

🤖 XXXBot 机器人项目 🤖

📄 项目概述

XXXBot 是一个基于微信的智能机器人系统，通过整合多种 API 和功能，提供了丰富的交互体验。本系统包含管理后台界面，

🔄 双协议支持

本系统现已支持两种微信协议版本，可以根据需要在配置文件中选择使用：

- ****849 协议****: 适用于 iPad 版本
- ****855 协议****: 适用于安卓 PAD 版本

通过在 `main_config.toml` 文件中设置 `Protocol.version` 参数，系统会自动选择相应的服务和 API 路径。详细配置

🚀 快速开始

✨ 主要特性

1. 🖥️ 管理后台

- 🏠 ****控制面板****: 系统概览、机器人状态监控
- 🛠️ ****插件管理****: 安装、配置、启用/禁用各类功能插件
- 📁 ****文件管理****: 上传、查看和管理机器人使用的文件
- 👤 ****联系人管理****: 微信好友和群组联系人管理
- 📊 ****系统状态****: 查看系统资源占用和运行状态

2. 💬 聊天功能

- 🗣️ ****私聊互动****: 与单个用户的一对一对话
- 👥 ****群聊响应****: 在群组中通过@或特定命令触发
- 📞 ****聊天室模式****: 支持多人持续对话，带有用户状态管理
- 💰 ****积分系统****: 对话消耗积分，支持不同模型不同积分定价
- 📸 ****朋友圈功能****: 支持查看、点赞和评论朋友圈

3. 🤖 智能对话

- 🔍 ****多模型支持****: 可配置多种 AI 模型，支持通过关键词切换
- 🖼️ ****图文结合****: 支持图片理解和多媒体输出
- 🎤 ****语音交互****: 支持语音输入识别和语音回复
- 😊 ****语音撒娇****: 支持甜美语音撒娇功能

4. 🔗 插件系统

- 🛠️ ****插件管理****: 支持加载、卸载和重载插件
- 🧰 ****自定义插件****: 可开发和加载自定义功能插件
- 🤖 ****Dify 插件****: 集成 Dify API，提供高级 AI 对话能力
- ⌚ ****定时提醒****: 支持设置定时提醒和日程管理
- 🏠 ****群欢迎****: 自动欢迎新成员加入群聊

2. 安装依赖

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

3. 安装 Redis

- Windows: 下载 Redis for Windows
- Linux: `sudo apt-get install redis-server`
- macOS: `brew install redis`

4. 安装 FFmpeg

- Windows: 下载安装包并添加到系统 PATH
- Linux: `sudo apt-get install ffmpeg`
- macOS: `brew install ffmpeg`

5. 配置

- 复制 `main_config.toml.example` 为 `main_config.toml` 并填写配置
- 设置管理员 ID 和其他基本参数

设置管理员：

在 `main_config.toml` 文件中的 `[XYBot]` 部分设置管理员：

```
[XYBot]
# 管理员微信ID，可以设置多个，用英文逗号分隔
admins = ["wxid_l2221111", "wxid_l111111"] # 管理员的wxid列表，可从消息日志中获取
```

设置 GitHub 加速代理：

在 `main_config.toml` 文件中的 `[XYBot]` 部分设置 GitHub 加速代理：

```
[XYBot]
# GitHub加速服务设置
# 可选值： "", "https://ghfast.top/", "https://gh-proxy.com/", "https://mirror.ghproxy.com/"
# 空字符串表示直连不使用加速
# 注意：如果使用加速服务，请确保以"/"结尾
github-proxy = "https://ghfast.top/"
```

