**Docker+Jenkins+SVN+Gradle**

**持续集成、集群部署**

# 安装docker-engine最新版

**1、centos系统更新**

yum -y update

**2、添加docker源**

tee /etc/yum.repos.d/docker.repo <<-'EOF'

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF

**3、安装docker-engine**

yum -y install docker-engine

**4、启动docker**

service docker start

# 希云cSphere免费版

<http://csphere.cn/hub/csphere>

## 部署控制器

curl -SsL -o /tmp/csphere-install.sh https://csphere.cn/static/csphere-install-v2.sh

sudo env ROLE=controller CSPHERE\_VERSION=0.13.0 /bin/sh /tmp/csphere-install.sh

## 部署Agent

打开浏览器，访问controller A主机的1016端口，第一次访问填入管理员邮箱密码注册，即可看到控制台的界面。 点击左侧的“主机”菜单，进入主机列表页面，点击“添加主机”并复制脚本，在Agent主机安装Agent程序，即可开始希云cSphere旅途。

sudo env ROLE=agent CONTROLLER\_IP=192.168.0.126 CONTROLLER\_PORT=1016 CSPHERE\_VERSION=0.13.0 AUTH\_KEY=39ce8737cb161b687616020be27bcbe8d8a44560c935253318e25ccc9ae82ddabca0659b2000584d /bin/sh /tmp/csphere-install.sh

# 添加docker命令

yum install wget

wget -P ~ https://github.com/yeasy/docker\_practice/raw/master/\_local/.bashrc\_docker;

echo "[ -f ~/.bashrc\_docker ] && . ~/.bashrc\_docker" >> ~/.bashrc; source ~/.bashrc

可执行：

docker-pid 可以获取某个容器的 PID：docker-pid myjenkins

docker-enter 可以进入容器或直接在容器内执行命令：docker-enter myjenkins

# 制作Tomcat容器(平台，项目团队跳过)

## 添加Tomcat配置文件

tomcat-users.xml

<?xml version='1.0' encoding='utf-8'?>

<tomcat-users>

<role rolename="manager-gui"/>

<role rolename="manager-status"/>

<role rolename="manager-script"/>

<role rolename="admin-gui"/>

<user username="eteng" password="yiteng" roles="manager-gui,manager-status,manager-script,admin-gui" />

</tomcat-users>

timezone

Asia/Shanghai

## Dockerfile

FROM tomcat:8-jre8

MAINTAINER "yitengshidai <hjx@e-teng.com.cn>"

ADD timezone /etc/

ADD tomcat-users.xml /usr/local/tomcat/conf/

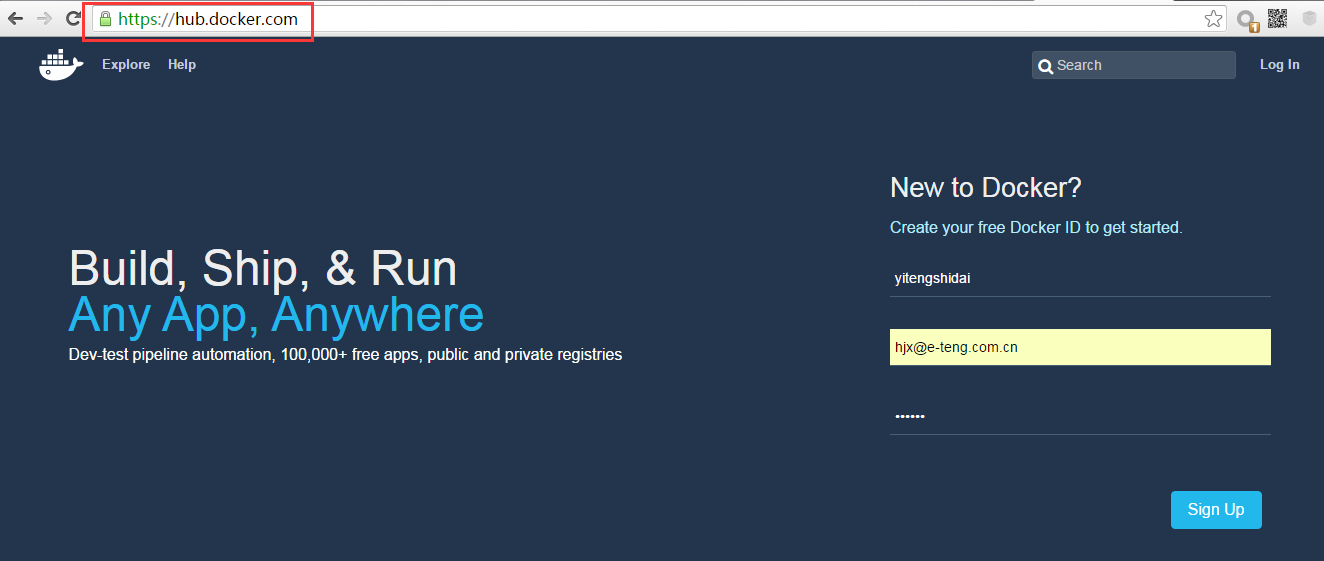
RUN rm -rf /usr/local/tomcat/webapps/docs

RUN rm -rf /usr/local/tomcat/webapps/examples

## 创建ettomcat镜像

docker build -t ettomcat /usr/local/dockerfile/tomcat

## 注册docker用户



Docker id:yitengshidai,密码：yiteng

## 标记本地镜像，归入仓库

docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]

docker tag ettomcat yitengshidai/tomcat:0.1

## 上传yitengshidai/tomcat镜像

docker login

Username: yitengshidai

Password: yiteng

Login Succeeded

docker push yitengshidai/tomcat:0.1

# 增加yitengshidai/tomcat容器

增加其他参数：-e JAVA\_OPTS=’-server –Xms256m –Xmx512m’

**部署两个tomcat,模拟集群环境**

docker run -d --name mytomcat -p 6001:8080 -e JAVA\_OPTS='-Xms128m -Xmx512m' yitengshidai/tomcat:0.1

docker run -d --name mytomcat2 -p 6002:8080 -e JAVA\_OPTS='-Xms128m -Xmx512m' yitengshidai/tomcat:0.1

# 制作Jenkins容器(平台，项目团队跳过)

## Jenkins容器修改时区

进入容器：docker-enter myjenkins

dpkg-reconfigure tzdata （6，67）

退出，重启容器：docker restart myjenkins

## 容器提交，创建镜像

docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

docker commit myjenkins yitengshidai/jenkins:0.1

查看镜像详细信息

docker inspect myjenkins yitengshidai/jenkins:0.1

## 上传yitengshidai/jenkins镜像

docker login

Username: yitengshidai

Password: yiteng

Login Succeeded

docker push yitengshidai/jenkins:0.1

# 增加yitengshidai/jenkins容器

docker run -d --name myjenkins -p 8080:8080 -v /var/jenkins\_home yitengshidai/jenkins:0.1

## Jenkins容器目录结构

进入容器：docker-enter myjenkins

Jenkins主目录：/var/jenkins\_home

项目工作空间及源代码：/var/jenkins\_home/workspace/[jenkins任务名]

War包产出物：/var/jenkins\_home/workspace/sinwar-demo/build/libs/\*.war

归档目录：/var/jenkins\_home/jobs/[jenkins任务名]

