docker save和docker export都能导出镜像包，咋看起来区别似乎不大。本文就针对这个问题，试图搞清楚docker save和docker export的功能是什么？适用于什么应用场景？

本文的测试的Docker版本如下，不保证所有版本的docker都能重现本文的结果。

```
>docker version

Client:
 Version:      17.07.0-ce-rc1
 API version:  1.31
 Go version:   go1.8.3
 Git commit:   8c4be39
 Built:        Wed Jul 26 05:19:44 2017
 OS/Arch:      windows/amd64

Server:
 Version:      17.07.0-ce-rc1
 API version:  1.31 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   8c4be39
 Built:        Wed Jul 26 05:25:01 2017
 OS/Arch:      linux/amd64
 Experimental: true
```

另外我是在Windows on bash里面操作docker，有些命令如 `ls` 并不是windows命令，如果想要复现我的试验，请换成相应的windows命令。

docker save

docker的命令行接口设计得很优雅，很多命令的帮助直接在后面加 `--help` 就可以查看。

docker save的帮助如下：

```
>docker save --help

Usage:  docker save [OPTIONS] IMAGE [IMAGE...]

Save one or more images to a tar archive (streamed to STDOUT by default)

Options:
      --help            Print usage
  -o, --output string   Write to a file, instead of STDOUT
```

从命令行帮助可以看出，docker save是用来将一个或多个image打包保存的工具。

例如我们想将镜像库中的postgres和mongo打包，那么可以执行：

```
docker save -o images.tar postgres:9.6 mongo:3.4
```

打包之后的 `images.tar` 包含 `postgres:9.6` 和 `mongo:3.4` 这两个镜像。

虽然命令行参数要求指定image，实际上也可以对container进行打包，例如：

```
>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
3623943d369f        postgres:9.6        "docker-entrypoint..."   3 hours ago         Up 3 hour

>docker save -o b.tar postgres
>docker save -o c.tar postgres:9.6
>ls -al
-rwxrwxrwx 1 root root 277886464 8月  26 14:40 b.tar
-rwxrwxrwx 1 root root 277886464 8月  26 14:41 c.tar
```

通过以上命令可以看到， `b.tar` 和 `c.tar` 是完全一模一样的。这说明，docker save如果指定的是container，docker save将保存的是容器背后的image。

将打包后的镜像载入进来使用docker load，例如：

```
docker load -i images.tar
```

上述命令将会把 `postgres:9.6` 和 `mongo:3.4` 载入进来，如果本地镜像库已经存在这两个镜像，将会被覆盖。

docker save的应用场景是，如果你的应用是使用docker-compose.yml编排的多个镜像组合，但你要部署的客户服务器并不能连外网。这时，你可以使用docker save将用到的镜像打个包，然后拷贝到客户服务器上使用docker load载入。

docker export

照例查看下docker export的帮助：

```
>docker export --help

Usage:  docker export [OPTIONS] CONTAINER

Export a container's filesystem as a tar archive

Options:
      --help            Print usage
  -o, --output string   Write to a file, instead of STDOUT
```

从帮助可以看出，docker export是用来将container的文件系统进行打包的。例如：

```
docker export -o postgres-export.tar postgres
```

docker export需要指定container，不能像docker save那样指定image或container都可以。

将打包的container载入进来使用docker import，例如：

```
docker import postgres-export.tar postgres:latest
```

从上面的命令可以看出，docker import将container导入后会成为一个image，而不是恢复为一个container。

另外一点是，docker import可以指定IMAGE[:TAG]，说明我们可以为镜像指定新名称。如果本地镜像库中已经存在同名的镜像，则原有镜像的名称将会被剥夺，赋给新的镜像。原有镜像将成为孤魂野鬼，只能通过IMAGE ID进行操作。

docker export的应用场景主要用来制作基础镜像，比如你从一个ubuntu镜像启动一个容器，然后安装一些软件和进行一些设置后，使用docker export保存为一个基础镜像。然后，把这个镜像分发给其他人使用，比如作为基础的开发环境。

docker save和docker export的区别

总结一下docker save和docker export的区别：

1. docker save保存的是镜像（image），docker export保存的是容器（container）；
2. docker load用来载入镜像包，docker import用来载入容器包，但两者都会恢复为镜像；
3. docker load不能对载入的镜像重命名，而docker import可以为镜像指定新名称。

脑洞

前面所讲的内容都是些基础知识，相信各位读者只要仔细看下官方文档就能知晓。这一节我来讲讲文档上没有的东西。

docker load和docker import都可以将tar包导入为镜像，我不禁脑洞一下，docker load能不能导入docker export的容器包，docker import能不能导入docker save的镜像包呢?

