**公司内网docker环境搭建**

安装使用说明书

V2.0

|  |  |  |  |
| --- | --- | --- | --- |
| 编写 |  | 编写时间 |  |
| 审批 |  | 审批时间 |  |
| 文档版本 |  | | |

# 前置操作

物理硬件设备要求：

* 服务器至少四台：每台配置如下

操作系统：centos7.3 64位(不能用32位)

系统配置：

cpu数量：16核

内存：128G

硬盘大小：2T

在安装docker和rancher之前，先执行以下操作，确保安装过程顺利进行。

* 确保Linux内核必须满足64位3.10以上
* 停止firewall

systemctl stop firewalld.service

* 禁止firewall开机启动

systemctl disable firewalld.service

* 设置aliyun的yum源

sudo wget -O /etc/yum.repos.d/CentOS-Base.repo <http://mirrors.aliyun.com/repo/Centos-7.repo>

* 添加EPEL源

sudo wget -P /etc/yum.repos.d/ <http://mirrors.aliyun.com/repo/epel-7.repo>

* 清理缓存并生成新的缓存

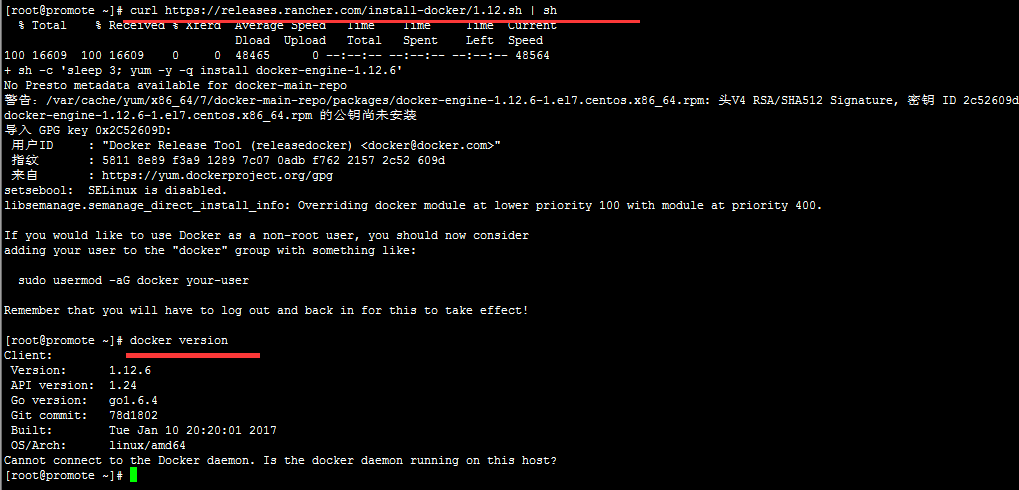
sudo yum clean all

sudo yum makecache

# 第2章 安装docker

**#step1**

curl https://releases.rancher.com/install-docker/1.12.sh | sh



**#step2**

sudo mkdir -p /etc/docker

**#step3 配置docker仓库地址**

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

{

"insecure-registries":["192.168.1.129:5000","docker.gtintel.cn:5000"],

"hosts": [

"unix:///var/run/docker.sock"

]

}

EOF

**#step4**

systemctl daemon-reload

**#step5**

systemctl enable docker

**#step6 重启docker**

systemctl restart docker

**#编辑docker.service文件**

vim /etc/systemd/system/multi-user.target.wants/docker.service

将ExecStart值修改为以下内容：

ExecStart=/usr/bin/dockerd --registry-mirror=https://p3g80nu4.mirror.aliyuncs.com

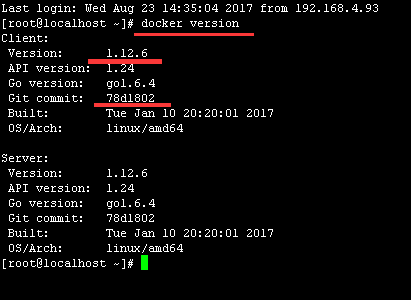
**#重启docker**

systemctl daemon-reload

systemctl restart docker

**#验证docker**

docker version



# 第3章 安装rancher

Rancher-server部署方式采用单节点，暂不支持集群部署。

Docker安装前一定要确保firewalld和SELinux服务关闭。

## 3.1 安装mysql数据库

**#step1 mysql官网下载最新的安装源包进行安装**

wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm

**#step2 查看安装包名称**

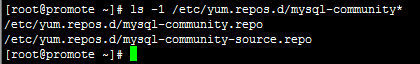
ls

**#step3 执行安装**

rpm -ivh mysql-community-release-el7-5.noarch.rpm

**#step4检查是否安装成功，如下图安装成功：**

ls -1 /etc/yum.repos.d/mysql-community\*



**#step5**

yum install mysql-server

**#step6 设置自启动**

service mysqld start

**#step7启动mysql**

vi /etc/rc.d/rc.local 把mysql加入开机自启动

service mysqld start

**#step8 修改mysql配置文件/etc/my.cnf**

文件最后加上编码配置：

[mysql]

default-character-set =utf8

**#step9 修改root密码（gt86589089），创建rancher数据库（cattle）**

#连接数据库

mysql -u root

#执行脚本

set password for 'root'@'localhost' =password('gt86589089');

CREATE DATABASE IF NOT EXISTS cattle COLLATE = 'utf8\_general\_ci' CHARACTER SET = 'utf8';

GRANT ALL ON cattle.\* TO 'cattle'@'%' IDENTIFIED BY 'cattle';

set password for 'cattle'@'%' =password('gt86589089');

FLUSH PRIVILEGES;

## 3.2 拉取rancher

#拉取rancher，指定版本为1.6.5。

docker pull rancher/server:v1.6.5

## 3.2启动rancher-server服务

**#step1 需指定rancher-server主机、端口、数据库账号和密码**

sudo docker run -d --name rancher-server --restart=unless-stopped -p 8080:8080 rancher/server:v1.6.5 --db-host 192.168.7.10 --db-port 3306 --db-user cattle --db-pass gt86589089 --db-name cattle --advertise-address 192.168.7.10:8080

参数说明：

--db-host 指定MySQL服务器的连接地址

--db-port 连接端口

--db-user 连接用户

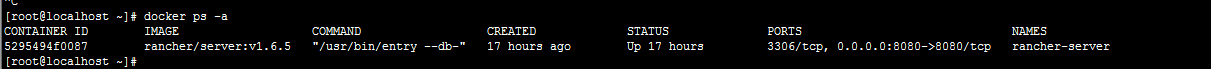
--db-pass 连接密码

--db-name 连接库名

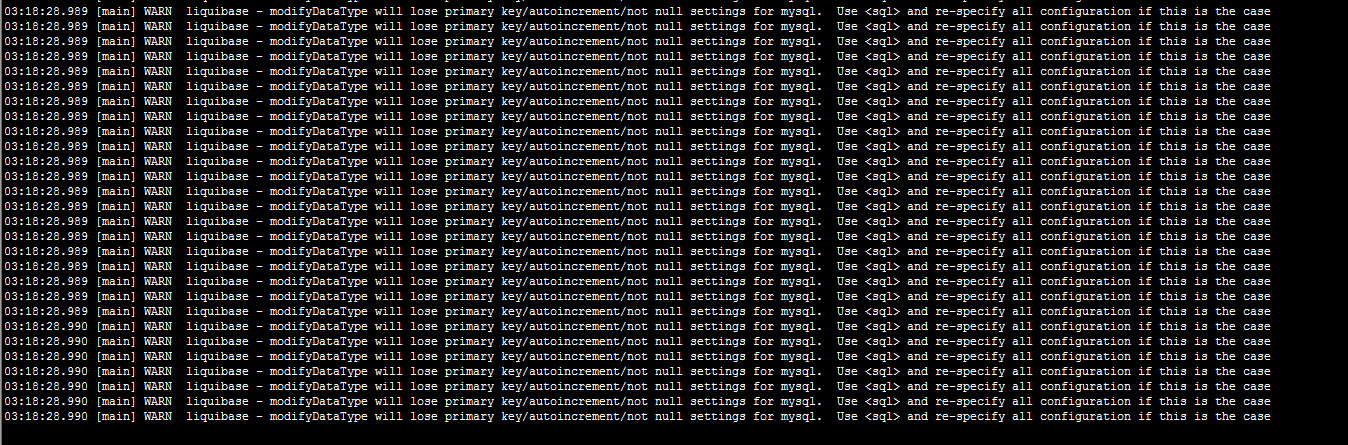
**#step2 访问http:// 192.168.7.10:8080地址**

如果出现rancher登录页面，则说明安装成功。

否则，需要重启docker（systemctl restart docker）再试。



如果是mysql安装的，第一次启动后一直无法正常进入系统，等待10分钟左右之后正常。

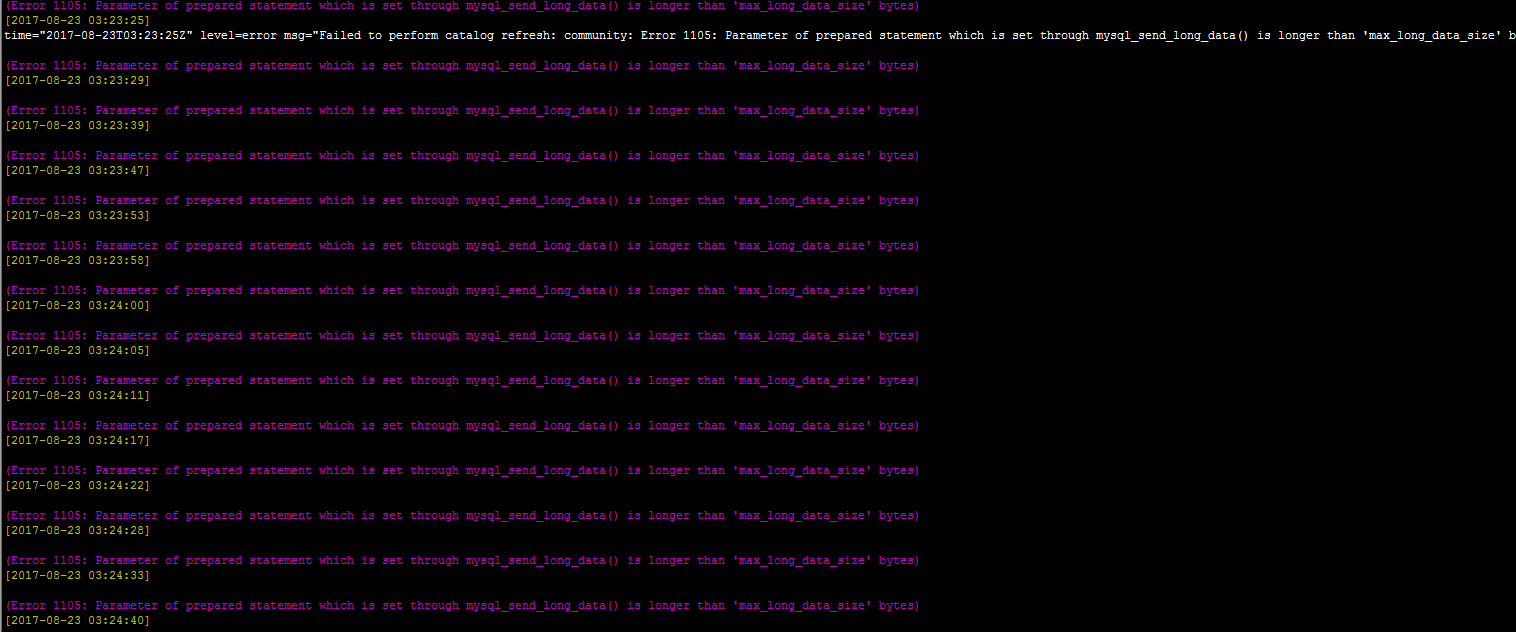


可以通过docker logs –f rancher-server来查看启动情况。

初次启动rancher-server服务的后台日志如下：



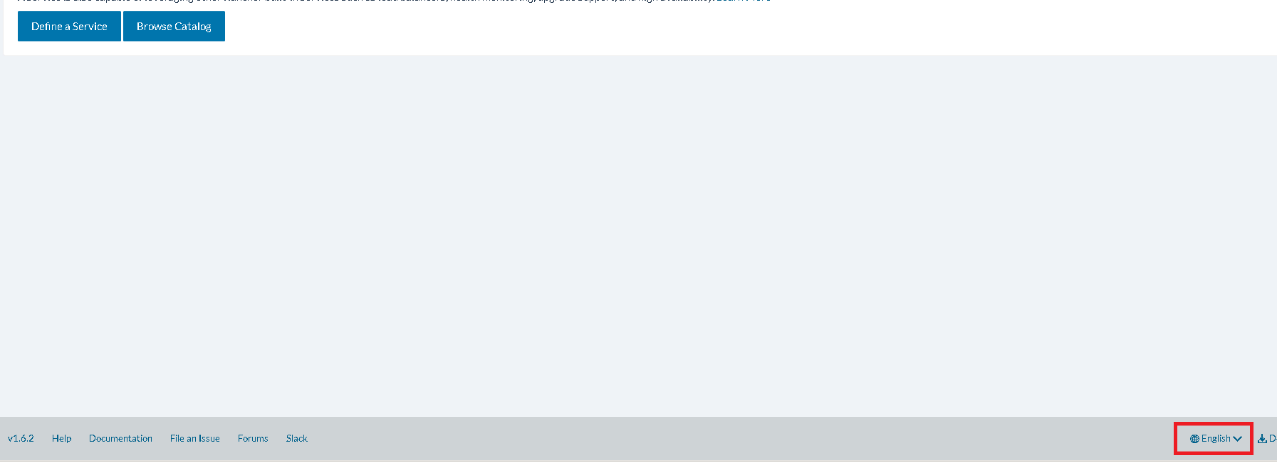
如果是mariadb安装，启动rancher后会报字段错误。



# 第4章 Rancher配置操作

## 4.1 将英文界面转换为中文界面

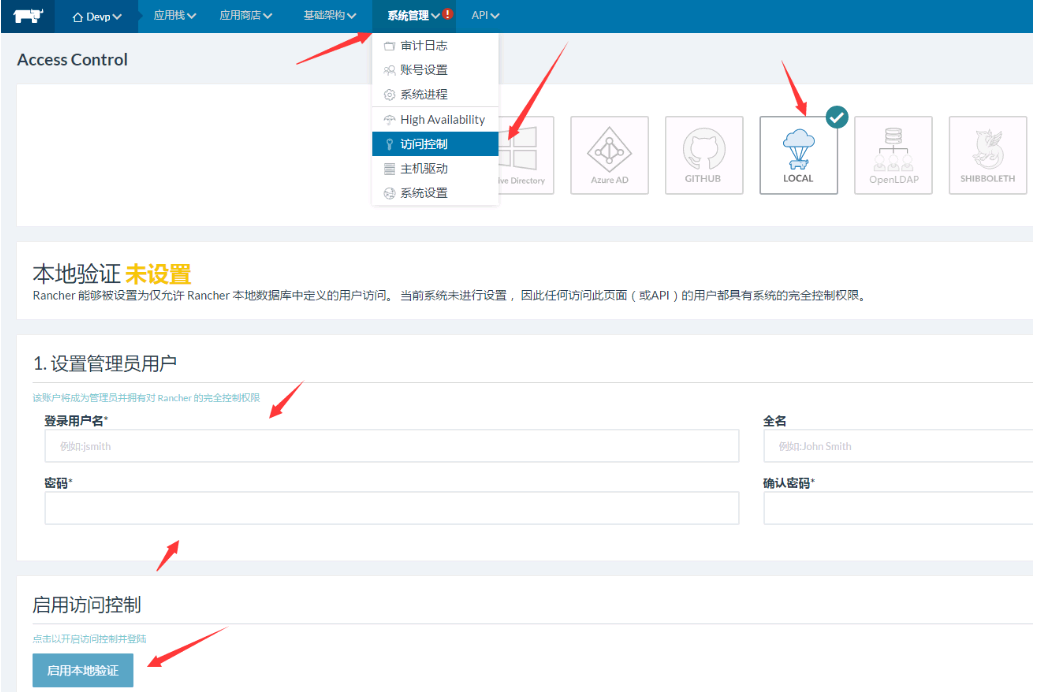
点击页面右下角的“English”图标，选择“简体中文”



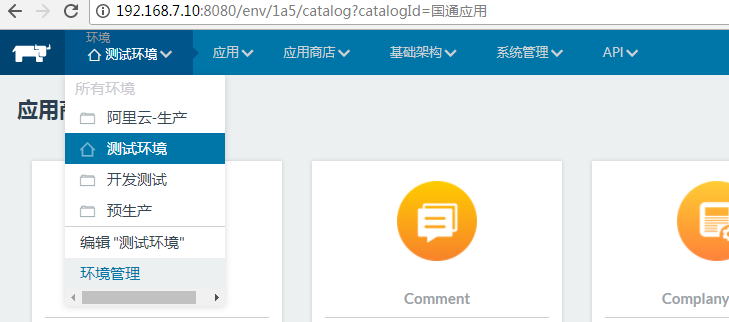
## 4.2 设置访问控制

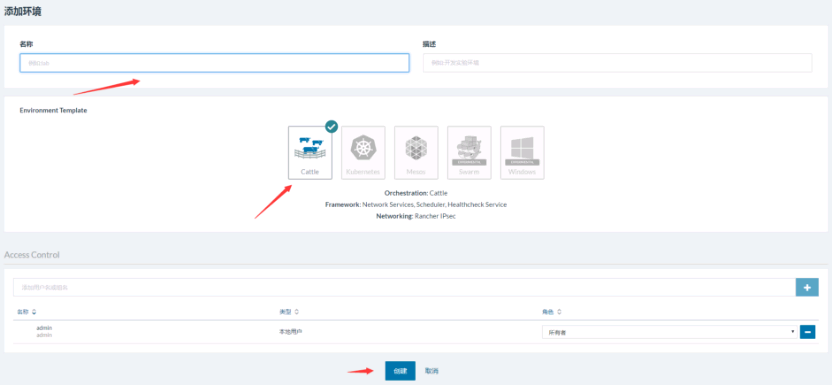
默认rancher是没有启用访问控制的，需要手工启用。

点击【系统管理】——【访问控制】——“LOCAL”图标，输入用户名和密码，最后点击“启用本地验证”按钮。



## 4.3 环境管理





根据业务需求，创建环境名称。

## 4.4 添加应用商店



以公司内网测试环境为例，

基础设施：<http://218.2.224.34:3000/root/GT_Infra.git>

国通应用：<http://192.168.1.34:3000/root/GT_App_Test.git>

## 4.5 添加主机





复制第5步中的脚本到Rancher的节点Docker中运行（rancher-agent进程）。如下：

sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.5 http://218.2.224.34:4445/v1/scripts/EF5F86C9EC7372A0F464:1483142400000:uBzypBNc4SF2mSaiqd7CPbCJLek

添加完成之后主机就会出现在主机列表中，并且安装一些初始化的docker。

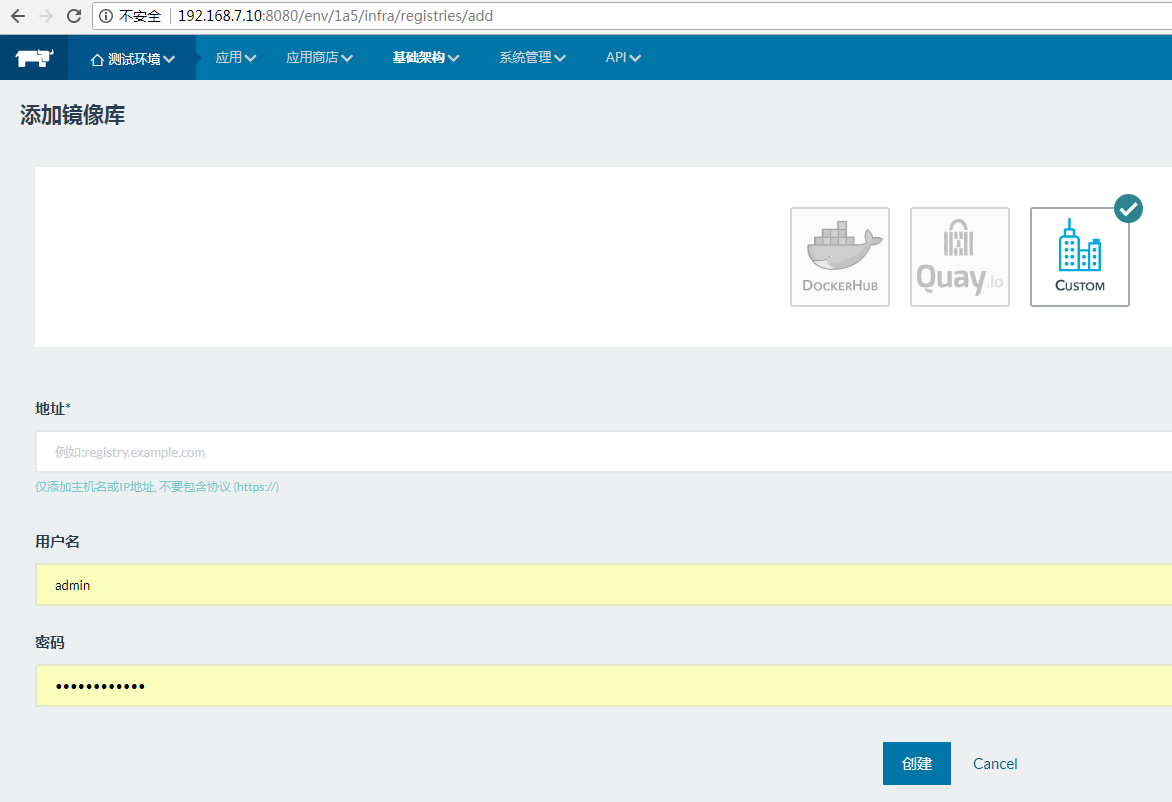
非常重要：如果rancher-server和agent装在同一台机器，则需要带参数-e CATTLE\_AGENT\_IP="192.168.7.10"

sudo docker run --rm --privileged –e CATTLE\_AGENT\_IP=”192.168.7.10” -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.5 http://218.2.224.34:4445/v1/scripts/EF5F86C9EC7372A0F464:1483142400000:uBzypBNc4SF2mSaiqd7CPbCJLek

