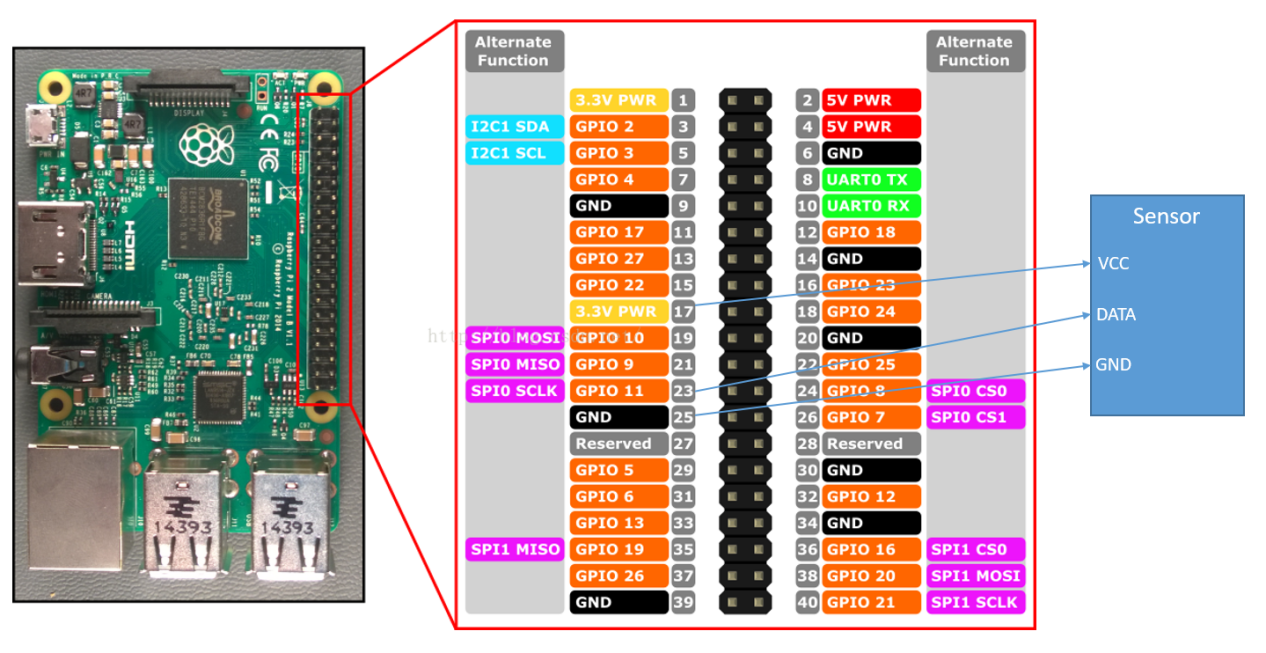
## 树莓派系统环境配置及kubeedge业务部署指南

### 1.硬件配置

所需硬件：树莓派4B、杜邦线、温度传感器、SD卡、读卡器





树莓派配置：

Raspberry PI 4B

CPU(s): 4

Memory: 4G

Disk: 16G

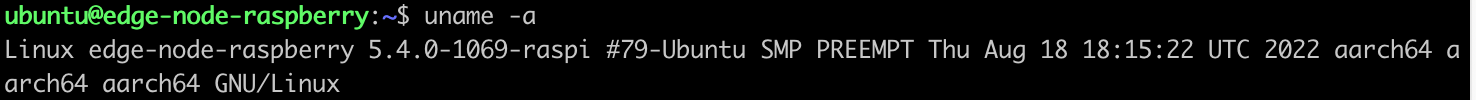
Vendor ID: ARM

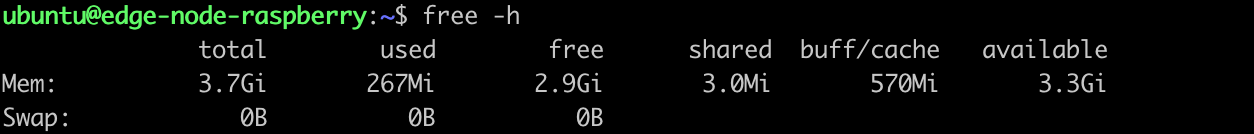
Model name: Cortex-A72

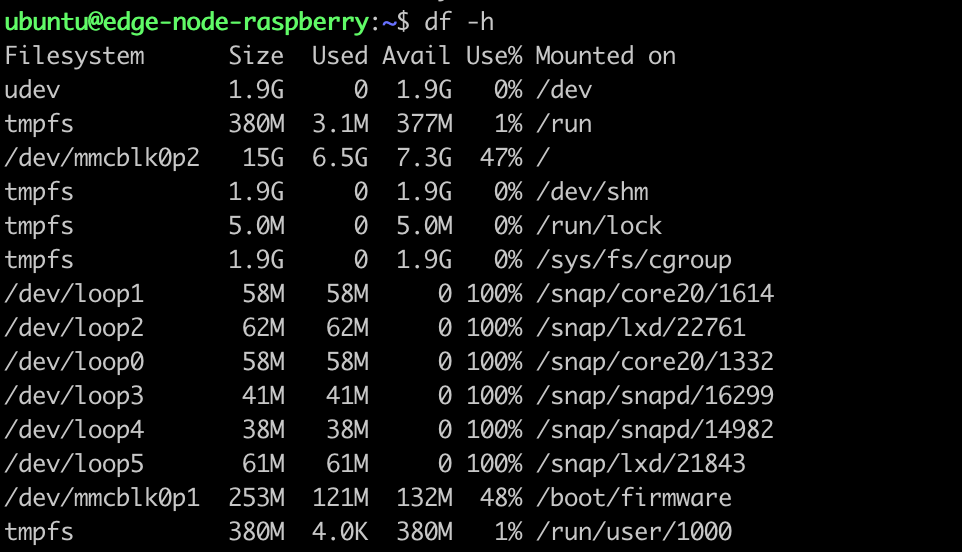
CPU max MHz: 1500.0000

CPU min MHz: 600.0000

温度传感器: DHT11







### 2.软件环境

**2.1. 操作系统**

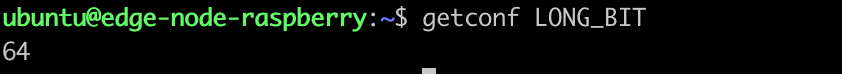
本实验为树莓派采用的操作系统是Ubuntu20.04.4 LTS（Liunx 5.4.0-raspi aarch64）。树莓派于2022年官方支持安装64位操作系统和软件，并且树莓派4B系列拥有支持64位的硬件架构，我们采用的也是最新支持的64位ubuntu系统。

