Docker 容器部署Tomcat服务器

# 1 Download

<https://oomake.com/download/docker-windows>

$wget <https://get.docker.com/rpm/1.7.1/centos-6/RPMS/x86_64/docker-engine-1.7.1-1.el6.x86_64.rpm>

yum install \*.rpm

|  |  |  |
| --- | --- | --- |
| 作者 | 联系方式 |  |
| 姜鹏 |  |  |
|  |  |  |

# 2 官方安装教程（Docker CE 社区版本）

https://docs.docker.com/install/linux/docker-ce/centos/#uninstall-old-versions

## 2.1先决条件

CentOS 7 + centos-extras（enabled by default）

overlay2 storage driver is recommended.

## 2.2 卸载旧版本

$ sudo yum remove docker \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

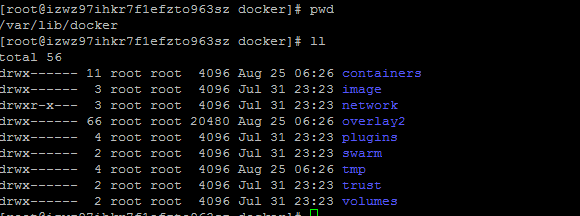
docker-latest-logrotate \

docker-logrotate \

docker-selinux \

docker-engine-selinux \

docker-engine



如果出现“none of these packages are installed.”证明安装成功。

## 2.3 Docker install

Docker CE安装三种选择：

* 1安装Docker资源库来安装，方便安装和升级。官方推荐。
* 2，手动下载RPM二进制包安装，适合无网络安装。
* 3，测试和开发环境选择自动化脚本安装Docker。

下面使用官方推荐的资源库来安装方式。

* 安装资源库

yum-utils：提供yum-config-manager 工具箱。

device-mapper-persistent-data 和lvm2 对于devicemapper storage driver 是必须的。

$ sudo yum install -y yum-utils \

device-mapper-persistent-data \

lvm2

* 安装稳定的资源库

使用如下命令安装稳定的资源库，即使你只想从edge or test 资源库中安装builds你也需要稳定的资源库。

$ sudo yum-config-manager \

--add-repo \

<https://download.docker.com/linux/centos/docker-ce.repo>

* 可选项: Enable edge and test 资源库。

docker.repo 文件中包含Enable edge and test 资源库，但是默认是 disabled，你可以使用如下命令连同稳定资源库一起都启用。

$ sudo yum-config-manager --enable docker-ce-edge

$ sudo yum-config-manager --enable docker-ce-test

You can disable the edge or test repository by running the yum-config-manager command with the --disable flag.

To re-enable it, use the --enable flag. The following command disables the edge repository.

$ sudo yum-config-manager --disable docker-ce-edge

* 安装DOCKER CE

$ sudo yum install docker-ce

安装成功后，Docker 还没有启动，docker group 已经创建了, 还没有加入用户。

可以通过 repo文件查看最高、最低、移除的版本。

$yum list docker-ce --showduplicates | sort -r

安装指定版本的DOCKER CE

$sudo yum install docker-ce-<VERSION STRING>

如：docker-ce-18.03.0.ce

* 启动 停止Docker

$ sudo systemctl start docker

$sudo systemctl stop docker

$service docker status

* 使用脚本安装

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sudo sh get-docker.sh

* 验证Docker CE 安装成功

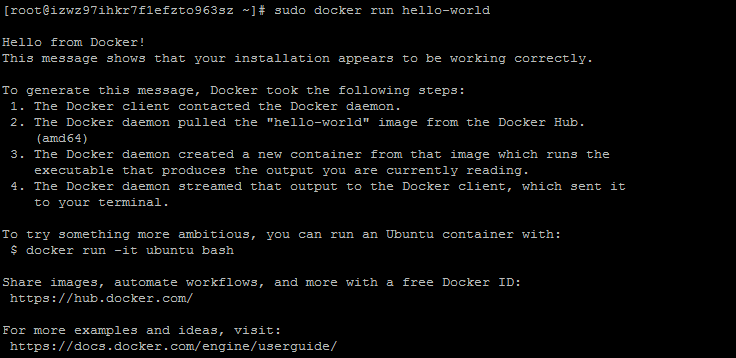
运行下载到本地的docker 镜像。

如果不存在，先下载到本地镜像。

$ sudo docker pull hello-world

docker有一个公共的img仓库，执行docker pull默认会在这个公共库里去下载对应的img，类似maven一样，maven的respository里有很多jar包。

$ sudo docker run hello-world



## 2.4移除Docker CE

卸载 Docker 安装包

$ sudo yum remove docker-ce

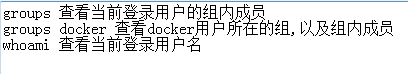
移除Images, containers, volumes, or 配置文件 。默认没有移除。