## 4.6 添加镜像库



点击“添加镜像库”按钮，勾选“Custom”图标，输入镜像库地址后，点击“创建”按钮。



添加如下镜像库：

192.168.1.129:5000

用户名和密码填写仓库访问的账号和密码，例如：admin/admin123

# 第5章 应用商店安装镜像

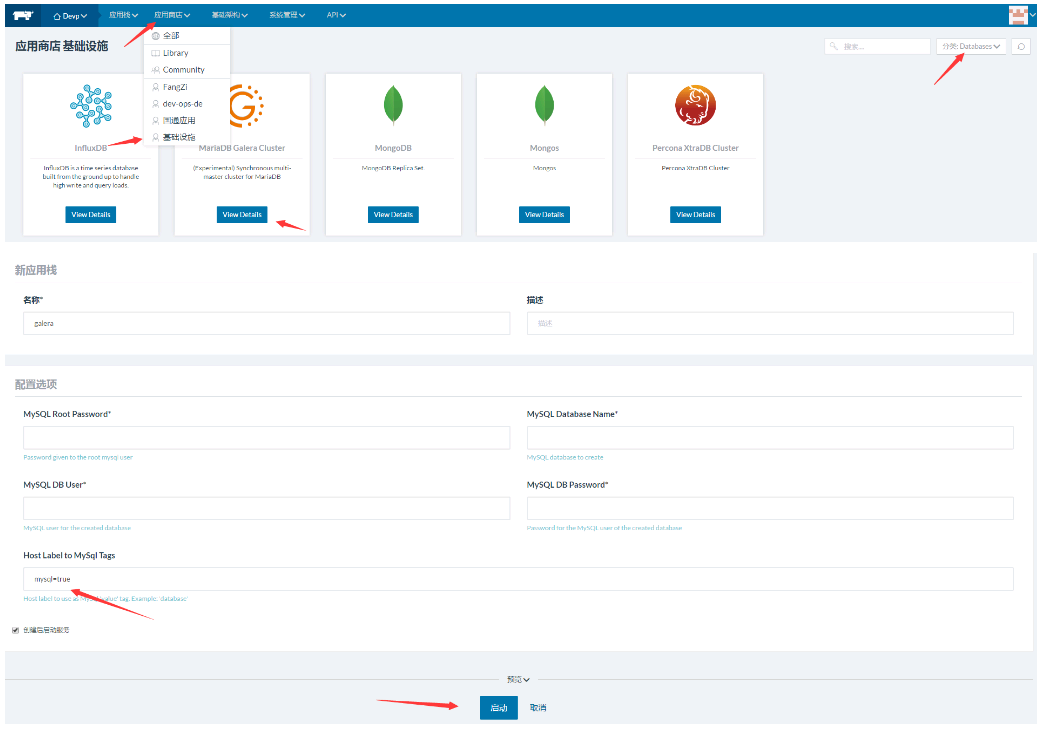
Rancher应用商店操作分为基础设施和国通应用两个方面。

基础设施包括zookeeper、kafka、mysql、monogodb等基础库。

国通应用包括基础的业务单元，如首页、黄页、名片、通知等。

## 5.1 基础设施

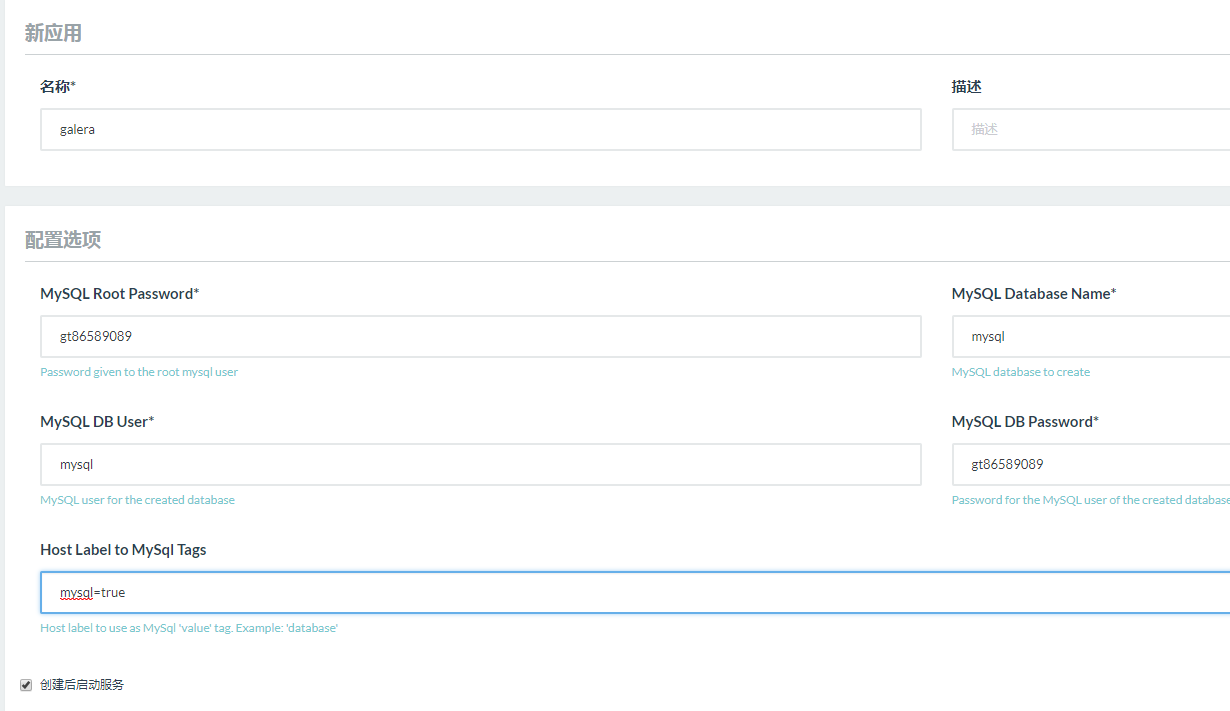
### 5.1.1 mysql集群（galera）



根据要求填写参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 值 | 说明 |
| MySQL Root Password | gt86589089 | 程序内部指定了该密码，所以必须填写。 |
| MySQL Database Name | mysql | 随便写 |
| MySQL DB User | mysql | 随便写 |
| MySQL Password | mysql | 随便写 |
| Host Label to MySql Tags | mysql=true |  |

注意Host Label为调度mysql最终运行在哪一台服务器上的标签。需要在Host中为主机添加标签。

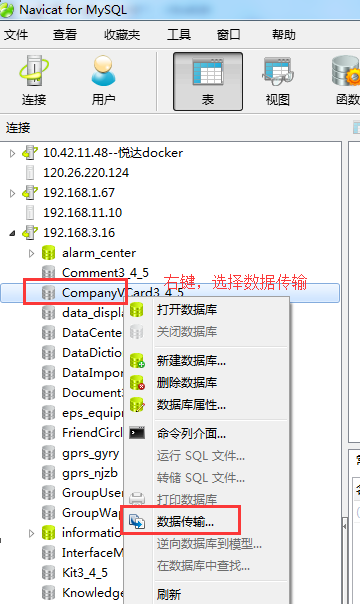


点击“添加标签”按钮，填写键和值。添加完成后，在主机页面就能看到应用的运行信息。

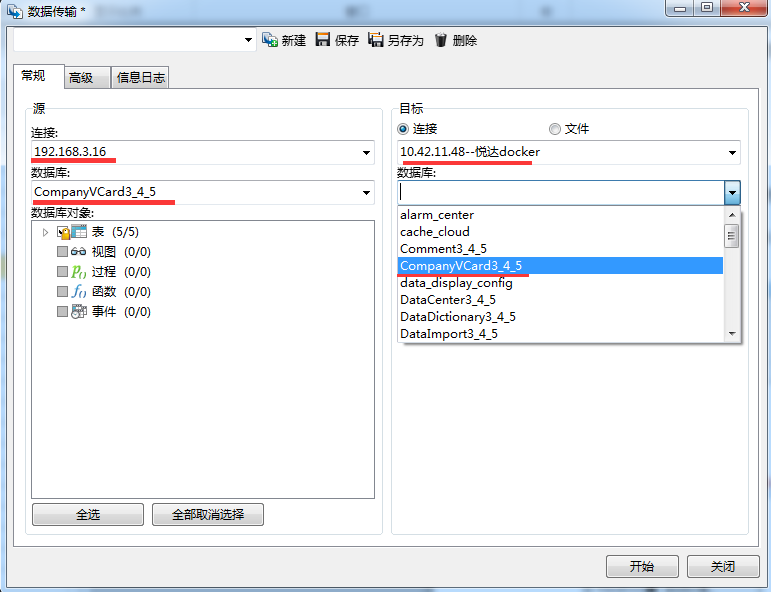
mysql数据库的数据导入：（前提是连接上openvpn，vpn账号密码为rancher管理平台账号密码。）

从预生产环境把mysql（192.168.3.16）里面的数据库备份出来，进行如下操作：

导出导入方式如下，



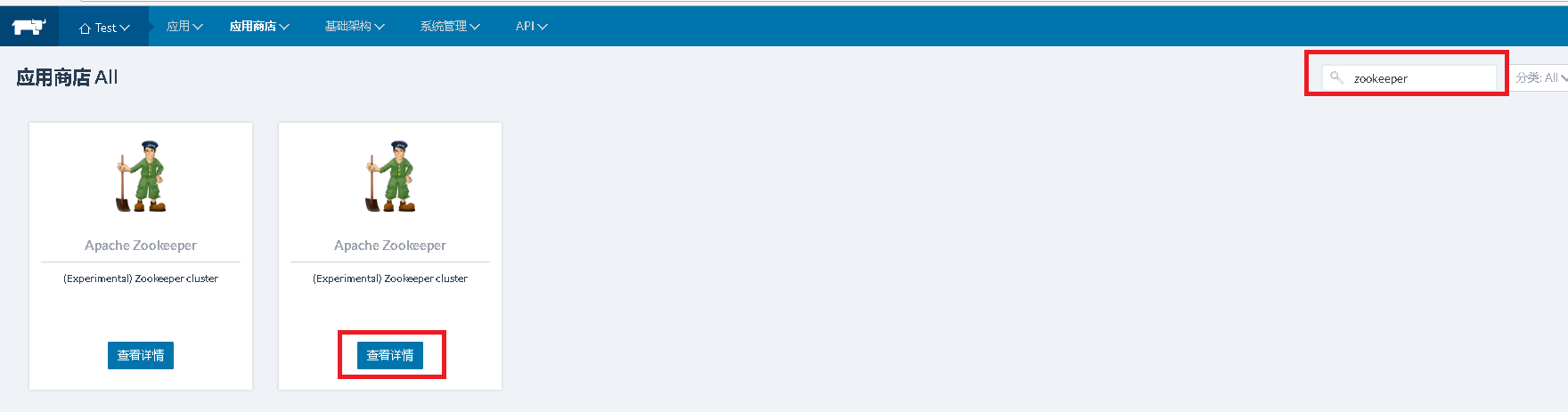
选择数据源及需要导入的新数据库信息，如下截图



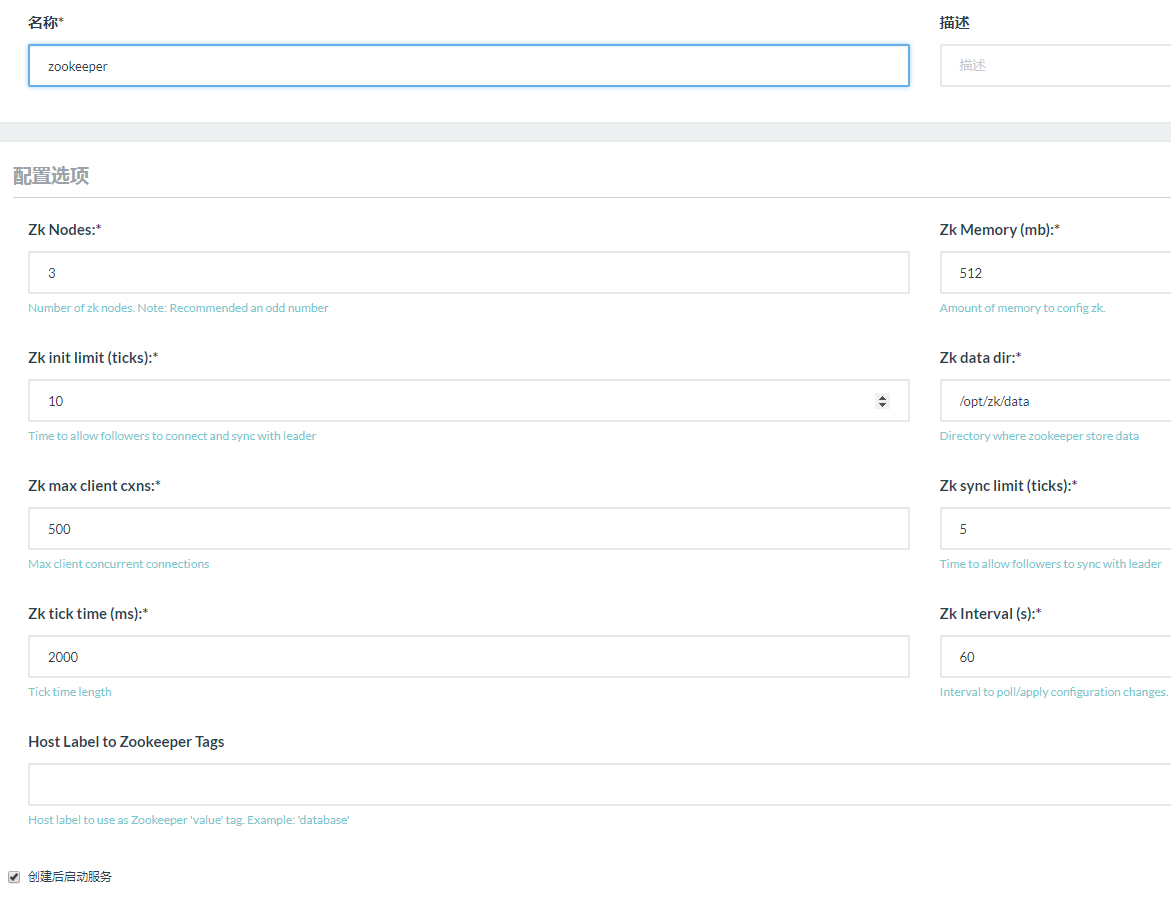
勾选完成后点击开始即开始数据传输。

### 5.1.2 zookeeper

应用商店——基础设施，直接搜索“zookeeper”，点击“查看详情”

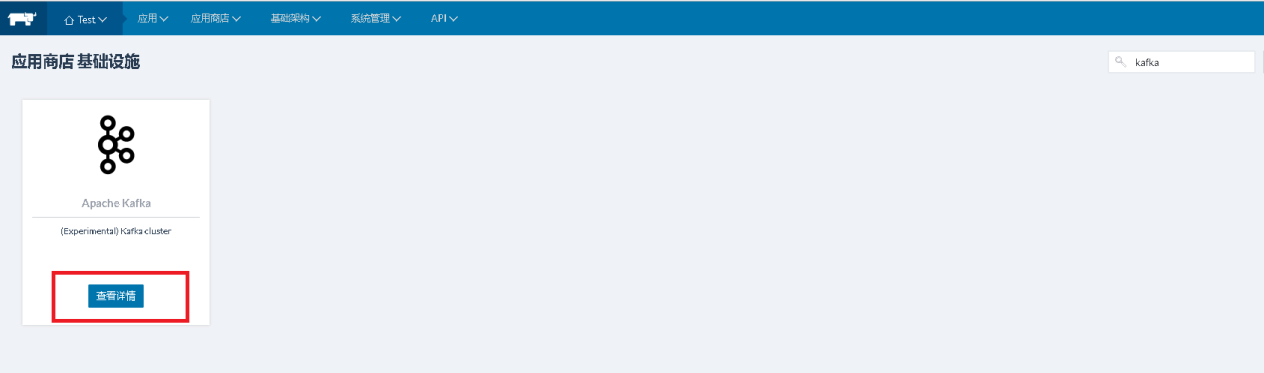


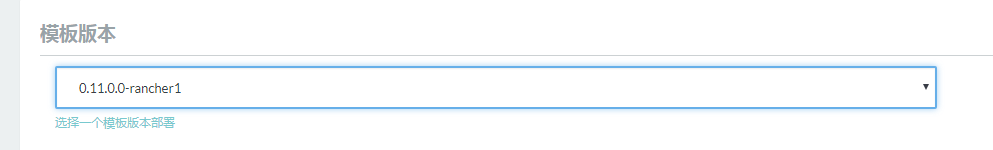
填写host label to zookeeper tags值，点击“启动”。



### 5.1.3 kafka

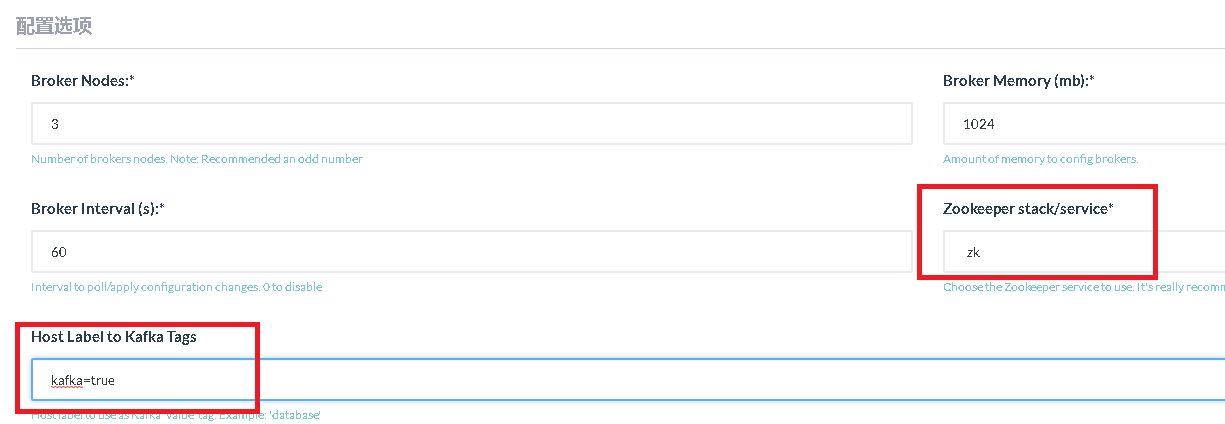
应用商店——社区共享，直接搜索“kafka”，点击“查看详情”





**Zookeeper stack/service** 选择“zk”

**Host label to kafka Tags** 填写kafka=true

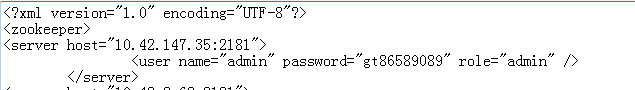


### 5.1.4 zookeeper配置

**（需要先安装VPN，跳转至VPN部署篇）**

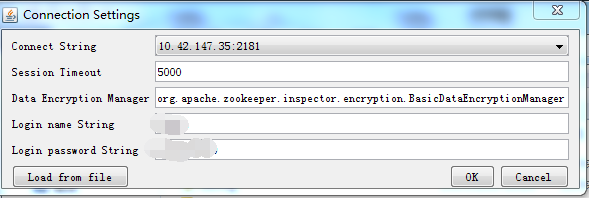
使用ZooInspector连接工具配置zookeeper。

#### 5.1.4.1 编辑ZooInspector的config.xml文件，添加host和用户账号。

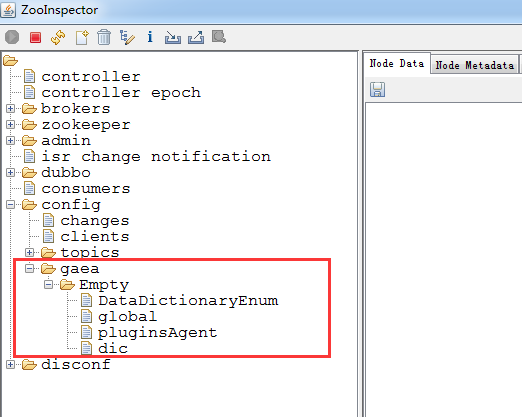


#### 5.1.4.2 运行zookeeper-dev-ZooInspector.jar工具

填写login name string和login password string



#### 5.1.4.3 在config目录下建造以下文件夹结构，如下图所示：



##### 一、Empty目录

###### 1、DataDictionaryEnum

{

GlobalDataDictionaryEnum: { Desc: "", Val: "{Apps:{Id:\"444BB40A-8D68-44E9-AB9C-F1F7546A3409\",ParentId:\"99999999-9999-9999-9999-999999999999\"},DataTypes:{Id:\"BDC0E49C-D73E-C21C-A7AE-08D291788182\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Tags:{Id:\"D49276A3-4885-C5DB-643B-08D2B800D641\",ParentId:\"99999999-9999-9999-9999-999999999999\"},KnowledgeTag:{Id:\"27488027-0EEC-4DD4-B5F3-1CE2D9F1AABF\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Plugins:{Id:\"5C53D9AF-9705-C47E-334D-08D2C3BA20E8\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Projects:{Id:\"1e2d3107-2964-c7b1-60f1-08d2718044cb\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Pages:{Id:\"174b2982-5b27-c33e-8d67-08d271804afc\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Design:{Id:\"5d340137-aa63-ced6-14eb-08d31a2b9a8b\",ParentId:\"99999999-9999-9999-9999-999999999999\"},Buttons:{Id:\"eca43a81-5430-cb3c-4d01-08d271805395\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}"},

ServiceDataDictionaryEnum:{ Desc: "", Val: "{AllServices:{Id:\"0ffe4550-6186-c8c8-e238-08d28849fb99\",ParentId:\"99999999-9999-9999-9999-999999999999\"},AllServers:{Id:\"0859f95a-8711-ca5b-cb4d-08d288618c0e\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}" },

TagDataDictionaryEnum:{ Desc: "", Val: "{SystemTag:{Id:\"0576d7e6-32ad-cd92-bb8f-08d28b625c54\",ParentId:\"99999999-9999-9999-9999-999999999999\"},CustomerTag:{Id:\"02be42cd-79ba-c6fd-5bfa-08d28b625c53\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}" },

MoodTagDictionaryEnum:{ Desc: "", Val: "{MoodTag:{Id:\"22cc5075-1947-c8bb-fc9f-08d2917cacb3\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}" },

GroupUserDictionaryEnum:{Desc: "", Val: "{UserExColumn:{Id:\"253fbedd-b2c2-c69a-f407-08d2a877e255\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}" },

SystemDataDictionaryEnum:{Desc: "", Val: "{SysGroupTag:{Id:\"93237738-4FC4-C994-56E8-08D2BCB55BC7\",ParentId:\"99999999-9999-9999-9999-999999999999\"}}"},

PluginsEnum:{Val:"{Empty:{Id:\"00000000-0000-0000-0000-000000000000\"},DataDictionaryWrapper:{Id:\"531cbf49-af43-c2a9-f647-08d35a1198e7\"},Notice:{Id:\"d5f16378-4910-c093-20ee-08d338045ed5\"}}"}

}

###### 2、global

{

CacheAddresses: {

Desc: "",

Val: "[\"192.168.1.224:6380\",\"192.168.1.71:6380\",\"192.168.1.81:6380\",\"192.168.1.103:6379\",\"192.168.1.104:6379\",\"192.168.1.105:6379\"]"

},

QuartzIp: {

Desc: "",

Val: "ww"

},

TimeOut: {

Desc: "",

Val: "\"00:15:00\"" },

IsDebug:{Desc:"",Val:true},

IsMySql:{Desc:"",Val:true},

PortStart:{Desc:"",Val:90},

TokenCount:{Val:5},

TokenTimeOut:{Val:30},

GateIp:{Desc: "", Val: "http://192.168.4.177" },

AliId:{Desc:"",Val:"gin6ZIIh3NaJcvFf"},

AliKey:{Desc:"",Val:"BKY98SroF6UrPM8fnD3VOk4vNpCF8m"},

AliNewsAddress:{Desc:"",Val:"http://news.gtintel.cn/"},

WaitCount:{Desc:"",Val:3},

ExceptionsAllowed:{Desc:"",Val:60},

DurationOfBreak:{Desc:"",Val:60},

CpuWarnPercent:{Desc:"",Val:89},

CpuBreakPercent:{Desc:"",Val:99},

MemWarnPercent:{Desc:"",Val:80},

MemBreakPercent:{Desc:"",Val:90},

EventStore:{Desc:"",Val:"tcp://admin:admin123@192.168.1.201:1113"},

DataCenterAgent: { Desc:"",Val:"{ContentType:\"application/json\",ServiceUrl:\"http://192.168.1.201:9045/gaeaapi/\"}"}

}

###### 3、PluginsAgent

{

"DataDictionaryAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://datadictionary-lb.datadictionary:9001/gaeaapi/\",RpcUrl:\"127.0.0.1:7001\"}"

},

"GroupUserAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://groupuser-lb.groupuser:9000/gaeaapi/\"}"

},

"InterfaceAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"VisitCardAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"SmsAgent": {

"Desc": "",

"Val": "{ContentType:\"text/plain\",ServiceUrl:\"http://112.74.76.186:8030/service/httpService/httpInterface.do?method=sendMsg\"}"

},

"LoginAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://login-lb.login:9021/gaeaapi/\"}"

},

"TemplateAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://192.168.1.132:8028/gaeaapi/\"}"

},

"ValidationAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://groupuserwrapper-lb.groupuserwrapper:9030/gaeaapi/\"}"

},

"PermissionAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://permission-lb.permission:9023/gaeaapi/\"}"

},

"KitAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"NoticeAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://noteice-lb.noteice:9010/gaeaapi/\"}"

},

"NewsAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://news-lb.news:9009/gaeaapi/\"}"

},

"GroupUserWrapperAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://groupuserwrapper-lb.groupuserwrapper:9030/gaeaapi/\"}"

},

"ReportServiceAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://192.168.1.42:8024/gaeaapi/\"}"

},

"UiConfigServiceAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"KnowledgeAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://knowledge-lb.knowledge:9011/gaeaapi/\"}"

},

"OperationAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://operation-lb.operation:9032/gaeaapi/\"}"

},

"PermissionWrapperAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://permission-lb.permission:8033/gaeaapi/\"}"

},

"PushAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"CommentAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://comment-lb.commnet:9041/gaeaapi/\"}"

