树莓派学习笔记

一、树莓派简介

树莓派（Raspberry Pi）是一款基于ARM架构的单板计算机，由英国树莓派基金会开发，旨在促进计算机科学教育。虽然体积小巧（仅信用卡大小），但具备完整计算机功能，可运行Linux等操作系统。

主要特点：

- •低功耗、低成本
- •GPIO接口支持电子项目开发
- •丰富的社区支持和生态系统
- •适合学习编程、物联网项目、家庭服务器等场景

使用的是**树莓派4B**型号，配备4GB内存，搭载64位四核Cortex-A72处理器，运行Raspberry Pi OS（基于Debian）。

二、树莓派系统烧录

1. 下载系统镜像

- •官方下载地址：https://www.raspberrypi.com/software/
- •推荐使用**Raspberry Pi Imager**工具进行烧录
- •选择带桌面的64位系统镜像
- 手动下载镜像地址
- https://www.raspberrypi.com/software/operating-systems/

2. 烧录步骤（SSD启动）

1. 将MicroSD卡插入读卡器并连接电脑
2. 打开Raspberry Pi Imager工具
3. 选择操作系统 → misc utility images---bootloader----usb Boot
4. 选择存储卡目标,点击烧录即可
5. 存储卡插入树莓派启动，自动设置为USB-Boot启动
6. SSD硬盘插入电脑，按以上步骤烧录Raspberry Pi OS (64-bit)系统，编辑配置设置wifi，ssh，WiFi地区选择CN
7. ssd插入树莓派即可启动，SSH按ip连接

三、一键获取系统硬件信息脚本

创建一个脚本文件：

```
nano system_report.sh
```

将以下内容复制粘贴到文件中：

```
#!/bin/bash
```

```
REPORT_FILE="/home/pi/pi_system_report_$(date +%Y%m%d_%H%M%S).txt"

{
    echo "====================== 树莓派系统全面配置报告 ======================"
    echo "生成时间: $(date)"
    echo "================================================================================"
    echo ""
    echo "---------------------- 1. 核心系统与硬件信息 ----------------------"
    echo ">> OS 版本:"
    cat /etc/os-release
    echo ""
    echo ">> 内核版本:"
    uname -a
    echo ""
    echo ">> 主机名:"
    hostname
    echo ""
    echo ">> 运行时间:"
    uptime
    echo ""
    echo ">> CPU 信息:"
    lscpu
    echo ""
    echo ">> ARM 频率和温度:"
    vcgencmd measure_clock arm
    vcgencmd measure_temp
    echo ""
    echo ">> 内存信息:"
    free -h
    echo ""
    echo ">> 磁盘使用:"
    df -h
    echo ""
    echo ">> GPU 内存:"
    vcgencmd get_mem gpu
    echo ""

    echo "---------------------- 2. 网络配置 ----------------------"
    echo ">> IP 地址:"
    hostname -I
    echo ""
    echo ">> 网络接口:"
    ip addr show
    echo ""
    echo ">> DNS 配置:"
    cat /etc/resolv.conf
    echo ""
    echo ">> 路由表:"
    ip route
    echo ""

    echo "---------------------- 3. 系统设置 ----------------------"
    echo ">> 环境变量:"
    printenv
    echo ""
```

```
    echo ">> 时区:"
    timedatectl status
    echo ""
    echo ">> 区域设置:"
    locale
    echo ""

    echo "---------------------- 4. 硬件与树莓派特定设置 ----------------------"
    echo ">> USB 设备:"
    lsusb
    echo ""
    echo ">> Config.txt 生效的配置 (数字):"
    vcgencmd get_config int
    echo ""
    echo ">> Config.txt 生效的配置 (字符串):"
    vcgencmd get_config str
    echo ""
    echo ">> 摄像头状态:"
    vcgencmd get_camera
    echo ""
    echo ">> 显示信息:"
    tvservice -s
    echo ""

} > "$REPORT_FILE"

echo "系统报告已生成至: $REPORT_FILE"
```

保存并退出（按 `Ctrl+X`，然后 `Y`，然后 `Enter`）。

给脚本添加执行权限并运行它：

使用方法：

```
chmod +x system_report.sh
./system_report.sh
```

脚本会在您的家目录（`/home/pi/`）下生成一个名为 `pi_system_report_日期时间.txt` 的文件，里面包含了您当前树莓派几乎所有的重要设置。

四、当前树莓派基本配置

硬件配置：

- **·型号**：Raspberry Pi 4 Model B Rev 1.5
- **·CPU**：4核Cortex-A72 @ 1.8GHz
- **·内存**：8GB LPDDR4
- **·存储**：外接SSD（470GB容量）
- **·系统**：Raspberry Pi OS 64-bit (Debian 12)

网络配置：

- **·主机名**：raspberrypi
- **·IP地址**：192.168.124.8（王村路由器）
- **·用户名**：pi
- **·密码**：已修改为安全密码

五、基础命令学习

1. 系统信息命令

```
hostname -I       # 查看IP地址
uname -a          # 查看系统信息
vcgencmd measure_temp  # 查看CPU温度
df -h             # 查看磁盘空间
lsblk            #查看挂载硬盘
free -h           # 查看内存使用
sudo ss -tulnp #查看端口信息
```

2. 文件操作命令

```
ls                # 列出文件
cd                # 切换目录
cd ~ #回到home
pwd               # 显示当前目录
mkdir             # 创建目录
rm                # 删除文件
cp                # 复制文件
mv                # 移动文件
nano              # 文本编辑
```

3. 网络管理命令

```
ping              # 测试网络连通性
ip addr           # 查看网络接口
nmcli             # NetworkManager命令行工具
```

4. 权限管理命令

```
sudo              # 超级用户权限
chmod             # 修改文件权限
chown             # 修改文件所有者
```

5. 进程管理命令

```
ps aux            # 查看所有进程
top               # 动态查看进程
htop              # 增强版top（需安装）
kill              # 终止进程
```

6. 包管理命令

```
sudo apt update           # 更新软件列表
sudo apt upgrade          # 升级所有软件
sudo apt install [软件包] # 安装软件
sudo apt remove [软件包]  # 卸载软件
```

7.Docker命令

```
#查看运行情况
docker ps
#查看指定容器的最近日志
docker logs <容器名或ID>
#查看 x-ui 容器的日志（假设容器名为 x-ui）
docker logs x-ui
#实时查看日志（类似 tail -f）
docker logs -f x-ui
#docker-compose启动
docker-compose up -d
#docker-compose停止
docker-compose down
```

六、基础服务安装

1. 设置临时网络代理

```
export http_proxy="http://192.168.124.23:10811"
export https_proxy="http://192.168.124.23:10811"
```

2. 系统更新

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y wget curl git
```

3. Docker安装

```
# 安装Docker
sudo apt install docker.io -y
sudo systemctl enable --now docker

# 将用户添加到docker组
sudo usermod -aG docker pi

# 注意：重新登录使更改生效
```

4. 独立Docker Compose安装

```
# 下载Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/download/v2.27.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 添加执行权限
sudo chmod +x /usr/local/bin/docker-compose

