树莓派

- 社区：https://shumeipai.nxez.com/hot-explorer
- 下载：https://www.raspberrypi.com/software/operating-systems

安装烧录镜像

树莓派4 安装 raspios-buster-arm64 位系统

01 准备资源

下载系统https://www.raspberrypi.com/software/operating-systems/下载当前最新64位版本，我下载的是桌面版本

也可考虑下载32为系统。
https://downloads.raspberrypi.org/raspios_full_armhf/images/raspios_full_armhf-2020-12-04/2020-12-02-raspios-buster-armhf-full.zip

下载烧录工具从 https://www.raspberrypi.org/software/ 下载烧录软件 。

02烧录系统

1. 打开软件选择use custom，选择我自己下载好的.img文件
2. 打开读取树莓派tf的sd卡，插入电脑
3. 在烧录的磁盘中加入一个空的文件名称为`ssh`的文件，支持ssh连接。[-可以不加-]添加一个文件名为`wpa_supplicant.conf`,里面配置如下，支持无线连接。配置好ssh连接和wifi连接，即可加电启动系统。

```shell
        #wpa_supplicant.conf 内容如下：
        # ssid指定wifi名称，psk指定wifi密码
        # 系统启动后，会把这些配置信息写入/etc/wpa_supplicant/wpa_supplicant.conf中
        # priority是优先级，数字越大越优先连接
        country=CN

ctrl_interface=DIR=/var/run/wpa_supplicant
```

```
GROUP=netdev
        update_config=1
        network={
          ssid="zfj23"
          psk="xxxxx33ff"
          key_mgmt=WPA-PSK
          priority=22
        }
        network={
          ssid="xxx"
          psk="xx"
          key_mgmt=WPA-PSK
          priority=2
        }
```

03 基础设置

- 替换apt镜像
  - https://mirrors4.tuna.tsinghua.edu.cn/help/raspbian/
- 固定ip
  - sudo vim /etc/dhcpcd.conf
    - 搜索static IP configuration
  - sudo passwd root 修改root密码
  - service networking restart 重启网络
  - apt-get install dnsutils 检查dns
  - vim /etc/resolv.conf 修改dns
  - sudo /etc/init.d/nscd restart 刷新dns缓存
- 免密钥ssh
  - ssh-keygen -t rsa
  - 复制id_rsa.pub到服务器
  - 本地新建config文件
- 打开vnc在桌面网络设置中打开vnc。或者使用命令sudo raspi-config

树莓派4b
防火墙(我只在LAN，暂时关闭)

ufw是一个主机端的iptables类防火墙配置工具，比较容易上手。如果你有一台暴露在外网的树莓派，则可通过这个简单的配置提升安全性。

安装方法

```
1 sudo apt-get install ufw
```

当然，这是有图形界面的(比较简陋)，在新立得里搜索gufw试试……

使用方法

****1 、启用****

1 sudo ufw enable
2 sudo ufw default deny

作用：开启了防火墙并随系统启动同时关闭所有外部对本机的访问（本机访问外部正常）。

2 、关闭

sudo ufw disable

3、查看防火墙状态

1sudo ufw status

4、开启/禁用相应端口或服务举例

sudo ufw allow 80 允许外部访问80端口
sudo ufw delete allow 80 禁止外部访问80 端口
sudo ufw allow from 192.168.1.1 允许此IP访问所有的本机端口
sudo ufw deny smtp 禁止外部访问smtp服务
sudo ufw delete allow smtp 删除上面建立的某条规则
ufw deny proto tcp from 10.0.0.0/8 to 192.168.0.1 port 要拒绝所有的流量从TCP的10.0.0.0/8 到端口22的地址192.168.0.1
可以允许所有RFC1918网络（局域网/无线局域网的）访问这个主机（/8,/16,/12是一种网络分级）：
sudo ufw allow from 10.0.0.0/8
sudo ufw allow from 172.16.0.0/12
sud}o ufw allow from 192.168.0.0/16

