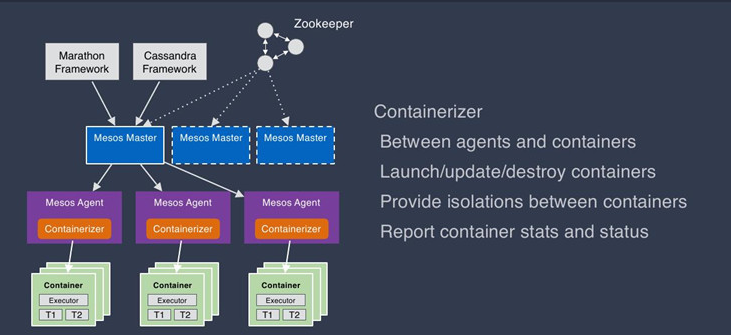
|  |
| --- |
| www.linuxfan.cn科技有限公司 |
| 项目：Marathon+Mesos+Zookeeper+Docker实战 |
| 版本：1-1 |

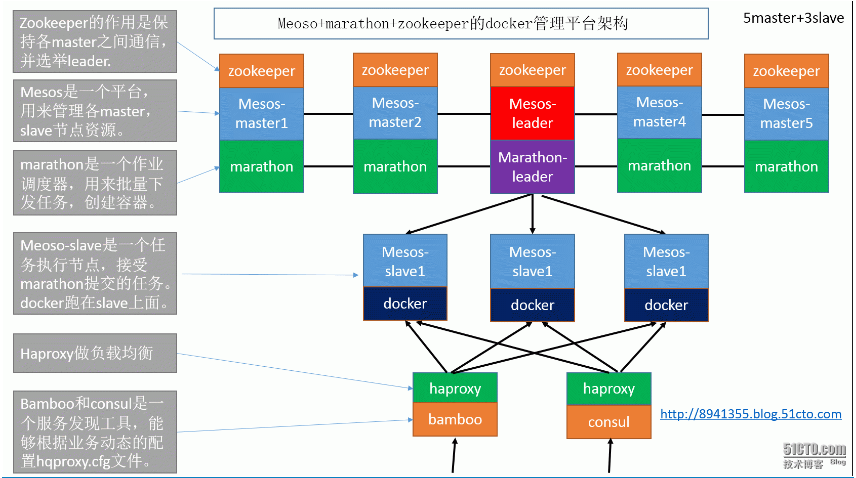
|  |
| --- |
| 作者：www.linuxfan.cn  2018-8-19 |

项目：Marathon+Mesos+Zookeeper+Docker实战

一、项目拓扑：



二、项目重点：



**1. Apache Mesos概述：**

**Apache Mesos：**Apache-Mesos是一款基于多资源(内存、CPU、磁盘、端口等)调度的开源集群管理套件，能使容错和分布式系统更加容易使用。官方网站http://mesos.apache.org/ ,软件能够自由下载使用；

Mesos实现了两级调度架构，它可以管理多种类型的应用程序。第一级调度是Master的守护进程，管理Mesos集群中所有节点上运行的Slave守护进程。集群由物理服务器或虚拟服务器组成，用于运行应用程序的任务，比如Hadoop和MPI作业。第二级调度由被称作Framework的“组件”组成。Framework包括调度器（Scheduler）和执行器（Executor）进程，其中每个节点上都会运行执行器。Mesos能和不同类型的Framework通信，每种Framework由相应的应用集群管理；  
Mesos Master协调全部的Slave，并确定每个节点的可用资源， 聚合计算跨节点的所有可用资源的报告，然后向注册到Master的Framework（作为Master的客户端）发出资源邀约。Framework可以根据应用程序的需求，选择接受或拒绝来自master的资源邀约。一旦接受邀约，Master即协调Framework和Slave，调度参与节点上任务，并在容器中执行；

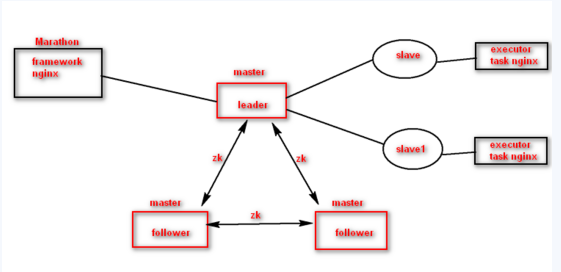
**Apache Mesos专业术语：**

Mesos Master：负责管理各个Framework和Slave，并将Slave上的资源分配给各个Framework；

Mesos Slave：负责管理本节点上的各个Mesos Task，为各个Executor分配资源；

Framework：计算机框架，如:Hadoop、Spark等，可以通过MesosSchedulerDiver接入Mesos；

Executor：执行器，在Mesos Slave上安装，用于启动计算框架中的Task；



**Apache Mesos处理流程：**

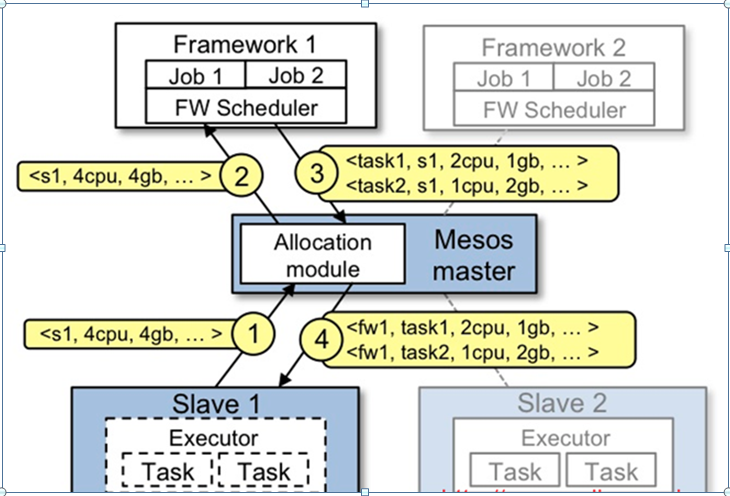
1.集群中的所有slave节点会和master定期进行通信，将自己的资源信息同步到master，master由此获知到整个集群的资源状况；

2.mater会和已注册、受信任的Framework进行交互，定期将最新的资源情况发送给Framework，当Framework前端有工作需求时，将选择接收资源，否则拒绝；

3.前端用户提交了一个工作需求给Framework；

4.Framework接收slave master发过来的资源信息；

5.Framework依据资源信息向slave发起任务启动命令，开始调度工作；



Slave是运行在物理或虚拟服务器上的Mesos守护进程，是Mesos集群的一部分。  
Framework由调度器（Scheduler）应用程序和任务执行器（Executor）组成，被注册到Mesos以使用Mesos集群中的资源；

    Slave 1向Master汇报其空闲资源：4个CPU、4GB内存。然后，Master触发分配策略模块，得到的反馈是Framework 1要请求全部可用资源；

    Master向Framework 1发送资源邀约，描述了Slave 1上的可用资源；

    Framework的调度器（Scheduler）响应Master，需要在Slave上运行两个任务，第一个任务分配<2 CPUs, 1 GB RAM>资源，第二个任务分配<1 CPUs, 2 GB RAM>资源；

    最后，Master向Slave下发任务，分配适当的资源给Framework的任务执行器（Executor）,接下来由执行器启动这两个任 务（如图中虚线框所示）。此时，还有1个CPU和1GB的RAM尚未分配，因此分配模块可以将这些资源供给Framework 2；

**2.zookeeper概述：**

ZooKeeper是用来给集群服务维护配置信息，提供分布式同步和提供组服务。所有这些类型的服务都使用某种形式的分布式应用程序。ZooKeeper是一个分布式的，开放源码的协调服务，是Hadoop和Hbase的重要组件，在此zookeeper与mesos结合实现为mesos的master保证其单点故障问题；

**ZooKeeper角色：**

领导者（leader）：领导者负责投票发起和决议，更新系统状态；

跟随者（follwoer）：follower用于接收客户请求并向客户端返回结果，在选主过程参与投票；

观察者：ObServer可以接受客户端连接，将写请求转发给leader节点，但ObServer不参加投票过程，只同步leader的状态，ObServer的目的是为了拓展系统，提高读取速度；

客户端：请求发起方；

**3.** **Marathon概述：**

是一个mesos框架，能够支持运行长服务，可以实现通过web页面或者api接口实现部署应用，创建docker的容器，比如web应用等。也是一种私有的Paas，实现服务的发现，为部署提供提供REST API服务，有授权和SSL、配置约束；

三、项目环境：

|  |  |  |  |
| --- | --- | --- | --- |
| 系统类型 | IP地址 | 主机名 | 所需软件 |
| Centos 7.4 1708 64bit | 192.168.100.101 | master | jdk-8u171-linux-x64.tar.gz marathon-0.15.2.tgz zookeeper-3.4.10.tar.gz  mesos |
| Centos 7.4 1708 64bit | 192.168.100.102 | master1 | jdk-8u171-linux-x64.tar.gz zookeeper-3.4.10.tar.gz  mesos |
| Centos 7.4 1708 64bit | 192.168.100.103 | master2 | jdk-8u171-linux-x64.tar.gz zookeeper-3.4.10.tar.gz  mesos |
| Centos 7.4 1708 64bit | 192.168.100.104 | slave1 | jdk-8u171-linux-x64.tar.gz  mesos  docker-engine |
| Centos 7.4 1708 64bit | 192.168.100.105 | slave2 | jdk-8u171-linux-x64.tar.gz  mesos  docker-engine |

四、项目实施步骤：

* 配置所有节点的域名解析（在此只显示master节点的配置）；
* 在所有节点进行安装mesos程序（在此只显示master节点的配置）；
* 在所有的master节点上进行安装zookeeper程序（在此只显示master节点的配置）；
* 配置master节点的zookeeper服务；
* 配置master1节点的zookeeper服务；
* 配置master2节点的zookeeper服务；
* 在所有的slave节点上进行安装docker程序（在此只显示slave1节点的配置）；
* 在所有的master节点启动zookeeper服务；
* 配置所有master节点的mesos服务（在此只显示master节点的配置）；
* 配置所有slave节点的mesos服务（在此只显示slave1节点的配置）；
* 访问mesos服务web页面，查看节点调度信息；
* 在master单一节点上安装部署marathon程序；
* 访问mesos的web页面查看marathon程序节点状态；
* 使用marathon的web页面提交测试任务；
* 访问mesos程序的web页面查看测试任务的提交情况；
* 查看slave节点的任务处理情况；
* 关闭marathon 上创建的测试应用；
* 使用Marathon API的形式添加docker容器nginx：
* **配置所有节点的域名解析（在此只显示master节点的配置）；**

