目 录

[GIO运维文档 4](#_Toc513455720)

[第1章 引言 4](#_Toc513455721)

[1.1 文档目的 4](#_Toc513455722)

[1.2 参考文档 4](#_Toc513455723)

[第2章 部署架构 5](#_Toc513455724)

[2.1 GIO系统架构图 5](#_Toc513455725)

[2.2 架构详解 5](#_Toc513455726)

[2.2.1 后端数据采集 5](#_Toc513455727)

[2.2.2 前端访问 6](#_Toc513455728)

[第3章 服务管理 6](#_Toc513455729)

[3.1 marathon 6](#_Toc513455730)

[3.1.1 介绍 6](#_Toc513455731)

[3.1.2 操作 6](#_Toc513455732)

[3.1.3 访问marathon 6](#_Toc513455733)

[3.1.4 marathon的启动与关闭 7](#_Toc513455734)

[3.2 frontend服务 7](#_Toc513455735)

[3.2.1 介绍 7](#_Toc513455736)

[3.2.2 操作 7](#_Toc513455737)

[3.2.3 frontend的启动与关闭 8](#_Toc513455738)

[3.2.4 frontend的基本操作 8](#_Toc513455739)

[3.3 gateway服务 10](#_Toc513455740)

[3.3.1 介绍： 10](#_Toc513455741)

[3.3.2 状态： 10](#_Toc513455742)

[3.3.3 gateway的启动与关闭 11](#_Toc513455743)

[3.3.4 gateway的基本操作 11](#_Toc513455744)

[3.4 accounts服务 11](#_Toc513455745)

[3.4.1 介绍 11](#_Toc513455746)

[3.4.2 状态 12](#_Toc513455747)

[3.4.3 accounts的启动与关闭 12](#_Toc513455748)

[3.4.4 accounts的基本操作 12](#_Toc513455749)

[3.5 charts-service服务 12](#_Toc513455750)

[3.5.1 charts-service的启动与关闭 13](#_Toc513455751)

[3.5.2 基本操作 13](#_Toc513455752)

[3.6 backend服务 13](#_Toc513455753)

[3.6.2 backend的启动与关闭 14](#_Toc513455754)

[3.6.3 backend基本操作 14](#_Toc513455755)

[3.7 query-service服务 14](#_Toc513455756)

[3.7.1 介绍 14](#_Toc513455757)

[3.7.2 状态 15](#_Toc513455758)

[3.7.3 query-service的启动与关闭 15](#_Toc513455759)

[3.7.4 query-service的基本操作 15](#_Toc513455760)

[3.8 shortener服务 15](#_Toc513455761)

[3.8.1 介绍 15](#_Toc513455762)

[3.8.2 状态 16](#_Toc513455763)

[3.8.3 shortener的启动与关闭 16](#_Toc513455764)

[3.8.4 shortener的基本操作 16](#_Toc513455765)

[3.9 id-service服务 16](#_Toc513455766)

[3.9.1 介绍 16](#_Toc513455767)

[3.9.2 状态 17](#_Toc513455768)

[3.9.3 id-service的启动与关闭 17](#_Toc513455769)

[3.9.4 id-service的基本操作 17](#_Toc513455770)

[3.10 vds-api服务 17](#_Toc513455771)

[3.10.1 介绍 17](#_Toc513455772)

[3.10.2 状态 18](#_Toc513455773)

[3.10.3 vds-api的启动与关闭 18](#_Toc513455774)

[3.10.4 *vds-api*的基本操作 18](#_Toc513455775)

[3.11 log2kafka服务 18](#_Toc513455776)

[3.11.1 介绍 18](#_Toc513455777)

[3.11.2 状态 18](#_Toc513455778)

[3.11.3 log2kafka的启动与关闭 19](#_Toc513455779)

[3.11.4 *log2kafka*的基本操作 19](#_Toc513455780)

[3.12 elacticsearch服务 19](#_Toc513455781)

[3.12.1 介绍 19](#_Toc513455782)

[3.12.2 状态 19](#_Toc513455783)

[3.12.3 elasticsearch的启动与关闭 20](#_Toc513455784)

[3.12.4 elasticsearch的基本操作 20](#_Toc513455785)

[3.13 kafka集群 20](#_Toc513455786)

[3.13.1 介绍 20](#_Toc513455787)

[3.13.2 状态 20](#_Toc513455788)

[3.13.3 kafka的启动与关闭 21](#_Toc513455789)

[3.13.4 kafka的基本操作 21](#_Toc513455790)

[3.14 online服务 22](#_Toc513455791)

[3.14.1 介绍 22](#_Toc513455792)

[3.14.2 服务状态 22](#_Toc513455793)

[3.14.3 online的启动与停止 22](#_Toc513455794)

[3.14.4 online的基本操作 22](#_Toc513455795)

[3.15 Offline服务 23](#_Toc513455796)

[3.15.1 介绍 23](#_Toc513455797)

[3.15.2 服务状态 23](#_Toc513455798)

[3.15.3 启动与停止 23](#_Toc513455799)

[3.15.4 offline基本操作 23](#_Toc513455800)

[3.16 Hadoop服务 24](#_Toc513455801)

[3.16.1 介绍 24](#_Toc513455802)

[3.16.2 Hadoop的启动与停止 24](#_Toc513455803)

[3.16.3 Hadoop基本操作 24](#_Toc513455804)

[3.17 Hbase集群 25](#_Toc513455805)

[3.17.1 介绍 25](#_Toc513455806)

[3.17.2 服务状态 25](#_Toc513455807)

[3.17.3 hbase的启动与停止 25](#_Toc513455808)

[3.17.4 hbase基本操作 25](#_Toc513455809)

[3.18 zookeeper集群 25](#_Toc513455810)

[3.18.1 介绍 25](#_Toc513455811)

[3.18.2 服务状态 26](#_Toc513455812)

[3.18.3 zookeeper启动与停止 26](#_Toc513455813)

[3.19 postgres数据库 26](#_Toc513455814)

[3.19.1 介绍 26](#_Toc513455815)

[3.19.2 服务状态 26](#_Toc513455816)

[3.19.3 postgres启动地址 26](#_Toc513455817)

[3.19.4 postgres基本操作 27](#_Toc513455818)

[3.20 redis服务 28](#_Toc513455819)

[3.20.1 介绍 28](#_Toc513455820)

[3.20.2 状态： 28](#_Toc513455821)

[3.20.3 redis的启动与关闭 28](#_Toc513455822)

[3.20.4 redis的基本操作 28](#_Toc513455823)

[第4章 常见的问题 29](#_Toc513455824)

[4.1 界面中没有添加角色的按钮 29](#_Toc513455825)

[4.2 集成应用和数据流排查流程 29](#_Toc513455826)

[4.2.1 online追数 31](#_Toc513455827)

[4.2.2 查看offline失败的任务 32](#_Toc513455828)

[4.2.3 容器服务升级 32](#_Toc513455829)

[4.2.4 非容器的升级 32](#_Toc513455830)

