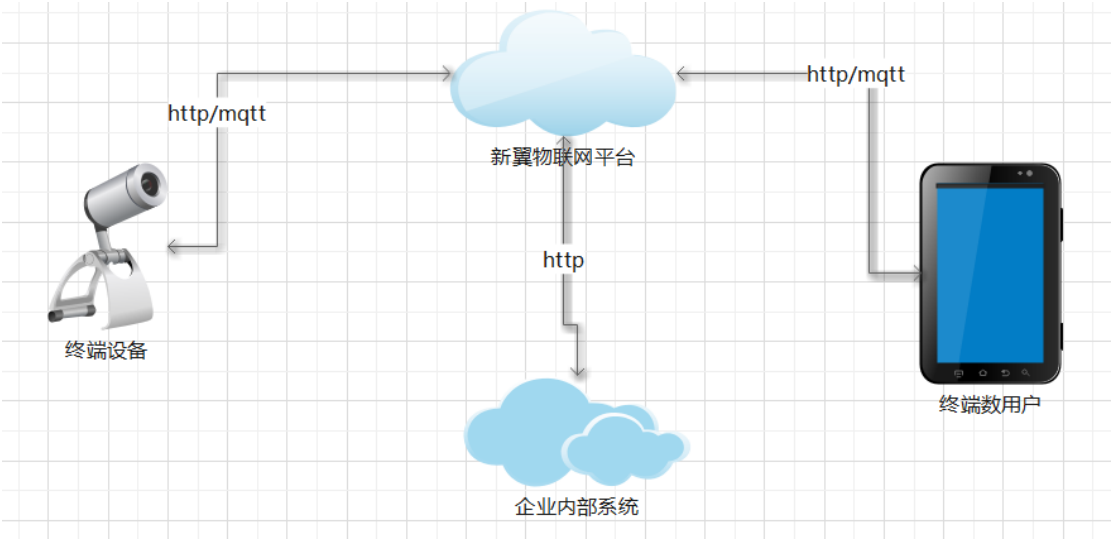
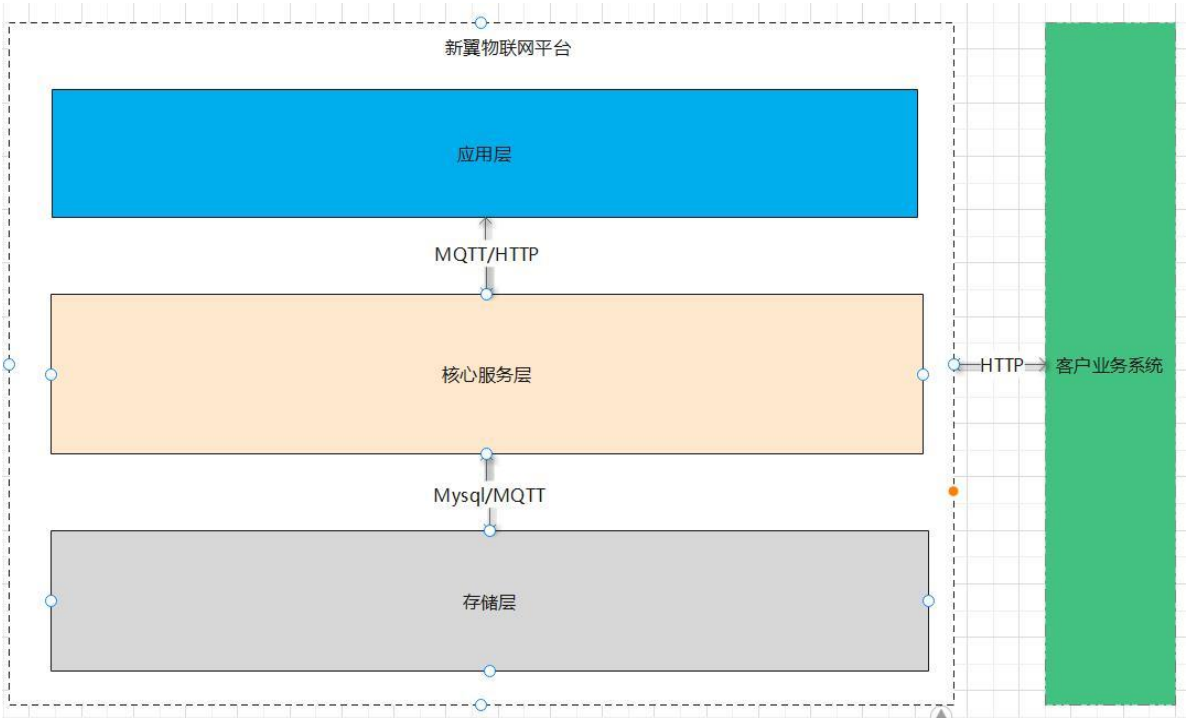
目录

