docker简明教程

Docker简明教程

来源: saymagic 发布时间: 2016-01-02 14:15 阅读: 11345 次 推荐: 28 原文链接 [收藏]

> Docker自从诞生以来就一直备受追捧，学习Docker是一件很炫酷、很有意思的事情。我希望通过这篇文章能够让大家快速地入门Docker， 并有一些学习成果来激发自己的学习兴趣。我也只是一个在Docker这条巨鲸上玩耍的小孩，全文如有不明确、不正确的地方，还请斧正。

Ubuntu上安装Docker

Docker的基础是Linux容器技术，因此学习Docker最好是使用Linux。这里推荐64位Ubuntu系统，因为在写此文 (2015-05-28)时，Docker还不支持32位，尽管民间有些土办法可以象征性的解决，但还是推荐初学Docker的尽量按照标准的来。如果手 边没有Ubuntu系统可以去Digitalocean、Ucloud等云服务商去租用一个Linux服务器。这样即使玩坏了也可以随时重新开始。

在Ubuntu中只需要运行一行命令即可实现Docker的安装：

```
sudo apt-get install docker.io
```

运行完后，可以在终端输入docker看到下面的界面证明我们安装成功了（注：提示权限问题就添加sudo，下文同）：

```
root@learndocker:~# docker
Usage: docker [OPTIONS] COMMAND [arg...]
 -H=[unix:///var/run/docker.sock]: tcp://host:port to bind/connect to or unix://path/to/socket to use

A self-sufficient runtime for linux containers.

Commands:
    attach    Attach to a running container
    build     Build an image from a Dockerfile
    commit    Create a new image from a container's changes
    cp        Copy files/folders from the containers filesystem to the host path
    diff      Inspect changes on a container's filesystem
    events    Get real time events from the server
    export    Stream the contents of a container as a tar archive
    history   Show the history of an image
    images    List images
    import    Create a new filesystem image from the contents of a tarball
    info      Display system-wide information
    inspect   Return low-level information on a container
    kill      Kill a running container
    load      Load an image from a tar archive
    login     Register or Login to the docker registry server
    logs      Fetch the logs of a container
    port      Lookup the public-facing port which is NAT-ed to PRIVATE_PORT
    pause     Pause all processes within a container
    ps        List containers
    pull      Pull an image or a repository from the docker registry server
    push      Push an image or a repository to the docker registry server
    restart   Restart a running container
    rm        Remove one or more containers
    rmi       Remove one or more images
    run       Run a command in a new container
    save      Save an image to a tar archive
    search    Search for an image in the docker index
    start     Start a stopped container
    stop      Stop a running container
    tag       Tag an image into a repository
    top       Lookup the running processes of a container
    unpause   Unpause a paused container
    version   Show the docker version information
    wait      Block until a container stops, then print its exit code

root@learndocker:~#
```

从上图可以看到，Docker的命令并不多，只有三十几个。例如我们可以输入`docker info`来查看我们安装的Docker信息:

```
root@learndocker:~# docker info
Containers: 0
Images: 0
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Dirs: 0
Execution Driver: native-0.2
Kernel Version: 3.13.0-52-generic
WARNING: No swap limit support
root@learndocker:~#
```

运行容器

安装好之后，我们就可以来开始Docker之旅了，

我们现在的Docker还是一个”裸”Docker，上面没有容器，等一下，什么是容器？所谓容器就是Docker中用来运行应用 的，Docker的容器很轻量级，但功能却强悍的很。也没有镜像。镜像？镜像简单理解就是容器的只读版本，用来方便存储与交流。此时，我们可以通过官方提 供给我们的镜像来进行学习。比如我们想在Docker中运行一个Ubuntu系统，很简单，Docker中得pull命令是用来获取镜像的，执行下面的命 令，就会从官方仓库里获取Ubuntu 14.04版本的系统：

```
docker pull ubuntu:14.04
```

images命令用来查看本机Docker中存在哪些镜像，运行 `docker images` 就会看到我们刚才获取的Ubuntu14.04系统:

```
root@learndocker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              14.04               07f8e8c5e660        4 weeks ago         188.3 MB
```

