基于Docker swarm和Rancher来快速实现PG PAAS私有云服务

嘉宾：王鹏冲
公司：平安银行
邮箱：wangpengchong001@qq.com
微信公众号：数据库技术圈

这是一个万众创业的时代

- DaoCloud - 风和投资，千万美金级别
- 时速云 - 朗玛峰创投， 数千万人民币
- 数人云 - 云启创投，3000万人民币
- 灵雀云 - 宽带资本，一千万美金

从一个故事开始

2017年春节后，几篇关于Docker和数据库的文章在朋友圈流传较广：
一位老外首先发表了他的意见：《why-databases-is-not-for-containers》
https://myopsblog.wordpress.com/2017/02/06/why-databases-is-not-for-containers/
里面列举了他认为容器技术不适合运行数据库的7个原因。

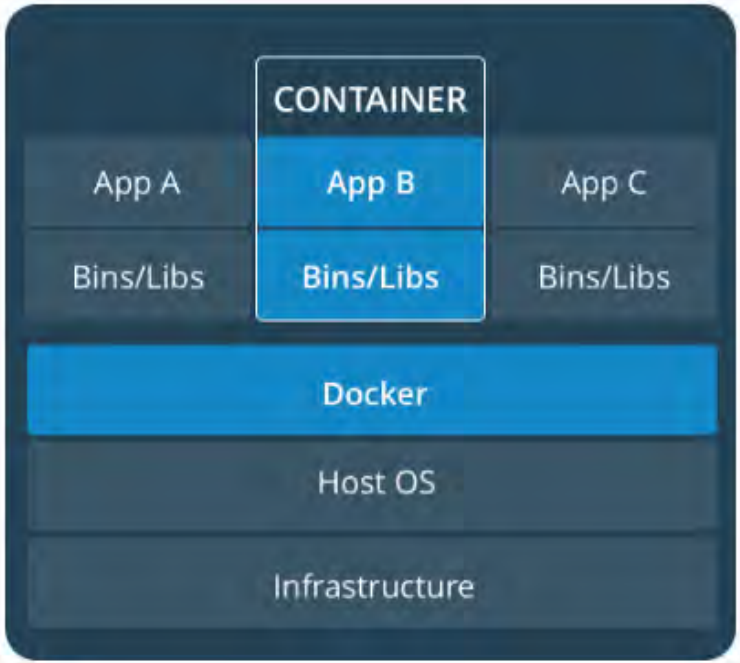
后来午夜咖啡的blog发表了一篇博文：
《2016年容器技术思考：Docker, Kubernetes, Mesos 将走向何方？》http://jolestar.com/container-ecosystem/
这篇文章较为全面地阐述了他对容器的技术生态圈的发展前景所做的思考，非常值得细细咀嚼。

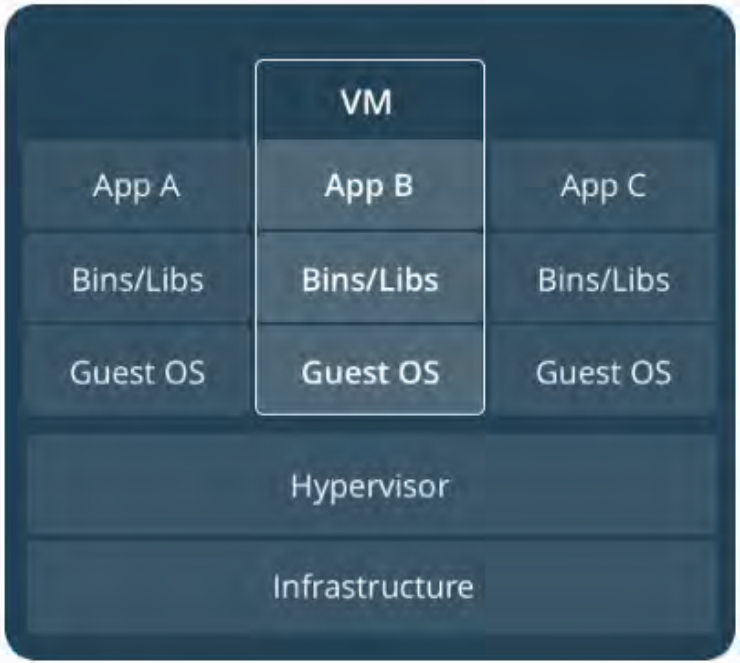
其中有一句话很鲜明地解释了docker与虚拟机的本质差别：
传统虚拟机封装的目标是操作系统，为操作系统提供虚拟化硬件环境，
而容器封装的目标是应用进程，其中内嵌的操作系统只是应用的依赖而已。

我想午夜咖啡在发表上述博文时，应该还没有看到《why-databases-is-not-for-containers》这篇文章。
所以后来他又专门发表了一篇博文：
论数据库容器化的目标和价值http://jolestar.com/the-containerized-value-of-database/

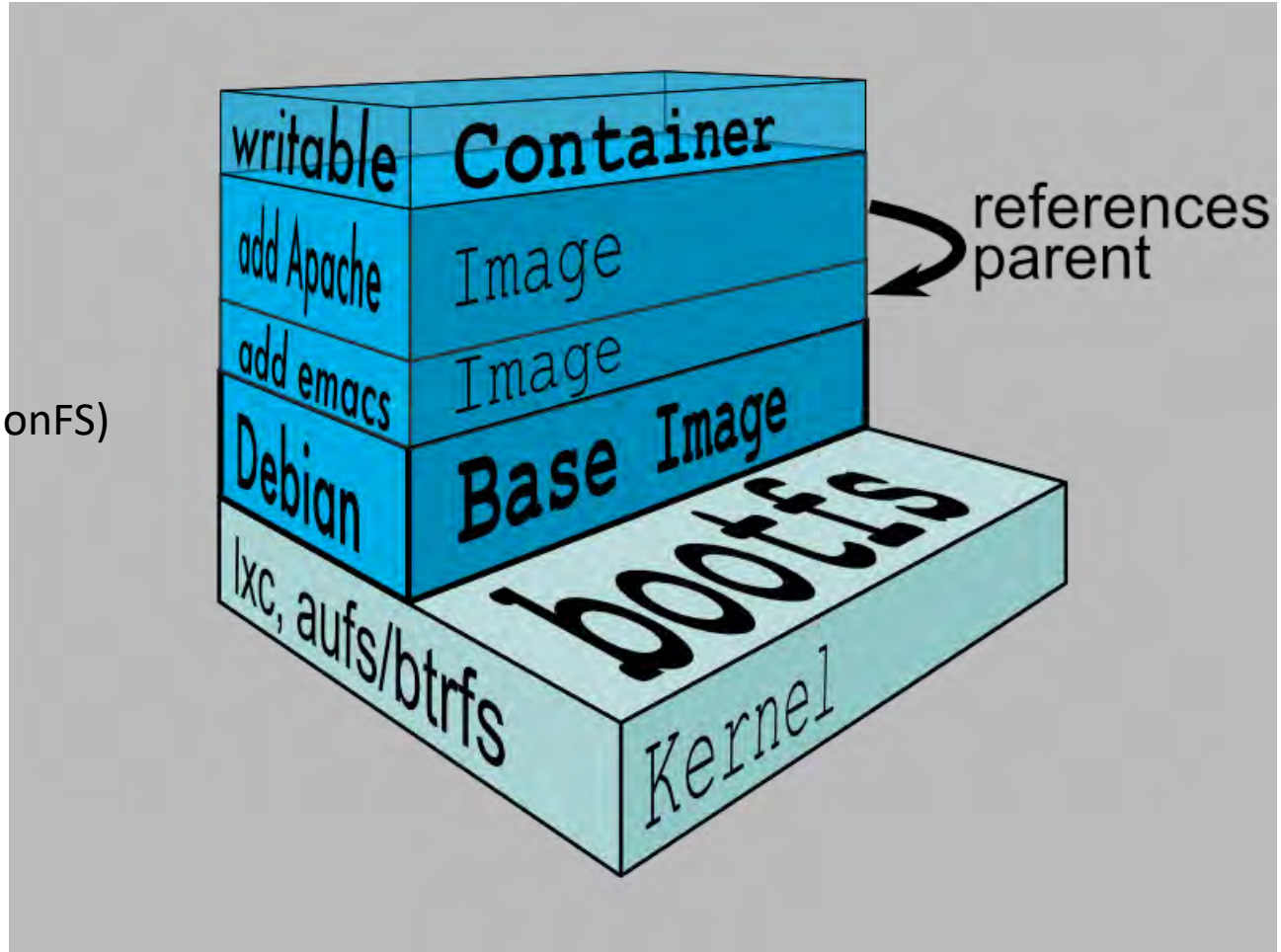
Comparing Containers and Virtual Machines

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. containers are more portable and efficient.





AUFS (AnotherUnionFS)



纸上得来终觉浅，绝知此事要躬行

- 环境：
- Host1
- Host2
- 操作系统：CoreOS Container Linux

说明：

制作PG镜像的步骤可以通过编辑一个大而全的dockerfile，然后执行一个命令就可以完成。

这里是为了更好理解docker镜像的制作过程和原理，所以分解成很多步骤、手工来做。

1、创建安装PG的容器

Host1宿主机器操作:

```
# docker pull ubuntu
# mkdir -p /backup/postgresql
# mkdir -p /storage/docker/postgresql
# docker run -it --name build_pg_img -v
    /storage/docker/postgresql:/var/lib/postgresql -v
    /backup/postgresql:/backup ubuntu /bin/bash
```

2、在容器内安装PG软件

连接到刚才创建的容器内：

```
# Docker exec –it build_pg_img /bin/bash
root@3fd84ba76393:~# apt-get update
root@3fd84ba76393:~# apt-get install vim
root@3fd84ba76393:~# apt-get install postgresql-9.5
```

3、配置用户ENV、CONF文件并建库

依旧在容器内操作，与linux上面initdb步骤没什么区别，此处不再展开。

本页PPT备注栏里面有详细过程。

4、创建PG的docker基础镜像

退出容器，到宿主机：

```
# docker ps -a |grep build_pg_img
# docker commit build_pg_img pg9.5_template
```

5、创建PG的客户化镜像

宿主机：

```
# vi dockerfile
    FROM  pg9.5_template
    USER postgres
    ENV PGHOME /usr/lib/postgresql/9.5
    ENV PGPORT 5432
    ENV PGDATA /var/lib/postgresql/data
    ENV PATH $PGHOME/bin:$PATH
    ENV LD_LIBRARY_PATH=$PGHOME/lib:$LD_LIBRARY_PATH
    EXPOSE 5432
    CMD ["/usr/lib/postgresql/9.5/bin/postgres", "-D", "/var/lib/postgresql/data"]
# docker build -t eg_postgresql .
```

6、基于新镜像来启动PG容器

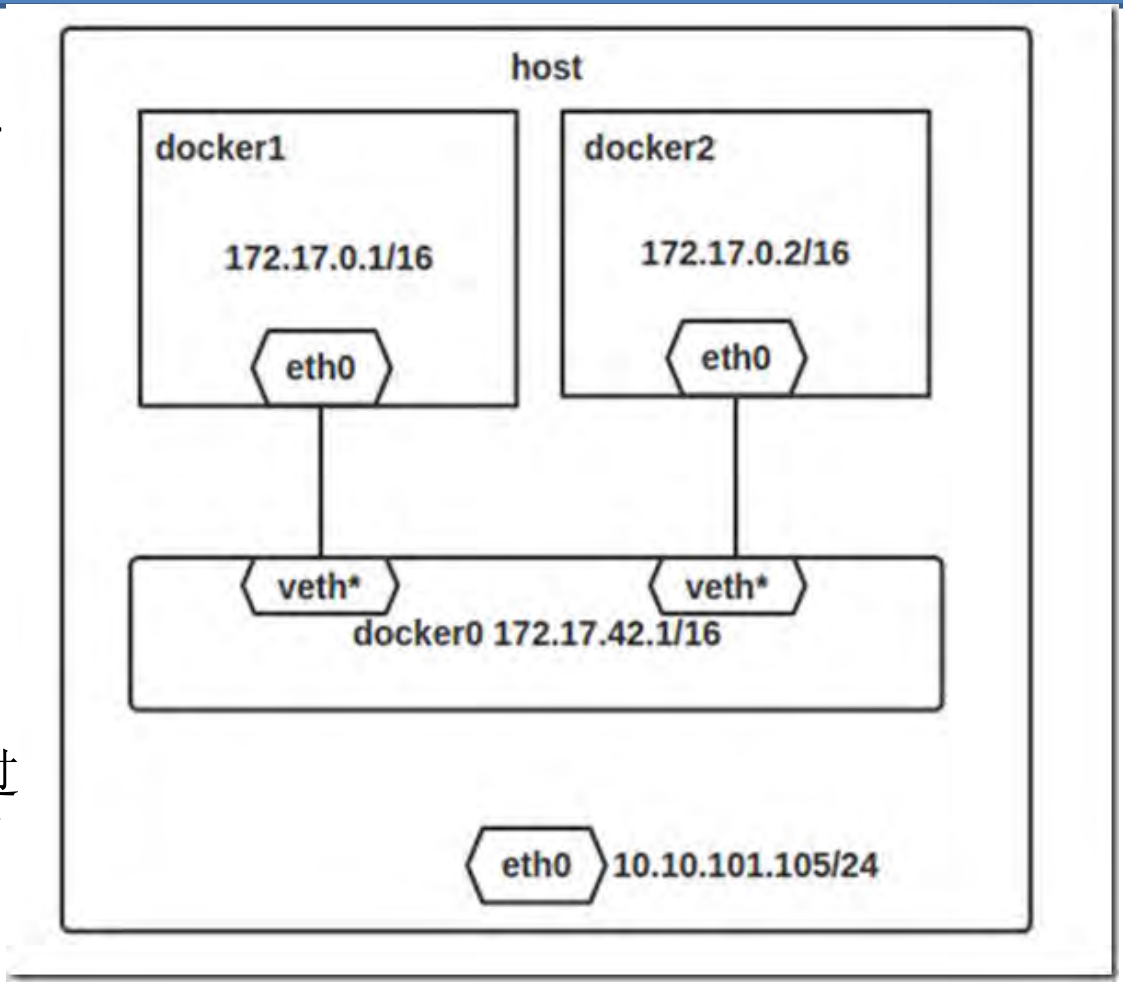
```
# docker run --name pg_pri -v
    /storage/docker/postgresql:/var/lib/postgresql -v
    /backup/postgresql:/backup -p 5436:5432 -d eg_postgresql


# docker ps  |grep pg_pri
    6ccecb527eaa        eg_postgresql       "/usr/lib/postgresql/"
    9 seconds ago       Up 9 seconds        5432/tcp            pg_pri
```