GIO运维文档

# 引言

## 文档目的

本文档主要提供gio系统的具体维护方案以及服务整体架构及规划。

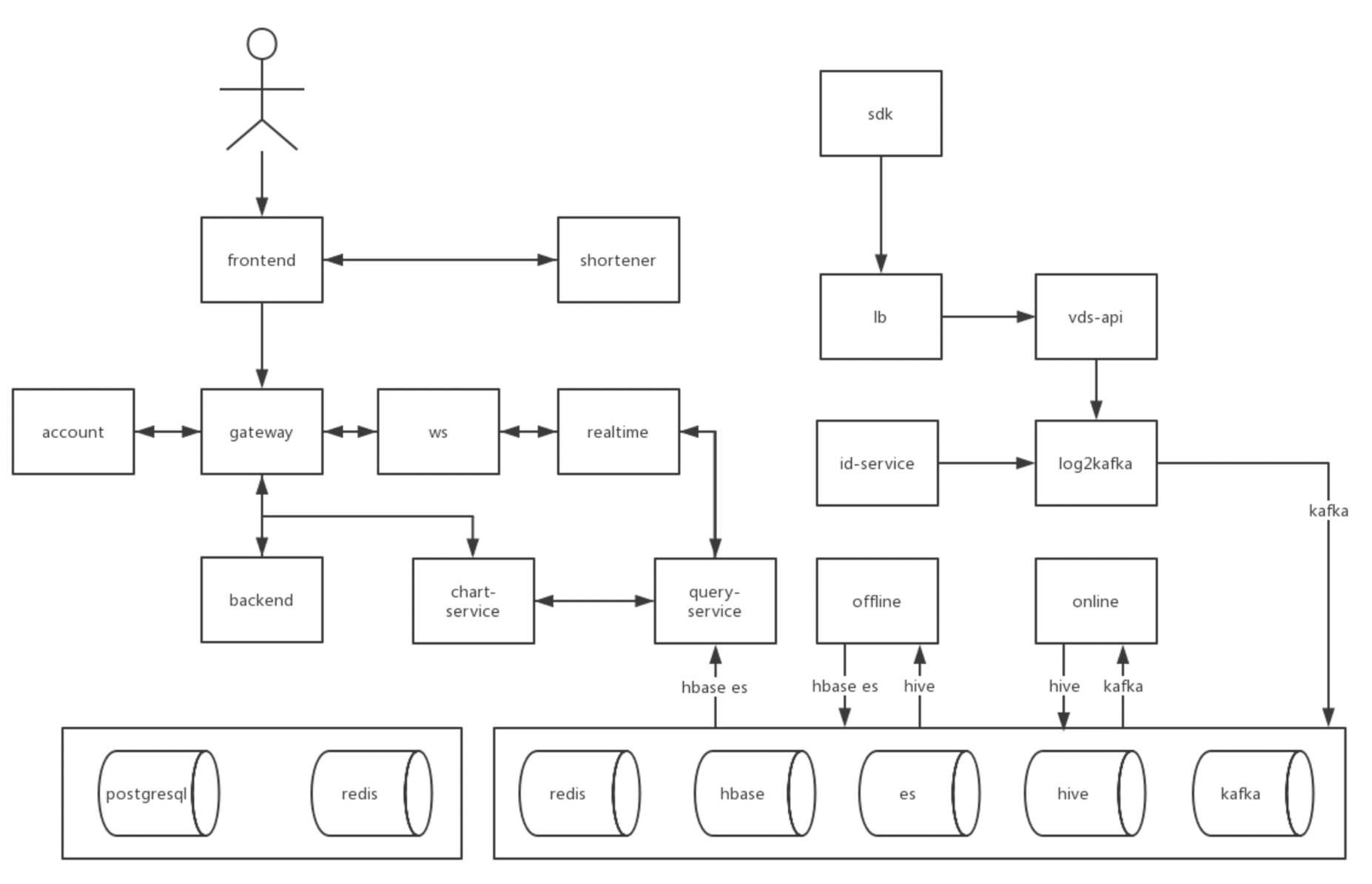
## 参考文档

|  |  |  |
| --- | --- | --- |
| 服务 | 版本 | 参考链接 |
| Hadoop | 2.7.3 | <https://hadoop.apache.org/docs/r2.7.3/> |
| spark | 2.1.1 | <http://spark.apache.org/docs/2.1.1/> |
| kafka | 0.8.2.2(二进制) | <http://kafka.apache.org/082/documentation.html> |
| hbase | 0.98 | <https://hbase.apache.org/book.html> |
| Postgres | 9.5 | <https://www.postgresql.org/docs/9.5/static/app-postgres.html> |
| docker | 1.13 | <https://docs.docker.com/v1.13/> |
| phoenix | 4.7 | <http://phoenix.apache.org/> |

* 备注：
* 以上都是官网链接，详情使用请参考官网使用方法

# 部署架构

## GIO系统架构图



## 架构详解

本架构分为两个部分：

一：后端数据采集部分

二：前端访问部分

### 后端数据采集

1. 需要对需要分析的网站或者app进行sdk集成，主要用于数据采集。
2. 然后数据会通过marathon-lb服务将数据传输至vds-api服务(前提是pg库里有该项目的ai)
3. vds-api服务收到数据后，log2kafka异步的读取vds的数据并且根据id-service将数据发送给kafka。
4. kafka收到数据后会，online服务会实时的去kafka里取数据并且将数据写入hive表中（原始数据）。
5. hive表里有了原始数据后offline服务会定时的是hive表里取数据进行计算，然后将数据写入hbase中。

### 前端访问

1. 用户首先进入frontend服务，frontend是一个页面渲染服务。

2．然后会通过gateway将请求跳转至account服务，gateway的作用就是链接各个服务，account就是一个认证服务。

3．然后会讲请求的一些指标数据会将请求通过gateway分发至backend服务。

4．backend服务会去chart-service里拿取数据。

5. chart-service也会去query-service拿取数据，而query-service会也会实时的去hbase请求数据

# 服务管理

## marathon

### 介绍

描述：介绍所有的前中端服务，也就是将所有的服务统一放置marathon管理，方便对服务的启动、停止、修改配置参数等操作

### 操作

登陆marathon服务器，使用docker ps 检查marathon是否在运行状态

命令：

docker ps ## 检查marathon是否启动

docker logs --tail 100 -f marathon-id ## 检查marathon运行日志

netstat -ltunp |grep 8080 ## 检查marathon端口是否被监听

### 访问marathon

用一台可以访问marathon内网ip的电脑，输入ip+端口即可访问marathon界面

浏览器输入：

<http://marathon>\_ip:8080

* 备注
* 确保可以ping通marathon服务器并且可以telnet通8080端口

### marathon的启动与关闭

启动：

登陆marathon服务器，切换到/apps/svr目录下执行install\_marathon.sh即可

cd /apps/svr && sh install\_marathon.sh

关闭：

可以使用docker rm进行关闭marathon

docker rm marathon

## frontend服务

### 介绍

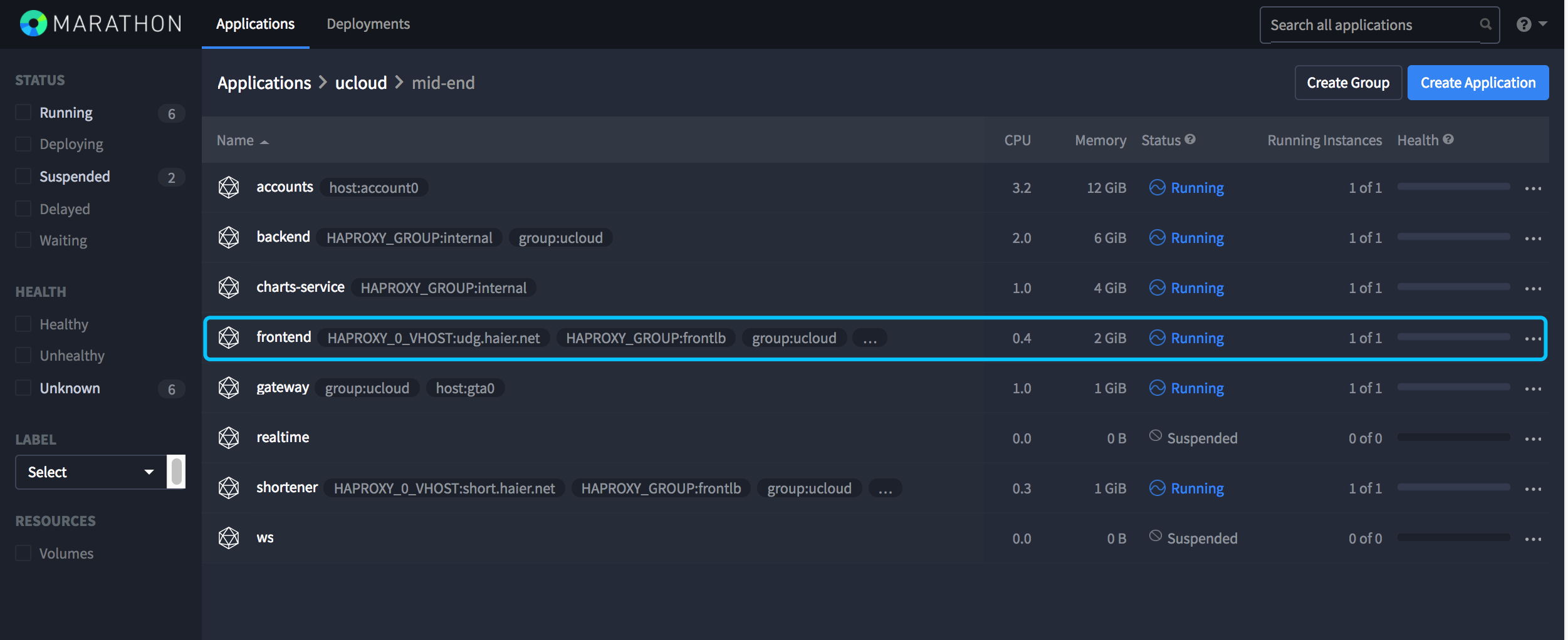
服务：frontend

描述：主要是前端页面的展示。

### 操作

状态

打开marathon界面frontend在ucloud/mid-end下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f front\_id ## 查看frontend的运行日志

