Docker的save和export命令的区别

我最近在玩Docker，一种应用程序容器和Linux的虚拟技术。它太酷了，创建Docker镜像和容器只需要几分钟。所有的工作都是开箱即用的。

在结束我一天的工作之前，我希望能保存下我的工作。但我在Docker的save和export命令之间，我凌乱了。我不知道它们之间有什么区别。所以，我上StackOverflow问了一个问题，接着得到mbarthelemy很棒的回复。

开源项目Docker，Red Hat新的虚拟化选择 http://www.linuxidc.com/Linux/2013-10/91051.htm

dockerlite: 轻量级 Linux 虚拟化 http://www.linuxidc.com/Linux/2013-07/87093.htm

Docker的搭建Gitlab CI 全过程详解 http://www.linuxidc.com/Linux/2013-12/93537.htm

Docker 和一个正常的虚拟机有何区别? http://www.linuxidc.com/Linux/2013-12/93740.htm

Docker 将改变所有事情 http://www.linuxidc.com/Linux/2013-12/93998.htm

以下是我发掘到的内容：

Docker是如何工作的（简单说明）

Docker是基于镜像的。镜像类似于已经包含了文件、配置和安装好的程序的虚拟机镜像。同样的，你可以像启动虚拟机一样启动多个镜像实例。运行中的镜像称为容器。你可以修改容器（比如删除一个文件），但这些修改不会影响到镜像。不过，你使用docker commit <container-id> <image-name>命令可以把一个正在运行的容器变成一个新的镜像。

举个例子：

# 像Docker官方的hello world例子一样，拉取一个叫busybox的镜像

sudo docker pull busybox

# 查看本地已经有哪些镜像

# 我们可以看到busybox

sudo docker images

# 现在让我们来修改下busybox镜像的容器

# 这次，我们创建一个文件夹

sudo docker run busybox mkdir /home/test

# 让我们再看看我们有哪些镜像了。

# 注意每条命令执行后容器都会停止

# 可以看到有一个busybox容器

sudo docker ps -a

# 现在，可以提交修改了。

# 提交后会看到一个新的镜像busybox-1

# <CONTAINER ID> 是刚刚修改容器后得到的ID

sudo docker commit <CONTAINER ID> busybox-1

# 再看看我们有哪些镜像。

# 我们现在同时有busybox和busybox-1镜像了。

sudo docker images

# 我们执行以下命令，看看这两个镜像有什么不同

sudo docker run busybox [ -d /home/test ] && echo 'Directory found' || echo 'Directory not found'

sudo docker run busybox-1 [ -d /home/test ] && echo 'Directory found' || echo 'Directory not found'

现在，我们有两个不同的镜像了（busybox和busybox-1），还有一个通过修改busybox容器得来的容器（多了一个/home/test文件夹）。下面来看看，是如何持久化这些修改的。

导出(Export)

Export命令用于持久化容器（不是镜像）。所以，我们就需要通过以下方法得到容器ID：

sudo docker ps -a

接着执行导出：

sudo docker export <CONTAINER ID> > /home/export.tar

最后的结果是一个2.7MB大小的Tar文件（比使用save命令稍微小些）。

保存(Save)

Save命令用于持久化镜像（不是容器）。所以，我们就需要通过以下方法得到镜像名称：

sudo docker images

接着执行保存：

sudo docker save busybox-1 > /home/save.tar

最后的结果是一个2.8MB大小的Tar文件（比使用export命令稍微大些）。

它们之间的不同

现在我们创建了两个Tar文件，让我们来看看它们是什么。首先做一下小清理——把所有的容器和镜像都删除：

# 查看所有的容器

sudo docker ps -a

# 删除它们

sudo docker rm <CONTAINER ID>

# 查看所有的镜像

sudo docker images

# 删除它们

sudo docker rmi busybox-1

sudo docker rmi busybox

译注：可以使用 docker rm $(docker ps -q -a) 一次性删除所有的容器，docker rmi $(docker images -q) 一次性删除所有的镜像。

现在开始导入刚刚导出的容器：

# 导入export.tar文件

cat /home/export.tar | sudo docker import - busybox-1-export:latest

# 查看镜像

sudo docker images

# 检查是否导入成功，就是启动一个新容器，检查里面是否存在/home/test目录（是存在的）

sudo docker run busybox-1-export [ -d /home/test ] && echo 'Directory found' || echo 'Directory not found'

使用类似的步骤导入镜像：

# 导入save.tar文件

docker load < /home/save.tar

# 查看镜像

sudo docker images

# 检查是否导入成功，就是启动一个新容器，检查里面是否存在/home/test目录（是存在的）

sudo docker run busybox-1 [ -d /home/test ] && echo 'Directory found' || echo 'Directory not found'

那，它们之间到底存在什么不同呢？我们发现导出后的版本会比原来的版本稍微小一些。那是因为导出后，会丢失历史和元数据。执行下面的命令就知道了：

# 显示镜像的所有层(layer)

sudo docker images --tree

执行命令，显示下面的内容。正你看到的，导出后再导入(exported-imported)的镜像会丢失所有的历史，而保存后再加载（saveed-loaded）的镜像没有丢失历史和层(layer)。这意味着使用导出后再导入的方式，你将无法回滚到之前的层(layer)，同时，使用保存后再加载的方式持久化整个镜像，就可以做到层回滚（可以执行docker tag <LAYER ID> <IMAGE NAME>来回滚之前的层）。

vagrant@Ubuntu-13:~$ sudo docker images --tree

├─f502877df6a1 Virtual Size: 2.489 MB Tags: busybox-1-export:latest

└─511136ea3c5a Virtual Size: 0 B

└─bf747efa0e2f Virtual Size: 0 B

└─48e5f45168b9 Virtual Size: 2.489 MB

└─769b9341d937 Virtual Size: 2.489 MB

└─227516d93162 Virtual Size: 2.489 MB Tags: busybox-1:latest