在使用docker run创建Docker容器时，可以用--net选项指定容器的网络模式，Docker有以下4种网络模式：

✓host模式
✓container模式
✓none模式。
✓bridge模式（默认）

我们通过端口映射-p 5436:5432 使得外部可以通过宿主机IP+5436端口即可访问容器内的PG服务。



7，测试PG容器

从宿主机进入容器内：

docker exec -it pg_pri /bin/bash

```
postgres@6ccecb527eaa:/$ ps -ef|grep post
postgres     1     0  0 07:42 ?        00:00:00 /usr/lib/postgresql/9.5/bin/postgres
/var/lib/postgresql/data
postgres     6     1  0 07:42 ?        00:00:00 postgres: logger process
postgres     8     1  0 07:42 ?        00:00:00 postgres: checkpointer process
postgres     9     1  0 07:42 ?        00:00:00 postgres: writer process
postgres    10     1  0 07:42 ?        00:00:00 postgres: wal writer process
postgres    11     1  0 07:42 ?        00:00:00 postgres: autovacuum launcher
cess
postgres    12     1  0 07:42 ?        00:00:00 postgres: archiver process
postgres    13     1  0 07:42 ?        00:00:00 postgres: stats collector process
postgres    14     0  0 07:43 ?        00:00:00 /bin/bash
postgres    22    14  0 07:43 ?        00:00:00 ps -ef
postgres    23    14  0 07:43 ?        00:00:00 /bin/bash
postgres@6ccecb527eaa:/$
```

如何以root用户进入容器？

- 因为我们的dockerfile里面指定过user是postgres，所以进入容器后默认用户是postgres用户，如果想以root用户进入，不能在容器内部su – root，会报错。
- 用root进入容器：

docker exec -u root -it pg_pri /bin/bash

8、将PG镜像传到host 2

1、将镜像打包

docker save -o eg_postgresql.tar eg_postgresql

2、传输到pg从库所在的机器上

scp eg_postgresql.tar core@192.168.11.2:/tmp/.

3、在pg从库机器（host2）上load image

docker load -i eg_postgresql.tar

9，主库PG容器做备份

1，在pg主库机器host1上操作，进入主库容器

host1 # docker exec -it pg_pri /bin/bash

2，主库创建流复制用户

```
CREATE USER repuser replication LOGIN CONNECTION LIMIT 3 ENCRYPTED
      PASSWORD 'repuser';
```

3，主库打开备份：

```
select pg_start_backup('Replition work');
```

4，退出容器，在宿主机器上操作数据文件的备份：

```
cd /storage/docker/postgresql
tar cvf data.tar data
scp data.tar core@192.168.11.2:/tmp/.
```

5，进入容器，主库结束备份

```
select pg_stop_backup(), current_timestamp;
```

10，将备份恢复到从库的宿主机卷

在pg从库机器host2上操作：

```
mkdir -p /backup/postgresql
mkdir -p /storage/docker/postgresql
cp /tmp/data.tar /storage/docker/postgresql/.
tar xvf data.tar
cd /storage/docker/postgresql/data
```

删除slave端(从master端拷过来的)的pid文件

```
rm -f $PGDATA/postmaster.pid
```

11， 编辑从库recover.conf

#cd /storage/docker/postgresql/data

vi recovery.conf

standby_mode = 'on'

primary_conninfo = 'host=192.168.17.61 port=5436 user=repuser password=repuser keepalives_idle=60'

trigger_file = '/var/lib/postgresql/primary_down'

12，启动从库，完成流复制搭建

启动从库容器

```
docker run --name pg_slv -v
    /storage/docker/postgresql:/var/lib/postgresql -v
    /backup/postgresql:/backup -p 5436:5432 -d eg_postgresql
```

```
host2# docker ps |grep pg_slv
   830b2ea66753        eg_postgresql       "/usr/lib/postgresql/"   About a
    minute ago   Up About a minute   0.0.0.0:5436->5432/tcp   pg_slv
```

进入从库容器

```
host2# docker exec -it pg_slv /bin/bash
```

思考，我们还要做什么，怎么做?

- ✓ 基于pg_pool2的自动切换，容器之间如何共享卷？
- ✓ 如何实现定期自动备份，是在宿主机做还是在容器内做？备份文件放哪？ 容器内？
- ✓ 如何配置docker上的zabbix来监控PG？zabbix agent运行在哪里？宿主机？PG容器？还是单独自己的容器？如何不同容器共享同一个zabbix agent？

开始Rancher+Docker Swarm

- 有了PG镜像，就可以将之上传到利用docker registry 搭建的私有镜像仓库中
- 然后利用公司统一的容器管理平台，实现基础架构资源的封装、容器管理与调度（CaaS）、弹性计算、CI/CD等目的，即实现私有PaaS服务。
- 在容器的世界里，开放人员交付的不再是代码库，而是一个在dev环境开发、调试通过的docker image。
- 测试人员不用再担心将开发封板移交的这个image部署到staging环境的docker后，跑起来再报错。

Yes ！ Build Once，Run Anywhere！

Docker是一个重新定义了程序开发测试、交付和部署过程的开放平台

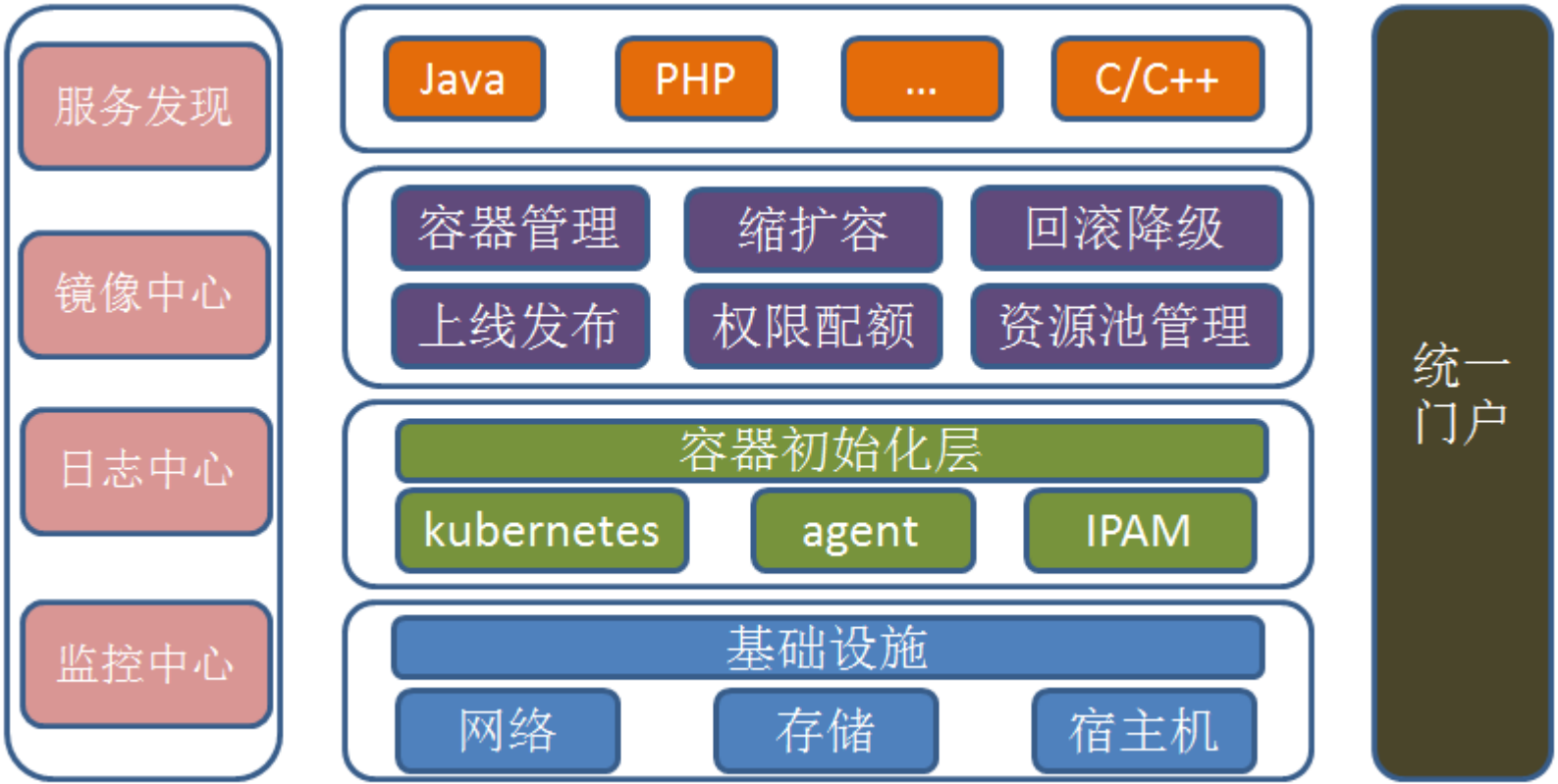
三大主流容器调度框架：Swarm、Mesos、Kubernetes

摘录自： http://dockone.io/article/2783

	swarm	Mesos	Kubernetes
应用隔离机制	docker	Mesos/docker/other	docker
资源类型	内存 cpu 端口	内存 cpu 端口 ulimit	内存 cpu 端口 ulimit
主机分组	Docker进程标签	Slave分组	Slave分组
调度策略	资源使用情况	资源使用情况	资源使用情况+应用节点均衡
编排	No	Yes	Yes
网络模式	Docker原生	支持自建	支持自建
主高可用	双主切换	zk	etcd
缩扩容	二次开发	API修改	API修改
健康检查	No	Yes	Yes
故障转移	No	Yes	Yes
应用滚动升级	No	No	Yes
负载均衡	No	Haproxy	Kube-proxy
集群规模	小	中	中，提升中
生产使用	较少	业界大规模使用	业界大规模使用

Docker compose
服务编排

58同城私有云架构图：


服务发现
镜像中心
日志中心
监控中心
Java
PHP
...
C/C++
容器管理
缩扩容
回滚降级
上线发布
权限配额
资源池管理
容器初始化层
kubernetes
agent
IPAM
基础设施
网络
存储
宿主机
统一门户


安全 | https://www.cnrancher.com/rancher/
RANCHER
产品
广纳贤才
Blog
DOCS
支持
新闻
活动
关于我们
别担心，容器其实很简单
Rancher，把一切都变得简单
下载 Rancher

Rancher被Gartner评为"四大最酷云基础设施供应商"之一！

Shirley Huang on 5月 27, 2017

随着越来越多的企业开始采用将本地、非本地、云端相结合的部署方式，正确选择云基础设施供应商变得更加关键。世界领先的信息技术研究和咨询公司Gartner，于本月公布了其对世界范围内各大云基础设施供应商的分析研究结果，评选出了四大"最酷云基础设施供应商（Cool Vendors in Cloud Infrastructure）"，上榜公司为Rancher Labs、Cloudistics、Joviam和Rescale。