安装filerun

https://docs.filerun.com/docker-arm
mariadb

- mysql-root-password
- mysql-usename
- mysql -password
- mysql-db
- 挂载的目录
  - /home/pi/FileRun/mariadb:/var/lib/mysql
  filerun
- fr-db-name
- fr-db-user
- fr-db-password
  - /home/pi/FileRun/mariadb
- 挂载目录
  - /home/pi/FileRun/filerun_web/html:var/www/html
  - /home/pi/挂载目录/user_files:/user-files
- mysql的连接信息配置目录
  - /var/www/html/system/data/autoconfig.php
- 配置php
  - 参考官方：https://www.php.net/manual/en/ini.core.php#ini.upload-max-filesize
  - 挂载一下配置文件
    - /home/pi/FileRun/php:/usr/local/etc/php

不同的容器之间的通讯还可以使用这种方式

新建网络

下面先创建一个新的 Docker 网络。

```
$ docker network create -d bridge my-net
```

-d 参数指定 Docker 网络类型，有 `bridge`、`overlay`。其中 `overlay` 网络类型用于 Swarm mode，在本小节中你可以忽略它。

连接容器

运行一个容器并连接到新建的 `my-net` 网络

```
$ docker run -it --rm --name busybox1 --network my-net busybox sh
```

打开新的终端，再运行一个容器并加入到 `my-net` 网络

```
$ docker run -it --rm --name busybox2 --network my-net busybox sh
```

再打开一个新的终端查看容器信息

```
$ docker container ls

CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
b47060aca56b        busybox             "sh"                11
minutes ago      Up 11 minutes
busybox2
8720575823ec        busybox             "sh"                16
minutes ago      Up 16 minutes
busybox1
```

下面通过 `ping` 来证明 `busybox1` 容器和 `busybox2` 容器建立了互联关系。在 `busybox1` 容器输入以下命令

```
/ # ping busybox2
PING busybox2 (172.19.0.3): 56 data bytes
64 bytes from 172.19.0.3: seq=0 ttl=64 time=0.072 ms
64 bytes from 172.19.0.3: seq=1 ttl=64 time=0.118 ms
```

用 ping 来测试连接 `busybox2` 容器，它会解析成 `172.19.0.3`。同理在 `busybox2` 容器执行 `ping busybox1`，

```
/ # ping busybox1
PING busybox1 (172.19.0.2): 56 data bytes
64 bytes from 172.19.0.2: seq=0 ttl=64 time=0.064 ms
64 bytes from 172.19.0.2: seq=1 ttl=64 time=0.143 ms
```

这样，`busybox1` 容器和 `busybox2` 容器建立了互联关系。

mysql问题

https://devopsbuild.com/docker-host-is-not-allowed-to-connect-to-this-mysql-server/

设置开机自动挂载

要实现开机自动挂载U盘，我们需要将U盘的设备信息写入到`/etc/fstab`文件中

1. 查看硬盘UUID信息

```
blkid
```

如上面列表中，我们看到U盘设备位置`/dev/sda1`的UUID是`3C646D4A646D07CA`，TYPE是"ntfs"
我们记录下信息准备后面使用：

```
/dev/sda1: UUID="224872DC4872AE63" TYPE="ntfs"
PARTUUID="d4202185-01"
```

2. 挂载信息写入配置文件

查看下`cat /etc/fstab`文件的内容

- 将以下信息，添加到/etc/fstab文件末尾

```
UUID=224872DC4872AE63 /home/pi/Newsmy/ ntfs defaults 0
0
```

上面的具体内容含义：

- 要挂载的分区设备号 UUID=3C646D4A646D07CA
- 挂载点 /home/ubuntu/udisk
- 文件系统类型 ntfs
- 挂载选项 defaults
- 是否备份 0
- 是否检测 0