# 验证安装
docker-compose --version
```

如果最后一条命令输出了版本号（如 `Docker Compose version v2.27.1`），说明安装成功。

5. Docker代理配置

```
# 创建配置目录
sudo mkdir -p /etc/systemd/system/docker.service.d

# 创建代理配置文件
sudo nano /etc/systemd/system/docker.service.d/http-proxy.conf
```

添加以下内容：

```
[Service]
Environment="HTTP_PROXY=http://192.168.31.109:10811/"
Environment="HTTPS_PROXY=http://192.168.31.109:10811/"
Environment="NO_PROXY=localhost,127.0.0.1,*.lan"
```

保存并退出 按 `Ctrl + X`，然后按 `Y`，再按 `Enter`。

重载配置并重启Docker：

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

6.预添加新WiFi网络的步骤

我们将直接在存放网络配置的目录中，创建一个新的文件来定义这个未来的网络。

1. **进入NetworkManager的配置目录** 这个目录存放着所有已知的网络连接。

   ```
   cd /etc/NetworkManager/system-connections/
   ```

2. **创建一个新的连接配置文件** 我们需要使用 `sudo` 因为这个目录属于 `root` 用户。请将命令中的 `新WiFi名称` 替换为您未来要连接的那个WiFi的真实名称（SSID），这样方便您自己识别。

   ```
   # 例如，如果新WiFi名叫"Office-WiFi"，就设置为 "sudo nano Office-WiFi.nmconnection"
   sudo nano "新WiFi名称.nmconnection"
   ```

3. **写入新WiFi的配置信息** 将下面的**模板内容完整地复制**并粘贴到您刚刚打开的 `nano` 编辑器中。

```
[connection]
id=您新WiFi的名称
uuid=
type=wifi
interface-name=wlan0

[wifi]
mode=infrastructure
ssid=您新WiFi的名称

[wifi-security]
key-mgmt=wpa-psk
psk=您新WiFi的密码

[ipv4]
method=auto

[ipv6]
method=auto
```

4. **修改模板中的关键信息** 您需要编辑上面模板中的**三处**地方：
   - `id=` 后面：填入您新WiFi的名称（SSID），例如 `id=Office-WiFi`。
   - `ssid=` 后面：再次填入您新WiFi的名称，例如 `ssid=Office-WiFi`。
   - `psk=` 后面：填入您新WiFi的**明文密码**，例如 `psk=Password12345`。
   - `uuid=` 后面**留空**。当NetworkManager第一次加载这个文件时，会自动为您生成一个唯一的UUID并填充进去。
5. **保存并退出**
   - 按 `Ctrl+X`
   - 按 `Y` 确认保存
   - 按 `Enter` 确认文件名并退出

七、应用服务安装

1. x-ui面板安装（网络代理服务）（Docker部署）

分为两个主要部分：**服务端部署 (x-ui)** 和 **web端配置 节点链接**

1. **x-ui 服务端面板**：将树莓派作为代理服务器，供局域网内其他设备使用。
2. **web端配置 节点链接**：让树莓派自身能通过代理智能分流访问网络

1.1准备工作

- **GitHub 参考地址**: https://github.com/vaxilu/x-ui
- **外置硬盘挂载点**: 确认外置硬盘已挂载，本文档示例路径为 `/docker`。

1.2创建部署目录：

```
mkdir -p home/pi/docker/x-ui-data
cd  home/pi/docker/x-ui-data
```

1.3创建docker-compose.yml：

```
nano docker-compose.yml
```

写入以下内容

```
services:
  x-ui:
    image: enwaiax/x-ui:latest
    container_name: x-ui
    restart: unless-stopped
    network_mode: host
    volumes:
      - ./db:/etc/x-ui/
      - ./cert:/root/cert/
    environment:
      - XUI_LOG_LEVEL=warn
      - XUI_LOG_FILE=/etc/x-ui/x-ui.log
```

1.4启动服务：

```
docker-compose up -d
```

访问地址：http://树莓派IP:54321
默认账号：admin/admin

1.5 web端配置核心

让树莓派本身可以通过代理进行智能分流，用于 `apt`、`curl` 等命令行工具。

1. **编辑配置文件**:

   web端点击设置----xray设置----右边粘贴json配置----重启面板**写入最终版 JSON 配置**: (此配置已包含智能分流规则和您提供的机场节点信息)

   ```
   {
     "log": {
       "access": "",
       "error": "",
       "loglevel": "warning"
     },
     "dns": {
       "servers": [
         "1.1.1.1",
         "8.8.8.8"
       ]
     },
     "stats": {},
     "policy": {
       "system": {
         "statsOutboundUplink": true,
         "statsOutboundDownlink": true
       }
     },
   ```

```
  "inbounds": [
    {
      "tag": "socks-in",
      "port": 10808,
      "listen": "127.0.0.1",
      "protocol": "socks",
      "sniffing": {
        "enabled": true,
        "destOverride": ["http", "tls"]
      },
      "settings": {
        "auth": "noauth",
        "udp": true
      }
    },
    {
      "tag": "http-in",
      "port": 10809,
      "listen": "127.0.0.1",
      "protocol": "http",
      "sniffing": {
        "enabled": true,
        "destOverride": ["http", "tls"]
      },
      "settings": {}
    }
  ],
  "outbounds": [
    {
      "tag": "proxy",
      "protocol": "shadowsocks",
      "settings": {
        "servers": [
          {
            "address": "gheianygh.yangliq.com",
            "method": "aes-128-gcm",
            "ota": false,
            "password": "319600b1-309f-4275-be70-56c9e4f62358",
            "port": 31002,
            "level": 1
          }
        ]
      },
      "streamSettings": {
        "network": "tcp"
      },
      "mux": {
        "enabled": false
      }
    },
    {
      "tag": "direct",
      "protocol": "freedom",
      "settings": {}
    },
    {
```

```
      "tag": "block",
      "protocol": "blackhole",
      "settings": {
        "response": {
          "type": "http"
        }
      }
    }
  ],
  "routing": {
    "domainStrategy": "AsIs",
    "rules": [
      {
        "type": "field",
        "ip": [
          "geoip:private",
          "geoip:cn"
        ],
        "outboundTag": "direct"
      },
      {
        "type": "field",
        "domain": [
          "geosite:cn"
        ],
        "outboundTag": "direct"
      },
      {
        "type": "field",
        "domain": [
          "geosite:category-ads-all"
        ],
        "outboundTag": "block"
      },
      {
        "type": "field",
        "network": "tcp,udp",
        "outboundTag": "proxy"
      }
    ]
  }
}
```

如需修改节点,v2ray导出后，对应修改outbounds，出站规则即可

1.6配置系统全局代理

1. **编辑环境变量文件**:

```
sudo nano /etc/environment
```

2. **在文件末尾添加以下内容**，将系统代理指向 V2Ray 提供的 HTTP 端口：

```
http_proxy="http://127.0.0.1:10809"
https_proxy="http://127.0.0.1:10809"
no_proxy="localhost,127.0.0.1,::1,192.168.0.0/16,10.0.0.0/8,172.16.0.0/12,.local"
```

3. **更改docker代理为127.0.0.1:10809**
4. **重启树莓派使配置生效**:

```
sudo reboot
```

重启后，树莓派的命令行环境即可自动使用 V2Ray 进行智能分流代理。

1.7（可选操作）Shell 配置文件配置全局局域网代理

代理通常设置在用户的 Shell 初始化文件中。根据你的 Shell（`bash`），检查以下文件：

```
# 查看当前用户的环境配置文件
cat ~/.bashrc | grep -i proxy
cat ~/.bash_profile | grep -i proxy
cat ~/.profile | grep -i proxy

# 如果使用 zsh（默认不是，但可检查）
cat ~/.zshrc | grep -i proxy
```

用 `nano` 编辑器打开文件：

```
nano ~/.bashrc
```

找到这两行（你刚才用 `grep` 已经看到了它们）：

```
export http_proxy="http://192.168.124.20:10811"
export https_proxy="http://192.168.124.20:10811"
```

删除或注释掉它们：

- •**删除**：直接删除这两行。
- **注释**（推荐，可保留记录）：在行首加 `#`，变成：

```
#export http_proxy="http://192.168.124.20:10811"
#export https_proxy="http://192.168.124.20:10811"
```

保存并退出：

- •按 `Ctrl + O` 保存 → 回车确认 → `Ctrl + X` 退出。

立即生效：

```
source ~/.bashrc  # 重新加载配置
env | grep -i proxy  # 验证代理变量是否已消失
```

2. 皎月连安装（内网穿透）（docker部署）

官网：皎月连一键内网穿透

2.1创建部署目录：

```
mkdir -p home/pi/docker/jiaoyuelian
cd home/pi/docker/jiaoyuelian
```

2.2创建docker-compose.yml：

1. **创建文件**： 在 `jiaoyuelian` 目录下，创建一个新的 `docker-compose.yml` 文件。

   ```
   nano docker-compose.yml
   ```

2. **粘贴内容**： 将以下内容完整复制并粘贴到文件中。

```
version: '3.8'
services:
  natpierce:
    image: xiyu505/natpierce:latest
    container_name: natpierce
    restart: always
    network_mode: host
    privileged: true
    volumes:
      - natpierce_data:/natpierce

volumes:
  natpierce_data:
```

2.3启动服务：

```
docker-compose up -d
```

2.4访问面板

访问地址：http://树莓派IP:33272
默认账号：zcjczy@126.com/带*1

3. Immich相册服务安装（docker部署）

3.1. 准备工作

在开始之前，请确保完成以下基础准备。

3.1.1. (前置要求) 确认64位操作系统

这是安装成功的**绝对前提**。请运行以下命令确认您的操作系统用户空间是64位的。

```
dpkg --print-architecture
```

预期输出必须是 `arm64`。如果输出是 `armhf`，则您需要停止安装，并为树莓派重装一个64位的操作系统。

文件格式ext4

```
df -Th
```

3.1.2. 系统与工具更新

确保系统及软件包为最新状态，并安装 `git` 工具用于下载配置。

```
sudo apt update && sudo apt full-upgrade -y
sudo apt install git -y
```

3.1.3. 准备外部存储

照片和数据库**必须**存放在外置硬盘（推荐SSD）上，而不是SD卡。

1. **连接硬盘**：将您的外置硬盘连接到树莓派的 USB 3.0 端口。
2. **确认挂载路径**：使用 `lsblk` 命令找到您的硬盘挂载点。本文档将以 `home/pi` 为例，**请在后续步骤中替换为您自己的真实路径。**
3. **创建数据目录**：

   ```
   # 请将 home/pi 替换为您的真实路径
   mkdir -p /home/pi/immich/photos
   #注意，数据库文件夹在sd卡上，因为外接硬盘是exFAT 或 NTFS格式。数据库要求ext4或其他支持linux权限的格式
   mkdir -p /home/pi/immich/database

   # 授予当前pi用户读写权限
   sudo chown -R pi:pi /home/pi/immich/
   ```

3.2. 获取并配置 Immich

3.2.1. 下载官方配置文件

```
# 创建docker/immich目录
mkdir -p /home/pi/docker/immich
cd /home/pi/docker/immich
# 下载 docker-compose.yml 和 .env 模板
wget -O docker-compose.yml https://github.com/immich-app/immich/releases/latest/download/docker-compose.yml
wget -O .env https://github.com/immich-app/immich/releases/latest/download/example.env
```

3.2.2. 配置 `.env` 文件

进入配置目录，并根据模板创建您的环境配置文件。

```
nano .env
```

在打开的编辑器中，修改以下关键内容：

```
# You can find documentation for all the supported env variables at
https://immich.app/docs/install/environment-variables

# The location where your uploaded files are stored
UPLOAD_LOCATION=/home/pi/immich/photos

# The location where your database files are stored. Network shares are not
supported for the database
DB_DATA_LOCATION=/home/pi/immich/database

# To set a timezone, uncomment the next line and change Etc/UTC to a TZ
identifier from this list:
https://en.wikipedia.org/wiki/List_of_tz_database_time_zones#List
# TZ=Etc/UTC
TZ=Asia/Shanghai
# The Immich version to use. You can pin this to a specific version like
"v1.71.0"
IMMICH_VERSION=release

# Connection secret for postgres. You should change it to a random password
# Please use only the characters `A-Za-z0-9`, without special characters or
spaces
DB_PASSWORD=postgres

# The values below this line do not need to be changed
################################################################################
##
DB_USERNAME=postgres
DB_DATABASE_NAME=immich

VIPS_NOVECTOR=1
IMMICH__MACHINE__LEARNING__ENABLED=false
```

更改四个设置`UPLOAD_LOCATION` `DB_DATA_LOCATION` `TZ=` `VIPS_NOVECTOR`分别为照片地址，数据库地址，时区，向量指令集。

关键点`VIPS_NOVECTOR=1`禁用VIPS指令，提高兼容性，解决exit 132问题

`IMMICH__MACHINE__LEARNING__ENABLED=false`关闭AI功能

修改完成后，按`Ctrl+X` -> `Y` -> `Enter`保存并退出。

3.3. 最终的 `docker-compose.yml` 文件

1. **打开 `docker-compose.yml` 文件**:

   ```
   nano docker-compose.yml
   ```

2. **完整替换文件内容**： 请删除文件内的所有内容，然后将下面已经为您准备好的完整配置**全部复制**并粘贴进去。

   YAML

   ```
   #
   ```

```
# WARNING: To install Immich, follow our guide:
[https://immich.app/docs/install/docker-compose]
(https://immich.app/docs/install/docker-compose)
#
# Make sure to use the docker-compose.yml of the current release:
#
# [https://github.com/immich-app/immich/releases/latest/download/docker-
compose.yml](https://github.com/immich-
app/immich/releases/latest/download/docker-compose.yml)
#
# The compose file on main may not be compatible with the latest release.

