🚀 Deployment Guide - Option Chain Analyzer

Overview

This guide covers deploying your Option Chain Analyzer app to various platforms, from local setup to cloud hosting.

🏠 Local Development Setup

Prerequisites

- Python 3.8 or higher
- pip (Python package manager)
- Git (optional, for version control)

Step-by-Step Setup

```bash
# 1. Create project directory
mkdir option_chain_analyzer
cd option_chain_analyzer

# 2. Create virtual environment
python -m venv venv

# 3. Activate virtual environment
# Windows:
venv\Scripts\activate
# Mac/Linux:
source venv/bin/activate

# 4. Install dependencies
pip install -r requirements.txt

# 5. Run the app
streamlit run app.py
```

Verify Installation

- Open browser to `http://localhost:8501`
- Upload sample CSV file

- Check if signals are generating correctly

☁️ Cloud Deployment Options

Option 1: Streamlit Community Cloud (FREE & EASIEST)

Pros:

- ✅ Free forever
- ✅ Automatic HTTPS
- ✅ Easy GitHub integration
- ✅ No server management

Steps:

1. **Push code to GitHub**

```bash
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/yourusername/option-chain-analyzer.git
git push -u origin main
```

2. **Deploy on Streamlit Cloud**

- Go to share.streamlit.io
- Sign in with GitHub
- Click "New app"
- Select your repository
- Choose `app.py` as main file
- Click "Deploy"

3. **Access your app**

- URL: `https://yourusername-option-chain-analyzer.streamlit.app`
- Share with anyone!

Limitations:

- 1GB RAM (sufficient for this app)
- Public repository required for free tier
- App sleeps after inactivity (wakes on visit)

Option 2: Heroku (Good for 24/7 Availability)

Pros:

- ✅ Always running (with paid plan)
- ✅ Custom domain support
- ✅ Database add-ons available

Setup Files Needed:

1. Procfile

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

2. runtime.txt

```
python-3.11.4
```

3. setup.sh

```bash
mkdir -p ~/.streamlit/

echo "\
[server]\n\
headless = true\n\
port = $PORT\n\
enableCORS = false\n\
\n\
" > ~/.streamlit/config.toml
```

Deployment Steps:

```bash
```

```
# 1. Install Heroku CLI
# Download from: https://devcenter.heroku.com/articles/heroku-cli

# 2. Login to Heroku
heroku login

# 3. Create Heroku app
heroku create option-chain-analyzer

# 4. Push to Heroku
git push heroku main

# 5. Open app
heroku open
```

Cost:

- Free tier: Limited hours/month
- Hobby: $7/month (always running)
- Professional: $25/month (performance + metrics)

Option 3: AWS EC2 (Full Control)

Pros:

- ✅ Complete customization
- ✅ Can run background jobs
- ✅ Add database, caching, etc.

Setup Steps:

1. **Launch EC2 Instance**

- Instance Type: `t2.micro` (free tier) or `t3.small`
- OS: Ubuntu 22.04 LTS
- Security Group: Allow ports 22 (SSH), 80 (HTTP), 443 (HTTPS)

2. **SSH into Instance**

```
bash
```

```
ssh -i your-key.pem ubuntu@your-instance-ip
```

3. Install Dependencies

```bash
# Update system
sudo apt update && sudo apt upgrade -y

# Install Python
sudo apt install python3-pip python3-venv -y

# Clone repository
git clone https://github.com/yourusername/option-chain-analyzer.git
cd option-chain-analyzer

# Setup virtual environment
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

4. Run with Supervisor (keeps app running)

Create `/etc/supervisor/conf.d/streamlit.conf`:

```ini
[program:streamlit]
directory=/home/ubuntu/option-chain-analyzer
command=/home/ubuntu/option-chain-analyzer/venv/bin/streamlit run app.py --server.port=8501
autostart=true
autorestart=true
stderr_logfile=/var/log/streamlit.err.log
stdout_logfile=/var/log/streamlit.out.log
```

```bash
# Start supervisor
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start streamlit
```

5. Setup Nginx Reverse Proxy

Create `/etc/nginx/sites-available/streamlit`:

```nginx
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_cache_bypass $http_upgrade;
    }
}
```

```bash
# Enable site
sudo ln -s /etc/nginx/sites-available/streamlit /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

6. **Setup SSL (HTTPS)**

```bash
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d your-domain.com
```

Cost:

- t2.micro: Free (first year) then ~$10/month
- t3.small: ~$15/month
- Elastic IP: ~$3.60/month if not attached to running instance

---

Option 4: Google Cloud Run (Serverless)

Pros:

- ✅ Pay only for usage

- ✅ Auto-scales
- ✅ Fast deployment

Setup Files:

1. Dockerfile

dockerfile

```
FROM python:3.11-slim

WORKDIR /app

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

COPY . .

EXPOSE 8080

CMD streamlit run app.py --server.port=8080 --server.address=0.0.0.0
```

2. .dockerignore

```
venv/
__pycache__/
*.pyc
.git/
.env
```

Deployment Steps:

bash

```
# 1. Install gcloud CLI
# Download from: https://cloud.google.com/sdk/docs/install

# 2. Initialize gcloud
gcloud init

# 3. Build and deploy
gcloud run deploy option-chain-analyzer \
  --source . \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated
```

Cost:

- First 2 million requests/month: FREE
- After that: ~$0.40 per million requests
- Very cost-effective for personal use

