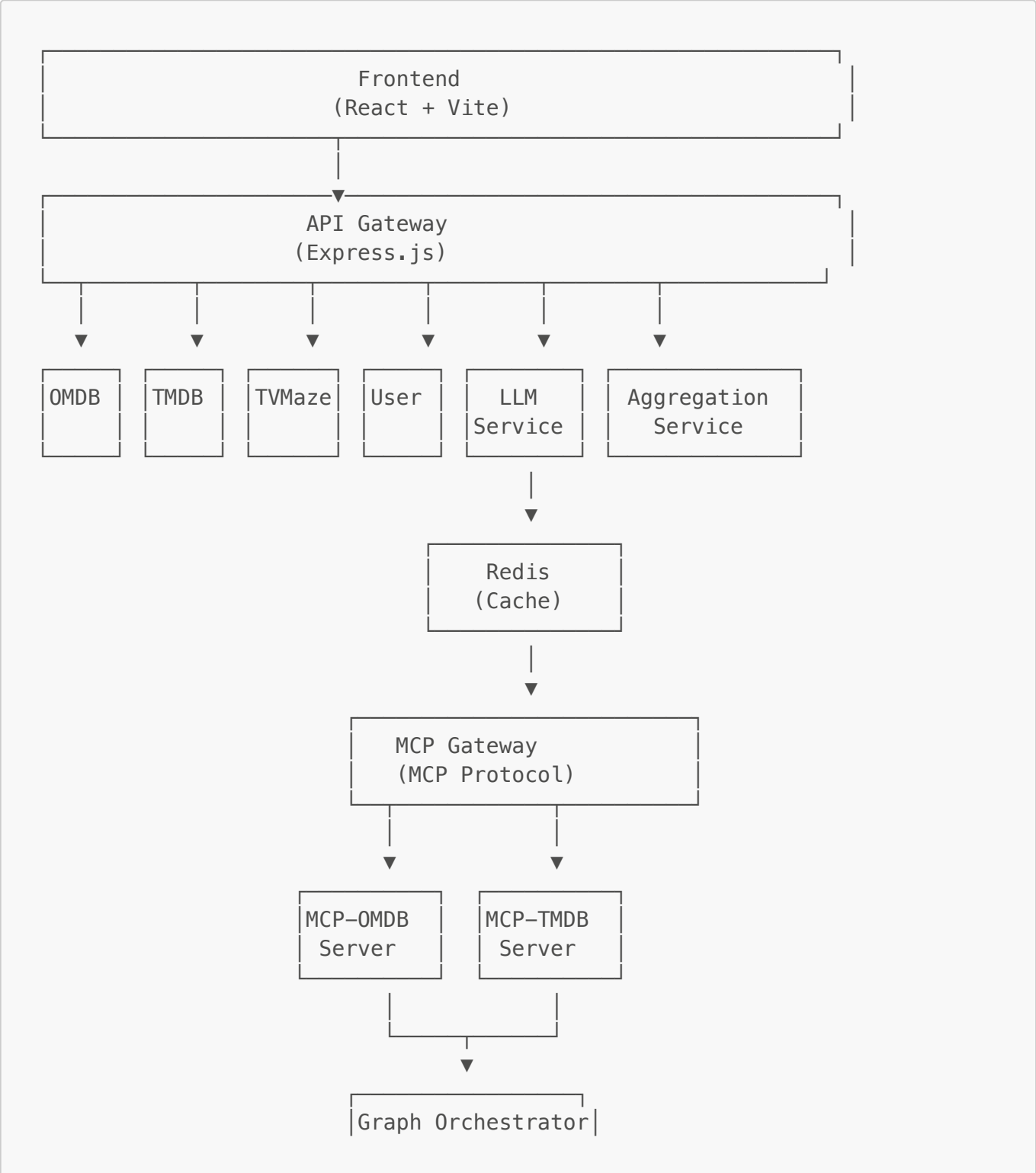
MovieHub - Configuration and Deployment Guide

项目概述

MovieHub 是一个基于微服务架构的电影与剧集聚合平台，整合了多个外部数据源（OMDB、TMDB、TVMaze）和 AI 服务（通义千问），提供电影搜索、评分聚合、AI 智能推荐等功能。

系统架构

微服务组件



```
                    │  (LangGraph)      │
                    └──────────────────┘
```