name: immich

services:
  immich-server:
    container_name: immich_server
    image: ghcr.io/immich-app/immich-server:${IMMICH_VERSION:-release}
    # extends:
    #   file: hwaccel.transcoding.yml
    #   service: cpu # set to one of [nvenc, quicksync, rkmpp, vaapi, vaapi-
wsl] for accelerated transcoding
    volumes:
      # Do not edit the next line. If you want to change the media storage
location on your system, edit the value of UPLOAD_LOCATION in the .env file
      - ${UPLOAD_LOCATION}:/data
      - /etc/localtime:/etc/localtime:ro
    env_file:
      - .env
    ports:
      - '2283:2283'
    depends_on:
      - redis
      - database
    restart: always
    healthcheck:
      disable: false

  immich-machine-learning:
    container_name: immich_machine_learning
    # For hardware acceleration, add one of -[armnn, cuda, rocm, openvino,
rknn] to the image tag.
    # Example tag: ${IMMICH_VERSION:-release}-cuda
    image: ghcr.io/immich-app/immich-machine-learning:${IMMICH_VERSION:-
release}
    # extends: # uncomment this section for hardware acceleration - see
[https://immich.app/docs/features/ml-hardware-acceleration]
(https://immich.app/docs/features/ml-hardware-acceleration)
    #   file: hwaccel.ml.yml
    #   service: cpu # set to one of [armnn, cuda, rocm, openvino, openvino-
wsl, rknn] for accelerated inference - use the `-wsl` version for WSL2 where
applicable
    volumes:
      - model-cache:/cache
    env_file:
      - .env
```

```
    restart: always
    healthcheck:
      disable: false

  redis:
    container_name: immich_redis
    image: docker.io/valkey/valkey:8-
bookworm@sha256:a137a2b60aca1a75130022d6bb96af423fefae4eb55faf395732db3544803
280
    healthcheck:
      test: redis-cli ping || exit 1
    restart: always

  database:
    container_name: immich_postgres
    image: ghcr.io/immich-app/postgres:14-vectorchord0.4.3-
pgvectors0.2.0@sha256:32324a2f41df5de9efe1af166b7008c3f55646f8d0e00d9550c16c9
822366b4a
    environment:
      POSTGRES_PASSWORD: ${DB_PASSWORD}
      POSTGRES_USER: ${DB_USERNAME}
      POSTGRES_DB: ${DB_DATABASE_NAME}
      POSTGRES_INITDB_ARGS: '--data-checksums'
      # Uncomment the DB_STORAGE_TYPE: 'HDD' var if your database isn't
stored on SSDs
      # DB_STORAGE_TYPE: 'HDD'
    volumes:
      # Do not edit the next line. If you want to change the database storage
location on your system, edit the value of DB_DATA_LOCATION in the .env file
      - ${DB_DATA_LOCATION}:/var/lib/postgresql/data
    shm_size: 128mb
    restart: always

volumes:
  model-cache:
```

3. 按 `Ctrl+X` -> `Y` -> `Enter` 保存并退出。

3.4. 启动与验证

3.4.1. 启动服务

在 `immich/docker` 目录下，运行启动命令。

```
docker-compose up -d
```

Docker会使用精确的哈希值去拉取镜像。如果您之前已经手动拉取过，它会直接跳过下载并立即启动。

3.4.2. 检查状态

等待几分钟，让所有服务完成初始化，然后运行以下命令查看状态：

```
docker-compose ps
```

您应该能看到所有服务都处于 `Up` 或 `running` 状态。

3.4.3. 初次访问

在您的电脑浏览器中，访问 `http://<你的树莓派IP地址>:2283`。您会看到 Immich 的欢迎页面，请根据指引注册您的第一个管理员账户。

```
#管理员账户
zcjczy@126.com
612.
Jia-Hzz LI
```

3.4.4.停止并移除机器学习容器

```

docker-compose stop immich-machine-learning
docker-compose rm -f immich-machine-learning
```

3.5 手动配置AI模型

模型分为两类， CLIP 模型（智能搜索）和人脸识别模型。

3.5.1准备工作

建立模型目录

```
#安装git-lfs
sudo apt-get install git-lfs

#总目录
cd home/pi/docker/immich  #此处存放.yml和.env文件
mkdir -p model-cache
#CLIP目录和人脸
cd model-cache
mkdir clip
mkdir facial-recognition
```

3.5.2下载模型

```
#人脸识别模型 下载
cd /home/pi/docker/immich/model-cache/facial-recognition
git clone https://huggingface.co/immich-app/buffalo_l
#人脸识别模型 启用
cd buffalo_l
git lfs install
git lfs pull
#CLIP模型 下载
cd /home/pi/docker/immich/model-cache/clip
git clone https://huggingface.co/immich-app/XLM-Roberta-Large-Vit-B-16Plus
#或更新更强的 nllb-clip-large-siglip__v1
#或原配ViT-B-32__openai
#CLIP模型 启用
cd XLM-Roberta-Large-Vit-B-16Plus
git lfs install
git lfs pul
```

3.5.3 docker模型文件夹映射

```
#编辑.yml
nano docker-compose.yml
#更改
  immich-machine-learning:
    container_name: immich_machine_learning
    # For hardware acceleration, add one of -[armnn, cuda, rocm, openvino, rknn]
to the image tag.
    # Example tag: ${IMMICH_VERSION:-release}-cuda
    image: ghcr.io/immich-app/immich-machine-learning:${IMMICH_VERSION:-release}
    # extends: # uncomment this section for hardware acceleration - see
[https://immich.app/docs/features/ml-hardware-acceleration]
(https://immich.app/docs/features/ml-hardware-acceleration)
    #   file: hwaccel.ml.yml
    #   service: cpu # set to one of [armnn, cuda, rocm, openvino, openvino-wsl,
rknn] for accelerated inference - use the `-wsl` version for WSL2 where
applicable
    volumes:
      - ./model-cache:/cache
```

`model-cache:/cache` 改为 `./model-cache:/cache`

3.5.4重启容器

```
docker-compose down
docker-compose up -d
```

3.5.5web端改模型

web端设置修改

3.5. 后续维护

3.5.1. 如何更新 Immich

1. **访问镜像更新地址**:
   - `immich-server`: https://github.com/immich-app/immich/pkgs/container/immich-server
   - `immich-machine-learning`: https://github.com/immich-app/immich/pkgs/container/immich-machine-learning
   - `database` (postgres): https://github.com/orgs/immich-app/packages/container/package/postgres
   - `redis` (valkey): https://hub.docker.com/_/valkey/tags
2. **查找新哈希值**： 按照我们之前实践过的方法，进入相应页面，找到新版本对应的 `linux/arm64` 的新哈希值。
3. **更新 `.yml` 文件**： 手动编辑 `docker-compose.yml` 文件，将旧的哈希值替换为新找到的哈希值。
4. **拉取并重启服务**： 在 `immich/docker` 目录下运行：

   ```
   docker-compose pull
   docker-compose up -d
   ```

3.5.2. 如何备份

您的所有核心数据都在外置硬盘上。请务必定期备份以下四个目录：

- `home/pi/immich/photos/backups`,`home/pi/immich/photos/library`,`home/pi/immich/photos/profile`,`home/pi/immich/photos/upload` 四个文件夹复制到home/pi/immich_backup`

备份方案

- 每天凌晨3点执行sh脚本，将以上四个文件夹复制到`home/pi/immich_backup`，windows电脑使用filefreesync软件创建同步文件夹即使同步，并设置定时计划将文件夹打包上传至云盘

1.树莓派操作