使用vim编辑`/etc/fstab`文件，追加上面的内容

```
sudo vim /etc/fstab
```

3. 使配置生效

```
sudo mount -a
```

4. 查看挂载情况

```
df -h
```

5.重启服务查看自动挂载效果

```
#重启
sudo reboot
```

使用`sudo lsblk`查看效果

树莓派部署Samba服务

Samba是一个能让Linux系统应用Microsoft网络通讯协议的软件，而SMB是Server Message Block的缩写，即为服务器消息块，SMB主要是作为Microsoft的网络通讯协议，后来Samba将SMB通信协议应用到了Linux系统上，就形成了现在的Samba软件。后来微软又把 SMB 改名为 CIFS（Common Internet File System），即公共 Internet 文件系统，并且加入了许多新的功能，这样一来，使得Samba具有了更强大的功能。

Samba最大的功能就是可以用于Linux与windows系统直接的文件共享和打印共享，Samba既可以用于windows与Linux之间的文件共享，也可以用于Linux与Linux之间的资源共享，由于NFS(网络文件系统）可以很好的完成Linux与Linux之间的数据共享，因而 Samba较多的用在了Linux与windows之间的数据共享上面。

使用samba和windows共享文件时候，请确保windows的NetBIOS(就是网上邻居功能)开启了。

Samba组成

一个samba服务器实际上包含了两个服务器程序：smbd和nmbd。

Smbd是samba的核心。它负责建立对话进程、验证用户身份、提供对文件系统和打印机的访问机制，只有smb服务启动，才能实现文件的共享。【TCP 445】

Nmbd实现了“Network Brower”（网络浏览服务器）的功能，实现NETBIOS(本地名称解析)功能【UDP137/138TCP 139】，如果该服务没有启动，则客户端不能通过Linux系统共享的工作组名称访问共享文件，而只能通过IP地址来访问共享的文件。

Samba还包含了一些实用工具。Smbclient是一个SMB客户工具，有shell-based用户界面并同FTP有些类似。应用它可以复制其它的SMB服务器资源，还可以访问其它SMB服务器提供的打印机资源。

1. 安装samba服务

```
sudo apt-get install samba
```

2. 创建账户与设置密码

这里把pi为用samba的登录用户，并设置密码

```
# 创建samba配置的密码文件
sudo touch /etc/samba/smbpasswd

# 添加smb账户
sudo smbpasswd -a pi
```

执行`smbpasswd`命令后会提示输入samba的账户密码，这个密码后面访问smb服务会用到，我这里使用ubuntu这个默认root用户所以不用新建，用户需要在系统中存在，没有则先用useradd创建。

3. 设置samba的配置文件

需要在smaba配置中指定相关的smb共享文件夹

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

将如下配置添加到smb.conf最后面

```
# 树莓派本磁盘配置，挂载的是test目录
log file = /var/log/samba/log.%m
max log size = 50
```

```
[local]
    comment = local
    path = /home/pi/test
    writable = yes
    browseable = yes
    valid user = pi
    available = yes
    create mask = 0777
    directory mask = 0777
    public = yes
    write list = root,pi

#  u盘挂载的配置，目录是挂载目录
[udisk]
    comment = udisk
    path = /home/pi/udisk
    writable = yes
    browseable = yes
    valid user = ubuntu
    available = yes
    create mask = 0777
    directory mask = 0777
    public = yes
    write list = root,pi
```

上面我添加了两个共享目录配置，一个是local共享的是树莓派本地的文件夹`/home/pi/test`，一个是udisk共享的是U盘挂载的`/home/pi/udisk`文件夹，大家可以根据自己情况设置一个或是多个

各个参数具体含义如下：

- udisk：分享名称
- comment：备注描述
- path：共享文件夹目录
- writable：是否可写入，不能写入就不能创建文件夹
- browseable：是否可以访问浏览
- valid user：允许哪个用户访问，这里需要按照指定的账户访问samba服务

