

# 第10章-容器管理与弹性扩容

学习目标：

## 1 容器管理工具Rancher

### 1.1 什么是Rancher

Rancher是一个开源的企业级全栈化容器部署及管理平台。Rancher为容器提供一揽子基础架构服务：CNI兼容的网络服务、存储服务、主机管理、负载均衡、防护墙..... Rancher让上述服务跨越公有云、私有云、虚拟机、物理机环境运行，真正实现一键式应用部署和管理。

<https://www.cnrancher.com/>

### 1.2 Rancher安装

(1) 下载Rancher 镜像

```
docker pull rancher/server
```

(2) 创建Rancher容器

```
docker run -d --name=rancher --restart=always -p 9090:8080 rancher/server
```

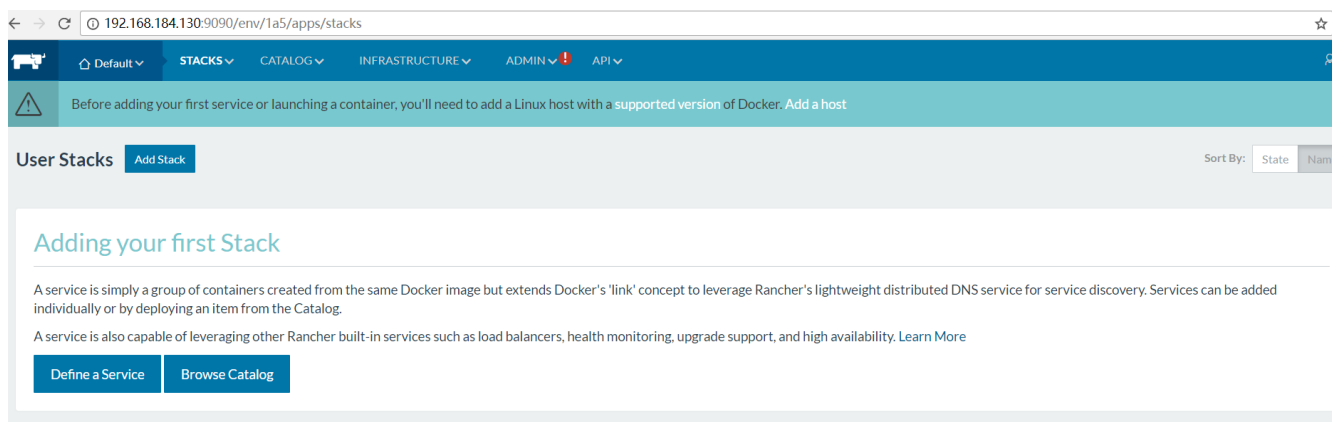
restart为重启策略

- no，默认策略，在容器退出时不重启容器
- on-failure，在容器非正常退出时（退出状态非0），才会重启容器
  - on-failure:3，在容器非正常退出时重启容器，最多重启3次
- always，在容器退出时总是重启容器
- unless-stopped，在容器退出时总是重启容器，但是不考虑在Docker守护进程启动时就已经停止了容器

