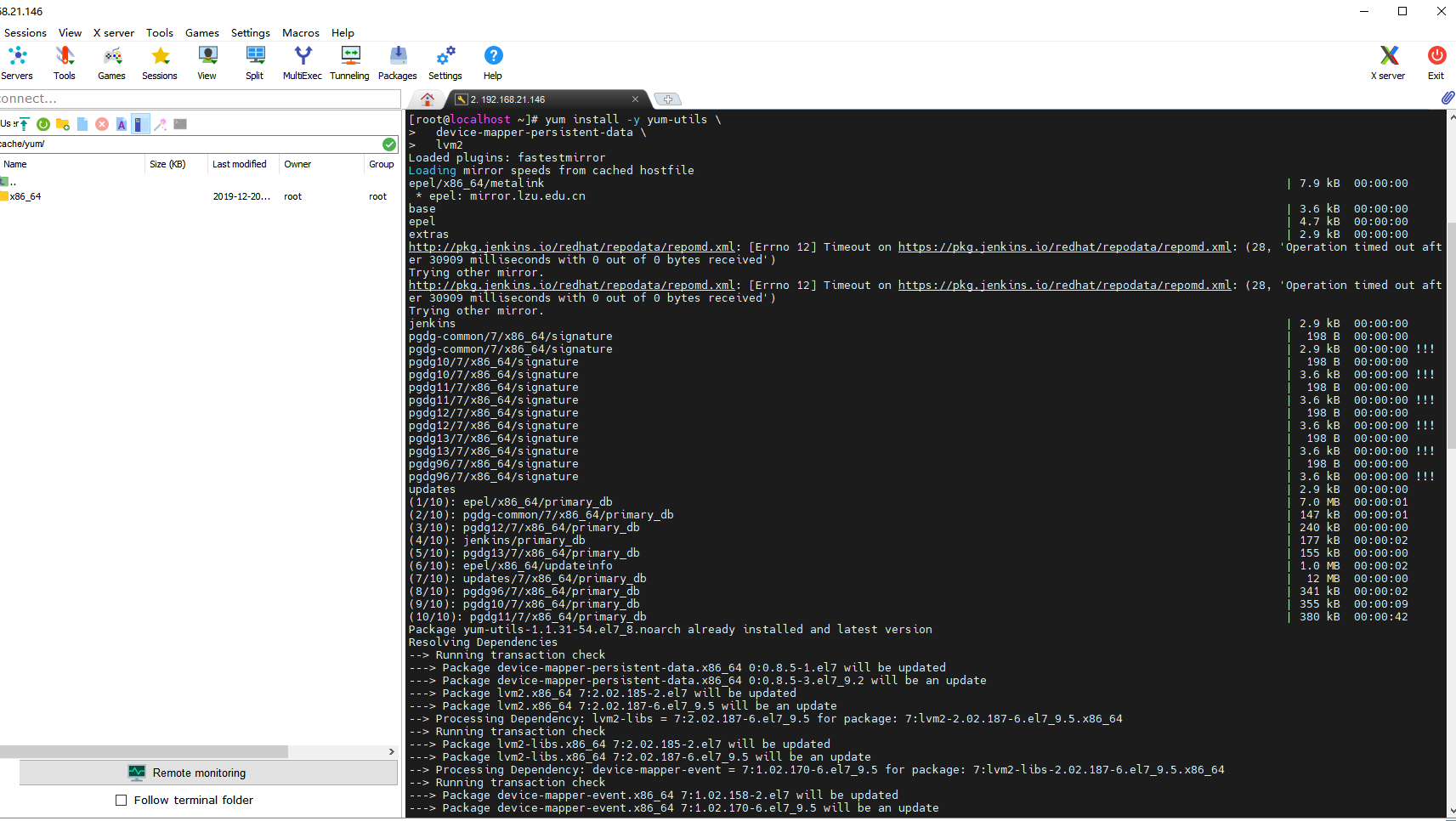