},

"InterfaceManageAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"DataDictionaryWrapperAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://datadictionarywrapper-lb.datadictionarywrapper:9035/gaeaapi/\"}"

},

"UiConfigAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"TimeServiceAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://timeservice-lb.timeservice:9043/gaeaapi/\"}"

},

"PositionAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://position-lb.position:9037/gaeaapi/\"}"

},

"DataCenterAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://datacenter-lb.datacenter:9045/gaeaapi/\"}"

},

"ContactAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"NoFactoryAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://nofactory-lb.nofactory:9051/gaeaapi/\"}"

},

"ComplanyVCardAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://complanyvcard-lb.complanyvcard:9027/gaeaapi/\"}"

},

"FormAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://formdesign-lb.formdesign:9049/gaeaapi/\"}"

},

"FlowAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"\"}"

},

"AlarmAgent": {

"Desc": "",

"Val": "{ContentType:\"application/json\",ServiceUrl:\"http://alarmcenter-lb.alarmcenter:9082/gaeaapi/\"}"

}

}

//该处ip地址和端口需要修改为实际服务的IP地址和端口

###### 4、dic

{

"Plugins": {

"Val": "{Id:\"0F8CFDAB-032C-CA3D-9005-08D40C42A48C\",ParentId:\"99999999-9999-9999-9999-999999999999\"}"

},

"App\_BFXM": {

"Val": "{Id:\"3fdeb5c2-15a7-ce68-b5d8-08d3fc967fa2\"}"

},

"Platform": {

"Val": "{Id:\"31fb2261-8835-cf98-fbce-08d418266a35\"}"

},

"Front": {

"Val": "{Id:\"28522671-281f-c79d-842b-08d3ffc0b04c\",ParentId:\"e1cc3e23-fe07-c6b5-8ead-08d3ffc08b27\"}"

},

"Back": {

"Val": "{Id:\"1164af9b-657b-cd9f-7445-08d3ffc0befc\",ParentId:\"e1cc3e23-fe07-c6b5-8ead-08d3ffc08b27\"}"

},

"OperateType": {

"Val": "{Id:\"1cf4977b-c264-c2dc-6a92-08d4182677fe\"}"

},

"MenuType": {

"Val": "{Id:\"907a74a9-9d4f-c06b-f268-08d4182684ba\"}"

},

"TargetType": {

"Val": "{Id:\"704b77ea-c53a-c5a5-bad9-08d418268e91\"}"

},

"RuleType": {

"Val": "{Id:\"2a0fc895-1916-c306-a6dc-08d418269c7c\"}"

},

"RuleRange": {

"Val": "{Id:\"d94c82c3-c14e-c164-5dd4-08d41826a652\"}"

},

"DeviceType": {

"Val": "{Id:\"abbc51d8-5f1d-cb2c-fb48-08d4173801f7\"}"

},

"ViewOperation": {

"Val": "{Id:\"57a78cff-b5f4-c8a8-6839-08d4173818a3\"}"

},

"Group": {

"Val": "{Id:\"CE443E66-389A-C3CC-5C94-08D401331553\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"User": {

"Val": "{Id:\"0A905274-CE85-CC09-E28A-08D401331D66\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"Text": {

"Val": "{Id:\"4FD5EC06-A05F-C154-36AD-08D401332BDD\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"Number": {

"Val": "{Id:\"B04AE58B-FB21-C75A-CE14-08D4013334B8\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"DateTime": {

"Val": "{Id:\"646B2238-2015-C4C1-0966-08D401334045\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"Role": {

"Val": "{Id:\"995BAFD5-5A24-CBCB-FC0B-08D4013347CE\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"DataSource": {

"Val": "{Id:\"0FA7BED5-916C-C076-FD96-08D402C5A65E\",ParentId:\"DBBA8537-D788-C5C9-56C3-08D401330233\"}"

},

"Self": {

"Val": "{Id:\"39DB5050-1F96-C16C-782F-08D401336FFE\",ParentId:\"309CEAE1-EF05-CA68-64B0-08D401335DDB\"}"

},

"Range": {

"Val": "{Id:\"4B735878-06F4-C7AB-443B-08D401337F7D\",ParentId:\"309CEAE1-EF05-CA68-64B0-08D401335DDB\"}"

},

"StandardMenu": {

"Val": "{Id:\"f623b738-26e0-c4b1-71b7-08d401328f55\",ParentId:\"c3467e78-079f-c9aa-d880-08d401326517\"}"

},

"CustomMenu": {

"Val": "{Id:\"b8e6619a-8157-c43a-c5d8-08d40132768f\",ParentId:\"c3467e78-079f-c9aa-d880-08d401326517\"}"

},

"NoticeUseType": {

"Val": "{Id:\"aef77d38-7a7e-cdc9-3cb3-08d3fd46d9a0\"}"

},

"NoticeLevel": {

"Val": "{Id:\"a0a43592-c0a5-c6c5-0762-08d4175db6ae\"}"

},

"ChinaRegions": {

"Val": "{Id:\"a6bf9322-e09b-c3b7-3b22-08d4225e4dfb\"}"

},

"EnterpriseScale": {

"Val": "{Id:\"cbcf524f-2842-cb6a-497d-08d422668564\"}"

},

"EnterpriseType": {

"Val": "{Id:\"5bd85afa-e530-cfb6-9d53-08d42268d221\"}"

},

"Industries": {

"Val": "{Id:\"ef5d015b-ca39-c1b2-c87b-08d4226b0088\"}"

},

"EmailType": {

"Val": "{Id:\"66ee6242-1583-ca73-b79b-08d433c4e130\"}"

},

"TelephoneType": {

"Val": "{Id:\"957d2527-270b-c044-e20d-08d433c3c38d\"}"

},

"JobTitle": {

"Val": "{Id:\"f24b63d4-62ed-c5cd-cc63-08d4347154eb\"}"

},

"CompanyName": {

"Val": "{Id:\"5d5b8b36-aa70-c952-a408-08d4347a6249\"}"

},

"DepartmentName": {

"Val": "{Id:\"badbafb9-1eaa-c385-41b9-08d4347c322f\"}"

},

"PositionCategory": {

"Val": "{Id:\"20136370-4eed-cce0-829c-08d439f20257\"}"

},

"AreaCategory": {

"Val": "{Id:\"b669d4f4-954b-ca9d-c582-08d439fa3250\"}"

},

"TimeZone": {

"Val": "{Id:\"7c84114a-6f5c-c2b0-a0e9-08d43a981e71\"}"

},

"TimeSystem": {

"Val": "{Id:\"70c8efb4-2080-c3d7-5aa3-08d43a982e9c\"}"

},

"TimeFormat": {

"Val": "{Id:\"03d74e2d-b822-c5db-931f-08d43a9aeef8\"}"

},

"KnowledgeItem": {

"Val": "{Id:\"4ba28af2-6b5e-c75e-395f-08d43e84a7e5\"}"

},

"KnowledgeSysItem": {

"Val": "{Id:\"09368d67-933f-cfce-2008-08d43e8492e1\"}"

},

"NewsCategory": {

"Val": "{Id:\"5e91df2a-78a7-c8a6-5a15-08d4338bb126\"}"

},

"CodeFreType": {

"Val": "{Id:\"7a333400-435b-ce9b-3710-08d4517e9735\"}"

},

"CodePartition": {

"Val": "{Id:\"a9b6f328-824b-ce20-999b-08d4518292dd\"}"

},

"CodeChangeType": {

"Val": "{Id:\"cf79b90c-a6c1-c3f1-cc82-08d4517d5530\"}"

},

"CodeDateFormat": {

"Val": "{Id:\"f1b8e102-7bcf-cc4d-0dc3-08d45180a91a\"}"

},

"CodeDeviceType": {

"Val": "{Id:\"872b18fb-1753-c959-1310-08d45f0bd4d5\"}"

},

"CodeSiteType": {

"Val": "{Id:\"3f21a8a4-11f3-c9d3-78cd-08d45f0d6065\"}"

},

"RelationClassify": {

"Val": "{Id:\"5b7fe3b0-8012-ca8d-d6f4-08d45a353ff3\",ParentId:\"4669bf01-2748-c0f2-32ca-08d45a3514c6\"}"

},

"GroupClassify": {

"Val": "{Id:\"3e614f15-0a60-ca11-9fe0-08d45badca24\"}"

},

"TimeModeType": {

"Val": "{Id:\"869eef28-611f-c0a0-ab5f-08d471a61566\"}"

},

"DayModel": {

"Val": "{Id:\"89ac8de5-09e4-ca51-e9a0-08d471a6374b\"}"

},

"PeriodModel": {

"Val": "{Id:\"9ace0cfc-0c65-c777-e358-08d471a643d4\"}"

},

"ScheduleModel": {

"Val": "{Id:\"56cb7132-1a65-c72d-6e4e-08d471a65433\"}"

},

"TimeType": {

"Val": "{Id:\"7ec59b14-74de-c106-774b-08d471a6235e\"}"

},

"FormKeys": {

"Val": "{Id:\"a2e6e0cf-293e-c3cb-25eb-08d4651e1674\"}"

},

"NewCarSale": {

"Val": "{Id:\"0bf719b5-20f2-cb7e-75b1-08d46f70143c\"}"

},

"FriendCircleTagDic": {

"Val": "{Id:\"600e5e53-f2fe-c098-be3c-08d475a270b9\"}"

},

"JobTitleSourceId": {

"Val": "{Id:\"54ad027b-53e7-cc53-a9f4-08d47e4fd15a\"}"

},

"JobTitleDicId": {

"Val": "{Id:\"9ea8821d-9d36-c8d3-d3ee-08d47fe09fd0\"}"

},

"FormClassifySourceId": {

"Val": "{Id:\"3e37b3cb-fe8d-c69f-dfb5-08d4b1771f55\"}"

},

"MainPage\_Template\_LBMB-01": {

"Val": "{Id:\"f1790b37-5320-11e7-abd5-000c29d77103\"}"

},

"MainPage\_Template\_LBMB-02": {

"Val": "{Id:\"7eedb379-5322-11e7-abd5-000c29d77103\"}"

},

"MainPage\_Template\_LBMB-03": {

"Val": "{Id:\"d8495f5b-5322-11e7-abd5-000c29d77103\"}"

},

"MainPage\_Template\_LBMB-04": {

"Val": "{Id:\"58025d0d-5323-11e7-abd5-000c29d77103\"}"

},

"MainPage\_Template\_LBMB-05": {

"Val": "{Id:\"a6ec8ab7-5323-11e7-aee7-000c293e3800\"}"

},

"MainPage\_Template\_LBMB-11": {

"Val": "{Id:\"1d208d60-5324-11e7-abd5-000c29d77103\"}"

},

"UserValidType": {

"Val": "{Id:\"dfead9b7-b68d-c34c-4db5-08d4a30fe180\"}"

},

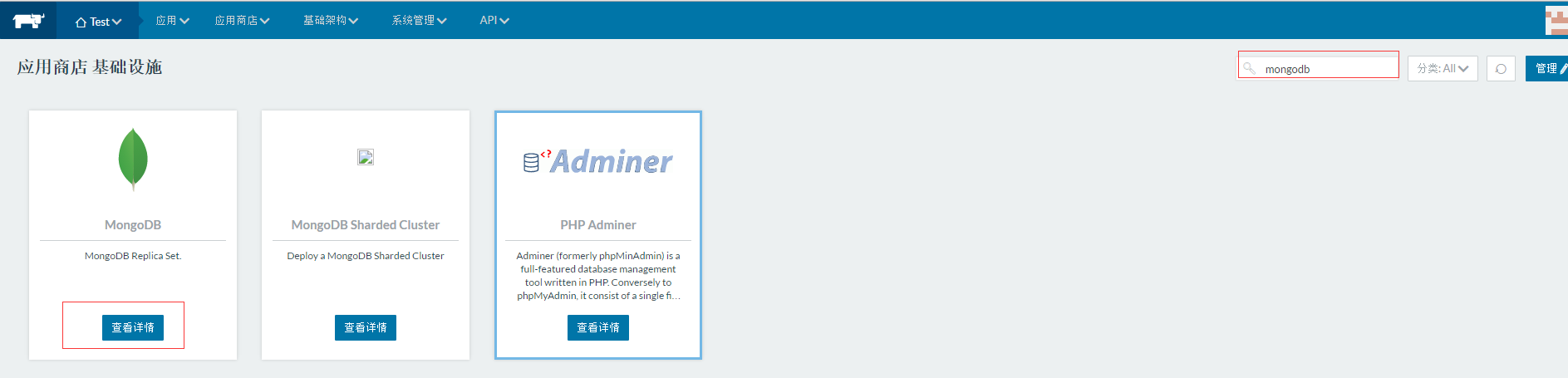
"User\_Relations": {

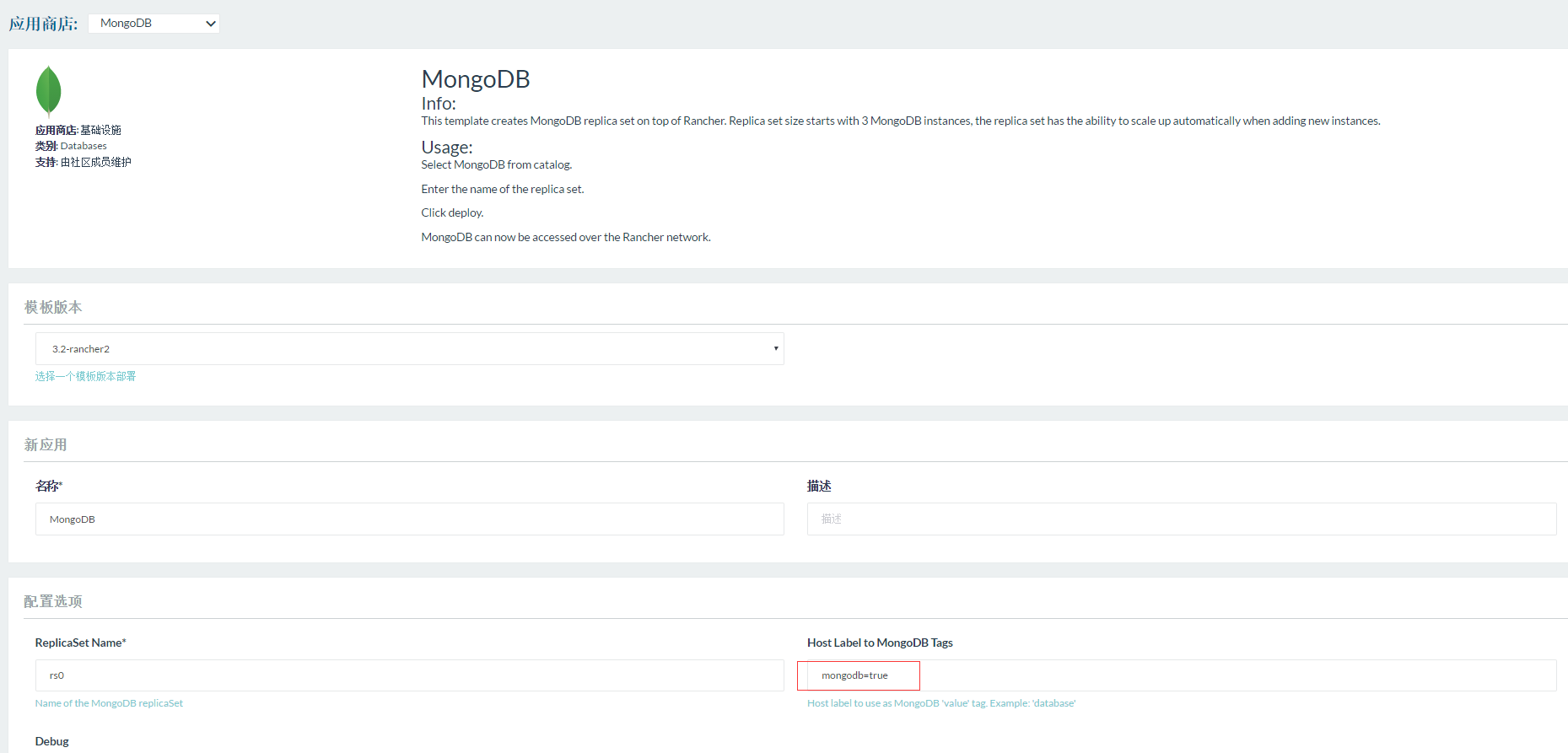
"Val": "{Id:\"445c5cb8-51db-cc0f-3e12-08d5081615d5\"}"

}

}

### 5.1.5 mongodb



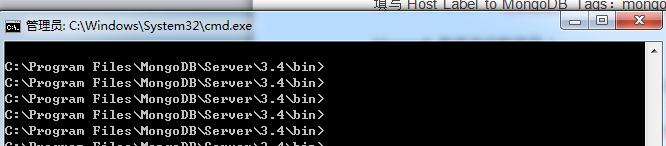


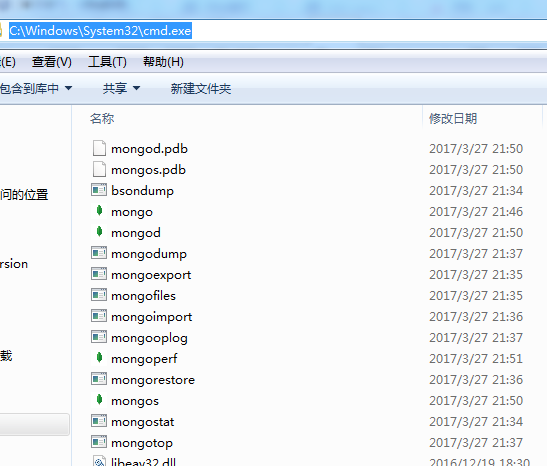
填写Host Label to MongoDB Tags：mongodb=true

Mongodb数据库的数据导入：（前提是连接上openvpn，vpn账号密码为rancher管理平台账号密码。）

从预生产环境把mongodb（192.168.3.18）里面的数据库备份出来，进行如下操作：

首先在mongodb工具所在路径下，输入cmd，进入命令行模式





在命令行模式中输入要备份的数据库的ip、名称、数据库连接账号和密码。

mongodump.exe /host 192.168.3.18 /db AlarmCenter /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

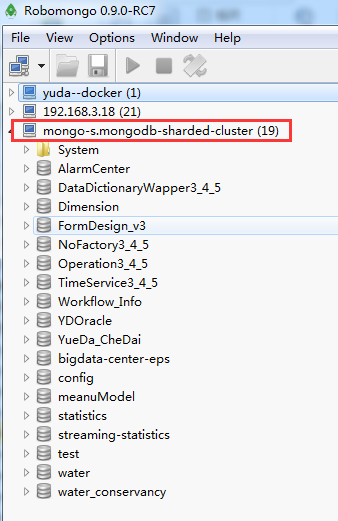
备份完成后，将其还原到新的数据库中，还是在命令行模式中进行如下操作：

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db water\_conservancy /dir "F:\data\AlarmCenter" /drop /username root /password gt86589089 /authenticationDatabase admin

备注：mongodb此次备份涉及到的数据库如下

|  |
| --- |
| AlarmCenter |
| DataDictionaryWapper3\_4\_5 |
| Dimension |
| FormDesign\_v3 |
| NoFactory3\_4\_5 |
| Operation3\_4\_5 |
| TimeService3\_4\_5 |
| Workflow\_Info |
| YDOracle |
| YueDa\_CheDai |
| bigdata-center-eps |
| meanuModel |
| statistics |
| streaming-statistics |
| test |
| water |
| water\_conservancy |

数据库备份还原后，可以在工具中查看已导入的结果。



完整的备份及还原语句见如下：

备份（数据源是预生产环境mongodb库的）：

mongodump.exe /host 192.168.3.18 /db AlarmCenter /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db DataDictionaryWapper3\_4\_5 /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db Dimension /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db FormDesign\_v3 /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db NoFactory3\_4\_5 /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db Operation3\_4\_5 /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db TimeService3\_4\_5 /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db Workflow\_Info /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db YDOracle /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db YueDa\_CheDai /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db bigdata-center-eps /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db menuModel /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db statistics /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db streaming-statistics /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db test /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db water /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

mongodump.exe /host 192.168.3.18 /db water\_conservancy /out "F:\data" /username GTAppUser /password GtApp86589089Hb /authenticationDatabase admin

还原：

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db AlarmCenter /dir "F:\data\AlarmCenter" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db DataDictionaryWapper3\_4\_5 /dir "F:\data\DataDictionaryWapper3\_4\_5" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db Dimension /dir "F:\data\Dimension" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db FormDesign\_v3 /dir "F:\data\FormDesign\_v3" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db NoFactory3\_4\_5 /dir "F:\data\NoFactory3\_4\_5" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db Operation3\_4\_5 /dir "F:\data\Operation3\_4\_5" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db TimeService3\_4\_5 /dir "F:\data\TimeService3\_4\_5" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db Workflow\_Info /dir "F:\data\Workflow\_Info" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db YDOracle /dir "F:\data\YDOracle" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db YueDa\_CheDai /dir "F:\data\YueDa\_CheDai" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db bigdata-center-eps /dir "F:\data\bigdata-center-eps" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db menuModel /dir "F:\data\menuModel" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db statistics /dir "F:\data\statistics" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db streaming-statistics /dir "F:\data\streaming-statistics" /drop /username root /password gt86589089 /authenticationDatabase admin

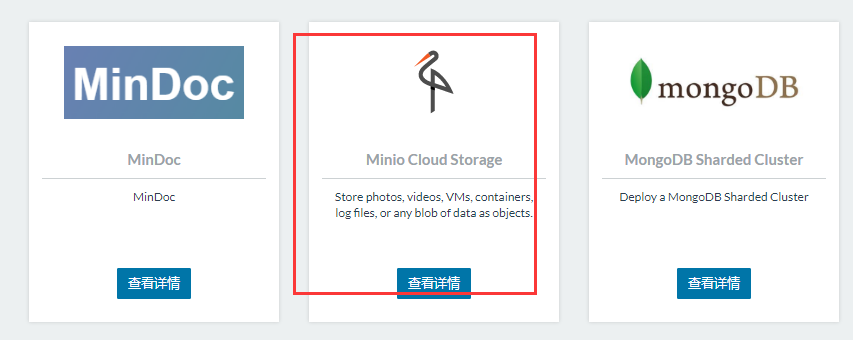
mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db test /dir "F:\data\test" /drop /username root /password gt86589089 /authenticationDatabase admin

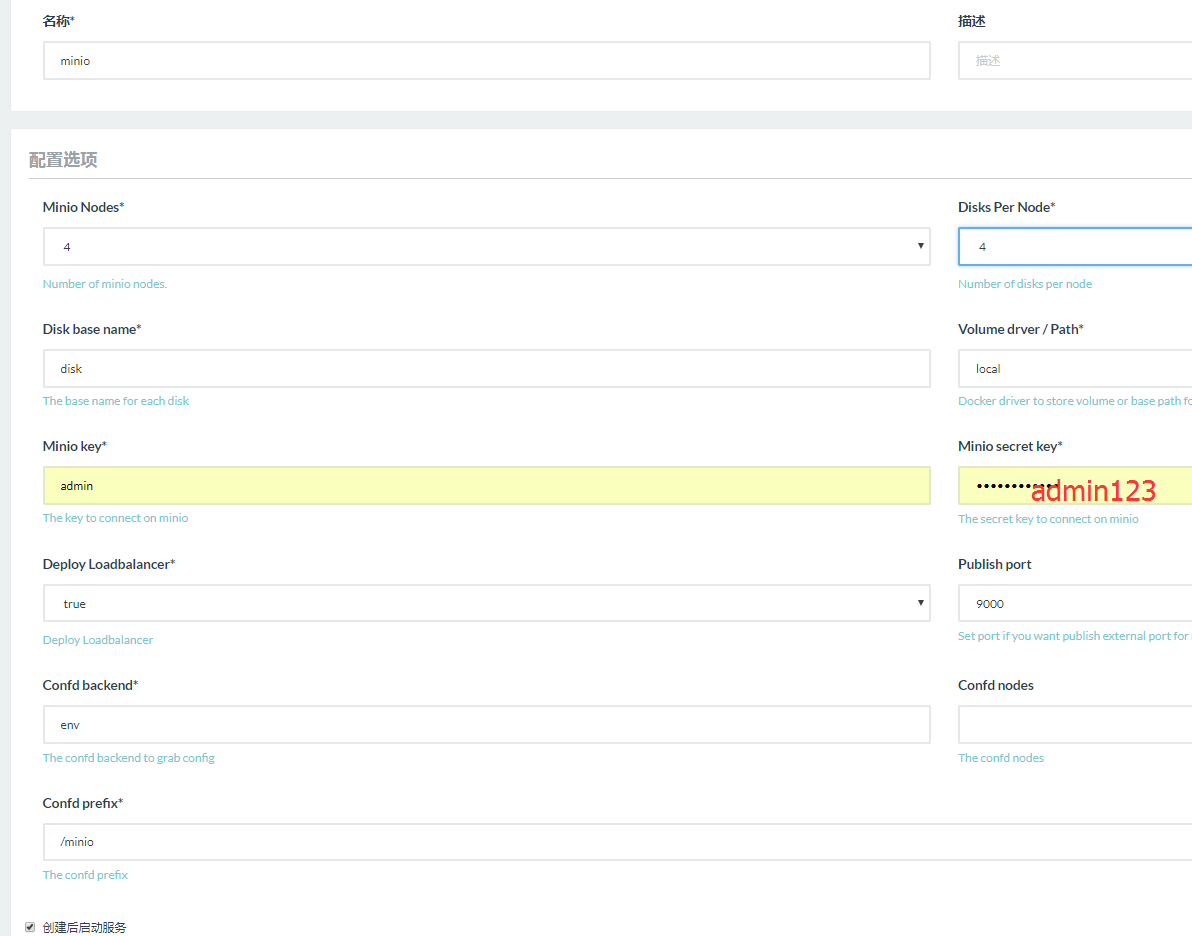
mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db water /dir "F:\data\water" /drop /username root /password gt86589089 /authenticationDatabase admin

mongorestore.exe /host mongo-s.mongodb-sharded-cluster /db water\_conservancy /dir "F:\data\water\_conservancy" /drop /username root /password gt86589089 /authenticationDatabase admin