现在，我们把刚刚的镜像运行起来，运行起来的镜像就叫做容器了，容器是可读写的，这样我们就可以在容器里做很多有意思的事情了。`run` 命令就是将镜像运行起来的，运行：

```
docker run -it ubuntu:14.04
```

仔细看，你会发现终端交互的用户名变掉了，说明我们进入到了容器的内部，效果如下:

```
root@learndocker:~# docker run -it ubuntu:14.04
root@79c761f627f3:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@79c761f627f3:/#
```

现在我们所做的任何操作都是针对于目前容器而言的，不会影响到原来的系统，例如，我们在里面安装下nginx服务器，运行如下命令：

```
sudo apt-get install -y nginx
```

完成后执行nginx -v就会发现我们已经将nginx安装成功:

```
root@79c761f627f3:/home# nginx -v
nginx version: nginx/1.4.6 (Ubuntu)
```

将容器转化为镜像

在上一小节中，我们已经在容器里安装好了nginx，接下来我们希望将这个容器内容保存下来，这样我们下次就无需再次安装了。这就是Docker中将容器转换为镜像的技术。

如果您还在刚刚的安装了nginx的终端里，执行exit退出此终端，回到系统本身的终端:

```
root@79c761f627f3:/home# exit
exit
root@learndocker:~#
```

ps命令可以查看我们当前都运行了哪些容器，加上-a参数后就表示运行过哪些容器，因为我们刚刚已经退出了安装nginx的容器，因此我现在想查看它的话，需要使用-a参数，执行如下命令：

```
docker ps -a
```

此时，就会显示出我们刚刚运行的容器，并且Docker会很贴心的随机给每个容器都起个Names方便标示。效果如下：

```
root@learndocker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                          PORTS               NAMES
79c761f627f3        ubuntu:14.04        /bin/bash           23 minutes ago      Exited (0) About a minute ago                       cocky_sinoussi
root@learndocker:~#
```

commit命令用来将容器转化为镜像，运行下面的命令，我们可以讲刚刚的容器转换为镜像：

```
sudo docker commit -m "Added nginx from ubuntu14.04" -a "saymagic" 79c761f627f3 saymagic/ubuntu-nginx:v1
```

其中，-m参数用来来指定提交的说明信息；-a可以指定用户信息的；79c761f627f3代表的是容器的id；saymagic/ubuntu-nginx:v1指定目标镜像的用户名、仓库名和tag 信息。创建成功后会返回这个镜像的ID信息。注意的是，你一定要将saymagic改为你自己的用户名。因为下文还会用到此用户名。

```
root@learndocker:~# sudo docker commit -m "Added nginx from ubuntu14.04" -a "saymagic" 79c761f627f3 saymagic/ubuntu-nginx:v1
8fbd47d369efca41eea3ba7f569ec994752788ef666fc315c02a0758533ffe24
```

这是我们再次使用docker images命令就会发现此时多出了一个我们刚刚创建的镜像:

```
root@learndocker:~# docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
saymagic/ubuntu-nginx   v1                  8fbd47d369ef        4 minutes ago       206.4 MB
ubuntu                  14.04               07f8e8c5e660        4 weeks ago         188.3 MB
```

此时，如果运行 docker run -it saymagic/ubuntu-nginx:v1 就会是一个已经安装了nginx的容器:

```
root@learndocker:~# docker run -it saymagic/ubuntu-nginx:v1
root@28e310f718d5:/# nginx -v
nginx version: nginx/1.4.6 (Ubuntu)
root@28e310f718d5:/#
```

存储镜像

我们刚刚已经创建了自己的第一个镜像，尽管它很简单，但这已经非常棒了，现在，我们希望它能够被更多的人使用到，此时，我们就需要将这个镜像上 传到镜像仓库，Docker的官方Docker Hub应该是目前最大的Docker镜像中心，所以，我们就将我们的镜像上传到Docker Hub。

