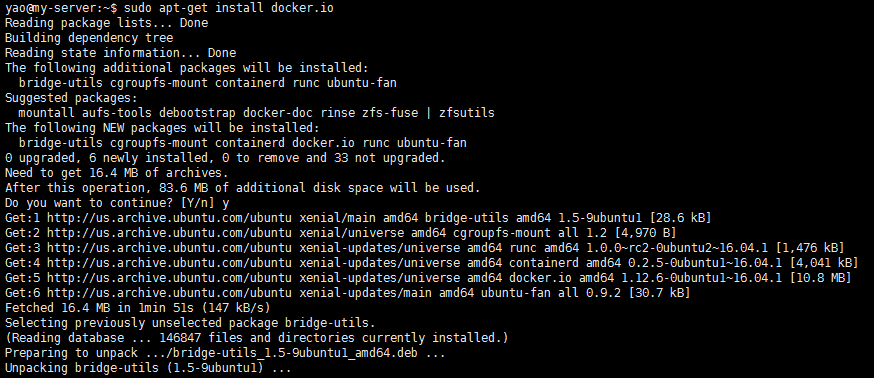
**Docker安装部署使用**

# 安装Docker

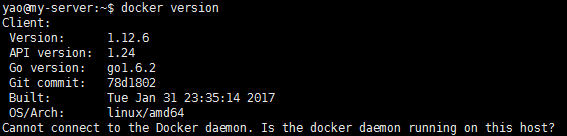
## **从官方的Ubuntu软件库安装**

sudo apt-get update

sudo apt-get install docker.io

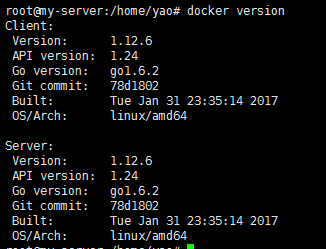


查看版本（注意这时候没有启动）

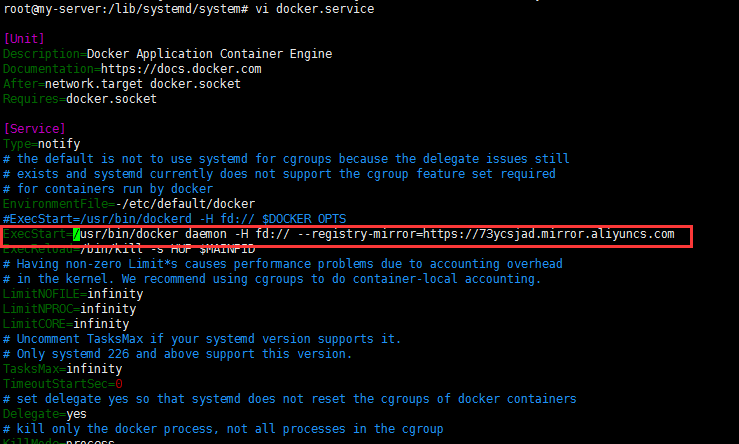


## 启动docker

service docker start



## 镜像加速度（pull加速）



1. 注册阿里云开发账号，获取镜像加速地址（<https://dev.aliyun.com/search.html>）
2. vi /lib/systemd/system/docker.service, 修改获取官方镜像地址(粘贴地址，Shift + Insert)，如下：

ExecStart=/usr/bin/docker daemon -H fd:// --registry-mirror=https://73ycsjad.mirror.aliyuncs.com

1. systemctl daemon-reload
2. systemctl restart docker

## 配置自启动

## 卸载

sudo apt-get remove docker.io

# 部署配置Rancher

## 下载并运行镜像

docker run -d --restart=unless-stopped -p 8080:8080 rancher/server

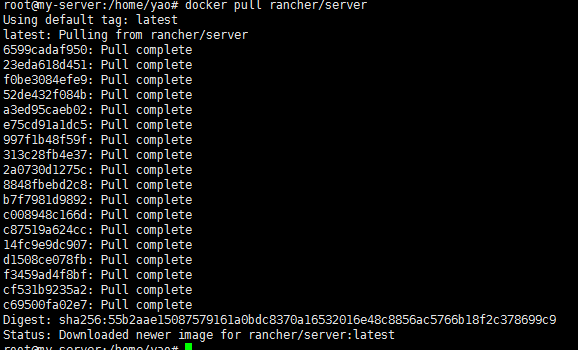
docker run -d --restart=unless-stopped \

-p 8080:8080 \

--volume /home/yao/rancher/mysql:/var/lib/mysql \

--volume /home/yao/rancher/cattle:/var/lib/cattle rancher/server

* 如果本地没有rancher镜像会自动从中心服务器拉去rancher镜像，并启动，可以通过 docker ps 进行查看，如下图：
* 如果下载失败，请多次尝试，或配置加速器；
* 最好映射出rancher对应的cattle、mysql目录，为后面的离线增加catalog准备条件；





## 手动启动(已经pull rancher/server)

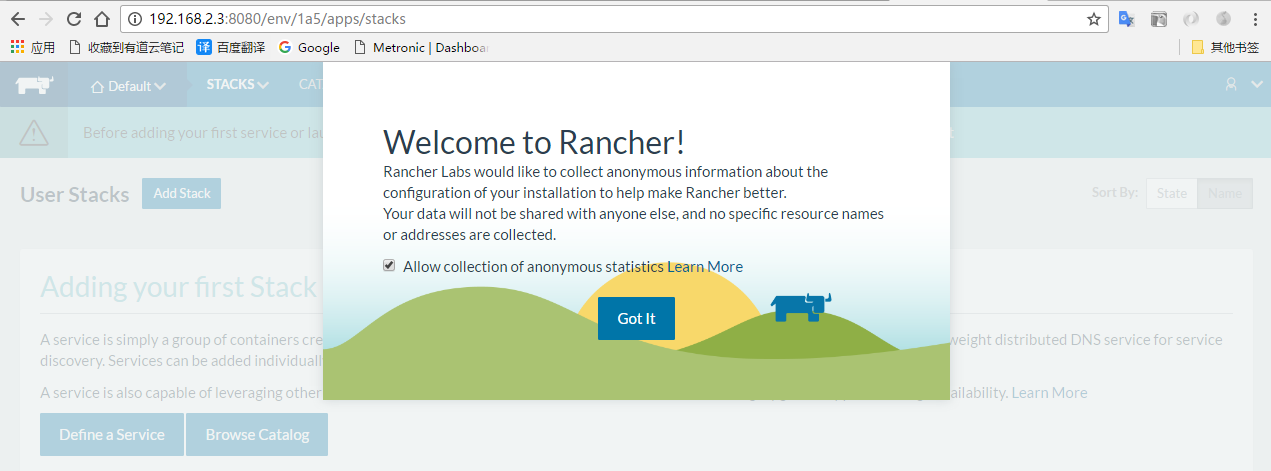
docker run -d --restart=always -p 8080:8080 rancher/server