通过树莓派官网提供的系统烧录软件在SD卡烧录系统。



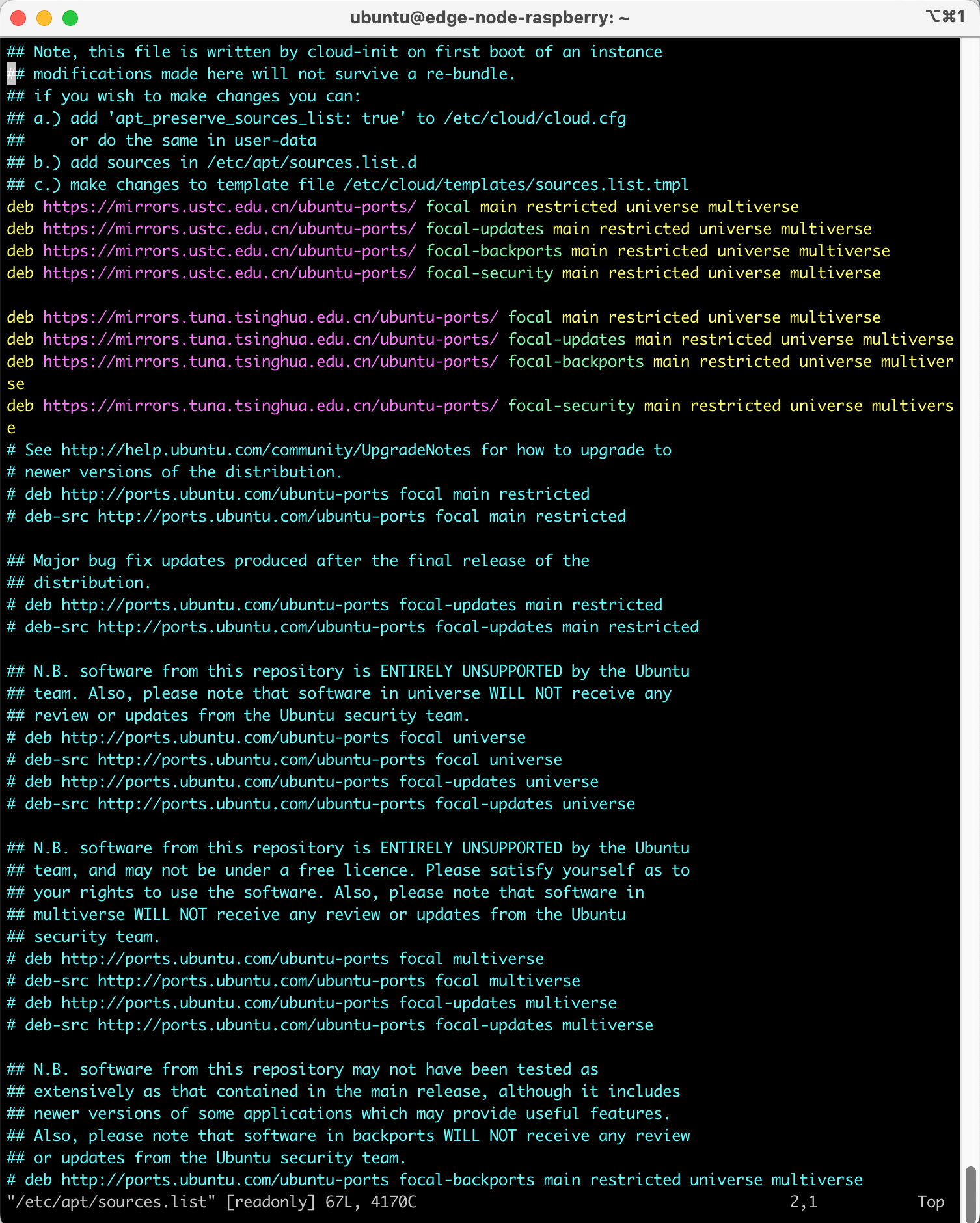
通过路由器找到树莓派所在的内网ip并进行设置





首先修改软件镜像源，删除原先的镜像源，添加中科大和清华镜像源：





Ubuntu20.04自动休眠禁止

sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target

根据情况需要配置VNC

首先安装桌面环境：

sudo apt update

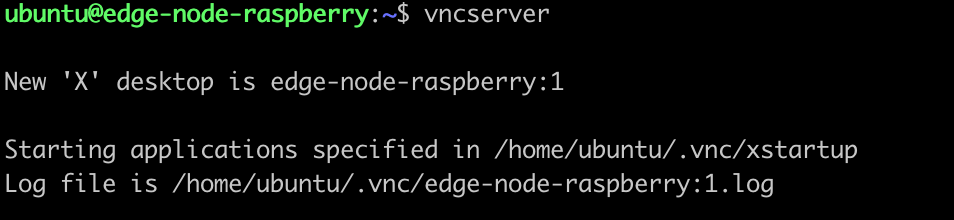
sudo apt upgrade

sudo apt install xfce4 xfce4-goodies

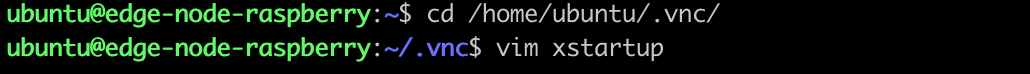
接下来安装VNC

sudo apt install tightvncserver

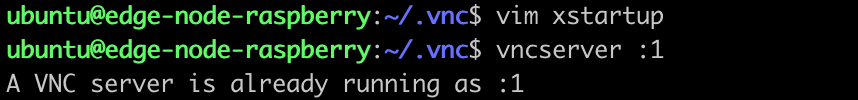
启动VNC，第一次配置时需要设置密码，默认端口1

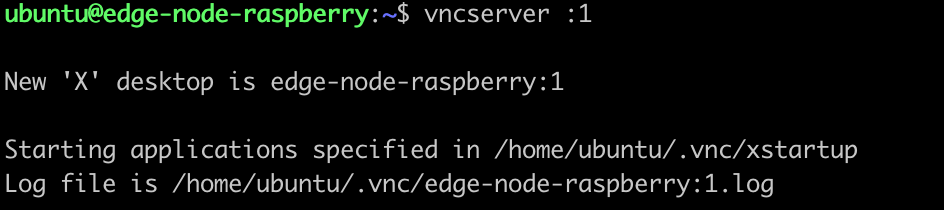


修改vnc配置文件









**安装docker**

sudo apt update

sudo apt install docker.io

sudo systemctl start docker

sudo systemctl enable docker

**为docker配置镜像加速**

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["https://0y4nh6yd.mirror.aliyuncs.com"]