(3) 在浏览器输入地址: <http://192.168.184.136:9090> 即可看到高端大气的欢迎页

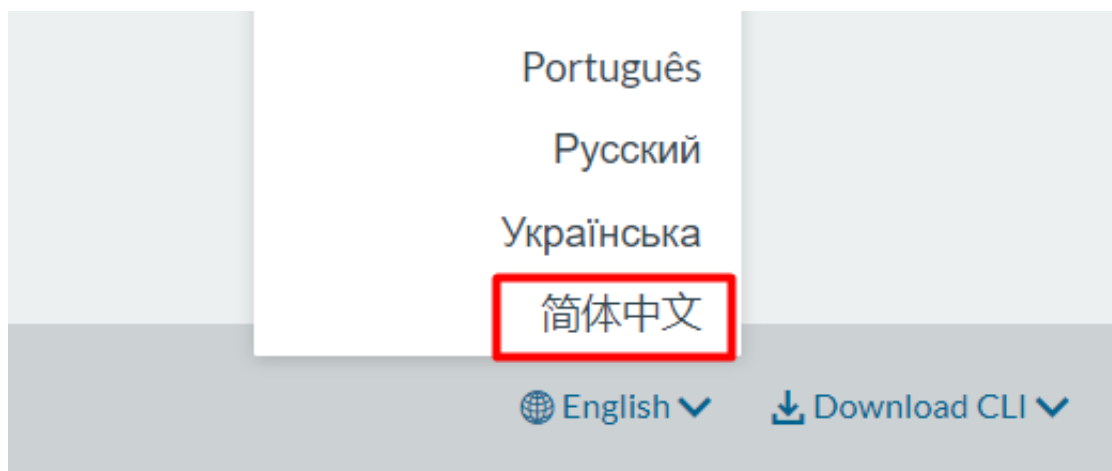


点击Got It 进入主界面



(4) 切换至中文界面

点击右下角的English 在弹出菜单中选择中文



切换后我们就可以看到亲切的中文界面啦~



## 1.3 Rancher初始化

### 1.3.1 配置私有仓库

#### (1) 修改daemon.json

```
vi /etc/docker/daemon.json
```

添加以下内容，保存退出。

```
{"insecure-registries":["192.168.184.135:5000"]}
```

此步用于让 docker信任私有仓库地址

#### (3) 重启docker 服务

```
systemctl restart docker
```

### 1.3.2 添加环境

Rancher 支持将资源分组归属到多个环境。每个环境具有自己独立的基础架构资源及服务，并由一个或多个用户、团队或组织所管理。

例如，您可以创建独立的“开发”、“测试”及“生产”环境以确保环境之间的安全隔离，将“开发”环境的访问权限赋予全部人员，但限制“生产”环境的访问权限给一个小的团队。

#### (1) 选择“Default -->环境管理” 菜单

在添加第一个服务或容器之前，必须至少添加一台安装了支持的Docker版本的Linux主机。添加主机

**环境** 添加环境

Rancher 支持将资源分组归属到多个**环境**。每个环境具有自己独立的基础架构资源及服务，并由一个或多个用户、团队或组织所管理。

例如，您可以创建独立的“开发”、“测试”及“生产”环境以确保环境之间的安全隔离，将“开发”环境的访问权限赋予全部人员，但限制“生产”环境的访问权限给一个小的团队。

状态	名称	描述	模板	编排
Unhealthy	Default	无描述	Cattle	Cattle

## (2) 填写名称，点击“创建”按钮

添加环境

名称: tensquare\_dev 描述: 十次方开发环境

环境模板

Cattle (选中) Kubernetes Mesos Swarm Windows

Orchestration: Cattle  
Framework: Network Services, Scheduler, Healthcheck Service  
Networking: Rancher IPsec

## (3) 按照上述步骤，添加十次方测试环境和生产环境

状态	名称	描述	模板	编排	默认
Unhealthy	Default	无描述	Cattle	Cattle	✓
Unhealthy	tensquare_dev	十次方开发环境	Cattle	Cattle	-
Active	tensquare_pro	十次方生产环境	Cattle	Cattle	-
Unhealthy	tensquare_test	十次方测试环境	Cattle	Cattle	-

## (4) 你可以通过点击logo右侧的菜单在各种环境下切换

tensquare\_dev 应用 应用商店 基础架构 系统管理 API

在添加第一个服务或容器之前，必须至少添加一台安装了支持的Docker版本的Linux主机。添加主机

**用户应用** 添加应用

创建第一个应用

服务是一组由相同docker镜像创建的容器，服务扩展了Docker的“link”概念以利用Rancher的轻量级分布式DNS服务用于服务发现。服务可以单独添加或通过应用商店部署。