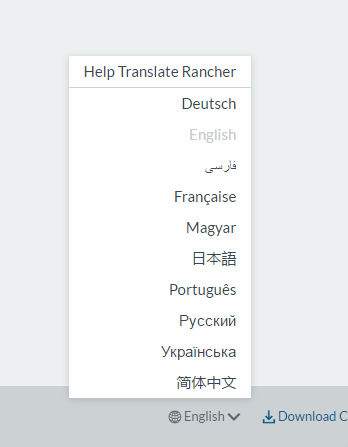


## 测试访问

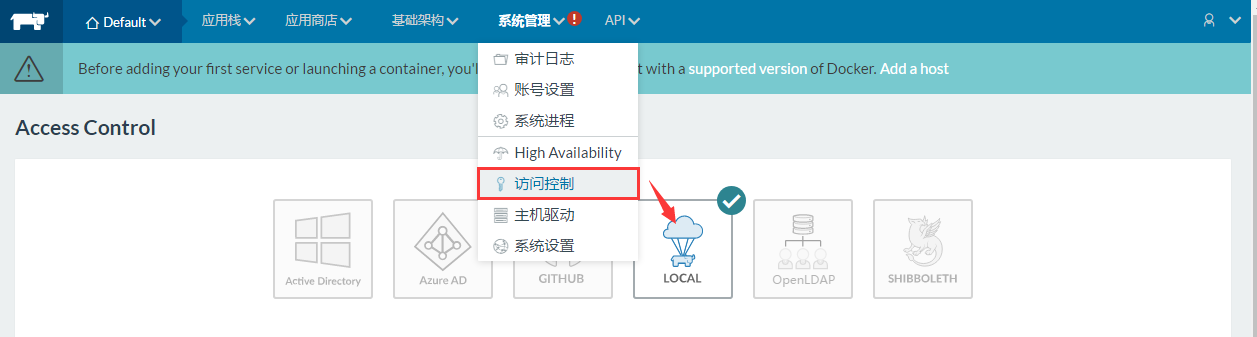
地址：<http://192.168.2.3:8080>

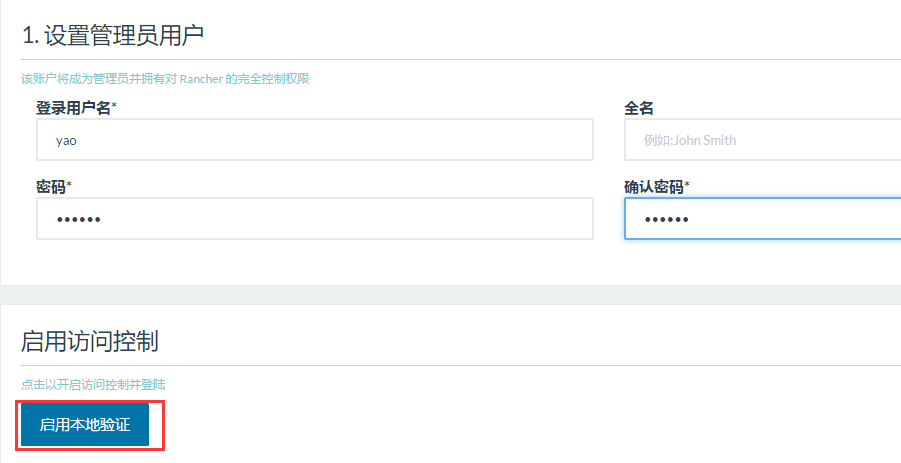


## 设置中文(右下角)

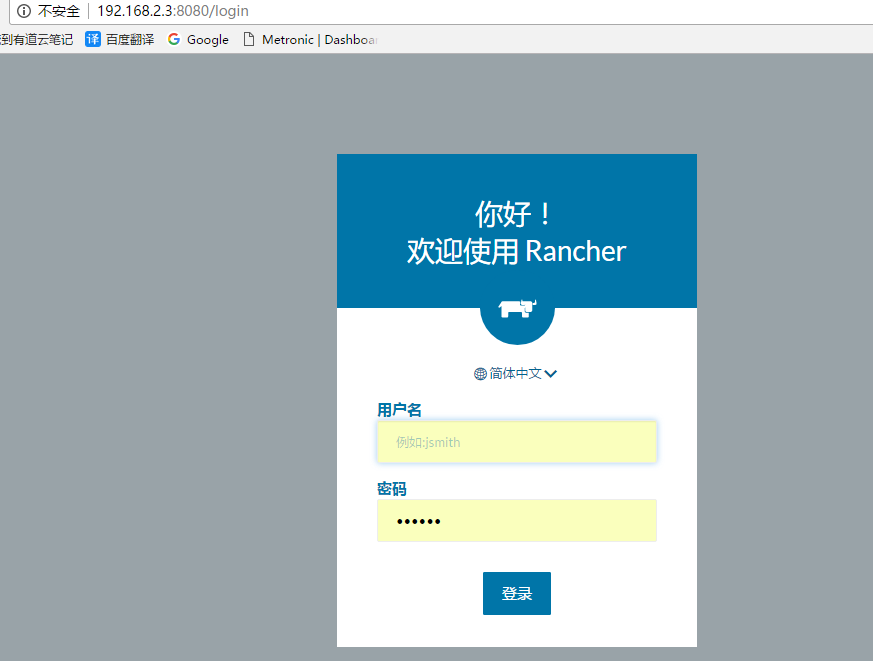


## 用户配置





**添加对应的用户名，密码即可（建议与用户一致）**



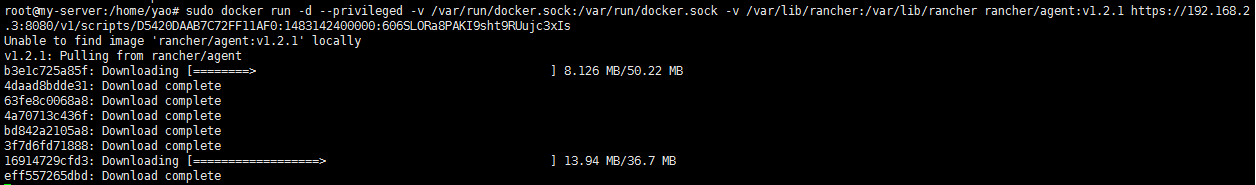
**右上角注销，即可弹出登录页面**

## 添加rancher节点

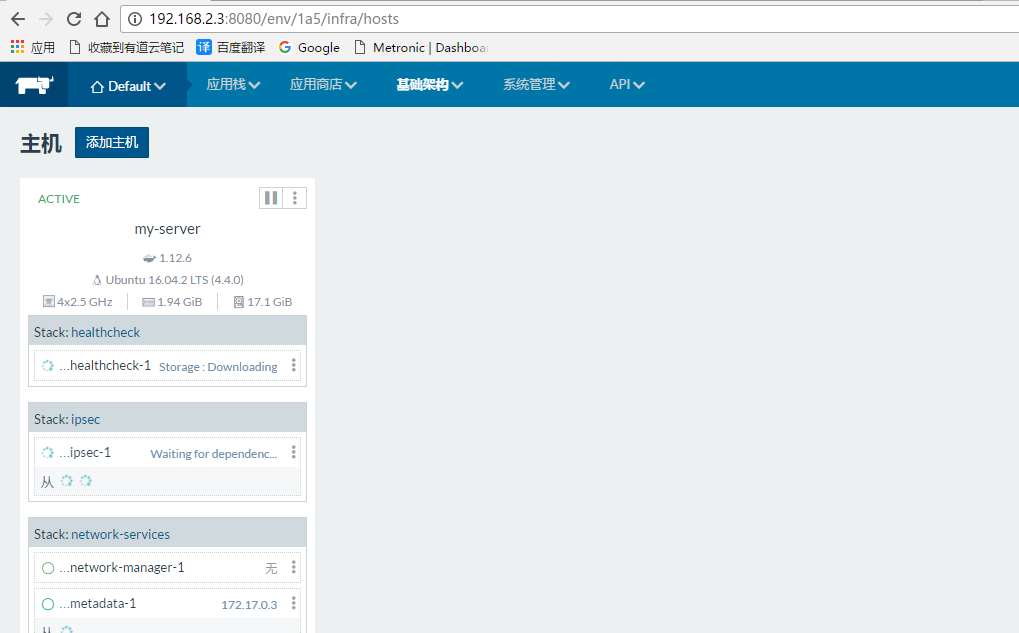




**复制以上文本，在shell端口录入**

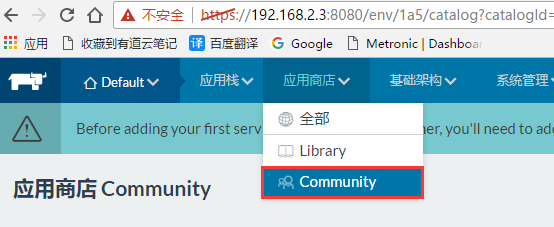
****

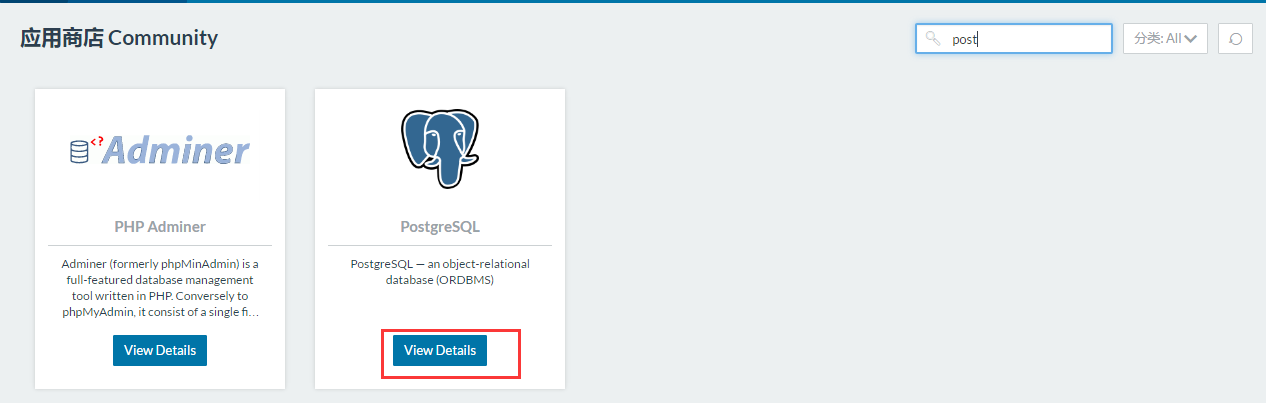
**等命令执行完成【自动下载 rancher/agent 镜像并运行】，在添加主机的页面，点击关闭**