}

EOF

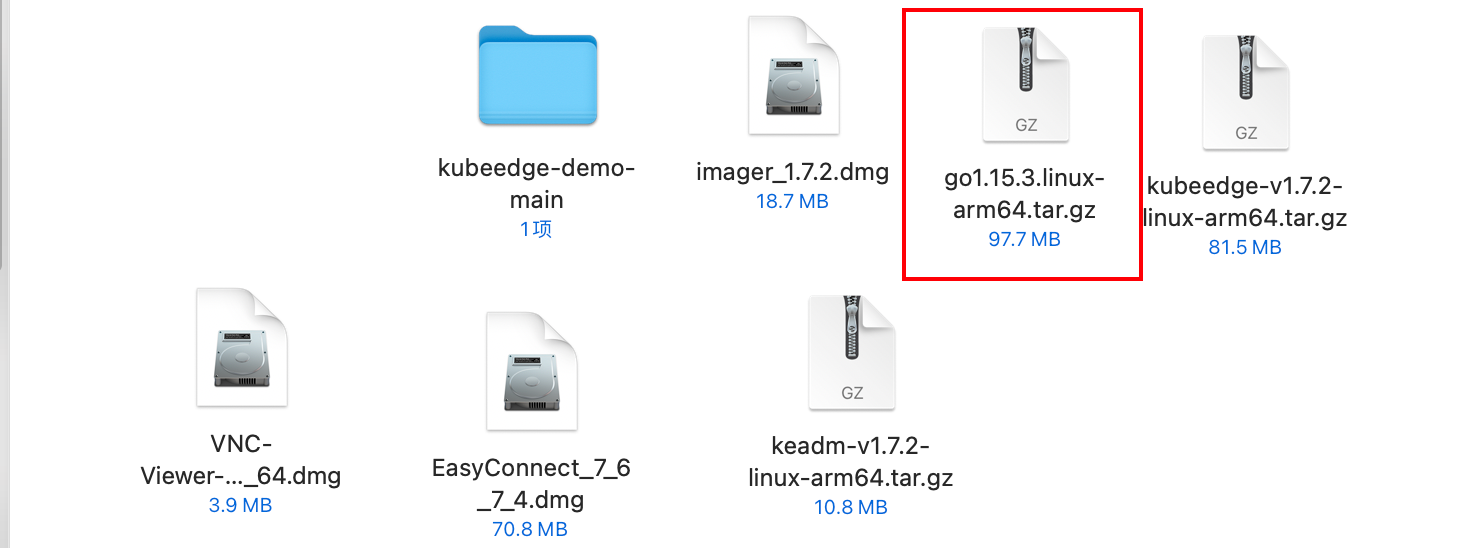
sudo systemctl daemon-reload

sudo systemctl restart docker



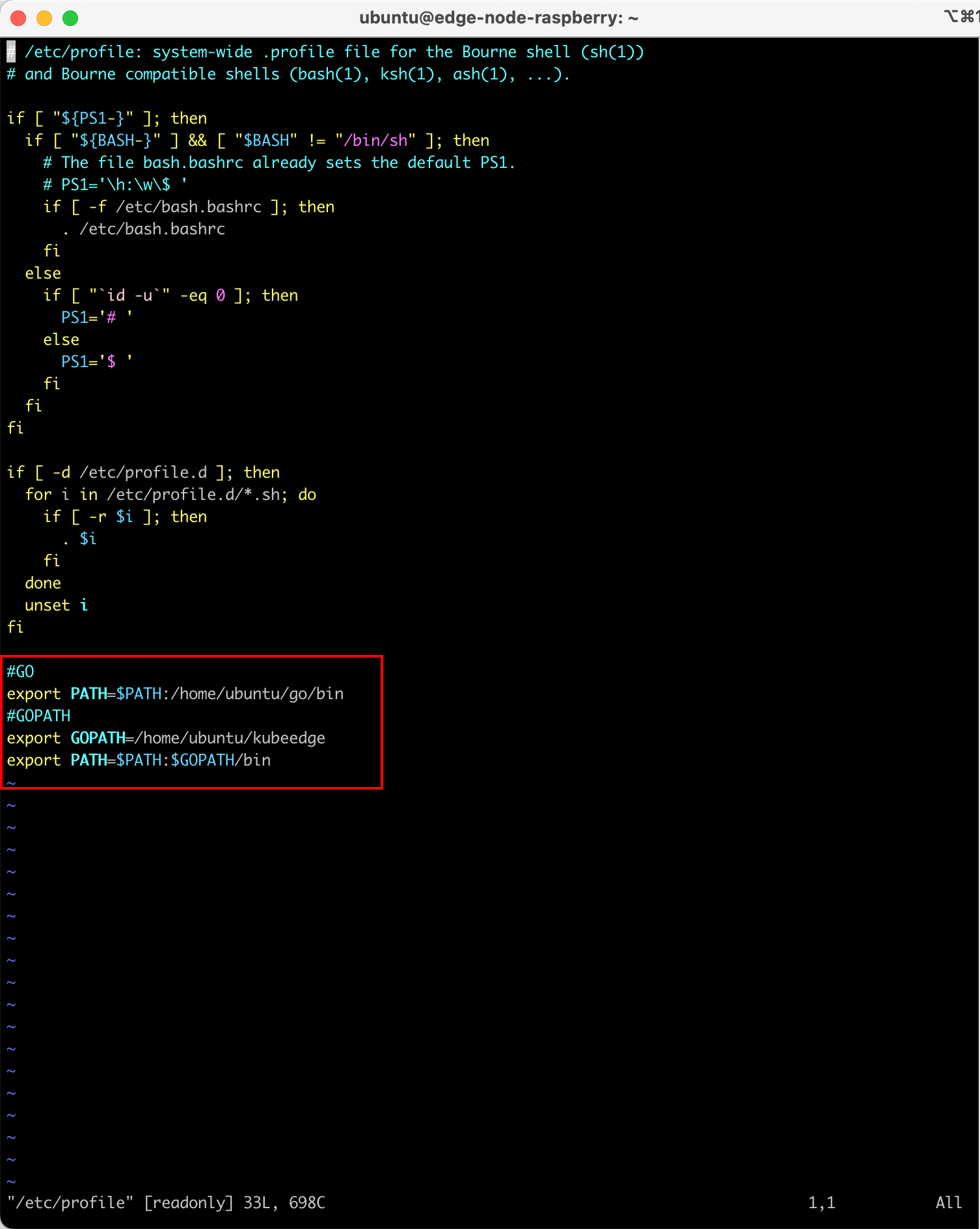
安装go语言环境

官网下载tar软件包通过sftp传输到树莓派上安装



tar -zxvf go1.15.3.linux-arm64.tar.gz

#在/etc/profile里加入环境变量，以便一次部署失败不用再次source