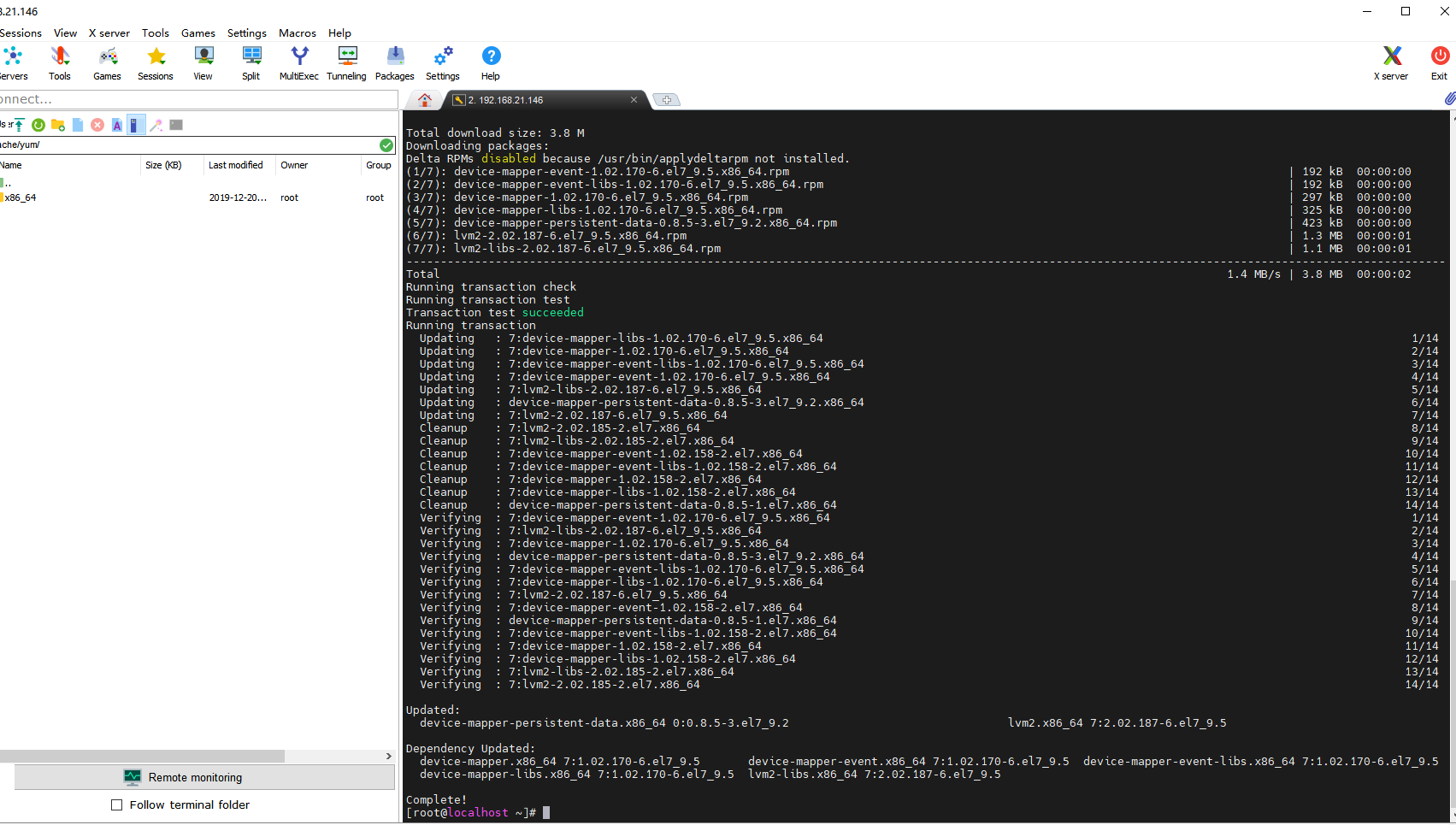
安装所需的软件包。