服务说明

服务名称	端口	功能描述
web-client	80	React 前端应用
api-gateway	3000	API 网关，路由请求到各个微服务
omdb	3003	OMDB API 服务封装
tmdb	3002	TMDB API 服务封装
tvmaze	3006	TVMaze API 服务封装
aggregation	3004	数据聚合服务，整合多源数据
llm	3001	LLM 服务，提供 AI 总结功能
user	3005	用户服务，管理观影清单
mcp-gateway	3007	MCP 协议网关
mcp-omdb	3009	MCP OMDB 服务器
mcp-tmdb	3008	MCP TMDB 服务器
graph-orchestrator	3010	LangGraph 编排服务，AI 智能搜索
redis	6379	缓存服务

环境要求

必需软件

- **Docker**: >= 20.10
- **Docker Compose**: >= 2.0
- **Node.js**: >= 18 (仅本地开发需要)
- **pnpm**: >= 8 (仅本地开发需要)

API 密钥

需要申请以下 API 密钥：

1. **OMDB API Key**
    - 网址: http://www.omdbapi.com/apikey.aspx
    - 免费版限制: 1000 次/天
2. **TMDB API Key**
    - 网址: https://www.themoviedb.org/settings/api
    - 免费版无限制

3. **通义千问 API Key**
    - 网址: https://dashscope.aliyun.com/
    - 需要阿里云账号

部署方式

方式一：Docker 部署（推荐）

1. 克隆项目

```
# 克隆项目仓库
git clone https://github.com/WilliamJiang1014/MicroServices-MovieHub.git

# 进入项目目录
cd MicroServices-MovieHub
```

2. 配置环境变量

在项目根目录下创建 `.env` 文件：

```
# 创建 .env 文件
cat > .env << 'EOF'
# OMDB API
OMDB_API_KEY=your_omdb_api_key_here

# TMDB API
TMDB_API_KEY=your_tmdb_api_key_here

# 通义千问 API
QWEN_API_KEY=your_qwen_api_key_here
QWEN_API_URL=https://dashscope.aliyuncs.com/compatible-mode/v1

# Redis
REDIS_URL=redis://redis:6379

# Node Environment
NODE_ENV=production
EOF
```

注意: 将 `your_xxx_api_key_here` 替换为实际的 API 密钥。

3. 构建并启动服务

```
# 构建所有服务
docker-compose build
```

```
# 启动所有服务
docker-compose up -d

# 查看服务状态
docker-compose ps

# 查看日志
docker-compose logs -f
```

4. 验证部署

```
# 检查服务健康状态
docker-compose ps

# 所有服务应显示 "healthy" 状态
# 访问应用
open http://localhost
```

5. 停止服务

```
# 停止所有服务
docker-compose down

# 停止并删除数据卷
docker-compose down -v
```

方式二：本地开发部署

1. 克隆项目并安装依赖

```
# 克隆项目仓库
git clone https://github.com/WilliamJiang1014/MicroServices-MovieHub.git

# 进入项目目录
cd MicroServices-MovieHub

# 安装 pnpm（如果未安装）
npm install -g pnpm

# 安装项目依赖
pnpm install
```

2. 配置环境变量

为每个服务创建 `.env` 文件：

```
# 复制环境变量模板
cp env.example .env

# 编辑 .env 文件，填入您的 API 密钥
nano .env
```

在 `.env` 文件中填入：

```
TMDB_API_KEY=your_tmdb_api_key_here
OMDB_API_KEY=your_omdb_api_key_here
QWEN_API_KEY=your_qwen_api_key_here
QWEN_API_URL=https://dashscope.aliyuncs.com/compatible-mode/v1
```

3. 启动 Redis

```
# 使用 Docker 启动 Redis
docker run -d --name moviehub-redis -p 6379:6379 redis:7-alpine
```

4. 构建必需的包

⚠ **重要**: 本地开发前必须先构建以下包：

```
pnpm --filter @moviehub/shared build
pnpm --filter @moviehub/mcp-gateway build
pnpm --filter @moviehub/mcp-provider-tmdb build
pnpm --filter @moviehub/mcp-provider-omdb build
pnpm --filter @moviehub/graph-orchestrator build
```

5. 启动所有服务

推荐方式（使用启动脚本，避免 macOS 文件描述符限制）：

```
./scripts/local-dev.sh
```

或手动启动：

```
pnpm dev
```

注意: macOS 用户如果遇到 `ENFILE: file table overflow` 错误，请使用启动脚本。

6. 访问应用

```
# 前端开发服务器
open http://localhost:5173

# API 网关
curl http://localhost:3000/health
```

7. 停止服务

```
# 停止所有开发进程
pkill -f 'pnpm.*dev' && pkill -f 'tsx watch' && pkill -f 'vite'

# 停止 Redis
docker stop moviehub-redis-local && docker rm moviehub-redis-local
```