$ sudo rm -rf /var/lib/docker

## 2.5 升级Docker

$yum -y upgrade instead of yum -y install

* 为Docker CE 新增用户



/etc/group 的内容包括用户组（Group）、用户组口令、GID及该用户组所包含的用户（User），每个用户组一条记录；

格式如下：

group\_name:passwd:GID:user\_list

在/etc/group 中的每条记录分四个字段：

第一字段：用户组名称；

第二字段：用户组密码；

第三字段：GID

第四字段：用户列表，每个用户之间用,号分割；本字段可以为空；如果字段为空表示用户组为GID的用户名；

* 新建一个用户组docker , 先查看是否存在docker组

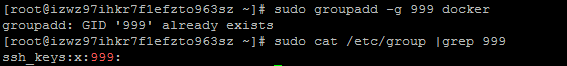
$sudo cat /etc/group | grep docker



新增一个docker组, 999作为GID

$sudo groupadd -g 999 docker

提示999 GID已经存在，换成1000





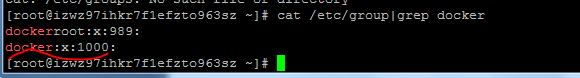
新增一个docker 用户到docker组：

$useradd docker -g docker

$passwd docker

查看docker组下面成员情况：





顺便提供官方有用的网址：

Share images, automate workflows, and more with a free Docker ID:

https://hub.docker.com/

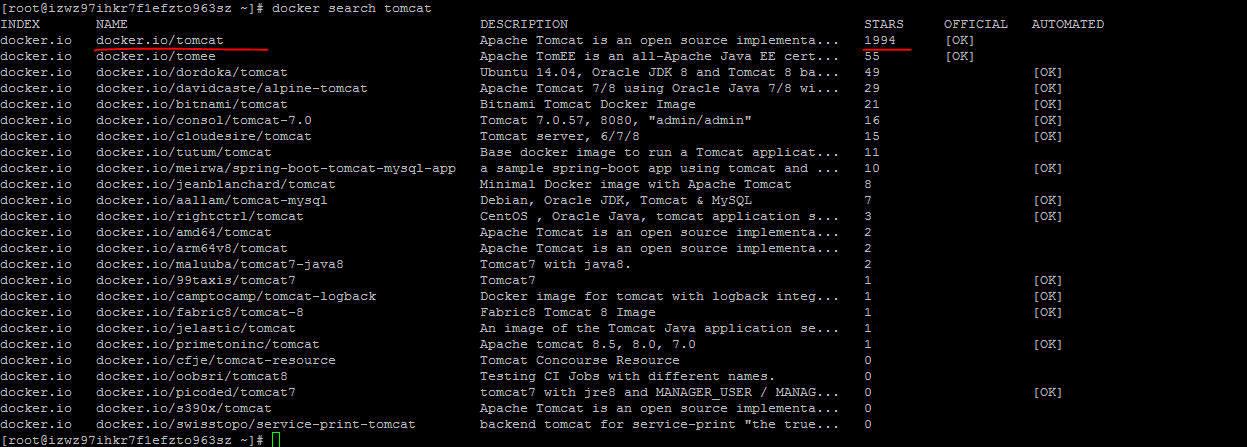
For more examples and ideas, visit:

<https://docs.docker.com/engine/userguide/>

# 3 Docker 部署Tomcat及Web 应用

* 查找服务器的Tomcat信息

$docker search tomcat



* 下载官方镜像starts最高的Tomcat

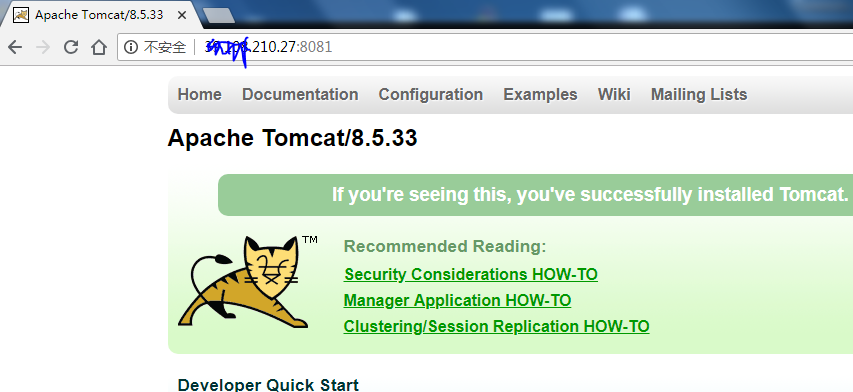
$docker pull docker.io/tomcat

* 启动Tomcat

$docker run -p 8081:8080 docker.io/tomcat

$docker run --name tomcat-1 --restart always -d -p 8081:8080 docker.io/tomcat

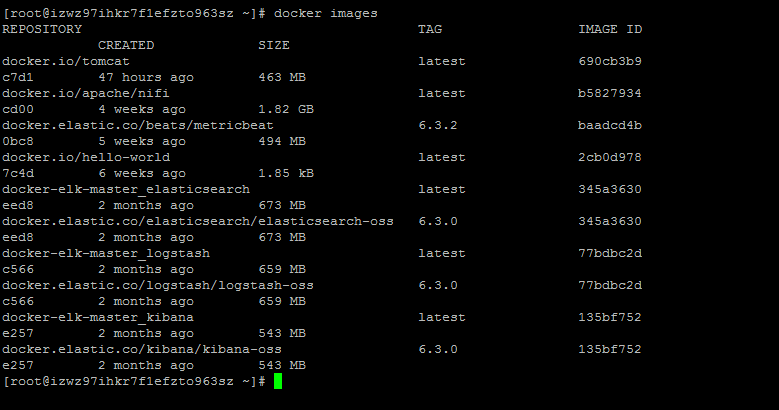
若端口被占用，可以指定容器和主机的映射端口 前者是外围访问端口：后者是容器内部端口