1 系统简介

1.1 架构介绍

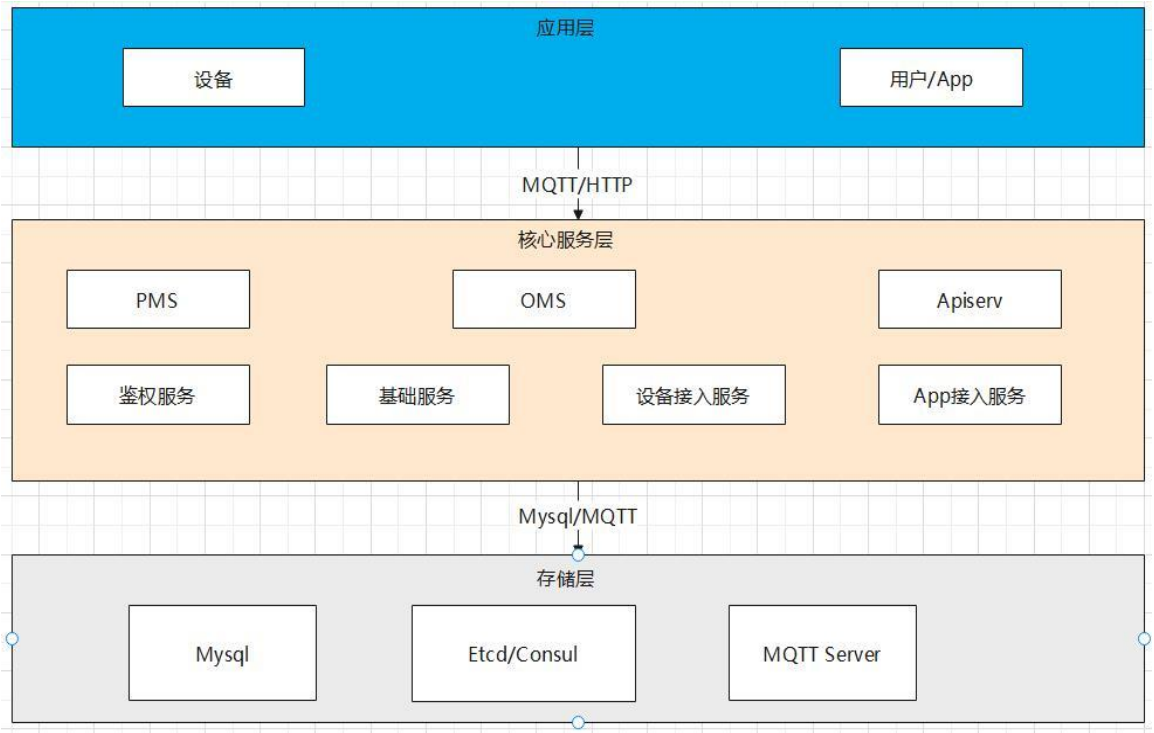
新翼物联网平台（英文简写：Xiots）以下简称为平台，采用微服务架构，通过注册/发现机制实现各个微服务之间的调度协作。





1.2 模块组成

平台各服务按照业务领域划分为：鉴权服务、基础服务、设备接入系统、App 接入系统、OMS、PMS、ApiService。



各服务组件主要功能如下：

鉴权服务（authserv）：实现设备、用户接入时的身份校验。

基础服务(baseserv)：实现平台内部基础接口，如用户控制消息的投递、用户与设备信息的查询等。

设备接入系统(device-connector)：设备通过 MQTT 连接到平台的核心业务，如上下线，事件上报，属性设置等。

App 接入系统(app-connetor)：用户通过 MQTT 连接到平台的核心业务，如上下线，设备控制、消息通知等。

OMS：运营管理系统。为企业用户调用平台能力提供支持。

接入请见《OMS.pdf》接入文档。

PMS：平台管理系统。为平台提供统一的 web 管理界面。

ApiService：企业用户的用户端接口，支持用户注册、登录、绑定/解绑设备，创建群组，群组转让等。

接入帮助请见《ApiService.pdf》的接入文档。

1.3 平台依赖

存储层使用关系型数据库：mysql，消息中心采用开源的 EMQX，注册中心支持 consul 与 etcd 两种分布式注册中心。

2 准备工作

2.1 系统要求

系统配置：所有服务均部署在同一台机器上，建议内存不小于 4G，CPU 为 i5 或以上，分布式部署，则节点内存大于 2G 即可。

操作系统：支持主流的 linux 发行版，如 centos、ubuntu。Windows 支持 win7 及以上。

2.2 第三方软件安装

您需要安装以下三款第三方软件:

数据库: mysql-server5.7 以上。

注册中心: etcd 3.4.15 或 consul 1.13.2。

MQTT Server : emqx 4.3.10 下载地址 : https://www.emqx.com/zh/try?product=enterprise

3 安装步骤

3.1 安装包介绍

系统安装包为针对各平台编译好的二进制包，且已经将各组件按照组件名称划分到不同目录。

每个组件目录中包含组件的二进制主程序与配置目录。

证书目录 cetificate: 所有组件启动时均需要读取证书，默认下载的安装包中有一个试用 3 个月的证书。

3.2 组件安装

直接解压安装包即完成组件安装。您可以根据需要将所有组件放在同一个目录，也可以分配到不同目录。

解压结束后, 需要将安装包中的 sql 文件导入到 Mysql 中, 导入后将创建数据库: xiots 。您需要为数据库 xiots 创建用户并分配可 select，delete，remove，update，insert 的权限。

3.3 组件配置

所有组件的配置都通过当前目录下的 config/config.json 进行。除了 PMS 外，所有组件的配置项均相同。各配置项说明见以下 JSON

```
{
    "server_id":"n2",
    "bind_address":"0.0.0.0:36012",
    "registry_center":"etcd",
    "registry_address":"127.0.0.1:2379",
    "registry_secure":false,
    "registry_tls":"",
    "cert_path":"/workspace/data/",
    "log_path":"/workspace/logs/device-connector/",
    "log_file":"roll.log",
    "log_level":"DEBUG"
}
```

server_id 服务器节点标识，可以为任意字符串与数字组合，同一服务不同的节点上 server_id 建议不同。

bind_address: http 服务绑定的 ip 与端口，如果是 0.0.0.0，则公网可以访问。

registry_center : 注册中心类型. Etcd 则表示以 etcd 为注册中心；consul：表示以 consul 为注册中心.

registry_secure: 注册中心是否采用安全链接，暂时请全部填 false。

registry_tls : 注册中心采用 tls 配置，目前请全部配置为空。

cert_path: 证书目录，请保证可读权限。

log_path：日志目录，请保证可读写。

log_file：滚动日志的文件名。

log_level：日志级别。有 INFO，DEBUG，ERROR，WARN。

PMS 配置说明：

相比其他组件，PMS 配置需要对以下三个配置项做额外说明：

"app_id":"aad094274f46650d6224300845dcb9c6"

"app_secret":"7432622f99fd8d83ebf23d78d27df09a"

"bind_address":"0.0.0.0:36001"

app_id ： pms 前端 js 的 appid，可以配置为任意的 32 个长度字符串。

app_secret: pms 前端 js 的 app_secret,可配置为任意的 32 个长度字符串。

bind_address：绑定地址，此地址为通过浏览器访问的地址。如例子所示，如果部署的服务器 ip 为：192.168.11.30，

则通过浏览器访问 http://192.168.11.30:36001/ 即可进入 pms 界面。

PMS 默认用户名密码为： admin:wingyouth123

3.4 服务启动

所有服务配置完，启动之前，需要您做以下两件事：

1、 在您的注册中心 key/value 中创建 xiots_config 目录。

2、 在其中增加以下配置项：

mysql //数据库配置，请填写步骤 3.2 时配置的数据库用户名与密码。

{"dsn":"iotdb_user:wingyouth@tcp(db.wingyouth.cloud)/xiots?charset=utf8mb4&parseTime=True&loc=Local","log_mode":4}

us_mqtt //app 的 mqtt，可以任意设置 32 位长度的用户与密码

{"host":"127.0.0.1:1883","user":"2ea662d49824fb43f49b33bcfd40cfd4","password":"d3e6a59e4ffb09eb9407d735f0e1f8f4","secret_key":"d394d64c39bc14de1767bf9dc2e65e4d"}

ds_mqtt //设备连接的 mqtt，可以任意设置 32 位长度的用户与密码

{"host":"127.0.0.1:1883","user":"f07e3851b2c7496c637ae7ba51beb1f9","password":"bb9e3b65ce25cd88414d711fd9e9c059","secret_key":"397227c61e4e715f13d4cd6b57b76294"}

intra_api //内部服务之间调用的校验 appid 与密钥，可设置任意 32 个长度的字符串

{"app_id":"91943e1465d1dd482c62252893479f58","app_secret":"0ec72c73381e213dbdc6b4fa2319fd98"}

以上配置值

以上步骤完成后，您可以直接启动各服务组件了。原则上 authserv 服务最先启动，其他组件启动顺序不限。

Linux 下启动 authserv 如下所示：

假设安装包解压在 /workspace/data/xiots 目录下，则：

```
cd /workspace/data/xiots/authserv/
nohup ./authserv &
```