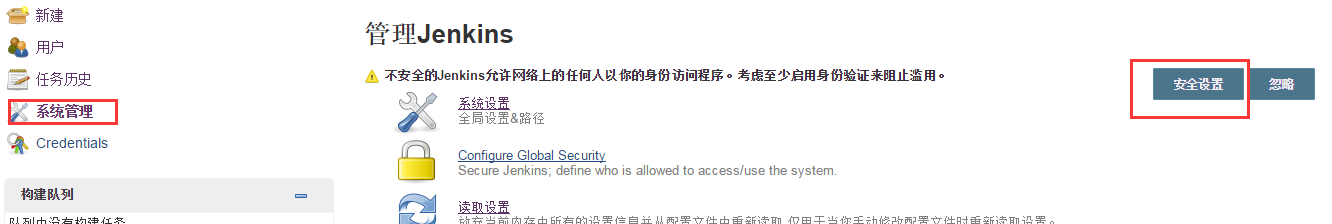
某次发布产出物：/var/jenkins\_home/jobs/[jenkins任务名]/builds/[发布序号]/archive/build/libs/\*.war

## 访问Jenkins

http://192.168.0.126:8080/

# Jenkins配置

## Jenkins安全设置





### 添加管理员

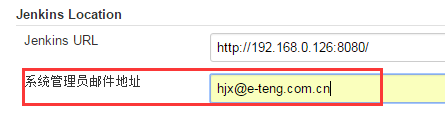
用户名：eteng，密码：yiteng

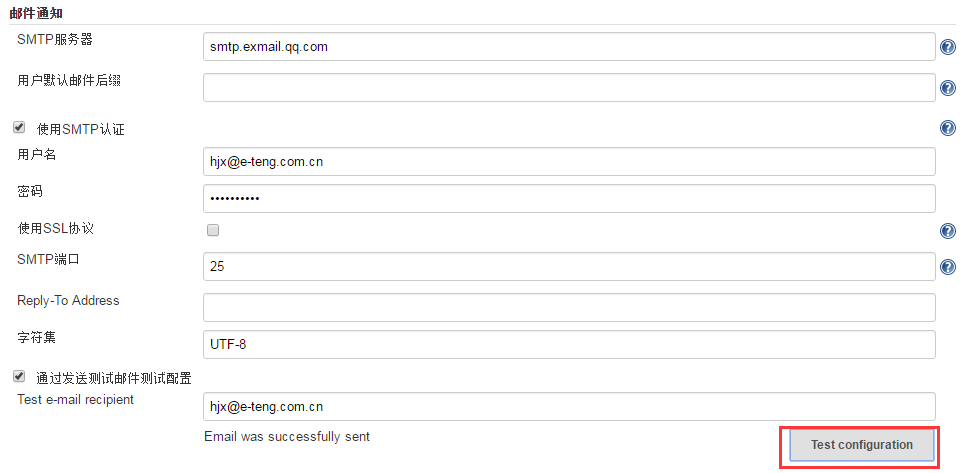


## Jenkins系统设置



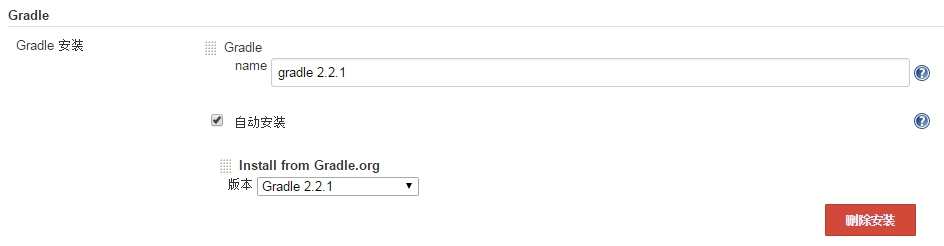
### 配置jenkins SMTP





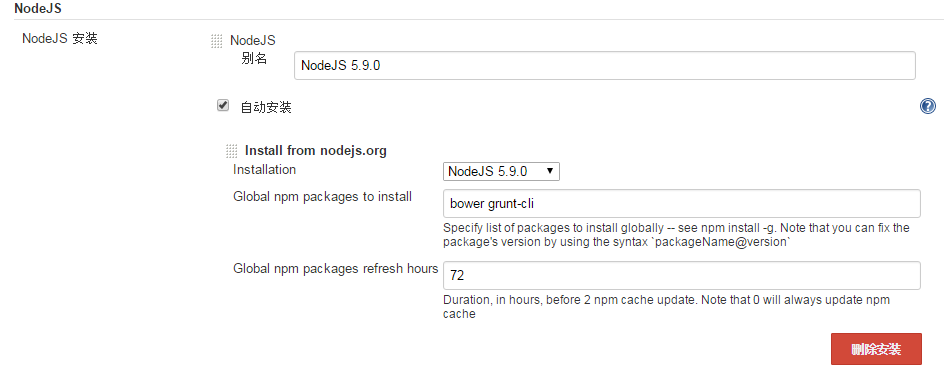
### 配置jenkins gradle

（运行任务时自动安装）



### 配置jenkins nodejs

（运行任务时自动安装）

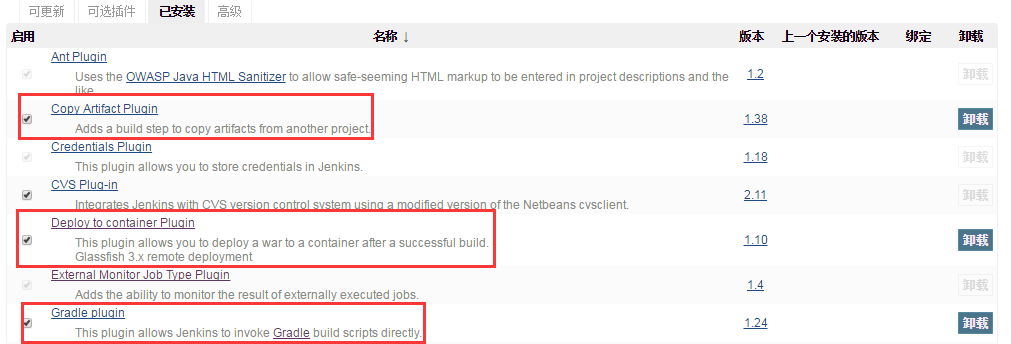


## 安装jenkins插件

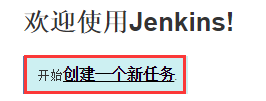


高级->升级站点->立即获取（获取可更新列表）

1. Gradle plugin:用于生成war
2. Deploy to container Plugin:用于将war发布至tomcat
3. Copy Artifact Plugin:用于版本回退
4. NodeJS Plugin:nodejs插件