sudo yum install -y yum-utils \

device-mapper-persistent-data \

lvm2





使用以下命令来设置稳定的仓库。

阿里云

$ sudo yum-config-manager \

--add-repo \

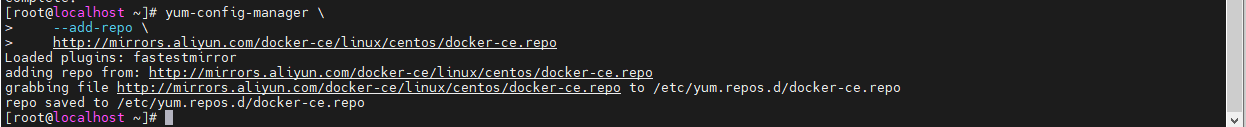
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

使用

yum-config-manager命令可以进行YUM配置管理。

|  |  |
| --- | --- |
| --add-repo=<软件仓库> | 从指定文件或URL添加（和启用）软件仓库 |

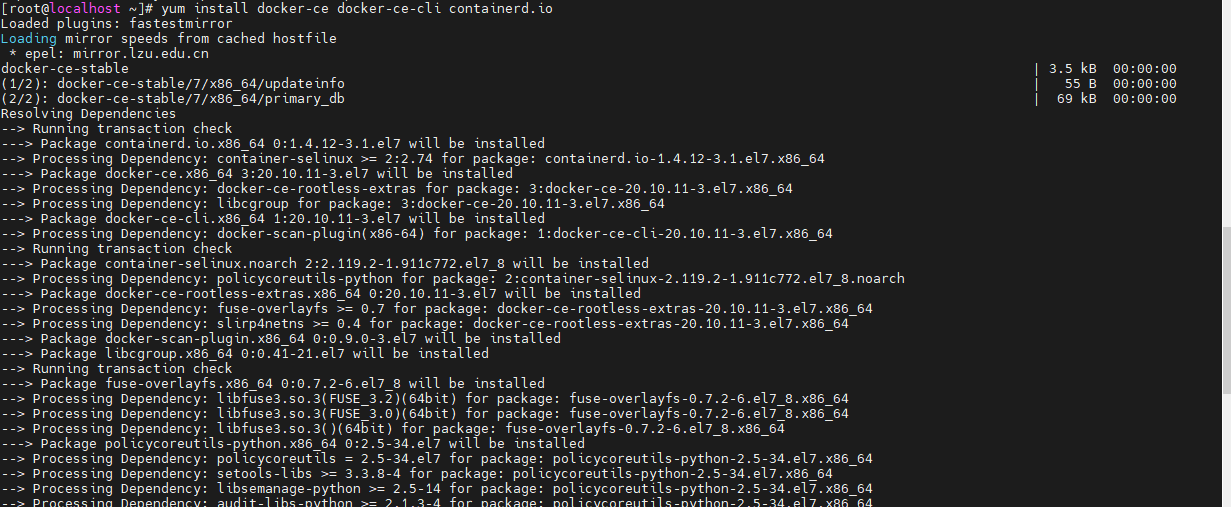
参考：<https://www.cnblogs.com/bulh/articles/12985279.html>