- ```
  # 脚本sh
  sudo nano /usr/local/bin/backup_immich.sh
  ```

- ```
  #!/bin/bash
  #脚本内容
  # 定义源目录和目标目录
  SOURCE_DIR="/home/pi/immich/photos"
  BACKUP_DIR="/home/pi/immich_backup"

  # 需要备份的子目录列表
  DIRS_TO_BACKUP=("backups" "library" "profile" "upload")

  # 创建备份目录（如果不存在）
  mkdir -p "$BACKUP_DIR"

  # 记录日志（可选）
  LOG_FILE="/home/pi/immich_backup/immich_backup.log"
  echo "===== 备份开始 [$(date '+%Y-%m-%d %H:%M:%S')] =====" >> "$LOG_FILE"

  # 循环备份每个目录
  for DIR in "${DIRS_TO_BACKUP[@]}"; do
      SOURCE="$SOURCE_DIR/$DIR"
      TARGET="$BACKUP_DIR/$DIR"

      # 检查源目录是否存在
      if [ -d "$SOURCE" ]; then
          echo "备份中: $SOURCE -> $TARGET" >> "$LOG_FILE"
          rsync -av --delete "$SOURCE/" "$TARGET/" >> "$LOG_FILE" 2>&1
      else
          echo "警告: 源目录不存在 [$SOURCE]" >> "$LOG_FILE"
      fi
  done

  echo "===== 备份完成 [$(date '+%Y-%m-%d %H:%M:%S')] =====" >> "$LOG_FILE"
  ```

- **设置脚本权限**

  ```
  sudo chmod +x /usr/local/bin/backup_immich.sh
  ```

配置每日定时任务

编辑当前用户的 cron 任务：

```
crontab -e
```

添加以下行（每天凌晨 3 点执行）：

```
0 3 * * * /usr/local/bin/backup_immich.sh
```

修改 `3` 为其他小时可调整备份时间（如 `0` 表示午夜）。

2.windows操作之同步

- 下载安装FileFreeSync软件
- 添加同步文件夹对，可同步添加study，work..等其他文件夹
- 设置同步方式为镜像，可过滤部分文件夹
- 另存为批处理文件
- 打开realtimesync，拖入批处理文件，设置周期执行

3.windows操作至定时压缩至百度云同步文件夹

- 创建backup.bat
- ```
  @echo off
  setlocal EnableDelayedExpansion

  REM ============ 核心配置 ============
  set "SOURCE_FOLDER=D:\workBackupNoZip"
  set "OUTPUT_FOLDER=D:\BaiduyunBackup"
  set "ZIP_PASSWORD=zn15188636576.."
  set "BANDIZIP_PATH=Bandizip.exe"
  set "TARGET_DAY_OF_WEEK=6"  REM 每周五备份 (1=周一,7=周日)
  set "FLAG_FILE=%OUTPUT_FOLDER%\last_backup.week"

  REM ============ 高级压缩配置 ============
  set "COMPRESSION_LEVEL=0"     REM 0(存储)-9(最大压缩)
  set "THREAD_COUNT=0"          REM 0=自动使用所有CPU核心
  set "EXCLUDE_LIST=*.tmp;*.bak"  REM 用分号分隔的排除文件类型
  REM ====================================

  REM 使用PowerShell获取ISO周数（兼容新版Windows）
  for /f "delims=" %%a in ('powershell -command "(Get-Date).Year"') do set
  "CurrentYear=%%a"
  for /f "delims=" %%a in ('powershell -command "(Get-
  Culture).Calendar.GetWeekOfYear((Get-Date),
  [System.Globalization.CalendarWeekRule]::FirstFourDayWeek,
  [DayOfWeek]::Monday)"') do set "CurrentWeek=%%a"
  for /f "delims=" %%a in ('powershell -command "[int](Get-Date).DayOfWeek +
  1"') do set "CURRENT_DOW=%%a"

  set "CURRENT_YEAR_WEEK=!CurrentYear!-!CurrentWeek!"

  REM 检查标志文件
  if exist "%FLAG_FILE%" (
      set /p LAST_BACKUP_WEEK=<"%FLAG_FILE%"
  ```
  ```

```
) else (
    set "LAST_BACKUP_WEEK="
)

echo 当前日期: %date%
echo 当前年份和周数: !CURRENT_YEAR_WEEK!
echo 上次备份: !LAST_BACKUP_WEEK!
echo 目标备份: 每周星期%TARGET_DAY_OF_WEEK%

if "!LAST_BACKUP_WEEK!"=="" (
    echo 首次运行，执行备份...
    goto :do_compress
) else if "!LAST_BACKUP_WEEK!" LSS "!CURRENT_YEAR_WEEK!" (
    echo 新的一周，执行备份...
    goto :do_compress
) else if !CURRENT_DOW! geq %TARGET_DAY_OF_WEEK% (
    echo 本周未备份，执行备份...
    goto :do_compress
) else (
    echo 本周已备份，跳过...
    goto :end_script
)

:do_compress
REM ----- 使用Bandizip压缩 -----
if not exist "%OUTPUT_FOLDER%" mkdir "%OUTPUT_FOLDER%"
if not exist "%SOURCE_FOLDER%" (
    echo 错误：源文件夹不存在 - %SOURCE_FOLDER%
    pause
    exit /b 1
)

set "EXCLUDE_SWITCH="
if defined EXCLUDE_LIST set "EXCLUDE_SWITCH=-ex:"!EXCLUDE_LIST!""

pushd "%SOURCE_FOLDER%"
echo 开始压缩到: %OUTPUT_FOLDER%

for /D %%F in (*) do (
    echo 正在处理: %%F
    "%BANDIZIP_PATH%" c -y -p:%ZIP_PASSWORD% -fmt:7z -l:%COMPRESSION_LEVEL% -
t:%THREAD_COUNT% %EXCLUDE_SWITCH% "%OUTPUT_FOLDER%\%%F.7z" "%%F\"
    if !errorlevel! equ 0 (
        echo ? 压缩成功
    ) else (
        echo ? 压缩失败 (错误码: !errorlevel!)
    )
    echo.
)

popd
echo !CURRENT_YEAR_WEEK! > "%FLAG_FILE%"
echo 备份完成! 标志已更新: !CURRENT_YEAR_WEEK!

:end_script
echo.
```

```
echo 任务状态: %errorlevel%
pause
```

- 打开windows任务计划---创建计划任务---触发器选择每周---勾选时间已过时直接执行该脚本

4. Samba文件夹局域网共享

步骤 1：安装 Samba 软件包

```
sudo apt update
sudo apt install samba samba-common-bin -y
```

步骤 2：配置 Samba（设置共享目录和权限）

1. 1.备原始配置文件：

```
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.bak
```

2. 2.编辑配置文件：

```
sudo nano /etc/samba/smb.conf
```

3. 3.在文件**末尾**添加以下配置段（请将 `/media/pi/SNF` 替换为您想共享的实际路径）：

```
# 我的SSD共享配置
[SSD-Share]
   comment = Raspberry Pi SSD Share
   path = /media/pi/SNF
   browseable = yes
   read only = no
   guest ok = no
   create mask = 0777
   directory mask = 0777
   force user = pi
```