首先，我们需要成为Docker Hub的用户，前往 https://hub.docker.com/ 进行注册。需要注意的是，为了方便下面的操作，你需要将你的用户名设为和我刚刚在上文提到的自定义用户名相同，例如我的刚刚将镜像的名字命名为是saymagic/ubuntu-nginx:v2，所以我的用户名为saymagic、注册完成后记住用户名、密码、邮箱。

login默认是用来登陆Docker Hub的，因此，输入如下命令来尝试登陆Docker Hub：

```
docker login
```

此时，就会输出交互，让我们输入Username、Password、Email,成功输入我们刚才注册的信息后就会返回Login Success提示:

```
root@learndocker:~# docker login
Username: saymagic
Password:
Email: saymagic@163.com
Login Succeeded
root@learndocker:~#
```

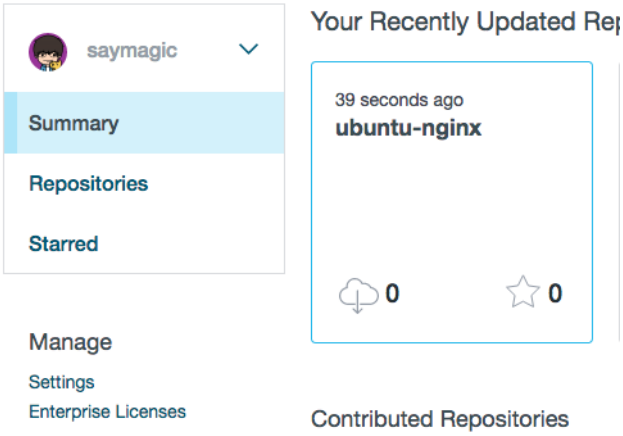
运行命令:

```
docker push saymagic/ubuntu-nginx:v1
```

这就是我们为什么将刚刚的镜像命名为saymagic/ubuntu-nginx:v1的原因，如果你上面步骤都操作正确的正确的话，是会得到下面的内容:

```
root@learndocker:~# docker push saymagic/ubuntu-nginx:v1
The push refers to a repository [saymagic/ubuntu-nginx] (len: 1)
Sending image list
Pushing repository saymagic/ubuntu-nginx (1 tags)
e9e06b06e14c: Image already pushed, skipping
a82efea989f9: Image already pushed, skipping
37bea4ee0c81: Image already pushed, skipping
07f8e8c5e660: Image already pushed, skipping
8fbd47d369ef: Image successfully pushed
Pushing tag for rev [8fbd47d369ef] on {https://cdn-registry-1.docker.io/v1/repositories/saymagic/ubuntu-nginx/tags/v1}
root@learndocker:~#
```

此时，不出意外的话，我们的镜像已经被上传到Docker Hub上面了，去Docker Hub上面看看：



果然，我们在Docker Hub上有了我们的第一个镜像，此时，其它的用户就可以通过命令`docker pull saymagic/ubuntu-nginx`来直接获取一个安装了nginx的ubuntu系统了。不信？那就自己实践一下吧！

Dockerfile使用

通过上面的学习，我们掌握了如何创建镜像、获取镜像、上传镜像、运行容器等等内容。有了上面的知识，我们来次实战。

我们刚刚使用了`commit`命令创建了一个安装nginx的镜像，但其实Docker创建镜像的命令还有`build`，`build`命令可以通过指定一个Dockerfile文件来实现将镜像创建过程自动化。Dockerfile文件有着特定的编写规则，但语法都还比较容易理解。这次我们不仅使用Dockerfile文件来创建一个像上文一样安装nginx的ubuntu镜像，还要发挥nginx的老本行来运行一个网页吧！DockFile可以很轻松的完成这个问题。首先将新建一个名字为`www`的文件夹，文件夹下面可以放一些HTML网页，比如新建一个`index.html`文件，随便写点内容：

```
<html>
<head>
Learn Docker
</head>
<body>
<h1>Enjoy Docker!</h1>
</body>
</html>
```