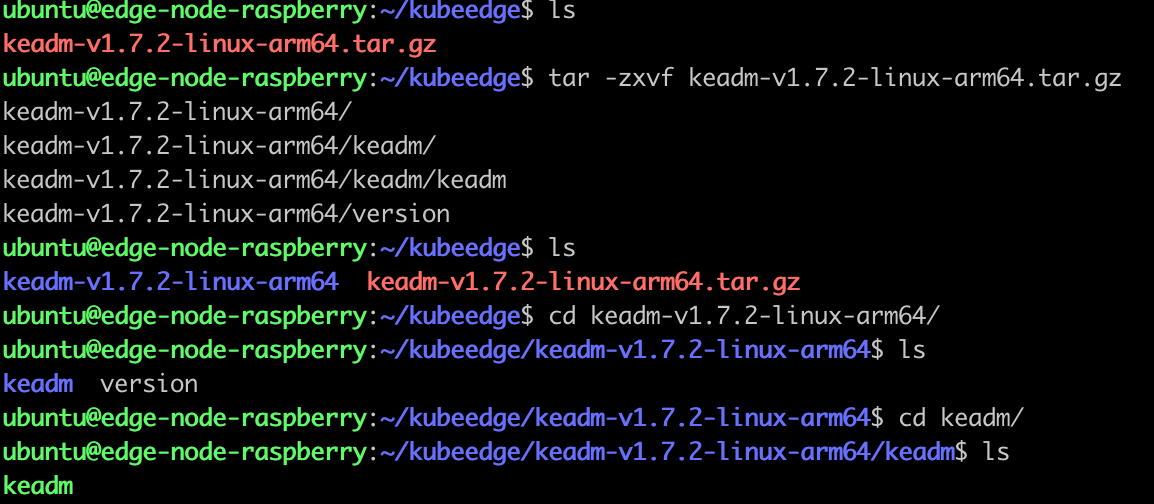


**2.2. 安装kubeedge**

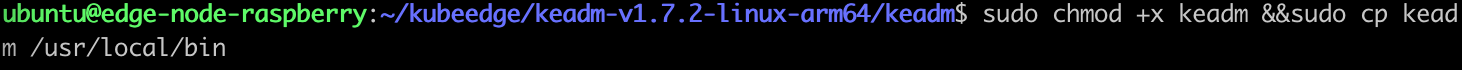
**1）keadm**

由于kubeedge版本更新，现在不再需要手动自己编译需要的可执行文件，在github下载后的压缩包内存在可执行文件。

树莓派边端不需要自己手动编译edgecore，只需要安装keadm进行管理，在加入云端时自动拉取edgecore并安装运行。