服务也能够利用其他Rancher内置服务，如负载均衡、健康监控、升级支持以及高可用。了解更多

定义一个服务 浏览应用商店

## 1.3.3 添加主机

(1) 选择基础架构-->主机 菜单，点击添加主机



(2) 拷贝脚本



(3) 在服务器（虚拟机）上运行脚本



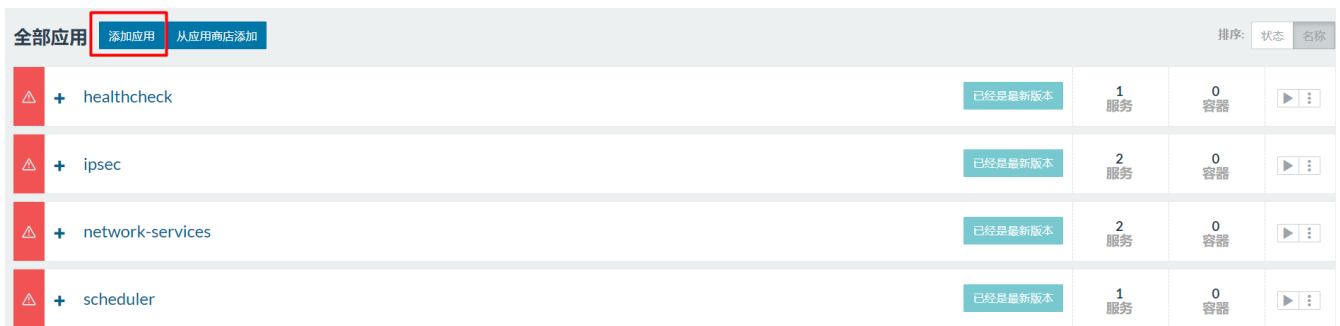
```
[root@pinyoyougou-docker ~]# sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:  
:9090/v1/scripts/92CAACB4B42BDB7A699A:1514678400000:wp9mbkDN4zyhivUiTurhetHS3Y  
Unable to find image 'rancher/agent:v1.2.10' locally  
Trying to pull repository docker.io/rancher/agent ...  
v1.2.10: Pulling from docker.io/rancher/agent  
b3e1c725a85f: Pull complete  
6a710864a9fc: Pull complete  
d0ac3b234321: Pull complete  
87f567b5cf58: Pull complete  
063e24b217c4: Pull complete  
d0a3f58caef0: Pull complete  
16914729cfd3: Pull complete  
4956f3e025a2: Pull complete  
64292afc18d7: Pull complete  
Digest: sha256:5618f36cdb6c6fe25baa3e72977f3a226d43ffd1d74dda6a8860e32b73a1e171  
  
INFO: Running Agent Registration Process, CATTLE_URL=http://192.168.184.130:9090/v1  
INFO: Attempting to connect to: http://192.168.184.130:9090/v1  
INFO: http://192.168.184.130:9090/v1 is accessible  
INFO: Configured Host Registration URL info: CATTLE_URL=http://192.168.184.130:9090/v1 ENV_URL=http://192.168.184.130:9090/v1  
INFO: Inspecting host capabilities  
INFO: Boot2Docker: false  
INFO: Host writable: true  
INFO: Token: xxxxxxxx  
INFO: Running registration  
INFO: Printing Environment  
INFO: ENV: CATTLE_ACCESS_KEY=4A9796CCA881CF11FEDD  
INFO: ENV: CATTLE_HOME=/var/lib/cattle  
INFO: ENV: CATTLE_REGISTRATION_ACCESS_KEY=registrationToken  
INFO: ENV: CATTLE_REGISTRATION_SECRET_KEY=xxxxxxx  
INFO: ENV: CATTLE_SECRET_KEY=xxxxxxx  
INFO: ENV: CATTLE_URL=http://192.168.184.130:9090/v1  
INFO: ENV: DETECTED_CATTLE_AGENT_IP=172.17.0.1  
INFO: ENV: RANCHER_AGENT_IMAGE=rancher/agent:v1.2.10  
INFO: Launched Rancher Agent: 9cdd147e7f79d92a027064ce8eccba4290a1ca99db403929d97f66ee2dbc832d  
[root@pinyoyougou-docker ~]#
```

(4) 点击关闭按钮后，会看到界面中显示此主机。我们可以很方便地管理主机的每个容器的开启和关闭



## 1.3.4 添加应用

点击应用-->全部(或用户) ， 点击“添加应用”按钮



填写名称和描述

### 添加应用

名称	描述
<input type="text" value="tensquare"/>	<input type="text" value="十次方"/>
可选:导入COMPOSE	
可选:docker-compose.yml	可选:rancher-compose.yml
<input type="text" value="docker-compose.yml文件的内容"/>	<input type="text" value="rancher-compose.yml文件的内容"/>
高级选项 ^	
<div>创建 取消</div>	

