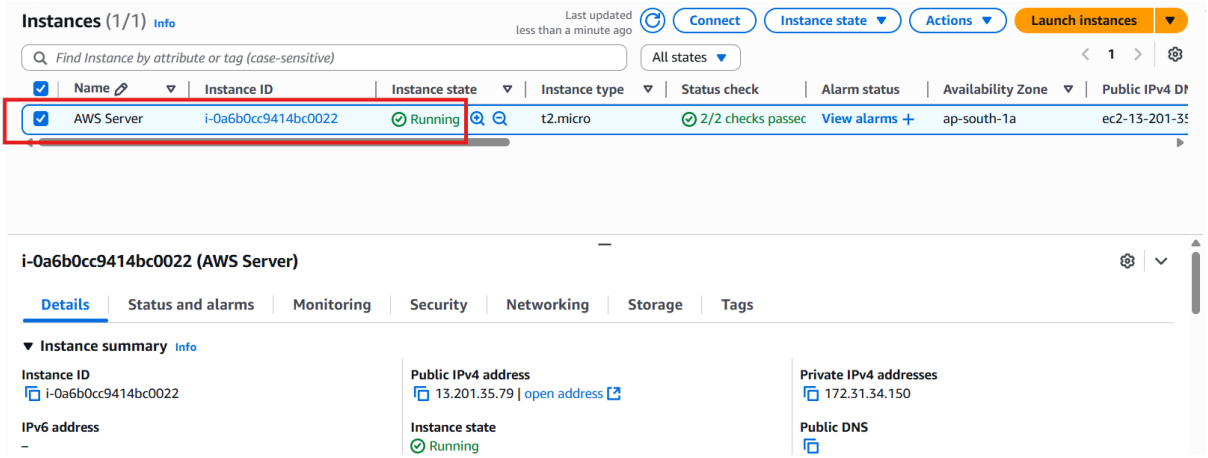
- Users access the Streamlit app through web server.
- The application process runs under a Python virtual environment and exposes Streamlit on port 8501.
- The app invokes the Google Generative AI SDK using an API key stored securely as an environment variable or a secrets file.
- Poppler is installed on the instance to enable conversion of PDFs/images for robust resume ingestion.

3. Pre-requisites

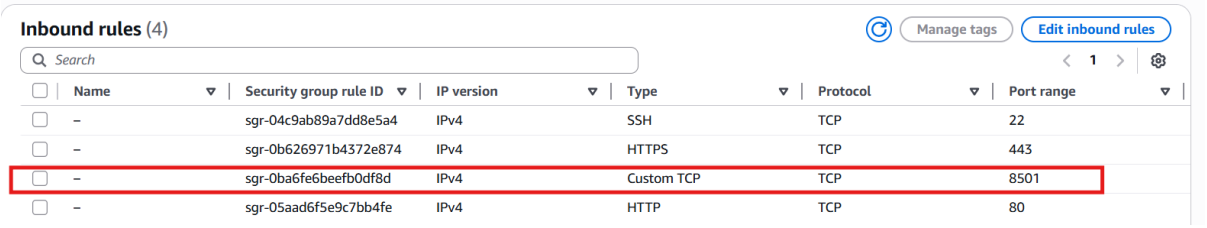
- An AWS account with permissions to create EC2 instances and security groups.
- A Linux-based EC2 AMI (e.g., Ubuntu 22.04 LTS), an SSH key pair, and a public IP or Elastic IP.
- Security group rules: allow SSH (22) from your IP. For testing, allow 8501 (Streamlit) temporarily. For production, allow 80/443 and remove public access to 8501.
- A Google Generative AI API key (from Google Cloud console).
- Git repository URL for the ATS application code.

4. Detailed Steps

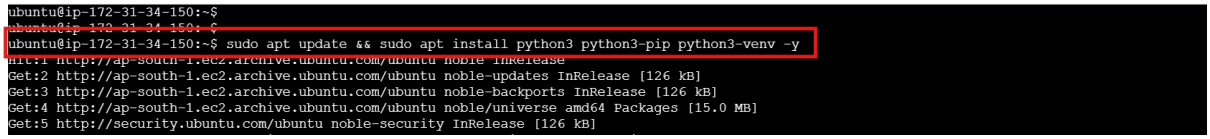
Create an EC2 instance:



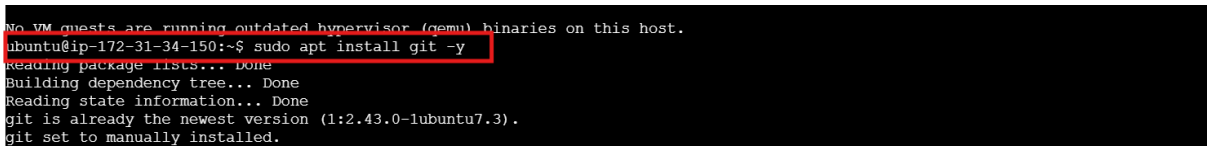
Edit and add custom port in inbound rules (as 8501 is the default port for streamlit):