[root@master ~]# cat <<END >>/etc/hosts

192.168.100.101 master

192.168.100.102 master1

192.168.100.103 master2

192.168.100.104 slave1

192.168.100.105 slave2

END

[root@master ~]# yum -y install ntpdate

[root@master ~]# /usr/sbin/ntpdate ntp1.aliyun.com

22 Sep 22:57:04 ntpdate[14520]: adjust time server 120.25.115.20 offset -0.001107 sec

[root@master ~]# echo "/usr/sbin/ntpdate ntp1.aliyun.com" >>/etc/rc.local

[root@master ~]# chmod +x /etc/rc.local

* **在所有节点进行安装mesos程序（在此只显示master节点的配置）；**

[root@master ~]# ls jdk-8u171-linux-x64.tar.gz

jdk-8u171-linux-x64.tar.gz

[root@master ~]# tar zxvf jdk-8u171-linux-x64.tar.gz

[root@master ~]# mv jdk1.8.0\_171/ /usr/local/java

[root@master ~]# ls /usr/local/java

bin db javafx-src.zip lib man release THIRDPARTYLICENSEREADME-JAVAFX.txt

COPYRIGHT include jre LICENSE README.html src.zip THIRDPARTYLICENSEREADME.txt

[root@master ~]# vi /etc/profile

export JAVA\_HOME=/usr/local/java

export PATH=$JAVA\_HOME/bin:$PATH

export CLASSPATH=$JAVA\_HOME/jre/lib/ext:$JAVA\_HOME/lib/tools.jar

:wq

[root@master ~]# source /etc/profile

[root@master ~]# java -version

java version "1.8.0\_171"

Java(TM) SE Runtime Environment (build 1.8.0\_171-b11)

Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)

[root@master ~]# wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

[root@master ~]# vi /etc/yum.repos.d/wandisco-svn.repo

[WANdiscoSVN]

name=WANdisco SVN Repo 1.9

enabled=1

baseurl=http://opensource.wandisco.com/centos/7/svn-1.9/RPMS/$basearch/

gpgcheck=1

gpgkey=http://opensource.wandisco.com/RPM-GPG-KEY-WANdisco

:wq

[root@master ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo

[root@master ~]# yum install -y apache-maven python-devel zlib-devel libcurl-devel openssl-devel cyrus-sasl-devel cyrus-sasl-md5 apr-devel apr-util-devel subversion-devel

[root@master ~]# rpm -ivh http://repos.mesosphere.com/el/7/noarch/RPMS/mesosphere-el-repo-7-1.noarch.rpm

[root@master ~]# yum -y install mesos

[root@master ~]# vi /etc/profile

export MESOS\_NATIVE\_JAVA\_LIBRARY=/usr/local/lib/libmesos.so

export MESOS\_NATIVE\_LIBRARY=/usr/local/lib/libmesos.so

:wq

[root@master ~]# source /etc/profile

* **在所有的master节点上进行安装zookeeper程序（在此只显示master节点的配置）；**

[root@master ~]# ls zookeeper-3.4.10.tar.gz

zookeeper-3.4.10.tar.gz

[root@master ~]# tar zxvf zookeeper-3.4.10.tar.gz

[root@master ~]# mv zookeeper-3.4.10 /usr/local/zookeeper

[root@master ~]# ls /usr/local/zookeeper

bin dist-maven lib README.txt zookeeper-3.4.10.jar.asc

build.xml docs LICENSE.txt recipes zookeeper-3.4.10.jar.md5

conf ivysettings.xml NOTICE.txt src zookeeper-3.4.10.jar.sha1

contrib ivy.xml README\_packaging.txt zookeeper-3.4.10.jar

[root@master ~]# vi /etc/profile

export ZOOKEEPER\_HOME=/usr/local/zookeeper

export PATH=$PATH:$ZOOKEEPER\_HOME/bin

[root@master ~]# source /etc/profile

* **配置master节点的zookeeper服务；**

[root@master ~]# mkdir /usr/local/zookeeper/data

[root@master ~]# mv /usr/local/zookeeper/conf/zoo\_sample.cfg /usr/local/zookeeper/conf/zoo.cfg

[root@master ~]# vi /usr/local/zookeeper/conf/zoo.cfg

12 dataDir=/usr/local/zookeeper/data

29 server.1=master:2888:3888

30 server.2=master1:2888:3888

31 server.3=master2:2888:3888

:wq

[root@master ~]# echo 1 >/usr/local/zookeeper/data/myid

* **配置master1节点的zookper服务；**

[root@master1 ~]# mkdir /usr/local/zookeeper/data

[root@master1 ~]# mv /usr/local/zookeeper/conf/zoo\_sample.cfg /usr/local/zookeeper/conf/zoo.cfg

[root@master1 ~]# vi /usr/local/zookeeper/conf/zoo.cfg

12 dataDir=/usr/local/zookeeper/data

29 server.1=master:2888:3888

30 server.2=master1:2888:3888

31 server.3=master2:2888:3888

:wq

[root@master1 ~]# echo 2 >/usr/local/zookeeper/data/myid

* **配置master2节点的zookeeper服务；**

[root@master2 ~]# mkdir /usr/local/zookeeper/data

[root@master2 ~]# mv /usr/local/zookeeper/conf/zoo\_sample.cfg /usr/local/zookeeper/conf/zoo.cfg

[root@master2 ~]# vi /usr/local/zookeeper/conf/zoo.cfg

12 dataDir=/usr/local/zookeeper/data

29 server.1=master:2888:3888

30 server.2=master1:2888:3888

31 server.3=master2:2888:3888

:wq

[root@master2 ~]# echo 3 >/usr/local/zookeeper/data/myid

* **在所有的slave节点上进行安装docker程序（在此只显示slave1节点的配置）；**

[root@slave1 ~]# cat <<END >>/etc/yum.repos.d/centos.repo

[local2]

name=docker

baseurl=ftp://192.168.100.100/docker

enable=1

gpgcheck=0

END

[root@slave1 ~]# yum clean all

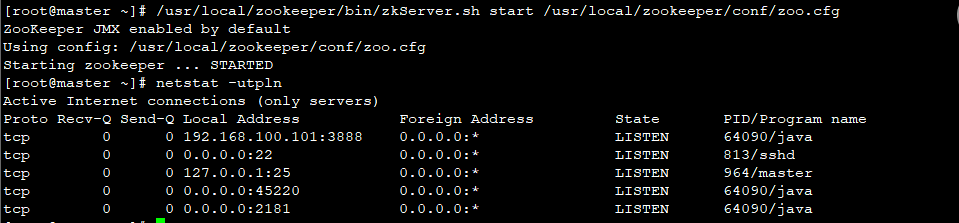
[root@slave1 ~]# yum -y install docker-engine ##网络卡顿多安装几次

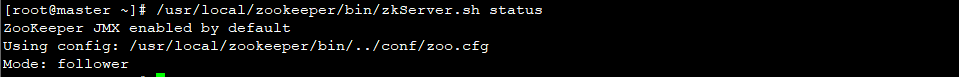
[root@slave1 ~]# systemctl start docker

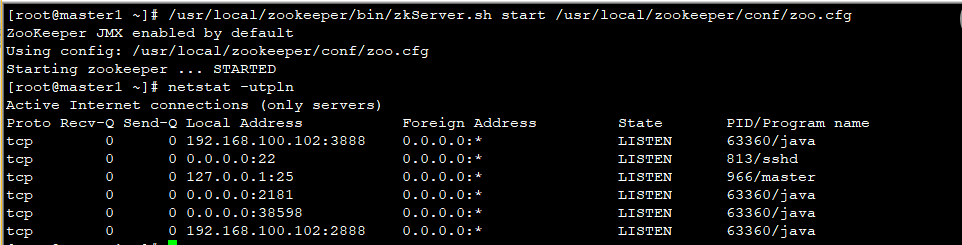
[root@slave1 ~]# systemctl enable docker

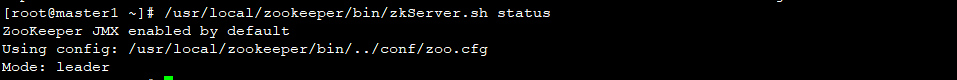
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

* **在所有的master节点启动zookeeper服务；**

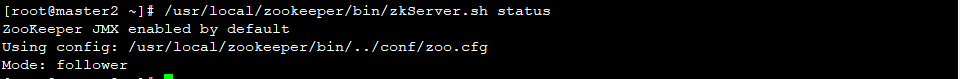












* **配置所有master节点的mesos服务（在此只显示master1节点的配置）；**

[root@master1 ~]# cat <<END >/etc/mesos/zk

zk://192.168.100.101:2181,192.168.100.102:2181,192.168.100.103:2181/mesos

END ##在mesos配置文件中指定zookeeper的节点

[root@master1 ~]# cat <<END >/etc/mesos-master/quorum

2

END ##指定节点数量

[root@master1 ~]# cat <<END >/etc/mesos-master/ip

192.168.100.102

END ##指定master节点的ip地址

[root@master1 ~]# cat <<END >/etc/mesos-master/hostname

192.168.100.102

END ##指定master节点的ip地址，避免使用dns解析

[root@master1 ~]# systemctl start mesos-master

[root@master1 ~]# systemctl enable mesos-master

[root@master ~]# netstat -utpln |grep mesos

tcp 0 0 192.168.100.102:5050 0.0.0.0:\* LISTEN 1613/mesos-master

* **配置所有slave节点的mesos服务（在此只显示slave1节点的配置）；**

[root@slave1 ~]# cat <<END >/etc/mesos/zk

zk://192.168.100.101:2181,192.168.100.102:2181,192.168.100.103:2181/mesos

END ##指定mesos集群的master节点的位置

[root@slave1 ~]# echo "192.168.100.104" >/etc/mesos-slave/ip

[root@slave1 ~]# echo "192.168.100.104" >/etc/mesos-slave/hostname

[root@slave1 ~]# echo "docker,mesos" >/etc/mesos-slave/containerizers ##注意：允许mesos在docker节点上进行创建容器

