

# OpenAct 脚本使用指南

---

本目录包含了完整的GitHub OAuth认证到Action调用的自动化脚本。

## 快速开始

### 1. 完整流程脚本 (推荐)

执行从OAuth认证到Action调用的完整流程：

```
# 设置GitHub应用凭据
export GITHUB_CLIENT_ID="0v23lihVkExosE0hR0Bh"
export GITHUB_CLIENT_SECRET="9c704ca863eb45c8175d5d6bd9f367b1d17d8afc"

# 运行完整流程
./scripts/complete_github_flow.sh
```

这个脚本会自动完成：

-  生成加密主密钥
-  启动AuthFlow服务器
-  创建GitHub OAuth2工作流
-  打开浏览器进行授权
-  等待授权完成
-  存储加密的访问令牌
-  执行真实的GitHub API调用
-  验证端到端集成
-  清理资源

### 2. 快速认证脚本

只进行OAuth认证，不执行Action：

```
./scripts/quick_github_auth.sh <client_id> <client_secret>
```

示例：

```
./scripts/quick_github_auth.sh 0v23lihVkExosE0hR0Bh
9c704ca863eb45c8175d5d6bd9f367b1d17d8afc
```

## 前置要求

## 1. GitHub OAuth应用设置

在GitHub中创建OAuth应用：

1. 访问 <https://github.com/settings/developers>
2. 点击 "New OAuth App"
3. 设置回调URL: <http://localhost:8080/oauth/callback>
4. 获取 Client ID 和 Client Secret

## 2. 系统依赖

确保安装了以下工具：

- **curl** - HTTP请求
- **jq** - JSON处理
- **python3** - 生成加密密钥
- **cargo** - Rust编译器
- **open** (macOS) 或浏览器 - 打开授权URL

## 3. 项目编译

确保项目可以正常编译：

```
cargo build --workspace --features server,sqlite,encryption
```

### 手动步骤

如果需要手动执行，可以按以下步骤：

#### 步骤1: 设置环境变量

```
export GITHUB_CLIENT_ID="your_client_id"
export GITHUB_CLIENT_SECRET="your_client_secret"
export AUTHFLOW_MASTER_KEY=$(python3 -c "import os,binascii;print(binascii.hexlify(os.urandom(32)).decode())")
export AUTHFLOW_STORE=sqlite
export AUTHFLOW_SQLITE_URL=sqlite:$(pwd)/authflow/data/authflow.db
```

#### 步骤2: 启动AuthFlow服务器

```
cd authflow
RUST_LOG=info cargo run --features server,sqlite,encryption &
```

#### 步骤3: 执行OAuth认证

使用 `authflow/scripts/` 目录下的现有脚本，或通过API手动创建工作流。

#### 步骤4: 执行Action

```
cd manifest
export CONNECTION_TRN="trn:authflow:demo-tenant:connection/github-username"
export GITHUB_BASE_URL="https://api.github.com"
cargo test e2e_github_get_user --test e2e_github -- --ignored --nocapture
```

### 故障排除

#### 常见问题

##### 1. 端口8080被占用

```
pkill -f "authflow.*server"
```

##### 2. 权限错误

```
chmod +x scripts/*.sh
```

##### 3. GitHub API 403错误

- 这是正常的，因为测试请求缺少User-Agent头
- 重要的是认证信息正确注入

##### 4. 数据库权限问题

```
mkdir -p authflow/data
chmod 755 authflow/data
```

#### 调试模式

启用详细日志：


```
export RUST_LOG=debug
```

查看数据库内容：

```
sqlite3 authflow/data/authflow.db "SELECT trn, provider, user_id,
created_at FROM connections;"
```

## 脚本输出说明

### 成功输出示例


 完整流程执行成功！

=====

 流程总结：

1.  生成并设置加密主密钥
2.  启动AuthFlow服务器
3.  创建GitHub OAuth2工作流
4.  执行OAuth认证流程
5.  用户浏览器授权完成
6.  访问令牌加密存储到SQLite
7.  Manifest读取并解密认证信息
8.  执行真实的GitHub API调用
9.  验证端到端集成成功

 认证信息已安全存储在：sqlite:./authflow/data/authflow.db

 连接TRN：trn:authflow:demo-tenant:connection/github-username

### 测试结果说明

- **Status: Success** - Action执行成功
- **final\_status: 200** - HTTP请求成功
- **ok: true** - 整体流程成功
- **http.status: 403** - GitHub API返回403（正常，因为缺少User-Agent）

## 安全注意事项

### 1. 不要提交敏感信息到Git

- Client Secret应该通过环境变量传递
- 主密钥会自动生成，不要硬编码

### 2. 生产环境使用

- 使用更安全的数据库配置
- 启用HTTPS
- 配置适当的CORS策略

### 3. 密钥管理

- 定期轮换GitHub应用密钥

- 使用密钥管理服务存储敏感信息



## 相关文档

- [AuthFlow文档](#)
- [Manifest文档](#)
- [OpenAct设计规范](#)
- [实现计划](#)