

持续集成与容器管理

学习目标：

- 掌握DockerMaven插件的使用
- 掌握持续集成工具Jenkins的安装与使用
- 掌握容器管理工具Rancher的安装与使用
- 掌握时间序列数据库influxDB的安装与使用
- 掌握容器监控工具cAdvisor的安装与使用
- 掌握图表工具Grafana的使用

1 DockerMaven插件

微服务部署有两种方法：

（1）手动部署：首先基于源码打包生成jar包（或war包），将jar包（或war包）上传至虚拟机并拷贝至JDK容器。

（2）通过Maven插件自动部署。

对于数量众多的微服务，手动部署无疑是非常麻烦的做法，并且容易出错。所以我们这里学习如何自动部署，这也是企业实际开发中经常使用的方法。

Maven插件自动部署步骤：

（1）修改宿主机的docker配置，让其可以远程访问

```
vi /lib/systemd/system/docker.service
```

其中ExecStart=后添加配置 `-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock`

修改后如下：

```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock
ExecReload=/bin/kill -s HUP $MAINPID
```

(2) 刷新配置，重启服务

```
systemctl daemon-reload
systemctl restart docker
docker start registry
```

(3) 在工程pom.xml 增加配置

```

<build>
  <finalName>app</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    <!-- docker的maven插件，官网：
https://github.com/spotify/docker-maven-plugin -->
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>0.4.13</version>
      <configuration>

<imageName>192.168.184.141:5000/${project.artifactId}:${project.version}
</imageName>
        <baseImage>jdk1.8</baseImage>
        <entryPoint>["java", "-jar",
"/${project.build.finalName}.jar"]</entryPoint>
        <resources>
          <resource>
            <targetPath>/</targetPath>
            <directory>${project.build.directory}
          </resource>
        </resources>
        <include>${project.build.finalName}.jar</include>
      </resource>
    </resources>
    <dockerHost>http://192.168.184.141:2375</dockerHost>
  </configuration>
</plugin>
</plugins>
</build>

```

以上配置会自动生成Dockerfile

```

FROM jdk1.8
ADD app.jar /
ENTRYPOINT ["java", "-jar", "/app.jar"]

```

(5) 在windows的命令提示符下，进入工程tensquare_parent所在的目录

```
mvn install
```

进入tensquare_base 所在的目录，输入以下命令，进行打包和上传镜像

```
mvn docker:build -DpushImage
```

执行后，会有如下输出，代码正在上传

```
c3fe59dd9556: Pushing [=====>] 353.6 MB
c3fe59dd9556: Pushing [=====>] 354.1 MB
c3fe59dd9556: Pushing [=====>] 354.7 MB
c3fe59dd9556: Pushing [=====>] 355.2 MB
c3fe59dd9556: Pushing [=====>] 355.7 MB
c3fe59dd9556: Pushing [=====>] 356.3 MB
c3fe59dd9556: Pushing [=====>] 356.7 MB
c3fe59dd9556: Pushed
```

(6) 进入宿主机，查看镜像

```
docker images
```

REPOSITORY	TAG	IMAGE ID
192.168.184.135:5000/tensquare_base	1.0-SNAPSHOT	83efa6b4478c
10 minutes ago		
192.168.184.135:5000/jdk1.8	latest	507438a0158f
6 hours ago		
jdk1.8	latest	507438a0158f
6 hours ago		

输出如上内容，表示微服务已经做成镜像

浏览器访问 <http://192.168.184.141:5000/v2/catalog>，输出

```
{"repositories":["tensquare_base"]}
```

浏览器访问 <http://192.168.184.141:5000/v2/catalog>，输出

```
{"repositories":["tensquare_base"]}
```

(7) 启动容器:

```
docker run -di --name=base -p 9001:9001  
192.168.184.141:5000/tensquare_base:1.0-SNAPSHOT
```

2 持续集成工具-Jenkins

2.1 什么是持续集成

持续集成 Continuous integration，简称CI

随着软件开发复杂度的不断提高，团队开发成员间如何更好地协同工作以确保软件开发的质量已经慢慢成为开发过程中不可避免的问题。尤其是近些年来，敏捷（Agile）在软件工程领域越来越红火，如何能再不断变化的需求中快速适应和保证软件的质量也显得尤其的重要。

持续集成正是针对这一类问题的一种软件开发实践。它倡导团队开发成员必须经常集成他们的工作，甚至每天都可能发生多次集成。而每次的集成都是通过自动化的构建来验证，包括自动编译、发布和测试，从而尽快地发现集成错误，让团队能够更快的开发内聚的软件。

持续集成具有的特点：

- 它是一个自动化的周期性的集成测试过程，从检出代码、编译构建、运行测试、结果记录、测试统计等都是自动完成的，无需人工干预；
- 需要有专门的集成服务器来执行集成构建；
- 需要有代码托管工具支持，我们下一小节将介绍Git以及可视化界面Gogs的使用

持续集成的作用：

- 保证团队开发人员提交代码的质量，减轻了软件发布时的压力；
- 持续集成中的任何一个环节都是自动完成的，无需太多的人工干预，有利于减少重复过程以节省时间、费用和工作量；

2.2 Jenkins简介

Jenkins，原名Hudson，2011年改为现在的名字，它是一个开源的实现持续集成的软件工具。官方网站：<http://jenkins-ci.org/>。

Jenkins 能实施监控集成中存在的错误，提供详细的日志文件和提醒功能，还能用图表的形式形象地展示项目构建的趋势和稳定性。

特点：

- 易安装：仅仅一个 `java -jar jenkins.war`，从官网下载该文件后，直接运行，无需额外的安装，更无需安装数据库；
- 易配置：提供友好的GUI配置界面；
- 变更支持：Jenkins能从代码仓库（Subversion/CVS）中获取并产生代码更新列表并输出到编译输出信息中；
- 支持永久链接：用户是通过web来访问Jenkins的，而这些web页面的链接地址都是永久链接地址，因此，你可以在各种文档中直接使用该链接；
- 集成E-Mail/RSS/IM：当完成一次集成时，可通过这些工具实时告诉你集成结果（据我所知，构建一次集成需要花费一定时间，有了这个功能，你就可以在等待结果过程中，干别的事情）；
- JUnit/TestNG测试报告：也就是用以图表等形式提供详细的测试报表功能；
- 支持分布式构建：Jenkins可以把集成构建等工作分发到多台计算机中完成；
- 文件指纹信息：Jenkins会保存哪次集成构建产生了哪些jars文件，哪一次集成构建使用了哪个版本的jars文件等构建记录；
- 支持第三方插件：使得 Jenkins 变得越来越强大

2.3 Jenkins安装

2.3.1 JDK安装

（1）将jdk-8u171-linux-x64.rpm上传至服务器（虚拟机）

（2）执行安装命令

```
rpm -ivh jdk-8u171-linux-x64.rpm
```

RPM方式安装JDK，其根目录为：`/usr/java/jdk1.8.0_171-amd64`

2.3.2 Jenkins安装与启动

(1) 下载jenkins

```
wget https://pkg.jenkins.io/redhat/jenkins-2.83-1.1.noarch.rpm
```

或将jenkins-2.83-1.1.noarch.rpm上传至服务器

(2) 安装jenkins

```
rpm -ivh jenkins-2.83-1.1.noarch.rpm
```

(3) 配置jenkins

```
vi /etc/sysconfig/jenkins
```

修改用户和端口

```
JENKINS_USER="root"  
JENKINS_PORT="8888"
```

(4) 启动服务

```
systemctl start jenkins
```

(5) 访问链接 <http://192.168.184.135:8888>

从/var/lib/jenkins/secrets/initialAdminPassword中获取初始密码串

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

(6) 安装插件



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Getting Started

✓ Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	Folders ** bouncycastle API ** Structs
Timestampers	Workspace Cleanup	Ant	Gradle	
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View	
Git	Subversion	SSH Slaves	Matrix Authorization Strategy	
PAM Authentication	LDAP	Email Extension	Mailer	
				** - required dependency

(7) 新建用户

Create First Admin User

用户名:

密码:

确认密码:

全名:

电子邮件地址:

完成安装进入主界面



2.4 Jenkins插件安装

我们以安装maven插件为例，演示插件的安装

(1) 点击左侧的“系统管理”菜单,然后点击



[系统设置](#)
 全局设置&路径



[Configure Global Security](#)
 Secure Jenkins; define who is allowed to access/use the system.



[全局工具配置](#)
 工具配置，包括它们的位置和自动安装器



[读取设置](#)
 放弃当前内存中所有的设置信息并从配置文件中重新读取 仅用于当您手动修改配置文件时重新读取设置。



[管理插件](#)
 添加、删除、禁用或启用Jenkins功能扩展插件。



[系统信息](#)
 显示系统环境信息以帮助解决问题。



[系统日志](#)
 系统日志从java.util.logging捕获Jenkins相关的日志信息。



[负载统计](#)
 检查您的资源利用情况，看看是否需要更多的计算机来帮助您构建。



[Jenkins CLI](#)
 从您命令行或脚本访问或管理您的Jenkins。



[脚本命令行](#)
 执行用于管理或故障探测或诊断的任意脚本命令。



[管理节点](#)
 添加、删除、控制和监视系统运行任务的节点。







(2) 选择“可选插件”选项卡，搜索maven，在列表中选择Maven Integration ，点击“直接安装”按钮

过滤:

可更新
 可选插件
 已安装
 高级

安装 ↓	名称	版本
<input type="checkbox"/>	Maven Artifact ChoiceListProvider (Nexus) A ChoiceListProvider to read Maven artifact information from a Nexus repository or MavenCentral	1.2.4
<input type="checkbox"/>	Maven Metadata Plugin for Jenkins CI server Adds a new build parameter type for resolving artifact versions reading the repository maven-metadata.xml	2.0.0
<input type="checkbox"/>	Maven Integration This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (JUnit, ...).	3.1.2
<input type="checkbox"/>	Maven Dependency Update Trigger	1.5
<input type="checkbox"/>	Unleash Maven Provides Maven release functionality using the Unleash Maven Plugin.	2.3.0
<input type="checkbox"/>	Maven Release Plug-in A plug-in that enables you to perform releases using the maven-release-plugin from Jenkins.	0.14.0
<input type="checkbox"/>	Config File Provider Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace	2.18
<input type="checkbox"/>	Jira Issue Updater Jenkins plugin which can update Jira issues in a maven or a freestyle job.	1.18