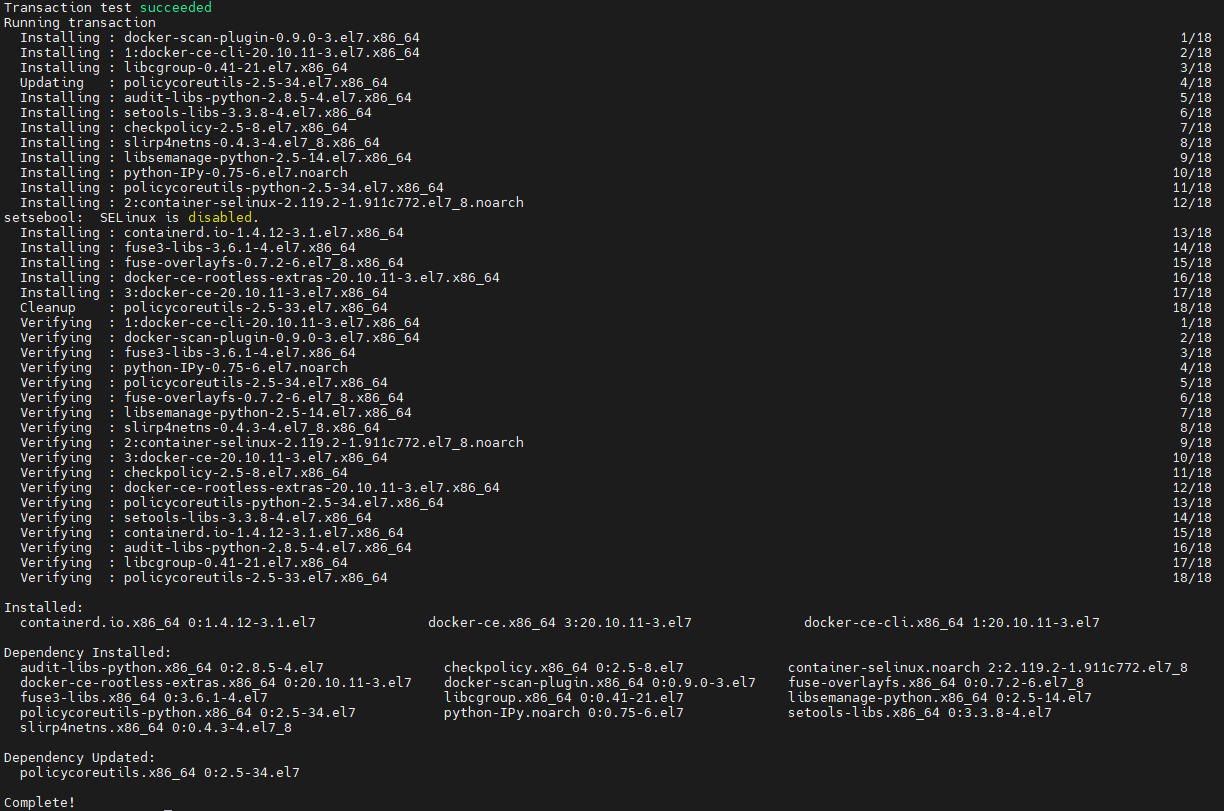


### **安装 Docker Engine-Community**

安装最新版本的 Docker Engine-Community 和 containerd，或者转到下一步安装特定版本：

$ sudo yum install docker-ce docker-ce-cli containerd.io





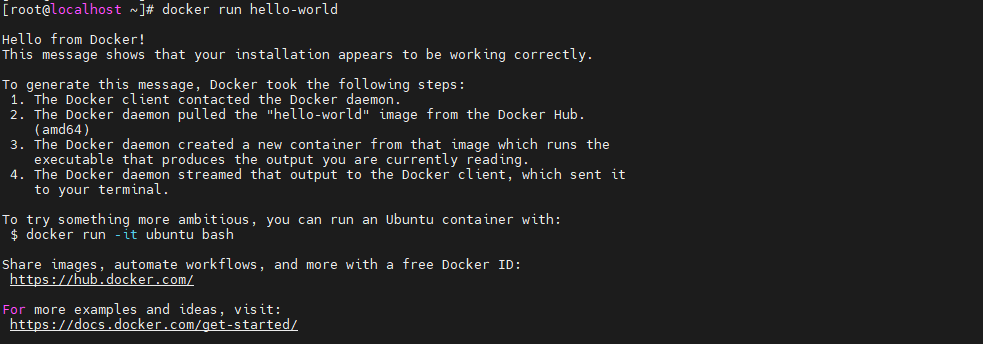
启动 Docker。

$ sudo systemctl start docker



运行 hello-world 映像来验证是否正确安装了 Docker Engine-Community 。

$ sudo docker run hello-world



国内从 DockerHub 拉取镜像有时会遇到困难，此时可以配置镜像加速器

运行 Docker 的云平台选择对应的镜像加速服务。

阿里云镜像获取地址：[https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors](https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors" \t "https://www.runoob.com/docker/_blank)，登陆后，左侧菜单选中镜像加速器就可以看到你的专属地址了：



