DeviceOn/EIM

一站式边缘设备远程智能管理方案

User Manual

赋华慧创
iConnect Everything

目录

1. 简介

DeviceOn/EIM 是一个专注于解决远程设备管理，监控和运维的解决方案。在物联网飞速发展的时代，边缘计算已成为各种物联网解决方案中最为重要的组成部分。通常，边缘计算网络中，边缘设备数量庞大，它们大都分散部署在不同的场域。而且这些设备可能基于不同的 CPU 架构，不同的操作系统，甚至不同的工业协议应用等，有时，有些设备部署的地域更是人迹罕至。面对如此复杂的环境，如何确保这些设备稳定正常地运行，如何监控操作系统，传感器以及工业协议产生的数据，如何监控设备的异常状态并产生告警，如何远程分析设备故障的原因，如何批量控制远端设备，以及如何快速地更新远端设备系统&应用，是所有工业边缘设备运维管理面临的难题。

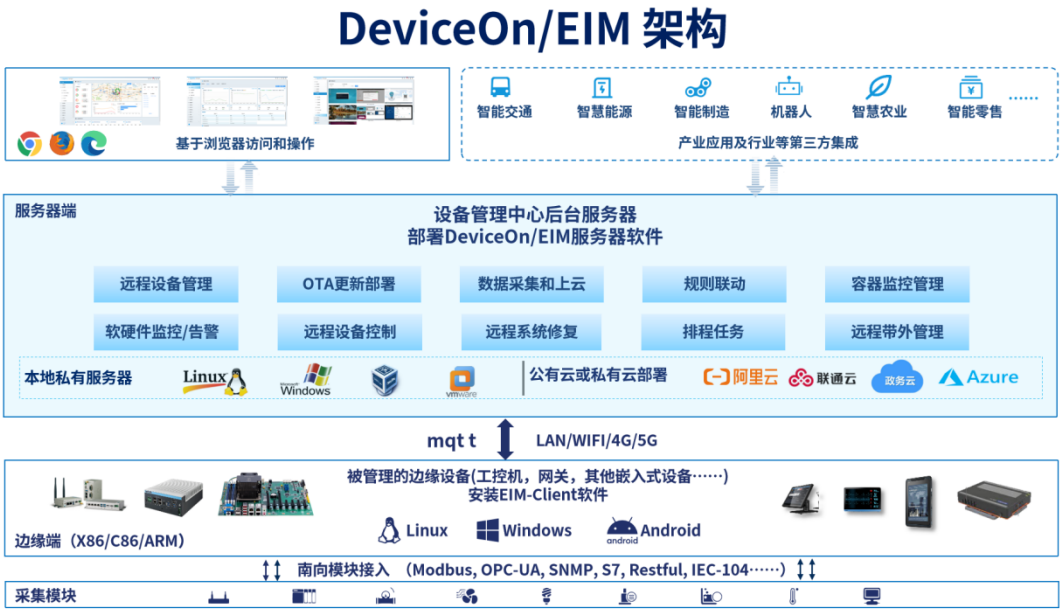
DeviceOn/EIM 的产生旨在解决以上的这些运维痛点，让运维人员通过远程控制的方式，快速便捷地处理一些复杂性低，冗余度极高的日常运维工作，和远程监控地方式，及时发现设备故障，排除设备故障。极大地提升边缘设备的运维效率，大大地降低边缘设备运维人力，物力，财力成本。同时助力企业数字化转型成功。

1.1. 什么是 DeviceOn/EIM

DeviceOn/EIM 是一个专注于解决远程设备管理，设备监控和运维，工业数据采集和显示的一站式工业物联网解决方案。该管理界面基于 Web 浏览器，使用简单，方便且功能丰富，便于集成，可以有效提升工业环境边缘设备的管理，运维效率，大大降低运维成本。

1.2. DeviceOn/EIM 架构

DeviceOn/EIM 是一款基于 MQTT 协议的 Client/Server 架构模式的系统。Client 负责边缘设备（基于 Android，Windows，Linux）的相关数据的采集，分析，整理，汇总等，并将结果上报至 Server，Server 会显示相关数据及分析结果，并把用户需求传递给 Client，实现对边缘设备的控制，监控等。其架构大致如下图所示：



1.3. DeviceOn/EIM 的应用场景

DeviceOn/EIM 专注于解决远程设备管理，监控和运维。所以基本上，DeviceOn/EIM 大致有以下应用场景：

- **场景一：DeviceOn/EIM 用于设备运维领域**

DeviceOn/EIM 的设备监控，设备管理，远程分析等系统，使 DeviceOn/EIM 特别适合于边缘设备&边缘计算节点，边缘服务器等的运维工作。有效提升运维效率，节约人力，物力，财力成本等方面，效果显著。目前已有很多成功案例，将 DeviceOn/EIM 用于设备运维领域。他们广泛地分布于教育，医疗，零售等领域。

- **场景二：DeviceOn/EIM 用于设备监控&告警**

DeviceOn/EIM 的设备监控，规则引擎，数据告警，屏幕监控等功能，使 DeviceOn/EIM 特别适合于多 CPU 架构，多种操作系统的边缘设备监控&告警场景。目前已有一些客户，将 DeviceOn/EIM 用于工厂，医院，电力能源行业等监控场景。

- **场景三：DeviceOn/EIM 用于设备管理**

DeviceOn/EIM 的设备管理，和设备监控，设备批量控制，OTA 更新等功能，使 DeviceOn/EIM 特别适合于远程设备管理，批量控制等场景。其完美简化了传统设备管理中，人为处理重复度极高，冗余度极高的工作的问题。提升设备管理的智能化，数字化。目前也有客户，将 DeviceOn/EIM 用于工厂，交通，零售等设备管理场景中。

- **场景四：DeviceOn/EIM 对工业现场数据的采集，监控和可视化**

某个客户，在工业现场有非常多的设备，且使用到各种不同的工业协议，如 modbus, SNMP, OPC-UA 等等，客户希望有一套系统，就可以采集不同协议的数据，并对这些协议数据进行监控，对异常数据进行报警，同时可以把这些数据转发到其他的工业应用中，采用 DeviceOn/EIM，不用任何编程，就可以快速搭建出相应的解决方案，可以满足这样的需求，客户非常满意。

还有其他客户将 DeviceOn/EIM 用于机房监控，水利闸门监控，燃气设备监控，工厂产线监控等各种场景，取得了很好的效果。

1.4. DeviceOn/EIM 的应用价值

DeviceOn/EIM 专注于解决远程设备管理, 数据采集, 监控和远程运维。通过 OTA 更新功能，极大方便了系统&系统软件的部署&更新。通过设备监控功能，能及时地发现设备异常，处理设备异常，让设备稳定地运行在健康区间内。通过批处理功能，能有效解决设备运维时，人为参与的重复性极高，复杂度较低的冗余工作。通过设备远程分析功能，能为设备异常提供更深入的分析工作，方便运维人员快速定位问题。通过设备告警功能，能将设备采集中重要的信息，通知给关键人员，或作出对应的操作。另外，DeviceOn/EIM 通过无代码的方式进行数据监控扩展，可以采集工业现场各种数据（包括如 modbus, OPC-UA, S7, SNMP 等等协议），并对采集的数据分析和转发等等，方便的拓展的对工业现场各种数据的监控管理。我们现将 DeviceOn/EIM 的主要功能描述如下：

1. 支持多架构多平台边缘设备

- 支持 X86，arm，arm64，mips 等 CPU 架构的边缘设备；
- 支持 Android，Linux，Windows 等系统；
- 支持 Linux 常见发行版如 debian, ubuntu, centos 等；

2. 边缘设备 OTA 软件更新

- 支持 Linux/Windwos/Android 的系统更新；

- 支持 Linux debian 包，压缩包，Docker compose/swarm 的部署&更新
- 支持 Windows exe 安装包，压缩包的部署&更新
- 支持 Android Apk 等部署&更新

3. 灵活高效的批处理

- 支持大量边缘设备同时进行 OTA 软件更新
- 支持大量边缘设备同时进行远程控制
- 支持多任务处理

4. 私有软件商店解决方案

- 私有软件仓库
- 软件版本管理
- 支持文件上传，下载，支持断点续传

5. 实时设备监控

- 设备状态统计及异常分析
- 应用状态，容器状态，软硬件状态实时监控
- 容器，Client，Server log 抓取

6. 远程故障分析系统

- 远程桌面（可穿透内网）
- 远程 SSH 登陆（可穿透内网）
- 远程屏幕监控
- 实时抓取 Client 设备备端的系统日志

7. 实时远程控制

- 支持计划性，周期性远程开关机，音量，背光控制
- 支持 App，Docker 等应用的管理
- 支持系统，软件，硬件等相关设定

8. 排程任务

- 远程 reboot、脚本等多种任务
- 自定义周期执行
- 日志结果可追踪

9. IBMC 协议支持

- 远程开机、远程系统备份/还原

10. 事件告警&数据转发

- 实时事件上报

- 基于规则的数据转发
- 可配置的事件处理

11. 灵活的数据协议接入系统

- 基于屋模型的宽泛数据模型
- 插件化接入
- 低代码协议 mapper：SNMP，Modbus, OPC-UA, IEC-104，S7.....

12. 支持服务器多平台部署

- 服务器支持公有云，私有云和本地服务器多种方式安装部署
- 支持实体机部署和虚拟机部署
- 支持 Docker 容器部署

以上是 DeviceOn/EIM 主要功能模块的简洁描述，随着我们不断地发展，相信会有更多，更丰富的功能添加进来。

2. 环境搭建

2.1. 概述

通过环境搭建章节，我们希望客户能够快速搭建起 DeviceOn/EIM 的试用环境，通过试用快速了解 DeviceOn/EIM 相关功能，以便可以根据自身需求，快速进行评估。要使用 DeviceOn/EIM，仅需以下三个步骤:

1. 服务器环境的建立
2. 设备环境的建立
3. 设备和服务器建立连线

当完成设备和服务器的连线后，就可以通过 DeviceOn/EIM 来对基于 Windows，Linux 和 Android 的设备进行远程监控管理了。

2.2. 服务器环境建立

从前面介绍的应用场景，我们已可以看到，DeviceOn/EIM 服务器可以支持多种平台的部署，目前已经验证过的平台有:

1. 客户自有内部服务器
2. WISE-PaaS 工业云平台(包括公有云和私有云)
3. 阿里云 ECS 虚拟机
4. 微软 Azure 云虚拟机

Server 端的配置要求如图:

下面我们分别进行介绍。

如果对下面的介绍有疑问，或者下面介绍的部署方式无法满足你的部署需求，请随时联系我们，我们可以提供专业的支持服务

2.2.1. ISO 镜像安装

如果你需要将 DeviceOn/EIM 安装到一台新的服务器或者虚拟机中，我们提供了一个可安装的 ISO 镜像，可以把这个镜像刻录到 U 盘上，然后通过 U 盘安装到本地服务器或者虚拟机中。这种方法非常方便，安装过程全自动，不需要人为干预。不过通过 ISO 镜像安装，安装程序会重新格式化本地服务器或者虚拟机的磁盘，所以这种方式针对的是全新的安装，如果你需要安装到已有系统的服务器或者虚拟机中，可参考后面第二种在线安装方式。

ISO 镜像下载地址：

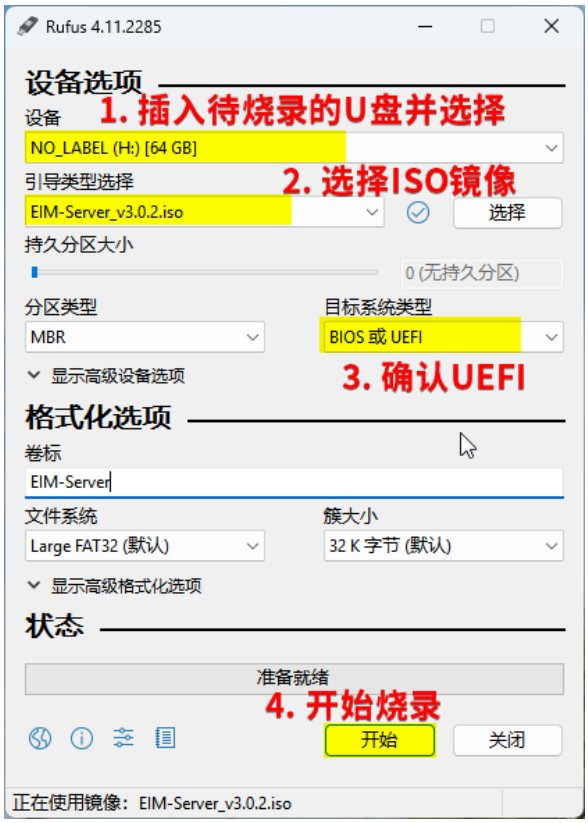
1. 百度云盘下载　提取码：xian

2. Google drive 下载

下载后，将 DeviceOn/EIM ISO 镜像烧录到 U 盘，建议使用 rufus 工具进行烧录.

Rufus 下载：https://rufus.ie/

Rufus 工具启动后界面如下图，插入 U 盘，选择 DeviceOn/EIM ISO 镜像，点击开始进行烧录。

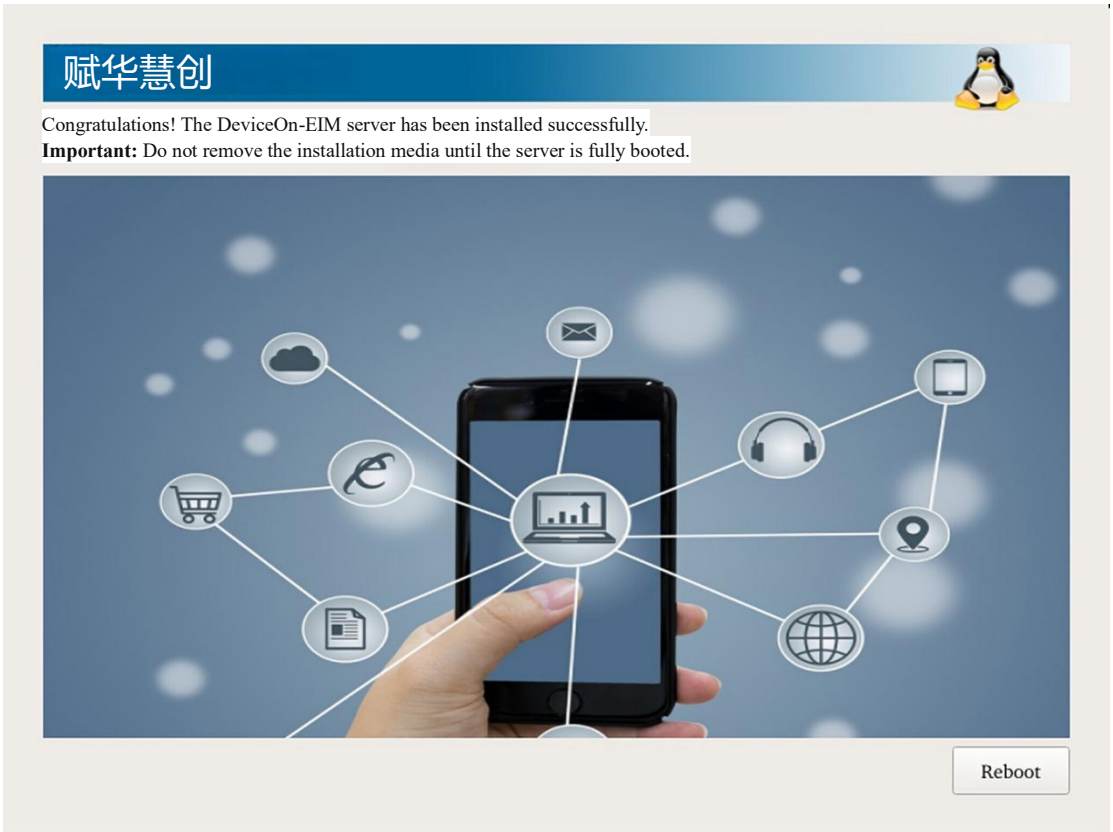


完成烧录后，插入 U 盘到服务器，上电进入 BIOS 设定界面，设定为从 U 盘启动，然后重新启动服务器，就可以自动完成整个安装过程。

注意：使用 DeviceOn/EIM 的 U 盘自动安装的方式会格式化整个硬盘，之前硬盘上的信息都会丢失。如果硬盘上有需要的资料，请务必备份好，防止丢失。如果是虚拟机，当然只会格式化虚拟出来的磁盘，并不会对整个物理磁盘做格式化。如果你是测试验证，建议可以先使用虚拟机环境。

服务器要求：

- CPU：X86_64
- 内存：4G 及以上
- 硬盘：32G 及以上



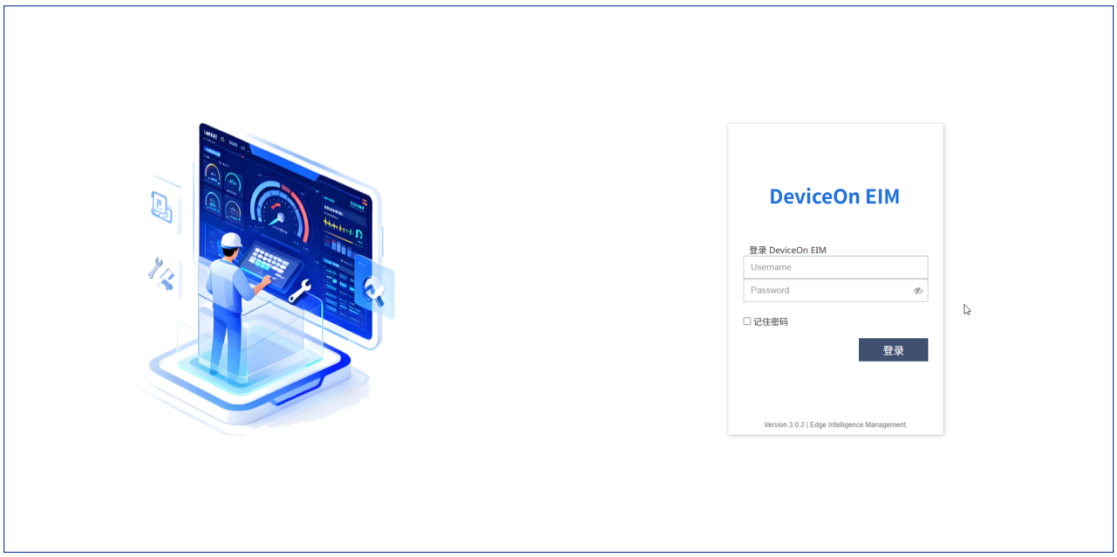
安装自动完成后，重新启动计算机或者虚拟机，DeviceOn/EIM 服务器环境就搭建好了，你就可以通过任何浏览器来访问 DeviceOn/EIM。访问地址如下：http://EIM_Server_IP:8080 这

里的“EIM_Server_IP”地址就是 DeviceOn/EIM 服务器的 IP 地址， DeviceOn/EIM 登录页面如下图：

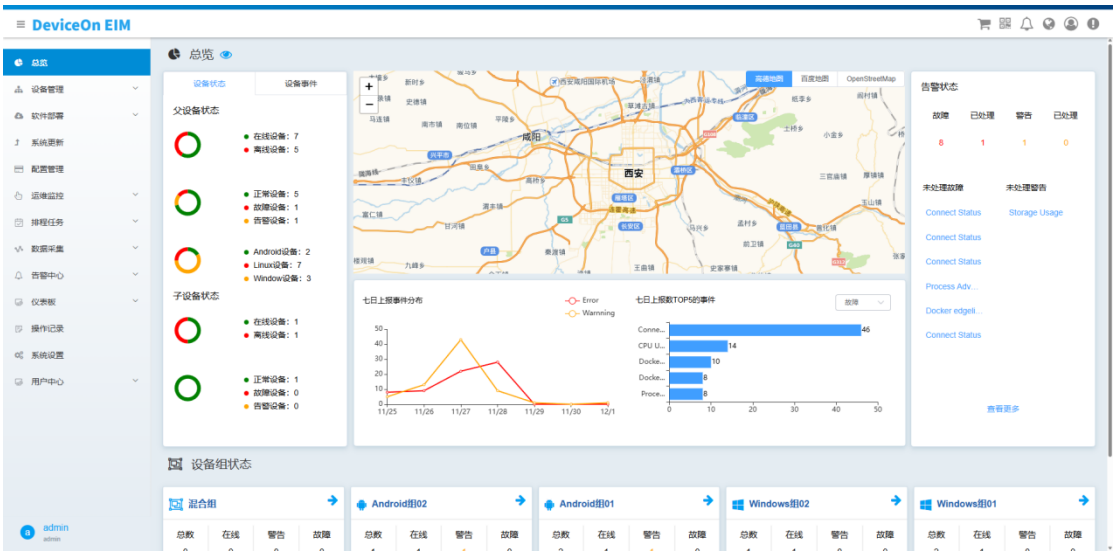
默认用户：admin

默认密码：admin

登录后，你可以修改 admin 的密码，也可以添加新的用户。



下图就是登录后，看到的 DeviceOn/EIM 的总览页面：



通过 overview 页面，我们可以直观的了解设备的定位、设备的状态统计、事件消息统计和处理情况等全方位的整体的了解。

另外，如果安装到虚拟机，可以安装到 VMware, Hyper-V, Virtual Box 等虚拟机中。甚至是 Windows 的 WSL 子系统中，也可以安装 DeviceOn/EIM Server。

同样，如果是 ISO 安装，建议分配给虚拟机 4G 以上内存，32G 以上的存储空间。之前我们的医院客户，就是将 DeviceOn/EIM 安装在他们 Windows 系统的 Hyper-V 虚拟机中的。

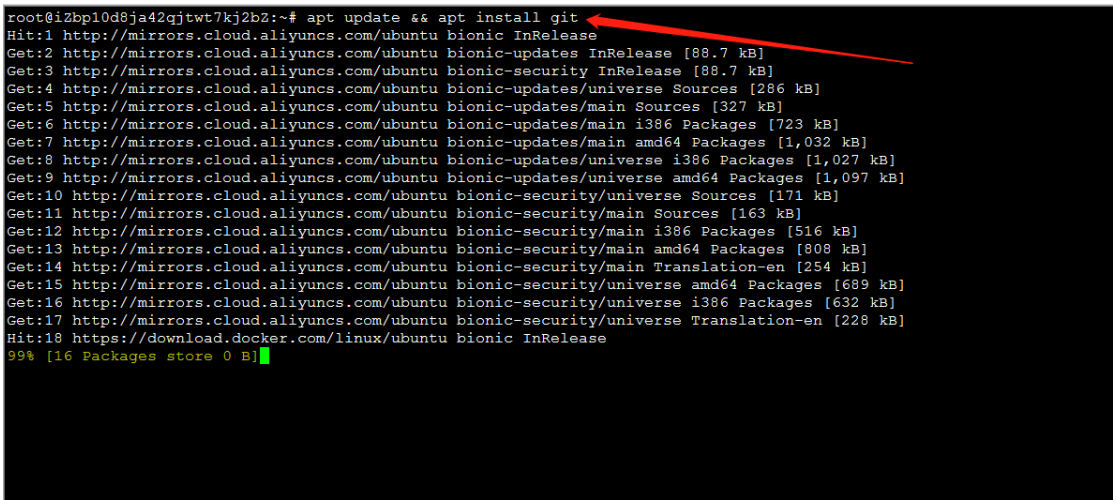
2.2.2. 在线安装

如需要将 DeviceOn/EIM 部署到云端虚拟机，比如微软 Azure 云虚拟机或者阿里云的 ECS 虚拟机等，或者本地的服务器，虚拟机已经有 Linux 系统，想在已有的系统上再安装 DeviceOn/EIM 服务器，就可以通过此方式进行安装，不过此方式是在安装时从网络下载 DeviceOn/EIM 软件包，所以安装速度，跟网速会有关系。在线安装只支持 Linux 系统，并且 Linux 系统要能够支持 Docker 运行环境。如果是 Windows 系统，可以在 Windows 系统下安装 Linux 虚拟机，或者使用 Windows WSL 子系统。在线安装步骤如下：

1. **安装 git 工具**

DeviceOn/EIM 可以通过在线方式进行安装，在线安装需要使用 git,所以首先要安装 git 工具，安装命令如下：

```
# apt update && apt instal git
```



2. 安装 Docker 运行环境

DeviceOn/EIM 在虚拟机中以 Docker 容器方式运行，所以也需要安装 docker 环境和命令：

（1）安装 docker

```
# curl -sSL https://get.daocloud.io/docker | sh
```

如果上述方式安装失败，请使用系统命令安装，

比如 Ubuntu 系统：`#apt install docker.io`

（2）安装 docker-compose 命令

使用系统安装命令安装，

比如 Ubuntu 系统：`# apt install docker-compose`

安装完成后，可以使用下面的命令检查版本，确认安装正确。

```
# docker version
```

```
# docker-compose version
```

3. 使用 git 命令下载 DeviceOn/EIM 的安装脚本文件

```
# git clone https://github.com/EdgeSolution/DeviceOn-EIM.git
```

4. 开放端口

DeviceOn/EIM 服务需要开放以下访问端口：

Server Name	Port	Note	Protocol
EIM-manager	8080	*必须的	http
EIM-ithings	8082	*必须的	http
mqtt	1883	*必须的	tcp
EIM-repo	30001	OTA	http
postgres	5432	*必须的	tcp
minio	9000	OTA	http
repeater	5901, 5500	VNC	tcp
novnc	9191	VNC	websocket
nps	8024, 50500-50510	Terminal	tcp
EIM-license	8081	Ensaas license Service	http
influxdb	8086	存储采集数据	http
Emqx Server web	18083	emqx 前端	http

5. 安装和启动 DeviceOn/EIM

当你通过 git 下载到安装文件后，虚拟机会生成“EIM-Server” 目录，进入该目前，执行 start.sh 脚本。

```
# cd EIM-Server
# chmod +x start.sh
#./start.sh
```

start.sh 会完成 DeviceOn/EIM 服务器的安装和启动，因为安装过程是在线安装，需要从网络上下载 DeviceOn/EIM Docker 镜像，根据网络速度不同，大约需要 10~20 分钟的时间完成安装，请耐心等待。 当安装完成后，就会自动启动 DeviceOn/EIM 服务，这时你就可以通过浏览器访问 DeviceOn/EIM 的服务了。

http://ServerIP:8080

SeverIP 就是 VM 对外的公网 IP 地址。

2.3. 设备环境的建立

DeviceOn/EIM 可以支持管理多种不同边缘设备操作系统，包括：

- **Windows**
- **Linux**
- **Android**

在需要被管理的边缘设备系统中，需要安装一个 DeviceOn/EIM Edge 客户端软件，客户端可以在出货时内置安装，也可以后安装，如果你购买的设备是有操作系统的，在拿到设备后，请先确认，你的系统中是否已经有安装客户端程序。当然，如果你手边的系统没有默认安装这个软件，但是也希望使用 DeviceOn/EIM，这个客户端软件也是可以后装的。

2.3.1. 首次安装

2.3.1.1. Linux Client

注意事项：以 root 身份进行安装

1. 安装步骤：

```
#tar -zxvf EIM-Client_x.x.x.tar.gz
#cd EIM-Client_x.x.x
#./install.sh
```

Note:

- **安装路径说明**

（1）默认安装路径

如果系统有/userdata 分区，优先安装在/userdata/EIM-Client 路径下；

否则，默认安装在/media/recovery/EIM-Client 下。

（2）指定安装路径

如果需要指定安装路径，则在执行安装脚本时添加参数，使用方法如下：

```
Usage: ./install.sh [OPTION [ARG]] ...
   -h          show this help statement
   -p          input the intallation path
```

例如：

```
#./install.sh -p /opt/EIM-Client
```

注意：需要说明的是，如果做系统 OTA 更新时 client 的安装路径不能格式化，否则 client 将被清除。

- **Client 开机自启动说明**

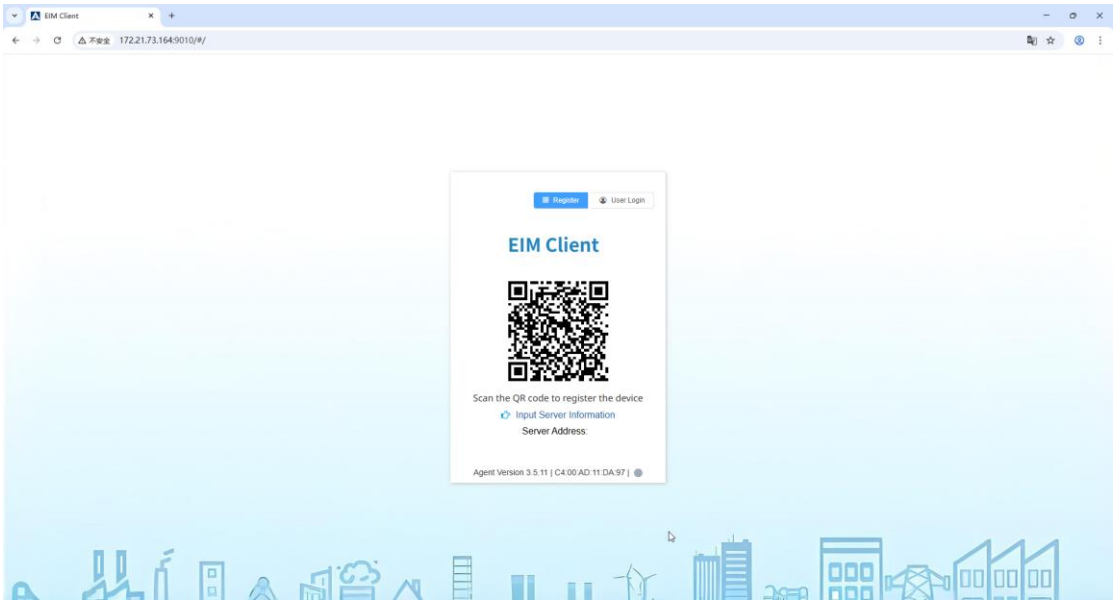
DeviceOn/EIM Linux client 以 service 的方式自启动，若不支持 systemctl，还需将安装路径下的 client 可执行文件写到开机脚本中。通常情况下 Ubuntu 等系统都是默认支持的。

2. 安装结果验证：

如何确定是否已安装，可以参照下面方法确认：

通过浏览器去访问设备

http://设备 IP:9010

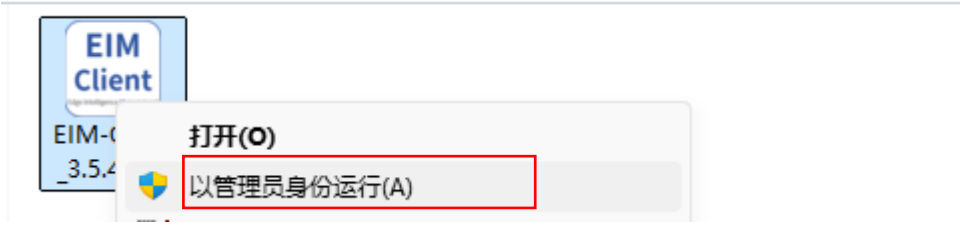


2.3.1.2. Windows Client

注意事项：请以管理员身份安装

1.安装步骤：

右键以管理员身份安装运行即可



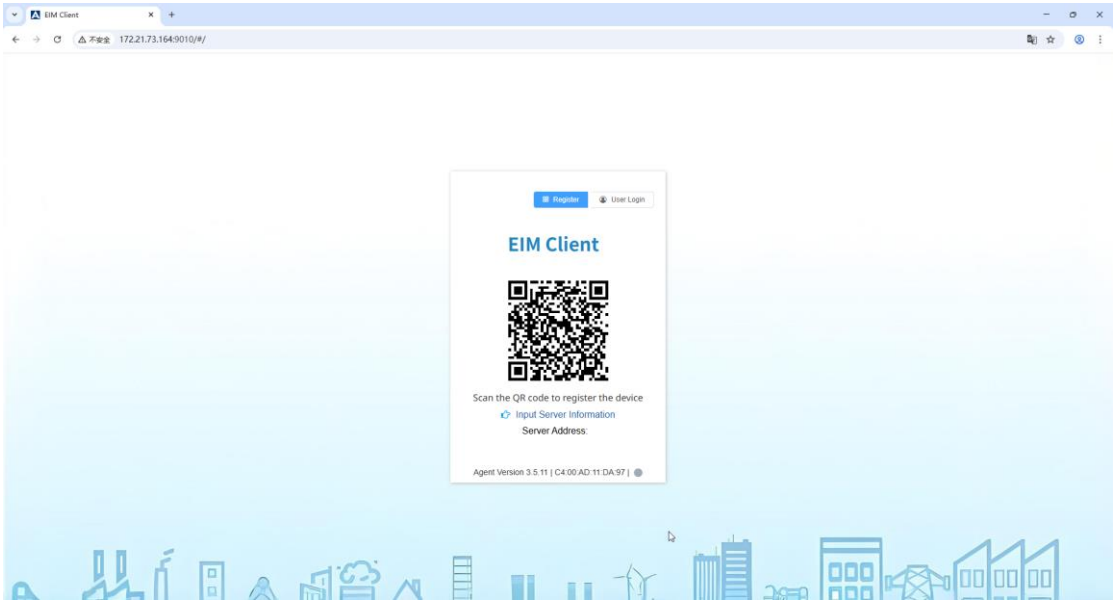
如果不修改安装路径，直接选择默认安装，next 即可完成安装。

2. 安装结果验证：

如何确定是否已安装，可以参照下面方法确认：

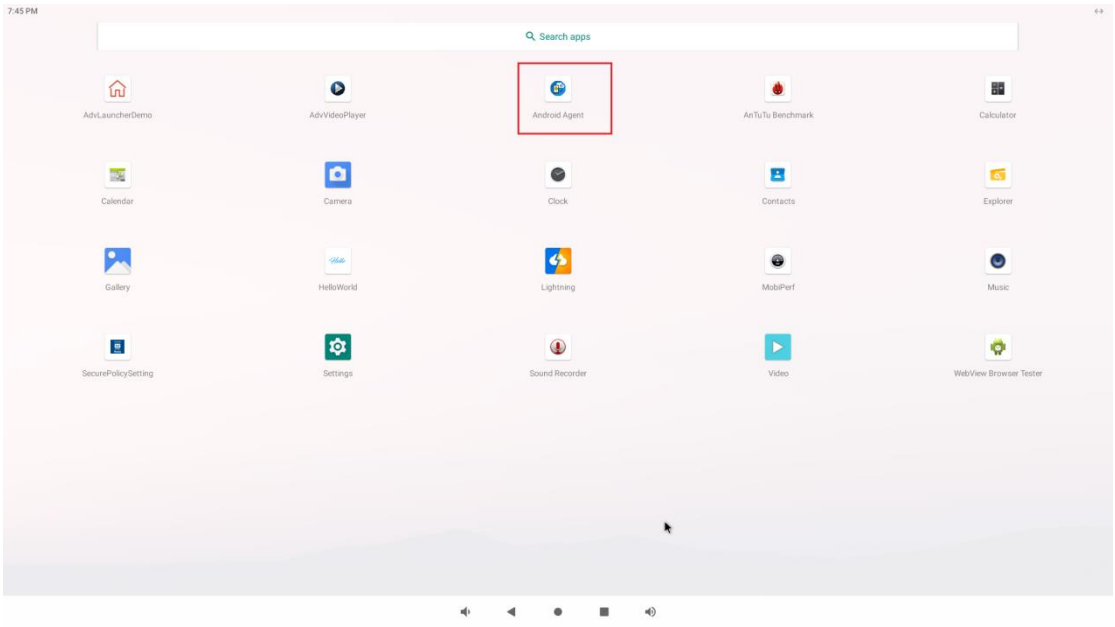
通过浏览器去访问设备

http://设备 IP:9010



2.3.1.3. Android Client

首次安装 Android Client app，需要将 Client apk 拷贝到 Android 设备本地，然后双击安装即可。如果在你的设备中，能够看到下面的应用，说明客户端程序已经安装了。



2.3.2. Client 的升级安装

2.3.2.1. Client 本地更新

Client 本地更新与首次安装基本一致，可以参考 2.3.1 章节

2.3.2.2. Client 远程自更新

EIM Client 支持远程自更新，假如您已经将 Client 连接到 Server 上（具体连接步骤参考 2.4 章节），通过 DeviceOn/EIM 的软件部署功能就可以远程自更新。具体操作如下：

step1：将新版本的 Client 上传到 DeviceOn/EIM 的软件仓库 repo 中。

Linux Client 上传到 linux zip 仓库，

Windows Client 上传到 Windows exe 仓库。

Android Client 上传到 Android apk 仓库

step2：在 DeviceOn/EIM 软件部署页面，选择相应的更新包，直接更新即可

我们支持批量部署和单个设备部署，灵活选用。批量部署则打开 batch 开关选择相应设备组，单个部署则直接选择对应设备即可。

2.4. 设备连接服务器

2.4.1. 概述

当 DeviceOn/EIM 服务器和客户端设备都准备好以后，整个环境就建立起来了，接下来要做的就是让设备连接上 DeviceOn/EIM 服务器，只要连接上服务器以后，DeviceOn/EIM 服务器就可以管理连线设备了。设备连线的本质，其实就是告诉边缘设备，服务的地址是什么，连线的用户名和密码是什么，设备获取这些信息后，就可以自动连接上去了。

我们提供了四种方式来进行设备连线：

1. **直接输入 IP 或者连接信息的方式 （适用于试用环境，输入很方便）**

在 Client 端直接输入服务器的 IP 地址或者连接信息。

2. **导入配置文件方式（适用于工厂预装和试用，不容易出错）**

从服务器端下载一个配置文件，该配置文件中包含了服务器的地址信息和连接信息，当然这些信息是经过加密过的。然后将这个配置文件拷贝到边缘设备上，导入到客户端程序，设备

就会自动连线了。

3. **扫描二维码方式（适用于现场批量部署，大批量部署效率高）**

可以通过扫描二维的方式来让设备注册连线，非常方便，这种方式特别适用大量设备需要部署的情况。但需要借助另外一个带有摄像头 Android 设备，比如 Android 手机，来完成整个扫码部署过程的。这种方式一开始可能不太好理解，需要你仔细的阅读下面的说明，不过一旦掌握，整个部署过程就会非常简单和方便，只要扫描两个二维码，就可以完成设备注册连线，无需任何其他操作。

需要说明的是，在使用上面 1/2/3 三种方式进行部署的时候，会需要打开服务器端网页来获取一些信息。

4. **命令行配置**

针对开发人员的需求，我们提供了一个工具(edityaml)，可以命令行直接配置 Server 信息给 Client，从而可以使 EIM Client 更通用于其他相同机制的 Server 上，即直接配置 Server mqtt 的连接信息后，Client 就会连接到指定 Server。

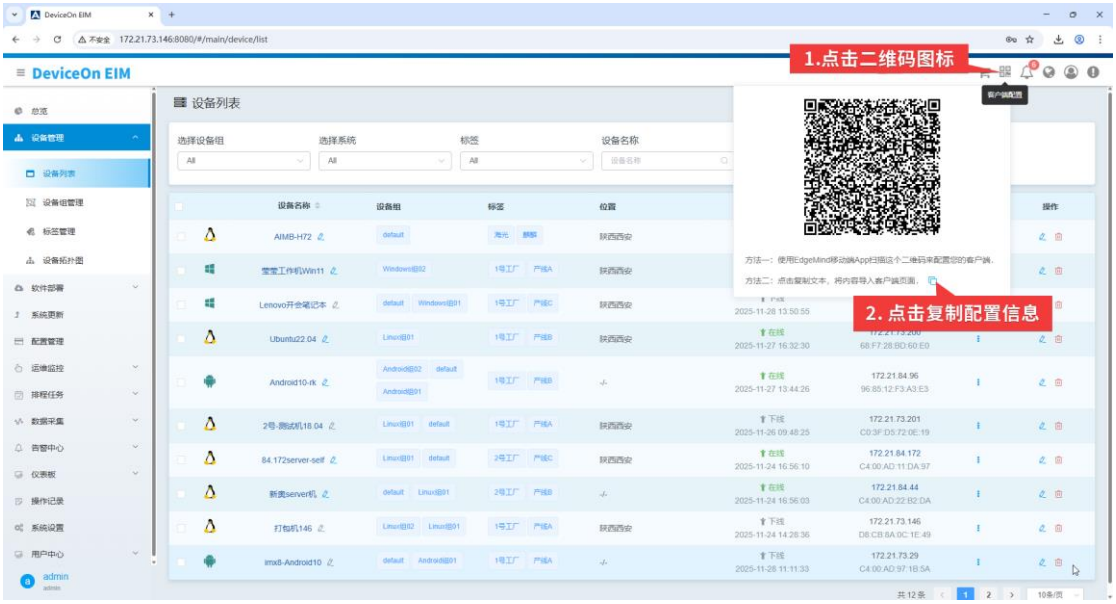
2.4.2. 连接配置方法 1——直接输入 IP 或者连接信息的方式

- **Linux/Windows**

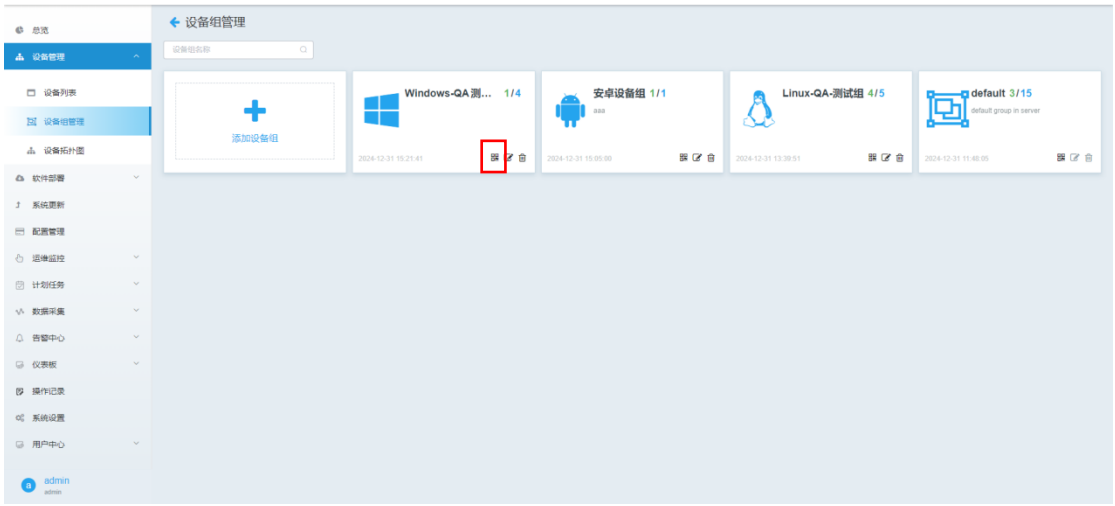
Linux 和 Windows 的这种配置方式一样，仅需两步。

step1：复制 Server 信息

打开 Server 页面，点击右上角二维码图标，弹出框中，点击下图中的复制按钮，复制 Server 信息。



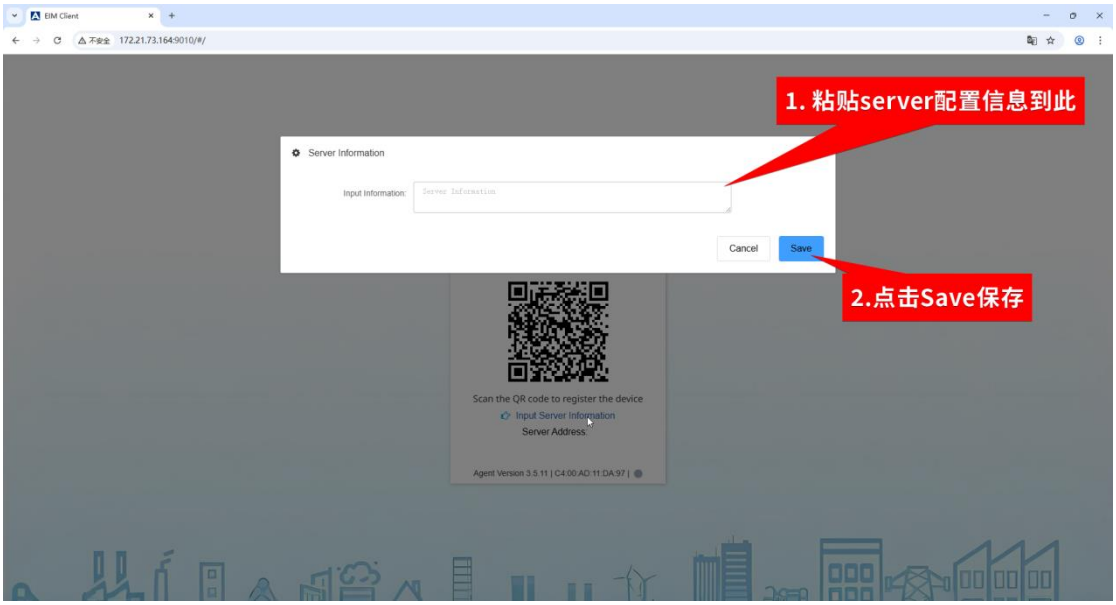
使用该注册信息注册，设备将默认被分配到 default group，如果期望将设备注册上线后直接分配到指定的设备组，则复制设备组对应的注册信息，如下图：



复制该 group 的注册信息

step2：Client 9010 配置页面粘贴 Server 信息

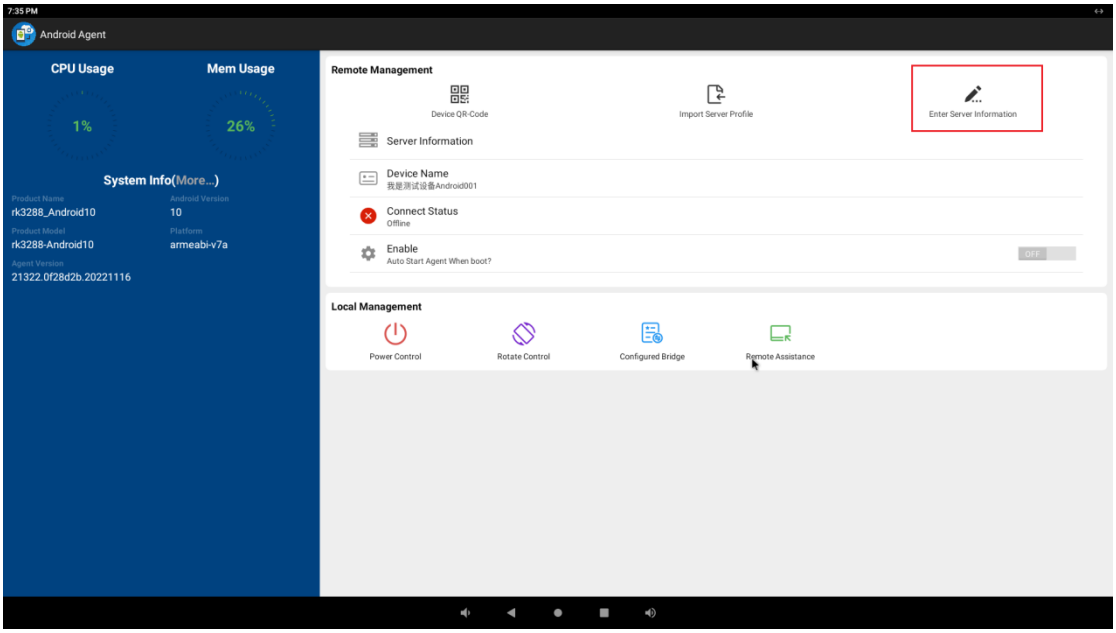
点击 Client 配置页面的“Input Server Information”，弹出输入框，将刚刚复制的 Server 信息粘贴进来即可。



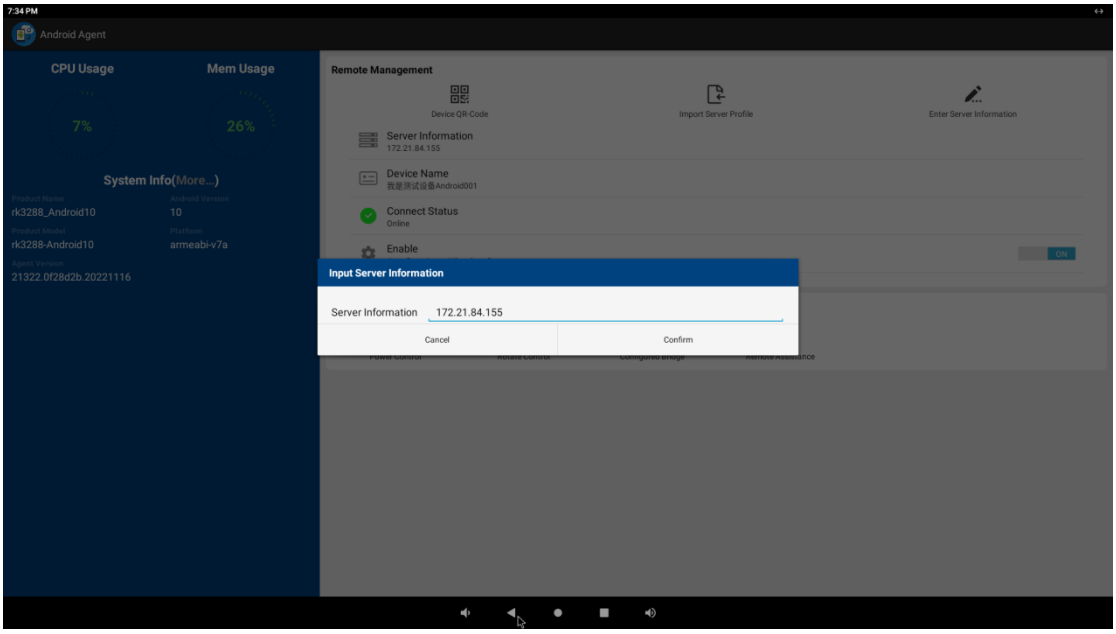
至此，Server 信息配置完成，Client 会自动连接到相应的 Server 上。

- **Android**

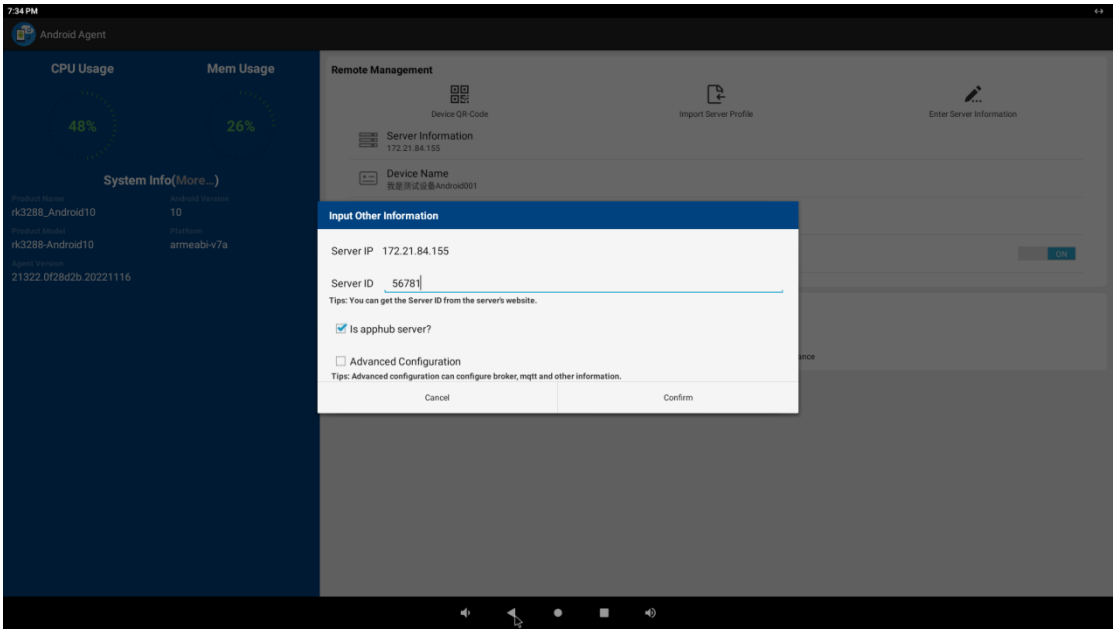
针对 Android 设备打开 Android 设备上的 Client 应用，点击右上角的“Enter Server Information”



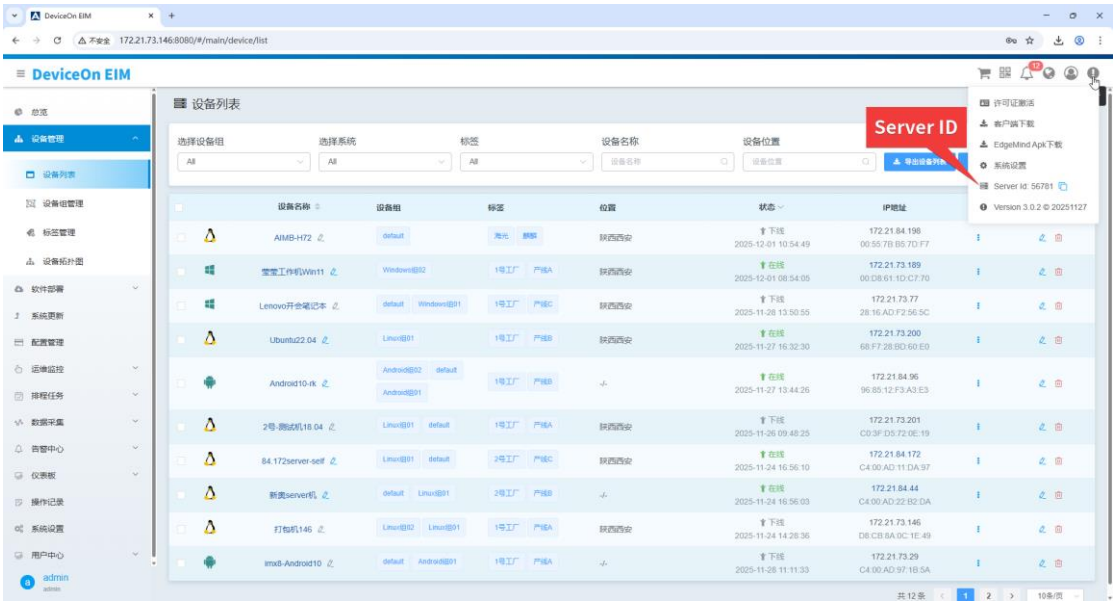
在弹出的输入框内，输入服务器的 IP 地址。如下图:



然后输入 Server ID（该信息由 Server 前端获取）



这个 ID 会显示在服务器的网页上，通过输入 IP 方式注册设备时，就还需要填入这个 Server ID，用户可以登录服务器，获取 Server ID，参考下图：



另外，也可以按照下面 Windows 和 Linux 的方式，不输入 IP，而是直接从服务器上复制加密的连接信息，粘贴到 Server information 栏位，也可以实现自动连接。

2.4.3. 连接配置方法 2——二维码扫描注册

要进行扫码部署，首先，我们需要准备一台带有摄像头的 Android 设备，比如 Android 手机或者 Android 平板，这台设备仅仅在扫码部署时需要，设备部署完成后，就不再需要。本章节后续我提到的 Android 手机，就是指这台扫码设备。赋华有提供一个 DeviceOn/EIM 的手

机端软件，该软件需要安装到 Android 扫码手机上，你可以让手机通过扫码下面的二维码的方式安装这个扫码软件到你的 Android 手机中。

如果无法成功，也可以直接从下面地址下载这个应用，再进行安装：

DeviceOn/EIM 扫码注册应用下载

安装完成后，手机上就可以看到这个扫码应用软件，应用的图标如下：

打开这个扫码软件，界面如下：

就是通过这个扫码软件，可以以零接触的方式，让设备快速完成和服务器的连线。整个过程，其实主要 2 大步即可：

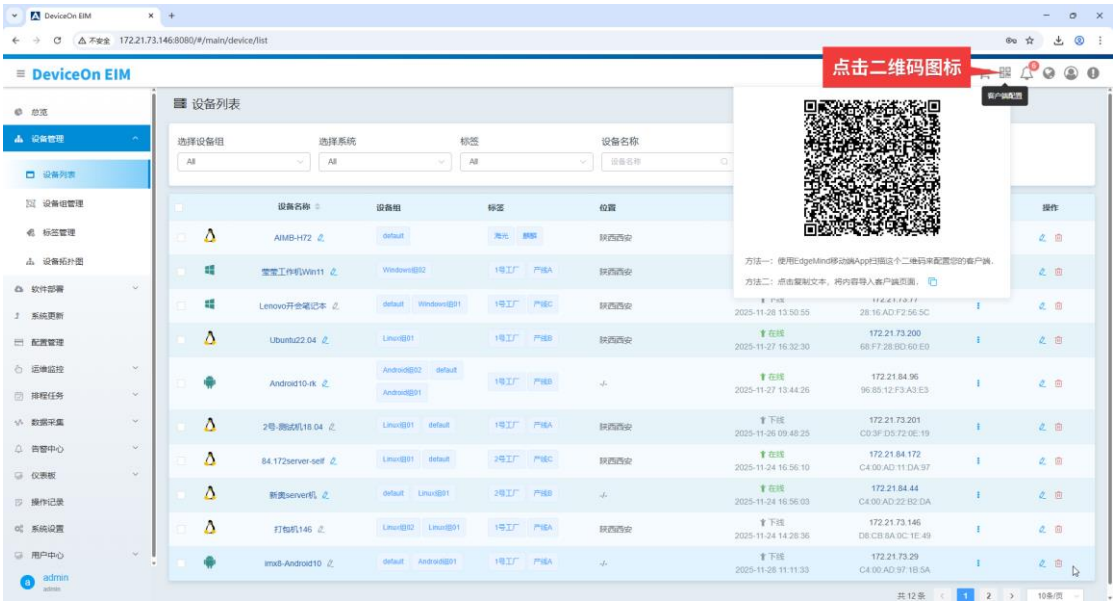
step1：扫描 DeviceOn/EIM Server 页面上的二维码获取服务器信息

打开扫码软件，扫描 DeviceOn/EIM 服务器端网页上显示的二维码，扫码软件就获得了 DeviceOn/EIM 服务器的信息，因为服务器的信息就包含在这个二维码中。

step2：扫描设备端二维码配置 Server 信息

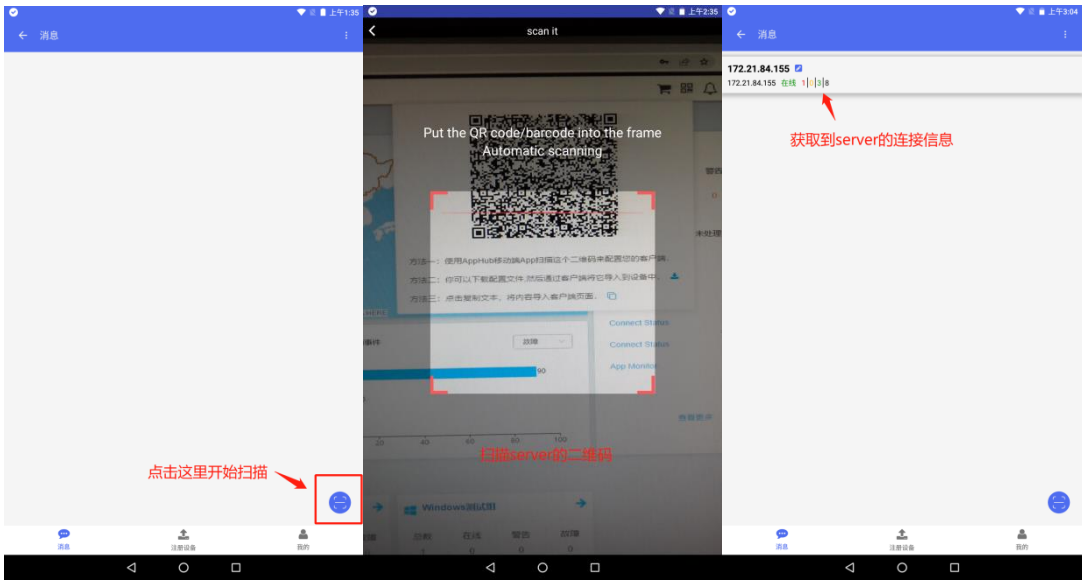
使用已打开的扫码软件，扫描设备上的二维码，扫码软件通过这个二维码，将服务器的地址信息传给设备，设备就知道了服务器的地址，就能自动连接到 DeviceOn/EIM 服务器了。至此，设备就自动上线了，在服务器页面上，也就可以看到上线的设备了。

下面，通过图文详细说明一下如何操作。打开 DeviceOn/EIM Server 页面，点击右上角二维码图标。

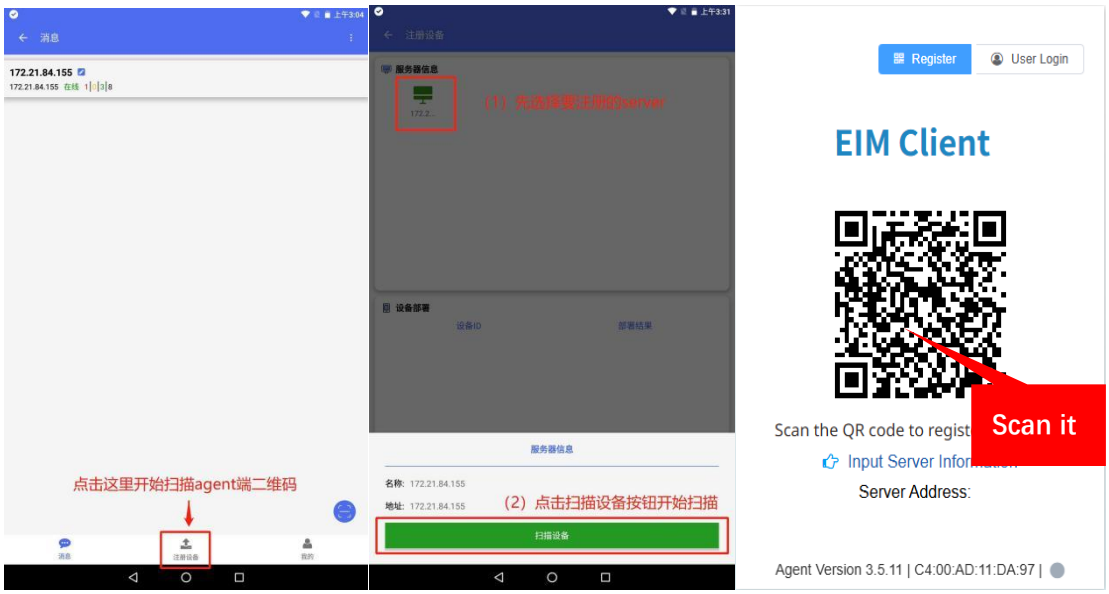


使用打开的 DeviceOn/EIM 扫码软件扫描该二维码，获取 Server 信息。

DeviceOn/EIM 扫码软件操作方法如下：

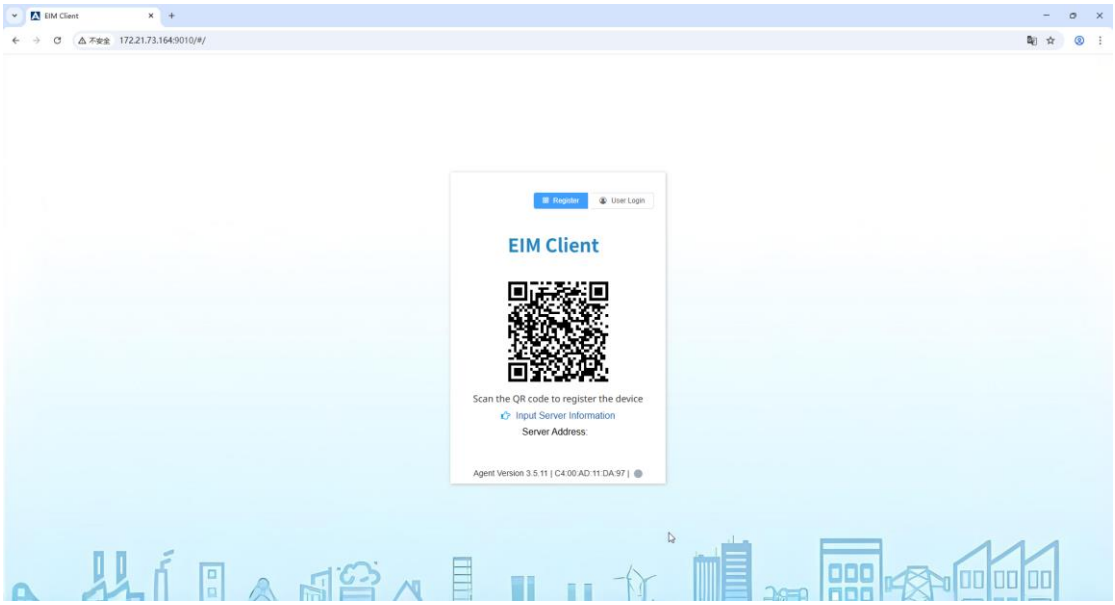


然后打开 Client 的二维码，继续扫描

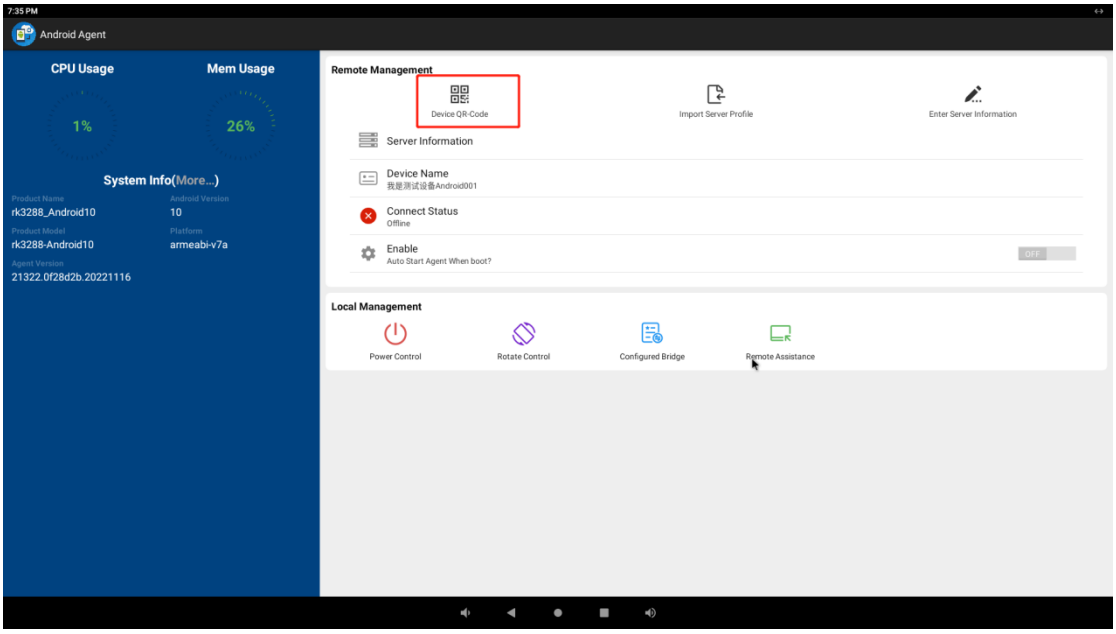


扫描成功后，在 DeviceOn/EIM 的扫码软件内会看到如下界面。

其中，Linux 和 Windows Client 的二维码就在 http://ip:9010 的配置页面，如下图：



Android Client 的二维码点击下图所示即可弹出：



当扫描了设备上的二维码，DeviceOn/EIM 手机端应用软件就获取了设备的 IP 地址信息，并通过 IP 和设备建立通信，将之前获得的 DeviceOn/EIM 服务器信息传递给设备,设备就会自动连线到服务器了。设备自动上线后，你可以从 DeviceOn/EIM 的网页端看到这个设备了，设备的状态是在线的。

注意:

1，如果你的服务器绑定了域名，在扫码时，需要通过 IP + 端口号的方式来访问服务器，显示二维码，另外扫码软件跟设备是通过 IP 地址进行通信的，所以，扫码手机和设备，必须在同一个网络中，手机必须要能够访问设备的 IP 地址。假如手机连接到了 4G/5G 网络，而设备连接在公司内部的网络，那手机就无法访问到设备的 IP 地址，扫码部署就会失败。服

务器和扫码手机不需要在同一网络中。

2，要确保设备是可以访问 DeviceOn/EIM 服务器的 IP 地址的，如果设备不能访问 DeviceOn/EIM 服务器的 IP 地址，即使扫码部署成功了，设备也无法连接上来。

DeviceOn/EIM 手机端软件其实还有更多功能，除了能进行扫码部署，还可以接收 DeviceOn/EIM 服务的实时消息，实时掌握 DeviceOn/EIM 管理的设备的运行情况，后续章节还会进一步介绍相关功能。

同 2.4.2 章节类似，如果希望设备注册上线后直接分配到指定的设备组，那么在扫描服务器二维码时，则需要扫描 group 对应的二维码。

2.4.4. 连接配置方法 3——命令配置

该工具(edityaml)适用于 Linux 和 Windows 系统。具体适用方法如下：

```
root@yy-Lenovo:/home# edityaml -h
this is a advantech yaml edit application for AppHub-Edge ..

Usage:
  edityaml [flags]
  edityaml [command]

Available Commands:
  clean       clean the yaml
  completion  generate the autocompletion script for the specified shell
  help        Help about any command

Flags:
  -t, --bind string                   apphub Edge client is binded [yes | no]
  -b, --broker string                 transport broker
  -f, --cafile_path string            mqtt broker cafile path
  -r, --certfile_path string          mqtt broker certfile path
  -c, --client_application_id string  apphub Edge client application id
  -i, --client_id string              mqtt client id
  -n, --device_name string            apphub Edge device name
  -e, --encrypt string                mqtt broker passwd is encrypt [yes | no]
  -h, --help                          help for edityaml
  -v, --insecure_skip_verify string   mqtt broker insecure verify
  -k, --keyfile_path string           mqtt broker keyfile path
  -o, --org_id string                 apphub Edge client org id
  -p, --passwd string                 mqtt broker passwd (need --encrypt option)
  -y, --path string                   config yaml path
  -q, --qos int                       mqtt Qos
  -s, --server_application_id string  apphub Edge server application id
  -a, --server_endpoint_id string     apphub Edge server endpoint id
  -l, --ssl string                    mqtt broker ssl
  -u, --usr string                    mqtt broker user name
      --version                       version for edityaml

Use "edityaml [command] --help" for more information about a command.
```

如果您有这方面的需求，且不清楚的，可以直接联系我们。

2.5. 产品 License 购买和激活

1. 本地服务器版本

目前本地服务器版本默认为免费试用版，可以免费管理 3 台设备，不需要激活;
如果超过 3 台，就需要申请购买软件许可进行激活;
在评估时，建议安装该免费版本，连接 3 台以内的设备，进行评估和免费试用。

2. 云端服务器版本

（1）阿里云版本，直接在阿里云市场购买和部署。阿里云市场购买
（2）微软 Azure 云市场版本，通过在线安装方式部署，但需要从赋华购买 License 激活。如有不清楚的，随时联系我们!

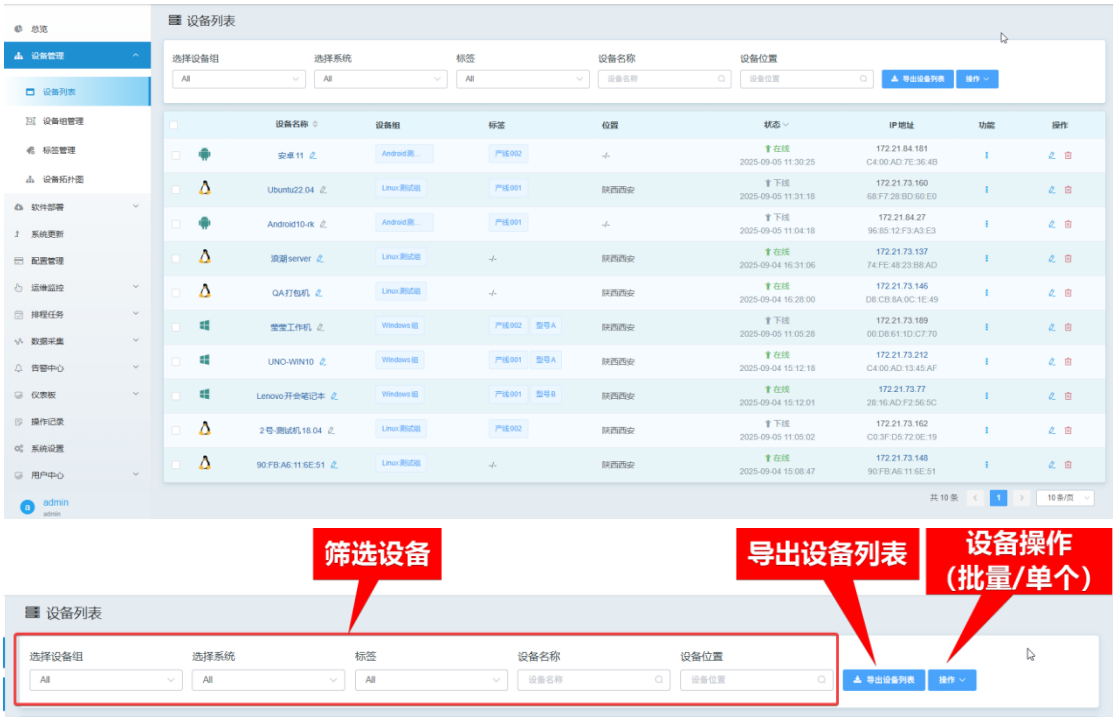
3. 设备管理

3.1. 设备管理

设备管理是 DeviceOn/EIM 的一个基本功能，主要用来查看设备基本信息，监控设备各种软硬件状态、实时显示设备在线状态、远程 KVM、远程终端、远程开关机、远程重启、远程获取客户端日志、客户端强制下线等。

3.1.1. 设备列表

设备第一次注册上线， 系统会自动将其划分到 default 组，进入设备管理页面，可以看到设备的在线情况、修改设备名称、查看设备详情、对设备进行简单的远程控制、将设备划分到指定的设备组，以及将设备列表导出保存为 Excel 等。如下图：



设备列表中，显示了所有和 DeviceOn/EIM Server 连接过的 Client 设备。用户可以通过设备管理 -> 设备拓扑图 查看更加结构化的 设备拓扑关系。

设备列表包含以下项目：

- **设备筛选**

根据设备组、操作系统筛选，或者搜索设备名称、标签、位置

- **设备列表导出**

根据筛选结果可以将当前设备列表中所有设备信息导入并保存到本地的 Excel 表格中。

- **设备操作**

可以批量亦可单个设备的操作；

通过勾选目标设备，可以对目标设备修改（修改设备组、标签或位置）、删除、重启和关机的操作。

- **设备名称**

设备首次注册上来默认显示 Mac 地址用户可以根据需求自定义显示名称。

鼠标点击设备名称会进入设备详情页面等，详细请参照章节 3.1.2。

- **设备组**

显示设备所在的设备组

- **标签**

显示设备的标签

- **位置**

根据设备 IP 获取的设备位置信息，可手动编辑

- **状态**

目前设备有 online 和 offline 两种状态，同时显示当前状态的更新时间；

设备状态亦可筛选查看。

- **IP 地址**

显示 IP 的同时，也是远程配置客户端的超链接

同时显示设备 ID，默认为设备的 Mac 地址

- **快捷功能**

在设备列表的功能列表，我们提供了 7 个快捷功能，分别如下：

（1）远程桌面：支持 Windows、Linux 和 Android

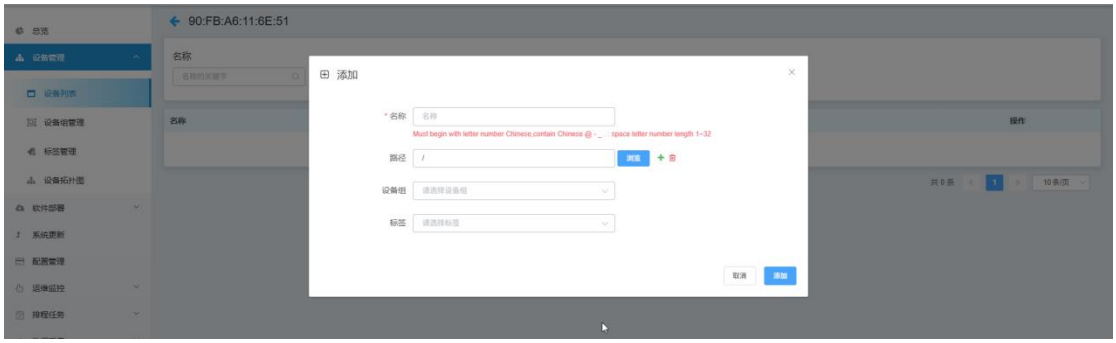
（2）远程终端：目前仅支持 Linux ssh 远程登录

（3）监控：监控设备的硬件、软件、外设等信息详细页面的快捷入口

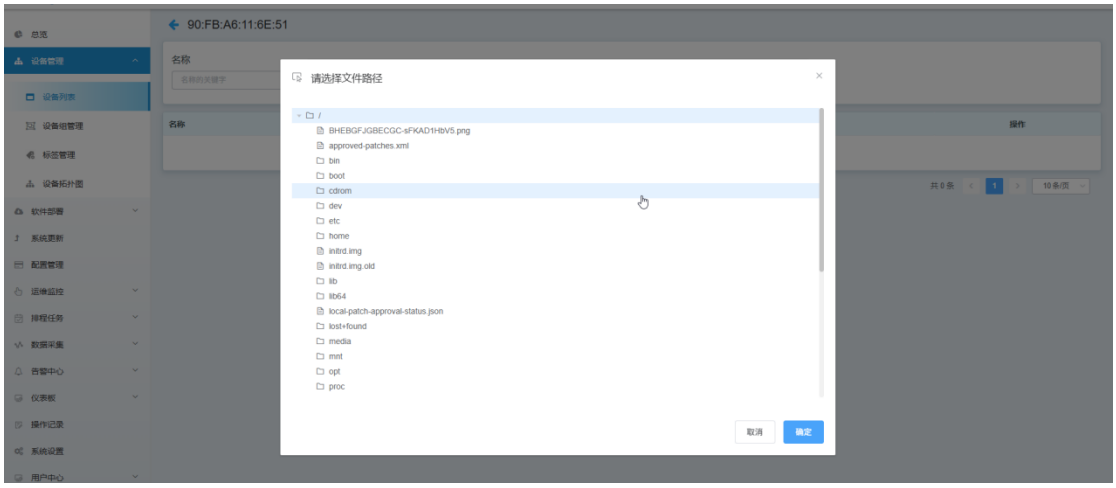
（4）下线：强制使客户端设备下线，客户端将清除 Server 的配置信息

（5）客户端日志：远程抓取客户端运行日志

（6）文件获取：通过添加设备端文件路径，可以远程将设备端的文件获取到本地



通过点击“浏览”，可以直观的查看设备端的文件目录，并选择指定文件



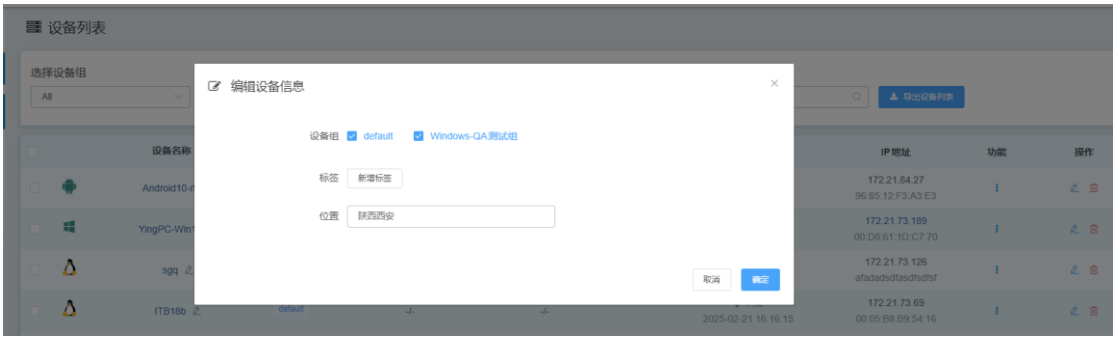
添加后，可以直接点击**下载**以获取文件

如果其他设备希望获取相同路径的文件，只需将该项**复制**到目标设备，然后再相应设备页面点击下载即可；

如果后期改配置项需要更新，也可以直接再次**编辑**它；

如果您不再需要该下载项，亦可直接将其**删除**掉。

- **编辑设备分组、标签和位置**

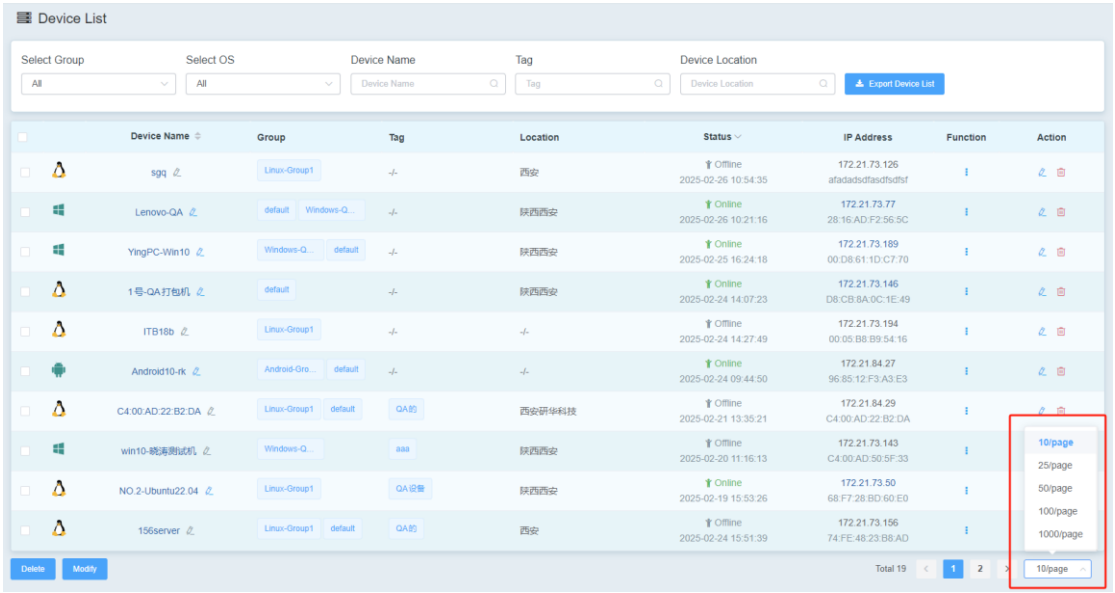


- **删除**

点击“**删除**”按钮，offline 设备可以直接删除掉

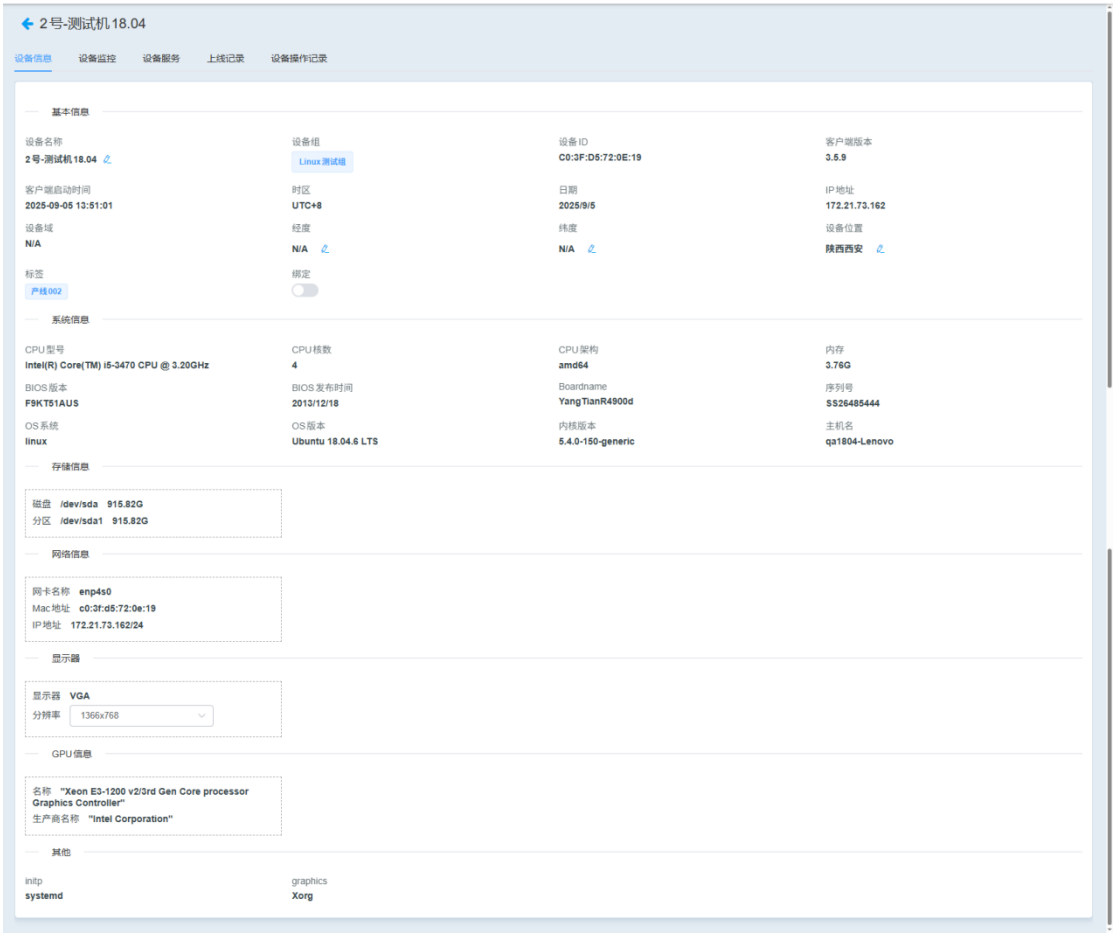
- **设备列表分页显示设定**

  可以根据需要指定每页显示项目条数



3.1.2. 设备详情

鼠标点击设备名称会进入设备详情页面。默认会切换到设备基本信息页面.



设备信息包含以下方面:

- **基本信息**

基本信息包含设备名称，Client 版本，设备 IP 地址, MAC 地址，时区信息，以及设备定位信息。

在设备信息页面设置经纬度便于在 overview 地图直观显示。

- **系统信息**

  包含 CPU 型号，架构&核数，内存容量，操作系统版本，BIOS 信息，BoardName，HostName 等信息。

- **存储信息**

  包含 Client 设备上所有硬盘，EMMC，Nand，SDCard，Flash 等各种存储设备的容量&分区信息。

- **网络信息**

  包含 Client 设备上所有物理网卡的基本信息。

- **显示器**

  枚举 Client 设备上显示器的类型，及对应支持的所有分辨率。

- **GPU 信息**

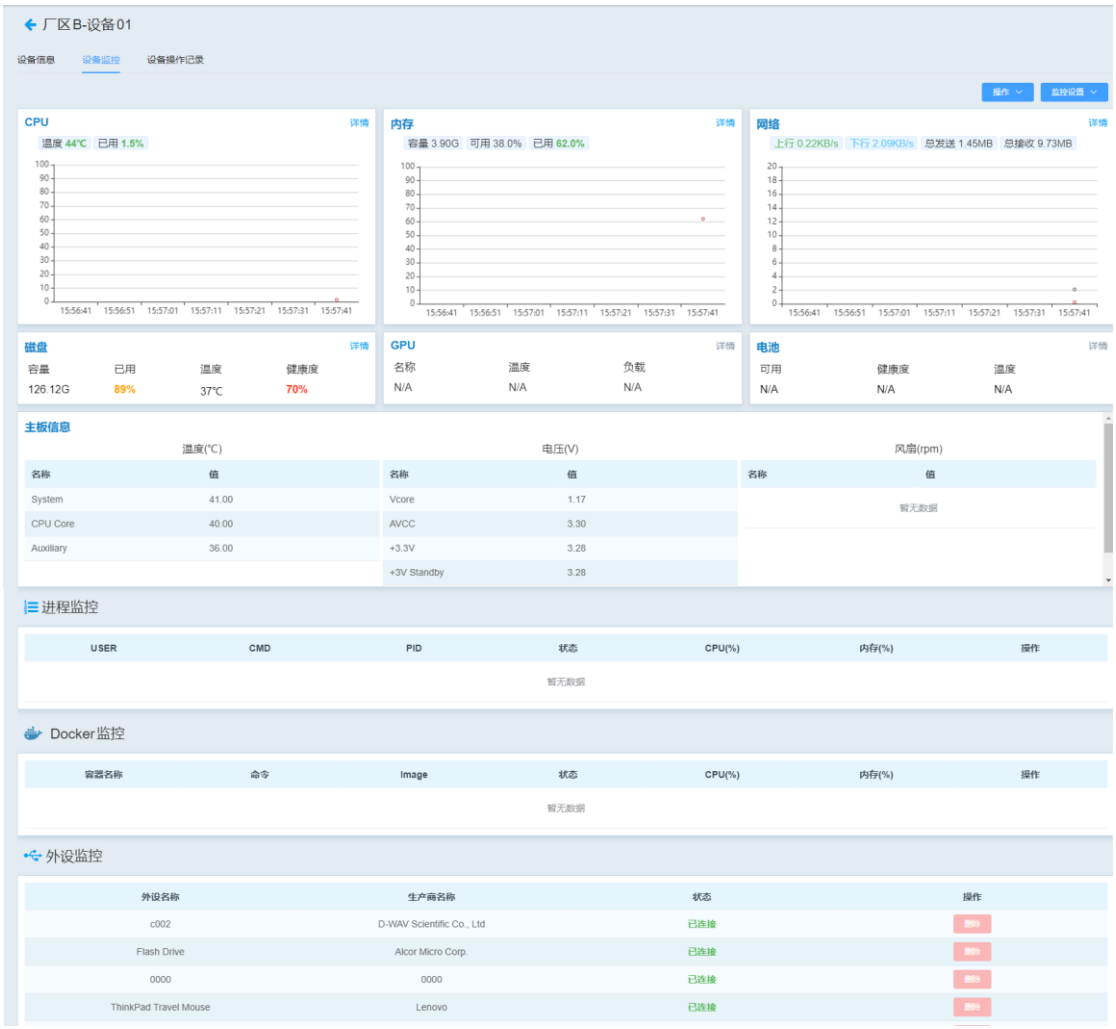
  枚举 Client 设备上的 GPU 的型号&厂商信息。

- **其它**

  枚举 Client 设备的一些其它信息。如 Linux 下是否使用 systemd 作为 init，图形系统是否基于 X11 等。

3.1.3. 设备监控详情

用户也可以通过设备详情页面，进入“设备监控” 页面，其如下图所示：



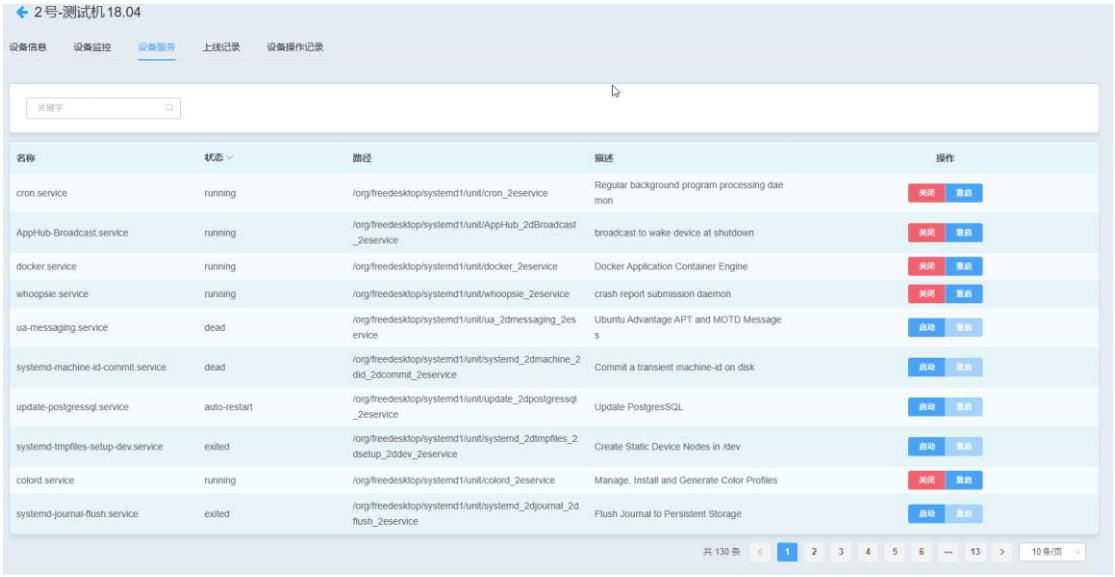
该页面可以实时监控 Client 设备的 CPU，Memory，网络，磁盘等使用量&相关信息。在网络方面，监控网口的上传，下载速率，总流量等。在磁盘方面，监控磁盘各个分区信息，使用率等，还可以监控磁盘温度，健康度等信息。

对于赋华自主的主板，提供对主板电压，温度，风扇转速等监控信息。非赋华主板不保证其主板信息的正确性。

关于如何设置硬件监控的阈值和软件监控列表，请参考章节 6.1。

3.1.4. 设备服务管理

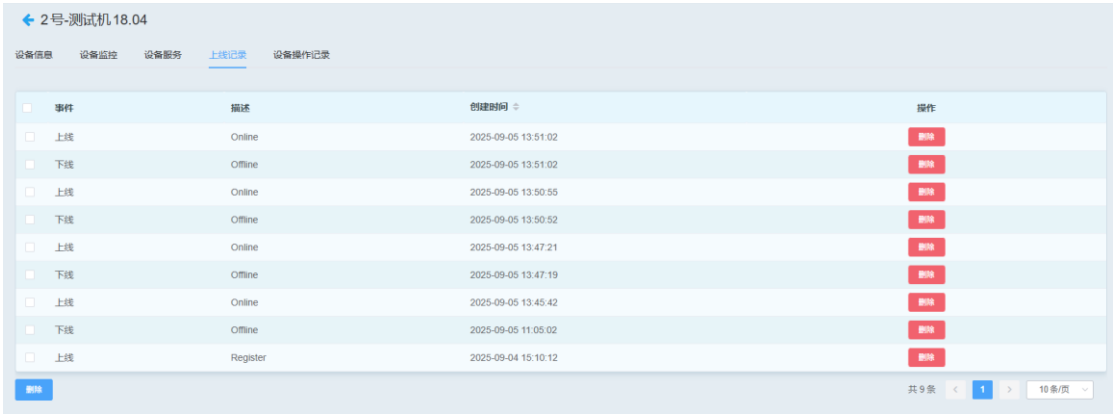
用户也可以通过设备详情页面，进入“**设备服务**”页面，其如下图所示：



可以远程关闭或者重启运行在设备端的服务。

3.1.5. 设备上线记录

用户也可以通过设备详情页面，进入“**上线记录**”页面，其如下图所示：



可以清楚的了解，设备的上线下记录。

3.2. 设备组管理

通过【设备管理】->【设备组管理】进入设备组管理页面，该页面主要是对设备组进行增删改查。如下图

- default 组

所有设备首次注册时，如果没有指定设备组，那么将默认分配到 default 组中，如果后面需要将其再分配到其他 group，则参考 3.1.1 章节其他操作。

- 添加设备组

设备组分为 Android、Linux、Windows、Mix 四种类型，在新建添加设备组时选定。其中 Android、Linux 和 Windows 三种类型根据操作系统划分，只能将对应正确的设备分到该类型的 group 下面，以方便批量操作；Mix 为混合组任意操作系统设备都可划分到该类型的 group。

- 编辑设备组

对于已经创建的设备组，仍然可以修改设备组的名称和描述信息

- 删除设备组

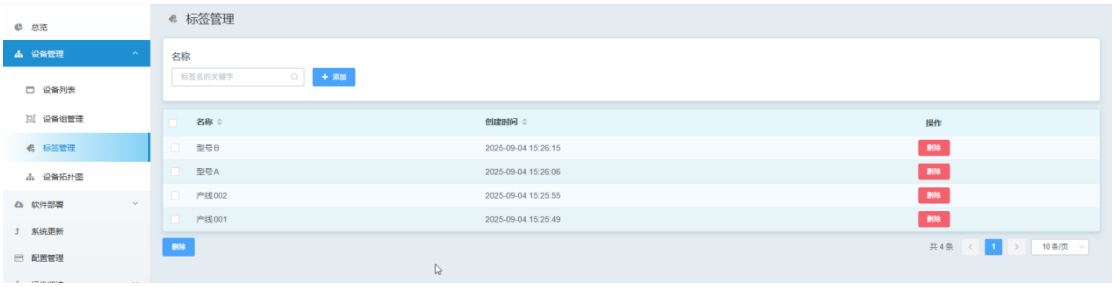
设备组下没有关联设备的情况下，可以删除该设备组

- 设备组二维码

每个设备组都有其独一的二维码，方便在设备注册时，直接将设备注册上线的同时直接划分到相应的 group 内。

当然，设备组卡片上可以直接显示当前设备总数和在线数，此外点击设备组卡片，可以直接进入该设备组下的设备列表。

3.3. 设备标签管理

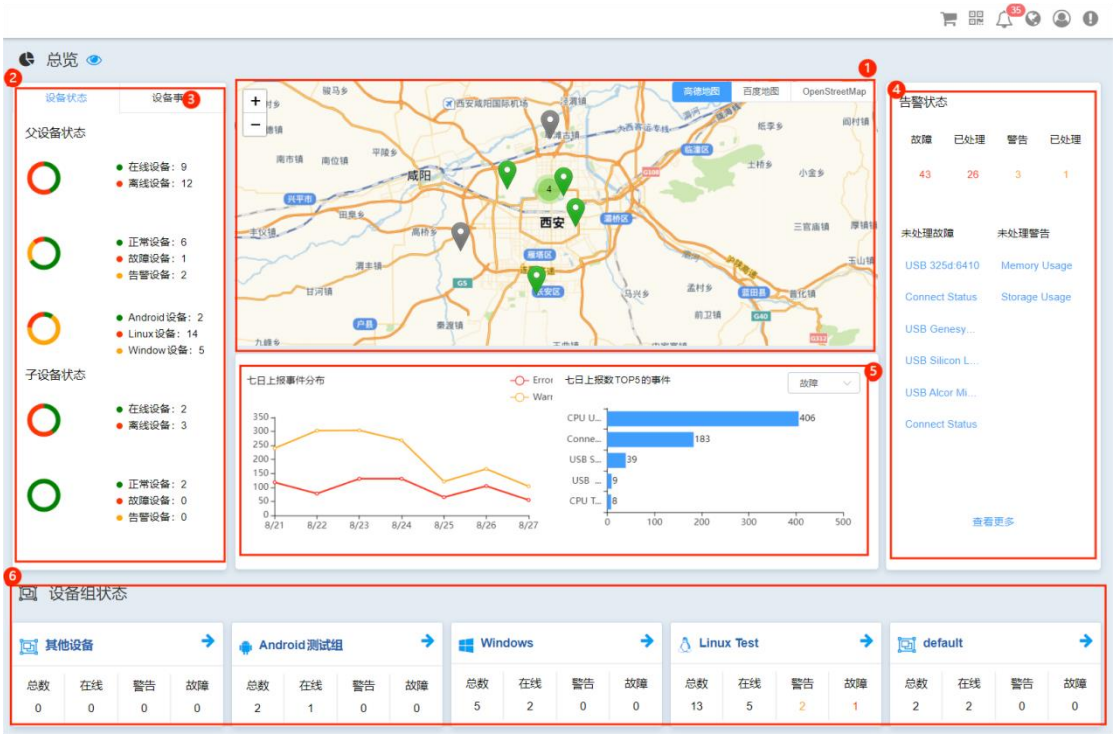


在设备标签管理页面创建标签后，就可以给设备添加不同的标签，增强设备辨识度更方便管理。

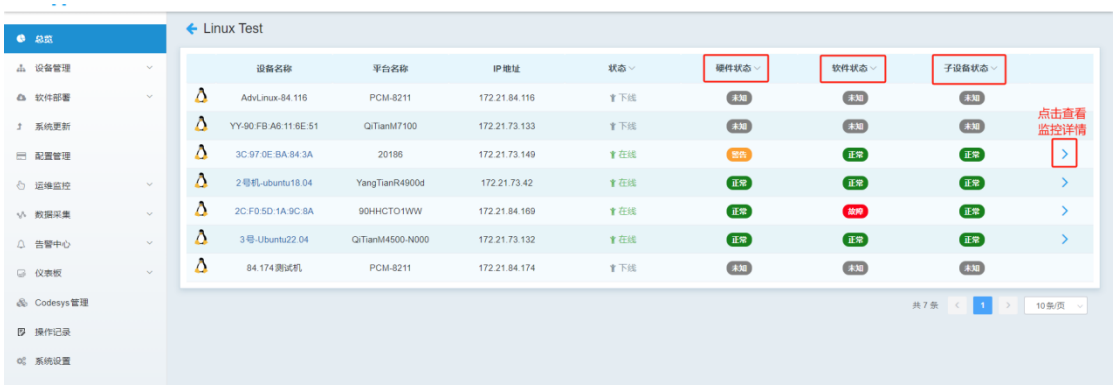
3.4. Overview

Overview 主要是对设备和子设备的在线状态、系统类型、异常状态和异常报警的统计和展示。

1. **设备定位：**
   支持百度地图和 openStreet 地图，可以满足国内外用户查看设备定位的需求。如果设备发生告警，点击可以查看该设备的位置和报警详情。
2. **设备的状态统计：**
   父设备统计：包含父设备的在线和离线数、Android/Linux/Windows 系统的设备数、正常/警告/故障的设备数
   子设备统计：包含子设备的在线和离线数、正常/警告/故障的子设备数
3. **设备事件统计：**
   图表显示七日内告警最多的五个父设备和子设备、七日内警告最多的五个父设备和子设备；
4. **告警状态：**
   实时显示所有设备发出的故障告警和警告告警并可以点击查看具体详情
5. **设备事件统计：**
   七日内上报最多的警告告警和故障告警、最近七日的警告告警数和故障告警数。
6. **设备组状态统计：**
   显示设备组中的设备总数、在线设备数、故障设备数、警告设备数。



点击设备组右上角的箭头进入设备状态页面，该页面主要查看该设备组所有设备的硬件状态、软件状态和子设备状态。如下图所示：



点击设备列表后进入该设备的监控的详细页面，该页面主要监控包括：

1. **硬件监控**
   - CPU 温度和使用率，以及 CPU 使用率的 TOP5 进程
   - 内存总量和使用率，以及内存使用率的 TOP5 进程
   - 网络上下行速率和总接受发送流量，以及网络详情
   - 磁盘的容量、使用率和健康度，以及磁盘各个分区的使用情况
   - GPU 信息，支持 GPU 资源监控
   - 电池余量、健康度和温度
2. **子设备监控**
   - 子设备的状态和信息

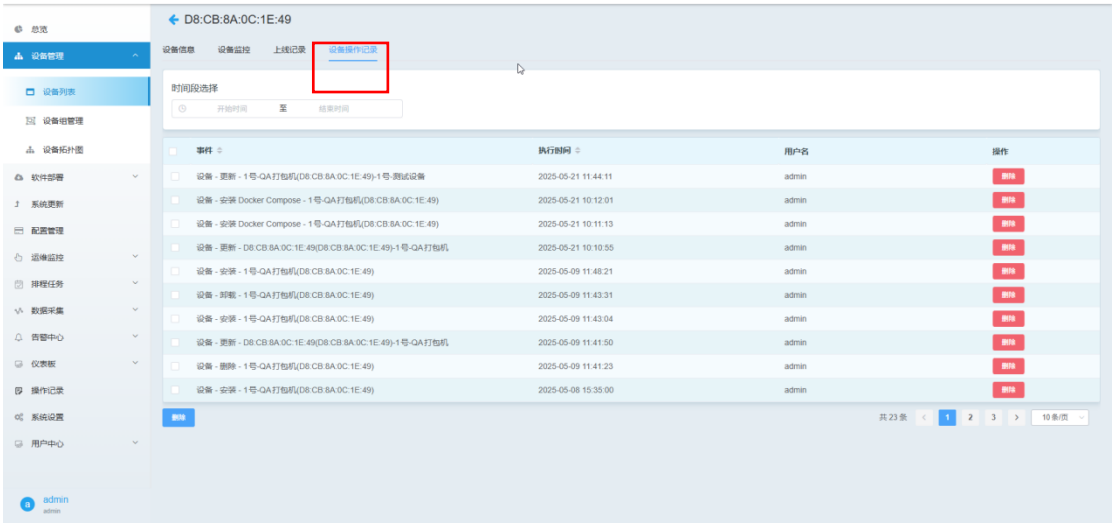
3. **软件监控**
   - Android 的 APP，或者 Windows/Linux 的进程，CPU 和内存使用情况的监控
   - docker 容器的 CPU 和内存监控
   - 远程重启、停止被监控的 APP 或者进程、以及 docker
4. **外设监控**
   - USB 外设连接的监控

除了上述监控，还可以在操作按钮处，对设备远程关机、重启、远程桌面、远程终端等。如下图所示：

3.5. 设备操作记录

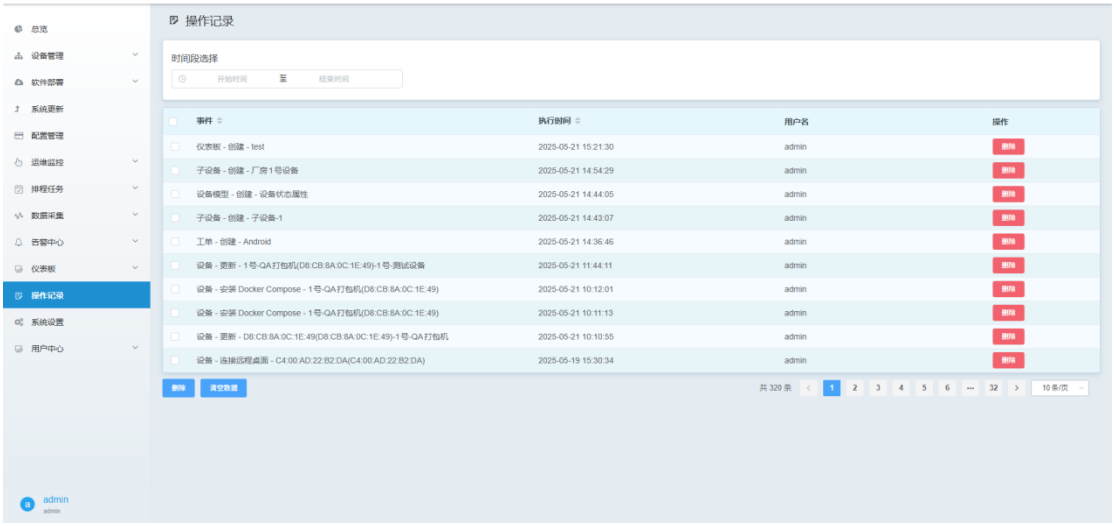
（1）查看指定设备的操作记录

如章节 3.1.2 描述，进入设备详情后，选择设备操作记录，即可进入设备操作记录页面。



（2）查看指定时间内的操作记录

导航栏【操作记录】点击筛选查看，如下图：



4. 设备 OTA 更新

4.1. 应用软件部署&更新

软件部署是的一个重要功能，当前我们支持 Linux/Windows/Android 各种软件安装包的部署&批量部署，以及普通文件的部署。我们希望通过我们提供的这种远程部署机制，能极大地简化实际生产环境中繁琐的部署工作。

不仅支持在线设备的软件更新，同时支持离线设备的软件更新，当离线设备再次上线后会自动执行更新任务。

下面我们将详细描述个部分内容。

4.1.1. 安卓 App 安装/更新/卸载

DeviceOn/EIM 支持 Android 设备的 App 安装、升级和卸载功能。其中安装顾名思义是安装新的应用到系统，升级是升级系统中已有的应用到更高版本，卸载是卸载系统中安装的第三方应用（系统预装的应用无法卸载）。