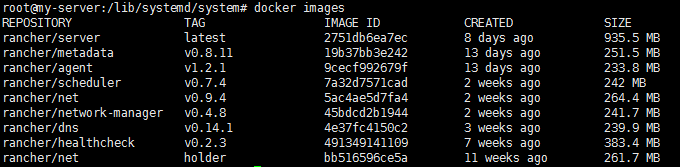


## 添加应用示例

### 根据应用商店分类，搜索对应的docker服务

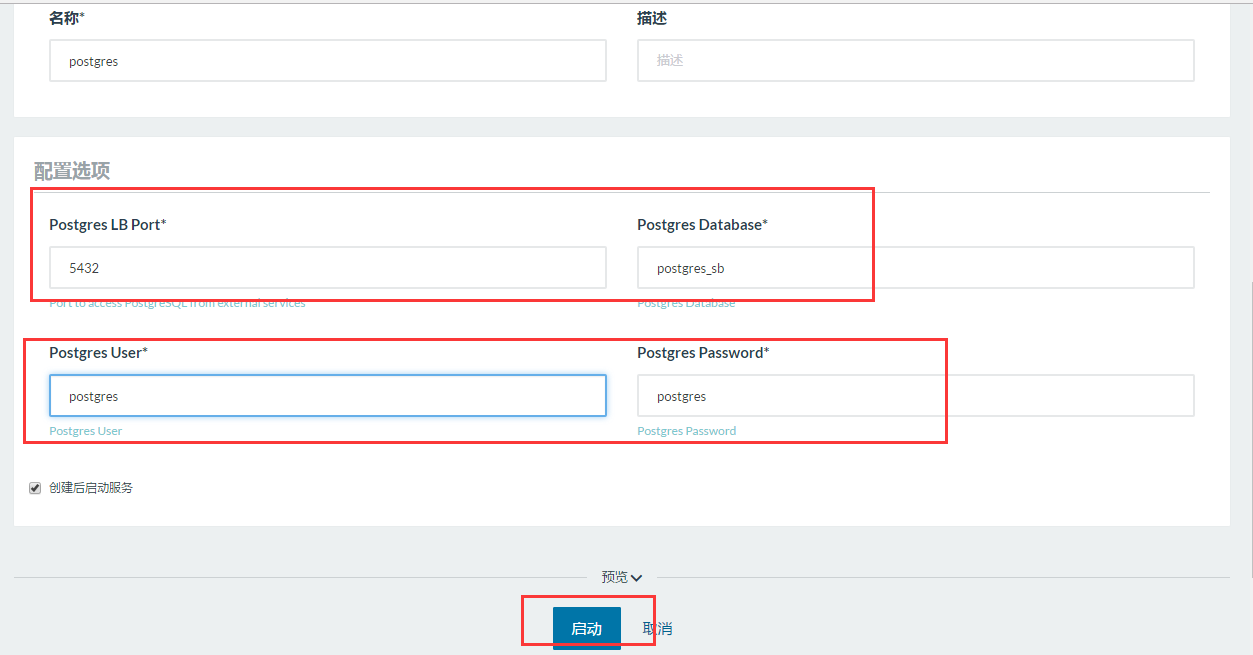


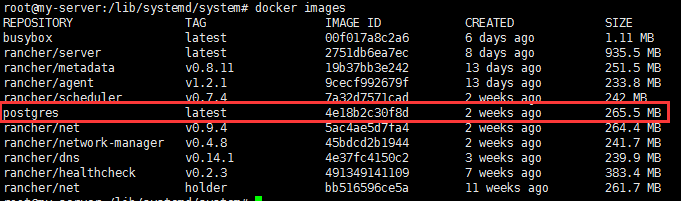




**启动postgre之前所有的镜像**

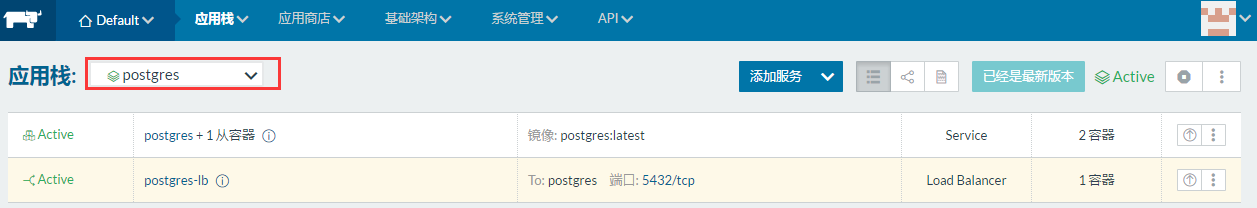
### 配置数据库端口等信息



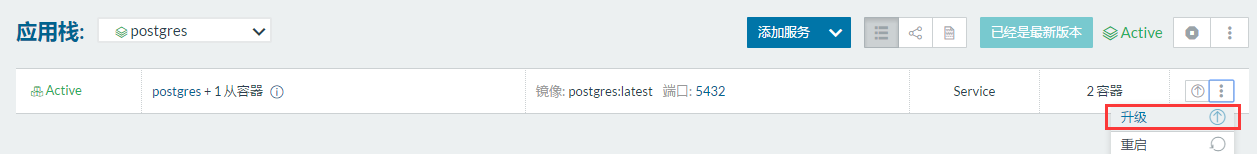


**启动后所有的镜像（原docker没有，会自动下载）**

### 启动（官方支持负载均衡）

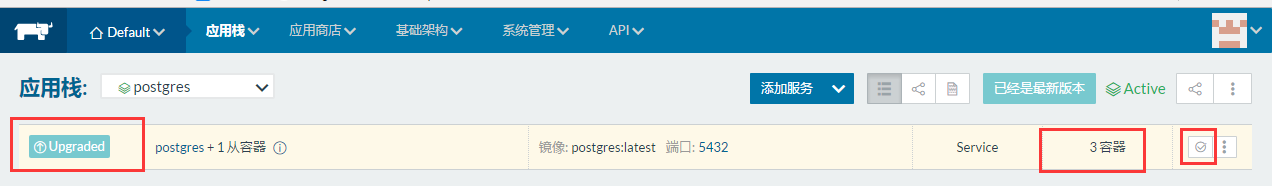


### 删除postgres-lb, 升级镜像postgres:latest，修改端口映射

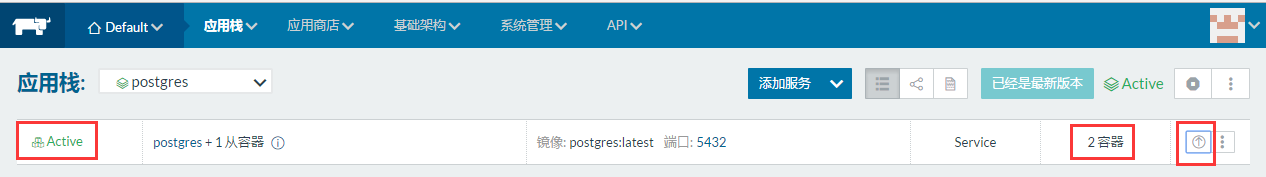


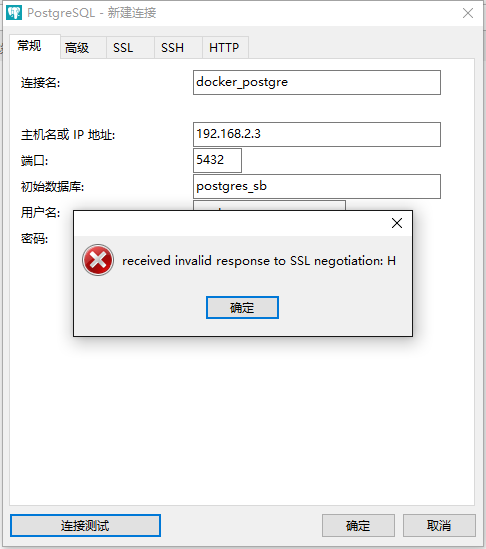


点击最下面的升级按钮（停止当前容器，并新创建一个容器）

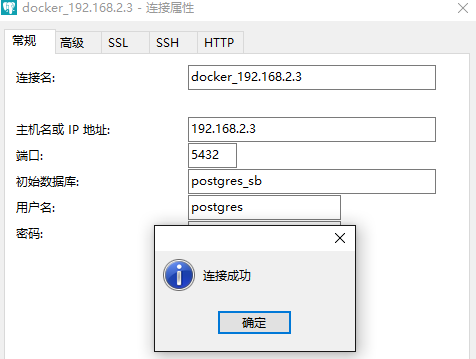


点击最右侧的升级（删除之前的容器）





**使用Navicat连接docker版的postgre失败，请检查负责均衡、端口映射**

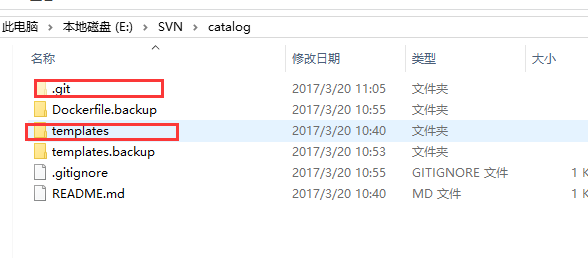


# Rancher上自定义Catalog

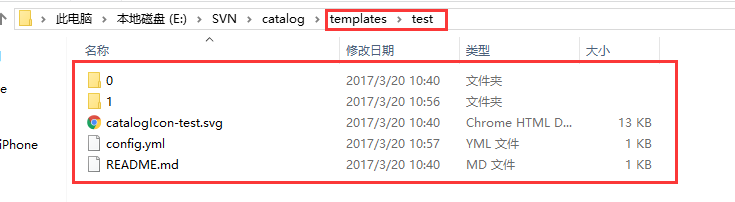
官方参照地址：<https://github.com/rancher/rancher-catalog>