## 创建项目发布任务(后台)

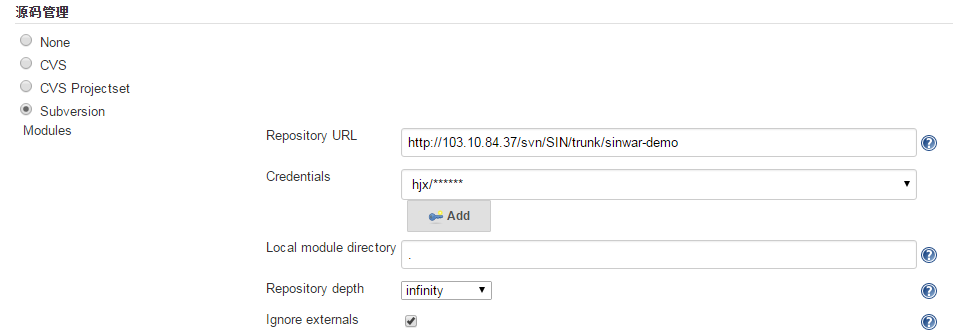


### 创建自由风格软件项目

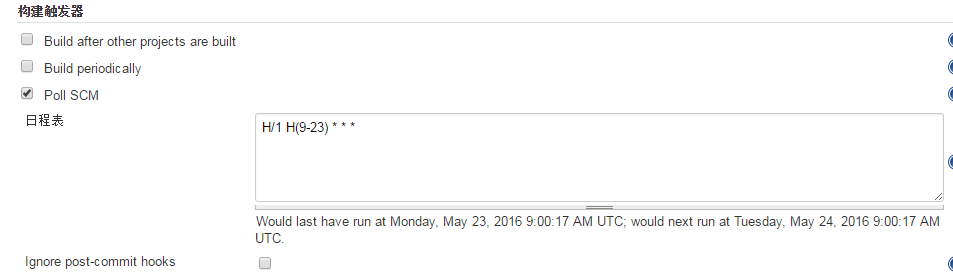




### 源码管理

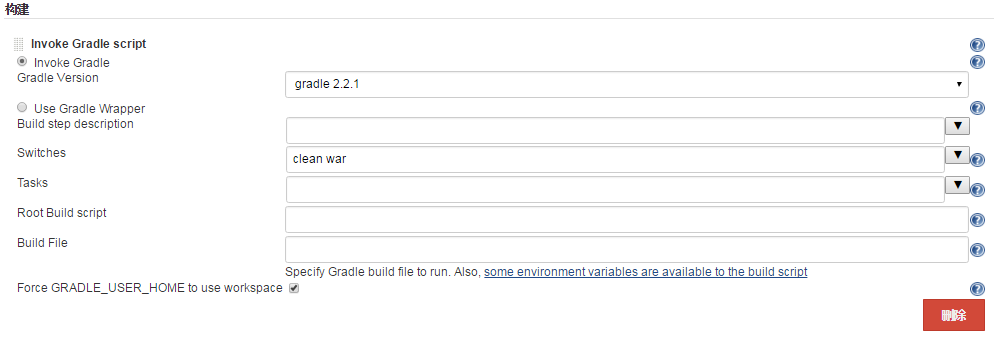


### 构建触发器



### 增加构建步骤

#### gradle

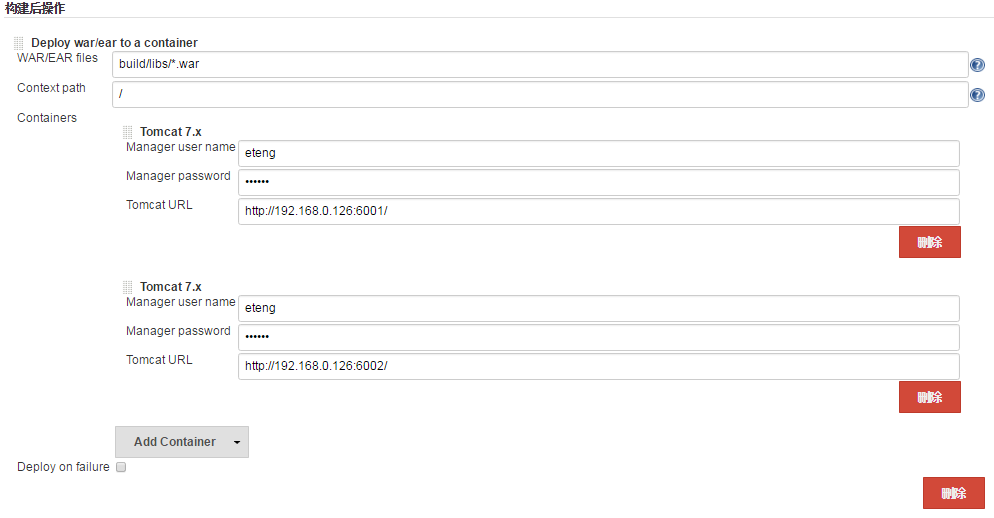


### 增加构建后操作步骤

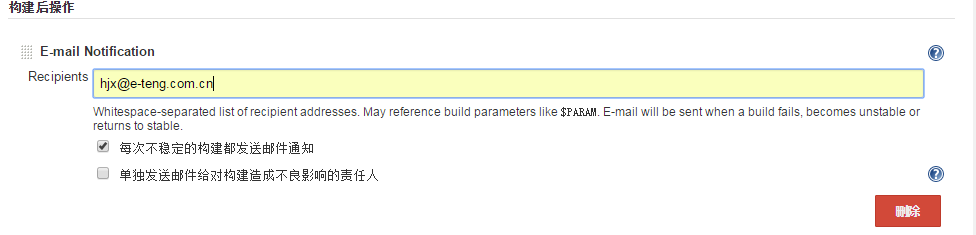
#### 归档构建成果war



#### 发布war到远程Tomcat



#### 发送任务邮件提醒



## 创建项目发布任务(前台)

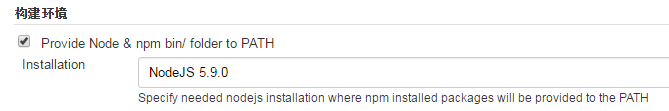
### Nginx配置说明

Nginx配置文件：/etc/nginx/nginx.conf

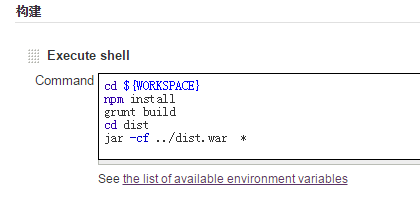
Nginx发布路径：/usr/share/nginx/html

通过tomcat进行发布dist.war，然后nginx指向tomcat/webapps/dist目录。

### 配置构建环境

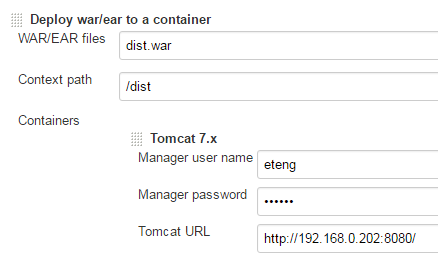


### 增加构建步骤



### 增加构建后操作步骤



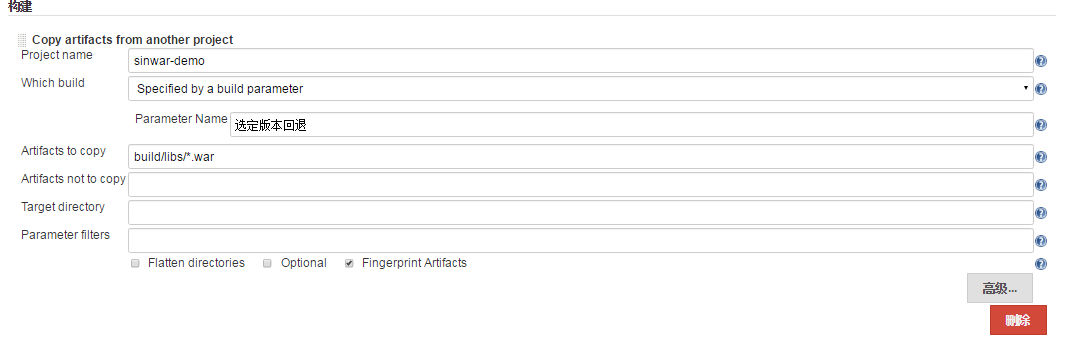


## 创建指定版本回退任务

### 参数化构建过程



### 拷贝历史版本



# 增加数据库容器

## 增加mysql/mysql-server容器

增加密码参数：-e MYSQL\_ROOT\_PASSWORD=my-secret-pw

docker run -d --name mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=eteng mysql/mysql-server