外网访问，Tomcat的猫猫出来咯。

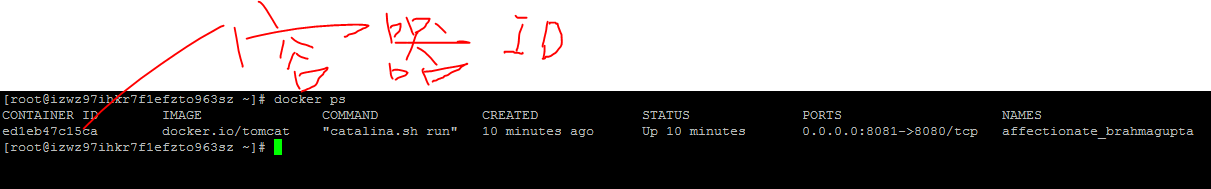
查看所有的镜像

$docker images



查看正在运行的容器

$docker ps



找到tomcat的实际安装路径

$docker exec -it ed1eb47c15ca /bin/bash



# 4 使用Docker镜像

## 4.1 docker pull NAME: TAG

如果不显示指定TAG，则默认会选择latest 标签

为避免镜像重名的情况，需要添加镜像的仓库地址：

$docker pull registry.hub.docker.com/ubuntu:14.04

等同于：

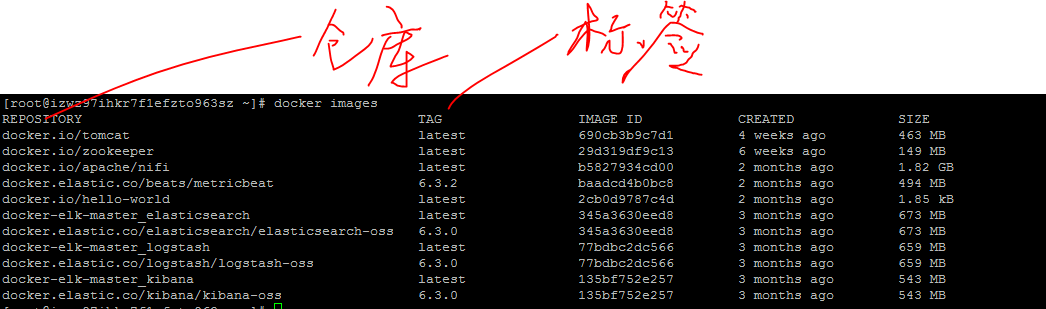
$docker pull ubuntu:14.04

下载镜像到本地，就可以使用该镜像了。如：利用ubuntu镜像创建一个容器，在其中运行bash命令

$docker run -it ubuntu:14.04 bash

## 4.2 查看镜像

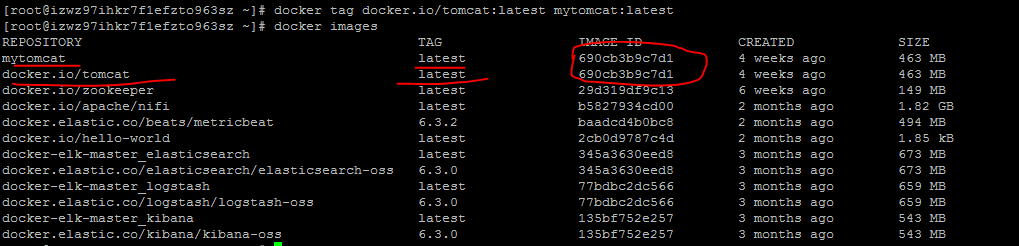
$docker images



使用镜像ID的前几位可区分的字符串来代替完整的镜像ID。

### 4.2.1 docker tag 添加镜像标签

$docker tag docker.io/tomcat:latest mytomcat:latest



### 4.2.2 inspect 查看详细信息

$docker inspect mytomcat:latest

### 4.2.3 history

--no-trunc 参数输出完整的命令



## 4.3 搜寻镜像

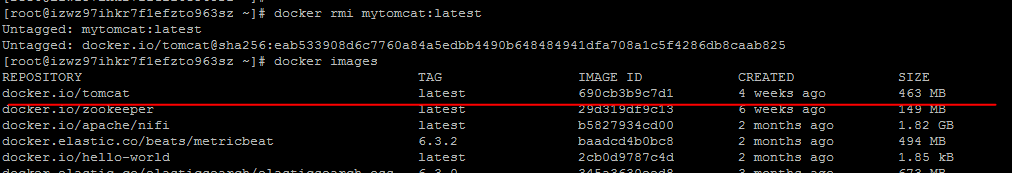
$docker search tomcat

## 4.4 删除镜像

### 4.4.1 根据镜像标签删除镜像

仅删除镜像的指定标签：

$docker rmi mytomcat:latest



可以看出mytomcat:latest已经不在镜像中。

### 4.4.1 根据镜像ID删除镜像

$docker ps -a 查看本机上存在的所有容器：



$docker rmi 998255a95a07

## 4.5 创建镜像

创建镜像方法有三种：

基于已有的镜像容器创建、基于本地模板导入、基于Dockerfile创建。

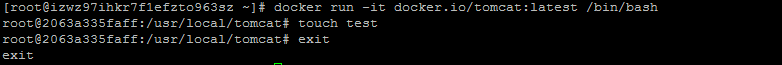
### 4.5.1基于已有的镜像容器创建

如：启动docker.io/tomcat:latest镜像，并新增一个test文件后退出。

$ docker run -it docker.io/tomcat:latest /bin/bash

root@2063a335faff:/usr/local/tomcat$ touch test

root@2063a335faff:/usr/local/tomcat$ exit



记住容器的ID为：2063a335faff

使用docker commit 来提交一个新镜像。

$docker commit -m “Add a new file” -a “Alan Jiang “ 2063a335faff test:0.1



提交成功后返回新创建的镜像ID信息。如上图：

$docker images

发现新创建的镜像已经存在了。



### 4.5.2 基于本地模板导入

如：基于redis-4.0.10.tar.gz 文件导入一个镜像：

$ cat redis-4.0.10.tar.gz | docker import - redis:4.0.10



$docker images



## 4.6 导出和载入镜像

docker save -o destfile repository:tag

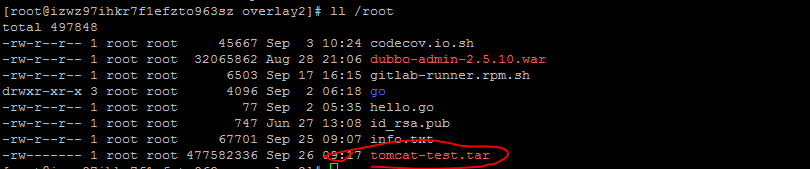
docker load --input file

docker load <file

导出镜像到/root下，保存为redis.4.0.10.tar文件。

$docker save -o /root/redis.4.0.10.tar redis:4.0.10

docker save -o /root/tomcat-test.tar test:0.1



$docker load --input /root/tomcat-test.tar



## 4.7 上传镜像

Docker push NAME:TAG

用户可以在Docker Hub上注册后，上传自己自制的镜像。

如：用户上传自己本地的test:latest镜像,可以先添加新的标签： user/test:latest， 然后 docker push 命令上传镜像。

$docker tag test:latest user/test:latest

$docker push user/test:latest

UserName:

Password:

Email:

# 5 操作Docker容器

容器是镜像运行的一个实例。

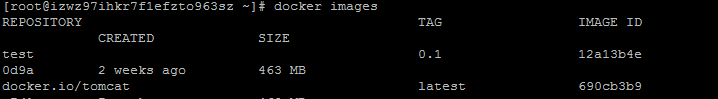
镜像是静态的只读文件，而容器带有运行时需要的可写文件层。

Docker容器是独立运行的一个（或一组）应用，以及他们必需的运行环境。

## 5.1 docker create 创建容器

例如：创建一个Tomcat容器：

先查看有无Tomcat镜像（如果本地没有，将从远程资源库PULL）

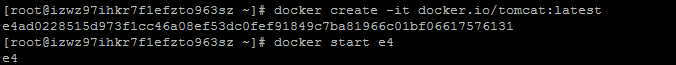


$docker create -it docker.io/tomcat:latest

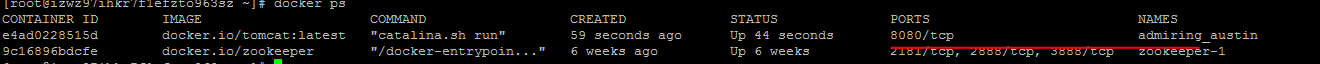
注意：创建的容器处于停止状态，需要docker start 来启动它。

## 5.2 docker start 启动容器

$docker start e4



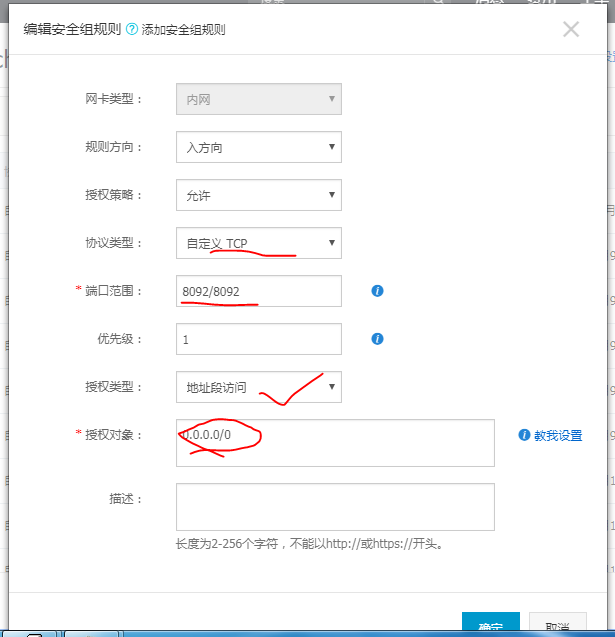
Docker ps 查看，发现容器已经启动。

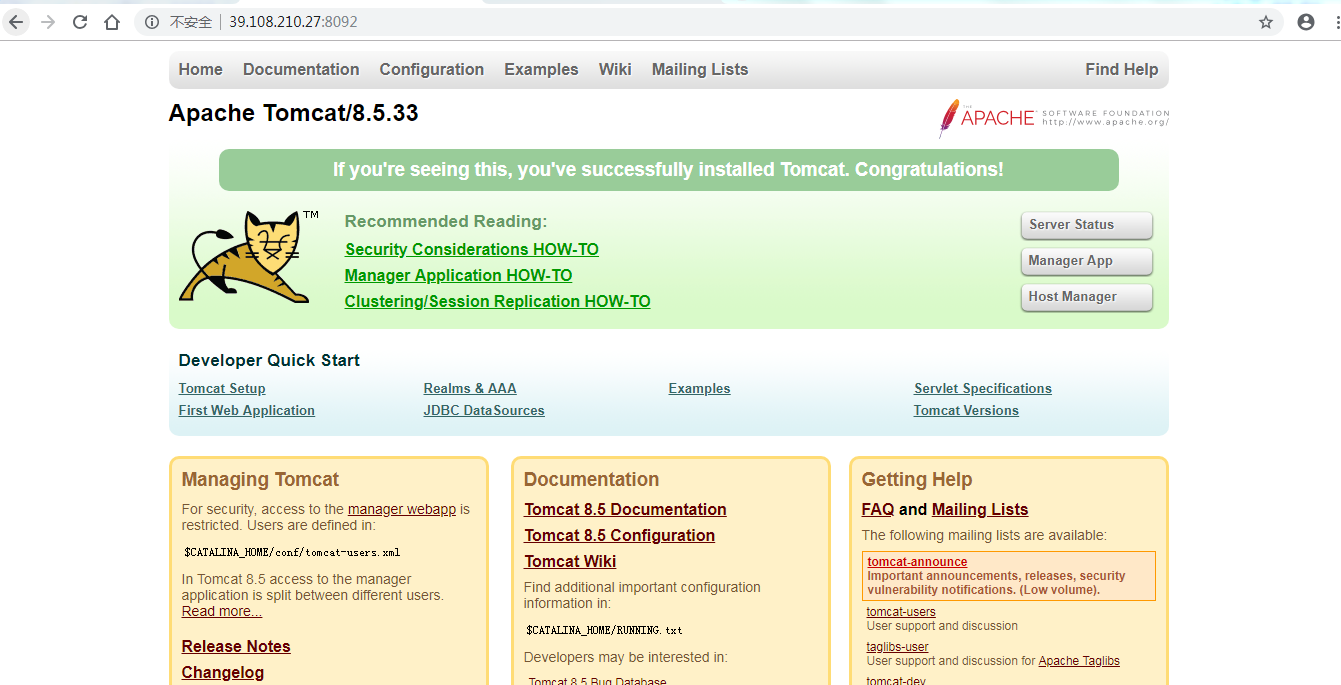


8080端口占用,可以指定容器和主机的映射端口

$docker run -p 8092:8080 docker.io/tomcat:latest

注意：云服务必须开放端口：





注意：docker create & docker start = docker run

注意：以上运行容器如果关闭终端，服务将停止，需要在后台运行容器。

$docker run -p 8092:8080 -d docker.io/tomcat:latest

容器运行模式即相关选项：

|  |  |
| --- | --- |
| 选项 | 说明 |
| -a | 是否绑定到stdin, stdout,stderror |
| -d | 是否在后台运行容器，默认为false |
| -i | 保持stdin 打开 |
| -t | 是否分配一个伪终端，默认为false |
| -p | 如何映射到本地主机端口 |
| -e | 指定容器内环境变量 |
|  |  |