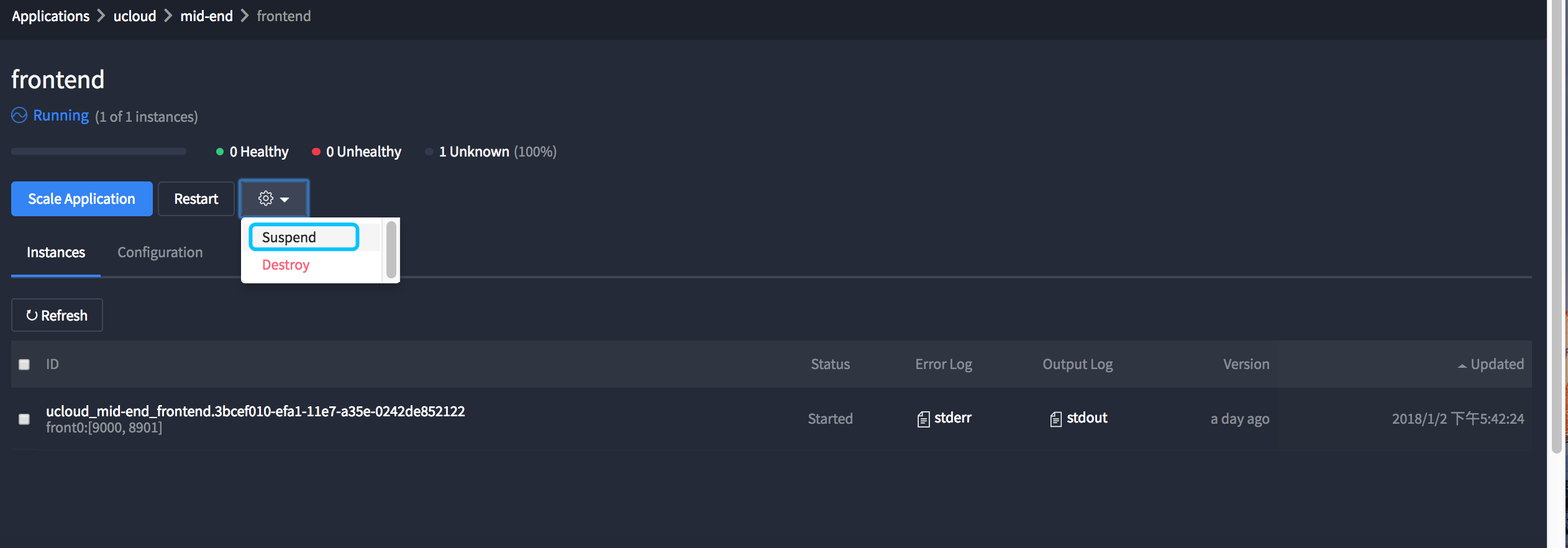
### frontend的启动与关闭

启动：

切换到/apps/svr/growing-frontend下执行install.sh即可

cd /apps/svr/growing-frontend && sh install.sh

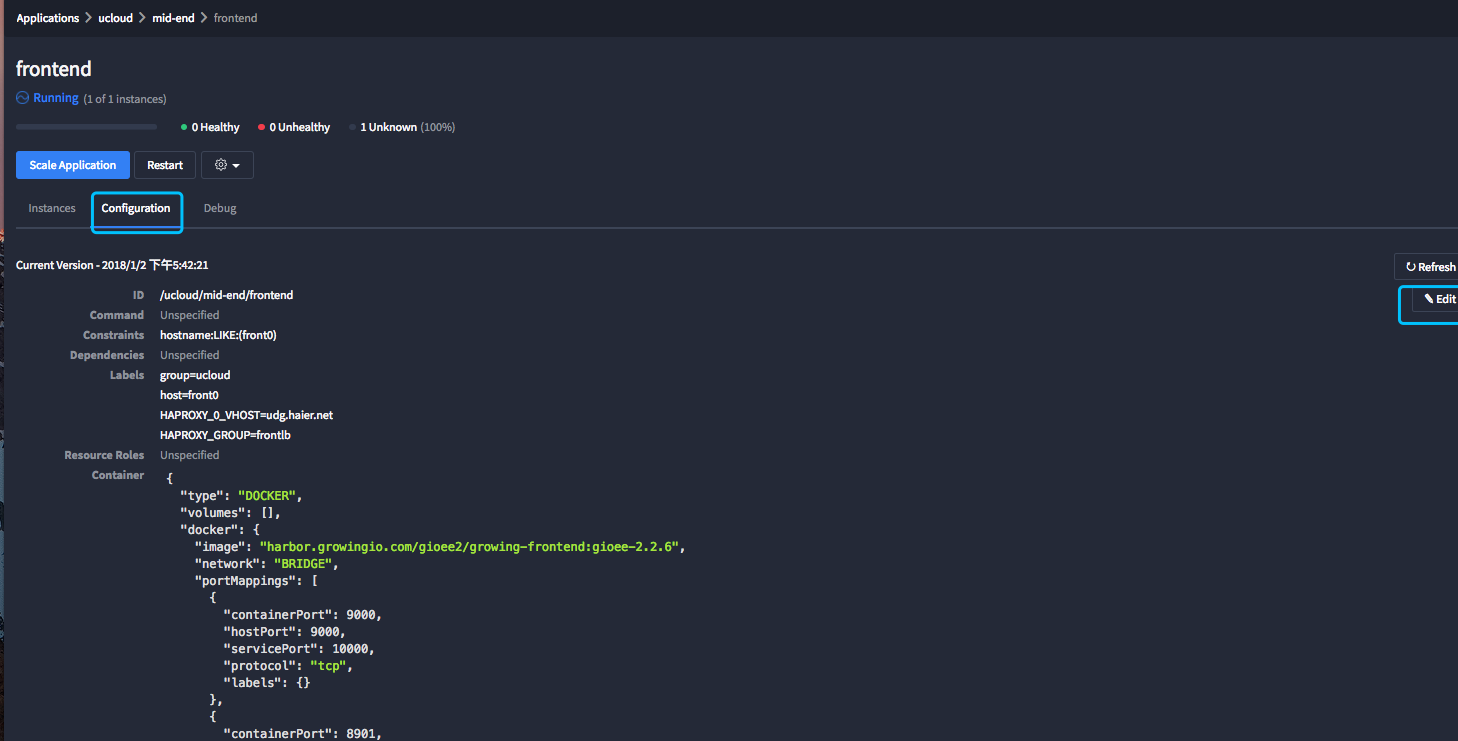
停止：



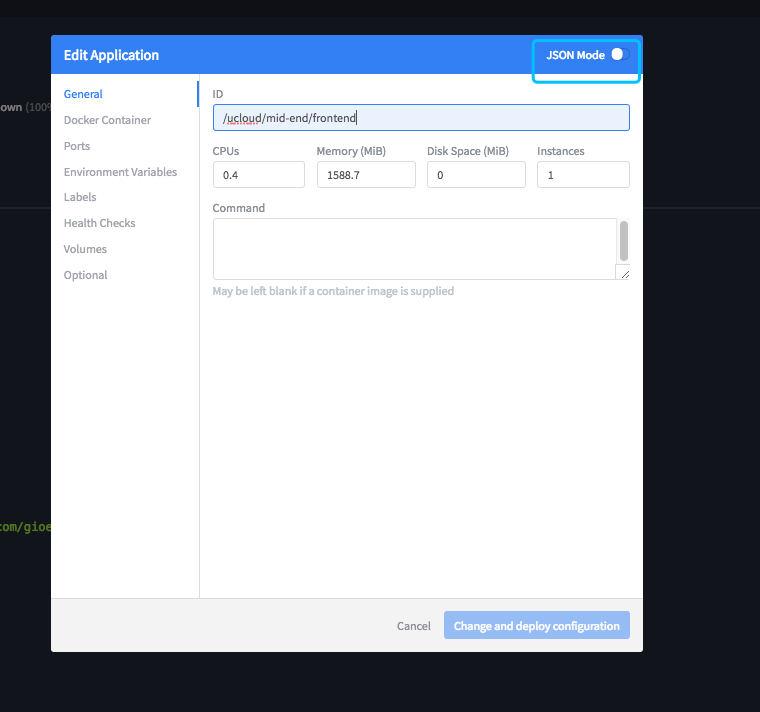
### frontend的基本操作

在执行启动install.sh服务的时候它会调用当前目录下的package.json这个文件是frontend的所有配置信息，也就是说是frontend的配置文件，执行成功后会在marathon界面看到frontend的状态信息，前提是marathon界面没有该服务，如果该服务已经正常运行，并不会影响当前运行的服务

修改配置：



点击配置—点击编辑会进入设置参数的界面



同时也支持json模式显示

## gateway服务

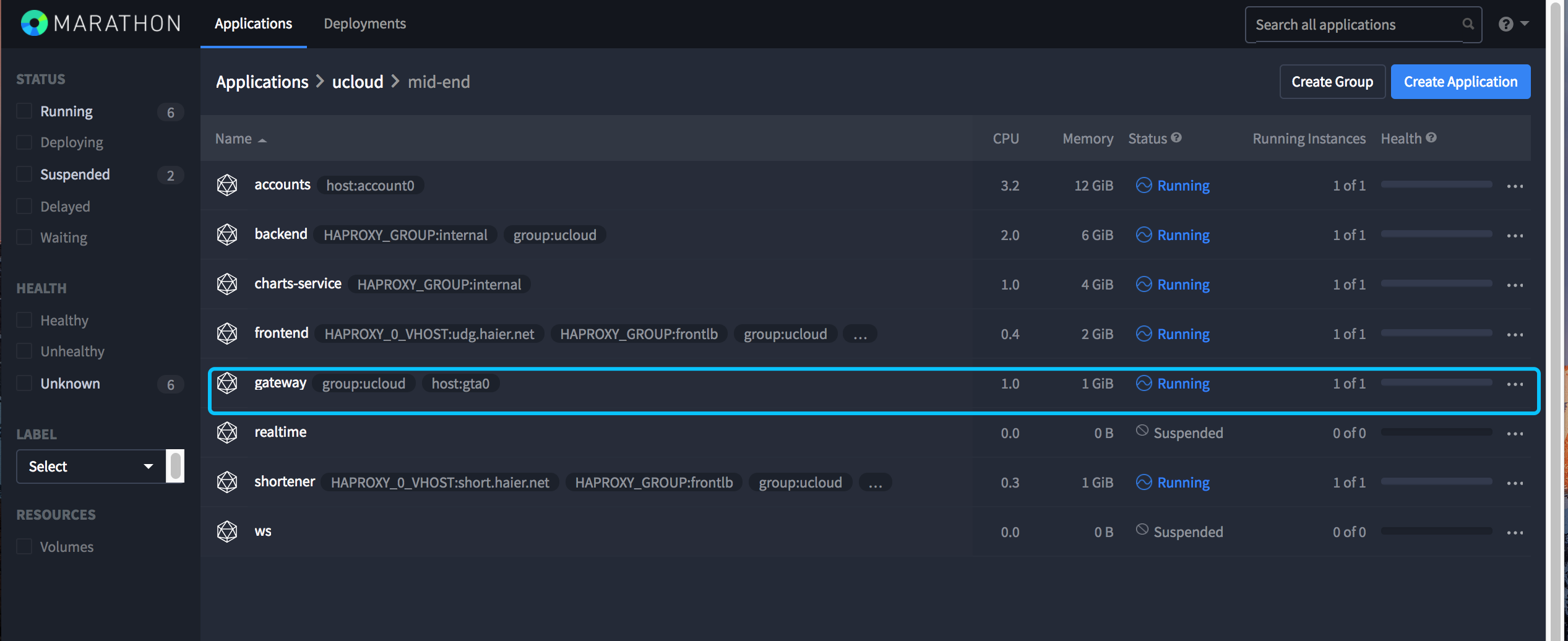
### 介绍：

服务：gateway

介绍：主要是链接account／backend和frontend

### 状态：

打开marathon界面gateway在ucloud/mid-end下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f gateway\_id ## 查看gatway的运行日志