## 编辑必须文件

### 文件结构



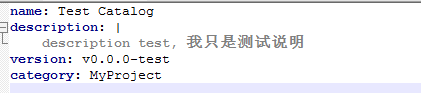
templates是根目录的第一节目录，这点很重要



test : 是容器的名称，也是这个catalog的名称，不允许重复；

0、1、2…. : 每次版本的模板文件，存放docker-compose.yml、rancher-compose.yml文件

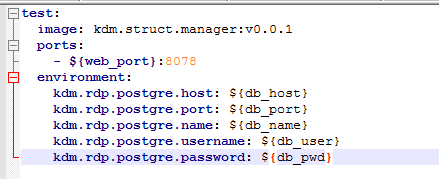
### config.xml



### catalogIcon-logo.svg

应用商量展示对应的catalog图片

### \*/docker-compose.yml



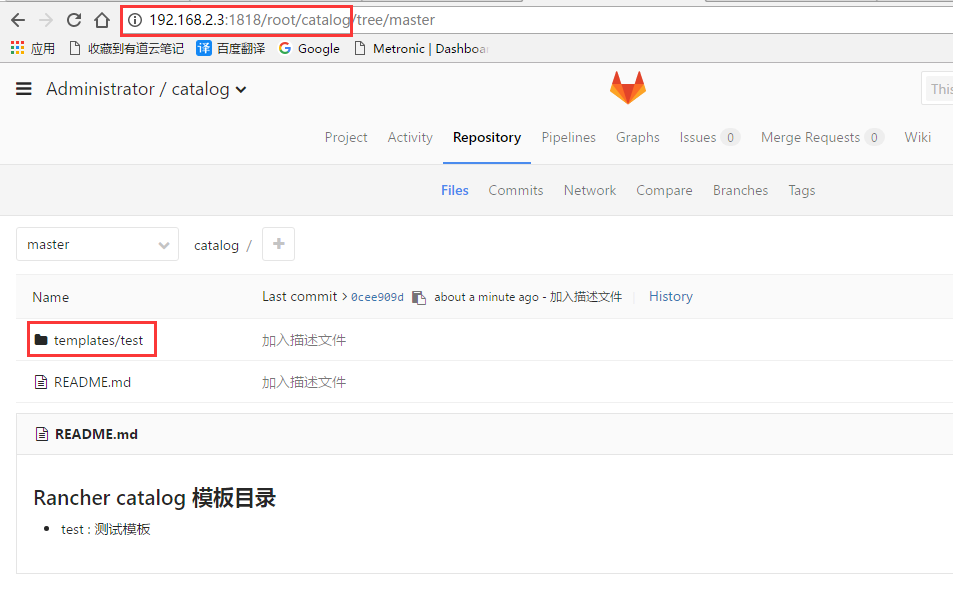
### \*/rancher-compose.yml





## 从gitlab引用

### 上传项目至公共仓库

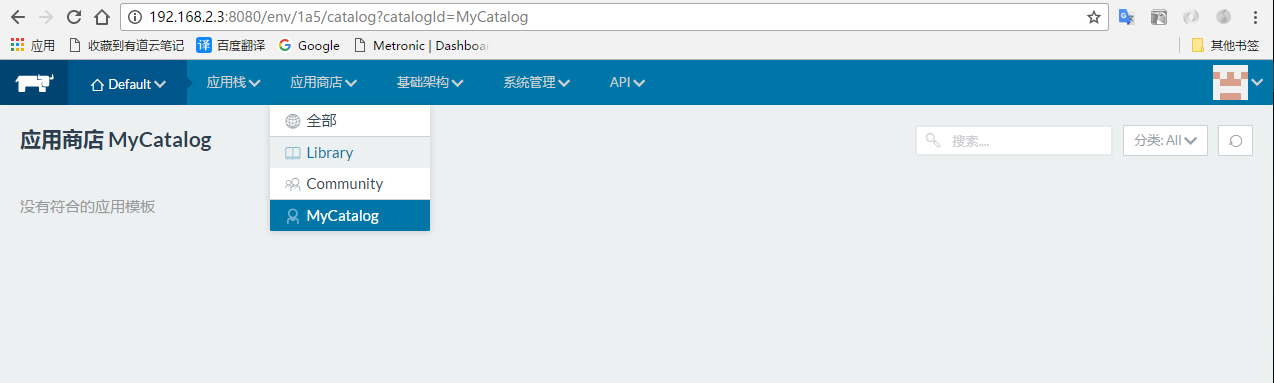


### 系统设置-添加应用商店

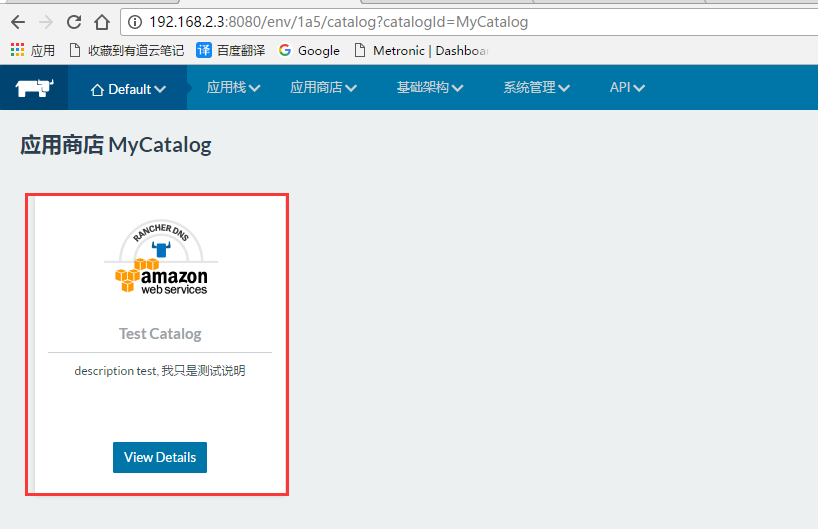




### 查看自己的商店



我测试的大概在10秒左右能刷出自己配置的Catalog，如果没有，请检查自己的配置文件；PS:可以拿官方的拿来测试；



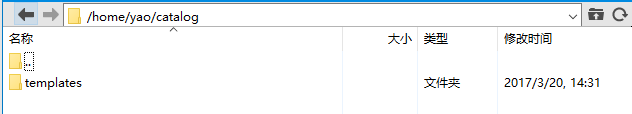
## 从本地引用

### 引用说明

将templates文件夹放到rancher可以访问的目录

### 上传templates文件夹

因为映射的文件夹只有root用户才能进行更改，且xftp等上传工具默认使用当前用户权限（非root），只能分两步进行，先时用xftp等工具上传templates、.git两个文件夹上传至夹至/home/yao/catalog目录下；.git文件夹是隐藏文件夹，使用ls -a可查看



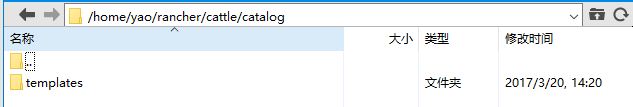


使用root用户将catalog文件夹移动到/home/yao/rancher/cattle/catalog文件夹下

mv /home/yao/catalog /home/yao/rancher/cattle/catalog

rm -R /home/yao/rancher/cattle/catalog

mkdir /home/yao/catalog

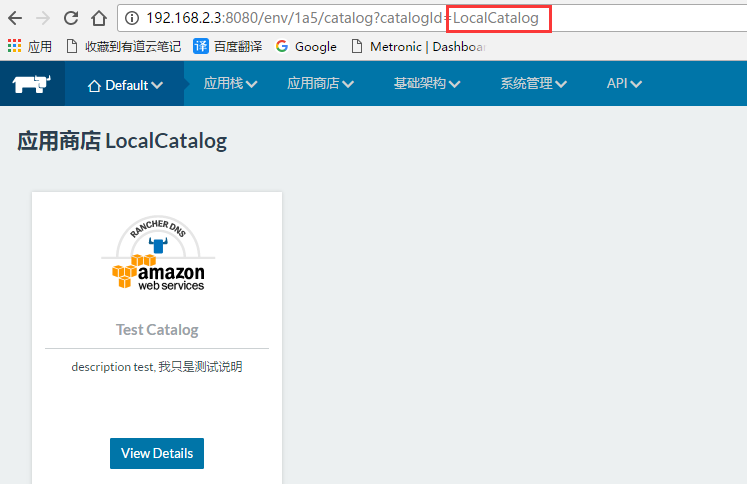


### 添加商店



注意：这里的地址是/var/lib/cattle/catalog/

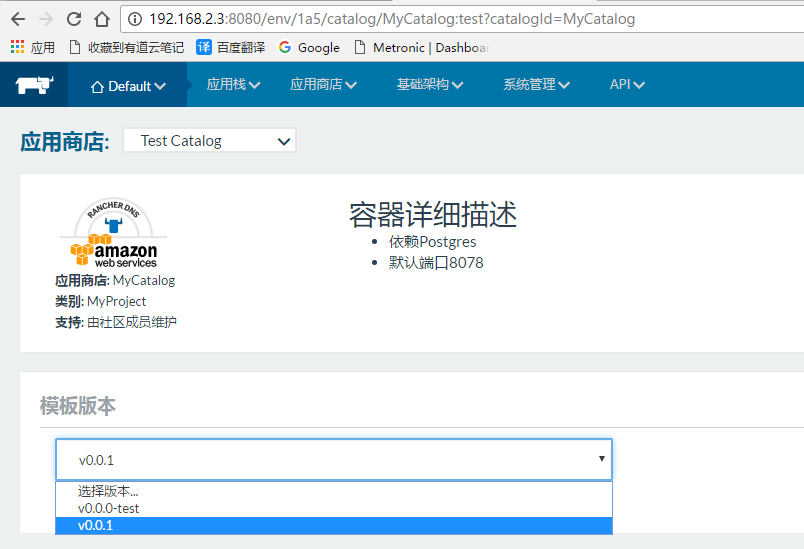
### 查看自己的商店



### 注意事项

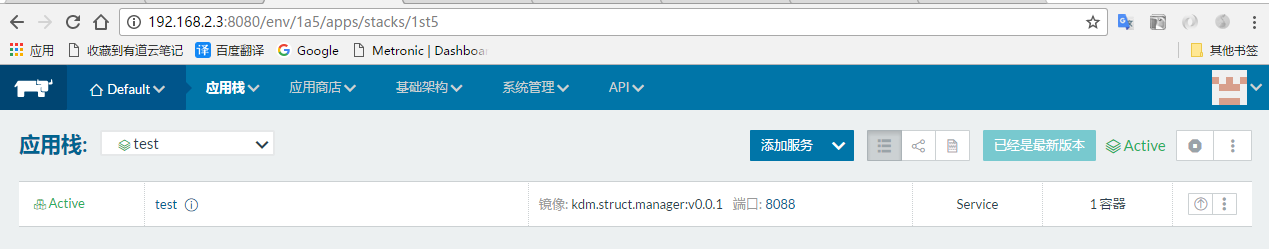
如果templates对应的版本有两个版本：V0.0.0、V0.0.1，如果V0.0.1还未push提交，那么.git文件下就没有更新动作，同样的，rancher也不会识别这个版本