Rancher是全球唯一提供Kubernetes、Mesos和Swarm三种调度工具的企业级分发版和商业技术支持的容器管理平台

KUBERNETES

强大并且灵活, Kubernetes 反映了Google多年的容器运行经验。

Read more

DOCKER SWARM

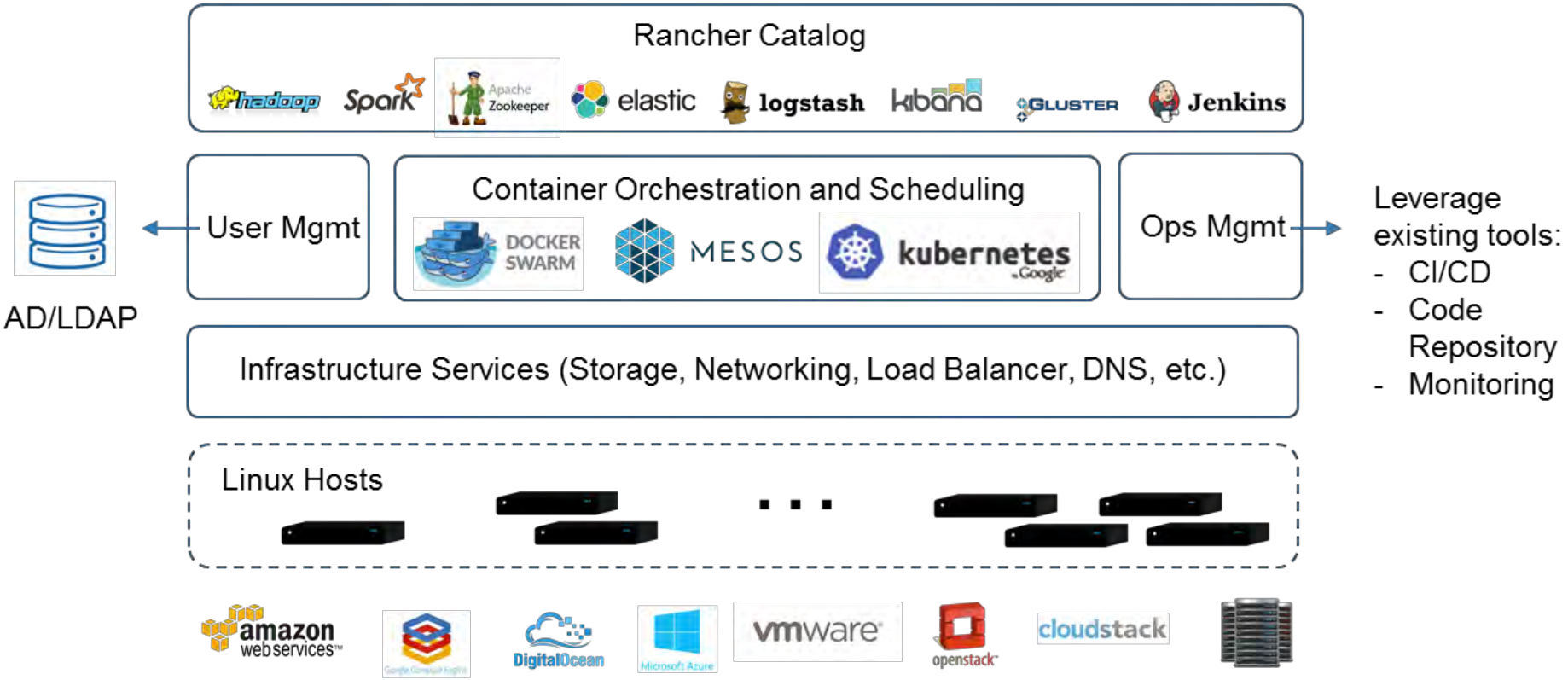
易学易用，精妙的设计, Docker Swarm 是提供给熟悉Docker使用者的原生集群。

Read more

MESOS

为了大规模分布式计算而生, Mesos 驱动了全球无数的网络规模企业

Read more


Rancher Catalog
hadoop
Spark
Apache Zookeeper
elastic
logstash
kibana
GLUSTER
Jenkins
AD/LDAP
User Mgmt
Container Orchestration and Scheduling
DOCKER SWARM
MESOS
kubernetes
Google
Ops Mgmt
Leverage existing tools:
- CI/CD
- Code Repository
- Monitoring
Infrastructure Services (Storage, Networking, Load Balancer, DNS, etc.)
Linux Hosts
amazon web services
DigitalOcean
Microsoft Azure
vmware
openstack
cloudstack

简而言之，Rancher就是一个企业级的容器管理平台。

Rancher主要由4部分组件构成：

1，基础架构资源的管理：无论该主机是物理机、虚拟机、公有云、私有云上的机器。

2，容器的编排和调度：Rancher支持将docker swarm、mesos、kubernetes的docker集群加入管理。使用Rancher提供的Cattle来管理service（基于docker swarm扩展开发的），或者使用它们原生的管理工具。

3，应用仓库：已经编排好的应用容器仓库，即插即用，自动升级，支持公共仓库和私有仓库。

4，企业管理：支持企业现有的AD、LDAP、GitHub用户权限认证，支持容器的环境分组和管理。

1，Rancher安装

Rancher的server也是以容器方式运行的，Rancher Server可以搭建为单节点的，也有HA架构的，我们搭建单节点的来测试即可。

1、初始化3台linux机器：

Linux host with 64-bit Ubuntu 16.04, which must have a kernel of 3.10+.

2、安装docker

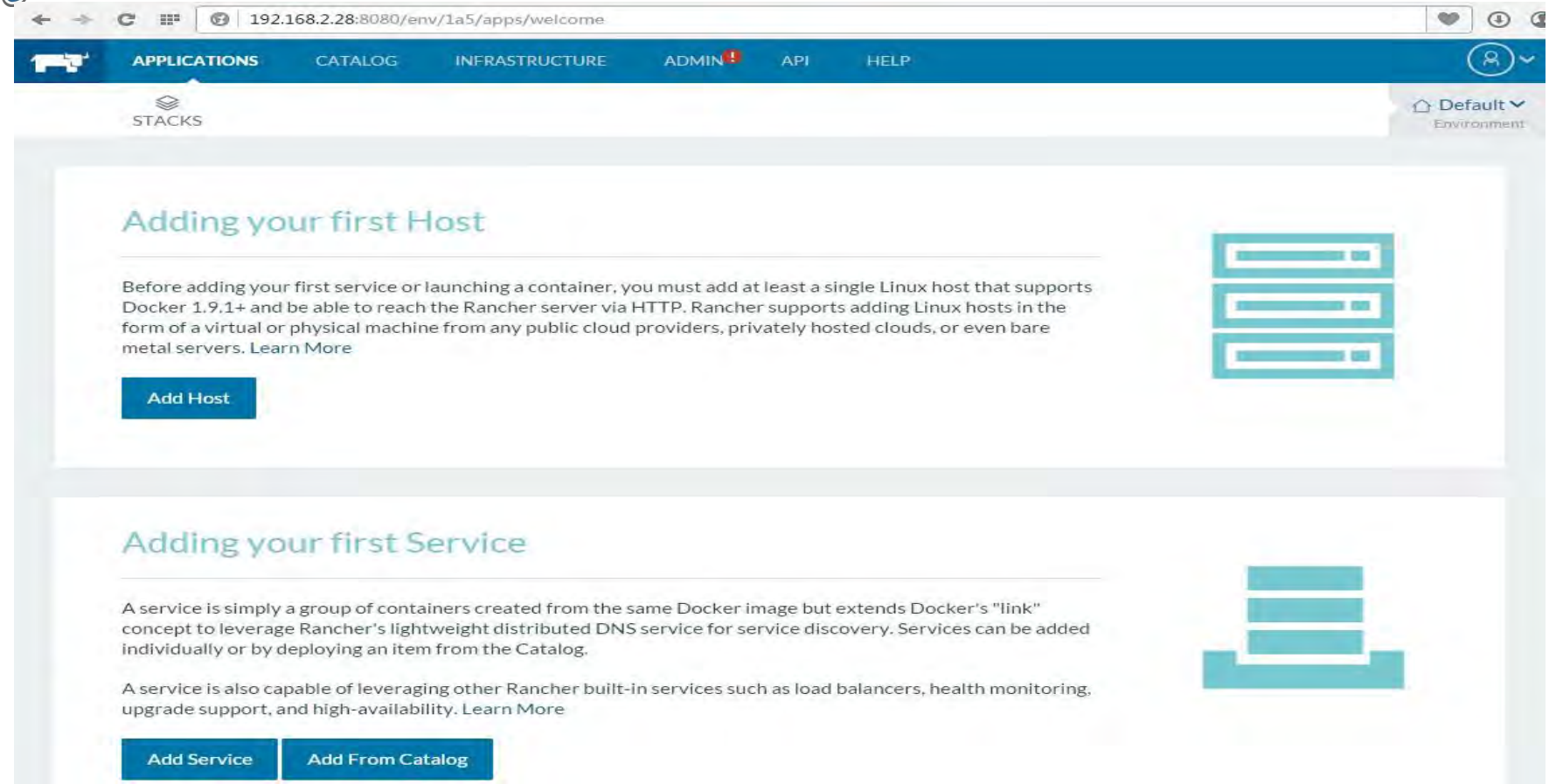
https://docs.docker.com/engine/installation/linux/ubuntu/

3、在其中一台机器上启动Rancher Server

```
$ sudo docker run -d --restart=unless-stopped -p 8080:8080 rancher/server:stable
# Tail the logs to show Rancher
$ sudo docker logs -f <CONTAINER_ID>
```

4、Rancher Server启动需要几分钟，完全启动后才可以访问UI：

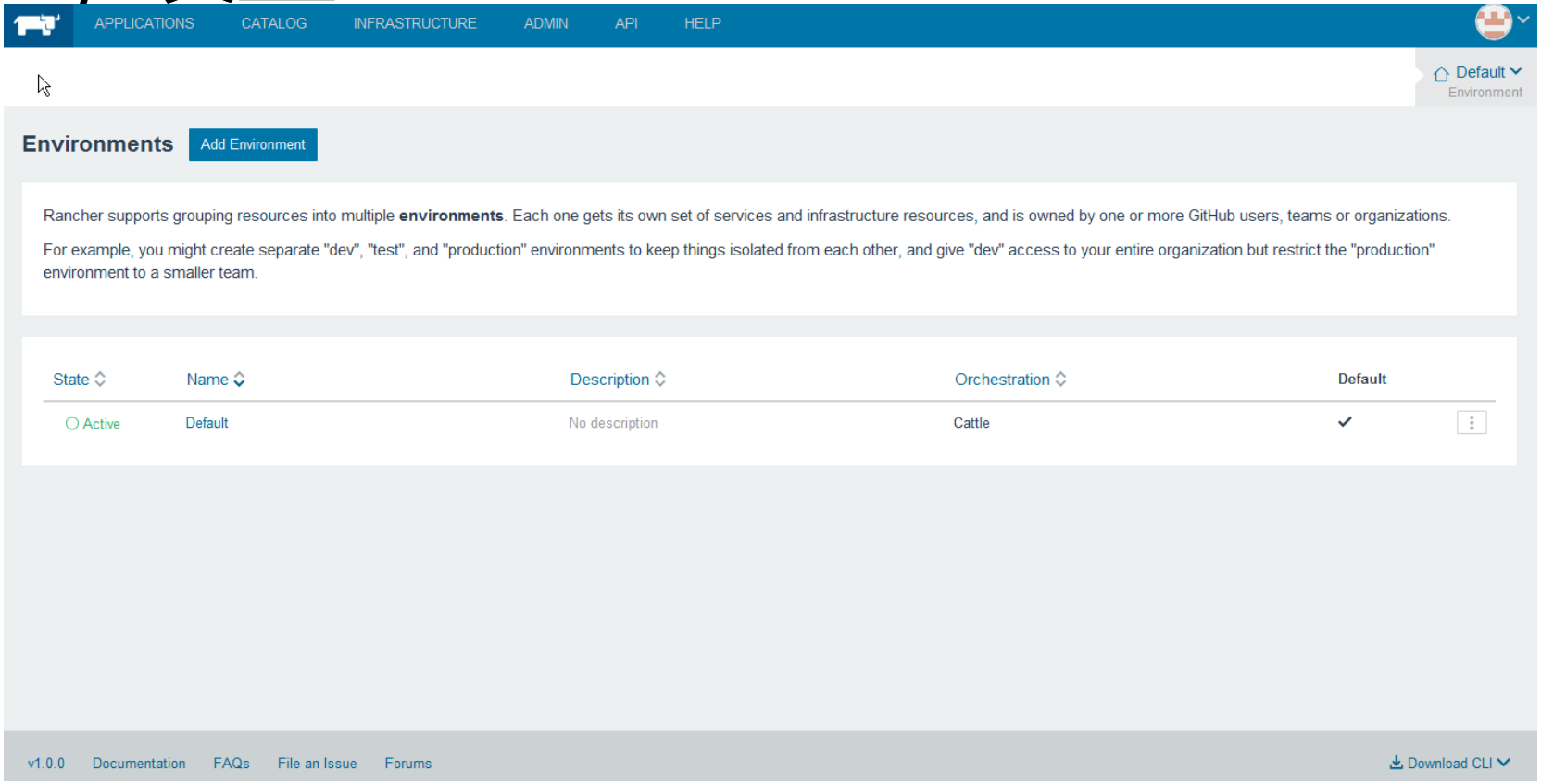
http://<SERVER_IP>:8080


192.168.2.28:8080/env/1a5/apps/welcome
APPLICATIONS
CATALOG
INFRASTRUCTURE
ADMIN
API
HELP
STACKS
Default
Environment
Adding your first Host
Before adding your first service or launching a container, you must add at least a single Linux host that supports Docker 1.9.1+ and be able to reach the Rancher server via HTTP. Rancher supports adding Linux hosts in the form of a virtual or physical machine from any public cloud providers, privately hosted clouds, or even bare metal servers. Learn More
Add Host
Adding your first Service
A service is simply a group of containers created from the same Docker image but extends Docker's "link" concept to leverage Rancher's lightweight distributed DNS service for service discovery. Services can be added individually or by deploying an item from the Catalog.
A service is also capable of leveraging other Rancher built-in services such as load balancers, health monitoring, upgrade support, and high-availability. Learn More
Add Service
Add From Catalog