- guest ok = yes 指定是否允许guest帐号访问
- log file: 设置SambaServer日志文件的存储位置以及日志文件名称。在文件名后加个宏%m（主机名），表示对每台访问Samba Server的机器都单独记录一个日志文件。如果pc1、pc2访问过Samba Server，就会在/var/log/samba目录下留下log.pc1和log.pc2两个日志文件。
- max log: 设置SambaServer日志文件的最大容量，单位为kB，0代表不限制。
网络连接配置：

```
hosts allow = 127. 192.168.1. 192.168.10.1
说明：表示允许连接到SambaServer的客户端，多个参数以空格隔开。可以用一个IP表示，也可以用一个网段表示。hosts deny 与hosts allow 刚好相反。
例如：hostsallow=172.17.2.EXCEPT172.17.2.50
表示容许来自172.17.2.*的主机连接，但排除172.17.2.50
hosts allow=172.17.2.0/255.255.0.0
表示容许来自172.17.2.0/255.255.0.0子网中的所有主机连接
hosts allow=M1, M2
表示容许来自M1和M2两台计算机连接
hosts allow=@pega
表示容许来自pega网域的所有计算机连接
```

挂载目录：

```
sudo mount -t ntfs /dev/sda1 /home/pi/Newsmy

df -hP
/dev/sda1       932G  303G  629G   33% /home/pi/Newsmy
```

4. 重启服务使配置生效

```
# 重启服务
sudo /etc/init.d/smbd restart

# 查看服务状态
sudo /etc/init.d/smbd status
```

也可以使用下面的命令效果相同

```
# 重启服务
sudo service smbd restart

# 查看服务状态
sudo service smbd status
```

测试连接：

```
sudo apt install libsmbclient libsmbclient-dev
sudo apt install smbclient
sudo smbclient -L //192.168.xx.xx//具体目录 -U 用户名

-L
显示服务器端所分享出来的所有资源
-U
指定用户名称
```

5. 设置开机自动启动

```
# 开机自启动
sudo systemctl enable smbd
```

6. 连接访问smb共享服务

在我们重启smb服务后，我们去访问smb的共享服务，网上有不少教程大家可以去搜索一下

树莓派监控器

https://zhuanlan.zhihu.com/p/430399915
https://github.com/nxez/pi-dashboard

homeAssistant

- 安装

    - 树莓派安装：https://www.home-assistant.io/installation/raspberrypi
    - https://www.home-assistant.io/installation/
- 不同的平台安装所具有的功能不一样，所以的功能如下(https://post.smzdm.com/p/a3dvvwld/):
    - Automations 是自动化，这是 HA 的灵魂；
    - Lovelace 是手机或者浏览器访问 HA 看到的前端界面，可以当成皮肤；
    - Blueprints 是来自社区的自动化和场景模板，为了降低自动化编写难度而生，通过蓝图可以快速创建家庭自动化实例；
    - Supervisor，管理员，是用来管理和更新 Home Assistant Core，管理操作系统，管理 docker（HA 和加载项），以及管理前三者之前的 API 和互动。
    - Integrations 是集成/组件，是指把各种智能家居和服务接入到 HA 这个平台的方法和代码的专有名称，可以理解为官方插件；
    - Add-on 是加载项商店，可以下载 HA 社区或者第三方开发的许多插件。Integration 是 HomeAssistant 的组成部分，而 Add-on 不是。HomeAssistant 通过配置加载 Integration 程序，通过 Supervisor 管理 Add-on 的安装/配置/启停。
- 体验网站 https://demo.home-assistant.io/#/lovelace/0

安装过程

docker container安装模式

```
version: '3'
services:
  homeassistant:
    container_name: homeassistant
    image: "ghcr.io/home-assistant/raspberrypi4-
homeassistant:stable"
    volumes:
      - /home/pi/homeassistant/config:/config
      - /etc/localtime:/etc/localtime:ro
    restart: unless-stopped
    privileged: true
```

```
    network_mode: host
```

