

# NOAH 知识图谱系统 — 完整安装与配置指南

适用系统： macOS / Linux（Windows 用户请参考附录） 技术水平要求： 会打开终端即可  
预计完成时间： 约 45 – 60 分钟（首次安装）

## 目录

- [1. 系统要求](#)
- [2. 第一步：安装 PostgreSQL 与 PostGIS](#)
- [3. 第二步：安装 Neo4j](#)
- [4. 第三步：安装 Python 环境](#)
- [5. 第四步：获取项目代码](#)
- [6. 第五步：配置数据库连接](#)
- [7. 第六步：导入 NOAH 数据库](#)
- [8. 第七步：运行数据迁移](#)
- [9. 第八步：启动 Web 界面](#)
- [10. 快速验证清单](#)
- [11. 常见问题与解决方案](#)
- [12. 附录：Docker 一键部署（可选）](#)

## 1. 系统要求

在开始之前，请确认你的电脑满足以下要求：

项目	最低要求	推荐配置
操作系统	macOS 12+ / Ubuntu 20.04+	macOS 14+ / Ubuntu 22.04+
内存 (RAM)	8 GB	16 GB

项目	最低要求	推荐配置
硬盘空间	5 GB 可用	10 GB 可用
Python	3.10+	3.12
网络	需要联网（下载依赖）	—

提示：内存要求主要来自 Neo4j（约需 4 GB）。如果你的电脑只有 8 GB 内存，请关闭其他大型程序（Chrome、Slack 等）再运行。

## 2. 第一步：安装 PostgreSQL 与 PostGIS

PostgreSQL 是项目的源数据库，PostGIS 是它的空间扩展（处理地理坐标数据）。

### macOS 安装

推荐使用 Homebrew（macOS 最常用的包管理器）。如果你还没有安装 Homebrew，先运行：

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

然后安装 PostgreSQL 和 PostGIS：

```
brew install postgresql@16
brew install postgis
```

安装完成后，启动 PostgreSQL 服务：

```
brew services start postgresql@16
```

验证安装是否成功：

```
psql --version
#  ? ? ? ? ? ? ? psql (PostgreSQL) 16.x
```

## Ubuntu/Debian 安装

```
sudo apt update
sudo apt install -y postgresql-16 postgresql-16-postgis-3
sudo systemctl start postgresql
sudo systemctl enable postgresql    # [?] [?] [?] [?]
```

### 3. 第二步：安装 Neo4j

Neo4j 是项目的目标数据库（知识图谱存储在这里）。

## macOS 安装

```
brew install neo4j
```

启动 Neo4j:

```
brew services start neo4j
```

## Ubuntu 安装

```
# [?] [?] [?] Neo4j [?] [?] [?] [?]
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -
echo 'deb https://debian.neo4j.com stable latest' | sudo tee /etc/apt/sources.list.d/neo4j.list

sudo apt update
sudo apt install -y neo4j

sudo systemctl start neo4j
sudo systemctl enable neo4j
```

## 设置 Neo4j 初始密码

Neo4j 首次启动需要修改默认密码（默认账号/密码均为 `neo4j`）。

打开浏览器，访问：<http://localhost:7474>

你会看到 Neo4j 浏览器界面：

1. `sudo apt-get install neo4j`
2. `sudo apt-get install neo4j`
3. `sudo apt-get install neo4j`

重要：把你设置的 Neo4j 密码记下来，后面配置文件里要用到。

验证 Neo4j 运行正常：

```
# cypher-shell -u neo4j -p password "RETURN 1"
```

## 4. 第三步：安装 Python 环境

检查 Python 版本

```
python3 --version
```

如果版本低于 3.10，用 Homebrew 安装新版：

```
brew install python@3.12
```

为什么要用虚拟环境？

虚拟环境（venv）是 Python 项目的“隔离沙盒”，防止不同项目的依赖包互相冲突。强烈建议使用。

## 5. 第四步：获取项目代码

方式 A：使用 Git 克隆（推荐）

```
git clone https://github.com/gitzhen0/noah-postgres-to-neo4j.git
cd noah-postgres-to-neo4j
```