## 增加nicktav/oracle11gr1-ee容器

参考：<https://hub.docker.com/r/nicktav/oracle11gr1-ee/>

安装完成后：

系统root密码 - mypassword

系统oracle密码 - mypassword

Oracle SYS用户密码 - mypassword

Oracle SYSTEM用户密码 – mypassword

Oracle用户首次登陆时，要求修改密码

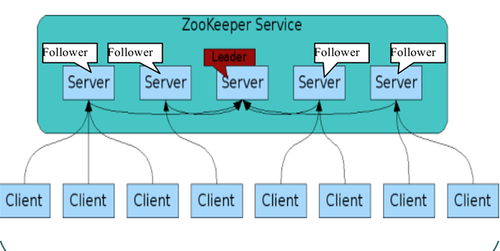
docker run -d –name oracle11g -h oracledb -p 10022:22 -p 1521:1521 -v /u01/app/oracle:/u01/app/oracle -v /u02/oradata:/u02/oradata -v /u02/flash\_recovery\_area:/u02/flash\_recovery\_area nicktav/oracle11gr1-ee

# 安装zookeeper集群

## Zookeeper简介

ZooKeeper是一个分布式的，开放源码的分布式**应用程序协调服务**，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

## 增加garland/zookeeper集群



参照文档：[使用不同Docker网络模型搭建Zookeeper集群](https://github.com/denverdino/aliyungo/wiki/Zookeeper-cluster-with-Docker)

[docker环境搭建zk集群](https://segmentfault.com/a/1190000004341503)

下载镜像

docker pull garland/zookeeper

部署zookeeper集群

docker run -d \

--name=zk1 \

--net=host \

-e SERVER\_ID=1 \

-e ADDITIONAL\_ZOOKEEPER\_1=server.1=localhost:2888:3888 \

-e ADDITIONAL\_ZOOKEEPER\_2=server.2=localhost:2889:3889 \

-e ADDITIONAL\_ZOOKEEPER\_3=server.3=localhost:2890:3890 \

-e ADDITIONAL\_ZOOKEEPER\_4=clientPort=2181 \

garland/zookeeper

docker run -d \

--name=zk2 \

--net=host \

-e SERVER\_ID=2 \

-e ADDITIONAL\_ZOOKEEPER\_1=server.1=localhost:2888:3888 \

-e ADDITIONAL\_ZOOKEEPER\_2=server.2=localhost:2889:3889 \

-e ADDITIONAL\_ZOOKEEPER\_3=server.3=localhost:2890:3890 \

-e ADDITIONAL\_ZOOKEEPER\_4=clientPort=2182 \

garland/zookeeper

docker run -d \

--name=zk3 \

--net=host \

-e SERVER\_ID=3 \

-e ADDITIONAL\_ZOOKEEPER\_1=server.1=localhost:2888:3888 \

-e ADDITIONAL\_ZOOKEEPER\_2=server.2=localhost:2889:3889 \

-e ADDITIONAL\_ZOOKEEPER\_3=server.3=localhost:2890:3890 \

-e ADDITIONAL\_ZOOKEEPER\_4=clientPort=2183 \

garland/zookeeper

查看集群(如果没有nc,通过 yum install nc 安装)

查看zk1(follower)

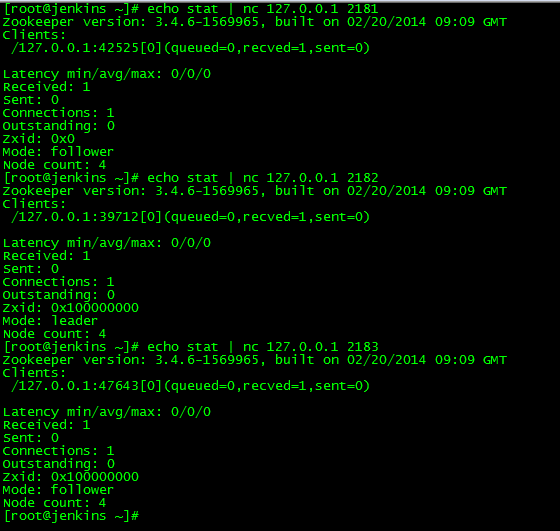
echo stat | nc 127.0.0.1 2181

查看zk2(leader)

echo stat | nc 127.0.0.1 2182

查看zk3(follower)

echo stat | nc 127.0.0.1 2183

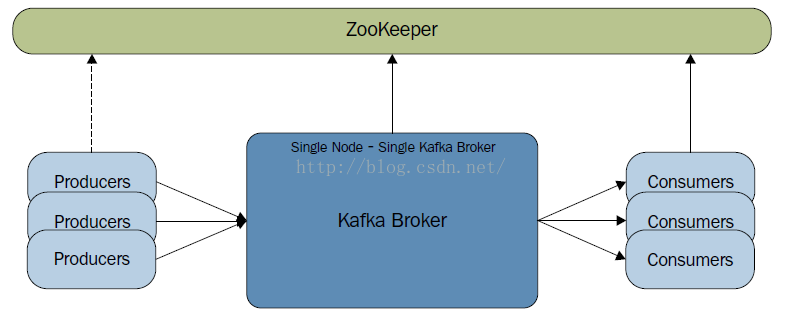


# 安装wurstmeister/kafka

kafka是一个消息队列服务器,服务称为broker, 消息发送者称为producer, 消息接收者称为consumer。

通常我们部署多个broker以提供高可用性的消息服务集群。典型的是3个broker;消息以topic的形式发送到broker,消费者订阅topic,实现按需取用的消费模式。

创建topic需要指定replication-factor(复制数目, 通常=broker数目);每个topic可能有多个分区(partition), 每个分区的消息内容不会重复。



配置说明参考：<http://www.tuicool.com/articles/QBrEFj>

## ~~安装扩展源EPEL~~

EPEL(http://fedoraproject.org/wiki/EPEL) 是由 Fedora 社区打造，为 RHEL 及衍生发行版如 CentOS、Scientific Linux 等提供高质量软件包的项目。

yum -y install epel-release

## ~~安装python-pip~~

Python pip是一个安装和管理 Python 包的工具 ,是easy\_install的替代品。

yum -y install python-pip

## ~~通过pip安装docker-compose~~

docker-compose：简化复杂容器应用的利器。Compose是用于定义和运行复杂Docker应用的工具。你可以在一个文件中定义一个多容器的应用，然后使用一条命令来启动你的应用，然后所有相关的操作都会被自动完成。

pip install -U docker-compose

## 增加wurstmeister/kafka容器

### 调整防火墙

CentOS7默认的防火墙不是iptables,而是firewall.