看到如下图时，表示已经完成

Credentials	 完成
SSH Credentials	 完成
JSch dependency	 完成
Maven Integration	 失败 - 详细
Apache HttpComponents Client 4.x API	 完成
Maven Integration	 完成

2.5 全局工具配置

2.5.1 安装Maven与本地仓库

(1) 将Maven压缩包上传至服务器（虚拟机）

(2) 解压

```
tar zxvf apache-maven-3.5.4-bin.tar.gz
```

(3) 移动目录

```
mv apache-maven-3.5.4 /usr/local/maven
```

(4) 编辑setting.xml配置文件 `vi /usr/local/maven/conf/settings.xml`，配置本地仓库目录,内容如下

```
<localRepository>/usr/local/repository</localRepository>
```

(5) 将开发环境的本地仓库上传至服务器（虚拟机）并移动到/usr/local/repository 。

```
mv reponsitory_boot /usr/local/repository
```

执行此步是为了以后在打包的时候不必重新下载，缩短打包的时间。

2.5.2 全局工具配置

选择系统管理，全局工具配置

（1）JDK配置

JDK

JDK 安装

JDK

别名

JDK

JAVA_HOME

/usr/java/jdk1.8.0_171-amd64

☐ 自动安装

新增 JDK

删除 JDK

系统下JDK 安装列表

设置javahome为 /usr/java/jdk1.8.0_171-amd64

（2）Git配置 （本地已经安装了Git软件）

Git

Git installations

Git

Name

Default

Path to Git executable

git

☐ 自动安装

Add Git

Delete Git

（3）Maven配置

Maven

Maven 安装

Maven

Name

Maven

MAVEN_HOME

/usr/local/maven

☐ 自动安装

删除 Maven

Save

Apply

2.6 代码上传至Git服务器

2.6.1 Gogs搭建与配置

Gogs 是一款极易搭建的自助 Git 服务。

Gogs 的目标是打造一个最简单、最快速和最轻松的方式搭建自助 Git 服务。使用 Go 语言开发使得 Gogs 能够通过独立的二进制分发，并且支持 Go 语言支持的所有平台，包括 Linux、Mac OS X、Windows 以及 ARM 平台。

地址: <https://gitee.com/Unknown/gogs>

(1) 下载镜像

```
docker pull gogs/gogs
```

(2) 创建容器

```
docker run -di --name=gogs -p 10022:22 -p 3000:3000 -v /var/gogsdata:/data  
gogs/gogs
```

(3) 假设我的centos虚拟机IP为192.168.184.141 在地址栏输入<http://192.168.184.141:3000> 会进入首次运行安装程序页面, 我们可以选择一种数据库作为gogs数据的存储, 最简单的是选择SQLite3。如果对于规模较大的公司, 可以选择MySQL

首次运行安装程序

如果您正在使用 Docker 容器运行 Gogs，请务必先仔细阅读 [官方文档](#) 后再对本页面进行填写。

数据库设置

Gogs 要求安装 MySQL、PostgreSQL、SQLite3、MSSQL 或 TiDB。

数据库类型 *

SQLite3

数据库文件路径 *

data/gogs.db

SQLite3 数据库文件路径。
作为服务启动时，请使用绝对路径。

应用基本设置

应用名称 *

Gogs

快用狂拽酷炫的组织名称闪瞎我们！

仓库根目录 *

/root/gogs-repositories

所有 Git 远程仓库都将被存放于该目录。

运行系统用户 *

git

该用户必须具有对仓库根目录和运行 Gogs 的操作权限。

域名 *

192.168.184.135

该设置影响 SSH 克隆地址。

☐ 使用内置 SSH 服务器

HTTP 端口号 *

3000

应用监听的端口号

应用 URL *

http://192.168.184.135:3000/

该设置影响 HTTP/HTTPS 克隆地址和一些邮箱中的链接。

日志路径 *

/root/gogs/log

存放日志文件的目录

☐ 启用控制台模式

可选设置

▶ 邮件服务设置

▶ 服务器和其它服务设置

▶ 管理员帐号设置

立即安装

点击“立即安装”

这里的域名要设置为centos的IP地址,安装后显示主界面

[首页](#)[发现](#)[帮助](#)[注册](#)[登录](#)

Gogs

一款极易搭建的自助 Git 服务



易安装

您除了可以根据操作系统平台通过 [二进制运行](#)，还可以通过 [Docker](#) 或 [Vagrant](#)，以及 [包管理](#) 安装。



跨平台

任何 [Go 语言](#) 支持的平台都可以运行 Gogs，包括 Windows、Mac、Linux 以及 ARM。挑一个您喜欢的就行！



轻量级



开源化

(4) 注册

注册

用户名*

邮箱*

密码*

确认密码*



验证码*

创建帐户

[已经注册? 立即登录!](#)

(5) 登录

登录

用户名或邮箱*

liubei

密码*

.....

☐ 记住登录

登录

[忘记密码?](#)

[还没帐户? 马上注册。](#)

北京市昌平区建材城西路金燕龙办公楼一层 电话：400-618-9090

(6) 创建仓库

创建新的仓库

所有者 *

 liubei

仓库名称 *

tensquare

伟大的仓库名称一般都较短、令人深刻并且 **独一无二** 的。

可见性

☐ 该仓库为 私有的

仓库描述

.gitignore

选择 .gitignore 模板

授权许可

请选择授权许可文件

自述文档 ?


Default

☐ 使用选定的文件和模板初始化仓库

创建仓库

取消

 liubei / tensquare

 取消关注

1


 点赞

0

 文件

 工单管理 0

 Wiki

 仓库设置

快速帮助

克隆当前仓库 不知道如何操作? 访问 [此处](#) 查看帮助!

HTTP

SSH

http://192.168.184.135:3000/liubei/tensquare.git



从命令行创建一个新的仓库

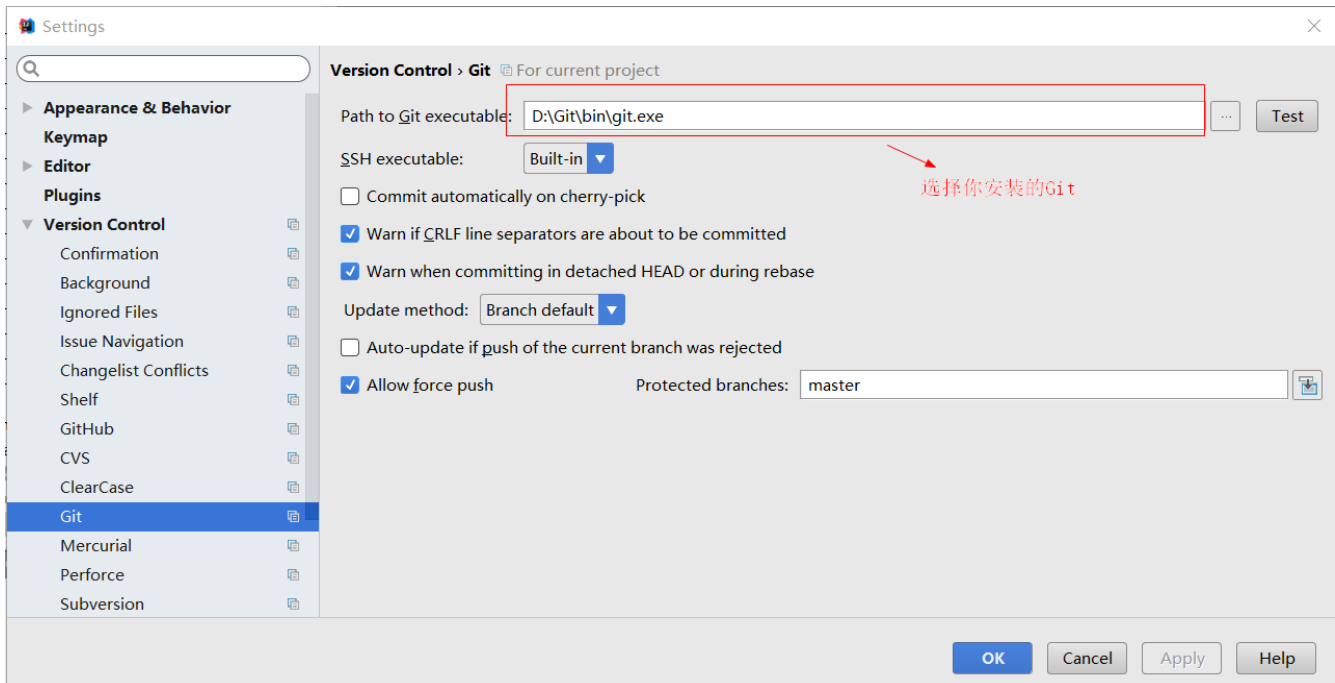
```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin http://192.168.184.135:3000/liubei/tensquare.git
git push -u origin master
```

2.6.2 提交代码

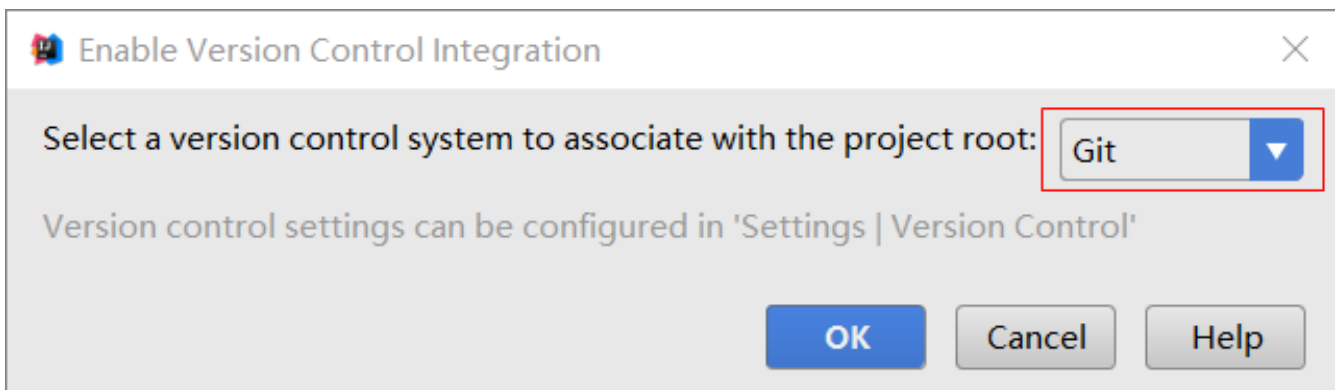
步骤：

(1) 在本地安装git(Windows版本)