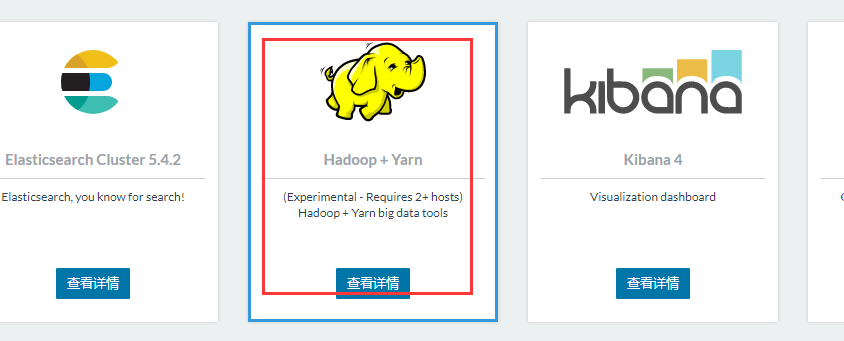
### 5.1.6 minio

在rancher应用商店搜素Minio，使用以下模板 Minio Cloud Storage，选择2017-03-16-rancher1版本后,Minio Nodes栏选择4个节点，Disks Per Node栏选择4个驱动，然后直接点击启动即可，等待启动成功。



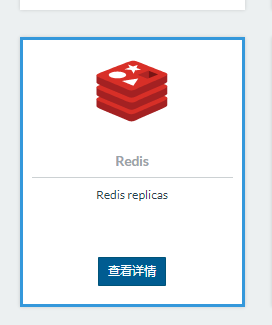


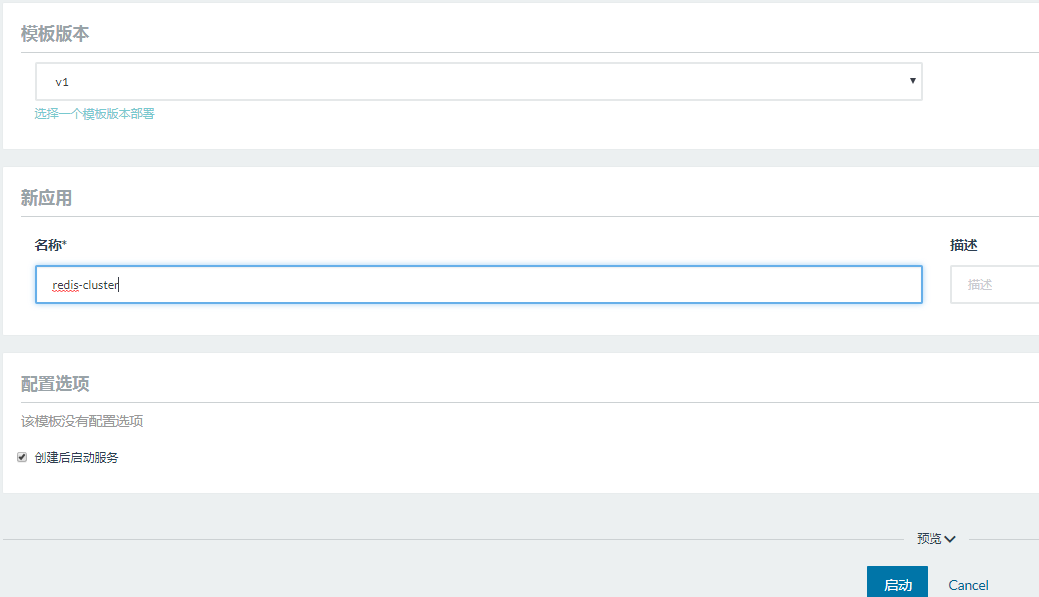
### 5.1.7 hadoop



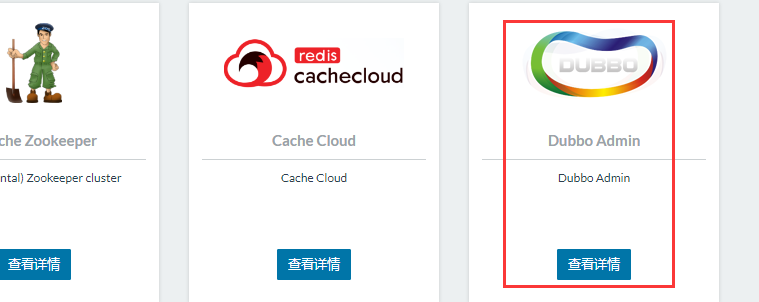
默认安装即可。

### 5.1.8 redis

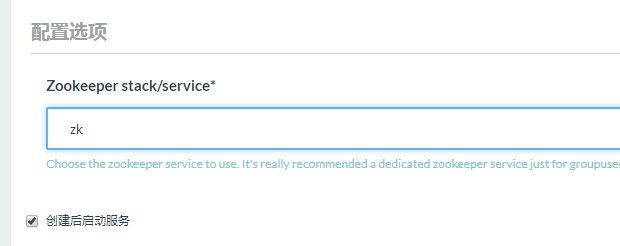
基础设施中搜索redis



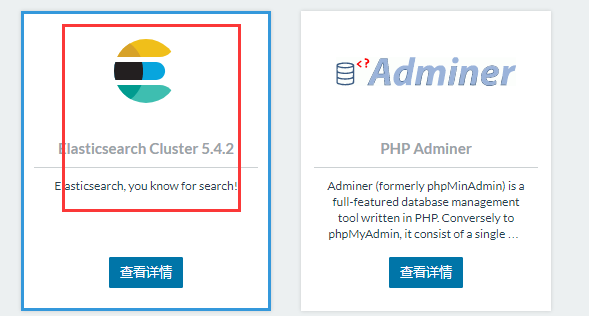
### 5.1.9 dubbo



配置选项选择zk



### 5.1.10 Elasticsearch

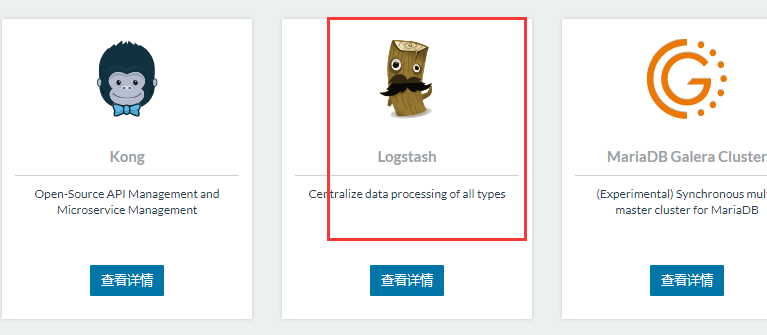


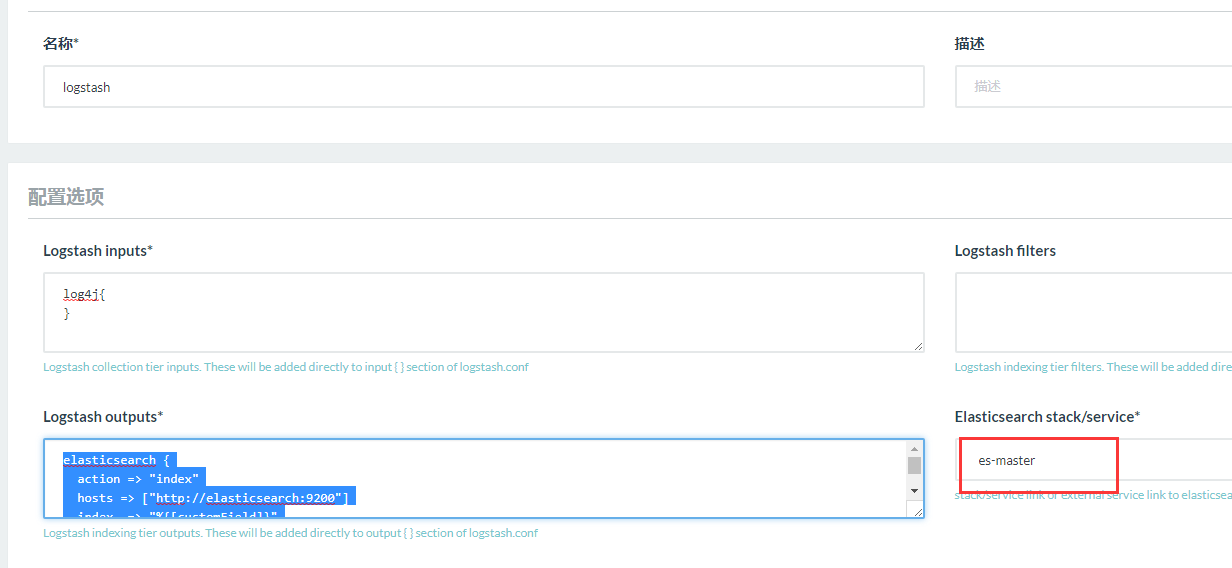
在rancher应用商店搜素Elasticsearch，使用以下模板 Elasticsearch Cluster 5.4.2，选择最新版本后直接点击启动即可，等待启动成功。

Elasticsearch启动要求：sysctl -w vm.max\_map\_count=262144

上面的指令在每台物理机上执行下。

### 5.1.11 Logstash





在rancher应用商店搜素Elasticsearch，使用以下模板 Logstash，选择版本2.4.1-rancher1(重要：5.1.1-rancher1版本有bug，不要使用),Logstash inputs一栏输入：

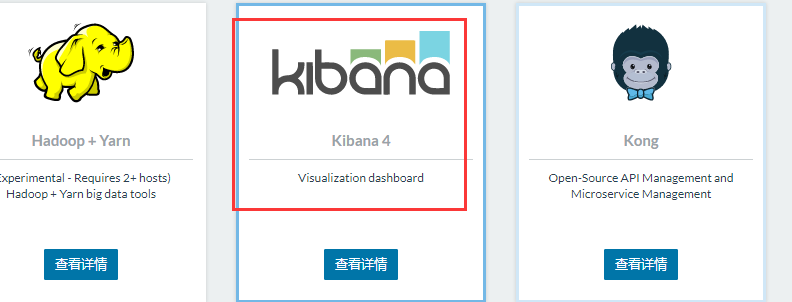
1. log4j{
2. }

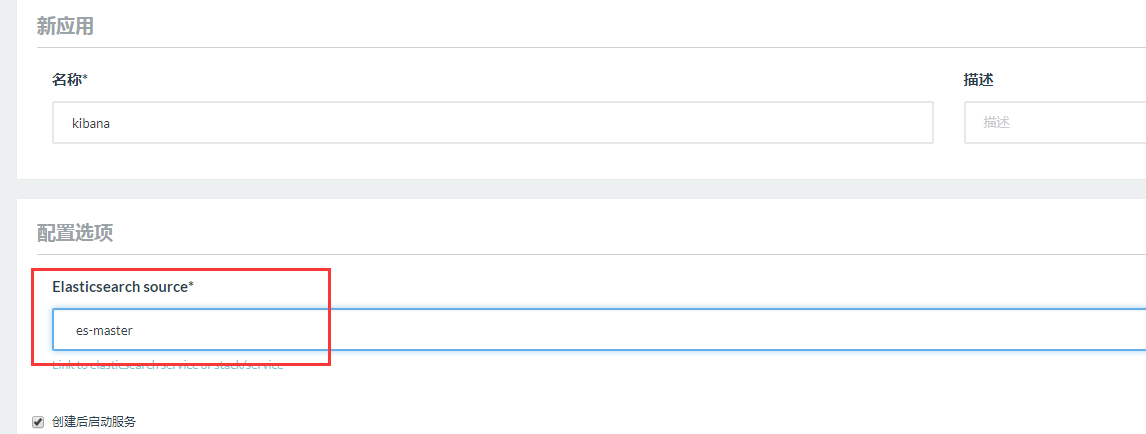
这样配置默认在本机端口监听5670 tcp端口，如需自定义配置则根据官方文档进行配置即可，注意logstash的版本为2.4.1。  
Logstash outputs一栏输入：

1. elasticsearch {
2. action => "index"
3. hosts => ["http://elasticsearch:9200"]
4. index => "%{[customField]}"
5. }
6. if [priority] == 'ERROR' {
7. kafka {
8. bootstrap\_servers => "broker.kafka:9092"
9. topic\_id => "logstash"
10. }
11. }

然后选择已经启动好的Elasticsearch(master)集群，点击启动，等待启动成功即可。

### 5.1.12 Kibana

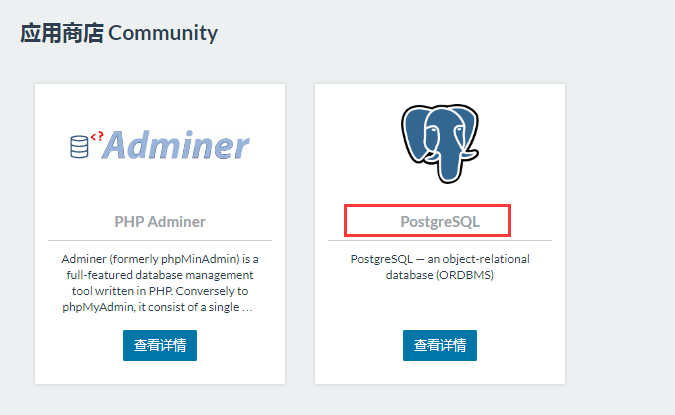


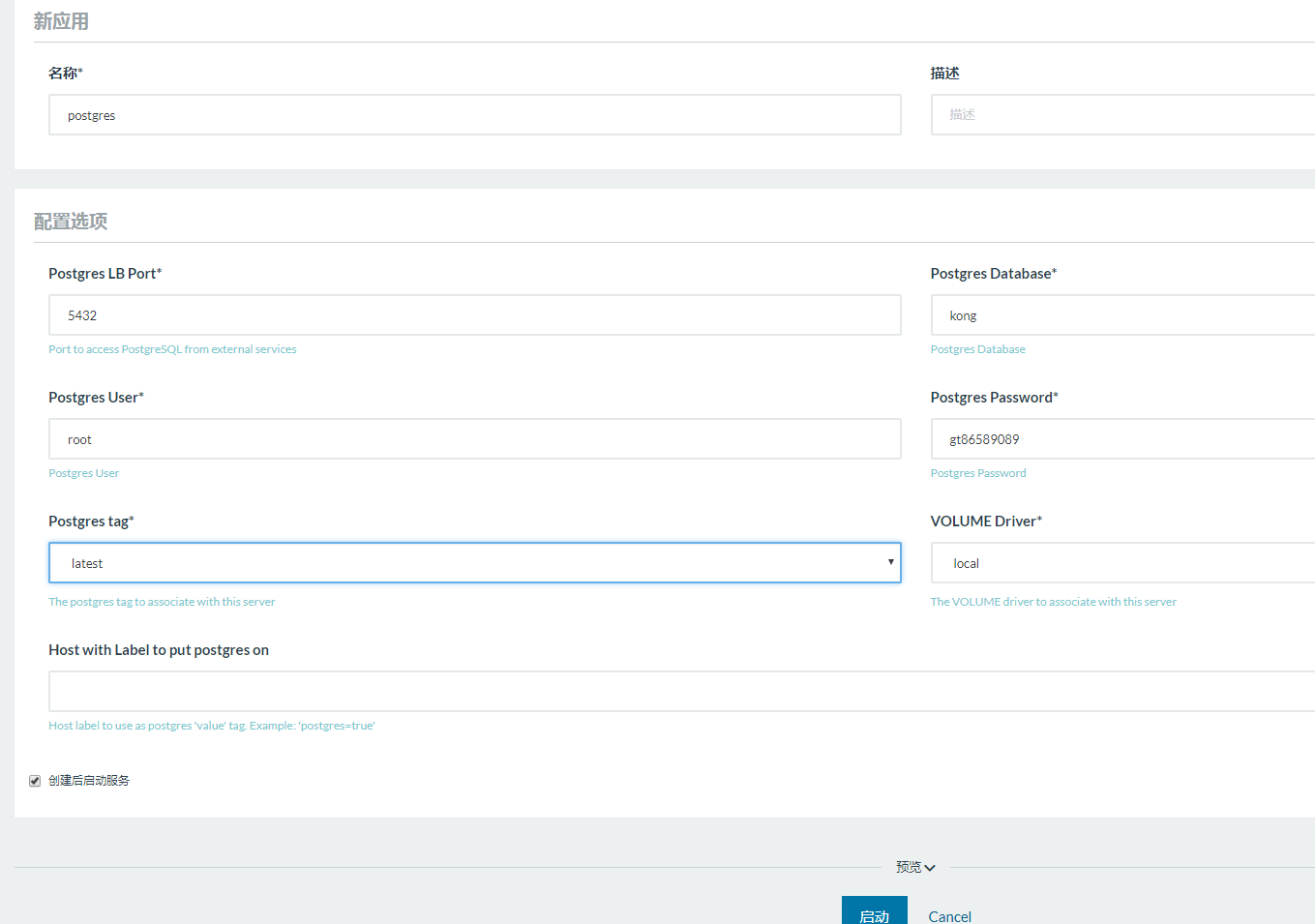


### 5.1.13 token



在社区共享中，搜索‘postgresql’