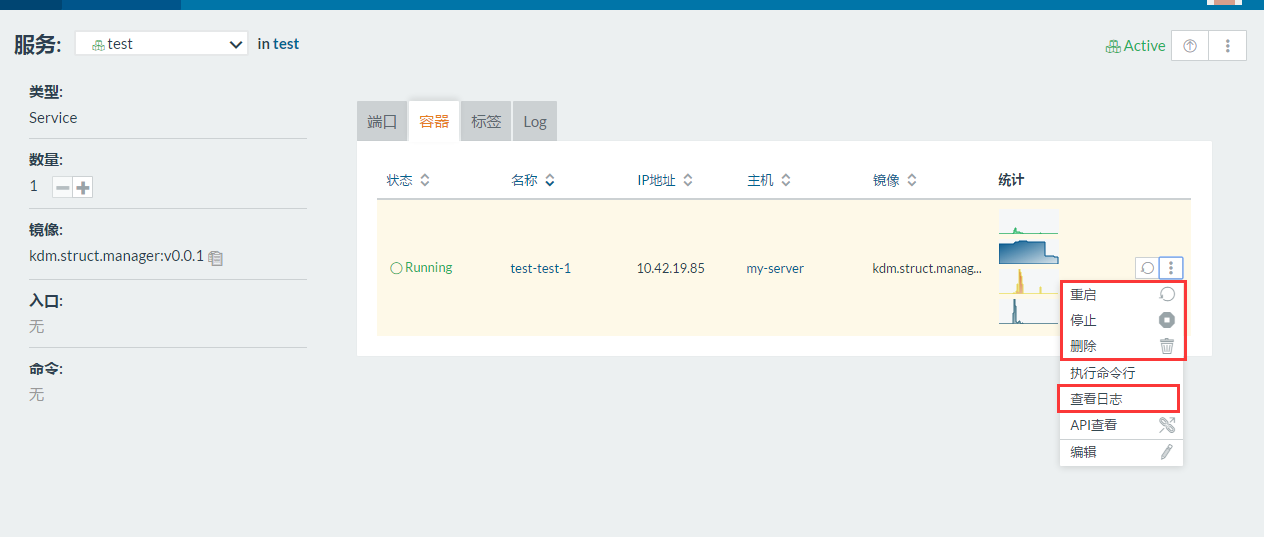
## 升级Catalog



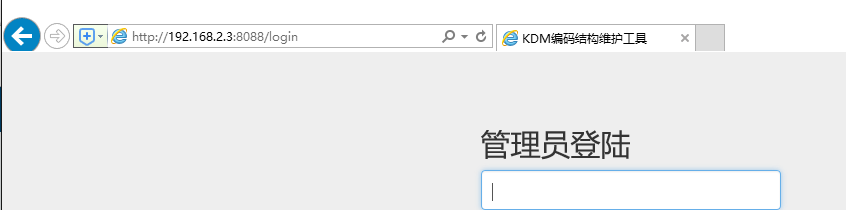
如果一直没刷新，可以尝试更新应用商店名称，从而刷新自己的商店

## 启动catalog





访问192.168.2.3:8088正常



# 搭建私有仓库registry

## 搭建最简单的仓库

默认情况下，会将仓库存放于容器内的/tmp/registry目录下，这样如果容器被删除，则存放于容器中的镜像也会丢失，所以我们一般情况下会指定本地一个目录挂载到容器内的/tmp/registry下

docker run -d --restart=unless-stopped -p 5000:5000 -v /home/yao/registry:/tmp/registry registry

* 测试访问

<http://192.168.2.3:5000/v2/>

* 查看所有镜像

<http://192.168.2.3:5000/v2/_catalog>

* 注意https错误，请将私有仓库加入到docker.service中

## 配置带用户权限的registry

* 创建用户信息

mkdir /home/yao/registry/auth/

docker run --entrypoint htpasswd registry:latest -Bbn yao ubuntu >> /home/yao/registry /auth/htpasswd

* 再次启动

docker run -d -p 5000:5000 --restart=always -v /opt/registry-var/auth/:/auth/ -e "REGISTRY\_AUTH=htpasswd" -e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm" -e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd -v /opt/registry-var/:/var/lib/registry/ registry:latest

## 仓库UI

<https://hub.docker.com/r/hyper/docker-registry-web/>

<https://hub.docker.com/r/konradkleine/docker-registry-frontend/>

sudo docker run \

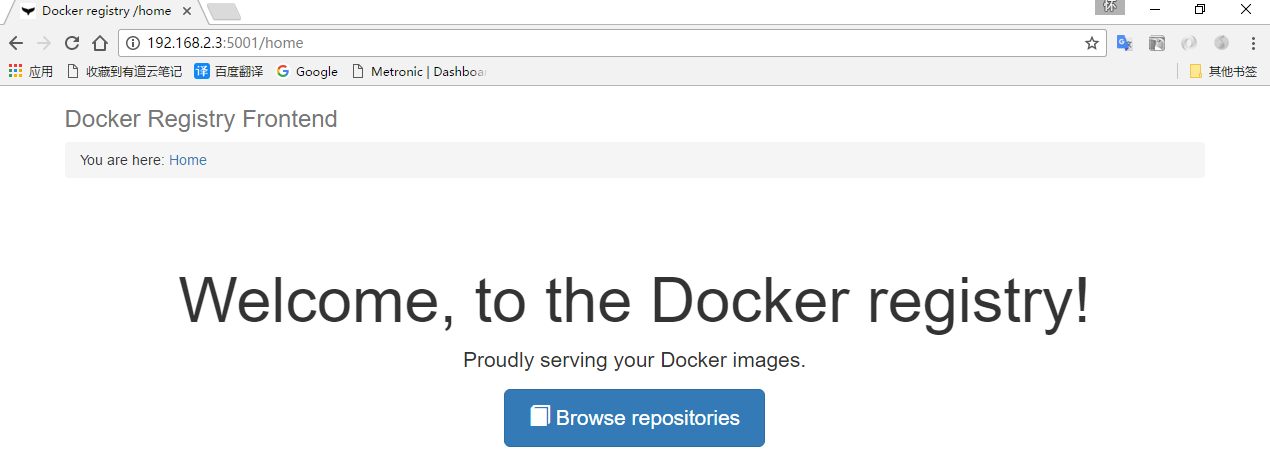
-d --restart=unless-stopped \

-e ENV\_DOCKER\_REGISTRY\_HOST=192.168.2.3 \

-e ENV\_DOCKER\_REGISTRY\_PORT=5000 \

-p 5001:80 \

konradkleine/docker-registry-frontend:v2

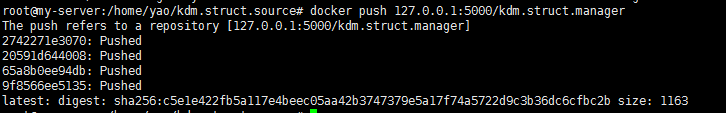


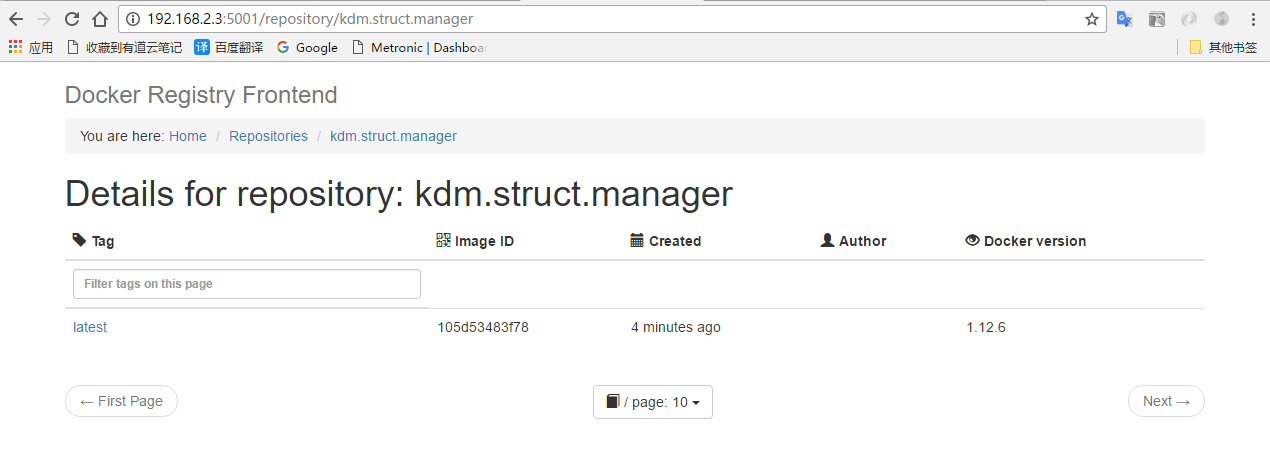
## 试试push到仓库

docker tag kdm.struct.manager 127.0.0.1:5000/kdm.struct.manager



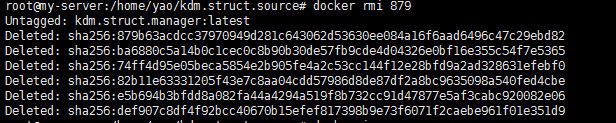
docker push 127.0.0.1:5000/kdm.struct.manager



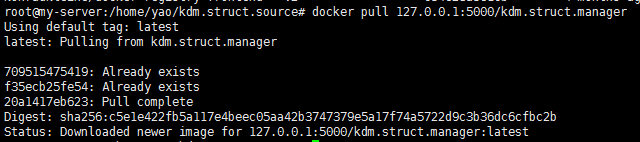


## 试试从仓库pull

docker rmi imageid



docker pull 127.0.0.1:5000/kdm.struct.manager



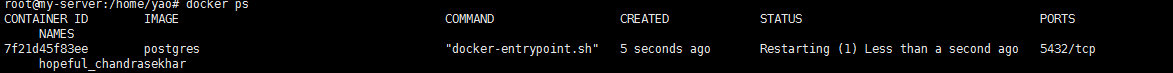
# 手动启动Postgres

下载镜像，指定用户密码端口，指定数据卷位置，并将其挂载到虚拟机上

docker run --name=my-postgres -d --restart=unless-stopped -e POSTGRES\_PASSWORD=postgres -e POSTGRES\_USER= postgres -p 5432:5432 -v /home/yao/postgres:/home/data postgres