App 操作过程分为两步进行:

第一步上传要安装/升级 App 的安装包到 DeviceOn/EIM Repo 仓库;

第二步在 DeviceOn/EIM 页面执行安装/升级/卸载操作。

其中第一步 DeviceOn/EIM Repo 仓库的操作请参考后面第 13 章内容。本章节主要讲述第二步 DeviceOn/EIM 页面的操作。

操作步骤为：【安卓 App】-> 选择要部署的设备组&设备 ->【App 市场】 -> 点击安装/卸载/更新，如下图所示

在 App 市场页面中操作栏为“卸载”的 app 是目前设备已经安装的第三方应用，

操作栏为“升级“的 app 是目前设备已经安装但 DeviceOn/EIM 软件仓库有更高版本的第三方应用，

操作栏为“安装“的 app 是目前设备未安装的但 DeviceOn/EIM 软件仓库中有安装包的第三方应用。

点击相应操作后，页面系统会自动跳转到【App 安装历史】，该页面会显示安装的结果，如果成功会显示安装成功，否则，会提示失败和安装失败原因。如下图是 App 安装结果的示意图

上面描述的是向单个机器安装（升级、卸载）App 的情况，我们还支持向多个机器同时部署 App 的操作，称为批量部署。批量部署需要先将设备加入到指定的设备组，然后选择指定设备组(设备组概念请参考 3.2 章节)，就可以进行相关部署操作。部署时，【App 安装历史】中，每一个设备会有一个安装包部署成功/失败的记录，供用户查询。

4.1.2. Linux 包的安装/更新/卸载

针对 Linux 系统，DeviceOn/EIM 支持 3 种类型安装包安装/卸载/更新，分别描述如下：

4.1.2.1. Debian 包

当前页面针对一个 group/device，一次部署任务仅支持单个 debian 包的安装/卸载/更新。如果需要一个部署任务同时部署多个包，可以参考配置管理章节（第 5 章）。或者多次部署来完成。debian 包是 Linux 标准包，如何制作在此不再详述。

4.1.2.2. 包含 install.sh & uninstall.sh 的压缩包

如何部署压缩包是一个简单议题，我们会在下文详细描述，在这里，我们描述如何生成一个包含 install.sh & uninstall.sh 的压缩包。我们会通过以下步骤来描述如何生成一个 DeviceOn/EIM 支持的压缩包。当前 DeviceOn/EIM 支持 zip 和 tar.gz 的压缩包格式。

Step1:创建 test 路径 并在其目录下生成 install.sh and uninstall.sh

```
$ mkdir test && cd test
$ touch install.sh
$ touch uninstall.sh
```

Step2:复制安装内容进你 test 目录

```
$ cp ${your_install_content} test/
```

Step3:编写 install.sh 和 uninstall.sh

```
1. $ vi install.sh or uninstall.sh
2. #!/bin/sh
3.
4. ### implment your install code
5.
6. if result is success; then
7.     exit 0;
8. else
9.     exit 1;
10. fi
```

在脚本里实现了你自己的安装代码后，你应当使用 exit 返回安装代码的执行结果。如果脚本返回 0，表示安装成功，如果返回 1，表示安装失败。

Step4:压缩 test 目录生成目标压缩包

```
$ tar cvzf test_1.0.1.tar.gz install.sh uninstall.sh ${your_target_file}
#或者
$ cd ../
$ ls test
$ tar cvzf  test_1.0.1.tar.gz test
```

注意：这个安装包的名字应该遵循这种格式：${app_name}_${package_version}.{tar.gz|zip}。

4.1.2.3. 包含 Deploy.xml 的压缩包

同 4.1.2.2 章节一样，当前 DeviceOn/EIM 支持 zip 和 tar.gz 的该种压缩包格式。我们通过以

下步骤来描述。

Step1:创建 test 路径 并在期目录下生成 Deploy.xml

```
$ mkdir test && cd test
$ touch Deploy.xml
$ touch APP/install.sh
$ touch APP/post_install.sh
```

Step2:复制安装内容进你 test 目录

```
$ cp ${your_install_content} test/
```

Step3:编辑 Deploy.xml

```
1. <?xml version="1.0" encoding="utf-8" standalone="no"?>
2. <DeployDescription>     ##根节点
1. <DeploySetting>     ##根节点
3. <DeployFileName>APP/install.sh</DeployFileName> ## 安装脚本的相对路径
4. <RebootFlag>1</RebootFlag> ## 安装完成后，是否需要重启
5. <ResultCheckScript>APP/post_install.sh</ResultCheckScript>     ##     post
   install 安装脚本的相对路径 RebootFlag and ResultCheckScript 是可选的。
6. </DeploySetting>
7. </DeployDescription>
```

在脚本里实现了你自己的安装代码后，你应当使用 exit 返回安装代码的执行结果。如果脚本返回 0，表示安装成功，如果返回 1，表示安装失败。

Step4:压缩 test 目录生成目标压缩包

```
$ zip -r test_1.0.1.zip Deploy.xml APP
```

注意：这个安装包的名字应该遵循这种格式：${app_name}_${package_version}.{tar.gz|zip}。

4.1.2.4. Linux 包的上传&OTA 更新

1. Linux 包的上传

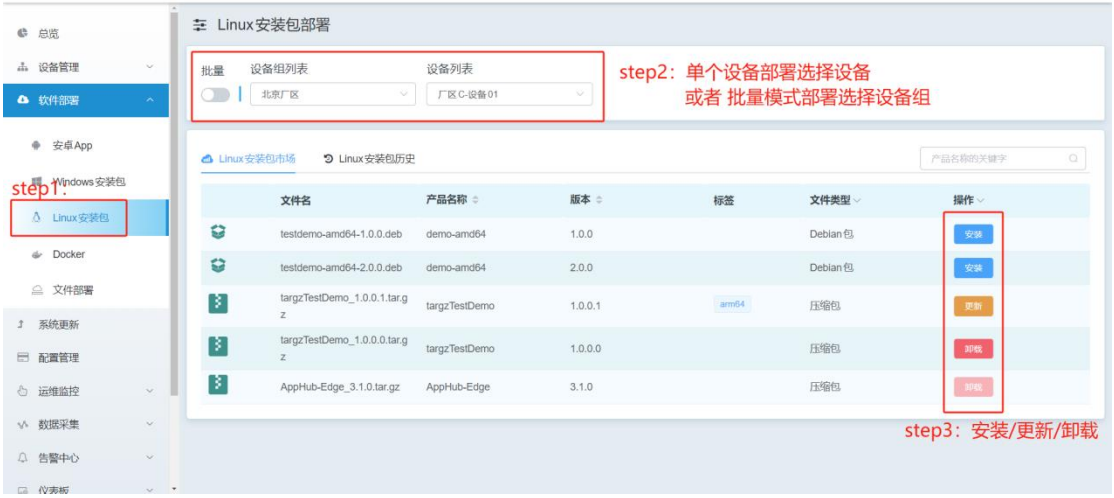
要实现包的部署，就需要将上面描述的压缩包和 debian 包上传至 DeviceOn/EIM 指定的仓库。为此我们已经实现了一个软件仓库来存储这些包，关于软件仓库上传 Linux 包的详细情况，请参考第 13.3.2 章节相关内容。

2. Linux 包的安装/卸载/更新

无论是 debian 包，还是上述的两种压缩包，都使用统一的界面进行安装，更新，删除。

操作步骤为：

【软件部署】->【Linux 安装包】-> 选择要部署的设备组&设备 ->【Linux 安装包市场】 -> 选择安装/卸载/更新。



点击相应操作后，页面系统会自动跳转到【Linux 安装包历史】，该页面会显示安装的结果，如果成功会显示安装成功，否则，会提示失败和安装失败原因。如果失败原因过于复杂，请参考 Client 的 log 进行判断。

注意：这里只能管理由 DeviceOn/EIM 安装的包，无法管理非 DeviceOn/EIM 安装的其他包。

上面描述的是向单个机器安装单个包的情况，我们还支持单个包向多个机器部署，称为批量部署。批量部署需要先将设备加入到指定的设备组，然后选择指定设备组(设备组概念请参考 3.2 章节)，就可以进行相关部署操作。部署结果会在【Linux 安装包历史】中，每一个设备会有一个安装包部署成功/失败的记录，供用户参考。

4.1.3. Windows 包的安装/更新/卸载

DeviceOn/EIM 支持两种类型的 Windows 包：Zip 压缩包和 exe 格式的安装包。

分别描述如下：

4.1.3.1. Zip 压缩包

Zip 压缩包的创建方法&注意事项与 Linux 的基本相同，请用户参考章节 4.1.2.2。只不过需要注意的是，压缩包中的脚本需要变成 Windows 下的批处理脚本，即 install.bat 和 uninstall.bat，或者 post_install.bat。其他注意事项基本和 Linux 相同。我们给出一个批处理文件的伪代码实现。

```
$ vi install.bat or uninstall.bat
### implment your install code

if result is success; then
    exit /B 0
```

```
6.  else
7.      exit /B 1
8.  fi
```

4.1.3.2. exe 包

当前只支持基于 Inno setup 和 Wix toolset 打包工具生成的 Windows 安装包，Inno setup 和 Wix toolset 都是开源的打包工具，具体打包步骤，请读者通过百度/Google 自行搜索研究学习，这里不再赘述。

4.1.3.3. Windows 包的上传&OTA 更新

Zip 压缩包和 exe 包的部署操作和 Linux 完全相同，具体操作方法如下：
【软件部署】->【Windows 安装包】->批量/单个模式选择->设备组/设备选择->软件包的选择->安装/升级/卸载。

4.2. Docker 的安装/更新/卸载

Docker 是一个用于开发、发布和运行应用程序的开放平台。Docker 能够将应用程序与基础设施分开，以便快速交付软件。使用 Docker，可以以管理应用程序的方式管理应用程序的基础环境。通过利用 Docker 快速搭建、测试和部署代码，可以显著地缩短产品的交付周期。Docker 容器使用本地容器的方式使生产环境和开发环境保持完全一致，从而大大简化了开发生命周期。容器也非常适合连续集成和连续交付（CI/CD）的情景。Docker 通常有以下应

用场景：

1. 开发者和其他同事共享开发环境；
2. 开发完成后，他们通过 Docker 来模拟生产环境进行相关测试等；
3. 发现 bug 后，他们在开发环境修复问题，并将开发环境使用 Docker 导入测试环境进行测试；
4. 测试完成后，更新 Docker image 并发布给客户，客户通过简单的 Docker image 更新就可完成生产环境的更新。

DeviceOn/EIM 当前支持两种基于 Docker 技术的部署&更新，即：Docker compose 部署和 Docker swarm 部署。接下来我们将分别描述之。

4.2.1. 建立 Docker 环境

Docker 部署要求对应的目标环境里（Client 所在环境）存在 Docker，对于 ubuntu/debian 等 Linux 环境，都可以通过以下命令安装 Docker 官网提供的最新 Docker 版本：

```
# 确保老版本 Docker 被移除
$ apt-get remove docker docker-engine docker.io containerd
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
```

为了提高兼容性，建议用户安装最新版本 Docker，笔者当前使用的 Docker 环境是 20.10.17，当然我们还测试了 18.xxx，19.xxx 等版本，都是兼容我们的 Docker 远程部署。关于其它安装细节，用户可以参考官方连接 https://docs.docker.com/engine/install。

Windows 环境安装 Docker 请参考官网连接：

https://hub.Docker.com/editions/community/docker-ce-desktop-Windows 。

无论是 Docker compose 还是 Docker swarm 都需要安装 Docker 环境。

4.2.2. Docker Compose 部署

Docker compose 是一个针对单主机的功能强大的容器编排工具。为了支持 Docker compose 的功能，用户应当在目标环境里（EIM Client 所在环境）安装 Docker compose。

如何安装请参考 Docker 官网 https://docs.docker.com/compose/install/

Compose 环境搭建好后， 就可以进行相关部署了。部署只需要提供 Compose 的 yaml 文件，具体语法请参考 Docker 官网 https://docs.docker.com/compose/compose-file/

我们下面给出几个例子来说明：

```
1.  version: '2'
2.
3.  services:
4.       alpine:
5.          image: arm64v8/alpine:latest
6.          restart: always
7.          container_name: alpine
8.          networks:
9.            - advnet
10.
11. networks:
12.     advnet:
13.       driver: bridge
```

该 compose 提供了一个名为 alpine 的 service，该 service 包含一个 Docker 容器，Docker image 为 arm64v8/alpine:latest。

我们再看一个例子:

```
1.  version: '3'
2.  services:
3.      mqtt:
4.         image: 00718/Androiddm-mosquitto-dev:v1.0
5.         restart: always
6.         ports:
7.                - "1883:1883"
8.                - "11880:11880"
9.         container_name: m2m_mosquitto
10.        networks:
11.             - advnet
12.     db:
13.        image: postgres:9.4
14.        restart: always
15.        environment:
16.               POSTGRES_PASSWORD: postgres
17.               POSTGRES_DB: aimlink
18.        ports:
19.               - "5432:5432"
20.        container_name: m2m-postgresSQL
```

```
21.        networks:
22.              - advnet
23. networks:
24.     advnet:
25.          driver: bridge
```

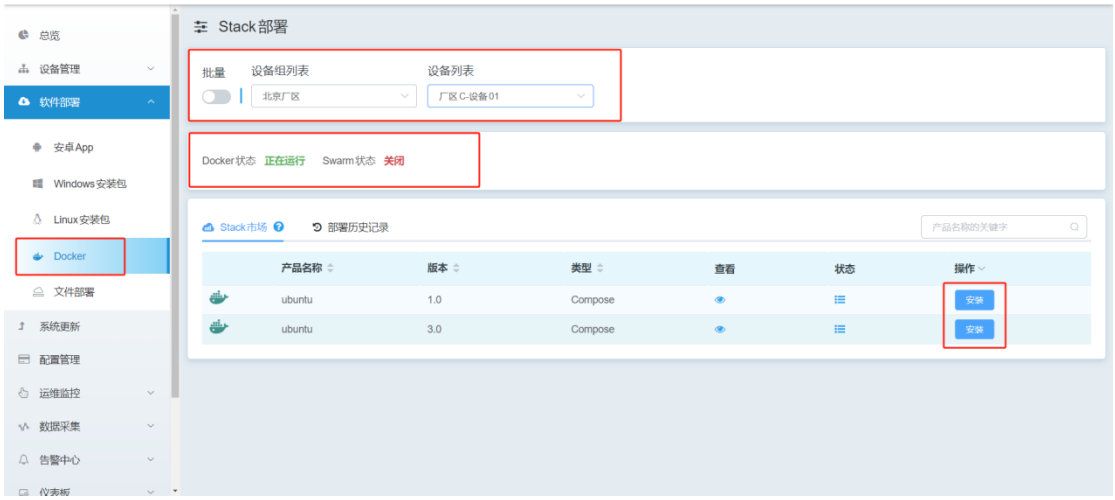
该 compose 支持两个 service，db 和 mqtt，mqtt 的 image 为 00718/Androiddm-mosquitto-dev:v1.0，它的对外端口为 1880 和 1883，同时这个 service 使用了由 compose 创建的名为 advnet 的网桥。用户完成自己的 compose yaml 后，就可以进行 Compose 部署了

step1：Docker compose 的上传

要实现 Compose 的部署，就需要将 Compose 的 yaml 文件上传至指定的仓库。为此我们已经实现了一个软件仓库来存储这些文件，关于软件仓库上传 Compose 的 yaml 的详细情况，请参考第 13 章相关内容。

step2：Docker compose 的部署

Docker swarm 和 Docker compose 使用相同的部署页面，操作方法&流程也基本一致。选择【软件部署】->【Docker】即可进入 Docker compose/swarm 部署界面，如下图:



进入部署页面后，首先需要选择部署的设备组和设备，页面此时就会显示该目标机器上 Docker 的运行状态和 swarm 的状态。如果 Docker 没有正常运行，请用户检查目标机器的 Docker 安装情况。

注意:只有 swarm 处于关闭状态下，Compose 才能正常使用！ 他们总是互斥的。

选择【Stack 市场】就可进入 Compose 部署/更新页面，在这里你可以选择安装/更新/删除部署，点击相应操作后，页面系统会自动跳转到[部署历史记录]页面，该页面会显示安装的结果，如果成功会显示安装成功，否则，会提示失败及失败原因。

部署完成后也可以查看相关部署里对应容器的运行状态，也可以查看对应部署 Compose yaml 文件，但不能修改。

上面描述的是向单个机器安装单个 Compose 的情况，我们还支持单个 Compose 向多个机器部署，称为批量部署。批量部署需要先将设备加入到指定的设备组，然后选择指定设备组

(设备组概念请参考 3.2 章节)，就可以进行相关部署操作。部署后，部署历史记录页面中，每一个设备会有一个安装部署成功/失败的记录，供用户参考。

4.2.3. Docker Swarm 部署

Docker Swarm 是一个针对多主机集群的功能强大的容器编排工具。为了支持 docker swarm 除了 docker 本身，你不需要安装其他额外的工具。但需要构建起一个 docker swarm 集群，或者至少需要提供一个 docker swarm manager 节点。关于如何构建一个 docker swarm 集群，请读者自行查询相关资料研究，这里不再赘述。

Docker swarm 的部署也需要 docker swarm 的 yaml 文件，它的格式和 docker compose 文件的语法和格式基本一致，只不过 Swarm 的 yaml 必须是 Compose yaml 的版本 3 及以上，Compose yaml 的 3 以下版本，是不支持 swarm 的。关于 Swarm 的 yaml，我们给出几个例子来说明:

```
1.   version: '3'
2.  services:
3.    redis:
4.      image: redis:alpine
5.      deploy:
6.        replicas: 6
7.        update_config:
8.          parallelism: 2
9.          delay: 10s
10.       restart_policy:
11.         condition: on-failure
12.
```

该 swarm 提供了一个名为 redis 的 service ， 他有 6 个实例，如果 task 运行失败 service 将会重启。我们再看一个实例:

```
1.  version: "3.3"
2.
3.  services:
4.    wordpress:
5.      image: wordpress
6.      ports:
7.        - 8080:80
```

```
8.      networks:
9.        - overlay
10.     deploy:
11.       mode: replicated
12.       replicas: 2
13.       endpoint_mode: vip
14.
15.   mysql:
16.     image: mysql
17.     volumes:
18.       - db-data:/var/lib/mysql/data
19.     networks:
20.       - overlay
21.     deploy:
22.       mode: replicated
23.       replicas: 2
24.       endpoint_mode: dnsrr
25.
26. volumes:
27.   db-data:
28.
29. networks:
30.   overlay:
```

该 swarm 有两个 service，wordpress 和 mysql，它们又各有两个实例。

以上就是 Docker swarm yaml 的 demo，细节可以参考相关 docker 的官方文档。Swarm yaml 文件的上传和部署，和 Compose 完全相同，这里不再赘述。

4.3. 普通文件部署

Linux、Windows、Android 均支持普通文件部署。同压缩包一样，需要将文件上传至指定的仓库，才能进行相关文件部署。关于更详细的上传文件描述，请参考第 13 章相关内容。选择【文件部署】->【文件市场】，选择部署的设备组&设备，即可进行部署。

- **部署普通文件**

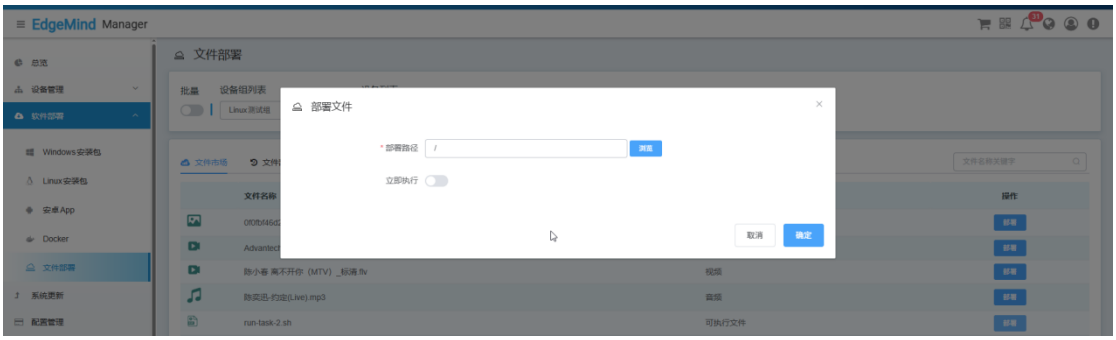
如上图所时，也支持文件的批量部署，即同一个文件部署到一个设备组里的所有设备上。部署后都会有相应的操作结果位于文件部署历史，供用户参考。

注意：同一个文件被部署到相同路径后，之前的同名文件会被覆盖。

- **部署可执行文件**

如果不勾选立即执行，那只是将可执行脚本或可执行文件远程部署到设备端；

如果勾选立即执行，那么设备端获取到文件后就会立即执行它；



4.4. 系统 OTA 更新

系统 OTA 更新是系统运维的一个关键点，DeviceOn/EIM 提供了一套简单、灵活的 OTA 更新机制，方便用户快速，批量地对大量设备做 OTA 更新。我们当前支持 Android、Linux、Windows 平台的系统 OTA 更新，涉及 x86/ARM 等多个平台。

4.4.1. Linux 系统 OTA 更新

当前我们已经支持 AdvLinuxTu, 赋华 rk3399 Debian/Ubuntu, 赋华 iMX8m yocto, 基于 jetson

的 ubuntu（赋华版）等平台，如果用户使用的是这些平台，就不需要再做适配，否则，请参考章节 4.4.1.3 进行适配。

4.4.1.1. OTA 系统更新包的制作&上传

我们对于 OTA 包的格式要求相对宽松，不同的平台可有不同的打包格式，但是 OTA 更新文件名需要满足以下格式：

```
1.  ${boardName}-\${version}.\${suffix}
```

boardName：必须和 EIM Client 获取的 boardName 一致；

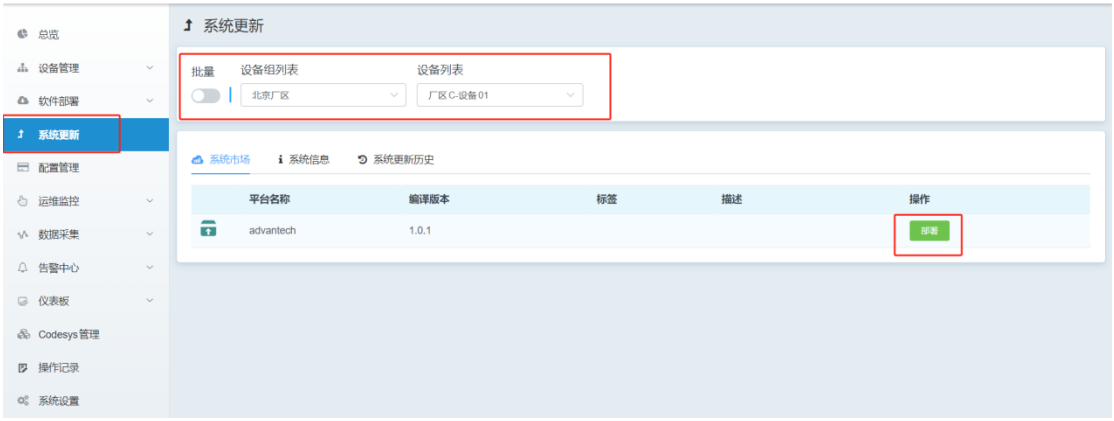
version：OTA 更新 image 的版本；

suffix：OTA 更新 image 的格式后缀，当前没有限制；

4.4.1.2. OTA 系统更新

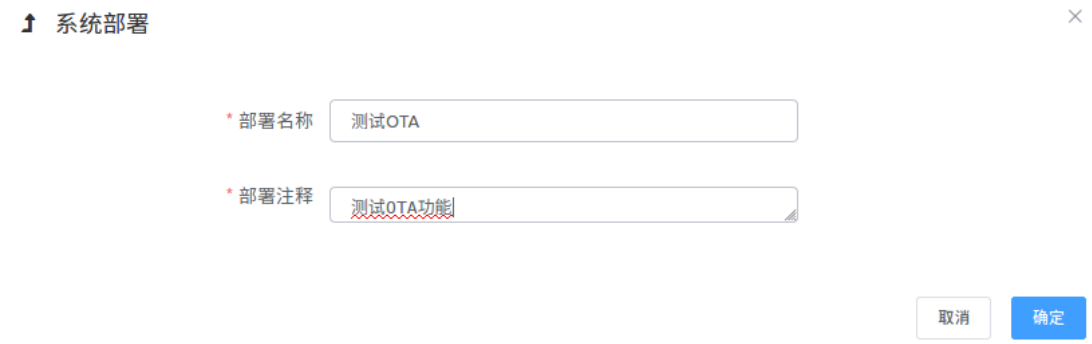
和上述安装包一样，OTA 更新包也需要上传至指定的仓库，才能进行 OTA 系统更新。关于更详细的上传 OTA 更新 image 描述，请参考第 13 章相关内容。

完成 OTA 包上传后，选择【系统更新】后，就可以进入系统 OTA 更新界面了。



选择目标设备组&设备，点击 OTA 更新部署按钮，就可进行 OTA 系统更新。

如过需要批量部署，也可以打开批量部署的开关，选择批量部署的设备组。当点击了部署按键后会弹出对话框，需要填写部署名称和注释，如下



点击确认按键后，开始正式部署更新。在更新的过程中，页面会跳转到[系统更新历史]，通过显示相关信息来报告 OTA 更新所处的阶段。由于系统 image 一般比较大，所以 OTA 的下载&更新都会花费较长时间，更新完成后，会显示对应的 OTA 更新结果，以供用户参考。一般地，目标系统 OTA 更新完成后，如果更新成功，都会重启系统，进入更新后的系统，这一阶段，EIM Client 就会有一段时间处于离线状态，该现象为正常现象。等待目标系统更新成功后，用户可以通过[系统信息]页面，获取新系统的相关版本信息。

4.4.1.3. DeviceOn/EIM 新平台适配 OTA

一般地，一个新平台，DeviceOn/EIM 为了支持其系统 OTA 更新，需要满足以下三个条件：

- **目标系统必须支持 recovery 或类 recovery 的系统更新机制；**

DeviceOn/EIM 只是一个 OTA 更新框架，它无法也不可能实现具体的系统更新功能，所以需要 recovery 或者类 recovery 的本地系统更新机制来配合。

- **EIM Client 必须安装在系统更新碰触不到的分区里；**

如果 EIM Client 安装在系统更新包含的分区内，系统更新后，该分区被重写，EIM Client 就不复存在，就无法正常提供 OTA 功能。

- **系统必须存在一个分区能容纳 OTA 更新包；**

如果系统没有足够的空间，将无法支持 OTA 功能。

如果满足以上条件，请按照以下步骤来为 DeviceOn/EIM 的 OTA 系统更新添加一个新平台支持。

Step1:提供一个该系统本地更新的脚本，并将其放入 EIM Client 的安装根路径。

该脚本的名字为 ota_update.sh，Windows 下为 ota_update.bat，你应当在这个脚本里做实际的本地系统更新的所有事情。我们给出该脚本的伪代码实现：

```
1.  #/bin/bash
2.  image_path=${1}
3.  ${update_tool} ${image_path} ${other_args}
```

```
4.  if success then
5.     exit 0
6.  else
7.    exit result # !0
8.  fi
```

step2:提供一个获取系统更新的脚本，并将其放入 EIM Client 的安装根路径。

该脚本名为 get_result.sh，Windows 下为 get_result.bat。用于获取系统更新的最终结果，如果没有提供该脚本，我们总是默认系统更新是成功的。我们给出该脚本的伪代码实现：

```
1.  #/bin/bash
2.  ${update_tool} ${get result} ${other_args}
3.  if sucess then
4.    echo ok # or echo success
5.  else
6.   echo failed
7.  fi
```

如果该脚本向标准输出输出了“”ok”或者“success”，我们总是认为 OTA 系统更新是成功的，否则，则认为是失败的。

Step3:至此就完成了一个新平台适配 DeviceOn/EIM OTA 更新的所有步骤。

4.4.2. Windwos 系统 OTA 更新

Windows 系统 OTA 的流程基本和 Linux 一致，由于系统 OTA 需要 OS 层面的支持，WIndwos 系统 OTA 更新相关，请先确保可以进行本地系统更新，然后便可以参照以上章节所描述进行 DeviceOn/EIM 支持的 Windwos 系统 OTA。

4.4.3. Android 系统 OTA 更新

DeviceOn/EIM Android OTA 功能的前提是设备要支持本地 OTA 功能。。

DeviceOn/EIM Android OTA 操作过程分为两步进行：

step1：上传 Android 系统更新包到 DeviceOn/EIM Repo 仓库

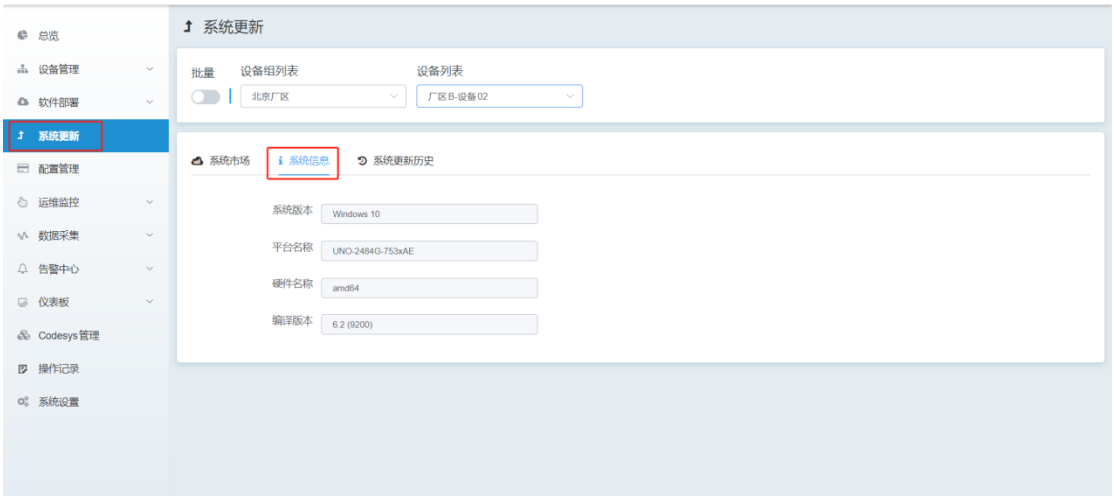
DeviceOn/EIM Repo 仓库的操作请参考后面第 13 章节内容

step2：在 DeviceOn/EIM 页面执升级操作。

操作步骤为：

【系统更新】->选择要部署的设备组&设备->【系统市场】 -> 选择更新

查看当前系统版本信息



开始部署

选择目标设备组&设备，点击 OTA 更新部署按钮，就可进行 OTA 系统更新。

如过需要批量部署，也可以打开批量部署的开关，选择批量部署的设备组。当点击了部署按键后会弹出对话框，需要填写部署名称和注释，如下

点击确认按键后，开始正式部署更新,然后就可以查看更新结果和历史，在更新历史记录里展示，如下图:

5. 配置管理

配置管理的目标是通过一个操作完成多个任务的部署，并且配置中的任务是可以保存起来的，在需要的时候可以进行再部署。配置的类型主要有 Android 配置、Linux 配置、Windows 配置。Android 配置的内容有 Android App、Android Kiosk、开机动画包、系统安全配置；Linux 配置的内容有 Docker compose&swarm 文件、tar&zip&deb 包；Windows 配置的内容有 exe 包和 zip 包。

5.1. 配置的创建

用户根据需求创建一个指定类型的配置。包含 Android 类型配置、Linux 类型配置和 Windows 类型配置。如下图所示：

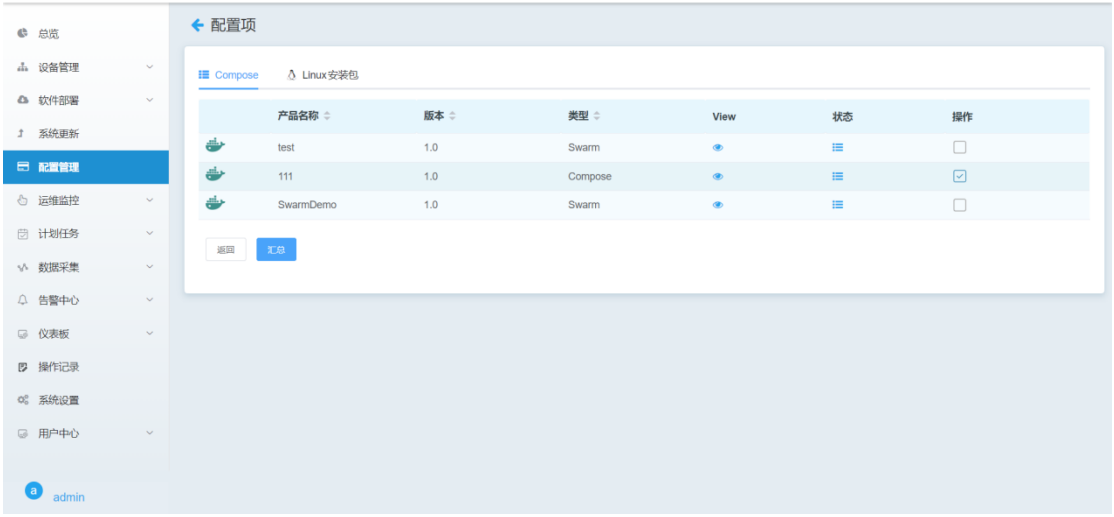
创建成功后会自动跳转到添加配置项页面，在该页面用户可以将需要的配置项添加到配置中，最终生成符合自己需求的配置。如下图所示：

Android 类型配置的配置项页面：

可以将安装多个 apk、设置 kiosk、设置开机动画和系统安全设置配置为一个任务清单，一次任务完成所有部署操作。

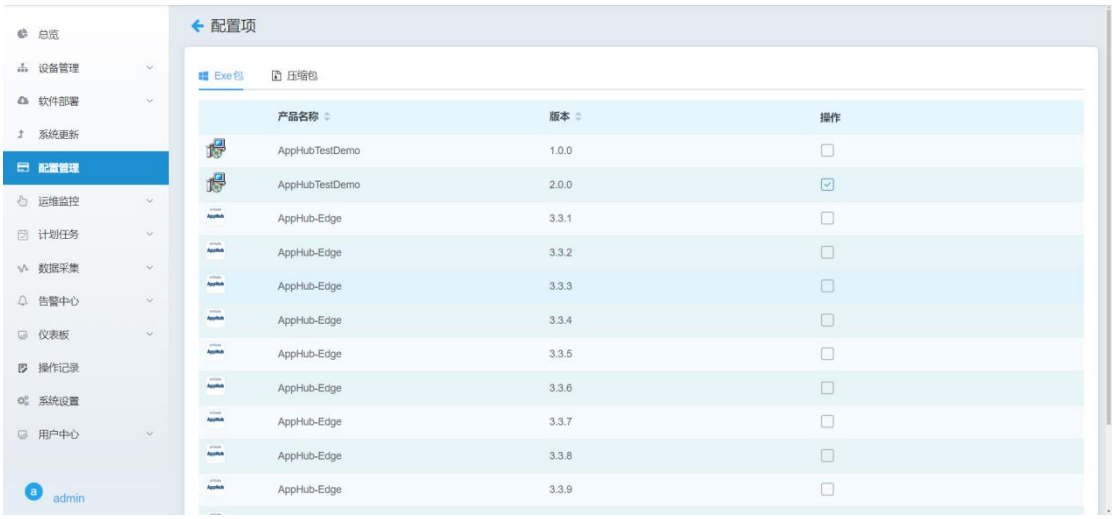
Linux 类型配置的配置项页面：

可以将部署安装 docker compose/swarm 和安装 deb 包、压缩包多个部署动作配置在一个任务清单，一次任务完成所有部署操作。



Windows 类型配置的配置项页面：

可以将部署安装 exe 安装包和安装压缩包多个部署动作配置在一个任务清单，一次任务完成所有部署操作。

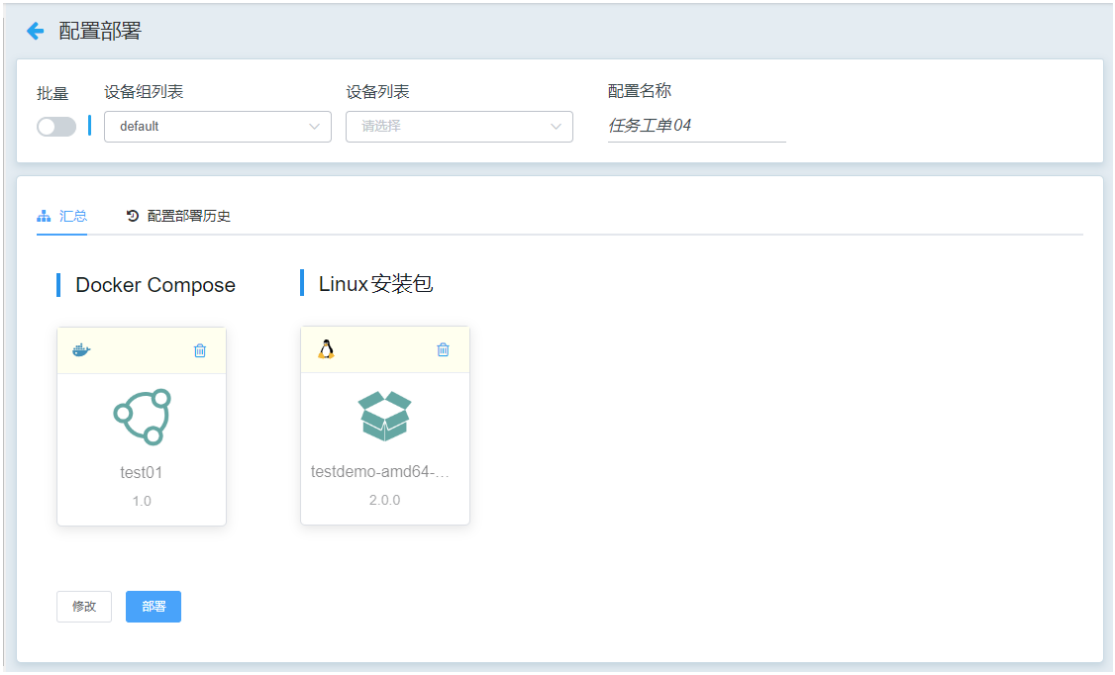


5.2. 配置的部署

配置项添加完成后点击汇总按钮进入配置汇总页面，该页面主要是部署配置到指定设备，完成对设备的相关配置。如下图所示：

Android 类型配置的部署页面：

Linux 类型配置的部署页面:



Windows 类型配置的部署页面：

另外需求注意的时候，DeviceOn/EIM 配置批量部署可以支持离线部署，即，

如果在选定的组中，有设备当时没有在线，也可以部署，当然，真正执行部署的时候，设备肯定是要上线的，这里所讲的离线部署，是指在选择组进行批量部署的时候，这时其中有些设备可以是离线的，DeviceOn/EIM 会知道哪些设备是离线的，所以它不会立刻执行部署，而是当下次离线的设备上线时，会自动执行这个部署。

所以在工单批量部署时，当切换【支持离线部署】功能后对于在线的设备，会立刻执行部署，对于离线的设备，会等设备下次上线时再自动执行部署。如下图：


总览
设备管理
设备列表
设备组管理
设备拓扑图
软件部署
系统更新
配置管理
配置部署
批量
设备
汇总
Docke
配置部署
* 部署名称
请输入部署名称
支持离线部署
取消
确定

6. 设备运维与监控

6.1. 监控设置

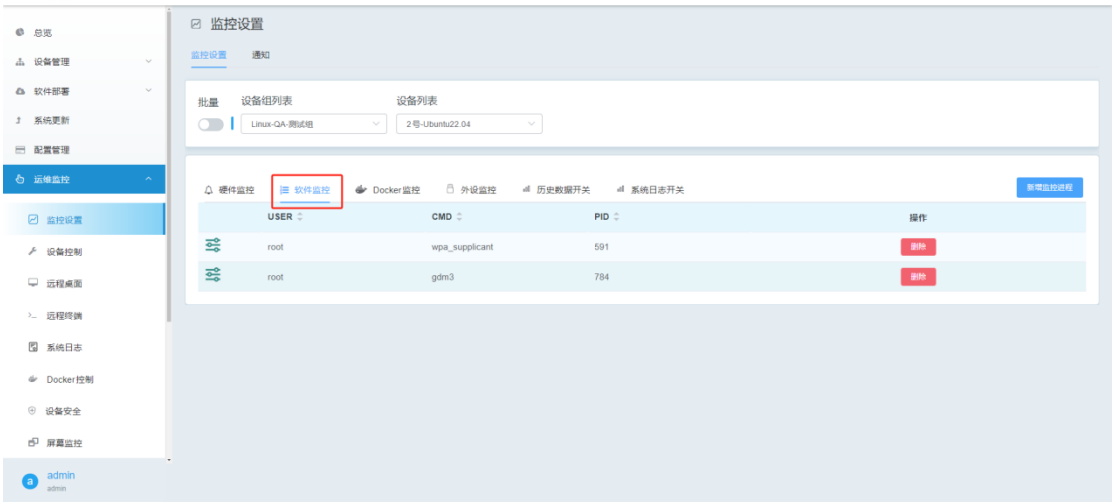
6.1.1. 硬件资源监控

目前，DeviceOn/EIM 支持的硬件监控有 CPU 温度、CPU 使用率、内存使用率、磁盘使用率、磁盘健康度、网络上下行速度和电池余量的监控告警，设置阈值并开启告警开关后，当资源状态达到阈值范围后，会主动告警通知。Linux、Windows 和 Android 的硬件阈值设置方法一致，硬件监控设置阈值如下图：

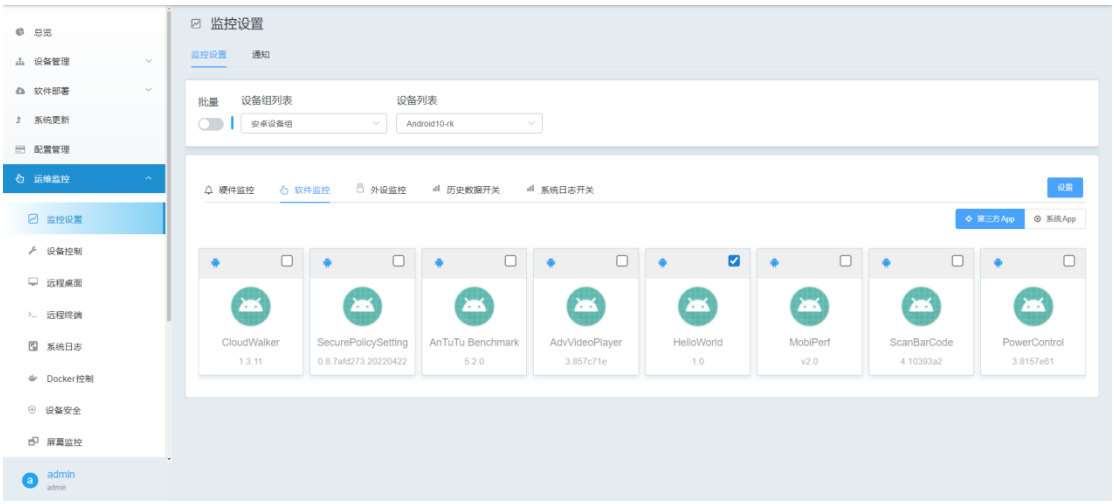
6.1.2. 软件进程监控

通过点击【新增监控进程】，获取当前系统运行的所有进程，勾选监控对象，将其加入监控列表后，如果被监控的软件状态发生变化，则会主动告警通知，如果不需要监控了也可以将其从软件监控列表中删除掉。

Linux 和 Windows 设置方法如下图所示：

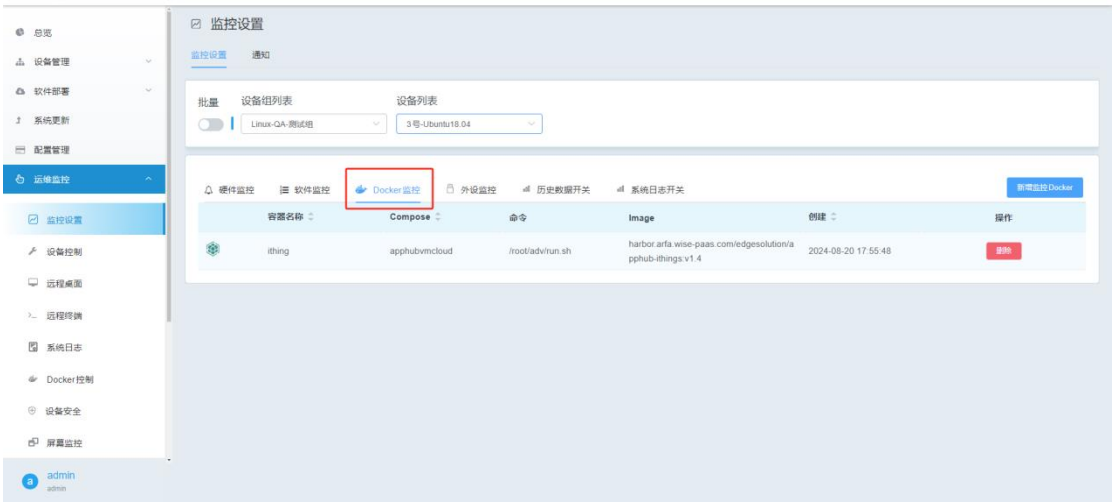


Android 的 apk 监控直接在列表中勾选即可将其加入监控列表，如下图:



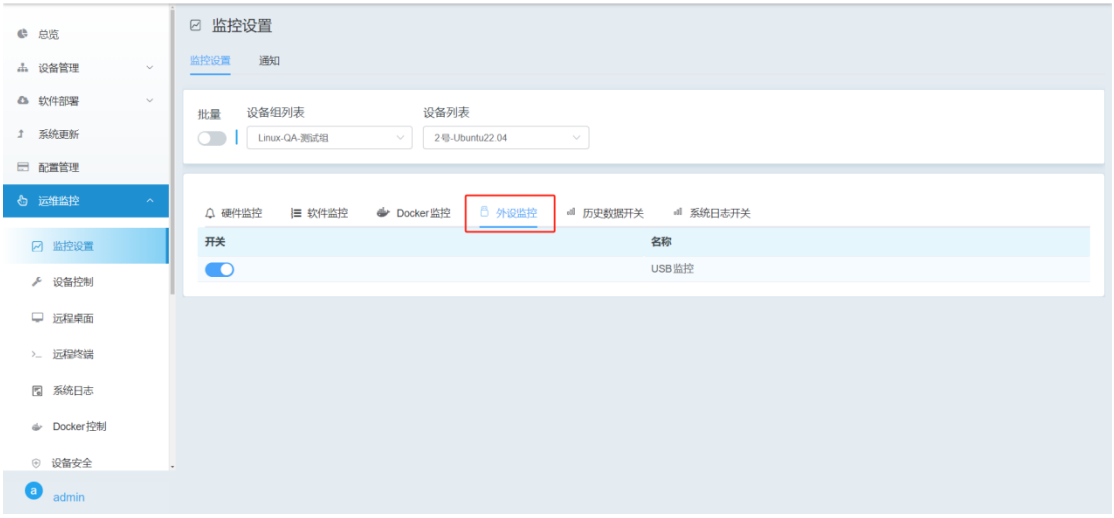
6.1.3. docker 容器监控

通过点击【新增监控 docker】，获取当前系统运行的所有容器，勾选监控对象，将其加入监控列表后，如果被监控的容器状态发生变化，则会主动告警通知，如果不需要监控了也可以将其从监控列表中删除掉。如下图所示:



6.1.4. 外设监控

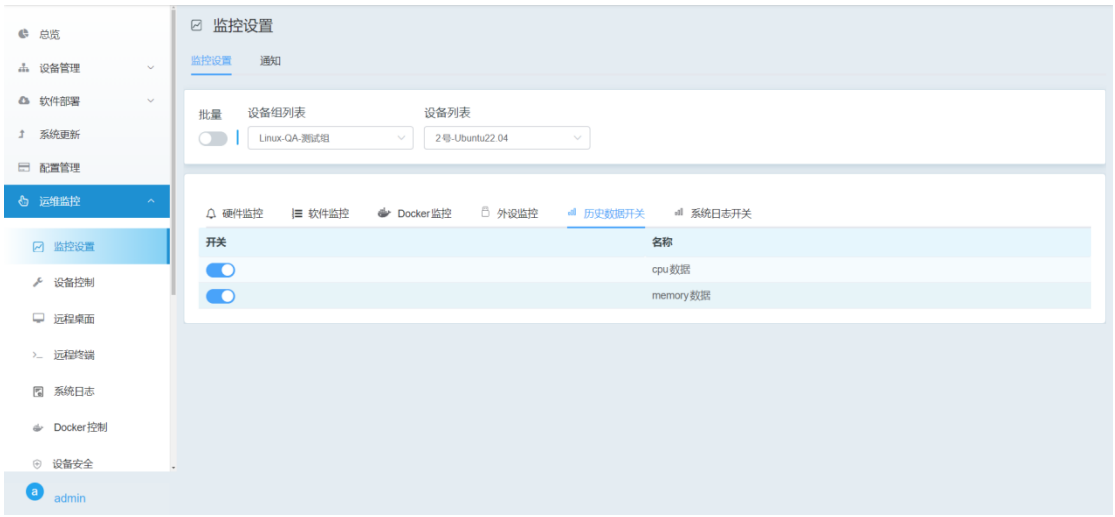
通过开启或关闭外设 USB 监控开关，可以根据需求设定是否有 USB 外设插入式需要报警通知。



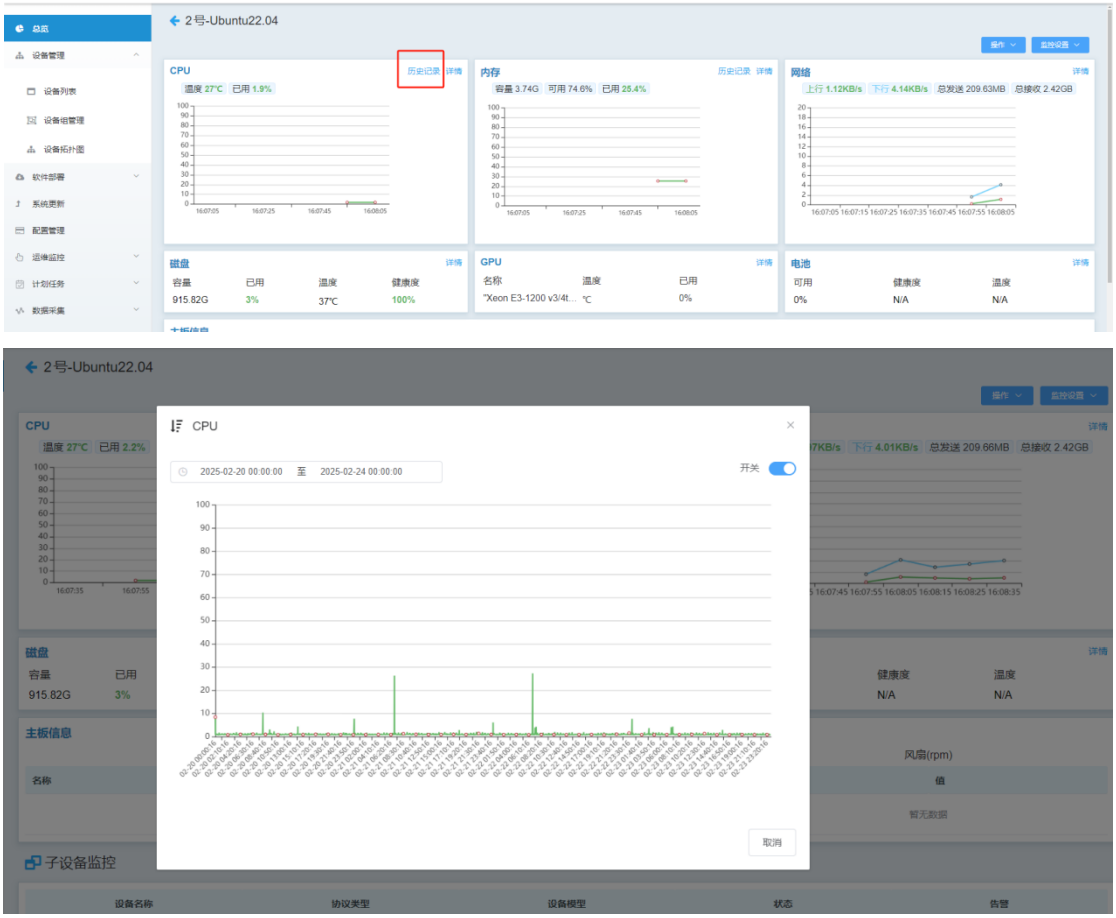
6.1.5. 历史数据开关

当开启数据开关后，CPU/memory 等监控过程中的数据会被保存起来，方便查看历史过程。

默认情况下是关闭的。



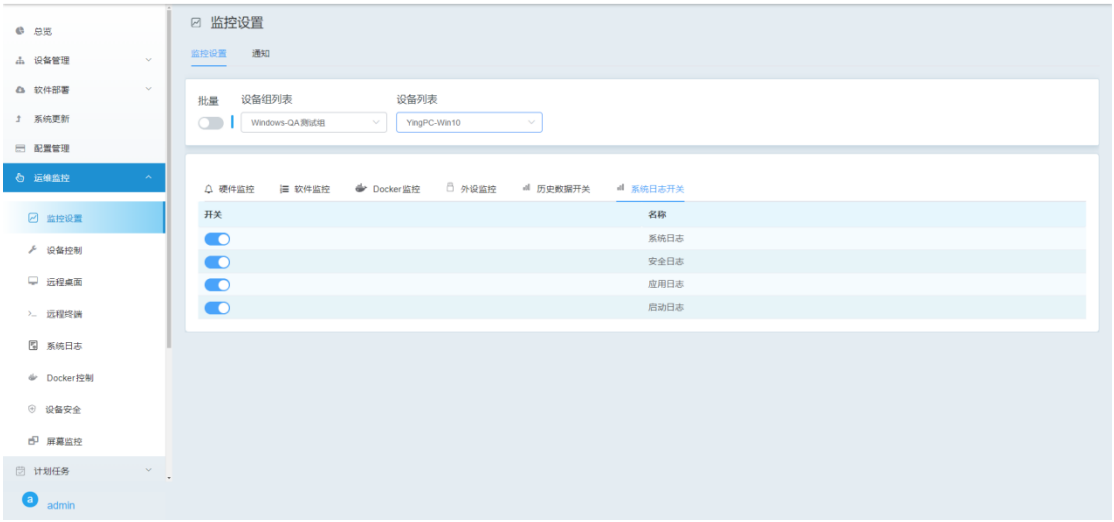
历史数据开启后，就可以在设备详细监控页面的 CPU/memory 曲线图查看历史，如下图所示：



