# NHS Waiting List Alert — 技术实现与部署手册（中英双语 · 完整代码示例）

**版本 1.2 · 更新 2025‑07‑04**

本文档在 1.1 版基础上补充：

1. **官方软件下载直链** 及 CLI 安装命令；
2. **脚本 & 样板代码**（ETL、FastAPI、Docker Compose）；
3. Grafana **监控仪表模板 JSON**；
4. **GP 白标落地流程脚本**（二维码生成、iframe SSO 示例）；
5. 全程中英文对照，方便跨国团队协作。

## 目录 / Table of Contents

1. 背景与目的 / Scope & Purpose
2. 架构图 / Architecture Overview
3. 工具下载与安装 / Tool Download & Installation
4. 代码示例 / Code Samples
5. 监控仪表模板 / Grafana Dashboards
6. GP 白标脚本 / White‑Label GP Scripts
7. 合规 & 运营附录

## 1. 背景与目的 / Scope & Purpose

*内容与上一版相同，此处略.*

## 2. 架构图 / Architecture Overview

*(沿用 1.1 版 Mermaid，不再复述)*

## 3. 工具下载与安装 / Tool Download & Installation

### 3.1 必备工具列表

| 工具 | 官网下载 | Windows 安装 | Ubuntu 安装 |  |  |
| --- | --- | --- | --- | --- | --- |
| **Docker Desktop** | <https://docs.docker.com/desktop/> | MSI 安装 → 勾选 “Use WSL 2” | `curl -fsSL <https://get.docker.com> | sh` |  |
| **Docker Compose v2** | 同上 | 附带在 Desktop | sudo apt install docker-compose-plugin |  |  |
| **Git** | <https://git-scm.com/downloads> | exe | sudo apt install git |  |  |
| **Fly.io CLI** | <https://fly.io/docs/hands-on/install-flyctl/> | PowerShell: `iwr <https://fly.io/install.ps1> | iex` | `curl -L <https://fly.io/install.sh> | sh` |
| **Python 3.11** | <https://www.python.org/downloads/> | Microsoft Store or MSVC installer | sudo add-apt-repository ppa:deadsnakes/ppa && sudo apt install python3.11 |  |  |
| **Prefect 2** | <https://docs.prefect.io/> | pip install prefect==2.\* | 同左 |  |  |
| **Grafana OSS** | <https://grafana.com/grafana/download> | ZIP + nssm 服务 | sudo apt install -y grafana |  |  |

**WSL 2 优化**：wsl --set-default-version 2 → wsl --shutdown → 在 ~/.wslconfig 添加：

[wsl2]  
memory=4GB  
processors=4

### 3.2 一键部署脚本

在项目根目录提供 setup.ps1（Windows）与 setup.sh（Linux）——已包含拉 Git、写 .env、启动 Compose 全流程。

# setup.sh 关键片段  
read -p "Enter WhatsApp Token: " TOKEN  
cat <<EOF > .env  
DB\_PASSWORD=$(openssl rand -base64 16)  
WHATSAPP\_TOKEN=$TOKEN  
META\_PHONE\_ID=your\_phone\_id  
EOF  
docker compose pull && docker compose up -d