#### ~~Firewall相关命令~~

#查看防火墙状态

firewall-cmd --state

#添加开放端口范围

firewall-cmd --zone=public --add-port=2181-2183/tcp --permanent

#重启firewall

systemctl restart firewalld

#查看防火墙开放端口明细

firewall-cmd --list-all

#### 卸载Firewall、安装iptables

参考：<http://www.cnblogs.com/kreo/p/4368811.html>

1. 禁用/停止自带的firewalld服务

#停止firewalld服务

systemctl stop firewalld

#禁用firewalld服务

systemctl mask firewalld

1. 安装iptable iptable-service

#先检查是否安装了iptables

service iptables status

#安装iptables

yum install -y iptables

#升级iptables

yum update iptables

#安装iptables-services

yum install iptables-services

1. 设置现有规则

#查看iptables现有规则

iptables -L -n

#先允许所有,不然有可能会杯具

iptables -P INPUT ACCEPT

#清空所有默认规则

iptables -F

#清空所有自定义规则

iptables -X

#所有计数器归0

iptables -Z

#允许来自于lo接口的数据包(本地访问)

iptables -A INPUT -i lo -j ACCEPT

#开放22端口

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

#开放21端口(FTP)

iptables -A INPUT -p tcp --dport 21 -j ACCEPT

#开放80端口(HTTP)

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

#开放443端口(HTTPS)

iptables -A INPUT -p tcp --dport 443 -j ACCEPT

#允许ping

iptables -A INPUT -p icmp --icmp-type 8 -j ACCEPT

#允许接受本机请求之后的返回数据 RELATED,是为FTP设置的

iptables -A INPUT -m state --state  RELATED,ESTABLISHED -j ACCEPT

#其他入站一律丢弃

iptables -P INPUT DROP

#所有出站一律绿灯

iptables -P OUTPUT ACCEPT

#所有转发一律丢弃

iptables -P FORWARD DROP

#保存上述规则

service iptables save

#注册iptables服务

#相当于以前的chkconfig iptables on

systemctl enable iptables.service

#开启服务

systemctl start iptables.service

#查看状态

systemctl status iptables.service

### 运行kafka容器

docker run -d --name kafka1 -e KAFKA\_ADVERTISED\_HOST\_NAME=192.168.0.202 -e KAFKA\_ADVERTISED\_PORT=9092 -e KAFKA\_BROKER\_ID=1 -e KAFKA\_ZOOKEEPER\_CONNECT=192.168.0.202:2181,192.168.0.202:2182,192.168.0.202:2183 -p 9092:9092 wurstmeister/kafka

docker run -d --name kafka2 -e KAFKA\_ADVERTISED\_HOST\_NAME=192.168.0.202 -e KAFKA\_ADVERTISED\_PORT=9093 -e KAFKA\_BROKER\_ID=2 -e KAFKA\_ZOOKEEPER\_CONNECT=192.168.0.202:2181,192.168.0.202:2182,192.168.0.202:2183 -p 9093:9092 wurstmeister/kafka

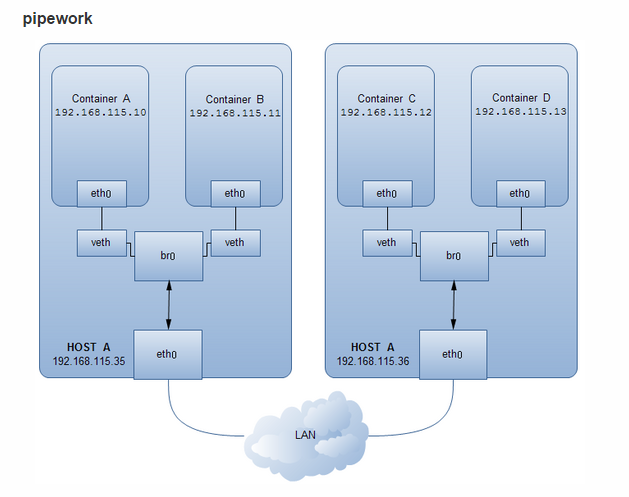
# 安装Hadoop集群

## ~~Docker网络增强插件Pipework~~

Pipework是由Docker的一个工程师设计的解决方案，它让容器能够在“任意复杂的场景”下进行连接。Pipework是Docker的一个网络功能增强插件，它使用了cgroups和namespacpace。

参考：<http://www.tuicool.com/articles/Y7rYrm>

### 网络拓扑图



### 安装步骤

1. 下载pipework、修改文件权限

curl -o /usr/local/bin/pipework <https://raw.githubusercontent.com/jpetazzo/pipework/master/pipework> && chmod +x /usr/local/bin/pipework

1. 安装网桥工具包brctl

yum install bridge-utils

1. 创建网桥br0、启动br0、修改br0 ip、删除物理网卡ip、将主机eth0桥接到br0上、删除默认路由、添加网关到br0路由。（这里由于是远程操作，中间网络会断掉，所以放在一条命令中执行。）

brctl addbr br0;\

ip link set dev br0 up;\

ip addr add 192.168.0.207/24 dev br0; \

ip addr del 192.168.0.207/24 dev eth0; \

brctl addif br0 eth0; \

route del default; \

route add default gw 192.168.0.1 dev br0

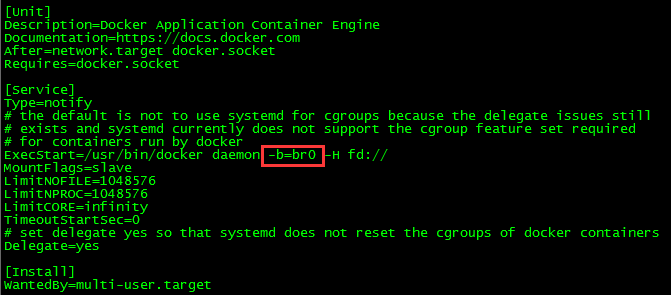
1. 修改docker启动项，调整docker使用网桥br0

**停止docker**

Service docker stop

**找到docker.service 启动文件,并编辑：**

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



**删除默认网桥接口docker0**

ifconfig docker0 down

brctl delbr docker0

**重新加载service daemon**

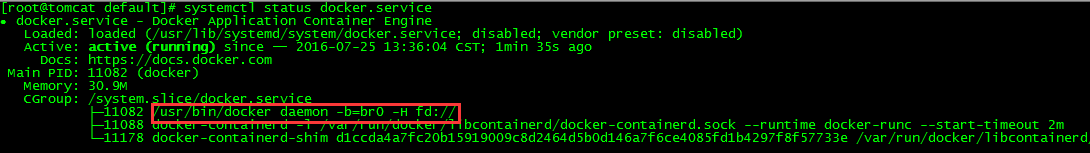
systemctl daemon-reload

**启动docker**

service docker start

**查看docker 启动状态**

systemctl status docker.service



1. 服务器重启（需要重新执行11.1红字部分,包括以下）

docker start amb-consul amb-server amb1 amb2

pipework br0 amb-server 192.168.0.230/24@192.168.0.1;\

pipework br0 amb1 192.168.0.231/24@192.168.0.1;\

pipework br0 amb2 192.168.0.232/24@192.168.0.1

## ~~使用Docker自定义网络~~

1. 安装网桥工具包brctl

yum install bridge-utils

1. 创建docker自定义网络br0,指定网桥名称br0、删除物理网卡ip、将主机eth0桥接到br0上、删除默认路由、添加网关到br0路由。（这里由于是远程操作，中间网络会断掉，所以放在一条命令中执行。）