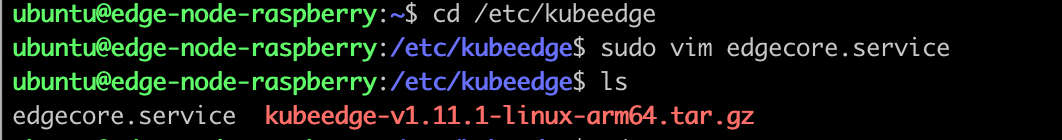


提高keadm权限并设置系统变量里的可执行文件

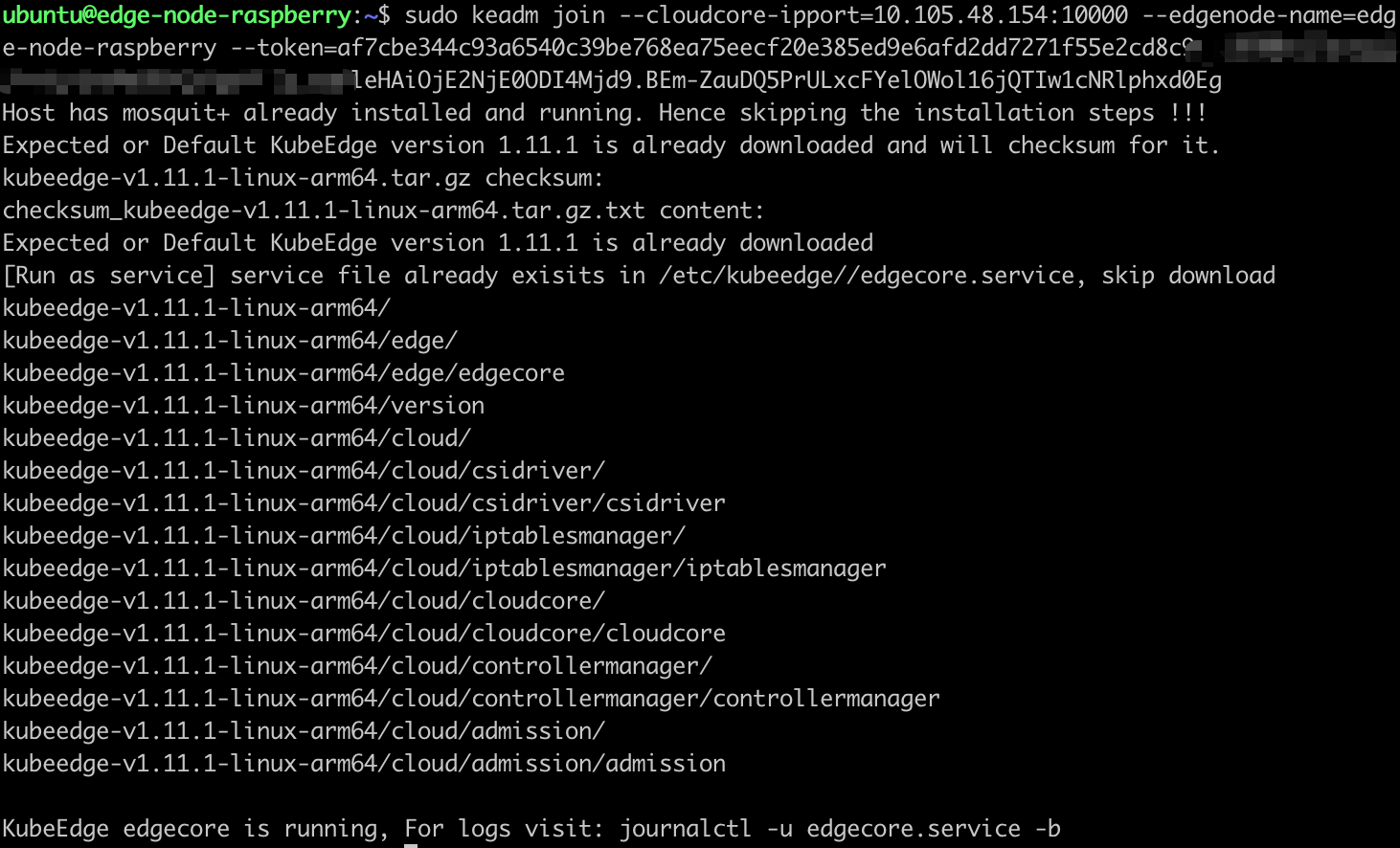


**2） 加入云端并安装运行edgecore**

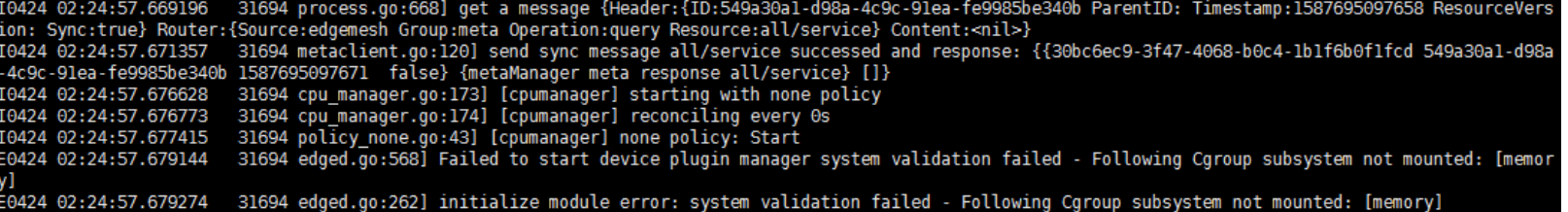
加入云端，但是因为github被墙了所以edgecore.service文件无法下载。解决方案自己创建该文件并修改其中的内容为网页中的内容。