   - • `[SSD-Share]`：共享名称，在电脑上会看到这个名称。
   - • `path`：要共享的目录路径（**您SSD的挂载点**）。
   - • `read only = no`：允许读写。
   - • `guest ok = no`：禁止访客访问，需要密码。
   - • `force user = pi`：强制将所有访问权限映射给 `pi` 用户，避免权限问题。

步骤 3：设置 Samba 用户密码

为系统用户 `pi` 设置一个专门的Samba访问密码：

```
sudo smbpasswd -a pi
```

按提示输入两次密码（**可以和系统登录密码不同，建议设置一下**）。

步骤 4：重启 Samba 服务使配置生效

```
sudo systemctl restart smbd nmbd
```

步骤 5：在 Windows 电脑上访问

1. 1.打开`文件资源管理器`，在地址栏输入：

   ```
   \\树莓派的IP地址
   ```

   例如：`\\192.168.124.8`
2. 2.在弹出的登录窗口中，用户名输入 `pi`，密码输入您刚才用 `smbpasswd` 设置的密码。
3. 3.勾选“记住我的凭据”，以后就能直接访问了。

可能出现的权限问题1

- 该目录可能属于 `root` 或其他用户，而当前用户（如 `pi`）没有访问权限。
- 检查权限的命令：

  ```
  ls -ld /home/pi/immich/photos/
  ```

  输出示例：

  ```
  drwx------ 26 dnsmasq pi 4096 Sep  4 13:48 /home/pi/immich/photos/
  ```

  1. 1所有权问题：
     - 所有者是 `dnsmasq` 用户（可能是错误配置）
     - 组是 `pi` 组
  2. 2.权限问题
     - `drwx------` 表示只有所有者（dnsmasq）有完全权限
     - 组用户（pi）和其他用户没有任何权限

  解决方案：

  1. 修改目录所有权（推荐）

  将目录所有者改为 `pi` 用户：

  ```
  sudo chown -R pi:pi /home/pi/immich/photos/
  ```

  2. 调整目录权限

  给予所有者完全权限，组用户读和执行权限：

  ```
  sudo chmod -R 750 /home/pi/immich/photos/
  ```

  或者如果需要其他用户也能访问（安全性较低）：

  ```
  sudo chmod -R 755 /home/pi/immich/photos/
  ```

可能出现的权限问题2

系统无法识别

Windows 10/11 默认安全策略禁止未经身份验证的来宾访问网络共享，而您的 Samba 共享可能被识别为"来宾访问"。

解决方案

修改 Windows 安全策略

按 `Win + R` 输入 `gpedit.msc` 打开组策略编辑器

导航到：

计算机配置 → 管理模板 → 网络 → Lanman 工作站

双击 **"启用不安全的来宾登录"**

选择 **"已启用"** → 点击"确定"

重启电脑或运行 `gpupdate /force`

5. AdGuard Home 安装打造家庭广告屏蔽系统

目标： 利用一台树莓派， 安装AdGuard Home，为家庭中的所有联网设备（手机、电脑、电视等）提供无广告、更安全的网络环境。

5.1 核心组件与资源

1. **软件：**
   - **AdGuard Home**: 开源、强大的网络广告拦截软件。
2. **官方网站与文档：**
   - **AdGuard Home 官网**: https://adguard.com/adguard-home/overview.html
   - **官方安装指南 (GitHub)**: https://github.com/AdguardTeam/AdGuardHome/wiki/Getting-Started
   - **官方 Docker 镜像**: https://hub.docker.com/r/adguard/adguardhome

5.2 安装部署 (Docker Compose 方案)

采用 Docker Compose 方式安装 AdGuard Home，便于管理和维护。

1. **前期准备**:
   - 在树莓派上安装好 Raspberry Pi OS Lite (64位)。
   - 安装 Docker 和 Docker Compose。
   - 为树莓派设置一个**静态 IP 地址** (例如: `192.168.31.203`)，这是后续所有配置的基础。（路由器管理界面，局域网管理指定设备地址即可）
2. **创建并配置 `docker-compose.yml`**:
   - 通过 SSH 登录树莓派。
   - 创建工作目录并进入：

     ```
     mkdir docker/adguard-home && cd docker/adguard-home
     ```

- 创建 `docker-compose.yml` 文件，并写入以下内容：

```
version: "3.7"
services:
  adguardhome:
    image: adguard/adguardhome
    container_name: adguardhome
    ports:
      # DNS 服务端口 (必须)
      - "53:53/tcp"
      - "53:53/udp"
      # 管理后台 Web UI 端口
      - "80:80/tcp"
      - "443:443/tcp"
      # 首次启动的设置向导端口
      - "3000:3000/tcp"
      # 加密 DNS 端口 (可选，但推荐)
      - "853:853/tcp"
    volumes:
      - ./workdir:/opt/adguardhome/work
      - ./confdir:/opt/adguardhome/conf
    restart: unless-stopped
```

3. **启动服务**:
   - 在 `docker-compose.yml` 所在目录下，运行：

```
docker-compose up -d
```

4. **首次向导**:
   - 浏览器访问 `http://[树莓派静态IP]:3000`。
   - 按照向导完成管理员账户的创建。
   - 基本无改动，都为默认

5.3 网络配置 (关键步骤)

1. **登录家庭路由器后台。**
2. **找到 DHCP 服务器设置**: 关键在于找到 `局域网(LAN)设置` 或 `DHCP服务` 菜单，**而不是“上网设置(WAN)”。**
3. **配置 DNS 服务器**:
   - **首选 DNS (DNS1)**: 填入树莓派的静态 IP 地址 (`192.168.31.203`)。
   - **备用 DNS (DNS2)**: **务必留空**，或重复填写树莓派的 IP。这可以防止设备绕过广告拦截。
4. **保存并重启路由器。**
5. 将手机、电脑等设备**断开网络后重连**，以获取新的 DNS 配置。

5.4 AdGuard Home 终优化配置

1. **常规设置 (Settings → General Settings)**
   - **过滤器更新周期**: 24 小时。
   - **启用 AdGuard「浏览安全」网页服务**: **勾选** (增强安全性，若无法上网取消勾选)。

- **日志/统计数据保留时间**: 90 天 (在性能和历史记录之间取得平衡)。

2. **DNS 设置 (Settings → DNS Settings)**
   - **上游 DNS 服务器**:
     - **关键心得**: 放弃使用因网络环境导致不稳定的国外 DNS (如 Cloudflare)，**专注使用国内稳定、快速的加密 DNS 服务。**
     - **最终配置 (清空默认值后填入)**:

```
https://doh.pub/dns-query
https://dns.alidns.com/dns-query
```

   - **请求处理方式**: 选择 **并行请求** (同时查询多个上游，取最快结果)。
   - **Bootstrap DNS 服务器**: 填入国内可靠的 DNS IP。

```
223.5.5.5
119.29.29.29
```