安装HACS&Xiaomi Miot

1.首先在映射出来的config文件下，新建`mkdir custom_components`

2.Downloading HACS：

Desc:HACS gives you a powerful UI to handle downloads of all your custom needs.

```
wget
https://github.com/hacs/integration/releases/latest/download/hacs.zip
```

3.Downloading hass-xiaomi-miot：

Desc:Xiaomi Miot For HomeAssistant
https://github.com/al-one/hass-xiaomi-miot/releases

4.解决github连接的问题

HACS经常下载插件失败，因为GitHub访问的原因。
解决思路

1. 修改supervisor 的容器的Hosts，或者修改软路由的，替换解析。
2. `sudo docker exec -it <容器ID> bash`
   进到HA容器的终端里面，vi /etc/hosts 修改容器的hosts文件，按i进入编辑模式，在最后面添加，为什么要添加因为github过不去(HACS需要和github绑定之后使用)！而且添加多次会被限制！！
3. 手动安装，每个库都有安装说明。安装路径在\config\custom_components下。
4. 修改HACS，下载代码，替换URL为CDN加速的url。

查看HACS代码，在
base.py中，增加方法,python要注意空格和缩进

```
    def replace_url(self, url):
        """替换原ulr为  cdn加速后的url"""
        # https: // cdn.jsdelivr.net / gh / user /
repo @ version / file
        if not
url.startswith('https://raw.githubusercontent.com/'):
            return url

        url =
url.replace("https://raw.githubusercontent.com/", "")
        splits = url.split("/")
        user = splits[0]
        repo = splits[1]
        version = splits[2]
        file = "/".join(splits[3:])
        new_url =
f"https://cdn.jsdelivr.net/gh/{user}/{repo}@{version}/
{file}"
        return new_url
```

然后替换url

```
  url = self.replace_url(url) # 替换url为cdn加速
```

修改为这样:

```
    async def async_download_file(self, url: str, *,
headers: dict | None = None) -> bytes | None:
        """Download files, and return the content."""
        if url is None:
            return None

        url = self.replace_url(url)

        self.log.debug("Downloading %s", url)
        timeouts = 0
```

```
        while timeouts < 5:
            try:
                request = await self.session.get(
                    url=url,
                    timeout=ClientTimeout(total=60),
                    headers=headers,
                )

                # Make sure that we got a valid result
                if request.status == 200:
                    return await request.read()

                raise HacsException(
                    f"Got status code {request.status}
when trying to download {url}"
                )
            except asyncio.TimeoutError:
                self.log.warning(
                    "A timeout of 60! seconds was
encountered while downloading %s, "
                    "using over 60 seconds to download
a single file is not normal. "
                    "This is not a problem with HACS
but how your host communicates with GitHub. "
                    "Retrying up to 5 times to
mask/hide your host/network problems to "
                    "stop the flow of issues opened
about it. "
                    "Tries left %s",
                    url,
                    (4 - timeouts),
                )
                timeouts += 1
                await asyncio.sleep(1)
                continue
```

```
            except BaseException as exception:  # lgtm
[py/catch-base-exception] pylint: disable=broad-except
                self.log.exception("Download failed -
%s", exception)

            return None
```

5.打开honmeasistant页面，添加集成

```
使用方法：
登陆HA > 配置 > 设备与服务 > 集成 > 添加集成 > 搜索“Xiaomi
Miot Auto”（如果搜索不到请清除你的浏览器缓存）
账号集成（Add devices using Mi Account):
自v0.4.4版本开始，插件新增支持账号集成时选择连接设备的模式
自动模式：（推荐）插件定期更新支持本地miot协议的设备，并自动将筛选设
备中符合条件的型号使用本地连接
本地模式：集成配置所筛选的设备都将使用本地连接，如勾选了不支持本地
miot协议的设备将不可用
云端模式：集成配置所筛选的设备都将使用云端连接，建议蓝牙、ZigBee设备
使用
```