## 方式 B：下载压缩包

如果你没有安装 Git，可以在 GitHub 页面点击 Code → Download ZIP，解压后进入项目目录。

## 创建并激活虚拟环境

进入项目目录后，运行：

```
# 创建虚拟环境
python3 -m venv venv

# 激活虚拟环境
source venv/bin/activate      # macOS / Linux
# venv\Scripts\activate      # Windows

# 验证虚拟环境是否激活
# (venv) zhenyang@macbook noah-postgres-to-neo4j %
```

## 安装依赖包

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

这一步会下载约 50 个 Python 库，根据网速大约需要 2-5 分钟。出现 `Successfully installed ...` 就说明成功了。

## 6. 第五步：配置数据库连接

### 复制配置文件模板

```
cp config/config.example.yaml config/config.yaml
```

### 编辑配置文件

用任意文本编辑器打开 `config/config.yaml`（可以用 VS Code、TextEdit 等）：

```
# 在 VS Code 中：
code config/config.yaml
```

```
# 配置 PostgreSQL 数据库
nano config/config.yaml
```

需要修改以下几处（用你自己的密码替换）：

```
# ① PostgreSQL 配置
source_db:
  host: localhost          # 数据库主机
  port: 5432                # 数据库端口
  database: noah_housing    # 数据库名称
  user: postgres            # 数据库用户
  password: postgres        # 数据库密码

# ② Neo4j 配置
target_db:
  uri: bolt://localhost:7687 # Neo4j 地址
  user: neo4j                 # Neo4j 用户名
  password: neo4j             # Neo4j 密码
  database: neo4j             # Neo4j 数据库名
```

## 配置 Anthropic API Key（用于自然语言查询）

自然语言查询功能需要 Anthropic API Key。有两种配置方式：

### 方式 A：环境变量（推荐）

```
# 创建 .env 文件
cp .env.example .env

# 编辑 .env 文件
nano .env
```

在 `.env` 文件中填入：

```
ANTHROPIC_API_KEY=sk-ant-api03-xxxxxxxx...
```

### 方式 B：在 Web 界面输入

启动 Streamlit 后，在左侧边栏的输入框直接粘贴 API Key，无需修改配置文件。

注意：如果你没有 Anthropic API Key，可以跳过这一步。Cypher 编辑器（Explore 页面）不需要 API Key 就能使用，只有自然语言查询（Ask 页面）需要。

---

## 7. 第六步：导入 NOAH 数据库

这一步把 NOAH 住房数据导入到你本地的 PostgreSQL 中。

### 创建数据库

```
# 创建 postgres 数据库
createdb noah_housing

# 安装 PostGIS
psql -d noah_housing -c "CREATE EXTENSION IF NOT EXISTS postgis;"
psql -d noah_housing -c "CREATE EXTENSION IF NOT EXISTS postgis_topology;"
```

验证 PostGIS 安装：

```
psql -d noah_housing -c "SELECT PostGIS_Version();"
# 3.4 USE_GEOS=1 USE_PROJ=1 ...
```

### 加载数据

项目的 `data/` 目录下包含已准备好的 SQL 数据文件：

```
# 进入 data 目录
ls data/

# 加载数据
psql -d noah_housing -f data/schema/01_create_tables.sql
psql -d noah_housing -f data/schema/02_create_indexes.sql
psql -d noah_housing -f data/samples/housing_projects.sql
psql -d noah_housing -f data/samples/zip_shapes.sql
psql -d noah_housing -f data/samples/affordability_data.sql
psql -d noah_housing -f data/samples/rent_burden.sql
```

### 验证数据导入

```
psql -d noah_housing -c "SELECT COUNT(*) FROM housing_projects;"
# 8604

psql -d noah_housing -c "SELECT COUNT(*) FROM zip_shapes;"
# 177
```

---

## 8. 第七步：运行数据迁移

这是核心步骤：把 PostgreSQL 里的数据迁移到 Neo4j 知识图谱。

确认虚拟环境已激活

```
# 创建虚拟环境 (venv)
# 激活虚拟环境
source venv/bin/activate
```

先检查连接状态

```
python main.py status
```

预期输出：

```
✅ PostgreSQL: Connected (noah_housing, 8604 housing projects)
✅ Neo4j: Connected (neo4j, 0 nodes)
```