3. **过滤器设置 (Filters → DNS blocklists)**
   - **核心理念**: **化繁为简。**与其订阅大量零散的规则列表，不如使用一个高质量的、由社区维护的聚合规则。
   - **清理默认规则**: 取消勾选或删除所有 AdGuard Home 默认自带的以及之前手动添加的广告拦截列表，避免重复和冲突。
   - **添加最终整合规则**:
     - 点击 **“添加拦截清单” -> “添加一个自定义列表”。**
     - **列表名称**: `Heidai Rules (推荐)`
     - **列表 URL (使用国内加速链接)**:

```
https://gcore.jsdelivr.net/gh/217heidai/adblockfilters@main/rules/adblockdns.txt
```

   - **为什么用这个规则?**
     - 它自动聚合了多个主流规则源。
     - 会自动去重并**移除已失效的域名**，优化性能。
     - 提供国内加速链接，完美解决因网络问题导致的规则更新失败（例如 `connection reset by peer` 错误）。
     - 有 Lite 版本 (`adblockdnslite.txt`) 可作为备用选项。
     - 项目地址：[217heidai/adblockfilters: 去广告合并规则，每8个小时更新一次。]

5.5 验证与问题排查笔记

1. **如何验证生效?**
   - **仪表盘**: AdGuard Home 首页的“DNS 查询总数”和“已拦截”数量应该在上网时持续增长。
   - **广告测试网站**: 访问 `https://adblock-tester.com/`，分数应在 90% 以上。
   - **实际体验**: 常用网站和 App 的广告明显消失。

八、自动运行脚本服务设计

1.每日监控报告发送至企业微信

该脚本能够收集系统状态、Docker容器信息、AdGuard Home统计数据以及Immich备份日志，并将格式化后的精美报告每日定时发送到企业微信群。

1.1 安装依赖和配置

安装Python

```
sudo apt update
sudo apt install python3-pip -y
```

创建脚本文件：

```
mkdir -p /home/pi/scripts/pi-monitor
nano /home/pi/scripts/pi-monitor/pi_monitor.py
```

1.2 创建并激活Python虚拟环境

为保持项目依赖的独立性，推荐使用虚拟环境。

```
# 1. 进入项目目录
cd /home/pi/scripts/pi-monitor/

# 2. 创建名为 venv 的虚拟环境
python3 -m venv venv

# 3. 激活虚拟环境
source venv/bin/activate
```

激活后，终端提示符前会出现 `(venv)` 标识。

1.3 安装依赖库

脚本运行需要 `requests` 库来与企业微信和AdGuard Home的API进行通信。

```
# 确保在已激活的虚拟环境中执行
pip install requests
```