--name=my-postgres 后面追加，安装Gitlab需要指定数据库

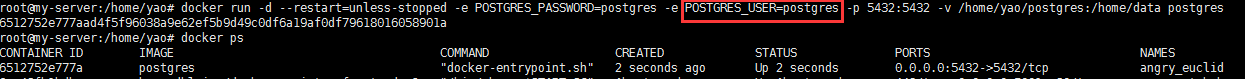


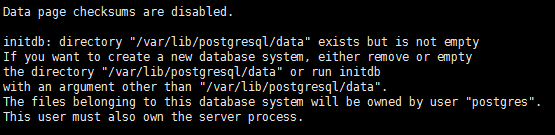


**状态：一直重启**

检查命令，查找错误、查看日志（docker logs 7f2） =>第一次运行的命令，设置用户名多了个一个空格

先删除这个容器docker rm -f 7f2（-f是强制删除，可以先停止，然后再删除）





**如果出现这个错误，请检查你映射的数据文件夹，是否正确**

# 手动搭建Gitlab

## 运行命令

sudo docker run \

-d --restart=unless-stopped \

--name gitlab \

--hostname 192.168.2.3 \

--publish 1818:80 \

--publish 2222:22 \

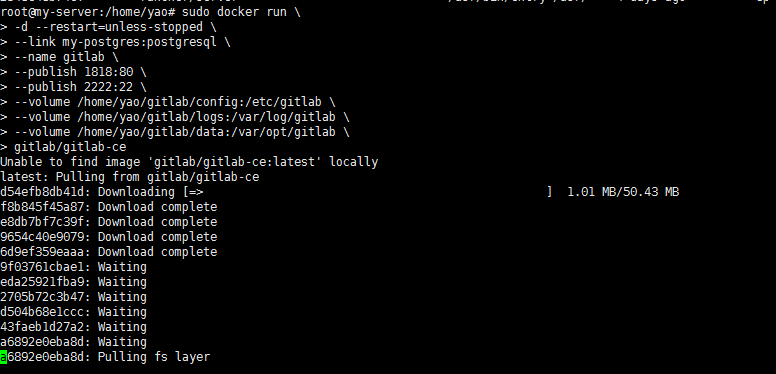
--volume /home/yao/gitlab/config:/etc/gitlab \

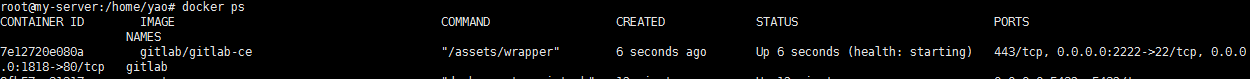
--volume /home/yao/gitlab/logs:/var/log/gitlab \

--volume /home/yao/gitlab/data:/var/opt/gitlab \

gitlab/gitlab-ce

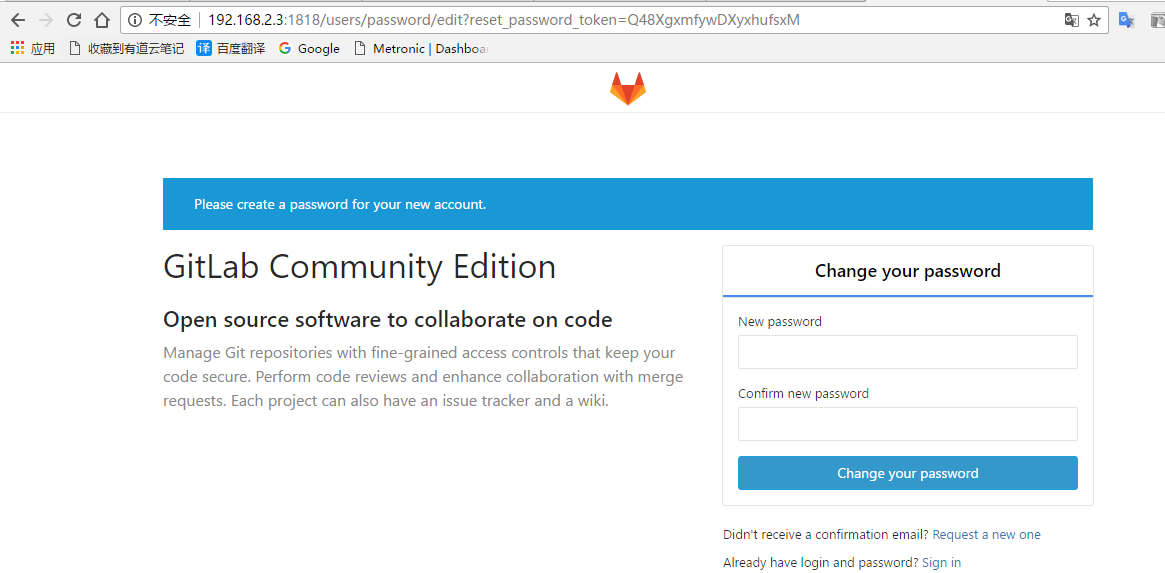
注意：gitlab对应的配置文件，数据库文件，日志文件夹，最好都被映射出来至用户目录；