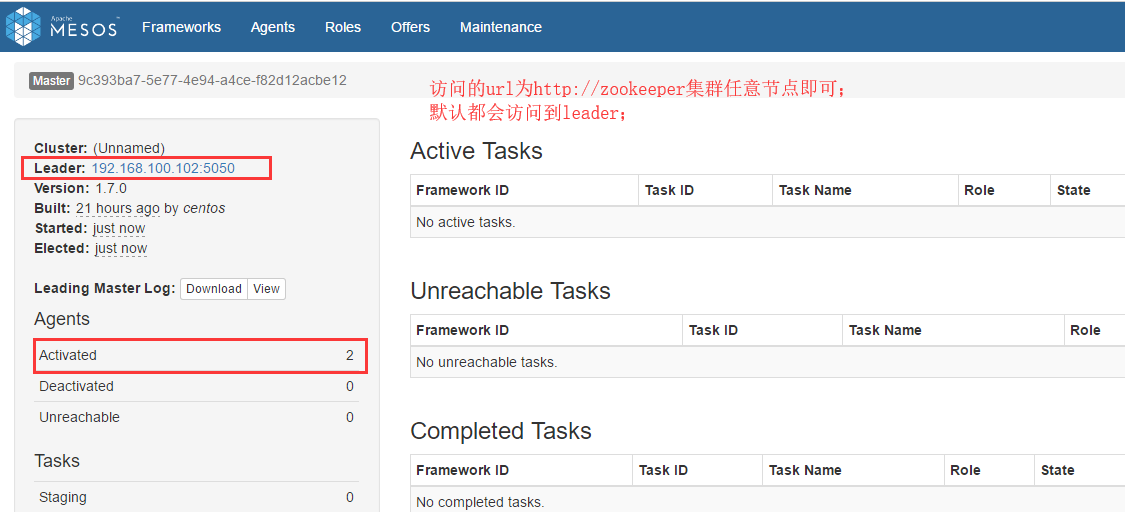
[root@slave1 ~]# systemctl start mesos-slave

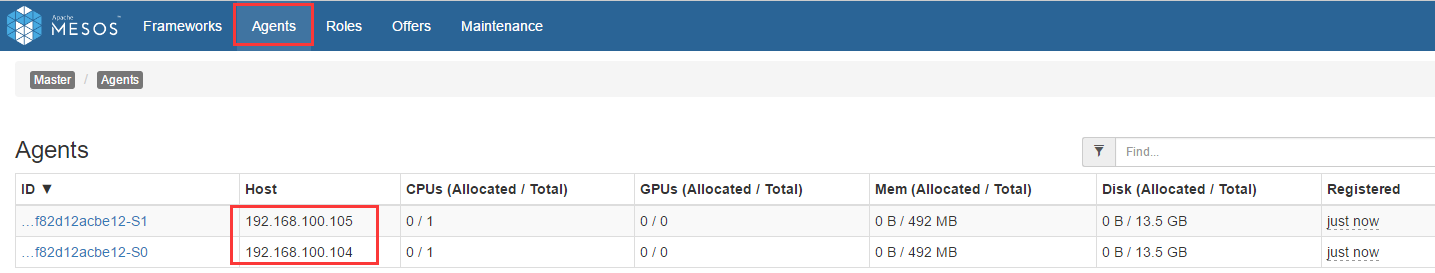
[root@slave1 ~]# systemctl enable mesos-slave

[root@slave1 ~]# netstat -utpln |grep mesos

tcp 0 0 192.168.100.104:5051 0.0.0.0:\* LISTEN 1835/mesos-slave

* **访问mesos服务web页面，查看节点调度信息；**





* **在master单一节点上安装部署marathon程序；**

[root@master ~]# ls marathon-0.15.2.tgz

marathon-0.15.2.tgz

[root@master ~]# tar zxvf marathon-0.15.2.tgz

[root@master ~]# mv marathon-0.15.2 /usr/local/marathon

[root@master ~]# ls /usr/local/marathon

bin Dockerfile docs examples LICENSE README.md target

[root@master ~]# /usr/local/marathon/bin/start --hostname 192.168.100.101 --master zk://192.168.100.101:2181,192.168.100.102:2181,192.168.100.103:2181/mesos --http\_address 0.0.0.0 &>/dev/null &

[1] 2687

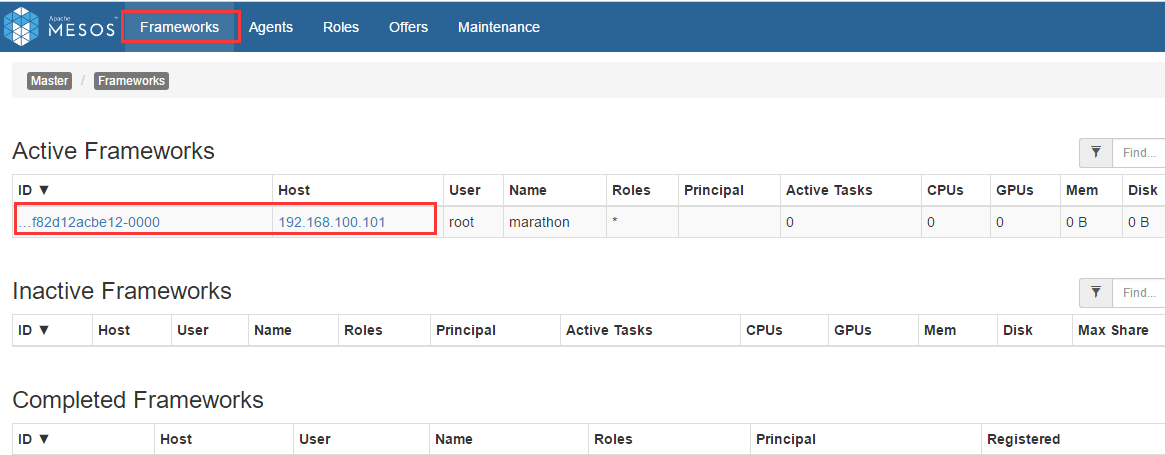
[root@master ~]# jobs -l

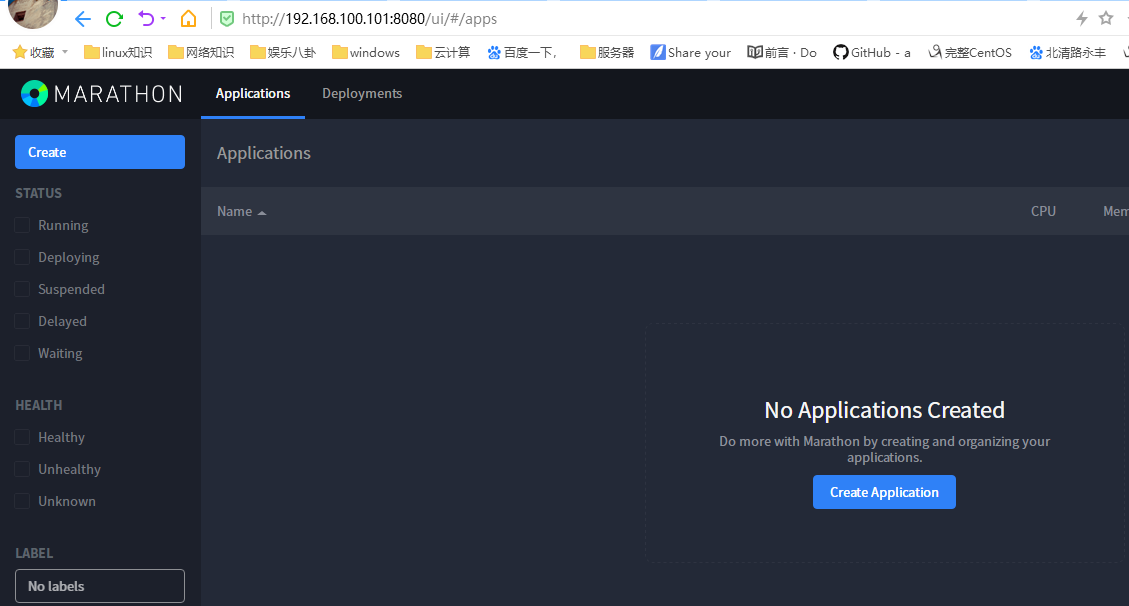
[1]+ 2687 运行中 /usr/local/marathon/bin/start --hostname 192.168.100.101 --master zk://192.168.100.101:2181,192.168.100.102:2181,192.168.100.103:2181/mesos --http\_address 0.0.0.0 &>/dev/null &

[root@master ~]# netstat -utpln |grep 8080

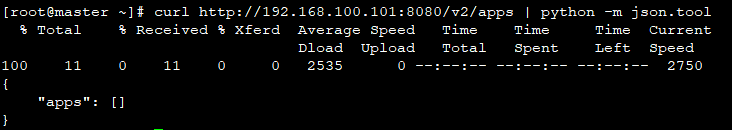
tcp 0 0 0.0.0.0:8080 0.0.0.0:\* LISTEN 2687/java