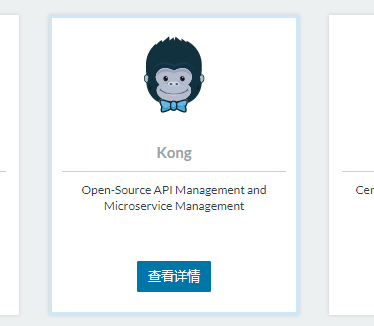




postgresql配置注意事项：  
Db User : root  
Db Pass : gt86589089  
Db Name : kong

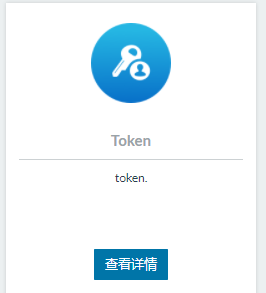
Postgres Tag:lastest

然后在基础设置里面找到kong安装



详细配置过程参见：<http://192.168.9.30:8182/docs/kong/379>（可以见下面的kong刷库步骤。）

在国通应用中找到‘token’安装

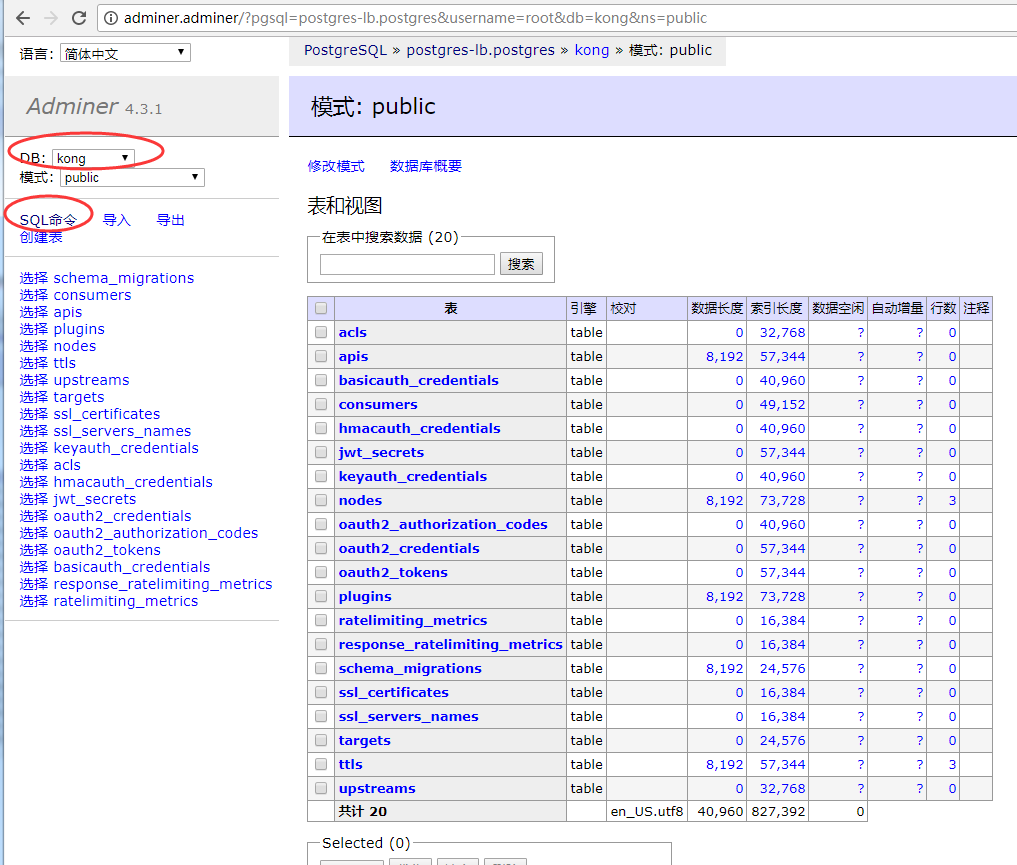


kong刷库步骤：

1. 连接vpn
2. 在浏览器输入地址：<http://adminer.adminer/>出现如下界面



1. 选择数据库PostgreSQLl，服务器：postgres-lb.postgres
2. 输入用户名密码后进入如下数据库选择kong



1. 点击SQL命令



1. 贴入SQL脚本语句，执行。

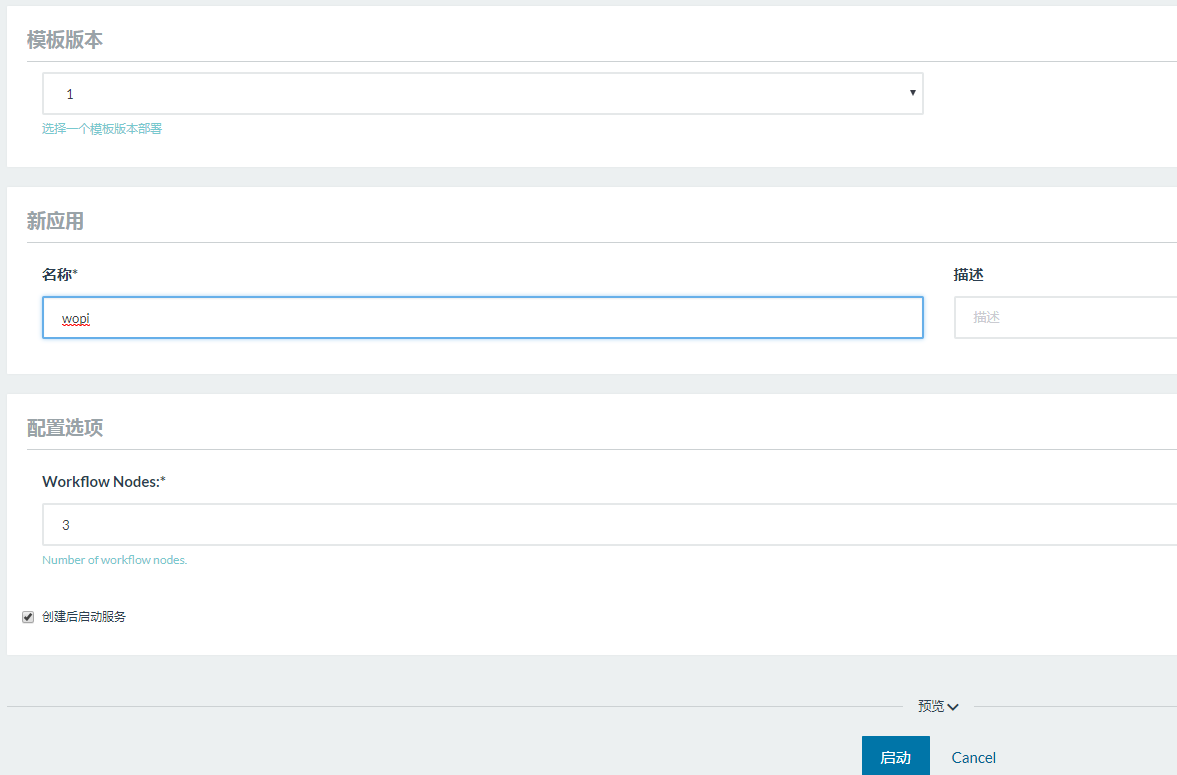


### 5.1.14 wopi

先安装wopi，然后在安装collabora。

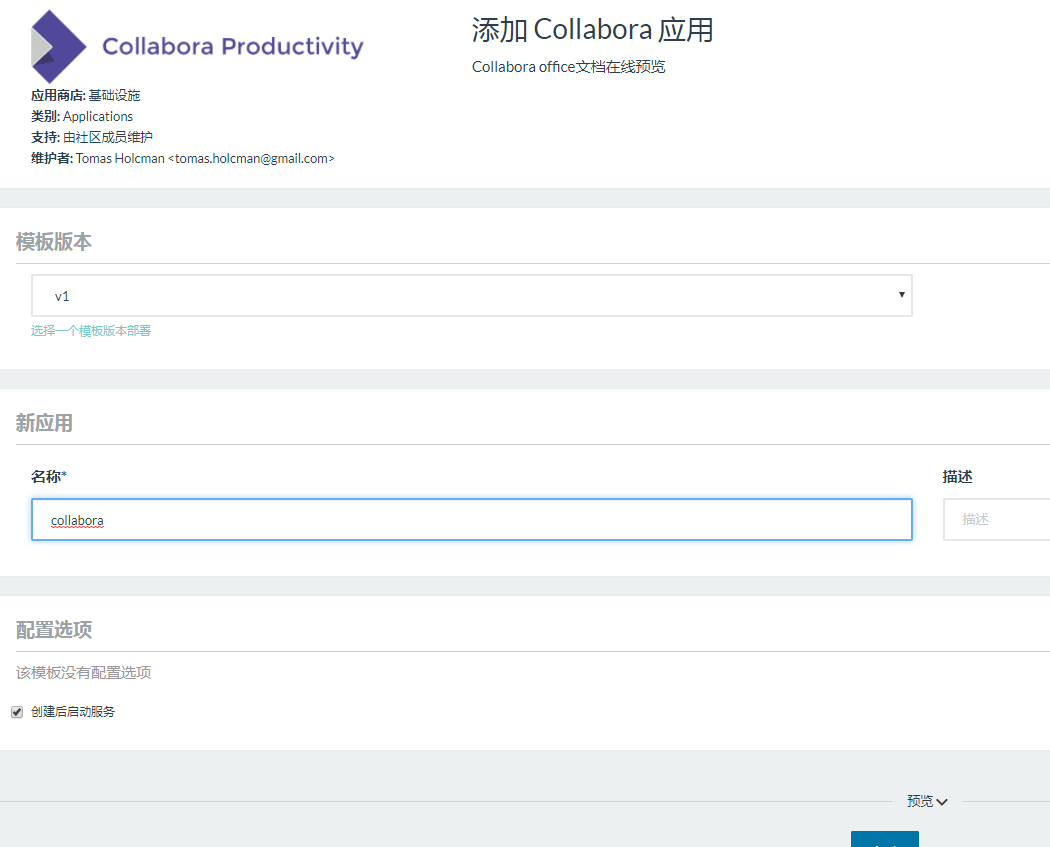
首先安装wopi服务，在国通应用中找到wopi，默认安装。

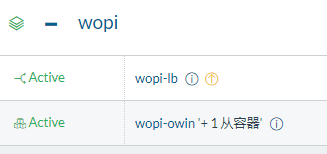
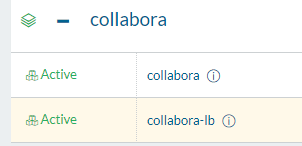




然后在基础设施中找到‘collabora’服务，默认安装。



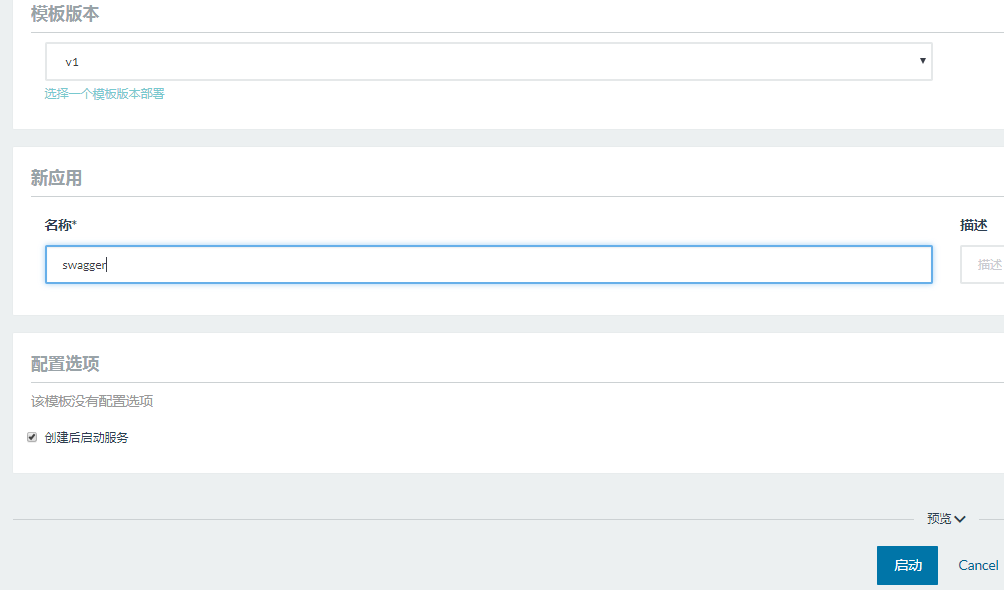




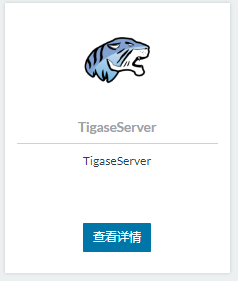
如上，安装后的2个服务。

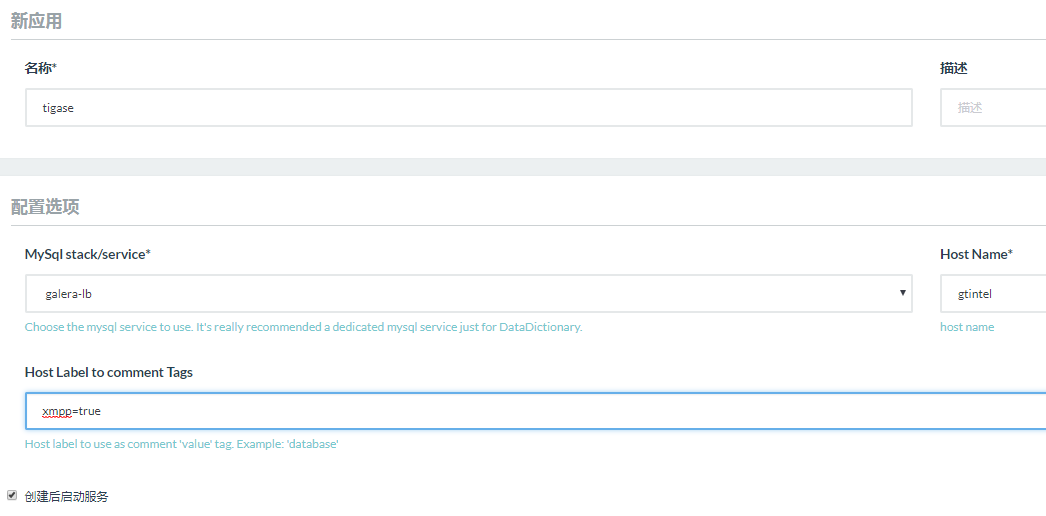
### 5.1.15 swagger

基础设置中搜索swagger，默认安装。



### 5.1.16 xmpp





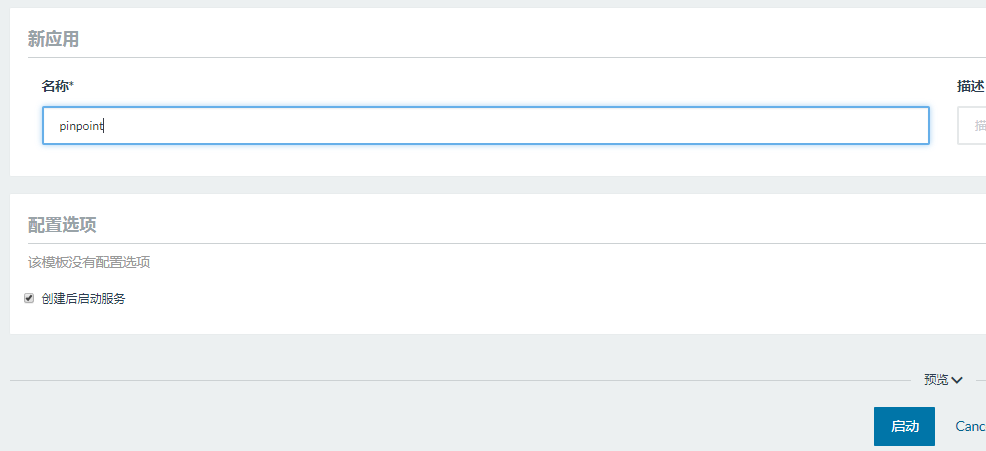
Mysql stack:galera-lb

Host Name:gtintel

Host Label to comment Tags: xmpp=true

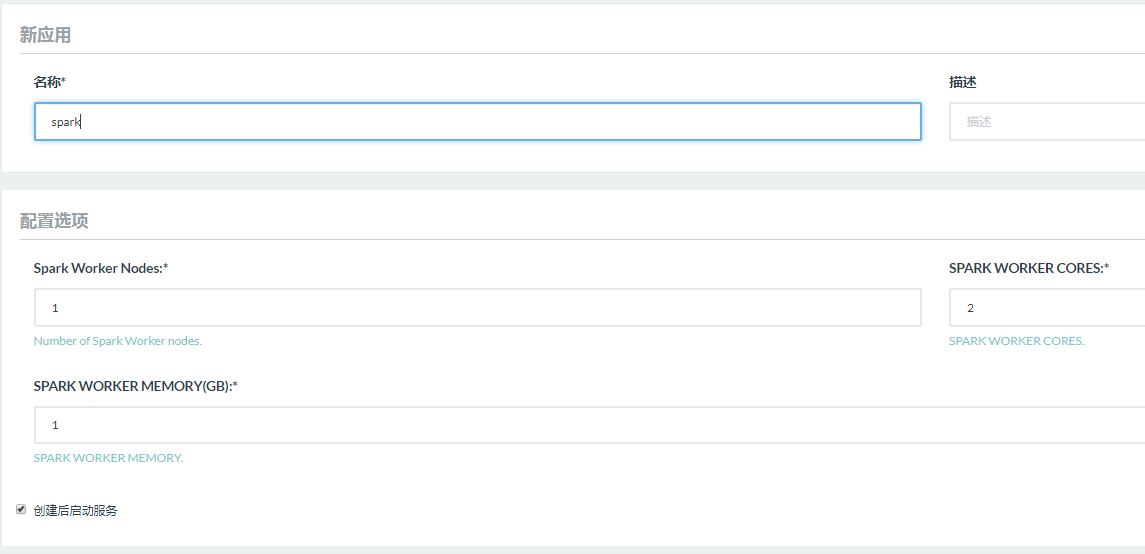
### 5.1.17 pinpoint



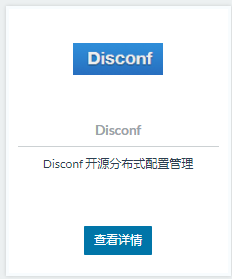


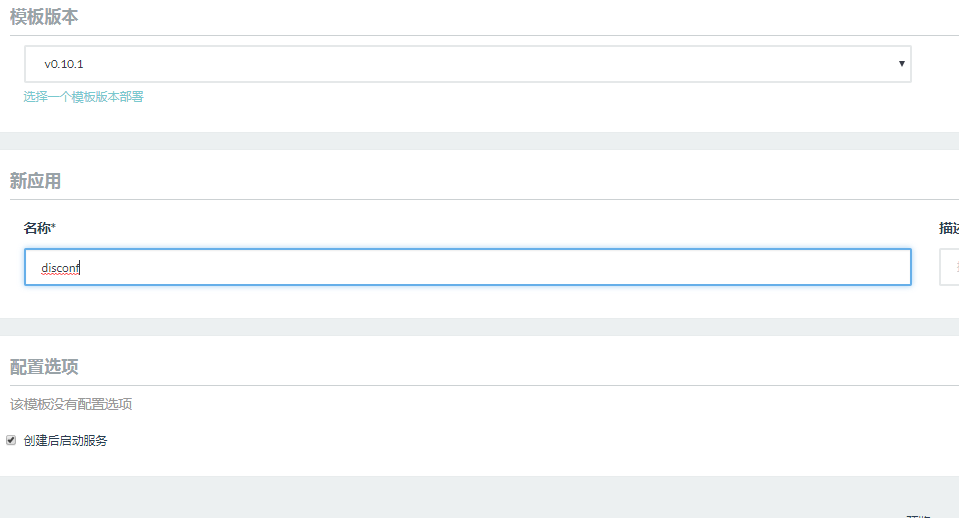
### 5.1.18 spark





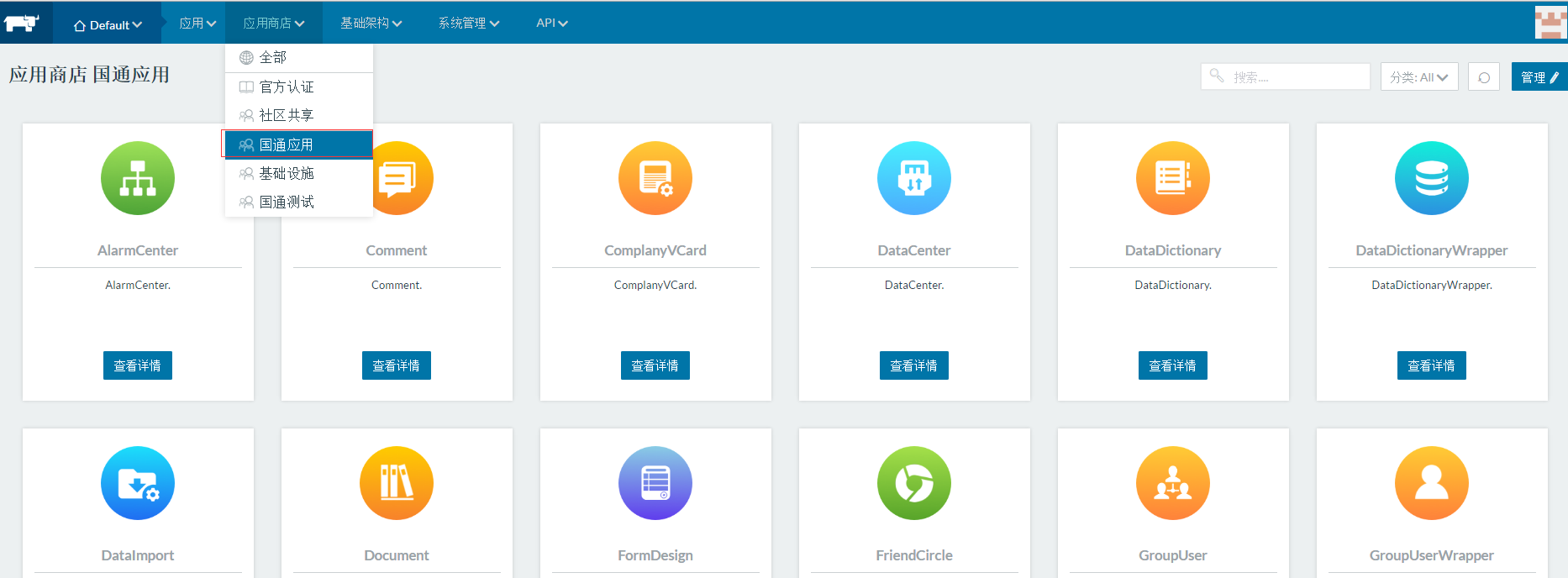
### 5.1.19 disconf





## 5.2国通应用

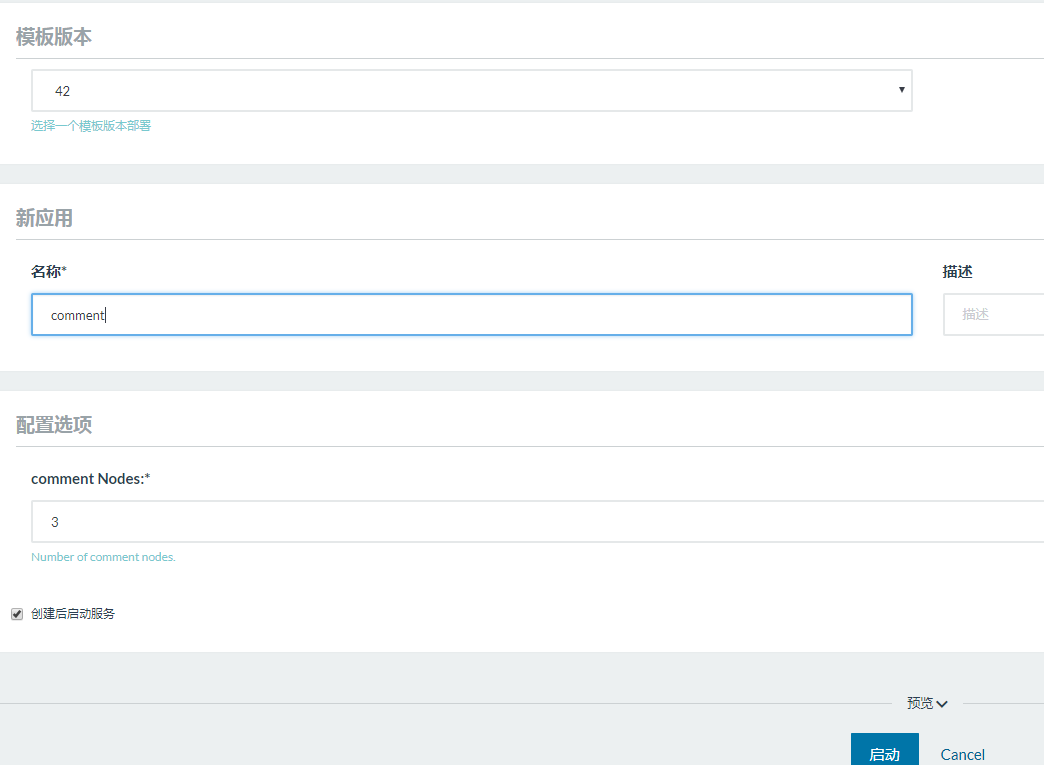
从应用商店中选取“国通应用”进行筛选。



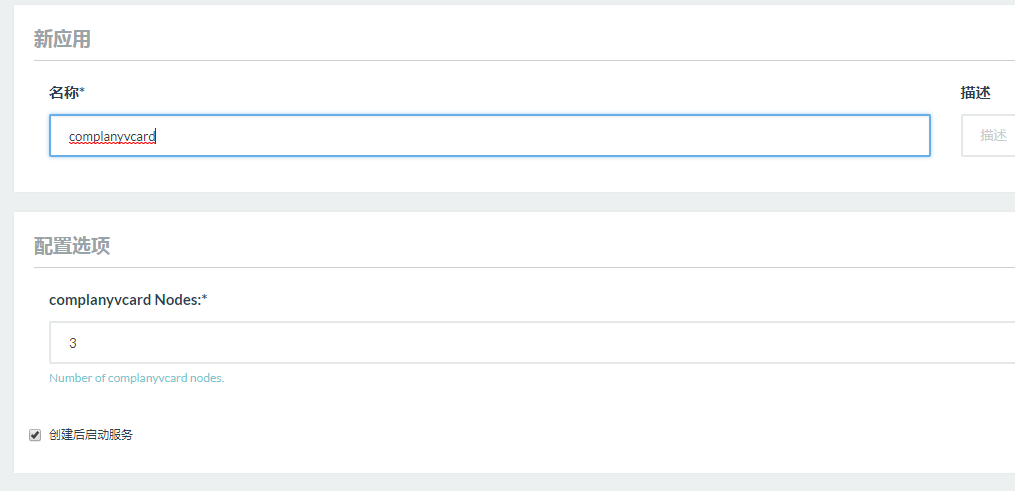
在用jenkins打包的之前，修改<http://192.168.1.161:8011/jenkins>里面