配置说明

Docker Compose 配置

主要配置文件：`docker-compose.yml`

关键配置项：

```
services:
  # 前端服务
  web-client:
    build: ./apps/web-client
    ports:
      - "80:80"
    depends_on:
      - api-gateway
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
"http://localhost/health"]
      interval: 30s
      timeout: 10s
      retries: 5
      start_period: 40s

  # API 网关
  api-gateway:
    build: ./services/api-gateway
    ports:
      - "3000:3000"
```

```
    environment:
      - NODE_ENV=production
    depends_on:
      - omdb
      - tmdb
      - tvmaze
      - aggregation
      - llm
      - user
```

Nginx 配置

前端 Nginx 配置：`apps/web-client/nginx.conf`

关键配置：

```
# API 代理
location /api/ {
    proxy_pass http://api-gateway:3000/api/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

# MCP 服务代理
location /api/mcp/ {
    proxy_pass http://graph-orchestrator:3010/;
}

location /api/mcp-gateway/ {
    proxy_pass http://mcp-gateway:3007/;
}
```

Vite 配置

本地开发代理：`apps/web-client/vite.config.ts`

```
export default defineConfig({
  plugins: [react()],
  server: {
    port: 5173,
    proxy: {
      // 注意：更具体的路径要放在前面，否则会被 /api 拦截
      '/api/mcp-gateway': {
        target: 'http://localhost:3007',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api\/mcp-gateway/, ''),
      },
      '/api/mcp': {
        target: 'http://localhost:3010',
```

```
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api\/mcp/, ''),
      },
      '/api': {
        target: 'http://localhost:3000',
        changeOrigin: true,
      },
    },
  },
});
```

功能验证

1. 基础功能测试

```
# 健康检查
curl http://localhost/health

# 搜索电影（关键词搜索）
curl "http://localhost/api/search?query=Inception"

# 获取电影详情
curl "http://localhost/api/movie/tmdb-27205"

# AI 智能搜索
curl -X POST http://localhost/api/mcp/execute \
  -H "Content-Type: application/json" \
  -d '{
    "workflow": "movie_search",
    "input": {
      "query": "科幻电影推荐"
    }
  }'
```

2. 前端功能测试

访问 http://localhost 并测试：

1. **关键词搜索**
   - 输入电影名称
   - 查看多源数据聚合结果
   - 点击电影卡片查看详情
2. **AI 智能搜索**
   - 切换到"AI 智能搜索"模式
   - 输入自然语言查询
   - 查看 AI 推荐结果

3. **观影清单**
    - 添加电影到清单
    - 设置观看状态（想看/在看/看过）
    - 调整观看进度
    - 添加个人评分
4. **电影详情**
    - 查看多源评分对比
    - 查看评分雷达图
    - 查看相似作品网络图
    - 查看 AI 生成的总结

故障排查

常见问题

1. 服务启动失败

```
# 查看服务日志
docker-compose logs <service-name>

# 例如：查看 API 网关日志
docker-compose logs api-gateway

# 查看所有服务日志
docker-compose logs -f
```

2. API 密钥错误

检查 `.env` 文件中的 API 密钥是否正确配置。

```
# 查看环境变量
docker-compose config
```

3. 端口冲突

如果端口被占用，修改 `docker-compose.yml` 中的端口映射：

```
ports:
  - "8080:80"  # 将 80 改为 8080
```

4. 服务健康检查失败

```
# 检查服务状态
docker-compose ps

# 重启特定服务
docker-compose restart <service-name>

# 重新构建并启动
docker-compose up -d --build <service-name>
```

5. Redis 连接失败

```
# 检查 Redis 服务
docker-compose logs redis

# 重启 Redis
docker-compose restart redis
```

日志查看

```
# 查看所有服务日志
docker-compose logs -f

# 查看特定服务日志
docker-compose logs -f web-client

# 查看最近 50 行日志
docker-compose logs --tail=50 api-gateway
```

技术栈总结

- **前端**: React 18, TypeScript, Vite
- **后端**: Node.js 18, Express.js, TypeScript
- **缓存**: Redis 7
- **容器化**: Docker, Docker Compose
- **AI 框架**: LangGraph
- **协议**: MCP (Model Context Protocol)
- **LLM**: 通义千问 (Qwen)
- **数据源**: OMDB, TMDB, TVMaze

项目仓库

GitHub: https://github.com/WilliamJiang1014/MicroServices-MovieHub

---

部署完成！ 访问 http://localhost 开始使用 MovieHub。