(2) 在IDEA中选择菜单：File -- settings，在窗口中选择Version Control -- Git

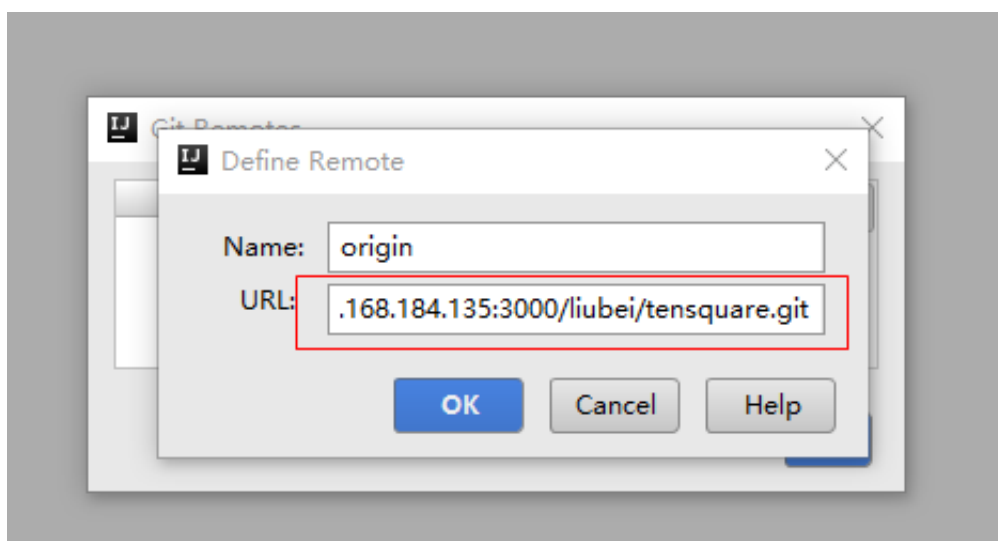
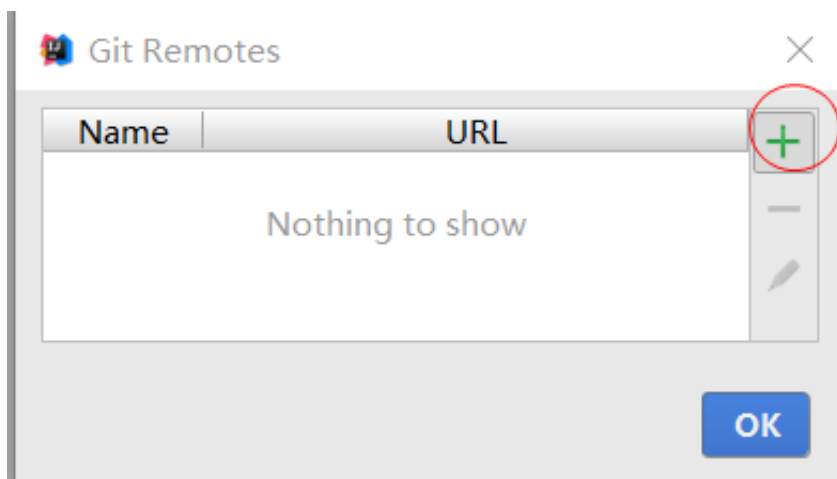


(3) 选择菜单VCS --> Enable Version Control Integration...



选择Git

(4) 设置远程地址：右键点击工程选择菜单 Git --> Repository -->Remotes...



- (5) 右键点击工程选择菜单 Git --> Add
- (6) 右键点击工程选择菜单 Git --> Commit Directory...
- (7) 右键点击工程选择菜单 Git --> Repository --> Push ...

2.7 任务的创建与执行

- (1) 回到首页，点击新建按钮 .如下图，输入名称，选择创建一个Maven项目，点击OK

Enter an item name

» Required field



构建一个自由风格的软件项目

这是Jenkins的主要功能。Jenkins将会结合任何SCM和任何构建系统来构建你的项目，甚至可以构建软件以外的系统。



构建一个maven项目

构建一个maven项目。Jenkins利用你的POM文件，这样可以大大减轻构建配置。



流水线

精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。



构建一个多配置项目

适用于多配置项目，例如多环境测试，平台指定构建，等等。



文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的内容，只要它们在不同的文件夹里即可。

（2）源码管理，选择Git

源码管理

☐ None

☒ Git

Repositories

Repository URL

Credentials

Add

高级...

Add Repository

Branches to build

源码库浏览器

Additional Behaviours

Add

（3）Build

Build

Root POM

Goals and options

高级...

命令：

```
clean package docker:build -DpushImage
```

用于清除、打包，构建docker镜像

最后点击“保存”按钮

(4) 执行任务

3 容器管理工具Rancher

3.1 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/>

3.2 Rancher安装

(1) 下载Rancher 镜像

```
docker pull rancher/server
```

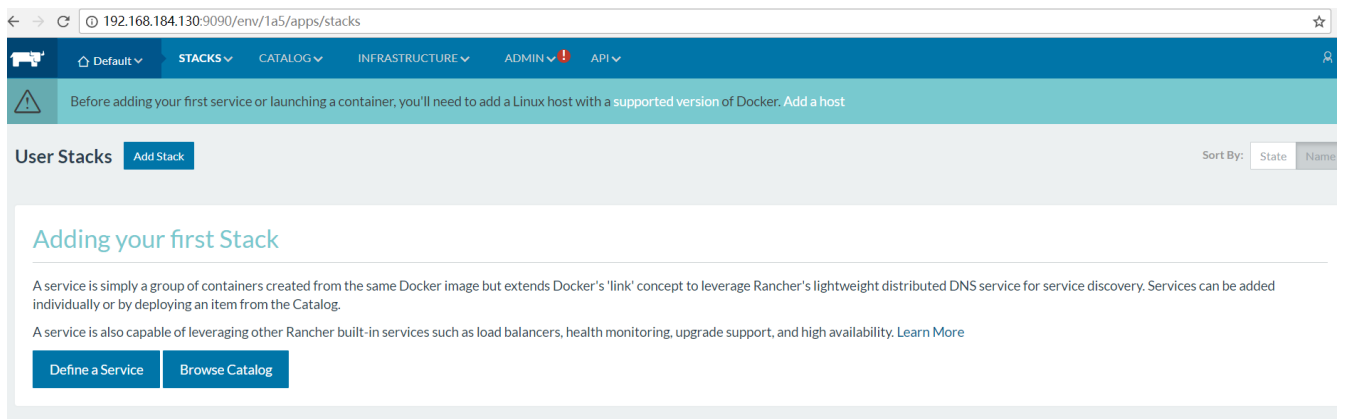
(2) 创建Rancher容器

```
docker run -di --name=rancher -p 9090:8080 rancher/server
```

(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

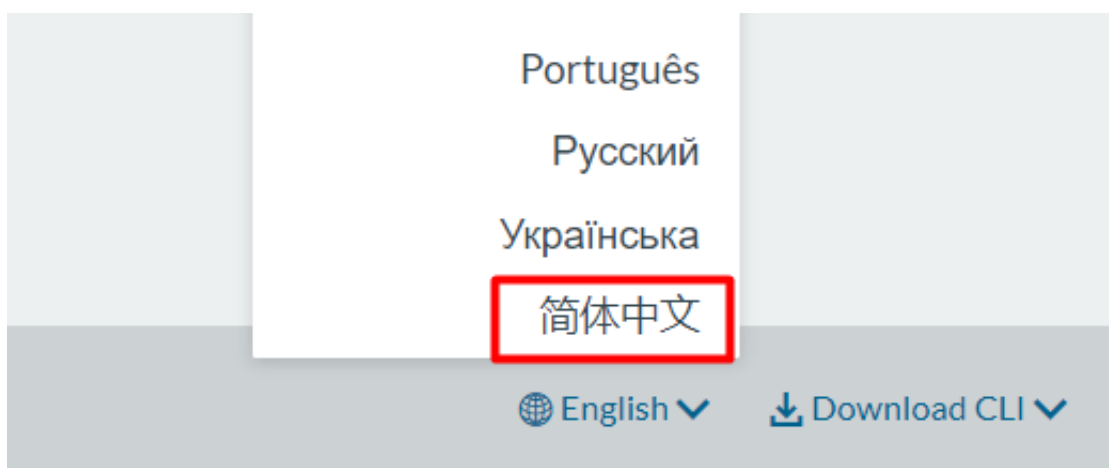


点击Got It 进入主界面



(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~



3.3 Rancher初始化

3.3.1 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



（2）填写名称，点击“创建”按钮

添加环境

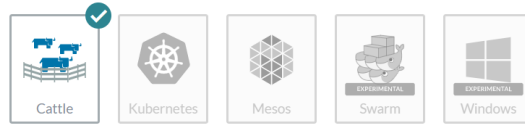
名称

tensquare_dev

描述

十次方开发环境

环境模板



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

3.3.2 添加主机

(1) 选择基础架构-->主机 菜单，点击添加主机

 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

主机 添加主机

当前没有主机或容器

(2) 拷贝脚本

1 启动一台Linux主机并在主机上安装好我们支持的版本的Docker。

2 确认安全组或防火墙允许以下通讯：
◦ 与其他所有主机之间的 UDP 端口 500 和 4500 (用于IPsec网络)

3 可选项: 在主机上增加标签

⊕ 添加标签

4 指定用于注册这台主机的公网IP。如果留空, Rancher会自动检测IP注册。通常在主机有唯一公网IP的情况下这是可以的。如果主机位于防火墙/NAT设备之后, 或者主机同时也是运行 `rancher/server` 容器的主机时, 则必须设置此IP。

例如: 1.2.3.4

5 将下列脚本拷贝到每一台主机上运行以注册 Rancher:

```
sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.10 http://192.168.184.130:9090/v1/scripts/92CAACB4B42BDB7A699A:1514678400000:wp9mbkDN4zyhivUiTurhetHS3Y
```



