WebhookTrader 逐步开发部署指南

项目文件结构

```
webhook-trader-server/
├── main.py          # FastAPI 主入口，webhook 接收 + 路由
├── executor.py      # 交易执行器（Paper Trading / Live Trading）
├── config.py        # 配置管理（从 .env 读取）
├── .env.example     # 环境变量模板
├── requirements.txt # Python 依赖
└── test_webhook.py  # 测试脚本，模拟 TradingView 发送 webhook
```

Step 1：本地开发 & 测试

1.1 在本地创建项目

bash

```
mkdir webhook-trader-server
cd webhook-trader-server
# 把上面的文件放到这个目录下
```

1.2 安装依赖

bash

```
# 建议用虚拟环境
python -m venv venv
source venv/bin/activate    # Mac/Linux
# venv\Scripts\activate     # Windows

pip install -r requirements.txt
```

1.3 创建 .env 配置文件

bash

```
cp .env.example .env
# 默认是 PAPER_TRADING=true，不需要改任何东西就能跑
```

1.4 启动服务器

bash

```
uvicorn main:app --host 0.0.0.0 --port 8000 --reload
```

你会看到类似输出：

```
2026-02-09 12:00:00 [INFO] ================================================
2026-02-09 12:00:00 [INFO] WebhookTrader server starting...
2026-02-09 12:00:00 [INFO] Mode: PAPER TRADING
2026-02-09 12:00:00 [INFO] Exchange: binance
2026-02-09 12:00:00 [INFO] Webhook endpoint: POST /api/webhook/tv
2026-02-09 12:00:00 [INFO] ================================================
```

1.5 测试 webhook

打开另一个终端窗口：

```
bash

python test_webhook.py
```

你会看到每个测试的请求和响应，包括：

- 健康检查
- BUY BTC/USDT（成功）
- SELL ETH/USDT（成功）
- BUY SOL/USDT（成功）
- 无效 action（返回 400）
- Raw webhook（调试用）
- 查看所有信号历史

也可以手动用 curl 测试：

```
bash

curl -X POST http://localhost:8000/api/webhook/tv \
 -H "Content-Type: application/json" \
 -d '{
  "strategy": "EMA Crossover",
  "action": "buy",
  "ticker": "BTCUSDT",
  "price": 97500,
  "qty": 0.01
 }'
```

1.6 查看 API 文档

浏览器打开 http://localhost:8000/docs ，FastAPI 自动生成的 Swagger UI， 可以直接在页面上测试每个接口。

Step 2：部署后端到 GCP 服务器

2.1 上传代码到服务器

```bash
# 在本地项目目录执行
scp -r ./* yu_xu@你的外部IP:/home/yu_xu/webhook-trader-server/
```

2.2 SSH 到服务器安装环境

```bash
ssh yu_xu@你的外部IP

# 安装 Python 和 pip（如果没有）
sudo apt update
sudo apt install -y python3 python3-pip python3-venv

# 进入项目目录
cd /home/yu_xu/webhook-trader-server

# 创建虚拟环境并安装依赖
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

# 创建 .env 文件
cp .env.example .env
```

2.3 测试运行

```bash
uvicorn main:app --host 0.0.0.0 --port 8000
```

浏览器访问 `http://你的外部IP:8000` ，看到 JSON 响应说明后端跑起来了。

> **注意**：需要在 GCP 防火墙开放 8000 端口，或者直接用 Nginx 反向代理（推荐，见 Step 3）。

2.4 用 systemd 让服务后台运行

创建 service 文件：

```bash
sudo nano /etc/systemd/system/webhook-trader.service
```

写入以下内容：

```ini
[Unit]
Description=WebhookTrader FastAPI Server
After=network.target

[Service]
User=yu_xu
WorkingDirectory=/home/yu_xu/webhook-trader-server
ExecStart=/home/yu_xu/webhook-trader-server/venv/bin/uvicorn main:app --host 127.0.0.1 --port 8000
Restart=always
RestartSec=5
Environment=PATH=/home/yu_xu/webhook-trader-server/venv/bin

[Install]
WantedBy=multi-user.target
```