注：192.168.0.207是服务器宿主机IP, 192.168.0.1是网关

docker network create -d bridge\

-o "com.docker.network.bridge.name"="br0"\

--subnet=192.168.0.0/24 \

--gateway=192.168.0.207 \

br0;\

ip addr del 192.168.0.207/24 dev eth0; \

brctl addif br0 eth0; \

route del default; \

route add default gw 192.168.0.1 dev br0

1. 重启服务器执行

ip addr del 192.168.0.207/24 dev eth0; \

service docker start;\

brctl addif br0 eth0; \

route del default; \

route add default gw 192.168.0.1 dev br0

1. Docker容器指定自定义网络及ip举例

docker run -d --name tt --net=br0 --ip=192.168.0.230 yitengshidai/tomcat:0.1

1. 公司修改后脚本文件:

**amb-consol:使用宿主机ip**

**amb-server:192.168.0.230**

**amb1(amb-agent):192.168.0.231**

**amb2(amb-agent):192.168.0.232**

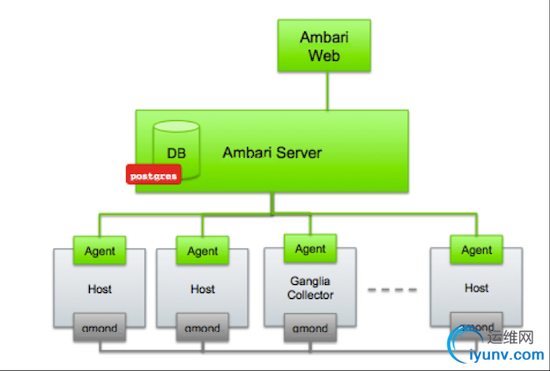
**ambary-shell(安装hdfs等服务需要):192.168.0.229**

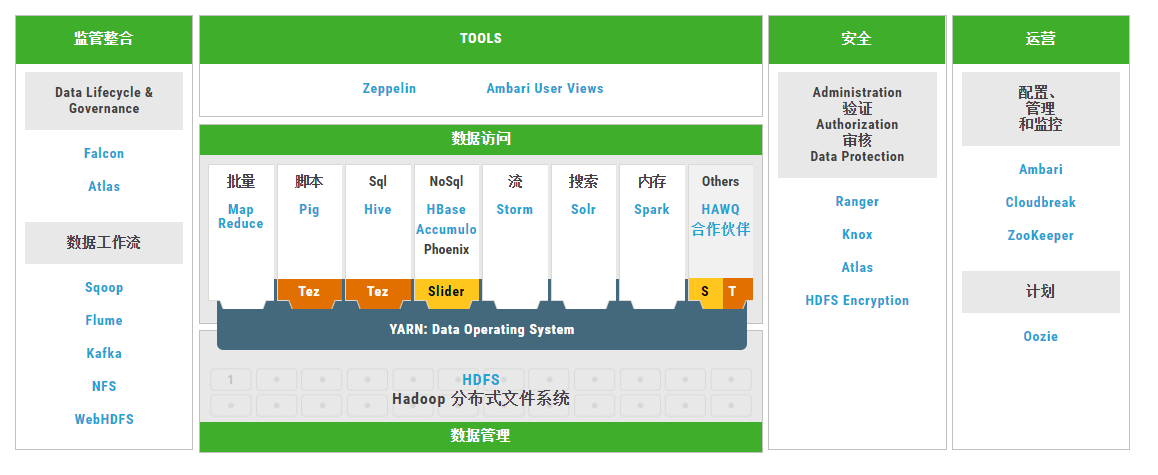
## 安装sequenceiq/ambari容器

<https://hub.docker.com/r/sequenceiq/ambari/>

ambaris: 是hortonworks开发的一个自动部署、管理、监控 HDP（Hortonworks Data Platform， http://hortonworks.com/hdp/，由hadoop生态环境中的一些开源软件组成）的开源软件。与之相对应的有cloudera manager， cloudera 发展时间更长，网上能搜到的资料更多， 但是其开源性好像没有hortonworks好。

Ambari Server与agent 交互从而 管理集群节点，这里agent的注册可以是被动，也就是server 通过ssh连接到agent，然后启动agent 去注册，所以需要两个配置， agent上需要能够ssh 私钥 无密码登陆，而且agent上要配置server的地址。





1、下载镜像

docker pull sequenceiq/ambari

2、下载ambari-functions脚本文件

curl -O <https://raw.githubusercontent.com/sequenceiq/docker-ambari/master/ambari-functions>

3、执行脚本：ambari-functions

source ambari-functions

source命令：

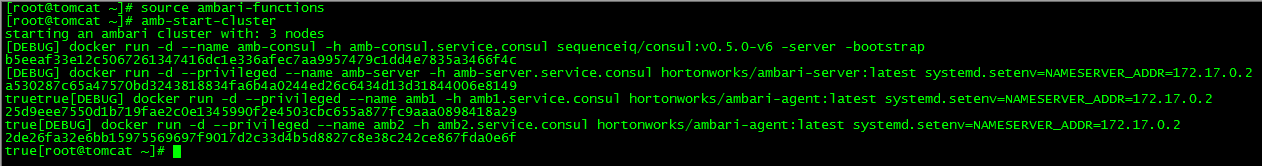
source命令也称为“点命令”，也就是一个点符号（.）,是bash的内部命令。

功能：使Shell读入指定的Shell程序文件并依次执行文件中的所有语句

source命令通常用于重新执行刚修改的初始化文件，使之立即生效，而不必注销并重新登录。

4、安装ambari-server和ambari-agent集群：将会启动1个consul server(分布式服务注册和发现consul)、1个Ambari server、2个Ambari agents

amb-start-cluster



5、创建并启动ambari-shell

* 创建ambari-shell容器

docker run -d --name shell --link amb-server:ambariserver hortonworks/ambari-server

* 进入ambari-shell容器

docker-enter shell



* 创建蓝图文件two-node-only-zookeeper（仅安装zookeeper服务）

vi /tmp/two-node-only-zookeeper

{

"host\_groups": [

{

"name": "master",

"components": [

{

"name": "ZOOKEEPER\_SERVER"

}

],

"cardinality": "1"

},

{

"name": "slave\_1",

"components": [

{

"name": "ZOOKEEPER\_CLIENT"

}

],

"cardinality": "1+"

}

],

"Blueprints": {

"blueprint\_name": "two-node-only-zookeeper",

"stack\_name": "HDP",

"stack\_version": "2.5"