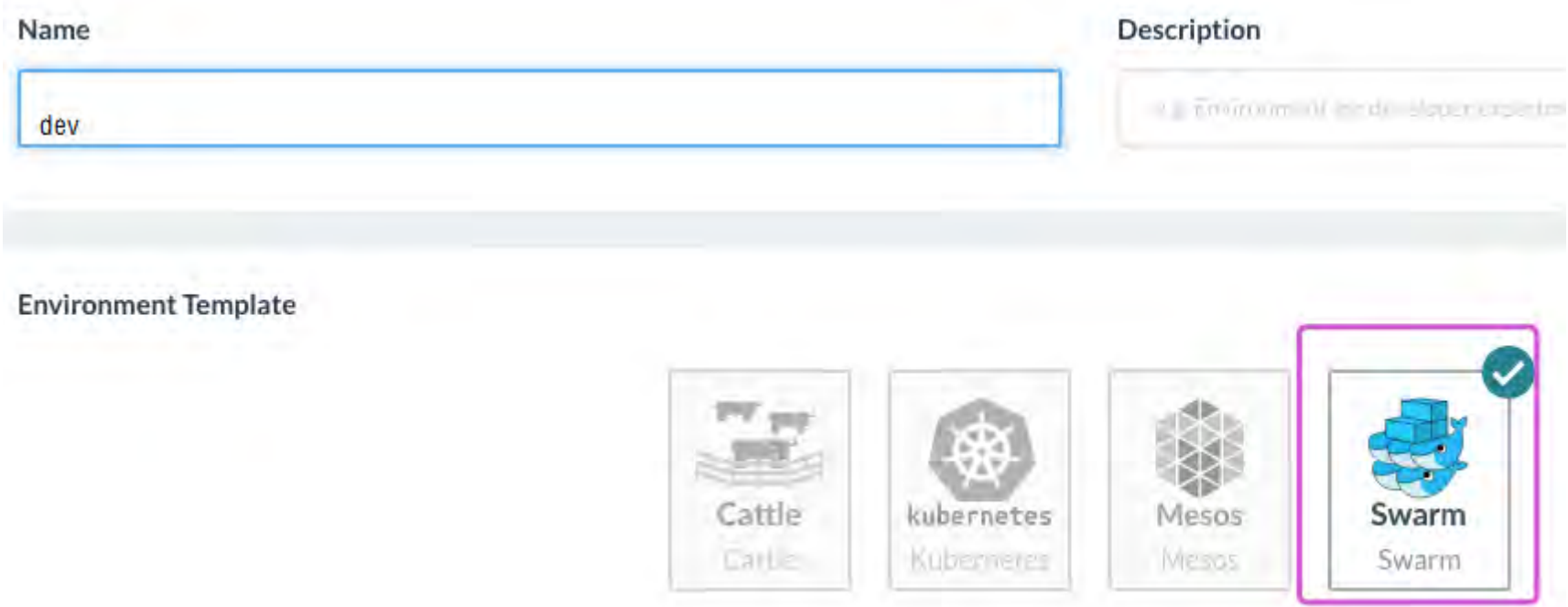
2，管理account

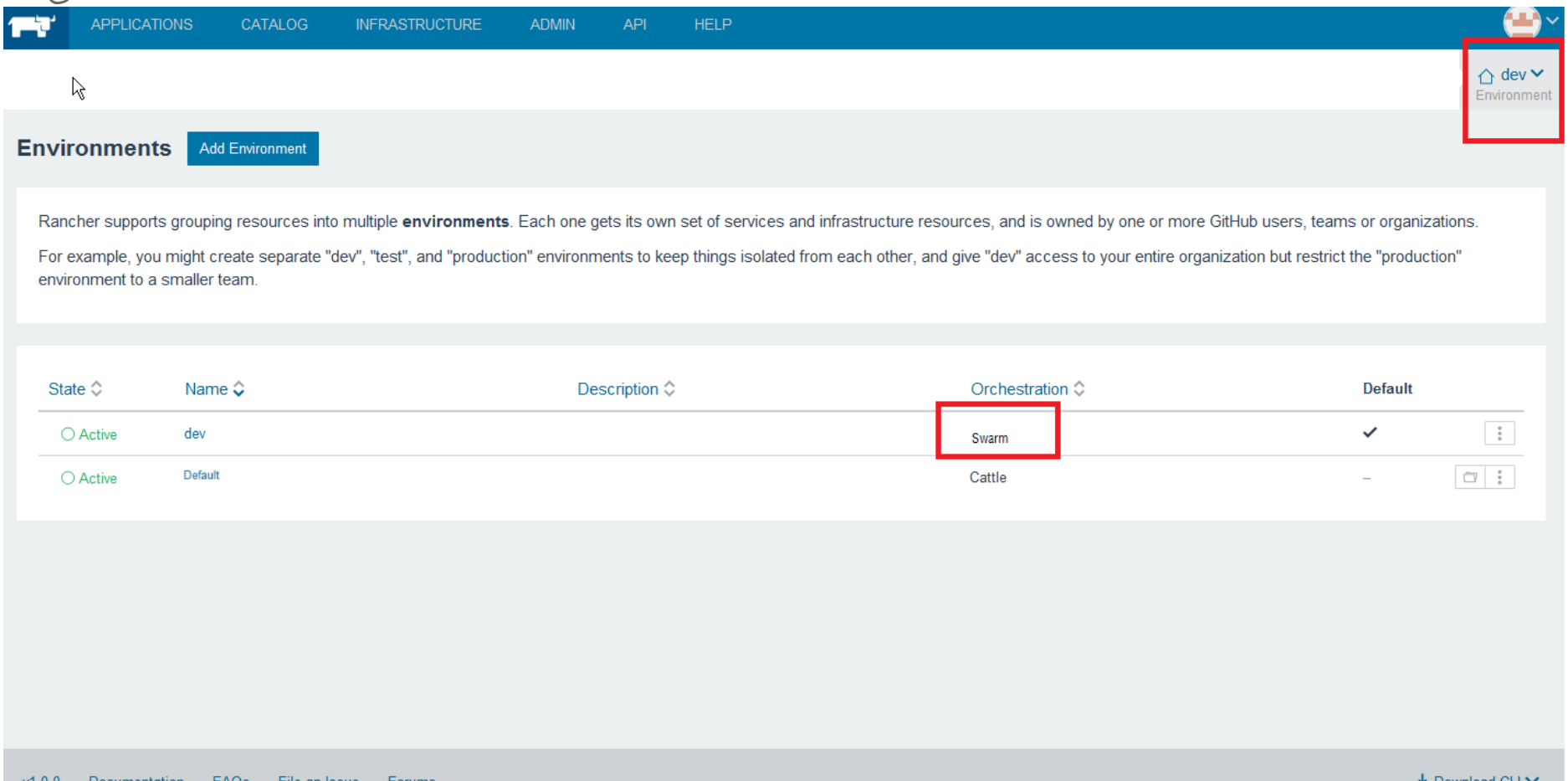

APPLICATIONS
CATALOG
INFRASTRUCTURE
ADMIN
API
HELP
AUDIT LOG
PROCESSES
ACCOUNTS
ACCESS CONTROL
SETTINGS
Access Control
Active Directory
GITHUB
LOCAL
OpenLDAP

3，设置Environment

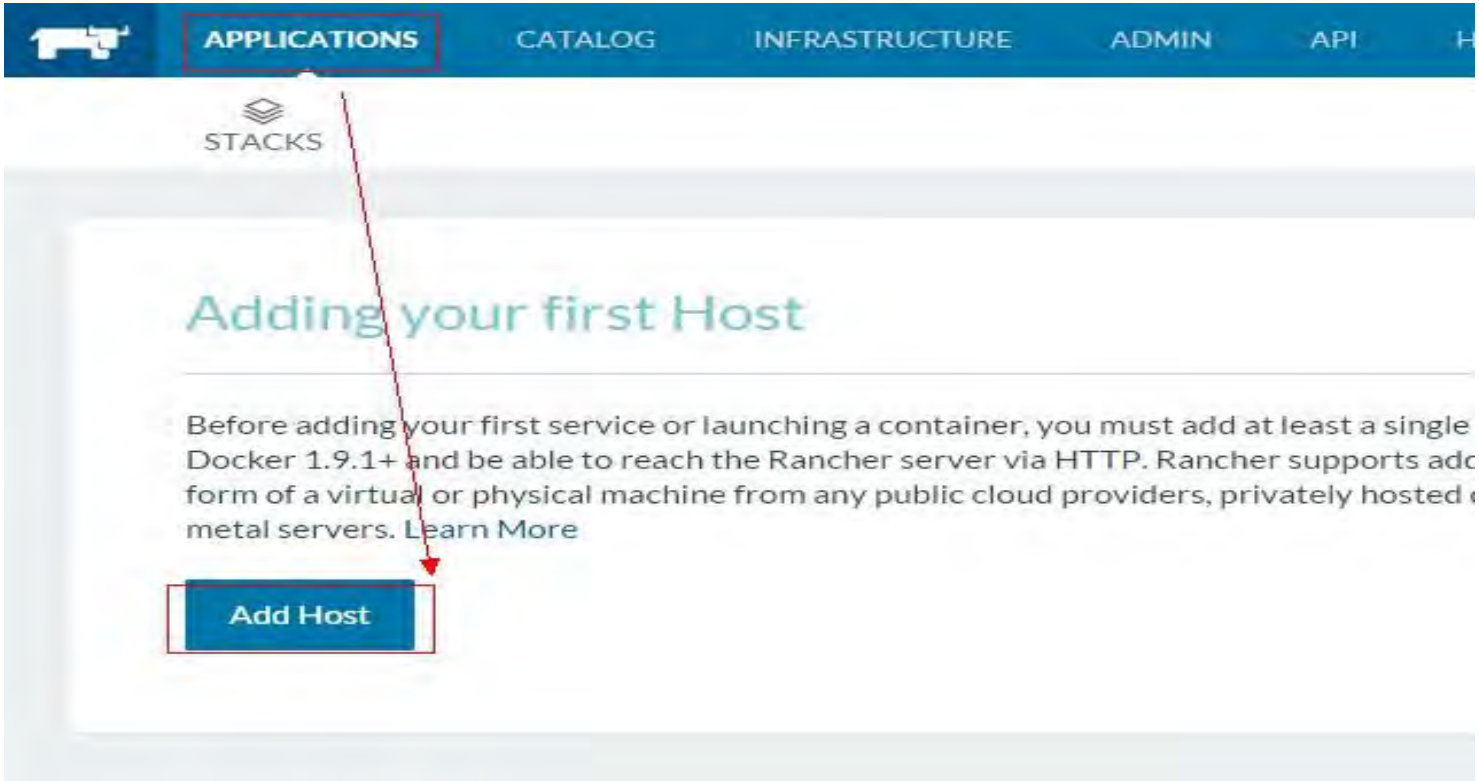


4，选择swarm作为引擎

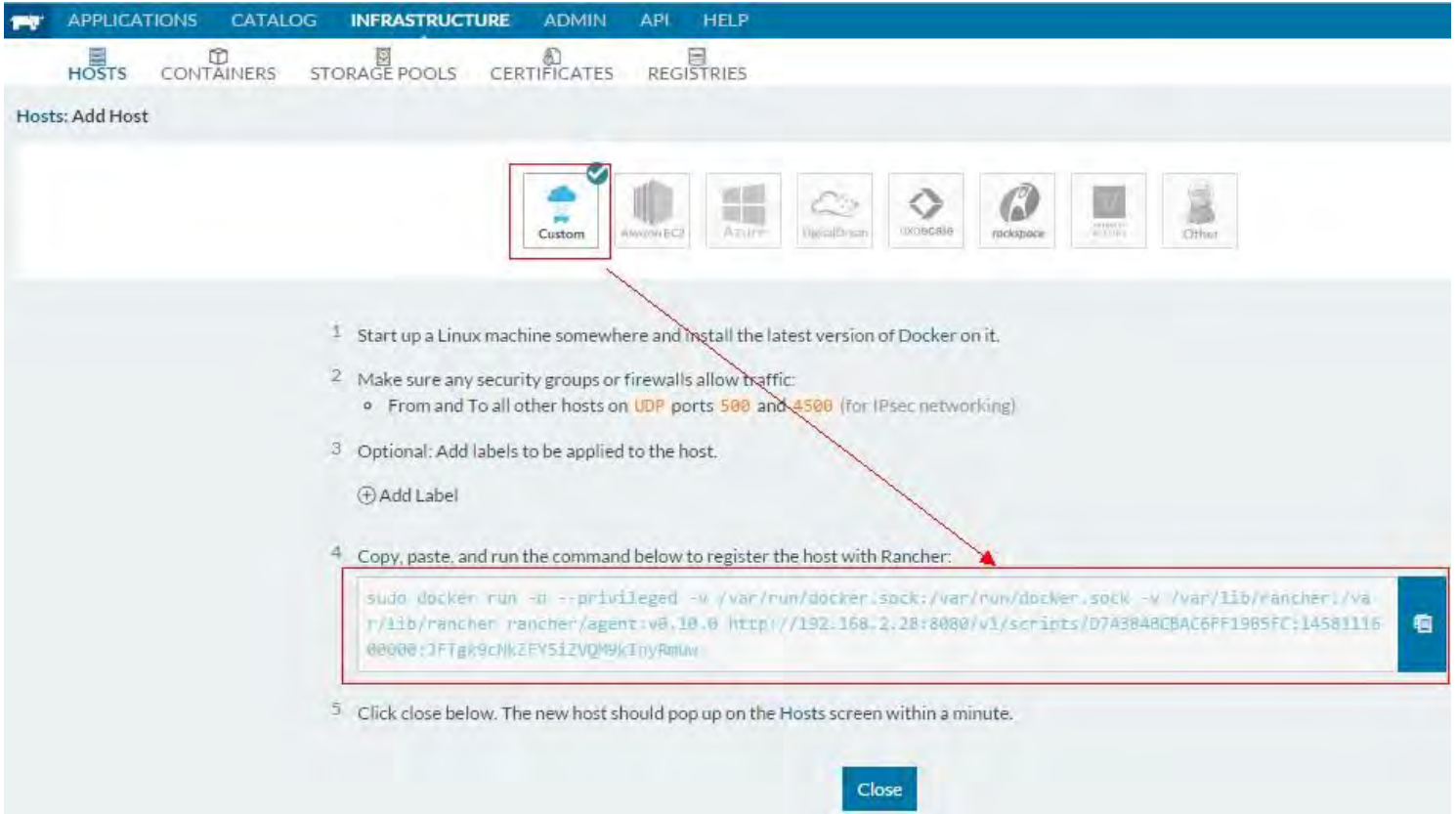



APPLICATIONS
CATALOG
INFRASTRUCTURE
ADMIN
API
HELP
dev
Environment
Environments
Add Environment
Rancher supports grouping resources into multiple environments. Each one gets its own set of services and infrastructure resources, and is owned by one or more GitHub users, teams or organizations.
For example, you might create separate "dev", "test", and "production" environments to keep things isolated from each other, and give "dev" access to your entire organization but restrict the "production" environment to a smaller team.
State
Name
Description
Orchestration
Default
Active
dev
Swarm
Active
Default
Cattle
v1.0.0
Documentation
FAQs
File an Issue
Forums
Download CLI