启动并设置开机自启：

```bash
sudo systemctl daemon-reload
sudo systemctl start webhook-trader
sudo systemctl enable webhook-trader

# 查看状态
sudo systemctl status webhook-trader

# 查看日志
sudo journalctl -u webhook-trader -f
```

Step 3：配置 Nginx 反向代理

3.1 创建 Nginx 配置

```bash
```

```
sudo nano /etc/nginx/sites-available/webhook-trader
```

写入：

```nginx
server {
    listen 80;
    server_name 你的外部IP;  # 之后换成你的域名

    # 后端 API — 反向代理到 FastAPI
    location /api/ {
        proxy_pass http://127.0.0.1:8000;
        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;
    }

    # 前端静态文件（Step 4 部署后启用）
    location / {
        root /home/yu_xu/webhook-trader-frontend/dist;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

3.2 启用配置

```bash
sudo ln -s /etc/nginx/sites-available/webhook-trader /etc/nginx/sites-enabled/
sudo rm /etc/nginx/sites-enabled/default   # 删除默认配置
sudo nginx -t                              # 测试配置是否正确
sudo systemctl reload nginx
```

3.3 测试

```bash
# 在任意机器上执行
curl http://你的外部IP/api/health
```

应该返回：

```json
```

{"status":"healthy","exchange_connected":true,"signals_processed":0,...}

Step 4：前端打包部署

4.1 本地创建 Vite 项目

```bash
npm create vite@latest webhook-trader-frontend -- --template react
cd webhook-trader-frontend
npm install
```

4.2 替换代码

将之前的 `trading-demo.jsx` 的内容复制到 `src/App.jsx`。

修改前端代码中的模拟逻辑，改为调用真实后端 API：

```javascript
// 例如：发送模拟信号时，改为调用后端
const handleSimulate = async () => {
 const response = await fetch('/api/webhook/tv', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
   strategy: 'Manual Signal',
   action: simAction,
   ticker: simPair.replace('/', ''),
   price: parseFloat(simPrice),
   qty: parseFloat(simQty),
  }),
 });
 const result = await response.json();
 // 更新 UI ...
};
```

4.3 打包并上传

```bash
npm run build

# 上传到服务器
scp -r dist/ yu_xu@你的外部IP:/home/yu_xu/webhook-trader-frontend/dist
```

4.4 验证

浏览器访问 `http://你的外部IP`，应该能看到 Dashboard 页面。 点击 Simulate 发送信号，应该能在 Logs 里看到后端处理结果。

Step 5：连接 TradingView（端到端测试）

5.1 在 TradingView 创建 Alert

1. 打开 TradingView，进入任意图表
2. 添加你的 Pine Script 策略（或用内置指标）
3. 点击「Alert」→ 创建新 Alert
4. Condition 选你的策略
5. **Webhook URL** 填入：`http://你的外部IP/api/webhook/tv`
6. **Message** 填入：

```
json

{
  "strategy": "{{strategy.order.alert_message}}",
  "action": "{{strategy.order.action}}",
  "ticker": "{{ticker}}",
  "price": {{close}},
  "qty": {{strategy.order.contracts}},
  "timestamp": "{{timenow}}"
}
```

7. 保存 Alert

5.2 验证信号

在服务器上实时看日志：

```
bash

sudo journalctl -u webhook-trader -f
```

当 TradingView 策略触发时，你会看到 webhook 到达、解析、和执行的日志。

同时访问 `http://你的外部IP/api/signals` 查看所有接收到的信号历史。

后续优化方向

- [ ] 配置 HTTPS（用 Certbot + Let's Encrypt 免费证书）
- [ ] 添加 webhook 签名验证（防止恶意请求）
- [ ] 接入 PostgreSQL 替换内存存储
- [ ] 添加 WebSocket 实现前端实时推送
- [ ] 接入真实交易所 API（取消注释 requirements.txt 中的 ccxt）
- [ ] 添加风控模块（最大仓位、止损、每日亏损上限）