6.1.6. 系统日志开关

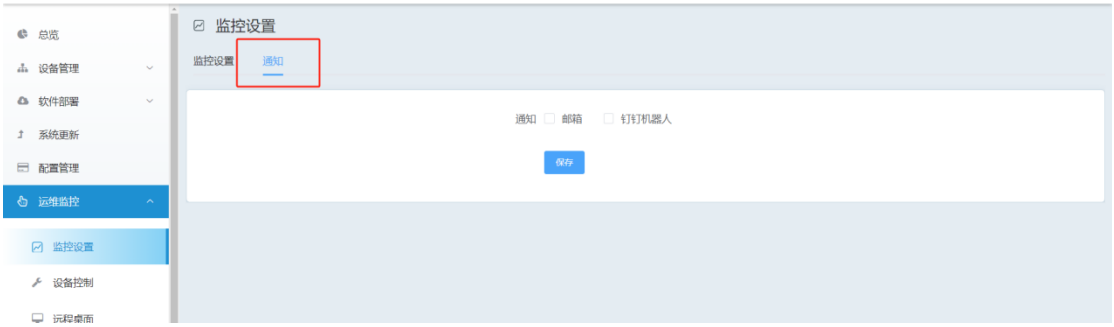
开启系统日志开关后，Client 会主动将设备上的系统日志抓取并上报给 Server，Server 端会记录保留，方便远程进行设备状况追踪。

默认情况下是关闭的。

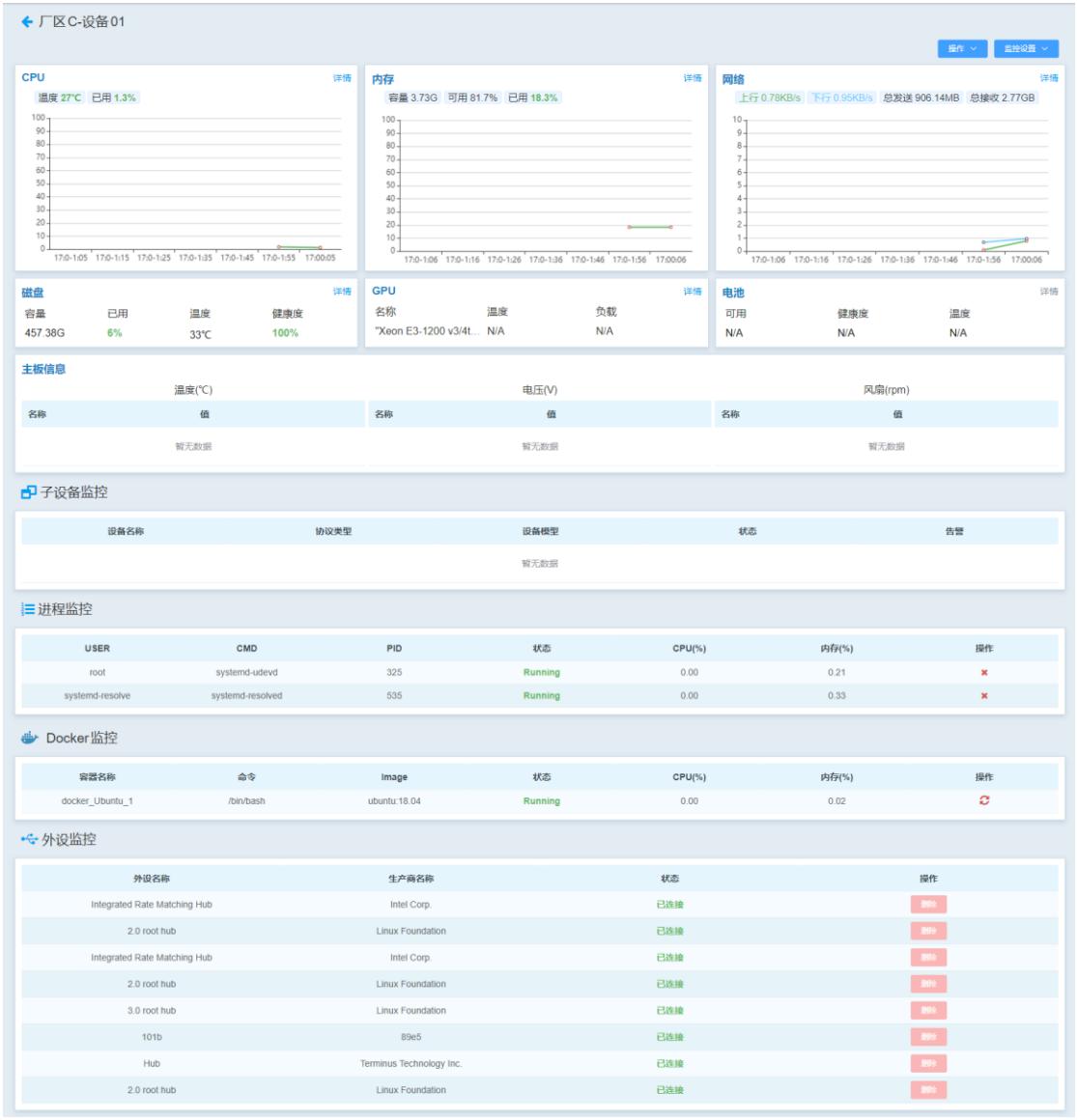


6.1.7. 告警通知

如果需要将软硬件的告警信息通过邮件或者钉钉的方式实时提醒，请在这里勾选：



设置完成后，在设备监控详细信息页面可以看到所有监控内容，如果软件进程或者 docker 容器异常停止，在这里还可以远程重启软件进程或者 docker 容器，如下图所示：



当监控对象的状态发生变化时，Client 会主动上报消息并提供实时通知。此外，您还可以在告警中心查看报警历史记录， 如下图所示：

6.2. 设备远程控制

远程控制主要是远程给设备下发一些指令起到远程控制设备的目的。

DeviceOn/EIM 远程控制功能分为两种情况：

情况一：单次控制即发送单次指令，主要执行一些系统设置相关的命令如：设备重启、设备关机、消息推送、媒体音量控制、背景亮度控制、设备绑定控制、命令发送等；

情况二：定时任务，即在某个时间点触发指令，可以是单次触发，也可以是重复触发。

6.2.1. 远程设置控制

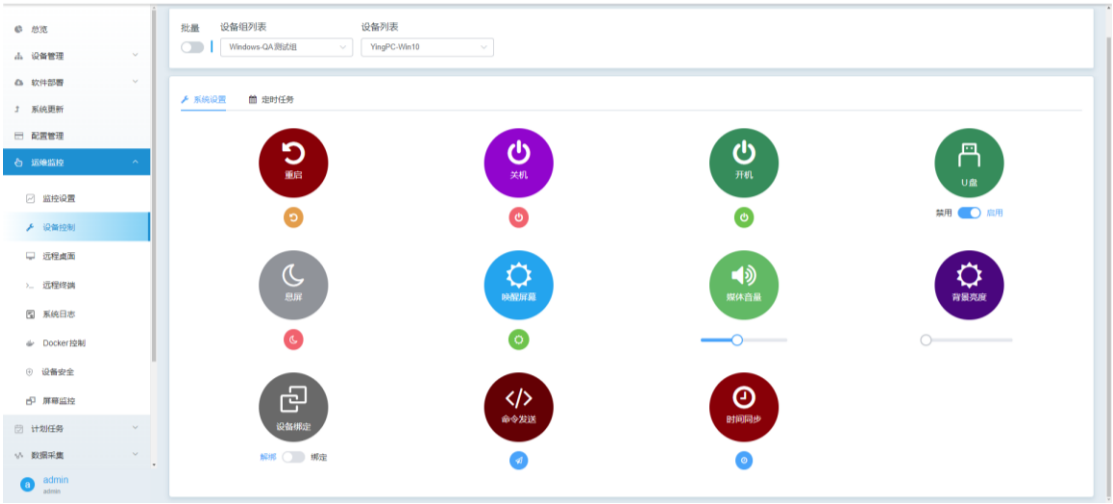
系统设置主要是对系统的一些属性或行为进行远程设置。因为系统差异所以不同系统支持的功能也会有差异，因此我们把功能分为三类介绍。

第一类是通用功能，如重启，关机，媒体音量，背景亮度，设备绑定，时间同步等；

第二类是 Windows/Linux 设备特有功能，如命令发送；

第三类是 Android 系统特有功能，如消息推送

下面图片是功能的操作页面。



下面详细介绍各个功能的含义和用法。

1. **通用功能**

- 重启

功能描述：重启远程设备的操作系统。

使用方法：点击重启图标下面的控制按键，然后在弹出的确认对话框中点击确认即可。

- 关机

功能描述：关闭远程设备的操作系统。

使用方法：点击关机图标下面的控制按键，然后在弹出的确认对话框中点击确认即可。

- 开机

功能描述：局域网内网络唤醒开机。

使用方法：点击开机图标下面的控制按键，然后在弹出的确认对话框中点击确认即可。

注意：这个功能需要硬件设备的支持。

- U 盘是否禁用

功能描述：当切换开关为“禁用”状态，Client 设备端 U 盘插入是无法识别无法正常使用的；当切换开关为“启用”状态，Client 设备端 U 盘插入后可以正常使用。

- 息屏/亮屏

功能描述：关闭屏幕或者唤醒屏幕

使用方法：点击“息屏”或者“唤醒屏幕”图标下面的控制按键，然后在弹出的确认对话框中点击确认即可。

- 媒体音量

功能描述：设置远程设备的多媒体音量。

使用方法：调节媒体音量图标下面的进度条来控制音量即可。

- 背景亮度

功能描述：设置远程设备的屏幕亮度。

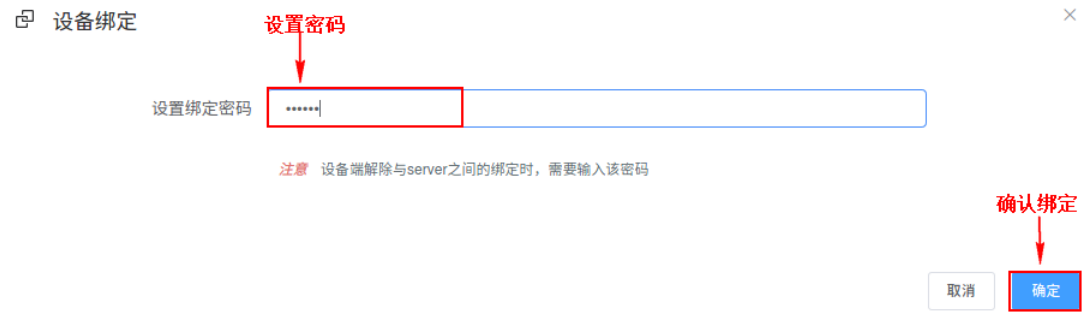
使用方法：调节背景亮度图标下面的进度条来控制屏幕亮度即可。

- 设备绑定

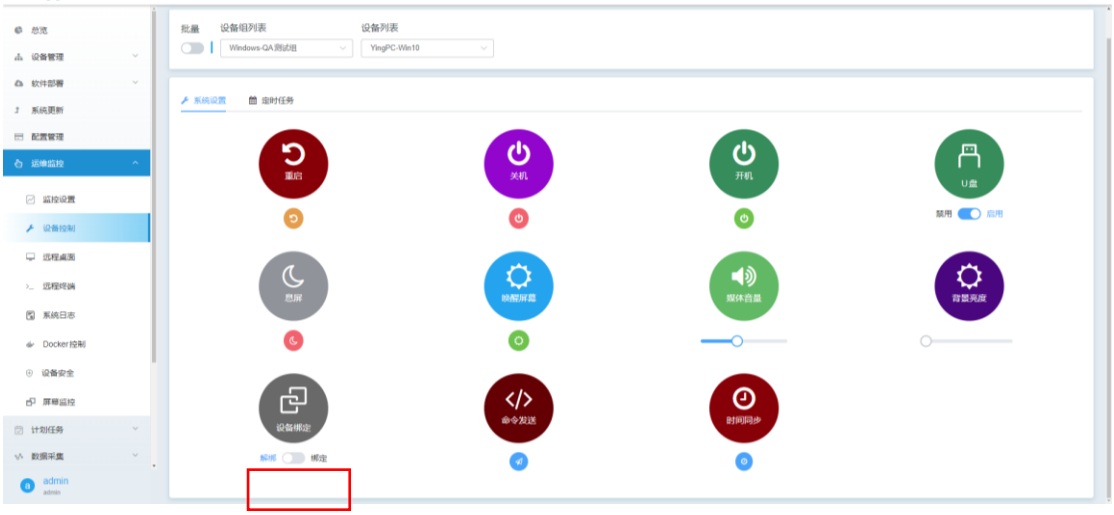
功能描述：将设备绑定到该 Server，绑定之后设备就无法被切换到其他 Server 除非先从本 Server 解除绑定。

使用方法：使用分为绑定和解绑，而且解绑又分为从 Server 端实现解绑和从设备端实现解绑。下面我们详细介绍。

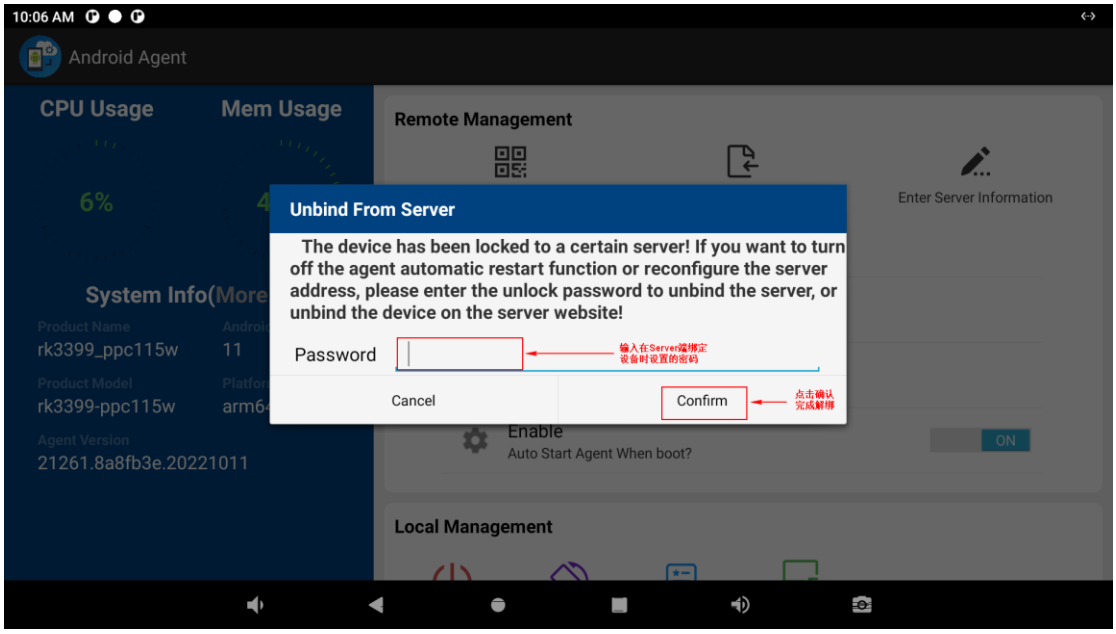
绑定方法：打开设备绑定图标下面的绑定开关，会弹出来一个对话框提示用户设置绑定密码，当用户输入后按下确认按键就可以完成设备绑定功能。注意这个密码很重要，因为如果从设备端解除绑定时需要输入该密码。



解绑方法一：从 Server 端完成解绑动作，只需要关闭设备绑定图标下面的绑定开关即可完成解绑动作。



解绑方法二：从设备端完成解绑动作，在设备端重新配置 Server 信息时，如果设备被绑定到了某个 Server，则任何一种配置方式（扫码配置，导入配置档，输入 Server 信息）都会弹出下面对话框，让用户输入解绑密码来先与旧 Server 解除绑定关系。该密码应该与在 Server 端进行绑定时输入的密码一致。



- 时间同步

功能描述：同步 Client 和 Server 的时间。

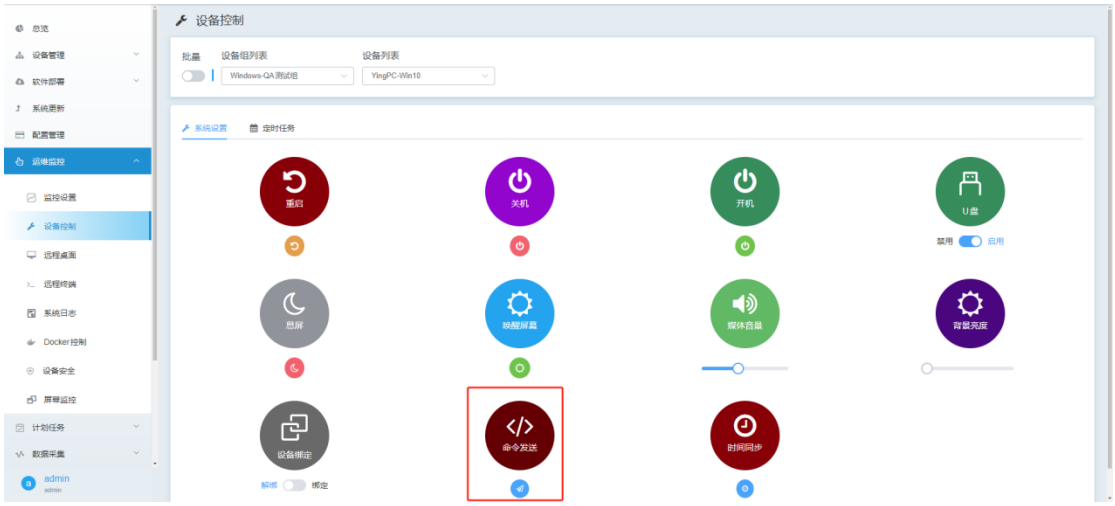
使用方法：有些 Client 没有时间同步服务，当需要设置时间同步的时候，点即“时间同步”图标下面的按钮即可。

2. Windows/Linux 特有功能

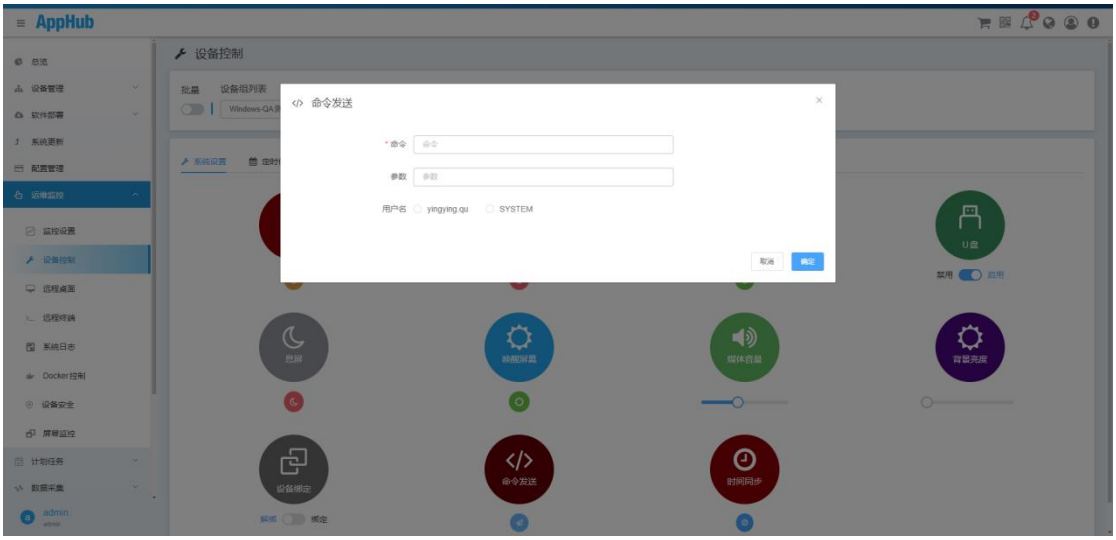
- 命令发送

Windows/Linux 命令发送功能用于向选定的设备发送一条指令。

命令发送方法时首先选择要接收命令的设备：

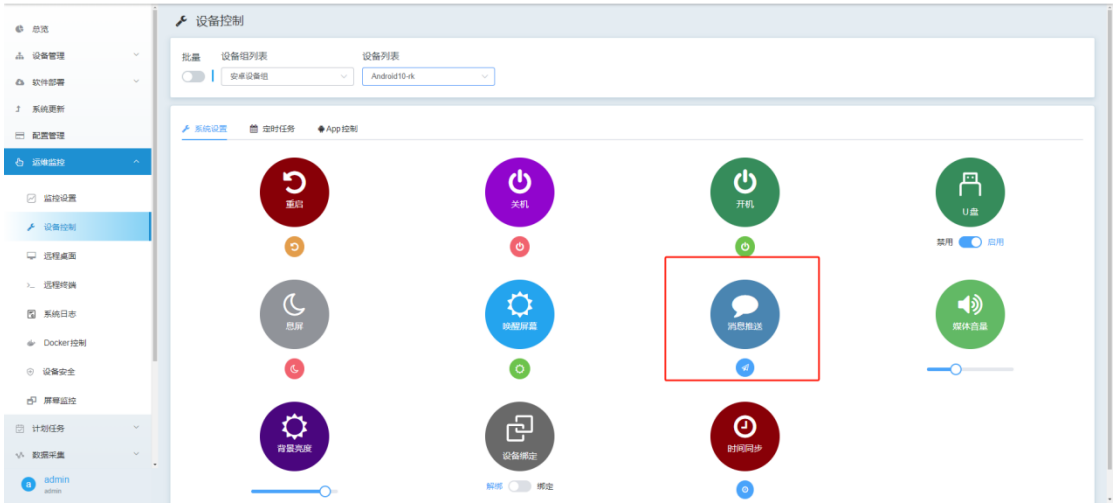


其次填写要执行的命令和参数：



需要说明的是，Linux 目前支持默认以 root 身份执行相应命令，Windows 需要选择用户。

3. Android 特有功能

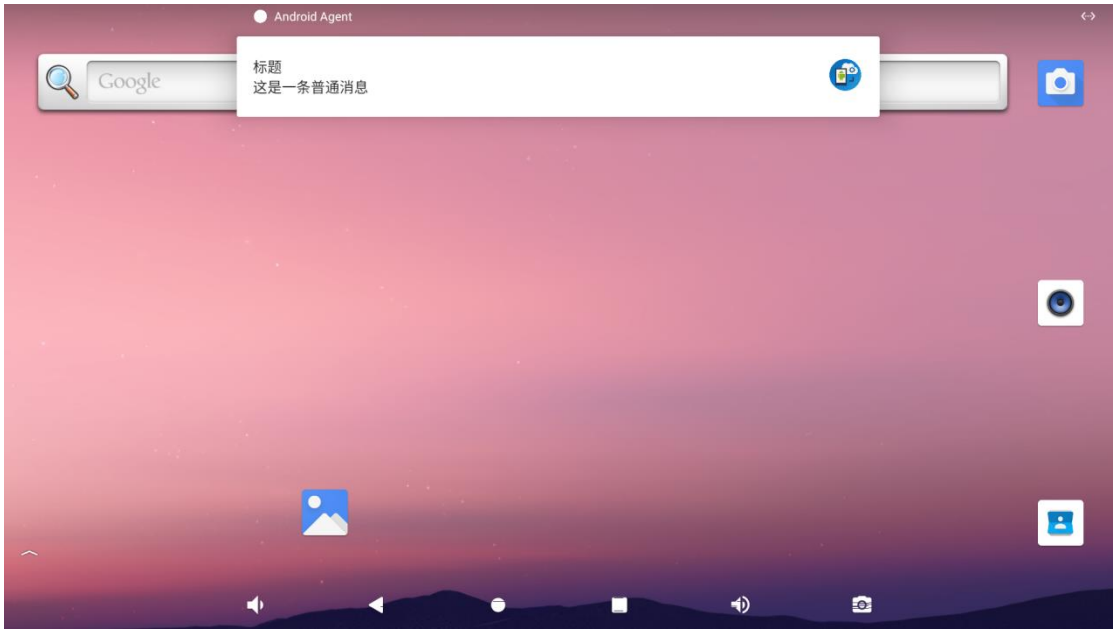


- 消息推送

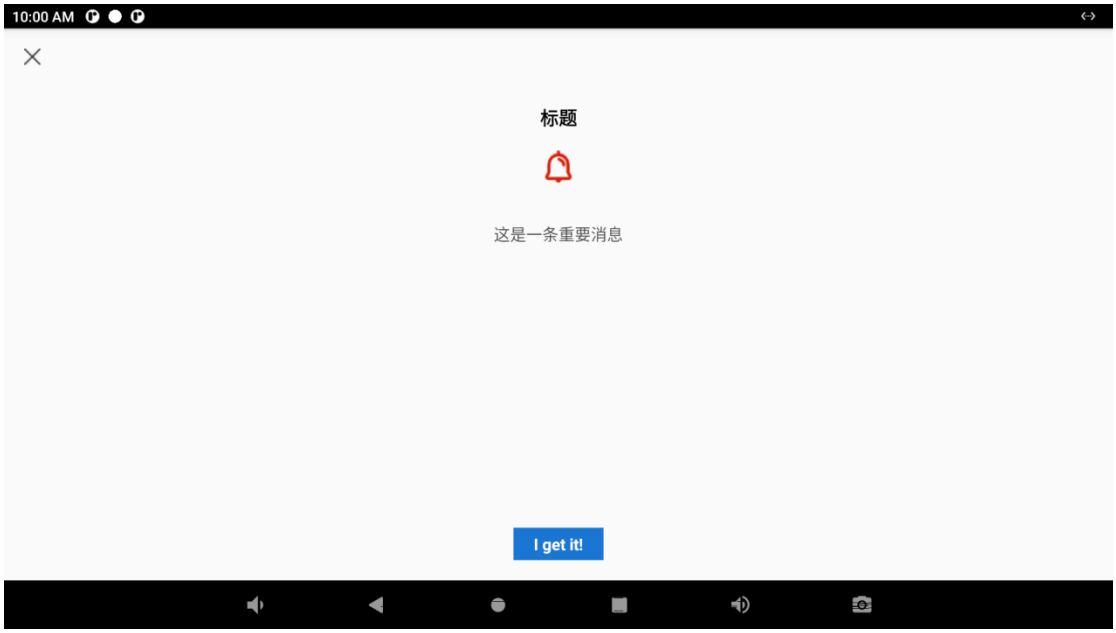
功能描述：向设备端推送一条消息。消息类型分为普通消息和重要消息，普通消息就是在设备端发送一条通知到通知栏，重要消息会直接全屏显示到设备端屏幕，需要用户确认才会关闭。

使用方法：点击消息推送图标下面的控制按键，然后在弹出的对话框中选择消息类型，并且输入消息标题和内容，最后点击确认即可。下面是操作示意图。

下图是设备端显示普通消息的效果：



下图是设备端显示重要消息的效果：



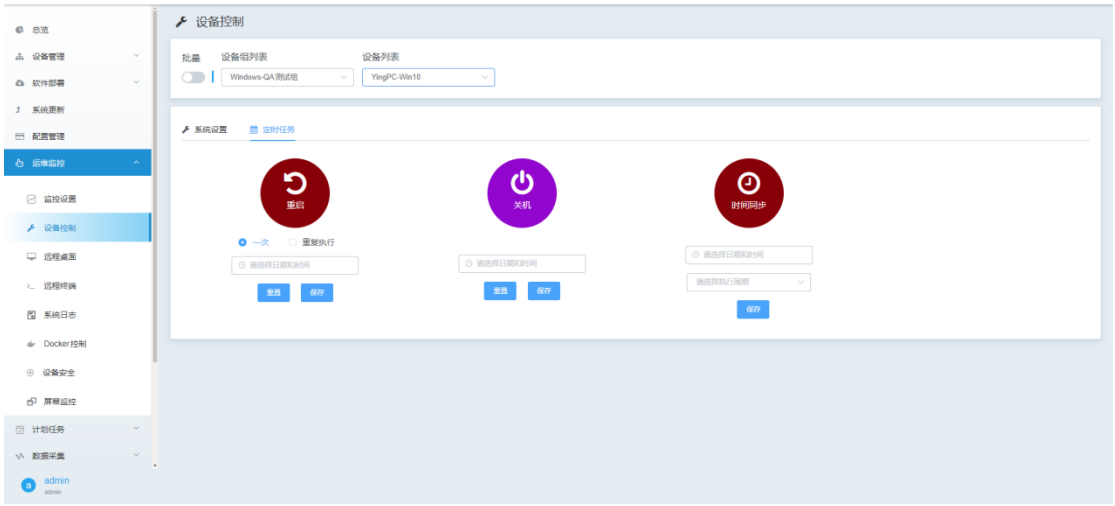
6.2.2. 定时任务

定时任务是提前设置的时程计划任务，当定时时间到达时开始执行，目前实现的功能有重启设备和设备关机两种功能。控制通常分为执行一次和重复执行。执行一次表示任务只需要在设置的时间执行一次就可以了，重复执行表示任务需要在设置的时间周期性执行。

- 单次执行的设置步骤：

【运维监控】->【设备控制】->选择要设置的设备组&设备 ->选择定时任务->选择一次->设置任务执行的时间->选择保存

如下图片表示设置 default 组中的 Android8-UR01 设备在 2024 年 9 月 3 日凌晨重启系统，并且在 2024 年 9 月 10 日 00：00 时关机

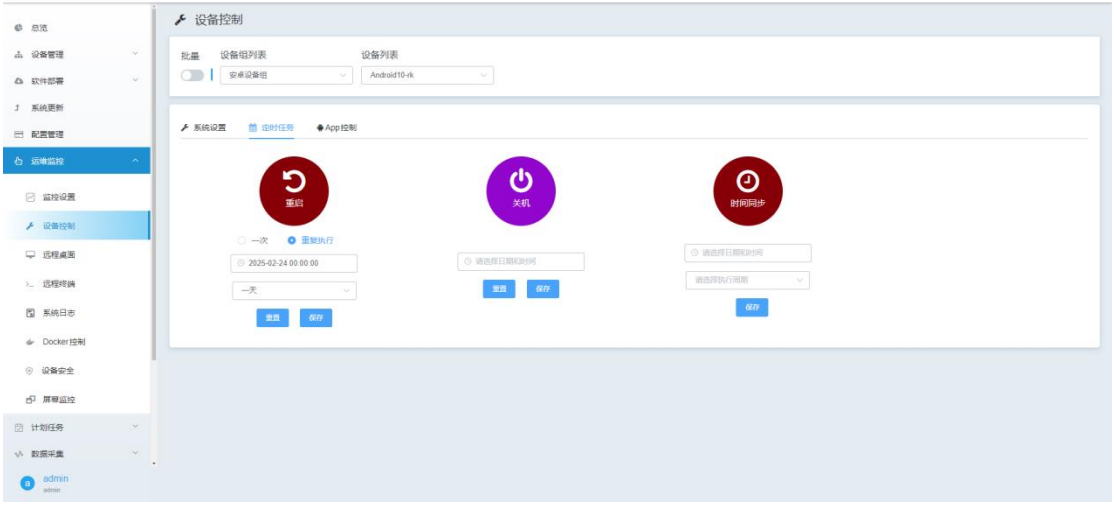


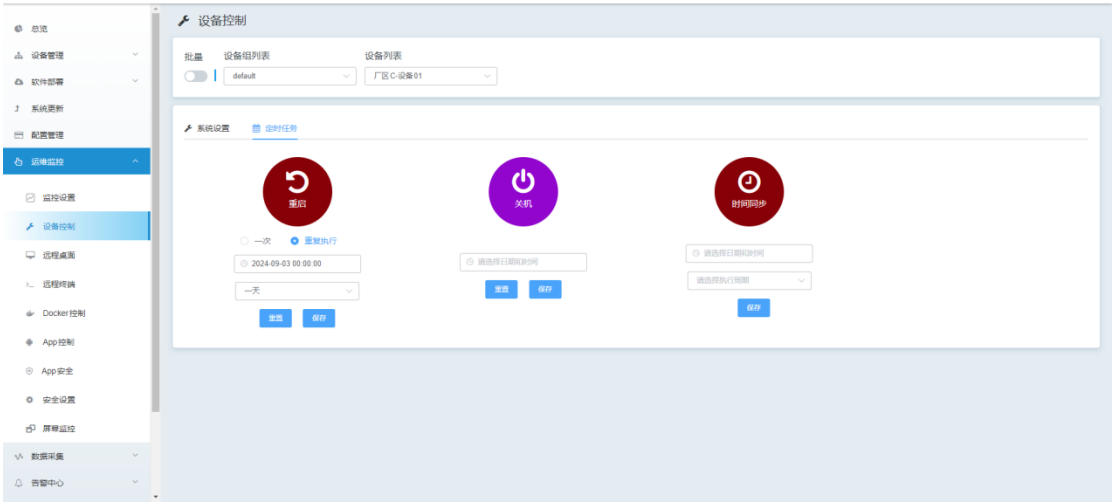
- 重复执行的设置步骤:

【运维监控】->【设备控制】->选择要设置的设备组&设备->选择定时任务-> 选择重复执行->设置任务执行的时间&任务执行周期->选择保存

其中任务执行周期有：一天、一周、一个月、三个月、六个月、一年等多个选项可供选择。

如下图片表示设置组中的某个设备从 2025 年 2 月 24 日起每天凌晨 0 点重启一次系统。





6.2.3. Android App 控制

App 控制包括 App 的打开，关闭，启用，禁止，设置 Kiosk，取消 Kiosk 等功能。

打开：打开指定应用

关闭：关闭指定应用

启用：启用指定应用（启用后，在桌面可以看到该应用，并且可以对该应用进行操作）

禁止：禁用指定应用（禁止后，在桌面就看不到该应用，并且无法对该应用进行操作）

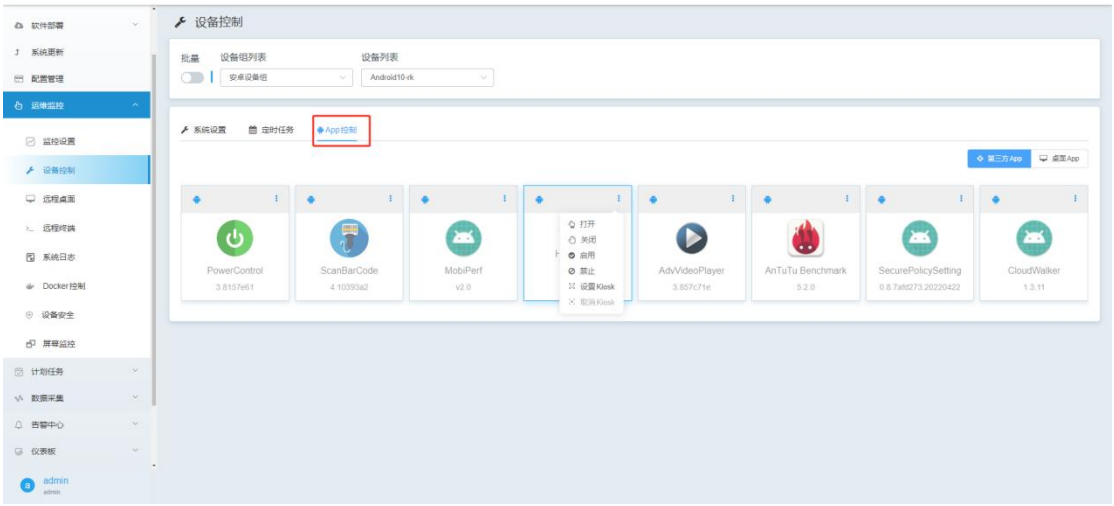
设置 Kiosk：开启 Kiosk 模式没并且将指定应用设置为 Kiosk App。

取消 Kiosk：取消某个应用的 Kiosk 模式。

说明：Kiosk 模式就是隐藏系统状态栏和导向栏，将指定应用全屏显示，并且需要该应用开机自启.

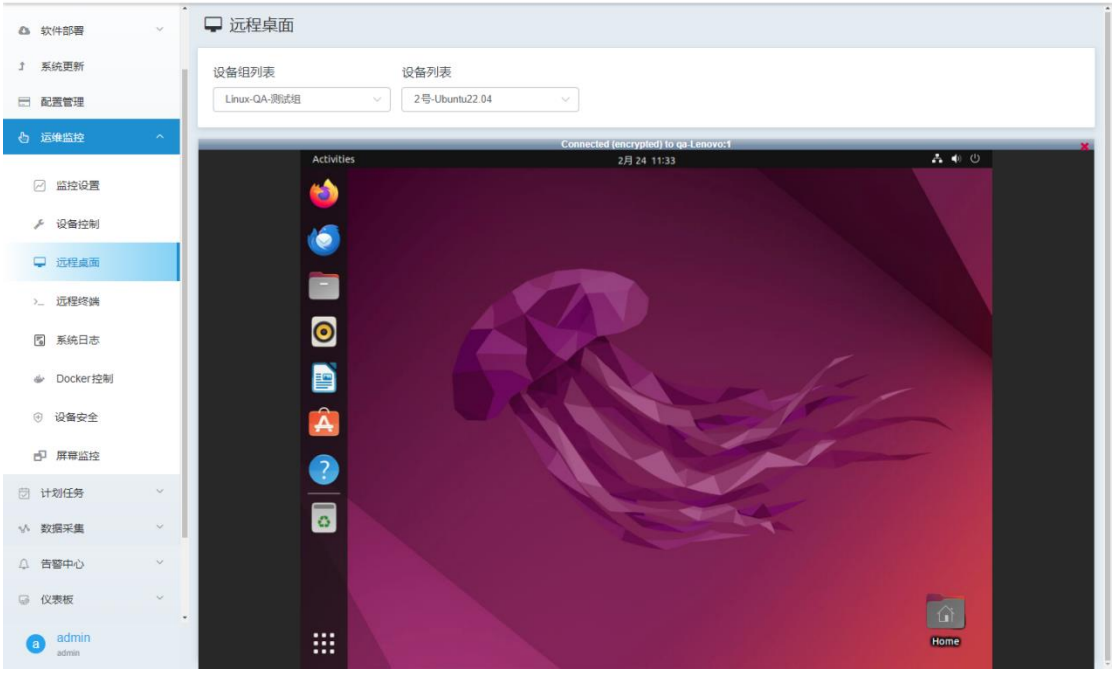
App 控制操作步骤为：

【运维监控】->【App 控制】->选择要部署的设备组&设备->选择第三方 App 或系统 App 或桌面 App -> 打开要控制 app 的隐藏菜单->执行操作



6.3. 远程桌面

远程桌面是将远端设备的桌面投影到 web 页面中，用户可以通过页面对远端设备桌面进行操作，如下图所示：



DeviceOn/EIM 远程桌面需要 Client 依赖于 VNC 功能，

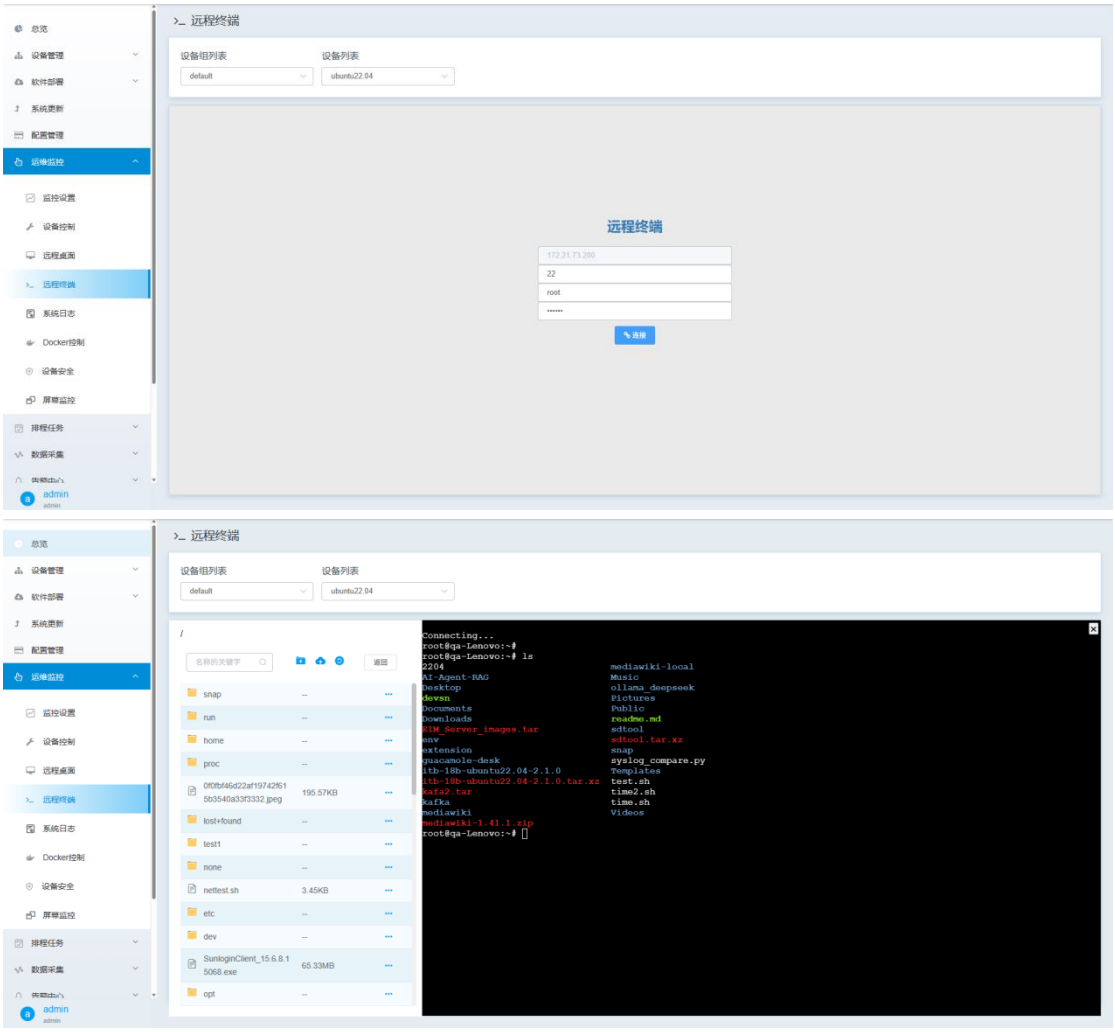
Linux 系统默认依赖于 X11Vnc，如果系统没有安装请先安装，例如 Ubuntu 环境下通过 apt 命令安装即可：`apt-get install x11vnc`

Windows 系统依赖于 UltraVNC，Client 安装后默认就会配置有，无需手动安装；

Android 系统 VNC 依赖于 DeviceOn/EIM team 自己开发的 VNC，该 VNC 当前已经适配众多赋华硬件平台和系统，Client 安装后默认配置有。

6.4. 远程终端/SFTP

远程终端是用户可以通过 web 页面远程 ssh 到指定设备的终端对设备进行操作，同时支持命令行交互操作和 SFTP 文件传输操作。如下图所示，输入相应的用户名和密码即可登录：

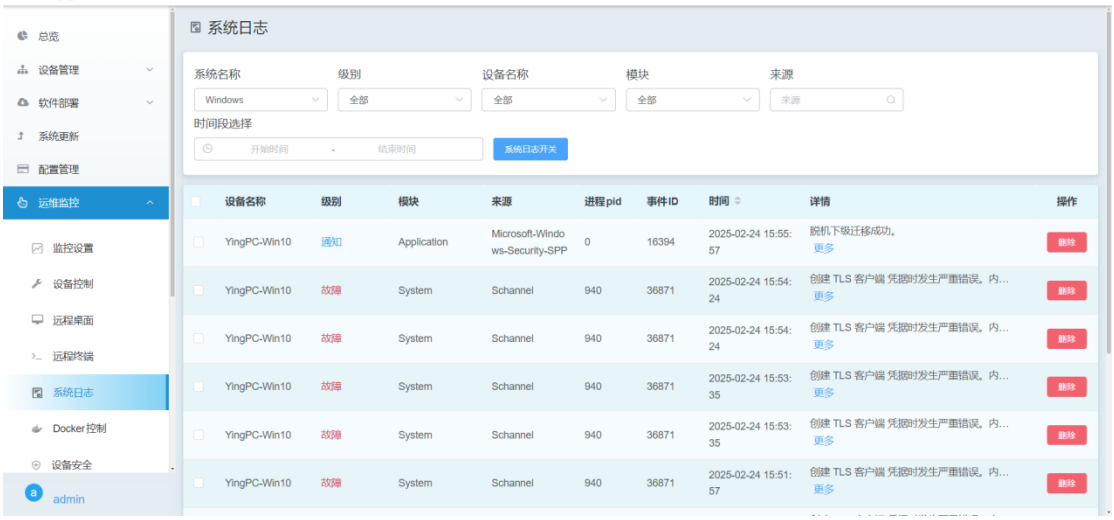


远程 SSH 当前仅支持 Linux 系统，需要注意的是，要使远端的 EIM Client 支持远程终端功

能，需要使能 EIM Client 端主机的 ssh 的登陆功能。

6.5. 系统日志

DeviceOn/EIM 的系统日志功能记录了 Windows 系统的系统日志、应用程序日志、启动日志和安全日志所有日志，以及 Linux 系统的 emerg (0)/alert (1)/crit (2)/err (3)/warning (4)/notice (5)共 5 个等级的系统日志。记录系统中硬件、软件和系统问题的信息，同时还可以监视系统中发生的事件。用户可以通过它来检查错误发生的原因，或者寻找受到攻击时击者留下的痕迹。



通过系统类型、设备名称、时间段、级别等关键字的筛选，可以精准定位设备系统日志内容。默认情况下系统日志是关闭的，需要用户在监控设置中打开，请参考 6.1.6 章节。

6.6. Docker compose/swarm 管理

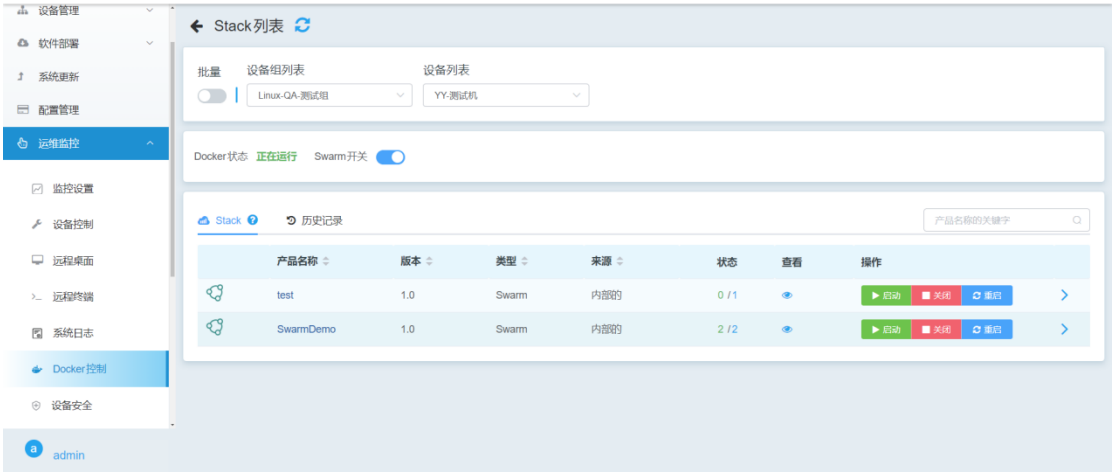
DeviceOn/EIM 支持 Windows，Linux 下的 Docker/compose/swarm 远程管理，一方面我们为 docker 运维管理提供了更人性化的交互界面（比 docker 的原生命令），通过图表，状态等描述，使相关分析更加清晰，简单，明了和专业。另一方面，我们通过批量化管理，使得 docker 管理更简单，更方便，更加节约成本。选

选择 DeviceOn/EIM 菜单栏【运维监控】->【Docker 控制】进入 docker/compose/swarm 管理页面，其如下图所示：

择目标设备组&设备，就可得到该目标机器上的 Docker stacks 的数量，docker 容器的数量和运行状态，以及 docker image 的数量和占用存储空间的大小。我们习惯上将 docker swarm 或者 docker compose 统称为 docker stacks（注意和原生的 docker stack 不是一个概念）。每一项都会有丰富的内容，下来我们将会分章描述他们。

6.6.1. Docker stacks 管理

Docker stacks 管理目标机器上所有的 docker compose 或者 docker swarm 集群。其管理页面如下图所示。

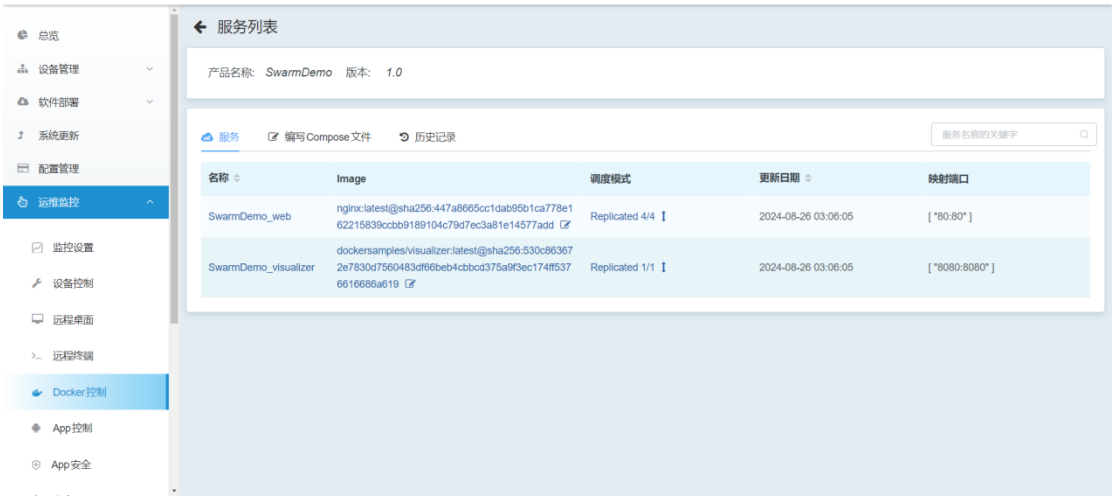


该页面会显示 docker 的运行状态，和 docker swarm 是否被开启，如果 swarm 处于关闭状态，当前可以控制的 stacks 便为 docker compose，否则就为 docker swarm。作为简单的测试，用户可以通过 swarm 开关来将目标机器设置为 swarm 的 manager 节点。 swarm 和

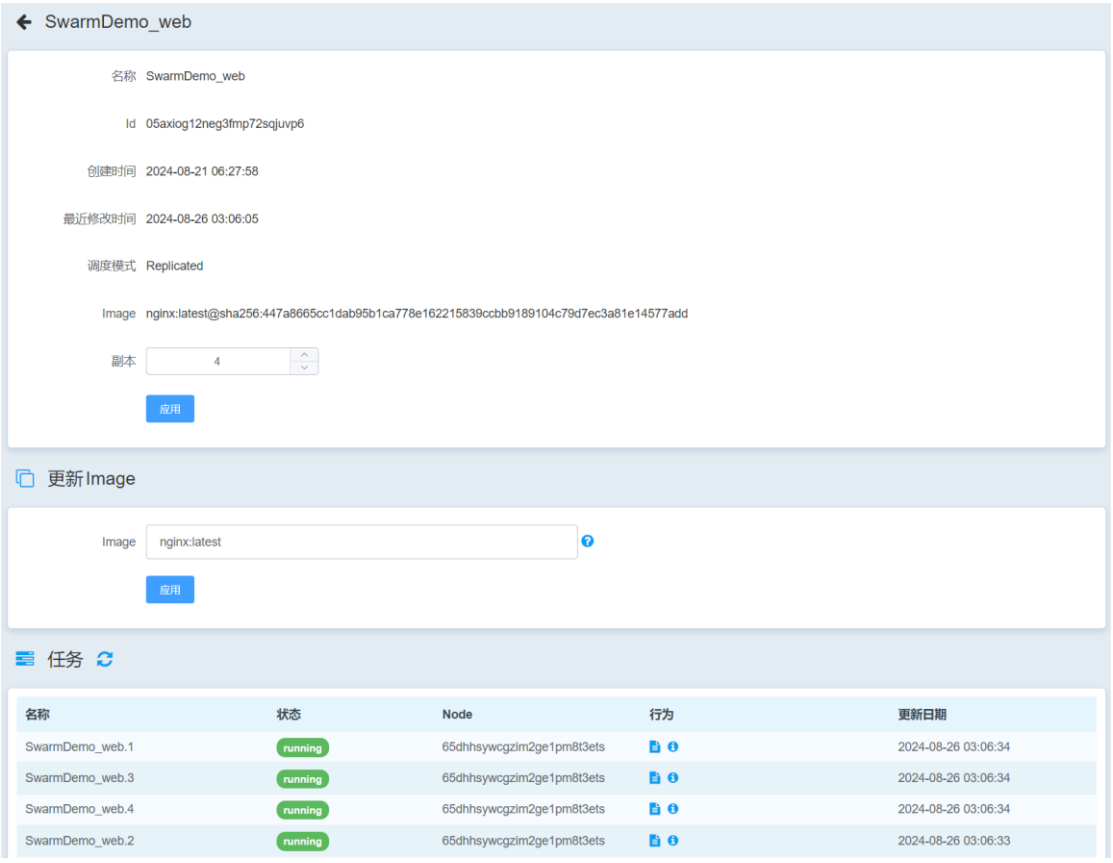
compose 有很多相似之处。

该页面还会列出目标机器上，所有的 docker stacks 的列表，及各个 docker stacks 的详细信息。用户还可以对具体的 stacks 做 Start/Stop/Restart 的操作，同时还可以查看该 stack 对应的 yaml 文件，以了解该 stack 的更详细情况。