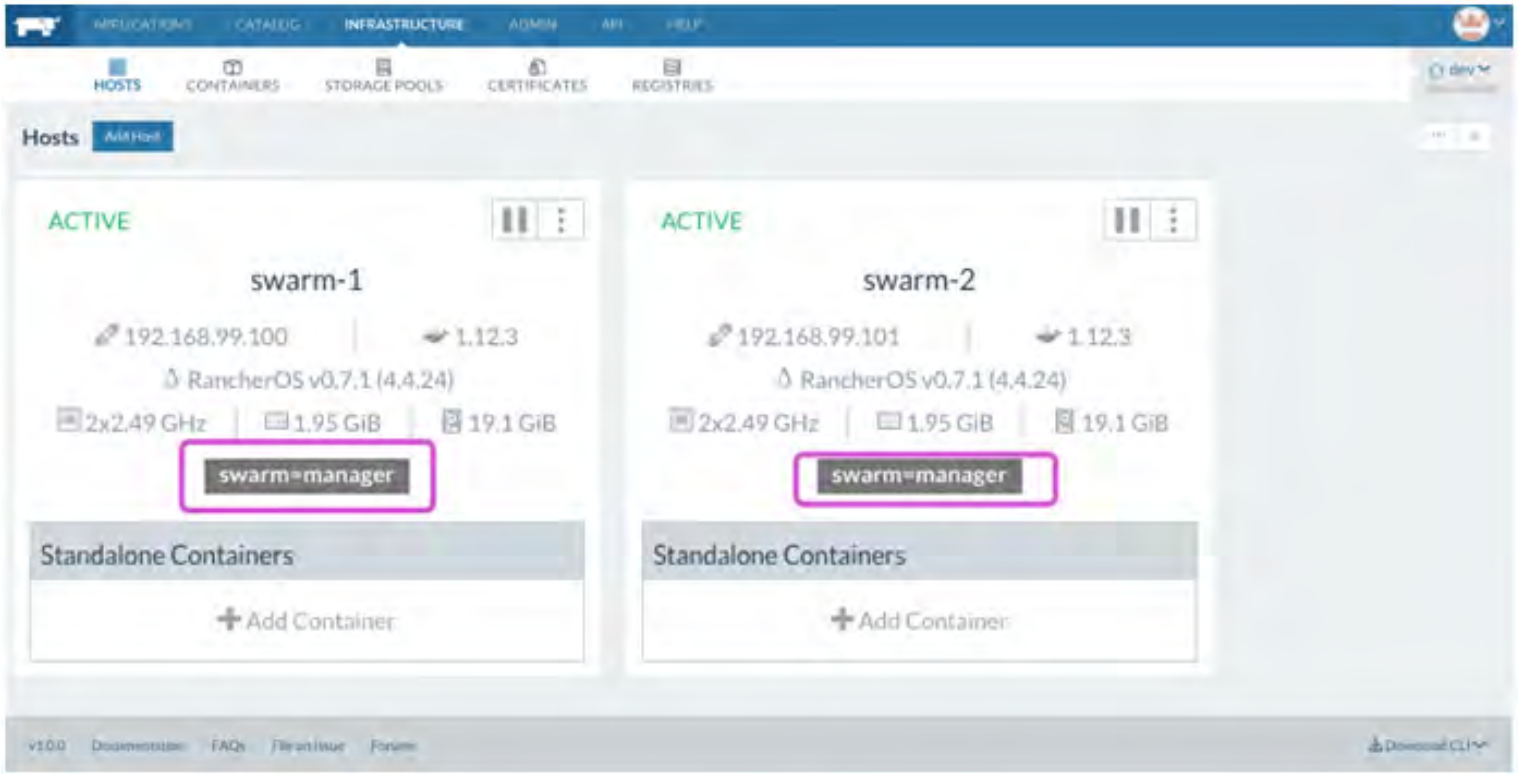
5，添加主机资源


APPLICATIONS
CATALOG
INFRASTRUCTURE
ADMIN
API
STACKS
Adding your first Host
Before adding your first service or launching a container, you must add at least a single
Docker 1.9.1+ and be able to reach the Rancher server via HTTP. Rancher supports adc
form of a virtual or physical machine from any public cloud providers, privately hosted
metal servers. Learn More
Add Host

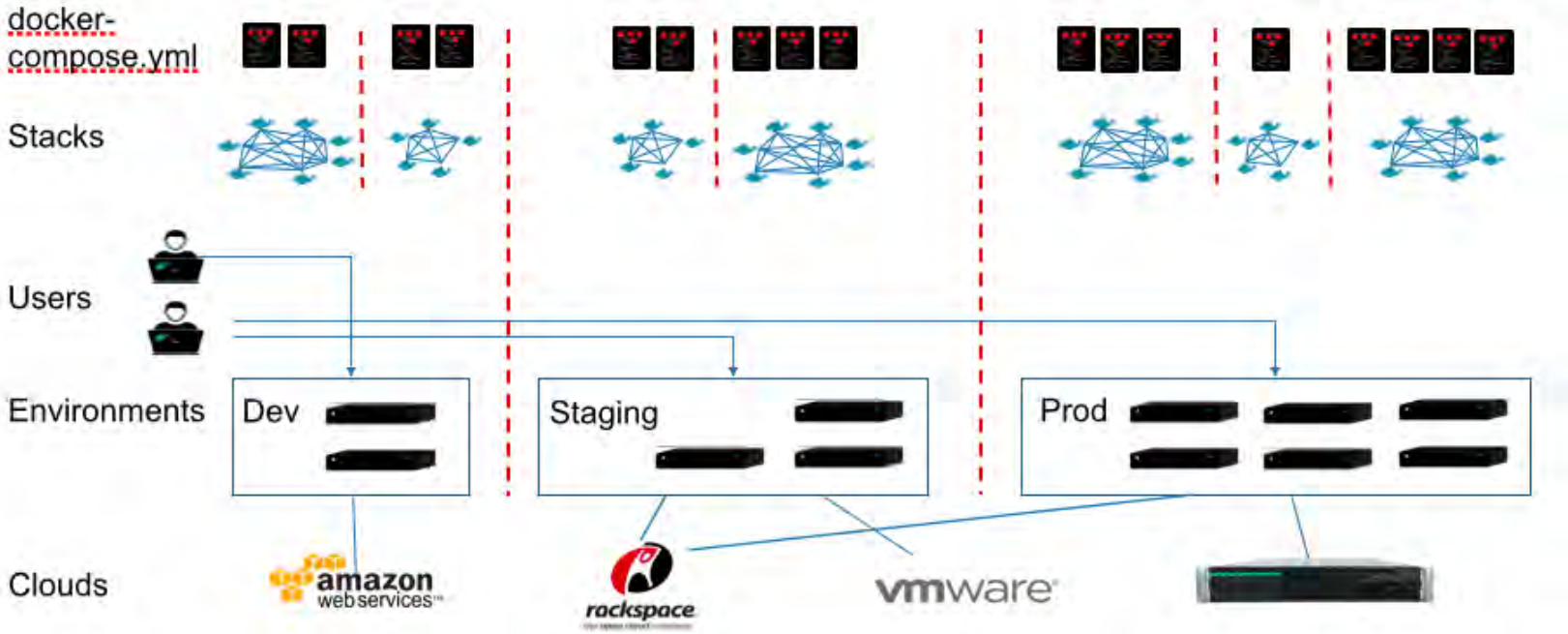
选择Add Host，弹出的界面选择 Custom，将此页面第4步框内出现的内容复制，在目标Host主机的命令行执行这个docker run命令，会自动安装启动一个Rancher Agent的容器：



6、按照同样的方式添加多个hosts



7，资源就绪，可以开始部署容器应用



接下来：
我们通过一个例子，用Rancher部署一整套服务

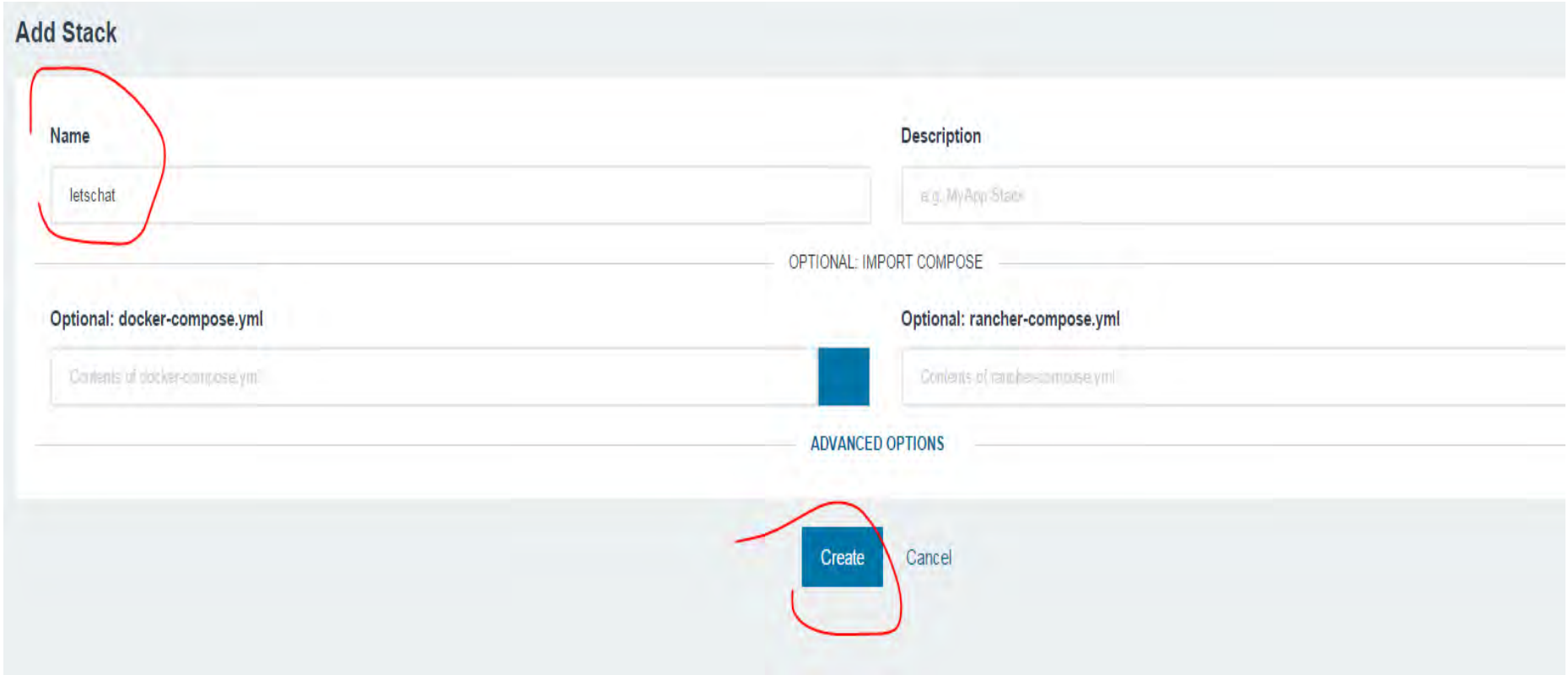
LetsChat application, for example, could consist of the following services:

1. A load balancer. The load balancer redirects Internet traffic to the “LetsChat” application.
2. A *web* service consisting of two “LetsChat” containers.
3. A *database* service consisting of one “Postgresql” container.