### gateway的启动与关闭

启动：

切换到/apps/svr/growing- gateway下执行install.sh即可

cd /apps/svr/growing- gateway && sh install.sh

关闭：

参考frontend服务

### gateway的基本操作

参考frontend服务

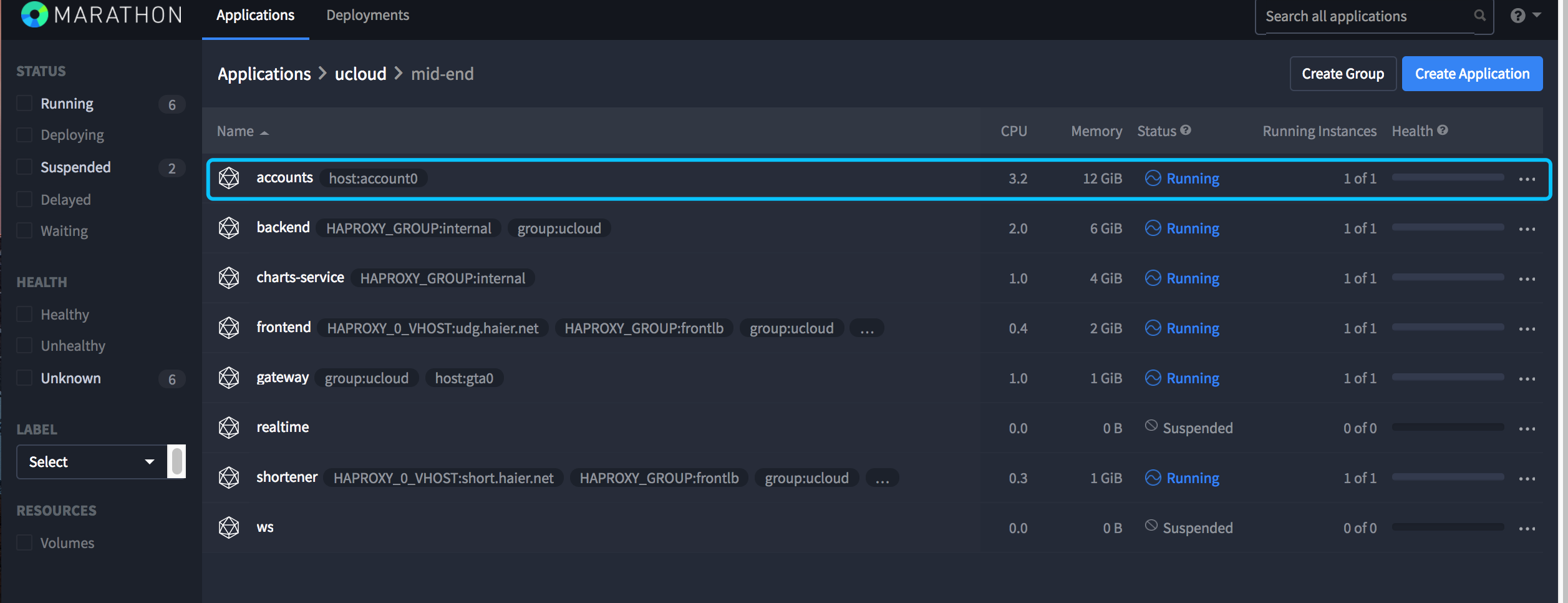
## accounts服务

### 介绍

服务：accounts

介绍：主要是认证服务

### 状态



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f account\_id ## 查看accounts的运行日志

### accounts的启动与关闭

启动：

切换到/apps/svr/growing- accounts下执行install.sh即可

cd /apps/svr/growing- accounts&& sh install.sh

关闭：

参考frontend

### accounts的基本操作

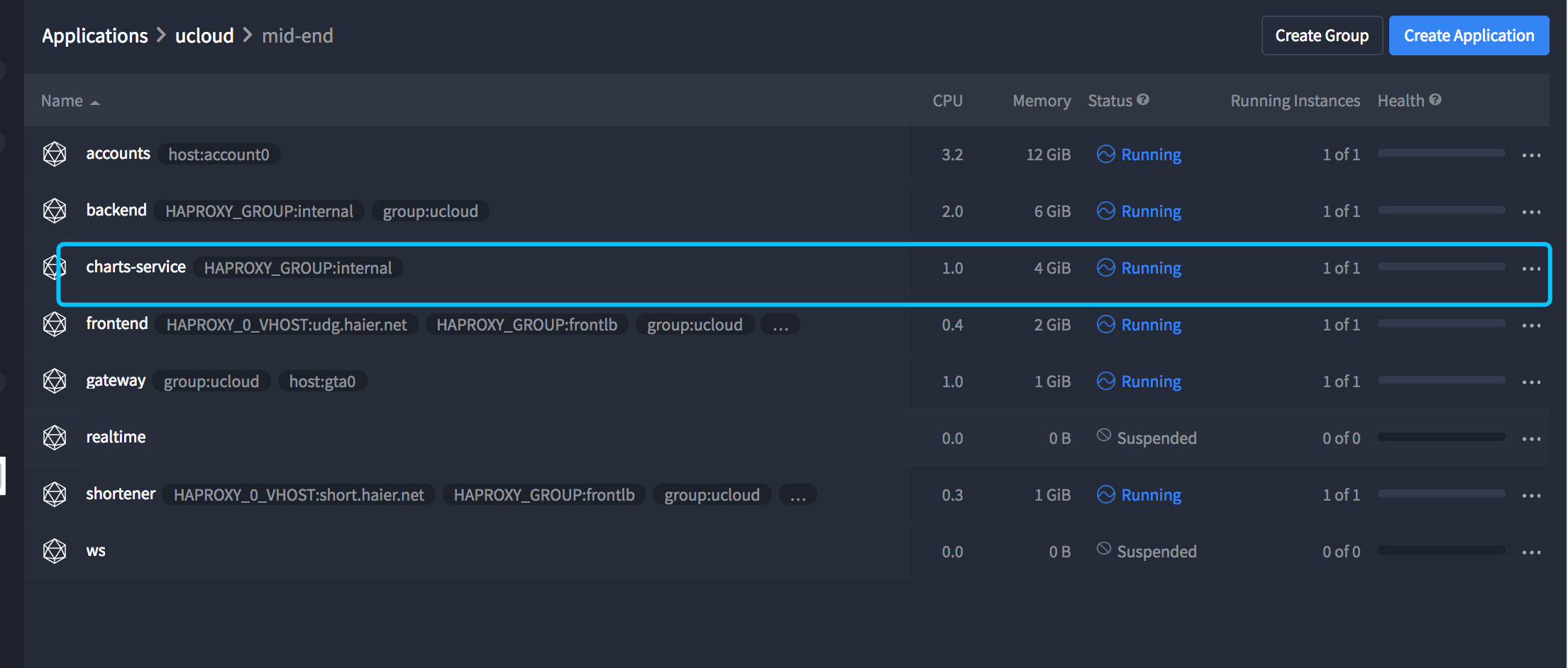
参考frontend服务

## charts-service服务

服务：charts-service

介绍：图显服务

打开marathon界面charts-service在ucloud/mid-end下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f charts\_id ## 查看charts的运行日志

