EdgeX相关文档

1. docker与docker-compose

首先是安装docker，可以使用一行命令直接完成：

```
sudo curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

安装成功后添加用户组：

```
sudo usermod -aG docker your-user（用户名）
```

随后安装docker-compose：

```
sudo curl -L https://github.com/docker/compose/releases/download/1.24.0-rc1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
```

2. EdgeX Foundry

2.1 主平台

拉取镜像，获取yml配置文件：

```
docker-compose -f docker-compose-geneva-redis.yml pull
```

启动全部容器

```
docker-compose -f docker-compose-geneva-redis.yml up -d
```

2.2 UI模块与MQTT设备模块

在docker-compose-geneva-redis.yml中添加如下关于UI的模块，并将MQTT模块的注释去除

```
services:
  ui:
    image: edgexfoundry/docker-edgex-ui-go:1.2.1
    ports:
      - "4000:4000"
    container_name: edgex-ui-go
    hostname: edgex-ui-go
    networks:
      - edgex-network
```

随后再次拉取镜像，重启模块。即可运行UI控制台

3. 虚拟设备

EdgeX与设备之间的通信采用MQTT协议来完成，具体地说依靠我们自主搭建的MQTT Broker来完成。

3.1 准备工作

进入127.0.0.1:8500，即EdgeX微服务管理界面。访问`Key/Value -> edgex -> devices -> 1.0 -> edgex-device-mqtt -> Driver`，修改**IncomingHost**字段和**ReponseHost**字段为Broker主机地址。当然，MQTT设备也需要监听Broker上某一个主题，以及在接受命令后返回Json数据的能力。

此外，需要自主编写MQTT设备配置文件，这可以参照EdgeX默认提供的随机数生成设备的配置文件来完成。

3.2 添加设备

进入127.0.0.1:4000，即EdgeX设备管理平台界面。在DeviceProfile一栏点击+号上传编写好的配置文件，在DeviceService一栏选择edgex-device-mqtt，点击Devices添加设备，并如实填写设备和Broker参数即可。

点击新添加的设备，点击commands，点击send按钮发送消息测试。如果返回Json数据则说明部署成功。

4. 规则引擎

规则引擎的数据流（Stream）以及响应（Actions）也是基于MQTT协议的。

4.1 规则引擎模块安装

仍然使用docker安装即可：

```
docker run -p 9081:9081 -d --name kuiper emqx/kuiper:$tag
```

4.2 数据流与主题的定义

我们的应用场景共设计了两个数据流，三个MQTT主题，具体地说即：

tempStream流、DataTopic_Temp主题，即传感器接受的温度消息

stateStream流、DataTopic_State主题，即华为开发板关于人脸识别的判别消息

CmdTopic_Arduino主题，即负责对Arduino发出命令的主题

然后即可具体的定义数据流，采用图形化控制台来进行定义即可。也可以采用命令行定义，比如：

```
CREATE STREAM tempStream () WITH (FORMAT="JSON", CONF_KEY="myDemo")
```

4.3 规则的定义

由于应用逻辑并不复杂，我们的规则也是比较简单的，仅有三条：

```
# 接受温度 >= 28摄氏度时，蜂鸣器响
SELECT * from tempStream where temperature >= 28
# 数字状态为1时，LED灯常亮
SELECT * from stateStream where state = 1
# 为0时，LED闪烁
SELECT * from stateStream where state = 0
```

具体的响应动作（action）实际上由图形化界面来进行添加，不赘述。需要指出的是，除了与MQTT服务器进行通讯以发送命令外，我们还设置了log()命令用于对日志进行持久化。