}

}

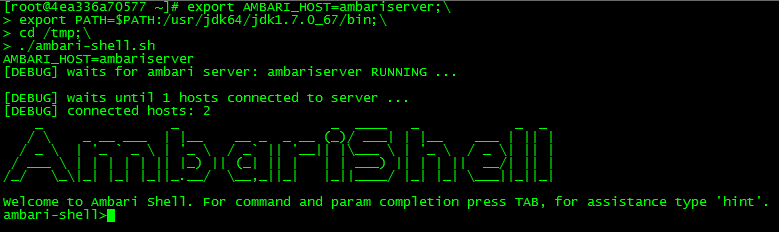
* 运行ambari-shell

export AMBARI\_HOST=ambariserver;\

export PATH=$PATH:/usr/jdk64/jdk1.7.0\_67/bin;\

cd /tmp;\

./ambari-shell.sh



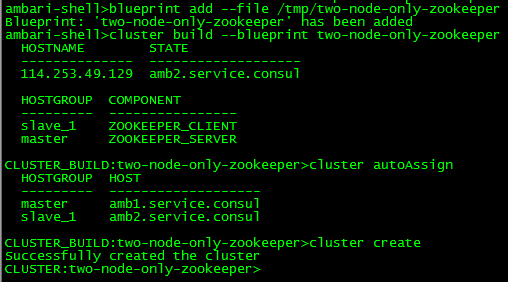
6、通过ambari-shell创建hadoop集群环境

blueprint add --file /tmp/two-node-only-zookeeper

cluster build --blueprint two-node-only-zookeeper

cluster autoAssign

cluster create



7、客户端机器添加网络路由

**Windows(需要以管理员身份运行cmd): route add 172.17.0.0 mask 255.255.0.0 192.168.0.207**



**将consul容器作为客户端机器的DNS服务器：netsh interface ip add dns "无线网络连接 2" 172.17.0.2**

**Linux: route add -net 172.17.0.0/16 192.168.0.207**

8、手动安装其他服务

ZooKeeper： ZooKeeper Server、ZooKeeper Client

HDFS: NameNode、SNameNode、DataNode、HDFS Client

YARN + MapReduce2: ResourceManager、History Server、App Timeline Server、

NodeManager、YARN Client、MapReduce2 Client

Tez: Tez Client

Pig: Pig Client

Slider: Slider Client

Hiver: Hive Metastore、HiveServer2、MySQL Server、WebHCat Server、

Hive Client、HCat Client

Spark: Livy Server、Spark History Server、Spark Client

Zeppelin: Zeppelin Notebook

Oozie:

### 修复quick link(hostname)的问题

1. vi /etc/ambari-agent/conf/public-hostname.sh

#!/bin/bash

echo ` ip addr |grep inet|grep -v 127.0.0.1|grep -v 172.17.0.1|grep -v inet6|awk '{print $2}'|sed 's/\/16//g'`

2、在amb1和amb2分别执行并重启容器

### 安装Zeppelin Notebook

需要spark安装了Livy Server

### 安装Oozie

1、数据库选用之前hive已经安装过的mysql。手工创建oozie用户和库，并分配权限。

GRANT ALL PRIVILEGES ON \*.\* TO 'oozie'@'%' IDENTIFIED BY 'oozie' WITH GRANT OPTION;

2、需要手工往amb-server复制并配置mysql jdbc驱动

复制jdbc驱动到amb-server

docker cp /root/mysql-connector-java-5.1.38.jar amb-server:/root

在amb-server执行

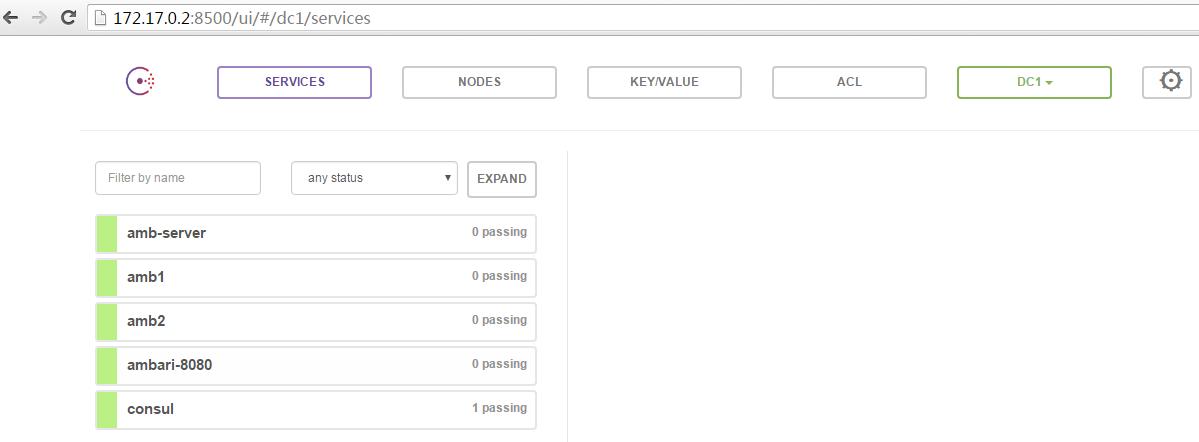
ambari-server setup --jdbc-db=mysql --jdbc-driver=/root/mysql-connector-java-5.1.38.jar