### charts-service的启动与关闭

启动：

切换到/apps/svr/charts-service下执行install.sh即可

cd /apps/svr/charts-service && sh install.sh

关闭：

参考frontend

### 基本操作

参考frontend服务

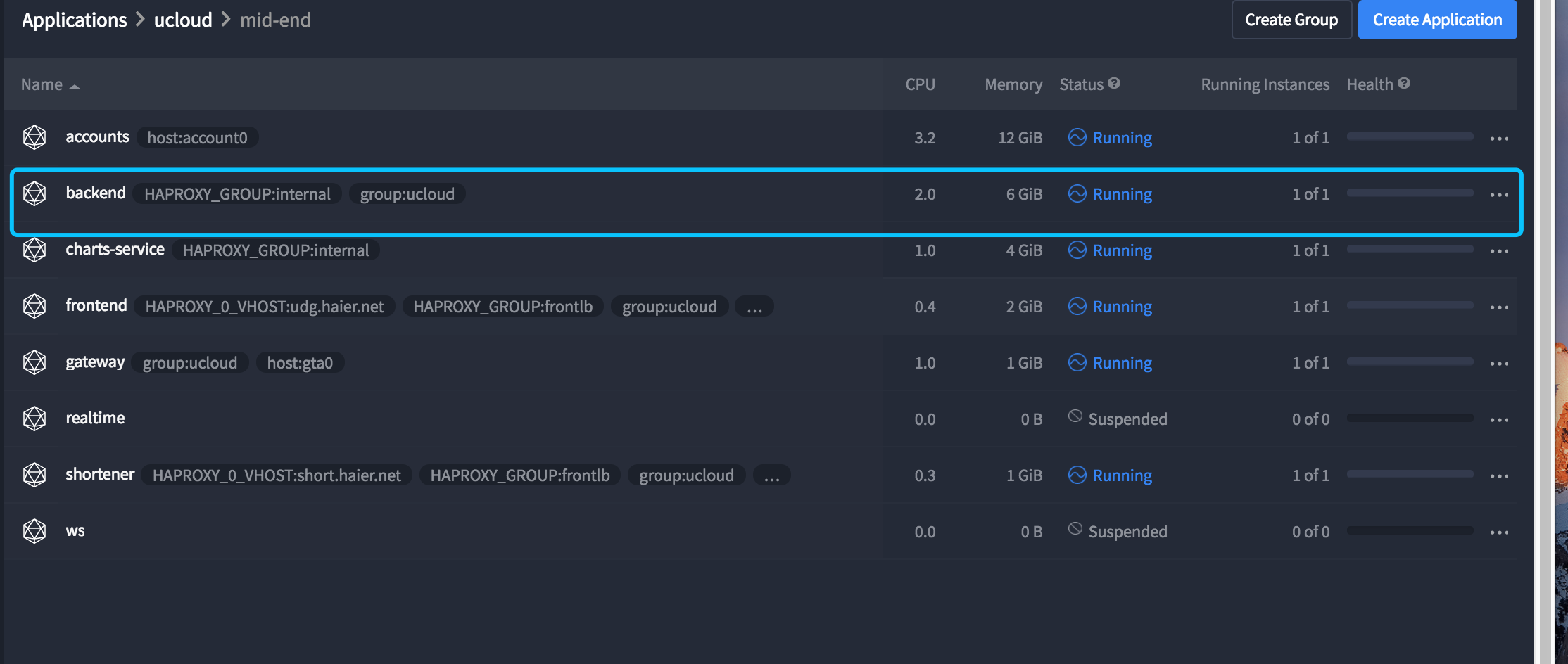
## backend服务

#### 介绍

服务：backend

介绍：后端数据展示的主服务

打开marathon界面backend在ucloud/mid-end下



命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f backend\_id ## 查看backend的运行日志

### backend的启动与关闭

启动：

切换到/apps/svr/growing backend下执行install.sh即可

cd /apps/svr/growing-backend && sh install.sh

关闭：

参考frontend

### backend基本操作

参考frontend服务

## query-service服务

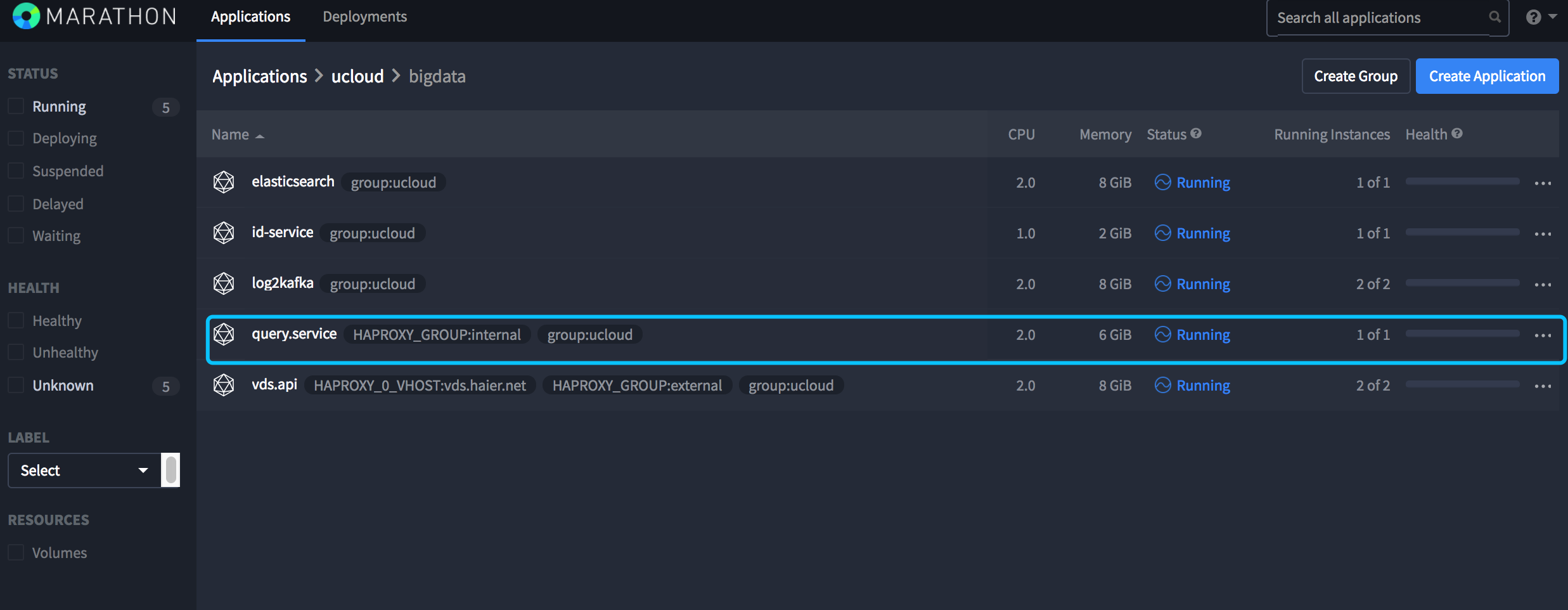
### 介绍

服务：query-service

介绍：后端数据查询服务

### 状态

打开marathon界面query-service在ucloud/bigdata下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f query\_id ## 查看query的运行日志

### query-service的启动与关闭

启动：

切换到/apps/svr/query-service下执行install.sh即可

cd /apps/svr/query-service && sh install.sh

关闭：

参考frontend

### query-service的基本操作

参考frontend服务

## shortener服务

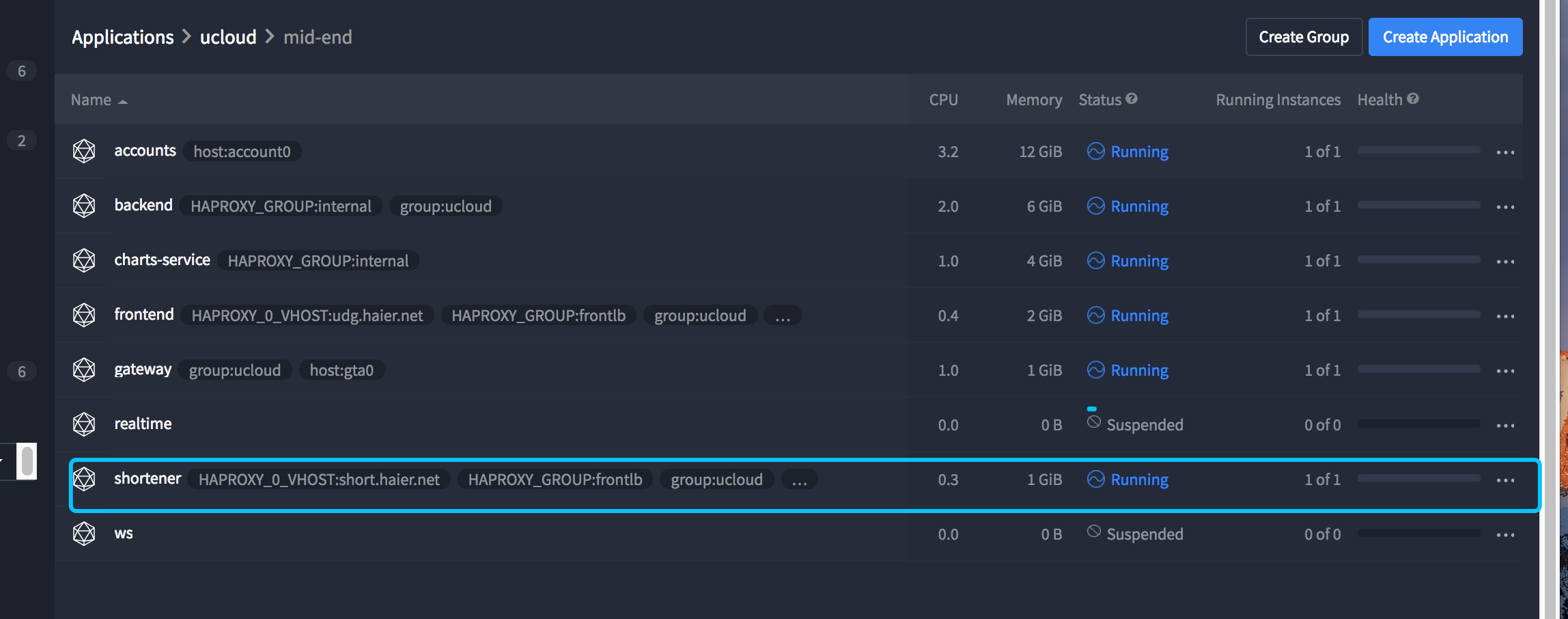
### 介绍

服务：shortener

介绍：短链服务，将长链接地址返回短连接地址，圈选app时需要扫描二维码，如果链接太长则无法识别，所有需要一个短链接服务转换下url地址。

### 状态

打开marathon界面shortener在ucloud/mid-end下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f shortener\_id ## 查看shortener的运行日志