## 5.3 终止容器

发现 docker stop tomcat-1 终止后，查看 docker ps -a|grep tomcat 发现容器又处于Up启动状态。

强制删除：

$docker stop dbb41c537468

$docker rm -f dbb41c537468

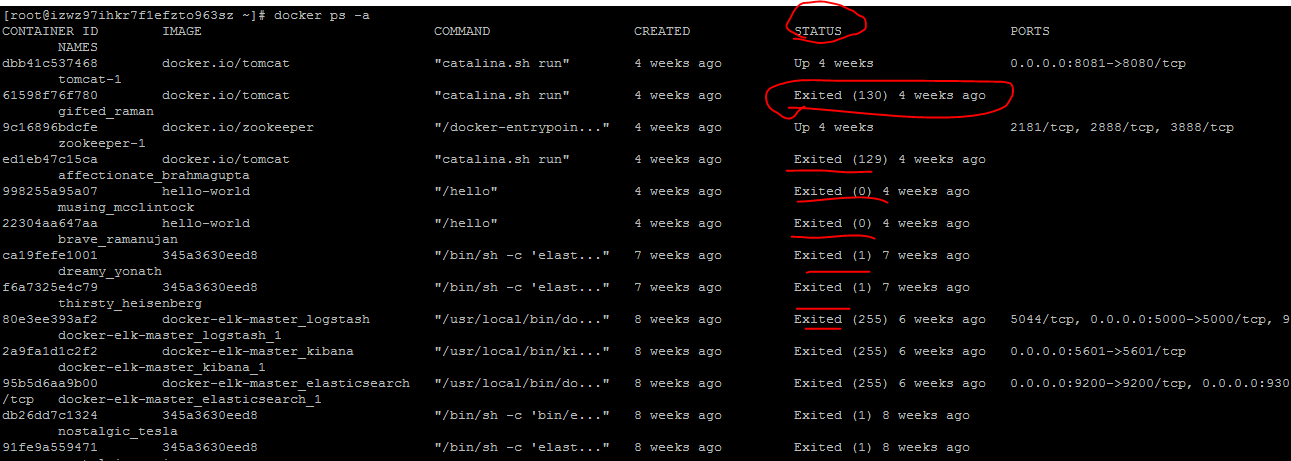
## 5.4 docker rm ID 删除容器

运行状态的容器必需先停止再删除

查看容器状态：

$docker ps -a

status 列状态值为：UP 表示正在运行。EXIT表示已经停止。



$ docker ps -a | grep Exited

# 6 Docker 数据管理

生产环境中使用DOCKER的过程中往往需要对数据进行持久化，或者多个容器之间共享数据，这必然涉及容器的数据管理操作。

## 6.1 数据卷和数据卷容器

数据卷：容器内数据直接映射到本地主机环境。

数据卷容器：使用特定容器维护数据卷。

数据卷是一个可供容器使用的特殊目录，它将主机操作系统目录直接映射进容器，类似与Mount操作。

## 6.2 数据卷重要特性：

可以在容器之间共享和重用，容器之间传递数据高效方便。

对数据卷的修改立马生效，无论是容器内操作还是本地操作。

对数据卷的更新不会影响镜像，解耦了应用和数据。

卷会一直存在，直到没有容器使用，可以安全卸载它。

## 6.3 数据卷的使用

* 在容器创建一个数据卷:docker run -v

使用docker run 命令时，使用 -v 可以创建多个数据卷。

* 挂载一个主机目录作为数据卷
* 挂载一个本地文件作为数据卷

# 参考文献

1 Step-by-Step High Availability with Docer and JavaEE

<https://community.oracle.com/docs/DOC-998210>

# 实践1 Docker 运行sonarqube服务

Step1: 安装数据库

$docker run --name postgresqllatest -v /data/postgresql/:/var/lib/postgresql/data/ -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=postgres -p 54321:5432 -d docker.io/postgres:latest运行docker run 经常会报错，提示容器名称已经占用，需要移除掉先前的或者使用新的容器名。

$docker ps -a



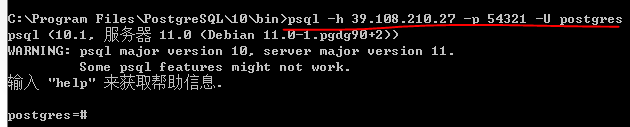
$docker stop ID

$docker rm -f ID

再次运行容器命令：

docker run --name postgresqllatest -v /data/postgresql/:/var/lib/postgresql/data/ -e POSTGRES\_USER=postgres -e POSTGRES\_PASSWORD=postgres -p 54321:5432 -d docker.io/postgres:latest

在WINDOWS客户端，连接云端通过docker启动的postgresql :



CREATE USER sonar WITH PASSWORD 'sonar';

Step2:

$docker run -d --name mysonarqube --link postgresqllatest -v /data/sonarqube/:/var/lib/sonarqube/data/ -p 9001:9000 -e SONARQUBE\_JDBC\_USERNAME=sonar -e SONARQUBE\_JDBC\_PASSWORD=sonar -e SONARQUBE\_JDBC\_URL=jdbc:postgresql://39.108.210.27:54321/sonar docker.io/sonarqube:latest