## 配置镜像加速器

针对Docker客户端版本大于 1.10.0 的用户

您可以通过修改daemon配置文件/etc/docker/daemon.json来使用加速器

sudo mkdir -p /etc/docker

sudo tee /etc/docker/daemon.json <<-'EOF'

{

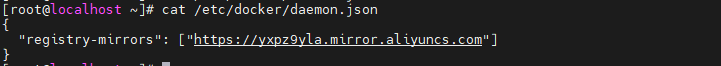
"registry-mirrors": ["https://yxpz9yla.mirror.aliyuncs.com"]

}

EOF

sudo systemctl daemon-reload

sudo systemctl restart docker



之后重新启动服务：

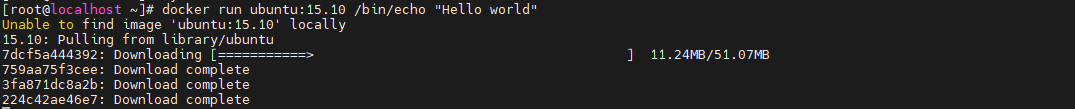
$ **sudo** systemctl daemon-reload  
$ **sudo** systemctl restart docker



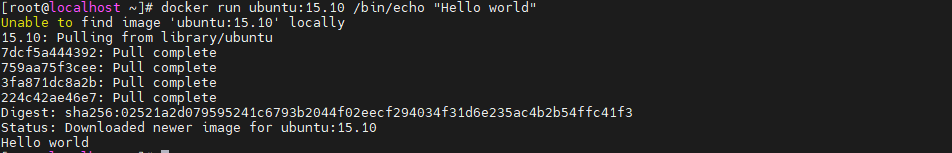
runoob@runoob:~$ docker run ubuntu:15.10 /bin/echo "Hello world"Hello world

以上命令完整的意思可以解释为：Docker 以 ubuntu15.10 镜像创建一个新容器，然后在容器里执行 bin/echo "Hello world"，然后输出结果。

docker run ubuntu:15.10 /bin/echo "Hello world"



Docker 首先从本地主机上查找镜像是否存在，如果不存在，Docker 就会从镜像仓库 Docker Hub 下载公共镜像。这里本地就是不存在。



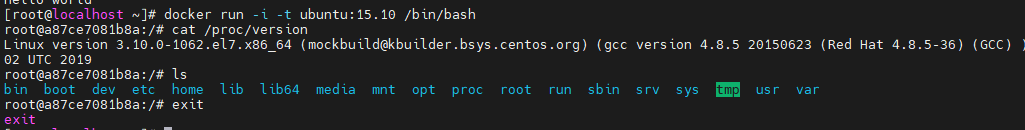
下载了镜像输出了helloworld

## 运行交互式的容器

我们通过 docker 的两个参数 -i -t，让 docker 运行的容器实现**"对话"**的能力：

runoob@runoob:~$ docker run -i -t ubuntu:15.10 /bin/bash

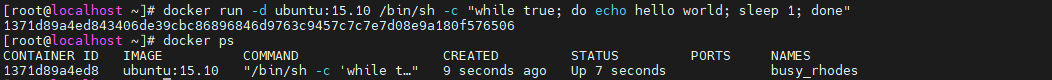
root@0123ce188bd8:/#



## 启动容器（后台模式）

使用以下命令创建一个以进程方式运行的容器

runoob@runoob:~$ docker run -d ubuntu:15.10 /bin/sh -c "while true; do echo hello world; sleep 1; done"2b1b7a428627c51ab8810d541d759f072b4fc75487eed05812646b8534a2fe63