再次用keadm申请加入云端。



使用systemctl status edgecore.service命令查看服务日志会发现服务没有跑起来，查看日志发现在启动初始化过程中失败了，原因是树莓派的Ubuntu20系统设置存在问题，这个问题广泛出现在linux内核版本5以上的树莓派系统中。



日志显示的是内存资源的cgroups机制没有开启。查看可用的cgroup:

cat /proc/cgroups

#subsys\_name hierarchy num\_cgroups enabled

cpuset 3 1 1

cpu 5 38 1

cpuacct 5 38 1

blkio 6 38 1

memory 0 53 0

devices 8 38 1

freezer 7 1 1

net\_cls 4 1 1

pids 2 43 1

通过修改启动文件增加cgroup的内存资源。



**3） 运行业务程序**

在云服务器主节点查询加入的节点情况

****

在树莓派上构建**计数器**应用程序，并在主节点创建设备实例。由于树莓派和云服务器架构不同，所以选择云端和树莓派分别本地编译，云端Ubuntu节点为x86架构，边缘raspberrypi节点为arm64架构，所以导致镜像在异构节点上无法创建容器。直接在raspberrypi节点创建镜像，并保证镜像名称和tag与deployment.yaml文件中一致，否则创建的pod将无法识别镜像，重新创建镜像后成功拉起容器，问题解决。

**构建本地镜像**

由于树莓派是arm64架构，所以我使用macbook m1芯片版进行镜像制作，并将制作好的镜像打包传给树莓派来减少中间编译过程的时间。

$ cd $GOPATH/src/github.com/kubeedge/examples/kubeedge-counter-demo/counter-mapper