### shortener的启动与关闭

启动：

切换到/apps/svr/growing-url-shortener下执行install.sh即可

cd /apps/svr/growing-url-shortener && sh install.sh

关闭：

参考frontend

### shortener的基本操作

参考frontend服务

## id-service服务

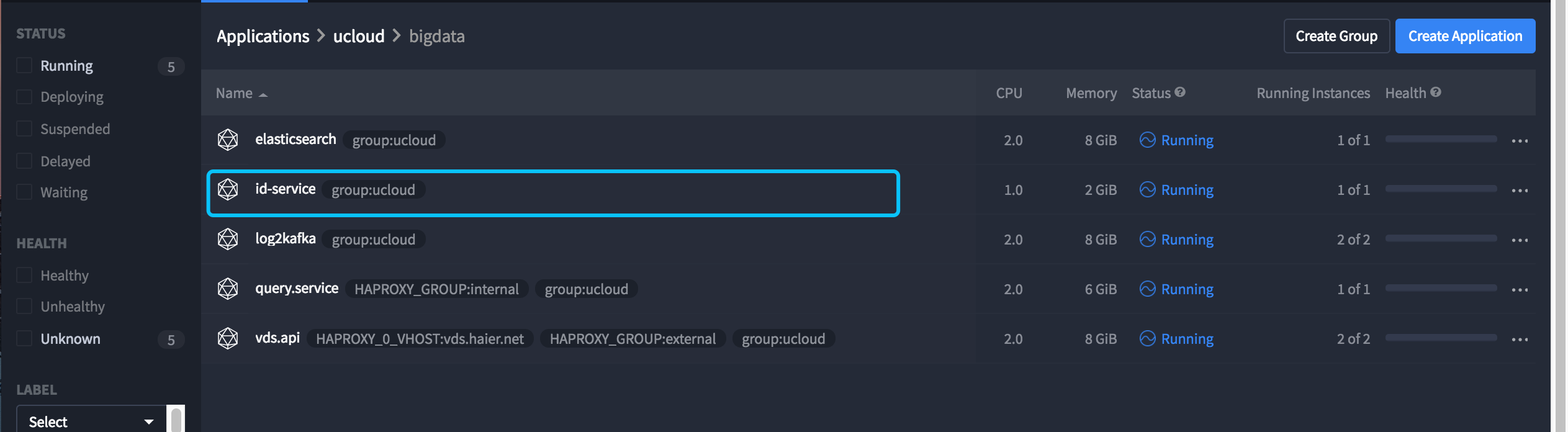
### 介绍

服务：id-service

介绍：根据访问的用户产生一个uid，用于判断多个设备是否是同一个用户

### 状态

打开marathon界面id-service在ucloud/bigdata下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f idservice\_id ## 查看idservice的运行日志

### id-service的启动与关闭

启动：

切换到/apps/svr/id-service 下执行update.sh即可

cd /apps/svr/id-service && sh update.sh

关闭：

参考frontend

### id-service的基本操作

参考frontend服务

## vds-api服务

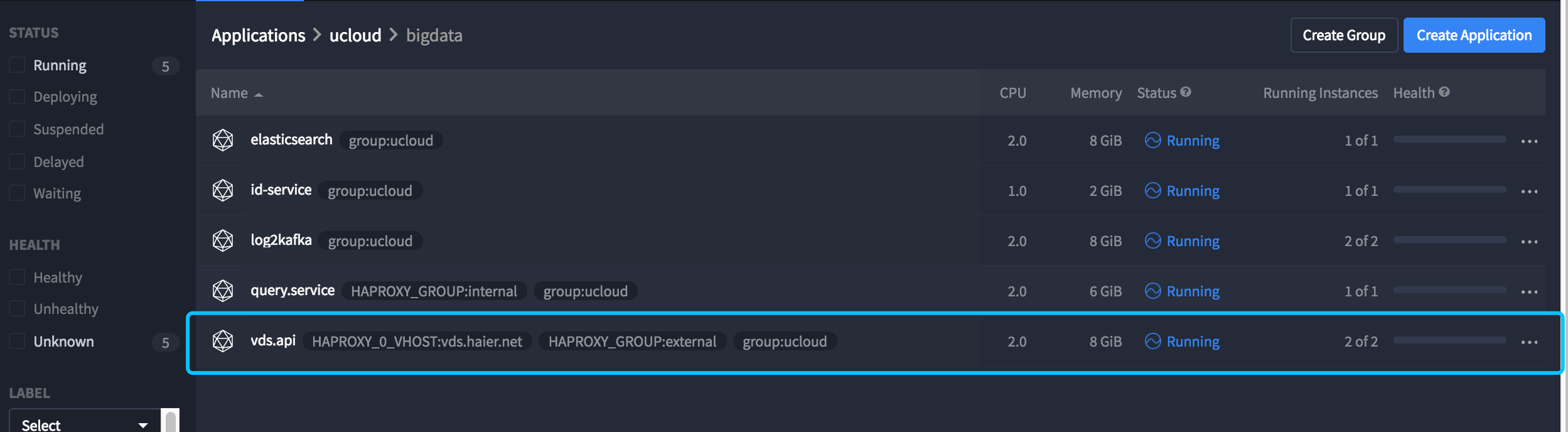
### 介绍

服务：vds-api（一般是两台）

介绍：后端数据收集上来存放的第一位置

### 状态

打开marathon界面vds-api在ucloud/bigdata下



status显示running就是运行状态

命令查看：

d docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f vdsapi\_id ## 查看vdsapi的运行日志

### vds-api的启动与关闭

启动：

切换到/apps/svr/vds-api/ 下执行install.sh即可

cd /apps/svr/vds-api && sh install.sh

关闭：

参考frontend

### vds-api的基本操作

参考frontend服务

## log2kafka服务

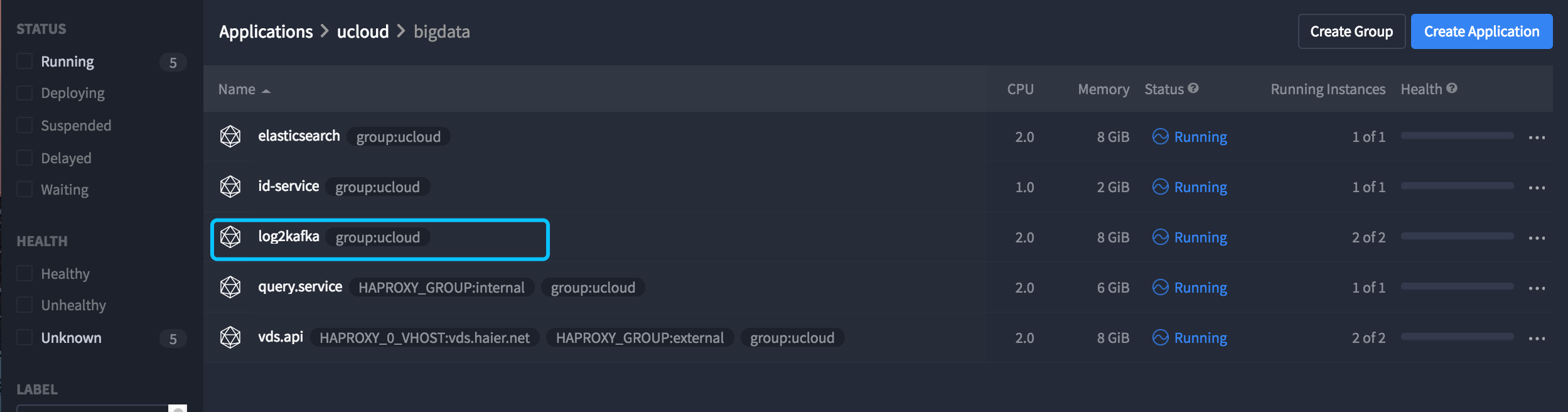
### 介绍

服务：log2kafka（一般是两台和vds成对出现）

介绍：会根据id-service服务返回的结果将数据传送至kafka服务

### 状态

打开marathon界面log2kafka在ucloud/bigdata下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f log2kafka\_id ## 查看log2kafka的运行日志