在`www`的同级目录下新建一个名为`Dockerfile`的文件，将DockerFile文件改写如下：

```
FROM ubuntu:14.04
MAINTAINER saymagic saymagic@163.com
RUN apt-get update
RUN apt-get install -y nginx
COPY ./www /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

我来整体的解释下这个Dockerfile文件，第一行是用来声明我们的镜像是基于什么构建的，这里我们指定为ubuntu 14.04，第二行的作用在于告诉别人你的大名。第三行和第四行的`RUN`命令用来在容器内部的shell里执行命令。第五行将当前系统的`www`文件夹拷贝到容器的`/usr/share/nginx/html`目录下，第六行声明当前需要对外开放80端口，最后一行表示运行容器时开启nginx。不理解没关系，因为这都是固定的语法，感兴趣可以多看相关内容。此时我们通过`build`命令来构建镜像，运行：

```
docker build -t="saymagic/ubuntu-nginx:v2" .
```

注意，最后的.表示Dockerfile在当前目录，也可指定其它目录。此时，再次运行`docker images`就会看到刚刚生成的镜像：

```
root@learndocker:/home/docker# docker images
REPOSITORY            TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
saymagic/ubuntu-nginx   v2                  11132a3bdd98        3 minutes ago       227.7 MB
saymagic/ubuntu-nginx   v1                  8fbd47d369ef        5 hours ago         206.4 MB
ubuntu                  14.04               07f8e8c5e660        4 weeks ago         188.3 MB
```

现在我们就可以运行刚刚的镜像了，和前面运行稍有不同，此时我们需要对外指定80端口，该行为通过-p参数指定，运行：

