Contents

TradeWatch Deployment Guide

Overview

This guide covers deployment options for the TradeWatch Global Trade Intelligence Platform, from development to production environments.

Prerequisites

System Requirements

- **CPU**: 4+ cores recommended for production
- **RAM**: 8GB minimum, 16GB recommended
- **Storage**: 50GB+ for database and logs

- **Network**: Reliable internet connection for real-time data feeds

Software Dependencies

- **Node.js**: 18.0.0 or higher
- **Python**: 3.9 or higher
- **PostgreSQL**: 13.0 or higher
- **Git**: Latest version
- **SSL Certificate**: Required for production (HTTPS)

Development Deployment

Local Development Setup

```bash
# Clone repository
git clone https://github.com/radsilent/TradeWatch.git
cd TradeWatch

# Install frontend dependencies
npm install

# Install backend dependencies
cd ai-processing
pip install -r requirements.txt
cd ..

# Set up environment variables
cp .env.example .env
# Edit .env with your configuration

# Start development servers
npm run dev &
cd ai-processing && python enhanced_real_data_api.py &
```

Development Environment Variables

```bash
# .env file
NODE_ENV=development
API_BASE_URL=http://localhost:8001
POSTGRES_HOST=localhost
POSTGRES_PORT=5432
POSTGRES_DB=tradewatch_dev
POSTGRES_USER=your_username
POSTGRES_PASSWORD=your_password
```

Production Deployment

Server Configuration

Option 1: Traditional VPS/Dedicated Server

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

# Install Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs

# Install Python and pip
sudo apt-get install -y python3.9 python3-pip python3-venv

# Install PostgreSQL
sudo apt-get install -y postgresql postgresql-contrib

# Install Nginx (reverse proxy)
sudo apt-get install -y nginx

# Install PM2 (process manager)
sudo npm install -g pm2
```

Option 2: Docker Deployment

```
# Dockerfile.frontend
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY . .
RUN npm run build
EXPOSE 5173
CMD ["npm", "run", "preview"]
```

```
# Dockerfile.backend
FROM python:3.9-slim
WORKDIR /app
COPY ai-processing/requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY ai-processing/ ./
EXPOSE 8001
CMD ["python", "enhanced_real_data_api.py"]
```

Production Build Process

Frontend Build

```
# Build optimized production bundle
npm run build

# Test production build locally
npm run preview
```

Backend Configuration

```
# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install production dependencies
pip install -r requirements.txt

# Set production environment variables
export ENVIRONMENT=production
export DEBUG=false
export CORS_ORIGINS="https://yourdomain.com"
```

Database Setup

PostgreSQL Configuration

```
-- Create production database
CREATE DATABASE tradewatch_prod;

-- Create application user
CREATE USER tradewatch_user WITH PASSWORD 'secure_password';

-- Grant privileges
GRANT ALL PRIVILEGES ON DATABASE tradewatch_prod TO tradewatch_user;

-- Connect to database and create schema
\c tradewatch_prod
```

```
# Run database migrations
python database/create_schema.py
```

Nginx Configuration

```
# /etc/nginx/sites-available/tradewatch
server {
    listen 80;
    server_name yourdomain.com;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com;

    ssl_certificate /path/to/ssl/certificate.crt;
    ssl_certificate_key /path/to/ssl/private.key;

    # Frontend
    location / {
        proxy_pass http://localhost:5173;
        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;
    }

    # Backend API
    location /api {
        proxy_pass http://localhost:8001;
        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;
    }
}
```

Process Management with PM2

PM2 Configuration

```
// ecosystem.config.js
module.exports = {
  apps: [
    {
      name: 'tradewatch-frontend',
      script: 'npm',
      args: 'run preview',
      cwd: '/path/to/TradeWatch',
      env: {
        NODE_ENV: 'production',
        PORT: 5173
      }
    },
    {
      name: 'tradewatch-backend',
      script: 'enhanced_real_data_api.py',
      cwd: '/path/to/TradeWatch/ai-processing',
      interpreter: 'python3',
      env: {
        ENVIRONMENT: 'production',
        PORT: 8001
      }
    }
  ]
};
```

Start Production Services

```
# Start applications with PM2
pm2 start ecosystem.config.js

# Save PM2 configuration
pm2 save

# Setup PM2 to start on system boot
pm2 startup
```

Cloud Deployment

AWS Deployment

Using EC2

```
# Launch EC2 instance (t3.medium or larger)
# Configure security groups:
# - Port 22 (SSH)
# - Port 80 (HTTP)
# - Port 443 (HTTPS)
# - Port 5432 (PostgreSQL - internal only)