点击“创建”按钮，列表中增加了新增的应用

全部应用		添加应用	从应用商店添加	排序:	状态	名称
+	healthcheck	已经是最新版本	1 服务	0 容器		
+	ipsec	已经是最新版本	2 服务	0 容器		
+	network-services	已经是最新版本	2 服务	0 容器		
+	scheduler	已经是最新版本	1 服务	0 容器		
+	tensquare 十次方	添加服务	0 服务	0 容器		

## 2 数据库环境搭建

### 2.1 Redis容器化部署

进入应用，点击添加服务 名称redis，镜像redis，端口映射6379

名称	描述
<input type="text" value="redis"/>	<input type="text" value="redis"/>
选择镜像*	<input type="checkbox"/> 创建前总是拉取镜像
<input type="text" value="redis"/>	
+ 端口映射	
公开主机端口	私有容器端口
<input type="text" value="6379"/>	<input type="text" value="6379"/>
/ TCP	

创建后使用客户端测试链接

```
redis-cli -h 192.168.184.136
```

测试成功



## 2.2 MySQL容器部署

镜像: centos/mysql-57-centos7 增加数据库服务

名称	描述
mysql	mysql

选择镜像\* ☐ 创建前总是拉取镜像

centos/mysql-57-centos7

端口映射

公开主机端口	私有容器端口	协议
3306	3306	TCP

[显示主机IP选项](#)

注意: 添加环境变量 MYSQL\_ROOT\_PASSWORD=123456

环境变量

[+ 添加环境变量](#)

变量	值
MYSQL_ROOT_PASSWORD	123456

高级技巧: 在键(Key)输入栏中粘贴一行或多行的key=value键值对能够批量输入。

点击创建按钮, 完成创建 上述操作相当于以下docker命令

```
docker run -di --name mysql -p 3306:3306 -e MYSQL_ROOT_PASSWORD=123456 centos/mysql-57-centos7
```

Active	mysql ①	镜像: centos/mysql-57-centos7 端口: 3306	服务	1 容器	<a href="#">+</a> <a href="#">-</a>
--------	---------	--------------------------------------	----	------	-------------------------------------

完成后服务列表中存在并且状态为激活 使用SQLyog测试链接, 执行建表语句

## 2.3 MongoDB容器化部署

名称mongo 镜像mongo 端口映射27017

## 2.4 Elasticsearch容器化部署

名称elasticsearch 镜像elasticsearch:5.6.8 端口映射9300 9200

添加后, 浏览器测试: <http://192.168.184.136:9200/>

## 2.5 RabbitMQ容器化部署

镜像: rabbitmq:management 端口映射 5671 5672 4369 15671 15672 25672

名称	描述
<input type="text" value="rabbitmq"/>	<input type="text" value="rabbitmq"/>

选择镜像\* ☐ 创建前总是拉取镜像

+ 端口映射

公开主机端口	私有容器端口	协议
<input type="text" value="5671"/>	<input type="text" value="5671"/>	TCP
<input type="text" value="5672"/>	<input type="text" value="5672"/>	TCP
<input type="text" value="4369"/>	<input type="text" value="4369"/>	TCP
<input type="text" value="15671"/>	<input type="text" value="15671"/>	TCP
<input type="text" value="15672"/>	<input type="text" value="15672"/>	TCP
<input type="text" value="25672"/>	<input type="text" value="25672"/>	TCP

显示主机IP选项

浏览器访问 <http://192.168.184.136:15672/>

## 3 微服务容器化部署

### 3.1 eureka部署

(1) 在用户应用界面中点击“添加服务”