```
docker run -p 80:80 saymagic/ubuntu-nginx:v2
```

此时，终端会卡住，这是正常的，因为Docker的思想是每个容器最好只开一个线程做一件事，此时我们打开了nginx服务器，所以终端卡住也没关系（当然是有办法来解决这个问题，但这里不做介绍）。现在我们可以通过浏览器访问localhost查看效果，如果是虚拟主机则需输入主机ip地址，然后就能看到了如下的页面：

DaoCloud实战

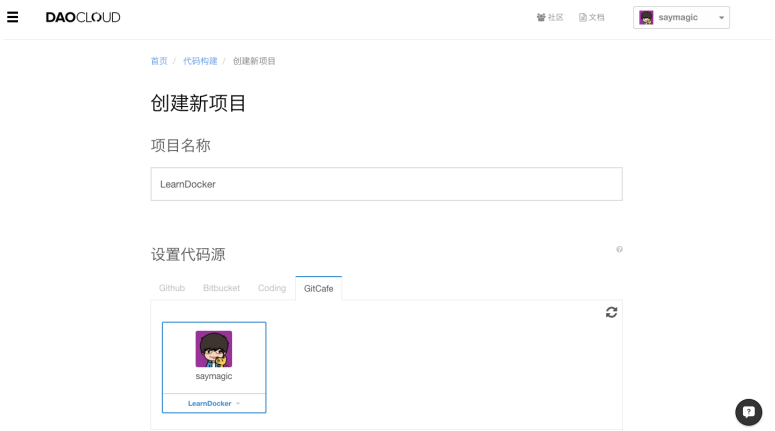
如果我们自己没有服务器，刚刚的网页我们只能在本地访问，好可惜。别急，现在我要隆重介绍一个Docker的好伙伴——DaoCloud，官网传送门：https://www.daocloud.io/

有了DaoCloud，我们只需要负责写Dockerfile，剩下的build、运行之类的东西都交给DaoCloud，我们只需要点一点按钮即可。

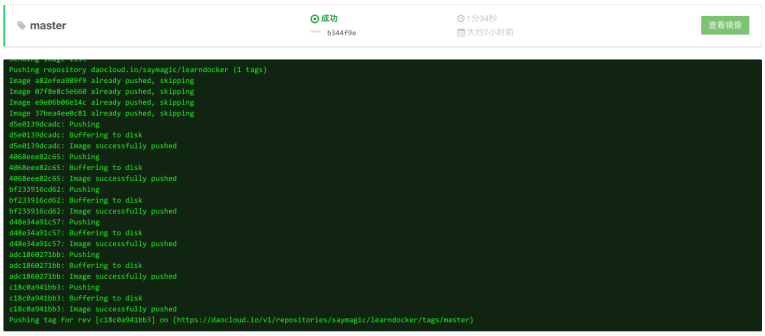
DaoCloud会将Github、GitCafe等git服务商作为代码源，这里我使用GitCafe，为了你下面的操作更加方便，你可以直接Fork我的项目，项目地址：

https://gitcafe.com/saymagic/LearnDocker 。

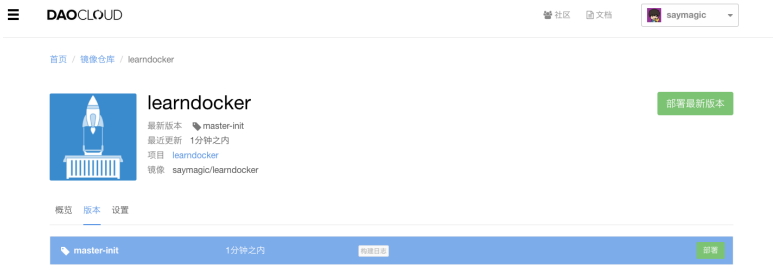
接着，我们需要去Daocloud注册一个账号，完成后，进入个人主页后选择代码构建->创建新项目->给项目起一个响亮的名字->同步GitCafe代码源->选择GitCafe下的LearnDocker项目：



最后，点击开始创建按钮。Daoloud就会马不停蹄的运行起来。如果细心的话你会发现，DaoCloud的build会比本地快很多，很迅速就会完成镜像的构建：

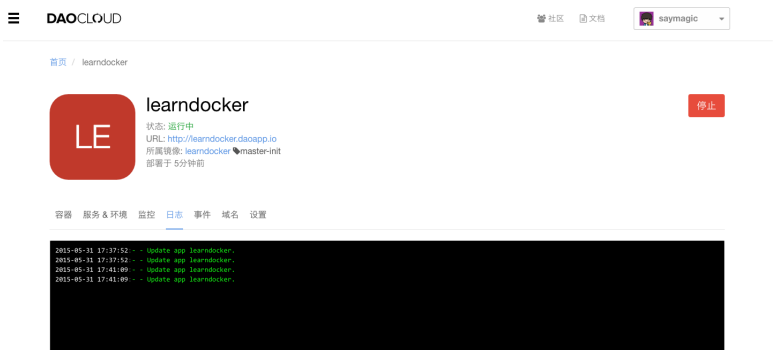


仅仅是构建镜像没什么意思， DaoCloud还可以将这个镜像在云端运行起来。我们点击绿色的查看镜像按钮，跳转到如下页面：

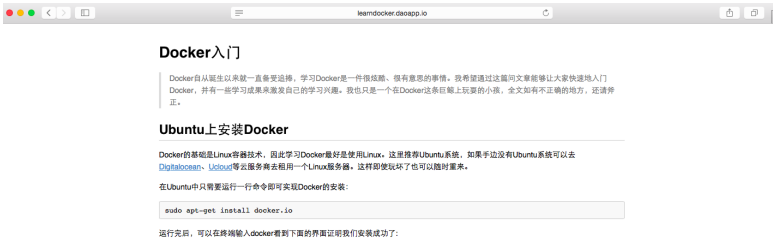


在DaoCloud中镜像需要运行在容器中，因为当前我们只构建了一个版本，所以选择部署最新版本和下面的部署按钮效果相同，点击任意一个，来到了这里：

我们给容器起一个霸气的名字learndocker，然后点击页面最下面的立即部署按钮，秒秒钟，我们的应用就运行了起来：



此时，注意到你自己运行成功的url后面的链接，将其复制到浏览器打开，你会发现，网页的内容就是本篇博客：



写在最后

Docker能做的事情远不止这些，更多有意思的事情还请读者慢慢用心去发现。