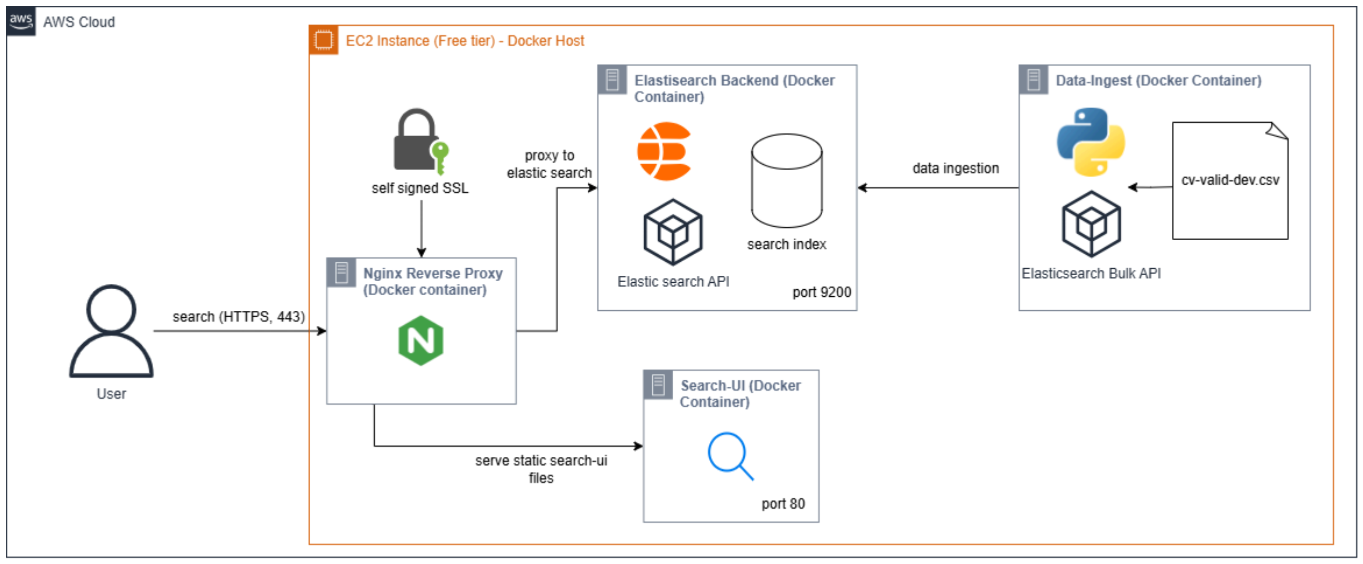
Task 3 Design


AWS Cloud
EC2 Instance (Free tier) - Docker Host
Elastisearch Backend (Docker Container)
Data-Ingest (Docker Container)
proxy to elastic search
self signed SSL
data ingestion
cv-valid-dev.csv
search index
Elastic search API
port 9200
Elasticsearch Bulk API
Nginx Reverse Proxy (Docker container)
search (HTTPS, 443)
User
Search-UI (Docker Container)
serve static search-ui files
port 80

The architecture is deployed on a single EC2 instance on AWS and containerised using Docker compose. There are 4 containers, nginx reverse proxy, search-ui, elasticseach-backend, and data ingestion.

Nginx Reverse Proxy

- Central entry point to app
- Enable https connections for users
- Routes incoming https traffic on port 443 to 2 services
    - /: forwards to search-ui
    - /elasticsearch/: proxies api requests to elasticsearch-backend container

Search-UI

- Frontend served via nginx
- Provides search interface to user
- Runs internally on port 80
- Sends api requests to /elasticsearch for data

ElasticSearch backend

- Hosts search index and listens on port 9200 internally
- Receives queries from frontend via nginx reverse proxy
- Returns matching results based on user search/filters

Data ingest

- Python used to programmatically load data from csv to index
- Uses elastic search bulk api via python client to ingest the data
- Runs once during deployment to add the data

Data flow

- CSV data ingested into elastic search index
- User accesses app via https

- Nginx serves frontend and proxies api calls
- Search ui sends search queries (if user filters through data or manually searches)
- Nginx forwards api request to elastic search
- Elastic search returns results from index
- Search ui displays results to user