用户应用 添加应用 从应用商店添加

tensquare 十次方

添加服务

0 服务

0 容器

(2) 填写名称、描述、镜像和端口映射，点击创建按钮

名称eureka 镜像 192.168.184.135:5000/tensquare\_eureka:1.0-SNAPSHOT

#### 添加服务

数量

☒ Run 1 个容器

☐ 总是在每台主机上运行一个此容器的实例

+ 添加从容器

---

名称  描述

选择镜像\*  ☒ 创建前总是拉取镜像

+ 端口映射

公开主机端口	私有容器端口	协议
<input type="text" value="6868"/>	<input type="text" value="6868"/>	TCP

### (3) 服务添加成功

应用:  添加服务

描述: 十次方

Active	eureka ①	镜像: 192.168.184.130:5000/tensquare_eureka:0.0.8-SNAPSHOT 端口: 6868	服务	1 容器	⊕ ⋮
--------	----------	---	----	------	-----

### (4) 我们现在访问以下我们的系统

<http://192.168.184.136:6868/> 可以正常访问

## 3.2 配置中心部署

### 创建容器

添加服务config 镜像 192.168.184.135:5000/tensquare\_config:1.0-SNAPSHOT

映射端口: 12000

测试 浏览器输入 <http://192.168.184.135:12000/base-dev.yml> 可以查看到配置文件内容

## 3.3 十次方公共模块安装

配置的maven命令是

```
clean install
```

## 3.4 基础信息微服务部署

(1) 将基础信息微服务上传至Docker私服

(2) 添加服务base-service 镜像tensquare\_base:1.0-SNAPSHOT 端口映射9100

名称	描述
base-service	基础信息微服务

选择镜像\* ☐ 创建前总是拉取镜像

192.168.184.130:5000/tensquare\_base:0.0.1-SNAPSHOT

端口映射

公开主机端口	私有容器端口	协议
9002	9002	TCP

(3) 测试微服务 浏览器打开网址 <http://192.168.184.136:9001/label> 看是否可以看到标签列表

## 3.5 其它微服务部署

学员实现

## 3.6 小节

(1) eureka部署

(2) 配置中心部署

(3) 准备配置文件（对应的环境）上传到码云

(4) 修改本地的微服务代码，连接配置中心

(5) 上传代码到git

(6) 在Jenkins系统中新建任务，完成持续集成，上传到私有仓库

(7) 在Rancher中创建服务，指定私有仓库中的镜像名称，自动下载并创建容器（服务）

软件：

(1) 码云 。作用是存储我们的配置文件。

(2) Gogs。与git配合管理我们的源码

(3) Jenkins。实现持续集成。从Git中获取源码，打包编译上传至Docker私有仓库

(4) Rancher。负责从私有仓库或中央仓库中下载镜像自动创建容器。

## 4 弹性扩容解决方案

### 4.1 什么是弹性扩容

### 4.2 负载均衡器

(1) 在Rancher将创建的base-service（基础信息微服务）删除

(2) 重新创建base-service，不设置端口映射

名称	描述
base-service	基础信息微服务

选择镜像\*

192.168.184.130:5000/tensquare\_base:0.0.1-SNAPSHOT

☐ 创建前总是拉取镜像

[+ 端口映射](#)

(3) 点击“添加服务”按钮右下角，弹出菜单选择“添加负载均衡”



(4) 填写负载均衡器名称ha-base-service 端口9001映射为9001，选择目标为base-service工程，点击创建按钮

名称: ha-base-service 描述: 基础信息微服务负载均衡器

端口规则

访问*	协议*	请求头信息	端口*	路径	目标*	端口*
公开	HTTP	例如: example.com	9002	例如: /foo	tensquare/base-service	9002

主机及路径规则将根据显示的顺序自上而下进行匹配。后端默认将随机命名。要自定义生成的后端，提供一个名称并在自定义的haproxy.cfg文件中引用该名称。显示自定义后端名称选项。显示主机IP选项。

(5) 创建后稍等几秒，负载均衡器启动成功

描述: 十次方

Active	base-db	镜像: centos/mysql-57-centos7 端口: 33062	服务	1 容器	
Active	base-service	镜像: 192.168.184.130:5000/tensquare_base:0.0.1-SNAPSHOT	服务	1 容器	
Active	eureka	镜像: 192.168.184.130:5000/tensquare_eureka:0.0.8-SNAPSHOT 端口: 6868	服务	1 容器	
Active	ha-base-service	到: base-service 端口: 9002/tcp	负载均衡	1 容器	
Active	redis	镜像: redis 端口: 6379	服务	1 容器	