监控

docker加速

- https://www.runoob.com/docker/docker-mirror-acceleration.html

安装cadvisor

参考：https://github.com/ZCube/cadvisor-docker
这个是arm64，官方不支持，所以自己构建

```
sudo docker run \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:ro \
```

```
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  --volume=/dev/disk/:/dev/disk:ro \
  --publish=8080:8080 \
  --detach=true \
  --name=cadvisor \
  --restart unless-stopped \
  zcube/cadvisor:latest
```

http://192.168.1.25:8080/containers

安装prometheus

启动:

```
sudo docker run  -d \
      --name prometheus --restart unless-stopped \
    -p 9090:9090 \
    -v
/home/pi/prometheus/prometheus.yml:/etc/prometheus/pro
metheus.yml \
    prom/prometheus
```

配置:

```
# 配置参考
# my global config
global:
  scrape_interval:     15s # Set the scrape interval
to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15
seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
```

```
  - static_configs:
    - targets:
      # - alertmanager:9093
# Load rules once and periodically evaluate them
according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"
# A scrape configuration containing exactly one
endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>`
to any timeseries scraped from this config.
  - job_name: 'prometheus'
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
    - targets: ['localhost:9090']

global:
  scrape_interval: 60s
  evaluation_interval: 60s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['192.168.1.25:9090']
  - job_name: "Pi_cadvisor"
    static_configs:
      - targets: ['192.168.1.25:8080']
  - job_name: "Docker_grafana"
    static_configs:
      - targets: ['192.168.1.25:3000']
```

http://192.168.1.25:9090/targets

安装portainer

sudo docker run -d -p 9000:9000 --name portainer --restart always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer

http://192.168.1.25:9000/

忘记密码

1.停止运行的portainerdocker stop '这里写你的portainer id 或者 name
2.下载帮助镜像docker pull portainer/helper-reset-password3.运行重置命令(前提是你安装的时候是根据官网步骤来的)官网地址：
https://documentation.portainer.io/v2.0/deploy/ceinstalldocker/
如果创建的时候修改了挂载的位置，这边路径也是需要修改的
docker run --rm -v portainer_data:/data portainer/helper-reset-password
4.执行完上一步命令，系统会随机一个密码
5.重启Portainer容器docker start '你的portainer id 或者 name

安装grafana

```
sudo docker run -d --name grafana --restart always  -p 3000:3000 grafana/grafana
```

安装exporter

- 社区提供的
- 例如，Node Exporter(https://github.com/prometheus/node_exporter)、MySQL Exporter、Fluentd Exporter。
- 详情：https://prometheus.io/docs/instrumenting/exporters/
- 用户自定义的
- 用户还可以基于Prometheus提供的Client Library创建自己的Exporter程序。
- Go（https://github.com/prometheus/client_golang）
- Java（https://github.com/prometheus/client_java）
- Python（https://github.com/prometheus/client_python）；

安装https://github.com/prometheus/node_exporter

sudo netstat -lntup|grep exporter

tcp6 0 0 :::9100 :::* LISTEN 33357/node_exporte

http://192.168.1.25:9100/metrics

傻妞
安装傻妞

```
一键命令
bash <(curl -sSL http://app.imdraw.com/install.sh)
docker版
安装:
docker run \
    -itd \
    --name sillygirl \
    --restart always \
   -p 8080:8080 \
    -v  "$(pwd)"/sillyGirl:/etc/sillyGirl \
    mzzsfy/sillygirl:latest
交互:
docker attach sillygirl
详情:
https://hub.docker.com/r/mzzsfy/sillygirl
node-onebot:
https://hub.docker.com/r/mzzsfy/node-onebot
cqhttp:
https://hub.docker.com/r/mzzsfy/go-cqhttp
```

安装node-onebot

https://hub.docker.com/r/mzzsfy/node-onebot