设置系统通知功能：

在 `main_config.toml` 文件中配置系统通知功能（微信离线、重连、重启等通知）：

```
# 系统通知设置
[Notification]
enabled = true # 是否启用通知功能
token = "your_pushplus_token" # PushPlus Token，必须在这里设置！
channel = "wechat" # 通知渠道：wechat(微信公众号)、sms(短信)、mail(邮件)、webhook、cp(
template = "html" # 通知模板
topic = "" # 群组编码，不填仅发送给自己

# 通知触发条件
[Notification.triggers]
offline = true # 微信离线时通知
reconnect = true # 微信重新连接时通知
restart = true # 系统重启时通知
```

```
error = true                                # 系统错误时通知

# 通知模板设置
[Notification.templates]
offlineTitle = "警告: 微信离线通知 - {time}" # 离线通知标题
offlineContent = "您的微信账号 <b>{wxid}</b> 已于 <span style=\"color:#ff4757;font-weight:bold;\">{time}</span> 离线。"
reconnectTitle = "微信重新连接通知 - {time}" # 重连通知标题
reconnectContent = "您的微信账号 <b>{wxid}</b> 已于 <span style=\"color:#2ed573;font-weight:bold;\">{time}</span> 重新连接。"
restartTitle = "系统重启通知 - {time}" # 系统重启通知标题
restartContent = "系统已于 <span style=\"color:#1e90ff;font-weight:bold;\">{time}</span> 重新启动。"
```

! 重要提示:

- PushPlus Token 必须在 main_config.toml 文件中直接设置，而不是通过网页界面设置
- 如果通过网页界面设置，可能会导致容器无法正常启动
- 请先在 [PushPlus 官网](#) 注册并获取 Token

协议配置

在 main_config.toml 文件中添加以下配置来选择微信协议版本:

```
[Protocol]
version = "849" # 可选值: "849", "855" 或 "ipad"
```

- **849**: 适用于 iPad 版本, 使用 /VXAPI 路径前缀
- **855**: 适用于安卓 PAD 版本, 使用 /api 路径前缀
- **ipad**: 适用于新版 iPad 协议, 使用 /api 路径前缀

系统会根据配置的协议版本自动选择正确的服务路径和 API 路径前缀。如果遇到 API 请求失败的情况, 系统会自动尝试使用另一种协议路径, 确保功能正常工作。

6. 启动必要的服务

需要先启动 Redis 和 PAD 服务 (注意启动顺序!) :

Windows 用户

- **! 第一步: 启动 Redis 服务** 
 - 进入 849/redis 目录, 双击 redis-server.exe 文件
 - 等待窗口显示 Redis 启动成功
- **! 第二步: 启动 PAD 服务** 
 - 根据你的协议版本选择相应的服务:
 - **849 协议 (iPad)** : 进入 849/pad 目录, 双击 linuxService.exe 文件
 - **855 协议 (安卓 PAD)** : 进入 849/pad2 目录, 双击 linuxService.exe 文件
 - 等待窗口显示 PAD 服务启动成功
-  请确保这两个服务窗口始终保持打开状态, 不要关闭它们!

然后启动主服务：

```
python app.py
```

Linux 用户

- **！ 第一步：启动 Redis 服务** 🟢

```
# 进入Redis目录
```

```
cd 849/redis
```

```
# 使用Linux配置文件启动Redis
```

```
redis-server redis.conf
```

- 如果 Redis 未安装，需要先安装：

```
# Ubuntu/Debian
```

```
sudo apt-get update
```

```
sudo apt-get install redis-server
```

```
# CentOS/RHEL
```

```
sudo yum install redis
```

- **！ 第二步：启动 PAD 服务** 🟡

根据你的协议版本选择相应的服务：

849 协议 (iPad)：

```
# 进入PAD目录
```

```
cd 849/pad
```

```
# 给执行文件添加执行权限
```

```
chmod +x linuxService
```

```
# 运行服务
```

```
./linuxService
```

855 协议 (安卓 PAD)：

```
# 进入PAD2目录
```

```
cd 849/pad2
```

```
# 给执行文件添加执行权限
```

```
chmod +x linuxService
```

```
# 运行服务
```

```
./linuxService
```

- ⚠️ 请确保这两个服务进程保持运行状态，可以使用如下命令检查：

```
# 检查Redis服务
ps aux | grep redis

# 检查PAD服务
ps aux | grep linuxService
```