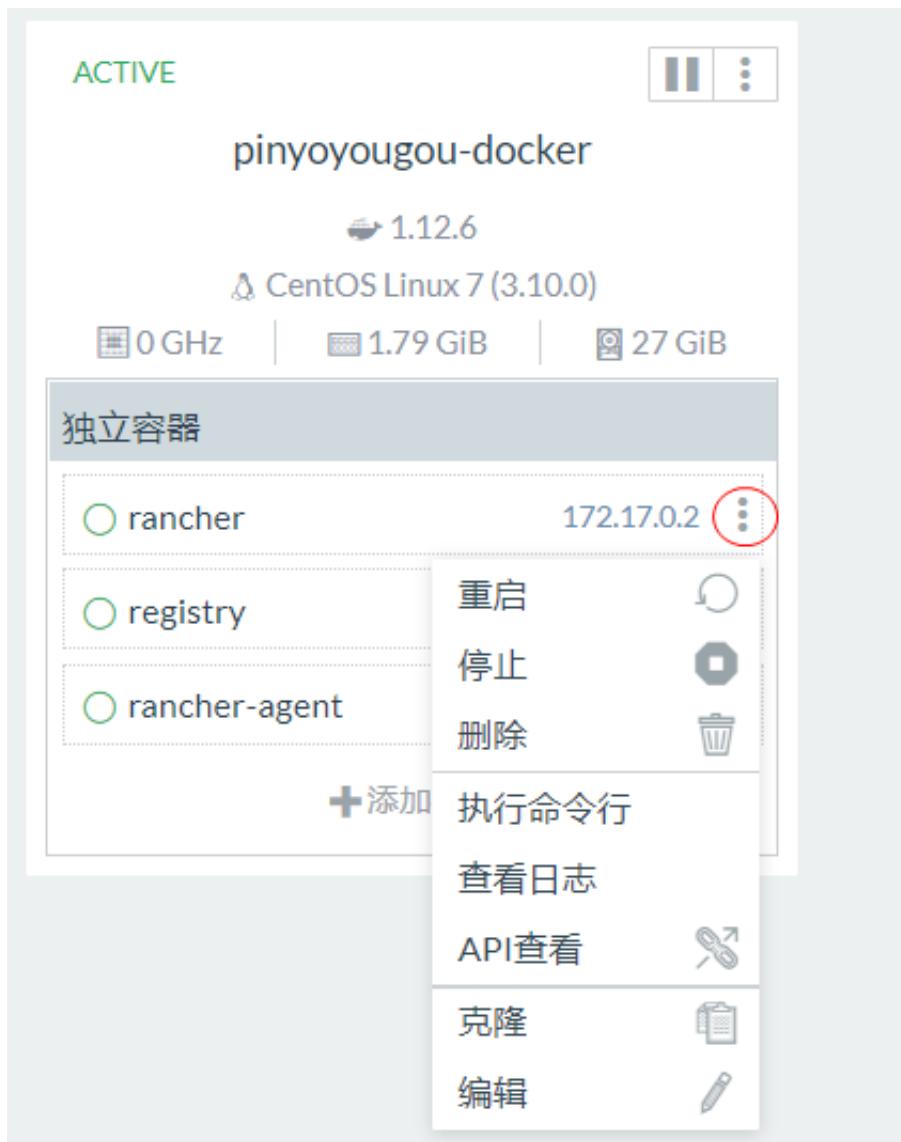
6 点击下面的关闭按钮, 新的主机注册后会显示在 主机 页面。

关闭

(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:/var/lib/rancher rancher/agent:v1.2.10 http://192.168.184.130:9090/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: sha256:5618f36c6b6c6fe25baa3e72977f3a226d43ffd1d74dda6a8860e32b73a1e171
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



3.3.3 添加应用

点击应用-->全部(或用户) ， 点击“添加应用”按钮

全部应用				排序: 状态 名称		
				添加应用	从应用商店添加	
+	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		

填写名称和描述

添加应用

名称

tensquare

描述

十次方

可选:导入COMPOSE

可选:docker-compose.yml

docker-compose.yml文件的内容

上传

可选:rancher-compose.yml

rancher-compose.yml文件的内容

上传

高级选项 ^

创建

取消

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	▶ ⋮

3.4 应用部署

3.4.1 MySQL部署

镜像：centos/mysql-57-centos7 增加数据库服务

名称

mysql

描述

mysql

选择镜像*

centos/mysql-57-centos7

☐ 创建前总是拉取镜像

端口映射

公开主机端口

3306

>

私有容器端口

3306

/

协议

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

MYSQL_ROOT_PASSWORD

=

值

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 ⓘ	镜像: centos/mysql-57-centos7 端口: 3306	服务·	1 容器	ⓘ ⋮
--------	---------	--------------------------------------	-----	------	-----

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

3.4.2 RabbitMQ部署

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

名称

rabbitmq

描述

rabbitmq

选择镜像*

rabbitmq:management

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

5671	>	5671	/	TCP	[-]
5672	>	5672	/	TCP	[-]
4369	>	4369	/	TCP	[-]
15671	>	15671	/	TCP	[-]
15672	>	15672	/	TCP	[-]
25672	>	25672	/	TCP	[-]

显示主机IP选项

浏览器访问 <http://192.168.184.136:15672/>

3.4.3 Redis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

redis

描述

redis

选择镜像*

redis

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

6379	>	6379	/	TCP	[-]
------	---	------	---	-----	-----

显示主机IP选项

创建后使用客户端测试链接

```
redis-cli -h 192.168.184.144
```

测试成功

3.4.4 微服务部署

(1) 搭建私有仓库

启动私有仓库容器

```
docker run -di --name=registry -p 5000:5000 registry
```

打开浏览器 输入地址http://192.168.184.144:5000/v2/_catalog看到 `{"repositories": []}` 表示私有仓库搭建成功并且内容为空

修改daemon.json

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

添加以下内容，保存退出。

```
{"insecure-registries":["192.168.184.144:5000"]}
```

(2) 修改docker配置，允许远程访问

```
vi /lib/systemd/system/docker.service
```

其中ExecStart=后添加配置 `-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock`

修改后刷新配置，冲洗服务

```
systemctl daemon-reload
systemctl restart docker
docker start registry
```

(3) 修改微服务工程，添加DockerMaven插件

(4) 连接mysql数据库，执行建库脚本

(5) 添加服务base-service 镜像192.168.184.144:5000/tensquare_base:1.0-SNAPSHOT 端口映射9001

名称	base-service	描述	基础信息微服务
选择镜像*	192.168.184.130:5000/tensquare_base:0.0.1-SNAPSHOT		
<input type="checkbox"/> 创建前总是拉取镜像			
+ 端口映射			
公开主机端口	9002	私有容器端口	9002 / TCP

(6) 测试微服务 浏览器打开网址 <http://192.168.184.144:9001/label> 看是否可以看到标签列表

3.6 扩容与缩容

3.6.1 扩容

(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	base-service	描述	基础信息微服务
选择镜像*	192.168.184.130:5000/tensquare_base:0.0.1-SNAPSHOT		
<input type="checkbox"/> 创建前总是拉取镜像			
+ 端口映射			

(3) 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.

(4) 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

(5) 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	<div></div>

(6) 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	⊕ ⋮

3.6.2 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	扩缩容服务	扩缩容动作 down tensquare/base-service 以步长 1	
Active	scale-base	扩缩容服务	扩缩容动作 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 influxDB

4.1 什么是influxDB

influxDB是一个分布式时间序列数据库。cAdvisor仅仅显示实时信息，但是不存储监视数据。因此，我们需要提供时序数据库用于存储cAdvisor组件所提供的监控信息，以便显示除实时信息之外的时序数据。

4.2 influxDB安装

(1) 下载镜像

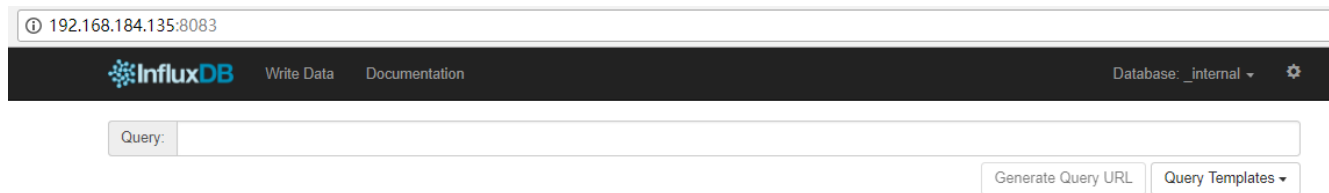
```
docker pull tutum/influxdb
```

(2) 创建容器

```
docker run -di \  
  -p 8083:8083 \  
  -p 8086:8086 \  
  --expose 8090 \  
  --expose 8099 \  
  --name influxsrv \  
  tutum/influxdb
```

端口概述: 8083端口:web访问端口 8086:数据写入端口

打开浏览器 <http://192.168.184.144:8083/>



4.3 influxDB常用操作

4.3.1 创建数据库

```
CREATE DATABASE "cadvisor"
```

回车创建数据库

```
SHOW DATABASES
```

查看数据库

4.3.2 创建用户并授权

创建用户

```
CREATE USER "cadvisor" WITH PASSWORD 'cadvisor' WITH ALL PRIVILEGES
```

查看用户

```
SHOW USRES
```

用户授权

```
grant all privileges on cadvisor to cadvisor
grant WRITE on cadvisor to cadvisor
grant READ on cadvisor to cadvisor
```

4.3.3 查看采集的数据

切换到cadvisor数据库，使用以下命令查看采集的数据

现在我们还没有数据，如果想采集系统的数据，我们需要使用**Cadvisor**软件来实现

5 cAdvisor

5.1 什么是cAdvisor

Google开源的用于监控基础设施应用的工具，它是一个强大的监控工具，不需要任何配置就可以通过运行在Docker主机上的容器来监控Docker容器，而且可以监控Docker主机。更多详细操作和配置选项可以查看Github上的cAdvisor项目文档。

5.2 cAdvisor安装

(1) 下载镜像

```
docker pull google/cadvisor
```

(2) 创建容器

```
docker run --volume=:/rootfs:ro --volume=/var/run:/var/run:rw --  
volume=/sys:/sys:ro --volume=/var/lib/docker:/var/lib/docker:ro --  
publish=8080:8080 --detach=true --link influxsrv:influxsrv --name=cadvisor  
google/cadvisor -storage_driver=influxdb -storage_driver_db=cadvisor -  
storage_driver_host=influxsrv:8086
```