### 3.3 小白快速上手 Checklist （逐步详解）

**目标**：让零基础用户在 20 分钟内完成本地全链路部署并收到第一条 WhatsApp 提醒。

| 步骤 | 详细操作 (Windows Server / Windows 10 + WSL 2) | 详细操作 (Ubuntu 22.04 LTS) |
| --- | --- | --- |
| **1. 安装 Docker & Git** | 1. 下载 **Docker Desktop** MSI：<https://desktop.docker.com/win/stable/amd64/Docker%20Desktop%20Installer.exe> 并勾选 *Enable WSL 2*. 2. 安装 **Git for Windows**：<https://git-scm.com/download/win>，保持默认选项。 3. 重新登录，docker -v 与 git --version 输出版本号即成功。 | ```bash |
| sudo apt update && sudo apt -y install docker.io git |  |  |
| sudo systemctl enable –now docker |  |  |
| sudo usermod -aG docker $USER && newgrp docker |  |  |
| ```确保 docker version 与 git --version 正常输出。 |  |  |
| **2. 克隆代码库** | 打开 **PowerShell**：git clone https://github.com/yourorg/waiting-alert.git``cd waiting-alert | 在终端执行同样两行 Git 命令。 |
| **3. 运行一键脚本** | *PowerShell*：Set-ExecutionPolicy RemoteSigned -Scope CurrentUser（首次）./setup.ps1按提示粘贴 **WhatsApp Cloud 永久令牌**，其余回车默认。 | chmod +x setup.sh && ./setup.sh 粘贴 **WHATSAPP\_TOKEN** → 回车。脚本将：① 自动生成随机 DB\_PASSWORD② 写 .env③ 拉取镜像并启动 docker compose. |
| **4. 验证容器状态** | docker compose ps 应显示：pg, etl, api, grafana 均为 **Up (healthy)**。 | 同左 |
| **5. 浏览器自检** | 1. 打开 http://<服务器IP>:8000/docs → 应见 FastAPI Swagger UI。2. 打开 http://<服务器IP>:3000→Grafana 登录页，初始账号 admin / admin。 | 同左，若本机部署可用 localhost 替代 IP。 |
| **6. 导入 WhatsApp Flow** | 1. 登录 **Meta Business Suite** → WhatsApp → *Flows*。2. 点击 *Import*，上传 flow/waiting\_alert\_setup.json（已随仓库提供）。3. 设置 **Webhook URL**：https://<服务器域名>/wa/webhook 并点 *Test*. | 同左；若仅本地测试，可暂留空 Webhook，使用 API 手动触发。 |
| **7. 实际消息测试** | 1. 用配置的商务号码向自己的 WhatsApp 发送 hi。2. 应收到欢迎语并提示输入邮编。输入 SW1A1AA、选择 Cataract。3. 系统返回等待时间图卡表示成功。 | 同左 |
| **8. （可选）生成二维码海报** | 执行：python scripts/gen\_qr.py GPODS123 SW1A1AA会在 qr/ 目录生成 PNG，可打印张贴。 | 同左 |

**到此为止**：已完成从零到服务运行、数据抓取、用户交互的全链路部署。

## 4. 代码示例 / Code Samples$1

### 4.5 预测服务 predict/service.py

"""Daily OLS + LOESS forecast for each (provider, specialty)."""  
import asyncpg, os, pandas as pd  
from statsmodels.regression.linear\_model import OLS  
from statsmodels.tools import add\_constant  
from statsmodels.nonparametric.smoothers\_lowess import lowess  
  
DSN = os.getenv("DATABASE\_URL")  
async def forecast():  
 pool = await asyncpg.create\_pool(DSN)  
 async with pool.acquire() as conn:  
 pairs = await conn.fetch("SELECT DISTINCT provider\_code, specialty FROM waiting\_times")  
 for provider, spec in pairs:  
 df = await conn.fetch(  
 """SELECT weeks, dataset\_date FROM waiting\_times  
 WHERE provider\_code=$1 AND specialty=$2  
 ORDER BY dataset\_date""", provider, spec)  
 pdf = pd.DataFrame(df, columns=["weeks", "dataset\_date"])  
 if len(pdf) < 4:  
 continue  
 y = pdf["weeks"].values  
 X = add\_constant(range(len(y)))  
 beta = OLS(y, X).fit()  
 next\_val = float(beta.predict([[1, len(y)]])[0])  
 if beta.rsquared < 0.2:  
 next\_val = lowess(y, range(len(y)), frac=0.6)[-1, 1]  
 await conn.execute("""  
 INSERT INTO waiting\_predicted(provider\_code,specialty,next\_weeks,pred\_on)  
 VALUES($1,$2,$3,now())  
 ON CONFLICT (provider\_code,specialty)  
 DO UPDATE SET next\_weeks=EXCLUDED.next\_weeks, pred\_on=EXCLUDED.pred\_on  
 """, provider, spec, next\_val)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import asyncio; asyncio.run(forecast())

### 4.6 数据库迁移脚本 db/schema.sql

CREATE TABLE IF NOT EXISTS waiting\_times (  
 provider\_code text,  
 specialty text,  
 weeks numeric,  
 source text,  
 dataset\_date date,  
 PRIMARY KEY (provider\_code,specialty,dataset\_date)  
);  
CREATE TABLE IF NOT EXISTS waiting\_predicted (  
 provider\_code text PRIMARY KEY,  
 specialty text,  
 next\_weeks numeric,  
 pred\_on timestamptz  
);  
-- nearest\_stats(materialized view)  
CREATE MATERIALIZED VIEW IF NOT EXISTS nearest\_stats AS  
SELECT wt.provider\_code, wt.specialty, wt.weeks, wp.next\_weeks  
FROM waiting\_times wt  
JOIN waiting\_predicted wp USING (provider\_code,specialty)  
WHERE wt.dataset\_date = (SELECT MAX(dataset\_date) FROM waiting\_times wt2 WHERE wt2.provider\_code=wt.provider\_code AND wt2.specialty=wt.specialty);  
CREATE INDEX IF NOT EXISTS idx\_nearest ON nearest\_stats(provider\_code);