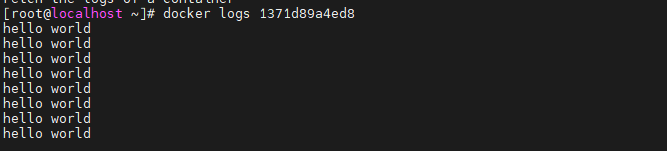


在宿主主机内使用 **docker logs** 命令，查看容器内的标准输出：

runoob@runoob:~$ docker logs 2b1b7a428627

后面的12位是容器ID。





## 停止容器

我们使用 **docker stop** 命令来停止容器:



会输出容器ID

此时再用docker ps查看



### **启动已停止运行的容器**

查看所有的容器命令如下：

$ docker ps -a

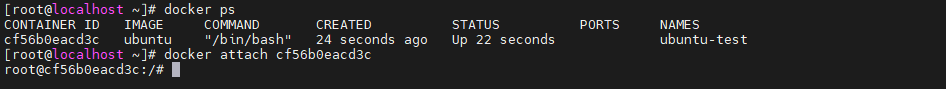
可以看出，一个docker可以运行多个容器。

使用 docker start 启动一个已停止的容器：

$ docker start b750bbbcfd88

容器在后台运行时，可以命令进入容器。

docker attach



### **删除容器**

删除容器使用 **docker rm** 命令：



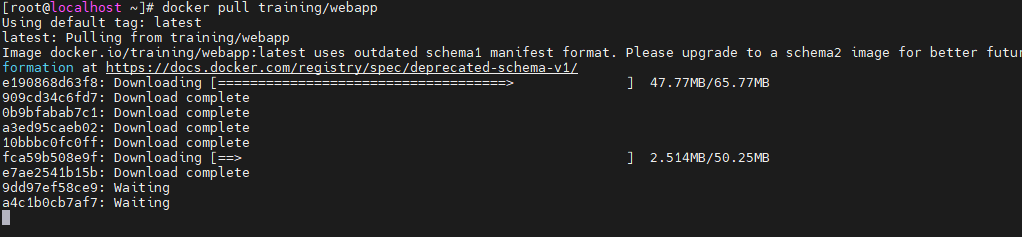
下面的命令可以清理掉所有处于终止状态的容器。

$ docker container prune

在docker容器中运行一个 Python Flask 应用来运行一个web应用。

runoob@runoob:~# docker pull training/webapp # 载入镜像

runoob@runoob:~# docker run -d -P training/webapp python app.py



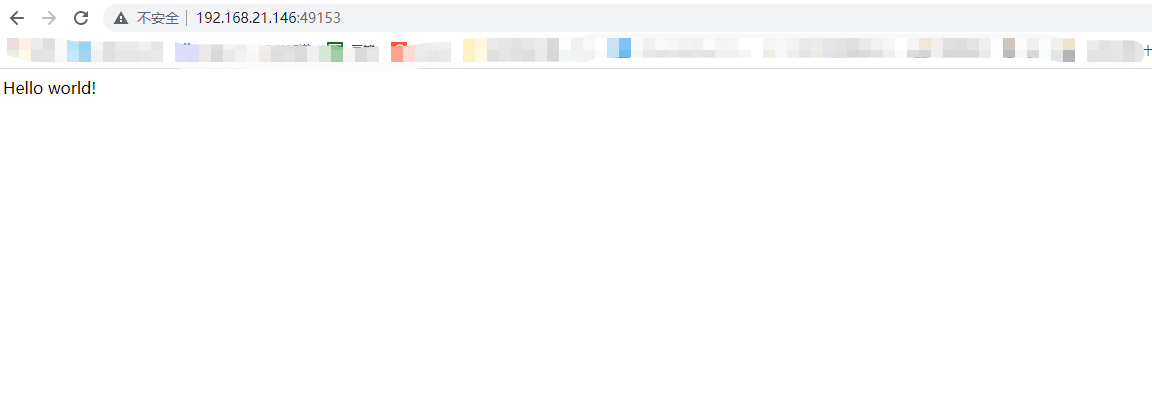


返回的这一串是容器ID

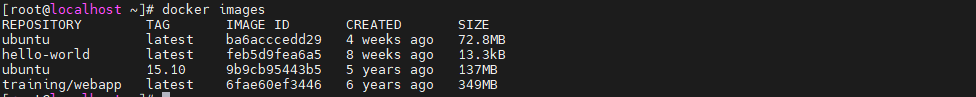


Docker 开放了 5000 端口（默认 Python Flask 端口）映射到主机端口 32769 上。