WEB前端访问地址

<http://192.168.184.144:8080/containers/>

性能指标含义参照如下地址

https://blog.csdn.net/ZHANG_H_A/article/details/53097084

再次查看influxDB，发现已经有很多数据被采集进去了。

6 Grafana

6.1 什么是Grafana

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。

6.2 Grafana安装

（1）下载镜像

```
docker pull grafana/grafana
```

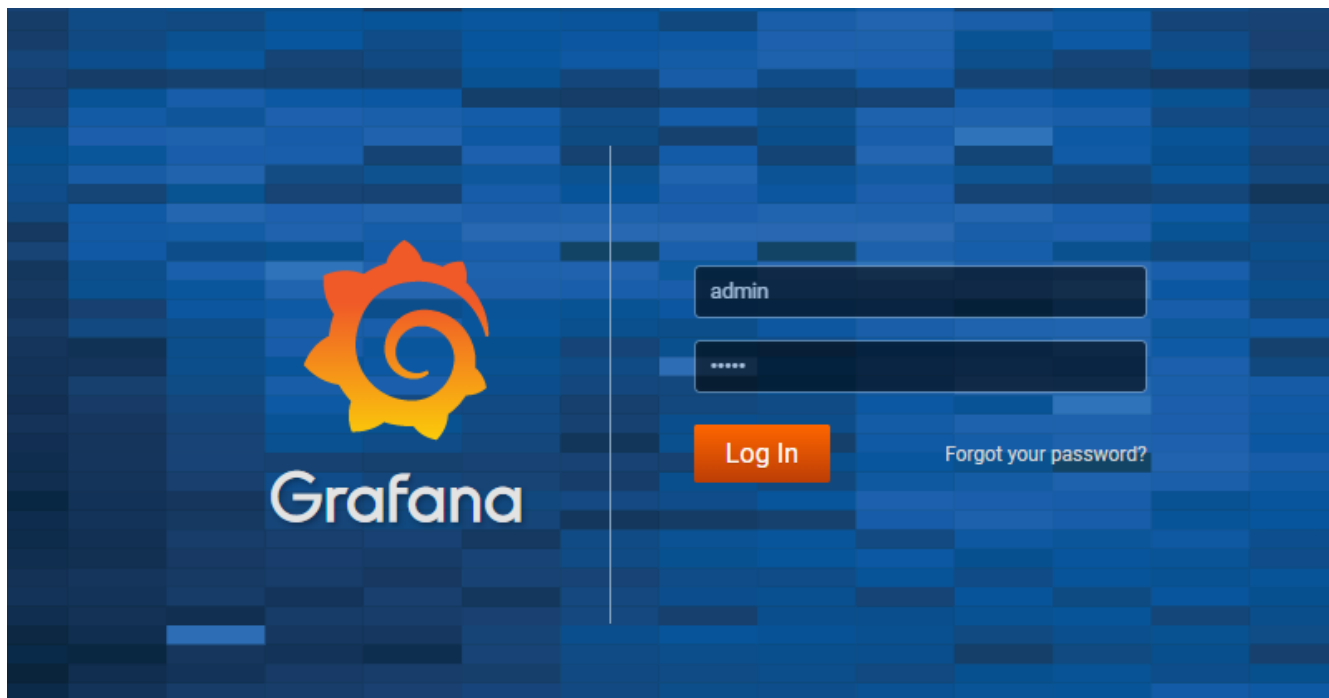
（2）创建容器

```
docker run -d -p 3001:3000 -e INFLUXDB_HOST=influxsrv -e  
INFLUXDB_PORT=8086 -e INFLUXDB_NAME=cadvisor -e INFLUXDB_USER=cadvisor -e  
INFLUXDB_PASS=cadvisor --link influxsrv:influxsrv --name grafana  
grafana/grafana
```

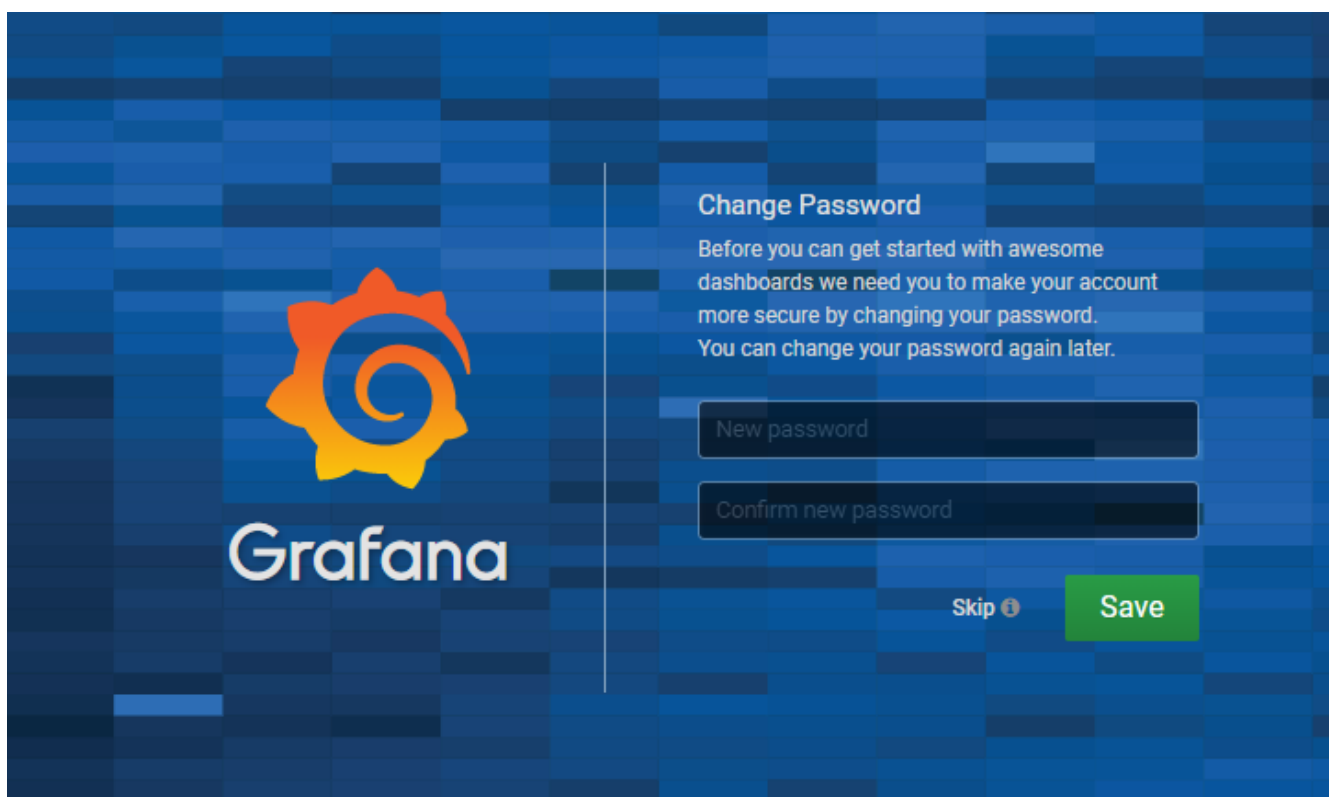
（3）访问

```
http://192.168.184.144:3001
```

