1. 查看容器的root用户密码

docker logs <容器名orID> 2>&1 | grep '^User: ' | tail -n1

因为**[Docker](http://lib.csdn.net/base/docker" \o "Docker知识库" \t "_blank)**容器启动时的root用户的密码是随机分配的。所以，通过这种方式就可以得到redmine容器的root用户的密码了。

1. 查看容器日志

docker logs -f <容器名orID>

注：该命令查看的是最近一次登录容器时所有执行的命令，不能查看到每次登录容器的所有的命令，如：黄色部分为登录后执行的命令，包括返回的结果

[root@bogon workfiles]# docker logs -f 0e0de3ce2b19

[root@0e0de3ce2b19 /]# eixt

bash: eixt: command not found

[root@0e0de3ce2b19 /]# docker ps

bash: docker: command not found

[root@0e0de3ce2b19 /]# docker ps -a

bash: docker: command not found

[root@0e0de3ce2b19 /]# exit

exit

又如：

[root@bogon workfiles]# docker logs -f 6f640095a5b7

[root@6f640095a5b7 /]# ls

anaconda-post.log dev home lib64 media opt root sbin sys usr

bin etc lib lost+found mnt proc run srv tmp var

[root@6f640095a5b7 /]# cd /usr

[root@6f640095a5b7 usr]# ls

bin etc games include lib lib64 libexec local sbin share src tmp

[root@6f640095a5b7 usr]# cd local/

[root@6f640095a5b7 local]# ls

bin etc games include lib lib64 libexec sbin share src

[root@6f640095a5b7 local]# cd src

[root@6f640095a5b7 src]# ls

[root@6f640095a5b7 src]# cd ..

[root@6f640095a5b7 local]# cd ..

[root@6f640095a5b7 usr]# ls

bin etc games include lib lib64 libexec local sbin share src tmp

[root@6f640095a5b7 usr]# cd /

[root@6f640095a5b7 /]# ls

anaconda-post.log dev home lib64 media opt root sbin sys usr

bin etc lib lost+found mnt proc run srv tmp var

[root@6f640095a5b7 /]# exit

exit

1. 查看正在运行的容器

docker ps

该命令只能查看所有正在运行中的容器，如果没有容器正在运行，则不会有运行的结果，如（只有表头）：

[root@bogon workfiles]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

[root@bogon workfiles]#

docker ps -a为查看所有的容器，包括已经停止的。

[root@bogon workfiles]# docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

6f640095a5b7 centos "/bin/bash" 4 minutes ago Exited (0) 3 minutes ago kickass\_bohr

0e0de3ce2b19 centos "/bin/bash" 11 minutes ago Exited (127) 11 minutes ago nostalgic\_rosalind

502c2be89cc9 centos "/bin/bash" 38 minutes ago Exited (0) 12 minutes ago naughty\_galileo

d5881c1adf06 centos "/bin/bash" 39 minutes ago Exited (0) 39 minutes ago sleepy\_bardeen

8d5f0a8ff5d8 centos "/bin/bash" 8 days ago Exited (127) 8 days ago naughty\_pike

a9fdd31d657f centos "/bin/bash" 8 days ago Exited (0) 8 days ago reverent\_morse

4查看所有容器的id

docker ps -a -q

查看所有的容器Id，是docker ps –a 的简化版，只取id字段，如：

[root@bogon workfiles]# docker ps -a -q

6f640095a5b7

0e0de3ce2b19

502c2be89cc9

d5881c1adf06

8d5f0a8ff5d8

a9fdd31d657f

5启动一个已停止的容器并附加（attach）

docker start –a <容器名orID>

6. 从存在的容器生成一个新的镜像

docker commit -m "centos\_java\_hadoop" -a "new\_cenots\_image" 8d5f0a8ff5d8

结果类似这样：

[root@bogon workfiles]# docker commit -m "centos\_java\_hadoop" -a "new\_cenots\_image" 8d5f0a8ff5d8

5c1c3ed10123875a0a806e5749e6f00cdc06bc140ade73e7202990ba5555b85b

7 停止一个正在运行的容器

docker stop <容器名orID>

[[root@bogon workfiles]# docker stop 6f6

6f6

6删除所有容器

docker rm $(docker ps -a -q)