(6) 访问测试 浏览器访问: <http://192.168.184.130:9002/label> 可以看到标签列表

## 4.3 部署WebHooks(钩子)

### 4.3.1 扩容

(1) 在选择菜单API -->WebHooks，点击“添加接收器”按钮

Default 应用 应用商店 基础架构 系统管理 API

在添加第一个服务或容器之前，必须至少添加一台安装了支持的Docker版本的Linux主机。添加主机

Experimental: More webhook features will be added in future releases, and the existing capabilities may change.

Receiver Hooks 添加接收器

Receiver hooks 提供一个URL，在访问该URL时能够触发Rancher内部相应的动作。

状态	名称	类型	详情
无receiver hooks.			

(2) 填写名称等信息，选择要扩容的服务，点击创建按钮



名称\*

scale-base

类型

扩缩容服务

操作

☒ 扩容 ☐ 缩容

目标服务\*

tensquare/base-service

步长

2

最小数量

1

最大数量

20

创建

取消

(3) 接收器列表中新增了一条记录，点击触发地址将地址复制到剪切板

Receiver Hooks					添加接收器
Receiver hooks 提供一个URL，在访问该URL时能够触发Rancher内部相应的动作。					
状态	名称	类型	详情	触发地址	
Active	scale-base	扩缩容服务	扩缩容动作 up tensquare/base-service 以步长 2		

(4) 使用postman测试:

POST

http://192.168.184.130:9090/v1-webhooks/endpoint?key=Hqmo1oZF3lFmSa5q4knUpDQJlfnHfcqUfo0ZPqHh&projectId=1a7

Params

Send

Authorization

Headers

Body

Pre-request Script

Tests

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization help

测试后，发现容器由原来的1个变为了3个

描述: 十次方					
Active	base-db ①	镜像: centos/mysql-57-centos7 端口: 33062	服务·	1 容器	⊕ ⋮
Active	base-service ①	镜像: 192.168.184.130:5000/tensquare_base:0.0.1-SNAPSHOT	服务·	3 容器	⊕ ⋮
Active	eureka ①	镜像: 192.168.184.130:5000/tensquare_eureka:0.0.8-SNAPSHOT 端口: 6868	服务·	1 容器	⊕ ⋮
Active	ha-base-service ①	到: base-service 端口: 9002/tcp	负载均衡	1 容器	⊕ ⋮
Active	redis ①	镜像: redis 端口: 6379	服务·	1 容器	⊕ ⋮

打开erueka，发现服务也有3个

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
TENSQUARE-BASE	n/a (3)	(3)	UP (3) - a1aaa18ebb08:tensquare-base:9002 , ba879bac3a8a:tensquare-base:9002 , b0f0495894ca:tensquare-base:9002

## 4.3.2 缩容

刚才我们实现了扩容，那么如何减少容器数量呢？我们来试试如何缩容

(1) 添加接收器 ,选择缩容，步长为1表示每次递减1个， 点击创建按钮

名称\*

dec-base

类型

扩缩容服务

操作

☐ 扩容 ☒ 缩容

目标服务\*

tensquare/base-service

步长

1

最小数量

1

最大数量

20

创建

取消



## (2) 创建成功后，复制触发地址

Receiver hooks 提供一个URL，在访问该URL时能够触发Rancher内部相应的动作。

状态	名称	类型	详情	触发地址
Active	dec-base	扩容服务	扩容动作 down tensquare/base-service 以步长 1	
Active	scale-base	扩容服务	扩容动作 up tensquare/base-service 以步长 2	

## (3) 使用postman测试

描述: 十次方

Active	base-db	镜像: centos/mysql-57-centos7 端口: 33062	服务	1 容器	
Active	base-service	镜像: 192.168.184.130:5000/tensquare_base:0.0.1-SNAPSHOT	服务	2 容器	
Active	eureka	镜像: 192.168.184.130:5000/tensquare_eureka:0.0.8-SNAPSHOT 端口: 6868	服务	1 容器	
Active	ha-base-service	到: base-service 端口: 9002/tcp	负载均衡	1 容器	
Active	redis	镜像: redis 端口: 6379	服务	1 容器	