清理此容器的网络占用

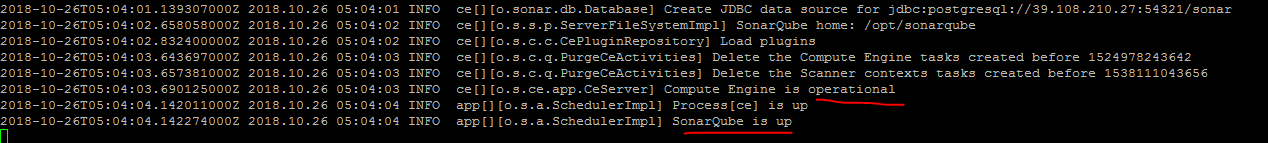
$docker network disconnect --force bridge mysonarqube

查看是否还有同名容器占用

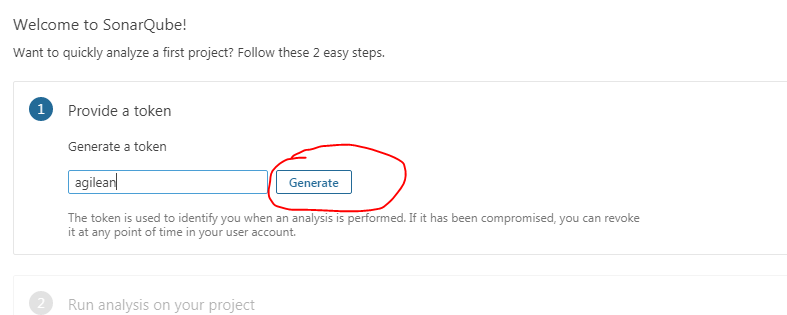
$docker network inspect mysonarqube

查看容器日志：

$docker logs -f -t --tail 600 mysonarqube



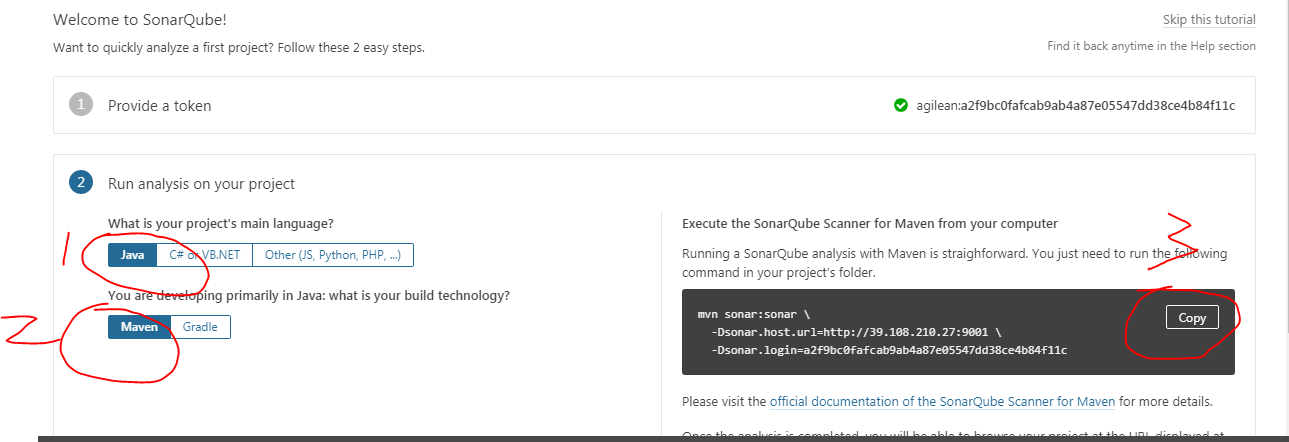
<http://39.108.210.27:9001/projects>



生成私有的Token.