然后启动主服务：

```
python app.py
```

▲ 方法二：Docker 安装

💡 **注意：**Docker 环境会自动启动 Redis 和 PAD 服务，无需手动启动。这是通过 `entrypoint.sh` 脚本实现的。脚本会根据 `main_config.toml` 中的 `Protocol.version` 设置自动选择启动 849 或 855 协议的 PAD 服务。

1. 使用 Docker Compose 一键部署

```
# 克隆代码库
git clone https://github.com/NanSsy/xxxbot-pad.git
cd xxxbot-pad

# 启动服务
docker-compose up -d
```

这将自动拉取最新的镜像并启动服务，所有数据将保存在 Docker 卷中。

2. 更新到最新版本

```
# 拉取最新镜像
docker-compose pull

# 重启服务
docker-compose down
docker-compose up -d
```

我们已经更新了 `docker-compose.yml` 文件，添加了 `pull_policy: always` 设置，确保每次启动容器时都会检查并拉取最新的镜像。更多更新相关的详细信息，请查看 [UPDATE_GUIDE.md](#) 文件。

3. 自定义管理员账号密码（可选）

编辑 `docker-compose.yml` 文件，修改环境变量：

```
environment:
  - ADMIN_USERNAME=your_username # 修改为您想要的用户名
  - ADMIN_PASSWORD=your_password # 修改为您想要的密码
```

访问后台

- 🌐 打开浏览器访问 `http://localhost:9090` 进入管理界面

- 👤 默认用户名: admin
- 🔑 默认密码: admin1234

🤖 Dify 插件配置

```
[Dify]
enable = true
default-model = "model1"
command-tip = true
commands = ["ai", "机器人", "gpt"]
admin_ignore = true
whitelist_ignore = true
http-proxy = ""
voice_reply_all = false
robot-names = ["机器人", "小助手"]
remember_user_model = true
chatroom_enable = true

[Dify.models.model1]
api-key = "your_api_key"
base-url = "https://api.dify.ai/v1"
trigger-words = ["dify", "小d"]
price = 10
wakeup-words = ["你好小d", "嘿小d"]
```

📖 使用指南

👑 管理员命令

- 登录管理后台查看各项功能
- 通过微信直接向机器人发送命令管理

💬 用户交互

- 🗉 私聊模式: 直接向机器人发送消息
- 👥 群聊模式:
 - 🙋 @机器人 + 问题
 - 💬 使用特定命令如 ai 问题
 - 🔔 使用唤醒词如 你好小d 问题

📞 聊天室功能

- 🙋 加入聊天: @机器人或使用命令
- 查看状态: 发送"查看状态"
- 暂时离开: 发送"暂时离开"
- 回来: 发送"回来了"
- 退出聊天: 发送"退出聊天"
- 查看统计: 发送"我的统计"
- 聊天排行: 发送"聊天室排行"

图片和语音

- 发送图片和文字组合进行图像相关提问
- 发送语音自动识别并回复
- 语音回复可根据配置自动开启

插件开发

插件目录结构

```
plugins/
├── YourPlugin/
│   ├── __init__.py
│   ├── main.py
│   ├── config.toml
│   └── README.md
```

基本插件模板

```
from utils.plugin_base import PluginBase
from WechatAPI import WechatAPIClient
from utils.decorators import *

class YourPlugin(PluginBase):
    description = "插件描述"
    author = "作者名称"
    version = "1.0.0"

    def __init__(self):
        super().__init__()
        # 初始化代码

    @on_text_message(priority=10)
    async def handle_text(self, bot: WechatAPIClient, message: dict):
        # 处理文本消息
        pass
```

常见问题

1. 安装依赖失败 🖥️

- 尝试使用 `pip install --upgrade pip` 更新 pip
- 可能需要安装开发工具: `apt-get install python3-dev`