Run migration:

cat db/schema.sql | docker compose exec -T pg psql -U postgres

### 4.7 Dockerfiles

``

FROM python:3.11-slim  
WORKDIR /app  
COPY etl/requirements.txt .  
RUN pip install -r requirements.txt  
COPY etl/ .  
CMD ["python", "runner.py"]

``

FROM python:3.11-slim  
WORKDIR /srv  
COPY api/requirements.txt .  
RUN pip install -r requirements.txt  
COPY api/ .  
ENV PYTHONUNBUFFERED=1  
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]

### 4.8 .env.example

DB\_PASSWORD=changeme  
DATABASE\_URL=postgresql://postgres:${DB\_PASSWORD}@pg:5432/postgres  
WHATSAPP\_TOKEN=EAAG...  
META\_PHONE\_ID=123456789  
PREFECT\_API\_KEY=pref-...  
FLY\_API\_TOKEN=

### 4.9 WhatsApp Flow JSON 模板

{  
 "name": "waiting\_alert\_setup",  
 "language": "en",  
 "version": "1",  
 "components": [  
 {  
 "type": "FLOW",  
 "flow": {  
 "name": "setup",  
 "screens": [  
 {"id": "welcome", "type": "TEXT", "text": "👋 欢迎！请输入邮编开始："},  
 {"id": "postcode", "type": "INPUT"},  
 {"id": "procedure", "type": "PICKLIST", "list": ["Cataract","Hip","Knee"]},  
 {"id": "thanks", "type": "TEXT", "text": "已记录！稍后将推送等待信息。"}  
 ]  
 }  
 }  
 ]  
}

上传至 *WhatsApp Manager → Flows*。Flow Webhook URL 设为 https://api.waitingalert.uk/wa/webhook。

### 4.10 本地端到端测试脚本

# 1. 触发 ETL (月度)  
docker compose exec etl python runner.py  
# 2. Inline API 查询  
curl -G 'http://localhost:8000/stats' --data-urlencode 'postcode=SW1A1AA' --data-urlencode 'procedure=CAT'  
# 3. 模拟阈值触发  
docker compose exec pg psql -U postgres -c "UPDATE waiting\_predicted SET next\_weeks=60 WHERE provider\_code='RJU';"  
# 4. 手动调用发送  
curl -X POST http://localhost:8000/admin/dispatch?user=1

出现 WhatsApp 消息即表示后端到端 OK。

## 5. 监控仪表模板 / Grafana Dashboards. 监控仪表模板 / Grafana Dashboards 监控仪表模板 / Grafana Dashboards

Screenshot & 说明：

1. **ETL Success Gauge** – 绿色≥0.95；<0.8 红色报警。
2. **WhatsApp Dispatch P95 Latency** – 阈值 1500 ms。
3. **Postgres Connection Count** – 连接 >90% 触发 Alertmanager。

在 Grafana → Dashboards → “Import” → 粘贴 JSON.

## 6. GP 白标脚本 / White‑Label GP Scripts

### 6.1 二维码批量生成

import qrcode, sys, csv  
with open('gp\_list.csv') as f:  
 for ods, postcode in csv.reader(f):  
 url = f"https://{ods}.waitingalert.uk/flow?pc={postcode}"  
 img = qrcode.make(url)  
 img.save(f"qr/{ods}.png")

### 6.2 iframe SSO 示例

<script src="https://unpkg.com/jwt-decode@4/dist/jwt-decode.js"></script>  
<iframe id="wa-frame" width="100%" height="600" src="about:blank"></iframe>  
<script>  
 const token = "{{SIGNED\_JWT\_FROM\_BACKEND}}";  
 document.getElementById('wa-frame').src = `https://ods123.waitingalert.uk/dashboard?jwt=${token}`;  
</script>

*JWT payload*:

{  
 "ods":"A12345",  
 "exp": 1750000000,  
 "scope":["view","export"]  
}

## 7. 合规 & 运营附录

* ICO 注册表单位置：<https://ico.org.uk/for-organisations/data-protection-fee/>
* WA 模板注册步骤：Business Settings → WhatsApp → Message Templates → 上传 JSON。
* 私立 Lead API 回调样例：POST /lead/callback with body {lead\_id, status, revenue}.

**完 / End of Document**