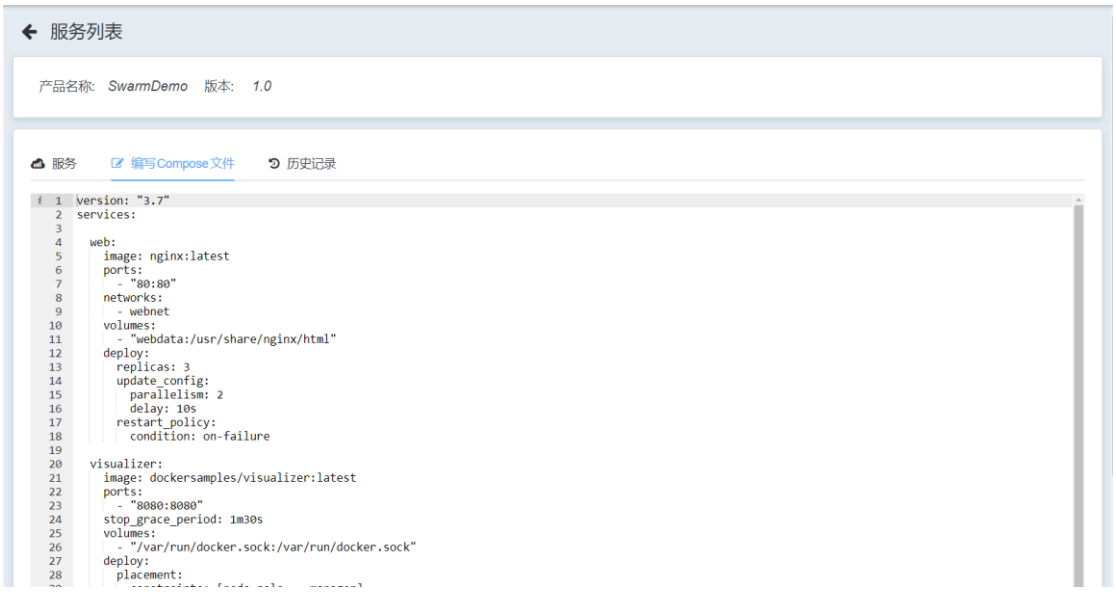
如果用户想了解关于该 stacks 更详细的情况，可以点击 stacks 的产品名称，进入 stacks 管理页面，如下图所示：



Stacks 管理页面包含当前部署的 swarm 产品的服务容器列表、镜像、调度模式和映射端口等。点击某个服务后，可以远程修改该容器的副本个数、镜像，并查看当前运行容器的具体情况，如下图：



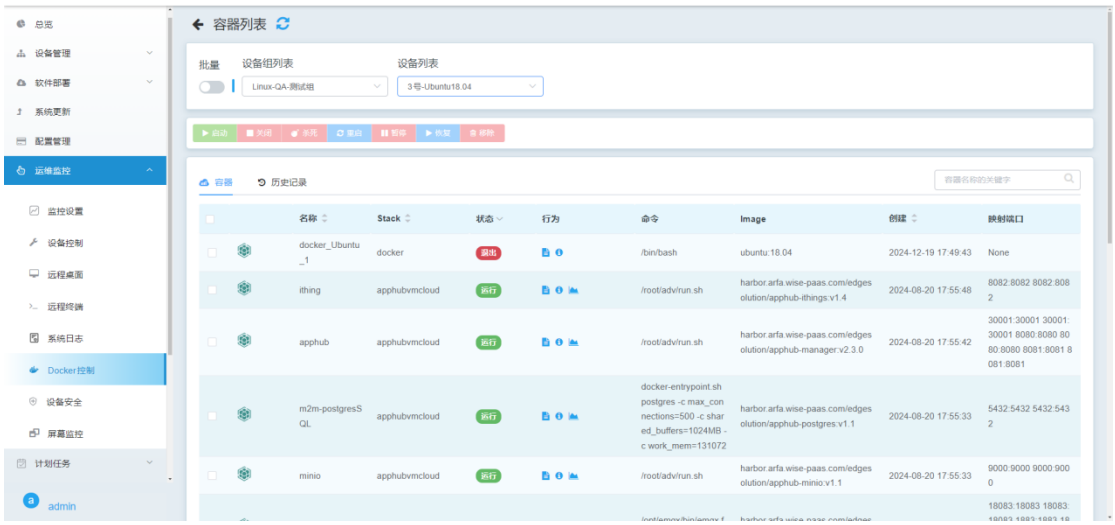
还可以直接在线编辑 yml 文档，远程更新当前部署的 swarm 服务，以及更新的历史记录如下图：



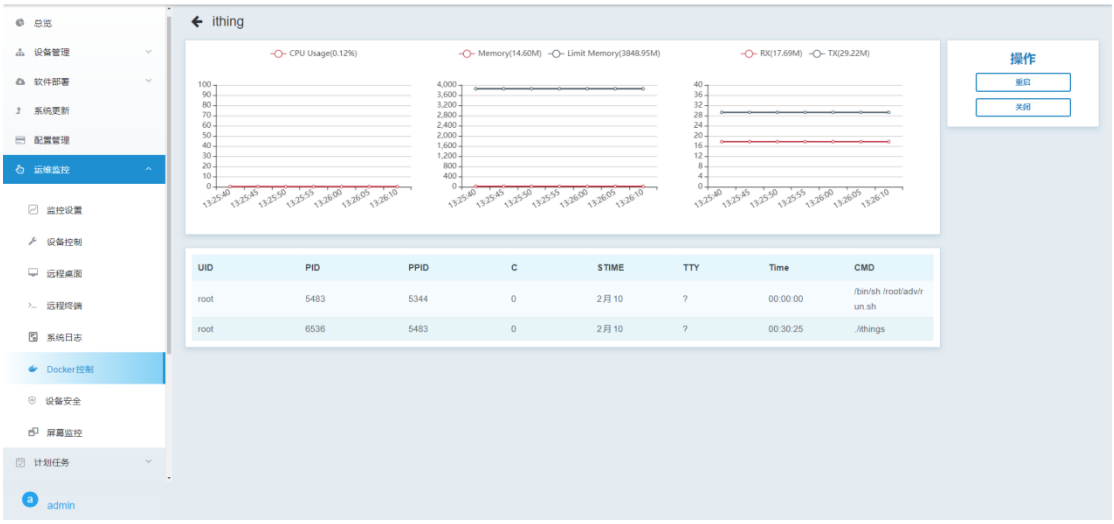
6.6.2. Docker 管理

Docker 容器管理会管理目标机器上所有的 docker container，是对某个具体 container 的管理。

包括每个容器的运行状态，容器的启动命令，容器的 image 名称，端口映射，创建时间等诸多信息。用户可以根据自己的需要，对所选容器进行 start, stop, kill, restart, pause, unpause, remove 等操作，其行为完全等价于 docker 相关指令。还可通过 log（等价于 docker log）查看该容器的启动 log，通过 Inspect Info（等价于 docker inspect）查看该容器运行的更详细状态。如下图所示：



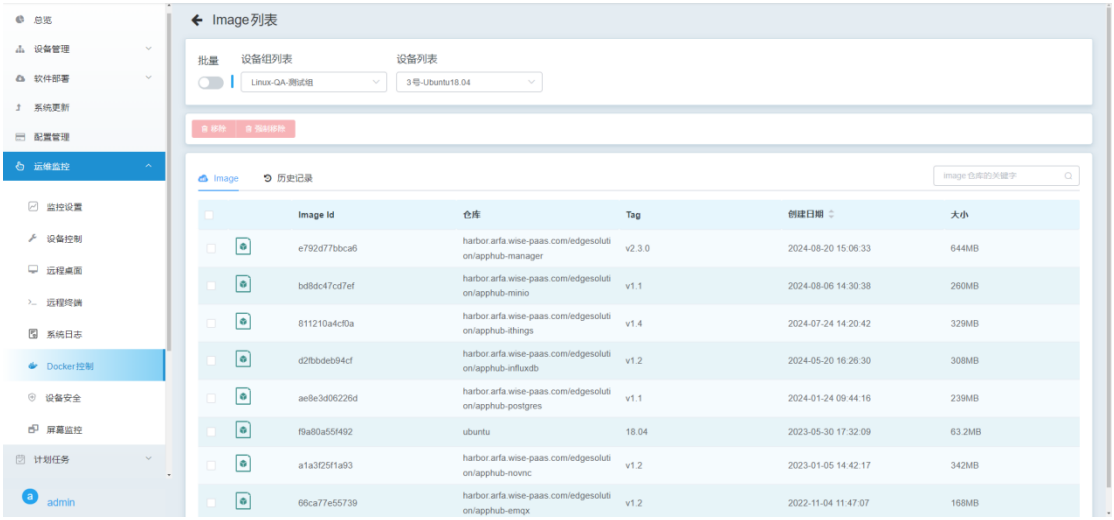
如过用户还想查看容器内的程序运行状况，点击 Monitor 进入容器内程序监控页面：



该页面会显示此容器占用的 CPU，内存以及网络流量的统计图，方便用户观察和分析，同时还会列出容器内所有运行程序的列表及相关信息，以方便客户观察&分析。

6.6.3. Docker image 管理

Docker image 管理会管理目标机器上所有的 docker image，其管理页面如下图所示。



用户可以通过该页面查看各个 docker image 的详细信息，并且对 image 进行删除和强制删除的操作。强制删除等价于 docker rmi -force 命令。

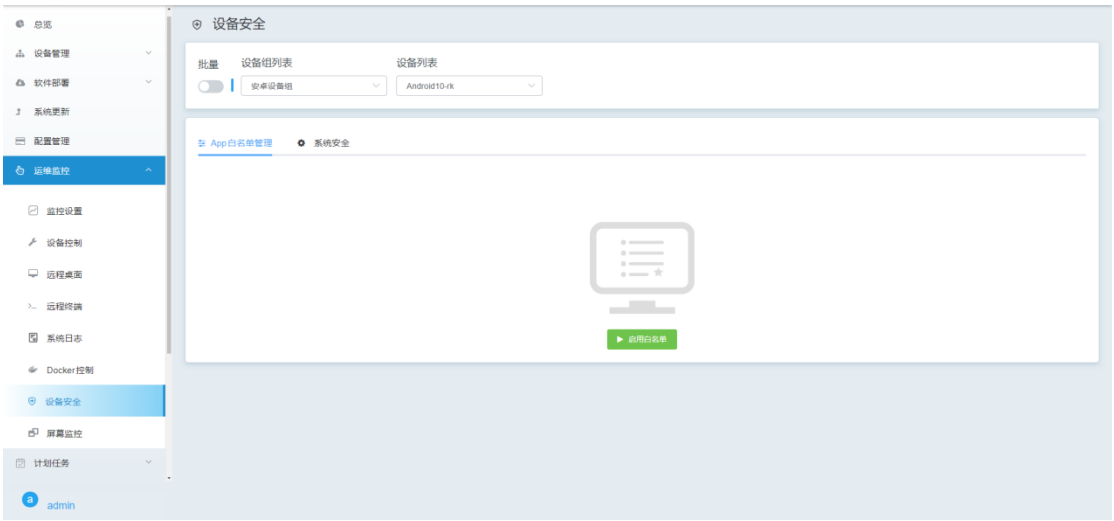
6.7. 设备安全

6.7.1. App 安全设定

在企业 Android 设备中经常会有客户要求只能安装和运行工作应用，而禁止安装和运行与工作无关的应用。这时 App 安全会起到很关键的作用，该功能可以让管理者将需要的 App 加入到白名单列表，把不需要的 App 加入到黑名单列表。加入白名单列表的 App 可以正常的打开关闭，加入黑名单的 App 则会被禁用，在桌面无法看到黑名单 App 的图标，也无法对黑名单 App 进行打开操作。

6.7.1.1. 开启 App 安全功能

系统默认情况下是没有开启 App 安全功能的，可以通过下面步骤开启 App 安全功能：**【运维监控】->【设备安全】->【App 白名单管理】**->选择要设置的设备组&设备 ->选择启动白名单



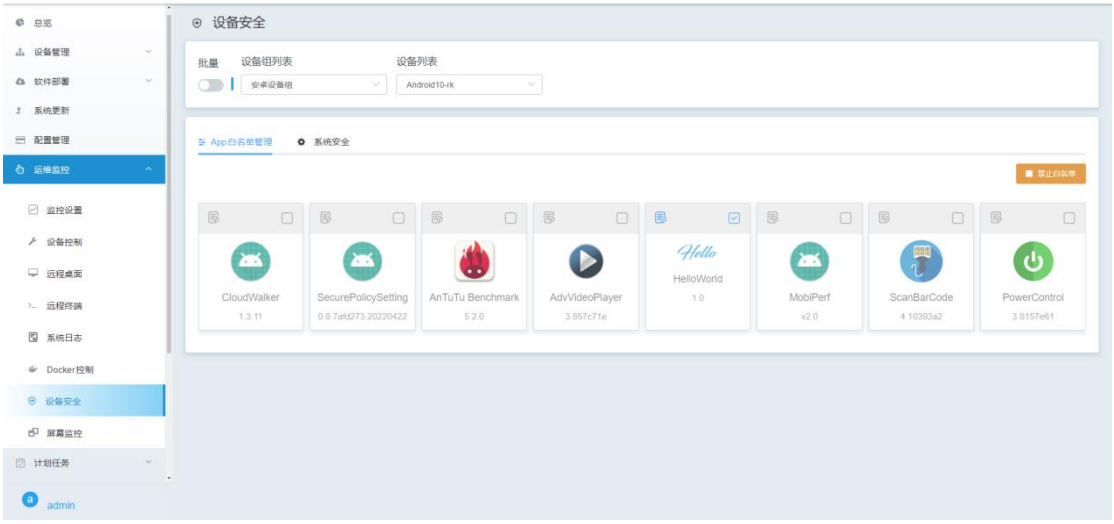
6.7.1.2. 设置 App 白名单

当 App 安全功能开启后，DeviceOn/EIM 默认会将系统应用添加到 App 白名单中，将第三方应用（后期安装的应用）添加到黑名单中，管理者可以通过 DeviceOn/EIM Server 进行控制来修改 App 白名单和黑名单。

操作步骤：**【运维监控】->【设备安全】->【App 白名单管理】**->选择要设置的设备组&

设备 ->将 App 添加或移除白名单（选中 App 右上角的选择框就表示将 App 添加到白名单中，相反表示将 App 从白名单中移除）。

如下将 default 组中的某设备 App 安全开启后，并且将 Helloworld App 勾选添加到了白名单，其他默认在黑名单。

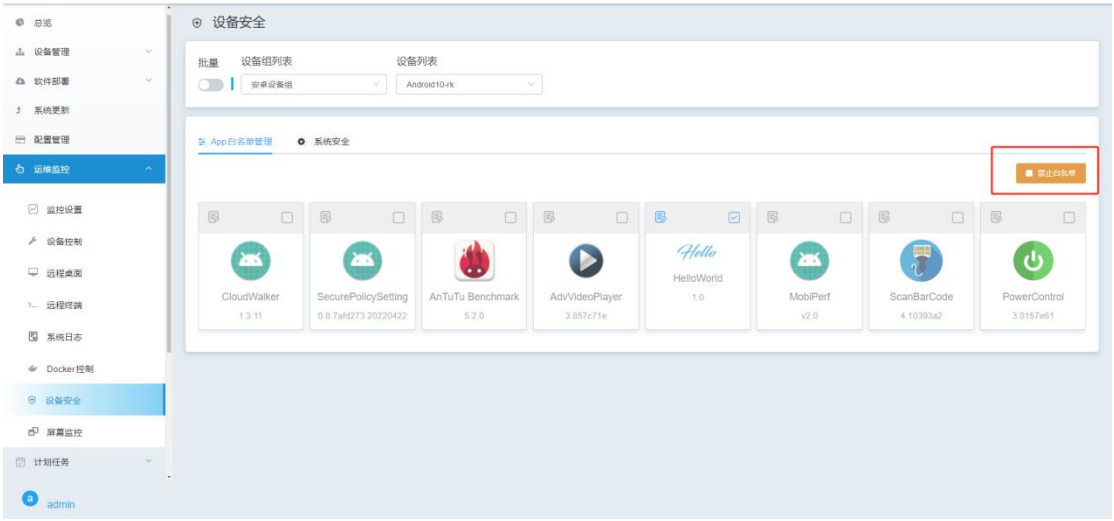


此时在系统桌面可以看到 Helloworld App 但是看不到且无法操作其他 App。

6.7.1.3. 关闭 App 安全功能

当关闭 App 安全功能后，会清除所有 App 白名单和 App 黑名单。这样设备上安装的所有应用都可以被正常看到和操作。

操作步骤：**【运维监控】->【设备安全】->【App 白名单管理】**->选择要设置的设备组&设备 ->选中禁止白名单选项。



6.7.2. 系统安全设置

安全设置又可称为 Android 系统企业级安全配置管理，是专为 Android 系统设计的一套设备安全相关的系统设定功能。安全设置适用于企业所有设备，通过多种配置策略来限制设备所提供的能力，是一种对企业级设备进行管理的良好方法。例如，我们可以禁用设备的 ADB 功能，禁用之后无论是 App 还是设备使用者都无法开启 ADB，这就避免了专业人员通过 ADB 方式对设备进行危险操作的可能性。

安全设置的核心实现由 Android 标准 API 提供，因此在适配性和安全性上都有保障，另外，API 提供的丰富功能可以满足工业场景下的多种需求。目前，我们已经提供了 30+项工业场景下常用的配置，具体如下：

1. 蓝牙：开闭蓝牙、禁用蓝牙功能。
2. WiFi：开闭 WiFi、禁用 WiFi、锁定到指定 WiFi 等。
3. USB：禁用 USB 传输。
4. 定位：开启定位、禁用定位。
5. ADB：开启 ADB、禁用 ADB。
6. 锁屏页：锁屏密码最低复杂度要求、锁屏密码最低长度要求、禁止可信代理解锁、禁止锁屏页显示通知、设置解锁方式。
7. 屏幕超时：设置超时时长、设置可选的超时时长范围、禁止修改超时时长。
8. 音量和背光：调节音量、禁止调节音量、禁用麦克风并静音、调节背光、自动背光。
9. 时区：自动时区和时间、禁止修改时区和时间。
10. 其它：禁用相机、禁用截图功能、禁用自动密码填充、记录设备安全日志、屏幕自动旋转。

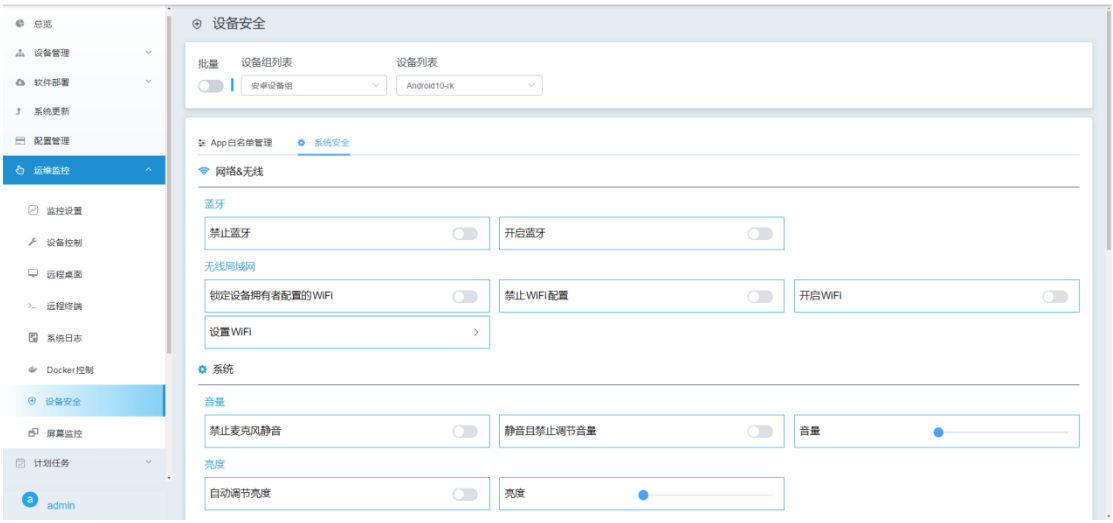
6.7.2.1. SecurityPolicySetting 安装&配置

要实现安全设置功能，首先需要在 Android 设备端安装 SecurityPolicySetting App，安装完成后需要借助 ADB 对 App 赋予 Device Owner 权限，获得该权限后，App 即可进行企业级的安全配置。

6.7.2.2. 安全设置

SecurityPolicySetting 配置完成后，即可对该设备进行相关设置了。可以按需选择配置相应项。【运维监控】->【设备安全】->【系统安全】->选择要设置的设备组&设备 ->操作设置项

如下图片表所示:

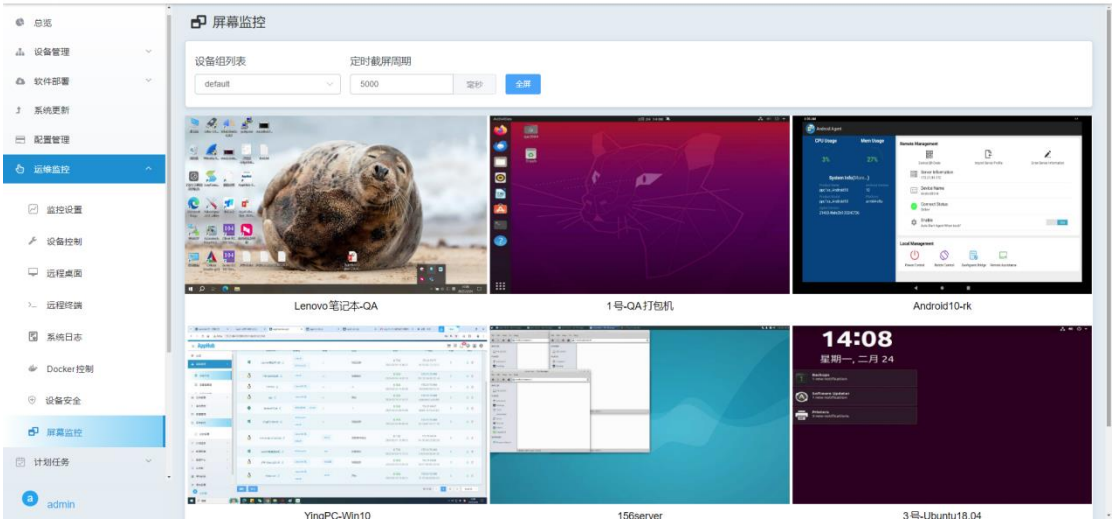


这里每次设置只能设置一项，设置多项需要分别单独操作，如果需要一次配置设置多项，那么在配置管理中可以实现，具体操作方法如下:

多项安全设置是依赖【配置管理】模块实现的，您可以在【配置管理】 ->添加 Android 任务清单->【系统设定】部分添加多个设置项，这些设置项随配置文件下发至设备后，即可在设备端完成多项安全设置的配置操作，具体操作步骤请看第 6 章配置管理章节。

6.8. 设备屏幕监控

屏幕监控是指用户通过 web 页面同时看到多个设备的桌面（实时刷新），多用于设备屏幕的监控。如下图所示:

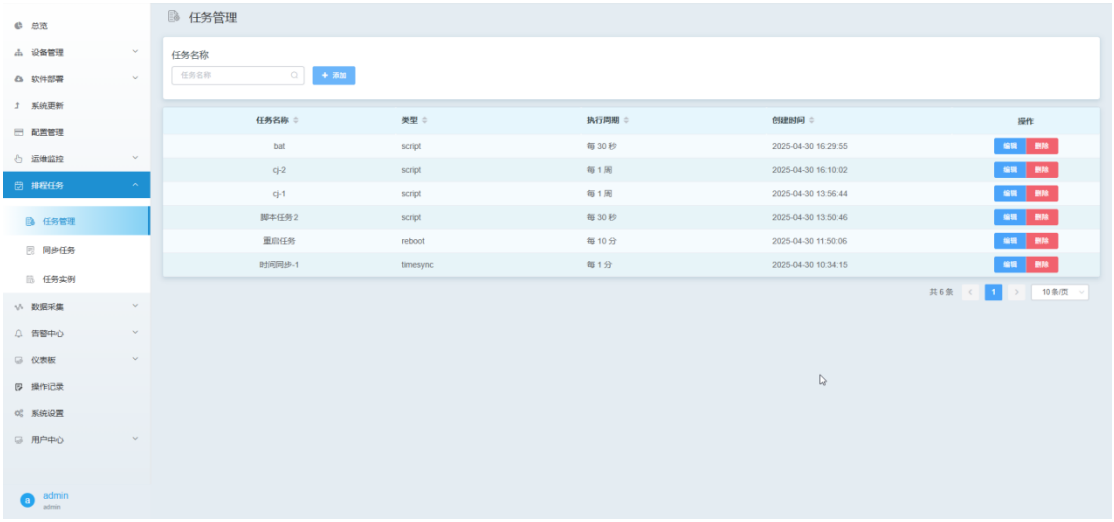


7. 排程任务

排程任务通过预先设置并按时执行的任务，可以对分布在各地的设备进行定期检查、维护、更新和数据收集等工作，无需人工现场干预。
目前支持的排程任务类型包括设备重启、远程脚本、时间同步和网络 ping 等，未来还会根据需要继续增加。

7.1. 任务管理

在任务管理列表，可以查看当前已经创建的任务，同时可以添加新任、编辑或者删除已存任务



此外，还可以根据需要添加新的任务、或者删除任务。

- **添加任务**

点击“添加”按钮后，在右侧弹出框的类型中进行添加。

任务名称：输入自定义的任务名称

选择类型：目前可以选择 reboot、script、ping 和 timesync。

执行周期：可以选择周期单位是秒、分、时、天、周、月，然后确定具体的执行时间。

示例：

下面以每周一的凌晨 0 点，执行特定脚本为例进行说明。

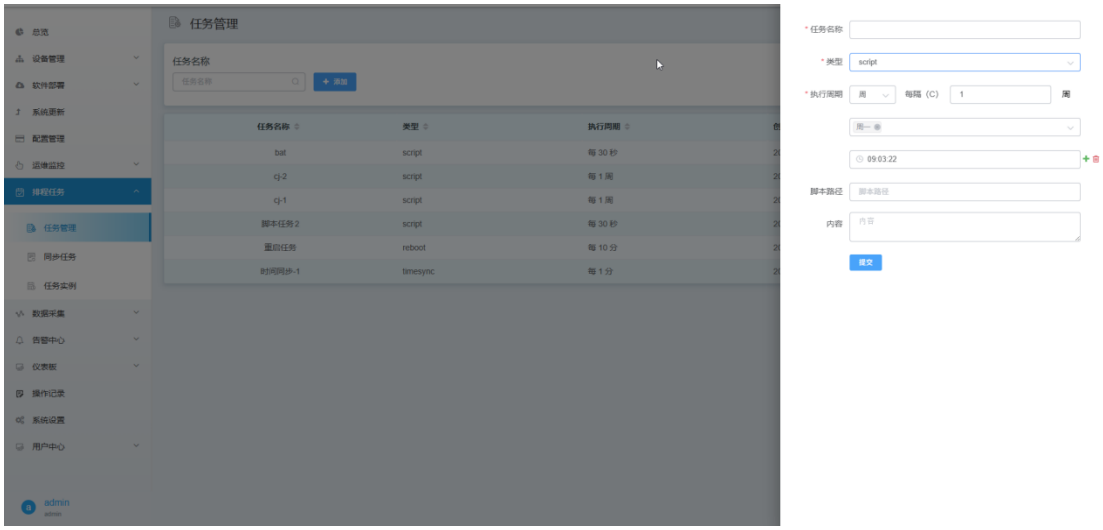
选择类型为“script”。，周期选择“周”，每隔 1 周，时间选择“周一”，00:00

（1）执行脚本在 Client 设备端已存在的情况下：

直接添加脚本路径

（2）执行脚本在 Client 设备端没有的情况下：

在内容中添加具体脚本内容，并在路径中添加将要该脚本保存的路径位置。



再次说明的是:

如果 Client 设备本地没有示例中的“/home/test.sh”那么就会将内容框内的内容同步到本地保存于“/home/test.sh”;

如果 Client 设备本地已经有示例中的“/home/test.sh”脚本，那么直接将脚本路径填入，内容框不需要输入。

- **编辑任务**

任务首次添加完成后，如果需要变动可以直接点击编辑按钮，再次编辑。编辑后需要再任务实例列表进行同步动作后，才会按照编辑后的任务来执行

- **删除任务**

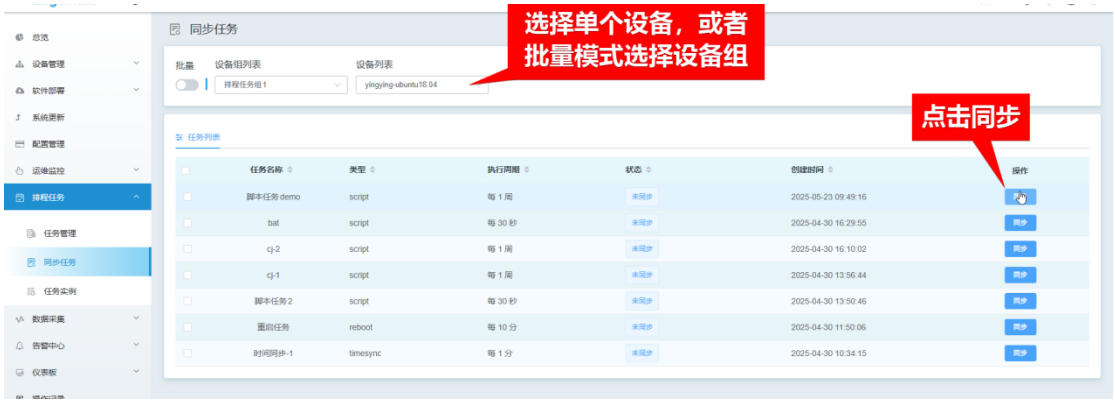
当任务没有设备同步执行的时候，可以被直接删除掉，否则需要先在实例列表中将任务从设备上移除掉后再删除

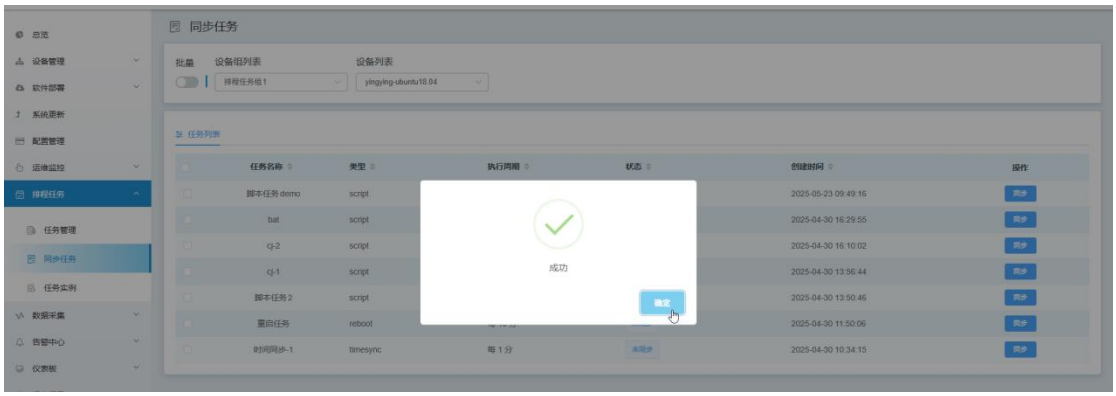
7.2. 同步任务

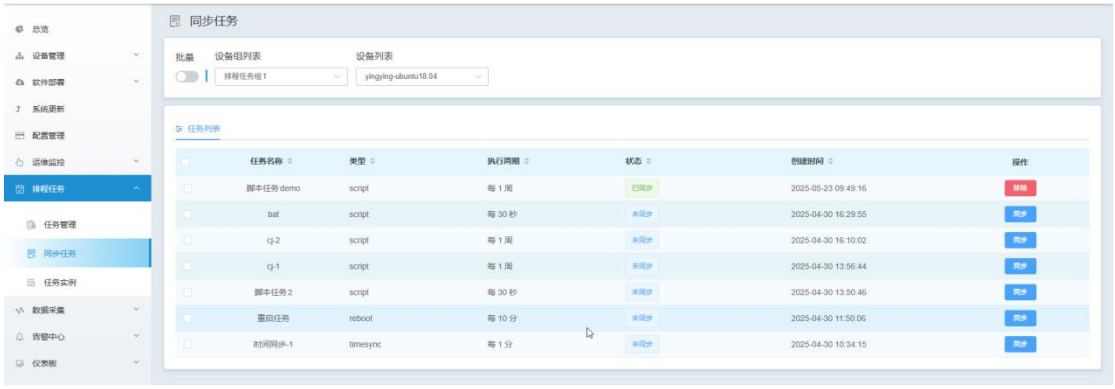
任务管理负责创建和编辑任务，那么同步任务就负责将任务给同步到 client 设备端。

- **同步**

在同步任务也，选择单个设备，或者批量模式选择设备组后，点击要同步的任务的“同步”按钮。







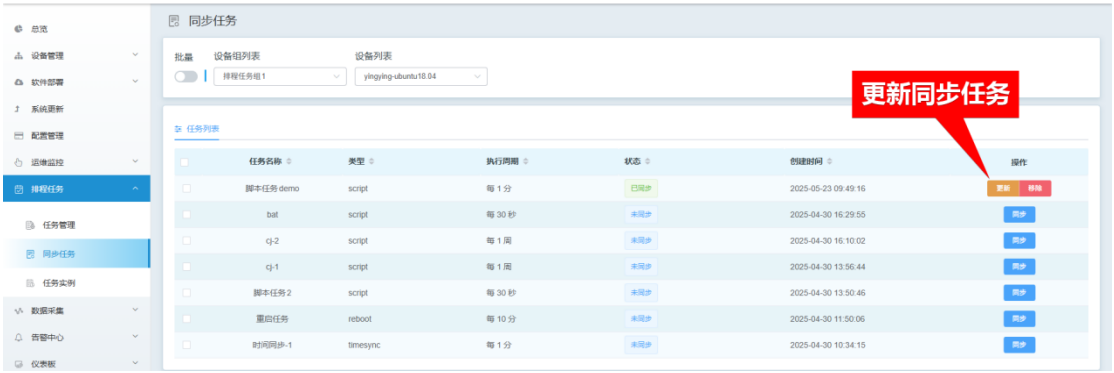
同步成功后，

在同步任务的列表可以看到状态及操作按钮的变化；

在任务实例列表中就可以看到具体执行情况了。

- **更新**

如果后期任务在任务管理中重新修改编辑保存后，在同步任务列表会出现更新按钮需要更新的时候，同样的在同步任务列表完成，如下截图：



- **移除**

  点击异常后，该任务将从设备上移除，即之后不在执行该排程任务。

7.3. 任务实例

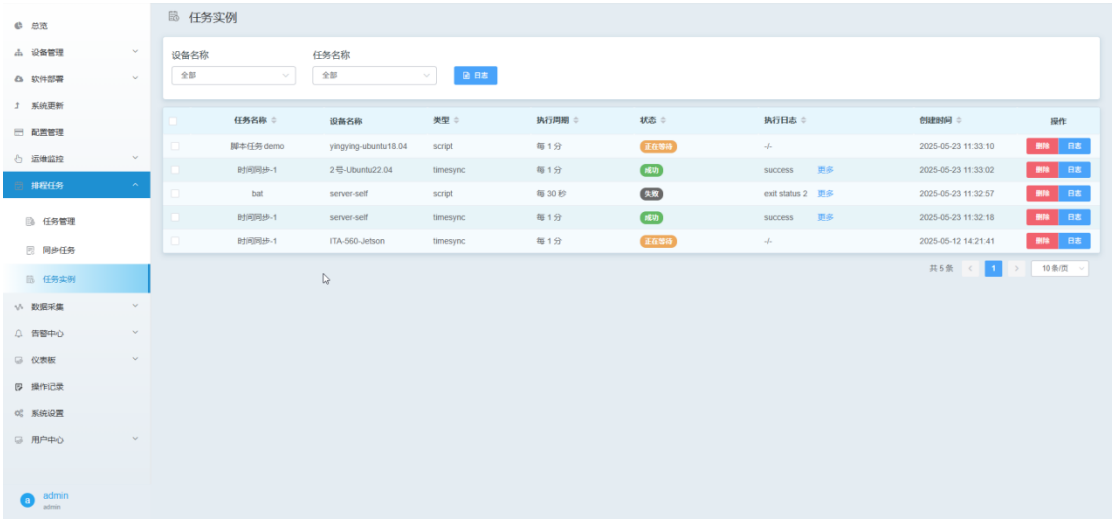
在同步任务列表中将任务同步到 client 设备端后，在任务实例列表就可以看到任务执行的情况了。

刚开始任务还没有开始执行的时候，状态是“正在等待”；

当任务开始执行后，状态就更新为最新一次执行结果；如果想看历史执行过程，可以点击日志查看。

如下图所示：

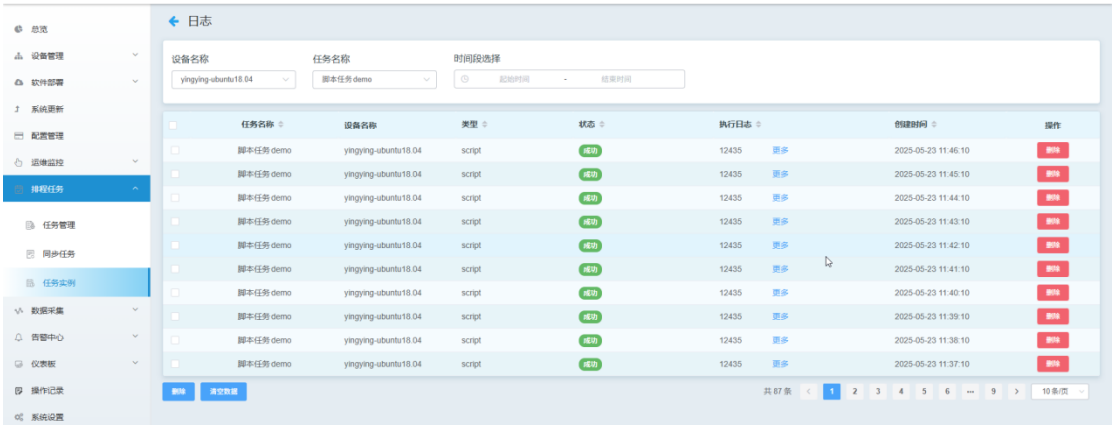
- **查看**



- **删除**

在任务实例列表中点击删除后，改任务将从设备上移除，等同于在同步任务列表中的移除功能。

- **日志**



8. 数据采集&监控&告警

在物联网蓬勃发展的时代，万物数字化已经是大势所趋。物联网中，这些巨量边缘设备因为拥有不同系统，不同的应用协议使它们的应用数据的格式更是千差万别。要更好地管理这些设备，就需要一种更加通用的数据格式的协议来统一它们。为此，物模型的概念诞生了，它是对现实世界中一个实际的事物的抽象描述。

物联网中，有些边缘设备不总是能和远端的 Server 保持连线，比如设备经常断线，或者有些低功耗设备经常需要进入休眠状态，它们仅在被唤醒的状态下上报自己的数据。然而对于远端的 web 应用，不可能总是等待处于断线状态设备。为了解决这个问题，通常在 Server 端维护一个保存设备最后一次上报数据的 Cache 即可，我们将其称为设备影子。

大量的 IT 设备中，数据都是以秒、毫秒级的频率产生着，不是所有的设备数据都需要连续不断地向远端的 Server 上报，他们更多是满足某个条件后，才产生数据的上报，有些更是以告警的方式出现，这便需要设备数据的监控和告警。

DeviceOn/EIM 在发展自己设备数据监控&告警系统的过程中，也融入以上所述的解决方案。

DeviceOn/EIM 的设备数据监控&告警旨在解决以下问题：

1）EIM Client 以插件的方式接入更多不同的设备数据协议；

2）DeviceOn/EIM 的设备数据统一化，便于 web 应用的管理&智能分析；

3）DeviceOn/EIM 实现设备影子的功能；

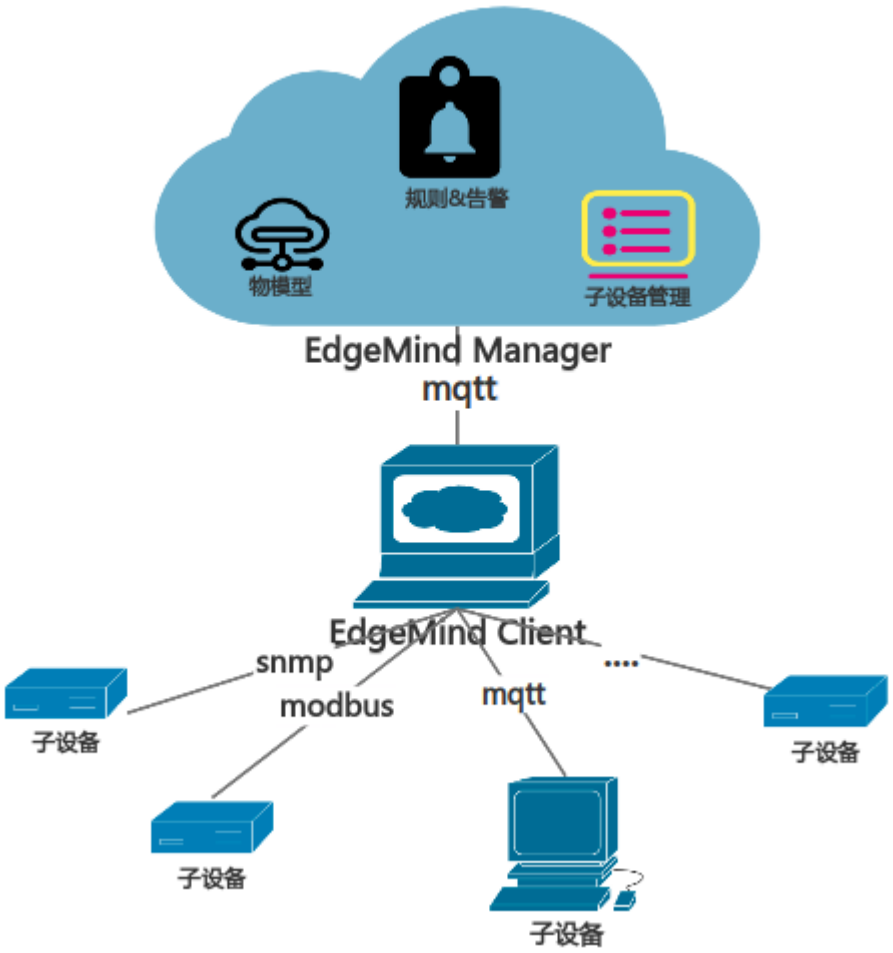
4）EIM Client 在一定情况下，实现低代码开发；

5）实现基于可配置规则的数据监控，数据转发；

6）实现动态，可配置的事件告警；

7）动态配置事件处理机制。

下图为 DeviceOn/EIM 设备数据监控&告警的架构图：

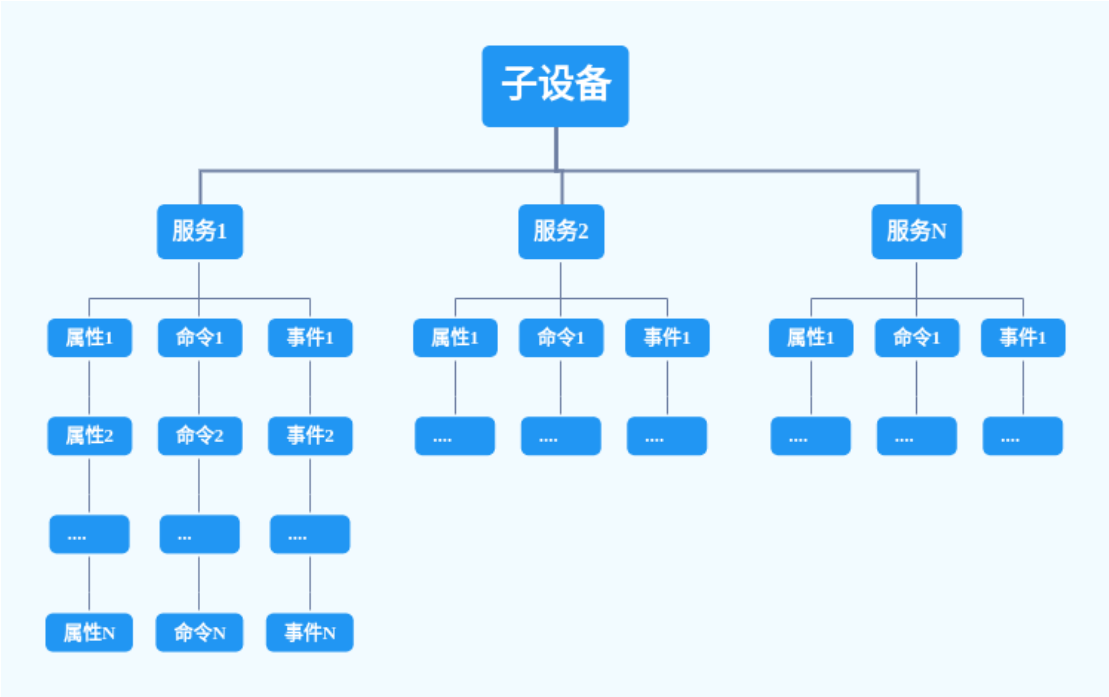


该图描述 DeviceOn/EIM 设备数据监控&告警的架构，子设备通过协议 Mapper 接入 EIM Client（请参考章节 8.1.5），DeviceOn/EIM Server 创建物模型并和远端的某个子设备进行绑定，绑定后，Server 端便可对远端的子设备进行相应的控制，也可以动态配置让远端子设备上报用户所需数据，还可以动态配置监控规则，让其产生事件，相关告警及事件处理机制等。该系统采用了物模型、设备影子、规则引擎、事件联动等物联网概念和子系统，使设备的数据接入更加广泛，下面我们将分别详细描述。

8.1. 基本概念

8.1.1. DeviceOn/EIM 的物模型

DeviceOn/EIM 的物模型是 DeviceOn/EIM 为 EIM Client 接入的子设备定义的数据模型，是 DeviceOn/EIM 子设备实体（如传感器，车载装置，以及各种边缘设备或者边缘控制器）的数字化描述。其从子设备属性、子设备命令、子设备事件三个维度描述一个子设备。其关系如下图所示。



下面我们分别描述各个概念：

功能类型	说明
子设备	Client 接入的设备 DeviceOn/EIM。
服务（Service）	子设备提供的功能服务模块。
属性（Property）	用于描述设备的各种属性、状态等，子设备的任何状态、特性都可以抽象成属性。例如子设备的 CPU 温度、CPU 占用率、内存容量、传感器值等。
命令（Command）	用于描述设备所能接受的命令，任何对设备的指令都可以抽象成设备的命令。如拉高 GPIO、打开风扇等，该部分未来会支持。DeviceOn/EIM
事件（Event）	用于描述设备产生的异常、通知等。我们通常用来将此作为设备告警等。

关于物模型的实现，各个平台大致相同，但有所差异。DeviceOn/EIM 的物模型实现依赖于子设备模型和子设备实例两个概念：

- **子设备模型**

子设备模型就是的物模型。DeviceOn/EIM 它是对一类设备的抽象，类似于面向对象编程中的类。它规定了这类设备有哪些属性、命令、支持的事件，通常创建一个子设备前，需要先创建对应的子设备模型。

- **子设备实例**

子设备实例是子设备模型的具体实现。它代表着一个具体的子设备，类似于面向对象编程中的对象。它为子设备模型中定义的属性、命令、事件赋予了实际的生命。一个模型可以创建很多个设备，但一个子设备只能对应一个子设备模型。一个子设备通常支持开启、关闭、状态上报、配置更新等常规操作，但设备模型是不具有的，子设备的产生依赖于设备模型。

8.1.2. 设备影子

设备影子是一个 JSON 文件，用于存储设备的在线状态、设备最近上报的设备属性值、应用服务器期望下发的配置。每个设备有且只有一个设备影子，设备可以获取和设置设备影子以此来同步设备属性值，这个同步可以是影子同步给设备，也可以是设备同步给影子。
设备影子上有 desired 区和 report 区：

- desired 区用于存储对设备属性的配置，即期望值。
  当需要修改设备的服务属性值时，可修改设备影子的 desired 属性值，设备在线时，desired 属性值立即同步到设备，如果设备不在线，待设备上线或上报数据时，desired 属性值同步到设备。
- report 区用于存储设备最新上报的设备属性值，即上报值。
  当设备上报数据时，平台刷新 report 属性值为设备上报的设备属性值。

DeviceOn/EIM 的子设备默认支持设备影子，用户通常是从设备影子中获取设备属性值。协议 Mapper 需要根据客户的需求周期性上报对应属性的值，关于协议 Mapper 请参考 8.1.5 小节。

8.1.3. 事件告警

事件告警主要是用户将设备模型中的事件和告警进行关联，当事件触发时会通过邮件或钉钉的方式给用户发出告警信息，让用户可以及时的处理告警。要实现事件告警主要有以下几步：[创建告警] -> [定义事件] -> [创建事件和告警的联动]。

8.1.4. 规则联动

规则联动主要包含触发器和执行动作两部分，触发器多为事件触发，执行动作有关机、重启、设置属性值、告警。

- 触发器：
  触发器分为特定设备的事件触发（指定设备）和设备模型的事件触发（绑定设备模型的所有设备）。
- 执行动作：
  当触发器满足条件触发时会自动执行相应的动作。主要包括关机、重启、特性设置、设警。等

8.1.5. 工业协议 Mapper 简介

Mapper 是子设备接入 EIM Client 的适配层，它负责将子设备协议转换并接入 EIM Client。Mapper 可以在集成在 Client 端，也可以单独部署在子设备端。
我们当前已经支持了 snmp、modbus、opcua，iec60875-5-104，西门子 S7， executer、restful 等 Mapper，**关于 Mapper 对协议的详细支持情况，请参考第 9 章**。 如果用户需要开发其它协议 Mapper，我们还提供了基于 C 的 MQTT 的 Mapper 开发 SDK，其地址位于

https://github.com/edgehook/ithings-edge-sdk-c, 用户可以基于该 SDK 及相关说明进行开发。

8.2. 子设备管理

子设备的管理包括子设备模型管理、子设备管理和设备拓扑图展示三个部分, 详细描述如下。

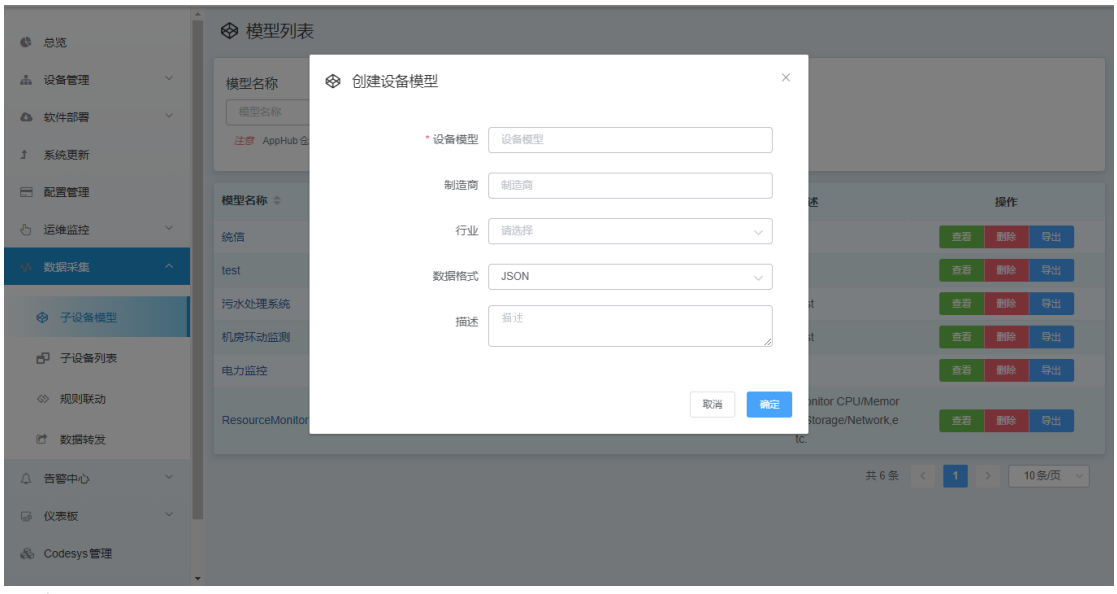
8.2.1. 子设备模型管理

子设备模型的管理主要分为子设备模型的增加、删除、编辑、导入导出等。子设备模型创建有手动新建和 excel 导入两种方式， 下面我们将分别描述。

8.2.1.1. 手动创建子设备模型

1. 新建子设备模型

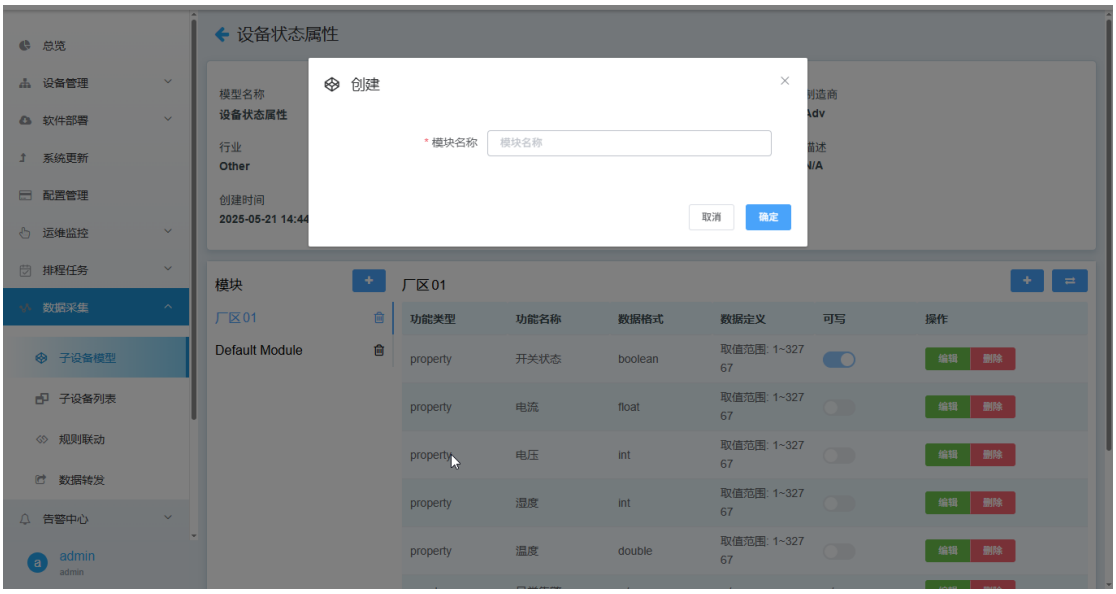
进入子设备模型列表界面， 点击添加， 如下弹框中填写相关信息后， 完成新建。



2. 新建模块

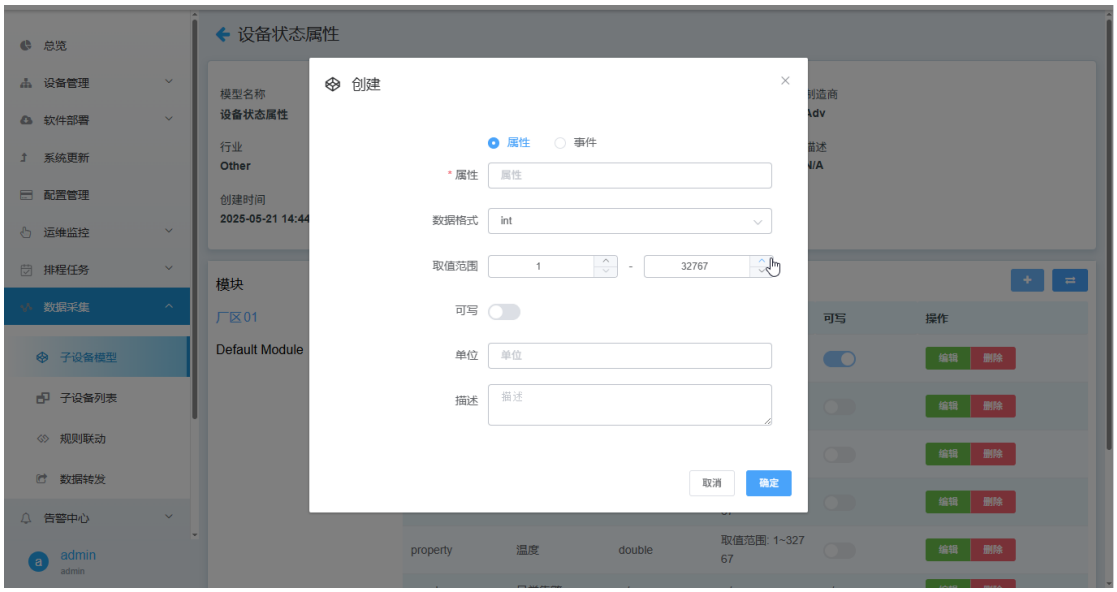
点击子设备模型名称，进入子设备模型的详情信息界面，如下，可以在此新建模块、事件和属性。