### log2kafka的启动与关闭

启动：

切换到/apps/svr/log2kafka 下执行update.sh即可

cd /apps/svr/log2kafka && sh update.sh

关闭：

参考frontend

### log2kafka的基本操作

参考frontend服务

## elacticsearch服务

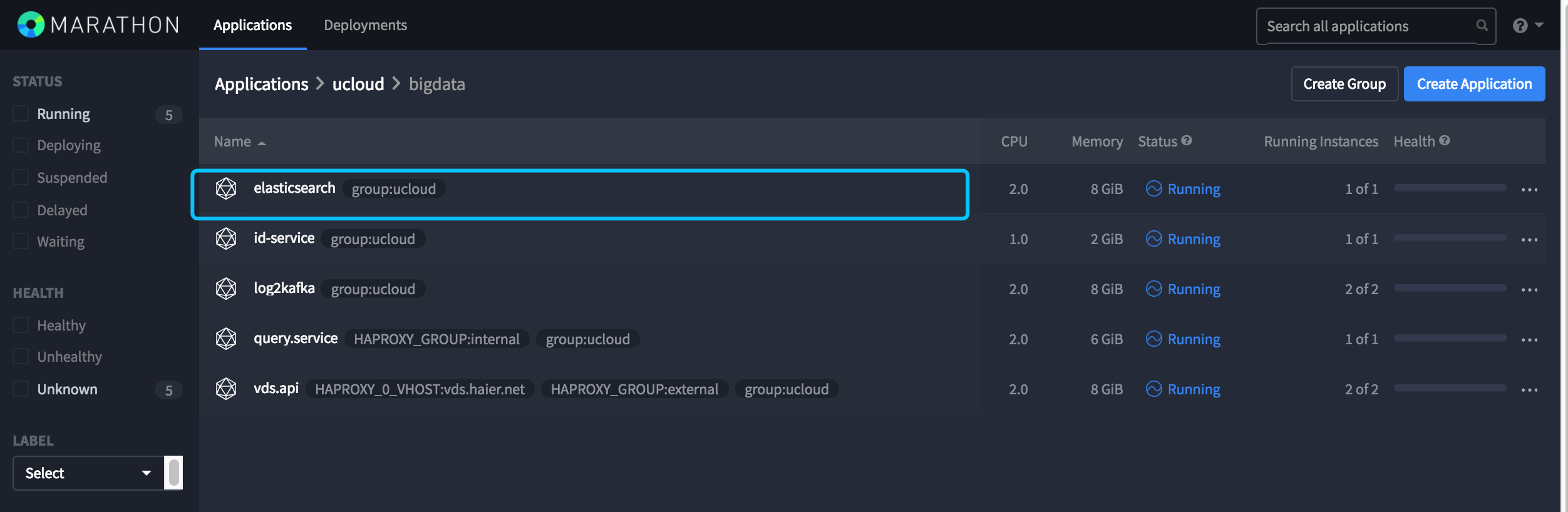
### 介绍

服务：elasticsearch

介绍：文本搜索服务

### 状态

打开marathon界面elasticsearch在ucloud/bigdata下



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f elasticsearch\_id ## 查看elasticsearch的运行日志

### elasticsearch的启动与关闭

切换到/apps/svr/elasticsearch下执行update.sh即可

cd /apps/svr/elasticsearch&& sh update.sh

关闭：

参考frontend

### elasticsearch的基本操作

参考frontend服务

## kafka集群

### 介绍

服务：kafka（一般是三台）

介绍：收集log2kafka传过来的数据，并且放到相对应的topic中

### 状态

命令：

cd /opt/growing-ansible ## 在ansbile服务器上运行

ansible -i inventories/project\_name/hosts kafka -m shell -a "jps |grep -i kafka"

### kafka的启动与关闭

启动：

ansible-playbook -i inventories/project\_name/hosts tool-scripts/start-kafka.yml # 启动所有kafka

关闭：

ansible-playbook -i inventories/project\_name/hosts tool-scripts/stop-kafka.yml # 关闭所有kafka

单台启动：(需要去kafka服务本地启动)

/apps/svr/kafka/bin/kafka-server-start.sh -daemon /apps/svr/kafka/config/server.properties

单台关闭：

kill kafka-id 即可

### kafka的基本操作

log2kfka会将采集上来的数据根据不通的topic传送至不同的topic中，kafka也会实时的接受数据

显示所有topic

切换至kafka服务器

cd /apps/svr/kafka

bin/kafka-topics.sh --zookeeper zookeeper:2181/kafka --list #指定zookeeper服务的主机名和端口

根据topic查看是否有数据显示

bin/kafka-console-consumer.sh --zookeeper zookeeper:2181/kafka --from-beginning --topic topic #写入需要查看的topic名称

* 因为是实时的如果有数据的话应该会实时的显示

## online服务

### 介绍

服务：online

介绍：实时处理kafka topic中的原始数据将数据写入hive中

### 服务状态

命令：

cd /opt/growing-ansible ## 在ansible服务器上运行

ansible -i inventories/project\_name/hosts online -m shell -a "jps"

### online的启动与停止

***启动：***

***切换至online服务器***

/apps/svr/online/bin/check.sh # 执行即可

关闭：

/apps/svr/online/bin/stop-all.sh

检查：

jps

### online的基本操作

online是在线实时任务，会把原始数据存放在hive中，hive没有安装用的是spark，所有如果要看数据的话需要登陆spark

cd /apps/svr/spark##切换至安装目录下

./bin/sql-spark --master local[2] ##进入hive中

use gio; ##切换至gio库

show tables; ## 查看所有表格

select \* from page; ## 通常查询page和visit表

show partitions page; ## 每半小时区分，还需要查看visit和action表，这是原始数据的表，如果没有数据需要查看他。

Select \* from page where time like ‘20180322%’;

## Offline服务

### 介绍

服务：offline

介绍：定时处理online写入hive中的原始数据，并且计算后写入hbase服务

### 服务状态

命令：

cd /opt/growing-ansible ## 在ansible服务器上运行

ansible -i inventories/project\_name/hosts offline -m shell -a "jps"

### 启动与停止

***启动：***

***切换至***offline***服务器***

/apps/svr/online/bin/check-server.sh # 启动server服务，主要作用是计算

/apps/svr/offline/bin/check-jobsubmitter.sh offline #启动offline服务，主要作用是提交任务

/apps/svr/offline/bin/check-jobsubmitter.sh loca #启动load服务，主要作用是提交任务

关闭：

/apps/svr/online/bin/kill-server.sh

/apps/svr/online/bin/kill-jobsubmitter.sh offline

/apps/svr/online/bin/kill-jobsubmitter.sh load

检查：

jps

### offline基本操作

offline会定时去hive中取原始数据然后在yarn上计算最终写入hbase中，hbase和spark做了集成

cd /apps/svr/phoneix/ ## 切换至phoneix安装目录，phoneix是一个集合插件

./bin/start-sql.sh ##进入hbase中

select \* from ai\_domain; ## 初次集成的时候如果后端数据上来并且kafka和hive没有问题这里就可以看到该项目的ai，然后去界面检测应该就可以检测到应用了，同时还有很多表new\_user，等等等

offline目录

cd /apps/svr/offline/ #offline安装目录

* 在安装目录下有main和load两个文件，这两个文件的主要作用是查看offline计算到那个时间点，一般offline是每一小时计算一次。

## Hadoop服务

### 介绍

服务：Hadoop（两主多从）

介绍：hadoop集群

### Hadoop的启动与停止

***启动：***

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/start-hdfs.yml

ansible-playbook -i inventories/project\_name/hosts tool-scripts/start-yarn.yml

关闭：

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/stop-hdfs.yml

ansible-playbook -i inventories/project\_name/hosts tool-scripts/ stop-yarn.yml

检查：

可以登陆hadoop服务查看服务进程监听端口，也可以访问hadoop master的50070端口检查集群是否启动，也可以检查当前存活的节点以及集群当前的情况

### Hadoop基本操作

可以参考hadoop的官方文档

yarn的介绍

在浏览器输入hadoop master的ip➕8088端口可以看到当前启动的所有任务（online+offline）根据online和offline启动的进程数不同，同时在第一行可以看到当前集群资源使用情况。

## Hbase集群

### 介绍

服务：hbase（两主多从）

介绍：hbase集群

### 服务状态

命令：

cd /opt/growing-ansible ## 在ansible服务器上运行

ansible -i inventories/project\_name/hosts hbase -m shell -a "jps"

### hbase的启动与停止

***启动：***

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/start-hbase.yml

关闭：

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/stop-hbase.yml

检查：

可以登陆hbase服务查看服务进程监听端口

### hbase基本操作

参考hbase的官方文档

## zookeeper集群

### 介绍

服务：zookeeper（一般是三台）

介绍：zookeeper集群

### 服务状态

命令：

cd /opt/growing-ansible ## 在ansible服务器上运行

ansible -i inventories/project\_name/hosts zookeeper -m shell -a "lsof –I :2181"

### zookeeper启动与停止

***启动：***

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/start-zookeeper.yml

关闭：

***切****换至ansible服务器执行*

cd /opt/growing-ansible

ansible-playbook -i inventories/project\_name/hosts tool-scripts/stop-zookeeper.yml

单机启动：

需要切换zookeeper服务器

/apps/svr/zookeeper/bin/zkServer.sh start

## postgres数据库

### 介绍

服务：postgres

介绍：存放GIO系统的数据

### 服务状态

命令：

切换至postgres服务器

ps –ef|grep postgres 查看进程是否存在

### postgres启动地址

***启动：***

***切****换至postgres服务器执行*

systemctl start postgres-9.5

*关闭：*

***切****换至postgres服务器执行*

systemctl stop postgres-9.5

### postgres基本操作

psql -U apps -d growing ## 使用apps用户进入growing的数据库

#### 在界面添加角色等按钮，修改项目为商业版

insert into contracts(id,project\_id,project\_version\_id,is\_active,created\_at,updated\_at,start\_date) values(3,5,3,'t','2017-12-12 00:00:07.767','2017-12-12 00:00:07.767','2016-10-14');

id： 设置新的id

project\_id：找出需要关联的项目（projects表里有）

project\_version\_id：找出商业版的id信息（project\_versions表里有）

is\_active：是否激活，为t

created\_at：创建时间，

updated\_at：更新时间

start\_date：开始时间

* 基本增删改查可以参考官网链接

#### 查看所有的项目

select name,keyid from projects;