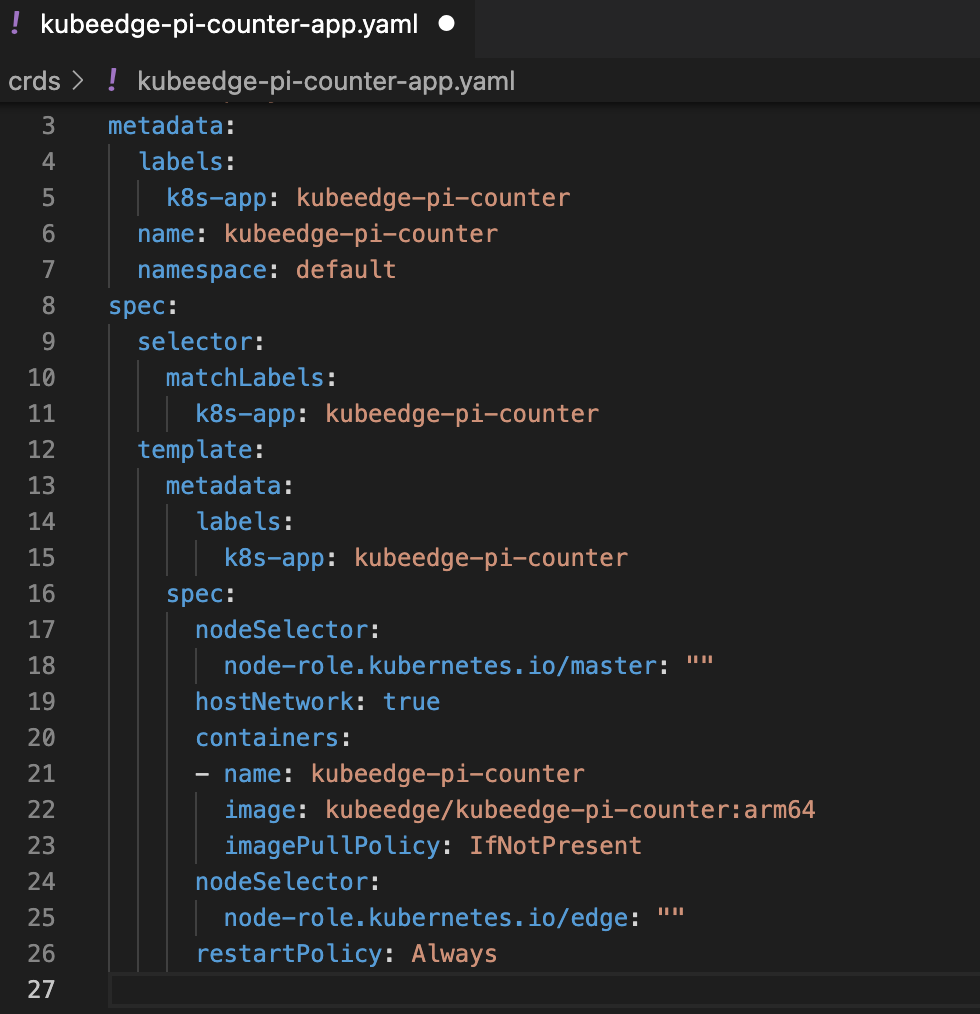
$ make

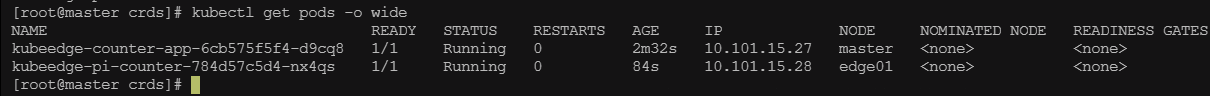
$ make docker

$ cd $GOPATH/src/github.com/kubeedge/examples/kubeedge-counter-demo/crds

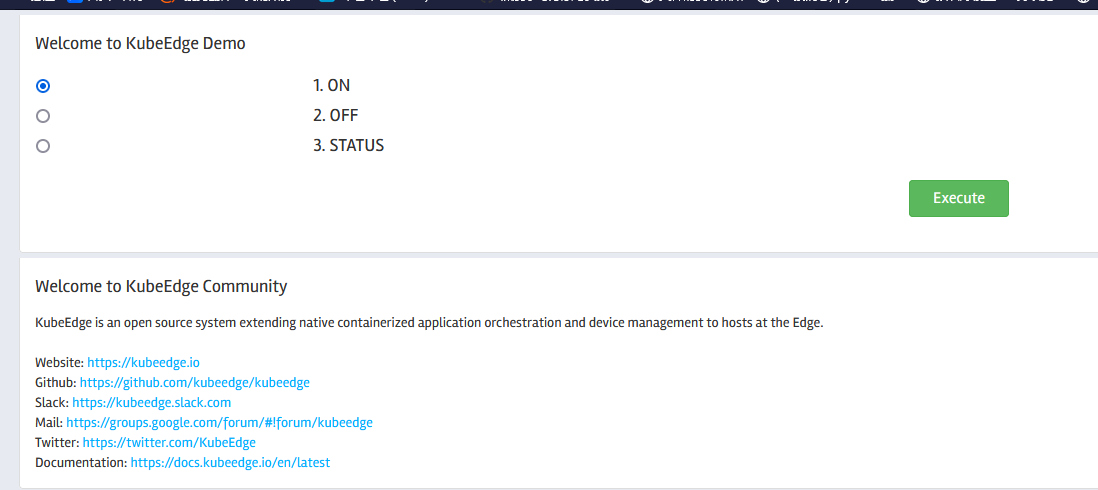
**创建实例**

$ kubectl create -f kubeedge-pi-counter-app.yaml

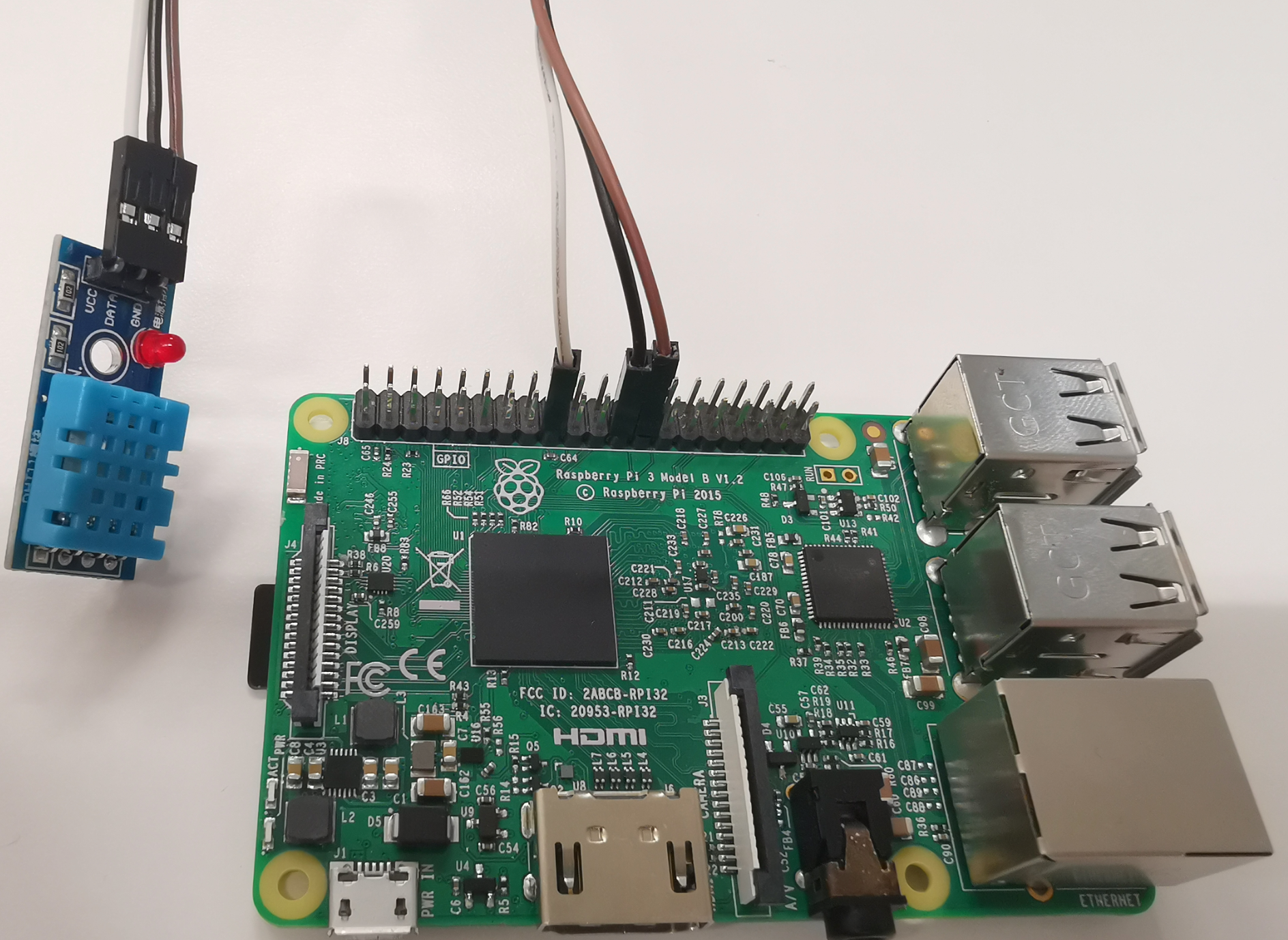




运行结果如下，和web网页demo结果一致，这里使用树莓派作为app部署调度硬件。



在树莓派上部署**温度测量**实例，参考官网给出的temperature-demo案例进行一下流程部署。



**创建设备实例**

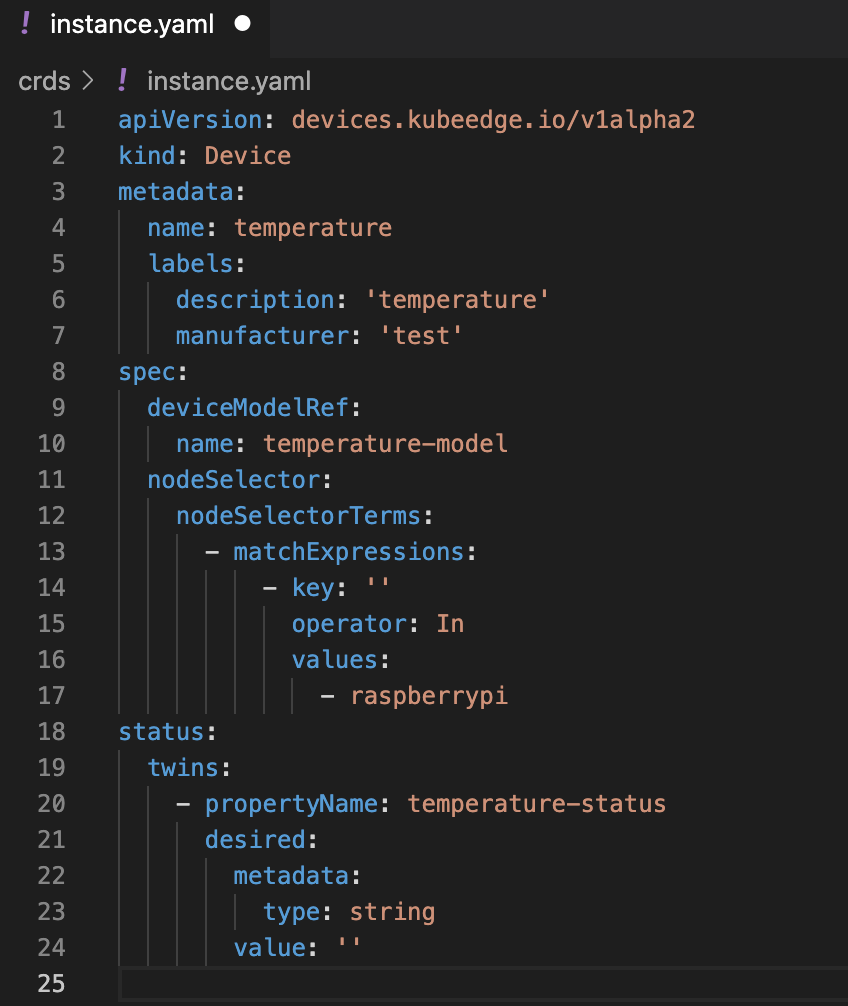
$cd $GOPATH/src/github.com/kubeedge/examples/temperature-demo/crds

$ kubectl apply -f devicemodel.yaml