击模块上方+按钮，输入模型名称，如下所示，点击确定完成新建。



3. 新建属性

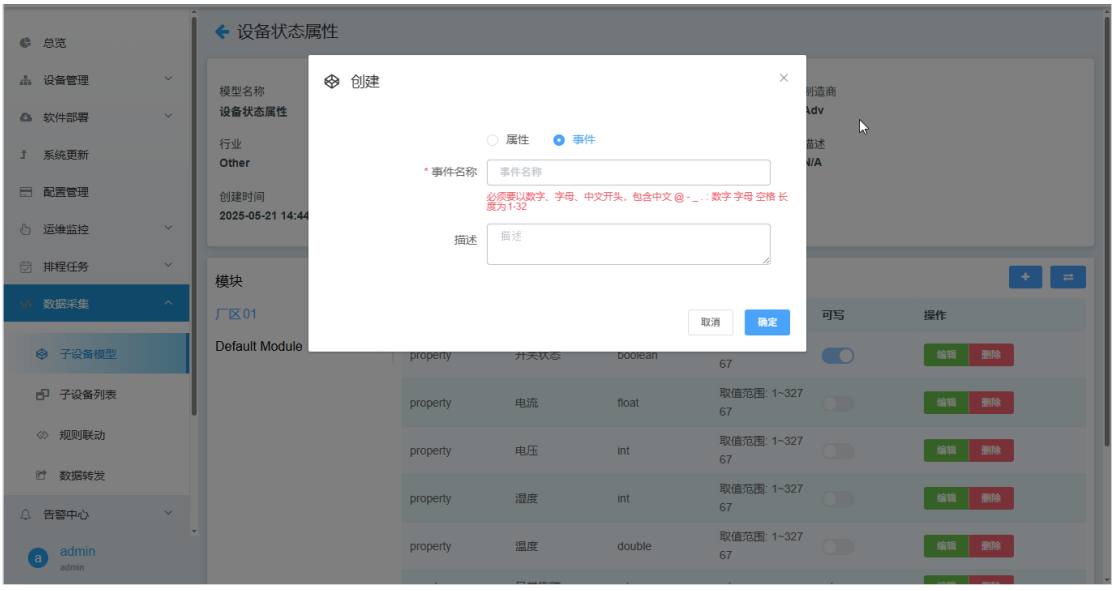
点击+按钮，选择属性，填写属性的相关信息，如图所示。



- 属性：属性名称
- 数据格式：新建属性的数据格式，包括 int、string、float、double、boolean、byte 等
- 取值范围：属性的取值范围
- 可写：是否可写
- 单位：属性值的单位
- 描述：其他信息

4. 新建事件

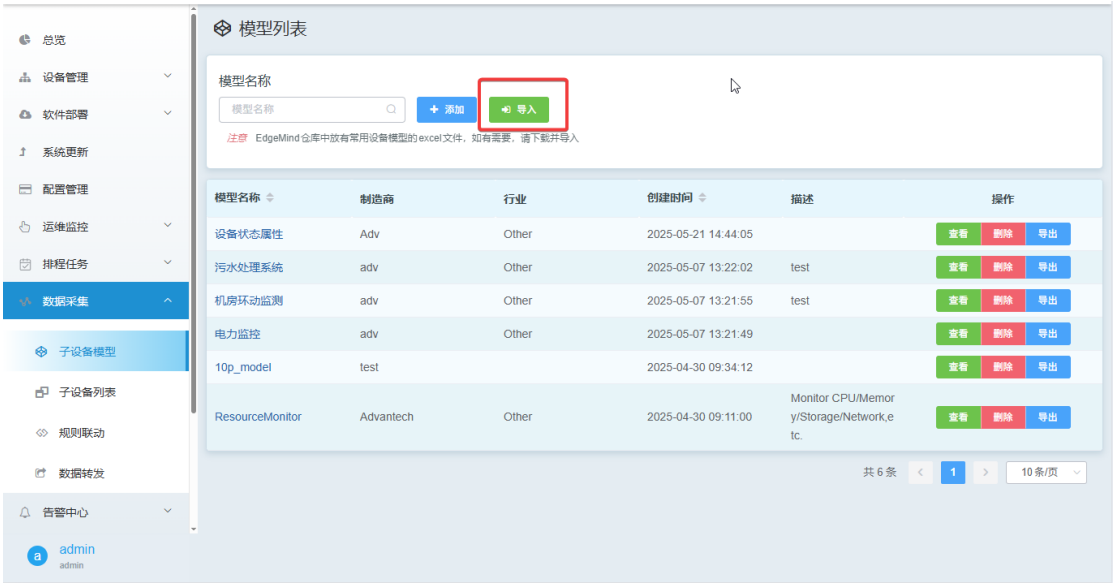
点击+按钮，选择事件，填写事件的相关信息，如图所示。



- 件名称：事件的名称
- 描述：其他信息

8.2.1.2. excel 导入子设备模型

模型导入是和模型导出功能对应。
点击导入，选择需要导入模型的 excel 文件，即可。

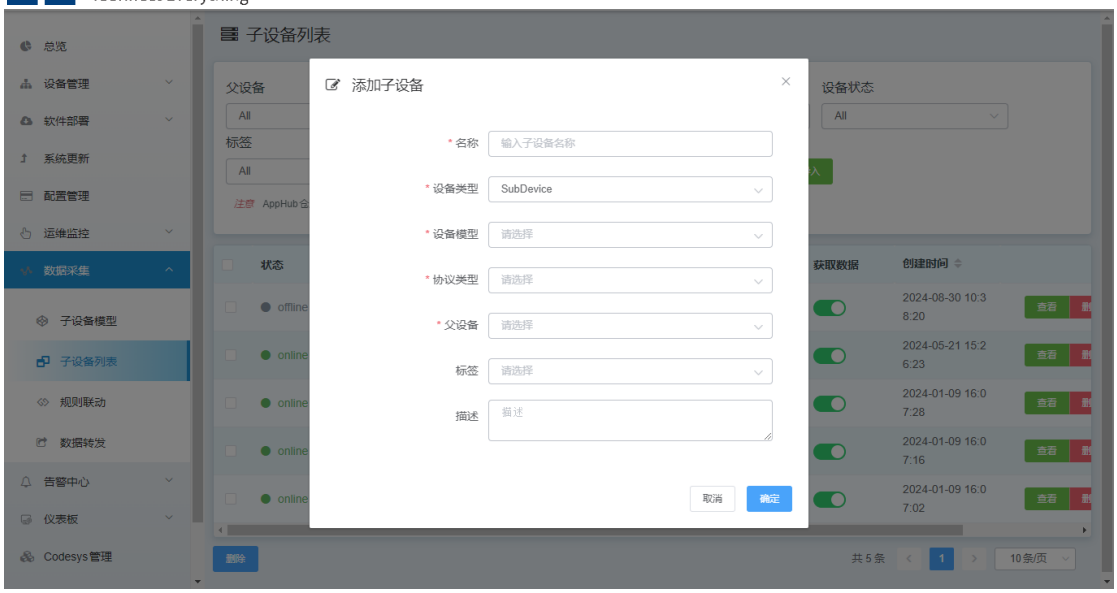


8.2.2. 子设备管理

同子设备模型一样，子设备创建也分为两种方式：手动创建 & excel 导入。详细描述如下，子设备就是模型的实例，通过创建子设备和配置子设备，就可以通过无代码方式快速采集到相关数据，目前子设备支持多种协议的数据采集，包括 Modbus, SNMP, OPC-UA, IEC60875-5-104，西门子 S7, executer、RESTful 等协议，详细可以参阅下个章节工业协议的介绍，在这个章节，我们以 executer 协议为例，对子设备的创建和配置进行说明。其他协议配置方式类似，只是具体配置的参数有差异，非常方便，无需编程，就可以实现数据的采集。

8.2.2.1. 手动创建子设备

1.新建子设备
点击添加，填写子设备信息的相关信息，点击确定完成新建，如下所示。

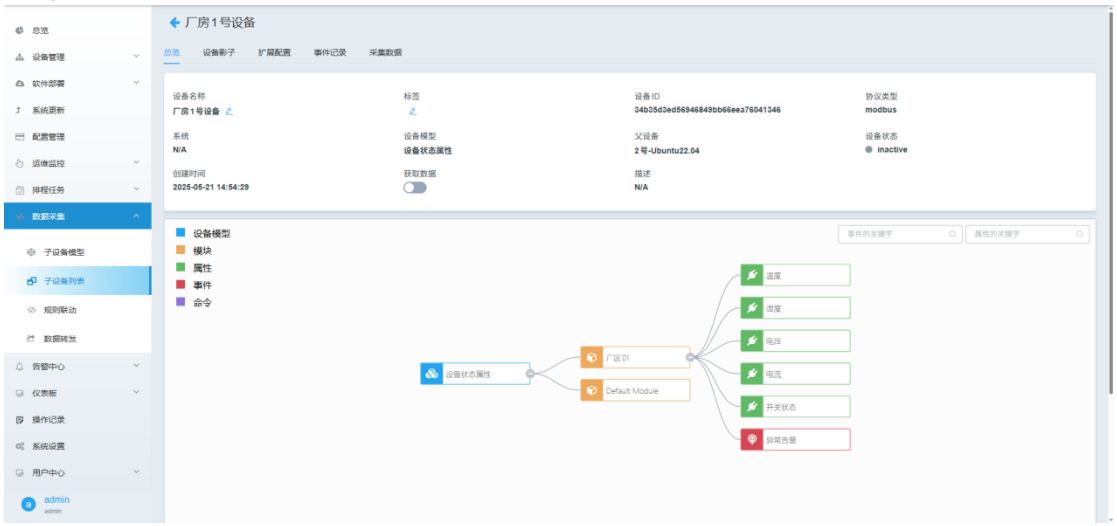


- 称：子设备名称
- 设备类型：SubDevice
- 设备模型：与子设备绑定的设备模型
- 协议类型：execute/modbus/restful 等
- 父设备：选择一个父设备挂载
- 描述：其他信息

2.基本属性配置

子设备的配置包括基本属性配置、具体协议的属性配置、还有事件配置。以下我们以 execute 协议为例，具体介绍一下子设备的配置。

首先点击子设备名称，在总览页签下，可以查看子设备基本信息和模型拓扑图。



点击拓扑图的设备模型名称，设置采集间隔时间（以及其他项完成设置）。

3. 其他属性配置

点击属性，在右侧的弹框中进行属性设置。

4. 事件配置

点击事件，在右侧的弹框中进行事件的触发条件和上报周期等配置。

- 条件：多个规则之间的关系是或还是且
- 规则：用户可以手动添加多个规则
- 检测周期：多长时间检测一次
- 持续周期：连续多少个检测周期都满足事件的触发条件，0 代表 1 个检测周期，1 代表 2 个检测周期，以此类推
- 报警：报警消息绑定（消息的内容需要在告警中心定义）
- 上报类型：一次或者持续上报

5. 事件/属性的配置查看

点击进入扩展配置，可对配置的属性和事件信息进行查看和删除。

6. 设备影子和采集数据查看

设备影子和采集数据都可以查看采集的数据，不同点在于采集数据是从时序数据库中取的，存取数据效率更高、存取数据量更多。2 种方式都支持图表和列表查看。

注：在 admin 权限下，系统设置页的 InfluxDb 中可配置时序数据库。

7. 子设备的事件列表

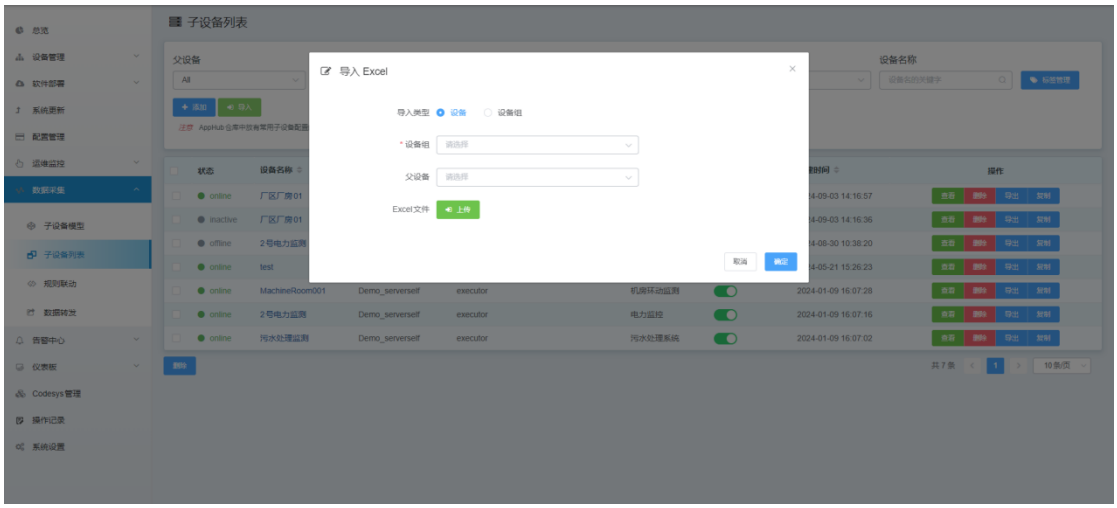
8.2.2.2. 子设备导入导出

一个子设备有大量的属性，事件和命令，对于同一种子设备，手动创建，配置会花费很大的时间，而且有很多重复性的配置工作。excel 导入导出提供了快速复制一个子设备的方式。复制完子设备后，针对该设备配置对应的不同配置项，即可快速创建子设备。

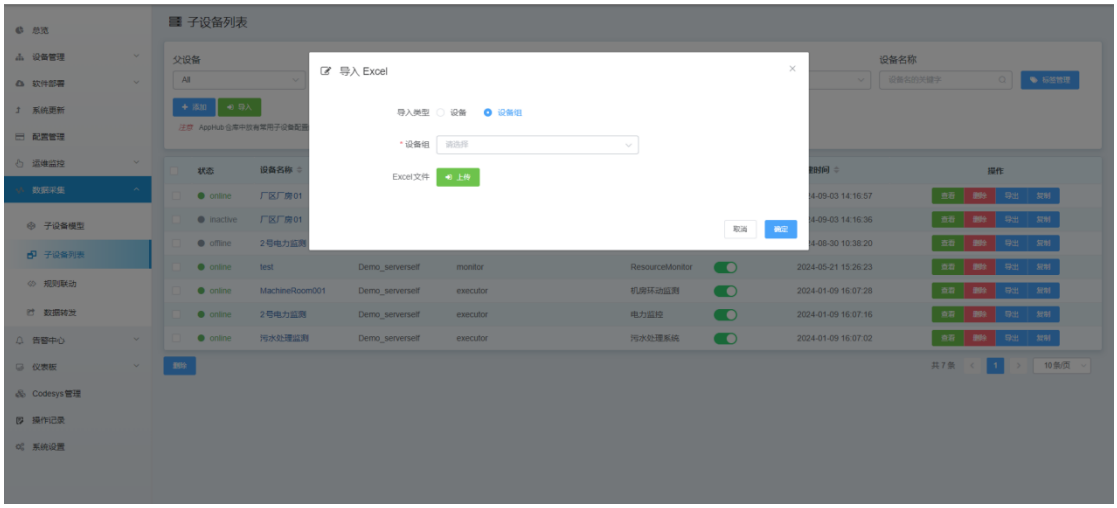
1. excel 导入

excel 导入分为 2 种：设备类型和设备组。

设备类型：在设备组下的某一个设备下，导入子设备。首先进入子设备界面，点击导入，如下弹框中，选择设备组，父设备，excel 文件，完成导入子设备。

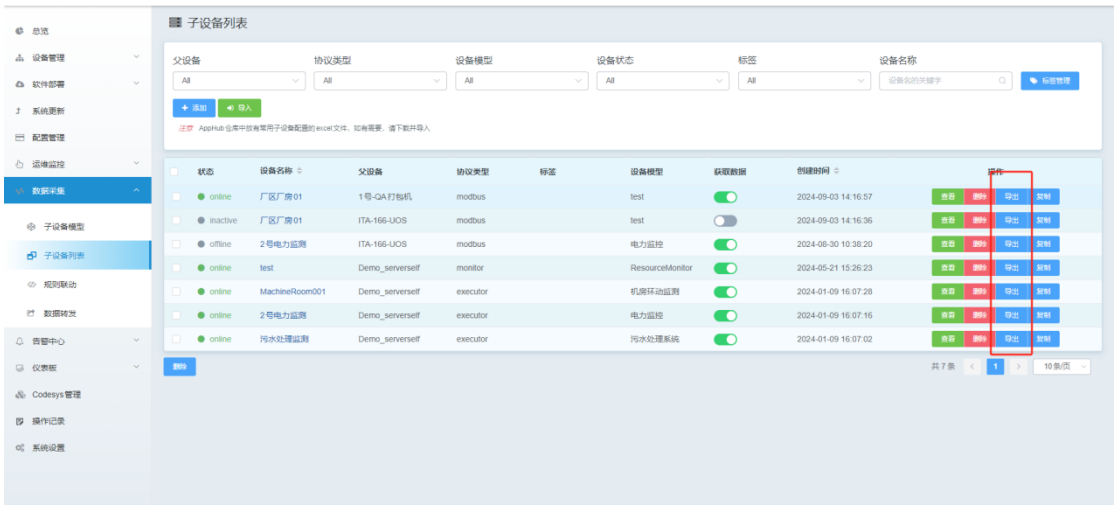


设备组类型：在设备组下的每一个设备下，导入子设备。如下所示，选择设备组，excel 文件，点击确定，完成导入。



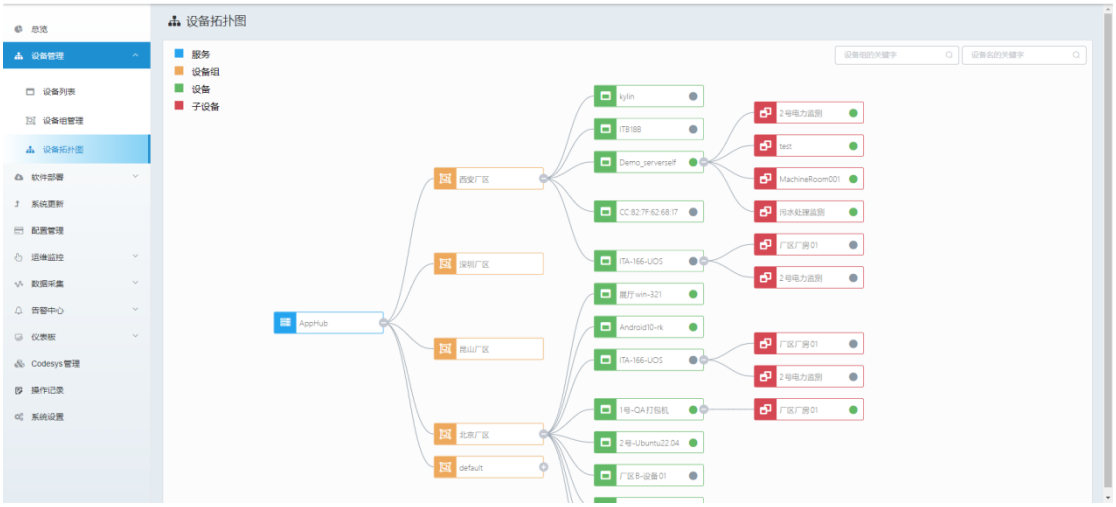
2. excel 导出

点击子设备列表的导出按钮，即可导出。



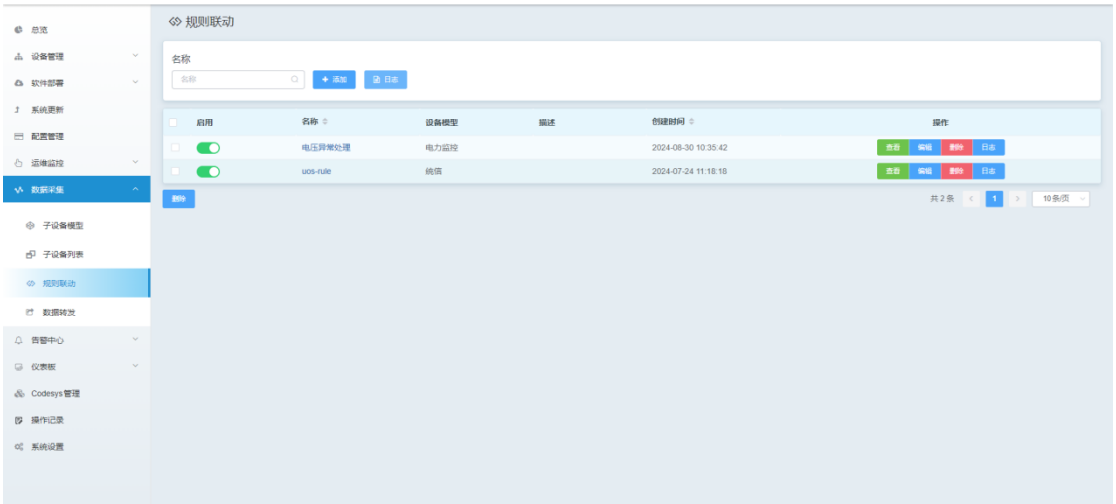
8.2.3. 设备拓扑图

设备拓扑图描述了 DeviceOn/EIM 的所有设备组，设备组下的所有 EIM Client 设备，以及 EIM Client 下所接入的所有子设备的拓扑关系图。从设备-设备拓扑图进入，即可查看设备的拓扑图，也支持设备名称的搜索。



8.3. 规则引擎，设备联动和数据转发

规则引擎包括规则联动和数据转发。规则联动指将事件和动作进行关联，从而实现场景联动；数据转发指将属性采集的数据和事件通过中间件转发到其他 Server。两种功能都支持新建、启动、编辑、删除、日志查看。



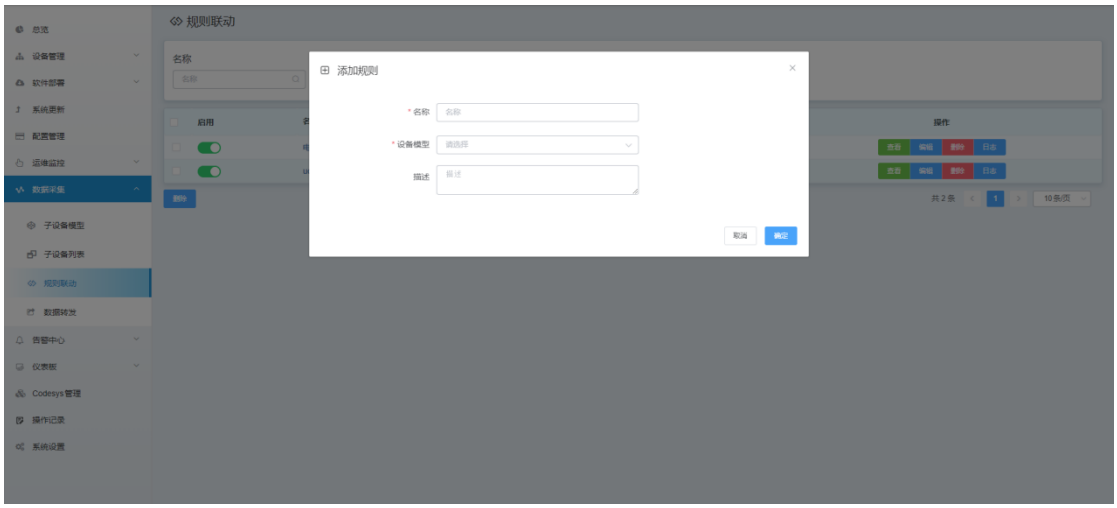
8.3.1. 规则联动

规则联动由触发器（事件）和执行动作组成，当事件触发时会执行规则中定义的动作，包括

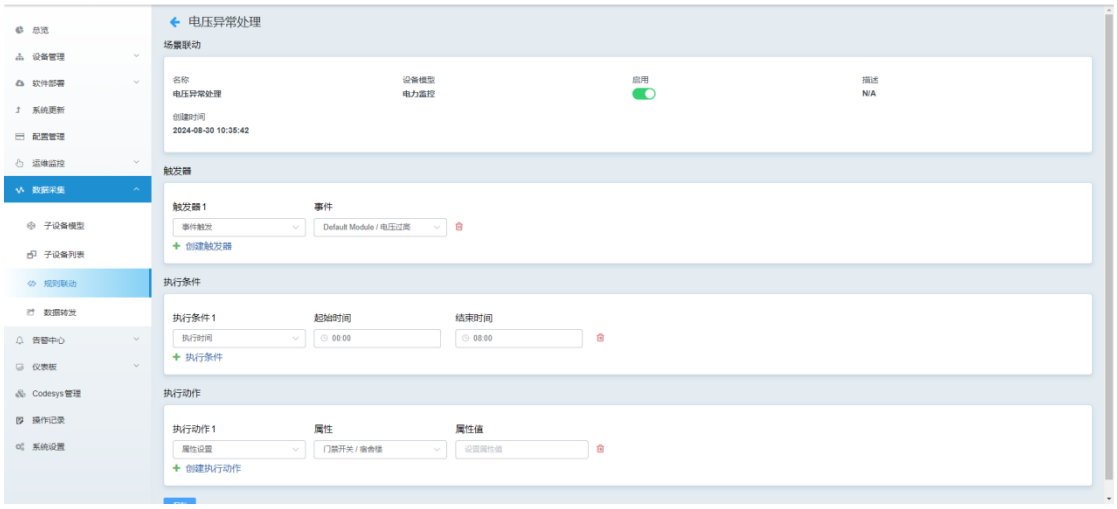
报警、属性设置、设备控制等。

1.创建关联规则

点击添加，输入规则名称，选择设备模型。



2.关联触发器（事件）和执行动作



触发器

触发器类型分为事件触发和特定事件触发，区别在于特定事件触发是指定子设备的事件才能触发。

- 执行条件

设置规则连动执行的时间范围

- 执行动作

执行动作分类：报警、属性设置、特定属性设置、设备控制、特定设备控制。

执行动作里的特定，是指事件触发的子设备为特定子设备时才会执行动作。

执行动作里的非特定，是指模型绑定的事件触发时，由触发事件的子设备执行动作。

3.启用规则

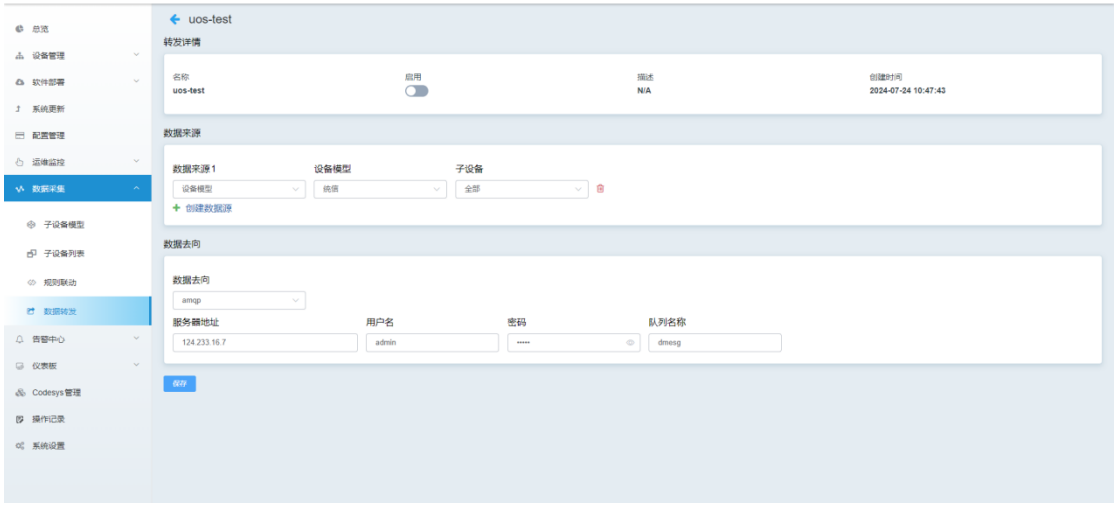
点击列表中启用开关(或规则详情页的启用开关)。

8.3.2. 数据转发

1. 创建转发规则

点击添加按钮，输入数据转发记录填写信息，点击确定，完成新建。

2.关联数据来源和数据去向。



- 数据来源：
  支持不同的设备模型。
- 数据去向：
  数据转发的目的是把数据转发给其他 Server 使用，目前支持 amqp、kafka 和 mqtt 协议，后期会加入其他。

3.启用规则

点击列表中启用开关(或规则详情页的启用开关)。

8.4. 嵌入式轻量仪表板和数据可视化

DeviceOn/EIM 内置的嵌入式轻量仪表盘功能，采用无代码方式，进行数据可视化，无需借助其他任何第三方产品，内含丰富多元的组件，通过简单的配置即可将 DeviceOn/EIM 子设备采集上来的数据进行自定义展示，从而实现了数据采集，监控，联动，可视化等功能，可以适用于不同自动化场景。

除了在线编辑外，用户可以预览、发布、拷贝、同步手机等多种方式对仪表盘进行管理，发布后的仪表盘可以再其他任意电脑上查看。

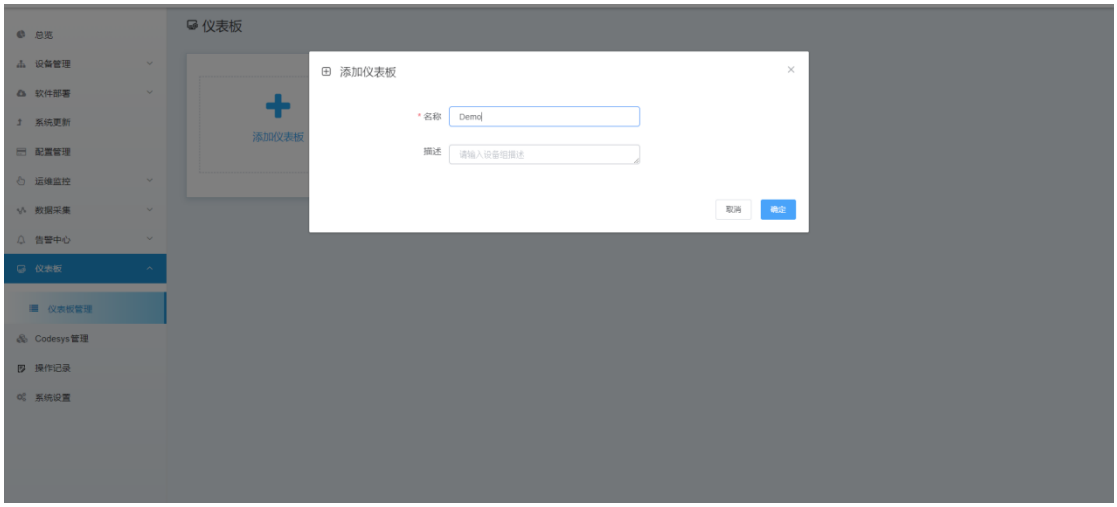
目前支持仪表、图表、列表等多种插件，以及图库涉及工业、交互、电力等，同时支持视频、图片、远程网页嵌入等丰富组件。

支持多种数据聚合，包括原始数据、平均值、最大最小值、求和、中位数、标准偏差等等。

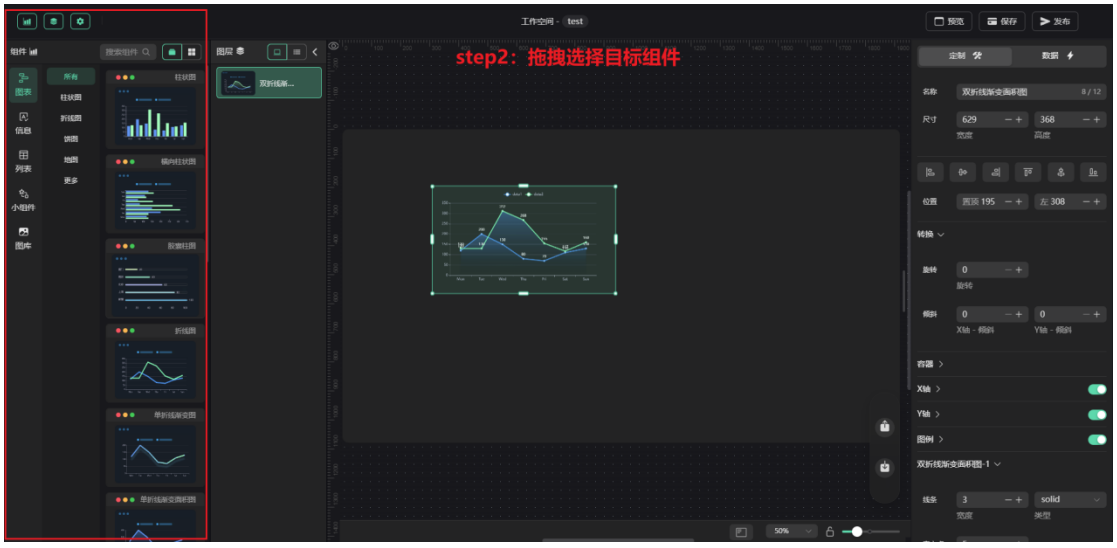
该仪表盘组件非常轻量，可运行在各种 X86,ARM 架构的嵌入式设备中，占用很少的系统资源，运行非常顺畅。使用于各种嵌入式 IoT 设备的数据可视化需求。

下面以呈现几个数据到折线图中为例，简单说明使用方法，仅需三步即可：

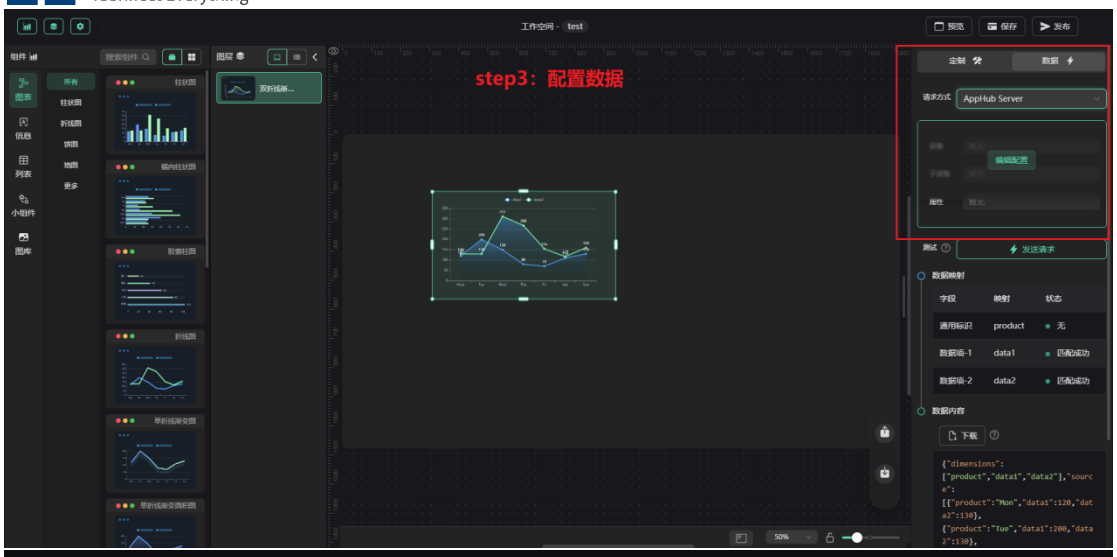
step1：新建仪表盘

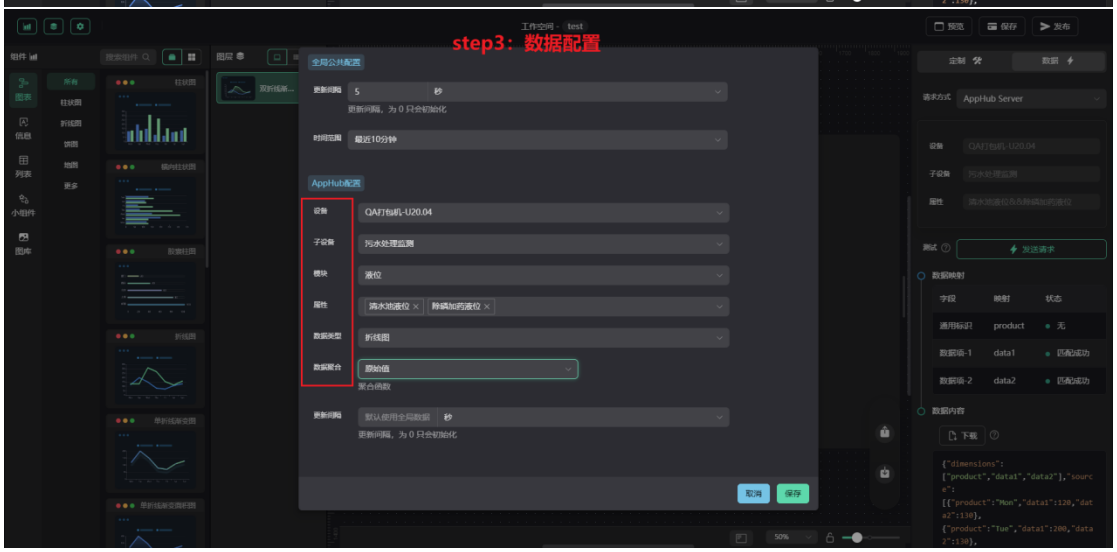


step2：拖拽选择目标组件



step3：数据配置

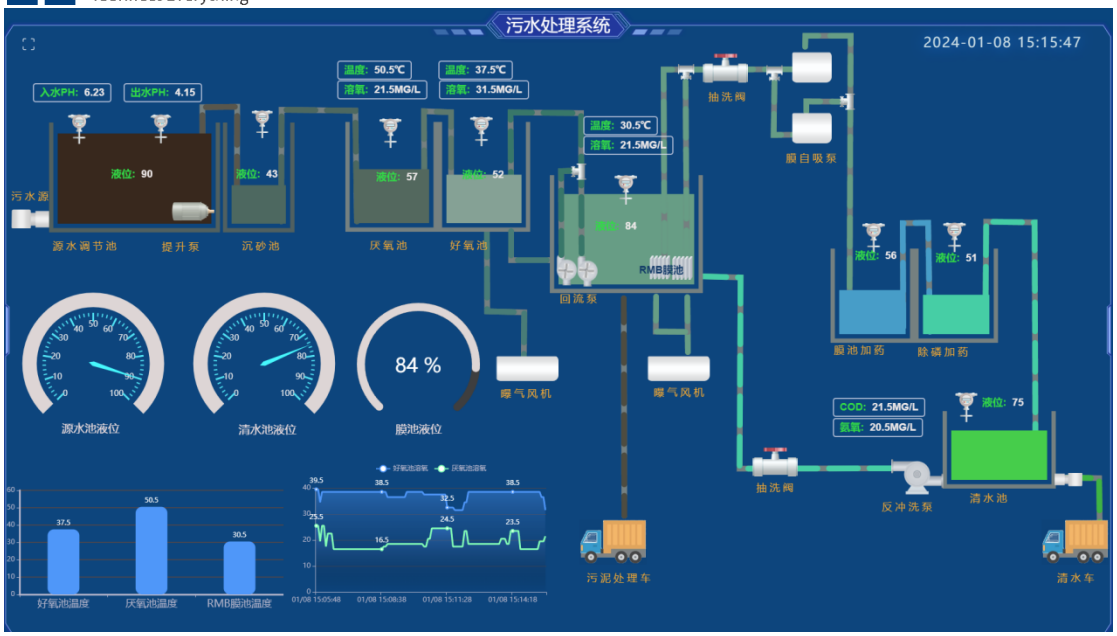




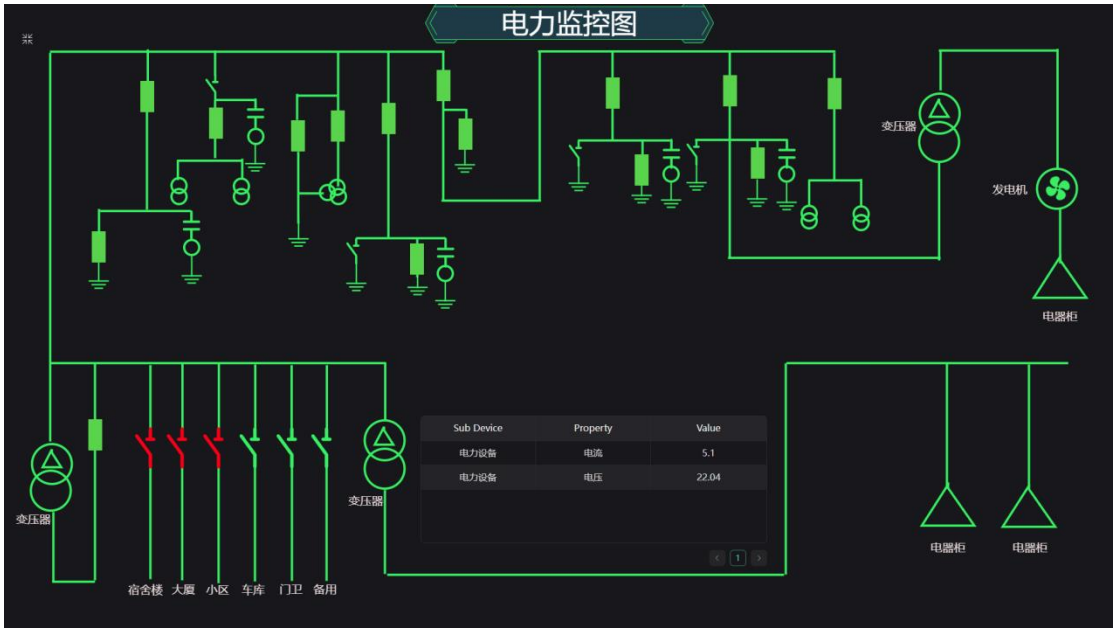
至此，子设备采集到的数据就以折线图的形式呈现出来了，下面是预览页面

如果希望更丰富的呈现，多个组件的进行组合即可，下面给出三个 demo:

demo1：污水处理系统



demo2：电力监控系统



demo3：机房动环监控系统

8.5. 边缘数据转发

除了前面提到的，我们在 DeviceOn/EIM Server 端可以实现数据的转发，另外，在 Client 端，也可以将采集到的数据直接转发到其他的软件系统，如赋华的 APM，iMachine,iEMS 等系统中。

9. 支持的工业协议

EIM Client 已经内部集成了 Modbus, SNMP, OPC-UA, IEC60875-5-104，西门子 S7, executer、RESTful 等协议 Mapper, 其负责将相关工业协议接入 DeviceOn/EIM. 下面我们将分别描述。通过这些协议 Mapper，就可以通过无代码的方式，将工业现场不同协议的数据采集到 DeviceOn/EIM 系统，并实现对数据的监控，异常告警，可视化，转发等等功能，使 DeviceOn/EIM 有很灵活的弹性和扩展功能，从而满足各种工业场景的需求。下面会介绍每个协议在子设备中的配置和采集。

9.1. Modbus 协议

Modbus Mapper 负责将工业现场 Modbus 协议的数据转换为 DeviceOn/EIM 物模型可以识别的数据格式，其主要用于工业现场数据的采集。对于 RTU 协议，一个串口设备节点通常只能抽象成一个子设备，Modbus TCP 则没有这样的限制。

9.1.1. Mapper 支持的功能特性

Modbus Mapper 对 Modbus 协议的支持情况如下所述：

- **支持 Modbus RTU 和 Modbus TCP**;
- **支持的 Modbus 功能码如下表所示：**

功能码描述	描述
ReadCoils（0x01）	读线圈
ReadDiscreteInputs(0x02)	读离散寄存器
ReadHoldingRegisters(0x03)	读保持寄存器
ReadInputRegisters(0x04)	读取输入寄存器
WriteMultipleCoils(0x0F)	写线圈
WriteMultipleRegisters(0x10)	写多个寄存器

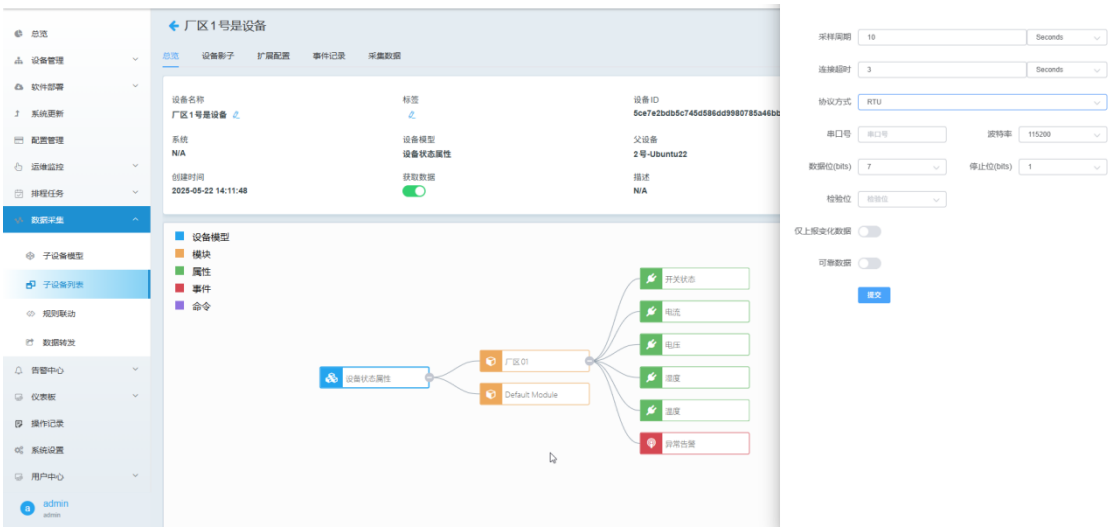
- **支持灵活配置字节序（大小端，byteSwap 等）；**
- **支持对原始数据进行相关公式计算；**

9.1.2. Modbus 子设备操作描述

DeviceOn/EIM 支持 Modbus RTU 和 Modbus TCP。对于 RTU 协议,**一个串口设备通常只能抽象成一个子设备**, Modbus TCP 则没有这样的限制。下面我们将描述 Modbus 的这两种设备。

1. 创建 Modbus RTU 设备

创建 Modbus RTU 设备需要知道远端（EIM Client 端）Modbus 协议栈基于哪个串口，以及串口的通信参数。



串口号代表 Modbus 设备的设备节点，Windwos 中为 COM0、COM1 等，Linux 下为/dev/ttyS*；波特率、数据位、检验位、停止位要和 EIM Client 接入的实际子设备（Modbus slave 设备）保持相同；Modbus 的采集周期依赖于 Modbus 的实际波特率和采集的点数，如果用户设置的采集周期远远小于 Modbus 的采集周期，这种结果是未定义的。

Modbus 的属性配置如下图所示：

配置属性的意义就是，将该物模型定义的设备属性和远端实际的 EIM Client 子设备中的相关寄存器绑定起来。

我们抽象的 Modbus 扩展配置有：

- **Modbus slave ID**
- **功能码**

  功能码代表支持哪种类型的 Modbus 寄存器，当前支持 Coils、Holding register、input register、discrete input register 等；
- **起始地址**

  起始地址为 Modbus 的相关寄存器的起始地址，注意其为 16 进值；
- **读写长度/数量**

  数量表示读取/写入 Modbus 对应寄存器数量。这里需要**特别注意**的是，属性的数据类型要和读取的寄存器数量匹配。比如，数据是 int 类型，对于 holding Register，需要的数量是 2 个，也可以是 1 个。大于 2 个，只会返回前两个 holding Register

的值，因为 int 只能装 4 个字节。当然 1 个数量的 holding Register（2 Byte）也是可以的。例如，int8 的只能读取 8 个或者小于 8 个的 Coils 或者 discrete input register 的。不能读取 1 个 holding register。

对于 float 类型的数据，目前只支持 float32 或者 float64 类型的数据。

- **大小端模式**

  当前 Mapper 支持的模式，大端模式，小端模式， 大端 byte Swap 模式， 小端 Byte Swap 模式。一般情况下，用户选择默认模式即可。

- **公式**

  当前公式只有两种，一种为原始数据。一种为 f(x)=ax+b 的函数，用户输入 a 和 b 的值，mapper 会将原始数据根据该计算公式产生对应结果。a 和 b 可以是整数，也可以是浮点数。

2. 创建 Modbus TCP 设备

创建 Modbus TCP 设备，需要知道远端（EIM Client 端）Modbus 协议栈基于哪个网络 IP 和端口。如下图所示：

TCP 的属性配置同 RTU 完全一致，请读者参考以上 RTU 章节描述。

9.2. SNMP 协议

SNMP 是广泛应用于 TCP/IP 网络的网络管理标准协议，目前协议模块支持 v2c 版本，该版引入了 GetBulk 和 Inform 操作，支持更多的标准错误码信息和数据类型。该协议模块能够将标准 SNMP 协议数据类型转换成 DeviceOn/EIM 物模型可识别的数据类型，并支持对工业现场 snmp 网络节点的数据采集、snmp 事件（Trap）上报以及 DeviceOn/EIM 规则事件上报等。下面我们将描述 SNMP 的子设备。

1. 创建 SNMP 设备

创建 SNMP 设备需要连接远端设备（网络节点）的 IP 地址以及采集间隔，并且要求远端设备（网络节点）已经开启 SNMP 服务。

SNMP 的属性配置如下图所示：

Trap 机制不属于 Client 对被 DeviceOn/EIM 管理设备（网络节点）的操作，它是被管理设备的自发行为。当被管理设备达到告警的触发条件时，会通过 SNMP 代理进程（用户 SNMP 进程）向 EIM Client 发送 Trap 消息，告知设备的异常情况，便于管理人员处理。

SNMP 的 Trap 上报机制设置如下图所示：

9.3. OPC-UA 协议

OPC-UA 的全名是 OPC Unified Architecture（OPC 统一架构）。简单来说，它是一种用于不同设备和系统之间进行通信的技术规范。通过 OPC-UA，各种设备和系统可以互相交流和共享数据，实现更高效的工业自动化。

OPC-UA 采用了现代化的网络通信技术，基于 Web 服务和互联网技术的基础之上。它使用了一种称为"面向对象"的方式来描述设备和系统之间的通信。在 OPC-UA 中，每个设备和系统都被抽象为一个对象，这些对象有自己的属性、方法和事件。通过读取和写入对象的属性，调用对象的方法，以及监听对象的事件，设备和系统可以进行数据交换和控制操作。此外，OPC-UA 还提供了多种传输方式，如 TCP/IP、HTTPS 等，可以根据实际情况选择最适合的传输方式。同时，OPC-UA 还提供了安全机制，确保通信的安全性和可靠性。