以下开始试验，准备以下两个文件：

```
>ls -al
-rwxrwxrwx 1 root root 271760384 8月  26 12:15 postgres-export.tar
-rwxrwxrwx 1 root root 398292480 8月  26 12:13 postgres-save.tar
```

其中 `postgres-export.tar` 是通过docker export导出的容器包， `postgres-save.tar` 是通过docker save保存的镜像包，两者都是基于 `postgres:9.6` 镜像。从文件大小可以直观的发现， `postgres-export.tar` 显然要比 `postgres-save.tar` 小100多M。

现在试试docker load容器包 `postgres-export.tar` ：

```
>docker load -i postgres-export.tar
open /var/lib/docker/tmp/docker-import-082344818/bin/json: no such file or directory
```

显然，docker load不能载入容器包。

那么，反过来，docker import载入镜像包可不可以呢?

```
>docker import postgres-save.tar postgres
sha256:8910feec1ee2fac8c152dbdd0aaab360ba0b833af5c3ad59fcd648b9a24d4838
>docker image ls
REPOSITORY                                  TAG                 IMAGE ID            CREATE
postgres                                    latest              8910feec1ee2        2 minu
```

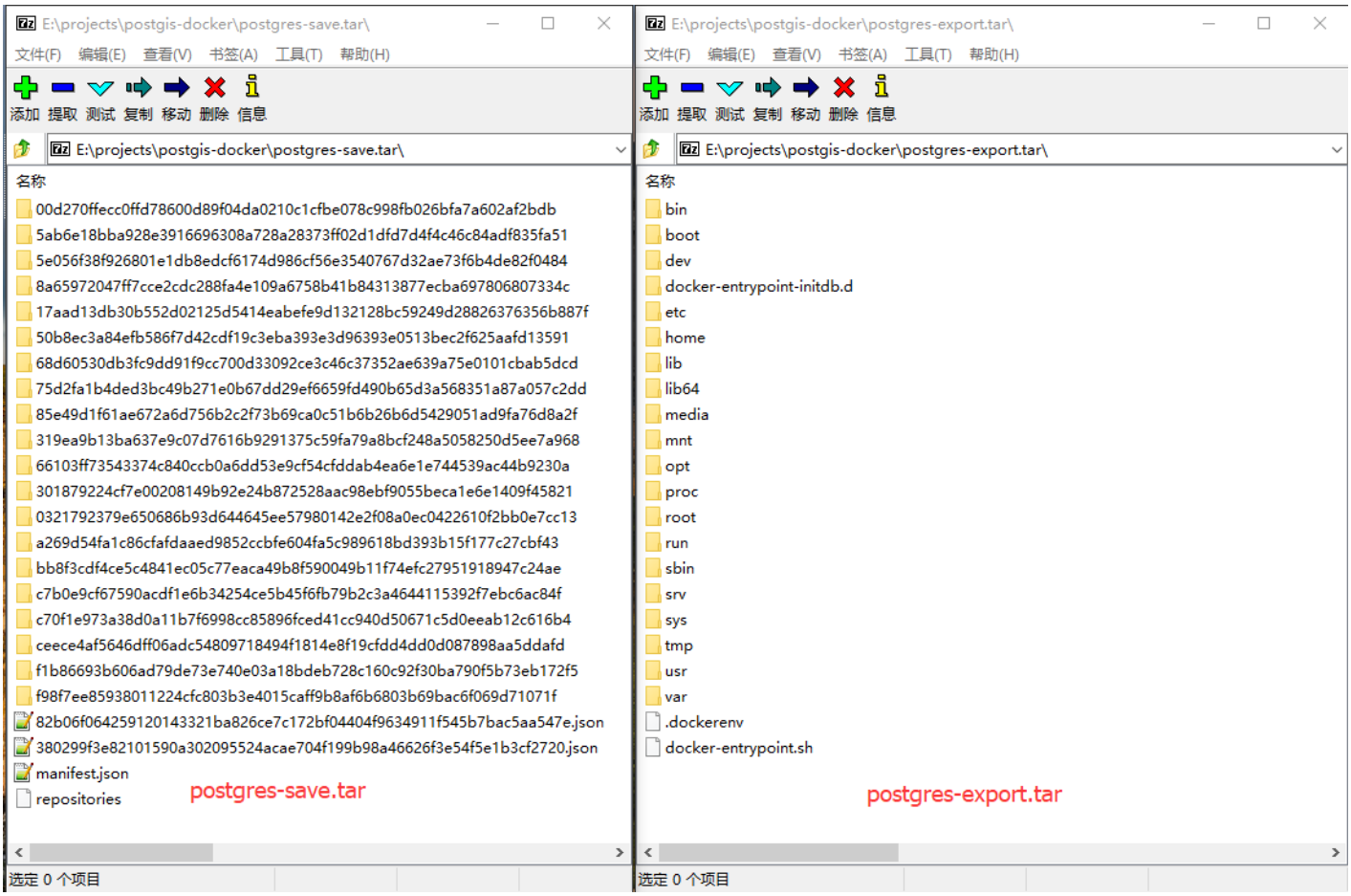
WTF，竟然成功了！！！

莫慌，再试试启动一个postgres容器：

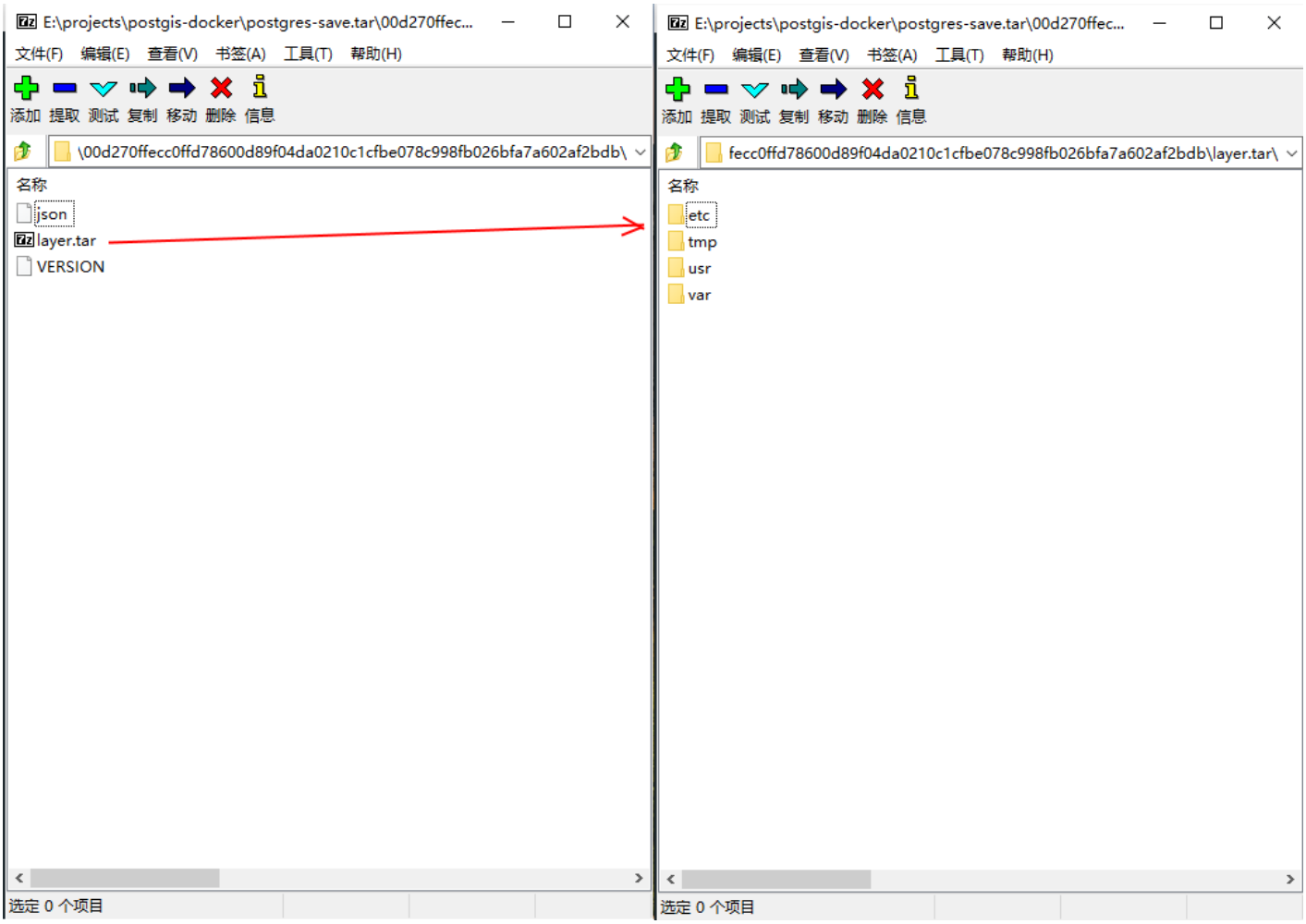
```
>docker run postgres
C:\Program Files\Docker\Docker\resources\bin\docker.exe: Error response from daemon: No commar
See 'C:\Program Files\Docker\Docker\resources\bin\docker.exe run --help'.
```