---系统管理---Managed files---json file。

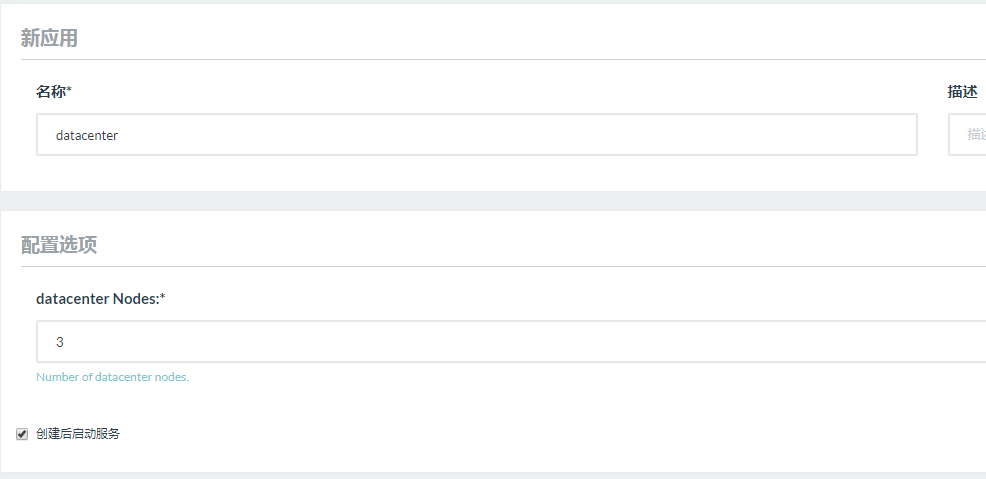
### 5.2.1 comment



### 5.2.2 ComplanyVCard



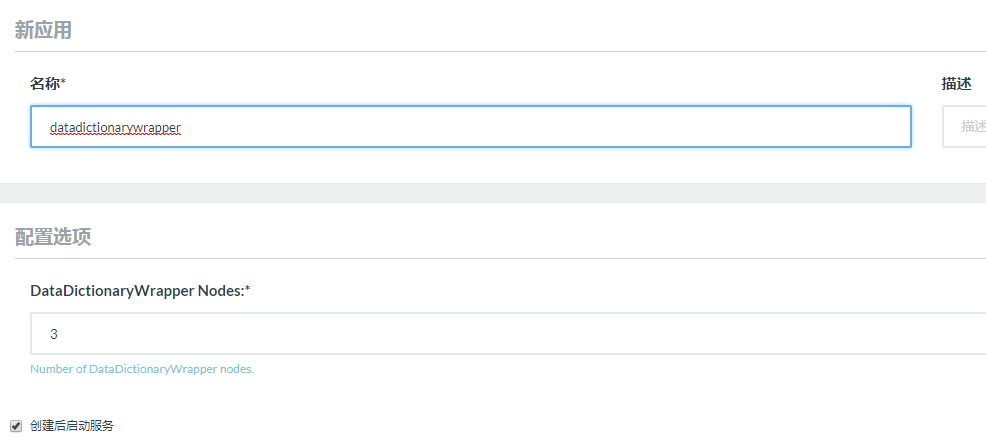
### 5.2.3 DataCenter



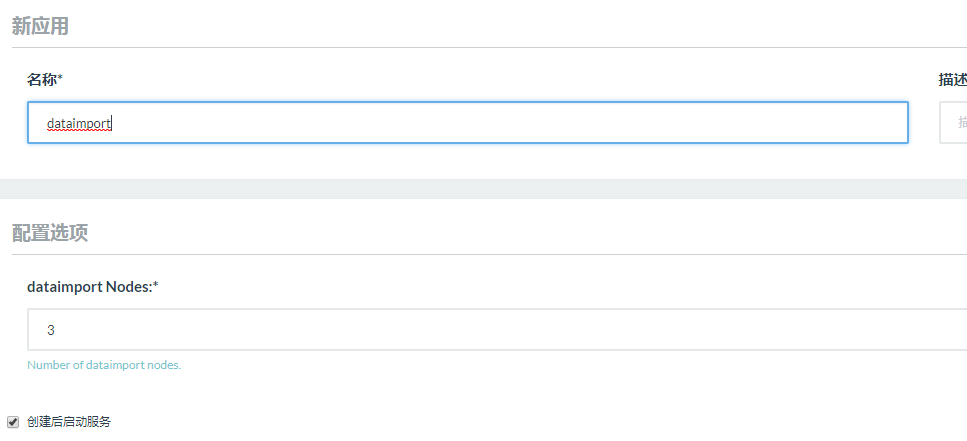
### 5.2.4 datadictionary



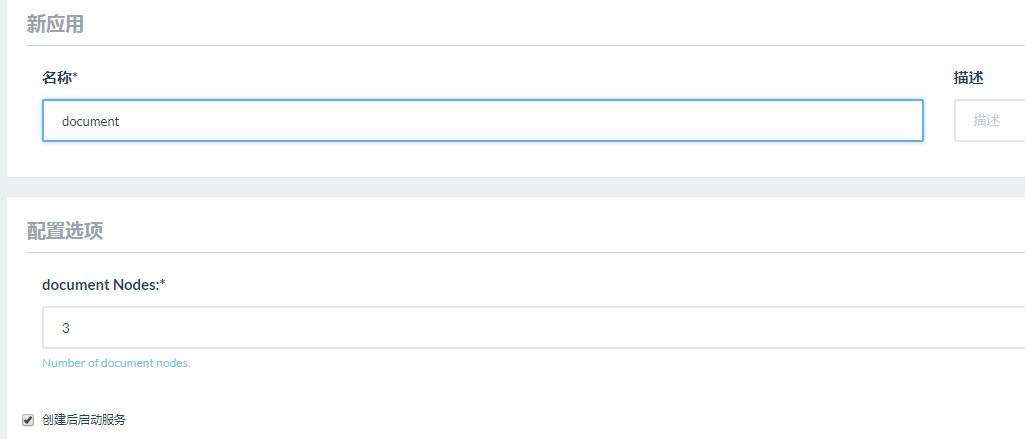
### 5.2.5 datadictionarywraper



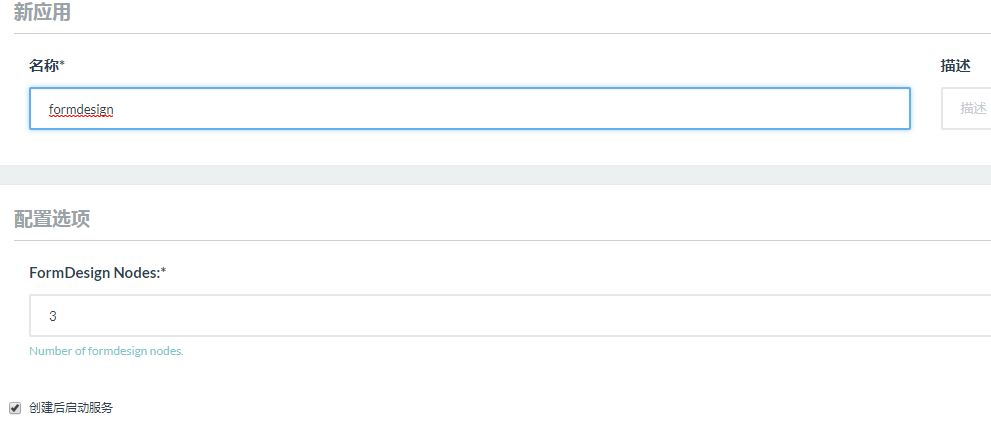
### 5.2.6 DataImport



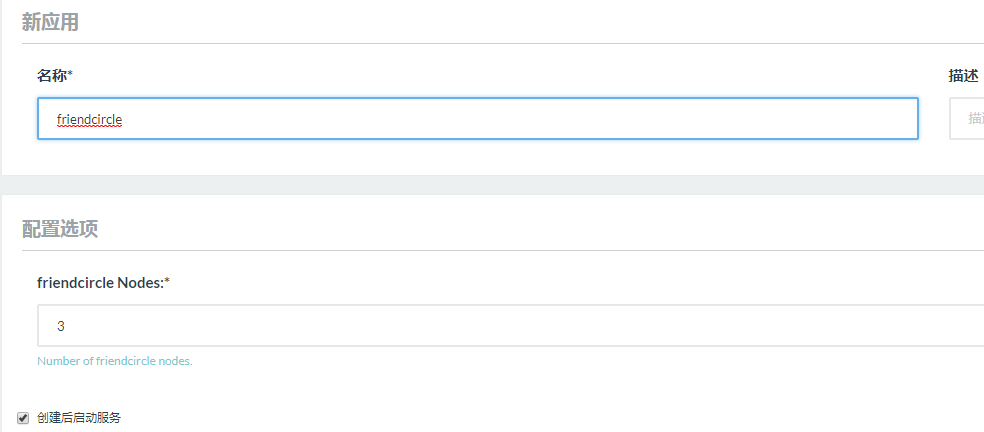
### 5.2.7 Document



### 5.2.8 FormDesign



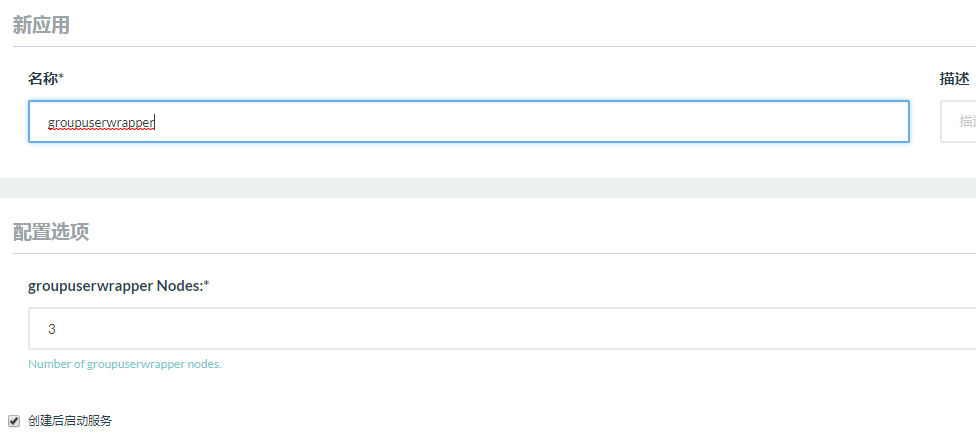
### 5.2.9 FriendCircle



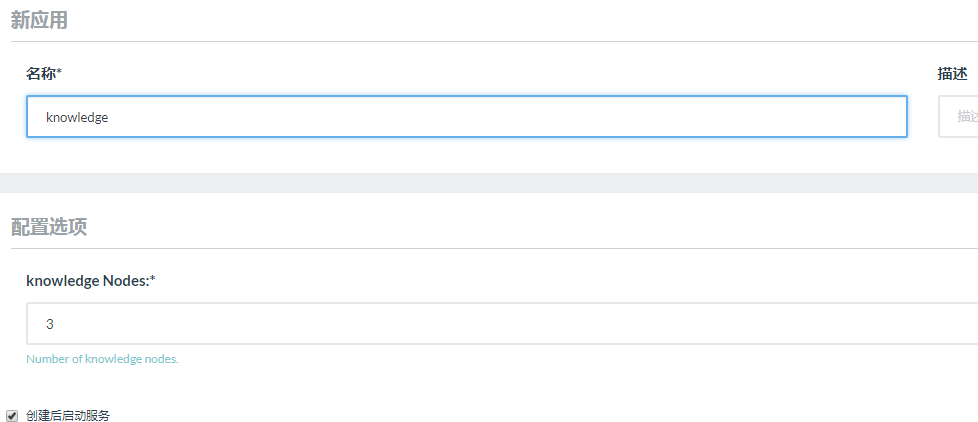
### 5.2.10 GroupUser



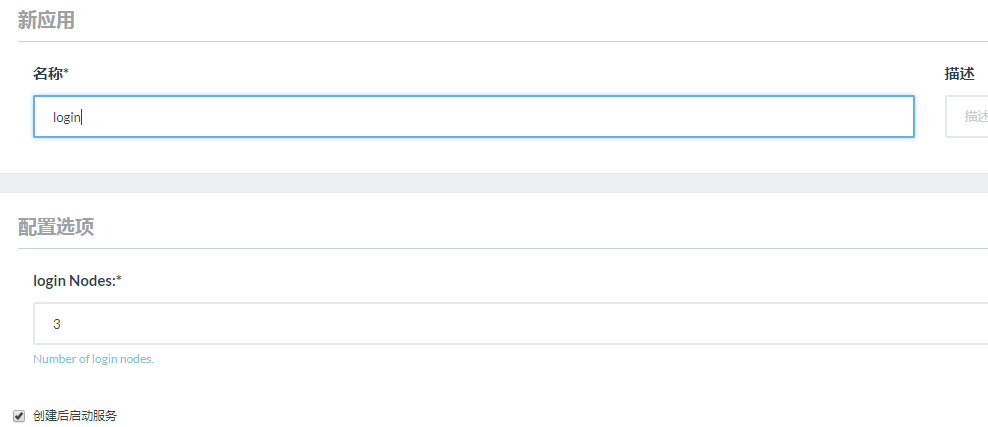
### 5.2.11 GroupUserWraper



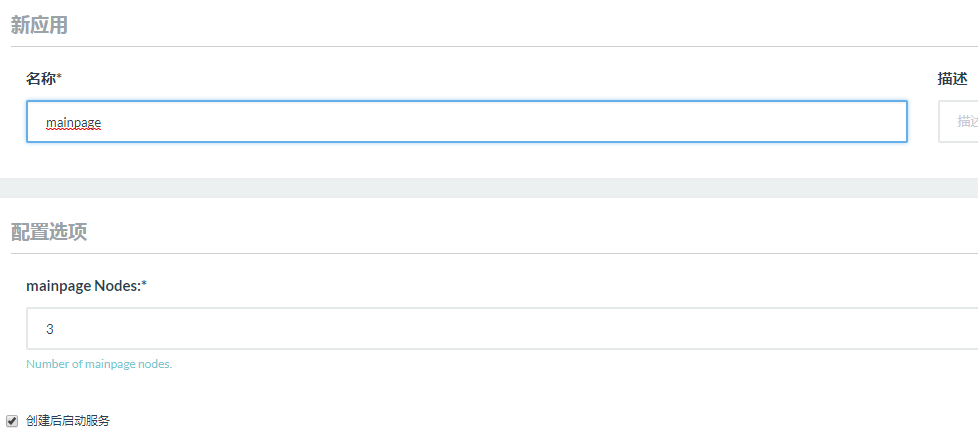
### 5.2.12 Knowledge



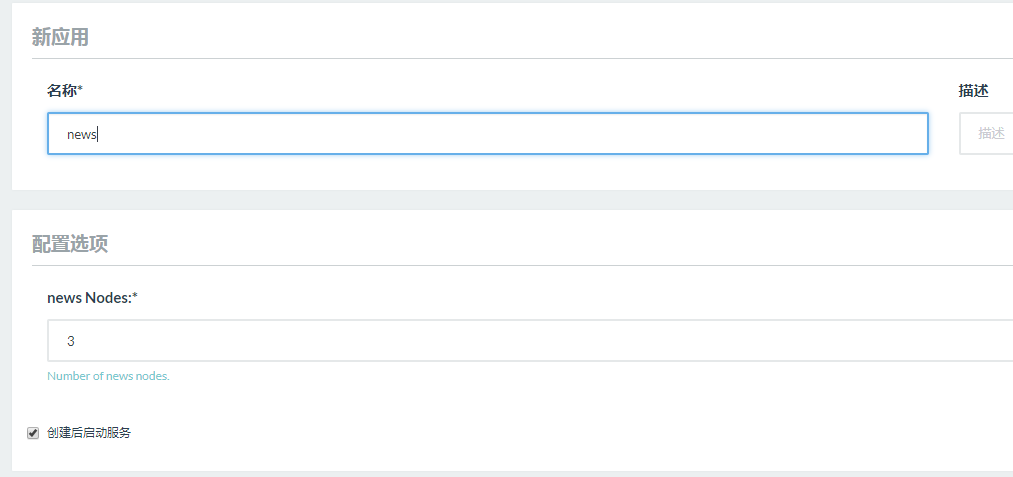
### 5.2.13 Login



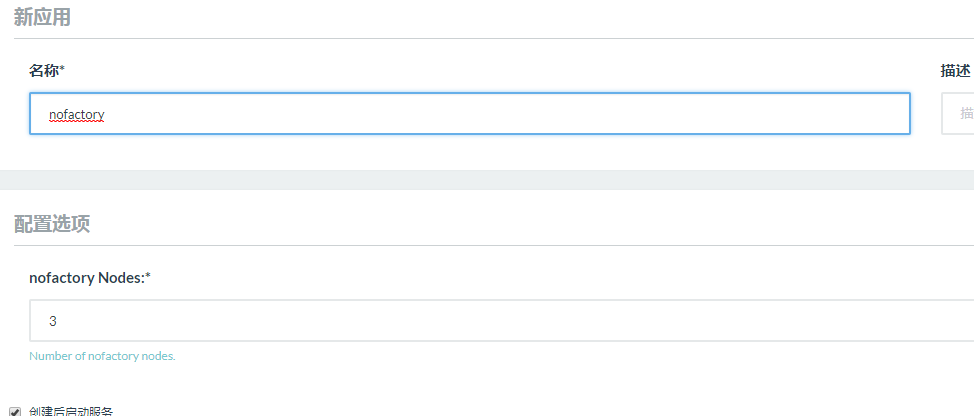
### 5.2.14 MainPage



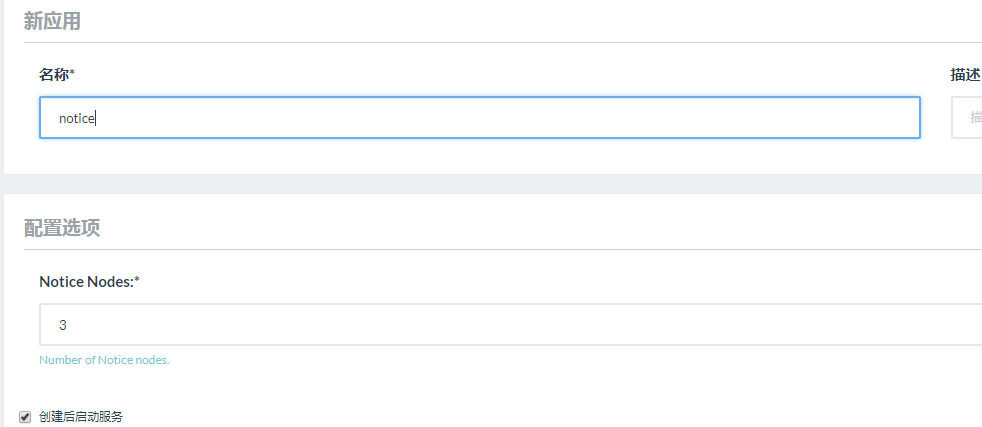
### 5.2.15 News



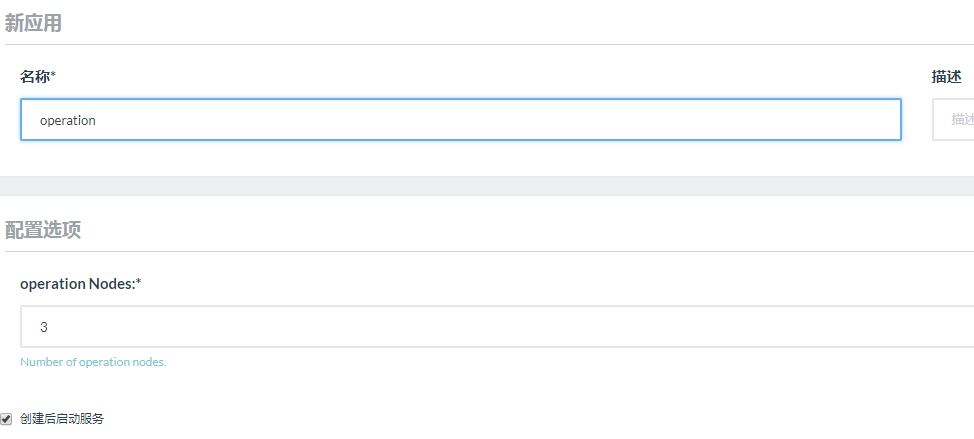
### 5.2.16 NoFactory



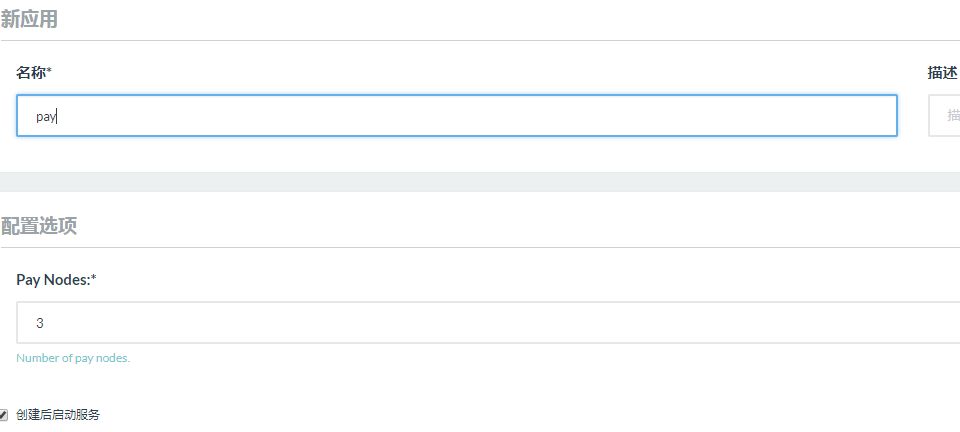
### 5.2.17 Notice



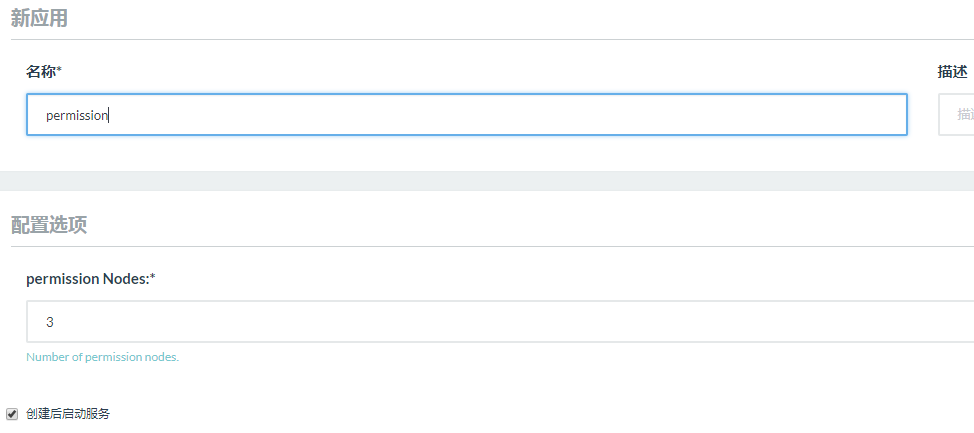
### 5.2.18 Operation



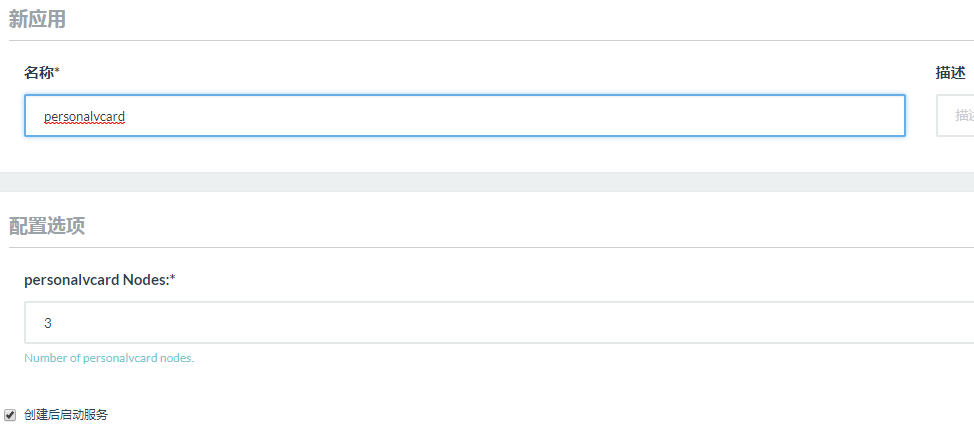
### 5.2.19 pay



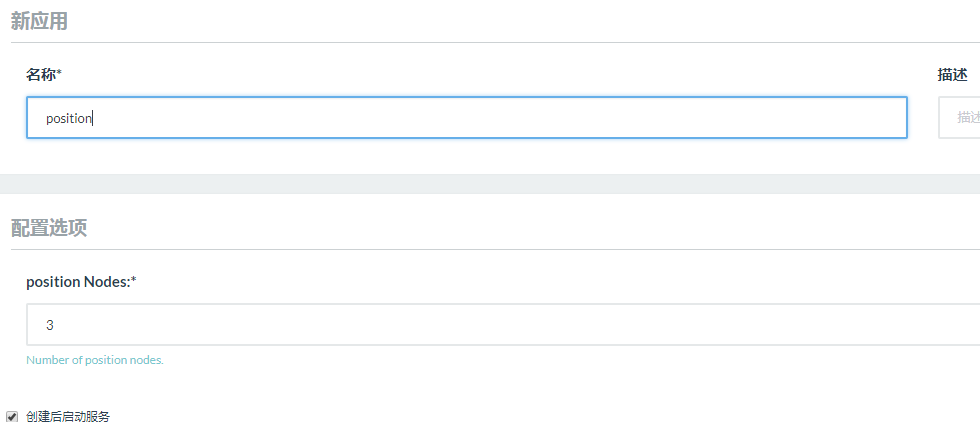
### 5.2.20 Permission



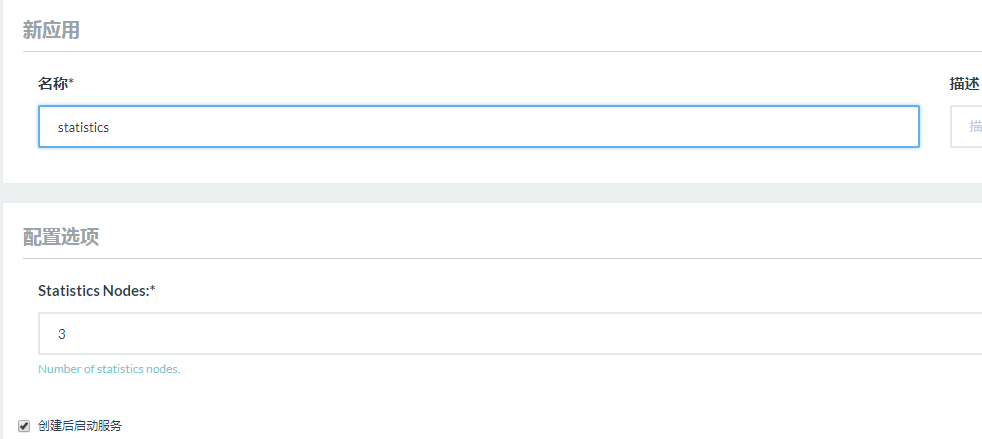
### 5.2.21 PersonalVCard



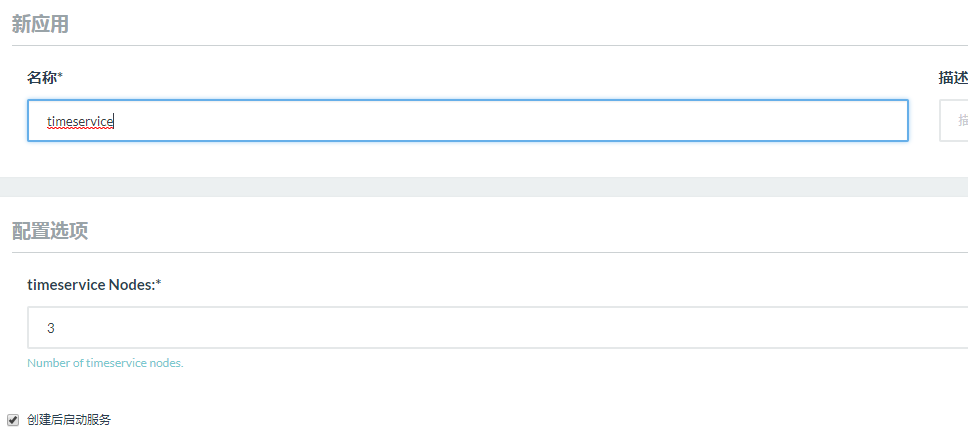
### 5.2.22 Position



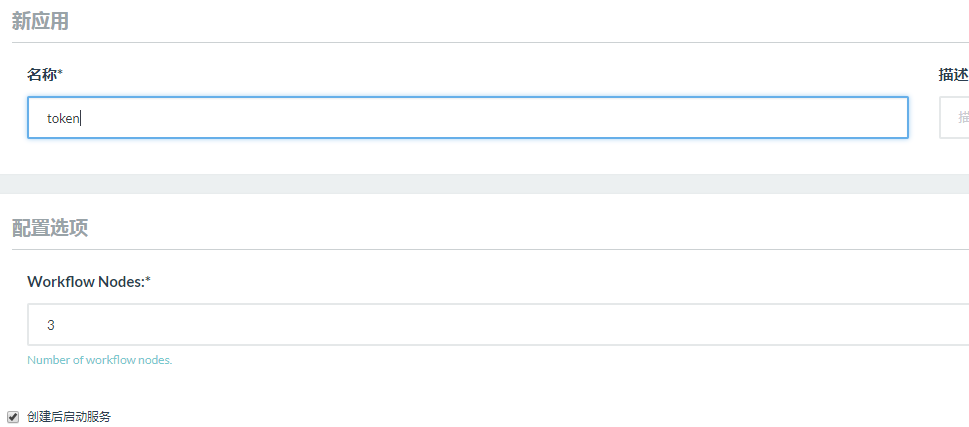
### 5.2.23 Staitics



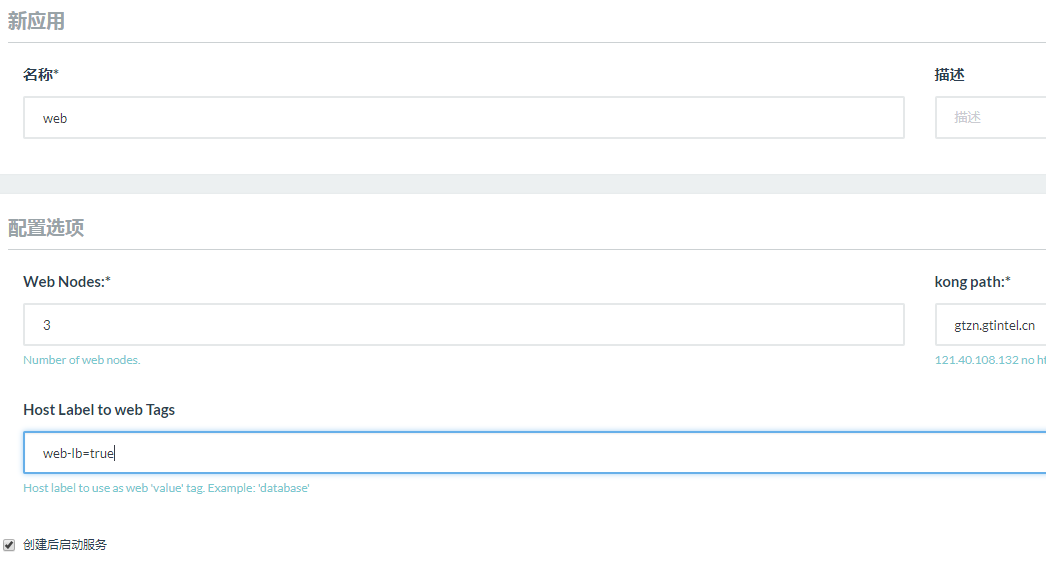
### 5.2.24 TimeService



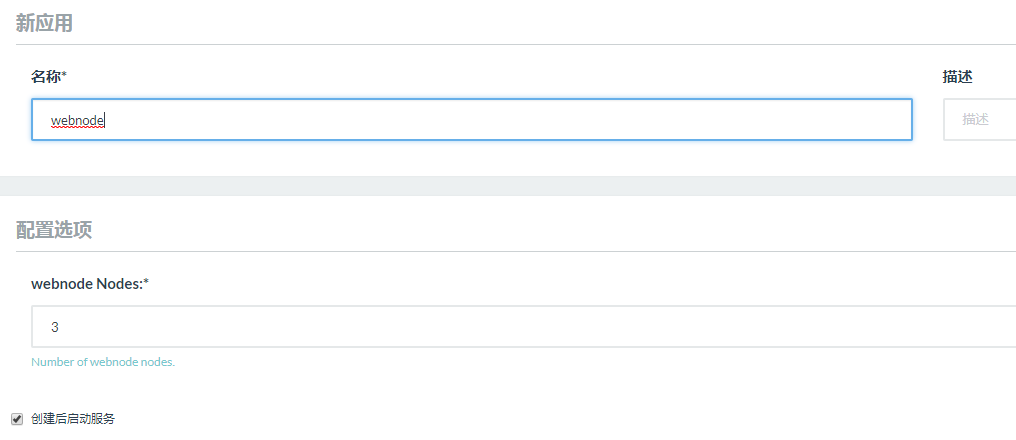
### 5.2.25 Token



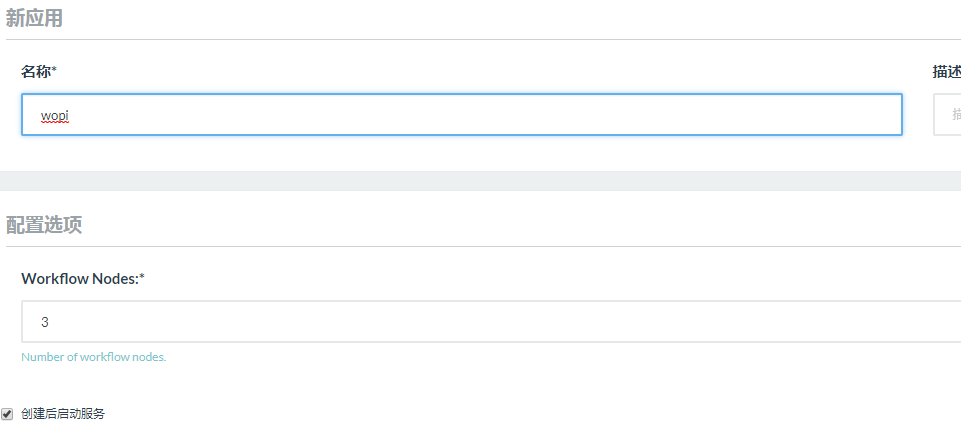
### 5.2.26 Web



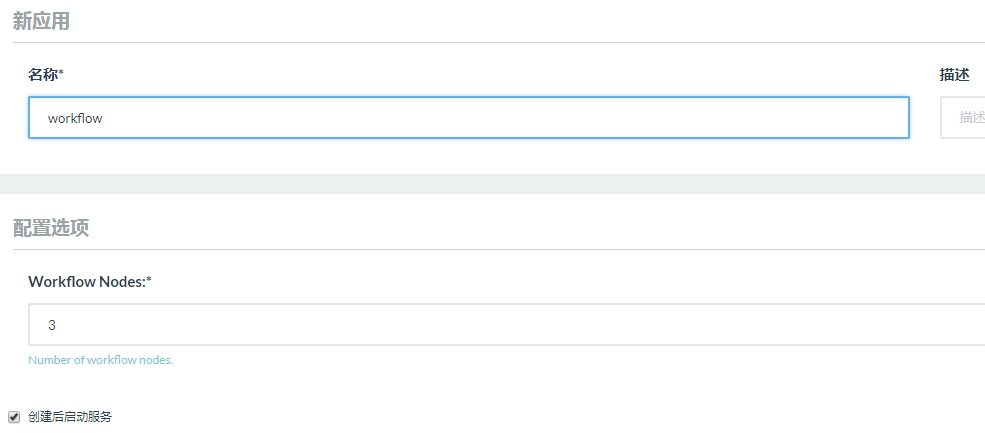
### 5.2.27 WebNode



### 5.2.28 Wopi



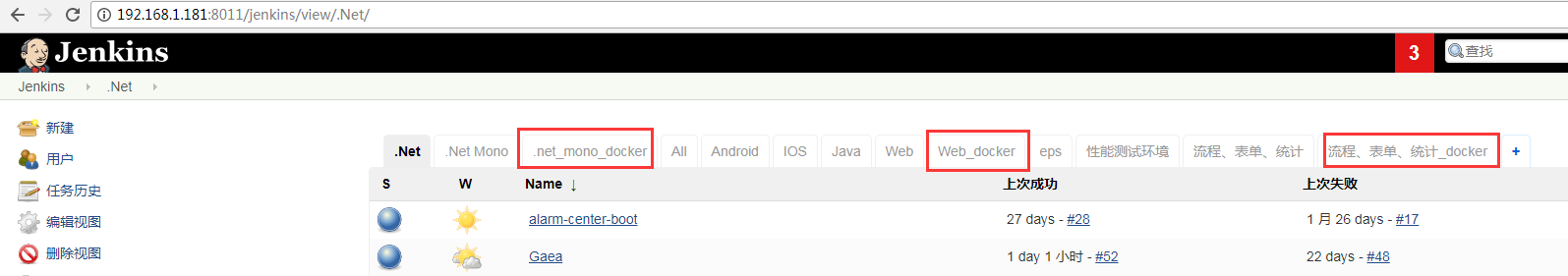
### 5.2.27 Workflow



备注：git用的是开发环境<http://192.168.1.31>

Jenkins用的是测试环境<http://192.168.1.181:8011/jenkins>

国通应用涉及到3大块内容的构建，如下





# 第6章 VPN部署

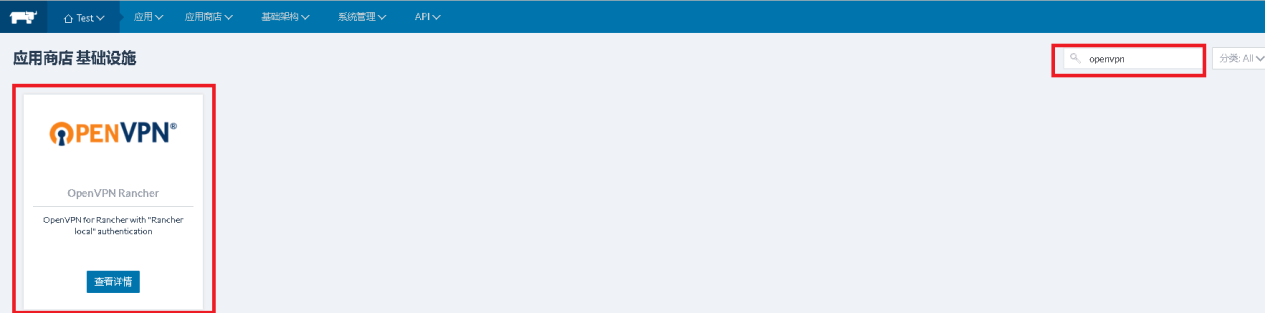
如果要修改容器中的配置信息，则必须通过vpn方式进行连接。否则，无法进行修改。

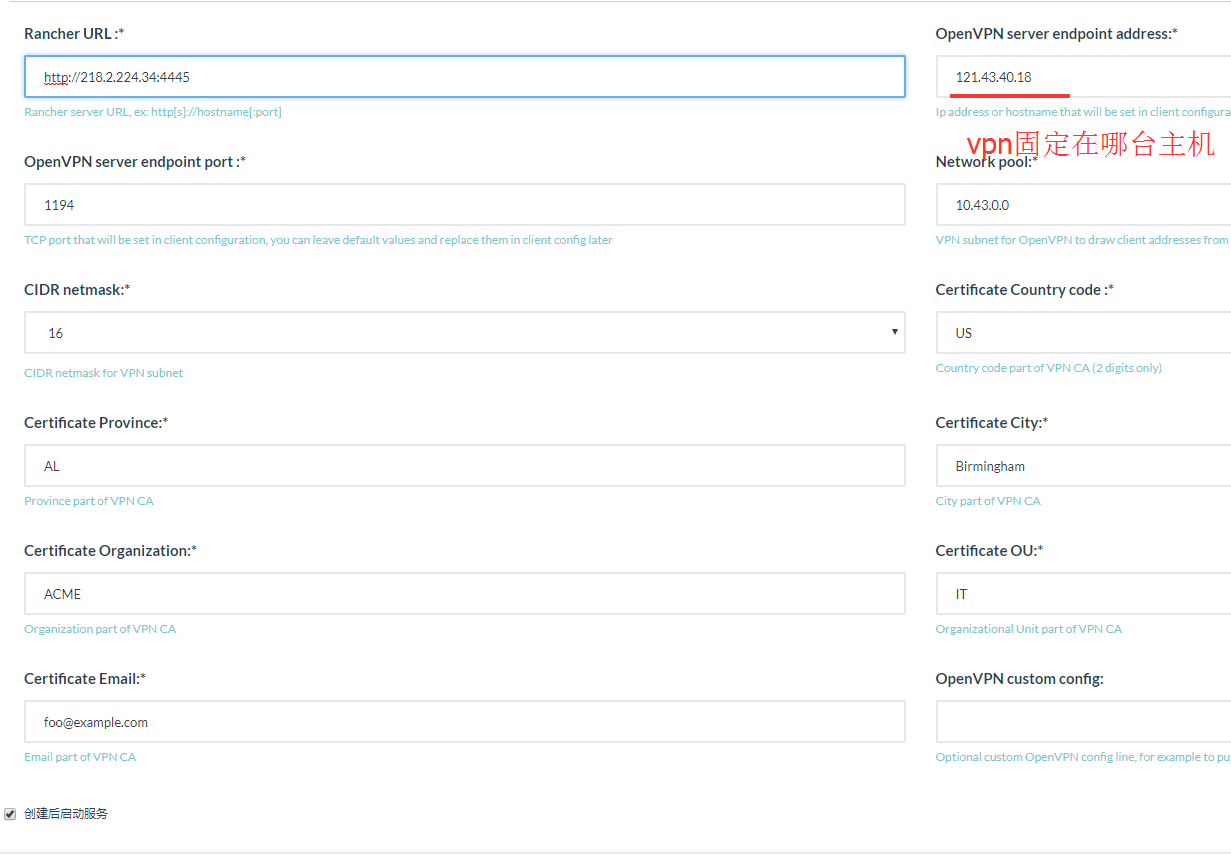
## 6.1 OpenVPN GUI安装

双击openvpn-install-2.4.2-I601.exe文件，进行默认安装。

## 6.2 Openvpn-rancher安装

应用商店-》基础设施，搜索框输入openvpn。





Rancher URL:填写rancher访问地址

Open VPN server endpoint address:填写OpenVPN server安装的主机IP。

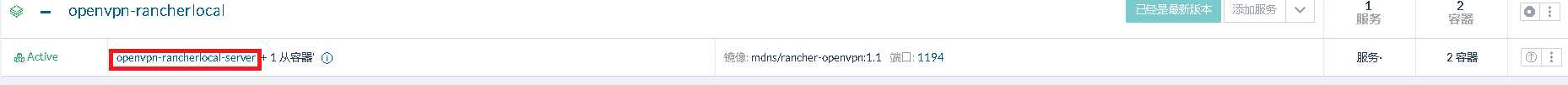
Host Label to Vpn Tags：填写运行实例的机器标签

## 6.3 配置client.ovpn

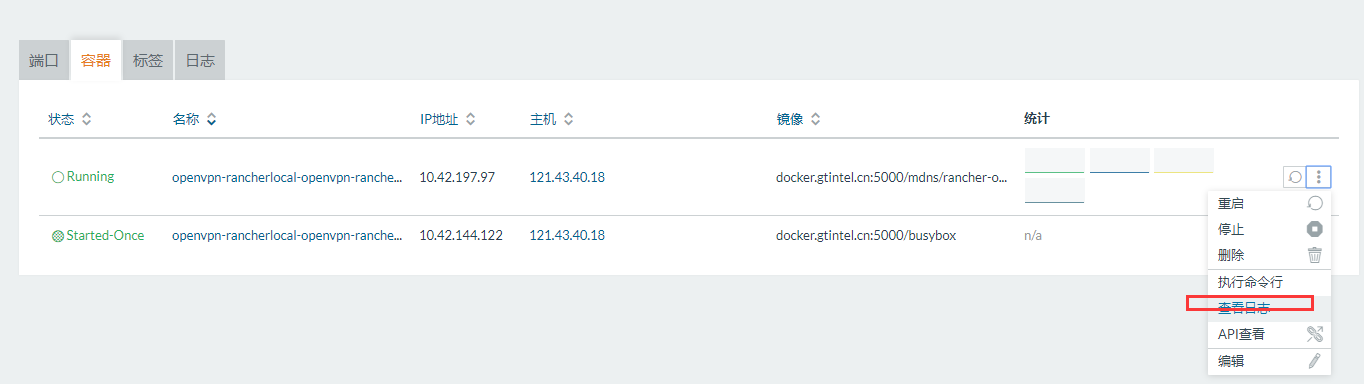
1. 在安装目录下，如C:\Program Files\OpenVPN\config，新建client.ovpn配置文件



1. 进入“应用”菜单下，双击 openvpn-rancherlocal-server容器



（3）点击running状态的openvpn-rancerlocal..后面的“查看日志”



1. 拷贝日志中的如下内容，进行整理，并将整理后的内容拷贝粘贴到client.ovpn文件中