OPC-UA 被称为数字化进程中自动化的里程碑，原因是 OPC-UA 是一项开放标准，适用于从机器到机器间的水平通信和从机器到云端的垂直通信。得益于具有平台独立性、强大的安全基础以及信息模型这些性质，OPC-UA 很好地赋能数字化，逐渐成为大家关注的焦点。

9.3.1. Mapper 支持的功能特性

OPC-UA Mapper 对 OPC-UA 协议的支持情况如下所述：

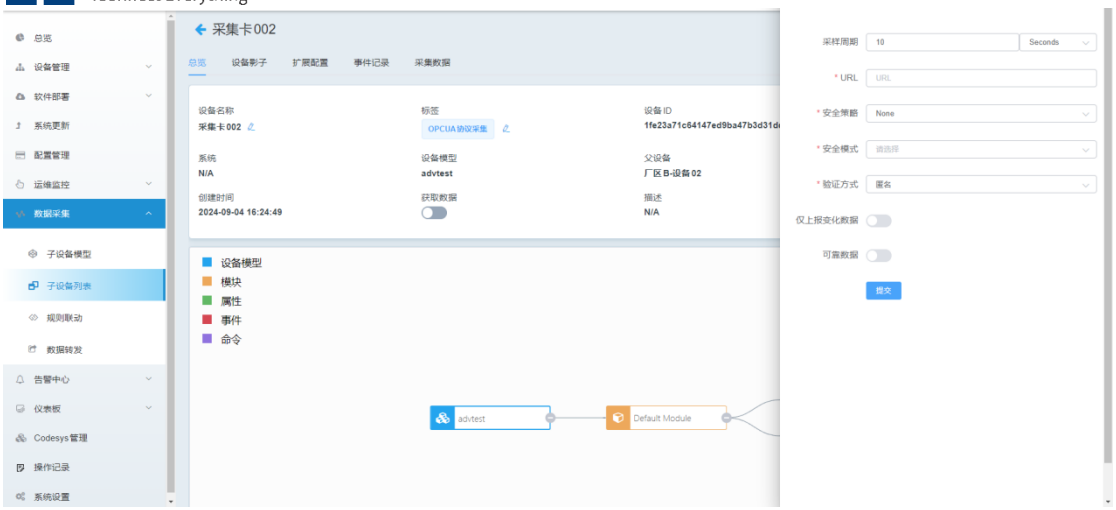
- **Mapper 实现了 OPC-UA Client 的功能；**
- **支持对变量节点的数据采集，和相关值的设定；**
- **方法节点调用方法在未来会支持；**
- **支持的安全策略有 None, Basic128Rsa15, Basic256, Basic256Sha256, Aes128Sha256RsaOaep, Aes256Sha256RsaPss 等，支持的消息的安全模式有 None，Sign，SignAndEncrypt 等；**
- **支持的用户身份验证包括：匿名认证，用户名/密码认证，和证书认证等；**
- **只支持基于 TCP 的 OPC-UA 传输；**

9.3.2. OPC-UA 子设备操作描述

下面我们将描述 OPC-UA 的子设备创建 & 属性绑定。

1. 创建子设备

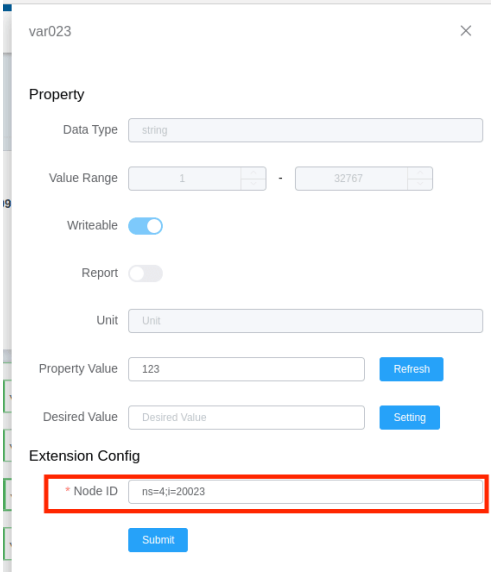
创建子设备时，**Protocol Type 要选择 opcua，**就会创建一个 opcua 的子设备。创建完子设备后，需要绑定设备，其需要绑定的信息如下图所示：



URL 为 OPC-UA Server 的地址，其格式必须是 **opc.tcp://**ip:port 的固定格式。其中安全策略，安全模式，认证方式，需要根据 OPC-UA 的 Server 的相关要求填写。

2. 属性绑定

OPC-UA 的属性绑定相对简单，只需要设置变量节点的 NodeId 即可，Mapper 将会读取该变亮化节点的 Value 值。其显示如下图:



该 NodeId 的格式，可以参考 UaExpert 中 Attribute 窗口中的 NodeId 格式，也可以直接 Copy 它。

9.4. IEC60875-5-104 协议

IEC60870-5-104 是一种电力自动化系统中常用的通信协议，使用 TCP/IP 协议作为底层通信协议，用于监视和控制电力系统中的各种设备，如变电站、发电机、开关等。国网电力系统使用 EC60870-5-104 来实现远程终端单元(RTU)和监控中心之间的数据交换。通过这个协议，监控中心可以实时获取 RTU 的运行状态、参数和告警信息，并下发控制指令到 RTU。这种双向的数据通信使得国网电力系统能够实现远程监控和控制，提高了系统的运行效率和可靠性。

IEC 60870-5-104 采用了面向对象的通信方式，数据传输高效可靠。它提供了灵活的数据组织和传输机制，可以支持各种监测和控制需求。作为国网电力系统的通信标准，1EC60870-

5-104 已经在该领域得到了广泛应用。它不仅被用于国网电力系统内部的通信，还被用于与其他电力系统以及能源调度中心的通信。通过来用统一的通信协议，国网电力系统能够与其他系统无缝集成，实现信息互通和资源共享，提升电力系统的整体运行效能。

9.4.1. Mapper 支持的功能特性

IEC 60870-5-104 Mapper 对 IEC 60870-5-104 协议的支持情况如下所述：

- **Mapper 实现了 IEC 60870-5-104 控制站(client)的功能；**
- **Mapper 自动实现总召唤，电能脉冲召唤，时钟同步等功能；**
- **满足中华人民共和国国家电网 DL/T634.5104-2009 实施细则标准；**
- **支持平衡传输模式；**
- **Mapper 支持的信息对象（ASDU）如下表所示：**

监视方向上的过程信息：

类型数字	类型标识	描述
1/30	M_SP_NA_1/M_SP_TB_1	**不带/带**时标 CP56Time2a 的单点信息
3/31	M_DP_NA_1/M_DP_TB_1	**不带/带**时标 CP56Time2a 的双点信息
5/32	M_ST_NA_1/M_ST_TB_1	**不带/带**时标 CP56Time2a 的步位置信息
7/33	M_BO_NA_1/M_BO_TB_1	**不带/带**时标 CP56Time2a 的 32 比特串
9/34	M_ME_NA_1/M_ME_TD_1	**不带/带**时标 CP56Time2a 的归一化测量值
11/35	M_ME_NB_1/M_ME_TE_1	**不带/带**时标 CP56Time2a 的标量化测量值
13/36	M_ME_NC_1/M_ME_TF_1	**不带/带**时标 CP56Time2a 的浮点型测量值
15/37	M_IT_NA_1/M_IT_TB_1	**不带/带**时标 CP56Time2a 的累计值
20	M_PS_NA_1	带状态检出的成组单点信息
21	M_ME_ND_1	不带品质描述的归一化测量值
38	M_EP_TD_1	带时标 CP56Time2a 的继电器保护装置事件
39	M_EP_TE_1	带时标 CP56Time2a 的继电器保护装置成组启动事件
40	M_EP_TF_1	带时标 CP56Time2a 的继电器保护装置成组输出电路信息

控制方向上的过程信息：

当前 Mapper 只支持简单的直控命令，不支持选控命令（会在未来支持）。

类型数字	类型标识	描述
45/58	C_SC_NA_1/C_SC_TA_1	**不带/带**时标 CP56Time2a 的单命令
46/59	C_DC_NA_1/C_DC_TA_1	**不带/带**时标 CP56Time2a 的双命令
47/60	C_RC_NA_1/C_RC_TA_1	**不带/带**时标 CP56Time2a 的步调节命令
48/61	C_SE_NA_1/C_SE_TA_1	**不带/带**时标 CP56Time2a 的设点命令，归一化值
49/62	C_SE_NB_1/C_SE_TB_1	**不带/带**时标 CP56Time2a 的设点命令，标量值
50/63	C_SE_NC_1/C_SE_TC_1	**不带/带**时标 CP56Time2a 的设点命令，短浮点值
51/64	C_BO_NA_1/C_BO_TA_1	**不带/带**时标 CP56Time2a 的 32 比特串

控制方向上的系统信息：

Mapper 在 Start 时会自动调用这些命令，如果需要强制调用这命令，可以考虑重启设备。

类型数字	类型标识	描述
100	C_IC_NA_1	总召唤命令
101	C_CI_NA_1	电能脉冲召唤命令
103	C_CS_NA_1	时钟同步命令

控制方向上的参数：

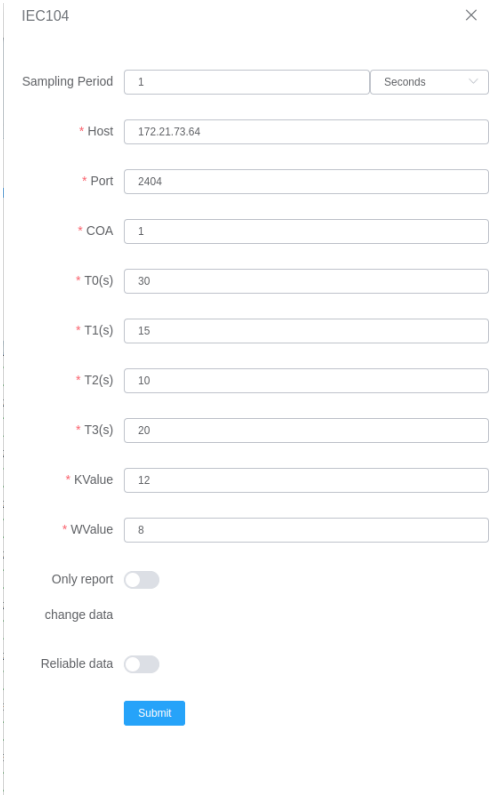
类型数字	类型标识	描述
110	P_ME_NA_1	归一化测量值
111	P_ME_NB_1	标量化测量值
112	P_ME_NC_1	浮点测量值

9.4.2. IEC60875-5-104 子设备操作描述

下面我们将描述 IEC104 的子设备创建 & 属性绑定。

1. 创建子设备

创建子设备时，**Protocol Type 要选择 iec104，**就会创建一个 iec104 的子设备。创建完子设备后，需要绑定设备，其需要绑定的信息如下图所示：



Host 和 Port：是受控站（Server）的地址和端口。

COA：受控站（Server）的站点地址，根据受控站的实际情况填写。

T0: TCP 连接建立的最大超时时间，根据受控站的实际情况填写，不清楚就使用默认值。

T1: 帧接收确认最长超时时间，超过此时间立即关闭连接。根据受控站的实际情况填写，不清楚就使用默认值。

T2: 发送一个接收确认的最大时间，根据受控站的实际情况填写，不清楚就使用默认值。

T3: 触发"TESTFR"保活的空闲时间值，根据受控站的实际情况填写，不清楚就使用默认

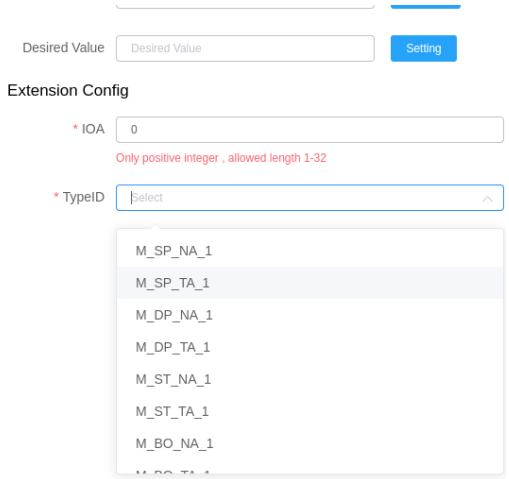
值。

K 值： I-frames 发送未收到确认的帧数上限， 一旦达到这个数，将停止传输。根据受控站的实际情况填写，不清楚就使用默认值。

W 值： 接收端最迟在接收了 w 次 I-frames 应用规约数据单元以后发出认可，w 不超过 2/3 K 值。根据受控站的实际情况填写，不清楚就使用默认值。

2. 属性绑定

IEC104 采集的目标是信息对象的值，既然是信息对象，就需要 IOA 和 TypeId，如下图所示：



IOA： 表示信息对象的地址

TypeID： 信息对象的类型描述符号，详细支持情况见 **Mapper 支持的功能特性**章节。

注意：

- 质量描述符在该属性的 ErrorMessage 字段以字符串描述，如 IV，NT，SB，BL，SPI 等（其详细内容参考 IEC 60870-5-104 协议文档所述）。其上报的值是单纯的 Value（例如单点信息的值，双点信息值等）。
- 累计值必须是一个 json 字符串，用户根据需要自行解压出想要的值。
- 事件类型的信息对象值是一字符串，其格式固定为 sprintf("%s; Elapsed Time: %d", SEP 值, elapsedTime)。
- 参数类型的属性只能设定参数，无法看到参数当前的值。

9.5. 西门子 S7 协议

西门子 Step7（简称 S7）协议是一种通信协议，主要用于西门子 S7 系列 PLC 的内部集成。它是一种运行在传输层之上的（会话层/表示层/应用层）、经过特殊优化的通信协议，其信息传输可以基于 MPI 网络、PROFIBUS 网络或者以太网。

S7 通信支持两种方式：基于客户端（Client）/服务器（Server）的**单边通信**和基于伙伴（Partner）/伙伴（Partner）的**双边通信**。其中，客户端（Client）/服务器（Server）模式是最常用的通信方式，也称作 S7 单边通信。在该模式中，只需要在客户端一侧进行配置和编程；服务器一侧只需要准备好需要被访问的数据，不需要任何编程（服务器的“服务”功能是硬件提供的，不需要用户软件的任何设置）。客户端通过 S7 通信协议，对服务器的数据进行读取或写入的操作。当两台 S7-PLC 进行 S7 通信时，可以把一台设置为客户端，另一台设置为服务器。S7 以太网通信协议主要用于将 PLC 连接到 PC 站（PG/PC-PLC 通信）。大多数情况下，西门子通信遵循传统的主从模式（master-slave）或者 CS 模式（client-Server）。其中 PC（master/client）将 S7 请求发送到现场设备（slave/Server）。

9.5.1. Mapper 支持的功能特性

S7 Mapper 对 S7 协议的支持情况如下所述：

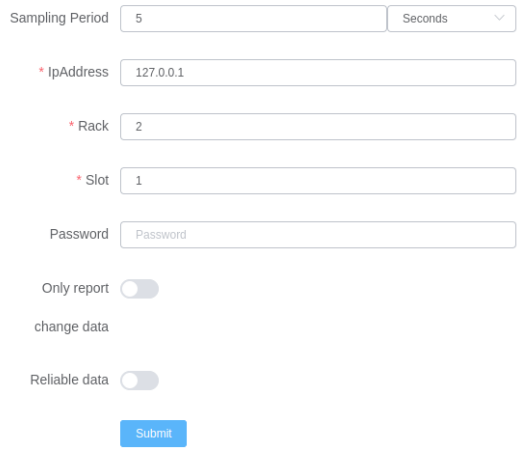
- **Mapper 实现了 S7 单边通信的客户端（client)功能，其传输层只支持 TCP 传输；**
- **支持读取/写入数据到 S7 PLC 的数据存储区域有：DB 区，Merker 区，输入点区域，输出点区域，Counter，Timer 等；**
- **每个区域都可以读出/写入的数据类型有 byte，char，int16，uint16，int32, uint32, float32 等数据类型；Counter 是一个 int 类型的数据，Timer 是时间类型的数据。**
- **支持 PLC CPU 状态的读取，开启/停止 PLC 等动作；**
- **支持 PLC 密码认证；**
- **不支持 PLC 的 Multiple read/write；**

9.5.2. 西门子 S7 子设备操作描述

下面我们将描述西门子 S7 子设备创建 & 属性绑定。

1. 创建子设备

创建子设备时，**Protocol Type 要选择 s7，**就会创建一个 s7 的子设备。创建完子设备后，需要绑定设备，其需要绑定的信息如下图所示：



IPAddress： PLC 的 IP 地址，其格式为 ip:port，一版不需要填写端口号，默认为 102。
Rack：机架，用于安装和组织 S7 PLC 的插槽的支架，一般为 0，具体根据实际情况填写。
Slot：S7 PLC 的插槽编号，一般地 S7-300/400 系列 PLC 为 2，S7-1200/1500 为 1。
Password：用于密码验证，没有可不填写。

2. 属性绑定

S7 属性绑定需要 4 个参数，即 PLC 的数据存储区域，DB 序号（如果是 DB 区域的话），起始地址，数据类型宽度等，其信息如下图所示：

Area：数据存储区域，有 DB，Merker，Input，Output，Timer，Counter 等，还有 PLCStatue（**不属于数据区域**）用于获取 PLC 的状态，和控制 PLC 的启动/停止。
DB Number： DB 序号，根据 Server 端是实际情况填写，其他数据区域不需要该参数。
Start：数据在这个区域的偏移地址，以 byte 为单位。
WordLen：读取/写入数据类型宽度，例如 s7wlbyte 的数据类型是 uint8，s7wlChar 类型是 int8，s7wlword 是 uint16，s7wlint 是 int16，s7wldword 是 uint32，s7wldint 是 int32，s7wlreal 是 float32， s7wlcounter 是 int 类型的计数器，s7wltimer 是时间类型等。
注意：

- PLC 的状态为 running/stopped 两种状态，如果为其他状态，表示和 PLC 没有建立正常连接；
- 创建 PLCStatus 属性，对该属性写 0 为 stop，其他值为 Start；

9.6. executor 协议

executor 协议主要用来调用远端（EIM Client 端）的 exe 程序、脚本、命令等，其适应于 Windwos、Linux 等系统。executor 设备除了采集周期外，不需要设置其他参数。executor 的设备属性配置如下图所示：

设备的扩展属性包含 exe 路径和参数，用以绑定具体的执行命令。exe 路径中仅包含 exe 的绝对路径，或相对于 Client 安装路径的相对路径，不能包含无意义的空格。其参数中可以包含多个用空格分开的参数，其类似本地执行相关命令。因为 exe 本身调用效率较低，所以其只能适合对实时性较低，采集周期为秒级以上的使用场景。

9.7. RESTful 协议

在 EIM Client 的周围，有一些设备只为用户提供 RESTful API 接口,为了获取这类设备的相关信息，DeviceOn/EIM 实现了 RESTful 协议，用以采集 RESTful 设备的相关数据。理论上，每一个提供 RESTful API 的硬件设备都可以抽象为 DeviceOn/EIM 的 RESTful 设备,RESTful 设备只需要提供 RESTful 的 baseurl。其如下图：

Host 为 RESTful API 的 baseurl 或者 host 的域名。RESTful 的属性配置如下图所示：

当前该 Mapper 仅支持 HTTP GET 的 RESTful API。RESTful API 包含 url 和参数，URL 为 baseURL+RESTful 路径中的内容，RESTful 参数可根据需要添加，其以 key 和 value 的方式，最终加入 HTTP Header 中。如果 RESTful API 回应的是一个 JSON 结构，且用户只希望读取该 JSON 结构中某一个位域值，就需要使用 ResponseID，其实质是一个过滤 key，其语法如下：

```
1. []：数组
2. 对象名：json 结构中的位域名
3. . ：该对象名表示的对像中的子位域，类似于指针
4. 如 json {
5. "a": "hello",
6. "b": {
7. "c": 12,
```

```
8.  "d": [0, 1, 2, 3],
9.  },
10. "e": [{"f":"hello"}],
11. };
```

如果要读取 a 位域的值，该 responseID 位 a；如果读取 d 位域的第一个值，该 responseID 位 b.d[0]。如果读取 f 位域的值，该 responseID 为 e[0].f。如果 JSON 是一个数组，便以[]开头，例如[0].a 等。

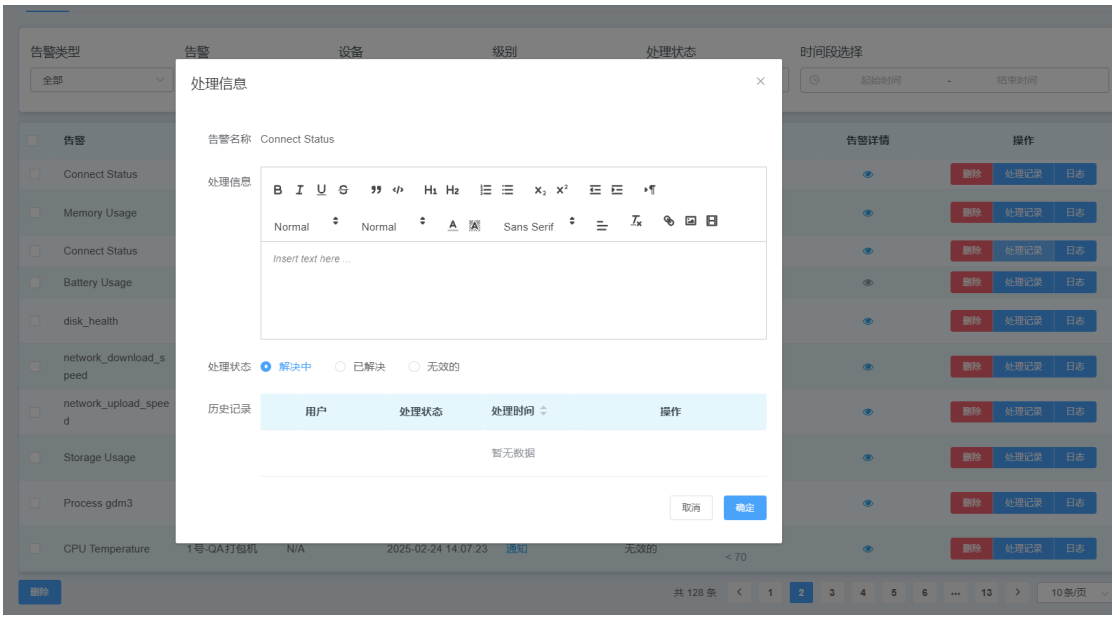
10. 告警中心

10.1. 告警管理

10.1.1. 告警列表

告警消息根据告警名称分类显示，显示每种告警当前的状态、描述等。
告警列表支持对告警的删除、批量删除、条件查询、日志查询、以及处理状态的修改和追踪。

当告警第一次发生时状态为未处理，巡检人员可以确认告警并处理，完成后点击**“处理记录”**，填写处理过程并修改状态。如下图：

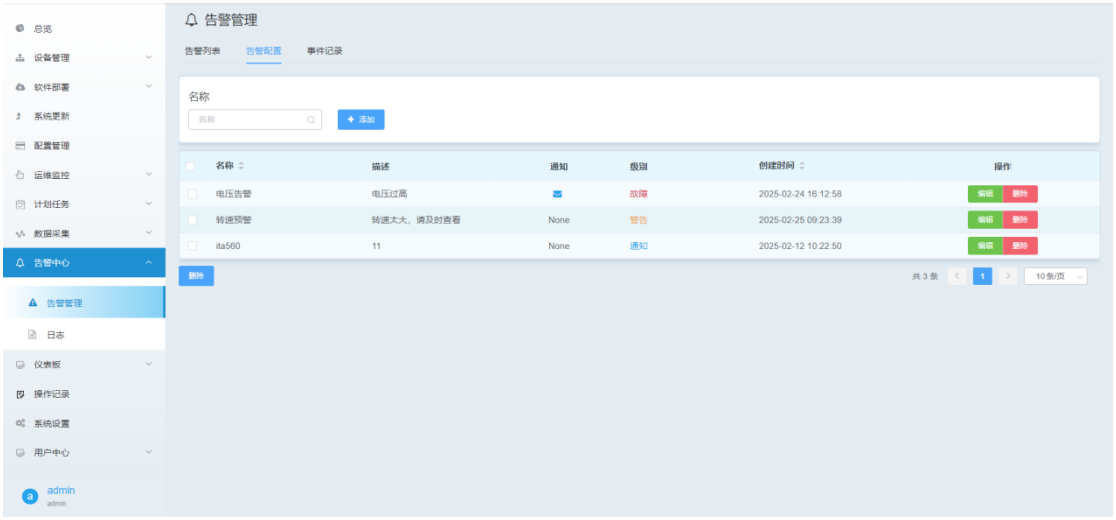


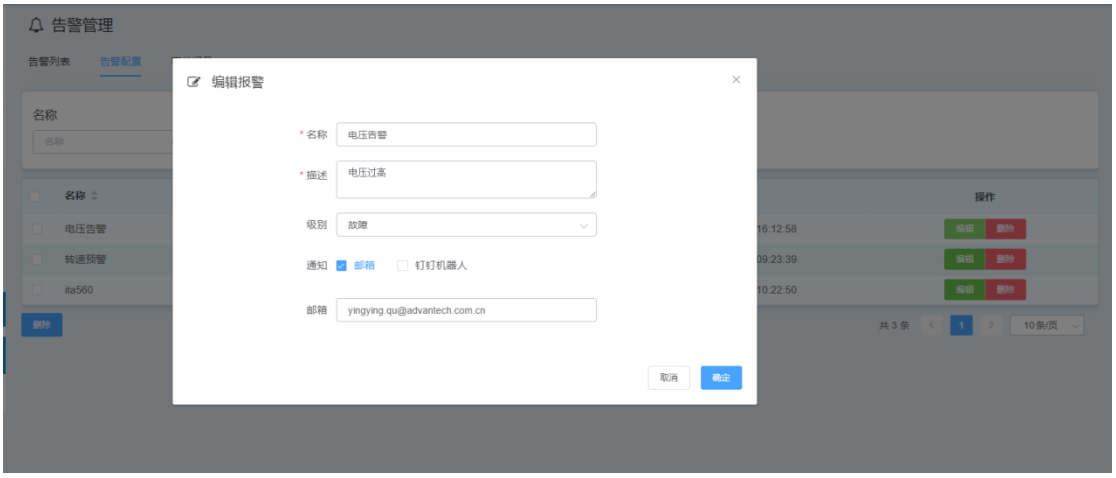
可以对某告警进行人为处理并在处理记录中显示，或者点击日志查看该报警消息的告警记

录。

10.1.2. 告警配置

用于子设备数据采集的告警消息配置，通过添加告警消息的名称、描述和等级，然后在子设备事件配置的时候关联，当子设备属性触发事件后，则会主动上报事件关联的告警消息。使用者还可以选择是否邮件通知或者钉钉通知

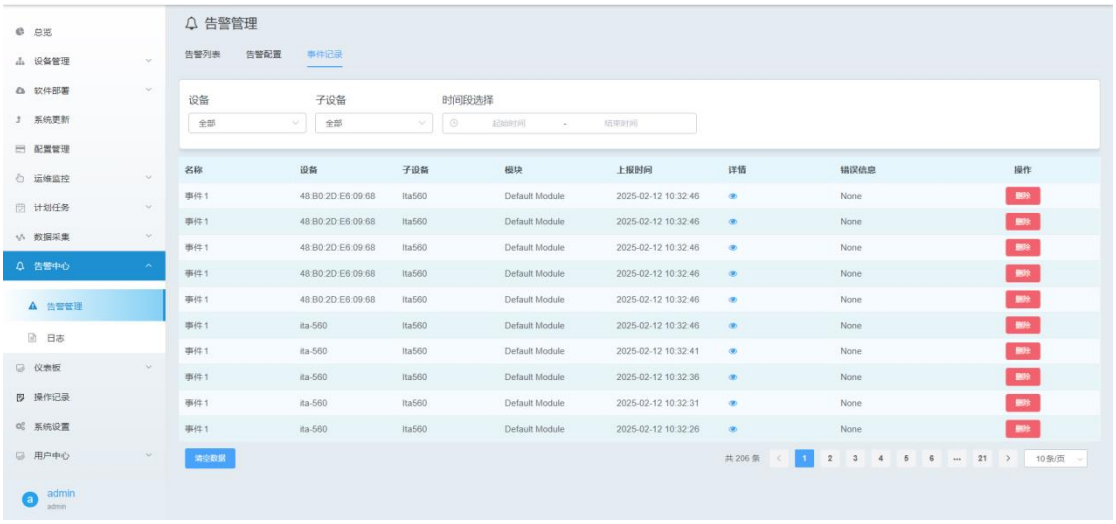




- 名称：事件告警名称
- 描述：事件告警的描述
- 级别：告警的级别，分为通知，警告，故障
- 通知：通知用户的方式，钉钉机器人和邮件。
  勾选钉钉机器人选项，可以选择钉钉机器人，如无机器人，点击添加按钮，在系统设置页面添加钉钉机器人。如勾选邮箱输入需要通知的邮箱地址。

10.1.3. 事件记录

记录当前 Server 上所有采集数据的子设备上报的事件。



10.2. 日志

从告警中心->日志，进入查看所有告警日志

也可以从事件告警/监控告警进入，查看的是某条告警下的历史告警记录。

11. DeviceOn/EIM Server 系统管理

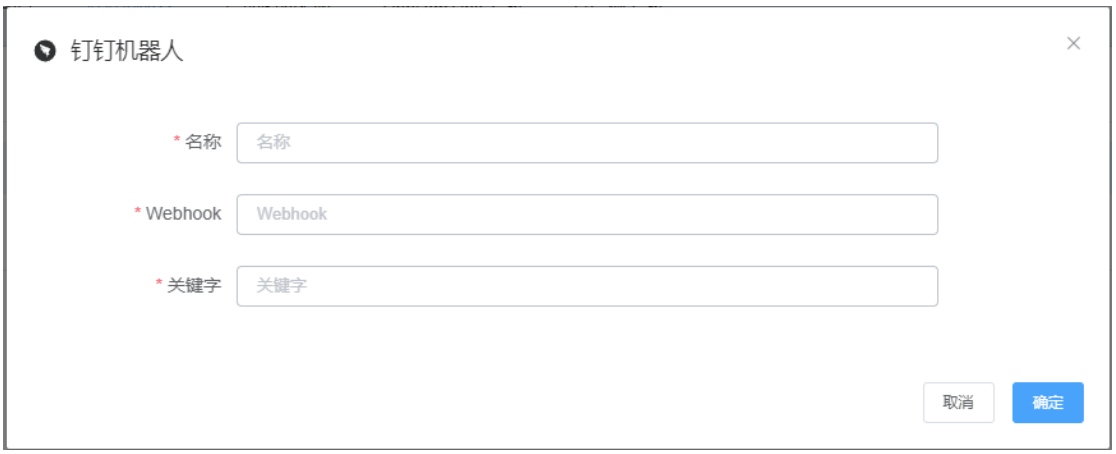
11.1. 系统设置

系统配置包括：菜单配置、influxDB 配置、邮箱服务器配置、钉钉机器人添加、产品图标更改、DeviceOn/EIM Apk 下载、DeviceOn/EIM 客户端下载。

- influxDB 配置：
  采集的数据是否存储到 influxDB 服务器中。
- 邮箱服务器配置：
  配置发送邮件的邮箱服务器，目前只支持 SNMP 协议。发送邮件主要用于注册用户、告警。
- 钉钉机器人的添加：
  通过 webhook 和关键字添加钉钉机器人，钉钉机器人主要用于报警提醒。
- 产品图标更改：
  可以修改产品名称和产品 logo
- 服务器配置：
  修改服务器 IP 地址。方便局域网 IP 映射外网时，访问地址的变化。

部分如下图所示：

注意：添加钉钉机器人时输入的关键字必须和钉钉机器人的关键字一致

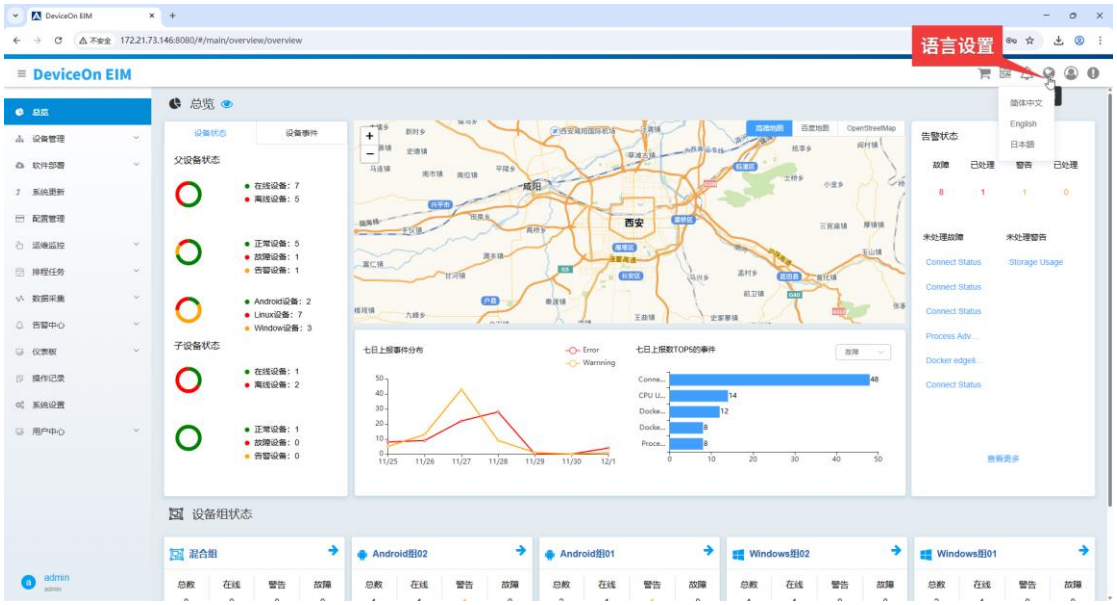


添加钉钉机器人的步骤：

1. 添加钉钉群，点击智能群助手
2. 点击添加机器人
3. 输入机器人名称、选择自定义关键字。

11.2. 语言设置

语言，目前支持中文、英文和日文，更多语言正在开发中。



12. 用户中心

用于管理系统中的用户和角色，确保不同用户拥有与其职责相匹配的访问权限。通过用户、角色和权限的精细管理，确保系统安全和高效运行。管理员可根据需求灵活配置，满足不同场景的权限管理要求。

12.1. 用户管理

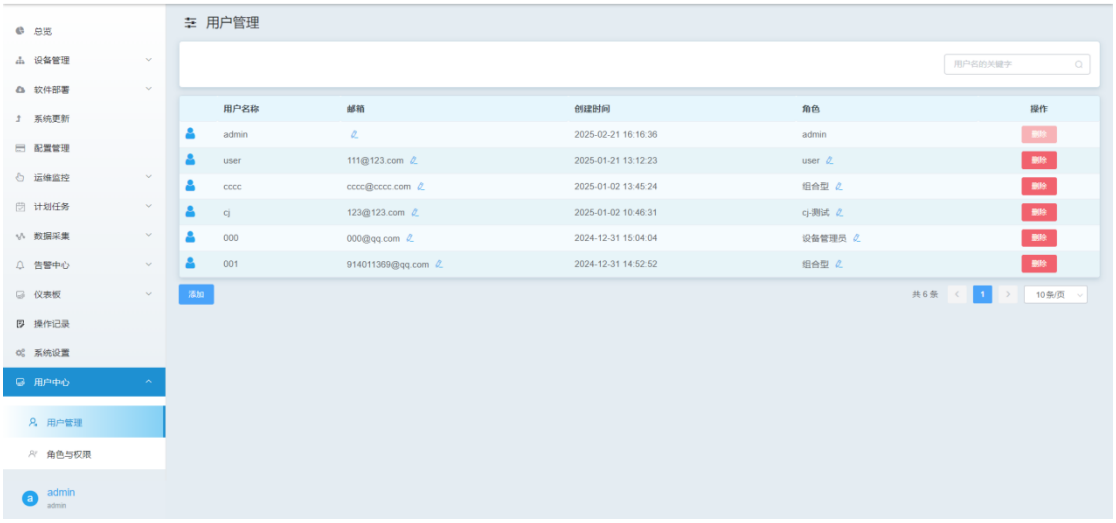
用户信息管理：支持添加、编辑、删除用户，管理基本信息如姓名、联系方式等。

密码管理：提供密码重置功能，确保账户安全。

权限管理：

（1）创建用户时，通过分配角色确保每个用户拥有匹配的权限

（2）动态调整：用户角色可随时调整，权限实时更新。



通过“添加”按钮，添加用户，如下图所示：

12.2. 角色与权限

角色创建与分配：支持创建不同角色，并为用户分配相应角色。

权限配置：每个角色可配置特定权限，确保用户只能访问与其职责相关的功能。

通过**“添加”**按钮创建角色，权限分类如下所示：

13. DeviceOn/EIM 软件仓库

13.1. 概述

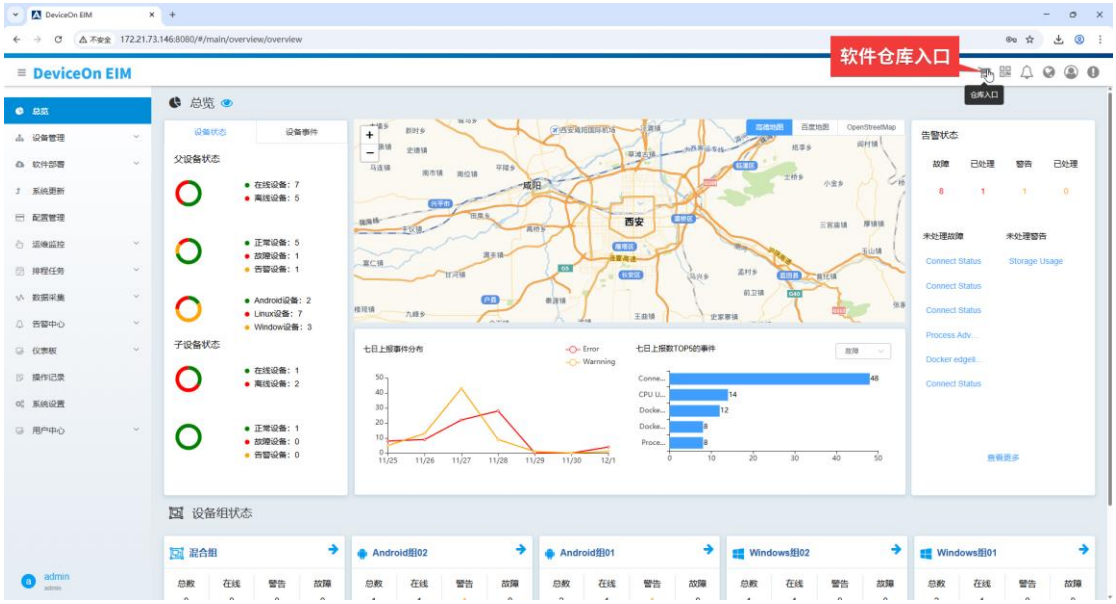
什么是私有软件仓库，为什么需要?
DeviceOn/EIM Repo 是一个 DeviceOn/EIM 的软件管理后台仓库，我们前面讲到，你可以使用 DeviceOn/EIM 来部署应用，更新系统，部署文件等等，在你进行这些操作之前，你都需要将相关的软件上传到软件仓库，才能在通过 DeviceOn/EIM Manager 把这些软件部署到远端的设备上。DeviceOn/EIM Repo 就是一个私有的软件仓库，和 DeviceOn/EIM Manager 是集成在一起的，共同组成 DeviceOn/EIM 整体方案，你可以把你需要部署的应用，文件，系统升级包等等，上传到 DeviceOn/EIM Repo 中，然后通过 DeviceOn/EIM Manager 进行单台或批量的部署。
一些操作系统厂商，如微软，或者 google 都有自己的 AppStore 应用商店，通过他们的商店，也可以进行应用的分发和部署，我们为什么还需要开发了一个 DeviceOn/EIM Repo 私有仓库呢? 因为，我们发现，这些现成的应用商店，并不能完全满足的工业客户的需求和使用场景，主要表现在以下这些方面:
(1)对于 Android 设备，很多工业 Android 设备都没有经过 Google GMS 认证和 AER 认证，系统中也没有集成 GMS 相关服务，无法使用 Google 的应用商店和企业 Android 相关功能。
(2)不管是 Google 的应用商店，还是微软的商店，都无法既支持 Android 应用，又支持 Windows 应用，更不要说 Linux 应用，Docker 容器应用等，而工业现场，通常有多种不同系统的设备并存，并希望可以统一管理。
(3)工业应用一般都是垂直行业的应用，很多工业用户并不想将这些应用上公有的应用商店。
(4)在许多工业现场，设备是无法访问互联网的，只是在一个内部网络中，无法使用外部各种应用商店。
(5)在工业应用场景中，一般是需要集中推送新版本到设备端，并且需要静默完成应用的安装升级，而不是让设备端使用者自行升级。
(6)一般的应用商店，只支持应用程序安装更新，并不支持系统更新部署，工业设备需要支持

系统更新的方案。

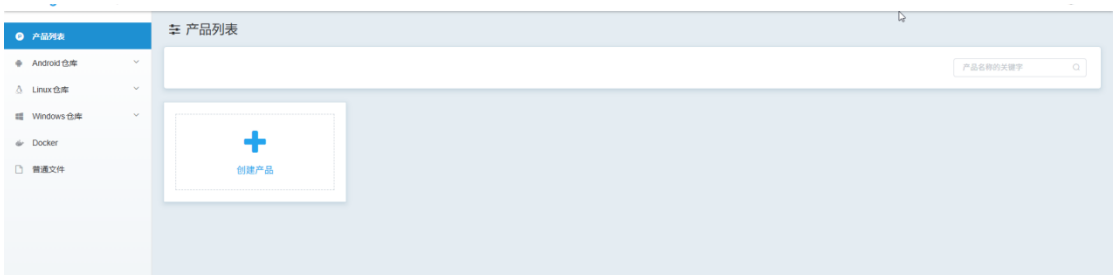
(7)在很多工业应用场景中，需要支持批量更新部署操作。

基于以上一些原因，目前系统厂商提供的应用商店，都无法很好的满足工业客户的需求。所以我们开发了 DeviceOn/EIM Repo 这个软件仓库方案，可以支持各种平台，各种类型的软件包的管理。DeviceOn/EIM Repo 也可以单独部署，比如你已经有其他的设备管理方案了，只是想用 DeviceOn/EIM Repo 来管理软件包，这也是可以的。

可以从主界面点击进入，如下图：



进入后，软件仓库的主界面如下图：



DeviceOn/EIM 软件仓库支持不同系统，不同类型的软件包管理，可以上传，下载，删除软件包，可以订阅感兴趣的软件包，实时收到更新通知，可以设定上级父仓库，可以设定软件包存放的地方，比如是存放本地，还是存放在云端，同时还支持多用户，多权限的管理，这些功能，我们后续会展开介绍。

13.2. 产品列表

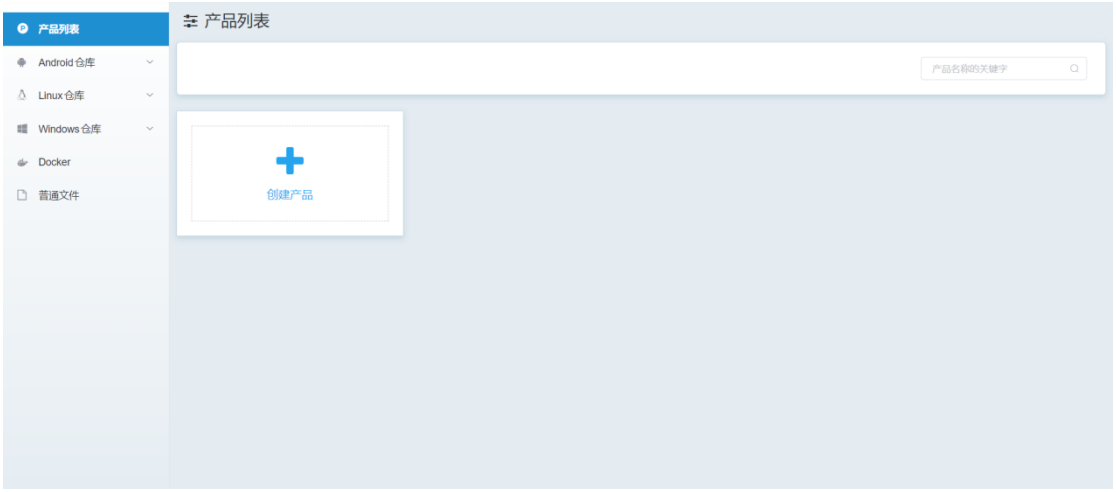
EIM 软件仓库除了可以存储不同种类的软件或者操作系统更新包之外，同样可以作为一个产品发布平台。产品列表的功能更是丰富了这一功能，可以将同一个产品下的多个软件或者多个不同种类的软件以及系统更新包以一个整体的形式进行展示和发布，同时附带产品说明功能，使产品客户可以更加直观便利的订阅和了解整个产品链。

以发布 Android TPC XX 产品为例。每次产品发布时要发布系统更新包以及系统上需要安装的相应的功能软件产品或者测试软件产品，同时要附带 user manual 或者修改记录。发布后

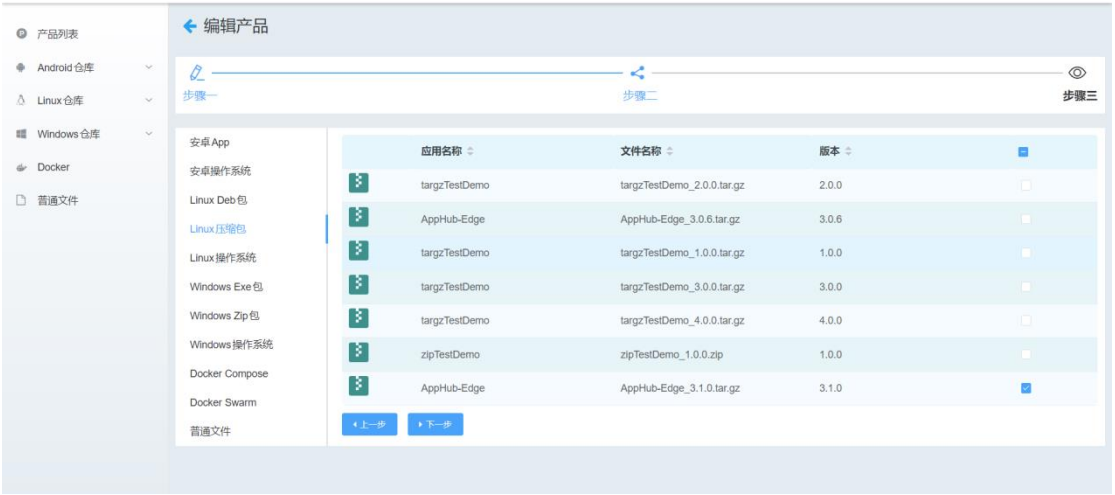
需要相关人员收到邮件通知。

创建产品步骤：

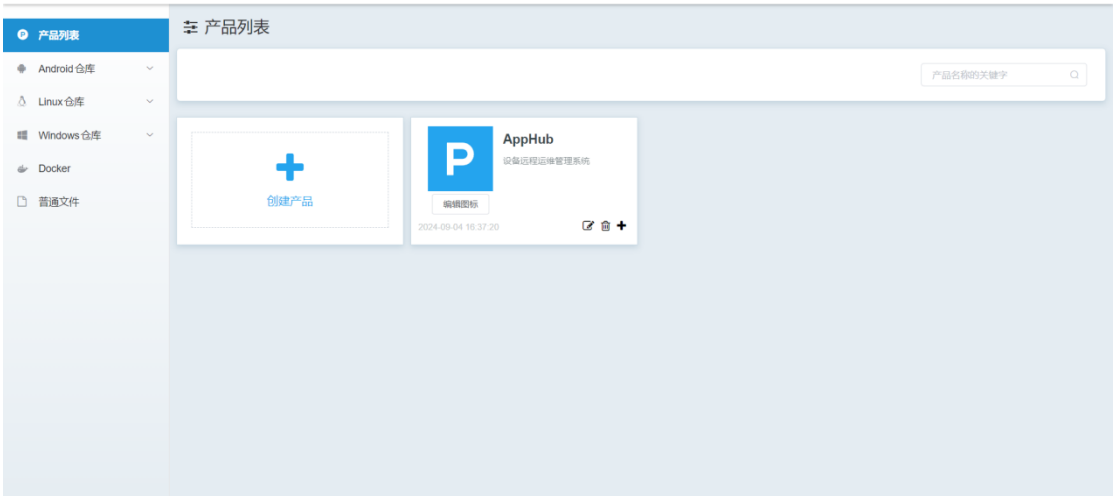
1. 点击【创建产品】开始创建。



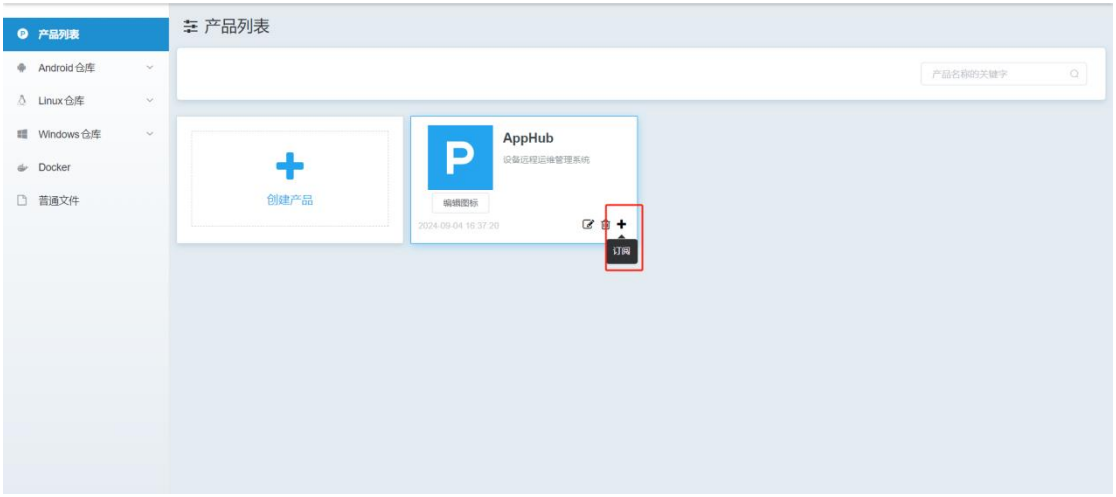
2. 编辑产品说明->勾选相应软件包->保存确定



3. 之后就可以在产品列表中看到该产品



如果用户需要订阅该产品则点击下图:



或者可以再次编辑或者删除。

当该产品内的相关系统更新包或者软件在软件仓库中有新上传时，订阅者可以自动收到邮件，而不需要分别单独订阅。

13.3. 仓库分类和软件包支持

DeviceOn/EIM 软件仓库可以让用户创建私有的 APP 应用商店方案，将开发的应用上传的这个软件仓库，DeviceOn/EIM 软件仓库就对上传的应用和软件进行管理了。

DeviceOn/EIM 软件仓库可以将仓库中的应用和设备上的应用进行版本比对，如果有新版本的应用，就会提示更新。用户可以通过 DeviceOn/EIM 将仓库中的应用远程推送的设备上，完成更新。

DeviceOn/EIM 软件仓库支持:

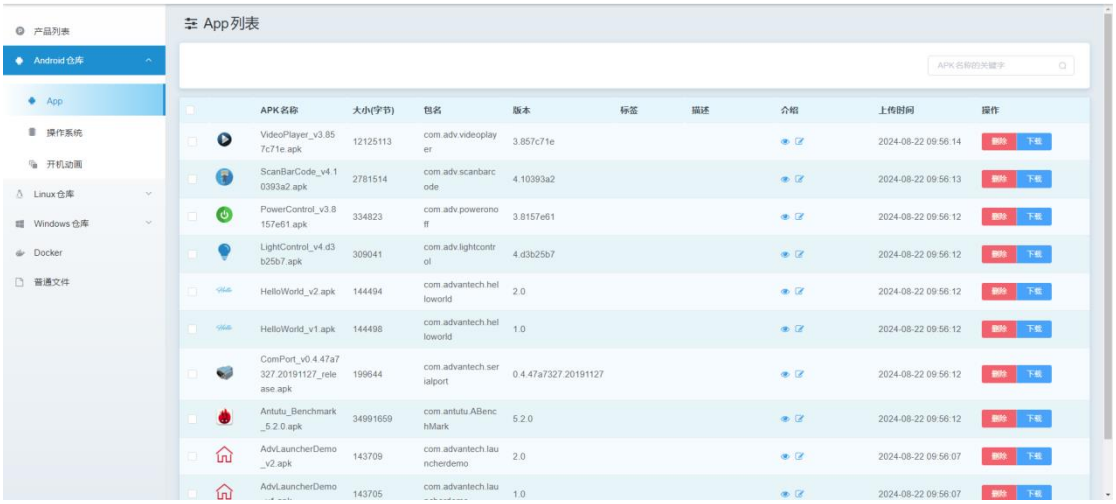
- 安卓软件仓库
- Linux 软件仓库
- Windows 软件仓库
- Docker 仓库
- 普通文件仓库

