Deployment Guide for Resume Tailor Agent

This guide covers various deployment options for the Resume Tailor Agent, from local development to production cloud deployment.

🏠 Local Development Deployment

Quick Start

Bash

```bash
# Clone and setup
git clone <your-repo>
cd resume-tailor-agent
python -m venv venv
source venv/bin/activate  # or venv\Scripts\activate on Windows
pip install -r requirements.txt

# Start web interface
streamlit run streamlit_app.py
```

Development with Auto-reload

Bash

```bash
# Install development dependencies
pip install watchdog

# Run with auto-reload
streamlit run streamlit_app.py --server.runOnSave true
```

☁️ Cloud Deployment Options

1. Streamlit Cloud (Easiest)

Prerequisites

- GitHub repository with your code
- Streamlit Cloud account (free)

Steps

1. Push code to GitHub repository
2. Go to https://share.streamlit.io/
3. Connect your GitHub account
4. Select repository and branch
5. Set main file path: `streamlit_app.py`
6. Add secrets for API keys in dashboard

Secrets Configuration

In Streamlit Cloud dashboard, add:

Plain Text

```
OPENAI_API_KEY = "your_openai_key"
ANTHROPIC_API_KEY = "your_anthropic_key"
```

Limitations

- No local Ollama support (cloud-only)
- Limited compute resources
- Public by default

2. Heroku Deployment

Prerequisites

- Heroku account
- Heroku CLI installed

Setup Files

Procfile:

Plain Text

```
web: streamlit run streamlit_app.py --server.port=$PORT --
server.address=0.0.0.0
```

runtime.txt:

Plain Text

```
python-3.10.12
```

requirements.txt: (ensure it includes)

Plain Text

```
streamlit>=1.28.1
python-docx>=0.8.11
requests>=2.31.0
openai>=1.51.0
anthropic>=0.7.8
```

Deployment Steps

Bash

```bash
# Login to Heroku
heroku login

# Create app
heroku create your-resume-tailor-app

# Set environment variables
heroku config:set OPENAI_API_KEY="your_key"
heroku config:set ANTHROPIC_API_KEY="your_key"

# Deploy
git add .
git commit -m "Deploy to Heroku"
git push heroku main
```

3. Google Cloud Platform (GCP)

Using Cloud Run

Dockerfile:

Plain Text

```
FROM python:3.10-slim

WORKDIR /app
```

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

COPY . .

EXPOSE 8080

CMD ["streamlit", "run", "streamlit_app.py", "--server.port=8080", "--
server.address=0.0.0.0"]
```

Deployment:

Bash

```bash
# Build and deploy
gcloud run deploy resume-tailor \
  --source . \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated \
  --set-env-vars OPENAI_API_KEY="your_key"
```

4. AWS Deployment

Using AWS App Runner

apprunner.yaml:

YAML

```yaml
version: 1.0
runtime: python3
build:
  commands:
    build:
      - pip install -r requirements.txt
run:
  runtime-version: 3.10.12
  command: streamlit run streamlit_app.py --server.port=8080 --
server.address=0.0.0.0
  network:
    port: 8080
    env: PORT
  env:
    - name: OPENAI_API_KEY
      value: "your_key"
```

Using EC2 with Docker

docker-compose.yml:

YAML

```yaml
version: '3.8'
services:
  resume-tailor:
    build: .
    ports:
      - "80:8080"
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
      - ANTHROPIC_API_KEY=${ANTHROPIC_API_KEY}
    restart: unless-stopped
```

5. Azure Deployment

Using Azure Container Instances

Bash

```bash
# Create resource group
az group create --name resume-tailor-rg --location eastus

# Deploy container
az container create \
  --resource-group resume-tailor-rg \
  --name resume-tailor-app \
  --image your-registry/resume-tailor:latest \
  --dns-name-label resume-tailor-unique \
  --ports 8080 \
  --environment-variables OPENAI_API_KEY="your_key"
```

🐳 Docker Deployment

Basic Dockerfile

Plain Text

```
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    gcc \
    && rm -rf /var/lib/apt/lists/*

# Copy requirements and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY . .

# Create non-root user
RUN useradd -m -u 1000 appuser && chown -R appuser:appuser /app
USER appuser

# Expose port
EXPOSE 8501

# Health check
HEALTHCHECK CMD curl --fail http://localhost:8501/_stcore/health

# Run application
CMD ["streamlit", "run", "streamlit_app.py", "--server.port=8501", "--
server.address=0.0.0.0"]
```

Multi-stage Build (Optimized)

Plain Text

```
# Build stage
FROM python:3.10-slim as builder

WORKDIR /app
COPY requirements.txt .
RUN pip install --user --no-cache-dir -r requirements.txt

# Runtime stage
FROM python:3.10-slim

WORKDIR /app
```

```
# Copy installed packages from builder
COPY --from=builder /root/.local /root/.local

# Copy application
COPY . .

# Make sure scripts in .local are usable
ENV PATH=/root/.local/bin:$PATH

EXPOSE 8501

CMD ["streamlit", "run", "streamlit_app.py", "--server.port=8501", "--
server.address=0.0.0.0"]
```

Docker Compose for Development

YAML