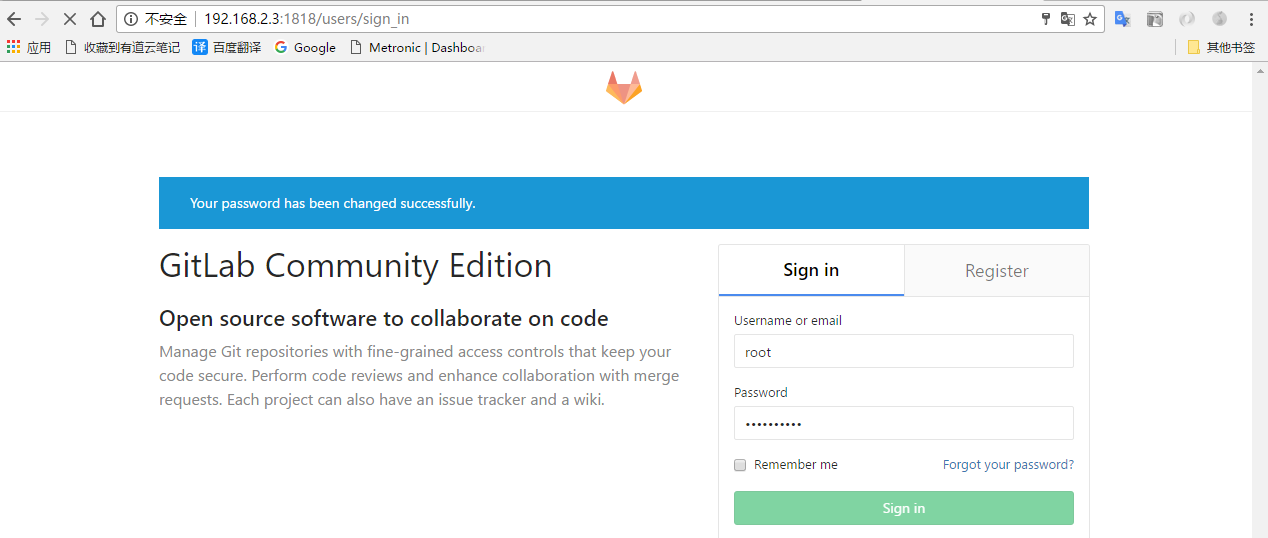




**这个要启动一会，切勿担心**

## 访问192.168.2.3:1818



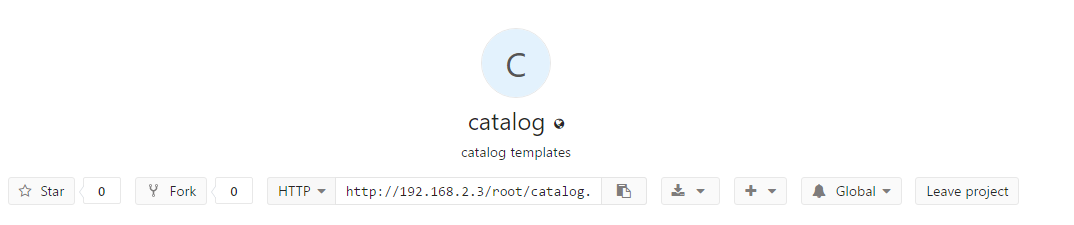


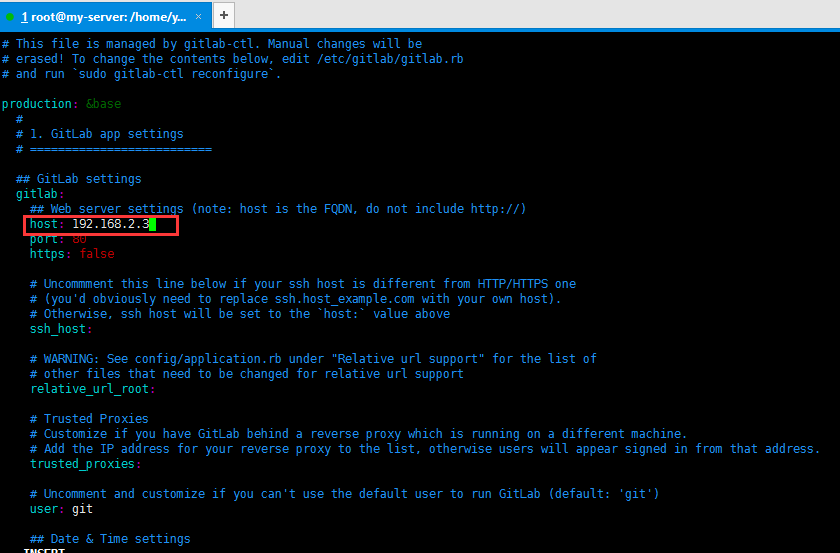
## 修改host

vi /home/yao/gitlab/data/gitlab-rails/etc/gitlab.yml

这个配置文件如果不是在docker里面倒是可以修改，但每次启动docker时，这个文件就会被启动的命令重新修改，所以vi命令下的修改无效。

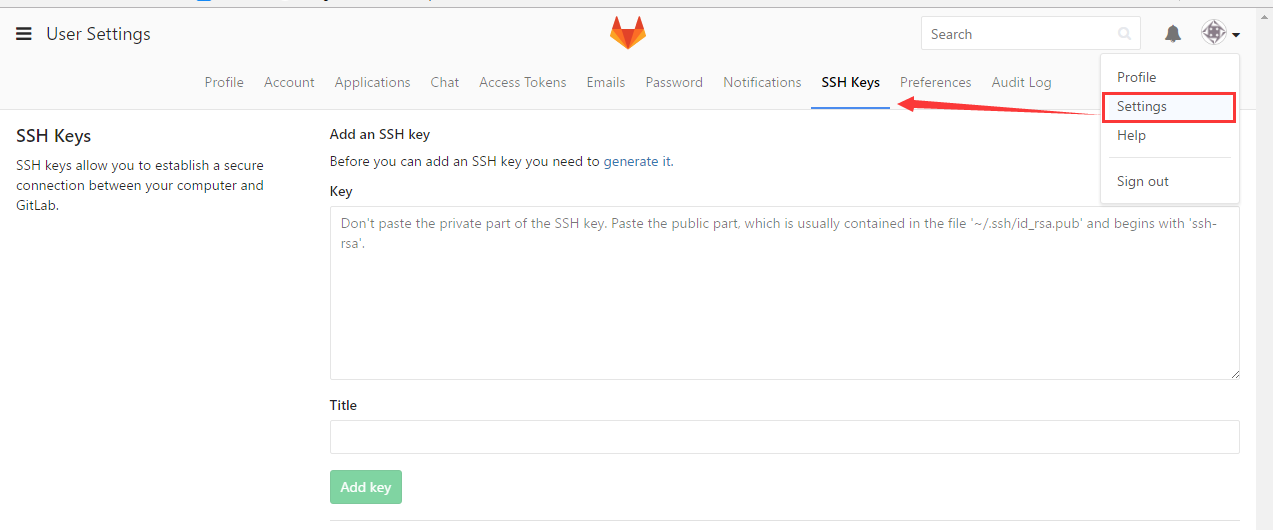
修改启动参数，加入--hostname 192.168.2.3，修改成功

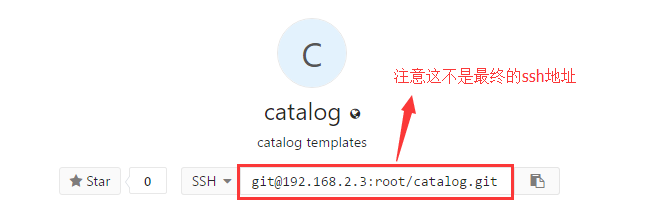


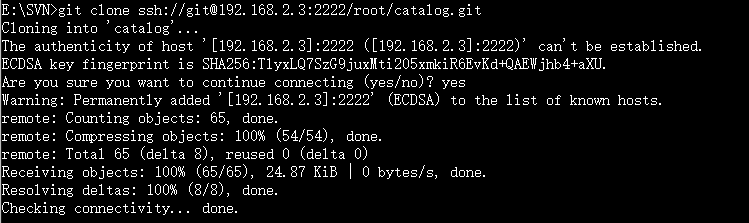


**记得重启服务器**

## 配置ssh下载



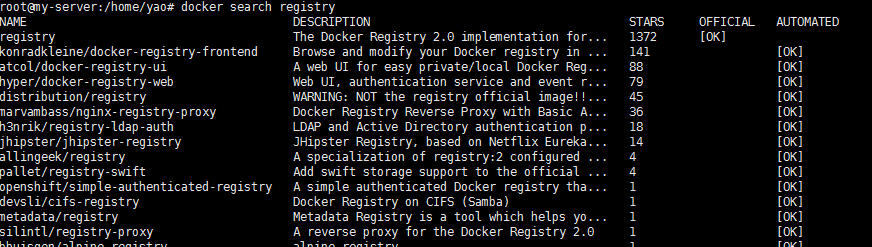




# 常用命令

## docker search

搜索可用docker镜像，如docker search registry



## docker pull

从远端拉取一个image

### 从默认的[Docker Hub](https://hub.docker.com/)拉取镜像

docker pull registry

### 从指定的服务器拉取镜像

docker pull docker.orientge.cn:5000/oge/vzdb

## docker build

### 编写Dockerfile

FROM fiadliel/java8-jre

VOLUME /logs

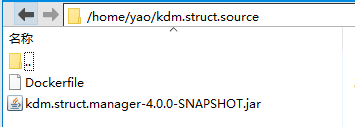
EXPOSE 8079

ADD kdm.struct.manager-4.0.0-SNAPSHOT.jar kdm.struct.manager-4.0.0-SNAPSHOT.jar

ENV JAVA\_OPTS=""

CMD ["java", "-jar", "kdm.struct.manager-4.0.0-SNAPSHOT.jar"]

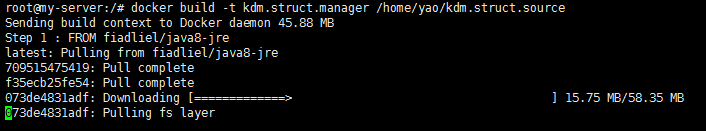
### 文件目录



### 打包命令

docker build -t kdm.struct.manager /home/yao/kdm.struct.source

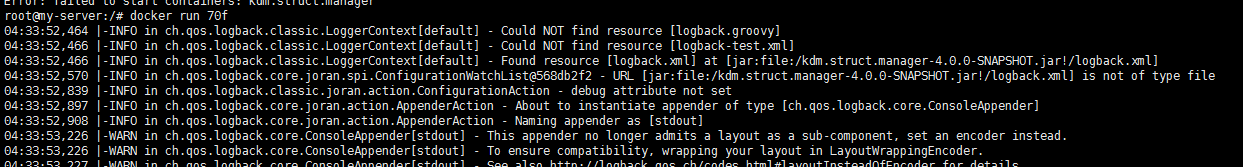
docker build -t kdm.struct.manager:v0.0.1 /home/yao/kdm.struct.source





### 测试运行

可以使用image id简写，使用这个镜像，启动新的容器：docker run 70f

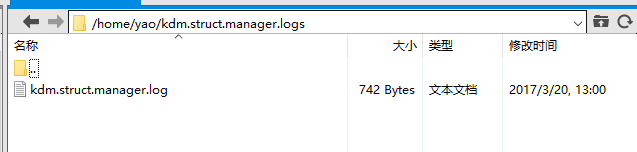


或者运行：

docker run -d --env kdm.rdp.postgre.name=kdm\_struct -p 8088:8078 kdm.struct.manager:v0.0.1



对应的日志也映射:



## docker load

load完全与save相反，会从文件还原image, 测试时先删除原image

docker load -i kdm.struct.manager.tar



## docker save

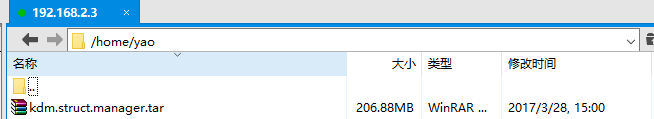
docker save -o kdm.struct.manager.tar kdm.struct.manager:v0.0.1

docker save -o kdm.struct.manager.tar docker.orientge.cn:5000/oge/kdm.struct.manager:v0.0.1

chmod 777 kdm.struct.manager.tar

可以用-o或者--output来指定输出，也可以直接用>重定向到文件

将指定镜像保存成 tar 归档文件， docker load 的逆操作。保存后再加载（saved-loaded）的镜像不会丢失提交历史和层，可以回滚。



## docker push

将镜像推送至远程仓库，默认为 Docker Hub

docker tag kdm.struct.manager 127.0.0.1:5000/kdm.struct.manager

docker push 127.0.0.1:5000/kdm.struct.manager

## docker rmi

docker rmi  镜像名/ID

删除所有未运行Docker容器

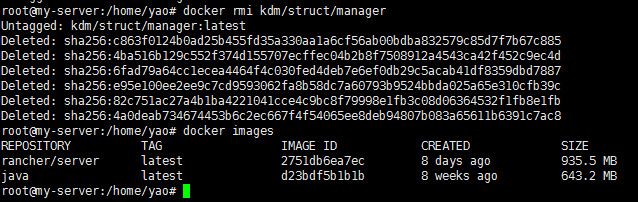
docker rm $(docker ps -a -q)

删除所有未打 tag 的镜像

docker rmi $(docker images | grep "^<none>" | awk "{print $3}")

删除所有镜像

docker rmi $(docker images -q)





注意：在删除镜像之前要先用 docker rm 删掉依赖于这个镜像的所有容器（哪怕是已经停止的容器），否则无法删除该镜像

## docker rm

删除一个或多个容器



## docker run

docker run [options "o">] <image> [ "nb">command] "o">[arg...]

