Deployment Guide 🚀

Complete guide for deploying the AI Trading Assistant to various platforms.

Table of Contents

Local Development

Prerequisites

- Python 3.10+
- pip
- Git
- 8GB+ RAM (for ML models)

Setup Steps

1. **Clone Repository:**

```
git clone <repository-url>
cd ai_trading_assistant
```

1. **Create Virtual Environment:**

```
# Windows
python -m venv venv
venv\Scripts\activate

# Linux/Mac
python3 -m venv venv
source venv/bin/activate
```

1. **Install Dependencies:**

```
pip install --upgrade pip
pip install -r requirements.txt
```

1. **Run Application:**

```
streamlit run src/main.py
```

1. **Access Application:**
   Open browser: `http://localhost:8501`

Development Mode

Enable hot-reload:

```
streamlit run src/main.py --server.runOnSave true
```

Enable debug mode in `config/config.py`:

```
DEBUG = True
```

Docker Deployment

Build Image

```
docker build -t ai-trading-assistant:latest .
```

Run Container

Basic:

```
docker run -p 8501:8501 ai-trading-assistant:latest
```

With environment variables:

```
docker run -p 8501:8501 \
  -e NEWS_API_KEY=your_key_here \
  ai-trading-assistant:latest
```

With volume mount (for data persistence):

```
docker run -p 8501:8501 \
  -v $(pwd)/data:/app/data \
  ai-trading-assistant:latest
```

Docker Compose

Create `docker-compose.yml`:

```yaml
version: '3.8'

services:
  app:
    build: .
    ports:
      - "8501:8501"
    environment:
      - NEWS_API_KEY=${NEWS_API_KEY}
    volumes:
      - ./data:/app/data
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8501/_stcore/health"]
      interval: 30s
      timeout: 10s
      retries: 3
```

Run with:

```
docker-compose up -d
```

Multi-Stage Build (Optimized)

Update Dockerfile:

```dockerfile
# Builder 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 --from=builder /root/.local /root/.local
COPY . .

ENV PATH=/root/.local/bin:$PATH

EXPOSE 8501
HEALTHCHECK CMD curl --fail http://localhost:8501/_stcore/health

ENTRYPOINT ["streamlit", "run", "src/main.py", "--server.port=8501", "--serv-
er.address=0.0.0.0"]
```

Streamlit Cloud

Deployment Steps

1. **Push to GitHub:**

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <your-repo-url>
git push -u origin main
```

1. **Deploy:**
   - Visit share.streamlit.io (https://share.streamlit.io)
   - Click “New app”
   - Select your repository
   - Set main file: `src/main.py`
   - Click “Deploy”

Configuration

Create `.streamlit/secrets.toml`:

```
NEWS_API_KEY = "your_api_key_here"
```

Add to `.gitignore`:

```
.streamlit/secrets.toml
```

In Streamlit Cloud dashboard:
- Go to App settings
- Add secrets in “Secrets” section
- Copy contents of secrets.toml

Custom Domain

1. In app settings, go to “General”
2. Add custom domain
3. Update DNS records as instructed

AWS Deployment

Using EC2

1. Launch EC2 Instance

- AMI: Ubuntu 22.04 LTS
- Instance Type: t3.medium (4GB RAM minimum)
- Security Group: Allow ports 22 (SSH), 8501 (Streamlit)

2. Connect and Setup

```
ssh -i your-key.pem ubuntu@your-ec2-ip

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

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

# Clone repository
git clone <your-repo-url>
cd ai_trading_assistant

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

3. Run with PM2

```
# Install PM2
sudo npm install -g pm2

# Create startup script
cat > run.sh << 'EOF'
#!/bin/bash
source venv/bin/activate
streamlit run src/main.py --server.port=8501 --server.address=0.0.0.0
EOF

chmod +x run.sh

# Start with PM2
pm2 start run.sh --name ai-trading-assistant
pm2 save
pm2 startup
```

4. Setup Nginx Reverse Proxy

```
sudo apt install nginx -y

sudo nano /etc/nginx/sites-available/trading-assistant
```

Add configuration:

```
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;
    }
}
```

Enable and restart:

```
sudo ln -s /etc/nginx/sites-available/trading-assistant /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Using ECS (Docker)

1. **Build and push image:**

```
aws ecr create-repository --repository-name ai-trading-assistant

docker build -t ai-trading-assistant .
docker tag ai-trading-assistant:latest <account-id>.dkr.ecr.<region>.amazonaws.com/ai-trading-assistant:latest
docker push <account-id>.dkr.ecr.<region>.amazonaws.com/ai-trading-assistant:latest
```

1. **Create ECS Task Definition**
2. **Create ECS Service**
3. **Setup Application Load Balancer**

Azure Deployment

Using Azure App Service

1. **Create Web App:**

```
az webapp create \
  --resource-group myResourceGroup \
  --plan myAppServicePlan \
  --name ai-trading-assistant \
  --runtime "PYTHON:3.10"
```

1. **Deploy:**

```
az webapp up \
  --name ai-trading-assistant \
  --resource-group myResourceGroup
```

Using Azure Container Instances

```
az container create \
  --resource-group myResourceGroup \
  --name ai-trading-assistant \
  --image <your-image> \
  --cpu 2 \
  --memory 4 \
  --ports 8501 \
  --environment-variables NEWS_API_KEY=your_key
```

Google Cloud Platform

Using Cloud Run

1. **Build and push:**