**a2f9bc0fafcab9ab4a87e05547dd38ce4b84f11c**

选择分析的语言和构建：

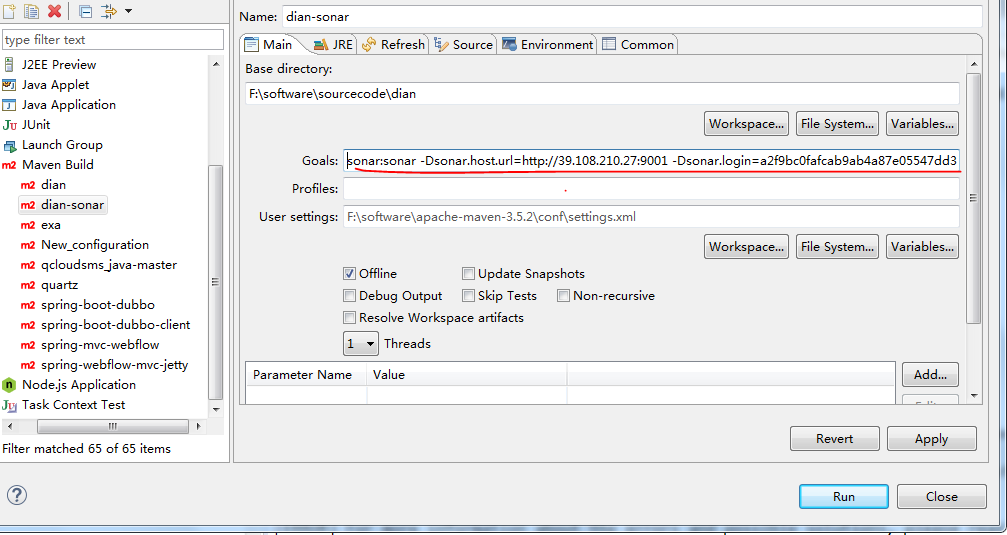


mvn sonar:sonar \

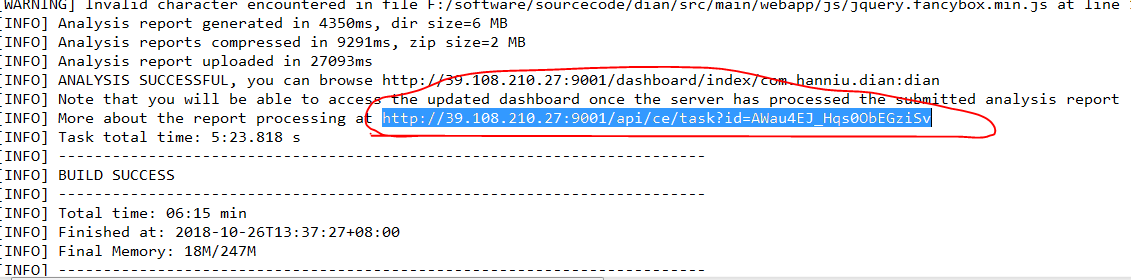
-Dsonar.host.url=http://39.108.210.27:9001 \

-Dsonar.login=a2f9bc0fafcab9ab4a87e05547dd38ce4b84f11c

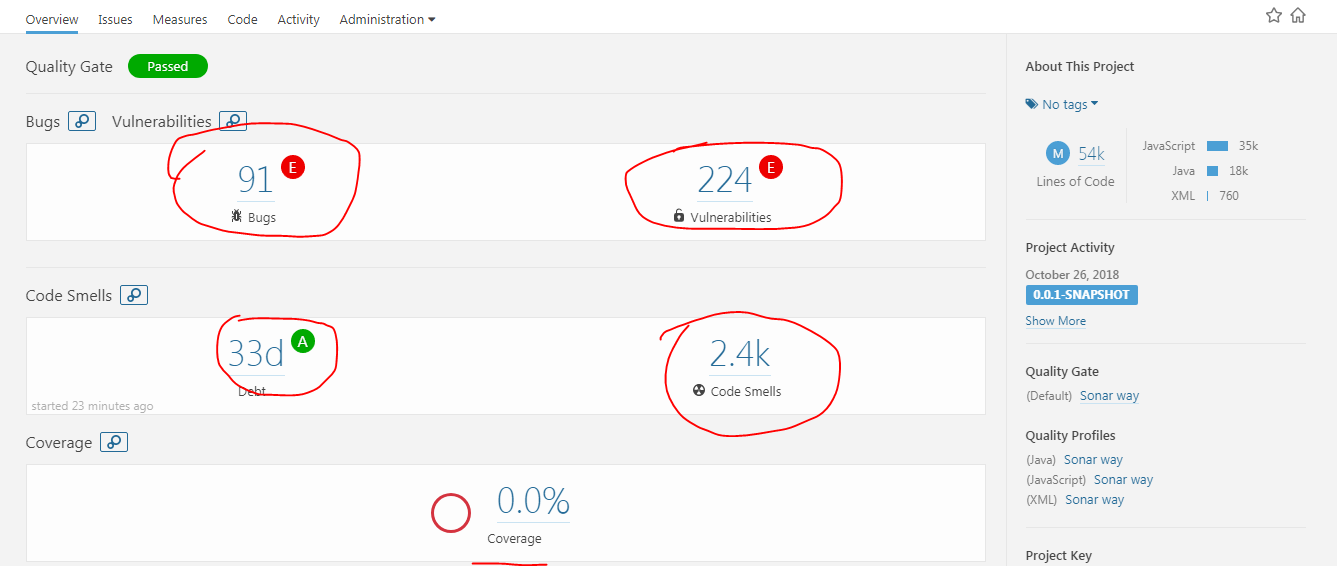
直接在ECLIPSE上对我的项目进行质量扫描：



运行完后，直接形成报告。



报告如下：



# 实践2 Dubbo+zookeeper

* Step1: 拉取zookeeper镜像

$docker pull zookeeper

$docker run --privileged=true -d --name zookeeper -v /data/zookeeper/:/var/lib/zookeeper/data/ --publish 2181:2181 -d zookeeper:latest

发现端口2181已经打开：