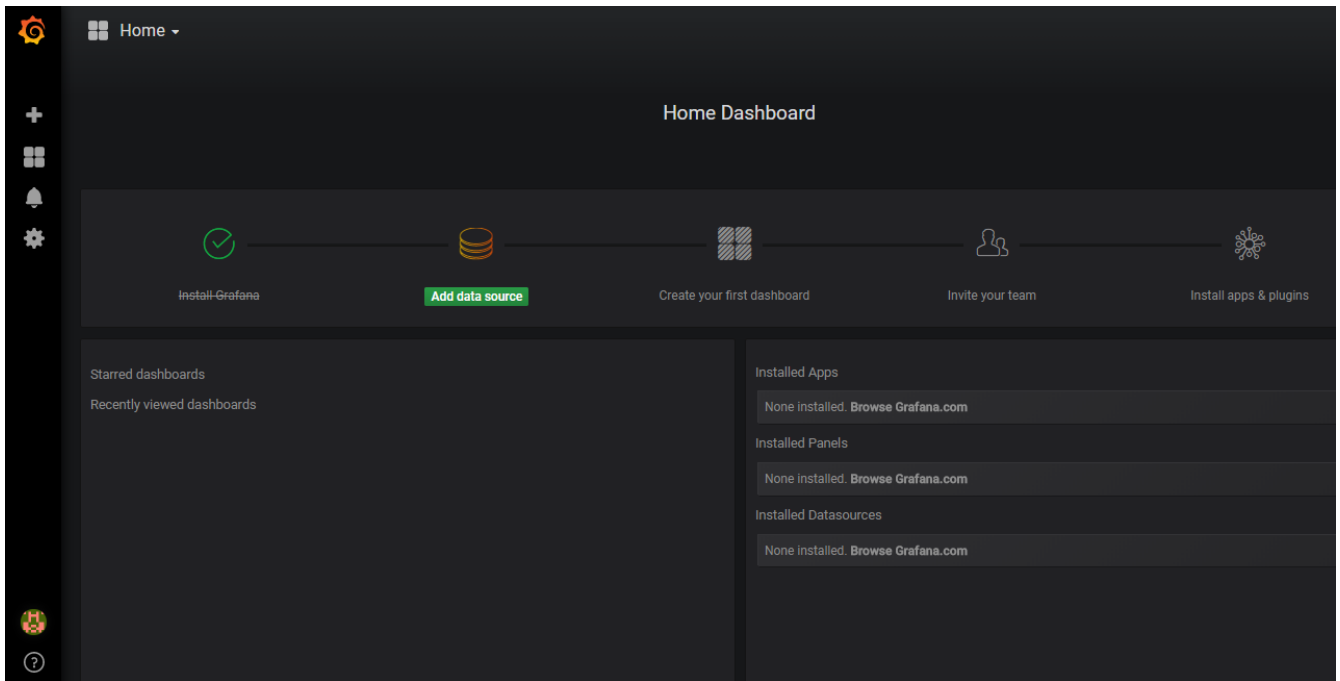
用户名密码均为admin



(4) 登录后提示你修改密码



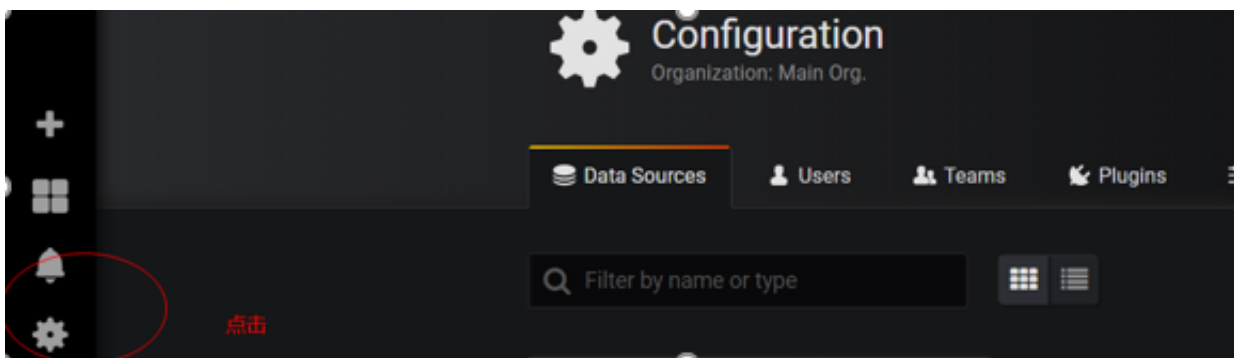
(5) 之后进入主页面



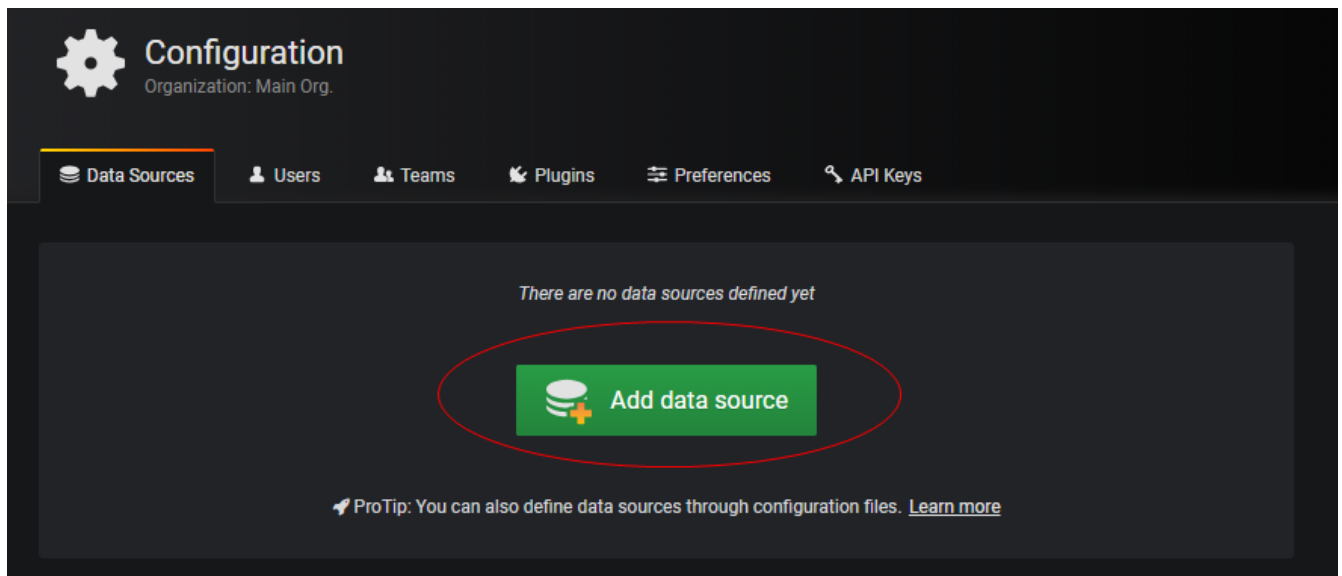
6.3 Grafana的使用

6.3.1 添加数据源

(1) 点击设置，DataSource



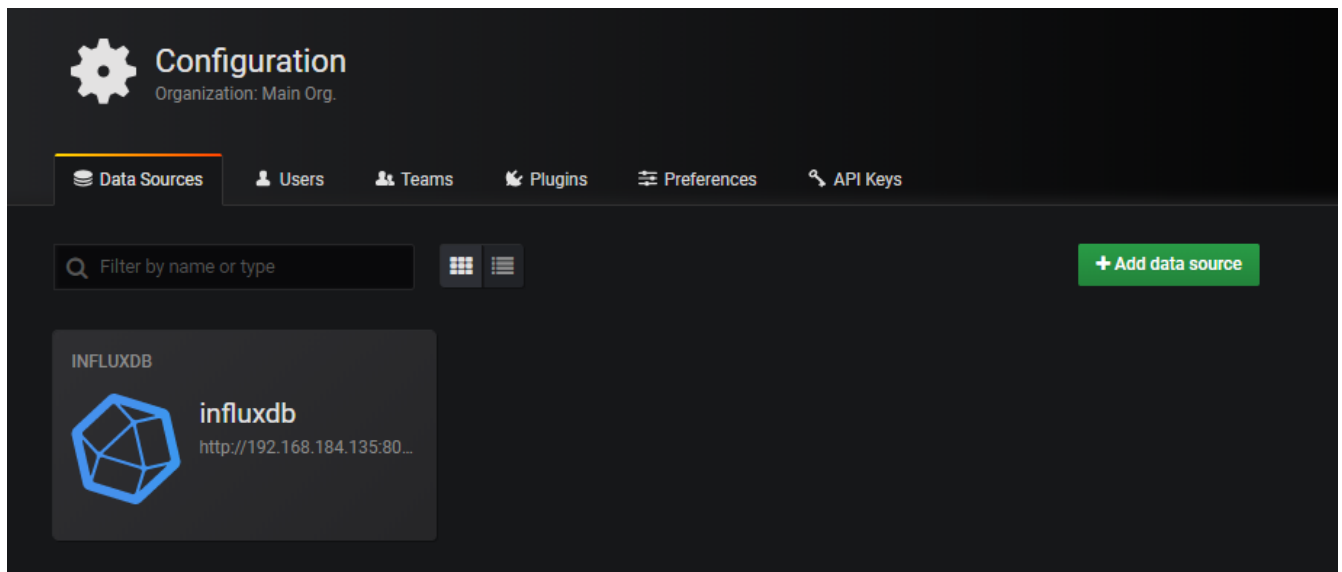
(2) 点击添加data source



(3) 为数据源起个名称，指定类型、地址、以及连接的数据库名、用户名和密码

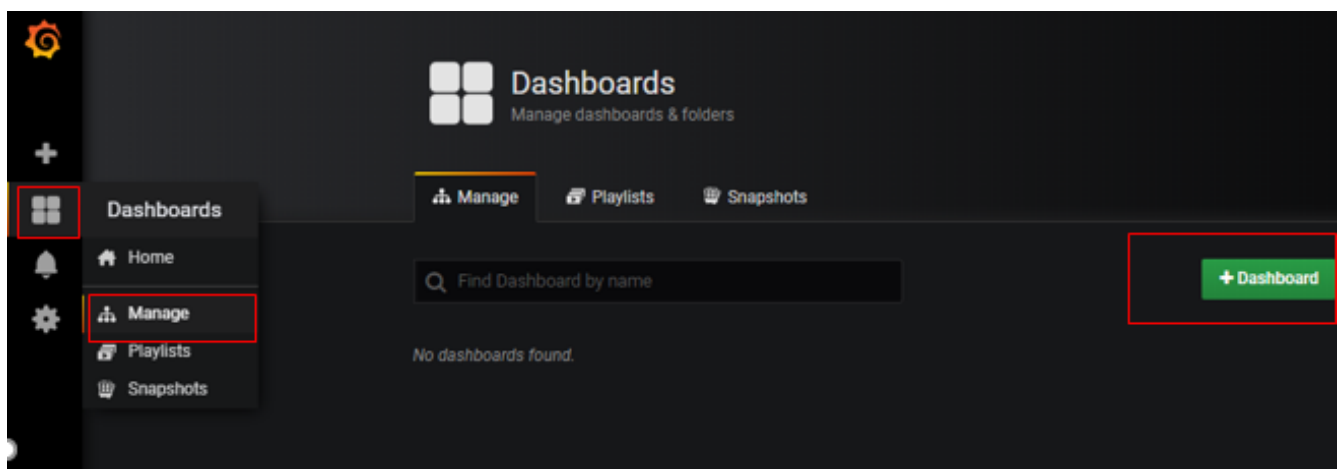
Name	influxdb	i	Default	<input type="checkbox"/>
Type	InfluxDB ▼			
HTTP				
URL	http://192.168.184.135:8086		i	
Access	Server (Default) ▼		Help ▶	
Auth				
Basic Auth	<input type="checkbox"/>	With Credentials i	<input type="checkbox"/>	
TLS Client Auth	<input type="checkbox"/>	With CA Cert i	<input type="checkbox"/>	
Skip TLS Verification (Insecure)		<input type="checkbox"/>		
Advanced HTTP Settings				
Whitelisted Cookies	<input type="text" value="Add Name"/> i			
InfluxDB Details				
Database	cadvisor			
User	cadvisor	Password	