如果看到 ❌ 错误，请参考第 11 节「常见问题」。

分析 Schema（可选，用于了解数据结构）

```
python main.py analyze
```

这会输出 PostgreSQL 数据库的表结构分析报告，帮助你了解数据是如何组织的。

运行迁移

```
python main.py migrate
```

迁移过程会显示进度条，大约需要 2 – 5 分钟：

```
Stage 1: Creating constraints and indexes... ✅
Stage 2: Migrating HousingProject nodes (8604)... [██████████] 100%
Stage 3: Migrating ZipCode nodes (177)... [██████████] 100%
Stage 4: Migrating AffordabilityAnalysis nodes (177)... [██████████] 100%
Stage 5: Migrating RentBurden nodes (2225)... [██████████] 100%
```

```
Stage 6: Creating LOCATED_IN_ZIP relationships... [██████████] 100%
Stage 7: Creating NEIGHBORS relationships (spatial)... [██████████] 100%
...
Migration complete! 11183 nodes, 16900+ relationships created.
```

## 验证迁移结果

```
python main.py audit
```

应该看到所有检查项通过：

- ✅ Node count parity: HousingProject 8604/8604
- ✅ Node count parity: ZipCode 177/177
- ✅ Relationship integrity: LOCATED\_IN\_ZIP 8604/8604
- ✅ Property coverage: 97.3% (target: 95%)
- ✅ Spatial relationships: NEIGHBORS 512 pairs

## 9. 第八步：启动 Web 界面

启动 Streamlit 应用

```
streamlit run app/Home.py --server.port 8505
```

终端会显示：

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8505>


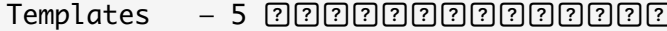

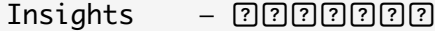
Network URL: <http://192.168.x.x:8505>

用浏览器打开 <http://localhost:8505>。

### 界面概览

你会看到 5 个页面（左侧边栏）：

```
🏠 Home      - [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]
? Ask       - [?] [?] [?]/[?] [?] [?] [?] [?] [?] AI [?] [?] [?] [?]
🔍 Explore   - Cypher [?] [?] [?] [?] [?] [?] [?] [?] [?] [?]
```

 Templates - 5   
 Insights - 

## 使用 Ask 页面（自然语言查询）

1. 在左侧边栏输入 Anthropic API Key（`sk-ant-` 开头）
2. 点击 Ask 页面
3. 在文本框输入问题，例如：
4. "哪些 ZIP 码的租金负担率超过 40%？"
5. "布鲁克林有多少保障性住房项目？"
6. "找到 ZIP 码 10001 相邻区域的所有住房项目"
7. 点击 Search 按钮


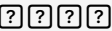
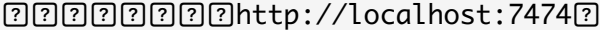
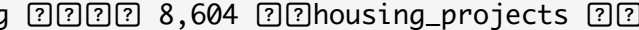
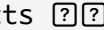

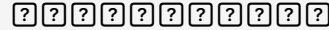
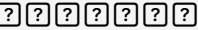
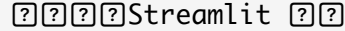

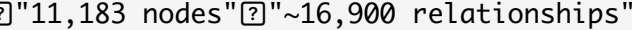

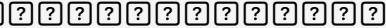
## 使用 Templates 页面（无需任何代码）

1. 点击 Templates 页面
2. 选择一个模板，例如「按行政区查看租金负担」
3. 用下拉菜单选择参数（行政区、阈值等）
4. 点击 Run ▶

---

## 10. 快速验证清单

完成所有步骤后，用这个清单确认一切正常：

```
□ PostgreSQL psql --version   
□ Neo4j http://localhost:7474/  
□ noah_housing  8,604 housing_projects   
□ python main.py status   
□ python main.py audit   
□ http://localhost:8505 Streamlit   
□ Home "11,183 nodes"~16,900 relationships"  
□ Templates 
```