* **访问mesos的web页面查看marathon程序节点状态；**

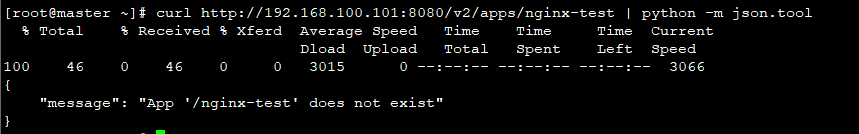




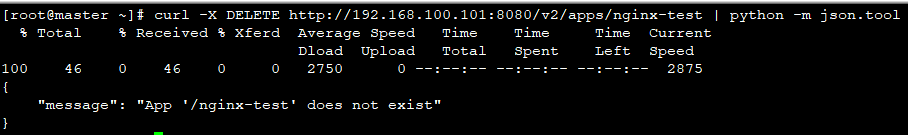
查看正在运行的应用：



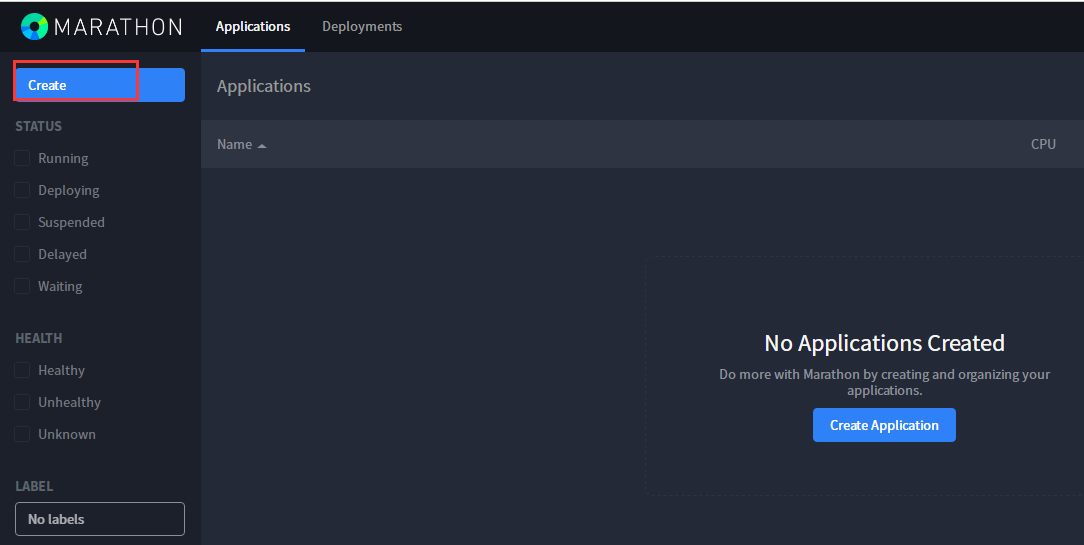
查看名称为nginx-test的应用：

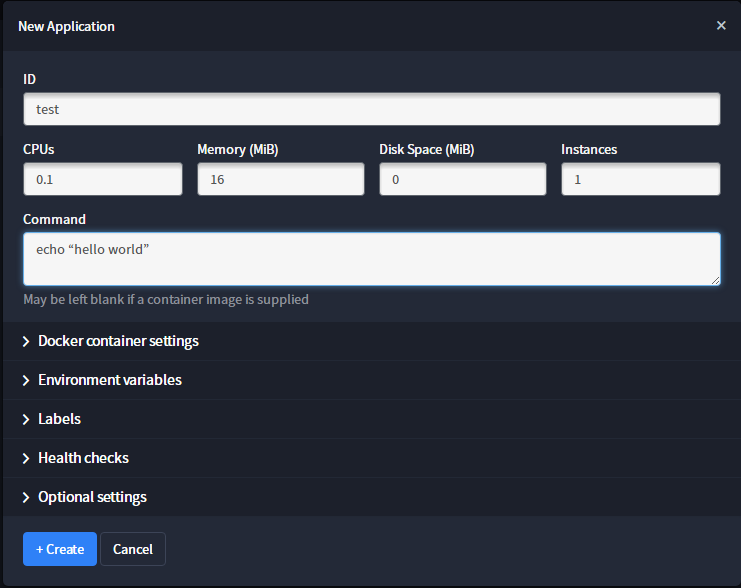


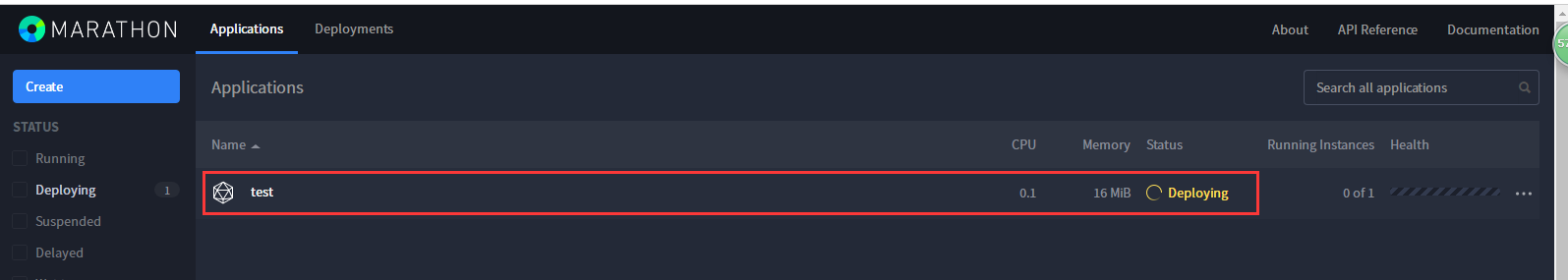
删除名称为nginx-test的应用：



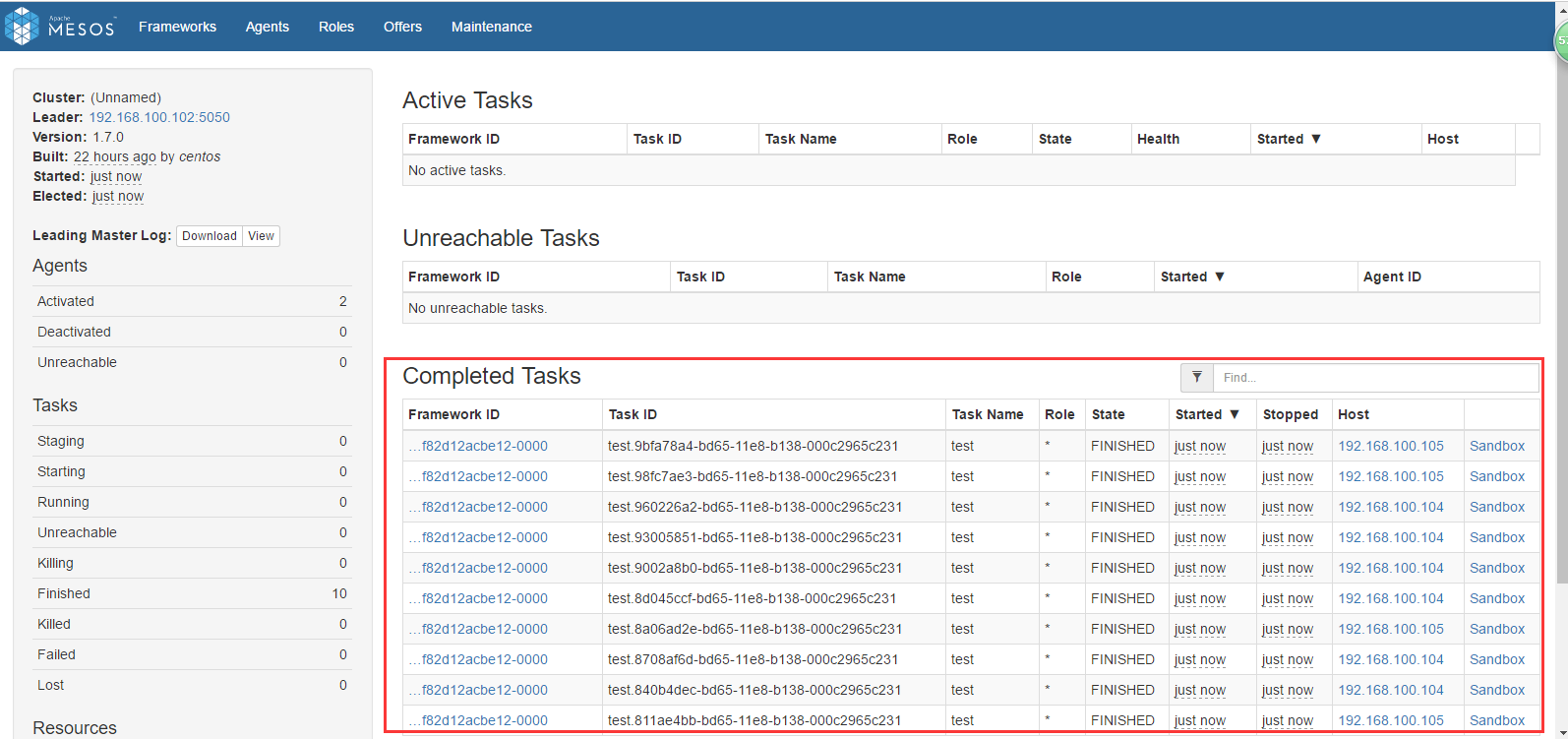
* **使用marathon的web页面提交测试任务；**

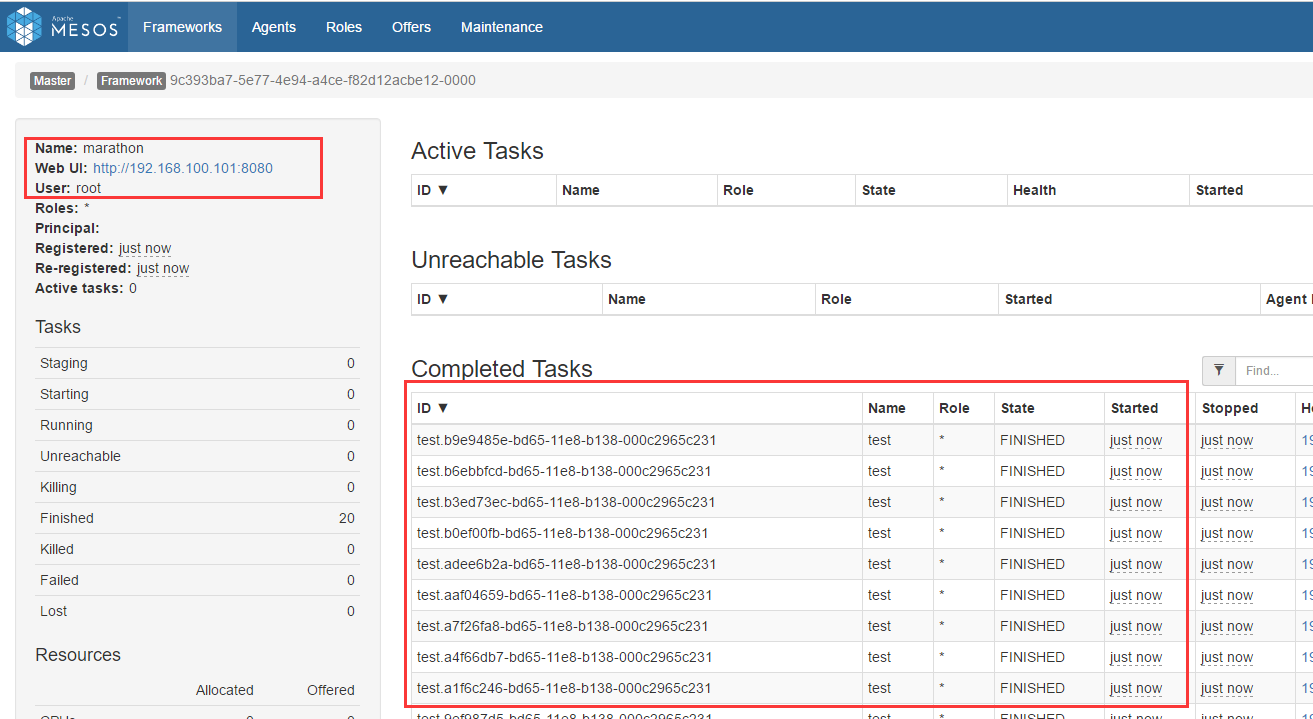


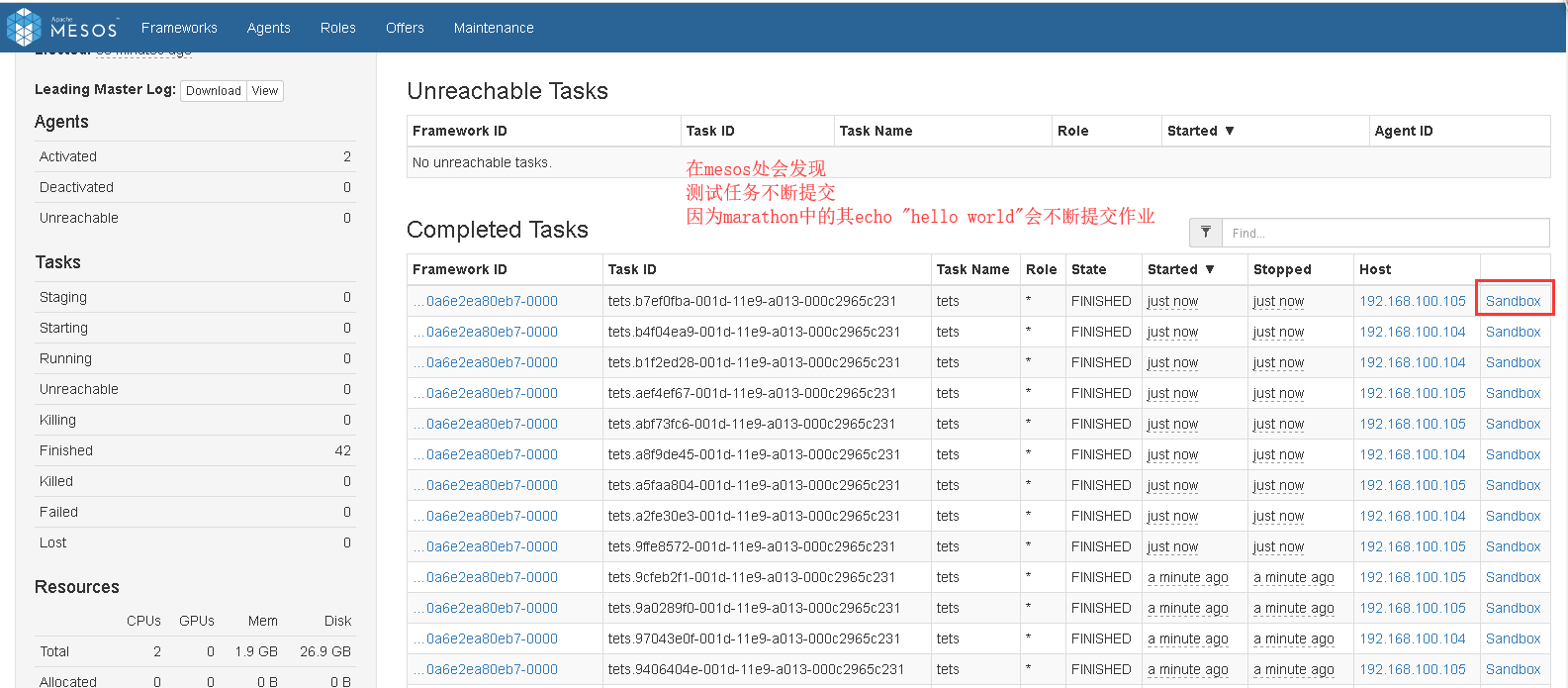