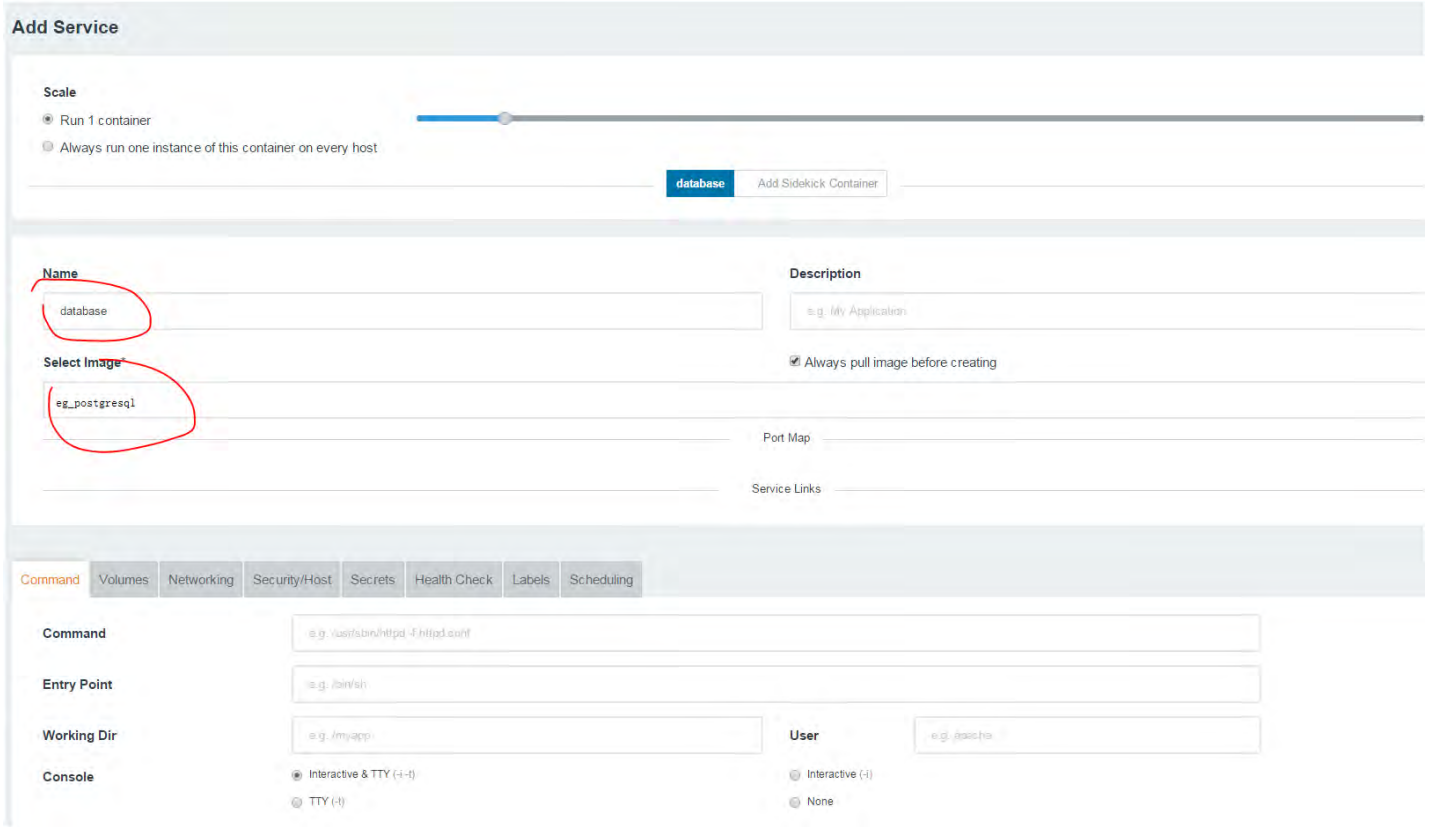
三个组件的连接关系如下：

Load balancer -> web service -> database

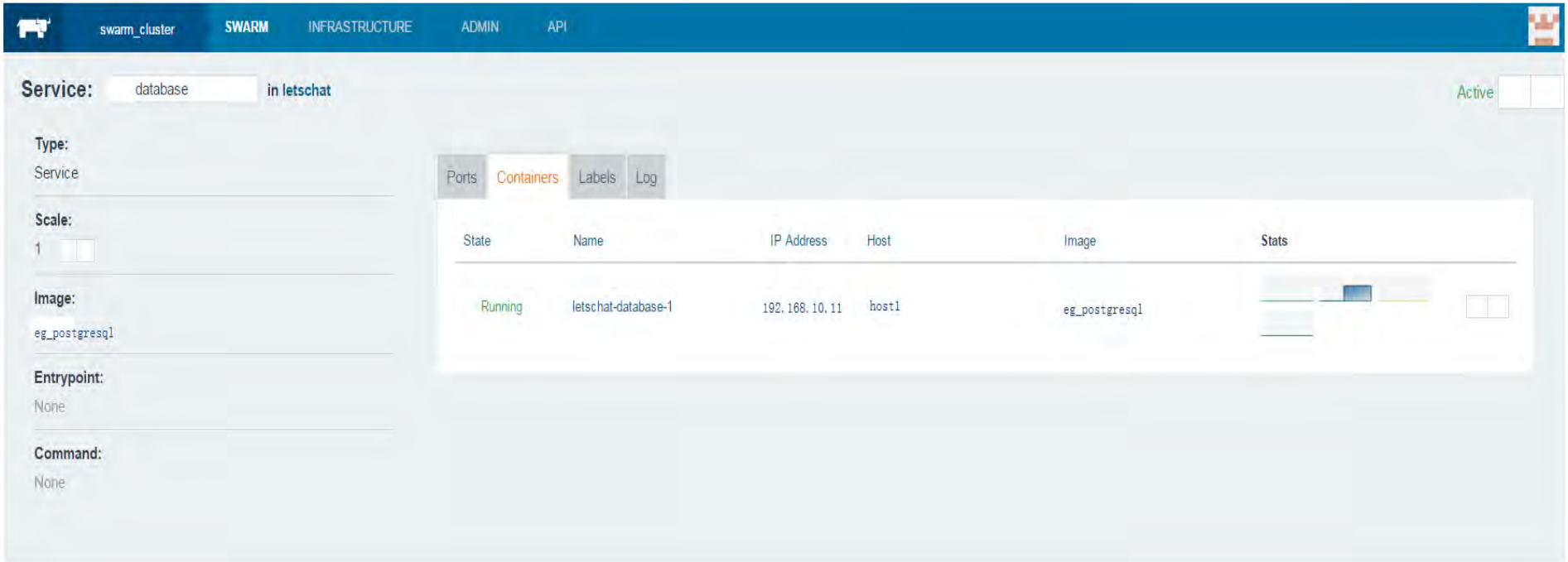
1、创建一个新的stack，命名为letschat



2、创建database的service

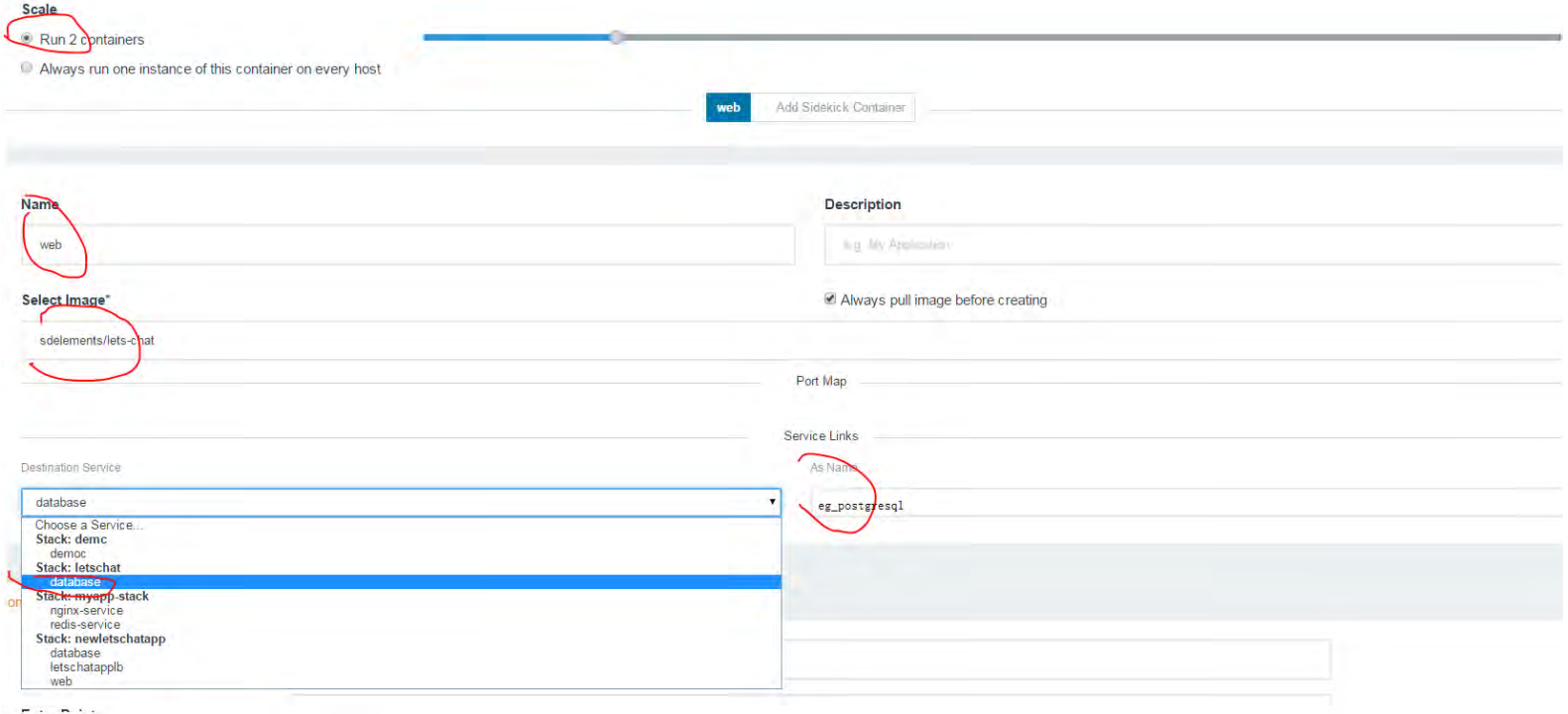


3、看到database已经启动

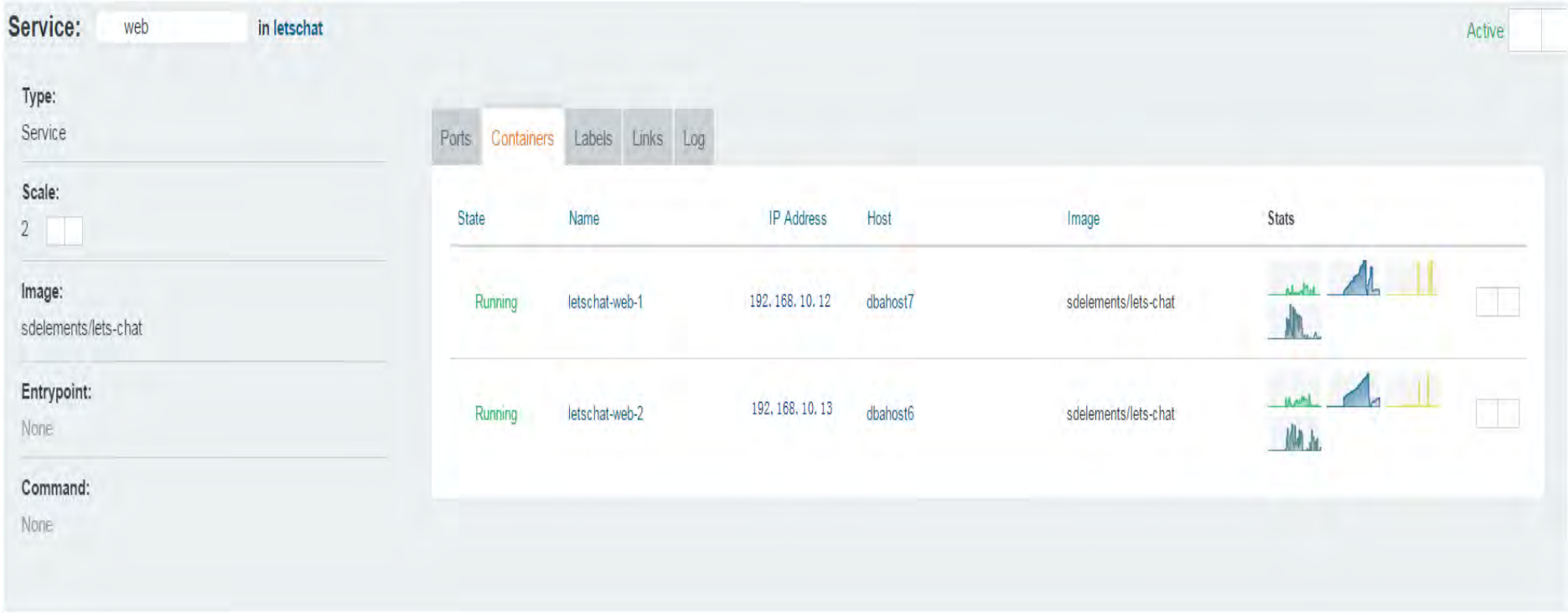


4、增加web的service

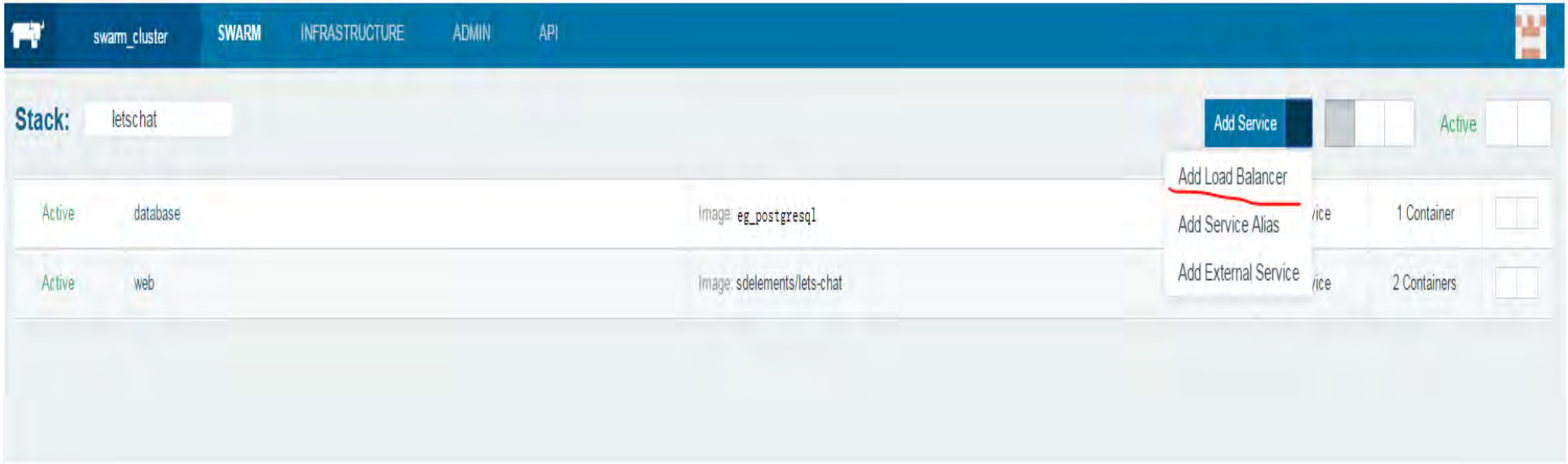
注意在Service Links处要增加前面我们创建的database的link：

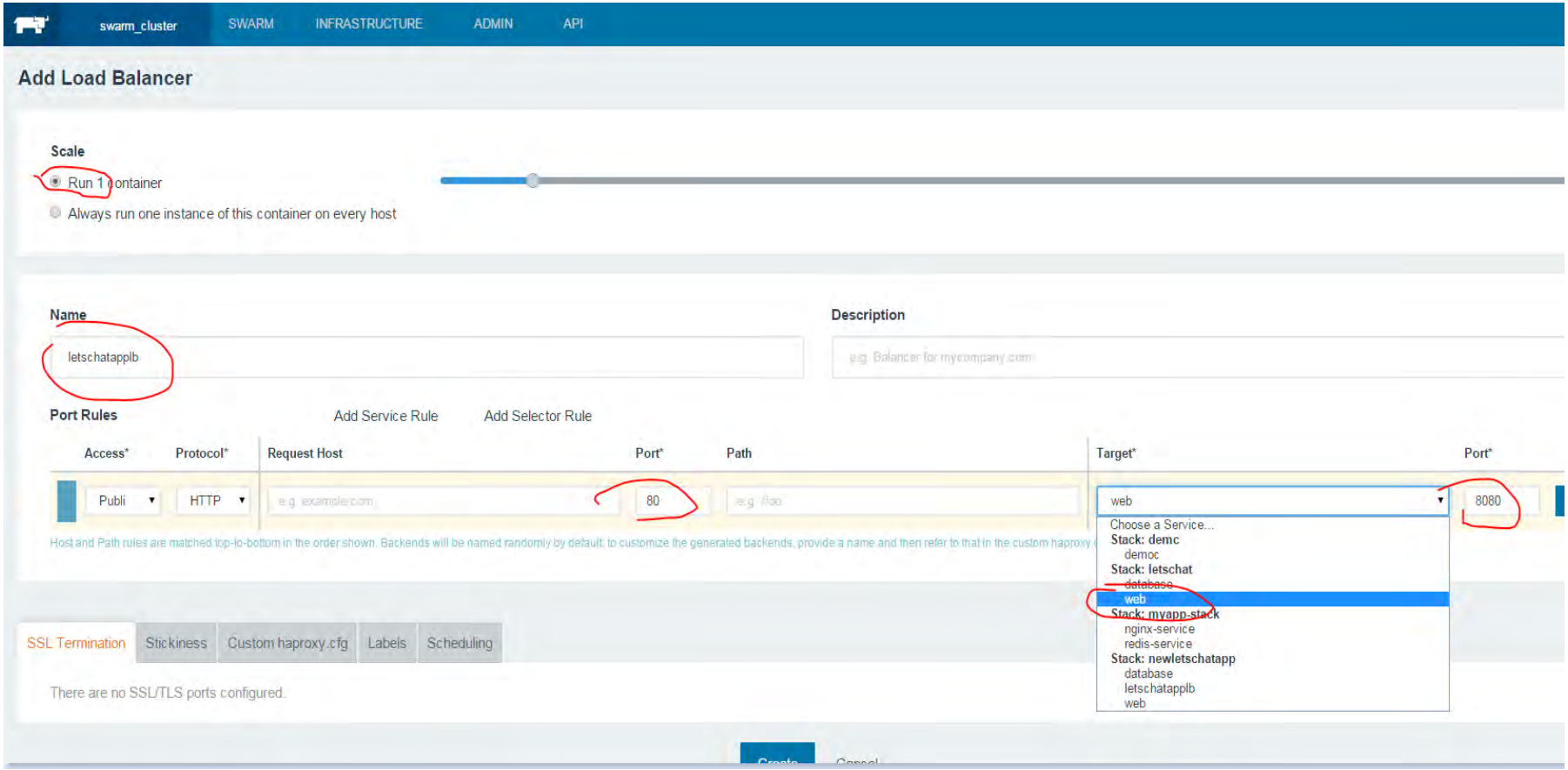


5、web service启动完毕

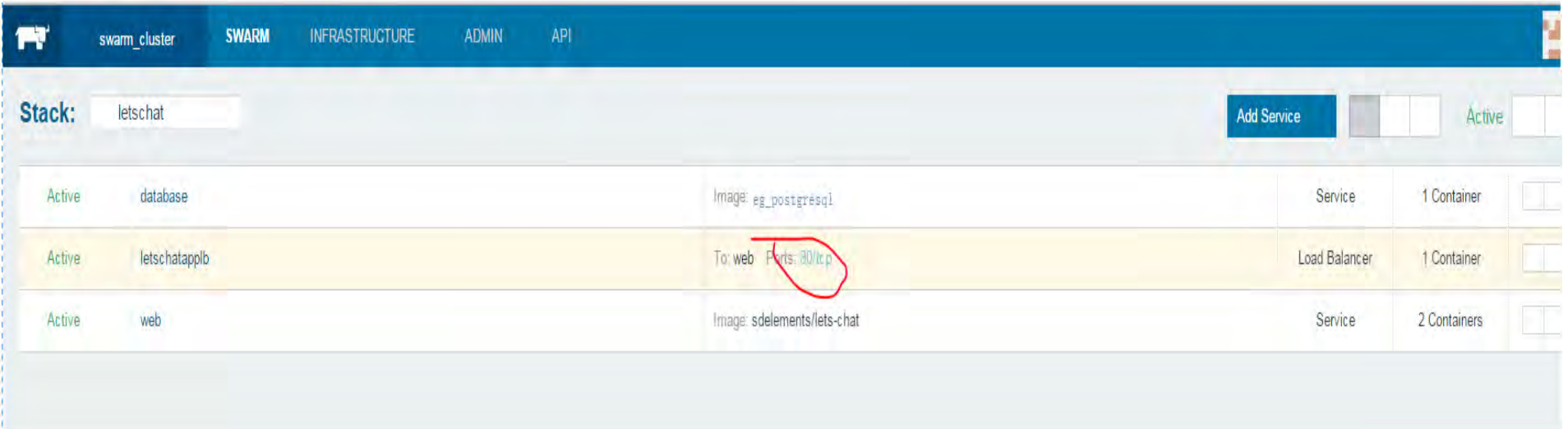


6，增加load banlancer的service：

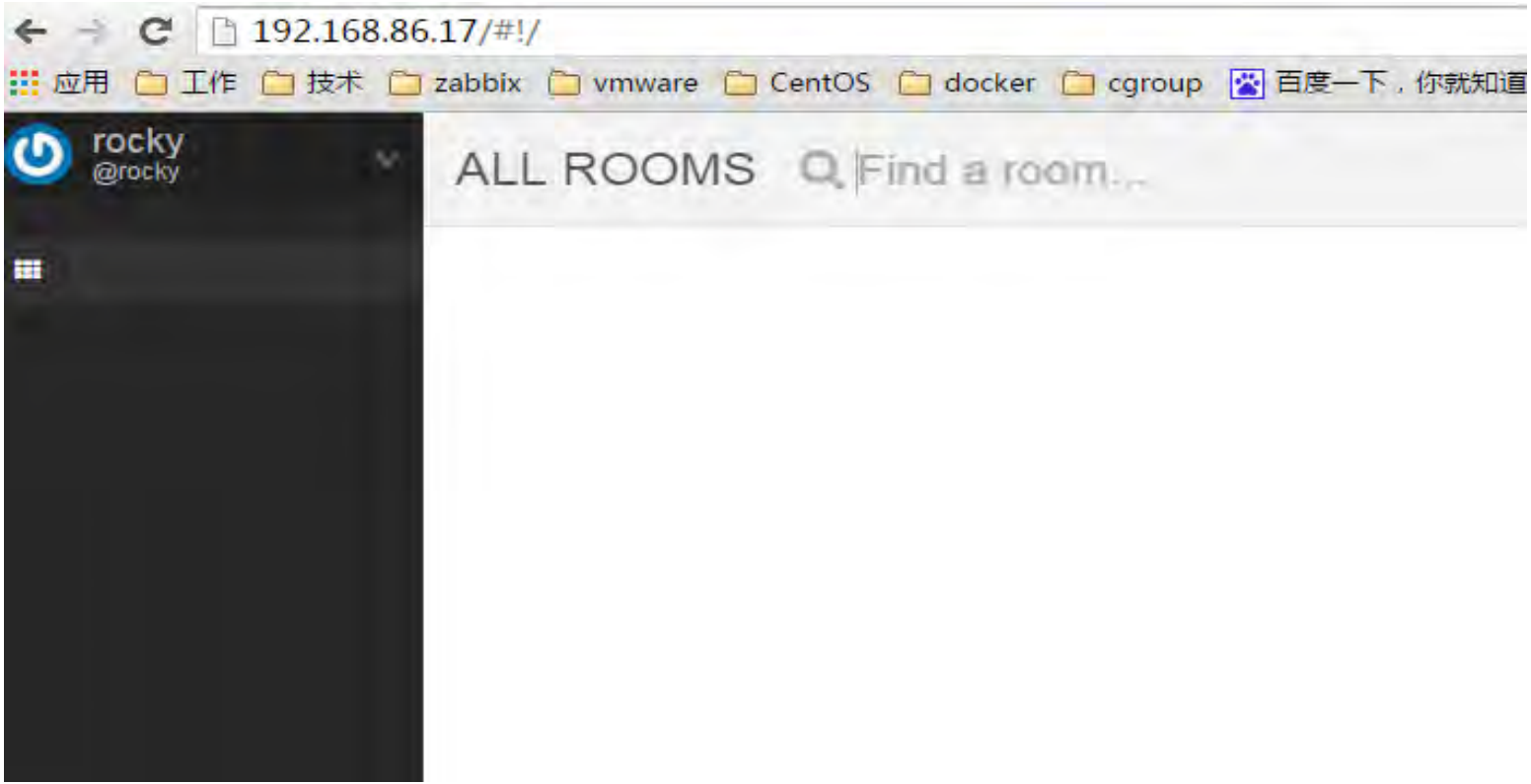



swarm_cluster
SWARM
INFRASTRUCTURE
ADMIN
API
Add Load Balancer
Scale
Run 1 container
Always run one instance of this container on every host
Name
letschatapplb
Description
e.g. Balancer for mycompany.com
Port Rules
Add Service Rule
Add Selector Rule
Access*
Protocol*
Request Host
Port*
Path
Target*
Port*
Publi
HTTP
e.g. example.com
80
e.g. /foo
web
8080
Host and Path rules are matched top-to-bottom in the order shown. Backends will be named randomly by default, to customize the generated backends, provide a name and then refer to that in the custom haproxy
Choose a Service...
Stack: democ
democ
Stack: letschat
database
web
Stack: myapp-stack
nginx-service
redis-service
Stack: newletschatapp
database
letschatapplb
web
SSL Termination
Stickiness
Custom haproxy.cfg
Labels
Scheduling
There are no SSL/TLS ports configured.