下面，我们分别对这些仓库进行介绍。

13.3.1. Android 仓库

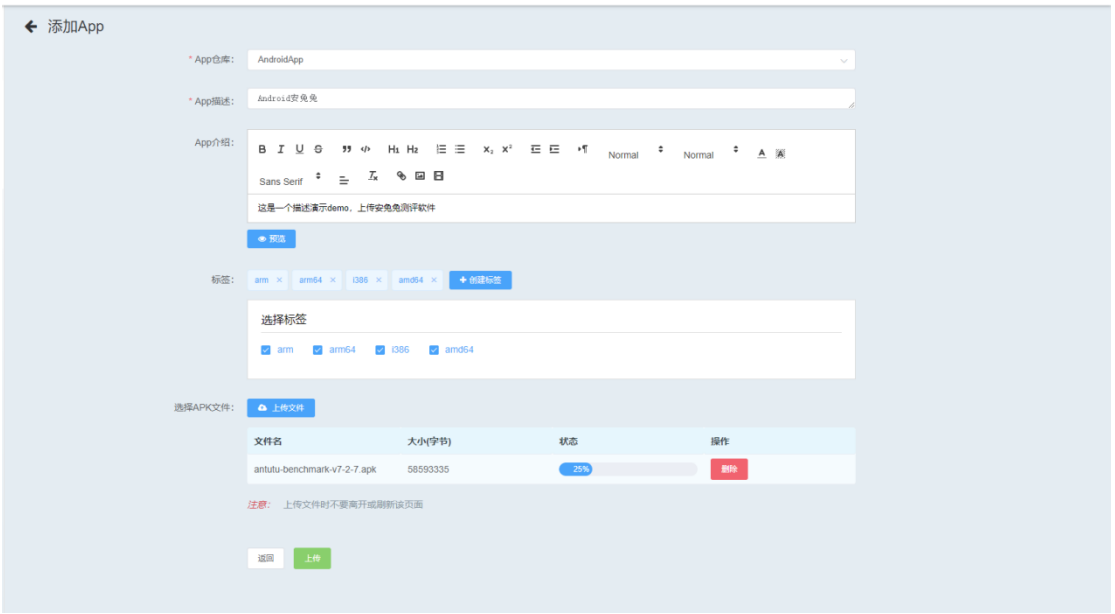
Android 应用仓库支持 Android App, Android 系统更新包，开机动画三种软件包的管理。

13.3.1.1. Android 应用仓库

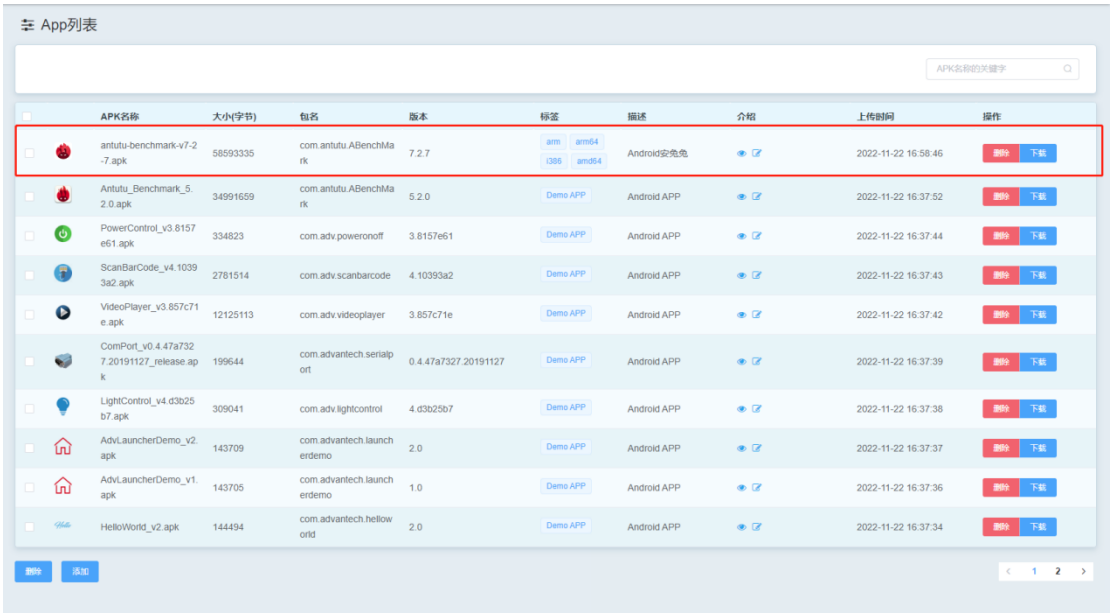


- 上传 apk

如果需要上传应用，则点击“添加”按钮，在接下来的页面中填写 APP 的描述(必填项)，并可以给 APP 添加标签（可选），如果对该 APP 有进一步的描述也可以在 App 介绍中添加介绍文本和图片（可选），最后选择 apk 文件，待 md5 计算完成后，点击上传完成上传动作，这样本地的应用就上传到仓库中了。上传时，Repo 会主动从应用中获取包的版本信息，包名等信息。如下图：



上传完成提示成功后，在 app 列表就可以看到刚刚上传好的 apk，如下图。



- 删除

如果不需要某 app 时，也可以从仓库中进行删除。可以选择单个 app 删除，也可以勾选多个批量删除。

- 下载

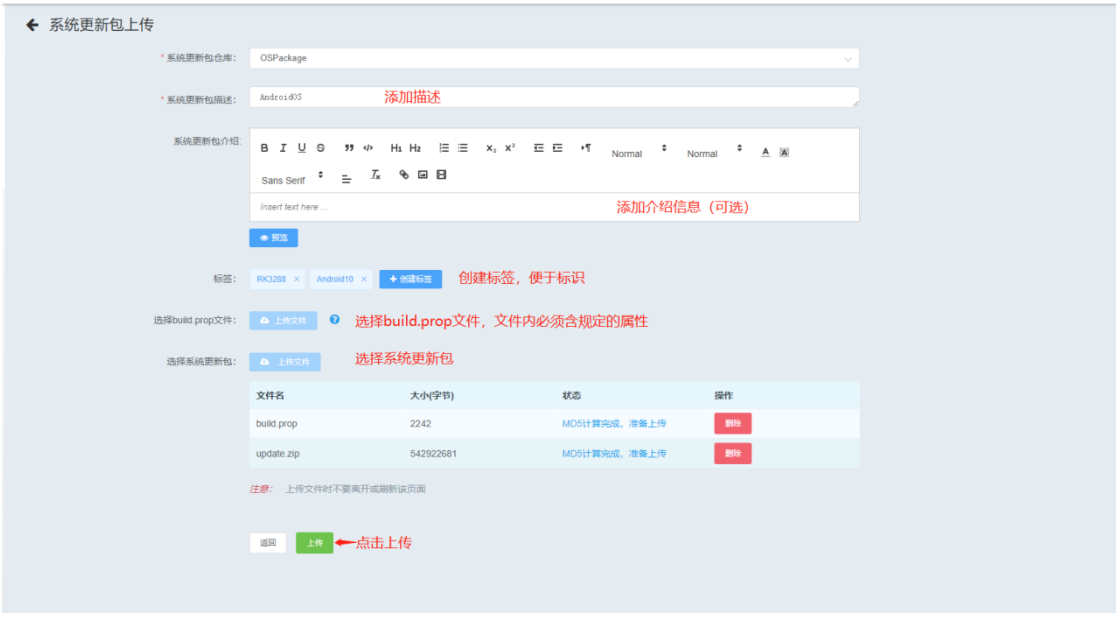
如果需要将仓库中的 app 下载到本地端，也可以点击“下载”。

13.3.1.2. Android 系统仓库

系统更新包一般需要系统开发者来提供，更新包比较大，通常在几百兆，上传时间也可能比较长，不过 DeviceOn/EIM 支持上传的断点续传，提升了上传大文件的可靠性。
直接上传更新包到系统更新仓库，操作方法同 app 类似。

- 上传

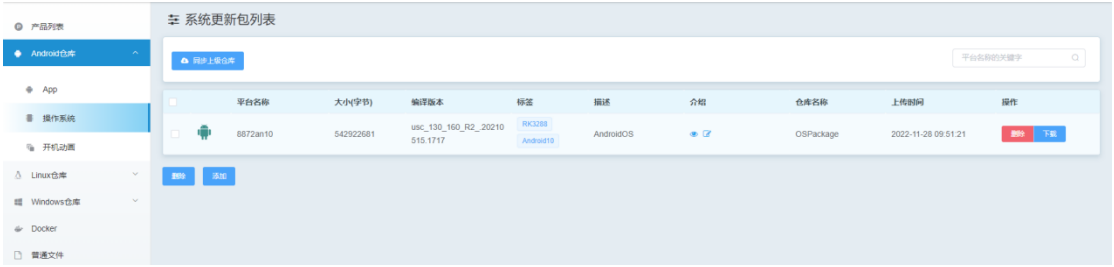
输入相应信息，并需要上传两个文件，build.prop 是对更新包的描述，update.zip 是真正的更新包。



其中，build.prop 文件是编译 BSP 更新包时产生的文件，包含了更新包的相关信息。如版本，支持的设备等。这个文件通常在 BSP 的下面这个目录中生成：/target/product/board_name/system/build.prop。

我们根据“ro.product.board”属性判断 platform，根据“ro.build.version.incremental”判断更新包的版本。

选择文件后，点击上传，即可完成上传。上传完成后在列表中即可看到，如下图：



因为一般更新包比较大，虽然有断点续传，但仍有可能会出现上传会失败，这种情况出现时，就需要重传。

- 删除

如果不需要某系统更新包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

- 下载

如果需要将仓库中的系统更新包下载到本地端，也可以点击“下载”。

13.3.1.3. 开机动画仓库

通过 DeviceOn/EIM 部署开机动画包后，可以改变 Android 设备的开机启动动画。

首先，需要制作一个新的开机动画包，Android 的开机动画是有标准的，这里并不会介绍如何制作开机动画，开机动画一般也是需要开发者来提供的，当你获得这个开机动画包之后，只要把它上传到开机动画仓库，就可以通过 DeviceOn/EIM 来进行部署了。

开机动画包是一个 zip 压缩包，选择添加上传即可：

这个功能需要 Android 系统配合，不是所有 Android 设备都可以支持，如果你的设备没有支持，又希望有这个功能，就需要联系赋华产品部门来客制添加这个功能。

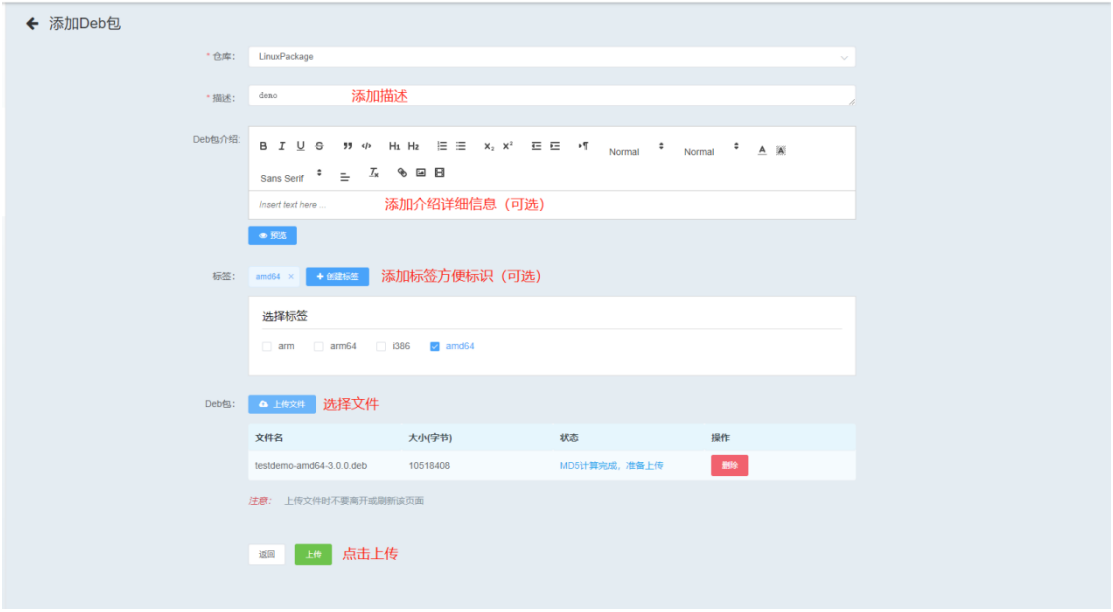
13.3.2. Linux 仓库

Linux 软件仓库支持 deb 包，zip 包和 tar.gz 包，系统更新包和 Docker/swarm。docker/swarm 将在 docker 部分统一介绍。

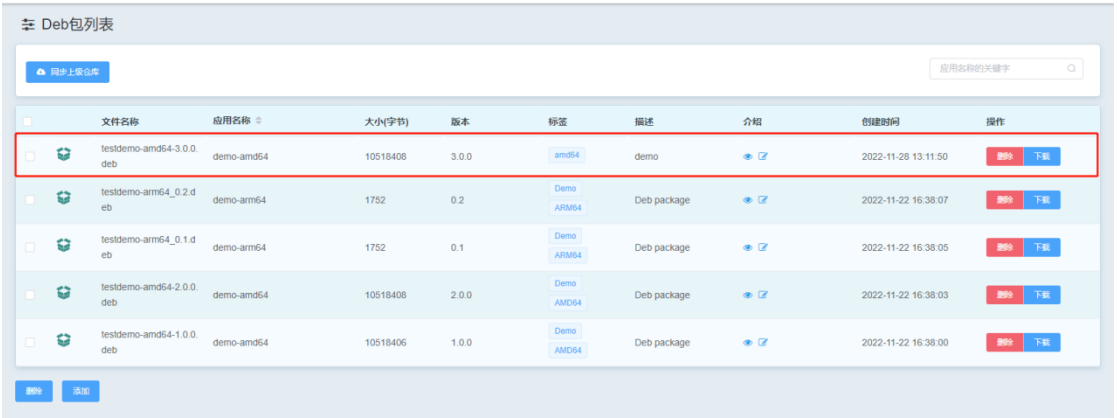
13.3.2.1. Deb 包

Deb 包是标准的 Linux 安装包，在 Ubuntu，debian，deepin 等很多常用的 Linux 发行版上，都支持 deb 包。

- 上传



添加完成后，在软【Linux 仓库-Deb 包】列表可以看到刚刚上传的软件包：



- 下载

如果需要将仓库中的软件包下载到本地端，也可以点击“下载”。

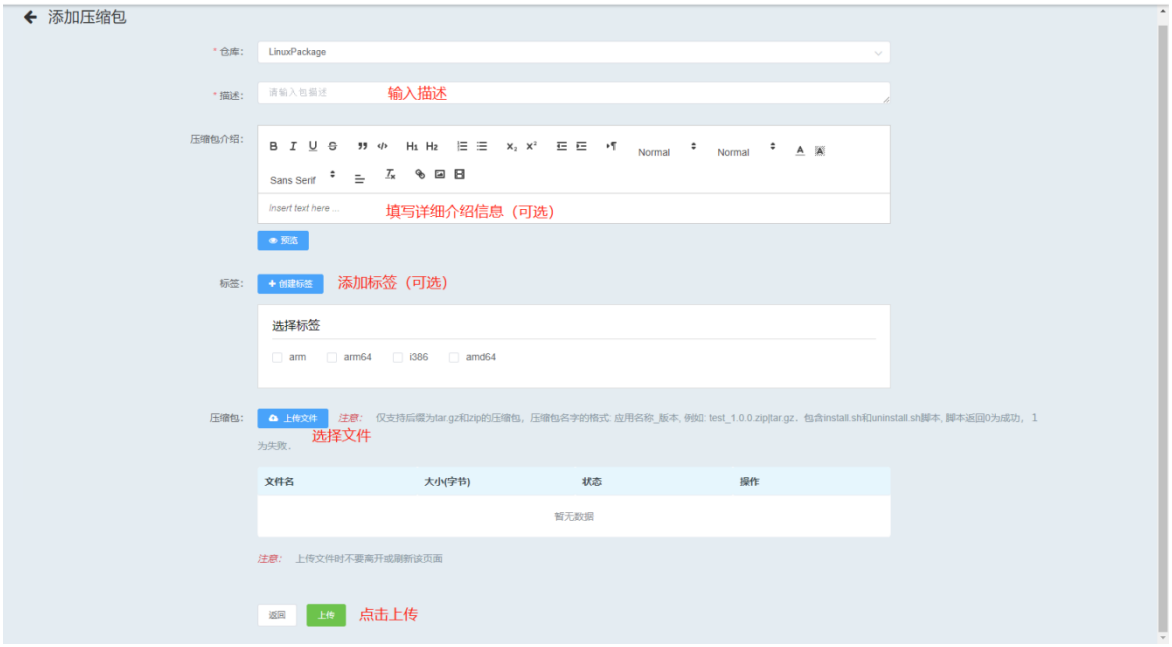
- 删除

如果不需要某软件包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.3.2.2. tar.gz/zip 压缩包

关于压缩包，需要符合一定规范，如何打包，参考章节 4.1.2.2/4.1.1.3。
同样的支持文件上传、下载、删除等动作。

- 上传



上传后在【Linux 仓库-压缩包】列表中可以看到软件包

- 下载

如果需要将仓库中的软件包下载到本地端，也可以点击“下载”。

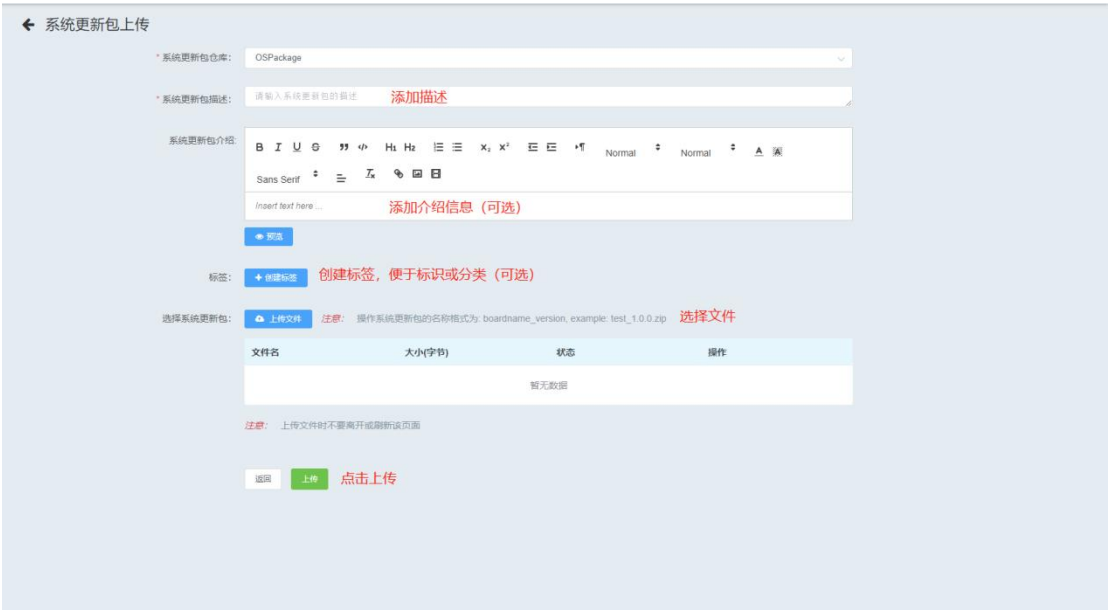
- 删除

如果不需要某软件包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.3.2.3. OS 更新包

操作系统部署，跟具体系统的关系会非常大，目前我们已经支持赋华 x86 和 ARM 等多个平台的系统更新。如果您有这方面的需求，请联系我们进行客制化。

- 上传



上传后在【Linux 仓库-操作系统】列表中可以看到系统更新包

- 下载

如果需要将仓库中的系统更新包下载到本地端，也可以点击“下载”。

- 删除

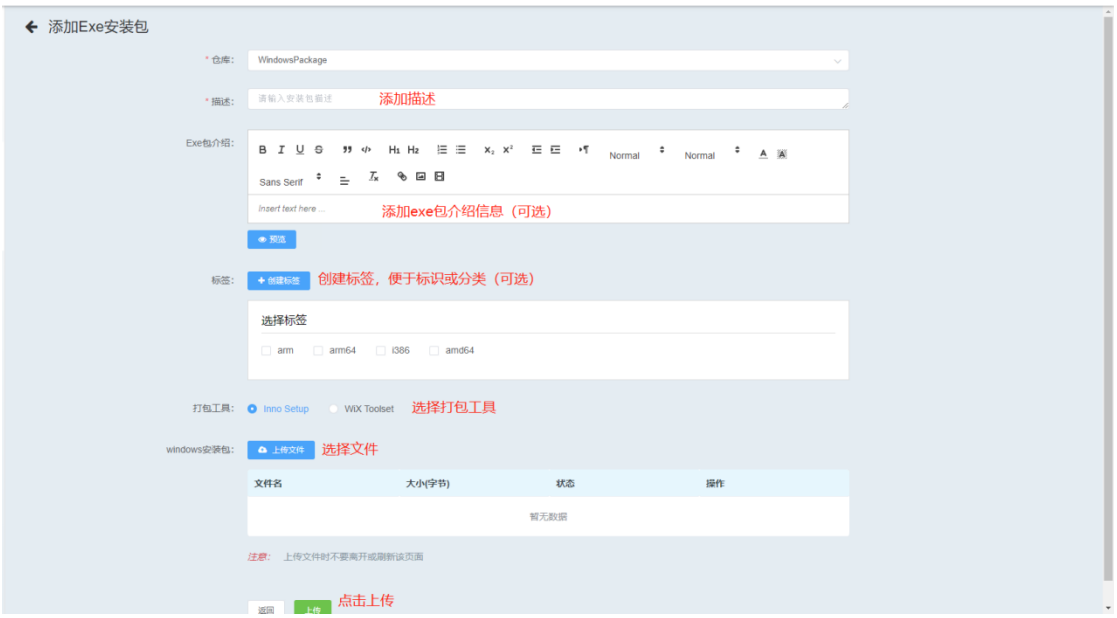
如果不需要某系统更新包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.3.3. Windows 仓库

13.3.3.1. Exe 包

Windows 的 exe 软件仓库可以上传 Inno setup 和 Wix Toolset 打包的安装包。Inno setup 和 Wix Toolset 是开源打包工具，具体打包方法这里不做介绍。

- 上传



上传后在【Windows 仓库-Exe 包】列表中可以看到软件包

- 下载

如果需要将仓库中的软件包下载到本地端，也可以点击“下载”。

- 删除

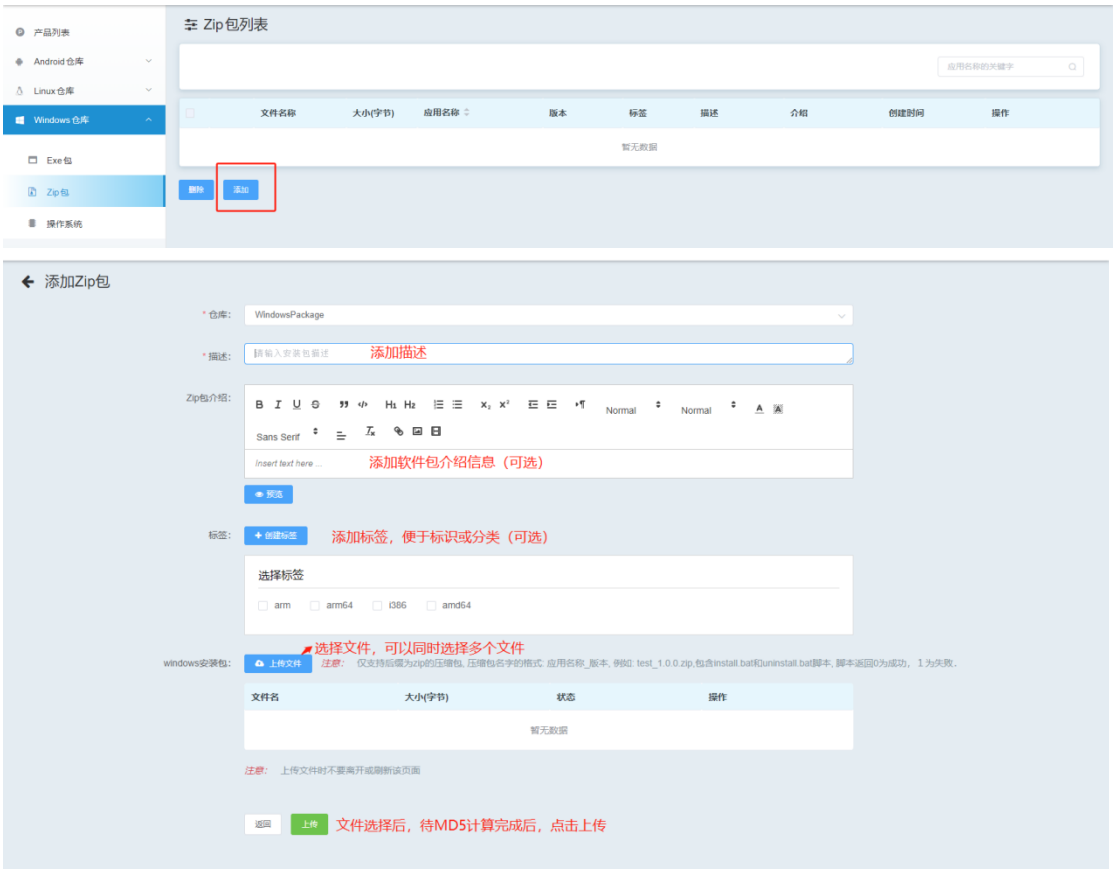
如果不需要某软件包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批

量删除。

13.3.3.2. Zip 包

zip 压缩包，需要符合一定规范，如何打包，参考章节 4.1.3.1。
同样的支持文件上传、下载、删除等动作。

- 上传



上传后在【Windows 仓库-Zip 包】列表中可以看到软件包

- 下载

如果需要将仓库中的软件包下载到本地端，也可以点击“下载”。

- 删除

如果不需要某软件包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.3.3.3. 操作系统

操作系统部署，跟具体系统的关系会非常大，如果您有这方面的需求，请联系我们进行客制化。
同样的支持文件上传、下载、删除等操作。

- 上传

上传完成后在【Windows 仓库-操作系统】列表中可以看到系统更新包

- 下载

如果需要将仓库中的系统更新包下载到本地端，也可以点击“下载”。

- 删除

如果不需要某系统更新包时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

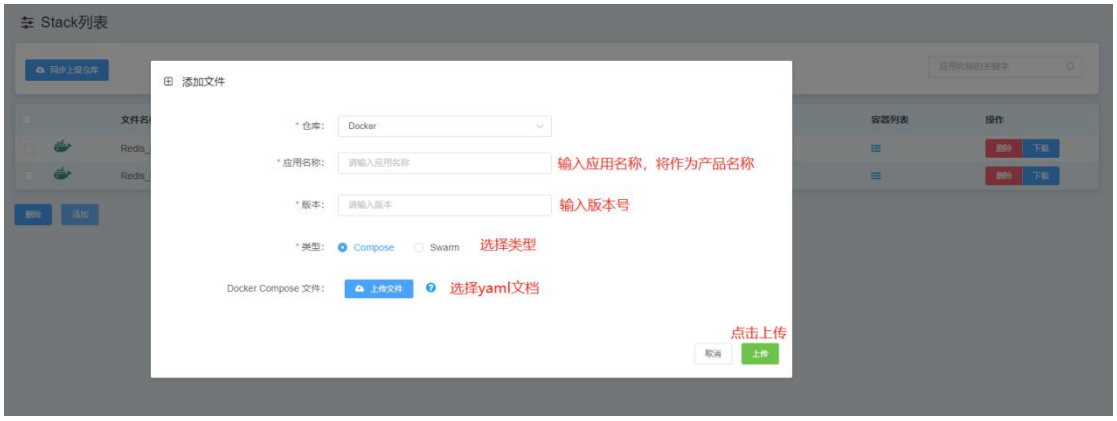
13.3.4. Docker 仓库

DeviceOn/EIM Docker 的部署支持 docker compose 和 docker swarm 两种类型，且都是通过 yaml 文件的方式进行部署，yaml 文件的格式为遵循官方标准。

我们只是管理 yaml 文件，真正的 image，仍需要上传到如 Docker hub 等外部 docker 仓库。

Docker 支持部署到 Windows 和 Linux 环境。

- 上传



上传完成后在【Docker 仓库】列表中可以看到系统更新包

- 下载

如果需要将仓库中的 yaml 档下载到本地端，也可以点击“下载”。

- 删除

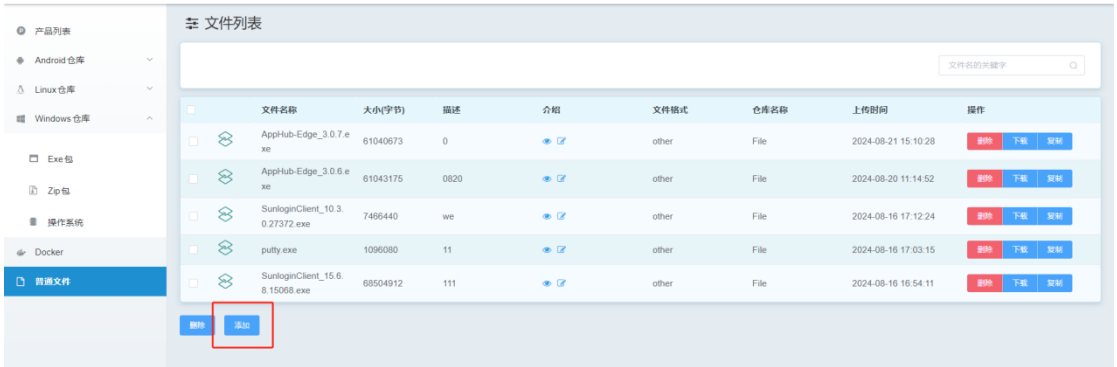
如果不需要某 yaml 档时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.3.5. 普通文件仓库

DeviceOn/EIM 也支持部署普通文件到远端设备，对于需要部署的文件，也要先将它上传到文件仓库。普通文件不做任何限制，即任何格式的文件都可以上传到仓库中，上传时，我们有初步的分类，即图片、视频、音频、压缩包、codesys 应用和其他。用户使用时可以进行文件分类的选择，也可以选择其他类型上传任何格式的文件。

需要特别说明的是，codesys 应用是专门针对 codesys manager 的功能做的分类，如果使用需要使用 DeviceOn/EIM 的 codesys manger 功能，请在上传 codesys 应用包时选择 codesys 应用。

- 上传



上传完成后在【普通文件】列表中可以看到系统更新包

- 下载

如果需要将仓库中的文件下载到本地端，也可以点击“下载”。

- 删除

如果不需要某文件时，也可以从仓库中进行删除。可以选择单个删除，也可以勾选多个批量删除。

13.4. 仓库存储配置

软件仓库因为要保存上传的软件，所以需要配置存储，本地版本或者 VM 版本，我们默认有配置了一个存储服务，直接使用了服务上的硬盘空间。此外，为满足不同客户的需求，我们同时也支持阿里云的 OSS，Azure 的 storage 或者 AWS S3 作为软件仓库的存储。DeviceOn/EIM 提供了配置页面，可以将仓库存储配置成阿里云的 OSS，Azure 存储或者 AWS S3 存储。

另外，如果是部署在 WISE-PaaS 平台，必须要配置外部存储，因为 WISE-PaaS 采用 K8S 架构，在 K8S 架构下，必须要通过专门的存储服务来实现文件永久保存，无法直接使用服务的磁盘，所以必须配置外部存储。WISE-PaaS 平台本身也有提供 blob 的存储服务，DeviceOn/EIM 就可以支持。同时也可以配置其他云存储服务。

下面分别对这几种支持的外部存储做一个简单的介绍：

(1)默认内置服务器存储

DeviceOn/EIM 默认内置存储，直接使用服务器的硬盘作为存储，无需配置，当然如果你想使用阿里云或微软的存储，也可以配置成外部存储。

(2)阿里云 OSS 存储配置

阿里云提供的存储服务，如果客户需要用阿里云的存储服务，在 DeviceOn/EIM 中可以进行配置，但需要客户自己额外申请购买阿里云的 OSS 存储服务。

(3)微软 Azure Storage 存储配置

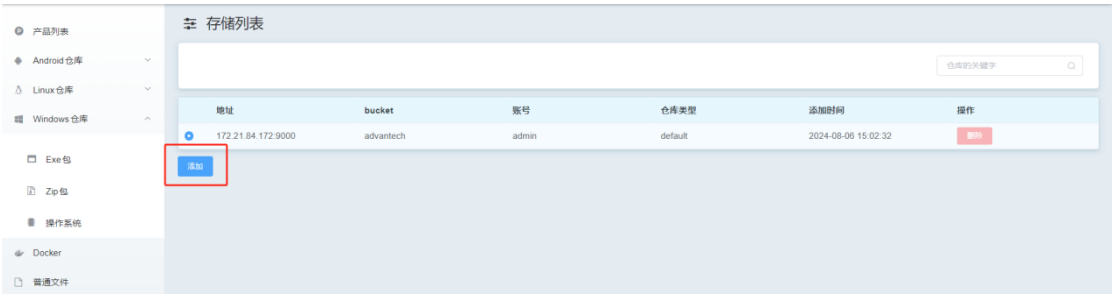
微软 Azure 提供的存储服务，如果客户需要用 Azure 的存储服务，在 DeviceOn/EIM 中可以进行配置，但需要客户自己额外申请购买 Azure Storage 存储服务。

(4)S3 存储配置

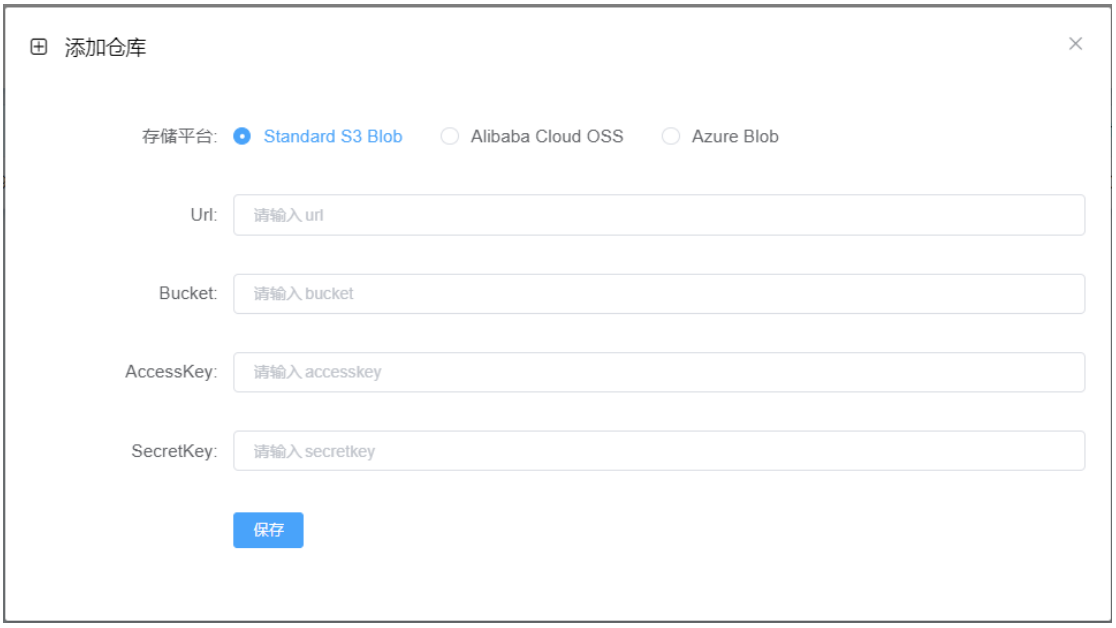
亚马逊提供的 AWS S3 的存储服务，还有很多第三方的存储服务厂商，提供兼容 S3 接口的存储服务，如果客户需要用 S3 兼容的各种存储服务，在 DeviceOn/EIM 中可以进行配置，

但需要客户自己额外申请购买 S3 存储服务。

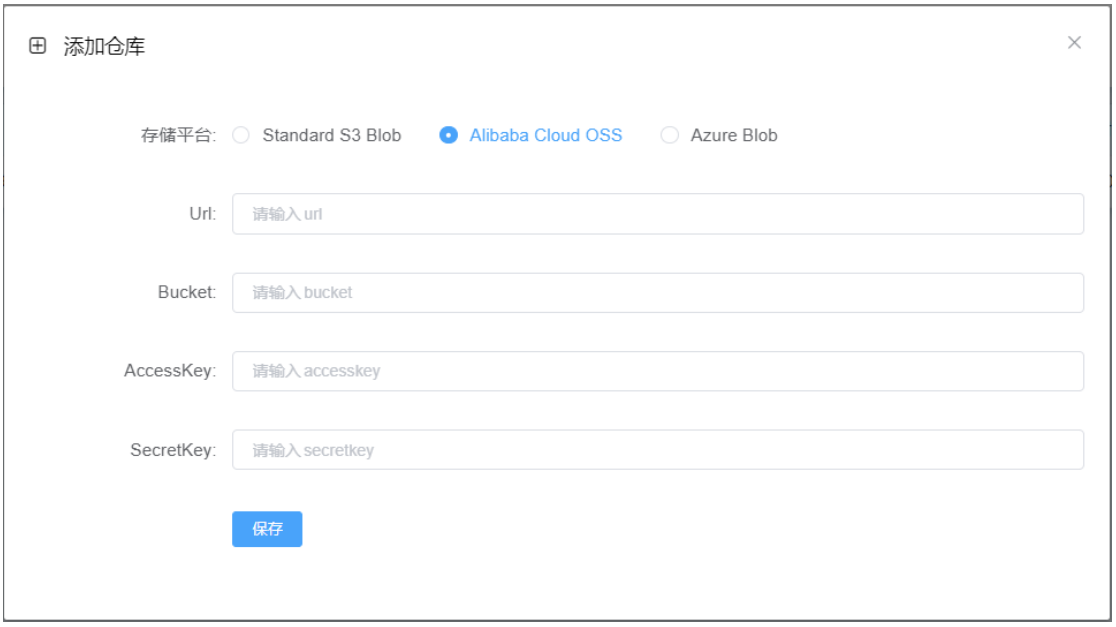
跳转到配置界面，安装如下方式添加外部存储：



如果使用标准 AWS S3 协议的存储，选择“Standard S3 Blob”，并输入相应信息后保存即可。



如果使用阿里云 OSS 存储，则选择“Alibaba Cloud OSS”，并输入相应信息后保存。

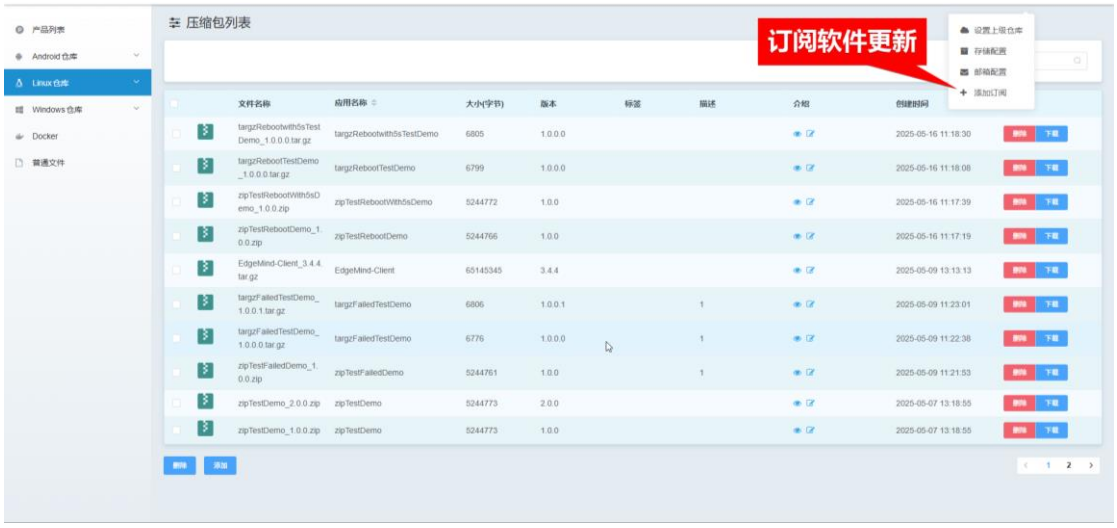


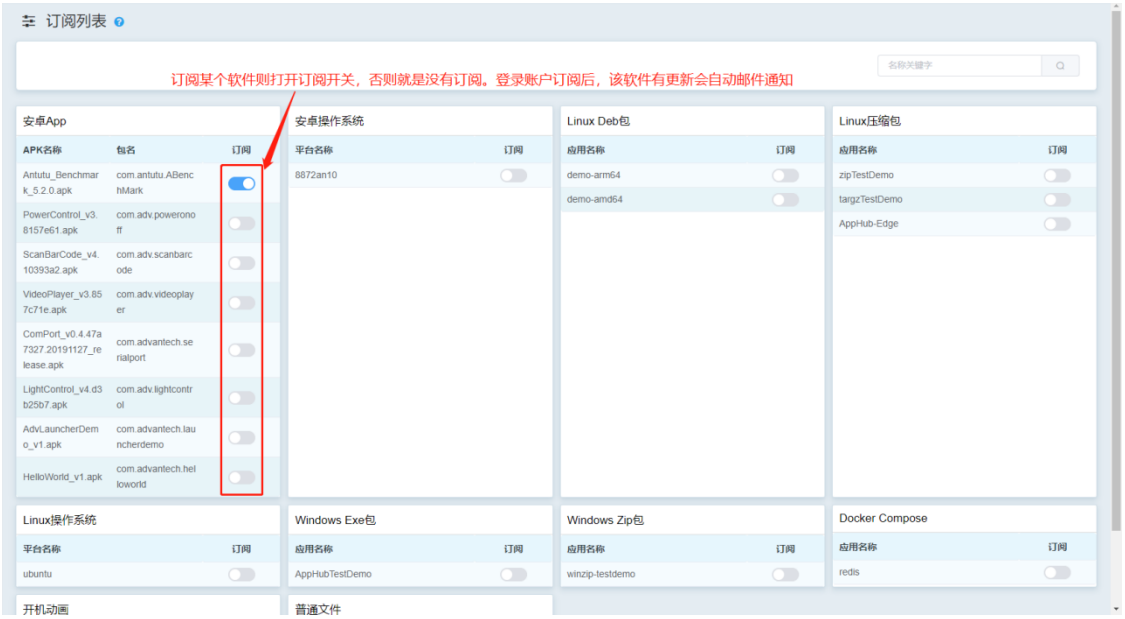
如果使用微软 Blob，则选择“Azure Blob”，并输入相应信息保存。

保存后，在存储列表中，点击选择相应的存储即可。

13.5. 订阅更新通知

DeviceOn/EIM 支持软件更新通知，用户登录后，可以订阅自己感兴趣的软件，订阅后，如果有该软件有新版本上传的 DeviceOn/EIM Repo，你就会收到有新版软件的通知，你可以根据需要在决定，是否要将你设备端的软件更新到新的版本。
订阅方法如下：





13.6. 上级仓库的级联和同步

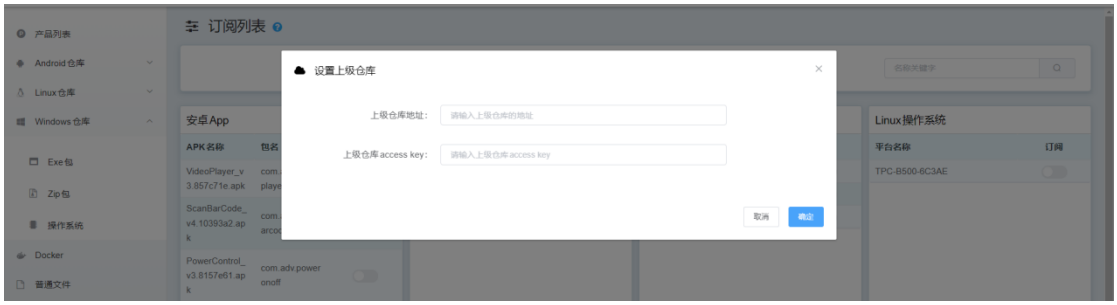
先来描述一个场景，比如有一个行业应用开发厂商，他开发了很多的行业应用软件，并且也在不断的更新和迭代，那他就有一个需求，如果快速的将新的版本通知他的客户，让他的客户知道，对于他的客户，当接收当新版本通知后，如何快速的下载这个新版的应用，如何快速的将新版本应用快速部署到自己的设备上去，也是一个问题。

DeviceOn/EIM 针对这个需求，提供了解决方案：

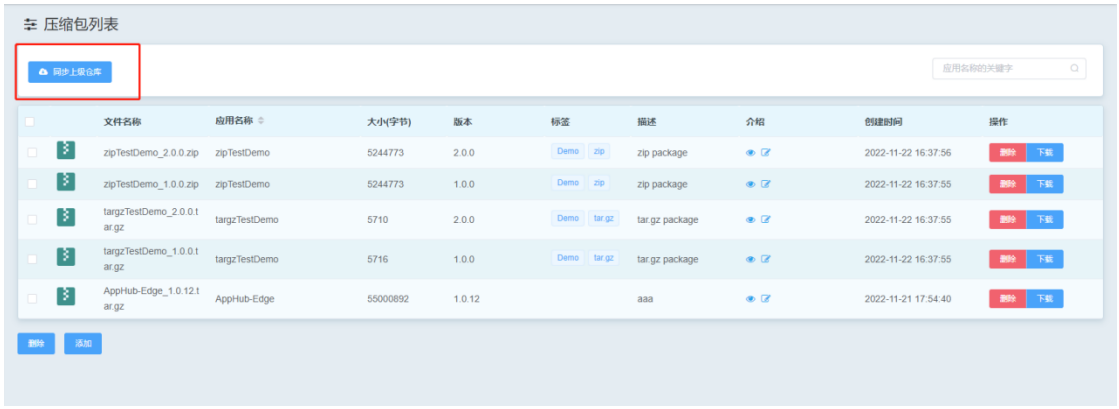
行业应用开发厂商，在云端搭建一个 DeviceOn/EIM 软件仓库，将相关行业应用上传到他的 DeviceOn/EIM 软件仓库中，他的客户如果对其中某些应用感兴趣，就会订阅这些应用，这样，行业应用厂商上传新版本应用时，相关的客户就会收到通知。这些客户使用 DeviceOn/EIM 来管理设备，在每个客户的案场，他们都有自己的 DeviceOn/EIM 服务器，并且通过 DeviceOn/EIM 服务器来管理设备，这时，每个客户可以在自己安装的 DeviceOn/EIM 服务器上，设定一个上级仓库，就是行业应用开发厂商的那个仓库，这样，如果行业应用厂商有发布新版本的话，客户可以在自己的 DeviceOn/EIM 中，将行业应用厂商发布的新版本直接同步到自己的 DeviceOn/EIM 仓库中，然后通过 DeviceOn/EIM 快速部署到自己管理的设备上，非常快捷方便。针对这种应用场景，我们已经有成功案例。

操作步骤：

进入上级仓库设置页面，设置上级仓库方法如下图，直接填入上级仓库的 URL 和 access key，确定即可：



配置上级仓库成功后，会在每种软件仓库列表左上方出现【同步上级仓库】按钮，否则不会有该按钮出现。如下图:



点击【同步上级仓库】进入该软件类型可同步的软件列表。这里面显示的软件，是本地仓库中没有的，或者是比本地仓库中新的版本，点击【同步】，就可以把上级仓库中的软件包，下载到本地仓库中。后续就可以根据需要，进行新版本的部署了。

13.7. 软件打包说明

13.7.1. 标准安装包

上述几种包格式中，Android Apk 包，window exe 安装包，Linux deb 包，以及 Docker 镜像等，都是标准的格式，其中 Windows 安装包，目前只支持 Inno Setup 和 WiX toolset 打包工具打成的安装包。关于标准格式的包打包，这里不做详细说明。如有疑问，可以联系我们。

13.7.2. 通用软件包

为了兼容更多的软件部署方式，DeviceOn/EIM 支持通用的基于压缩包的管理和部署，压缩包的格式就是标准的 zip 格式或者 tar.gz 格式，但压缩包内部的内容以及文件命名，需要遵循一定的规范，打出来的包，DeviceOn/EIM 才能够进行部署，接下来，重点说明一下压缩包的打包方式：
压缩包文件名定义如下：
包名_版本信息_自定义内容.zip
包名_版本信息_自定义内容.tar.gz
上面自定义内容是可选的，如果没有自定义内容，那压缩包文件名就变成:
包名_版本信息.zip
包名_版本信息.tar.gz

文件名中版本信息的定义，遵循 https://semver.org/定义的规范的。比如下面是一个版本比对关系的例子：
1.0.0-alpha<1.0.0-alpha.1<1.1.0-alpha.beta<1.0.0-beta<1.0.0-beta.2<1.0.0-beta.11<1.0.0-rc.1<1.0.0
比如 test_1.0.0-beta.11_arm.tar.gz 的版本，就比 test_1.0.0-beta.2_arm.tar.gz 新。

(1)Linux 压缩包可以支持 zip 包和 tar.gz 包
压缩包中必须包含 install.sh 和 uninstall.sh，其中 install.sh 是安装脚本，uninstall.sh 是卸载脚本，至于具体如何安装，卸载，完全是由开发者在对应的脚本中实现，DeviceOn/EIM 只负责下载该压缩包，解压，然后在安装时去执行 install.sh，卸载时，就执行 uninstall.sh。默认脚本返回值为 0 表示安装/卸载成功，1 则表示安装/卸载失败。
详细打包步骤，*参考* 4.1.2.2.

(2)Windows 压缩包
原理上跟 Linux 压缩包完成一样，Windows 只支持 zip 格式文件，安装脚本和卸载脚本变成是 install.bat 和 uninstall.bat 批处理文件。
Client 会根据安装脚本和卸载脚本的返回值，来判断安装或卸载是成功还是失败，脚本的返回值 0 表示成功，1 表示失败。
压缩包下载到设备后，DeviceOn/EIM 客户端会对压缩包进行解压，然后执行里面的 install 文件，执行完成后，会删除解压目前中的文件，只保留 uninstall 脚本。
详细打包方式参考 4.1.3.1。

我们有提供打安装包示例，可以从下面网址下载参考：

1)百度云盘下载 提取码：xian

2)Google driver 下载

进入 package_demo 目录，可以下载和查看安装包示例,关于打包，如有疑问，也请随时跟我们联系。

14. FAQ

1. 如果使用中有问题或疑问，如何联系咨询?

可以联系 DeviceOn/EIM 产品经理：Jiangwei@fuhuaxc.com.cn 也可以联系你购买产品的产品经理或赋华销售。

2. DeviceOn/EIM 管理方案是免费的吗？是否可以免费试用?

如果需要通过试用来了解 DeviceOn/EIM 的功能，建议先安装本地版本进行试用，该版本可以免费管理 3 台设备，只要不超过 3 台，就可以免费使用，如果超过 3 台，就需要购买软件许可进行激活。建议先免费使用，确认的确可以满足你的需求后，再来购买。

3. DeviceOn/EIM 服务器可以部署在云端吗?

是的，除了可以部署在本地的服务器上，DeviceOn/EIM 也可以部署到云端环境，支持部署在阿里云，微软 Azure 云等云平台。详细可以联系我们询问。

4. DeviceOn/EIM 服务器本地版本对服务器硬件有什么要求?

DeviceOn/EIM 服务器占用的资源很少，一般普通的配置的 64 位电脑都可以运行 DeviceOn/EIM，DeviceOn/EIM 服务器支持部署在 Windows 机器或者 Linux 机器。Windows 平台需要安装虚拟机环境。在开发和试用时，你可以尝试部署在你开发的台式机上，或者笔记本上，甚至是虚拟机中，都是没有问题的。

5. DeviceOn/EIM 可以管理哪些设备?

DeviceOn/EIM 可以支持所有的赋华的 Windows, Linux 和 Android 设备，并且支持 ARM 和 X86 平台。从技术上讲，DeviceOn/EIM 也可以支持管理非赋华的设备，如果你的场域内既有赋华设备，又有其他厂商的设备，但需要统一使用 DeviceOn/EIM 来进行管理，我们可以提供技术支持。另外，有些支持工业协议的设备（如 modbus, OPC-UA, S7, restful API, SNMP 等等），可以作为子设备，接入到 DeviceOn/EIM 中来，并且可以通过这些协议采集数据，监控数据，数据可视化和转发等。

6. 如何获得 DeviceOn/EIM 相关安装软件?

请联系赋华销售或者赋华 DeviceOn/EIM 产品的产品经理。可以从下面地址下载最新的本地版本服务器和客户端程序：下载地址(百度云盘)：链接：
https://pan.baidu.com/s/1kuqIMkCbecQVlyGYrXUtyg 提取码：xian

7. 当我安装 DeviceOn/EIM 的客户端应用到我的设备时，安装失败了，是为什么?

首先，赋华的很多设备已经默认预装的 DeviceOn/EIM 的客户端程序，不再需要额外安装，如果确认你购买的设备没有预装 DeviceOn/EIM 客户端程序，你也可以后安装。

8. 当使用手机进行扫码设备注册时，扫码失败，是什么原因?

要确保手机和设备在同一个局域网内，因为扫码的时候，手机会通过局域网广播来找到设备，再将服务器的信息部署到设备，让设备去连接服务器，如果手机和设备不在局域网内，就找不到需要部署的设备，比如，如果你的手机连接了 4G/5G 网络，而设备在公司的内网，就会部署失败，这时，其实需要让你的手机也连接到公司的内网中。

9. 手机上显示部署成功，但在 DeviceOn/EIM 的服务器网页上，看不到该设备有上线。

首先，如果服务器绑定了域名，在扫码服务器端的二维码时，一定要通过 IP 地址+端口的方式来访问服务器，而不是通过域名访问。其次，请确认设备网络正常，并且设备可以访问到服务器的 IP 地址。最后，请确认服务器的 IP 地址没有发生变化，如果服务器的 IP 地址发生了变化，就需要重新部署。所以建议服务器设定为固定 IP 地址，如果是部署在云端，最好通过域名来访问。

10. DeviceOn/EIM 是否支持第三方整合?

支持，目前已经有很多内外部客户集成了 DeviceOn/EIM，如内部 iFactory，IoTedge，DeviceOn，DeviceOn/BI，Edge365 等，他们分别用于不同的场景。关于更多整合合作详情，请联系 DeviceOn/EIM team 相关人员。

上海赋华慧创智能信息技术科技有限公司

地址:上海市静安区江场三路 56、58 号 9 楼 902-1 室

网址: www.fuhuaxc.com

赋华慧创官方微信