虽然能够成功地导入为一个镜像，然而这个镜像并不能使用。

要搞清楚到底是怎么回事，我们先看看镜像包和容器包由什么区别：



从上面可以看出右边的 `postgres-export.tar` 的内容是一个linux系统的文件目录，猜测就是一个linux镜像。而 `postgres-save.tar` 里面到底是什么内容呢？点开一个文件夹看看：



其实就是一个分层的文件系统。Docker镜像实际上就是由这样的一层层文件进行叠加起来的，上层的文件会覆盖下层的同名文件。如果将 `postgres-save.tar` 中的各层文件合并到一起，基本就是 `postgres-export.tar` 的内容。由于 `postgres-save.tar` 里面的各层文件会存在很多重复的文件，这也解释了为什么 `postgres-save.tar` 会比 `postgres-export.tar` 大100多M。

docker load必须要载入的是一个分层文件系统，而 `postgres-export.tar` 并不具有这样的结构，因此无法载入。

而docker import仅仅是将tar包里面的文件复制进来，所以不管tar包里面的文件结构是怎样的，都可以载入进来，所以能够载入 `postgres-save.tar` 。但 `postgres-save.tar` 并不是一个有效的操作系统镜像，因此当我试图以改镜像启动容器时，容器并不能启动。

我们再来看看docker import的帮助：

```
Usage:  docker import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

Import the contents from a tarball to create a filesystem image

Options:
  -c, --change list      Apply Dockerfile instruction to the created image
      --help             Print usage
  -m, --message string   Set commit message for imported image
```

似乎和docker commit很像：

```
Usage:  docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

Create a new image from a container's changes

Options:
  -a, --author string    Author (e.g., "John Hannibal Smith
                         <hannibal@a-team.com>")
  -c, --change list      Apply Dockerfile instruction to the created image
      --help             Print usage
  -m, --message string   Commit message
  -p, --pause            Pause container during commit (default true)
```

发现docker import和docker commit都有 `--change` 和 `--message` 选项。我们可以将docker import理解为将外部文件复制进来形成只有一层文件系统的镜像，而docker commit则是将当前的改动提交为一层文件系统，然后叠加到原有镜像之上。

关于docker save和docker export的区别讲得差不多，拜了个拜。

参考文献

1. docker save帮助文档
2. docker load帮助文档
3. docker export帮助文档
4. docker import帮助文档
5. docker commit帮助文档