其他组件均可按照此方式启动。

3.5 MQTT Server 配置

服务启动后，需要配置 emqx 的鉴权方式。步骤如下：

1、 进入 emqx 的安装目录 vim etc/acl.conf，修改为如下配置：

```
{allow, {user, "dashboard"}, subscribe, ["$SYS/#"]}.
{allow, {ipaddr, "127.0.0.1"}, pubsub, ["$SYS/#", "#"]}.
{deny, all, subscribe, ["$SYS/#", {eq, "#"}]}.
```

2、 vim etc/emqx.conf 更改以下配置项：

```
allow_anonymous = false
acl_nomatch = deny
broker.shared_dispatch_ack_enabled = true
```

3、 vim etc/plugins/auth_http.conf

```
##==============以下为 auth 鉴权==========
auth.http.auth_req = {{auth_server}}/auth/client/auth
auth.http.auth_req.headers.content_type = application/json
auth.http.auth_req.method = post
auth.http.auth_req.params = clientid=%c,username=%u,password=%P,ipaddress=%a
##==============以下为 super user 鉴权==========
auth.http.super_req.url =  {{auth_server}}/auth/super/auth
auth.http.super_req.method = post
auth.http.super_req.headers.content-type = application/json
auth.http.super_req.params = clientid=%c,username=%u,password=%P,ipaddress=%a

##==============以下为 acl 鉴权============
auth.http.acl_req =  {{auth_server}}/auth/client/acl
auth.http.acl_req.method = post
auth.http.acl_req.headers.content-type = application/json
auth.http.acl_req.params = access=%A,username=%u,clientid=%c,ipaddr=%a,topic=%t
```

注意，其中{{auth_server}} 请根据 authserv 的配置项 bind_address 的绑定端口与所在服务器 ip 组成形如 http://192.168.33.10:36006/ 这样的 url。

3.6 分布式部署

平台的每个服务均支持多节点，分布式部署。您只需要将服务的文件与证书复制到不同节点，并保证节点之间的网络连通正常即可。

4 系统接入

4.1 支持的系统

目前支持接口调用的服务有：ApiServ、OMS。

4.2 接入方式

所有支持接入的服务均通过 HTTP 协议接入。

4.3 鉴权方式

服务的接口调用采用了签名+JWT 的方式进行鉴权。调用接口之前，均需要通过 appid（pms 的应用管理里添加应用即可）调用对应服务的授权接口获取 JWT 的 Token。然后将获取到的 token 填入 header 的 Token 字段传到服务端以进行身份校验。

PHP 示例如下：

```php
<?php
$curl = curl_init();
```

```
curl_setopt_array($curl, array(
  CURLOPT_URL => '192.168.33.10:36003/oms/user/add',
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => '',
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => 'POST',
  CURLOPT_POSTFIELDS =>'{"username":"13424240433","password":"000000"}',
  CURLOPT_HTTPHEADER => array(
    'Content-Type: application/json',
    'Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBfaWQiOiIiLCJpc3MiOiJ3aW
5neW91dGgtand0IiwiZXhwIjoxNjY5NzIzMzc5fQ.oYa__pJcolCLAXHIv1F3iIaYW7532u0TEG9dm1
t6ZGo'
  ),
));
$response = curl_exec($curl);
curl_close($curl);
echo $response;
```

其中，header 中的 Token 值通过 4.4 中的签名算法，调用对应服务的授权接口获得。

4.4 签名算法

获取 token 时，需要通过一定的签名算法调用授权接口。以下为 postman 中的 javascript 的签名代码：

```
var app_id = "*********"; //pms 里添加的 appid
var rid = "*********"; //可以为随机的字符串或者时间戳
var secret = "*********";//对应的密钥
var param = app_id + rid  + secret;
var sign = CryptoJS.MD5(param, {
    asString: true
  });
```

此处 sign 即为最终的签名字符串。

最后，构造 post 的 body 体中 json 格式字符串。

```
{"app_id":"{{iot-oms-appid}}","timestamp":"{{rid}}","sign":"{{sign}}"}
```

PHP 实例如下：

```
<?php
$curl = curl_init();
curl_setopt_array($curl, array(
  CURLOPT_URL => '192.168.33.10:36003/oms/auth/token',
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => '',
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
```

```
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => 'POST',
  CURLOPT_POSTFIELDS =>'{"app_id":"d0ac73cef081e3653a252f81d989f483","timestamp
":"PPR1.180610.011","sign":"54ce0f7b8be76840cd8683b2fd53a6fc"}'
));
$response = curl_exec($curl);
curl_close($curl);
echo $response;
```