# Connect and deploy
ssh -i your-key.pem ubuntu@your-ec2-ip
# Follow traditional VPS setup steps
```

Using ECS (Docker)

```
# docker-compose.yml
version: '3.8'
services:
  frontend:
    build:
      context: .
      dockerfile: Dockerfile.frontend
    ports:
      - "5173:5173"
    environment:
      - NODE_ENV=production
    depends_on:
      - backend

  backend:
    build:
      context: .
      dockerfile: Dockerfile.backend
    ports:
      - "8001:8001"
    environment:
      - ENVIRONMENT=production
      - POSTGRES_HOST=database
    depends_on:
      - database

  database:
    image: postgres:13
    environment:
      - POSTGRES_DB=tradewatch
      - POSTGRES_USER=tradewatch
      - POSTGRES_PASSWORD=secure_password
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:
```

Google Cloud Platform

Using Cloud Run

```
# Build and deploy backend
gcloud builds submit --tag gcr.io/PROJECT_ID/tradewatch-backend ai-processing/
gcloud run deploy tradewatch-backend --image gcr.io/PROJECT_ID/tradewatch-backend --platform managed

# Build and deploy frontend
gcloud builds submit --tag gcr.io/PROJECT_ID/tradewatch-frontend .
gcloud run deploy tradewatch-frontend --image gcr.io/PROJECT_ID/tradewatch-frontend --platform managed
```

Microsoft Azure

Using Container Instances

```
# Create resource group
az group create --name TradeWatchRG --location eastus

# Deploy containers
az container create --resource-group TradeWatchRG --name tradewatch-app --image your-registry/tradewatc
```

Environment Configuration

Production Environment Variables

```
# Backend (.env)
ENVIRONMENT=production
DEBUG=false
SECRET_KEY=your-secret-key
DATABASE_URL=postgresql://user:password@host:port/database
CORS_ORIGINS=https://yourdomain.com
API_RATE_LIMIT=10000
LOG_LEVEL=INFO

# Frontend (.env.production)
VITE_API_BASE_URL=https://yourdomain.com/api
VITE_ENVIRONMENT=production
VITE_ENABLE_ANALYTICS=true
```

Monitoring and Logging

Application Monitoring

```
# Install monitoring tools
npm install -g pm2-logrotate
pm2 install pm2-server-monit

# Configure log rotation
pm2 set pm2-logrotate:max_size 10M
pm2 set pm2-logrotate:retain 30
```

Health Checks

```
# API health check endpoint
curl https://yourdomain.com/api/health

# Database connection check
curl https://yourdomain.com/api/health/database
```

Log Management

```
# View application logs
pm2 logs tradewatch-backend
pm2 logs tradewatch-frontend

# Monitor real-time logs
tail -f /var/log/nginx/access.log
tail -f /var/log/nginx/error.log
```

Security Considerations

SSL/TLS Configuration

- Use strong SSL certificates (Let's Encrypt recommended)
- Enable HTTP Strict Transport Security (HSTS)
- Configure secure headers in Nginx

Database Security

- Use strong passwords for database users
- Restrict database access to application servers only
- Enable SSL connections for database
- Regular security updates and patches

API Security

- Implement rate limiting
- Use CORS restrictions for production
- Validate all input data
- Monitor for suspicious activity

Backup and Recovery

Database Backups

```
# Automated daily backups
0 2 * * * pg_dump -h localhost -U tradewatch_user tradewatch_prod > /backups/tradewatch_$(date +\%Y\%m\

# Restore from backup
psql -h localhost -U tradewatch_user tradewatch_prod < backup_file.sql
```

Application Backups

```
# Backup application files
tar -czf tradewatch_app_$(date +%Y%m%d).tar.gz /path/to/TradeWatch

# Backup configuration
cp /etc/nginx/sites-available/tradewatch /backups/
cp /path/to/TradeWatch/.env /backups/
```

Scaling Considerations

Horizontal Scaling

- Load balancer configuration for multiple application instances
- Database read replicas for improved performance

- CDN integration for static assets

Performance Optimization

- Enable Nginx caching for static assets
- Database query optimization and indexing
- Redis caching for frequently accessed data
- Image optimization and compression

Troubleshooting

Common Issues

1. **Port conflicts**: Ensure ports 5173 and 8001 are available
2. **Database connections**: Check PostgreSQL service status
3. **CORS errors**: Verify CORS_ORIGINS configuration
4. **SSL certificate**: Ensure certificates are valid and properly configured

Debug Commands

```
# Check service status
systemctl status nginx
systemctl status postgresql
pm2 status

# View error logs
journalctl -u nginx -f
pm2 logs --err
```

Deployment Guide v2.1.0 Last Updated: January 2025