$ sed -i "s#edge-node#<your-edge-node-name>#g" instance.yaml

$ kubectl apply -f device.yaml

根据自己设置的节点名称情况修改节点名称的配置，这里修改成raspberrypi。



**构建本地镜像**

在树莓派节点执行以下操作

cd $GOPATH/src/github.com/kubeedge/examples/temperature-demo

docker build -t kubeedge/temperature-mapper-demo:arm64 .

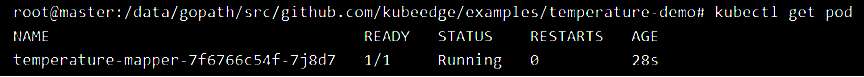


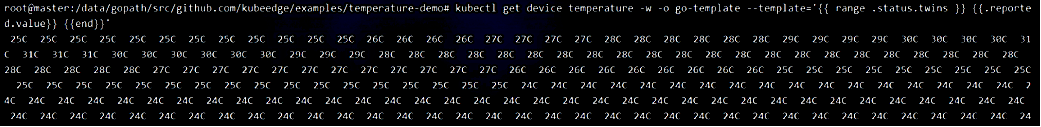
**部署业务**

在云端节点部署业务容器，云端通过kube-scheduler服务结合nodeSelector的标签选择机制将温度监测pod调度至树莓派节点。

cd $GOPATH/src/github.com/kubeedge/examples/temperature-demo/

容器调度和运行结果如下：

kubectl create -f deployment.yaml



树莓派环境配置及业务部署的详细流程可见GitHub中raspberry\_kubeedge文件夹。内部还包括业务代码及导出的arm64镜像。