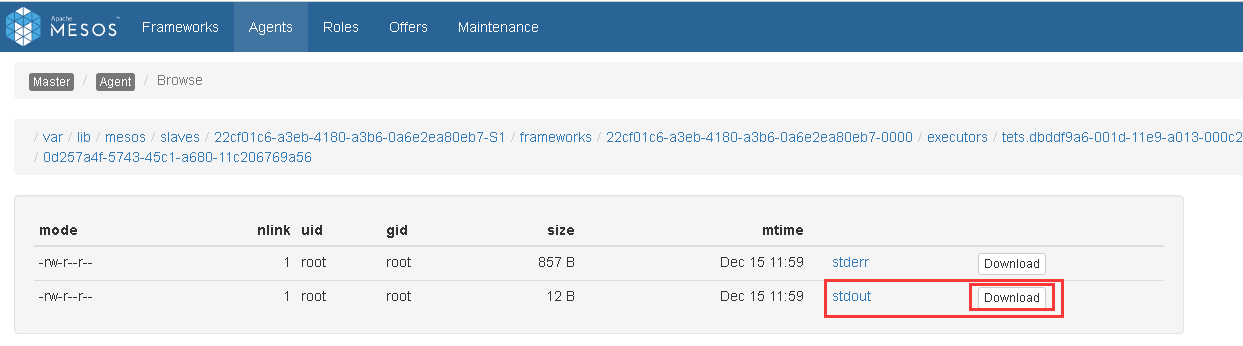


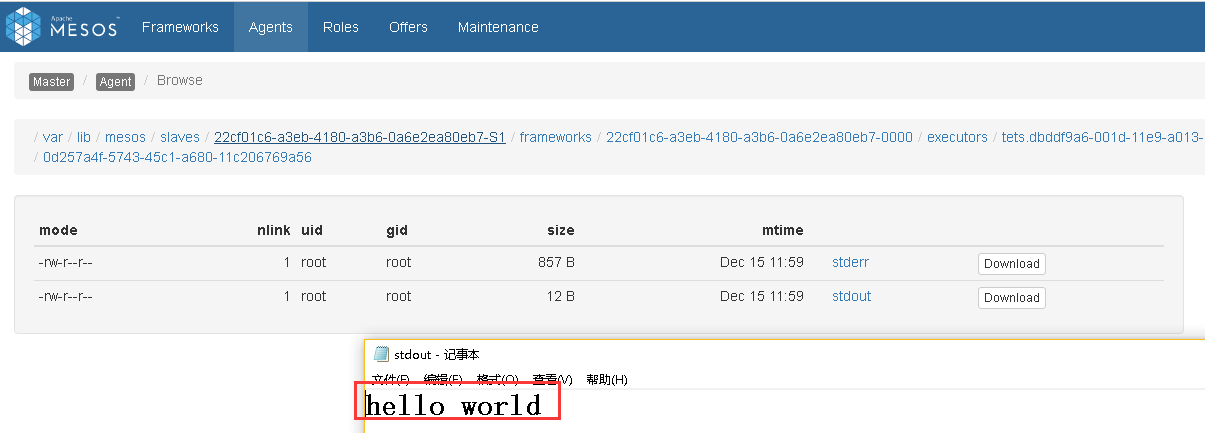
* **访问mesos程序的web页面查看测试任务的提交情况；**

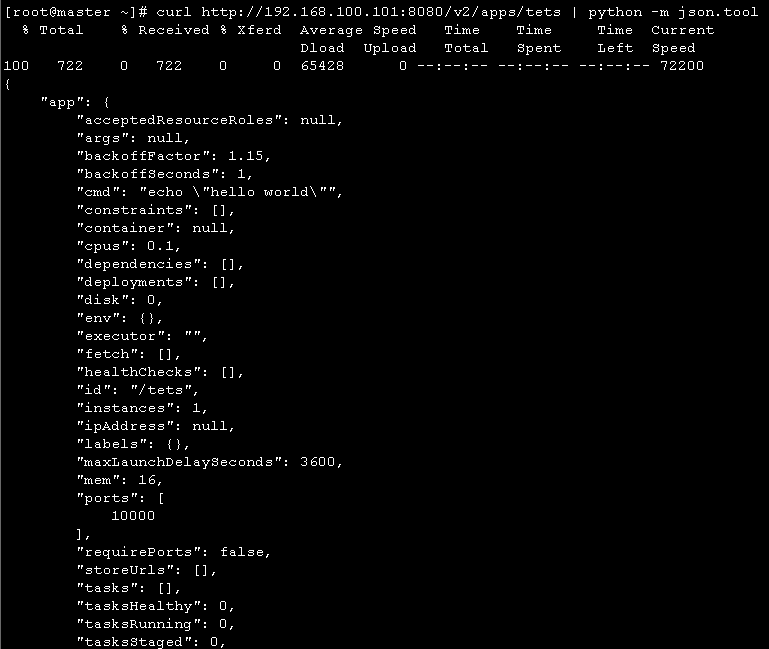




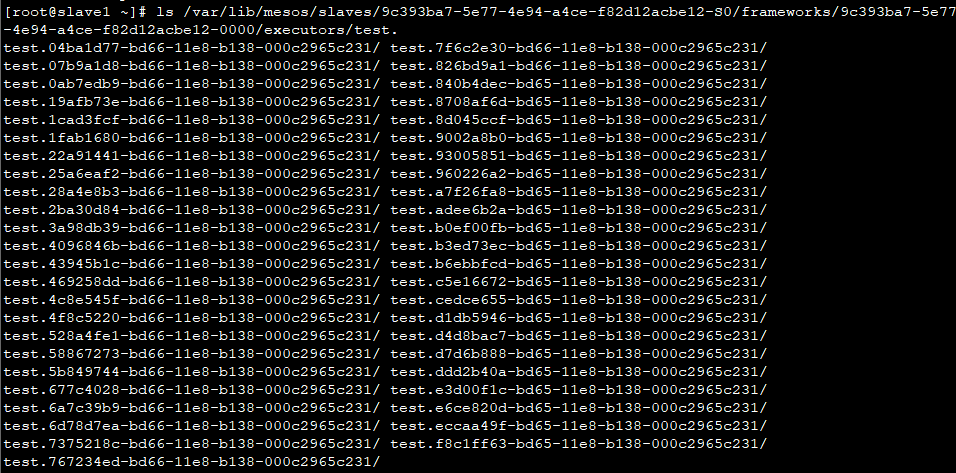




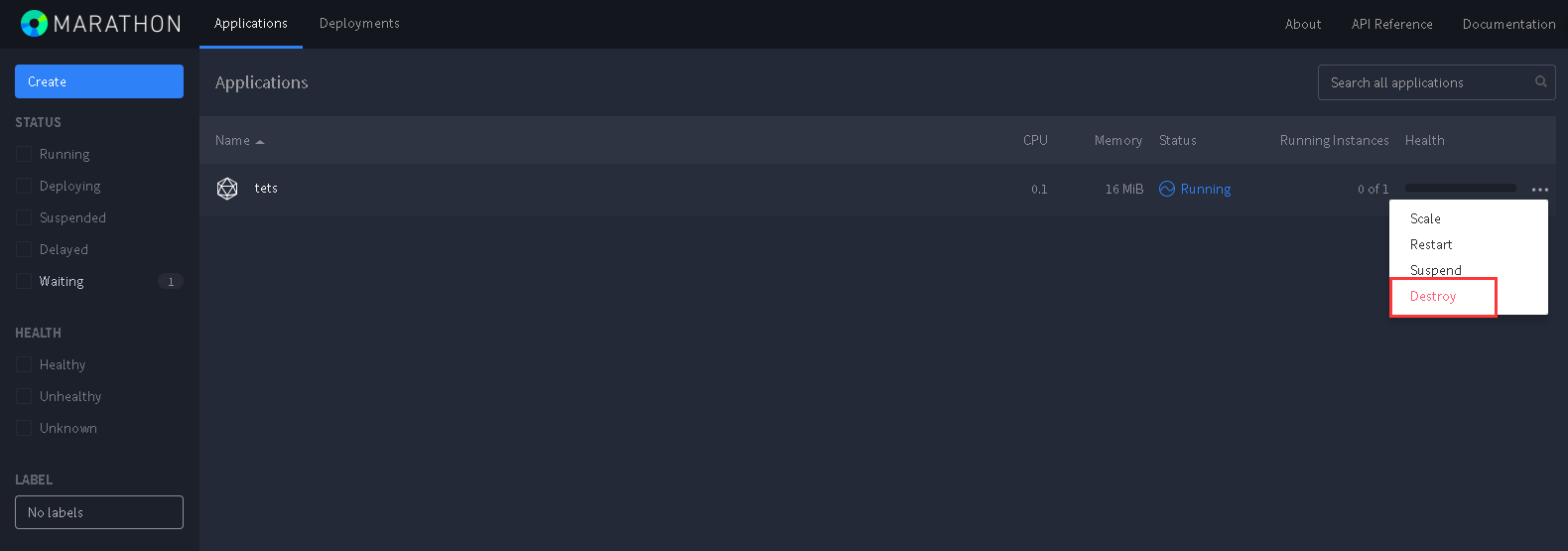


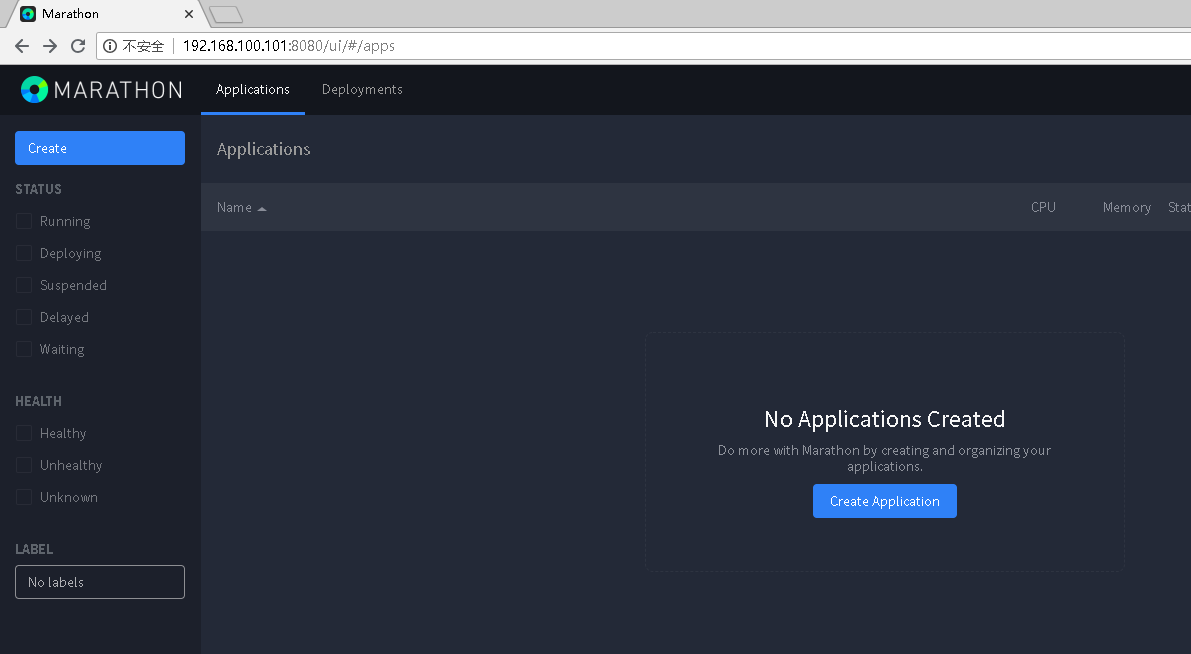


* **查看slave节点的任务处理情况；**

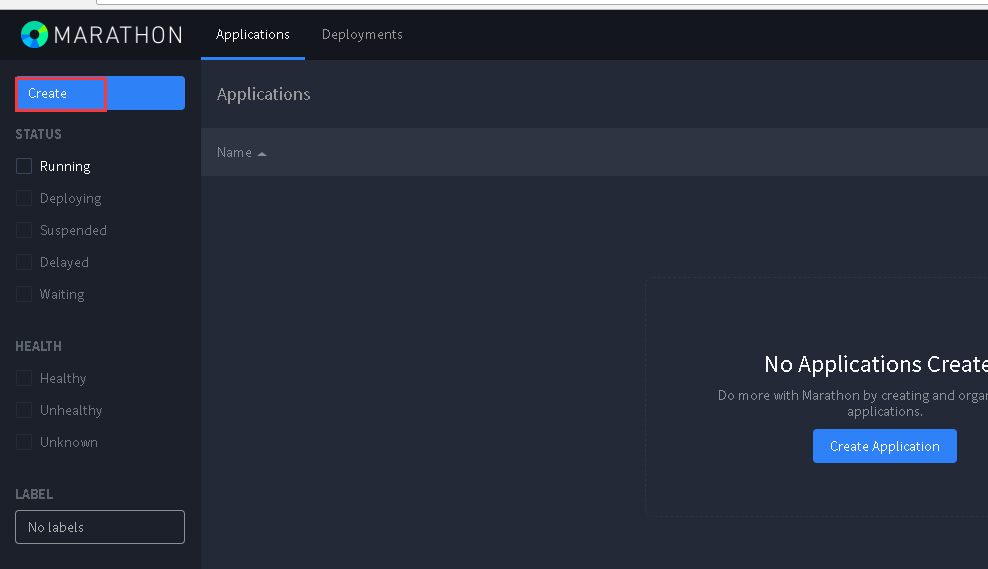


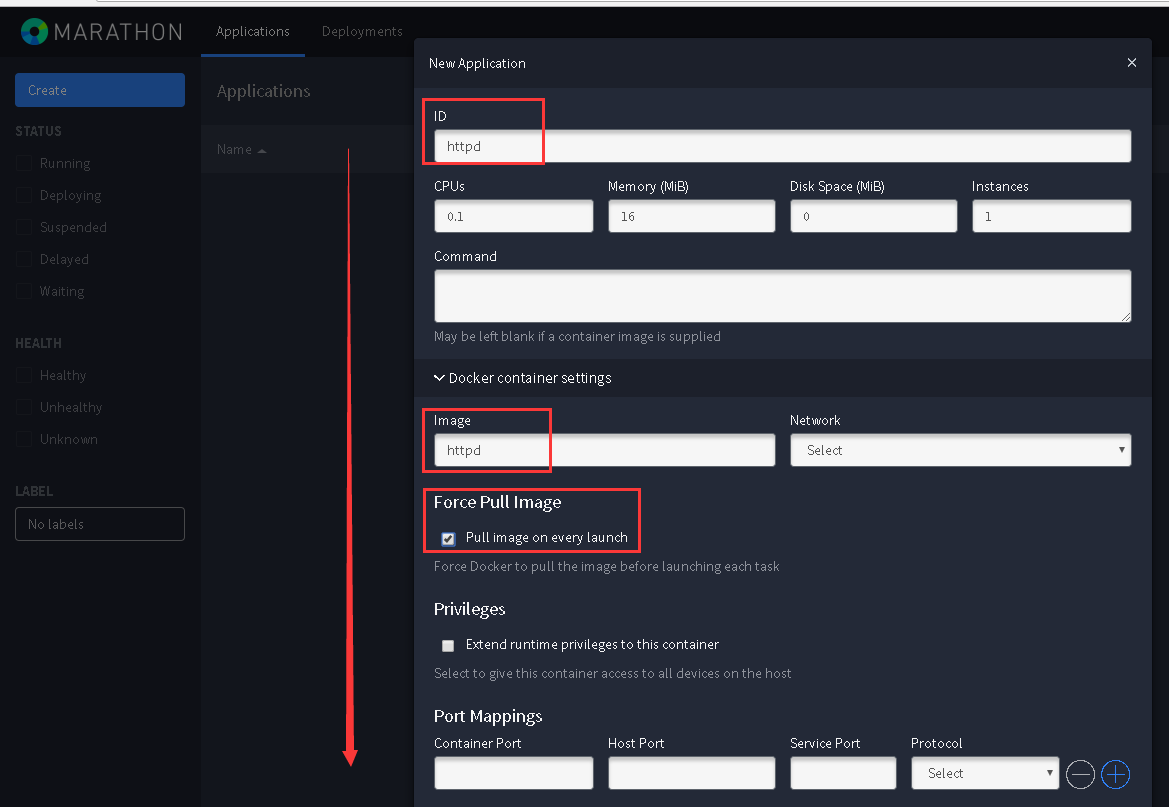
* **关闭marathon 上创建的测试应用；**

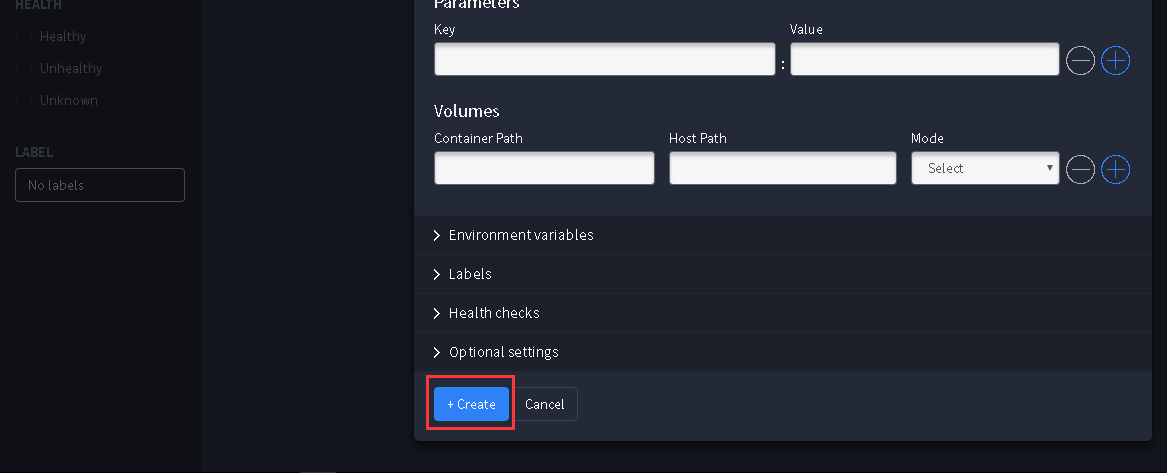