点击保存。数据源建立成功



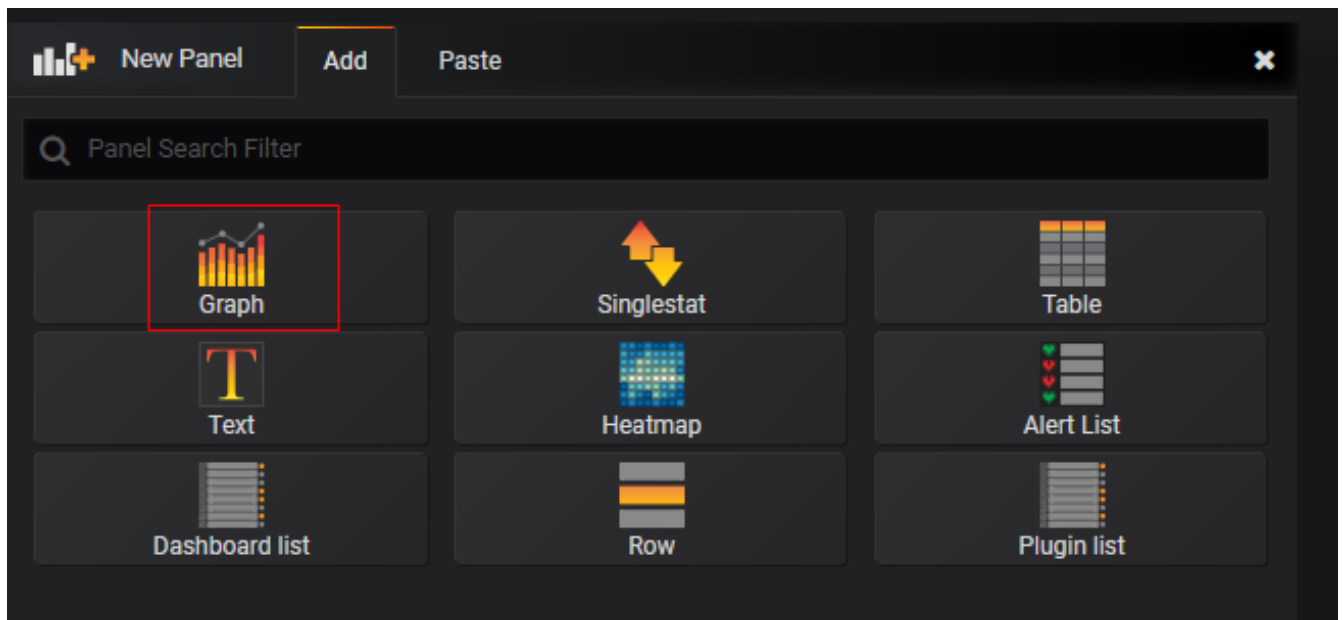
6.3.2 添加仪表盘

(1) 选择Dashboards --Manager

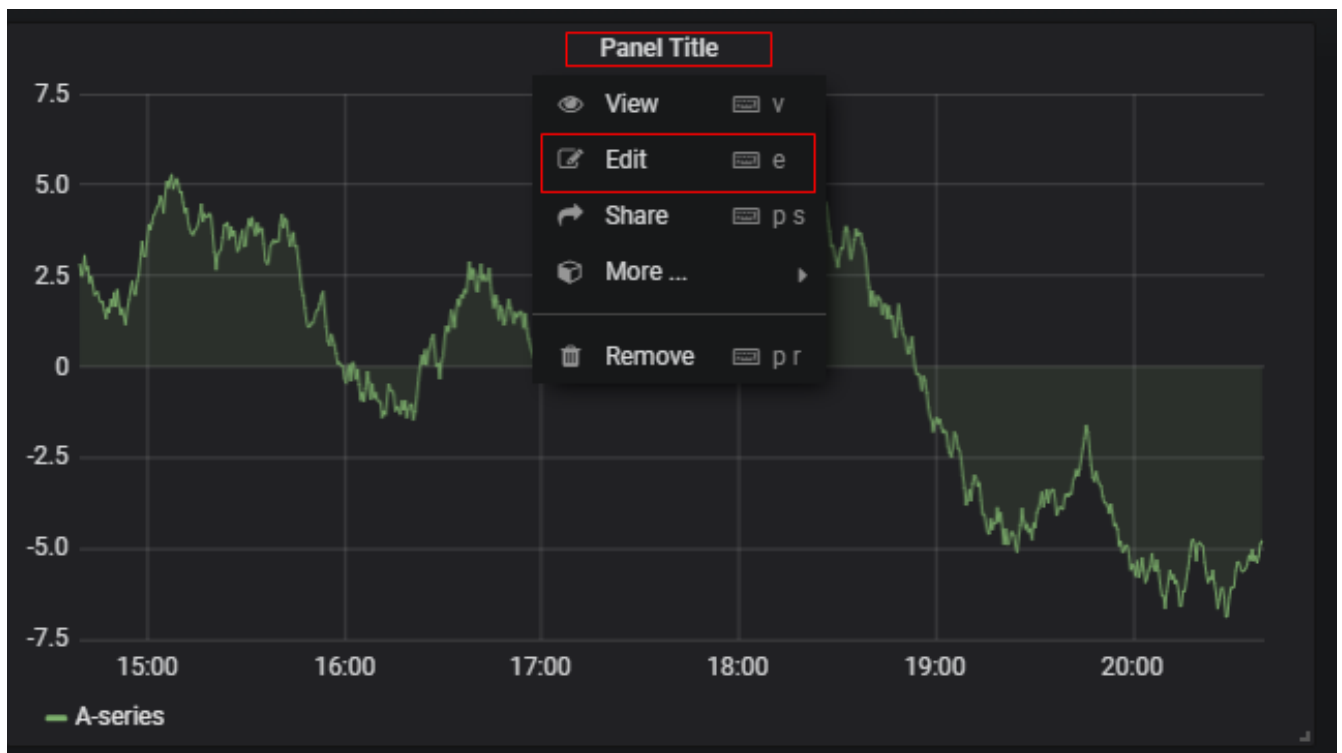


(2) 点击“添加”按钮

(3) 点击Graph 图标



(4) 出现下面图表的界面，点击Panel Title 选择Edit (编辑)