```
gcloud builds submit --tag gcr.io/PROJECT_ID/ai-trading-assistant
```

1. **Deploy:**

```
gcloud run deploy ai-trading-assistant \
  --image gcr.io/PROJECT_ID/ai-trading-assistant \
  --platform managed \
  --port 8501 \
  --memory 4Gi \
  --allow-unauthenticated
```

Using Compute Engine

Similar to AWS EC2 setup above.

Environment Variables

Required Variables

None (all optional)

Optional Variables

```
# News API (for enhanced sentiment analysis)
NEWS_API_KEY=your_newsapi_key

# Custom settings
STREAMLIT_SERVER_PORT=8501
STREAMLIT_SERVER_ADDRESS=0.0.0.0
```

Setting Environment Variables

Linux/Mac:

```
export NEWS_API_KEY=your_key
```

Windows:

```
set NEWS_API_KEY=your_key
```

Docker:

```
docker run -e NEWS_API_KEY=your_key ...
```

Streamlit Cloud:
Add to secrets in dashboard

Performance Optimization

1. Caching

Already implemented with `@st.cache_data`:
- Data fetching
- Model training
- Calculations

Adjust cache TTL in `config/config.py`:

```
CACHE_TTL = 3600  # 1 hour
```

2. Resource Limits

Docker:

```
docker run \
  --memory=4g \
  --cpus=2 \
  ai-trading-assistant
```

Kubernetes:

```
resources:
  limits:
    memory: "4Gi"
    cpu: "2"
  requests:
    memory: "2Gi"
    cpu: "1"
```

3. Database for Caching

For production, use Redis:

```
import streamlit as st
import redis

@st.cache_resource
def init_redis():
    return redis.Redis(host='localhost', port=6379)
```

4. Load Balancing

Use multiple instances behind load balancer:
- AWS ELB
- Azure Load Balancer
- GCP Load Balancer
- Nginx

5. CDN for Static Assets

- CloudFlare
- AWS CloudFront
- Azure CDN

Monitoring

Application Monitoring

1. **Streamlit Analytics:**
   Built-in analytics in Streamlit Cloud

2. **Custom Logging:**

```
import logging

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    filename='app.log'
)
```

1. **Health Check Endpoint:**
   Already available at: `http://your-app/_stcore/health`

Infrastructure Monitoring

- **AWS:** CloudWatch
- **Azure:** Application Insights
- **GCP:** Cloud Monitoring
- **Self-hosted:** Prometheus + Grafana

SSL/TLS Configuration

Let’s Encrypt (Free SSL)

```
sudo apt install certbot python3-certbot-nginx

sudo certbot --nginx -d your-domain.com
```

Auto-renewal:

```
sudo systemctl status certbot.timer
```

Backup Strategy

Database Backups

Not applicable (stateless application)

Configuration Backups

```
# Backup config
cp config/config.py config/config.py.backup

# Backup secrets
cp .streamlit/secrets.toml .streamlit/secrets.toml.backup
```

Code Backups

Use Git for version control

Scaling

Vertical Scaling

Increase resources:
- More CPU
- More RAM
- Faster disk

Horizontal Scaling

Multiple instances:
- Use load balancer
- Share session state
- Stateless design

Auto-scaling

AWS:

```
# Auto Scaling Group
# Launch Template with application
# Target tracking policy
```

Kubernetes:

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: ai-trading-assistant
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: ai-trading-assistant
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
```

Troubleshooting

Common Issues

Port Already in Use:

```
# Find process
lsof -i :8501
# Kill process
kill -9 <PID>
```

Memory Issues:
- Increase container memory
- Reduce cache TTL
- Optimize data loading

Slow Loading:
- Enable caching

- Use faster instance
- Optimize queries

Connection Timeout:
- Check firewall rules
- Verify security groups
- Check health endpoint

Security Best Practices

1. **Use HTTPS:** Always in production
2. **Environment Variables:** Never commit secrets
3. **Regular Updates:** Keep dependencies updated
4. **Access Control:** Implement authentication if needed
5. **Rate Limiting:** Prevent abuse
6. **Input Validation:** Sanitize user inputs
7. **CORS:** Configure properly

Maintenance

Update Application

```
git pull
pip install -r requirements.txt --upgrade
pm2 restart ai-trading-assistant
```

Update Dependencies

```
pip install --upgrade pip
pip list --outdated
pip install <package> --upgrade
pip freeze > requirements.txt
```

Clear Cache

```
streamlit cache clear
```

Cost Estimation

Streamlit Cloud

- Free tier: Public apps
- Teams: $250/month

AWS EC2

- t3.medium: ~$30/month
- ◦ Storage: ~$5/month
- ◦ Data transfer: Variable

Docker/VPS

- DigitalOcean: $24-48/month
- Linode: $24-48/month
- Hetzner: €20-40/month

Support

For deployment issues:
1. Check logs
2. Review documentation
3. Contact development team

Deployment Checklist:
- [ ] Dependencies installed
- [ ] Environment variables set
- [ ] SSL configured
- [ ] Monitoring enabled
- [ ] Backups configured
- [ ] Security hardened
- [ ] Performance optimized
- [ ] Documentation updated

Good luck with your deployment! 🚀