```yaml
version: '3.8'
services:
  resume-tailor:
    build: .
    ports:
      - "8501:8501"
    volumes:
      - .:/app
      - /app/venv
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
      - ANTHROPIC_API_KEY=${ANTHROPIC_API_KEY}
    command: streamlit run streamlit_app.py --server.runOnSave true

  ollama:
    image: ollama/ollama:latest
    ports:
      - "11434:11434"
    volumes:
      - ollama_data:/root/.ollama
    command: serve

volumes:
  ollama_data:
```

🔧 Production Configuration

Environment Variables

Bash

```bash
# Required for API-based LLMs
OPENAI_API_KEY=your_openai_key
ANTHROPIC_API_KEY=your_anthropic_key

# Optional configuration
DEFAULT_MODEL=local
LOG_LEVEL=INFO
MAX_UPLOAD_SIZE=200
STREAMLIT_SERVER_PORT=8501
STREAMLIT_SERVER_ADDRESS=0.0.0.0
```

Streamlit Production Config

.streamlit/config.toml:

Plain Text

```
[server]
port = 8501
address = "0.0.0.0"
maxUploadSize = 200
enableCORS = false
enableXsrfProtection = true

[browser]
gatherUsageStats = false
showErrorDetails = false

[logger]
level = "info"

[client]
showErrorDetails = false
```

Nginx Reverse Proxy

Plain Text

```
server {
    listen 80;
    server_name your-domain.com;
    location / {
        proxy_pass http://localhost:8501;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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;
        proxy_cache_bypass $http_upgrade;
    }
}
```

📊 Monitoring and Logging

Application Monitoring

Python

```python
# Add to streamlit_app.py
import logging
import time

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)

# Add performance monitoring
@st.cache_data
def monitor_performance(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result = func(*args, **kwargs)
        end_time = time.time()
        logging.info(f"{func.__name__} took {end_time - start_time:.2f}
seconds")
        return result
    return wrapper
```

Health Check Endpoint

Python

```python
# Add to streamlit_app.py
import streamlit as st
from streamlit.web import cli as stcli
import sys

def health_check():
    """Simple health check for monitoring"""
    return {"status": "healthy", "timestamp": time.time()}

# Add health check route (requires custom Streamlit setup)
```

Log Aggregation

For production, consider:

- **ELK Stack** (Elasticsearch, Logstash, Kibana)
- **Grafana + Prometheus**
- **Cloud-native solutions** (CloudWatch, Stackdriver)

🔒 Security Best Practices

API Key Management

Bash

```bash
# Use secret management services
# AWS Secrets Manager
aws secretsmanager get-secret-value --secret-id openai-api-key

# Google Secret Manager
gcloud secrets versions access latest --secret="openai-api-key"

# Azure Key Vault
az keyvault secret show --name openai-api-key --vault-name your-vault
```

HTTPS Configuration

Bash

```bash
# Let's Encrypt with Certbot
sudo certbot --nginx -d your-domain.com
```

```
# Or use cloud provider SSL certificates
```

Input Validation

Python

```python
# Add to streamlit_app.py
import re

def validate_file_upload(uploaded_file):
    """Validate uploaded files"""
    if uploaded_file.size > 200 * 1024 * 1024:  # 200MB limit
        raise ValueError("File too large")

    if not uploaded_file.name.endswith('.docx'):
        raise ValueError("Invalid file type")

    return True
```

📈 Scaling Considerations

Horizontal Scaling

- Use load balancers
- Implement session affinity
- Consider stateless design
- Use external storage for uploads

Vertical Scaling

- Monitor CPU/memory usage
- Optimize LLM model selection
- Implement caching strategies
- Use async processing for heavy tasks

Database Integration

For user management and job tracking:

Python

```python
# Example with SQLite
import sqlite3

def init_database():
    conn = sqlite3.connect('resume_tailor.db')
    conn.execute('''
        CREATE TABLE IF NOT EXISTS jobs (
            id INTEGER PRIMARY KEY,
            user_id TEXT,
            job_title TEXT,
            company TEXT,
            status TEXT,
            created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    conn.close()
```

🚀 CI/CD Pipeline

GitHub Actions Example

.github/workflows/deploy.yml:

YAML

```yaml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.10'
      - name: Install dependencies
        run: |
          pip install -r requirements.txt
      - name: Run tests
        run: |
          python test_parser.py
          python test_llm.py

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${{secrets.HEROKU_API_KEY}}
          heroku_app_name: "your-resume-tailor-app"
          heroku_email: "your-email@example.com"
```

📋 Deployment Checklist

Pre-deployment

- [ ] All tests passing
- [ ] Environment variables configured
- [ ] API keys secured
- [ ] Dependencies updated
- [ ] Documentation updated

Deployment

- [ ] Build successful
- [ ] Health checks passing
- [ ] SSL certificate configured
- [ ] Domain configured
- [ ] Monitoring setup

Post-deployment

- [ ] Application accessible
- [ ] All features working
- [ ] Performance acceptable

- [ ] Logs being collected
- [ ] Backup strategy in place

This deployment guide covers the most common scenarios for deploying the Resume Tailor Agent. Choose the option that best fits your needs, budget, and technical requirements.