整理前：

2017/7/4 下午4:54:31remote 192.168.3.12 1194

2017/7/4 下午4:54:31client

2017/7/4 下午4:54:31dev tun

2017/7/4 下午4:54:31proto tcp

2017/7/4 下午4:54:31remote-random

2017/7/4 下午4:54:31resolv-retry infinite

2017/7/4 下午4:54:31cipher AES-128-CBC

2017/7/4 下午4:54:31auth SHA1

2017/7/4 下午4:54:31nobind

2017/7/4 下午4:54:31link-mtu 1500

2017/7/4 下午4:54:31persist-key

2017/7/4 下午4:54:31persist-tun

2017/7/4 下午4:54:31comp-lzo

2017/7/4 下午4:54:31verb 3

2017/7/4 下午4:54:31auth-user-pass

2017/7/4 下午4:54:31auth-retry interact

2017/7/4 下午4:54:31ns-cert-type server

2017/7/4 下午4:54:31<ca>

2017/7/4 下午4:54:31-----BEGIN CERTIFICATE-----

2017/7/4 下午4:54:31MIIEmzCCA4OgAwIBAgIJAI2+h4UwOMLZMA0GCSqGSIb3DQEBCwUAMIGPMQswCQYD

2017/7/4 下午4:54:31VQQGEwJVUzELMAkGA1UECBMCQUwxEzARBgNVBAcTCkJpcm1pbmdoYW0xDTALBgNV

2017/7/4 下午4:54:31BAoTBEFDTUUxCzAJBgNVBAsTAklUMRAwDgYDVQQDEwdBQ01FIENBMRAwDgYDVQQp

2017/7/4 下午4:54:31EwdFYXN5UlNBMR4wHAYJKoZIhvcNAQkBFg9mb29AZXhhbXBsZS5jb20wHhcNMTcw

2017/7/4 下午4:54:31NzA0MDg1MzE1WhcNMjcwNzAyMDg1MzE1WjCBjzELMAkGA1UEBhMCVVMxCzAJBgNV

2017/7/4 下午4:54:31BAgTAkFMMRMwEQYDVQQHEwpCaXJtaW5naGFtMQ0wCwYDVQQKEwRBQ01FMQswCQYD

2017/7/4 下午4:54:31VQQLEwJJVDEQMA4GA1UEAxMHQUNNRSBDQTEQMA4GA1UEKRMHRWFzeVJTQTEeMBwG

2017/7/4 下午4:54:31CSqGSIb3DQEJARYPZm9vQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC

2017/7/4 下午4:54:31AQ8AMIIBCgKCAQEAtwkBoSeYkZRevXkyWOdOvI0EjGSns56QgZ3dE1lEo2zsE13Q

2017/7/4 下午4:54:31GUO46lpdzgMJO1gX8qc0j0RqtZeE4BD1XGBuS9369fbFiWyNXFsMI/BtAfPZQ9lw

2017/7/4 下午4:54:31rCWXniW73IH/B75uTMDV5OGPdyt3NREjBJNbGVwcHA8/nlXdr7pYaopIPtEcvMPw

2017/7/4 下午4:54:31Yxg+kq/RxDYX2Grikw7ztLkgWV5XJoauCZa8Fbj86ik+hTzTeobmRgSrSLqXV8+8

2017/7/4 下午4:54:3193F91O20ygVARiQ9LQRGZ8PQmiBmTrKyFhVkdLLalIIKXCFHzoAz+Yxyo+6hR+Bz

2017/7/4 下午4:54:31VUYP6EcTSvGcV0Ukv0RWhAATCXsP7akoKhUXbQIDAQABo4H3MIH0MB0GA1UdDgQW

2017/7/4 下午4:54:31BBQ9eC8TIJR1gnjOKShv8v1h2i8ijTCBxAYDVR0jBIG8MIG5gBQ9eC8TIJR1gnjO

2017/7/4 下午4:54:31KShv8v1h2i8ijaGBlaSBkjCBjzELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkFMMRMw

2017/7/4 下午4:54:31EQYDVQQHEwpCaXJtaW5naGFtMQ0wCwYDVQQKEwRBQ01FMQswCQYDVQQLEwJJVDEQ

2017/7/4 下午4:54:31MA4GA1UEAxMHQUNNRSBDQTEQMA4GA1UEKRMHRWFzeVJTQTEeMBwGCSqGSIb3DQEJ

2017/7/4 下午4:54:31ARYPZm9vQGV4YW1wbGUuY29tggkAjb6HhTA4wtkwDAYDVR0TBAUwAwEB/zANBgkq

2017/7/4 下午4:54:31hkiG9w0BAQsFAAOCAQEAen/LT6BF5RNWZMBgILLaf5RyliKbX/sw0ZLwV/22+fMQ

2017/7/4 下午4:54:31HxgrW2MyqeGJCs2hrv3QsJLO/leV6XNUP8k6sGpSAJGiiOzVy2FYmhTz2cCp5Vlo

2017/7/4 下午4:54:311WCP0KT5wWjhsJHkUwq8P7zqazKdQ8vk1znjY+GH4H2a2pTCYQluMmBhwJjMOCBZ

2017/7/4 下午4:54:31dBf/rVee0LCqW6efeugkA2XqM6Bmo3i/4AiDisxeYlb3VSOP7ZzIIS5TclkNRASO