启动一个容器，在其中运行指定命令

常用命令解释

docker run -d --restart=unless-stopped -p 8080:8080 rancher/server

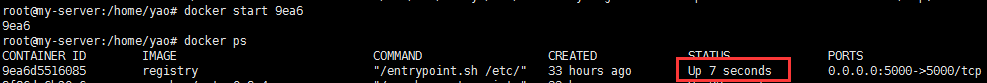
启动镜像rancher/server，端口映射8080=>8080，除了stopped命令，其他都会重启

sudo docker run -t -i training/sinatra /bin/bash

启动镜像training/sinatra， 并启动一个 bash 终端，允许用户进行交互

## docker start|stop|restart

启动、停止和重启一个或多个指定**容器**。



## docker ps

列出所有运行中容器

-a 显示所有容器，包括当前没有运行的容器

--before="nginx" 列出在某一容器之前创建的容器，接受容器名称和ID作为参数；

--since="nginx" 列出在某一容器之后创建的容器，接受容器名称和ID作为参数；

-f [exited=<int>] 列出满足

exited=<int> 条件的容器；

-l 仅列出最新创建的一个容器；

--no-trunc 显示完整的容器ID；

-n=4 列出最近创建的4个容器；

-q 仅列出容器ID；

-s 显示容器大小。

## docker pause/ unpause

暂停/恢复某一容器的所有进程

## docker kill

杀死一个或多个指定容器进程。

## docker inspect

检查镜像或者容器的参数，默认返回 JSON 格式。

docker inspect kdm.struct.manager:v0.0.1

docker inspect gitlab/gitlab-ce

## docker logs

获取容器运行时的输出日志

## docker export

将指定的容器保存成 tar 归档文件， docker import 的逆操作。导出后导入（exported-imported)）的容器会丢失所有的提交历史，无法回滚。

## docker import

从归档文件（支持远程文件）创建一个镜像， export 的逆操作，可为导入镜像打上标签。导出后导入（exported-imported)）的容器会丢失所有的提交历史，无法回滚。

## docker top

查看一个正在运行容器进程，支持 ps 命令参数

## docker images

docker images [options "o">] [name]

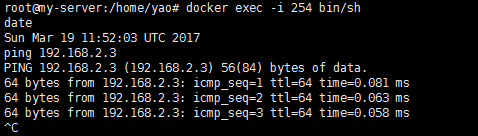
列出本地所有镜像。其中 [name] 对镜像名称进行关键词查询。

## docker attach

连接到启动的容器

## docker exec

docker exec-i bb2 /bin/sh



输入exit退出

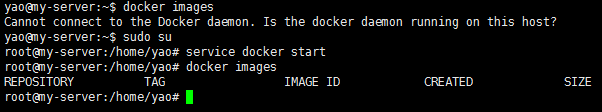
## 其他

docker tag imageid name:tag

docker tag 13b docker.orientge.cn:5000/oge/kdm.struct.manager:v0.0.1

# 常见问题处理

## Cannot connect to the Docker daemon. Is the docker daemon running on this host?



权限问题，当前用户没权限，root用户可以运行

解决方法：

sudo su //切换到root

service docker start //启动docker service

docker images //显示所有images

## server gave HTTP response to HTTPS client

从我公司内网拉取一个vzdb镜像：如docker  pull docker.orientge.cn:5000/oge/vzdb



Docker默认从docker hub拉取镜像的，使用https客户端拉取，但私有仓库是http的，如果需要访问http仓库，需修改docker.service，增加的命令如下：--insecure-registry docker.orientge.cn:5000

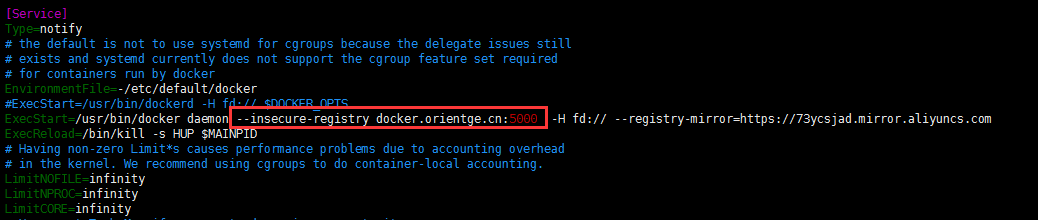
vi /lib/systemd/system/docker.service

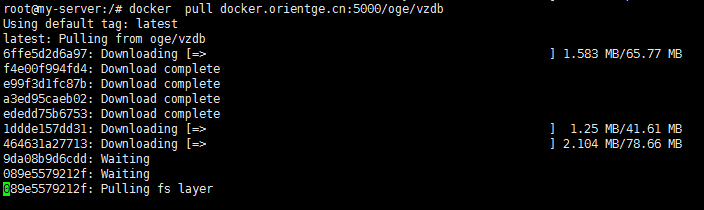
ExecStart=/usr/bin/docker daemon --insecure-registry docker.orientge.cn:5000

-H fd:// --registry-mirror=https://73ycsjad.mirror.aliyuncs.com

service docker stop

service docker start



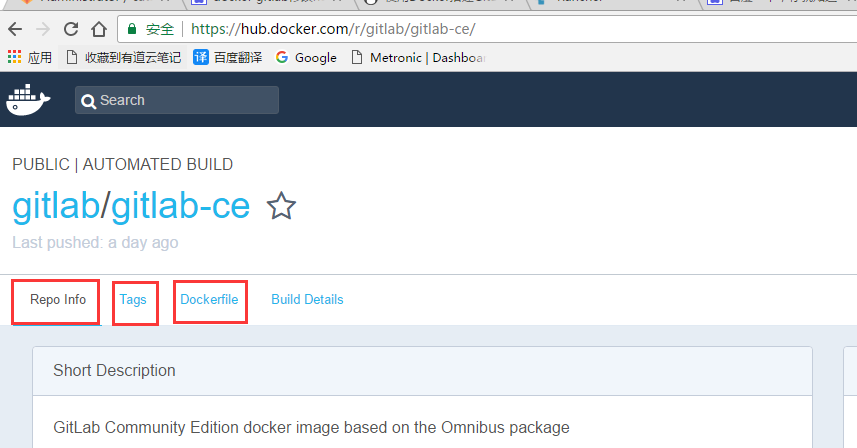


docker  pull docker.orientge.cn:5000/oge/kdm-rdp

# 高级技巧

## 怎么知道官方的镜像有哪些参数，变量可以设置？

### 从docker hub搜索该镜像的介绍



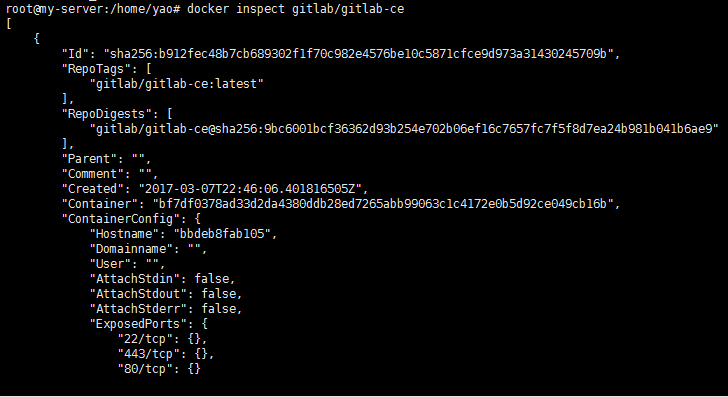
Repo info : 介绍了一般启动改镜像的方法（常用的参数都有说明）

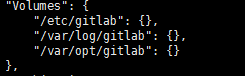
Tags: 该镜像的所有版本

Dockerfile : 镜像打包时候的命令（这里面就有详细参数了）

### dockers inspect 命令

docker inspect gitlab/gitlab-ce:latest





# Maven私服打包image、制作catalog模板

Sonatype Nexus Repository OSS 是 Apache Maven 私服仓库，永久免费。增加 Docker、NuGet、npm、Bower、PyPI、Ruby Gems的支持；基于 JDK 1.8+；

* Java maven的jar包仓库
* docker的镜像仓库

下载：<https://www.sonatype.com/download-oss-sonatype>

Linux上安装nexus3参照文档：<http://blog.csdn.net/sinat_34017357/article/details/51776641>

## Dockerfile

FROM openjdk:8u121

MAINTAINER yao

VOLUME /sonatype-work

EXPOSE 8081

ADD nexus-3.2.1-01 nexus-3.2.1-01

ADD sonatype-work sonatype-work

WORKDIR nexus-3.2.1-01/bin

CMD ./nexus run

## 打包运行命令

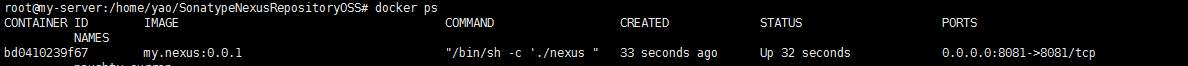
tar zxvf nexus-3.2.1-01-unix.tar.gz

docker build -t my.nexus:0.0.1 /home/yao/SonatypeNexusRepositoryOSS

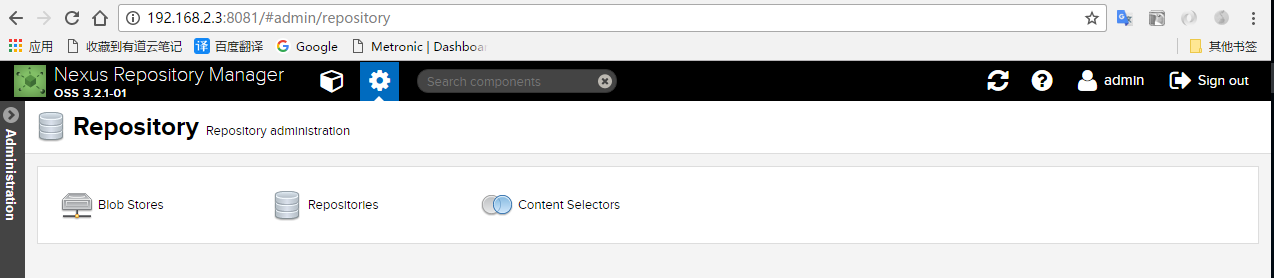
docker run -d --restart=always -p 8081:8081 my.nexus:0.0.1

docker rmi my.nexus:0.0.1

docker run -t -i my.nexus:0.0.1 /bin/bash



## 登陆



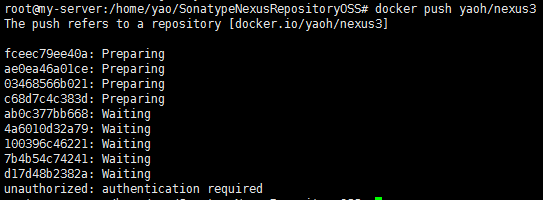
管理员：admin/admin123

## 上传镜像到我的docker hub

<https://hub.docker.com/r/yaoh/nexus3/>

docker tag imageid yaoh/nexus3

docker run -d --restart=always -p 8081:8081 yaoh/nexus3



需要登录