3、需要再oozie server安装的容器上，手工安装zip。

Yum -y install zip

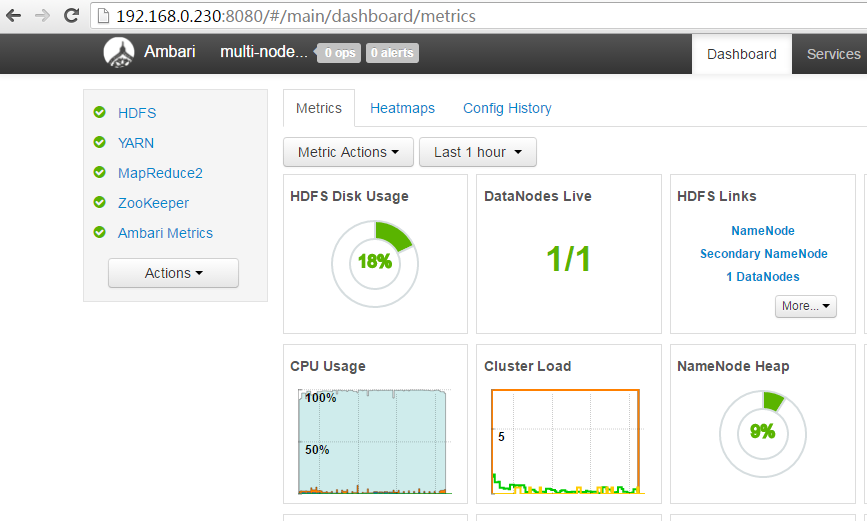
### Consul管理页面

(使用宿主机ip): <http://172.17.0.2:8500/>



### ambari server管理页面

<http://192.168.0.230:8080/> 用户密码 admin/admin



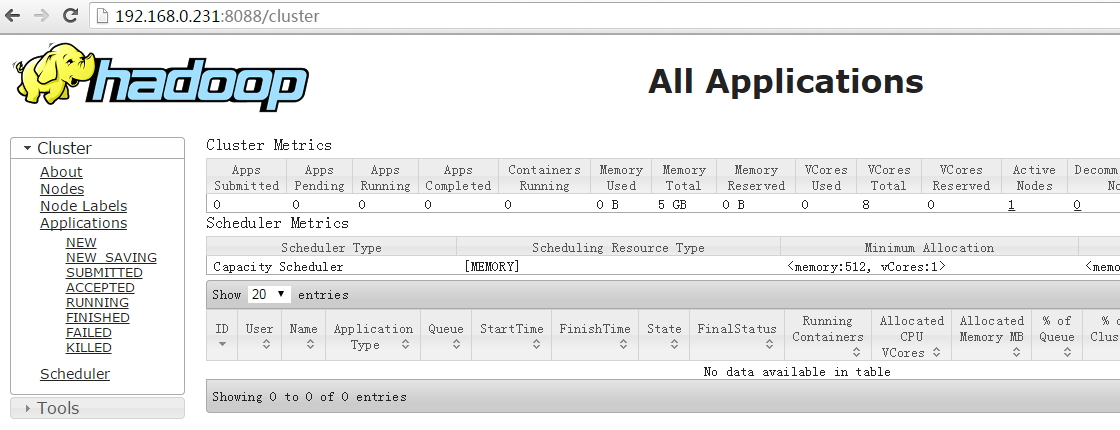
### HDFS NameNode管理界面

<http://192.168.0.231:50070/>



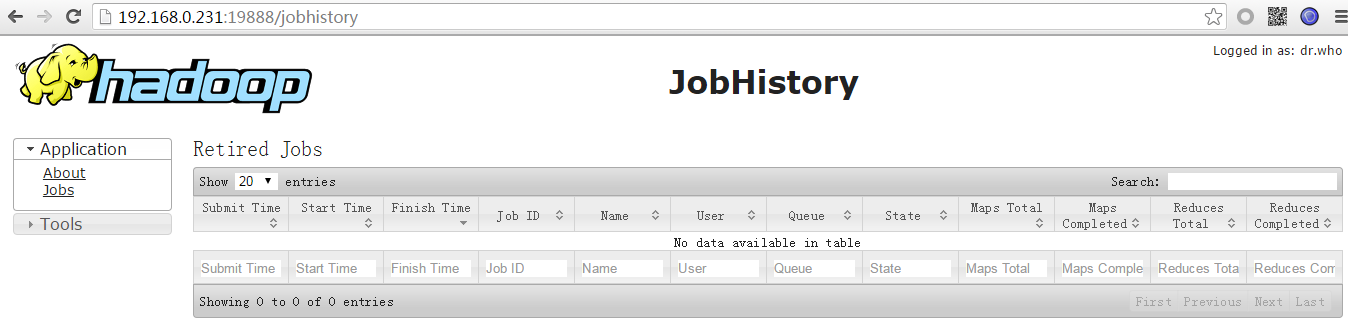
### Yarn ResourceManager 管理页面

<http://192.168.0.231:8088/>

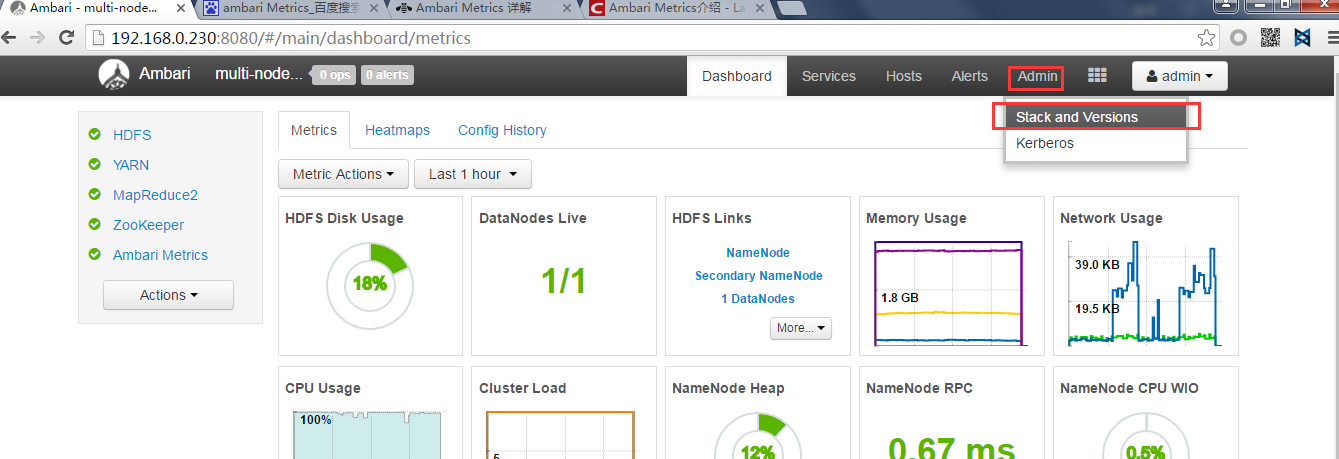


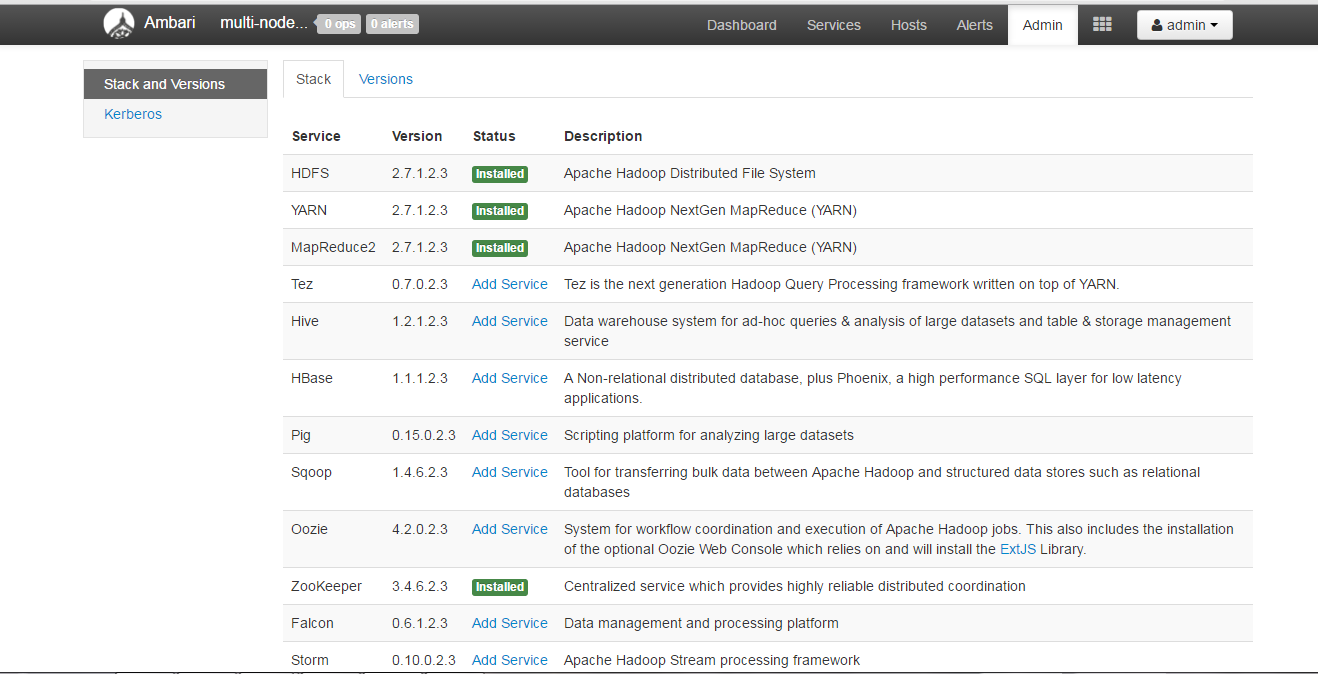
### MapReduce2 JobHistory 管理页面

<http://192.168.0.231:19888/>



### 查看已安装组件和版本





## Hadoop开发

参考 [使用Eclipse插件连接配置Mapreduce说明与教程(hadoop-eclipse-plugin 2.6)](http://www.voidcn.com/blog/huitoukest/article/p-5708822.html)

### 查找DFS Master配置

登陆ambari server管理界面->HDFS->Configs->Advanced-> Advanced hdfs-site

-> dfs.namenode.rpc-address



### 查找Map/Reduce(v2) Master配置

登陆ambari server管理界面->YARN->Configs->Advanced-> Advanced yarn-site

-> yarn.resourcemanager.scheduler.address



Eclipse插件：