#### 修改用户的密码

切换至growing\_accounts库

\c growing\_accounts

update accounts set password\_digest=’password’ where id = ‘需要修改用户名的user\_id’;

## redis服务

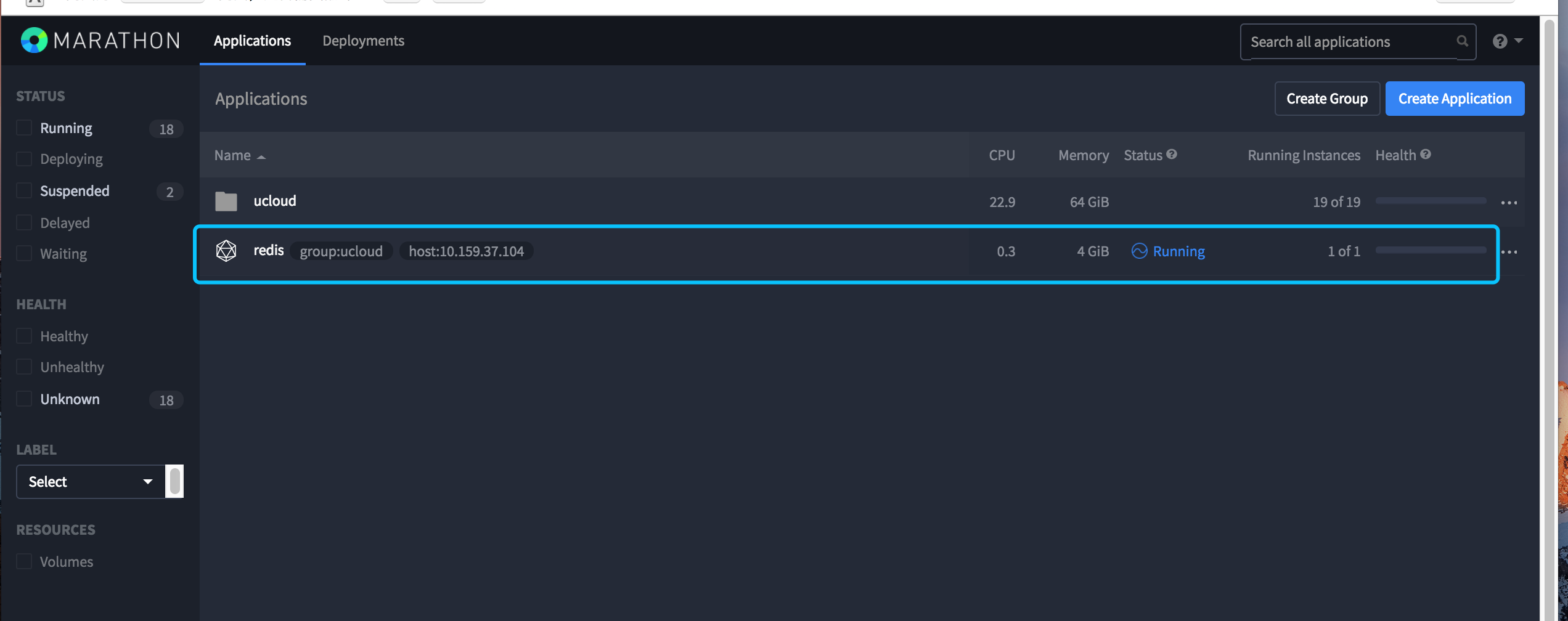
### 介绍

服务：redis

介绍：缓存服务

### 状态：

打开marathon界面就可以看到redis



status显示running就是运行状态

命令查看：

docker ps ## 可以查看当前运行的容器

docker logs --tail 100 -f redis\_id ## 查看redis的运行日志

### redis的启动与关闭

启动：

切换到/apps/svr/redis 下执行install.sh即可

cd /apps/svr/redis && sh install.sh

关闭：

参考frontend

### redis的基本操作

#### 概览数据为0

如果offline重新计算数据，数据已经计算完成（在单图中可以看到数据），可以删除redis的key清理redis的缓存

redis-cli keys "\*qa29" | xargs redis-cli del

#### 在用户管理界面缺少添加角色的按钮

pg数据库确定修改好，然后删除redis的key才可以生效

keys role\*

删除

# 常见的问题

## 界面中没有添加角色的按钮

切换至数据库服务器

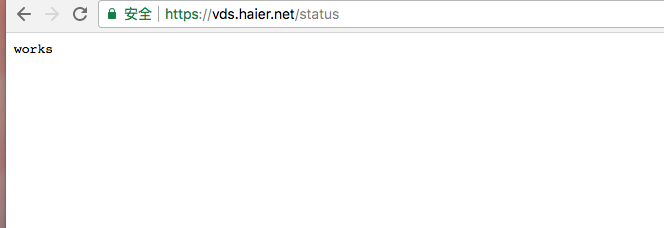
psql -U apps -d growing

insert into contracts(id,project\_id,project\_version\_id,is\_active,created\_at,updated\_at,start\_date) values(3,5,3,'t','2017-12-12 00:00:07.767','2017-12-12 00:00:07.767','2016-10-14');

* id： 设置新的id
* project\_id：找出需要关联的项目（projects表里有）
* project\_version\_id：找出商业版的id信息（project\_versions表里有）
* is\_active：是否激活，为t
* created\_at：创建时间，
* updated\_at：更新时间
* start\_date：开始时间

## 集成应用和数据流排查流程

在浏览器输入vds的域名➕status 是否返回works，如果返回即负载均衡正常



2.检查vds-api是否接受到数据

去 vds-api服务器，然后去/data/vds-api/events-log目录下查看文件是否有数据存在。

3.检查log2kafka是否有问题

去vds-api服务器，切入log2kafka的容器内，进入到logs目录下，查看sendlog是否有报错输出正常的情况应该是空的log文件，然后检查其他的日志文件是否有报错。

4.检查kafka是否获取原始数据

切换至kafka服务器，在kafka的安装目录下/apps/svr执行看是否会实时的输出数据

bin/kafka-console-consumer.sh --zookeeper zookeeper:2181/kafka --from-beginning --topic topic ## topic根据不同应用进行选择

1. 检查hive是否有原始数据

切换online服务器，切换至spark的安装目录/apps/svr/spark下，然后进入spark中，查看page和visit以及其他表是否有数据存在。

cd /apps/svr/spark/bin

./spark-sql --master local[4]

use gio;

select \* from page where time like ‘20180324%’; #查看page、visit表,由于数据量比较大，最好使用count统计下

1. 检查hbase是否有数据

去offline服务器，切换到phoenix安装目录下，然后进入phoenix中，查看ai\_domain表是否有集成的那个应用的ai

cd /apps/svr/phoenix/bin

./start-sql.sh

select \* from ai\_domain; #这个表记录的每个项目的ai

8．去页面检查

在页面进行检查应该就可以检查到应用了。

9．前提要检查所有服务的状态是否全部启动，是否启动有问题

### online追数

由于磁盘空间满了或者其他原因，导致online没有将原始数据读区至hive中，offline 没有统计出数。

操作：

1. 登陆online服务器

cd /apps/svr/spark/bin

./spark-sql --master local[4]

2、 切换到gio数据库。

use gio; #切换至gio数据库

3、查看online原始数据，正常是每半小时会有一条记录

show partitions action;

* 看显示的时间，如果缺少一段时间的数据，也就是online没有读取那段时间的原始数据，如缺少201803241600 – 201803242400，输出没有这个时间段。

4、停止online和offline服务

详情停止参考online和offline的服务停止命令（注意定时任务，先注释，数据追上以后删除注释）

5、修改online的配置文件

cd /apps/svr/online/conf

vim common.conf

修改online读区数据配置kafka.reset,重启online.开始追原始数据.(earliest/latest)

# app.after: 1487203200000 从指定的时间节点读取kafka数据

action {

expire.hour: 24

} 这个配置，读取数据截止小时，即只读取最近24小时

追数完成后修改回默认配置，重启online

6、确定元数据是否正常获取

重复1、2、3操作查看是否有缺失时间短的数据

7、元数据已经OK，需要配置跑offline jobs 。

登陆offline服务器

cd /apps/svr/offline

8、修改时间

将时间修改至缺失数据前的时间点然后修改MainFrame.pos 和 Load.pos文件，重启offline服务。

9、等待验证数据

### 查看offline失败的任务

数据库： postgres

登陆方式：psql -U apps -d rules

查看rules的job\_todo表

统计当前所有任务的状态

select clz, count(clz) from job\_todo group by clz;

### 容器服务升级

### 升级的大概步骤

1. gio提供升级包的镜像
2. 将gio提供的景象load到需要升级服务的docker镜像中。
3. 打开marathon界面修改镜像名称的版本即可

导入docker images中

docker load -i package.tar.gz

查看镜像

docker ps

### 非容器的升级

### 升级的大概步骤

1. gio提供需要升级的jar包或者服务软件包
2. 将服务软件包拷贝至服务器
3. 替换之前的服务或者jar包
4. 重新启动