2. 语音识别失败 🎤

- 确认 FFmpeg 已正确安装并添加到 PATH
- 检查 SpeechRecognition 依赖是否正确安装

3. 无法连接微信 📱

- 确认微信客户端和接口版本是否匹配
- 检查网络连接和端口设置
- 如果使用 PAD 协议，确认 PAD 服务是否正常运行
- ⚠ Windows 用户请确认是否按正确顺序启动服务：先启动 Redis，再启动 PAD
- 检查 `main_config.toml` 中的协议版本设置是否正确（849 用于 iPad，855 用于安卓 PAD）

4. Redis 连接错误 🟡

- 确认 Redis 服务器是否正常运行
- 🔴 Windows 用户请确认是否已启动 `849/redis` 目录中的 `redis-server.exe`
- 检查 Redis 端口和访问权限设置
- 确认配置文件中的 Redis 端口是否为 6378
- 💡 提示：Redis 窗口应显示“已就绪接受指令”或类似信息

5. Dify API 错误 🤖

- 验证 API 密钥是否正确
- 确认 API URL 格式和访问权限

6. Docker 部署问题 🐳

- 确认 Docker 容器是否正常运行：`docker ps`
- 查看容器日志：`docker logs xxxbot-pad`
- 重启容器：`docker-compose restart`
- 查看卷数据：`docker volume ls`
- 💡 注意：Docker 容器内会自动启动 PAD 和 Redis 服务，无需手动启动
- 如果需要切换协议版本，只需修改 `main_config.toml` 中的 `Protocol.version` 设置并重启容器
- ⚠ Windows 用户注意：Docker 容器使用的是 Linux 环境，不能直接使用 Windows 版的可执行文件

7. 无法访问管理后台 🔴

- 确认服务器正常运行在 9090 端口
- 尝试使用默认账号密码: `admin/admin123`
- 检查防火墙设置是否阻止了端口访问

技术架构

- 后端：Python FastAPI
- 前端：Bootstrap, Chart.js, AOS
- 数据库：SQLite (aiosqlite)
- 缓存：Redis
- 微信接口：PAD 协议或 WeChatAPI
- 外部服务：Dify API, Google Speech-to-Text
- 容器化：Docker
- Web 服务：默认端口 9090，默认账号 `admin/admin123`

项目结构

```
XXXBot/
├── admin/                # 管理后台
│   ├── static/          # 静态资源
│   ├── templates/       # HTML模板
│   └── friend_circle_api.py # 朋友圈API
├── plugins/             # 插件目录
│   ├── Dify/            # Dify插件
│   ├── Menu/            # 菜单插件
│   ├── SignIn/          # 签到插件
│   └── YujieSajiao/      # 语音撒娇插件
├── database/            # 数据库相关
├── utils/               # 工具函数
├── WechatAPI/           # 微信API接口
├── 849/                 # PAD协议相关
│   ├── pad/             # 849协议客户端（适用于 iPad）
│   ├── pad2/            # 855协议客户端（适用于安卓 PAD）
│   └── redis/           # Redis服务
├── app.py               # 主应用入口
├── main.py              # 机器人主程序
├── entrypoint.sh        # Docker入口脚本
├── Dockerfile           # Docker构建文件
├── requirements.txt      # 依赖列表
└── main_config.toml     # 主配置文件
```

协议和许可

本项目基于 [MIT 许可证](#) 开源，您可以自由使用、修改和分发本项目的代码，但需保留原始版权声明。

⚠ 注意：本项目仅供学习和研究使用，使用前请确保符合微信和相关服务的使用条款。使用本项目所产生的任何法律责任由使用者自行承担。

鸣谢

本项目的开发离不开以下作者和项目的支持与贡献：



HenryXiaoYang
[个人主页](#)

项目：XYBotV2 - 本项目的重要参考源
提供了微信机器人的基础架构和核心功能，为本项目的开发提供了宝贵的参考。



与本项目作者共同完成的开发工作
在功能扩展、界面设计和系统优化方面做出了重要贡献。

heaven2028
个人主页

同时感谢所有其他贡献者和使用的开源项目。

联系方式

- **GitHub:** <https://github.com/NanSsye>
- **官方交流群:** 请查看上方[快速开始](#)部分的二维码



管理后台界面展示