1.4 监控脚本编写

将以下完整代码保存为 `/home/pi/scripts/pi-monitor/pi_monitor.py` 文件。

注意： 脚本顶部的 `__init__` 方法内包含了所有需要您根据实际情况修改的配置项。

```
#!/home/pi/scripts/pi-monitor/venv/bin/python3
# -*- coding: utf-8 -*-
"""
树莓派状态监控脚本（功能增强与美化版）
"""

import json
```

```
import subprocess
import requests
from requests.auth import HTTPBasicAuth
import datetime
import os

class PiStatusMonitor:
    def __init__(self):
        # --- 请在这里集中配置您的信息 ---

        # 企业微信机器人Webhook Key
        self.wechat_key = "875d0f20-....."

        # AdGuard Home 配置
        self.adguard_host = "127.0.0.1"
        self.adguard_port = 80
        self.adguard_user = "zcj..."
        self.adguard_pass = "hua..."

        # 文件路径配置
        self.immich_backup_log = "/home/pi/immich_backup/immich_backup.log"

        # --- 配置结束 ---

        # 根据配置生成内部变量
        self.wechat_webhook = f"[https://qyapi.weixin.qq.com/cgi-
bin/webhook/send?key=](https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=)
{self.wechat_key}"
        self.adguard_api_url = f"http://{self.adguard_host}:
{self.adguard_port}/control/stats"
        self.session = requests.Session()
        self.session.auth = HTTPBasicAuth(self.adguard_user, self.adguard_pass)

    def run_command(self, cmd):
        """执行shell命令并返回结果"""
        try:
            result = subprocess.run(cmd, shell=True, capture_output=True,
text=True, timeout=30)
            return result.stdout.strip() if result.returncode == 0 else f"Error:
{result.stderr.strip()}"
        except subprocess.TimeoutExpired:
            return "命令执行超时"
        except Exception as e:
            return f"命令执行异常: {str(e)}"

    def get_system_info(self):
        """获取系统基本信息"""
        info = {}
        temp_output = self.run_command("vcgencmd measure_temp")
        info['cpu_temp'] = temp_output.replace("temp=", "").replace("'C", "°C")
if "temp=" in temp_output else temp_output
        uptime_output = self.run_command("uptime -p")
        info['uptime'] = uptime_output.replace("up ", "") if uptime_output and
not uptime_output.startswith("Error") else "N/A"
        memory = self.run_command("free -h | grep Mem | awk '{print $3\"/\"$2}'")
```

```
        info['memory'] = memory if memory and not memory.startswith("Error") else
"N/A"
        disk_output = self.run_command("df -h / | awk 'NR==2{print $3\"/\"$2\"
(\"$5\")'}")
        info['disk_usage'] = disk_output if disk_output and not
disk_output.startswith("Error") else "N/A"
        ip = self.run_command("hostname -I")
        info['ip_address'] = ip.split()[0] if ip and not ip.startswith("Error")
else "N/A"
        return info

    def get_docker_info(self):
        """获取Docker容器信息"""
        containers = []
        docker_ps = self.run_command("docker ps --format '{{.Names}}|{{.State}}|
{{.Image}}'")
        if docker_ps and not docker_ps.startswith("Error"):
            for line in docker_ps.split('\n'):
                if '|' in line:
                    name, state, image = line.split('|', 2)
                    status_color = "#2ECC71" if state == "running" else "#E74C3C"
                    containers.append({
                        'name': name,
                        'status': state,
                        'image': image.split(':')[0],
                        'color': status_color
                    })
        return containers

    def get_adguard_stats(self):
        """通过一次API调用获取AdGuard Home完整统计数据"""
        stats = {'basic': {}, 'top': {}}
        try:
            response = self.session.get(self.adguard_api_url, timeout=10)
            if response.status_code == 401:
                stats['basic']['error'] = "认证失败,请检查用户名或密码"
                return stats

            response.raise_for_status()
            data = response.json()

            # 基础统计
            dns_queries = data.get('num_dns_queries', 0)
            blocked_filter = data.get('num_blocked_filtering', 0)
            stats['basic'] = {
                'dns_queries': dns_queries,
                'blocked_filter': blocked_filter,
                'blocked_percentage': round((blocked_filter / max(1,
dns_queries)) * 100, 2),
                'blocked_safebrowsing': data.get('num_replaced_safebrowsing', 0),
                'blocked_parental': data.get('num_replaced_parental', 0),
                'avg_processing_time': data.get('avg_processing_time', 0) * 1000
            }

            # Top统计 (API返回的是字典列表,需转换)
            def transform_top_data(data_list):
```

```
                if not isinstance(data_list, list): return []
                return [list(item.items())[0] for item in data_list]

            stats['top'] = {
                'top_queried': transform_top_data(data.get('top_queried_domains',
[]))[:5],
                'top_blocked': transform_top_data(data.get('top_blocked_domains',
[]))[:5],
                'top_clients': transform_top_data(data.get('top_clients', []))
[:3]
            }
        except requests.exceptions.RequestException as e:
            stats['basic']['error'] = f"API连接错误: {e}"
        except json.JSONDecodeError:
            stats['basic']['error'] = "API返回非JSON数据"
        return stats

    def get_immich_backup_info(self):
        """获取Immich备份信息"""
        backup_info = {"last_backup": "无备份日志文件", "status": "unknown"}
        if os.path.exists(self.immich_backup_log):
            try:
                with open(self.immich_backup_log, 'r', encoding='utf-8') as f:
                    lines = f.readlines()

                backup_info["last_backup"] = "日志中无备份记录"
                for line in reversed(lines):
                    clean_line = line.strip().replace("=====", "").strip()
                    if "备份完成" in clean_line:
                        backup_info['last_backup'] = clean_line
                        backup_info['status'] = "success"
                        break
                    elif "备份开始" in clean_line:
                        backup_info['last_backup'] = clean_line
                        backup_info['status'] = "in_progress"
                        break
            except Exception as e:
                backup_info['last_backup'] = f"日志读取错误: {e}"
        return backup_info

    def send_to_wechat(self, message):
        """发送消息到企业微信"""
        payload = {"msgtype": "markdown", "markdown": {"content": message}}
        try:
            response = requests.post(self.wechat_webhook, json=payload,
timeout=10)
            response.raise_for_status()
            return True
        except Exception as e:
            print(f"发送到企业微信失败: {e}")
            return False

    def generate_adguard_card(self, title, value, percent, color):
        """生成AdGuard统计卡片"""
        colors = {"blue": "#3498DB", "red": "#E74C3C", "green": "#2ECC71",
"yellow": "#F39C12"}
```

```
        text_color = colors.get(color, "#3498DB")
        percent_str = f"<font color=\"#7F8C8D\">({percent}%)</font>" if percent
else ""
        return f"> **{title}**\n> ## <font color=\"{text_color}\">{value}</font>
{percent_str}\n"

    def generate_report(self):
        """生成手机友好的监控报告"""
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        system_info = self.get_system_info()
        docker_info = self.get_docker_info()
        adguard_stats = self.get_adguard_stats()
        backup_info = self.get_immich_backup_info()

        markdown_msg = f"# 🍓 树莓派状态报告\n> <font color=\"comment\">
{current_time}</font>\n\n"

        # 系统概览
        markdown_msg += f"<font color=\"#3498DB\">--- 系统概览 ---</font>\n"
        markdown_msg += f" <font color=\"#2ECC71\">🌡️ CPU温度</font>: **
{system_info.get('cpu_temp', 'N/A')}**\n"
        markdown_msg += f"- <font color=\"#2ECC71\">⏱️运行时间</font>:
{system_info.get('uptime', 'N/A')}\n"
        markdown_msg += f"- <font color=\"#2ECC71\">🧠内存使用</font>:
{system_info.get('memory', 'N/A')}\n"
        markdown_msg += f"- <font color=\"#2ECC71\">◇磁盘使用</font>:
{system_info.get('disk_usage', 'N/A')}\n"
        markdown_msg += f"- <font color=\"#2ECC71\">🌐主IP地址</font>:
`{system_info.get('ip_address', 'N/A')}`\n\n"

        # Docker服务
        markdown_msg += f"<font color=\"#3498DB\">--- 🐳Docker服务 ---</font>\n"
        if not docker_info:
            markdown_msg += "> <font color=\"comment\">无运行中的容器</font>\n"
        for container in docker_info:
            markdown_msg += f"· `{container['name']}`: <font color=\"
{container['color']}\">{container['status']}</font> \n"

        # AdGuard Home
        markdown_msg += "<font color=\"#3498DB\">--- 🛡️DNS拦截 ---</font>\n"
        if 'error' in adguard_stats['basic']:
            markdown_msg += f"> <font color=\"warning\">⚠️ 获取数据失败</font>:
{adguard_stats['basic']['error']}\n"
        else:
            basic = adguard_stats['basic']
            markdown_msg += self.generate_adguard_card("DNS查询总数", f"
{basic.get('dns_queries', 0):,}", None, "blue")
            markdown_msg += self.generate_adguard_card("已被过滤器拦截", f"
{basic.get('blocked_filter', 0):,}", basic.get('blocked_percentage', 0), "red")
            markdown_msg += self.generate_adguard_card("恶意/钓鱼网站", f"
{basic.get('blocked_safebrowsing', 0):,}", None, "green")
            markdown_msg += self.generate_adguard_card("成人网站拦截", f"
{basic.get('blocked_parental', 0):,}", None, "yellow")

        # Immich 备份
```

```
        status_color = "#2ECC71" if backup_info['status'] == "success" else
"#E74C3C"
        markdown_msg += f"\n<font color=\"#3498DB\">--- 📷 Immich 备份
---</font>\n"
        markdown_msg += f"> <font color=\"info\">最后记录</font>:
{backup_info['last_backup']}\n"
        markdown_msg += f"> <font color=\"info\">状态</font>: <font color=\"
{status_color}\">{backup_info['status'].upper()}</font>\n"

        return markdown_msg

    def run(self):
        """执行监控并发送报告"""
        print("--- 开始收集树莓派状态信息 ---")
        report = self.generate_report()
        print("\n--- 报告预览 ---\n" + report + "\n--- 报告结束 ---\n")

        print("--- 正在发送到企业微信 ---")
        if self.wechat_key == "YOUR_WEBHOOK_KEY":
             print("警告:尚未配置企业微信KEY, 已跳过发送。")
             return

        success = self.send_to_wechat(report)
        if success:
            print("✅ 消息发送成功!")
        else:
            print("❌ 消息发送失败!")

if __name__ == "__main__":
    monitor = PiStatusMonitor()
    monitor.run()
```

1.5 设置定时任务 (Cron Job)

使用 `cron` 服务实现脚本的每日自动执行。

编辑定时任务

```
crontab -e
```

添加任务指令 在文件末尾添加以下一行，设定脚本在每天早上6点整运行。

```
0 6 * * * /home/pi/scripts/pi-monitor/venv/bin/python3 /home/pi/scripts/pi-
monitor/pi_monitor.py >> /home/pi/scripts/pi-monitor/monitor.log 2>&1
```

- `0 6 * * *`：时间设定，表示每天6:00。
- `/home/pi/.../python3`：**必须**使用虚拟环境中的Python解释器完整路径。
- `/home/pi/.../pi_monitor.py`：脚本的完整路径。
- `>> ... 2>&1`：将所有输出（包括错误）追加到日志文件 `monitor.log`，便于排错。

保存并验证

- 保存退出编辑器（`nano` 中为 `Ctrl+X` -> `Y` -> `Enter`）。

- 使用 `crontab -l` 命令检查任务是否已成功添加。