2017/7/4 下午4:54:31l7gsresANfQmBFUlsyByEx2/tqHH2BTYLEjs/Fk+3WRQCPAvziCL8uinOGcmNT8i

2017/7/4 下午4:54:31ezOuieXQvg3s3M34f0fEo2TVkauFESLz5yT90MfFww==

2017/7/4 下午4:54:31-----END CERTIFICATE-----

2017/7/4 下午4:54:31</ca>

整理后：

remote 192.168.3.24 1194 //修改此处的IP地址为openvpn-rancher实际安装主机

client

dev tun

proto tcp

remote-random

resolv-retry infinite

cipher AES-128-CBC

auth SHA1

nobind

link-mtu 1500

persist-key

persist-tun

comp-lzo

verb 3

auth-user-pass

auth-retry interact

ns-cert-type server

<ca>

-----BEGIN CERTIFICATE-----

MIIEmzCCA4OgAwIBAgIJAI2+h4UwOMLZMA0GCSqGSIb3DQEBCwUAMIGPMQswCQYD

VQQGEwJVUzELMAkGA1UECBMCQUwxEzARBgNVBAcTCkJpcm1pbmdoYW0xDTALBgNV

BAoTBEFDTUUxCzAJBgNVBAsTAklUMRAwDgYDVQQDEwdBQ01FIENBMRAwDgYDVQQp

EwdFYXN5UlNBMR4wHAYJKoZIhvcNAQkBFg9mb29AZXhhbXBsZS5jb20wHhcNMTcw

NzA0MDg1MzE1WhcNMjcwNzAyMDg1MzE1WjCBjzELMAkGA1UEBhMCVVMxCzAJBgNV

BAgTAkFMMRMwEQYDVQQHEwpCaXJtaW5naGFtMQ0wCwYDVQQKEwRBQ01FMQswCQYD

VQQLEwJJVDEQMA4GA1UEAxMHQUNNRSBDQTEQMA4GA1UEKRMHRWFzeVJTQTEeMBwG

CSqGSIb3DQEJARYPZm9vQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOC

AQ8AMIIBCgKCAQEAtwkBoSeYkZRevXkyWOdOvI0EjGSns56QgZ3dE1lEo2zsE13Q

GUO46lpdzgMJO1gX8qc0j0RqtZeE4BD1XGBuS9369fbFiWyNXFsMI/BtAfPZQ9lw

rCWXniW73IH/B75uTMDV5OGPdyt3NREjBJNbGVwcHA8/nlXdr7pYaopIPtEcvMPw

Yxg+kq/RxDYX2Grikw7ztLkgWV5XJoauCZa8Fbj86ik+hTzTeobmRgSrSLqXV8+8

93F91O20ygVARiQ9LQRGZ8PQmiBmTrKyFhVkdLLalIIKXCFHzoAz+Yxyo+6hR+Bz

VUYP6EcTSvGcV0Ukv0RWhAATCXsP7akoKhUXbQIDAQABo4H3MIH0MB0GA1UdDgQW

BBQ9eC8TIJR1gnjOKShv8v1h2i8ijTCBxAYDVR0jBIG8MIG5gBQ9eC8TIJR1gnjO

KShv8v1h2i8ijaGBlaSBkjCBjzELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkFMMRMw

EQYDVQQHEwpCaXJtaW5naGFtMQ0wCwYDVQQKEwRBQ01FMQswCQYDVQQLEwJJVDEQ

MA4GA1UEAxMHQUNNRSBDQTEQMA4GA1UEKRMHRWFzeVJTQTEeMBwGCSqGSIb3DQEJ

ARYPZm9vQGV4YW1wbGUuY29tggkAjb6HhTA4wtkwDAYDVR0TBAUwAwEB/zANBgkq

hkiG9w0BAQsFAAOCAQEAen/LT6BF5RNWZMBgILLaf5RyliKbX/sw0ZLwV/22+fMQ

HxgrW2MyqeGJCs2hrv3QsJLO/leV6XNUP8k6sGpSAJGiiOzVy2FYmhTz2cCp5Vlo

1WCP0KT5wWjhsJHkUwq8P7zqazKdQ8vk1znjY+GH4H2a2pTCYQluMmBhwJjMOCBZ

dBf/rVee0LCqW6efeugkA2XqM6Bmo3i/4AiDisxeYlb3VSOP7ZzIIS5TclkNRASO

l7gsresANfQmBFUlsyByEx2/tqHH2BTYLEjs/Fk+3WRQCPAvziCL8uinOGcmNT8i

ezOuieXQvg3s3M34f0fEo2TVkauFESLz5yT90MfFww==

-----END CERTIFICATE-----

</ca>

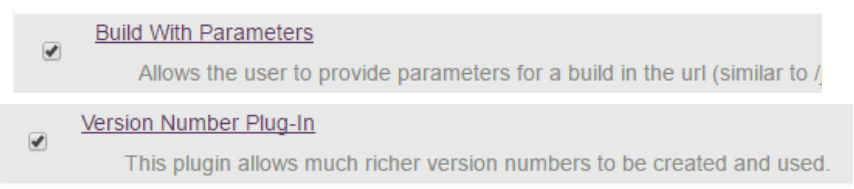
1. 运行OpenVPN GUI程序，右键点击“连接”，输入用户名和密码（该处的用户名和密码与rancher访问的账号相同）
2. Vpn连接成功后，当前对话框会消失，右下角出现连接成功图标。如果连接失败，则当前连接对话框一直存在,会打印错误信息。

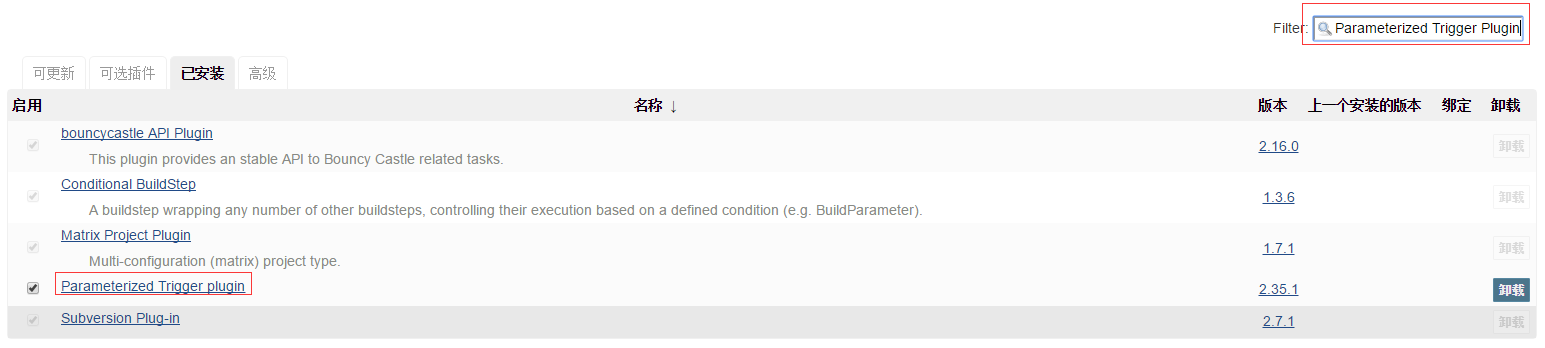


# 第7章 Jenkins配置

## 7.1 安装插件

需要安装插件：Build With Parameters、Version Number Plug-In、Parameterized Trigger Plugin





1. 登录<http://192.168.1.181:8011/jenkins/>，
2. 输入账号和密码: admin/86589089181
3. 访问菜单【系统管理】——【管理插件】-【可选插件】，过滤框分别输入要安装的插件，勾选记录，点击【直接安装】按钮。

## 7.2 安装docker依赖环境

在编译的jenkins主机上安装docker和mono服务，用来完成docker镜像的生成。

#### 7.2.1 docker部署

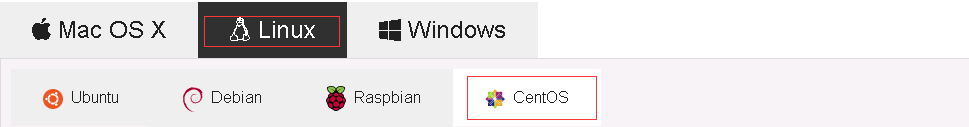
参考“第2章 Docker部署”完成docker安装。

#### 7.2.2 mono部署

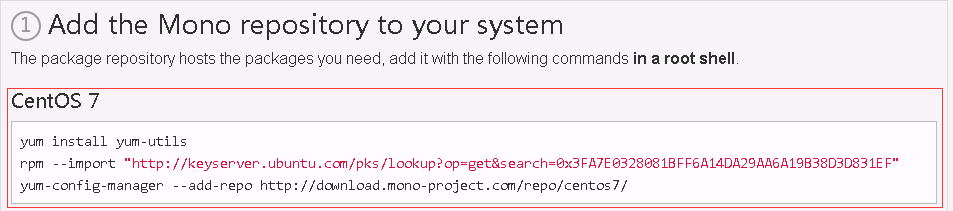
Mono编译Xbuild工具。

步骤1：打开mono官网 <http://www.mono-project.com/download/#download-lin-centos>

步骤2：选择linux-centos主机



步骤3：执行下列命令



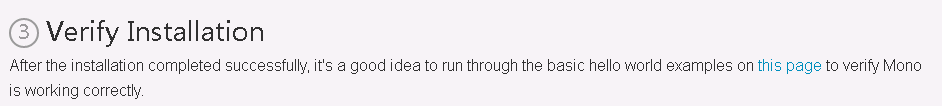
# yum install yum-utils

# rpm --import "http://keyserver.ubuntu.com/pks/lookup?op=get&search=0x3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF"

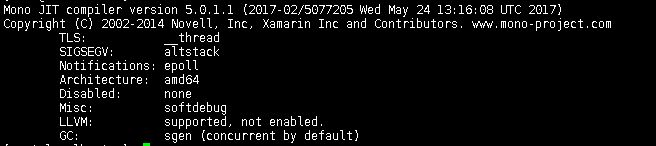
# yum-config-manager --add-repo <http://download.mono-project.com/repo/centos7/>



# yum install mono-devel



# 执行mono --version进行验证，如果OK，则说明安装成功。



#### 7.2.3 nuget部署

Nuget起到还原程序包的作用。

登录<https://www.nuget.org/，下载nuget.exe>



在/home/gtarch/目录下，新建nuget目录，将nuget.exe文件拷贝至/home/gtarch/nuget目录。

#### 7.2.4 Jenkins账户加入docker用户组

（1）将jenkins/gtarch用户加入docker组

#sudo gpasswd -a jenkins docker

#sudo gpasswd -a gtarch docker

# sudo usermod -aG docker $(whoami)

（2）重启服务

# sudo service docker restart

# sudo service jenkins restart

# systemctl daemon-reload

#### 7.2.5 登录docker注册服务器

#docker login 192.168.1.129:5000  //仓库地址

用户名：admin

密码：admin123

登录成功，则提示Login Succeeded

登录成功后，在/root/.docker/config.json文件中保存注册信息，如下：

[root@localhost .docker]# vi config.json

{

"auths": {

"192.168.1.129:5000": {

"auth": "YWRtaW46YWRtaW4xMjM="

}

}

}

如果登录失败，则需要查看当前目录下是否已存在注册信息，可以通过docker logout删除注册信息，重新注册。或者重新启动docker服务（systemctl restart docker）

## 构建项目

#### Gaea.Comment.Owin\_Mono\_Catalog（评论\_配置项）

**.Net服务（例如：字典、人组、名片、朋友圈等）构建，可以参考该项目构建模式，参数基本上不需要做太大修改。**

评论配置项通过构建评论项目来自动执行，无需手工构建。该项目主要用来打包docker镜像，并推送docker镜像。

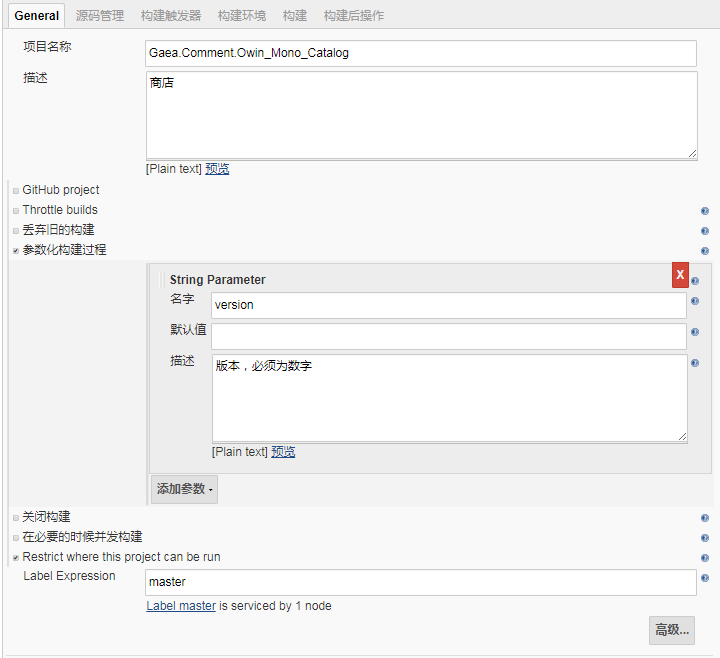
1. 点击【新建】菜单，填写项目名称Gaea.Comment.Owin\_Mono\_Catalog

copy from 填写Gaea.Comment.Owin。

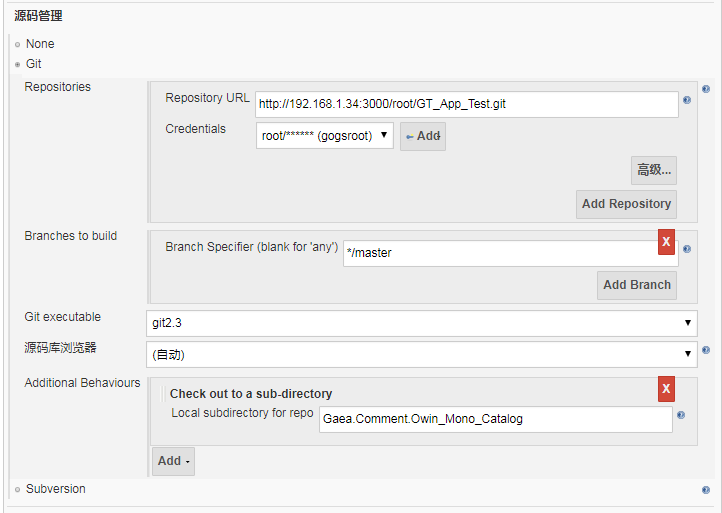


（2）点击【配置】，修改配置项，如下：

* **General**



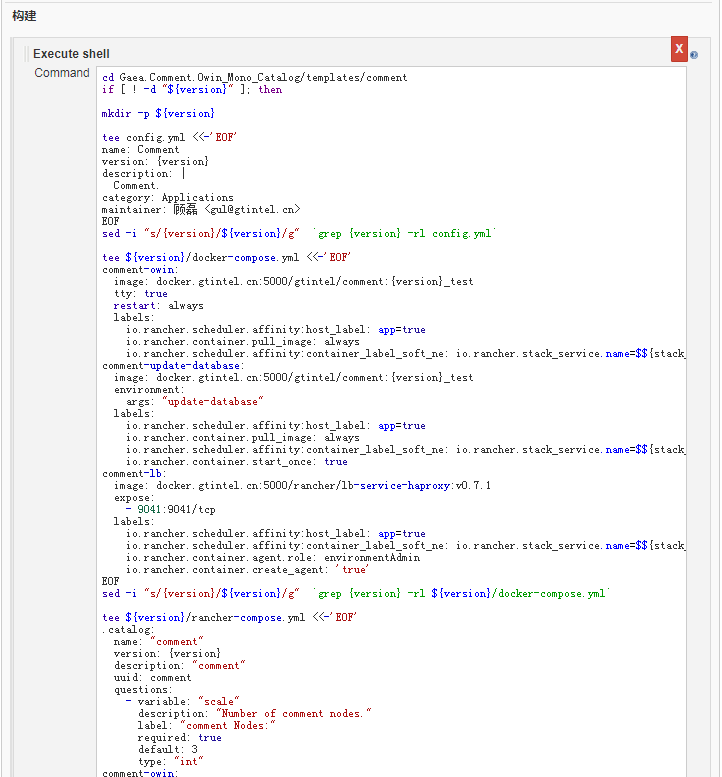
* **源码管理**



* **构建触发器/构建环境**



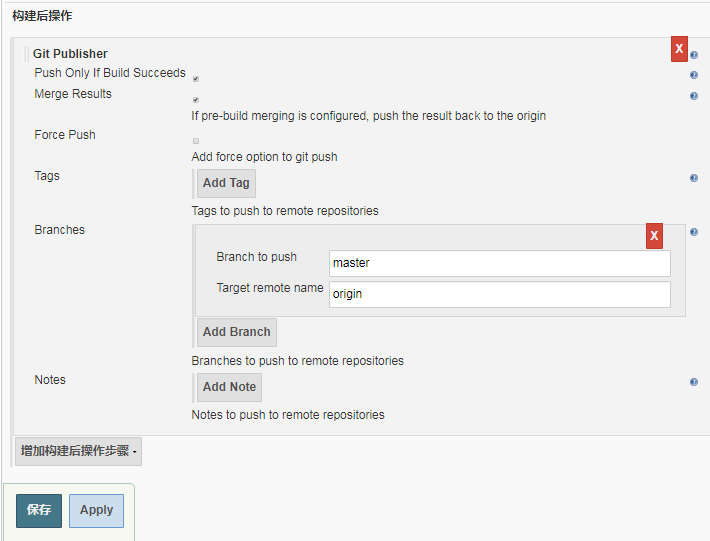
* **构建**



Execute shell Command内容如下：

|  |
| --- |
| cd Gaea.Comment.Owin\_Mono\_Catalog/templates/comment  if [ ! -d "${version}" ]; then  mkdir -p ${version}  tee config.yml <<-'EOF'  name: Comment  version: {version}  description: |  Comment.  category: Applications  maintainer: 顾磊 <gul@gtintel.cn>  EOF  sed -i "s/{version}/${version}/g" `grep {version} -rl config.yml`  tee ${version}/docker-compose.yml <<-'EOF'  comment-owin:  image: docker.gtintel.cn:5000/gtintel/comment:{version}\_test  tty: true  restart: always  labels:  io.rancher.scheduler.affinity:host\_label: app=true  io.rancher.container.pull\_image: always  io.rancher.scheduler.affinity:container\_label\_soft\_ne: io.rancher.stack\_service.name=$${stack\_name}/$${service\_name}  comment-update-database:  image: docker.gtintel.cn:5000/gtintel/comment:{version}\_test  environment:  args: "update-database"  labels:  io.rancher.scheduler.affinity:host\_label: app=true  io.rancher.container.pull\_image: always  io.rancher.scheduler.affinity:container\_label\_soft\_ne: io.rancher.stack\_service.name=$${stack\_name}/$${service\_name}  io.rancher.container.start\_once: true  comment-lb:  image: docker.gtintel.cn:5000/rancher/lb-service-haproxy:v0.7.1  expose:  - 9041:9041/tcp  labels:  io.rancher.scheduler.affinity:host\_label: app=true  io.rancher.scheduler.affinity:container\_label\_soft\_ne: io.rancher.stack\_service.name=$${stack\_name}/$${service\_name}  io.rancher.container.agent.role: environmentAdmin  io.rancher.container.create\_agent: 'true'  EOF  sed -i "s/{version}/${version}/g" `grep {version} -rl ${version}/docker-compose.yml`  tee ${version}/rancher-compose.yml <<-'EOF'  .catalog:  name: "comment"  version: {version}  description: "comment"  uuid: comment  questions:  - variable: "scale"  description: "Number of comment nodes."  label: "comment Nodes:"  required: true  default: 3  type: "int"  comment-owin:  scale: ${scale}  health\_check:  response\_timeout: 2000  healthy\_threshold: 2  port: 9041  unhealthy\_threshold: 3  initializing\_timeout: 60000  interval: 2000  reinitializing\_timeout: 60000  comment-lb:  scale: 1  start\_on\_create: true  lb\_config:  certs: []  port\_rules:  - priority: 1  protocol: http  service: comment-owin  source\_port: 9041  target\_port: 9041  health\_check:  response\_timeout: 2000  healthy\_threshold: 2  port: 42  unhealthy\_threshold: 3  initializing\_timeout: 60000  interval: 2000  reinitializing\_timeout: 60000  EOF  sed -i "s/{version}/${version}/g" `grep {version} -rl ${version}/rancher-compose.yml`  git add ${version}/docker-compose.yml  git add ${version}/rancher-compose.yml  git commit -am "commit ${version} Catalog"  fi |

* **构建后操作**



* **执行构建**

构建成功后，在添加应用界面可以看到最新的模板版本。



#### Gaea.Comment.Owin\_Mono（评论）

**.Net服务（例如：字典、人组、名片、朋友圈等）构建，可以参考该项目构建模式，参数基本上不需要做太大修改。**

1. 点击【新建】菜单，填写项目名称Gaea.Comment.Owin\_Mono，

copy from 填写Gaea.Comment.Owin

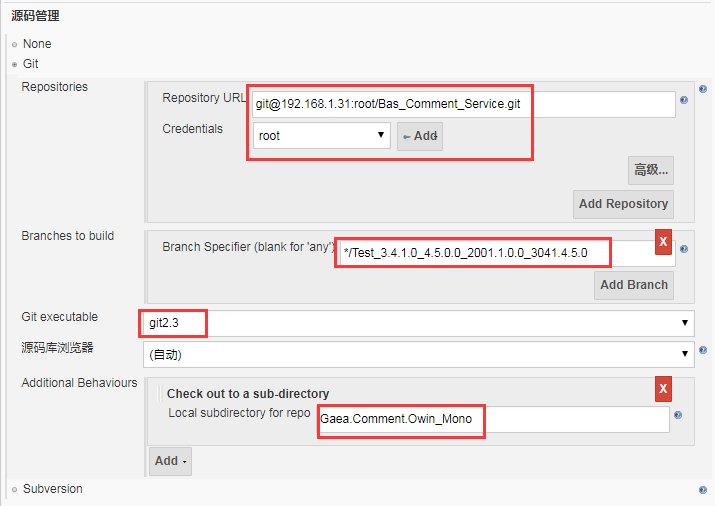


（2）点击【配置】修改配置项

* **General项：**



* **源码管理项**



* **构建触发器 /触发环境：**



* **构建：**



1#执行shell脚本，Nuget还原包：

mono /home/gtarch/nuget/nuget.exe locals all -clear

mono /home/gtarch/nuget/nuget.exe restore ./Gaea.Comment.Owin\_Mono/src/Gaea.Comment.sln -nocache -source http://192.168.1.129:8081/repository/NugetDevp/

2#执行shell脚本，Mono编译Xbuild工具：

MONO\_IOMAP=all xbuild /p:WarningLevel=4 /p:NoWarn=1591 /p:AllowUntrustedCertificate=True /p:Configuration=Release /p:ExcludeGeneratedDebugSymbol=false /p:ExcludeXmlAssemblyFiles=false /p:FilesToIncludeForPublish=AllFilesInProjectFolder Gaea.Comment.Owin\_Mono/src/Gaea.Comment.sln

3#执行shell脚本，利用dockerfile打包Docker镜像，推送镜像：

cd Gaea.Comment.Owin\_Mono/src/Gaea.Comment.Owin/bin/Release

docker build -t 192.168.1.129:5000/gtintel/comment:${version}\_test .

docker push 192.168.1.129:5000/gtintel/comment:${version}

# 第8章 rancher环境信息

**dubbo管理工具**

<http://admin.dubbo:8080/>

root

admin123

**数据库管理工具**

<http://adminer.adminer/>

<http://mongoclient.adminer:3000/>

**调用栈管理工具**

<http://pinpoint-web.pinpoint:8080/#/main>

admin

admin123

**微服物聚合管理工具**

[http://kong-dashboard.kong:8080](http://kong-dashboard.kong:8080/#/config)

**文件服务管理工具**

<http://lb.minio:9000>

admin

admin123

**Swagger服务地址**

<http://swagger-ui.swagger:8080/>

**日志服务**

<http://kibana-vip.kibana>

**Kafka管理地址**

<http://kafka-manager.kafka:9000/>