(5) 定义标题等基础信息

Graph General Metrics Axes Legend Display Alert Time range

Info

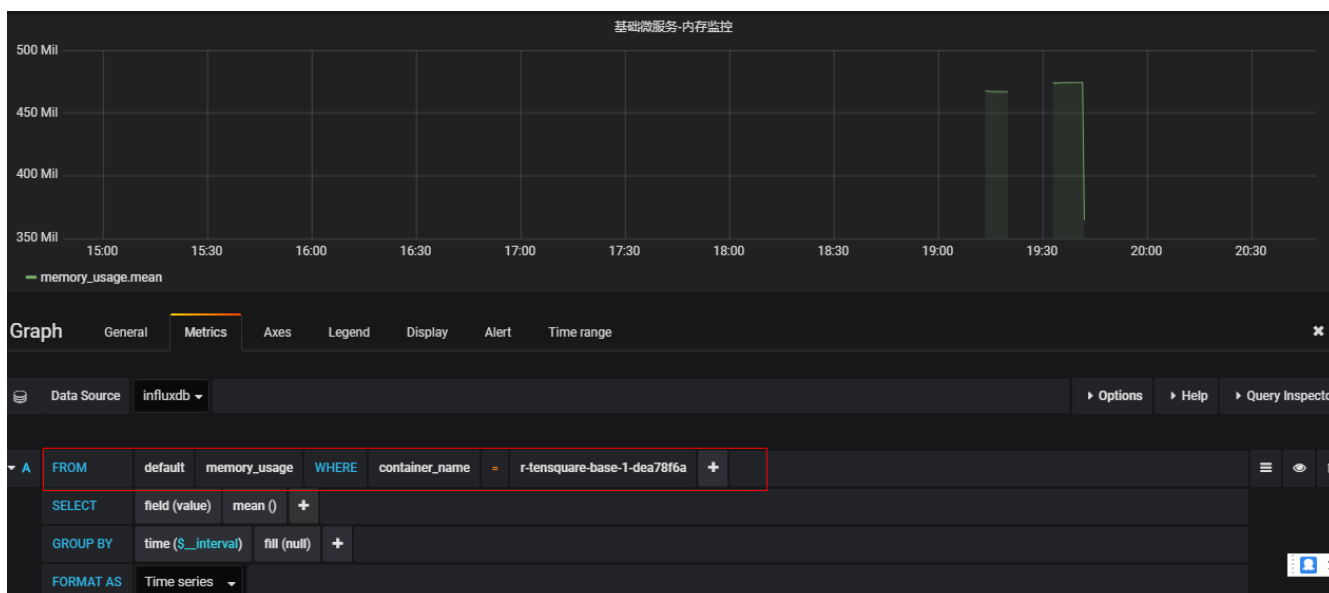
Title 基础微服务-内存监控

Description Panel description, supports markdown & links

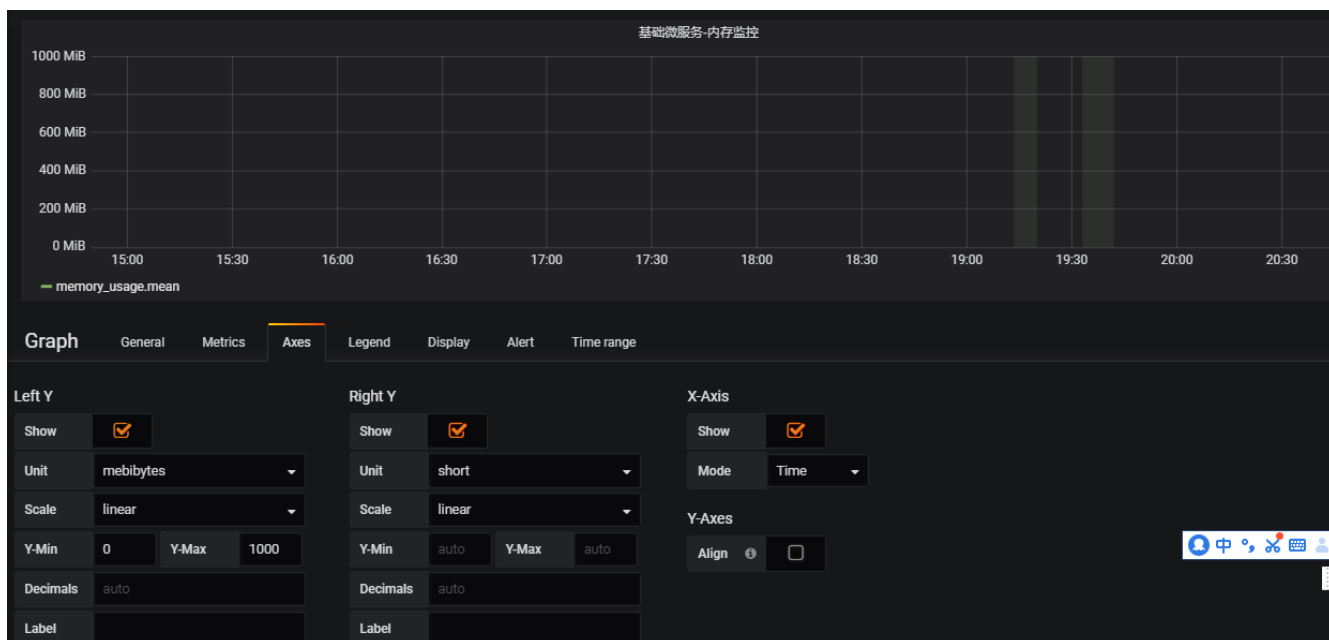
Transparent ☐

Repeat For each value of

(6) 设置查询的信息为内存，指定容器名称



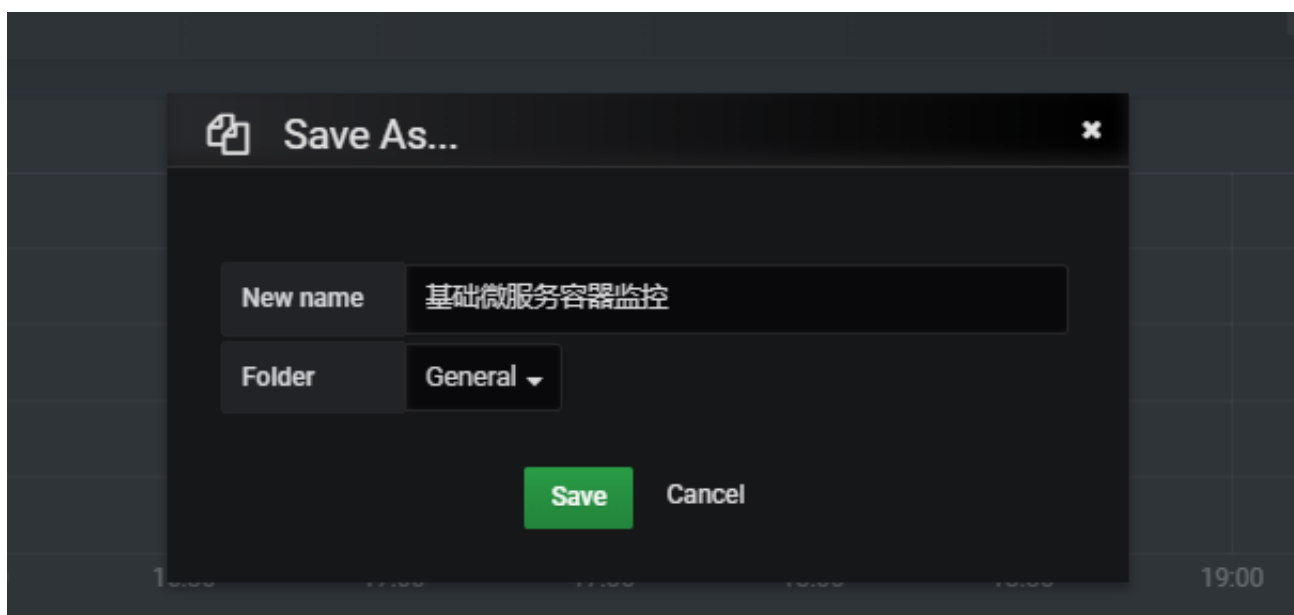
(7) 指定y轴的单位 为M



(8) 保存



填写名称

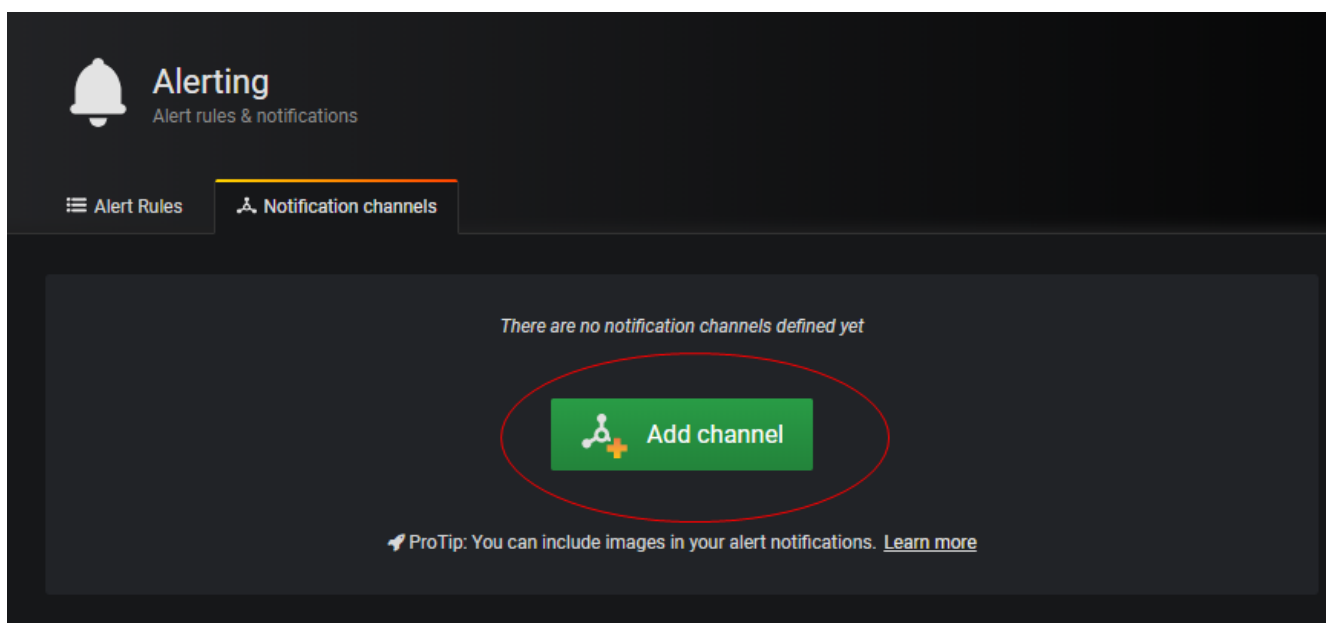


6.4.3 预警通知设置


(1) 选择菜单 alerting--> Notification channels



(2) 点击Add channel 按钮



(3) 填写名称，选择类型为webhook ,填写钩子地址



Alerting

Alert rules & notifications

Alert Rules

Notification channels

New Notification Channel

Name

scale-base

Type

webhook

Send on all alerts

☐

Include image

☒

Webhook settings

Url

http://192.168.184.135:9090/v1-webhooks/endpoint...

Http Method

POST

Username

Password

Save

Send Test

Back

这个钩子地址是之前对base微服务扩容的地址

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	scale-base	扩容容服务	扩容容动作 up ? 以步长 2		

(4) 点击SendTest 测试 观察基础微服务是否增加容器

(5) 点击save保存

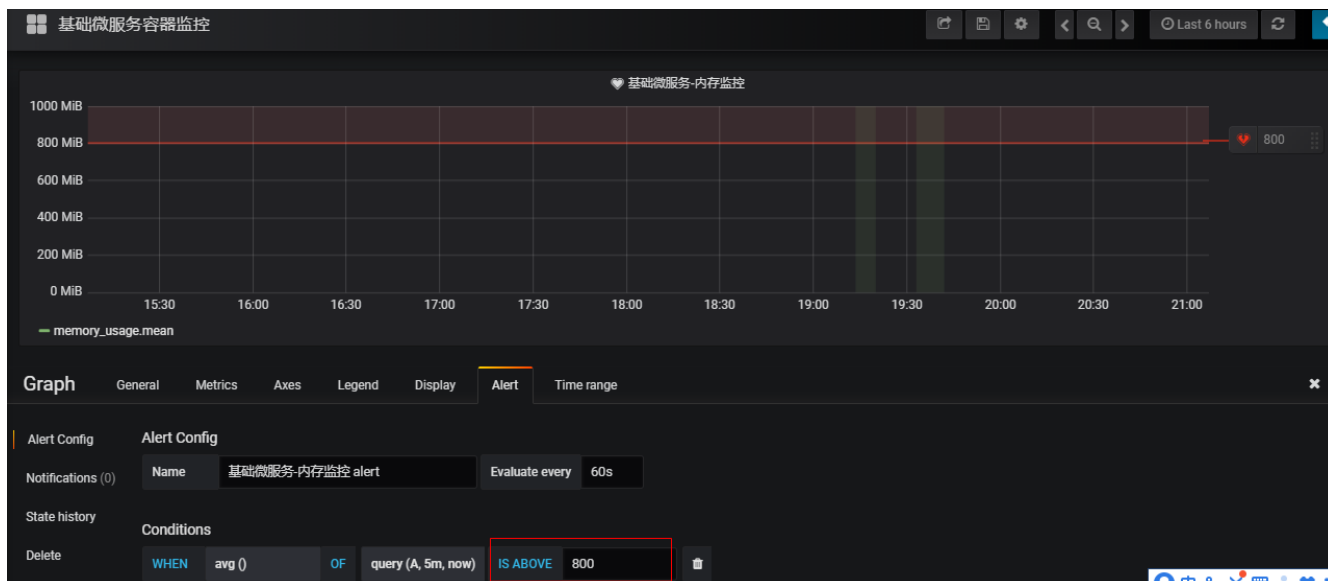
(6) 按照同样的方法添加缩容地址

6.4.4 仪表盘预警设置

(1) 再次打开刚刚编辑的仪表盘

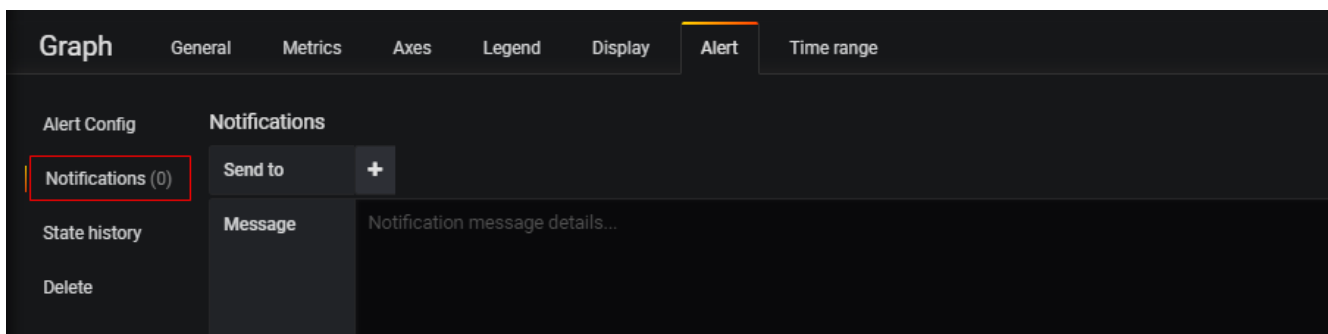


(2) 点击 Create Alert



设置预警线

(3) 选择通知



Graph

GeneralMetricsAxesLegendDisplayAlertTime range

Alert Config

Notifications

Notifications (1)

State history

Delete

Send to

scale-base %

+

Message

内存超过800M

保存更改