7，在statck页面，点击load balancer里面的80/tcp这个链接



打开letschat的网页，开始聊天吧：



我们也可以不通过UI，通过CLI来部署

DOCKER-COMPOSE.YML

```
VERSION: '2'
SERVICES:
 LETSCHATAPPLB:
  #IF YOU ONLY HAVE 1 HOST AND ALSO CREATED THE HOST IN THE UI,
  # YOU MAY HAVE TO CHANGE THE PORT EXPOSED ON THE HOST.
  PORTS:
  - 80:80/TCP
  LABELS:
    IO.RANCHER.CONTAINER.CREATE_AGENT: 'TRUE'
    IO.RANCHER.CONTAINER.AGENT.ROLE: ENVIRONMENTADMIN
  IMAGE: RANCHER/LB-SERVICE-HAPROXY:V0.4.2
 WEB:
  LABELS:
    IO.RANCHER.CONTAINER.PULL_IMAGE: ALWAYS
  TTY: TRUE
  IMAGE: SDELEMENTS/LETS-CHAT
  LINKS:
  - DATABASE:MONGO
  STDIN_OPEN: TRUE
 DATABASE:
  LABELS:
    IO.RANCHER.CONTAINER.PULL_IMAGE: ALWAYS
  TTY: TRUE
  IMAGE: EG_POSTGRESQL
  STDIN_OPEN: TRUE
```

RANCHER-COMPOSE.YML

```
VERSION: '2'
SERVICES:
 LETSCHATAPPLB:
  SCALE: 1
  LB_CONFIG:
    CERTS: []
    PORT_RULES:
    - HOSTNAME: ''
      PATH: ''
      PRIORITY: 1
      PROTOCOL: HTTP
      SERVICE: QUICKSTARTGUIDE/WEB
      SOURCE_PORT: 80
      TARGET_PORT: 8080
  HEALTH_CHECK:
    PORT: 42
    INTERVAL: 2000
    UNHEALTHY_THRESHOLD: 3
    HEALTHY_THRESHOLD: 2
    RESPONSE_TIMEOUT: 2000
 WEB:
  SCALE: 2
 DATABASE:
  SCALE: 1
```

执行命令部署docker应用

先配置好Rancher CLI：

$ rancher config

Set the Rancher URL

URL []: http://<SERVER_IP>:8080/

Set the access key, i.e. username

Access Key []: <accessKey_of_account_api_key>

Set the secret key, i.e. password

Secret Key []: <secretKey_of_account_api_key>

然后cd到保存有这两个文件的路径下：docker-compose.yml and rancher-compose.yml

只用执行一条命令，即可部署完毕整套服务：

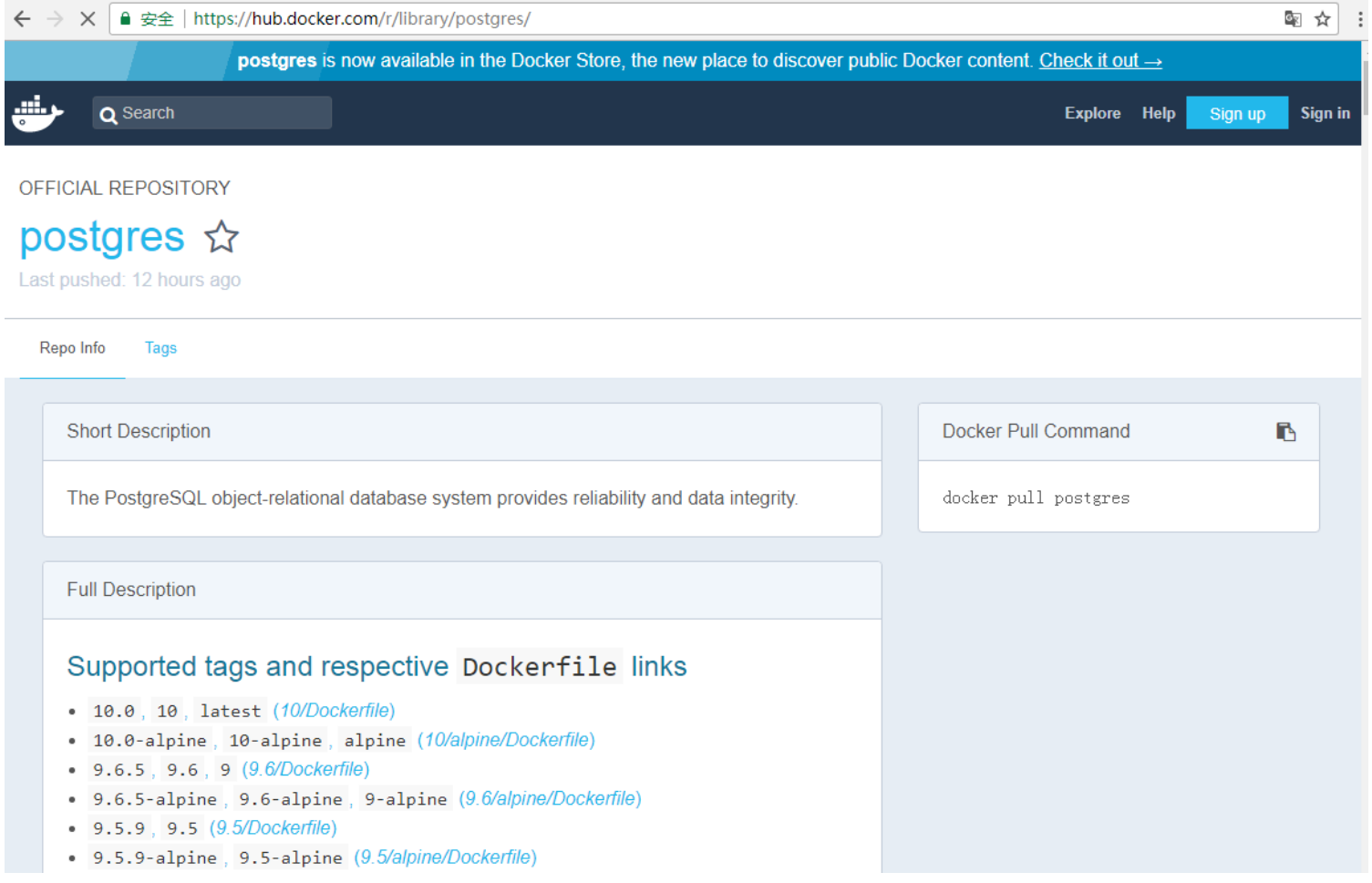
$ rancher up -d -s NewLetsChatApp

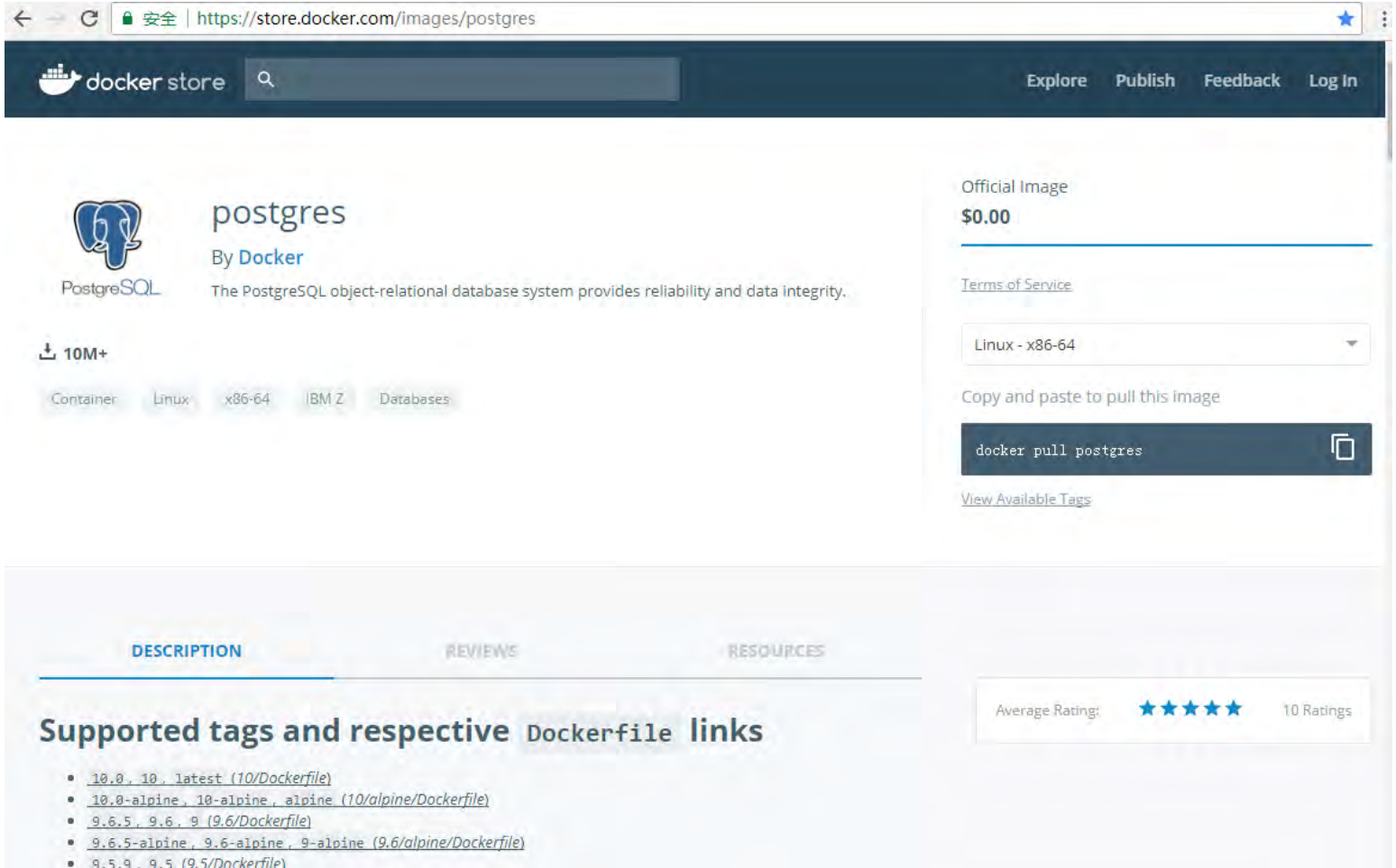
来自容器技术大会Container Day 2017的报道：

Rancher Labs与海航科技集团联合发布了海航科技容器公有云平台，并与华为共同推出基于容器的轻量级PaaS应用服务框架

Rancher Labs更隆重推出**首个能在同一平台上管理任何Kubernetes集群**的最新平台技术——Rancher 2.0，使用户能够**轻松的应用各种技术创新以及Kubernetes的生态系统**，而不需要重新组建一个完整的平台。

Rancher 2.0的新功能包括：
管理来自各处的Kubernetes：随着越来越多的云提供商支持Kubernetes集群服务，用户不再需要创建自己的集群。Rancher 2.0能够让用户管理来自诸如谷歌容器引擎（GKE）等云服务提供商的现有Kubernetes集群，亦可管理位于本地数据中心上的Kubernetes集群。
多集群管理：Rancher 2.0可集中管理用户身份验证、监测和健康检查，以便为IT管理员提供更高的可视性和控制力。Rancher 2.0充分利用Kubernetes中复杂的、基于角色的访问控制（RBAC）功能，为用户提供共享集群以及主机访问权限。
优异的用户体验：Rancher已有的优秀的用户体验再次得到了显著提升，使得Rancher平台上的Kubernetes体验也拥有了如Docker命令行般的简洁，以及Docker Compose般的优雅。用户在笔记本电脑上就能够在建立Kubernetes集群方面获得同样出色的体验。
丰富的应用目录：Rancher目录已得到扩展，可支持Docker Compose、Kubernetes模板和Helm charts，以便用户能够访问更多容器化应用。


安全 | https://hub.docker.com/r/library/postgres/
postgres is now available in the Docker Store, the new place to discover public Docker content. Check it out →
Search
Explore
Help
Sign up
Sign in
OFFICIAL REPOSITORY
postgres
Last pushed: 12 hours ago
Repo Info
Tags
Short Description
The PostgreSQL object-relational database system provides reliability and data integrity.
Docker Pull Command
docker pull postgres
Full Description
Supported tags and respective Dockerfile links
10.0, 10, latest (10/Dockerfile)
10.0-alpine, 10-alpine, alpine (10/alpine/Dockerfile)
9.6.5, 9.6, 9 (9.6/Dockerfile)
9.6.5-alpine, 9.6-alpine, 9-alpine (9.6/alpine/Dockerfile)
9.5.9, 9.5 (9.5/Dockerfile)
9.5.9-alpine, 9.5-alpine (9.5/alpine/Dockerfile)


安全 | https://store.docker.com/images/postgres
docker store
Explore
Publish
Feedback
Log in
PostgreSQL
postgres
By Docker
The PostgreSQL object-relational database system provides reliability and data integrity.
10M+
Container
Linux
x86-64
IBM Z
Databases
Official Image
$0.00
Terms of Service
Linux - x86-64
Copy and paste to pull this image
docker pull postgres
View Available Tags
DESCRIPTION
REVIEWS
RESOURCES
Supported tags and respective Dockerfile links
10.0, 10, latest (10/Dockerfile)
10.0-alpine, 10-alpine, alpine (10/alpine/Dockerfile)
9.6.5, 9.6, 9 (9.6/Dockerfile)
9.6.5-alpine, 9.6-alpine, 9-alpine (9.6/alpine/Dockerfile)
9.5.9, 9.5 (9.5/Dockerfile)
Average Rating:
10 Ratings

部分内容参考如下网站，鸣谢!

https://www.docker.com/

https://www.cnrancher.com/

http://rancher.com

http://dockone.io

http://tonybai.com/2016/04/14/an-introduction-about-rancher/

http://blog.chinaunix.net/uid-29757900-id-5676591.html

Thanks!