## 11. 常见问题与解决方案

问题 1: `psql: command not found`

原因: PostgreSQL 未加入系统 PATH。

解决方案 (macOS) :

```
echo 'export PATH="/opt/homebrew/opt/postgresql@16/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

---

问题 2: `Cannot connect to Neo4j / ServiceUnavailable`

排查步骤:

```
# 检查 Neo4j 是否安装
brew services list | grep neo4j      # macOS
sudo systemctl status neo4j         # Linux

# 启动 Neo4j
brew services start neo4j            # macOS
sudo systemctl start neo4j          # Linux

# 检查端口是否监听
lsof -i :7687
```

常见原因: Neo4j 启动需要 15 – 30 秒, 如果刚启动就立刻连接可能失败, 等一下再试。

---

问题 3: `password authentication failed for user "postgres"`

解决方案:

```
# 使用 psql 连接数据库
sudo -u postgres psql

# 修改 postgres 用户密码
ALTER USER postgres PASSWORD 'postgres';
\q
```

然后更新 `config/config.yaml` 中的密码。

---

问题 4: `PostGIS extension not found`

解决方案 (macOS) :

```
brew install postgis
psql -d noah_housing -c "CREATE EXTENSION IF NOT EXISTS postgis;"
```

---

问题 5: `pip install` 报错 / 某个包安装失败

常见解决方案:

```
# 更新pip
pip install --upgrade pip

# 安装psycopg2 和 PostgreSQL 驱动
pip install psycopg2-binary # 安装二进制驱动

# 安装neo4j 驱动
pip install neo4j==5.14.0
```

---

问题 6: `Port 8505 is already in use`

Streamlit 已经在运行了。

```
# 杀死占用 8505 端口的进程
lsof -ti :8505 | xargs kill

# 重新运行
streamlit run app/Home.py --server.port 8506
```

---

问题 7: Ask 页面显示 `API key not valid`

- 确认 API Key 以 `sk-ant-` 开头

- 确认 Anthropic 账户有可用额度
- API Key 只存在浏览器会话中，刷新页面后需要重新输入

---

问题 8：迁移完成后 Neo4j 节点数为 0

```
# 检查 Neo4j 状态
python main.py status

# 检查 Neo4j 数据库
# 查询 http://localhost:7474, 返回
# MATCH (n) RETURN count(n)
```

如果为 0，可以重新运行迁移（迁移使用 MERGE，重复运行是安全的）：

```
python main.py migrate
```

---

## 12. 附录：Docker 一键部署（可选）

如果你有 Docker，可以用一条命令启动所有服务，跳过手动安装 PostgreSQL 和 Neo4j：

安装 Docker

从 <https://www.docker.com/products/docker-desktop> 下载安装 Docker Desktop。

启动所有服务

```
# 启动所有服务
docker compose up -d
```

等待约 60 秒（Docker 需要拉取镜像），然后访问：

- Streamlit UI: <http://localhost:8505>
- Neo4j 浏览器: <http://localhost:7474>

## 查看日志

```
docker compose logs -f app      # Streamlit [?]
docker compose logs -f neo4j    # Neo4j [?]
```

## 停止服务

```
docker compose down
```

注意： Docker 方式会自动创建空的 Neo4j 数据库。你仍然需要手动导入 PostgreSQL 数据并运行迁移（ `python main.py migrate` ），或者修改 `docker-compose.yml` 挂载已有的 Neo4j 数据目录。

## 小结

完成上述步骤后，你的本地环境应该是这样的：

```
PostgreSQL (localhost:5432)
├─ noah_housing [?]
│   ├── housing_projects 8,604 [?]
│   ├── zip_shapes       177 [?]
│   ├── noah_affordability_analysis 177 [?]
│   └─ rent_burden       2,225 [?]
└─

Neo4j (localhost:7687)
├─ neo4j [?]
│   ├── HousingProject [?] 8,604 [?]
│   ├── ZipCode [?]      177 [?]
│   ├── AffordabilityAnalysis [?] 177 [?]
│   ├── RentBurden [?]   2,225 [?]
│   └─ [?]              ~16,900 [?]
└─

Streamlit UI (localhost:8505)
├─ 5 [?] [?] [?] [?] [?] [?]
└─
```

如有其他问题，请查阅 `docs/guides/USER_GUIDE.md` 或在 GitHub 提交 Issue。