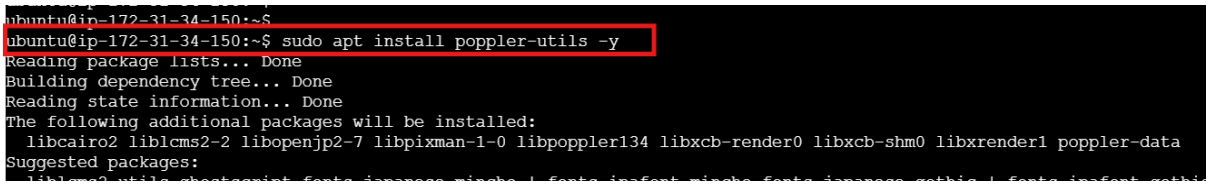
Now update packages, install python and create a venv:



Now install git for fetching and connecting to the repo:



Installing poppler for our ATS to read the image resume as well:



Cloning the repo for fetching the code of our ATS system:

```
ubuntu@ip-172-31-34-150:~$
ubuntu@ip-172-31-34-150:~$
ubuntu@ip-172-31-34-150:~$ git clone https://github.com/Anu-Anurag/Application-Tracking-System
Cloning into 'Application-Tracking-System'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 45 (delta 18), reused 7 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (45/45), 15.14 KiB | 2.52 MiB/s, done.
Resolving deltas: 100% (18/18), done.
```

Check whether the repo is cloned successfully or not:

```
ubuntu@ip-172-31-34-150:~$
ubuntu@ip-172-31-34-150:~$
ubuntu@ip-172-31-34-150:~$ cd Application-Tracking-System/
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ ls
README.md  app.py  index.html  packages.txt  requirements.txt
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
```

Now setup and activate the virtual environment:

```
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ python3 -m venv venv
ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ source venv/bin/activate
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
```

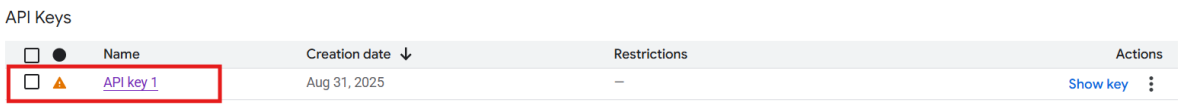
Updating the pip package manager and downloading dependencies from requirements.txt:

```
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ pip install --upgrade pip
Requirement already satisfied: pip in ./venv/lib/python3.12/site-packages (24.0)
Collecting pip
  Downloading pip-25.2-py3-none-any.whl.metadata (4.7 kB)
Downloading pip-25.2-py3-none-any.whl (1.8 MB)
   ---------------------------------------- 1.8/1.8 MB 30.8 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-25.2
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ pip install -r requirements.txt
Collecting streamlit (from -r requirements.txt (line 1))
  Downloading streamlit-1.49.1-py3-none-any.whl.metadata (9.5 kB)
```

Installed Google GenAI console in venv:

```
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ pip install google-generativeai
Requirement already satisfied: google-generativeai in ./venv/lib/python3.12/site-packages (0.8.5)
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in ./venv/lib/python3.12/site-packages (from google-generativeai) (0.6.15)
```

Generate an API key from Google Cloud console:



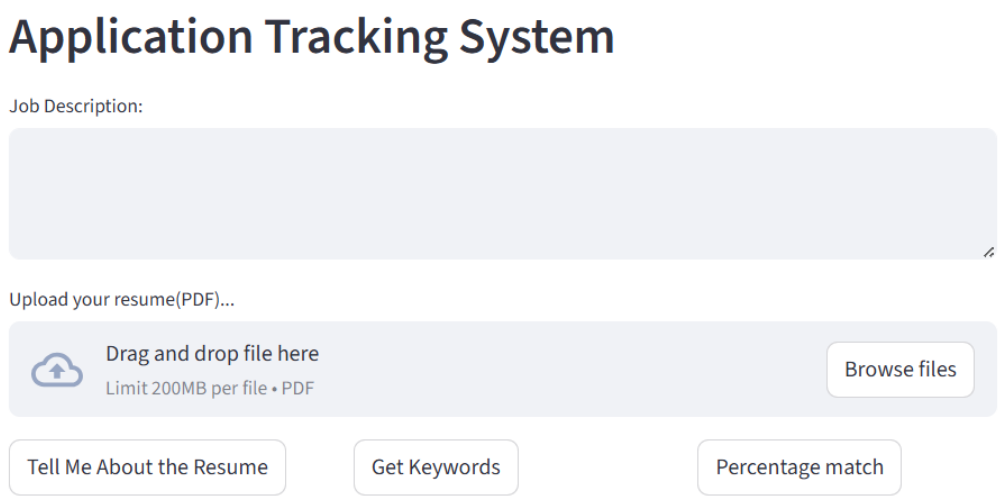
Create a directory and add the API key in a file inside that directory:

```
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ mkdir -p .streamlit
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ vi .streamlit/secrets.toml
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
```

Finally, run the Streamlit app:

```
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$
(venv) ubuntu@ip-172-31-34-150:~/Application-Tracking-System$ streamlit run app.py --server.port 8501 --server.enableCORS false
2025-09-01 08:41:03.251
Warning: the config option 'server.enableCORS=false' is not compatible with
'server.enableXsrfProtection=true'.
As a result, 'server.enableCORS' is being overridden to 'true'.

More information:
In order to protect against CSRF attacks, we send a cookie with each request.
To do so, we must specify allowable origins, which places a restriction on
cross-origin resource sharing.

If cross origin resource sharing is required, please disable server.enableXsrfProtection.


Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.31.34.150:8501
  External URL: http://13.201.35.79:8501
```

And our ATS web app is running fine:



Well Done…