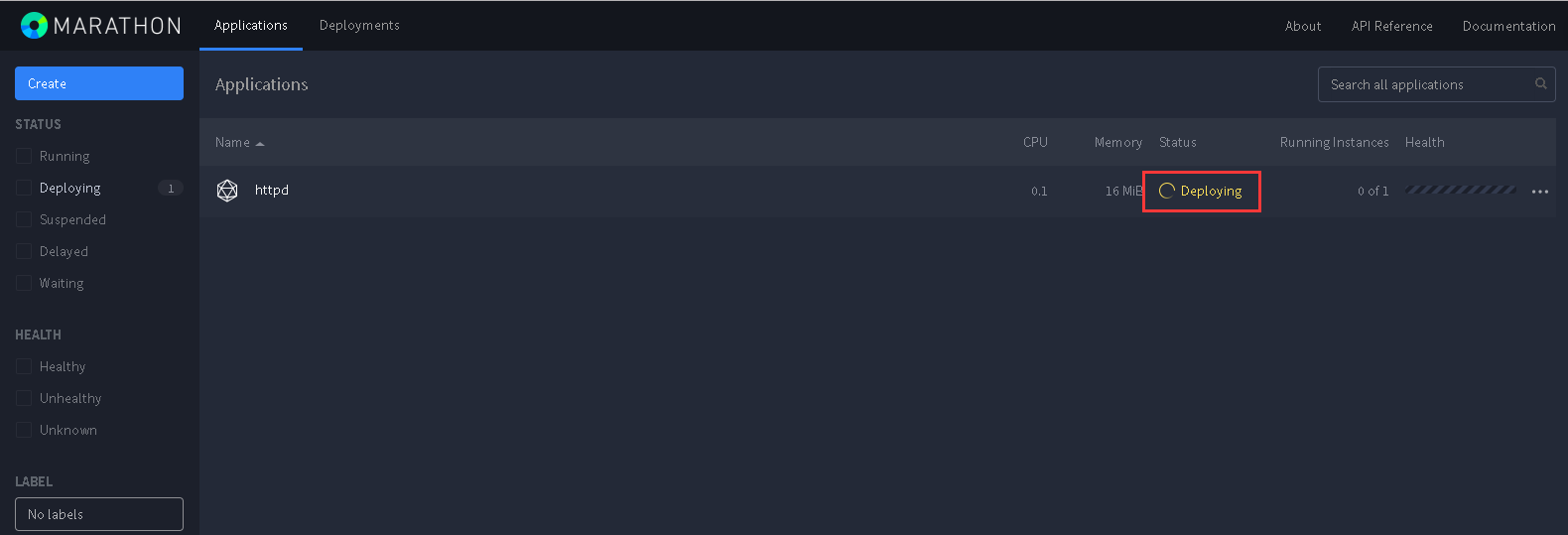


* **使用marathon的web页面提交httpd容器的创建任务；**

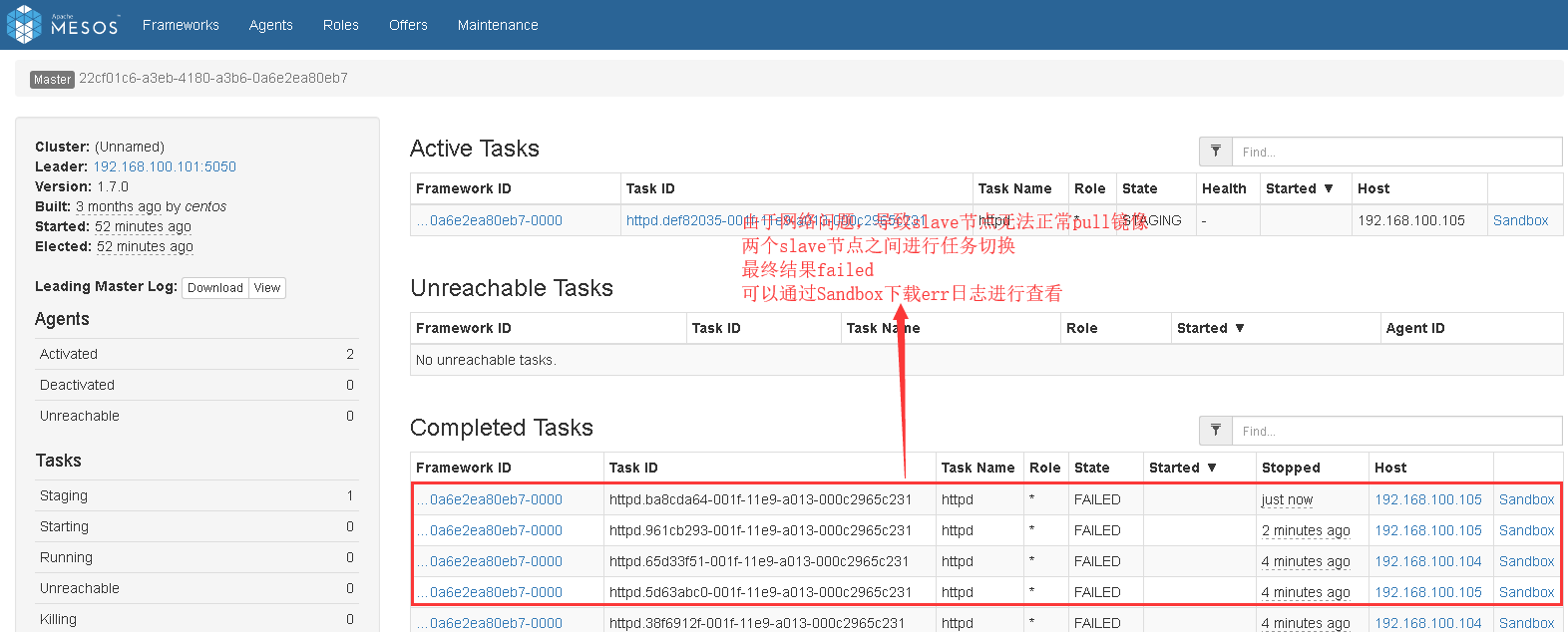




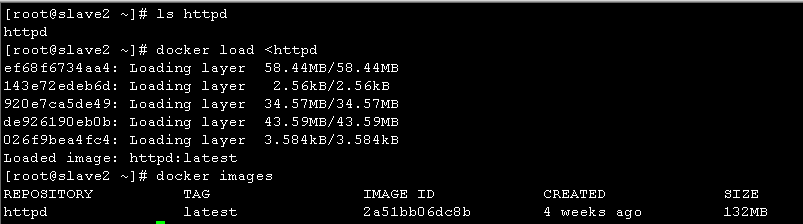


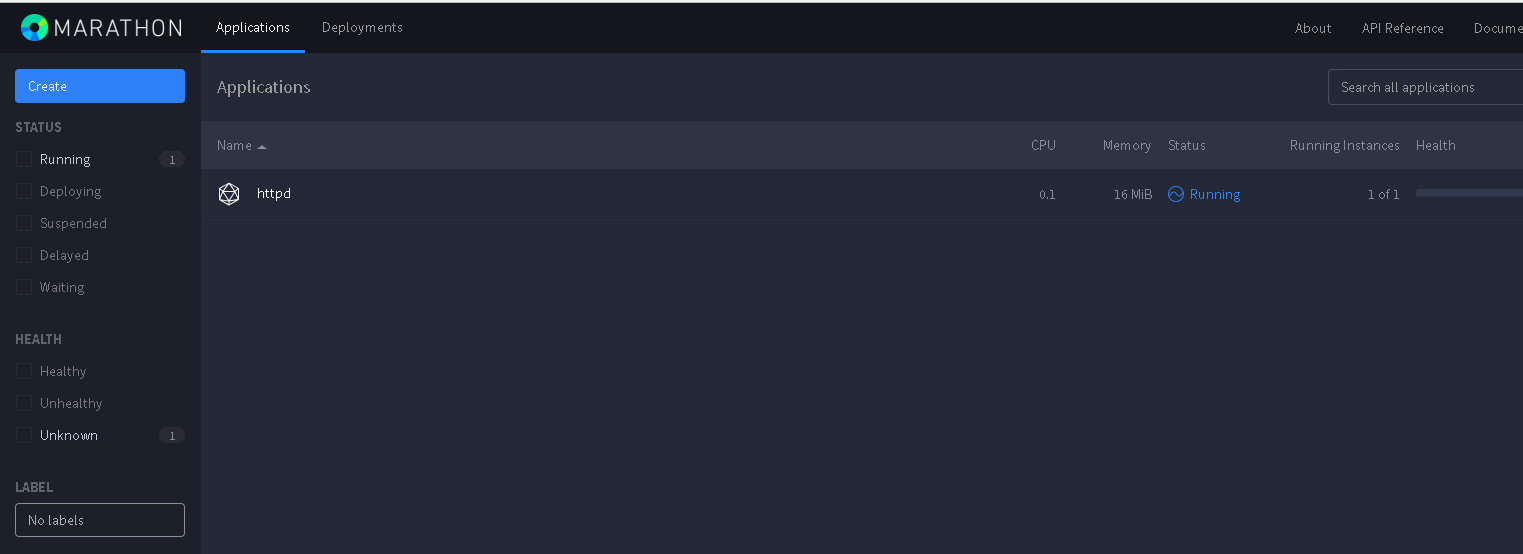


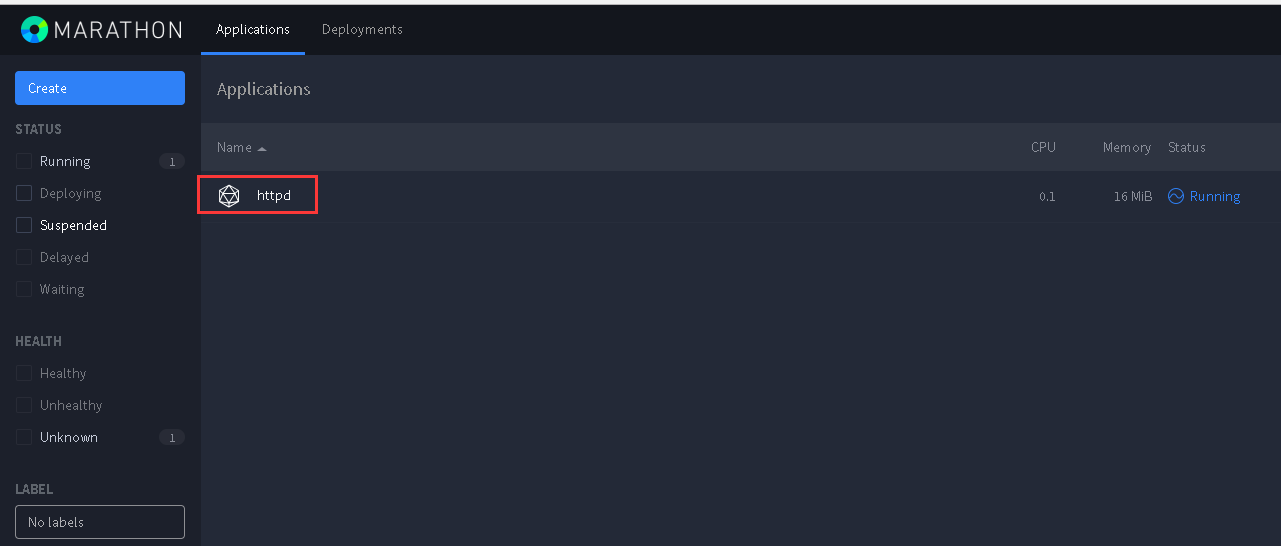
* **访问mesos程序的web页面查看httpd容器任务的提交情况；**

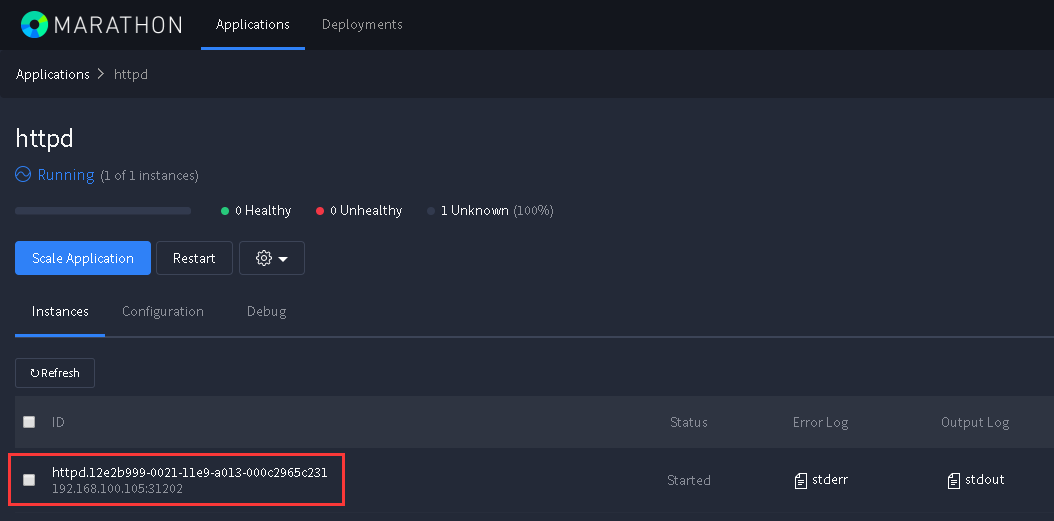


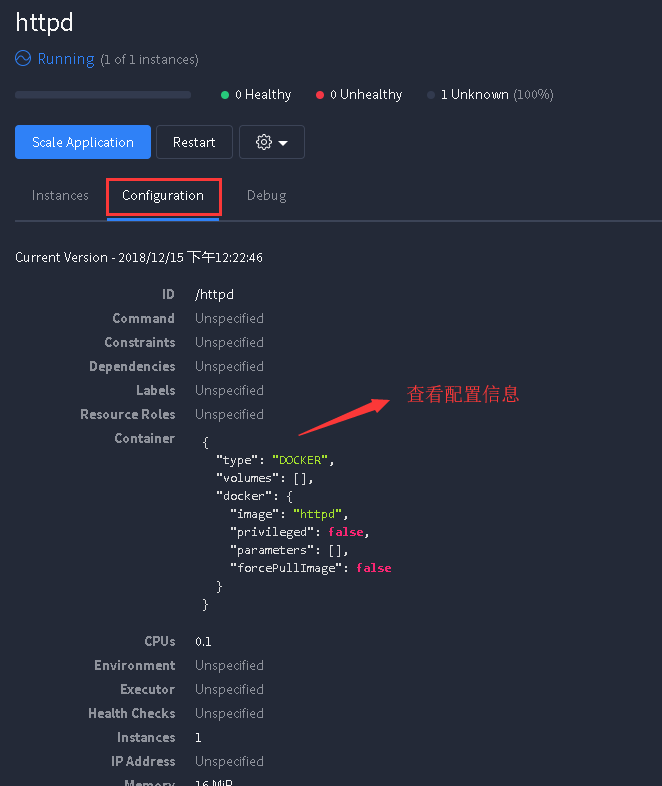


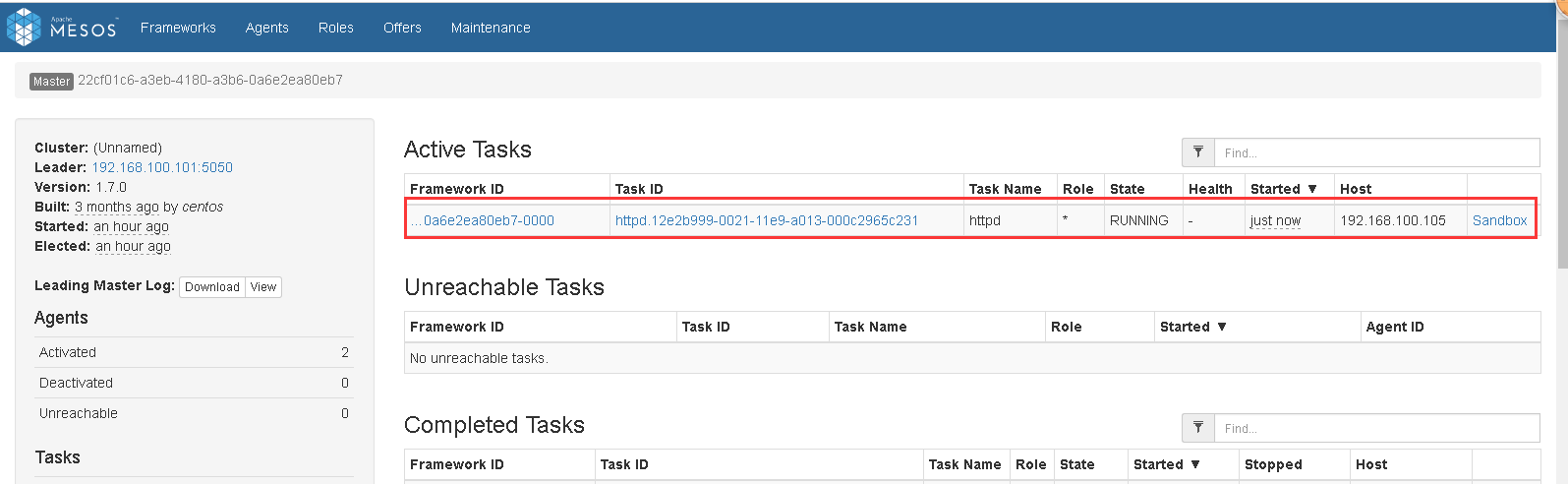