这时我们可以通过浏览器访问WEB应用



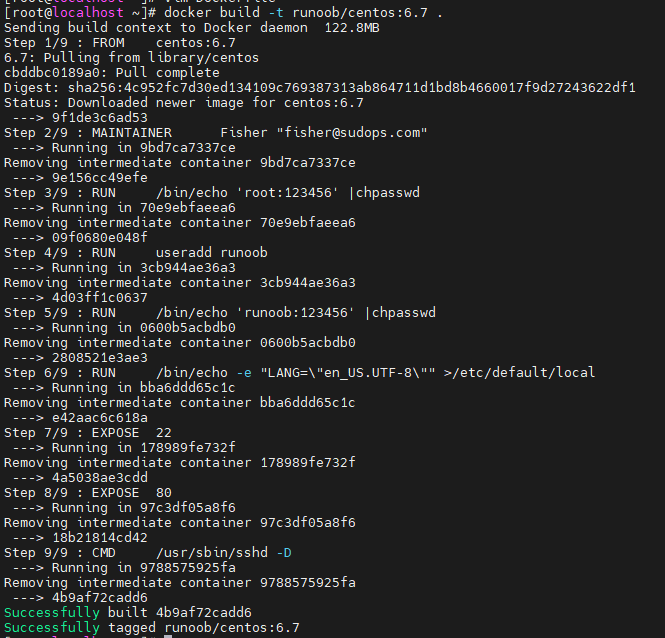
可以使用 **docker images** 来列出本地主机上的镜像。



使用命令 **docker build** ， 从零开始来创建一个新的镜像。为此，我们需要创建一个 Dockerfile 文件，其中包含一组指令来告诉 Docker 如何构建我们的镜像。

使用 Dockerfile 文件，通过 docker build 命令来构建一个镜像。

runoob@runoob:~$ docker build -t runoob/centos:6.7 .





可以使用 docker tag 命令，为镜像添加一个新的标签。

runoob@runoob:~$ docker tag 860c279d2fec runoob/centos:dev

指定容器端口绑定到主机端口。

runoob@runoob:~$ docker run -d -p 5000:5000 training/webapp python app.py

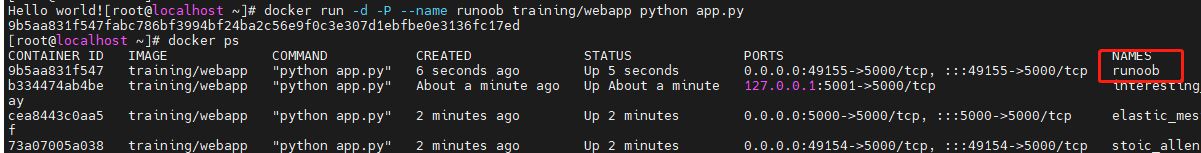
容器绑定的网络地址，比如绑定 127.0.0.1。

runoob@runoob:~$ docker run -d -p 127.0.0.1:5001:5000 training/webapp python app.py



当我们创建一个容器的时候，docker 会自动对它进行命名。另外，我们也可以使用 **--name** 标识来命名容器，例如：

runoob@runoob:~$ docker run -d -P --name runoob training/webapp python app.py



Dockerfile 是一个用来构建镜像的文本文件，文本内容包含了一条条构建镜像所需的指令和说明。