Deployment Guide

This guide provides detailed instructions for deploying the AI Trading App on your Ubuntu VPS.

System Requirements

- **OS**: Ubuntu 20.04 LTS or later
- **RAM**: Minimum 2GB (recommended 4GB)
- **CPU**: Single core minimum
- **Disk**: 5GB free space
- **Network**: Stable internet connection

Pre-deployment Checklist

- [ ] VPS with Ubuntu installed
- [ ] Root or sudo access
- [ ] Node.js v18.20.8 installed
- [ ] Go 1.21.6 or later installed
- [ ] OpenAlgo instance running with API key
- [ ] Abacus.AI API key obtained

Installation Steps

1. Prepare the VPS

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

# Install required system dependencies
sudo apt-get install -y build-essential git curl wget tesseract-ocr
```

2. Upload the Project

Transfer the `trading-app` directory to `/root/trading-app` on your VPS.

```
# Using scp from your local machine:
scp -r trading-app root@YOUR_VPS_IP:/root/

# Or using rsync:
rsync -avz trading-app root@YOUR_VPS_IP:/root/
```

3. Run Deployment Script

```
# Navigate to project directory
cd /root/trading-app

# Make scripts executable
chmod +x *.sh

# Run deployment
./deploy.sh
```

The script will prompt you for:
- **OpenAlgo API Key**: Your OpenAlgo API key
- **Abacus.AI API Key**: Your Abacus.AI API key

4. Verify Installation

```
# Check service status
./status.sh

# View logs
./logs.sh backend
./logs.sh frontend
```

Services should show as “active (running)”.

5. Access the Application

Open your browser and navigate to:

```
http://YOUR_VPS_IP:3000
```

Login with default credentials:
- Username: `admin`
- Password: `admin123`

Configuration

Backend Configuration

Edit `/root/trading-app/backend/.env`:

```
# Database
DB_PATH=./data/trading.db

# Upload directory
UPLOAD_DIR=./data/uploads

# Server port
PORT=8080

# OpenAlgo configuration
OPENALGO_URL=https://openalgo.mywire.org
OPENALGO_API_KEY=your_openalgo_api_key_here

# Abacus.AI API key for AI chat
ABACUS_API_KEY=your_abacus_api_key_here

# JWT Secret (change in production)
JWT_SECRET=your_secret_key_change_this_in_production
```

Frontend Configuration

Edit `/root/trading-app/frontend/.env.local`:

```
NEXT_PUBLIC_API_URL=http://YOUR_VPS_IP:8080
NEXT_PUBLIC_WS_URL=ws://YOUR_VPS_IP:8080/ws
```

Restart After Configuration Changes

```
./restart.sh
```

Setting Up Firewall (Optional but Recommended)

```
# Install UFW
sudo apt-get install -y ufw

# Allow SSH (IMPORTANT: Do this first!)
sudo ufw allow 22/tcp

# Allow application ports
sudo ufw allow 3000/tcp  # Frontend
sudo ufw allow 8080/tcp  # Backend

# Enable firewall
sudo ufw enable

# Check status
sudo ufw status
```

Setting Up Reverse Proxy with Nginx (Optional)

For production with custom domain:

```
# Install Nginx
sudo apt-get install -y nginx certbot python3-certbot-nginx

# Create Nginx configuration
sudo nano /etc/nginx/sites-available/trading-app
```

Add the following configuration:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        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;
    }

    location /api {
        proxy_pass http://localhost:8080;
        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;
    }

    location /ws {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
    }
}
```

Enable the site:

```
# Create symbolic link
sudo ln -s /etc/nginx/sites-available/trading-app /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# Restart Nginx
sudo systemctl restart nginx

# Get SSL certificate (after DNS is configured)
sudo certbot --nginx -d your-domain.com
```

Monitoring and Maintenance

View Logs

```
# Real-time backend logs
./logs.sh backend

# Real-time frontend logs
./logs.sh frontend

# System logs
sudo journalctl -u trading-backend -f
sudo journalctl -u trading-frontend -f
```

Monitor Resources

```
# Memory usage
free -h

# Disk usage
df -h

# CPU and process info
htop  # or top
```

Backup Database

```
# Create backup directory
mkdir -p /root/backups

# Backup database
cp /root/trading-app/backend/data/trading.db /root/backups/trading-$(date +
%Y%m%d_%H%M%S).db

# Set up automatic daily backup (cron)
crontab -e
# Add this line:
# 0 2 * * * cp /root/trading-app/backend/data/trading.db /root/backups/trading-$(date
+\%Y\%m\%d_\%H\%M\%S).db
```

Updating the Application

```
cd /root/trading-app

# Stop services
./stop.sh

# Pull updates or upload new files
# (upload your updated files here)

# Rebuild backend
cd backend
go build -o trading-server ./cmd/main.go

# Rebuild frontend
cd ../frontend
npm install --legacy-peer-deps
npm run build

# Start services
cd ..
./start.sh
```

Uninstalling

To completely remove the application:

```
# Stop services
cd /root/trading-app
./stop.sh

# Disable services
sudo systemctl disable trading-backend
sudo systemctl disable trading-frontend

# Remove service files
sudo rm /etc/systemd/system/trading-backend.service
sudo rm /etc/systemd/system/trading-frontend.service

# Reload systemd
sudo systemctl daemon-reload

# Remove application directory
cd /root
rm -rf trading-app
```

Troubleshooting Deployment

Issue: Services fail to start

Solution:

```
# Check logs for errors
./logs.sh backend
./logs.sh frontend

# Verify configuration
cat backend/.env
cat frontend/.env.local

# Check if ports are available
sudo netstat -tulpn | grep :8080
sudo netstat -tulpn | grep :3000
```

Issue: Permission denied errors

Solution:

```
# Fix ownership
sudo chown -R $USER:$USER /root/trading-app

# Fix permissions
chmod +x /root/trading-app/*.sh
```

Issue: Out of memory

Solution:

```
# Check memory
free -h

# Add swap space if needed
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

# Make permanent
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Security Best Practices

1. **Change Default Credentials**: Immediately after deployment
2. **Use Strong Passwords**: For all user accounts
3. **Enable Firewall**: As shown in the firewall section
4. **Regular Updates**: Keep system and application updated
5. **Secure API Keys**: Never expose API keys in public repositories
6. **Use HTTPS**: Set up SSL/TLS with Let’s Encrypt
7. **Regular Backups**: Backup database and configuration files
8. **Monitor Logs**: Regularly check logs for suspicious activity

Support and Help

If you encounter issues during deployment:

1. Check the main README.md troubleshooting section
2. Review logs: `./logs.sh backend` or `./logs.sh frontend`
3. Verify all prerequisites are met
4. Ensure API keys are correct
5. Check network connectivity to OpenAlgo and Abacus.AI

Deployment completed successfully? You're ready to start trading with AI! 🚀