# 4.4 Grafana实现弹性扩容

Grafana是一个可视化面板（Dashboard），有着非常漂亮的图表和布局展示，功能齐全的度量仪表盘和图形编辑器。支持Graphite、zabbix、InfluxDB、Prometheus和OpenTSDB作为数据源。

Grafana主要特性：灵活丰富的图形化选项；可以混合多种风格；支持白天和夜间模式；多个数据源。

## 4.4.1 Grafana应用安装（从应用商店）

(1) 选择“应用--全部” 菜单， 点击从应用商店添加按钮




(2) 搜索grafana

应用商店 All

grafana

分类: All

管理



Grafana

Visualization dashboard

查看详情

### (3) 点击查看详情

#### 新应用

名称*	描述
grafana	描述

#### 配置选项

<b>Http Port*</b> 3001 <small>http port to access Grafana</small>	<b>Admin Username*</b> admin <small>Grafana admin username</small>
<b>Admin Password*</b> password <small>Grafana admin password</small>	<b>Secret Key*</b> su2Tong2zoocle <small>Signing secret key</small>
<b>Install Plugins</b> grafana-clock-panel <small>Include/Install Grafana Plugins (comma-separated plugin-ids)</small>	

### (4) 点击启动按钮，等待激活

应用: grafana

添加服务

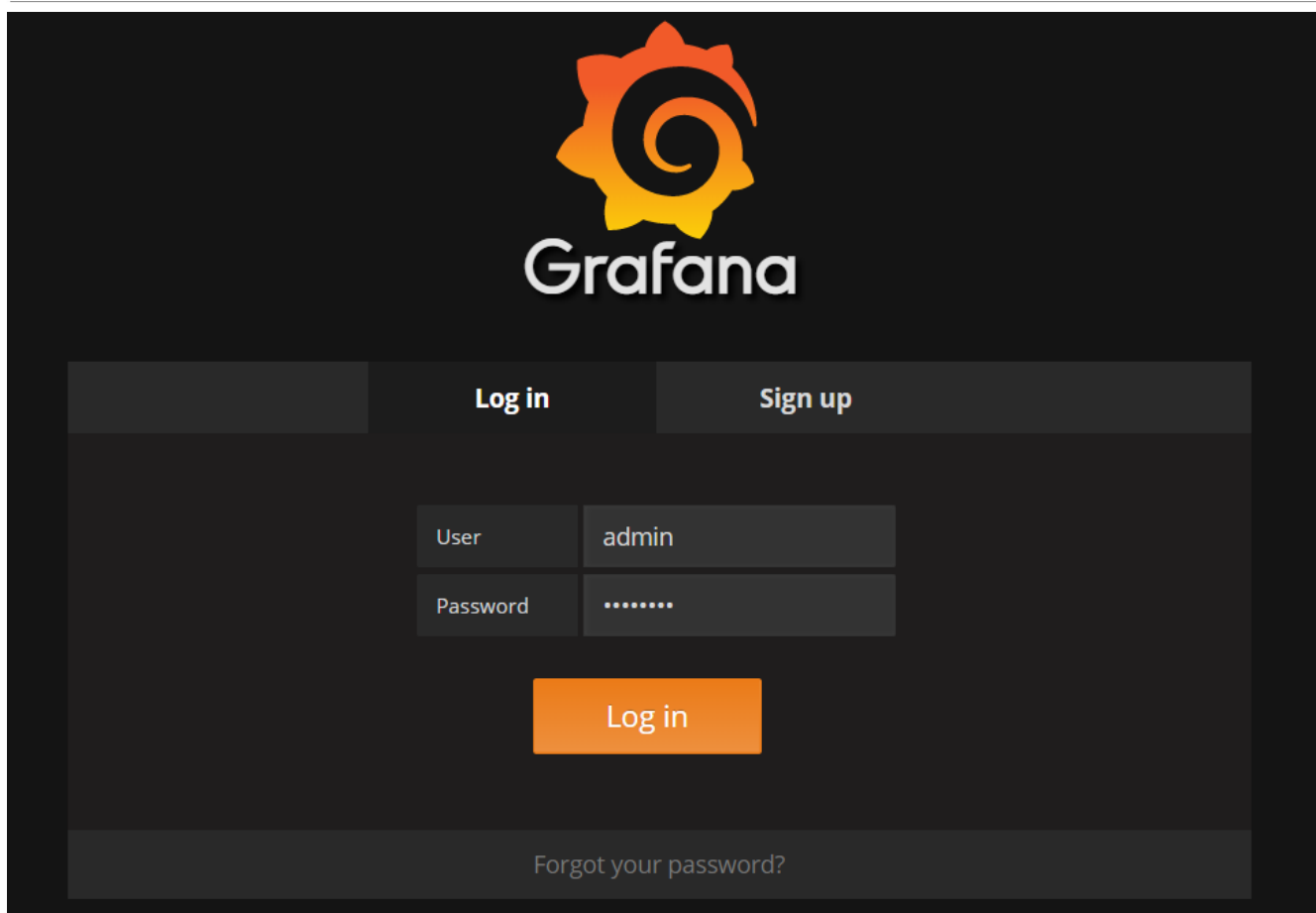
已经是最新版本

Activating grafana (In Progress) ①

镜像: grafana/grafana:4.2.0 端口: 3000

服5

### (5) 激活后浏览器输入 <http://192.168.184.136:3000> 进入登录页



用户名admin 密码password

## 4.4.2 安装（Rpm）

（1）下载及安装

```
wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana-5.1.3-1.x86_64.rpm  
sudo yum localinstall grafana-5.1.3-1.x86_64.rpm
```

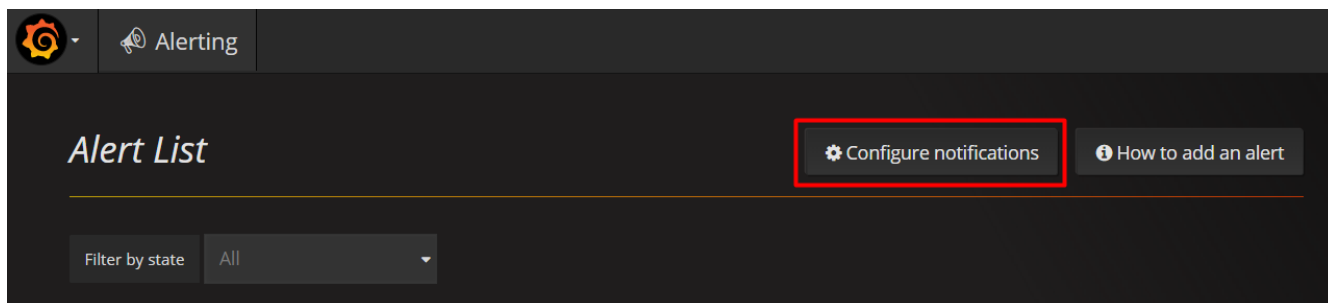
（2）启动服务：

```
systemctl start grafana-server
```

端口是3000

## 4.4.3 Alerting（预警）

（1）选择菜单 alerting-->alert List



(2) 点击Configure notifications按钮，点击new channel，参照下图填写信息

The screenshot shows the 'New Channel' configuration page. It has a title 'New Channel' at the top. Below it is a form with several fields: 'Name' (filled with 'scale-base'), 'Type' (a dropdown menu filled with 'webhook'), 'Send on all alerts' (a checkbox), and 'Include image' (a checkbox). Below these is a section titled 'Webhook settings' containing 'Url' (filled with 'http://192.168.184.130:9090/v1-webhooks/endpoint'), 'Http Method' (a dropdown menu filled with 'POST'), 'Username' (empty), and 'Password' (empty). At the bottom of the form are two buttons: 'Save' (green) and 'Send Test' (blue).

(3) 点击SendTest 测试 观察基础微服务是否增加容器

(4) 点击save保存

(5) 按照同样的方法添加缩容地址

## 4.5 完成其它模块微服务的部署

学员实现 代码略。要求每个模块都有负载均衡器