🔒 Security Best Practices

1. Environment Variables

Never hardcode sensitive data. Use `.env` file:

```python
# .env
EMAIL_PASSWORD=your_email_password
TELEGRAM_BOT_TOKEN=your_bot_token
API_KEY=your_api_key
```

```python
# In app.py
import os
from dotenv import load_dotenv

load_dotenv()

email_password = os.getenv('EMAIL_PASSWORD')
bot_token = os.getenv('TELEGRAM_BOT_TOKEN')
```

Add to `requirements.txt`:

```
python-dotenv==1.0.0
```

2. Input Validation

```python
```

```python
def validate_csv(df):
    """Validate uploaded CSV"""
    required_columns = ['Strike Price', 'Call OI', 'Put OI']

    if not all(col in df.columns for col in required_columns):
        raise ValueError("Invalid CSV format")

    if df['Strike Price'].isnull().any():
        raise ValueError("Strike Price cannot be null")

    return True
```

3. Rate Limiting (for API endpoints)

```python
from streamlit import cache_data
import time

@cache_data(ttl=60)  # Cache for 60 seconds
def rate_limited_function():
    # Your API calls here
    pass
```

4. HTTPS Only

Always use HTTPS in production. Free options:

- Streamlit Cloud: Automatic HTTPS
- Let's Encrypt: Free SSL certificates
- Cloudflare: Free CDN + SSL

📊 Monitoring & Logging

1. Streamlit Analytics

Built-in metrics in Streamlit Cloud:

- Page views
- Active users
- Error rates

2. Custom Logging

```python
import logging

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('app.log'),
        logging.StreamHandler()
    ]
)

logger = logging.getLogger(__name__)

# Usage
logger.info("User uploaded file")
logger.warning("Missing OI Change data")
logger.error("Signal generation failed")
```

3. Error Tracking (Sentry)

```python
import sentry_sdk

sentry_sdk.init(
    dsn="your-sentry-dsn",
    traces_sample_rate=1.0
)
```

🚀 Performance Optimization

1. Caching

```python
@st.cache_data(ttl=300)  # 5 minutes
def expensive_calculation(df):
    # Heavy computation
    return result
```

2. Lazy Loading

```python
with st.expander("Advanced Analysis"):
    # Only loads when expanded
    display_advanced_features()
```

3. Async Operations

```python
import asyncio

async def fetch_data():
    # Async operations
    pass

if st.button("Fetch"):
    asyncio.run(fetch_data())
```

🔄 CI/CD Pipeline

GitHub Actions Workflow

Create `.github/workflows/deploy.yml`:

```yaml
```

```yaml
name: Deploy to Streamlit Cloud

on:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2

    - name: Set up Python
      uses: actions/setup-python@v2
      with:
        python-version: 3.11

    - name: Install dependencies
      run: |
        pip install -r requirements.txt

    - name: Run tests
      run: |
        python -m pytest tests/

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
    - name: Deploy to Streamlit
      run: echo "Auto-deployed to Streamlit Cloud"
```

📱 Mobile Responsiveness

Streamlit handles most mobile optimization, but you can improve it:

```python
```

```
# Detect mobile
is_mobile = st.sidebar.checkbox("Mobile View", value=False)

if is_mobile:
    # Single column layout
    st.markdown("### Signals")
    display_signal()
    st.markdown("### Charts")
    display_charts()
else:
    # Multi-column layout
    col1, col2 = st.columns(2)
    with col1:
        display_signal()
    with col2:
        display_charts()
```

🌐 Custom Domain Setup

Streamlit Cloud

1. Go to app settings
2. Click "Custom domain"
3. Add CNAME record: `your-app.streamlit.app`

AWS/Heroku

1. Buy domain (Namecheap, GoDaddy)
2. Point A record to your IP
3. Setup SSL with Let's Encrypt

🆘 Troubleshooting

Issue: App not loading

```bash
# Check logs
streamlit run app.py --logger.level=debug
```

Issue: Memory errors

```python
# Use chunking for large files
@st.cache_data
def load_large_csv(file):
    chunks = pd.read_csv(file, chunksize=1000)
    df = pd.concat(chunks)
    return df
```

Issue: Slow performance

- Enable caching: `@st.cache_data`
- Use lazy loading
- Optimize dataframe operations
- Deploy to faster server

📋 Pre-Deployment Checklist

- [ ] All sensitive data in environment variables
- [ ] Error handling for all user inputs
- [ ] Logging configured
- [ ] SSL/HTTPS enabled
- [ ] Backup strategy in place
- [ ] Performance testing done
- [ ] Mobile responsiveness checked
- [ ] Documentation updated
- [ ] README with setup instructions
- [ ] License file added

🎯 Recommended Stack for Production

For Personal Use:

- Streamlit Community Cloud (FREE)
- GitHub for code hosting
- No database needed initially

For Team/Commercial:

- AWS EC2 (t3.small) + Nginx + SSL
- PostgreSQL for data storage
- Redis for caching
- Sentry for error tracking
- Cloudflare for CDN + DDoS protection

Cost Estimate (Commercial):

- EC2: $15/month
- Database: $15/month
- Sentry: Free tier
- Cloudflare: Free tier
- **Total: ~$30/month**

📚 Additional Resources

- Streamlit Documentation
- Heroku Python Guide
- AWS EC2 Tutorial
- Let's Encrypt Setup
- Docker Best Practices

Need help? Open an issue on GitHub or contact the development team.

Happy Trading! 🚀📊