删除单个容器

docker rm <容器名orID>

[root@bogon workfiles]# docker rm 0e0

0e0

[root@localhost local]# docker rm a9fdd31d657f

a9fdd31d657f

注：在删除之前要先停止容器的运行

7停止、启动、杀死一个容器

1. docker stop <容器名orID>

[root@localhost local]# docker stop 8d5f0a8ff5d8

1. docker start <容器名orID>

启动一个容器并附加到此容器中：

[root@localhost local]# docker start -a 8d5f0a8ff5d8

docker kill <容器名orID>

[root@localhost local]# docker kill 8d5f0a8ff5d8

8d5f0a8ff5d8

1. 查看所有镜像
2. docker images

[root@localhost local]# docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

centos\_with\_java\_done latest 4be7d94a30f3 2 days ago 1.32 GB

centos latest 97cad5e16cb6 5 weeks ago 196.5 MB

1. 删除所有镜像
2. docker rmi $(docker images | grep none | awk '{print $3}' | sort -r)

rmi是移除镜像

1. 运行一个新容器，同时为它命名、端口映射、文件夹映射。以redmine镜像为例
2. docker run --name redmine -p 9003:80 -p 9023:22 -d -v /var/redmine/files:/redmine/files -v /var/redmine/mysql:/var/lib/mysql sameersbn/redmine
3. 一个容器连接到另一个容器
4. docker run -i -t --name sonar -d -link mmysql:db tpires/sonar-server
5. sonar

容器连接到mmysql容器，并将mmysql容器重命名为db。这样，sonar容器就可以使用db的相关的环境变量了。

13．在docker hub中搜索可用的容器

[root@localhost local]# docker search centos

NAME DESCRIPTION STARS OFFICIAL AUTOMAT

centos The official build of CentOS. 2905 [OK]

jdeathe/centos-ssh CentOS-6 6.8 x86\_64 / CentOS-7 7.2.1511 x8... 49 [OK]

nimmis/java-centos This is docker images of CentOS 7 with dif... 18 [OK]

gluster/gluster-centos Official GlusterFS Image [ CentOS-7 + Glu... 14 [OK]

million12/centos-supervisor Base CentOS-7 with supervisord launcher, h... 12 [OK]

14.拉取镜像

docker pull <镜像名:tag>

如

docker pull sameersbn/redmine:latest

1. 当需要把一台机器上的镜像迁移到另一台机器的时候，需要保存镜像(在原机器上保存)与加载镜像（在新的机器上加载）。  
   机器
2. docker save busybox-1 > /home/save.tar

使用scp将save.tar拷到机器b上，然后：

docker load < /home/save.tar

1. 构建自己的镜像（通过文件dockerfile构建镜像）

docker build -t <镜像名> <Dockerfile路径>

如Dockerfile在当前路径：

docker build -t xx/gitlab .

1. 重新查看container的stdout
2. # 启动top命令，后台运行
3. $ ID=$(sudo docker run -d ubuntu /usr/bin/top -b)
4. # 获取正在running的container的输出
5. $ sudo docker attach $ID
6. top - 02:05:52 up 3:05, 0 users, load average: 0.01, 0.02, 0.05
7. Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
8. Cpu(s): 0.1%us, 0.2%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
9. Mem: 373572k total, 355560k used, 18012k free, 27872k buffers
10. Swap: 786428k total, 0k used, 786428k free, 221740k cached
11. ^C$

$ sudo docker stop $ID

1. 后台运行(-d)、并暴露端口(-p)
2. docker run -d -p 127.0.0.1:33301:22 centos6-ssh
3. 从**[Container](http://lib.csdn.net/base/docker" \o "Docker知识库" \t "_blank)**中拷贝文件出来（到宿主机）
4. sudo docker cp 7bb0e258aefe:/etc/debian\_version .

拷贝7bb0e258aefe中的/etc/debian\_version到当前目录下。  
注意：只要7bb0e258aefe没有被删除，文件命名空间就还在，可以放心的把exit状态的container的文件拷贝出来

**坑**

1. ubuntu14下的docker是没有service服务。去除每次sudo运行docker命令，需要添加组：
2. # Add the docker group if it doesn't already exist.
3. $ sudo groupadd docker
4. #改完后需要重新登陆用户
5. $ sudo gpasswd -a ${USER} docker
6. ubuntu14的febootstrap没有-i命令
7. Dockerfile中的EXPOSE、docker run --expose、docker run -p之间的区别  
   Dockerfile的EXPOSE相当于docker run --expose，提供container之间的端口访问。docker run -p允许container外部主机访问container的端口