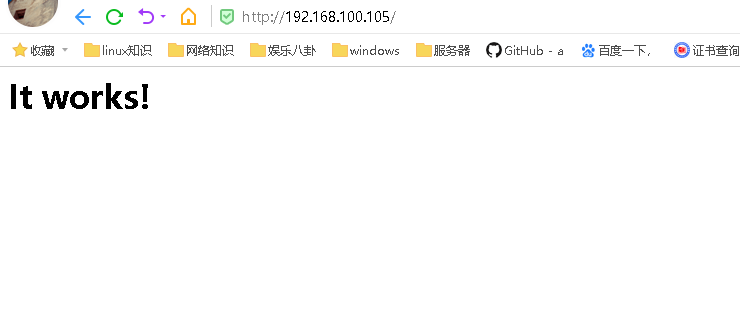












* **使用Marathon API的形式添加docker容器nginx：**

[root@slave1 ~]# vi nginx.json

{

"id":"nginx",

"cpus":0.1,

"mem":10,

"instances": 1,

"constraints":[["hostname", "UNIQUE",""]],

"container": {

"type":"DOCKER",

"docker": {

"image": "nginx",

"network": "HOST",

"portMappings": [

{"containerPort": 80,"hostPort": 0,"servicePort": 0, "protocol":"tcp" }

]

}

}

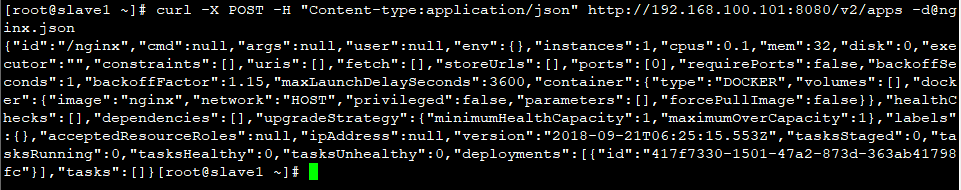
}

:wq

注意：json文件中的每一行内容的开头和结尾一定不能有空格，但可以有空行，否则手动通过curl的方式创建容器实例，会有各种各样的报错！！！

注意：在节点机器上手动创建的docker容器，这些容器信息是不会在marathon和mesos里展示的。这个nginx容器是在slave1节点机上创建的（注意：如果slave1节点机宕机或docker服务重启，那么这个nginx容器就会自动漂移到其他的slave节点机上；另外，通过上面方式创建好的容器，在单个slave节点机上删除后，容器也会自动转移到其他slave节点机器上，这样就实现了在slave节点出现故障时容器自动转移的高可用功能）。

[root@slave1 ~]# curl -X POST -H "Content-type:application/json" http://192.168.100.101:8080/v2/apps -d@nginx.json



* **访问marathon页面和mesos页面发现nginx任务正在处理，访问正在处理的slave节点的进程，发现nginx镜像正在下载；**

